Eleonora Sibilio

Formal Methods for Wireless Systems

Ph.D. Thesis

Università degli Studi di Verona
Dipartimento di Informatica

Advisor:
prof. Massimo Merro

Series N°: **TD-08-10**

Università di Verona
Dipartimento di Informatica
Strada le Grazie 15, 37134 Verona
Italy

ad Alessio,
per essere nato.

Abstract *Wireless systems* consist of wireless devices which communicate with each other by means of a *radio frequency channel*. This networking paradigm offers much convenience, but because of the use of the wireless medium it is inherently vulnerable to many threats. As a consequence, security represents an important issue. Security mechanisms developed for wired systems present many limitations when used in a wireless context. The main problems stem from the fact that they operate in a centralised manner and under the assumption of a "closed world". Formal techniques are therefore needed to establish a mathematically rigorous connection between modelling and security goals.

In the present dissertation we apply the well-known formalism of process calculus to model the features of wireless communication. The scientific contributions are primarily theoretical. We propose a timed process calculus modelling the communication features of wireless systems and enjoying some desirable time properties. The presence of *time* allows us to reason about *communication collisions*. We also provide *behavioural equivalences* and we prove a number of algebraic laws. We illustrate the usability of the calculus to model the Carrier Sense Multiple Access scheme, a widely used MAC level protocol in which a device senses the channel before transmitting.

We then focus on security aspects, in particular we propose a *trust model* for mobile ad hoc networks, composed only of mobile nodes that communicate each other without relying on any base station. We model our networks as multilevel systems because trust relations associate security levels to nodes depending on their behaviour. Then we embody this trust model in a process calculus modelling the features of ad hoc networks. Our calculus is equipped with behavioural equivalences allowing us to develop an observational theory. We ensure *safety despite compromised nodes* and *non interference* results. We then use this calculus to analyse a secure version of a leader election algorithm for ad hoc networks. We also provide an encoding of the endairA routing protocol for ad hoc networks.

Finally, we extend the trust-based calculus with timing aspects to reason about the relationship between trust and time. We then apply our calculus to formalise the routing protocol ARAN for ad hoc networks.

Acknowledgements. I want to thank prof. Massimo Merro for giving me the possibility to be admitted to the Ph.D. program under his supervision. I am grateful for his teaching during these three years and for his advice during the writing-up.

I am also grateful to prof. Luca Viganò for always finding time for his comments and to Dr. Davide Quaglia for his "engineering" help during my work.

Particular thanks to my Ph.D. colleagues Elisa Burato and Marco Volpe for their friendship, good humour and encouragement.

I am infinitely debtor to my parents for their presence in spite of the distance and their enormous moral support.

Many thanks to my brother Enrico for being "contagious" in the passion for the research, for his suggestions and wide-ranging discussions.

Incomparable thanks to my brother Leonardo and my sister-in-law Rossana for giving me the biggest happiness and motivation: my wonderful nephew Alessio.

Contents

List of Tables

List of Figures

1

Introduction

Wireless systems are commonly associated with telecommunications networks whose interconnections between devices (or nodes) are implemented without the use of wires. This kind of network uses *electromagnetic waves* for the carrier, such as *radio waves, infrared radiation, microwaves*, to transmit information.

Packet Data technology is a form of packet switching technology used to transmit digital data via radio communications links. It was developed in the mid-1960s and was put into practical application in 1969 in the Advanced Research Projects Agency Network (ARPANET), created by the Defence Advanced Research Projects Agency (DARPA) of the United States Department of Defence. It was the world's first operational packet switching network, and the predecessor of the contemporary global Internet. Initiated in 1970, the ALOHANET, based at the University of Hawaii, was the first large-scale packet radio project. Using ALOHANET, a number of experiments have been performed in order to develop methods to arbitrate access to a shared radio channel by network nodes. These experiments were generally considered to be successful, and also marked the first demonstration of Internetworking, as data were routed between the ARPANET, PRNET (a packet radio network created by DARPA), and SATNET (a satellite packet radio network) networks.

During the 1980s the wide evolution of business, trade and industry introduced in everyday life new demands in communication technologies. Therefore these technologies, needed for such things as pagers and wireless telephones, would be perfected to the point that they became widely available consumer products. Initially developed only for voice communication services, at the end of 1990s, with the enormous increase of Internet usage, it became necessary the development of wireless technologies for data transfer. Thus WAP and GPRS technologies were born, characterised by a long communication range. Besides these, wireless standards with shorter range were developed: among these, the IEEE 802.11 for wireless LAN was introduced and improved during the years.

Today, the range of application of wireless technologies includes data communications, Internet access, multimedia applications, and mobile payment services. Moreover they offer a number of advantages over alternative solutions, as increased capacity, reduced power and reduced interference from other signals.

Important features of wireless systems are: (i) half-duplex radio frequency channel: on a given channel, a node can either transmit or receive but cannot do both at the same time; (ii) local broadcast communication: a message sent by a node reaches only the nodes within the radio transmission range of the sender; (iii) mobility: nodes move while remaining connected to the network, breaking links with old neighbours and establishing fresh links with new devices, and (iv) time: many activities in wireless systems require the need for a common notion of time among devices. During the last ten years or so, wireless systems have represented a strong stimulus to the research community which has devoted to them hundreds of papers.

1.1 Problem Statement

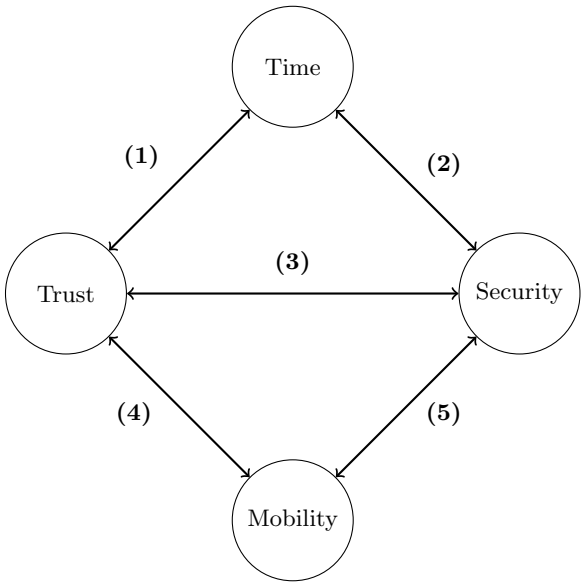
In spite of the enormous proliferation of wireless technology, many technical challenges remain in designing robust and secure wireless networks. The current situation in the world of wireless communications leads to new requirements from users. Ubiquitous systems presuppose that the user can access the communications anytime and anywhere. The arrival of mobile communications brings a dynamic aspect into the digital environment and extends security-related requirements. Moreover, the "open" and shared nature of their medium makes wireless systems very vulnerable to many threats, some ones due to atmospheric problems or to technical failures, but the most ones are due to human misbehaviours. The gap between these current and emerging features and requirements and the vision of secure wireless applications indicates that much work remains to do in order to make concrete this reality. Formal techniques are therefore needed to establish a mathematically rigorous connection between modelling and desired security goals.

Traditional security mechanisms, widely and successfully used for wired systems, present a number of limitations when used in a wireless context. The main problems stem from the fact that these systems operate in centralised manner and under the assumption of a "closed world". Indeed, if in traditional digital systems the main security requirements were addressed by installing firewalls at network entries and by providing security level monitoring, these solutions are inadequate for wireless systems. Indeed, wired communication may be seen as a static environment having more or less defined threats, lists of closed communicating entities and identities and static mostly direct trust relationships between them. This is not the case of wireless systems, where the number and the identity of users are not known in advance.

A more *soft* approach to security seems to be more appropriate [131]. The goal of soft security mechanisms is to stimulate the collaborative adherence to common ethical norms by participants in a community and the collaborative enforcement of them. In other words, they describe a social control model which acknowledges that malicious users can exist among good ones and attempts to make them known to all the community in order to isolate them.

In everyday life every interactions between persons or organisations are based on some kind of *trust* relationships. In the modern wireless world, the concept of trust plays a significant rôle because relations in the virtual world more and more

Figure 1.1 Time, Trust, Security and Mobility in Wireless Networks



reflect real life. Trust precludes some elements of certainty in the relationships among participants and it exploits the mechanisms of cooperation, vital in wireless systems that do not rely on the presence of a fixed infrastructure. The *trust management approach* for wireless systems is a soft security mechanism developed as an answer to the inadequacy of traditional authorisation mechanisms. Trust management systems determine whether or not a request should be allowed, by defining languages for expressing authorisations and access control policies, and by providing a trust management engine for determining when a particular request is authorised. They operate in distributed systems and eliminating the closed world assumption of traditional access control systems. Trust establishment in the context of wireless networks is still an open and challenging field.

In Figure 1.1, we give a scheme for the relationships between specific aspects considered in this dissertation and playing important rôles in wireless systems: time, trust, security and mobility. Time has implications with security and trust; trust is related with security; mobility has implications with trust and security.

- Arrow (1): in trust models for wireless networks, the timing factor is important because more recent trust information should have more influence on the trust establishment process. More generally, a notion of time would allow to record past behaviours. In this manner, a malicious agent, which has been previously detected, cannot regain trust after some time;
- Arrow (2): many security protocols for wireless networks require the presence of timing information, because they present parameters that may vary with time. Usually these parameters certify the freshness of a message. In this set-

ting, some security properties, such as authentication and secrecy, can be reformulated in a timed setting;
- Arrow (3): security mechanisms are designed to protect infrastructure and information. Usually security objectives formulation is based on the analysis of trust to all possible entities that may potentially interact with a system. The concept of trustworthiness is essential in open and distributed environments. Traditional security solutions are totally inadequate in this setting and then decentralised trust management systems were developed as an answer to their inadequacy;
- Arrow (4): mobility has impact on the trust establishment process, as it introduces issues related to user credentials management, indirect trust establishment and mutual authentication between previously unknown and untrusted entities;
- Arrow (5): in the traditional digital world the main security requirements were addressed by installing firewalls at network borders and security level monitoring. Fixed communication may be seen as a static environment having more or less defined threats, lists of communicating entities and static, mostly direct trust relationships between them. The arrival of mobile communications brings a dynamic aspect into the digital environment and extends security-related requirements.

1.2 Summary of Contributions

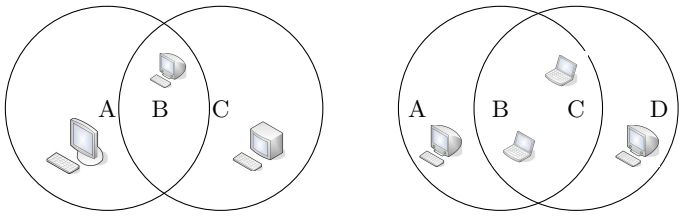
The *goal* of the present dissertation is modelling and reasoning about wireless systems and their security needs by means of formal methods. We focus on the well-known formalism of *process calculus*, that during the last years has been extended to cope with the possibility to model features of wireless systems and to formalise and analyse some protocols. Moreover, we focus on *trust-based security*, developing trust models explicitly tailored for wireless systems. The scientific contributions are primarily theoretical. In the sequel, we provide a summary of them.

A Timed Calculus for Wireless Systems

We propose a timed calculus for wireless systems called TCWS, in which all wireless devices are assumed to be *synchronised*, using some *clock-correction synchronisation protocol* [76,221] correcting the local clock of each node to run in par with a global time scale. Time proceeds in *discrete* steps represented by occurrences of a simple action tick, in the style of Hennessy and Regan's TPL [111], to denote idling until the next clock cycle. As in Hennessy and Regan's TPL [111], and Prasad's timed CBS [195], our TCWS enjoys three basic time properties:

- *time determinism*: the passage of time is deterministic, i.e. a network can reach at most one new state by performing the action tick;
- *patience*: nodes will wait indefinitely until they can communicate;
- *maximal progress*: data transmissions cannot be delayed, they must occur as soon as a possibility for communication arises.

Figure 1.2 Hidden and Exposed terminal problem



The operational semantics of our calculus is given in terms of a *labelled transition system*. We follow a *two-phase* approach [181] separating the execution of actions for synchronisation from the passage of time.

In this calculus we focus on *communication interferences*. In the literature, there exist a number of process calculi modelling wireless systems [91, 92, 94, 163, 167, 178, 214]. All these calculi rely on the presence of some MAC-level protocol to remove interferences. However, in wireless systems *collisions cannot be avoided* although there are protocols to reduce their occurrences. We believe that communication collisions represent a serious concern that should be taken into account in a timed model for wireless systems.

As a case study we use our calculus to model the *Carrier Sense Multiple Access* (CSMA) protocol [123]. According to this scheme, stations transmit only when the channel is sensed free. As we will show, this protocol allows to prevent certain forms of collisions, although it suffers from two well-known problems: the *hidden terminal problem* and the *exposed terminal problem*. Figure 1.2 gives a graphical representation of these two problems. Suppose A, B, and C are three stations that want to communicate. If the station B is within the transmission range of both A and C, and A and C cannot hear each other, then A is a *hidden terminal* with respect to C and vice versa. This is because C could transmit to B when A is already transmitting to B (as it doesn't hear A) causing a collision at B (Figure 1.2 on the left).

Suppose A, B, C, and D are four stations, where the two receivers A and D are out of range of each other and the two transmitters B and C are in range of each other. Here, if a transmission between B and A is taking place, node C is prevented from transmitting to D as it concludes after carrier sense that it will interfere with the transmission by its neighbour B. However D could still receive the transmission of C without interference because it is out of range from B. Therefore, C is an *exposed terminal* with respect to B, and vice versa (Figure 1.2 on the right).

We propose as main program equivalence a timed variant of weak *reduction barbed congruence* [172]. As an efficient proof method for our timed barbed congruence we propose a security variant of *labelled bisimilarity* [169]. We then apply our bisimulation proof-technique to prove a number of algebraic properties.

A Calculus of Trustworthy Ad hoc Networks

We then focus on trust-based security aspects. We propose a process calculus for mobile ad hoc networks which embodies a *behaviour-based multilevel trust model*. We call this calculus CTAN.

We model our networks as *multilevel systems* [28] where each device is associated to a security level depending on its behaviour. Thus, trust relations associate security levels to nodes.

None of process calculi recently used to model different aspects of wireless systems and cited above addresses the notion of trust. In our calculus, each node is equipped with a local trust store containing a set of assertions. These assertions supply trust information about the other nodes, according to a local security policy. Our calculus is not directly concerned with cryptographic underpinnings. However, we assume the presence of a hierarchical key generation and distribution protocol [122, 211]. Thus, messages are transmitted at a certain security level relying on an appropriate set of cryptographic keys.

We provide the operational semantics of our calculus in terms of a labelled transition system where each transition is annotated with a security level. For simplicity, our operational semantics does not directly express node mobility. However, we will show how to adapt the mobility approach proposed in [92] to annotate our labelled transitions with the necessary information to represent node mobility.

Our calculus guarantees that only authorised nodes receive sensible information. Thus, our networks enjoy two desirable security properties: *safety preservation* and *safety despite compromised nodes*. The first property ensures that a node m transmitting at level ρ may only synchronise with nodes receiving at level ρ or above, according to the local knowledge of both sender and receivers. The second property says that bad nodes, once detected, may not interact with good nodes. In this manner, bad nodes (recognised as such) are isolated from the rest of the network.

In CTAN, our program equivalence is a security variant of weak reduction barbed congruence. Moreover, along the lines of [70], we propose a labelled bisimilarity, called *δ-bisimilarity*, parameterised on security levels. Our bisimilarity is a congruence and an efficient proof method for our reduction barbed congruence.

Information flow properties are a particular class of security properties for controlling the flow of information among different entities. The seminal idea of *non interference* proposed in [96] aims at assuring that "*variety in a secret input should not be conveyed to public output*". In a multilevel computer system [28] this property is reformulated saying that information can only flow from low levels to higher ones. We prove a non-interference result using our notion of δ-bisimilarity. Formally, high-level behaviours can be arbitrarily changed without affecting low-level equivalences, that is equivalence parameterised on low security levels. Then a network is *interference free* if its low security level behaviour is not affected by any activity at high security level.

As case studies, we use our calculus to formalise and analyse a *secure* version of the leader election algorithm for mobile ad hoc networks [227]. We then propose an encoding of the *endairA* routing protocol for ad hoc networks [9].

Time vs Trust

We introduce a simple timed variant of the calculus previously presented. We call it TCTAN. First of all, we extend our trust model proposed for CTAN with a very simple notion of time to express the validity of trust information. In this manner, more recent trust information should have more influence on the trust establishment process. Then, we develop appropriate changes in operational semantics in order to distinguish between instantaneous and timed actions. As in TWCS, we adopt the fictitious clock approach: a global clock is supposed to be updated whenever all the processes agree on this, by globally synchronising on the special action, called tick, representing the passing of one time unit. All the other actions are assumed to be instantaneous.

In TCTAN we assume the presence of clock synchronisation protocols following the *untethered clock approach* [75, 188, 202], achieving a common notion of time without synchronisation. A global time scale is maintained while letting the local clocks run untethered.

In this last calculus we work with configurations. A *configuration* $t \triangleright M$ is a pair composed by a time indicator and a network, where t indicates the global time. As in TWCS, we separate the execution of actions from the passage of time. We provide the operational semantics of our calculus in terms of a labelled transition system.

We adapt the behavioural theory of CTAN in order to model it to timing extensions. Moreover, we prove that all the properties of CTAN are preserved by this calculus and that it also enjoys the basic time properties of time determinism, patience and maximal progress. As case study, we use our calculus to formalise the secure, on-demand routing protocol ARAN [209], that uses digital certificates with timestamps.

1.3 Outline

Our thesis is divided into two parts. In the first part we give an overview of background arguments, describing for each of them related work. In the second part we focus on contribution arguments.

Part I: Background

- In **Chapter 2** we give an overview of wireless networks. We also deal with the rôle of time, security aspects and mobility;
- In **Chapter 3** we describe traditional access control, and we introduce the concept of trust and trust management systems;
- In **Chapter 4** we present an overview of foundational process calculi, timed process calculi and some recent calculi for wireless networks;
- In **Chapter 5** we give an overview of some formal methodologies for the modelling and analysis of wireless systems.

Part II: Contributions

Some of the material of the second part has been presented in our papers [165,166] and in technical report [164].

- In **Chapter 6** we describe our timed calculus for wireless systems;
- In **Chapter 7** we present our calculus of trustworthy ad hoc networks;
- In **Chapter 8** we present a timed extension of our trust-based calculus;
- In **Chapter 9** we conclude the dissertation with a discussion of research contributions and directions for future work;
- In **Appendix A** we provide supportive material and full proofs.

1.4 Publications

In the following we report publications concerned with the present dissertation:

[166] Massimo Merro and Eleonora Sibilio. *A Timed Calculus for Wireless Systems.* In proceedings of the 3rd International Conference on Fundamentals of Software Engineering (FSEN), volume 5961 of Lecture Notes in Computer Science, pages 228-243. Springer, 2009;

[164] Massimo Merro and Eleonora Sibilio. *A Timed Calculus for Wireless Systems.* Technical Report 75/2009, Department of Computer Science - University of Verona, 2009, submitted to a journal;

[165] Massimo Merro and Eleonora Sibilio. *A Calculus of Trustworthy Ad Hoc Networks.* In proceedings of the 6th International Workshop on Formal Aspects in Security and Trust (FAST2009), volume 5983 of Lecture Notes in Computer Science, pages 157-172. Springer, 2010.

Part I

Background

2

Wireless Communication

2.1 Introduction

In the last years, the communication has become increasingly mobile and ubiquitous. As a result, new requirements have to be accomplished and new features have to be added to communication systems. Indeed it has been proven that traditional ways of networking, i.e. with physical cables, are inadequate to completely meet these new systems. For example, let us think of some kind of monitoring, tracking and controlling in an hostile environment, e.g. nuclear reactor control, fire detection or traffic monitoring. It should be immediately get across that in these settings the development of traditional networkings could meet some difficulties. The most obvious obstacle is the "immobility" of traditional infrastructures: if users must be connected to a network by physical cables, their movements are dramatically reduced.

Wireless communication is, by any measure, the fastest growing segment of the communications; it has become very popular in industry, business, commerce and in everyday life. Indeed, wireless technologies provide a wide choice of capabilities and features and then they are in position to satisfy different demands. Wireless technology spans from user applications, such as personal area networks, ambient intelligence, and wireless local area networks, to real-time applications, such as cellular, and ad hoc networks. Important features of wireless systems are: (i) half-duplex radio frequency channel, (ii) local broadcast communication, (iii) (possible) node mobility and (iv) time.

Wireless networks can be classified according to the coverage area of their devices. Moreover, different types can be considered according to their features.

Many activities in wireless systems require the need for a common notion of *time* among devices. For example, we can think of wireless sensor networks in which devices are equipped with sensor hardware to sense physical phenomena. The data sensed by these devices have to be combined and evaluated to derive knowledge about the environment where they are working. This operation is called *data-fusion*. The fusion of individual sensor readings is possible only by exchanging messages that are timestamped by each sensor's local clock. Many other aspects are greatly influenced by temporal relationships, e.g., whichever the media access control protocol may be or real-time and temporal synchronisation constraints

characterising the hosts communications. Moreover, many security protocols require the presence of timing information, because they present parameters that vary with time. Usually these parameters certify the freshness of a message. In this setting, some security properties, such as authentication and secrecy, can be reformulated in a timed setting. All this requires the need for a common notion of time among the devices. Protocols that provide such a common notion of time are called *clock synchronisation protocols*.

In spite of the enormous proliferation of this technology, many technical challenges remain in designing robust and safe wireless networks and then it is necessary the development of modern security applications. The gap between these current and emerging features and requirements and the vision of secure wireless applications indicates that much work remains to be done to make concrete this reality.

Operating in open and shared media, wireless communication is inherently less secure than wired communication. Mobile wireless devices usually have limited resources, such as bandwidth, storage space, processing capability, and energy, which makes security enforcement hard. Wireless networks have to face different attacks or threats that may easily compromise their functionality. The effects of these attacks are not trivial. For this reason, it is rather urgent the necessity of security mechanisms aimed either at preventing the attacks or at minimising their effects. When operating in wireless scenario, new security requirements has to be added to the traditional ones and these last ones have to be reformulated in terms of the new features.

An important aspects in wireless networks having many implications with security aspects is *mobility*. In wireless networks, mobility is associated with the ability of a user to access telecommunication services from different locations and different devices. In the traditional digital world the main security requirements were addressed by installing firewalls at network borders and security level monitoring. Fixed communication may be seen as a static environment having more or less defined threats, borders, lists of communicating entities (and identities) and static, mostly direct trust relationships between them. The arrival of mobile communications brings a dynamic aspect into the digital environment and extends security-related requirements. Thus security solutions overcoming these problems have to be developed.

We end this introduction with an outline of the present chapter: in Section 2.2 we deal with wireless networks, in particular in Section 2.2.1 we provide a classification of them, whereas in Section 2.2.2 we consider emerging types, finally in Section 2.2.3 we describe some wireless protocols. In Section 2.3 we describe the rôle of time in wireless systems and introduce the clock synchronisation protocols. In Section 2.4 we consider some security aspects in wireless systems, in particular in Section 2.4.1 we deal with their threats and vulnerabilities and in Section 2.4.2 with security requirements. In section 2.5 we deal with mobility aspect. Finally, in Section 2.6 we give a brief summary of the arguments discussed in this chapter.

2.2 Wireless Networks

With the term *wireless network* we refer to any type of computer network that is wireless, i.e. the interconnections are implemented without the use of wires. Like all networks, a wireless network is a collection of *devices*, also called *nodes* or *stations*, transmitting data over a shared network medium. This is the most obvious characteristic of wireless networks: the transmission takes place over a *wireless channel*, which is usually a *radio channel*, but can also be an *infrared channel*. In traditional networks, communication channels are *full-duplex*, that is a node can transmit and receive at the same time. As a consequence, *collisions* caused by two simultaneous transmissions are immediately detected and repaired by retransmitting the message after a randomly-chosen period of time. This is not possible in wireless networks where channels are *half-duplex*: on a given channel, a node can either transmit or receive but cannot do both at the same time.

Usually, in a network, there are three methods for transmitting a message:

- *Unicast*: when the message is sent to a single destination node. For instance web client-server interactions use unicast transmissions;
- *Broadcast*: when the message is sent to all network nodes. For instance, a message advertising the foreseen network unavailability is broadcast to all nodes;
- *Multicast*: when the message is sent to a subset of the network nodes. For instance, users connect to a network to access the Internet services, e.g., video streaming or videoconference. Different groups of users sharing the same interests can coexist. The size of the group can be very high or on the contrary, very limited.

However, in wireless networks radio signals span over a limited area, called *transmission cell*, and therefore reach only a (possibly empty) subset of devices in the system. Nodes being in the transmission cell of a node m are called *neighbours* of m. In a classical wired network, with a single broadcast transmission a message sent by a node may reach all the nodes in the network. It is not the same with wireless networks, where a message sent by a node reaches only the nodes within the radio transmission range of the sender. That is why in wireless networks it is dealt with *semi-broadcast* or *local broadcast* communications. To reach all nodes in the network, a message must be forwarded by nodes which receive it. Actually, even the devices within a cell might not be reachable due to environmental conditions such as walls, obstacles, etc. For these reason, the wireless communication must be taken into account the probability of packets losses.

In recent years, wireless technologies have become very popular such that we can say today they are the medium of choice of many applications. In the following sections, we give a classification of wireless networks and a description of the most interesting emerging types.

2.2.1 A Classification of Wireless Networks

According to either the supported data rate or the coverage area or the technological factors, different classifications of wireless networks could be showed.

The simplest and the most common one is based on the coverage area of wireless devices. In [57, 90] we can find a comprehensive overview of wireless networks according to this kind of classification.

Wireless Body Area Networks

A *Wireless Body Area Network* (WBAN) has a very limited transmission range: only 1 or 2 meters. These networks are studied to cover only the space around a human body, allowing the interconnection of "wearable" devices, like hearphones, palms and so on. The most interesting and attractive feature of a WBAN is the possibility of connecting similar devices and the ability of autoreconfiguration, making easy some common operations, like either the removal or the addition of new devices.

Wireless Personal Area Networks

A *Wireless Personal Area Network* (WPAN) provides wireless interconnections of personal devices placed around an individual person's workspace. Typically, a wireless personal area network uses a technology that permits communication within a very short range (about 10 meters). Whereas a WBAN is dedicated to the interconnections of wearable devices, a WPAN is used to create a communication in the immediately proximity of a user. The interconnection could also be established between mobile devices and a fixed infrastructure, like Internet. One such technology is Bluetooth [104], which has been used as the basis for the standard IEEE 802.15 [1].

Wireless Local Area Networks

A *Wireless Local Area Network* (WLAN) technology provides a range from 100 to 500 meters about, and high-speed wireless data connections between mobile devices (such as laptops, PDAs, phones and home entertainment equipment) or between mobile devices and nearby fixed base stations, as access points. In order to provide connectivity between a wireless station and a broadband wired network, an access point may have two interfaces, a wireless interface and a wired interface. Due to their rapid installation, flexibility, scalability and good throughput at low cost, WLANs are the most spread wireless networkings. The standard on which they are developed is the IEEE 802.11 [123]. This standard provides many useful capabilities, as mobility and quality of service support. The basic building block of an 802.11 network is the *Basic Service Set* (BSS), which is a set of stations that have successfully synchronised. Communications take place within a specific area, called the *basic service area*, defined by the propagation features of the wireless medium. When a station is in the basic service area, it can communicate with the other members within the BSS. There exist different types of BSSs:

- *Independent BBS* (IBBS): in a IBBS devices communicate directly with each other without using any distribution system. IBSSs are sometimes referred to as *ad hoc BSSs* or *ad hoc networks*;

- *Infrastructure BSS*: they are distinguished from the independent ones by the presence of an access point. Access points are used for all communications in infrastructure networks, including communication between mobile nodes in the same service area. Then in an infrastructure network, if a station wants to communicate, it must be associated to an access point to obtain network services;
- *Extended Service Set* (ESS): they are created by linking several BSSs together within a backbone network. ESSs can create coverage in small offices and homes, but they cannot cover larger areas.

Wireless Metropolitan Area Networks

A *Wireless Metropolitan Area Network* (WMAN) extends the coverage range of a local network, covering a metropolitan area or campus. Then it provides services, as Internet connection, to a very large number of users. WMANs are based on IEEE 802.16 [2] standard, also referred to as *Worldwide interoperability for Microwave Access* (WiMAX), and allow Internet connectivity for LANs in a metropolitan region. This technology provides inexpensive broadband accesses and supports for user mobility. A WMAN is composed by three elements:

- a *subscriber station* (SS) or *mobile subscriber station* (MSS), which is user's terminal;
- a *base station* (BS), that provides radio access functionality;
- a *network control and management system*, that includes entities supporting routing, network management, scheduling and coordination services, multimedia session management services, security and mobility services.

User terminals are connected to base stations, which are connected to the network control and management system.

Wireless Wide Area Networks

A *Wireless Wide Area Network* (WWAN) is network covering a large geographic area and provide an high speed wireless communication. WWANs can have a transmission range varying from 1,5 to 1,8 kilometres. Access to home services from a remote location and user mobility are provided. These networks are used above all as a mobile telecommunications cellular networks even if they can also used to transmit data.

2.2.2 Emerging Wireless Networks

In this subsection, we describe the most relevant upcoming types of wireless networks.

Cellular networks

A *cellular network* is an infrastructure-base network composed of several radio cells each served by at least one fixed-location transceiver, that is the radio access device, also known as *base station*. The base stations are connected together by a

wireline network. They are installed and managed by an authorised operator. A base station provides that mobile devices can access to the backbone with one-hop. Each base station serves only a limited physical area, called a *cell*, hence the name cellular. Mobile devices can move from one cell to another one. In a given cell, they are connected to the base station via wireless channels. By connecting their backbones together and setting up appropriate roaming agreements, different network operators can jointly provide larger coverage. In any cell is possible to use several portable transceivers that can be moved through more than one cell during transmission. In this manner it is enabled ever worldwide mobility for users.

Cellular networks have been deployed in the last decade, and are proliferating throughout the world. Today, cellular networks are so popular that in some countries, the number of mobile subscribers already exceeds the number of fixed telephone lines. Originally, cellular networks provided only voice communication services and the sending and receiving of short text messages. Today, the range of applications includes data communications, Internet access, multimedia applications, and mobile payment services. Moreover they offer a number of advantages over alternative solutions, as increased capacity, reduced power and reduced interference from other signals.

Ad hoc networks

An *ad hoc network* consists only of nodes that relay each others' traffic, that is a set stations communicating with each other without relying on any base station. The most diffused network paradigm in this context considers nodes as mobile devices. Then we talk about *Mobile Ad hoc NETwork* (MANET). Especially during the last years, such networks have been a strong stimulus to the research community. Indeed, this is a new attracting area for its potential to provide ubiquitous connectivity without the assistance of any fixed infrastructure.

We can distinguish between the following two kinds of MANETs:

- *Mobile ad hoc networks in hostile environments*: they designate a set of mobile nodes operating in an environment where the presence of a "strong" attacker is expected, e.g. in a military setting. In this context, more then in other categories, secure and safety are central and essential needs. In these networks, in compliance with the rôle of each of the users, each device knows a pre-load appropriate set of cryptographic keys in order to protect the communication. An authority in charge of the preloading phase. As long as a node is not compromised, it is reasonable to assume that it will have a highly cooperative behaviour with respect to the other nodes of the network. The security challenges typically encountered in this kind of networks include secure routing, prevention of traffic analysis, resistance of a captured device to reverse engineering and key retrieval;
- *Self-organised mobile ad hoc networks*: they do not have authority to take care of the initialisation phase. This means that the network is purely peer-to-peer, and that the nodes have to organise by themselves secure communications, establishing safety associations. Such networks are very vulnerable. One of the most serious problems is the unpredictable presence of malicious nodes. Without appropriate mechanisms, such a network can go into stopping due to the presence of attackers.

Vehicular networks

A *vehicular ad hoc network* (VANET) is potentially the largest instantiation of the mobile ad hoc networking technology. It provides communications among nearby vehicles and between vehicles and nearby fixed equipment. In other words, nodes act both as end points and as routers. By their features, VANETs are between the two cases of mobile ad hoc networks just described: indeed they cannot be fully self-organised, but they also cannot be placed under the strict control of a single authority. Their main goal is ensuring safer and more efficient driving conditions, enabling a variety of applications for safety, traffic efficiency and driver assistance. To this end, a special electronic device is placed inside each vehicle which provides ad hoc network connectivity for the passengers. For the presence of this electronic device, each of such a vehicle is a node in the VANET and can receive and send messages through the wireless medium over the network.

For the reasons explained above, most of the concerns of interest to MANETs are of interest in VANETs, but some details are different. Rather than moving at random, vehicles tend to move in an organised fashion. The interactions with road-side equipment can likewise be characterised fairly accurately. And finally, most vehicles are restricted in their range of motion, for example by being constrained to follow an asphalted road.

A VANET can integrate multiple ad hoc networking technologies such as WiFi IEEE 802.11 b and g [123] standards, WiMAX IEEE 802.16 [2] and Bluetooth [104].

Wireless mesh networks

A *wireless mesh network* is a network composed of radio nodes organised in a mesh topology. A *mesh topology* is a particular type of networking topology where each node may act as an independent router, regardless of whether it is connected to another network or not. Wireless mesh network are often made up of one *Wireless Hot Spot* (WHS), connected to the Internet, and of several *Transit Access Points* (TAPs). The TAPs are responsible for sorting the traffic between mobile stations, e.g. cell phones and other wireless devices, and the WHSs in multi-hop fashions. In order to provide wireless Internet connectivity in a wide geographic area, wireless mesh networks can represent a feasible solution. One of the most interesting feature is that when one node can no longer operate, the remaining nodes can still communicate with each other, directly or through intermediate nodes. Wireless mesh networks can be implemented with various wireless technology including the IEEE 802.11 [123] and IEEE 802.16 [2] standards, cellular technologies or combinations of them. Finally, for its inner features and its nature, a wireless mesh network can be seen as a special type of a wireless ad hoc network.

Sensor networks

A *wireless sensor network* consists of a large number of spatially distributed autonomous sensors and a few base stations (or sinks). The sensor nodes are tiny devices that are equipped with sensing circuits. They cooperatively collect data to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. Moreover, the sensor nodes are typically

equipped with a radio transceiver or other wireless communications device, a small microcontroller, and an energy source, usually a battery. The sinks are more powerful devices with respect to sensor nodes: they collect the data gathered by the sensor nodes and then send them to some specific application units for final processing. Initially developed for military applications, such networks are now used in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring, healthcare applications, home automation and traffic control. As we stated above, the sensor nodes are usually battery powered. Since recharging the batteries is often impractical, or even impossible in some deployment scenarios, this has a profound effect on the design of sensor networks. As a result, all wireless sensor networking mechanisms are designed to reduce the energy consumption of the sensor nodes and increase network lifetime as much as possible.

2.2.3 Applications

In this section, we describe some wireless protocols.

Bluetooth, IEEE 802.11, and IEEE 802.16

Bluetooth [104] is a communication protocol primarily designed for low power consumption, with a short range based on low-cost transceiver microchips in each device. It makes it possible for these devices to communicate with each other when they are in range. Bluetooth provides a way to connect and exchange information between devices such as mobile phones, telephones, laptops, personal computers, printers, GPS receivers, digital cameras, and video game consoles through a short-range radio frequency bandwidth. Because the devices use a radio broadcast communications system, they do not have to be in line of sight of each other. In particular, the process by which a pair of devices authenticate with each other and establishes the key is called *device pairing*.

The 802.11i [124] is also known as WPA2, is an amendment to the IEEE 802.11 [123] standard specifying security mechanisms for wireless networks. This has substituted the previous security specification, the Wired Equivalent Privacy (WEP), after the disclosure of some flaws. The involved entities are the peer, the access point and an authentication server. The peer authentication process involves an handshake, in which the access point acts like a pass through the server, on behalf of the peer. Actually, the process may either consists of a challenge-response mechanism, or a password-based mechanism.

The 802.16 [2] and its amendment IEEE 802.16e-2005 are standards for wireless metropolitan area networks. They contain the PKMv2 protocol, enhancing security features with respect to the old version. The protocol has a notable mixture of security components. Parties at stake are base stations and mobile stations, that need to securely transmit and receive. Base and mobile stations can mutually authenticate, and it was missing in the first version. The chain of keys that are sequentially computed to finally derive the ultimate key, used in the communication, are now derived from both the contribution of all the two parties.

Routing Protocols

Routing is a fundamental function in every network that is based on multi-hop communications, as mobile ad hoc networks and wireless sensor networks. These systems act as end-systems and they also perform routing functions. In this context, it is useful to follow the distinction in [222] between forwarding and routing processes. When a node is looking up the appropriate routing information and a packet arrives at that node, the process of *forwarding* is activated: it appropriately allows to relay the packet towards its destination. Routing information are computed by the *routing* process, and is defined by routing protocol or routing algorithm.

A *routing protocol* is used to determine the appropriate paths on which data should be transmitted in a network. According to [57], routing protocols for wireless systems can be classified into:

- *Topology-based* protocols: they rely on traditional routing concepts, such as maintaining routing tables or distributing link-state information, but they are adapted to the special requirements of mobile ad hoc networks. Topology-based protocols can be further divided into three groups:
  - *Proactive* protocols: they try to maintain consistent routing information within the system so that at any time, every node knows how to route packets to all other nodes in the network. A disadvantage is that usually periodic exchanges of routing information among the nodes are required. If only a few pairs of nodes communicate with each other, proactive protocols can waste a lot of resources unnecessarily. An advantage is that as routing information is always up-to-date and available, packets can be sent to any destination virtually with no delay. Examples of proactive routing protocols are OLSR [65] and DSDV [187];
  - *Reactive* protocols: a route is established between a source and a destination only when it is needed. For this reason, reactive protocols are also called *on-demand* protocols. A disadvantage is that it may happen that a node wants to communicate with another node but no working route to that other node is available. Thus the communication must be delayed until such a route is discovered. Examples of on-demand protocols are DSR [129] and AODV [186];
  - *Hybrid* protocols: they try to combine the advantages of the proactive and the reactive approaches. An example of an hybrid protocol has been proposed in [107];
- *Position-based* protocols: they use information about the physical locations of the nodes to route data packets to their destinations. The advantage of position-based routing is that the nodes do not need to maintain routing information or to discover routes explicitly, and therefore the control overhead of these protocols tends to be smaller. The disadvantages that they rely on additional hardware, as GPS or some other positioning service, in each node or some other mechanisms by which the nodes can determine their own location.

Initial work in ad hoc routing has considered only the problem of providing efficient mechanisms for finding paths, without considering security. In wireless networks

routing control messages are sent over wireless channels. However, due to the lack of physical protection, some of the routers could be corrupted and not follow the routing protocol faithfully [136]. Obviously, this can have undesirable effects on the operations of the network. There are a number of different attacks to manipulate the routing in an ad hoc network. According to [209], typical routing attacks are:

- *Modification* attacks, which cause a redirection of network traffic and DoS attacks by altering control message fields or by forwarding routing messages with fake values;
- *Impersonation* attacks, also known as *spoofing*, when a node misrepresents its identity;
- *Fabrication* attacks, which involve the generation of false routing messages.

Since an adversary can paralyse the activity of a network by attacking this basic functionality, several "secure" routing protocols have been proposed. Comprehensive surveys of these protocols can be found in [57,120]. Among them we remember, SRP [184], Ariadne [121], endairA [9], SAODV [238], SEAD [119] and ARAN [209].

2.3 Time Synchronisation

For most applications and algorithms that run in a wireless system, time is a central aspect. Let us think to sensor networks. In them, each device is equipped with sensor hardware to sense physical phenomena. The data sensed by these devices have to be combined and evaluated to derive knowledge about the environment where they are working. This operation is called *data-fusion*. It in turn allows to the devices to react intelligently to changes in their environment. For all these activities, in order to determine the direction of the phenomena, temporal relationships among different events originated by different devices, i.e. the event X has happened before the event Y, have to be exhibited. Moreover, to estimate the speed of the phenomena, real-time issues, i.e. X and Y have happened within a specific time interval, and then time differences between events originating from different devices have to be calculated. The fusion of individual sensor readings is possible only by exchanging messages that are timestamped by each device's local clock.

This is just one example attesting the importance of time in wireless systems. Many other situations can be exhibited. For instance, many security protocols require the presence of timing information, because they present parameters that vary with time. Usually these parameters certify the freshness of a message. In this setting, some security properties, such as authentication and secrecy, can be reformulated in a timed setting. All this requires the need for a common notion of time among the devices.

A *computer clock* is an electronic device that counts oscillations in an accurately-machined quartz crystal, at a particular frequency. It is also defined as an ensemble of hardware and software components used to provide an accurate, stable, and reliable time-of-day function to the operating system and its clients. Computers clocks are essentially *timers*. The timer counts the oscillations of the crystal, which is associated with a counter register and a holding register. For each oscillation in

the crystal, the counter is decremented by one. When the counter becomes zero, an interrupt is generated and the counter is reloaded from the holding register. Therefore, it is possible to program a timer to generate an interrupt 60 times a minute, where each interrupt is called a *clock tick*, by setting an appropriate value in the holding register. At each clock tick, the interrupt procedure increments the clock value stored in memory. The term *software clock* normally refers to the time in a computer clock to stress that it is just a counter that gets incremented for crystal oscillations. The interrupt handler must increment the software clock by one every time an interrupt (i.e., a clock tick) occurs.

The clock value can be scaled to get the time of the day; the result can be used to timestamp an event on that computer. In practice, the quartz crystals in each of the devices will run at slightly different frequencies, causing the clock values to gradually diverge from each other. This divergence is formally called the *clock skew*, which can lead to an inconsistent notion of time. Clock synchronisation is performed to correct this clock skew.

In centralised systems, there is no need for synchronised time because there is no time ambiguity. A device gets the time by simply issuing a system call to the kernel. When another process tries to get the time, it will get either an equal or a higher time value. Thus, there is a clear ordering of events and the times at which these events occur. On the other hand, in distributed systems each device has its own internal clock and its own notion of time. In practice, these clocks can easily drift seconds per day, accumulating significant errors over time. Also, because different clocks tick at different rates, they may not remain always synchronised although they might be synchronised when they start. This clearly poses serious problems to applications that depend on a synchronised notion of time. In distributed systems, where each machine has its own physical clock, clock synchronisation has significant importance. *Clock synchronisation protocols* ensure that physically distributed processors have a common notion of time. It has a significant effect on many areas like security systems, fault diagnosis and recovery, scheduled operations, database systems, and real-world clock values.

Researchers have developed successful clock synchronisation protocols for wired networks over the past few decades. However, due to the features of wireless systems, neither logical time [141, 142] nor classical clock synchronisation algorithms [143, 213] are applicable in their setting. Indeed, algorithms implementing logical time assume that causal relationships manifest themselves in network messages between entities generating the event. This is not the case of real world where wireless networks operate. Instead, classical clock synchronisation algorithms are not adapted to wireless systems because they rely on two important assumptions: (i) the ability to periodically exchange messages between nodes that have to be synchronised, and (ii) the ability to estimate the time it takes for a message to travel between two nodes to be synchronised. In the wireless scenario, these assumptions cannot be made, especially when we consider mobile devices; for example, the frequent temporary networks partitions may not permit to devices to be synchronised when they sense two different events. Moreover, in wireless path may be arbitrarily delayed, ruling out good estimation of message delay. Thus the assumptions (i) and (ii) above are violated. Finally, a wireless network could comprise a large

number of devices to be synchronised. This constitutes a scalability challenge for classical synchronisation protocols.

During the last years, many appropriate clock synchronisation protocols tailored for wireless networks are developed. Good overviews of them can be found in [76, 221].

In [221] the authors have classified synchronisation protocols based on two kinds of features:

1. **Synchronisation issues**: in this kind of classification, we can find different options:
   - *Master-slave* vs *peer-to-peer synchronisation*: the former assigns one node as the master and the other nodes as slaves. The slave nodes consider the local clock reading of the master as the reference time and attempt to synchronise with the master. In the latter any node can communicate directly with every other node in the network. This eliminates the risk of master node failure, which would prevent further synchronisation. Even if peer-to-peer configurations offer more flexibility, they are more difficult to control;
   - *Clock correction* vs *untethered clocks*: in the first option, the local clock in each node is correct by running on a par with a global time scale or an atomic clock, used to provide a convenient reference time. In untethered approach, the protocol tends to achieve a common notion of time without synchronisation. This approach is widely spreading as it allows to save a considerable amount of energy;
   - *Internal synchronisation* vs *external synchronisation*: in internal synchronisation approach a global time base, called real-time, is not available from within the system and the goal is to minimise the maximum difference between the readings of local clocks of the nodes. In external synchronisation approach a standard source of time such as Universal Time (UTC) is provided. There is an atomic clock that provides actual real-world time, usually called reference-time. The local clocks of nodes seek to adjust to this reference time in order to be synchronised;
   - *Probabilistic* vs *deterministic synchronisation*: the former provides a probabilistic guarantee on the maximum clock off-set with a failure probability that can be bounded or determined. The latter guarantees an upper bound on the clock offset with certainty;
   - *Sender-to-receiver* vs *receiver-to-receiver synchronisation*: the first method synchronises a sender with a receiver by transmitting the current clock values as timestamps. The second method is based on the statement of fact that if any two receivers receive the same message in single-hop transmission, they receive it at approximately the same time. Thus the receivers exchange the time at which they received the same message and compute their offset based on the difference in reception times.
2. **Application-dependent features**: the options of this kind of classification are the following:
   - *Single-hop* vs *multi-hop networks*: in a single-hop network, a node can directly communicate and exchange messages with any other node in the network. The need for multi-hop communication arises due to the increase

in the size of wireless networks. In such settings, nodes in one domain communicate with nodes in another domain via an intermediate node. Communication can also occur as a sequence of hops through a chain of pairwise-adjacent nodes;

- *Stationary networks* vs *mobile networks*: in stationary networks, nodes do not move. On the contrary, in a mobile network, nodes can move, and they connect with other nodes only when entering in communication range of them. Mobility is an inherent advantage of a wireless environment but it induces more difficulties in synchronisation. It leads to frequent changes in network topology and demands that the protocol be more robust, because it requires re-synchronisation of nodes and recomputation of the neighbourhoods;
- *MAC-layer based approach* vs *standard approach*: the Media Access Control (MAC) layer is a part of the Data Link Layer of the Open System Interconnection (OSI) model. This layer is responsible for (i) providing reliability to the layers above it with respect to the connections established by the physical layer and (ii) preventing transmission collisions so that the message transmission between one sender and the intended receiver nodes does not interfere with transmission by other nodes.

2.4 Security Aspects

The Merrian-Webster dictionary on-line [162] defines *security* as *the quality or state of being secure - to be free from danger*. This definition seems to be very generic; it is particularly referred to the protection of life, things, resources from damage due to either intentional or unintentional human actions or natural catastrophes, e.g. earthquakes, hurricanes, storms and so on. When we want to explicitly refer to the protection of information, we have to talk about *information security*, meaning protection of information and information systems from unauthorised access, use, disclosure, disruption, modification or destruction. Information security is defined as *the preservation of confidentiality, integrity* and *availability of information* [127], commonly known as the CIA (Confidentiality, Integrity and Availability) properties.

When designing communication systems, it is essential to include more strict security requirements. In particular, in wireless networks the situation is very complicated for different reasons. Due to their peculiar features, they are vulnerable against both external and internal attacks. Due to the open nature of wireless medium, attacking its availability is not a complex task. Without protection, anyone in the transmission range of the sender can intercept its signal. Mobile wireless devices usually have limited resources, such as bandwidth, storage space, processing capability, and energy. In this context, it is challenging to implement and use the cryptographic algorithms and protocols required for the creation of security services. Indeed costly security solution may not be affordable. Moreover, some devices, as sensors, are not tamper-resistant; thus it is possible to reprogram or simply destroy them. Furthermore, due to their inherent distributed nature, it is very difficult to localise the failure point or the actual cause of malfunction.

Even worse, very often wireless networks are employed in hostile environment, such as battlefields, or in emergency situations, as in hospitals, or in other civilian and scientific applications, such as in monitoring of pollution. In all these conditions, wireless devices are exposed to many threats, some ones due to atmospheric problems or to technical failures, but the most ones are due to human misbehaviours. These last threats introduce relevant issues, as it is impossible, or anyway extremely complicated, to prevent the human actions and consequently it is extremely difficult to anticipate the kind of misbehaviour that will affect a network while not yet deployed. In addition, rapid deployment of new networking technologies and of new services are encouraged by competition. Indeed, wireless technologies are changing rapidly and new features and products are continuously introduced. As a result, there is no time to put right and implement complete protection mechanisms. Consequently, very often the protection mechanisms are designed a posteriori and can be not appropriate.

More difficulties arise when considering wireless networks without fixed infrastructure. Compared with infrastructure-based wireless networks, security management is more challenging due to unreliable communication, intermittent connection, node mobility, and dynamic topology. As we stated, these systems do not have any entry points such as routers, gateways, etc, then nodes have to configure and organise themselves in a cooperative way without external supports. On the contrary, fixed infrastructures are typically present in wired networks and can be used to monitor all network traffic that passes through them. A mobile node can see only a portion of a network, that is the packets it sends or receives together with other packets within its radio range. This introduces some difficulties.

Summarising, compared with traditional networks, the security management in wireless systems is more difficult due to the following features:

- *resource constraints*: they reduces the possibility of designing costly security solutions;
- *unreliable communications*: the shared-medium nature and unstable channel quality of wireless links may result in high packet-loss rate and re-routing instability. This implies that the security solution in wireless ad hoc networks cannot rely on reliable communication;
- *node mobility* and *dynamic topology*: nodes can leave and join the network and move independently, so the network topology can change frequently. The highly dynamic operation of a MANET can cause traditional techniques to be unreliable. Indeed, as nodes are mobile they may enter or leave the a route frequently, adding complexity in the design of protocols for these networks;
- *scalability*: it is a key problem when we consider a large network size.

In [57, 106, 128, 150] we can find an exhaustive analysis of security vulnerabilities, threats, attacks, requirements, and challenges for wireless networks. In Sections 2.4.1 and 2.4.2 we give an overview of them.

2.4.1 Security Threats and Vulnerabilities

As we stated, the first obvious feature of wireless networks is that communication takes place over a wireless channel. Such a channel suffers from a number

of vulnerabilities. *Attacks on channel* include any actions that aim at interrupting, disturbing, or stopping the normal activities of the system, intentionally or unintentionally. We can divide these attacks into five categories:

- *Passive attacks*: they are able to retrieve and copy data from the network, but do not influence over its behaviour. Indeed passive attacks consist in listening to the traffic in the network and analysing the captured data without interacting with the network. *Eavesdropping* is a common type of passive attack; it can be defined as the interception of information or data by an unintended party. It consists in placing an antenna at an appropriate location and overhearing the information that the victim transmits or receives. Usually, the protection against such misdeeds is achieved by encrypting that information;
- *Active attacks*: they directly obstruct the development of the services. An active attacker try to modify the content of the message exchanged between parties: in this way the attacker can capture, delete, redirect or alter the data so that the potential recipient of the message will no more be able to receive it. Moreover, an active attacker can also modify the routing or the control messages of the network. Finally, the attack can be made in terms of consumption of resources, battery or memory, computational power of the network. An example of this kind of attacks is *the Man in the Middle (MitM)*: an attacker interposes between two parties for suspicious purposes;
- *External attacks*: they come from outside the logical communication group or network. For example the external attacker may be a person using an high power transceiver able to launch a remote attack. The attacker can also be an internal node of the network, carrying out some attacks against an adjacent communication group;
- *Internal attacks*: they include the attacks originating from within the same communication group. With respect to external attacks, they are more difficult to prevent and for this reason they can be more dangerous;
- *Misbehaviour*: it can be considered an internal active attack as it is an unauthorised behaviour of an internal node of a network. More precisely this node does not appropriately collaborate in the cooperative process of its neighbourhood. The scope of a misbehaving node is not to damage other nodes but to save its energy, memory and resources. For these reasons, it is called *selfishness*. If a network has a centralised control authority, the selfishness attack does not occur.

From the *functional point of view*, the different threats can be organised in the following categories:

- *Common attacks*: this category comprises all passive and active attacks on wireless channel;
- *Denial of Service Attacks* (DoS): it is a special class of active attacks, deserving a category of its own. It attempts to disrupt the function of a service, making network resources unavailable to its intended users or destroying the services availability. This kind of attack can also be carried out by a group of collaborating attackers. The disruption can range from physical destruction of network equipment (*power exhaustion attack*) to attacks to the occupation of network's bandwidth (*jamming attack*). In the first one, an attacker imposes a

particularly complex task to a node in order to consume its battery charge (in case of sensor networks) or to fill its buffers or memory. In jamming attacks, the attacker constantly emits radio frequency signals, thus any member of the network in the affected area will not be able to send or receive any packet;

- *Node compromise*: in general a node is *compromised* when it is under the control of an attacker, that can gain access or control to the node itself after its deployment and can either read or modify its internal memory. The ultimate goal of this attack is, in most cases, to obtain the secret keys stored within a trusted node;
- *Side-channel attacks*: these are attacks on the cryptographic materials of a node. In order to compromise a node, it is also possible to attack its hardware through this kind of attacks. The main objective of side channel attacks is to obtain confidential data stored within the node;
- *Impersonation attacks*: the *cheat on identities* is numbered in this category. The attacker pretends to be a legitimate node in a communication group by forging the valid identity of a legitimate node. Examples of this kind of attacks are the *replication* and the *Sybil* attacks. In the former a malicious node can create multiple fake and pseudonymous identities. In the Sybil attack [180], a node use multiple identities to deceive other nodes. The attacker can either fabricate new identities or steal them from legitimate nodes;
- *Protocol-specific attacks*: wireless networks base their correct engineering on many different protocols (routing, data aggregation, time synchronisation ...) Specific attacks are direct to them. As a result, the internal services of the network are influenced.

2.4.2 Security Requirements

The effects of attacks in a network are not trivial, because very often they make the services unavailable. For this reason it is necessary to develop security mechanisms aimed either at preventing the attacks or at minimising their effects. In this section, we deal with the requirements usually expected for secure systems and in particular how these requirements are fulfilled in wireless networks. In [218] we can find a description of the basic requirements that are considered universal in any communication network. However, for wireless networks, additional requirements must be considered. Thus, to the so called CIA properties, we need to add new appropriate requirements. Again in [57, 128, 150] we can find an exhaustive list of these requirements:

- *Confidentiality*: it is the term used to indicate the preventing of the disclosure of information to unauthorised individuals or systems. It is based on the principle that transmitted data must be known only by specific and authorised parties. In order to provide this level of security, data must be encrypted. In some cases, it may be not mandatory, for instance where the data is public by itself (e.g. a temperature of a room) and when no other information can be derived from it. On the other hand, in many particular situations (e.g. a private household, a credit card transaction) the physical data obtained by the network can be deemed as sensitive, and then must be read only by internal and authorised entities. Moreover, certain cryptographic materials, such as security credentials

and secret keys, must be hidden from unauthorised and external entities. If an unauthorised party obtains sensitive information in any way, a breach of confidentiality has occurred;

- *Integrity*: it means that data cannot be maliciously modified or deleted without authorisation. This ensures that the received data are the same as those sent. Unlike confidentiality, integrity is, in most cases, a mandatory property. The data to be protected are not only the users' data, but also the data related to the control of the network;
- *Authentication*: it is necessary to ensure that data, transactions, communications or documents are genuine. It is also important for authenticity to validate that both parties involved are who they claim to be. In other words, nobody cheats on identities, pretending to be another party. In wireless networks, and in particular in mobile wireless networks, the authentication requirement is very important to make possible the legitimation of registered members of the network;
- *Authorisation*: it is the basic requirement for access control mechanisms, granting appropriate access to resources (connectivity, data, information providing, ...) based both on the user's identity and the organisation's policy. Authorisation requirement allows that only authorised nodes in a network can perform specific tasks;
- *Availability*: the services of a wireless network must be accessible by its users whenever it is necessary, in normal activities or under attacks. This requirement should also provide higher priority to very important communications, such as an emergency call from a cellular phone and it should also guarantee a fair distribution of the radio channel to mobile users in the transmission range;
- *Freshness*: the data produced by a network must be recent. This requirement is particularly important for sensor networks, because they are data-centric and their existence is due to the collection of physical data from an environment;
- *Forward* and *Backward Secrecy*: a node can be deployed to substitute a failed node. In this context, there are two important properties to be considered: forward secrecy and backward secrecy. The first one is referred to the feature of substituted node to not be able to read any future messages after it leaves the network, whereas the second one indicates that a substitutive node should not be able to read any previously transmitted message;
- *Self-Organisation*: it is a specific property related to the autonomous nature of ad hoc networks. Nodes must be independent and flexible enough to autonomously react against problematic situations, organising and healing themselves. These problematic situations can be caused either by external or internal attackers trying to influence over the behaviour of the elements of the system or by extraordinary circumstances in the environment or in the network itself;
- *Auditing*: this property is particularly referred to sensor networks. The elements of a sensor network must be able to store any significant events that occur inside the network. Auditing information is useful to analyse the behaviour of the system after a failure, e.g. to examine the causes of such a failure. This property is also closely related to self-organisation: in order to adjust their behaviour, sensor nodes must be able to know the state of their neighbourhood;

- *Non-repudiation*: a node cannot deny the sending of a message it has previously sent. It can be also considered the repudiation of receipt, where the recipient tries to deny the reception of the message. Moreover, it should not be possible for a user of a given service provided by a given operator to deny that he did not use that service. In other words, the operator must to be able to prove that the specific user really used that specific service. In order to overcome this problem, it is necessary to used certain "evidence", that is redundant piece of information, as a nonce or a timestamp, that uniquely identify the message. They can be used to detect and discard duplicates and replay of the same messages;
- *Privacy and Anonymity*: in some situations, the nodes should not reveal their location nor the party with which they communicate. As confidentiality, privacy and anonymity are not mandatory properties.

2.5 Mobility vs Security

In wireless networks, *mobility* is associated with the ability of a user to access telecommunication services from different locations and different devices. With the term *nomadism* it is indicated a discrete terminal mobility which implies the ability of the terminal to be connected to different networks, for example, at home and in the office. Session continuity is not supported in this case. In a distinguished way from nomadism, the term mobility implies continuous uninterrupted user access to a service even while changing location or device. Ubiquitous mobility is often expressed in terms of "anywhere, anytime and any device" connectivity. Wireless does not mean imperatively mobile, as mobility is a service; its realization requires additional support from both the part of the network and the user. A user can always move within a WiFi cell but without mobility support he cannot move to a neighbouring cell. As a consequence, the use of wireless devices raises mobility support requirements.

Mobility classification includes user, session, code and terminal mobility. *User* or *personal mobility* allows a user to be reachable on different terminals at the same logical address. *Session mobility* allows a user to continue a session even when changing a terminal. *Service mobility* allows a user to maintain access to his services while changing devices or network service providers. It should be possible for a user to update services definitions from any terminal. Thus, with service mobility, a user has the possibility to store his preferences either locally or at the dedicated server, that is associated with the user's address. *Code mobility* allows software entities (codes, objects or processes) to be relocated or moved from one terminal to another during their execution.

In this dissertation, we concentrate on the terminal mobility in ad hoc wireless networks. *Terminal mobility* allows a device to change its location while continuing to maintain all services and sessions running. Mobile devices break links with old neighbours and establish fresh links with new devices. The end points of these new links may belong either to the same or to different subnets, either to the same or different administrative domains and they may support the same or different access technologies. Terminal mobility may be classified into micromobility and

macromobility, according to the locality impact. *Micromobility* refers to mobility over a small area. Usually this means mobility within a single IP domain. *Macromobility* represents mobility over a large area. This includes mobility support and associated address registration procedures that are needed when the mobile node moves between IP domains.

From the description of mobility above provided, it can be noticed that its support introduces new challenges and increases the complexity of modern wireless systems. In the traditional digital world, the main security requirements such as confidentiality, integrity, availability and non-repudiation were addressed by installing firewalls at network borders and security level monitoring. Fixed communication may be seen as a static environment having more or less defined threats, borders, lists of communicating entities (and identities) and static, mostly direct trust relationships between them. The arrival of mobile communications brings a dynamic aspect into the digital environment and extends security-related requirements. Not only confidentiality of data but also confidentiality of location, traffic and identity should be addressed. The use of cryptography becomes vitally necessary but difficult to implement because of the limited processing capabilities of mobile devices. The need to maintain a security state while moving among networks introduces new threats to security solutions. According to [57], mobility has the following security implications:

- devices become a way to permanently trace the movements of a user and hence jeopardising his privacy;
- mobility also means that a given device must be able to roam across wireless networks controlled by different operators. As mentioned above, this requires that appropriate roaming agreements are made between operators;
- to be mobile the device must be small, meaning that it has limited storage, computing power, and energy. This can lead to poor engineering of the security protocols;
- a mobile station can easily be stolen, with the risk that it is misused or reverse engineered and that the data that it contains are accessed.

In a mobile network, secure channel binding must be provided in order to assure that end-points of a secure channel are the same of those authenticated by each other. This requirement becomes very important since a masquerade attack can be easily realized in a mobile scenario. Moreover, a visited network should identify and authenticate a user that asks the network to grant access, dynamically build trust relations with other administrative domains in order to authenticate a visitor, distribute/negotiate encryption keys with users, correctly account user's activity and recognise malicious behaviour in a visitor.

Obviously, in this scenario it is critical to know what entities can access what resources, from where and when. A user is often served by an entity that is not his identity provider and trust establishment between a user and a service provider becomes a necessity. In a wireless context, new security mechanisms have to be developed as complex processes of entities recognising, definition of privileges and restriction for them and access granting according to these privileges. We analyse more accurately these issues in the next chapter.

2.6 Chapter Summary

Wireless networks are quickly becoming the networks of choice. This is not only due to the large bandwidth, but also to the flexibility and freedom they offer. As stated in [57], new trends in the design of wireless networks lead to increase the number of security challenges and extend security related requirements in terms of safety and cooperation.

The trend in next-generation wireless systems is toward an IP-based infrastructure with the support of heterogeneous wireless access technologies. One of the challenges for these systems is the design of efficient mobility management solutions.

Many often, there is no direct communication link between the two nodes that want to communicate. However, it is possible to pass through one or more intermediate nodes. This is known as a *multi-hop*. Multi-hopping communication increases the "security distance" between the device under the control of the operator (base station, access point) and/or the mobile station. Consequently, appropriate measures must be taken to prevent misbehaviours against the correct functionality of the network.

Wireless devices provide more flexibility to the users as they are programmable; for example a user can easily increase the performances of a device by installing new applications on it. But at the same time, this flexibility can be misused to prepare attacks; in this context selfishness may become an uncontrolled threat.

In this chapter, we provided a classification of wireless networks, according to the coverage area of the devices. Then we described the most popular network types, as cellular, ad hoc, vehicular, mesh and sensor networks. We briefly described some applications in the area of short range communication, as Bluetooth, IEEE 802.11 and IEEE 802.16. We provided a more detailed description of wireless routing protocols.

We described clock synchronisation protocols that provide such a common notion of time. Finally, we dealt with security and with mobility aspects.

In the next chapter we discuss traditional access control mechanisms, focusing on their inadequacy for wireless systems. Then we introduce the concept of trust and describe trust management systems.

3

Security Mechanisms

3.1 Introduction

The purpose of *access control* is to limit the actions or operations that a legitimate user of a computer system can perform. Access control constrains what a user can do directly, as well what programs executing on behalf of the users are allowed to do. In this way access control seeks to prevent activity which could lead to a breach of security. A widely applied access control policy is the *access control list*. Roughly speaking, an access control list is a list of permissions attached to an object. It specifies which users or system processes are granted access to objects, as well as what operations are allowed to be performed on given objects. ACLs work well when access policies are set in a centralised manners. However, they are less suited to ubiquitous systems where the number of users may be very large (think of sensor networks) and/or continuously changing. In this scenario users may be potentially unknown and, therefore, untrusted.

Emerging wireless systems have one issue in common: the need to grant or restrict access to resources according to some security policy. However, different systems and applications have different notions of what a resource is. Moreover different applications have different access-granting or -restricting policies. The criteria on which a decision is based may differ greatly among the various applications (or even between different instances of the same application). The security mechanism should be able to handle those different criteria.

In [131] Jøsang has distinguished between *hard security*, for the traditional information security mechanisms, and *soft security*, for the so called social control mechanisms. Cryptographic algorithms and firewalls are examples of hard security mechanisms, and they have the general property of allowing complete access or no access at all. Hard security also assumes complete certainty. It is clear how these strong assumptions cannot meet the features of wireless networks. While the goal of traditional information security is to preserve the CIA properties, the goal of soft security mechanisms is to stimulate the quality of a specific community in terms of the ethical behaviour and the integrity of its members. Soft security mechanisms seem to be well suited for wireless systems. Among these mechanisms, we can find the ones that take into consideration the concept of *trustworthiness*.

Current security technology offers us some capability to build in a certain level of security into our communication, i.e. cryptographic algorithms for privacy and digital signatures, authentication protocols for proving authenticity and access control methods for managing authorisation and so on. However, these methods cannot manage the more general concept of trustworthiness, that is essential in open and distributed environments as wireless settings. In everyday life, every interaction between persons or organisations is based on some kind of trust relationships. In the modern wireless world, the concept of trust plays a significant rôle because relations in the virtual world more and more reflect real life.

Trust and security are two concepts that cannot be desegregated. Also the concepts of trust and *reputation* are closely related in trust management systems and they are firmly routed in sociology and psychology. In a first approximation, trust can be defined as the belief that another party (a person, an organisation, but also a device) will behave according to a set of well established rules and will thus meet expectations of another party. Focusing on computing-oriented definitions, trust is a binary directional relationship between two parties, called *trustor* and *trustee*. In general, the trustor is the subject that trusts an entity or a service, and the trustee is the entity that is trusted. Trust information is usually represented as a collection of assertions on the reliability of the parties. The trust establishment process includes specification of valid assertions, their generation, distribution, collection and evaluation. Trust evaluation is performed by applying specific policies to assertions; the result is a trust relation between the trustor and the trustee.

The *trust management approach* to distributed system is a soft security mechanisms developed as an answer to the inadequacy of traditional authorisation mechanisms. Trust management engines avoid the need to resolve identities in an authorisation decision. Instead, they express privileges and restrictions in a programming language. A trust management systems (TMS) may be either a *credential-based* or a *behaviour-based* system. In credential-based systems peers or nodes use certificates in order to establish trust with other peers. Behaviour-based systems are often called experience-based as in these models an entity A trusts another entity B based on its experience on B's past behaviour. These systems heavily rely on the concept of reputation.

We end this introduction with an outline of the present chapter. In Section 3.2 we describe traditional access control, providing an overview of the most common mechanisms and policies. In particular, we describe in Section 3.2.1 the Access Control List and in Section 3.2.2 some access control policies. Then we provide a classification of the policies according to their administration in Section 3.2.3. In Section 3.3 we introduce trust context. In particular in Section 3.3.1 we analyse security issues related to trust and in Section 3.3.2 we describe the concepts of trust and reputation. In Section 3.4 we describe the trust management approach and in Section 3.4.1 we analyse trust management systems for MANETs. Finally, in Section 3.5 we give a summary of the chapter.

3.2 Traditional Access Control

Access control is the process of limiting actions or operations that a legitimate user of a computer system can perform. It constraints what a user can do directly, as

well as what programs executing on behalf of the users are allowed to do. Access control principles were proposed by Sandhu and Samarati in [207]. In their paper, the authors have stated that access control is enforced by a *reference monitor*: any attempted access by a user, or program executing on behalf of that user, to objects in the system has to be mediated by the reference monitor. An authorisation database, administered and maintained by a security administrator, records the associations between each user and the operations that the user is actually authorised to perform. The administrator sets these authorisations on the basis of the *security policy* of the organisation. When the reference monitor receives a request from a user, it consults the authorisation database in order to determine if that user can perform that operation. It is important to make a clear distinction between *authentication* and access control. Correctly establishing the identity of the user is the responsibility of the authentication service. Access control assumes that authentication of the user has been successfully verified prior to enforcement of access control via a reference monitor. For these reasons, it is important to understand that access control is not an exhaustive solution for securing a system but it has to coexist with other security services.

It is also important to understand the distinction between *policies* and *mechanisms*. Policies are high level guidelines which determine how accesses are controlled and access decisions determined. Mechanisms are low level software and hardware functions which can be configured to implement a policy. A desirable goal is to develop access control mechanisms which are largely independent of the policy for which they could be used. In this manner, mechanisms can be used to serve a variety of security purposes.

In general, there do not exist bad or good policies but there exist policies which ensure more protection than others. However, not all systems have the same protection requirements. Policies suitable for a given system may not be suitable for another. For instance, very strict access control policies, which are crucial to some systems may be inappropriate for environments where users require greater flexibility. As a consequence, the choice of access control policy depends on the particular features of the environment to be protected.

In this section we give an overview of most popular access control policies and mechanisms for traditional systems. We briefly discuss why they are inappropriate for wireless systems.

3.2.1 Access Control List

Over the years, a number of abstractions have been developed in dealing with access control. Perhaps the most fundamental of these is the realization that all resources controlled by a computer system can be represented by data stored in objects, e.g., files. Therefore protection of objects is the crucial requirement, which in turn facilitates protection of other resources controlled via the computer system.

Any activity in a system is initiated by entities known as *subjects*. Subjects are typically users or programs executing on behalf of users. Subjects initiate actions or operations on *objects*. These actions are permitted or denied according to the authorisation modes established in the system.

In the simplest case, access policies are expressed by an *access matrix* [14], where there is a row for each subject, and a column for each object. Each cell of

the matrix specifies the access authorised for the subject in the row to the object in the column. The task of access control is to ensure that only the operations authorised by the access matrix actually get executed. As stated in [207], this is achieved by means of a reference monitor, which is responsible for mediating all attempted operations by subjects on objects. Note that the access matrix model clearly separates the problem of authentication from that of authorisation. In systems using an access matrix, all users are known to a system in advance. Obviously, this is not the case of wireless systems. Moreover, access control matrices do not provide a scalable solution. It might not only impose performance problems but also increases the probability of an administration mistake. In a large system the access matrix will be enormous in size, and most of its cells are likely to be empty. In order to overcome these limitations and implement access policies in more compact and efficient way several solutions have been proposed. An example in this direction is the *Access Control List*.

An Access Control List (ACL) [207] is a list describing which access rights a user has on an object (resource). Such a list of users is defined together with the corresponding resource. This approach corresponds to storing the matrix seen above by columns. With ACLs it is easy to determine which modes of access a subject has for an object. It is also easy to revoke all accesses to an object by replacing the corresponding list with an empty one. On the other hand, to determine all the accesses that a subject has, it is necessary to examine the ACL of every object in the system with respect to a subject. Similarly, if all accesses of a subject need to be revoked, all lists must be visited one by one. In practice revocation of all accesses of a subject is often done by deleting the user account corresponding to that subject. This is acceptable if a user is leaving an organisation. However, if a user is reassigned within the organisation, it would be more convenient to retain the account and change its privileges to reflect the changed assignment of the user.

ACLs do not need to physically exist in one location but may be distributed throughout the system. ACLs have been used in distributed systems because they are conceptually easy to understand. However, there are a number of fundamental reasons for which ACLs are inadequate for wireless networks. For example, ACLs are suitable to situations where access policies are set in a centralised manner, but they are less suited for the situation where the system interacts with a large amount of users that is continuously changing. Even if they are simple to implement, they are not efficient for doing security checks at runtime. Blaze, Feigenbaum and Keromytis in [45] and Krukow in [138] have provided a list of these inadequacies and their reasons:

- *Authorisation = Authentication + Access Control List*: authentication deals with establishing identity. In traditional static environments, e.g., operating systems, the identity of an entity is well known. In Internet and wireless applications this is often not the case. This means that before an access control list is to be used some form of authentication must have been performed. In distributed systems, often public key based authentication protocol are used, which usually relies on centralised and global certification authorities;
- *Delegation*: it is necessary for scalability of a distributed system. It enables decentralisation of administrative tasks. Existing distributed system security mechanisms usually delegate directly to a certified entity. In such systems,

policy (or authorisations) may only be specified at the last step in the delegation chain (the entity enforcing policy), most commonly in the form of an ACL. The implication is that high level administrative authorities cannot directly specify overall security policy but they can only certify lower level authorities. This authorisation structure leads easily to inconsistencies among locally specified sub-policies;

- *Expressibility* and *Extensibility*: a generic security mechanism must be able to handle new and several conditions and restrictions. The traditional ACL approach has not provided sufficient expressibility or extensibility. Thus, many security policy elements that are not directly expressible in ACL form must be hard coded into applications. This means that changes in security policy often require reconfiguration, rebuilding, or even rewriting of applications;
- *Local trust policy*: the number of administrative entities in a distributed system can be quite large. Each of these entities may have a different trust model for different users and other entities. For example, system A may trust system B to authenticate its users correctly, but not system C; on the other hand, system B may trust system C. It follows that the security mechanism should not enforce uniform and implicit policies and trust relations.

3.2.2 Access Control Policies

We now discuss some different policies which commonly occur in computer systems:

- classical mandatory policies;
- classical discretionary policies;
- the Bell-LaPadula Model.

We have added the "classical" adjective to the first two items to underline that mandatory and discretionary policies have been recognised by security researchers and practitioners for a long time. However, in recent years there is increasing consensus that there are legitimate policies which have aspects of both of these. The Bell-LaPadula model is an example of this fact.

Mandatory Access Control Policies

Mandatory Access Control (MAC) refers to a type of access control by which the operating system constraints the ability of a subject (or user) to access or more generally to perform some specific operations on an object. A subject is always a process or a thread, whereas objects are files, directories, memory segments and so on. *Security levels* are associated to subjects and objects. The security level associated with an object reflects the sensitivity of the information contained in that object, i.e, the potential damage which could result from unauthorised disclosure of the information. The security level associated with a subject reflects the subject's trustworthiness not to disclose sensitive information to unauthorised subjects. In the simplest case, the security level is an element of a hierarchical ordered set. A *dominion* relation is established among the elements of the set, according to the chosen order relation. For example, if the order relation is $\geq$ then we say that a level l *dominate* a level l' if $l \geq l'$.

Any operation by any subject on any object is tested against the policy, in order to decide if the security requirements are enough to allow the operation. The policy is the set of authorisation rules enforced by the operating system, Thus, whenever a subject attempts to access an object, an authorisation rule examines the policy and decides whether to grant the access or not. In particular, the following two principles are required to hold:

- *Read down*: a subject's security level must dominate the security level of the object being read, according to the order relation of the hierarchical set;
- *Write up*: a subject's security level must be dominated by the security level of the object being written, according to the order relation of the hierarchical set.

Satisfaction of these principles ensures secrecy, because it prevents more sensitive information, that is those contained in high level objects, to flow to objects at lower levels.

Mandatory access control can as well be applied for the protection of information integrity. The integrity level associated with an object reflects the degree of trustworthiness that can be placed in the information stored in the object, and the potential damage that could result from unauthorised modification of the information. The integrity level associated with a user indicates the user's trustworthiness for inserting, modifying or deleting data and programs at that level. Similarly to secrecy, for integrity the following principles are required:

- *Read up*: a subject's integrity level must be dominated by the integrity level of the object being read, according to the order relation of the hierarchical set,
- *Write down*: a subject's integrity level must dominate the integrity level of the object being written, according to the order relation of the hierarchical set.

Satisfaction of these principles safeguard integrity by preventing less reliable information, i.e. stored in low objects, to flow to high objects.

With mandatory access control, the security policy is centralised and it is directly controlled by a security policy administrator. Users do not have the possibility to override the policy and, for example, grant access to resources otherwise restricted. Then there is a central policy to be enforced for all users.

Discretionary Access Control Policies

Similarly to mandatory policies, *Discretionary Access Control* (DAC) policies govern the access of users to the information on the basis of the user's identity and authorisations. For each subject and each object in the system, the authorisations specify the access modes, e.g., read, write, or execute, that the subject can perform on the object. Each request of a subject to access an object is checked against these authorisations.

Even if both discretionary and mandatory access control govern the ability of subjects to access objects, on the contrary on mandatory, discretionary access control allows users the possibility to make policy decision or assign security levels. Let us consider a system in which user can create objects. Thus in DAC systems the individual user may, at his own discretion, determine who is authorised to access the objects he creates. Instead, in MAC systems, the creator of an object,

if he is not the central authority, does not have the ability to determine who has authorised access to it.

If on the one hand the flexibility of discretionary policies makes them suitable for a variety of systems and applications, on the other hand they do not provide real assurance on the flow of information in a system. For example, a subject who is able to read data can pass it to other subjects not authorised to read it. The reason is that discretionary policies do not impose any restriction on the usage of information, that is dissemination of information is not controlled. By contrast, as we stated above, dissemination of information is controlled in mandatory systems.

Discretionary access control policies can be classified into *closed*, when default decision of the reference monitor is permission, and in *open*, when decision could also be applied by specifying denials instead of permissions. In this last case, each access request by a user is checked against the specified negative authorisations and it is granted only if no authorisation denying the access exists.

The Bell-LaPadula Model

The earliest security policy model was proposed by Bell and LaPadula in the *Bell-LaPadula Model* [28], also known as *Multi-Level Security*. The Bell-LaPadula model focuses on data confidentiality and restricted access to classified information, in contrast to the *Biba Integrity Model* [40], a formal state transition system of computer security policy that describes a set of access control rules designed to ensure data integrity.

The Bell-LaPadula model is a state machine model initially developed to enforce access control in government and military applications. As usual, the entities of an information system are divided into subjects and objects. It describes a set of access control rules which use security levels on objects and subjects, called *security labels* for the former and *clearances* for the latter. Security levels range from the most sensitive (e.g."Top Secret"), down to the least sensitive (e.g., "Unclassified" or "Public"), expressed in terms of a lattice.

In the model it is defined a notion of a secure state and it is proven that each state transition preserves security by moving from secure state to secure state, inductively proving that the system satisfies the security goals of the model. The transition from one state to another state is defined by transition functions. A system state is defined *secure* when it allows access modality of subjects to objects only according to a security policy. To determine whether a specific access modality is allowed, the clearance of a subject is compared to the security labels of the object to determine if the subject is authorised for the specific access modality. The model defines two MAC rules and one DAC rule with three security properties:

1. The *Simple Security Property*: a subject at a given security level may not read an object at a higher security level (also known as *no read-up*);
2. The $\star$*-property*: a subject at a given security level must not write to any object at a lower security level (also known as *no write-down)*;
3. The *Discretionary Security Property*: it uses of an access matrix to specify the discretionary access control.

Among the limitations of this model, we can remember that it is restricted to confidentiality and it does not provide for policies for changing access rights. Indeed

it is oriented to systems with static security levels. Moreover, a low subject can detect the existence of high objects when it is denied access. This is the so called *covert channel*, that is a means by which two components of a system that are not permitted to communicate do so anyway by affecting a shared resource. Finally, an information hiding can happen: two components of the system that are permitted to communicate about one set of things, exchange information about disallowed topics by encoding contraband information in the legitimate traffic.

3.2.3 Administration of Access Control

According to [207], we can have a classification of the policies based on their administration, that is who is authorised to modify allowed access. When we described mandatory and discretionary policies, we stated that the first one is controlled by a security administrator, who is the only one who can change security levels to subjects or objects. Instead, in discretionary policies, the individual user may, at his own discretion, determine who is authorised to access the objects he has created. This allows a wide range of administrative policies:

- *Centralised*: a single entity or a group is allowed to grant and revoke authorisations to the users. This is the case of mandatory policies;
- *Hierarchical*: according to a hierarchy, a central authority is responsible for assigning administrative responsibilities to other administrators. The administrators can then grant and revoke access authorisations to the users of the system;
- *Cooperative*: authorisations on resources need the cooperation of several authorities;
- *Ownership*: the individual user is the owner of the resources he has created. Thus, at his own discretion, he can determine who is authorised to access these objects, as for discretionary policies;
- *Decentralised*: the owner of an object can also grant other users the privilege of administering authorisations on it.

In many distributed systems, and in wireless networks, centralised administration of access rights is infeasible both because the large number of users and because their specific features (i.e. ad hoc networks). All the other requirements are basilar in these systems, as delegation, cooperation, and so on, but very often existing access control do not implement them. Moreover, in the access control mechanisms described in previous sections, the operations considered are elementary such as read, write and execute. Modern applications make demands for more complex actions.

All previous access control mechanisms were originally designed for wired systems. Wireless networks use a shared medium and therefore access control methods used in wired networks are not suitable for them. An additional challenge in wireless networks is that the user can appear anywhere in the network thus policies for controlled networks access becomes essential. Network security that relies on physical constraints is no longer effective.

Access control mechanisms operate at a number of levels in a system, from the physical to the application layer. These systems are vulnerable to environmental changes that make invalid assumptions in their design. The dynamic nature

of ubiquitous networking is the most challenging aspect of access control design. Traditional access control mechanisms are not suitable to the ubiquitous and open environment where the number and identities of users are not known in advance. The number of users is extremely large and their behaviour is difficult to predict, so the risks for service providers change dramatically in such circumstances. It becomes impossible for an administrator to analyse system logs and adapt security policies to the actual situation. Thus a mechanism should be developed to provide access control to resources and to automated management of access policies. Although progress has been made, much remains to be done. The concept of *trust* seems to represent a promising basis for such a direction, as we describe in the next sections.

3.3 Trust in Wireless Systems

As we explained in Section 2.4, security can generally be defined as the quality or state of being secure - to be free from danger. In particular, the term *information security* is referred to the protection of information. It is commonly defined as the preservation of the CIA properties, that is confidentiality, integrity and availability of information. It is often assumed that the owner of information has an interest in keeping them free from danger, and in preserving their CIA properties. However, in many situations it is necessary a protection from harmful information and from those who offer services, so that the problem is reversed. Traditional security solutions are totally inadequate for protecting against for example deceitful service providers that give false or misleading information.

When designing communication systems, it is essentially to include more strict security requirements and not only those related to the CIA properties. In particular, in wireless networks the situation is very complicated.

In [131] Jøsang has distinguished between *hard security* and *soft security*. Hard security is referred to the traditional information security mechanisms, like authentication and access control, whereas soft security indicates the so called social control mechanisms. In Section 3.2 we described some traditional access control security mechanisms. Cryptographic algorithms and firewalls are other examples of *hard security* mechanisms, and they have the general property of allowing complete access or no access at all. Hard security also assumes complete certainty. It is clear how these strong assumptions cannot meet the features of wireless networks. In contrast to traditional information security where security policies are often explicitly defined for a specific security domain by a security manager, soft security is based on an implicit security policy collaboratively emerging from the whole community. In [131] soft security is defined as *the collaborative adherence to common ethical norms by participants in a community and the collaborative enforcement of them.* Among these mechanisms, we can find the ones that take into consideration the concept of *trustworthiness.*

Current security technology offers us with some capability to build in a certain level of security into our communication, i.e. cryptographic algorithms for privacy and digital signatures, authentication protocols for proving authenticity and access control methods for managing authorisation and so on. In Section 3.2 we explained

why security solutions devised for wired networks cannot be used to protect the wireless systems, focusing on access control mechanisms. In particular, these methods cannot manage the more general concept of trustworthiness, that is essential in open and distributed environments as wireless settings. Indeed current security technology is lacking the complementary tool for managing trust effectively; for this reason there is rapidly growing literature on the theory and applications of alternative systems based on the concept of trustworthiness. While the goal of traditional information security is to preserve the CIA properties, the goal of soft security mechanisms is to stimulate the quality of a specific community in terms of the ethical behaviour and the integrity of its members. What constitutes ethical norms within a community will be dynamically defined by certain key entities in conjunction with the user. Soft security mechanisms make it possible to identify and punish *bad members* of a community, that is those participants who do not respect the norms, and to recognise and reward *good members*, that is who adhere to the norms.

3.3.1 Trust vs Security

According to Buttyán and Hubaux [57], trust and security are two tightly coupled concepts that cannot be desegregated. For example, cryptography is a means to implement security but it is highly dependent on trusted key exchange. Similarly, trusted key exchange cannot take place without requisite security services in place. It is because of this inter-reliance that often both of these terms are used interchangeably when defining a secure system.

In a first approximation, trust can be defined as the belief that another party (a person, an organisation, but also a device) will behave according to a set of well established rules and will thus meet one's expectations. This notion is fundamental in all human societies (and also in many animal groups); generally, a breach of trust is considered to be a major offence. But trust is a fuzzy and abstract notion, because it considered across persons or across areas of competence: no matter how close they are to each other, different people may trust very different things, even in front of the same evidence. Likewise, a person A may trust a person B for the accomplishment of a certain task, but not another.

Buttyán and Hubaux in [57] have collocated trust with respect to security and cooperation:

- *Trust preexists security*: trust is a natural phenomenon, and it has existed for millennia, before any concept of security was invented. Security is simply a technique to infer trust: if A trusts something, security can help A trusting something else. For example, if A trusts that a personal computer is not compromised and that the security protocols and the cryptographic algorithms it uses are not flawed, then A will carry out e-banking transactions with the legitimate belief that they are safe. It should be clear from this simple example that any security mechanism requires some level of trust in its underlying components;
- *Cooperation reinforces trust*: in the informal definition that we have provided, trust is about the ability to predict the behaviour of another party. A reasonable assumption is the selfishness of the parties in a system. Therefore, if a system

is designed in such a way that the socially desirable behaviour coincides with a party's interest, then it is likely that the party will indeed behave as desired. Hence there is the possibility of a virtuous cycle, because if an observer B notices a cooperative behaviour of A, B could believe that A will continue to be cooperative in the future. Thus the trust of B in A will increase. This also encourages B to be cooperative, which will reinforce the trust that A has in B, and so on.

Because of the complex features of trust, and as it is very deeply rooted in the human nature, trust is difficult to quantify and to model. It is in fact easier to describe the reasons to trust someone or something, which are the following, according to [57]:

- *Moral values*: any society has its rules, and in many cases it is assumed that other parties obey these rules, typically because of their education or because they fear bad publicity. In any case their misbehaviour should be disclosed. So for example, we trust a large cellular operator to protect our privacy as long as there is no strong reason (e.g., a legal enquiry) to depart from that attitude;
- *Experience about a given party*: previous interactions are of course revealing about the trustworthiness of a given party. These interactions can be either first hand (direct) or be reported by other parties (indirect), meaning that reputation is a fundamental component of trust;
- *Rule enforcement organisation* and *mechanism*: if the risks are high, the obedience to the rules is further encouraged by a specialised agency. For example, the way cellular operators use the radio spectrum is usually regulated by a governmental agency or the way mobile users make use of the radio spectrum is usually controlled by the operator. Moreover, technical mechanisms must be deployed to either make attacks more difficult or to encourage the desired behaviour;
- *Usual behaviour*: although malicious behaviour refers to poorly understood psycho-logical mechanisms, it is possible to consider that one behaviour is much more frequent than another. For example, network users will often keep trying to set up a communication in spite of the fact that the network is congested, but very few will make the effort to jam a given area simply to enjoy complicating other peoples life.

3.3.2 Trust and Reputation

The concepts of trust and reputation are closely related and they are firmly routed in sociology and psychology. Although there is no universal definition for these concepts, due their rich connection with different disciplines, we focus on computing-oriented definitions.

Trust is a binary directional relationship between two parties, called *trustor* and *trustee*. In general, the trustor is the subject that trusts an entity or a service, and the trustee is the entity that is trusted. Trust enables a trustor to reduce uncertainty in its future interactions with a trustee, whose actions may affect the state of the trustor. Trust relations may be *revoked* on the basis of newly obtained evidence.

The term trust has been used with a variety of meanings [161]. The definition of trust adopted by Gambetta in [88] is often referred as *reliability trust* and it is defined as the *belief or subjective possibility by which an individual A expects that another individual B performs a given action on which the welfare of A depends.* The author introduces the dependency of trust from the context, meaning that a trust relationship has a *scope*, as it applies to a specific purpose or domain of actions. In these terms, trust may be viewed as a quantitative value, that is a quantifiable relation between two entities. According to Grandison [102], trust is *the quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee, within a specified context.* In [155], trust implies a risk of some sort and it is strongly linked to confidence and context. In [37] it is introduced a formal representation of trust relationships and a way to *dynamic* trust evaluation. In [132] the authors distinguish between functional and referral trust, and between direct and indirect trust. *Functional trust* is the belief in an entity's ability and willingness to carry out or support a specific function on which the relying party depends. *Referall trust* is the belief in an entity's ability to recommend another entity with respect to functional trust. A *direct trust* relation occurs when the trustor trusts the trustee directly, without relying on intermediates. An *indirect trust* relation occurs when the trustor trusts a trustee based on one or more opinions from third parties.

Reputation is defined as the opinion held by the trustor towards the trustee, based both on its past experience and *recommendations* of other trustees. A recommendation is simply an attempt at communicating a party's reputation from one community context to another. Reputation is an important concept for the trust evaluation, but it is often confused with trust. Social network researchers deal with reputation as a quantity derived from the implicit social network which is globally visible to all members of the network. The difference between trust and reputation can be illustrated by the following statements:

a) I trust you because of your good reputation;
b) I trust you despite your bad reputation.

Assuming that the two sentences relate to the same trust scope, statement a) reflects that the relying party is informed of the trustee's reputation and bases his trust on that. Statement b) reflects that the relying party has some private knowledge about the trustee, e.g. through direct experience or intimate relation, and that these factors replace any (negative) reputation that a person might have. This observation reflects that trust is a personal and subjective phenomenon that is based on various factors or evidence, and that some of those have more weight than others. Personal experience typically carries more weight than second hand trust referrals or reputation, but in the absence of personal experience, trust often has to be based on recommendation from others. An individual's subjective trust can be derived from a combination of received recommendations and personal experience.

Trust represents an active and decisive concept: if one entity trusts another entity then, the latter is allowed to perform certain actions. Reputation may serve as a source of trust; however, it does not directly define allowed actions. Trust is subjective, while reputation is also subjective but is not based on personal observa-

tions. Reputation usually comes from the context and it does not reflect personal experience of the interested party. As for trust, there are several possible definitions of reputation. For instance, in [7] reputation is defined as an expectation about an individual's behaviour based on observations of its past behaviour. In [134] reputation is proposed as a meaning of building trust; one entity can trust another entity based on its good reputation.

We can summarise the features of trust as follows:

- trust is *not symmetric*: if a user A trusts another user B, it does not mean that B trusts user A;
- trust is *not distributive*: if a user A trusts users B and C in pair, the statement "A trusts B and A trusts C" separately is not true;
- trust is *not transitive*: if a user A trusts a user B and B trusts a user C, it does not follow that A trusts user C.

3.4 Trust Management Systems

Trust management and trustworthy computing are becoming increasingly significant in a distributed environment, since they assist the systems in making sensible interactions with unknown parties by providing a basis for more detailed and automated decisions [203].

Multi-agent systems often use access control mechanisms to manage shared resources. The problem of access control can be splitted in two subproblems: (i) determining whether or not a request should be allowed, and (ii) enforcing the decision. Trust management systems solve the first subproblem by defining languages for expressing authorisations and access control policies, and by providing a trust management engine for determining when a particular request is authorised. Traditional access control mechanisms are centralised and operate under a closed world assumption in which all of the parties are known. Trust management systems generalise access control mechanisms by operating in distributed systems and eliminating the closed world assumption. Over the last years, a number of trust management systems have been developed, some focusing on authentication [225, 231, 239], others for specialised purposes [25, 63, 113], others for general purpose authorisation [43, 46, 74], and others based on logics [5, 15, 145].

An enlightening formalisation of trust management systems has been proposed by Weeks in [230]. In this paper the author has introduced a mathematical framework for expressing trust management systems. The framework helps in understanding a number of existing systems, making a comparison among them. According to the framework in [230], trust management systems are usually composed by the following components: (i) *evidence manager*, to collect and classify evidence, (ii) *mathematical model*, to formulate evidence into opinion, and then to use that opinion to predict the result of future interactions, and (iii) *policy manager*, that collaborates with the evidence manager and the mathematical model in defining policies for making decisions and granting authorisations to users. The semantics of the policy manager is then defined via a least fixpoint in a lattice. The trust management framework proposed in [230] is monotonic, in the sense that authorisations granted to a user can only increase. To demonstrate the flexibility of

this framework, the author reformulated the systems KeyNote [43] and SPKI [74] using his framework. The definition of the semantics via the least fixpoint also makes it clear that the framework cannot handle revocation. There are some other aspects of trust management systems that do not fit well within this framework. One example is REFEREE [63], in which the trust management engine directly interprets policies and credentials, without finding a fixpoint meaning.

In traditional trust management systems, as those mentioned above, decisions are based on the credentials presented by user. However, more generally, trust management systems can be classified into *credential-based* and *behaviour-based* systems. In *credential-based systems* peers use certificates in order to establish trust with other peers. Their trust management is limited to verifying credentials and restricting access to resources according to the application of defined policies. When a peer requests an access to a resource, the resource owner provides access only if it can verify the credentials of the requesting peer. These systems are used when there is an implicit trust in the resource owner by the requesting peers. They do not incorporate the need of the requesting peer to establish trust on the resource owner. For these reasons they are a good solution for decentralised systems. One of the first implementations of a credential-based management system was PolicyMaker [45, 47, 48]. KeyNote [44] appeared as an improvement of PolicyMaker and REFEREE [63] as a trust management system for web services.

Behaviour-based systems are often called experience-based as in these models an entity A trusts another entity B based on its experience on B's past behaviour. These systems heavily relies on the concept of *reputation*. One of the first attempts to build a reputation-based trust management systems for e-commerce was SPORAS [237]. REGRET [205] is a reputation-based model developed in the context of multi-agents. Based on beliefs, Jøsang proposed a subjective logic [135] in order to derive reputation values. Other reputation systems use probabilistic methods such as the Beta function [133].

The main differences between credential and reputation systems can be summarises as follows: the former ones produce a score that reflects the relying party's subjective view of an entity's trustworthiness, whereas reputation systems produce an entity's (public) reputation score as seen by the whole community. Secondly, transitivity of trust paths and networks is an explicit component in credential systems, whereas reputation systems usually do not take transitivity into account, or only in an implicit way. Finally, trust systems take subjective expressions of (reliability) trust about other entities as input, whereas reputation systems take ratings about specific (and objective) events as input.

In the following we give brief description of some of the credential-based and behaviour-based trust systems cited above.

Credential-based TMSs

PolicyMaker and KeyNote

Blaze et al. [45, 46] have developed the traditional notion of trust management based on the *compliance checking problem*: "Does a set C of credentials prove that a request r complies with a local security policy σ?". The same authors in

the same papers have also developed a first prototype trust management system called *PolicyMaker*.

The most general form of proof of compliance in PolicyMaker is undecidable and several natural restrictions are NP hard. PolicyMaker considers a version of the proof of compliance problem which requires that all assertions are monotonic (assertions are fully programmable functions that are part of credentials). This leads to a restricted notion of proof of compliance which is decidable in polynomial time. *KeyNote* [44], the successor of PolicyMaker, restricts the language of assertions to a simple domain specific language so that resource usage is proportional to program size. KeyNote is less general than PolicyMaker but has simpler syntax and semantics, and requires less computational power. Architectural tradeoffs between Policy Maker and KeyNote is considered in [45].

The RT Family

The *Rôle-based Trust management* (RT) framework is a family of languages for policies and credentials which combines the strengths of rôle-based access control [79, 206] and trust management. It was developed by Li, Mitchell and Winsborough in [146, 147]. The RT framework consists of languages together with an engine which works by translating credentials into Datalog rules. This enables proof of compliance checking in polynomial time. RT supports concepts of intersection rôles, manifold rôles and delegation of rôle activation; these enhance the expressive power, compared to other frameworks. Furthermore, RT supports distributed credentials and distributed credential discovery and it is considered to be the state of the art for credential based trust management systems.

Behaviour-based TMSs

Bayesian Systems

Jøsang and Ismail [133] and Mui, Mohtashemi and Halberstadt [176] were among the first authors in developing reputation systems based on a Bayesian probabilistic approach with beta priors. *Bayesian systems* take binary ratings as input, i.e. positive or negative, and are based on computing reputation scores by statistical updating of beta probability density functions (PDF). The *a posteriori*, i.e. the updated, reputation score is computed by combining the *a priori*, i.e. previous, reputation score with the new rating. The reputation score can be represented in the form of the beta PDF parameter tuple (α, β), where α and β represent the amount of positive and negative ratings respectively, or in the form of the probability expectation value of the beta PDF, and optionally accompanied with the variance or a confidence parameter. The advantage of Bayesian systems is that they provide a theoretically sound basis for computing reputation scores, and the only disadvantage that it might be too complex to understand.

Belief Models

Belief theory is a framework related to probability theory, but where the sum of probabilities over all possible outcomes not necessarily add up to 1, and the

remaining probability is interpreted as uncertainty. Jøsang [130] has proposed a belief/trust metric called *opinion* denoted by $\omega_x^A = (b, d, u, a)$, which expresses the relying party A's belief in the truth of statement x. Here b, d, and u represent belief, disbelief and uncertainty respectively where $b, d, u \in [0, 1]$ and $b+d+u = 1$. The parameter $a \in [0, 1]$, which is called the *relative atomicity*, represents the base rate probability in the absence of evidence, and is used for computing an opinion's probability expectation value $E(\omega_x^A) = b + au$, meaning that a determines how uncertainty shall contribute to $E(\omega_x^A)$. When the statement x for example says "David is honest and reliable", then the opinion can be interpreted as reliability trust in David. As an example, let us assume that Alice needs to get her car serviced, and that she asks Bob to recommend a good car mechanic. When Bob recommends David, Alice would like to get a second opinion, so she asks Claire for her opinion about David. When trust and trust referrals are expressed as opinions, each transitive trust path Alice $\longrightarrow$ Bob $\longrightarrow$ David, and Alice $\longrightarrow$ Claire $\longrightarrow$ David can be computed with the *discounting operator*, where the idea is that the referrals from Bob and Claire are discounted as a function Alice's trust in Bob and Claire respectively. Finally the two paths can be combined using the *consensus operator*.

The Trust Structure Frameworks

Carbone, Nielsen and Sassone [58] have defined the *trust structure framework*, a formal model for trust based on the work of Weeks [230], but focusing on information rather than authorisation. They have a set $\mathcal{D}$ of trust values, built through a particular operator on a complete lattice. A *trust structure* is a triple $T = (\mathcal{D}, \preceq, \sqsubseteq)$, where $\preceq$ is the trust ordering and $\sqsubseteq$ is the information ordering. These partial orderings are used to built complete lattices on $\mathcal{D}$.

The goal of the framework is to define, given a set of principals P and a trust structure T, a unique global trust state, `gts`, to represent every principal's trust in every other principal. Mathematically, the computation is very similar to that proposed by Weeks. Principals define their trust policies, a sort of "web of trust", and by means of them each local policy makes reference to other principals' local policies using mutual recursion. Global trust is the function determined collectively by the web of policies. This amounts to say that `gts` is the least fixpoint of the universal set of local policies.

The framework presented in [58] described above could be considered as an "hybrid" model, not really a credential ones, although it may be instantiated to obtain certain credential-based systems, and on the other hand, not really an experience-based model, although it may be instantiated to certain simple experience-based system. Nielsen and Krukow in [182] have started with the trust model in [58] and have added probabilities. They have explicitly model *risk*, e.g. the expected cost of an interaction, as well as trust. Each interaction is modelled as having a set of possible *outcomes*. Thus they have explicitly consider a structure, *Out*, modelling the possible outcomes. The risk of each outcome depends on the probability of the outcome when interacting with a principal, and on the intrinsic cost or benefit of the outcome. Costs are represented by probability density functions on some range of cost values. The main contribution is the discovery that the notion of *event structures*, well studied in the theory of concurrency, can faithfully model the important concepts of observation and outcomes of interactions. In this setting, observations

are events and an outcome of an interaction is a maximal set of consistent events, representing the past evidence for that particular outcome. They consider a trust structure $T = (\mathcal{D}, \preceq, \sqsubseteq)$ where $\mathcal{D}$ has the general form $\mathcal{D} = Out \longrightarrow Ev$ where Ev represents evidence. Further, they have presented a general procedure to convert trust values into probabilities of outcomes. Krukow, Nielsen and Sassone [140] have proposed an alternative way of recording behavioural information. They have presented a logical policy-based framework for reputation systems in which principals specify policies which state precise requirements on the past behaviour of other principals that must be fulfilled in order for interaction to take place. The framework consists of a formal model of behaviour, based on event structures, a declarative logical language for specifying properties of past behaviour and efficient dynamic algorithms for checking whether a particular behaviour satisfies a property from the language.

3.4.1 Trust Management Systems for MANETs

We explained that the activities of MANETs highly depend on the distributed cooperation among nodes and, at the same time, they are susceptible to node misbehaviour. Thus the establishment of trust relationships within the network could serve as the basis for higher level security solutions. However, the specific features of MANETs pose challenges for the trust management area. We can think of constrained energy, memory, computation, communication capabilities, the wireless nature of communications, the dynamically changing topology and the lack of fixed infrastructure. There are a number of consequences: each node needs to manage trust relationships with other nodes individually, connectivity cannot be assured, and thus stable hierarchies of trust relations cannot be supported, trust establishment needs to support evidence that may be uncertain and incomplete. Moreover, node misbehaviour can affect not only network operations but also the trust evaluation framework itself.

For all these considerations, trust establishment protocols for MANETs should:

- be decentralised and not based on online trusted parties. Instead, they should support distributed, cooperative evaluation, based on uncertain evidence;
- support and exploit the diversity in the roles and the capabilities of the nodes in the deployments by allowing for flexibility in the trust establishment process;
- support trust revocation in a controlled manner;
- scale to large deployments, be flexible to membership changes and entail acceptable resource consumption.

In Section 2.5 we reasoned how mobility in MANETs has impact on security. It has also impact on the trust establishment process. It introduces issues related to user credentials management, indirect trust establishment and mutual authentication between previously unknown and untrusted entities. For example, in cellular network access providers and service providers may trust each other on contractual basis. In such a way a user may connect to a network managed by an authority with which he has not established direct trust. Since trust relations are not transitive, special trust infrastructures and mechanisms are required to establish indirect trust between a roaming user and a visited network. In a mobile network,

it is critical to know what entities can access what resources, from where and when. A user is often served by an entity that is not his identity provider and trust establishment between a user and a service provider becomes a necessity. Trust management represents a complex process of entities recognising, definition of privileges and restriction for them and access granting according to these privileges. In daily communications, interaction with multiple identity and service providers becomes an important part of trust management. All parties involved must have specific trust relationships. Entities verifying trust information often do not have direct means to estimate trustworthiness of the corresponding entity. Thus trust models must take into consideration also the conditions in which a party can trust other parties.

Credential-based trust management systems for MANETs aim at defining mechanisms for pre-deployment knowledge on the trust relationships within the network, usually represented by certificates, to be spread, maintained and managed either independently or cooperatively by the nodes. Trust decisions are mainly based on the provision of a valid certificate which proves that the target node is considered trusted either by a certification authority or by other nodes that the issuer trusts. It is generally outside the scope of certificate-based frameworks to evaluate the behaviour of nodes and base trust decisions on that evaluation.

In behaviour-based trust management systems for MANETs, each node performs trust evaluation based on continuous monitoring of the behaviour of its neighbours, in order to evaluate how cooperative they are. Although a mechanism that determines the identities of the other nodes is usually assumed to exist, it is generally outside the scope of behaviour-based trust establishment models to securely authenticate other nodes and to determine whether they are legitimate members of the network. In that sense, behaviour-based models are more reactive than certificate-based models. As an example, if a node makes unauthorised use of the network and behaves selfishly or maliciously, it will not manage to gain or retain a trust level that will allow it to cooperate with other nodes, and it will be thus isolated. Trust is evaluated both *independently*, by each node, based on observations and statistical data that is continuously accumulated by monitoring the network traffic, and *cooperatively* through sharing recommendations and spreading reputation. The main objective of these behaviour-based models is to isolate the nodes that either act maliciously because they have been compromised, or selfishly to preserve resources, by assigning and recommending low levels of trust.

Comprehensive surveys of trust management systems for ad hoc networks can be found in [10, 24, 201]. In the following we give a brief description of two of these frameworks.

Trust Establishment in Pure Ad-hoc Networks

Pirzada and McDonald [190] have proposed a behavioural model for ad hoc networks. They have called *pure* ad hoc networks those networks that which have no required centralised infrastructure. In Section 2.2.1 we have called them self-organised mobile ad hoc networks. Pirzada and McDonald have described a trust model for finding trustworthy routes in ad hoc networks that is entirely based on direct trust evaluation. In their model, they have made use of independent trust

agents that reside on network nodes, each one gathering network traffic information in passive mode by applying appropriate taps at different protocol layers. The information gathered from these events is classified into trust categories, so that the *situational trust* $TS(i, j, x)$ for node j according to node i can be computed using the information of trust category x. Moreover, weights $W_i(x)$ are assigned according to the utility and importance of each trust category for i. The general trust is thus computed as the trust that the trustor node i assigns to the trustee node j based upon all previous transactions in all situations, according to their significance.

The Directed Graph Approach

A different view on trust evaluation with respect to [190] has been proposed by Theodorakopoulos and Baras in [223]. They have presented a behavioural trust model for ad hoc networks which mainly focus on the evaluation of indirect trust as the combination of opinions from neighbouring nodes, assuming that some mechanism exists for these nodes to assign their opinions based on local observations. The process of indirect trust evaluation is formulated as a shortest path problem on a weighted directed graph, where graph nodes represent network nodes and edges represent trust relations. The edges are weighted with the trust value that the issuer node has on the target node and the confidence value it assigns on its opinion, depending on the number of the previous interactions and positive direct evaluations. The theory of semirings is being used for formalising two versions of the trust inference problem: finding the trust-confidence value that node i should assign to node j, based on the trust-confidence values of the intermediate nodes, and finding a sequence of nodes that has the highest aggregate trust value among all trust paths from i to j.

3.5 Chapter Summary

The purpose of access control is to limit the actions or operations that a legitimate user of a computer system can perform. Access control constrains what a user can do directly, as well what programs executing on behalf of the users are allowed to do. In this way access control seeks to prevent activity which could lead to breach of security. Traditional access control mechanisms do not work well with wireless systems. A more suitable approach is represented by trust management systems.

In this chapter we described the main features of access control and we analysed some traditional access control mechanisms and policies. Traditional access control mechanisms are classified as hard security, as well as cryptographic algorithms and firewalls. They have the general property of allowing complete access or no access at all. Hard security also assumes complete certainty. It is clear how these strong assumptions cannot meet the features of wireless networks. More *soft* security mechanisms are necessary. While the goal of traditional information security is to preserve the CIA properties, the goal of soft security mechanisms is to stimulate the quality of a specific community in terms of the ethical behaviour and the integrity of its members. Trust management approaches are mechanisms of this kind. Then

we introduce the concept of trust and we describe some trust management systems. In the next chapter we deal with process calculi.

4

Process Calculi

4.1 Introduction

Process algebra is an active area of research in concurrency theory, the theory of parallel and distributed systems in computer science. It has been developed during the early seventies of the twentieth century. Baeten in [17] have addresses the history of this research area. Until that moment, the only part of concurrency theory that existed was the theory of Petri nets, proposed by Petri in his Ph.D. thesis in 1962 [189].

The term process algebra is used in different meanings. The word "process" refers to the *behaviour* of a *system*. A system is anything showing behaviour, in particular the execution of a software system, the actions of a machine or even the actions of a human being. Behaviour is the total of events or actions that a system can perform, the order in which they can be executed and maybe other aspects of this execution, such as timing or probabilities. We always describe only some aspects of behaviour; we can say that we have only an observation, and an action is the chosen unit of observation.

The word "algebra" denotes that we take an algebraic/axiomatic approach in talking about behaviour. If we consider the definition of a *group* in mathematical algebra, we can say that a group is any mathematical structure with operators satisfying the group axioms. In other words, a group is any model of the equational theory of groups. Likewise, we can say that a process algebra is any mathematical structure satisfying the axioms given for the basic operators. An element of a process algebra is a *process*. By using axioms, it is possible to perform *calculations* with processes.

The term *process calculus*, used for the first time by Milner in [169], denotes an approach largely algebraic, but which may also include the use of logic or other mathematical disciplines.

The simplest model of behaviour is to consider it as an input/output function. We can observe a value, the input, given at the beginning of the process, and then at some moment we can observe another value (or the same) as outcome or output. This model has been the basis in the development of finite state *automata theory*. In automata theory, a process is modeled as an automaton. An automaton has a number of states, some are initial states, other ones final states, and a number

of transitions from one state to another one. A transition denotes the execution of a basic action, that constitutes the basic unit of behaviour. A run is a path from an initial state to a final state, a behaviour is the set of executions from the initial state to a final state. Two automata can be compared, that is it is possible to establish if they are equal. This is expressed by a notion of *equivalence*. On automata, the basic notion of equivalence is language equivalence, based on the algebra of regular expressions.

During the years, studying more complex and dynamic systems, the automaton model was found to be incomplete. Basically, the notion of interaction between systems is not considered. The description of parallel or distributed systems, or so-called *reactive systems*, is based on this notion of interaction: during the execution from initial state to final state, a system may interact with another system. The theory behind reactive systems is the concurrency theory and process calculus was developed as as an approach to concurrency theory.

A process calculus is characterised by a *syntax*, a *formal semantics*, and *behavioural relations*. The syntax describes the structure of the terms (processes) of the calculus and it is is defined inductively. The semantics gives reasoning about the system dynamics. We can distinguish between *operational*, *denotational* and *axiomatic semantics*. Finally, behavioural equivalences allow to establish when two processes have the same observable behaviour, that is they are indistinguishable for an observer. The foundational process calculi, as CCS [168,169], CSP [56,117], ACP [30,31], π-calculus [171], take point-to-point communication as primitive.

Important developments have occurred since the formulation of these basic process calculi. One important aspect to be taken into consideration is *time*. There may be good reasons to introduce time in such a way that suitable timing becomes relevant for the correct behaviour of a complex system, or because for certain protocols depends on the timing of certain actions or simply to pay attention to performance aspects of a system.

Process calculi that incorporate some form of timing have been studied extensively by now. According to purposes and applications, different versions of describing timing have been presented in the literature [18–20,23,111,175,195,199].

Developed with the goal of providing very general process models, most specialised calculi are derived from their seminal ideas. It seemed that point-to-point fashion led to complicated and sometimes little intuitive encodings when dealing with more modern systems. In the recent years, a particular attention has been addressed to the development of process calculi and observational theories to obtain a formal foundation of the modelling and analysis of wireless communications. Specific aspects of wireless communications must be taken into account in these new process calculi: local broadcasting, the presence of obstacles that disables the reception, the possibility of interferences and collisions, and node mobility. These aspects have been taken into account in calculi as [77,92,94,95,163,167,178,214].

The present chapter is organised as follows. In Section 4.2 we present an overview of foundational process calculi, in Section 4.3 we deal with timed process calculi, whereas in Section 4.4 we describe some recent calculi for wireless networks. Finally, in Section 4.5 we give a summary of the chapter.

4.2 Foundational Process Calculi

Process algebra can be defined as *the study of the behaviour of parallel or distributed systems by algebraic means.* Besides parallel composition to describe a system, process algebra can usually also provide alternative composition (choice) and sequential composition (sequencing). Moreover, we can reason about such systems using algebra, i.e. equational reasoning. By means of this equational reasoning, we can do *verification*, i.e. establishing whether a system satisfies certain properties.

Among the basic laws on operators of process algebra, we can distinguish between the following:

- *static laws*, also called *structural laws*, that do not involve actions but list some general properties of the operators involved, such as commutativity, associativity, distributivity, and so on;
- *dynamic laws*, that involve action executions directly. For instance, there is no static law connecting parallel composition to the other operators. Such a connection is at the heart of process algebra, and it makes calculation possible. In most process algebras, this law allows to express parallel composition in terms of the other operators, and is called the *expansion theorem.* This theorem is an example of dynamic law. Process algebras with an expansion theorem are called *interleaving* process algebras, those without are called *partial order* or *true concurrency.*

Milner in [169] has made a distinction between process calculus and process algebra, considering the first one largely algebraic but where also other mathematical disciplines can be used.

Typically, there are three elements that constitute a process calculus:

1. **Syntax**: the structure of the terms of a process calculus is defined inductively. Processes are basic terms of process calculi. The simplest entities are *channel names* whose purpose is to provide means of communication. Processes use them to interact and pass values to one other by referring to them in interactions; moreover values received can be used again in further interactions. When among values also channel names are passed, we say that it is a *name-passing* process calculus, otherwise it is called *value-passing* process calculus. Processes evolve performing *actions*; thus in the syntax the capabilities for action (i.e. input/output ...) are also described. In addition to names, one needs a means to form new processes: the crucial operators, always present in some form or other, allow parallel composition of processes, specification of which channels to use for sending and receiving data, sequentialisation of interactions, hiding of interaction points, recursion or process replication;
2. **Semantics**: a formal semantics gives reasoning about the system dynamics. At the beginning of the studies in concurrency area, we could distinguish three main styles of formal reasoning about computer programs, focusing on giving semantics to programming languages:
   - *Operational semantics*: a computer program is modeled as an execution of an abstract machine. A state of such a machine is a valuation of variables and a transition between states is an elementary program instruc-

tion. Pioneer of this field is McCarthy [160]. Operational semantics is usually expressed in two style: *Structural Operational Semantics* (SOS) by Plotkin [191], generating Labelled Transition Systems (LTS), so that states are closed terms and state transitions are obtained from a collection of inference rules, and the *Chemical Abstract Machine* (CHAM) [34], giving the semantics in terms of a set of reduction rules for term rewriting and a structural congruence to rearrange terms. It can also be taken into account a binary relation, called *reduction relation*, explaining how a system can evolve independently of its environment, whereas the SOS explains not only activity within the system but also how processes can interact with their external environment. It is possible to choose only the former or the latter or both to provide an operational semantics; when both of them are used, it is necessary to formally prove the correspondence;

- *Denotational semantics*: it is more abstract than operational semantics, as computer programs are usually modeled by a function transforming input into output [210];
- *Axiomatic semantics*: emphasis is put on proof methods proving whether programs are correct. Central notions are program assertions, program statement and postcondition, and invariants. Preliminary works can be found in [80, 115];

3. **Behaviour equivalences**: a central topic for process calculi is to establish when two processes have the same *observable behaviour*, that is when they are indistinguishable for an external observer. The notion of equivalence studied for process calculus is usually not language equivalence, as for the automata. Particular notions of *behavioural equivalences* are involved. Several notions of them can be found in literature, sharing some properties:
   - two processes are equivalent only if identical interactions are observed to any environment;
   - the equivalence is preserved by some key construct of the calculus.

   Bisimulation is an example of behavioural equivalence; intuitively two processes P and Q are *bisimilar* if they match each other's actions.

The problem to be faced at the beginning of the study in concurrency was how to give semantics to programs containing parallel operator. It was immediately clear that the idea of a behaviour as an input/output function needed to be abandoned. A program could still be modeled as an automaton, but the notion of language equivalence is no longer appropriate. Moreover, in modelling automata global variables were used: a state of a system is given as a valuation of the program variables, that is, a state is determined by the values of the variables. In concurrent systems, the independent execution of parallel processes makes difficult and sometimes impossible to determine the values of global variables at a given moment. It turns out to be simpler to let each process have its own local variables, and to denote exchange of information explicitly.

One of the people studying the semantics of parallel programs in the early seventies was Hans Bekič. Working on the denotational semantics of ALGOL and PL/I, it arose the problem of how to give a denotational semantics for parallel composition. In [27] Bekič have proposed a preliminary solution, that is definitely considered a precursor of the later expansion law of process algebra. In general,

Bekič has contributed a number of basic ingredients to the emergence of process algebra.

Before describing the process calculi closely related to wireless networks and then to process calculi presented in the present thesis, we first give a background on what may be called "pure" or "foundational" calculi.

CCS

The work of Milner about his process theory CCS has culminated in the publication of the book [168], that was later updated in [169] with the notion of bisimulation. In these books we have for the first time in history a complete process algebra, with a set of equations and a semantical model. Moreover in [169] the author have used for the first time the expression "process calculus" with the meaning previously explained. They are presented the basic language and a value-passing variant of CCS. The CCS provides a static and non-distributed model of processes with synchronous point-to-point communication. We now give a brief description of the value-passing CCS, where, with respect to the basic language, the variables and values to be passed through channels are introduced.

Actions are of the form

$$\alpha ::= \bar{a}(v) \quad | \quad a(v) \quad | \quad \tau$$

to mean the output of a value v on channel a, the input of a value v on channel a and the silent action, respectively. Among the processes, we report

$$P ::= \mathbf{0} \quad | \quad \alpha'.P \quad | \quad P + Q \quad | \quad A \stackrel{\text{def}}{=} P \quad | \quad P \mid Q \quad | \quad P\backslash K$$

where

$$\alpha' ::= \bar{a}(v) \quad | \quad a(x) \quad | \quad \tau$$

is the output, input and silent prefix, respectively. Process $\mathbf{0}$ denotes the terminal process, $\alpha'.P$ is the prefix process, $P+Q$ is the non-deterministic choice, $A \stackrel{\text{def}}{=} P$ indicates a recursive definition of P, $P \mid Q$ is the parallel composition of P and Q, $P\backslash K$ denotes the restriction process, to mean that actions contained in K cannot be used for interactions with the environment but only for internal communications in P.

A labelled transition relation over processes is provided as operational semantics of the calculus. Below, we report the rule for the parallel composition, when both branches interact internally via handshake. The other rules are quite straightforward:

$$\frac{\bar{a}(v).P_1 \xrightarrow{\bar{a}(v)} P_1 \quad a(x).P_2 \xrightarrow{a(v)} \{^{v}/_{x}\}P_2}{\bar{a}(v).P_1 \mid a(x).P_2 \xrightarrow{\tau} P_1 \mid \{^{v}/_{x}\}P_2}$$

where $\{^{v}/_{x}\}P_2$ means the substitution of variable x with value v in P_2.

Among the different behavioural relations used for the CCS, we cite the *strong bisimulation*, in which each action α of one process must be matched by an action α of the other, and the *weak bisimulation*, where the requirement is relaxed and it is required that each τ action is matched by zero or more τ actions. It is proved that they are congruence, that is they are preserved under all operators of the calculus.

CSP

The works of Hoare et al. [56,117] give a good overview of CSP, that was introduced some years before in [116] as a value-passing calculus. In processes there is a distinction between two choice operators:

- *Deterministic* (or external) choice: it allows the future evolution of a process to be defined as a choice between two component processes, and allows the environment to resolve the choice by communicating an initial event for one of the processes;
- *Non-deterministic* (or internal) choice: it allows the future evolution of a process to be defined as a choice between two component processes, but does not allow the environment any control over which of the component processes will be selected.

Moreover, the parallel operator is represented as an interleaving operator to model completely independent concurrent activity. It does not hide communications and thus allows any number of processes to participate in the same event.

The operational semantics of CSP is usually expressed in denotational style. The three major denotational models of CSP are the *traces model*, the *stable failures model*, and the *failures/divergences model*. Semantic mappings from process expressions to each of these three models provide the denotational semantics for CSP. Behavioural reasoning are not based on equivalences but on *refinements*. The notion of refinement consists in establishing a relation between components of a system which captures the fact that the system satisfies at least the same conditions as another systems. This leads to a more abstract notion, where it is not required that the implementation and its specification must be indistinguishable, as in bisimulation, but it is required only that the observations of a correct implementation are contained in those of its specification.

ACP

In [29] Bergstra and Klop started the work that led to ACP [30, 31]. In [29] they have used the phrase "process algebra" for the first time. In the following, we quote the exact description:

A *process algebra* over a set of atomic actions A is a structure

$$\mathcal{A} = \langle \mathtt{A}, +, \cdot, \between, a_i(i \in I) \rangle$$

where $\mathtt{A}$ is a set containing A, the a_i are constant symbols corresponding to the $a_i \in A$, and $+$ (union), $\cdot$ (concatenation or composition, left out in the axioms), $\between$ (left merge) satisfy for all $x, y, z \in \mathtt{A}$ and $a \in A$ the following axioms:

$A1$	$x + y = y + x$
$A2$	$x + (y + z) = (x + y) + z$
$A3$	$x + x = x$
$A4$	$(xy)z = x(yz)$
$A5$	$(x + y)z = xz + yz$
$A6$	$(x + y) = x\lfloor\!\lfloor z + y\lfloor\!\lfloor z$
$A7$	$ax\lfloor\!\lfloor y = a(x\lfloor\!\lfloor y + y\lfloor\!\lfloor x)$
$A8$	$a\lfloor\!\lfloor y = ay.$

Bergstra and Klop have defined a process algebra with alternative, sequential and parallel composition, but without communication. A model was established based on projective sequences (a process is given by a sequence of approximations by finite terms), and in this model, it is established that all recursive equations have a solution.

π-calculus

Research on networks of processes where processes are mobile and configuration of communication links is dynamic has been dominated by the π-calculus by Milner, Parrow, and Walker [171]. Good textbooks can be considered [170, 208].

This calculus can be considered a development of previous process calculi where processes are considered static. Central to the π-calculus is the notion of *name*. The simplicity of the calculus lies in the dual rôle that names play as *communication channels* and *variables*. This implies, most importantly, that communicated names can be used as new communication links after reception. This is achieved syntactically by action prefixes

$$\pi ::= \bar{x}y \quad | \quad x(z) \quad | \quad \tau \quad | \quad [x = y]\pi.$$

The first three prefixes are similar in the value-passing version of the CCS, and represent the sending of name y via channel x, the receiving of any name z via channel x and the silent action, respectively. The last one is a conditional capability: it is π if x and y are the same name. The syntax of the processes is the following:

$$P ::= M \quad | \quad P_1 \mid P_2 \quad | \quad \nu z P \quad | \quad !P$$
$$M ::= \mathbf{0} \quad | \quad \pi.P \quad | \quad M_1 + M_2$$

Most of them has the same meaning as in CCS. The restriction $\nu z P$ limits the scope of the name z to the process P, similar to restriction $P\backslash K$ in CCS, but here the scope of a restriction may change dynamically as the result of communication, the so-called *scope extrusion*. Replication $!P$ is the means to express infinite behaviour via the equation $!P \equiv P \mid !P$ of the structural congruence. The *structural congruence* is a relation over processes: informally, two processes are structurally congruent, if they are identical up to structure. In particular, parallel composition

is commutative and associative. The basis definition of structural congruence is a collection of axioms that allows manipulation of the term-structure.

The actions are given by $\alpha ::= \bar{x}y \mid xy \mid \bar{x}(z) \mid \tau$: the first is the sending of the name y via the name x, the second is the receiving of y via x, the third is the sending of a fresh name via x (bound output) and the last one is the silent action.

The original semantics of the π-calculus has been given in terms of *labelled transition system* in the SOS style [191], where transactions are of the form $P \xrightarrow{\alpha} P'$, for some action α. An example of transition is:

$$\frac{P \xrightarrow{\bar{x}z} P' \;\; z \neq x}{\nu z P \xrightarrow{\bar{x}(z)} P'} \qquad \frac{P \xrightarrow{\bar{x}(z)} P' \;\; Q \xrightarrow{xz} Q' \;\; z \notin \mathrm{fn}(Q)}{P \mid Q \xrightarrow{\tau} \nu z(P' \mid Q')}$$

The first rule expresses the extrusion of the scope of a name, whereas the second one expresses that a process P performing a bound output $\bar{x}(z)$ can interact with a process Q that can receive z via x and z becomes restricted in Q. Alternatively, the behaviour of a π-process can be described by means of a *reduction semantics*, where reduction are of the form $P \rightarrow P'$, modelling an internal evolution of the process P. Indeed this ignores the capability of a process to interact with the environment. An example of reduction is:

$$(\bar{x}y.P_1 + M_1) \mid (x(z).P_2 + M_2) \rightarrow P_1 \mid \{^{y}/_{z}\}P_2.$$

In the π-calculus, labelled and reduction semantics coincide.

For that concerns behavioural equivalences *barbed congruence* and *bisimulation* were defined. Barbed congruence is based on the notion of *barbs*, that are observable predicates of the form $P \downarrow_a$ for each name a, which detects the possibility of a process of accepting a communication with the environment at a. An equivalent labelled transition semantics can be defined by exhibiting auxiliary predicates.

4.3 Timed Process Calculi

Important developments have occurred since the formulation of the basic process calculi CCS [168,169], CSP [56,117] and ACP [30,31]. As we explained in Section 4.2, the π-calculus [171] can be considered a development in terms of mobility. Indeed in π-calculus processes are mobile and configuration of communication links is dynamic. Most of the developments of process calculi are collected in the impressive handbook [32].

One important aspect that has be taken into consideration is *time*. There may be good reasons to introduce time in such a way that suitable timing becomes relevant for the correct behaviour of a complex system. This is, for example, the case of most controllers. Their correct behaviour, often derived from physical laws, usually involves actions performed between certain timebounds. Moreover, for certain data communication protocols, whether they behave at an appropriate level of abstraction like a queue depends on the timing of certain actions. Still another reasons may be that there is simply a need to pay attention to performance aspects of a system.

The introduction of aspects of time into the setting of process calculi has received much attention in research and, considering that time is a complex subject, it is not surprising the proliferation of proposals. Indeed, process calculi that incorporate some form of timing enabling quantitative analysis of time performance have been studied extensively by now. The literature regarding this development is quite extensive. We confine our discussion to approaches which are the basis for our calculi described in the second part of the thesis.

According to purposes and applications, different versions of describing timing have been presented in the literature. Baeten and Reniers in [23] have made a distinction between the following choices:

1. **The nature of the time domain**:
   - *discrete time* vs *dense time*: this choice is with respect to which type of time domain is used to describe timing. In discrete time, the time domain is of a discrete nature, whereas in dense time the time domain is of a continuous nature, i.e. between every two moments in time there is another moment in time. The axiom of ACP has been extended to discrete time by Baeten and Bergstra in [19]. Baeten, Bergstra and Reniers [20] have increased the framework of [19] with the silent step τ. In [18] Baeten and Bergstra have proposed a real-time version of ACP; in [103] it is proposed a language based on ACP to describe and verify real-time systems using a discrete time scale; in [181] Nicollin and Sifakis have presented a temporal extension of CCS, the ATP algebra, with a notion of discrete global time; in [199] Reed has presented a real-timed extension of CSP, called *Timed* CSP, while CCS is the starting point for the TCCS calculus of Moller and Tofts [175] and of the calculus called TPL of Hennessy and Regan [111];
   - *linear time* vs *branching time*: in a linear time domain each two moments in time are ordered by some total ordering $\leq$, in a branching time domain this is only a partial ordering.
2. **The way in which time is syntactically described**:
   - *time-stamped description* vs *two-phase description*: if the description of time is attached to the atomic actions, we speak of a time-stamping mechanism. If, on the other hand, time delay is separated from action execution, we have a two-phase approach. For instance, the calculi in [19, 103, 111, 175, 181] have introduced special actions to model the passage of time, although the basis for all those proposals may be found in [36].
3. **The way in which time is semantically incorporated**:
   - *absolute timing* vs *relative timing*: sometimes, it is convenient to describe the passage of time with respect to a global clock, then we have absolute timing, sometimes, it is convenient to describe passage of time relative to the previous action, then we have the relative timing;
   - *time-determinism* vs *time-nondeterminism*: a choice to be made with far reaching consequences is whether a delay may determine a choice. In the literature three versions are encountered. Firstly, if the delay by itself may not determine any choice, we speak of strong time-determinism. Secondly, if a delay of t time by itself cannot determine a choice between alternatives that allow a delay of t time, we speak of weak time-determinism. Finally, if a delay can determine a choice we speak of time-nondeterminism;

- *durational actions* and *instantaneous actions*: it is possible either to assume that actions have no duration, and hence are instantaneous, or that they have a duration. In the latter case, the duration can be specified or unspecified. Example of calculi with durational actions are [67, 68];
- *urgent actions* vs *multi-actions*: if actions occurring at the same time can be ordered, these are called urgent actions, otherwise these are called multi-actions.

Baeten and Middelburg [21] have proposed several timed process algebras treated in a common framework, and related by embeddings and by conservative extensions relations. All the theory presented in the book are generalisations of the ACP process algebra without timing. These process algebras, called $\mathrm{ACP}^{\mathrm{sat}}$, $\mathrm{ACP}^{\mathrm{srt}}$, $\mathrm{ACP}^{\mathrm{dat}}$ and $\mathrm{ACP}^{\mathrm{drt}}$, allow the execution of two or more actions consecutively at the same point in time, separate the execution of actions from the passage of time, and consider actions to have no duration. The process algebra $\mathrm{ACP}^{\mathrm{sat}}$ is a (dense)real-time process algebra with absolute time, $\mathrm{ACP}^{\mathrm{srt}}$ is a (dense)real-time process algebra with relative time. Similarly, $\mathrm{ACP}^{\mathrm{dat}}$ and $\mathrm{ACP}^{\mathrm{drt}}$ are discrete-time process algebras with absolute time and relative time, respectively. In these process algebras the focus is on unsuccessful termination or deadlock. The framework of Baeten and Reniers in [22] extends the framework of [21] to model successful termination for the relative-time case.

The *stochastic process calculus* modelling paradigm has been introduced as an extension of classical process calculi with timing information. This model mainly aims at the integration of functional design with quantitative analysis of computer systems. Time is represented by exponentially distributed random variables that are assigned to each activity in the model. Thus, the semantic model of this stochastic model can easily be transformed into a continuous time Markov chain. This can help to compute performance measures as well as dependability measures. Among these calculi we remember CCS^{+} [219], PEPA [114], EMPA [33], ES-SPA [55] and Sπ [197].

In the following we describe more in details two process calculi, the TPL and the TCBS, that use similar approaches to those used for our timed process calculi described in the second part of this thesis.

TPL

Papers like [199] and [18] have proposed very descriptive languages with which it is possible to describe the minutiae of detailed timing considerations in complex systems. According to Hennessy and Regan [111], there are certain applications for which these languages may be inappropriate because the description may be unnecessary complex. For this reason, Hennessy and Regan have proposed in [111] a very simple and intuitive timed process calculi, the *Timed Process Language* (TPL). TPL presents a mathematically simple notion of time that may be useful in particular application areas such as protocol verification. Indeed protocols are typical examples in which time affects the behaviour of just a small part of the overall system. Hence it is unnecessary using complex formal language to model them. TPL is designed so that the specification of the time-independent part of the system may be treated as usual in process calculi whereas the time-dependent part may be carried out by a simple extension.

The idea of [111] is to introduce into the CCS a specific action modelling time passing. The authors have used the symbol σ to denote it. When a process execute a σ action it means that it is idling or doing nothing until the next clock cycle. The σ action shares many of the properties of the usual actions of CCS, but it is distinguished by certain of its specific properties. This is due to the fact that it represents the passage of time and this part it treated separately in TPL. Thus TPL is characterised by the following time properties:

- *discrete time*: time proceeds in discrete steps represented by occurrence of the action σ;
- *actions are instantaneous*: time is not associated directly with communication actions but occurs independently. This is retained by the implicit assumption underlying CCS that all communications are instantaneous, since in TPL there is a distinction between the passage of time, scanned by σ actions, and all other actions are performed in between occurrences of this time action;
- *time determinism*: the passage of time is deterministic, i.e. the process can reach at most one new state by performing σ action;
- *patience*: a process can idle, i.e. it can perform a σ action, indefinitely until it can communicate. This is an intuitive assumption underlying the usual (asynchronous) theories of process calculus that all process may idle indefinitely and that the semantic theory is formulated in terms of the actions which a process may perform;
- *maximal progress*: a process cannot delay if it can perform a communication, i.e. communications cannot be delayed, they must occur as soon as possible. This is expressed in TPL saying that if a process can perform a τ-action, then it cannot execute a σ action. Indeed, as in CCS, a τ-action represents a synchronisation between a sender and a receiver process. The maximal progress property has been introduce in [71] and it is a common features of many proposed timed process calculi.

The syntax of TPL processes is very similar to the syntax of CCS processes, except for the *delay process* $\sigma.P$, to mean that the process P can do nothing until the next clock cycle and from that moment it behaves as P. This syntax is used to force delay. Moreover, TPL has another new process, the *timeout process* $\lfloor P \rfloor (P')$, coming from the ATP algebra of [181]. The behaviour of the process $\lfloor P \rfloor (P')$ is properly decided by the passage of time in favour of the right hand process, that is if $\lfloor P \rfloor (P')$ can execute a σ action then it evolves into P'. The dynamism of the system is expressed by an operational semantics, divided in two parts. The first is a slight generalisation of the standard operational semantics of CCS where the new action σ plays no rôle, whereas the second part is defined in term of this action. Thus in the first part the transitions are of the form $\stackrel{\alpha}{\longrightarrow}$, where the action α ranges over the sending, the reception and the τ-action as in CCS, whereas in the second part the transitions are of the form $\stackrel{\sigma}{\longrightarrow}$. Below, we report three rules, the first explains how the delay process behaves, the other ones models the actions that the timeout process can perform:

$$\frac{-}{\sigma.P \stackrel{\sigma}{\longrightarrow} P} \qquad \frac{P \stackrel{\alpha}{\longrightarrow} P''}{\lfloor P \rfloor (P') \stackrel{\alpha}{\longrightarrow} P''} \qquad \frac{P \stackrel{\tau}{\not\longrightarrow}}{\lfloor P \rfloor (P') \stackrel{\sigma}{\longrightarrow} P'}.$$

The requirement of the last rule is for the maximal progress, as stated above.

The goal of the authors of [111] is to extended the semantic theory of processes based on testing [109]. Using the operational semantics, Hennessy and Regan have defined an operational preorder on timed processes based on *must* testing, where also time has to be considered. The characterisation of testing preorder is expressed in terms of *barbs* [192, 226], that is sequences of the form $s_1 A_1 \dots s_k A_k$, where s_i is a sequence of actions that a process can perform to arrive at a specific state and A_i is the set of next possible action from that state. Thus two processes are compared according to the barbs they can exhibit and the barbs may be compared using defining orders.

TCBS

Prasad [195] has proposed a timed variant of his *Calculus of Broadcasting Systems* (CBS) [194], called TCBS. CBS is a simple and natural CCS-like calculus modelling broadcast behaviours: processes send data one at a time and data are received instantaneously by all others. This is the most important feature that allows to distinguish CBS from almost all other process calculi, which use point-to-point communication.

In TCBS the passage of time is dense. Besides to standard constructs in process calculus, in the syntax of the processes of TCBS we find the *timeout process* $\delta : P$, whose meaning is similar to the timeout process $\lfloor P \rfloor(P')$ of [111]. The communication rules are modelled by means of an operational semantics, containing the relations $\xrightarrow{w!}$, $\xrightarrow{w?}$, $\xrightarrow{\tau}$ and $\xrightarrow{\delta:}$ over processes, modelling transmission, receiption, silent action and passage of δ instants of time. We write only the rules for timeout process:

$$\frac{-}{\delta : P \xrightarrow{\delta:} P} \quad \frac{P \xrightarrow{w?} P'}{\delta : P \xrightarrow{w?} P'} \quad \frac{P \xrightarrow{w!} P'}{0 : P \xrightarrow{w!} P'} \quad \frac{-}{(\delta + \delta') : P \xrightarrow{\delta:} \delta' : P}.$$

The third rule underlines the assumption that if a communication occurs then it cannot be delayed. This is expressed by 0 delay. Indeed also the TCBS calculus, as TPL described above, is characterised by the maximal progress property and the time determinism and the patience properties, as well. The last rules together with rule

$$\frac{P \xrightarrow{\delta:} P' \;\; P' \xrightarrow{\delta':} P''}{P \xrightarrow{(\delta+\delta'):} P''}$$

introduces a further property of TCBS, that is *time additivity*. This properties is associated with density; it does not hold for TPL because it is a discrete timed calculus. Additivity is expressed by the following statement:

$$P \xrightarrow{\delta:} \xrightarrow{\delta':} P' \text{ iff } (\delta + \delta')P \xrightarrow{(\delta+\delta'):} P'.$$

For that concerns the behavioural equivalences, Prasad has proposed a strong bisimulation and a weak bisimulation, that, as usual, abstracts over τ actions, and in which also time passing actions are taken into consideration. A delay prefix

operator can be derived up to weak bisimulation if time dependent behaviour is allowed.

Developed with the goal of providing very general process models, most specialised calculi are derived from the seminal ideas of calculi described in previous sections. In the next section, we provide a description of process calculi for wireless systems.

4.4 Process Calculi for Wireless Systems

The majority of process calculi take point-to-point communication as primitive. However, in order to implement a calculus to model systems with broadcast communications, as wireless networks, it seemed that point-to-point fashion led to complicated and sometimes little intuitive encodings. Thus dedicated calculi with broadcast as primitive could avoid many problems. Examples of peculiarities of wireless devices with respect to the more conventional wired communication are:

- the local broadcasting, meaning that a transmission spans over the limited area, called *cell*, and does not reach all nodes in the network, but only the so called *neighbours*;
- within the cell, there may be the presence of obstacles that disables the reception;
- nodes may not transmit and receive at the same time, thus, there may be undetected interferences and collisions when two devices try to transmit at the same time;
- node are mobile, that is they can break links with old neighbours and establishing fresh links with new devices.

In the last years many process calculi to describe wireless networks have been proposed. In this section we describe some of them.

CWS

Mezzetti and Sangiorgi [167] have proposed a *Calculus of Wireless Systems* (CWS) to describe interferences. CWS has nodes, which represent the devices of the system, that can be composed in parallel. Inside a node there is a sequential process, which models the behaviour of that device. Each node has a location and a radius that define the cell over which that node can transmit. CWS does not model the movement of devices.

When developing the semantics for CWS, the authors have decided to refine the view on transmissions and observe, for each node, the change of state between transmission and reception (and vice versa), rather then single transmissions. Further, in accordance with physical wireless devices, they have assumed that, when a device is not performing a transmission, its antenna is in reception mode. An *event* is the state change; they call *begin transmission* the event which corresponds to a device which initiates a transmission, and *end transmission* the event which corresponds to a transmitter which finishes its transmission.

The authors have presented a Reduction Semantics (RS) and a Labelled Transition Semantics (LTS). The RS and the LTS differ in the approach followed to

check the interferences. In the LTS, the derivation of a transition takes a set of active transmitters, i.e., a set of devices that are currently engaged in a transmission as a parameter; then, the various possibilities of interferences are checked against such a parameter. In the RS, by contrast, such parameter is absent; the checks for interferences are confined into the rules of the semantics itself. As a consequence, however, the derivation of a reduction has to be decomposed into three separate sub-derivations, which are defined using some auxiliary relations on an extension of the calculus. Each one of such auxiliary relations implements a logical distinct sub-component of the mechanism with which transmissions are performed in wireless systems, e.g., individuation of the transmitter, individuation of its cell, and, for each receiver in the cell, the individuation of possible interferences. The main extensions of the calculus are given by markers that are placed on the network nodes and that represent a partial state of the node within the whole reduction.

The main technical result of the paper in [167] is the equivalence between the two semantics.

CBS$^\sharp$

In [178] Nanz and Hankin have proposed a framework for specification and security analysis for ad hoc networks. We describe more in detail this calculus in Section 5.3.

CMAN

In [94] Godskesen has suggested a *Calculus for Mobile Ad hoc Networks* (CMAN), based on an extension of the π-calculus. The goal of CMAN is to facilitate the modelling of two very important features of ad hoc networks: (i) *mobility*, for which nodes autonomously change localities and thereby change their connections and hence the topology of the network, and (ii) *spatially oriented* broadcast, for which messages will only reach those nodes within the communication range of the emitting node. Thus the author has assumed that nodes may move arbitrarily, change their neighbour relationship and thereby change the network topology. The syntax of a node is $\lfloor p \rfloor_l^\sigma$, where l is its location and σ is the set of the locations of its neighbour, letting the topology be explicitly part of the network syntax and letting the topology change as a consequence of computational steps. Nodes composed by parallel composition constitute a *network*. In CMAN broadcast is *atomic* in the sense that all neighbours at the time of the broadcast, and only those, can listen to and receive the broadcasted message.

Connection and disconnection of nodes are bidirectional and they are modelled by reduction rules. Spatially oriented broadcast is also realized by a broadcast reduction rule, labelled by the location l of the emitting node, where the node at location l broadcasts to all nodes to which it is connected in the current topology. It is also modelled the topic that a message could be lost.

A novel contribution of the work in [94] is that the author has chosen to work with a family of broadcast reductions, one for each locality in the network. This allows an external observer to observe the locality (node) in charge of the synchronous broadcast. However, since it may be unrealistic for an observer to cover the whole network, it is introduced the notion of a hidden node $\nu k \lfloor \langle t \rangle.r \rfloor_k$, i.e. a

node with the location name k restricted. It may connect to other nodes extruding its location name and subsequently send/receive messages to/from its neighbours, but the emission from a hidden node cannot be observed by an external observer, hence the reduction obtained is not a broadcast reduction.

Moreover the author has defined a semantic choosing to abstract from observability of node mobility. CMAN is equipped with a natural reduction semantics and congruence, and a co-inductive sound and complete bisimulation characterisation. Finally, some primitives to model cryptographic operations are added, modelling a possible attacks to the ARAN [209] routing protocol.

ω-calculus

For many aspect Singh, Ramakrishnan and Smolka in [214] have proposed a process calculus very similar to the one presented in [94], modelling the two principal features of a mobile ad hoc network: broadcast spatially oriented communication and mobility. The authors have called their calculus the ω-calculus, operating a separation of a node's communication and computational behaviour, described by an ω-process, from the description of its physical transmission range, referred to as an ω-process *interface*. Ideally, the specification of a mobile ad hoc network node's control behaviour should be independent of its neighbourhood information, even if in a traditional process calculus the model must intermix the computation of neighbourhood information with the protocol's control behaviour, rendering such models unnatural and complex.

The syntax of the calculus comprises a set of nodes, each of which runs a process. Processes are specified by extending π-calculus process expressions with broadcast-communication primitives, thus is quite standard, while the major difference with respect to the π-calculus can be seen in the syntax of node expressions, because the author have design with $P : G$ a basic node with process P and interface G. As usual, networks are given by the parallel composition of nodes. The topology of the network is expressed as a node-connectivity graph, where an edge between two nodes indicates that they are in the same transmission range. The ω-process interfaces are comprised of *groups*, which operationally function as local-broadcast ports. A group of a node represents the maximal sets of neighbouring nodes. Mobility is captured in the ω-calculus via the dynamic creation of new groups and dynamically changing process interfaces. The authors have provided a formal semantics in terms of labelled transition systems and structural-equivalence rules.

The authors have showed that the state reachability problem is decidable for finite-control ω-processes. They also have proved that their calculus is a conservative extension of the π-calculus. Then they have defined a bisimulation equivalence, showing that it is a congruence. Congruence results are also established for a weak version of bisimulation equivalence, which abstracts away from two types of internal actions: τ-actions, as in the π-calculus, and μ-actions, signalling node movement. They additionally have defined a symbolic semantics for the ω-calculus extended with the mismatch operator, along with a corresponding notion of symbolic bisimulation equivalence, and established congruence results for this extension as well. They have illustrated the practical utility of the calculus by outlining a formal ω-calculus model of a leader election protocol for mobile ad hoc

networks [227] and the AODV protocol [186]. Finally, they have described how the operational semantics is directly encoded in Prolog, forming the first step in constructing a model checker for the calculus.

CMN

Merro in [163] has proposed a *Calculus for Mobile ad hoc Networks* (CMN). This calculus focuses on an observational theory for mobile ad hoc networks more than on topology of the network or on the modelling of routing protocols, as for [214] and [94]. A network is represented as a collection of nodes, composed in parallel. Messages are broadcasted and received via channels, that may be public or private to a set of node. The restriction operator ν is used to model the privacy. A node has this syntax: $n[P]^{\mu}_{l,r}$, where n is the network address, P is the sequential process running in this node, l is the location, r is the transmission radius, μ is the mobility tag (a node may be stationary or mobile: in the first case the tag will be $\mathtt{s}$, in the second $\mathtt{m}$). The location l and the radius r are used to determine the cell over which a node can broadcast values through channels: only nodes within the cell of the transmitting node will receive the message sent. It is assumed the presence of protocols to avoid transmission collisions; moreover the calculus does not deal with cryprographic underpinnings.

The dynamics of the calculus is specified by the *reduction relation* $\twoheadrightarrow$ over networks that, as usual, relies on another relation, the *structural congruence*, $\equiv$, which allows manipulations of term structure to bring potential inter-actors together. Beyond the standard rules in process calculi, the reduction rules model broadcast, movement and message loss. Broadcast is successfull when the nodes are at a specific distance from the transmitting node; it is used a function $\mathrm{d}(l, l')$, where l and l' are respectively the locations of the transmitting and receiving node. If r is the radius and the computed distance is equal or less than r, then the communication may success. The message loss is modelled assuming that there exists no receiving. For the movement, the only requirement is that the mobility is setted to $\mathtt{m}$. It is also defined a labelled transition system, dividing into two set of rules, one for processes and one for networks.

For that concerns behavioural equivalences, in CMN only the transmission of messages can be observed. In fact, in a broadcasting calculus, an observer cannot see whether a given process actually receive a particular broadcasted value. For this reason the notion of observability is represented by the transmission of messages that can be detected by a pervasive observer, i.e. an observer that can listen anywhere, at any channel. At this point it is defined the *reduction barbed congruence* that is the largest symmetric binary relation over networks that respect some specific properties (reduction closed, barb preserving and contextual). The author has defined an appropriate notion of simulation/bisimulation for ad hoc networks, proving that this bisimilarity implies reduction barbed congruence, and hence represents a valid method for proving that two networks are reduction barbed congruent. Actually, it is proved that bisimilarity is more than a proof technique; it represents a complete characterisation of reduction barbed congruence. Indeed reduction barbed congruence is contained in the labelled bisimilarity. As usual in process calculi, it is proved the correspondence between labelled bisimilarity and reduction semantics. Since the observational theory is developed, using

(bi)simulation proof method, they are formally proved some non-trivial properties of ad hoc networks.

RBPT and CNT

In [91] Ghassemi, Fokkink and Movaghar have presented a process calculus for modelling and reasoning about mobile ad hoc networks and their protocols. Their calculus is called *Restricted Broadcast Process Theory* (RBPT). They have modelled the essential modelling concepts of ad hoc networks, i.e. local broadcast, connectivity of nodes and connectivity changes.

In CWS [167], CMAN [94], CMN [163], and the ω-calculus [214], the topology is defined as a part of the syntax, while the semantics of CBS$^\sharp$ [178] is quantified over a set of node configurations. In the former calculi (except CWS, where the topology is considered static), a process evolves syntactically (to reflect topology changes) by the application of mobility rules defined in the semantics, while in the latter, the underlying configuration changes arbitrary in the semantics. The authors have transfered topology concepts completely to the semantics, similarly to CBS$^\sharp$, although their labelled transition systems and the equivalence relations are completely different. In [178] a transition to the next state is examined for all possible valid graphs (those contained in the network topology fixed a priori) whereas in [91] a transition is examined for all graphs containing the connections used in a communication, constituting valid topologies. A valid topology is a set of connectivity relations between nodes such that the topology invariant is satisfied. A *topology invariant* is a set of predicates defined over nodes. In RBPT, it is the network behaviour that defines a set of valid topologies under which such behaviour is correct, rather than the underlying topology dictates the network behaviour.

The authors have provided a formal operational semantics for their process calculus, parametrised over the invariants, and they have defined equivalence relations on protocols and networks. They have showed how their calculus can be applied to prove correctness of a simple ad hoc routing protocol.

The work in [91] has been refined in [92], where the same authors have provided an equational theory for RBPT. Then they have exploited an extended calculus called *Computed Network Theory* (CNT). The previous notion of invariant is refined with then notion of *restrictions*. They have assumed a binary relation $>$ over the set of locations (or node addresses), which imposes connection relations between addresses. A relation while $A > B$ denotes a node with address A is connected to a node with address B. A network restriction is a set of relations $>$. The transitions of the operational semantics are now subscripted by a set of network restrictions, those that are necessary to the transition to fire.

4.5 Chapter Summary

In this chapter we sketched a brief history of process calculus. The early work centred around giving semantics to programming languages involving a parallel construct. Here, two changes were needed: first of all, abandoning the idea that a program is a transformation from input to output, replacing this by an approach

where all intermediate states are important, and, secondly, replacing the notion of global variables by the paradigm of message passing and local variables.

In the 1970s, both these steps were taken, and the process calculi CCS and CSP evolved. In doing so, process calculus became an underlying theory of all parallel and distributed systems, extending formal language and automata theory with the central ingredient of interaction. We briefly described the so called foundational process calculi, as CCS, CSP, ACP and π-calculus.

Important developments have occurred since the formulation of the basic process calculi (CCS, CSP, ACP). Among these developments is noteworthy the introduction of timing aspects into this setting. According to purposes and applications, different versions of describing timing have been presented in the literature. In the present chapter, we described general features of timed process calculi and analysed some timed process calculi useful to understand our approaches.

In the following years, much work has been done, and many process calculi have been formulated and many extensions were added. In particular, we described some process calculi developed to address the features of wireless networks. They concern with mobility, interferences, spatially oriented broadcast.

In the next chapter we give an overview of formal techniques for security analysis in wireless networks.

5

Formal Techniques for Security Analysis

5.1 Introduction

The usual security goal of information and communication systems is preventing passive attacks and detecting active ones. When and where an attack occurs this is a deliberate attempt to compromise the system that reaches a non-desirable state or it behaves in a non-desirable way, and then its functionalities are compromised.

Formal specification and security analysis have a long research history in computer science, as for the development of network protocols. However, only few works to date have attempted to bring these two branches of research together, let alone in the small context of wireless networks.

Security goals can be achieved by *physical* protection or *protocol/algorithmic* measures. Typical examples of physical protection are when a server is locked in a room under continuous video surveillance or when a device is placed inside some tamper resistant packaging, e.g., smart cards. Physical protection is very effective, but it is often very expensive as well and, even worse, it is not always applicable. For instance, in wireless systems the communication takes place over a radio channel. As a consequence, the access to the wireless channel cannot be prevented by physical means. When physical protection is not feasible or very expensive, algorithmic measures can be used. Most of these algorithmic measures are based on cryptographic algorithms and on protocols that allow for secure communications over insecure channels at the cost of physically protecting a limited amount of key material only.

A protocol defines a sequence of interactions between entities designed to achieve a specific goal. However, the particular nature of wireless systems poses a number of challenges in designing and also analysing protocols. Formal techniques are therefore needed to establish a mathematically rigorous connection between protocol modelling and desired goals. We can divide protocol analysis techniques into *automated-based methods*, in which tools and automation are used, *proof-based methods*, focusing on translating protocol steps directly into some mathematical formalism enriched with proof techniques, and *language-based methods*, that start out by modelling a security protocol in some process calculus and then they exploit standard static analysis techniques such as model checking, static analysis, type systems, or abstract interpretation. Very often automated-based and language-

based methods come together, as the latter often support tools. Among proof-based methods we remember BAN logics, Strand Spaces, and theorem proving.

In general, protocol analysis consists of two main tasks: *falsification*, that is that is finding flaws in protocols that are not correct, and *verification*, that is proving their correctness. Automated-based techniques usually focus on falsification, whereas proof-based methods focus on verification, even if they may sometimes only be able to indirectly indicate flaws.

According to the protocol classes they analyse, protocol analysis techniques can be distinguished into *security protocols* and *communication protocols*. The goal of a security protocol is to establish a security property between communication partners, e.g. authentication or secrecy. These protocols usually make use of cryptographic mechanisms such as encryption and digital signatures in order to execute correctly in hostile environments where an attacker may deliberately try to compromise the protocol. On the other hand, communication protocols are designed to establish communication between agents in a network, which includes various tasks such as link establishment, routing, or end-to-end transfer of data. In comparison to security protocols, communication protocols are usually far more complex in terms of message contents and required internal computation.

The use of cryptographic primitives is very common in the area of systems communication. On the one hand, various mathematical issues concerning these primitives have been covered, leading to a better understanding of the foundations of cryptography, on the other hand, the use of these cryptographic primitives does not give full guarantees about the fulfilment of the security requirements analysed in Section 2.4.2. In wireless communications many unpredictable threats can rise up and most of them cannot be prevented by cryptographic primitives. Starting from these observations, a branch of research in computer security assumes cryptographic primitives to be perfect, and uses a black box view of cryptography.

Formal methods have been proved efficient modes both to better define the goals of the security protocols and the mechanisms to achieve them, and to offer a precise description of the interactions among the involved entities. The protocol to be analysed is described in a given language. Thus in such a language a formal specification of the security properties to be verified is presented. In order to prove whether or not these security properties are accomplished, it is simulated an *hostile* environment, in which the presence of malicious agents is considered. Thus the protocol is proved secure or not by formally analysing its development in this environment. More precisely, the protocol is running together with honest participants and malicious ones.

We end this introduction with an outline of the present chapter. Section 5.2 gives an overview of some formal methodologies that have became popular in the last decades for the modelling and analysis of computing systems. In Section 5.3 we describe formal models for wireless security. Finally, in Section 5.4 we provide a summary of the arguments of the chapter.

5.2 Methodologies

In wireless networks, the flexibility provided by the open broadcast medium and the cooperativeness of the mobile devices, in particular in MANETs, introduces

new security risks. As part of rational risk management, it should be possible to identify these risks and take appropriate actions. In some cases this is possible, in other cases we may have to accept that vulnerabilities exist and seek to take appropriate actions when we believe someone is attacking us. In [57, 158] we can find a detailed overview of current methodologies for security analysis in wireless networks and its applications. In the following sections, we deal with the most common ones.

Model Checking

Model checking techniques have been initially developed to reason about correctness of discrete state systems. Then they have been extensively used also to verify real time and hybrid systems. In particular in [16, 82, 152, 156, 159, 228] the modelling and analysis of security protocols have been studied.

The model checking technique allows to check whether a given model satisfies a given logical formula. We can formulate a general model checking problem as follows: *given a property, expressed in a given logic, it is verified whether it is satisfied by a given model, specified in some formal language*. Such a language can be source code descriptions in a hardware description language or a special-purpose language. A finite state machine is used to represent the executed code. For example, if the assumed finite state machine is a direct graph, the nodes represent the possible states reached during the execution and the edges represent the transitions from one state to another one. Moreover, in particular states specific properties may held, represented by atomic propositions. The main practical challenge of model checking is to overcome the *state explosion problem*: the size of the state space grows exponentially with the size of the system description. As finite state model checking methodology is considered, it is sometimes not possible to analyse protocols in all generality. As a consequence, finitary restrictions or drastic simplifications have to be made.

A first attempt to overcome the limitations of model checking technique has been presented by Basin, Mödersheim, and Viganò in [26], where the authors have presented the model checker OFMC, that combines two ideas for analysing security protocol. The first idea is the use of lazy data types as a simple way of building efficient on-the-fly model checkers for protocols with very large, or even infinite, state spaces. The second idea is the integration of symbolic techniques and optimisations for modelling a lazy Dolev-Yao intruder [73], whose actions are generated in a demand-driven way.

Among the tools for model checking used for wireless networks, we cite SPIN model checker [118] and AVISPA toolkit [16, 228]. In particular, SPIN has been used to automatically find vulnerabilities in the discovery route process of ad hoc on demand routing protocols, whereas the model checking functionalities of the AVISPA toolkit have been used in [224] to verify some building blocks of security protocols for wireless sensor networks.

Bhargavan, Obradovic, and Gunter [38] have used SPIN to verify routing protocols for mobile ad-hoc networks. For a loop-freedom property expressed in temporal logic they can use the model checker SPIN to expose flaws on a fixed network setup. Andel and Yasinsac [13] have also used SPIN to automatically analyses a route

discovery process. SPIN allows automated evaluation tracing the adversary's actions that leads to an attack. In particular, the protocols analysed were DSR [129] and SRP [184].

Model checking technique is also used in [215] to analyse the authentication process in 802.11i standard.

Static Analysis

Static analysis is a static technique for predicting safe and computable approximations to the set of a value that the objects of a program may assume during its execution [183]. The goal of static analysis is to automatically identify many common coding problems before a program is released. Static analysis tools examine the text of a program statically, without attempting to execute it. Theoretically, they can examine either a program's source code or a compiled form of the program [62]. Static analysis tools usually offer an high degree of automation and therefore they may require less human intervention with respect to other techniques, e.g. theorem provers. Moreover, they are efficient: the analysis has polynomial times with respect to the specification. On the other hand, even if static analysers may discover a lot of security flaws, like buffer overruns, access problems, exception handling, format string problems, however they cannot solve all security problems. Generally, static analysis tools look for a fixed set of patterns, or rules, in the code. Although more advanced tools allow new rules to be added over time, if a rule has not been written yet to find a particular problem, the tool will never find that problem.

Available tools are either commercial, e.g. [69, 193, 196], or academic, e.g. [42, 52]. In academic area, it is worthy to notice the work of Bodei et al. [49–52], who have shown that such techniques, namely *control flow analysis*, can also be applied to process calculi, and have derived secrecy and authentication properties in variants of the π-calculus [171] and Spi Calculus [6]. In [236] static analysis is used to investigate which extensions are really necessary, and which can be omitted, without compromising the correctness of the protocol.

Automated Theorem Proving

Automated theorem proving (ATP), or *automated deduction*, is currently considered the most well-developed subfield of automated reasoning, and it consists in proving of mathematical theorems by computer programs. These programs show that some statements are a logical consequences of a set of axioms or hypotheses. ATP systems are used in a wide variety of domains, e.g., mathematics, management, electronics. The language used to write conjecture, axioms and hypotheses is a logic, in particular the first-order logic. First-order theorem proving is one of the most mature subfields of automated theorem proving. The first-order logic is expressive enough to allow the specification of some kind of problems. On the other hand, it is semi-decidable, and a number of sound and complete calculi have been developed, enabling fully automated systems. More expressive logics, such as higher order and modal logics, allow the convenient expression of a wider range of problems than first-order logic. Anyway, theorem proving for these logics is less well developed. The power of an ATP system, is actually its formality, that allows

no ambiguity. Actually, this is the power of using a formal method approach for solving some kind of problems given certain premises.

ATP is a technique widely used among the formal methods community for the verification of security properties. Among the literature, we cite the Isabelle/HOL toolkit [126], Coq [66], ProVerif [41] and Athena [216].

In [42] the author have used ProVerif to analyse the standard protocol defined by the Bluetooth specification.

The work in [232], by Yang and Baras, focuses on developing formal models (a combination of model checking and theorem proving) for ad hoc routing protocols, aimed at modelling insider attackers. Insider attackers are those that, besides replaying messages generated by legitimate parties, and degrading communication by jamming the lower layers of communication, have the legitimate keys necessary to generate authentic messages. The protocol under investigation is the SAODV [238]. The authors in [232] have aimed at testing the presence of uncompromised paths through the network. An uncompromised path exists when the routing service is guaranteed, despite of the presence of any number of insider attackers.

Type Systems

Another program analysis approach to protocol security is based on *type systems*. The first attempt has been proposed by Abadi in [3], where the author have developed formal rules for achieving secrecy properties in security protocols. This approach has successively been extended for the π-calculus [171] by Abadi and Blanchet in [4]. Data are classified into *Public, Secret*, and *Any*, that subsumes both the other levels. The goal of the typing system is to establish that if a process P typechecks, it does not leak values of level *Any*. In other words, this means that P does not leak its secret inputs. The notion of leaking is formalised in terms of testing equivalence.

Abadi and Blanchet in [4] have also shown the relation of the type systems approach to the Prolog-based protocol checker ProVerif [41] which translates a process into logic-programming rules which can be automatically evaluated. Fournet, Gordon and Maffeis [87] have considered the problem of statically verifying the conformance of the code of a system to an explicit authorisation policy. They have formalised their safety criterions in the setting of a process calculus, and have presented a verification technique based on a type system. Hence, they can verify policy conformance of code that uses a wide range of the security mechanisms found in distributed systems, ranging from secure channels down to cryptographic primitives, including encryption and public key signatures. In [64] the authors have defined an extension to the higher-order π-calculus [208] for analysing protocols that rely on remote attestation. They have also provided a static analysis technique for ensuring safety in the presence of arbitrary attackers.

5.3 Formal Analysis for Wireless Security

In this sections, we focus on formal languages able to model wireless secure protocols. We also describe very general schemata for the definition and analysis of security properties.

Non-Interference and General Schemata

In the literature, several efforts have been made to prevent the unauthorised information flow in multilevel computer systems [28], i.e. systems where processes and objects are bound to a specific security level. The seminal idea of *non interference* proposed in [96] aims at assuring that information can only flow from low levels to higher ones. The first taxonomy of non-interference-like properties has been uniformly defined and compared by Focardi and Gorrieri in [81, 82] in the context of a CCS-like process calculus. In particular, processes in the calculus were divided into high and low processes, according to the level of actions that they can perform. To detect whether an incorrect information flow (i.e. from high to low) has occurred, a particular non interference-like property has been defined, the so-called *Non Deducibility on Compositions* (NDC). Intuitively a system is NDC if the set of its low level views cannot be modified by composing such a system with any high level process. The NDC property has been reformulated in terms of network security. For instance, in [84–86] the low-level processes are used to specify the communication primitives of a cryptographic protocol, whereas an high-level process is intended to be any possible adversary. The behaviour of the protocol running in isolation is compared with the behaviour of the protocol running in parallel with an adversary.

Focardi and Martinelli in [86] have proposed a *Generalised* Non Deducibility on Compositions (GNDC) as a uniform approach for the definition and the analysis of various security properties. The GNDC is actually based on this idea of checking a system against all the possible hostile processes. This general schema has the following form:

$$E \text{ satisfies } P^{\alpha}_{\triangleleft} \text{ iff } \forall X \in \text{ Env } : E \,||\, X \triangleleft \alpha(E)$$

Basically, the general property $P^{\alpha}_{\triangleleft}$ requires that the system E satisfies a specification $\alpha(E)$ when composed (in parallel) with any (possibly hostile) environment X. The property is parametric with respect to $\alpha(E)$ and $\triangleleft$ that can be instantiated in order to obtain different security properties. In particular, $\alpha(E)$ is a function between processes that, given E, specifies which should be its intended correct behaviour, whereas $\triangleleft$ is a relation between processes representing the actual notion of observation. Thus, with $E \,||\, X \triangleleft \alpha(E)$ it is checked if process E shows a correct behaviour even in the presence of an adversary X.

This universal quantification over all the possible intruders could be problematic when trying to check a property, since the verification may span over infinitely many processes (one for each intruder). Usually, this problem is overcome by analysing the case where only the *most powerful* intruder is considered. If the property holds in the presence of the most powerful intruder then it will certainly hold even if less powerful ones are considered. Basically, the most powerful intruder is a process that knows a set of messages φ, can communicate only over fixed channels, can receive every message passing over these channels (increasing in such a way its knowledge) and, finally, can send over these channels every message that it can deduce starting from φ.

Gorrieri, Locatelli and Martinelli [98] have proposed a timed version of GNDC, called *timed* Generalised Non Deducibility on Compositions (tGNDC). It rephrases

the analogue GNDC, but in a timed setting. In this context, $\lhd$ and α are, respectively, a timed behavioural relation and a timed property. In [100] the tGNDC has been used to verify properties of the μTESLA protocol [188] for sensors networks.

Process Calculi

The formalism of process calculus introduced in Chapter 4 has been successively extended to cope with the possibility to detect flaws in communication and security protocols [82, 153]. A paradigmatic example of how errors can be found in security protocols using a process algebra based formalism and an analysis tool is the case of the Needham-Schroeder protocol [179]. Lowe has found an authentication attack [151] to this protocol. This attack can be described as follows: as a consequence of the interleaving of the two sessions, an intruder discovers sensitive data, by pretending to be one of the honest participant. The attack was found by running a tool for formal automated analysis, on an algebraic specification of the protocol, based on CSP [56, 117]. Lowe subsequently proposed to fix the protocol [152]. By running the modified specification with the same tool, he did not find any attack.

In the setting of wireless networks, here we describe the CBS$^\sharp$ process calculus. In [178] Nanz and Hankin have proposed a framework for the specification and the security analysis for mobile wireless networks. The authors have called their calculus CBS$^\sharp$, as they considered the CBS calculus [194] the first calculus to have broadcast as communication primitive as a direct ancestor of their calculus.

In CBS$^\sharp$ there is clear distinction between the processes and the topology of the network, which can change independently by the actions. The goal of the authors is to present a framework, based on a broadcast calculus and static analysis, which allows mobile wireless networks and their security to be formally described and analysed. They have focused their attention on the routing, a central issue in wireless systems.

In CBS$^\sharp$ the syntax of a node is $n[P, S]$, where n is the location, P is the process and S is a private store. Indeed, according to the authors, nodes are not memoryless but store information in routing tables with impact on future actions. Furthermore, the local accessibility allows a clear distinction between the event that information become available to an eavesdropper and the event that the same information is disclosed to the attacker capturing the store. As usual in process calculi for wireless systems, a network N is a parallel composition of nodes.

The authors in [178] have extended CBS by introducing the *local broadcast*: sent messages are not received globally but only by adjacent neighbours of the sender. The notion of adjacency is made explicit by the *connectivity graph*, indicated with G, that is a common graph whose vertex set is a subset of locations of nodes, indicated with $m, n, \ldots$, and an edge (m, n) means that nodes at locations m and n are neighbours. Connectivity graphs are used to describe the connections between nodes for a particular moment in time. As usual the operational semantics is expressed by a labelled transition relation, equipped by a connectivity graph, a reduction relation and a structural equivalence. The transitions of the operational semantics are of the form:

$$N \xrightarrow{(U,m)\#}_G N'$$

In a label $(U, m)\#$, the following communication modes can be expressed: sending $(U, m)!$, reception $(U, m)?$, and loss (U, m): of the term U sent out by m. Nodes may drop messages if they are not within the transmission range of the node who is broadcasting.

The authors have developed notions of behavioural equivalence over networks that include the notion of connectivity and then they have identified a security property expressed through the behavioural equivalences of the calculus. This property expresses *topology consistency* as a building block for routing security, which focus on the notion of *mediated process equivalence*: processes are considered equivalent if they have the same capabilities to store items. Therefore security is determined by the relation between the actual environment conditions and what nodes believe about their environment. The belief is expressed by routing tables built by the private store.

Finally, the authors have showed how to combine their calculus and the *control flow analysis* into the proposed framework. They have applied a control flow analysis technique over the networks, that leads to an over approximation of the set of terms which may be transmitted, together with their senders, and the set of terms which may be stored, together with the location of the storage. This enable to automatically check whether networks are mediated equivalent and prove security properties for network protocols. In particular, they have described μSAODV, a simplified version of the combination of the routing protocol AODV [186] and its security extension SAODV [238]. The framework has thus been used for automated verification. From the analysis results for μSAODV the authors have concluded that the protocol is insecure and have presented a simple attack.

Process calculus has been used to formalise and analyse routing protocols in some other works. For instance, the CMAN of Godskesen [94] has been used to provide a formalisation of an attack on the cryptographic routing protocol ARAN [209]. The author has used his weak bisimulation $\approx$ to show that a simplified version of ARAN is insecure. In particular, called A the version of the routing protocol and given a specification of an intruder I, Godskesen has proved that $A \not\approx A \mid I$.

Singh, Ramakrishnan and Smolka [214] have used their ω-calculus for modelling the AODV [186] routing protocol. In particular, they have used a PROLOG encoding of the semantics of their calculus to develop and analyse the protocol.

5.4 Chapter Summary

A verification of correctness for wireless protocols is an essential requirement for successfully continuing with the developing of the related technologies. Indeed, the huge proliferation of commercial proposals in the area, as well as the widespread usage in everyday life, demands assurance for essential security properties over the wireless links. Recent literature demonstrates that the use of formal methods and tools, popular means for the analysis of security aspects of computer protocols, can provide the solution to the quest for verification.

In this chapter, we surveyed several approaches for modelling and verifying wireless secure procedures and we briefly described their application to the verifi-

cation of common standards in the area of short range communication (Bluetooth, IEEE 802.11, IEEE 802.16, routing protocols).

Part II

Contributions

6

A Timed Calculus for Wireless Systems

6.1 Introduction

The IEEE 802.11 standard [123] contains a series of specifications for wireless LAN technologies. The basic building block of an 802.11 network is the *Basic Service Set* (BSS), which is a set of stations that have successfully synchronised and that use radio transceivers to broadcast messages. In *Independent* BSS (IBSS), stations communicate with each other without using any *distribution system*. IBSS networks are sometimes referred to as *ad hoc networks*. In this chapter, we propose a formal model for IBSS networks paying particular attention to *communication interferences*. Communication interferences represent one of the main concern when evaluating the performances of a network in terms of *network throughput*, i.e. the average rate of successful message delivery over a communication channel.

In concurrent systems, an interference occurs when the activity of a component is damaged or corrupted because of the activities of another component. In Ethernet-like networks, communication channels are full-duplex: a node can transmit and receive at the same time. As a consequence, *collisions* caused by two simultaneous transmissions are immediately detected and repaired by retransmitting the message after a randomly-chosen period of time. This is not possible in wireless networks where radio signals span over a limited area, called *transmission cell*, and channels are *half-duplex*: on a given channel, a device can either transmit or receive, but cannot do both at the same time. As a consequence, in wireless systems communication collisions can only be detected at destination by receivers exposed to different transmissions.

We analysed in detail in Section 4 the formalism of process calculus. In the last twenty-five years, process calculi [30,60,112,169,171] have been intensively used to study the semantics of concurrent/distributed systems, and to develop verification techniques for such systems. In the literature, there exist a number of process calculi modelling wireless systems [91,92,94,163,167,178,214]. All these calculi rely on the presence of some MAC-level protocol to remove interferences. However, in wireless systems *collisions cannot be avoided* although there are protocols to reduce their occurrences (see, for instance, the IEEE 802.11 CSMA/CA protocol [123] for unicast communications). We believe that communication collisions represent a

serious concern that should be taken into account in a timed model for wireless systems.

Many protocols for wireless networks rely on a *common notion of time* among the devices, provided by some clock synchronisation protocol. Most clock synchronisation protocols for ad hoc networks [89, 148, 174, 212, 220, 235] follow the clock correction approach correcting the local clock of each node to run in par with a global time scale, as described in Section 2.3. We remember that an excellent survey of existing clock synchronisation protocols for sensor networks and more generally for ad-hoc networks can be found in [221]. This approach heavily relies on *network connectivity*. In a connected network all nodes are in touch with each other, although not always directly. Wireless networks are usually assumed to be connected; disconnected devices can be considered as not being part of the network as, in general, they need to re-authenticate to rejoin the network.

We propose a *Timed Calculus for Wireless Systems TCWS*, in which all wireless devices are assumed to be *synchronised*, using some clock-correction synchronisation protocol. Thus, TCWS is a process calculus with *absolute timing*, where all timing refers to an absolute clock.

Time proceeds in *discrete* steps represented by occurrences of a simple action tick, in the style of Hennessy and Regan's TPL [111], to denote idling until the next clock cycle. This is known as the *fictitious clock*. The calculus is value-passing, and message transmission requires a positive amount of time. The operational semantics is given in terms of a *labelled transition system* in the SOS style of Plotkin. We follow a *two-phase* approach [181] separating the execution of actions for synchronisation from the passage of time. Then we have

$$M \xrightarrow{\lambda} N$$

for synchronisation, where the label λ does not range over tick, and

$$M \xrightarrow{\mathsf{tick}} N$$

for the time passing. The meaning of these transitions is that the network M can execute the action λ and tick, respectively, and then evolve into the network N. As usual for ad hoc networks, the communication mechanism is *broadcast*.

As in Hennessy and Regan's TPL [111], and Prasad's timed CBS [195], our TCWS enjoys three basic time properties:

- *time determinism*: the passage of time is deterministic, i.e. a network can reach at most one new state by performing the action tick;
- *patience*: nodes will wait indefinitely until they can communicate;
- *maximal progress*: data transmissions cannot be delayed, they must occur as soon as a possibility for communication arises.

We provide a notion of network well-formedness to take into account node-uniqueness, network connectivity, transmission exposure, and transmission consistency. Then, we prove that our labelled transition semantics preserves network well-formedness.

As a case study we use our calculus to model the *Carrier Sense Multiple Access* (CSMA) scheme [123]. According to the CSMA scheme, stations transmit only

when the channel is sensed free. As we will show, this protocol allows to prevent certain form of collisions, although it suffers of two well-known problems: the *hidden terminal problem* and the *exposed terminal problem*.

A central concern in process calculi is to establish when two terms have the same *observable behaviour*, that is, they are indistinguishable in any context. *Behavioural equivalences* are fundamental for justifying program transformations. Our program equivalence is a timed variant of (weak) *reduction barbed congruence*, a branching-time contextually-defined program equivalence. Barbed equivalences [172] are simple and intuitive but difficult to use due to the quantification on all contexts. Simpler proof techniques are based on *labelled bisimilarities* [169], which are co-inductive relations that characterise the behaviour of processes using a labelled transition system. We define a *labelled bisimilarity* which is a proof method for our timed reduction barbed congruence. We then apply our bisimulation proof-technique to prove a number of algebraic properties.

We end this introduction with an outline of the chapter. In Section 6.2, we provide both syntax and operational semantics of our calculus. In Section 6.3 we propose a notion of network well-formedness to rule out inconsistent networks and we prove that network well-formedness is preserved at run time. In Section 6.4, we prove that TCWS enjoys the above cited time properties. In Section 6.5, we use our calculus to study the CSMA protocol. In Section 6.6, we equip TCWS with a notion of observational equivalence along the lines of Milner and Sangiorgi's barbed congruence. In Section 6.7, we propose a labelled bisimilarity as a proof method for our observations equivalence. More precisely, we prove that our bisimilarity is a congruence and that it implies our observational equivalence. In Section 6.8 we then use our bisimilarity to prove a number of algebraic properties. In Section 6.9 we present related work. Finally, in Section 6.10 we give a summary of the arguments discussed in this chapter.

6.2 The Calculus

6.2.1 Syntax

In Table 6.1, we define the syntax of TCWS in a two-level structure, a lower one for *processes* and a upper one for *networks*. We use letters $a, b, c, \ldots$ for logical names, x, y, z for *variables*, u for *values*, and v and w for *closed values*, i.e. values that do not contain free variables. Closed values actually denote messages that are transmitted as TCP/IP packets. Packets contain a number of auxiliary informations such as the network address of the transmitter. So, sometimes we write $m{:}v$ to mean a message v transmitted by node m. With an abuse of notation, structured messages of the form $m{:}v$ are ranged by the same letters v and w. We write $\tilde{u}$ to denote a tuple $u_1, \ldots, u_k$ of values.

Networks are collections of nodes (which represent devices) running in parallel and using a unique common channel to communicate with each other. We use the symbol $\mathbf{0}$ to denote the empty network, while $M_1 \mid M_2$ represents the parallel composition of two sub-networks M_1 and M_2. Nodes cannot be created or destroyed. All nodes have the same transmission range. We write $n[W]_t^{\nu}$ for a node

Table 6.1 The Syntax

Values			
u	$::=$	x	variable
	$\mid$	v	closed value
Networks			
M, N	$::=$	$\mathbf{0}$	empty network
	$\mid$	$M \mid N$	parallel composition
	$\mid$	$n[W]_t^{\nu}$	node
Processes			
W	$::=$	P	inactive process
	$\mid$	A	active process
P, Q	$::=$	nil	termination
	$\mid$	$!\langle u\rangle.P$	broadcast
	$\mid$	$?(x).P$	receiver
	$\mid$	$\mathsf{tick}.P$	delay
	$\mid$	$\lfloor ?(x).P \rfloor Q$	receiver with timeout
	$\mid$	$[u_1 = u_2]P, Q$	matching
	$\mid$	$H\langle \tilde{u}\rangle$	recursion
A	$::=$	$\langle v\rangle^t.P$	active sender
	$\mid$	$(x)_v.P$	active receiver

named n (the device network address) executing the sequential process W. The tag ν denotes the set of (the names of) the neighbours of n. Said in other words, ν contains all nodes in the transmission cell of n. In this manner, we model the network topology. Notice that the network topology could have been represented using some kind of restriction operator à la CCS over node names. We preferred our notation to keep at hand the neighbours of a node. The variable t is a semantic tag ranging over positive integers to represent *node exposure*. Thus, a node $n[W]_t^{\nu}$, with $t > 0$, is exposed to a transmission (or more transmissions) for the next t instants of time.

Processes are sequential and live within the nodes. For convenience, we distinguish between *non-active* and *active processes*. An active process is a process which is currently transmitting or receiving. An *active node* is a node with an active process inside. The symbol nil denotes the skip process. The sender process $!\langle v\rangle.P$ allows to broadcast the value v. Once the transmission starts the process evolves into the active sender process $\langle v\rangle^{\delta_v}.P$ which transmits the message v for the next δ_v time units, the time necessary to transmit v. In the construct $\langle v\rangle^t.P$ we require $t > 0$. The receiver process $?(x).P$ listens on the channel for incoming messages. Once the reception starts the process evolves into the active receiver process $(x)_w.P$ and starts receiving. Only when the channel becomes free the receiver calculates the CRC (Cyclic Redundancy Check) to check the integrity of the received packets. Upon successful reception the variable x of P is instantiated with the transmitted message w. The process $\mathsf{tick}.P$ models sleeping for one time unit. The process $\lfloor ?(x).P \rfloor Q$ denotes a receiver with timeout. This operator comes from

the process algebra ATP put forward in [181]. Intuitively, this process either starts receiving a value in the current instant of time, evolving into an active receiver, or it idles for one time unit, and then continues as Q. Process $[v_1 = v_2]P, Q$ is the standard "if then else" construct: it behaves as P if $v_1 = v_2$, and as Q otherwise. In processes $\mathsf{tick}.P, ?(x).P$, $\lfloor ?(x).P \rfloor Q$, and $!\langle v \rangle.P$ the occurrence of process P is said to be *guarded*. We write $H\langle \tilde{v} \rangle$ to denote a process defined via a definition $H(\tilde{x}) \stackrel{\text{def}}{=} P$, with $\mid \tilde{x} \mid = \mid \tilde{v} \mid$, where $\tilde{x}$ contains all variables that appear free in P. Defining equations provide *guarded recursion*, since P may contain only guarded occurrences of process identifiers, such as H itself. We use a number of notational conventions. $\prod_{i \in I} M_i$ means the parallel composition of all sub-networks M_i, for $i \in I$. We write $!\langle v \rangle$ for $!\langle v \rangle.\mathsf{nil}$, and $\langle v \rangle^\delta$ for $\langle v \rangle^\delta.\mathsf{nil}$. We recall that in the active sender process $\langle v \rangle^t.P$ it holds that $t > 0$. However, sometimes, for convenience, we write $\langle v \rangle^0.P$ assuming the following syntactic equality $\langle v \rangle^0.P = P$.

In the terms $?(x).P$, $\lfloor ?(x).P \rfloor Q$, and $(x)_v.P$ the variable x is bound in P. This gives rise to the standard notion of α-conversion. We identify processes and networks up to α-conversion. We assume there are no free variables in our networks. The absence of free variables in networks is trivially maintained as the network evolves. We write $\{^v/_x\}P$ for the substitution of the variable x with the value v in P. We define *structural congruence*, written $\equiv$, as the smallest congruence which is a commutative monoid with respect to the parallel operator.

Given a network M, $\mathsf{nds}(M)$ returns the names of the nodes which constitute the network M. For any network M, $\mathsf{actsnd}(M)$ and $\mathsf{actrcv}(M)$ return the set of active senders and active receivers of M, respectively. Thus, for instance, for $N = m[!\langle w \rangle]_t^\nu \mid n[\langle v \rangle^r.P]_{t'}^{\nu'}$ we have $\mathsf{nds}(N) = \{m, n\}$ and $\mathsf{actsnd}(N) = \{n\}$. Given a network M and an active sender $n \in \mathsf{actsnd}(M)$, the function $\mathsf{active}(n, M)$ says for how long the node will be transmitting. For instance, if N is the network defined as before, $\mathsf{active}(n, N) = r$. If n is not an active sender then $\mathsf{active}(n, N) = 0$. Finally, given a network M and a node $m \in \mathsf{nds}(M)$, the function $\mathsf{ngh}(m, M)$ returns the set of neighbours of m in M. Thus, for N defined as above $\mathsf{ngh}(m, N) = \nu$.

6.2.2 Operational Semantics

We give the operational semantics of our calculus in terms of a Labelled Transition System (LTS). Table 6.2 contains a simple LTS for processes. Rules (SndP) and (RcvP) model the beginning of a transmission. In rule (SndP) a sender evolves into an active sender. For convention we assume that the transmission of a value v takes δ_v time units. In rule (RcvP) a receiver evolves into an active receiver $(x)_{m:v}.P$ where m is the transmitter's name and v is the value that is supposed to be received after δ_v instants of time. The process $\lfloor ?(x).P \rfloor Q$ can start a reception in the current instant of time, as $?(x).P$, or it can idle for one time unit evolving into Q. Rules (RcvTO) and (Timeout) model these two different behaviours, respectively. The remaining rules regards time passing. Rules (Nil-tick), (Rcv-tick), and (Tick) are straightforward. In rule (ActSnd) the time necessary to conclude the transmission is decreased. In rule (ActRcv) the derivative does not change as a reception terminates only when the channel is sensed free. Notice that sender processes do not perform tick-actions. This is to model the maximal progress property.

Table 6.2 LTS - Process transitions

$$\text{(SndP)}\ \frac{-}{!\langle v\rangle.P \xrightarrow{!v} \langle v\rangle^{\delta_v}.P} \qquad \text{(RcvP)}\ \frac{-}{?(x).P \xrightarrow{?w} (x)_w.P}$$

$$\text{(RcvTO)}\ \frac{-}{\lfloor ?(x).P \rfloor Q \xrightarrow{?w} (x)_w.P} \qquad \text{(Timeout)}\ \frac{-}{\lfloor ?(x).P \rfloor Q \xrightarrow{\mathsf{tick}} Q}$$

$$\text{(Nil-tick)}\ \frac{-}{\mathsf{nil} \xrightarrow{\mathsf{tick}} \mathsf{nil}} \qquad \text{(Rcv-tick)}\ \frac{-}{?(x).P \xrightarrow{\mathsf{tick}} ?(x).P}$$

$$\text{(Tick)}\ \frac{-}{\mathsf{tick}.P \xrightarrow{\mathsf{tick}} P}$$

$$\text{(ActSnd)}\ \frac{r > 0}{\langle v\rangle^{r}.P \xrightarrow{\mathsf{tick}} \langle v\rangle^{r-1}.P} \qquad \text{(ActRcv)}\ \frac{-}{(x)_v.P \xrightarrow{\mathsf{tick}} (x)_v.P}$$

Table 6.3 LTS - Begin transmission

$$\text{(Snd)}\ \frac{P \xrightarrow{!v} A}{m[P]^{\nu}_{t} \xrightarrow{m!v} m[A]^{\nu}_{t}} \qquad \text{(Rcv)}\ \frac{m \in \nu \quad P \xrightarrow{?m:v} A}{n[P]^{\nu}_{0} \xrightarrow{m?v} n[A]^{\nu}_{\delta_v}}$$

$$\text{(RcvPar)}\ \frac{M \xrightarrow{m?v} M' \quad N \xrightarrow{m?v} N'}{M \mid N \xrightarrow{m?v} M' \mid N'} \qquad \text{(Sync)}\ \frac{M \xrightarrow{m!v} M' \quad N \xrightarrow{m?v} N'}{M \mid N \xrightarrow{m!v} M' \mid N'}$$

$$\text{(Coll)}\ \frac{m \in \nu \quad t' := \mathsf{max}(t, \delta_v)}{n[(x)_w.P]^{\nu}_{t} \xrightarrow{m?v} n[(x)_{\perp}.P]^{\nu}_{t'}} \qquad \text{(Exp)}\ \frac{m \in \nu \quad W \neq (x)_w.P \quad t' := \mathsf{max}(t, \delta_v)}{n[W]^{\nu}_{t} \xrightarrow{m?v} n[W]^{\nu}_{t'}}$$

$$\text{(OutRng)}\ \frac{m \notin \nu \quad m \neq n}{n[W]^{\nu}_{t} \xrightarrow{m?v} n[W]^{\nu}_{t}} \qquad \text{(Zero)}\ \frac{-}{\mathbf{0} \xrightarrow{m?v} \mathbf{0}}$$

We have divided the LTS for networks in two sets of rules corresponding to the two main aspects of a wireless transmission. Table 6.3 contains the rules to model the initial synchronisation between the sender and its neighbours. Table 6.4 contains the rules for modelling time passing and transmission ending.

Let us comment on the rules of Table 6.3. Rule (Snd) models a node starting a broadcast of message v to its neighbours in ν. By maximal progress, a node which is ready to transmit will not be delayed. Rule (Rcv) models the beginning of the reception of a message v transmitted by a station m; m is in the transmission range of the receiver. This happens only when the receiver is not exposed to other transmissions i.e. when the exposure indicator is equal to zero. The exposure indicator is then updated because node n will be exposed for the next δ_v instants

of time. The reception will finish only when the receiver senses the channel free (see rule (Time-0) of Table 6.4).

Rule (RcvPar) models multiple receptions. Rule (Sync) serves to synchronise the components of a network with a broadcast transmission originating from a node m. In rule (Coll) an active receiver n is exposed to a transmission originating from a node m. This transmission gives rise to a *collision* at n. We use the symbol $\perp$ to indicate that a collision has happened. In this case, we may need to update the exposure indicator of n to the maximum between the current value and the duration of the transmission causing the collision. Rule (Exp) models the exposure of a node n (which is not an active receiver) to a transmission originating from a transmitter m. In this case, n does not take part in the transmission. Notice that a node $n[?(x).P]_0^\nu$ might execute rule (Exp) instead of (Rcv). This is because a potential (synchronised) receiver might miss the synchronisation with the sender for several reasons (internal misbehaving, radio signals problems, etc). Such a situation will give rise to a failure in reception at n (see rule (RcvFail) in Table 6.4). Rule (OutRng) regards nodes which are out of the range of a transmission originating from a node m. Rule (Zero) is similar but regards empty networks. Rules (RcvPar) and (Sync) have their symmetric counterpart.

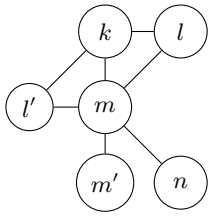
Let us explain the rules in Table 6.3 with an example.

Example 6.1. Consider the network

$$Net \stackrel{\text{def}}{=} k[!\langle v\rangle.?(x).P]_0^{\nu_k} \mid l[?(x).Q]_0^{\nu_l} \mid m[!\langle w\rangle]_0^{\nu_m} \mid n[?(y).R]_0^{\nu_n}$$

with the following communication topology: $\nu_k = \{l, m, l'\}$, $\nu_l = \{k, m\}$, $\nu_m = \{k, l, n, l', m'\}$, and $\nu_n = \{m\}$.

Figure 6.1 Network topology of Example 6.1



We draw the network topology of Net in Figure 6.1: circles represent nodes, whereas a connection line between two nodes represents that those nodes are neighbouring. There are two possible broadcast communications originating from stations k and m, respectively. Let us suppose k starts broadcasting. By applying rules (Snd), (Rcv), (Exp), (OutRng), (RcvPar), and (Sync) we have:

$$\begin{aligned} Net &\xrightarrow{k!v} k[\langle v\rangle^{\delta_v}.?(x).P]_0^{\nu_k} \mid l[(x)_{k:v}.Q]_{\delta_v}^{\nu_l} \mid m[!\langle w\rangle]_{\delta_v}^{\nu_m} \mid n[?(y).R]_0^{\nu_n} \\ &= \quad Net_1 \ . \end{aligned}$$

Now, by maximal progress, the station m must start transmitting at the same instant of time. Supposing $\delta_v < \delta_w$ we have:

Table 6.4 LTS - Time passing/End transmission

$$\text{(Time-0)}\ \frac{W \xrightarrow{\mathsf{tick}} W' \quad W \neq (x)_w.P}{\begin{array}{c} n[W]^\nu_0 \xrightarrow{\mathsf{tick}} n[W']^\nu_0 \\ n[(x)_w.P]^\nu_0 \xrightarrow{\mathsf{tick}} n[\{^w/_x\}P]^\nu_0 \end{array}}$$

$$\text{(Time-t)}\ \frac{t>0 \quad W \xrightarrow{\mathsf{tick}} W' \quad W \not\xrightarrow{?v}}{n[W]^\nu_t \xrightarrow{\mathsf{tick}} n[W']^\nu_{t-1}} \qquad \text{(RcvFail)}\ \frac{t>0 \quad P \xrightarrow{?\perp} A}{n[P]^\nu_t \xrightarrow{\mathsf{tick}} n[A]^\nu_{t-1}}$$

$$\text{(Zero-tick)}\ \frac{-}{\mathbf{0} \xrightarrow{\mathsf{tick}} \mathbf{0}} \qquad \text{(Par-tick)}\ \frac{M \xrightarrow{\mathsf{tick}} M' \quad N \xrightarrow{\mathsf{tick}} N'}{M \mid N \xrightarrow{\mathsf{tick}} M' \mid N'}$$

$$\begin{array}{rcl} Net_1 & \xrightarrow{m!w} & k[\langle v\rangle^{\delta_v}.?(x).P]^{\nu_k}_{\delta_w} \mid l[(x)_\perp.Q]^{\nu_l}_{\delta_w} \mid m[\langle w\rangle^{\delta_w}]^{\nu_m}_{\delta_v} \mid n[(y)_{m:w}.R]^{\nu_n}_{\delta_w} \\ & = & Net_2 \ . \end{array}$$

Now, node l is exposed to a collision and its reception is doomed to fail. Notice that, although node m was already exposed when it started transmitting, node n will receive correctly the message w from m.

Let us comment on rules of Table 6.4. Rules (Time-0) and (Time-t) model the passage of one time unit for non-exposed and exposed nodes, respectively. In rule (Time-t) the exposure indicator is decreased, whereas in rule (Time-0) this is not necessary as time indicator is equal to 0. Notice that for $W = !\langle v\rangle.P$ none of these two rules can be applied, as for maximal progress no transmission can be delayed. Notice also that, for $W = ?(x).P$ and $t > 0$, rule (Time-t) cannot be applied. In this case, we must apply rule (RcvFail) to model a failure in reception. This may happen, for instance, when the receiver misses the preamble starting a transmission, or when a receiver wakes up in the middle of an ongoing transmission. Rule (Zero-tick) is straightforward. Rule (Par-tick) models time synchronisation. This is possible because our networks are connected. Rule (Par-tick) has its symmetric counterpart.

Example 6.2. Let us continue with the previous example. Let us show how the system evolves after δ_v and δ_w time units. We write $\xrightarrow{\mathsf{tick}}{}^{\delta_v}$ to mean the passage of δ_v instants of time. We recall that $0 < \delta_v < \delta_w$. For simplicity let us define $\delta := \delta_w - \delta_v$:

$$\begin{array}{rcl} Net_2 & \xrightarrow{\mathsf{tick}}{}^{\delta_v} & k[?(x).P]^{\nu_k}_{\delta} \mid l[(x)_\perp.Q]^{\nu_l}_{\delta} \mid m[\langle w\rangle^{\delta}]^{\nu_m}_0 \mid n[(y)_{m:w}.R]^{\nu_n}_{\delta} \\ & \xrightarrow{\mathsf{tick}} & k[(x)_\perp.P]^{\nu_k}_{\delta-1} \mid l[(x)_\perp.Q]^{\nu_l}_{\delta-1} \mid m[\langle w\rangle^{\delta-1}]^{\nu_m}_0 \mid n[(y)_{m:w}.R]^{\nu_n}_{\delta-1} \\ & \xrightarrow{\mathsf{tick}}{}^{\delta-1} & k[(x)_\perp.P]^{\nu_k}_0 \mid l[(x)_\perp.Q]^{\nu_l}_0 \mid m[\mathsf{nil}]^{\nu_m}_0 \mid n[(y)_{m:w}.R]^{\nu_n}_0 \\ & \xrightarrow{\mathsf{tick}} & k[\{^\perp/_x\}P]^{\nu_k}_0 \mid l[\{^\perp/_x\}Q]^{\nu_l}_0 \mid m[\mathsf{nil}]^{\nu_m}_0 \mid n[\{^{m:w}/_y\}R]^{\nu_n}_0 \end{array}$$

Table 6.5 LTS - Matching and Recursion

$$\text{(Then)}\ \frac{n[P]_t^\nu \xrightarrow{\lambda} n[P']_{t'}^\nu}{n[[v=v]P,Q]_t^\nu \xrightarrow{\lambda} n[P']_{t'}^\nu} \qquad \text{(Else)}\ \frac{n[Q]_t^\nu \xrightarrow{\lambda} n[Q']_{t'}^\nu \quad v_1 \neq v_2}{n[[v_1=v_2]P,Q]_t^\nu \xrightarrow{\lambda} n[Q']_{t'}^\nu}$$

$$\text{(Rec)}\ \frac{n[\{\tilde{v}/\tilde{x}\}P]_t^\nu \xrightarrow{\lambda} n[P']_{t'}^\nu \quad H(\tilde{x}) \stackrel{\text{def}}{=} P}{n[H\langle\tilde{v}\rangle]_t^\nu \xrightarrow{\lambda} n[P']_{t'}^\nu}$$

Notice that, after δ_v instants of time, node k will start a reception in the middle of an ongoing transmission (the transmitter being m). This will lead to a failure at k.

In the rest of the chapter, the metavariable λ will range over the following labels: $m!v$, $m?v$, and tick. In Table 6.5 we report the obvious rules for nodes containing matching and recursive processes (we recall that only guarded recursion is allowed).

6.3 Well-Formedness

The syntax presented in Table 6.1 allows to derive inconsistent networks, i.e. networks that do not have a realistic counterpart. Below we give a number of definitions to rule out ill-formed networks.

As network addresses are unique, we assume that there cannot be two nodes with the same name in the same network.

Definition 6.3 (Node uniqueness). *A network M is said to be* node-unique *if whenever $M \equiv M_1 \mid m[W_1]_t^\nu \mid n[W_2]_{t'}^{\nu'}$ it holds that $m \neq n$.*

We also assume network connectivity, i.e. all nodes are connected to each other, although not always directly. We recall that all nodes have the same transmission range. Formally,

Definition 6.4 (Network connectivity). *A network M is said to be* connected *if*

- *whenever $M \equiv N \mid m[W_1]_t^\nu \mid n[W_2]_{t'}^{\nu'}$ with $m \in \nu'$ it holds that $n \in \nu$;*
- *for all $m, n \in \mathsf{nds}(M)$ there is a sequence of nodes $m_1, \ldots, m_k \in \mathsf{nds}(M)$, with neighbouring $\nu_1, \ldots, \nu_k$, respectively, such that $m{=}m_1$, $n{=}m_k$, and $m_i \in \nu_{i+1}$, for $1 \leq i \leq k{-}1$.*

The next definition is about the consistency of exposure indicators of nodes. Intuitively, the exposure indicators of active senders and active receivers must be consistent with their current activity (transmission/reception). Moreover, the neighbours of active senders must have their exposure indicators consistent with the duration of the transmission.

Definition 6.5 (Exposure consistency). *A network M is said to be* exposure-consistent *if the following conditions are satisfied.*

1. *If $M \equiv N \mid m[(x)_v.P]^{\nu}_t$, with $v \neq \bot$, then $0 \leq t \leq \delta_v$.*
2. *If $M \equiv N \mid m[\langle v \rangle^r.P]^{\nu}_t$, then $r \leq \delta_v$.*
3. *If $M \equiv N \mid m[\langle v \rangle^r.P]^{\nu}_t \mid n[W]^{\nu'}_{t'}$, with $m \in \nu'$, then $0 < r \leq t'$.*
4. *Let $M \equiv N \mid n[W]^{\nu}_t$ with $t{>}0$. If $\mathsf{active}(k, N) \neq t$ for all k in $\nu \cap \mathsf{actsnd}(N)$, then there is k' in $\nu \backslash \mathsf{nds}(N)$ such that whenever $N \equiv N' \mid l[W']^{\nu'}_{t'}$, with $k' \in \nu'$, then $t' \geq t$.*

The next definition is about the consistency of transmitting stations. The first and the second part are about successful transmissions, while the third part is about collisions.

Definition 6.6 (Transmission consistency). *A network M is said to be* transmission-consistent *if the following conditions are satisfied.*

1. *If $M \equiv N \mid n[(x)_v.Q]^{\nu}_t$ and $v \neq \bot$, then $\mid \mathsf{actsnd}(N) \cap \nu \mid \leq 1$.*
2. *If $M \equiv N \mid m[\langle w \rangle^r.P]^{\nu}_t \mid n[(x)_v.Q]^{\nu'}_{t'}$, with $m \in \nu'$ and $v \neq \bot$, then (i) $v = m{:}w$, and (ii) $r = t'$.*
3. *If $M \equiv N \mid n[(x)_v.P]^{\nu}_t$, with $\mid \mathsf{actsnd}(N) \cap \nu \mid > 1$, then $v = \bot$.*

Definition 6.7 (Well-formedness). *A network M is said to be* well-formed *if it is node-unique, connected, exposure-consistent, and transmission-consistent.*

In the sequel, we will work only with well-formed networks. Finally, we prove that network well-formedness is preserved at runtime. In particular, the preservation of exposure- and transmission-consistency are the more interesting and delicate results.

Proposition 6.8. *Let M be a node-unique network. If $M \xrightarrow{\lambda} M'$ then M' is node-unique.*

Proof By transition induction. The result easily follows by the fact that no inference rule creates new nodes. □

Proposition 6.9. *Let M be a connected network. If $M \xrightarrow{\lambda} M'$ then M' is connected.*

Proof By transition induction. The result easily follows by the fact that no inference rule changes the network topology. □

Proposition 6.10 (Exposure consistency). *Let M be an exposure consistent network. If $M \xrightarrow{\lambda} M'$ then M' is exposure consistent.*

Proof The proof proceeds by transition induction on the derivation of $M \xrightarrow{\lambda} M'$, for $\lambda \in \{m!v, m?v, \mathsf{tick}\}$. We show the most significant cases, derived by an application of rules (Sync), (RcvPar), and (Par-tick). The other cases are straightforward. Here we show just a few cases. The full proof can be found in Section A.1 at page 168.

- Let $M \xrightarrow{m!v} M'$ by an application of rule (Sync) with $M = M_1 \mid M_2$, $M_1 \xrightarrow{m!v} M_1'$ and $M_2 \xrightarrow{m?v} M_2'$, and $M' = M_1' \mid M_2'$, where M_1' and M_2' are exposure consistent by inductive hypothesis. We have to prove that M' respects the clauses of Definition 6.5.
  - Clauses 1-2. In these cases the result follows directly by inductive hypothesis.
  - Clause 3. Let $M' \equiv \prod_i n_i[W_i']_{t_i'}^{\nu_i} \mid h[\langle v\rangle^r.P]_{t_h'}^{\nu_h} \mid n[W']_{t_n'}^{\nu_n}$, with $h \in \nu_n$. We have to prove that $r \leq t_n'$. We only consider the case when $h \in \mathsf{nds}(M_1)$ and $n \in \mathsf{nds}(M_2)$ (or viceversa). The other cases are easier and follow by inductive hypothesis. There are two possibilities.
    - $h \neq m$. By Lemma A.1(7) at page 167 we have
    $$M \equiv \prod_i n_i[W_i]_{t_i}^{\nu_i} \mid h[\langle v\rangle^r.P]_{t_h}^{\nu_h} \mid n[W]_{t_n}^{\nu_n}$$
    for appropriate processes and tags. Now, if $m \in \nu_n$ by Lemma A.1(4) we have $t_n' = \max(t_n, \delta_v)$. As M is exposure consistent it holds that $r \leq t_n$ and hence also $r \leq t_n'$. On the other hand, if $m \notin \nu_n$ by an application of Lemma A.1(3) we have $t_n' = t_n$. As M is exposure consistent it follows that $r \leq t_n = t_n'$.
    - $h = m$. By Lemma A.1(2) it follows that
    $$M \equiv \prod_i n_i[W_i]_{t_i}^{\nu_i} \mid h[!\langle v\rangle.P]_{t_h}^{\nu_h} \mid n[W]_{t_n}^{\nu_n}$$
    for appropriate processes and tags, with $r = \delta_v$. Since $h \in \nu_n$, by Lemma A.1(4) we have $t_n' = \max(t_n, \delta_v)$. As a consequence, $r \leq t_n'$.
  - Clause 4. Let
  $$M \equiv N \mid n[W]_t^{\nu} = \prod_i n_i[W_i]_{t_i}^{\nu_i} \mid n[W]_t^{\nu}$$
  and
  $$M' \equiv N' \mid n[W']_{t'}^{\nu} = \prod_i n_i[W_i']_{t_i'}^{\nu_i} \mid n[W']_{t'}^{\nu}$$
  with $t' > 0$ and $\mathsf{active}(k', N') \neq t'$ for all $k' \in \nu \cap \mathsf{actsnd}(N')$. We have to prove that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N')$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i' \geq t'$. We can distinguish two cases:
    - If $m \notin \nu$ by Lemma A.1(3) we have $t' = t$. By Lemma A.1(6), it follows that $\mathsf{actsnd}(N) \subseteq \mathsf{actsnd}(N')$. As a consequence, $\nu \cap \mathsf{actsnd}(N) \subseteq \nu \cap \mathsf{actsnd}(N')$. Since $t' = t$ we can derive that for all $k \in \nu \cap \mathsf{actsnd}(N)$ it holds that $\mathsf{active}(k, N') \neq t$. By Lemma A.1(6) and Lemma A.1(7) if $k \neq m$ then $\mathsf{active}(k, N) = \mathsf{active}(k, N')$. Since $m \notin \nu$ it follows that for all $k \in \nu \cap \mathsf{actsnd}(N)$ it holds $\mathsf{active}(k, N) \neq t$. Since M is exposure consistent it follows that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N)$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i \geq t$. Notice that $\nu \setminus \mathsf{nds}(N) = \nu \setminus \mathsf{nds}(N')$. Moreover, by Lemma A.1(3) and A.1(4) we have $t_i \leq t_i'$, for all i. This allows us to derive that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N')$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i' \geq t_i \geq t = t'$.

· If $m \in \nu$ then by Lemma A.1(4) we have $t' = \max(t, \delta_v)$. By definition of neighbouring of a node $m \in \nu$ implies $m \neq n$. By Lemma A.1(2) it follows that $m \notin \mathsf{actsnd}(N)$, $m \in \mathsf{actsnd}(N')$, and $\mathsf{active}(m, N') = \delta_v$. Since $\mathsf{active}(k', N') \neq t'$ for all $k' \in \nu \cap \mathsf{actsnd}(N')$, and $m \in \nu \cap \mathsf{actsnd}(N')$, it follows that $t' \neq \delta_v$. Since $t' = \max(t, \delta_v)$, it follows that $t' = t$. By Lemma A.1(6), it follows that $\mathsf{actsnd}(N) \subseteq \mathsf{actsnd}(N')$. As a consequence, $\nu \cap \mathsf{actsnd}(N) \subseteq \nu \cap \mathsf{actsnd}(N')$. Since $t' = t$ we can derive that for all $k \in \nu \cap \mathsf{actsnd}(N)$ it holds that $\mathsf{active}(k, N') \neq t$. By Lemma A.1(6) and Lemma A.1(7) if $k \neq m$ then $\mathsf{active}(k, N) = \mathsf{active}(k, N')$. Since $m \notin \mathsf{actsnd}(N)$ it follows that for all $k \in \nu \cap \mathsf{actsnd}(N)$ it holds $\mathsf{active}(k, N) \neq t$. Since M is exposure consistent it follows that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N)$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i \geq t$. Notice that $\nu \setminus \mathsf{nds}(N) = \nu \setminus \mathsf{nds}(N')$. Moreover, by Lemmas A.1(3) and A.1(4) we have $t_i \leq t'_i$, for all i. This allows us to derive that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N')$ such that if $\hat{k} \in \nu_i$, for some i, then $t'_i \geq t_i \geq t = t'$.

□

Let us prove now that our LTS preserves transmission consistency.

Proposition 6.11 (Transmission consistency). *Let M be both an exposure consistent and a transmission consistent network. If $M \xrightarrow{\lambda} M'$ then M' is transmission consistent.*

Proof The proof proceeds by transition induction on the derivation of $M \xrightarrow{\lambda} M'$, for $\lambda \in \{m!v, m?v, \mathsf{tick}\}$. We show the most significant cases, derived by an application of rules (Sync), (RcvPar), and (Par-tick). The other cases are straightforward. We show just a few cases. The full proof can be found in Section A.1 at page 170.

- Let $M \xrightarrow{m!v} M'$ by an application of rule (Sync), with $M = M_1 \mid M_2$, $M_1 \xrightarrow{m!v} M'_1$ and $M_2 \xrightarrow{m?v} M'_2$, and $M' = M'_1 \mid M'_2$, where M'_1 and M'_2 are transmission consistent by inductive hypothesis. We have to prove that M' respects the clauses of Definition 6.6. Let examine the three clauses one by one.
  - Clause 1. Let

$$M' \equiv N' \mid n[(x)_w.Q]^{\nu_n}_{t'_n} = \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid n[(x)_w.Q]^{\nu_n}_{t'_n}$$

    with $w \neq \bot$. We have to prove that $\mid \mathsf{actsnd}(N') \cap \nu \mid \leq 1$. By Lemma A.1(2) at page 167 we have

$$M' \equiv N' \mid n[(x)_w.Q]^{\nu_n}_{t'_n} \equiv \prod_j n_j[W'_j]^{\nu_j}_{t'_j} \mid m[\langle v \rangle^{\delta_v}.P]^{\nu_m}_{t'_m} \mid n[(x)_w.Q]^{\nu_n}_{t'_n}$$

    and

$$M \equiv N \mid n[W]^{\nu_n}_{t'_n} = \prod_j n_j[W_j]^{\nu_j}_{t_j} \mid m[!\langle v \rangle.P]^{\nu_m}_{t_m} \mid n[W]^{\nu_n}_{t_n}$$

    for appropriate processes and tags.
    There are two possibilities.

- If $m \notin \nu_n$ then by Lemma A.1(3) we have $W = (x)_w.Q$. By Lemmas A.1(6) and A.1(7) we have $\mathsf{actsnd}(N') = \mathsf{actsnd}(N) \cup \{m\}$. Since M is transmission consistent, we have $\mid \mathsf{actsnd}(N) \cap \nu_n \mid \leq 1$. Since $m \notin \nu_n$ it follows that $\mid \mathsf{actsnd}(N') \cap \nu_n \mid \leq 1$.
- If $m \in \nu$ then by Lemma A.1(5) we have $W = ?(x).Q$ (the case $W = \lfloor ?(x).P \rfloor Q$ is similar) and $t_n = 0$. By Lemmas A.1(6) and A.1(7) we have $\mathsf{actsnd}(N') = \mathsf{actsnd}(N) \cup \{m\}$. Since $t_n = 0$, $m \in \nu_n$, and M is exposure consistent, clause 3 of Definition 6.5 allows to derive that $\mathsf{actsnd}(N') \cap \nu_n = \{m\}$. Hence, $\mid \mathsf{actsnd}(N') \cap \nu_n \mid \leq 1$.

– Clause 2. Let

$$M' \equiv \prod_i n_i[W_i']^{\nu_i}_{t_i'} \mid h[\langle w_1 \rangle^r.P]^{\nu_h}_{t_h'} \mid n[(x)_{w_2}.Q]^{\nu_n}_{t_n'}$$

with $h \in \nu_n$ and $w_2 \neq \bot$. We have to show that $w_2 = m{:}w_1$ and $r = t_n'$. There are two cases.

1. Suppose $h \neq m$. In this case, by Lemma A.1(2) at page 167 we have the following situation:

$$M' \equiv \prod_j n_j[W_j']^{\nu_j}_{t_j'} \mid m[\langle v \rangle^{\delta_v}.R]^{\nu_m}_{t_m'} \mid h[\langle w_1 \rangle^r.P]^{\nu_h}_{t_h'} \mid n[(x)_{w_2}.Q]^{\nu_n}_{t_n'}$$

and

$$M \equiv \prod_j n_j[W_j]^{\nu_j}_{t_j} \mid m[!\langle v \rangle.R]^{\nu_m}_{t_m} \mid h[\langle w_1 \rangle^r.P]^{\nu_h}_{t_h} \mid n[W]^{\nu_n}_{t_n}$$

for appropriate processes and tags.
Now, there are two sub-cases.
   a) If $m \notin \nu_n$ then by Lemma A.1(3) we have $W = (x)_{w_2}.Q$ and $t_n' = t_n$. Since M is transmission consistent we derive $w_2 = m{:}w_1$ and $r = t_n'$.
   b) If $m \in \nu_n$ then by Lemma A.1(5) we have $W = ?(x).Q$ (the case $W = \lfloor ?(x).P \rfloor Q$ is similar) and $t_n = 0$. However, since M is exposure consistent by clause 3 of Definition 6.5 it must be $t_n > 0$. So, this case is not possible.
2. Suppose $h = m$. This case easily follows by an application of Lemma A.1(2) and Lemma A.1(5).

– Clause 3. Let

$$M' \equiv N' \mid n[(x)_w.P]^{\nu_n}_{t_n'} = \prod_i n_i[W_i']^{\nu_i}_{t_i'} \mid n[(x)_w.P]^{\nu_n}_{t_n'}$$

with $\mid \mathsf{actsnd}(N') \cap \nu_n \mid > 1$. We want to show that $w = \bot$. By an application of Lemma A.1(2) at page 167 it holds that

$$M' \equiv \prod_j n_j[W_j']^{\nu_j}_{t_j'} \mid m[\langle v \rangle^{\delta_v}.Q]^{\nu_m}_{t_m'} \mid n[(x)_w.P]^{\nu_n}_{t_n'}$$

and

$$M \equiv \prod_j n_j[W_j]^{\nu_j}_{t_j} \mid m[!\langle v \rangle.Q]^{\nu_m}_{t_m} \mid n[W]^{\nu_n}_{t_n}$$

for appropriate processes and tags. Since $\mid \mathsf{actsnd}(N') \cap \nu_n \mid > 1$, it must be $W'_j = \langle w_j \rangle^r.P_j$, for some j. By Lemma A.1(6) we derive that $W_j = W'_j$. At this point we reason by contradiction. Suppose $w \neq \bot$. Then, by Lemma A.1(5) we have $W = ?(x).P$ (the case $W = \lfloor ?(x).P \rfloor Q$ is similar) and $t_n = 0$. However, since M is exposure consistent, by clause 3 of Definition 6.5 it must be $t_n > 0$. This contradiction allows us to conclude that $w = \bot$. □

Theorem 6.12 (Subject reduction). *If M is a well-formed network, and $M \xrightarrow{\lambda} M'$ for some label λ and network M', then M' is well-formed as well.*

Proof The proof follows by an application of Propositions 6.8, 6.9, 6.10, and 6.11. □

6.4 Time Properties

We now prove that TCWS enjoys three desirable time properties: (i) *time determinism*, (ii) *patience*, and (iii) *maximal progress*.

Theorem 6.13 formalises the determinism nature of time passing: a network can reach at most one new state by executing the action tick.

Theorem 6.13 (Time Determinism). *Let M be a well-formed network. If $M \xrightarrow{\mathsf{tick}} M'$ and $M \xrightarrow{\mathsf{tick}} M''$ then M' and M'' are syntactically the same.*

Proof By induction on the length of the proof of $M \xrightarrow{\mathsf{tick}} M'$. The base cases are when the transition is derived by the application of one of the following rules: (Time-0), (Time-t), (RcvFail), and (Zero-tick). It is straightforward to prove that the statement holds for these rules. As to the inductive case, let $M \xrightarrow{\mathsf{tick}} M'$ by an application of rule (Par-tick). This implies that $M = M_1 \mid M_2$, for some M_1 and M_2, with $M_1 \xrightarrow{\mathsf{tick}} M'_1$, $M_2 \xrightarrow{\mathsf{tick}} M'_2$ and $M' = M'_1 \mid M'_2$. As $M = M_1 \mid M_2$, the transition $M \xrightarrow{\mathsf{tick}} M''$ can be derived only by applying rule (Par-tick) where $M_1 \xrightarrow{\mathsf{tick}} M''_1$, $M_2 \xrightarrow{\mathsf{tick}} M''_2$ and $M'' = M''_1 \mid M''_2$. By inductive hypothesis it holds that M'_i and M''_i are syntactically the same, for $i \in \{1, 2\}$. This implies that M' and M'' are syntactically the same. □

In [111, 195], the maximal progress property says that processes communicate as soon as a possibility of communication arises. However, unlike [111, 195], in our calculus message transmission requires a positive amount of time. So, we generalise the property saying that transmissions cannot be delayed.

Theorem 6.14 (Maximal Progress). *Let M be a well-formed network. If there is N such that $M \xrightarrow{m!v} N$ then $M \xrightarrow{\mathsf{tick}} M'$ for no network M'.*

Proof By induction on the structure of M. In $M = \mathbf{0}$ the statement does not apply. If M is composed by only one node and $M \xrightarrow{m!v} N$, this can be derived only by an application of rule (Snd) with $M = m[!\langle v \rangle.P]^\nu_t$ and $N = m[\langle v \rangle^{\delta_v}.P]^\nu_t$. Because sender nodes cannot perform tick-actions, there is no network M' such

that $M \xrightarrow{\mathsf{tick}} M'$. Let M be composed by at least two nodes. If $M \xrightarrow{m!v} N$ by an application of rule (Sync) then $M = M_1 \mid M_2$ for some M_1 and M_2, with $M_1 \xrightarrow{m!v} M_1'$, $M_2 \xrightarrow{m?v} M_2'$ and $N = M_1' \mid M_2'$ (the converse is similar). In this case the only rule for deriving a tick-transition from M is (Par-tick). However, the inductive hypothesis guarantees that $M_1 \xrightarrow{\mathsf{tick}} \widehat{M}$ for no network $\widehat{M}$; then $M \xrightarrow{\mathsf{tick}} M'$ for no network M'. □

The last time property is patience. In [111, 195] patience guarantees that a process will wait indefinitely until it can communicate. In our setting, this means that if no transmission can start then it must be possible to execute a tick-action to let time pass.

Theorem 6.15 (Patience). *Let M be a well-formed network. If $M \xrightarrow{m!v} M'$ for no network M' then there is a network N such that $M \xrightarrow{\mathsf{tick}} N$.*

Proof By contradiction and then by induction on the structure of M. We prove that if $M \xrightarrow{\mathsf{tick}} N$ for no network N then there is a network M' such that $M \xrightarrow{m!v} M'$. Let us proceed by induction on the structure of M. We show just a few cases. The full proof can be found in Section A.1 at page 173.

- Let $M = \mathbf{0}$. Then $M \xrightarrow{\mathsf{tick}} M$ by an application of rule (Zero-tick). So, the statement does not apply.
- Let $M = n[W]_t^\nu$. We proceed by induction on the structure of P.
  - If $W = \mathsf{nil}$ and $t = 0$ then $M \xrightarrow{\mathsf{tick}} n[P]_0^\nu$ by an application of rules (Nil-tick and) and (Time-0). Thus the statement does not apply.
  - If $W = !\langle v\rangle.P$ then $M \not\xrightarrow{\mathsf{tick}}$. However, $M \xrightarrow{m!v} m[\langle v\rangle^{\delta_v}.P]_t^\nu$, by an application of rule (Snd), in contradiction with the hypothesis.
  - If $W = ?(x).P$ and $t = 0$ then $M \xrightarrow{\mathsf{tick}} n[W]_0^\nu$, by an application of rules (Rcv-tick) and (Time-0). Thus the statement does not apply.
  - If $W = \mathsf{tick}.P$ and $t = 0$ then $M \xrightarrow{\mathsf{tick}} n[W]_0^\nu$ by an application of rules (Sigma) and (Time-0). Thus the statement does not apply.
  - If $W = \lfloor ?(x).P \rfloor Q$ and $t = 0$ then $M \xrightarrow{\mathsf{tick}} n[Q]_{t\ominus 1}^\nu$, by an application of rules (Timeout) and (Time-0). Thus the statement does not apply.
  - If $W = \langle v\rangle^r.P$, with $r > 1$, and $t = 0$ then by an application of rules (ActSnd) and (Time-0) we have $M \xrightarrow{\mathsf{tick}} n[\langle v\rangle^{r-1}.P]_0^\nu$ and the statement does not apply.
  - If $W = (x)_v.P$, with $t > 0$, then by an application of rules (ActRcv) and (Time-t) we have $M \xrightarrow{\mathsf{tick}} n[(x)_v.P]_{t\ominus 1}^\nu$ and the statement does not apply.
- Let $M = M_1 \mid M_2$. A transition of the form $M \xrightarrow{\mathsf{tick}} M'$ can be derived only by an application of rule (Par-tick). Thus if M cannot perform a tick-action then at least one of the premises of rule (Par-tick) does not hold:
  - If $M_1 \xrightarrow{\mathsf{tick}} M_1'$ for no network M_1', then by inductive hypothesis we have $M_1 \xrightarrow{m!v} M_1'$, for some M_1'. As $M = M_1 \mid M_2$ is a well-formed network,

by Lemma A.4 at page 173 it holds that $M \xrightarrow{m!v} M_1' \mid M_2'$, for some M_2', in contradiction with the hypothesis.

– If $M_2 \xrightarrow{\mathsf{tick}} M_2'$ for no network M_2', then we can reason as in the previous sub-case.

□

6.5 A Case Study: the Carrier Sense Multiple Access Scheme

The *Carrier Sense Multiple Access* (CSMA) scheme [123] is a widely used MAC level protocol in which a device senses the channel before transmitting. More precisely, if the channel is sensed free, the sender starts transmitting immediately, that is in the next instant of time (we recall that in wireless systems channels are half-duplex); if the channel is busy, that is some other station is transmitting, the device keeps listening the channel until it becomes idle and then starts transmitting immediately. This strategy is called *1-persistent* CSMA. More generally, in a *p-persistent* CSMA strategy, where p is a probability, the sender transmits with probability p, and waits for the next available time slot, with probability $1 - p$.

In our calculus, we can easily model the 1-persistent CSMA scheme using receivers with timeout where the sender process $!\langle v\rangle.P$ is replaced by the process defined below:

$$!!\langle v\rangle.P \stackrel{\text{def}}{=} \lfloor ?(x).!\langle v\rangle.P \rfloor !\langle v\rangle.P \ .$$

The next example shows how 1-persistent CSMA affects the behaviour of a wireless system. Let us consider the network:

$$Net \stackrel{\text{def}}{=} k[!!\langle v\rangle.?(x).P]_0^{\nu_k} \mid l[?(x).Q]_0^{\nu_l} \mid m[\mathsf{tick}.!!\langle w\rangle]_0^{\nu_m} \mid n[?(y).R]_0^{\nu_n}$$

with the following communication topology: $\nu_k = \{l, m, l'\}$, $\nu_l = \{k, m\}$, $\nu_m = \{k, l, n, l', m'\}$, and $\nu_n = \{m\}$, as in Figure 6.1. Here, node k senses the channel free and, according to the CSMA scheme, in the next instant of time it will start transmitting. Thus,

$$\begin{aligned} Net &\xrightarrow{\mathsf{tick}} k[!\langle v\rangle.?(x).P]_0^{\nu_k} \mid l[?(x).Q]_0^{\nu_l} \mid m[!!\langle w\rangle]_0^{\nu_m} \mid n[?(y).R]_0^{\nu_n} \\ &= \quad Net_1 \ . \end{aligned}$$

In Net_1, node m it is currently listening the channel to check whether it is free. By applying rules (Snd), (Rcv), (Exp), (OutRng), (RcvPar), and (Sync) node k can start transmitting:

$$\begin{aligned} Net_1 &\xrightarrow{k!v} k[\langle v\rangle^{\delta_v}.?(x).P]_0^{\nu_k} \mid l[(x)_{k:v}.Q]_{\delta_v}^{\nu_l} \mid m[(x)_{k:v}.!\langle w\rangle]_{\delta_v}^{\nu_m} \mid n[?(y).R]_0^{\nu_n} \\ &= \quad Net_2 \ . \end{aligned}$$

Now, since k has already started its transmission, node m senses the channel busy and it must wait until the channel becomes free. Notice that in this manner there are no collisions at l and/or k. In fact, after δ_v instants of time we have:

$$\begin{aligned} Net_2 &\xrightarrow{\mathsf{tick}}{}^{\delta_v} k[?(x).P]_0^{\nu_k} \mid l[(x)_{k:v}.Q]_0^{\nu_l} \mid m[(x)_{k:v}.!\langle w\rangle]_0^{\nu_m} \mid n[?(y).R]_0^{\nu_n} \\ &\xrightarrow{\mathsf{tick}} k[?(x).P]_0^{\nu_k} \mid l[\{^{k:v}/_x\}Q]_0^{\nu_l} \mid m[!\langle w\rangle]_0^{\nu_m} \mid n[?(y).R]_0^{\nu_n} \\ &= Net_3 \end{aligned}$$

where node l has successfully received value v from k. Notice that after δ_v instants of time node m senses the channel free, and by maximal progress it will start transmitting in the next instant of time.

Notice that, using a CSMA scheme, there is always a chance of stations transmitting at the same time, caused by the fact that different stations sensed the medium free and decided to transmit at once. As an example, consider the network:

$$Net' \stackrel{\text{def}}{=} k[!!\langle v\rangle.?(x).P]_0^{\nu_k} \mid l[?(x).Q]_0^{\nu_l} \mid m[!!\langle w\rangle]_0^{\nu_m} \mid n[?(y).R]_0^{\nu_n}$$

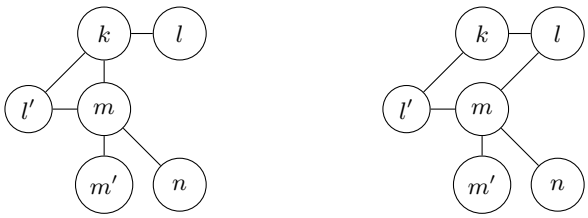
with the same communication topology as before. In this scenario, both nodes k and m want to start transmitting. And since both of them sense the channel free, they will start transmitting in the next instant of time. Thus, assuming $\delta_v < \delta_w$, we have:

$$\begin{aligned} Net' &\xrightarrow{\mathsf{tick}} k[!\langle v\rangle.?(x).P]_0^{\nu_k} \mid l[?(x).Q]_0^{\nu_l} \mid m[!\langle w\rangle]_0^{\nu_m} \mid n[?(y).R]_0^{\nu_n} \\ &\xrightarrow{k!v} k[\langle v\rangle^{\delta_v}.?(x).P]_0^{\nu_k} \mid l[(x)_{k:v}.Q]_{\delta_v}^{\nu_l} \mid m[!\langle w\rangle]_{\delta_v}^{\nu_m} \mid n[?(y).R]_0^{\nu_n} \\ &\xrightarrow{m!w} k[\langle v\rangle^{\delta_v}.?(x).P]_{\delta_w}^{\nu_k} \mid l[(x)_{\perp}.Q]_{\delta_w}^{\nu_l} \mid m[\langle w\rangle^{\delta_w}]_{\delta_v}^{\nu_m} \mid n[(y)_{m:w}.R]_{\delta_w}^{\nu_n} \ . \end{aligned}$$

In this situation, node l is exposed to a collision caused by the two transmissions.

Notice that, the CSMA scheme is not always a good idea. Let us consider, for instance, the previous network Net where nodes l and m are not neighbours anymore, that is $\nu_l = \{k\}$ and $\nu_m = \{k, n, l', m'\}$. The network topology is drawn in the picture on the left of Figure 6.2. Now, suppose that m wants to send a message to n.

Figure 6.2 Network topologies for CSMA examples



Then, the CSMA scheme delays the transmission without any reason, only because m is exposed to the transmission originating from k. This is a well-known problem, not prevented by CSMA, called *exposed terminal problem*.

The CSMA scheme suffers from another well-known problem called *hidden terminal problem*. This happens when two transmitters sense the channel free, because they are not in each other's transmission cell, and start transmitting causing a collision to a third node lying in the transmission cells of both. As an example, you can consider, for instance, the previous network Net with the following communication topology: $\nu_k = \{l, l'\}$, $\nu_l = \{k, m\}$, $\nu_m = \{l, n, l', m'\}$, and $\nu_n = \{m\}$. The network topology is drawn in the picture on the right of Figure 6.2. In this case, both transmissions at k and m will fire causing (after two instants of time) an interference at l.

In unicast communications, to reduce the number of collisions due to the hidden terminal problem, the CSMA scheme may be used together with a CA (Collision Avoidance) protocol. In this protocol, before transmitting the message, two special packets (RTS/CTS) are sent to reserve the channel [123]. On one hand this technique reduce the number of collisions, on the other hand the transmission of extra data may drastically reduce the performances of the network. Moreover, the CSMA/CA protocol does not help for broadcast communications, as it is designed only for unicast transmissions.

6.6 Behavioural Semantics

In this section we propose a notion of timed behavioural equivalence for our wireless networks. Our starting point is Milner and Sangiorgi's barbed congruence [172], a standard contextually-defined program equivalence. Intuitively, two terms are barbed congruent if they have the same *observables*, in all possible contexts, under all possible *evolutions*. The definition of barbed congruence strongly relies on two crucial concepts: a reduction semantics to describe how a system evolves, and a notion of observable which says what the environment can observe in a system.

From the operational semantics given in Section 6.2.2 it should be clear that a wireless network evolves transmitting messages. Notice that a transmission in a network does not require any synchronisation with the environment. Thus, we can define the reduction relation $\twoheadrightarrow$ between networks using the following inference rule

$$\text{(Red)}\ \frac{M \xrightarrow{m!v} N}{M \twoheadrightarrow N}\ .$$

We write $\twoheadrightarrow^*$ for the reflexive and transitive closure of $\twoheadrightarrow$.

Now, let us focus on the definition of an appropriate notion of observable. In our calculus, as in CCS [169] and in π-calculus [171], we have both transmission and reception of messages. However, in broadcast calculi only the transmission of messages may be observed [137,163]. In fact, an observer cannot detect whether a given node actually receives a broadcast value. In particular, if the node $m[!\langle v\rangle.P]^{\nu}_{t}$ evolves into $m[\langle v\rangle^{r}.P]^{\nu}_{t}$ we do not know whether some of the neighbours have actually synchronised for receiving the message v. On the other hand, if a *non-exposed* node $n[?(x).P]^{\nu}_{0}$ evolves into $n[(x)_{m:v}.P]^{\nu}_{t}$, then we can be sure that some node in ν has started transmitting. Notice that a node n can certify the reception of a message v from a transmitter m only if it receives the whole message without collisions.

Following Milner and Sangiorgi [172] we use the term "barb" as synonymous of observable.

Definition 6.16 (Barbs). *Let M be a well-formed network. We write $M \downarrow_n$, if $M \equiv N \mid m[\langle v \rangle^r.P]_t^\nu$, for some m, v, r, P, t and ν, such that $n \in \nu$ and $n \notin \mathsf{nds}(N)$. We write $M \Downarrow_n$ if there is M' such that $M \rightarrow^* M' \downarrow_n$.*

The barb $M \Downarrow_n$ says that there is a transmission at M reaching the node n of the environment. The observer can easily detect such a transmission placing a receiver with timeout at n. Say, something like $n[\lfloor ?(x).\mathbf{0} \rfloor ! \langle w \rangle.\mathbf{0}]_t^{\nu'}$, where $M \mid n[\lfloor ?(x).\mathbf{0} \rfloor ! \langle w \rangle.\mathbf{0}]_t^{\nu'}$ is well-formed, and $f \in \nu'$, for some fresh name f. In this manner, if n is currently exposed to a transmission then, after a tick-action, the fresh barb at f is definitely lost. One may wonder whether the barb should mention the name m of the transmitter, which is usually recorded in some specific field of the packets. Notice that, in general, due to communication collisions, the observer may receive incomprehensible packets without being able to identify the transmitter. In fact, if $M \downarrow_n$ there might be several nodes in M which are currently transmitting to n. So, in our setting, it does not make sense to put the name of the transmitter in the barb.

Now, everything is in place to define our timed notion of reduction barbed congruence. In the sequel, we write $\mathcal{R}$ to denote binary relations over well-formed networks.

The first property that our timed notion of reduction barbed congruence has to satisfy is the preservation of the barb.

Definition 6.17 (Barb preserving). *A relation $\mathcal{R}$ is said to be* barb preserving *if whenever $M \mathcal{R} N$ it holds that $M \downarrow_n$ implies $N \Downarrow_n$.*

As we are interested in weak behavioural equivalences, the definition of reduction closure is given in terms of weak reductions.

Definition 6.18 (Reduction closure). *A relation $\mathcal{R}$ is said to be* reduction-closed *if $M \mathcal{R} N$ and $M \rightarrow M'$ imply there is N' such that $N \rightarrow^* N'$ and $M' \mathcal{R} N'$.*

When comparing two networks M and N, time must pass in the same manner for M and N.

Definition 6.19 (tick-closure). *A relation $\mathcal{R}$ is said to be* tick-closed *if $M \mathcal{R} N$ and $M \xrightarrow{\mathsf{tick}} M'$ imply there is a network N' such that $N \rightarrow^* \xrightarrow{\mathsf{tick}} \rightarrow^* N'$ and $M' \mathcal{R} N'$.*

Our relation is preserved by the parallel operator.

Definition 6.20 (Contextuality). *A relation $\mathcal{R}$ is said* contextual *if $M \mathcal{R} N$, for M and N well-formed, implies $M \mid O \mathcal{R} N \mid O$ for all networks O such that $M \mid O$ and $N \mid O$ are well-formed.*

Finally, everything is in place to define timed reduction barbed congruence.

Definition 6.21 (Timed reduction barbed congruence). Timed reduction barbed congruence*, written $\cong$, is the largest symmetric relation over well-formed networks which is barb preserving, reduction-closed,* tick*-closed, and contextual.*

6.7 A Bisimulation Proof Method

The definition of timed reduction barbed congruence is simple and intuitive. However, due to the universal quantification on parallel contexts, it may be quite difficult to prove that two terms are barbed congruent. Simpler proof techniques are based on labelled bisimilarities. In this section, we propose an appropriate notion of bisimulation between networks. As a main result, we prove that our labelled bisimilarity is a proof-technique for timed reduction barbed congruence.

First of all we have to distinguish between transmissions which may be observed and transmissions which may not be observed.

$$\text{(Shh)}\ \frac{M \xrightarrow{m!v} N \quad \mathsf{ngh}(m,M)\subseteq\mathsf{nds}(M)}{M \xrightarrow{\tau} N} \qquad \text{(Obs)}\ \frac{M \xrightarrow{m!v} N \quad \nu:=\mathsf{ngh}(m,M)\setminus\mathsf{nds}(M)\neq\emptyset}{M \xrightarrow{m!v\blacktriangleright\nu} N}$$

Rule (Shh) models transmissions that cannot be detected by the environment. This happens if none of the potential receivers is in the environment. Notice that, although there is no explicit rule, τ-actions propagate through parallel composition.

Lemma 6.22. *If $M \xrightarrow{\tau} M'$ then $M \mid N \xrightarrow{\tau} M' \mid N$ and $N \mid M \xrightarrow{\tau} N \mid M'$.*

Rule (Obs) models a transmission of a message that may be potentially received by the nodes ν of the environment. Notice that this transmission can be really observed at some node $n \in \nu$ only if no collisions arise at n during the transmission of v.

In the sequel, we use the metavariable α to range over the following actions: τ, tick, $m?v$, and $m!v\blacktriangleright\nu$. Since we are interested in *weak behavioural equivalences*, that abstract over τ-actions, we introduce a standard notion of weak action: $\Rightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}$; $\stackrel{\alpha}{\Longrightarrow}$ denotes $\Rightarrow\xrightarrow{\alpha}\Rightarrow$; $\stackrel{\hat{\alpha}}{\Longrightarrow}$ denotes $\Rightarrow$ if $\alpha=\tau$ and $\stackrel{\alpha}{\Longrightarrow}$ otherwise.

Definition 6.23 (Bisimilarity). *A relation $\mathcal{R}$ over well-formed networks is a* simulation *if $M\,\mathcal{R}\,N$ implies that*

- $\mathsf{nds}(M)=\mathsf{nds}(N)$
- *whenever $M \xrightarrow{\alpha} M'$ there is N' such that $N \stackrel{\hat{\alpha}}{\Longrightarrow} N'$ and $M'\,\mathcal{R}\,N'$.*

A relation $\mathcal{R}$ is called bisimulation *if both $\mathcal{R}$ and its converse are simulations. We say that M and N are* bisimilar*, written $M \approx N$, if there is some bisimulation $\mathcal{R}$ such that $M\,\mathcal{R}\,N$.*

The requirement that two bisimilar networks must have the same nodes is quite reasonable. Technically, this is necessary to prove that the bisimilarity is a congruence.

In order to prove that our labelled bisimilarity implies timed reduction barbed congruence we have to show its contextuality.

Theorem 6.24 ($\approx$ is contextual). *Let M and N be two well-formed networks such that $M \approx N$. Then $M \mid O \approx N \mid O$ for all networks O such that $M \mid O$ and $N \mid O$ are well-formed.*

Proof We prove that the relation

$$\mathcal{S} \stackrel{\text{def}}{=} \{(M \mid O\,,\, N \mid O) : M \approx N,\ \ M \mid O \text{ and } N \mid O \text{ well-formed } \}$$

is a bisimulation by a case analysis. We show details for one case. The full proof can be found in Section A.1 at page 174.

- Since $M \approx N$ by definition of bisimilarity it follows $\mathsf{nds}(M) = \mathsf{nds}(N)$. This implies $\mathsf{nds}(M \mid O) = \mathsf{nds}(N \mid O)$.
- Let $M \mid O \xrightarrow{m!v\blacktriangleright\nu} \widehat{M}$, by an application of rule (Obs) because $M \mid O \xrightarrow{m!v} \widehat{M}$, with $\nu = \mathsf{ngh}(m, M \mid O) \setminus \mathsf{nds}(M \mid O) \neq \emptyset$.
  There are two possible cases:
  - $M \mid O \xrightarrow{m!v} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!v} M'$ and $O \xrightarrow{m?v} O'$, with $\widehat{M} = M' \mid O'$. Since $M \xrightarrow{m!v} M'$ it follows that $m \in \mathsf{nds}(M)$ and hence $\nu = (\mathsf{ngh}(m, M) \setminus \mathsf{nds}(M)) \setminus \mathsf{nds}(O)$. Let $\nu' := \mathsf{ngh}(m, M) \setminus \mathsf{nds}(M)$, since $\nu \neq \emptyset$ it follows that $\nu' \neq \emptyset$. By an application of rule (Obs) we have $M \xrightarrow{m!v\blacktriangleright\nu'} M'$. Now, since $M \approx N$ there is N' such that $N \stackrel{m!v\blacktriangleright\nu'}{\Longrightarrow} N'$ with $M' \approx N'$ and $\nu' = \mathsf{ngh}(m, N) \setminus \mathsf{nds}(N) \neq \emptyset$. Since the action $m!v\blacktriangleright\nu'$ can be generated only by an application of rule (Obs), there are N_1 and N_2 such that

    $$N \xrightarrow{\tau}{}^{*} N_1 \xrightarrow{m!v} N_2 \xrightarrow{\tau}{}^{*} N' \ .$$

    Since $O \xrightarrow{m?v} O'$, by several applications of Lemma 6.22 and one application of rule (Sync) we have:

    $$N \mid O \xrightarrow{\tau}{}^{*} N_1 \mid O \xrightarrow{m!v} N_2 \mid O' \xrightarrow{\tau}{}^{*} N' \mid O' \ .$$

    As

    $$\begin{aligned} \nu &= \mathsf{ngh}(m, M \mid O) \setminus \mathsf{nds}(M \mid O) \\ &= (\mathsf{ngh}(m, M) \setminus \mathsf{nds}(M)) \setminus \mathsf{nds}(O) \\ &= (\mathsf{ngh}(m, N) \setminus \mathsf{nds}(N)) \setminus \mathsf{nds}(O) \\ &= \mathsf{ngh}(m, N \mid O) \setminus \mathsf{nds}(N \mid O) \\ &\neq \emptyset \ . \end{aligned}$$

    by an application of rule (Obs) we can derive

    $$N \mid O \xrightarrow{\tau}{}^{*} N_1 \mid O \xrightarrow{m!v\blacktriangleright\nu} N_2 \mid O' \xrightarrow{\tau}{}^{*} N' \mid O' \ .$$

    By Theorem 6.12, both $M' \mid O'$ and $N' \mid O'$ are well-formed. As $M' \approx N'$ it follows that $(M' \mid O'\,,\, N' \mid O') \in \mathcal{S}$.
  - $M \mid O \xrightarrow{m!v} \widehat{M}$, by an application of rule (Sync), because $M \xrightarrow{m?v} M'$ and $O \xrightarrow{m!v} O'$, with $\widehat{M} = M' \mid O'$. Since $O \xrightarrow{m!v} O'$, it follows that $m \in \mathsf{nds}(O)$ and hence $\nu = (\mathsf{ngh}(m, O) \setminus \mathsf{nds}(O)) \setminus \mathsf{nds}(M)$. Since $M \approx N$ there is N' such that $N \stackrel{m?v}{\Longrightarrow} N'$ with $M' \approx N'$. By several applications of Lemma 6.22 and one application of rule (Sync) it follows that:

    $$N \mid O \xrightarrow{\tau}{}^{*} N_1 \mid O \xrightarrow{m!v} N_2 \mid O' \xrightarrow{\tau}{}^{*} N' \mid O' \ .$$

By definition of bisimilarity, $M \approx N$ implies $\mathsf{nds}(M) = \mathsf{nds}(N)$. Moreover, since $M \mid O$ and $N \mid O$ are well-formed and $m \in \mathsf{nds}(O)$, by node uniqueness it follows that $m \notin \mathsf{nds}(M)$ and $m \notin \mathsf{nds}(N)$. Thus,

$$\begin{aligned}\mathsf{ngh}(m, N \mid O) \setminus \mathsf{nds}(N \mid O) &= (\mathsf{ngh}(m, O) \setminus \mathsf{nds}(O)) \setminus \mathsf{nds}(N) \\ &= (\mathsf{ngh}(m, O) \setminus \mathsf{nds}(O)) \setminus \mathsf{nds}(M) \\ &= \mathsf{ngh}(m, M \mid O) \setminus \mathsf{nds}(M \mid O) \\ &= \nu \\ &\neq \emptyset \ .\end{aligned}$$

With this premise, by an application of rule (Obs) we can derive

$$N \mid O \xrightarrow{\tau}{}^{*} N_1 \mid O \xrightarrow{m!v\blacktriangleright\nu} N_2 \mid O' \xrightarrow{\tau}{}^{*} N' \mid O' \ .$$

By Theorem 6.12, both $M' \mid O'$ and $N' \mid O'$ are well-formed. As $M' \approx N'$ it follows that $\big(M' \mid O' \, , \, N' \mid O'\big) \in \mathcal{S}$.

□

In Theorem 6.25 we state our soundness result for which our bisimilarity implies our reduction barbed congruence.

Theorem 6.25 (Soundness). *Let* ngh *be a neighbouring function and* M *and* N *two well-formed networks wrt* ngh *such that* $M \approx N$*. Then* $M \cong N$*.*

Proof We have to prove that the labelled bisimilarity is contextual, barb preserving, reduction- and tick-closed. Contextuality follows from Theorem 6.24. Reduction and tick-closure follow by definition. As to barb preservation we reason by contradiction. If $M \downarrow_n$, we can choose as a context to observe this barb the network $O \stackrel{\text{def}}{=} n[\lfloor ?(x).\mathbf{0} \rfloor ! \langle w \rangle.\mathbf{0}]_t^{\nu}$ such that $M \mid O$ and $N \mid O$ are well-formed, and $f \in \nu$, for some fresh name f. Since $M \downarrow_n$ the network $M \mid O$ will never (even in the future) perform an output action $n!w\blacktriangleright\nu$. Let us assume that $N \not\Downarrow_n$. We would have $N \mid O \stackrel{\mathsf{tick}}{\Longrightarrow} \stackrel{n!w\blacktriangleright\nu}{\Longrightarrow}$. However, this is in contradiction with the hypothesis, as by Theorem 6.24 we have $M \mid O \approx N \mid O$. So, it must be $N \Downarrow_n$. □

6.8 Algebraic Laws

In Theorem 6.26, we report a number of algebraic properties on well-formed networks that can be proved using our bisimulation proof-technique. We briefly explain them:

- Laws 1 and 2: they show different but equivalent nodes that do not interact with the rest of the network;
- Law 3: it is about exposed and sleeping nodes;
- Law 4: it is about successful reception. Here, node n will receive correctly the value v because all its neighbours will not interfere during the current transmission;
- Laws 5 and 6: they are about collisions. In both cases the transmission at m will cause a collision at n;

- Law 7: it tells about the blindness of receivers exposed to collisions. In particular, if all neighbours of a transmitter are exposed, then the content of the transmission is irrelevant as all recipients will fail. Only the duration of the transmission may be important as the exposure indicators of the neighbours may change.

Theorem 6.26.

1. $n[\mathsf{nil}]^{\nu}_{t} \approx n[Sleep]^{\nu}_{t}$, *where* $Sleep \stackrel{\text{def}}{=} \mathsf{tick}.Sleep$.
2. $n[\mathsf{nil}]^{\nu}_{t} \approx n[P]^{\nu}_{t}$, *if* P *does not contain sender processes.*
3. $n[\mathsf{tick}^{r}.P]^{\nu}_{s} \approx n[\mathsf{tick}^{r}.P]^{\nu}_{t}$ *if* $s \leq r$ *and* $t \leq r$.
4. $m[\langle v\rangle^{r}.P]^{\nu}_{t} \mid n[(x)_{m:v}.Q]^{\nu'}_{r} \mid M \approx m[\langle v\rangle^{r}.P]^{\nu}_{t} \mid n[\mathsf{tick}^{r}.\{^{m:v}/_{x}\}Q]^{\nu'}_{r} \mid M$, *if* $m \in \nu'$ *and for all* $k \in \nu' \setminus m$ *it holds that* $M \equiv k[\mathsf{tick}^{s}.R]^{\nu_k}_{t_k} \mid M'$, *with* $s \geq r$.
5. $m[!\langle v\rangle.P]^{\nu}_{s} \mid n[(x)_{w}.Q]^{\nu'}_{t} \approx m[!\langle v\rangle.P]^{\nu}_{s} \mid n[(x)_{\perp}.Q]^{\nu'}_{t}$, *if* $m \in \nu'$.
6. $m[\langle v_1\rangle^{r}.!\langle v_2\rangle.P]^{\nu}_{s} \mid n[(x)_{w}.Q]^{\nu'}_{t} \approx m[\langle v_1\rangle^{r}.!\langle v_2\rangle.P]^{\nu}_{s} \mid n[(x)_{\perp}.Q]^{\nu'}_{t}$, *if* $m \in \nu'$.
7. $m[!\langle v\rangle.P]^{\nu}_{t} \mid N \approx m[!\langle w\rangle.P]^{\nu}_{t} \mid N$, *if* $\delta_v = \delta_w$, *and for all* $n \in \nu$ *it holds that* $N \equiv n[W]^{\nu'}_{t'} \mid N'$, *with* $t' > 0$.

Proof By exhibiting the appropriate bisimulations. Let us prove, for instance, Law 5. The other proofs can be found in Section A.1 at page 176. For convenience, let us define:

- $A \stackrel{\text{def}}{=} m[!\langle v\rangle.P]^{\nu}_{s} \mid n[(x)_{w}.Q]^{\nu'}_{t}$
- $B \stackrel{\text{def}}{=} m[!\langle v\rangle.P]^{\nu}_{s} \mid n[(x)_{\perp}.Q]^{\nu'}_{t}$.

Let

$$\mathcal{S} \stackrel{\text{def}}{=} \{(A, B) \mid \text{ for all } s \text{ and } t\} \cup Id$$

where Id is the identity relation over network terms. We prove that $\mathcal{S}$ is a bisimulation. We proceed by case analysis on the possible transitions of A. Notice that by maximal progress, no tick-actions may be performed.

- If $A \xrightarrow{h?v'} A'$. The most interesting case is when $h \in \nu \cap \nu'$. In this case, by an application of rules (Coll), (Exp), and (RcvPar) we have $A' = m[!\langle v\rangle.P]^{\nu}_{s'} \mid n[(x)_{\perp}.Q]^{\nu'}_{t'}$, where $t' = \max(t, \delta_{v'})$ and $s' = \max(s, \delta_{v'})$. Similarly, we have $B \xrightarrow{h?v'} A'$ and we are done.
- If $A \xrightarrow{m!v \blacktriangleright \hat{\nu}} A'$, with $\hat{\nu} = \nu \setminus \{n\} \neq \emptyset$, then since $m \in \nu'$, by an application of rules (Snd), (Coll), (Sync), and (Obs) it follows that $A' = m[\langle v\rangle^{\delta_v}.P]^{\nu}_{s} \mid n[(x)_{\perp}.Q]^{\nu'}_{t'}$ with $t' = \max(t, \delta_v)$. Similarly, we have $B \xrightarrow{m!v \blacktriangleright \hat{\nu}} A'$ and we are done.
- If $A \xrightarrow{\tau} A'$, because $A \xrightarrow{m!v} A'$ and $\nu = \{n\}$. This case is similar to the previous one.

□

6.9 Related Work

In this section, we describe some related work.

Process Calculi without Time

We described in detail all the following related calculi but CBS$^\sharp$ [178] in Section 4.4, whereas we described CBS$^\sharp$ in Section 5.3. Here we remember some of their features and we give a comparison of them with TCWS under different aspects.

Nanz and Hankin [178] have introduced a calculus for mobile wireless networks (CBS$^\sharp$), relying on graph representation of node localities. The main goal of the paper is to present a framework for specification and security analysis of communication protocols for mobile wireless networks. Merro [163] has proposed a process calculus for mobile ad hoc networks with a labelled characterisation of reduction barbed congruence. Godskesen [94] has proposed a calculus for mobile ad hoc networks (CMAN). The paper proves a characterisation of reduction barbed congruence in terms of a contextual bisimulation. It also contains a formalisation of an attack on the cryptographic routing protocol ARAN. Singh, Ramakrishnan, and Smolka [214] have proposed the ω-calculus, a conservative extension of the π-calculus. A key feature of the ω-calculus is the separation of a node's communication and computational behaviour from the description of its physical transmission range. The authors have provided a labelled transition semantics and a bisimulation in "open" style. The ω-calculus is then used for modelling a leader election protocol and the AODV routing protocol for MANETs. Ghassemi, Fokkink, and Movaghar [91] have proposed a process algebra for mobile ad hoc networks (RBPT) where, topology changes are implicitly modelled in the (operational) semantics rather than in the syntax. The authors have proposed a notion of bisimulation for networks parameterised on a set of topology invariants that must be respected by equivalent networks. This work is then refined in [92] where the authors have proposed an equational theory for an extension of RBPT. Mezzetti and Sangiorgi [167] have instead proposed the CWS calculus, a lower level calculus to describe interferences in wireless systems. In their LTS there is a separation between transmission beginning and transmission ending. In CWS, CMAN, CMN and in ω-calculus the topology is defined as a part of the syntax, while the semantics of CBS$^\sharp$ is quantified over a set of node configurations. In the former (except CWS, where the topology is considered static), A process evolves syntactically (to reflect topology changes) by the application of mobility rules defined in the semantics, while in the latter, the underlying configuration changes arbitrary in the semantics.

All the previous calculi but CWS abstract from the presence of interferences and moreover are developed for mobile networks. In CWS a node is specified by $n[P]^c_{l,r}$ denoting process P, synchronised on channel c, deployed at a node with logical address n, physical location l and transmission range r. The topology of a network is derived by a function d, defining which nodes are located within transmission range of each other. Our work was definitely inspired by [167] for the choice of the representation of network topology and the representation of collisions, even if with some differences. In TCWS we have $n[\![W]\!]^\nu_t$ for a device with network address n, executing the sequential process W with the set of neighbours

ν. Then we do not need to explicit the physical location of a node as we do not use a distance function to calculate the set of neighbours of a node, because we write them in the syntax with the tag ν. Moreover we consider a unique channel. Finally, in CWS in order to identify whether and where two transmissions interfere with each other, a transmission is not atomic, but is modelled by its two boundary events: *begin-transmission* and *end-transmission*. Physically, these events represent a modification of the antenna communication mode. In order to study communication interferences, we consider time-consuming transmissions and semantic tags representing exposure and transmission durations are updated according to.

Process Calculi with Time

None of the calculi mentioned above deals with time, although there is an extensive literature on timed process algebra, as we stated in Section 4.3. From a purely syntactic point of view, the earliest proposals are extensions of the three main process algebras, ACP, CSP and CCS. For example, [18] presents a real-time extension of ACP, [199] contains a denotational model for a timed extension of CSP, while CCS is the starting point for [175]. In [18] and [199] time is real-valued, and at least semantically, associated directly with actions. The other major approach to representing time is to introduce special actions to model the passage of time, which the current calculus shares with [19,103,175,181] and [233,234], although the basis for all those proposals may be found in [35]. TWCS shares many of the assumptions of the languages presented in these papers. For example, all the papers above assume that actions are instantaneous and only the extension of ACP presented in [103] does not incorporate time determinism; however maximal progress is less popular and patience is even rarer.

More recent works on timed process algebra include the following papers. Aceto and Hennessy [8] have presented a simple process algebra where time emerges in the definition of a *timed observational equivalence*, assuming that beginning and termination of actions are distinct events which can be observed. Hennessy and Regan [111] have proposed the timed calculus called TPL, as timed version of CCS, enjoying time determinism, maximal progress, and patience. Our action tick takes inspiration from theirs. The authors have developed a semantic theory based on testing and characterised in terms of a particular kind of ready traces. Prasad [195] has proposed a timed variant of his CBS [194], called TCBS. In TCBS a time out can force a process wishing to speak to remain idle for a specific interval of time; this corresponds to have a priority. TCBS also assumes time determinism and maximal progress. We described both TPL and TCBS in detail in Section 4.3. Corradini et al. [67] have dealt with *durational actions* proposing a framework relying on the notions of reduction and observability to naturally incorporate timing information in terms of process interaction. Corradini and Pistore [68] have studied durational actions to describe and reason about the performance of systems. Actions have lower and upper time bounds, specifying their possible different durations. Their *time equivalence* refines the untimed one. As we described, also Baeten and Middelburg [21] have proposed several timed process algebras treated in a common framework, and related by embeddings and conservative extensions relations. In these process algebras the focus is on unsuccessful termination or deadlock. In [22] Baeten and Reniers extend the framework of [21] to model suc-

cessful termination for the relative-time case. Laneve and Zavattaro [144] have proposed a timed extension of π-calculus where time proceeds asynchronously at the network level, while it is constrained by the local urgency at the process level. They have proposed a timed bisimilarity whose discriminating is weaker when local urgency is dropped.

Our time model is inspired by TPL of [111] for the following properties: discrete time, separation in the LTS between synchronisation and time passing, time determinism, patience and maximal progress. On the contrary of TPL our actions are not instantaneous and in this case we took inspiration by [67] and [68]. Also our definition of timed reduction barbed congruence takes inspiration from [67].

6.10 Chapter Summary

We proposed a timed process calculus for wireless systems, called TCWS, exposed to communication collisions. In our model, time and collisions are treated in a completely *orthogonal* way. The operational semantics of our calculus is given in terms of a labelled transition system. We provided a notion of network well-formedness and we proved that it is preserved at run-time. We proved that the calculus enjoys a number of desirable time properties. As a case study we modelled the Carrier Sense Multiple Access (CSMA) scheme. The main behavioural equivalence of our calculus is a timed variant of reduction barbed congruence and, as efficient proof method for it, we defined a labelled bisimilarity. We then applied our bisimulation proof-technique to prove a number of algebraic properties.

For simplicity, in TCWS we relied on a static network topology. As a consequence, our results mainly apply to *stationary networks*. Notice that movement is not relevant in important classes of wireless systems, most notably *sensor networks* (not all sensor networks are stationary, but the stationary case is predominant).

In the next two chapters we propose two calculi for MANETs. In CTAN in Chapter 7 we embody a behavioural trust model and we adopt a mobility model following the approach of [92]. In TCTAN in Chapter 8 we extend CTAN with a simple notion of time.

7

A Calculus of Trustworthy Ad Hoc Networks

7.1 Introduction

Ad hoc networking is a new area in wireless communications that is attracting the attention of many researchers for its potential to provide ubiquitous connectivity without the assistance of any fixed infrastructure. In describing mobile ad hoc networks (MANETs) in Section 2.2.2 we said that they are self-configuring networks of mobile devices, also called nodes, communicating with each other via radio transceivers. The communication is usually broadcast. Ad hoc networks may operate in a standalone fashion, or may be connected to the larger Internet. They can be used wherever a wired backbone is infeasible and/or economically inconvenient, for example, to provide communications during emergencies, special events (expos, concerts, etc.), or in hostile environments. Lack of a fixed infrastructure, node mobility, shared wireless medium, cooperative behaviour, and physical vulnerability are some of the features that make challenging the design of a *security scheme* for mobile ad hoc networks.

In this scenario, *trust management* and *trustworthy computing* are becoming increasingly significant as they assist the systems in making sensible interactions with unknown parties by providing a basis for more detailed and automated decisions [203]. With the rapid development of network technologies and applications, the existing network architecture exposes serious insufficiencies, and various network attacks have kept emerging, such as malicious attacks, junk mails and computer virus etc. All this makes the current networks vulnerable and untrustworthy. So network security has to face big challenges.

The *Trusted Computing Group* (TCG) [105] has been developed to enhance endpoint trustworthiness and standardise open specifications for high trusted computing, but does not provide the network trustworthiness. The trustworthiness of network is an urgent problem, and it is the inevitable trend in the development of high trusted computing. According to Feng, Lin and Li [78, 185], in trustworthy networks security and survivability must be provided on network services, controllability must be provided on network architecture, user's behaviours should be monitored and some abnormal behaviours should be handled.

In brief, trustworthiness is reflected in three ways:

- *Trustworthiness of network services*: traditional security mechanism can resolve the trustworthiness of service providers' identities, but cannot provide the trustworthiness of their behaviours. In trustworthy network, service providers should be honest and provide secure services, cannot propagate malicious programs and leak private information of users. Quality of Service (QoS) and security control mechanisms are important technologies to realize the trustworthiness of network services;
- *Trustworthiness of information transmission*: it means that information cannot be deleted, tampered or damaged during transmission. Confidentiality, integrity and availability of information should be ensured, which can be realized by security protocols, encryptions and signatures;
- *Trustworthiness of users*: it includes the trustworthiness of users identity and behaviours, which can be realized by rigorous authentication and correct authorisation.

In this chapter we focus on *trustworthiness of users identity* and *behaviours*. It is very important for trustworthy networks and it can be realized by access control technology. As we described in Section 3.2, access control systems typically authenticate principals and then solicit access to resources. Access control is essentially addressed to limit the privileges of users to access system resources, and users can only do some operations which have been authorised to him. Correct authorisation can ensure the trustworthiness of user's behaviours. The trustworthiness of users is evaluated according to their behaviours. So system attributes of users not only includes their identity, but also their trust value, considered into the access control in trustworthy network. Moreover, it is required the real-time detection and auditing on system states and behaviours of users, which can reduce the security risks to a great extent before network attacks emerge.

In Section 3.2 we described some traditional access control mechanisms; they rely on the definition of specific permissions, called access policies, which are recorded in some data structure such as Access Control Lists (ACLs). ACLs work well when access policies are set in a centralised manners. However, they are less suited to ubiquitous systems where the number of users may be very large and/or continuously changing. In this scenario users may be potentially unknown and, therefore, untrusted.

In order to overcome these limitations, Blaze, Feigenbaum and Lacy [46] have introduced the notion of *Decentralised Trust Management* as an attempt to define a coherent framework in which safety critical decisions are based on trust policies relying on partial knowledge. Trust formalisation is the subject of several academic works. Among them we remember [37,88,102,132,155]). In Section 3.3 we analysed the concept of trust and reputation and we also described the trust management approach. In particular, we argued that stable hierarchies of trust relations cannot be supported in MANETs as evidence may be uncertain and incomplete, and only sporadically collected and exchanged. However, in behaviour-based trust models each node may perform trust evaluation based on continuous monitoring of misbehaviours of neighbours (*direct trust*). Misbehaviours typically include dropping, modification, and misrouting of packets at network layer. Trust evaluation may also depend on node reputation relying on cooperation with other nodes (*indirect trust*).

In section 2.4.1 we showed that there are various threats to ad hoc networks, of which the most interesting and important is *node subversion*. In this kind of attack, a node may be reverse-engineered, and replaced by a malicious node. A *bad* (or *compromised*) node can communicate with any other node, good or bad. Bad nodes may have access to the keys of all other bad nodes, whom they can impersonate if they wish. They do not execute the authorised software and thus do not necessarily follow protocols to identify misbehaviour, revoke other bad nodes, vote honestly or delete keys shared with revoked nodes. So, trust frameworks for ad hoc networks should support *node revocation* to isolate malicious nodes.

In Section 2.5 we discussed another key feature of MANETs: *node mobility*. Devices move while remaining connected to the network, breaking links with old neighbours and establishing fresh links with new devices. This makes security even more challenging as the compromise of a legitimate node or the insertion of a malicious node may go unnoticed in such a dynamic environment. Thus, a mobile node should acquire trust information on new neighbours, and remove trust information on old neighbours that cannot be monitored anymore.

In this paper, we propose a *process calculus* for *mobile ad hoc networks* which embodies a *behaviour-based multilevel trust model*. We call this *Calculus of Trustworthy Ad hoc Networks CTAN*.

Our trust model is decentralised and supports both *direct trust*, by monitoring nodes behaviour, and *indirect trust*, by collecting recommendations and spreading reputations. No information on past behaviours of nodes is recorded. Our model takes into consideration challenges introduced by mobility. For instance, we model loss of trust information. This could happen when a node moves, changing its neighbourhood. In this case, trust information concerning old neighbours must be deleted as they cannot be directly verified. We model our networks as *multilevel systems* [28] where each device is associated to a security level depending on its behaviour. Thus, trust relations associate security levels to nodes. We described many process calculi recently used to model different aspects of wireless systems [92, 94, 95, 163, 167, 178, 214]. However, none of these papers address the notion of trust. In our calculus, each node is equipped with a local trust store containing a set of assertions. These assertions supply trust information about the other nodes, according to a local security policy. Our calculus is not directly concerned with cryptographic underpinnings. However, we assume the presence of a hierarchical key generation and distribution protocol [122, 211]. Thus, messages are transmitted at a certain security level relying on an appropriate set of cryptographic keys. As usual, the operational semantics of the calculus is given in terms of a labelled transition system, where actions are executed at a certain security level. Then the transitions are of the form

$$M \xrightarrow{\lambda}_{\rho} N$$

indicating that the network M can perform the action λ, at security level ρ, evolving into the network N. For simplicity, our operational semantics does not directly express node mobility. However, we will show how to adapt the approach proposed in [92] to annotate our labelled transitions with the necessary information to represent node mobility.

Our calculus guarantees that only authorised nodes receive sensible information. The *safety up to a security level* property says that a synchronisation at a certain security level ρ is safe if the involved parties trust each other at that security level. Our networks enjoy two desirable security properties: *safety preservation* and *safety despite compromise*. The first property ensures that under all possible evolutions, no invalid synchronisation arises. The second property says that bad nodes, once detected, may not interact with good nodes. In this manner, bad nodes (recognised as such) are isolated from the rest of the network.

In CTAN, our program equivalence is a security variant of weak reduction barbed congruence. Along the lines of [70], we define a labelled bisimilarity parameterised on security levels. Intuitively, two networks are δ-bisimilar if they cannot be distinguished by any observer that can only perform actions at security level at most δ. Our bisimilarity is a congruence and an efficient proof method for our reduction barbed congruence. A *non interference* result expressed in terms of information flow is also proved by means of labelled bisimilarity.

As case studies, we use our calculus to analyse a *secure* version of the leader election algorithm for mobile ad hoc networks [227]. We then provide an encoding of the *endairA* routing protocol for ad hoc networks [9]. In our encoding, routing paths are associated with a certain security level σ as they are composed only by nodes at security level at least σ. This is quite a desirable property in a multilevel network where information at certain security level is supposed to travel along trusted nodes.

We end this introduction with an outline of the chapter. In Section 7.2, we describe our behaviour-based trust model. In Section 7.3, we describe CTAN, in particular in Section 7.3.1 we provide the syntax and in Section 7.3.2 we give the operational semantics of our calculus. In Section 7.4, we present our mobility model. In Section 7.5, we prove our safety properties. In Section 7.6, we propose a notion of observational equivalence along the lines of Milner and Sangiorgi's barbed congruence. In Section 7.7, we propose a labelled bisimilarity as a proof method for our observations equivalence. More precisely, we prove that our bisimilarity is a congruence and it implies our observational equivalence. In Section 7.8, we use our bisimulation to prove a non-interference result. In Section 7.9, we use our calculus to study a secure version of the leader election protocols for MANETs and an instance of the eindairA routing protocol for MANETs. In Section 7.10, we present related work. Finally, in Section 7.11 we give a summary of the arguments discussed in this chapter.

7.2 Trust model

In this section we propose a behaviour-based multilevel decentralised trust model for MANETs. As in most trust models for distributed systems [46, 64, 149, 198], each node comes together with an extra component called *trust manager*. A trust manager consists of two main components: the *monitoring module* and the *reputation handling module*. The first module monitors the behaviour of neighbours, while the second one collects/spreads recommendations and evaluates trust information about other nodes using a local security policy. The continuous work of

Table 7.1 Trust framework

$m, n \in Nodes$	node name
$\langle \mathcal{S}, < \rangle$	complete lattice
$\mathsf{bad}, \mathsf{trust}, \mathsf{low}, \mathsf{high} \in \mathcal{S}$	security level
$A \in Assertions = Nodes \times Nodes \times \mathcal{S}$	assertion
$T \subseteq \wp(Assertions)$	trust store
$\mathcal{P} : \wp(Assertions) \to \wp(Assertions)$	policy function

the trust manager results in a *local trust store* T containing the up-to-date trust relations.

Trust information may change over time due to mobility, temporary disconnections, recommendations, etc. The main objective of the model is to isolate *bad nodes*, i.e. nodes which do not behave as expected. For this reason, we support *node revocation*. This may happen when a node detects a misbehaviour of another node, and spreads this information to its neighbours. Repudiable evidences allow bad nodes to falsely accuse good nodes. Thus, recommendations are always evaluated using a local security policy implementing an appropriate metric.

The basic elements of our model are nodes (or principals), security levels, assertions, policies and trust stores. We use $k, l, m, n, \ldots$ to range over the set *Nodes* of node names. We assume a complete lattice $\langle \mathcal{S}, < \rangle$ of security levels: $\mathsf{bad} < \mathsf{trust} < \mathsf{low} < \mathsf{high}$. The bad security level is associated to a misbehaviour, trust is the lowest security level associated to trusted behaviour, low is associated to more trusted behaviour, i.e. for more sensible data, whereas high indicates the highest trusted behaviour. We use the Greek letter ρ for security levels belonging to $\mathcal{S}$. The set of *assertions* is defined as $Assertions = Nodes \times Nodes \times \mathcal{S}$. Thus, an assertion $\langle m, n, \rho \rangle$ says that a *node m trusts a node n at security level ρ*. A local trust store T contains a set of assertions, formally $T \subseteq \wp(Assertions)$. When a node m (the trustor) wants to know the security level of a node n (the trustee), it has to check its own trust store T. For convenience, we often use T as a partial function of type $Nodes \to Nodes \to \mathcal{S}$, writing $T(m, n) = \rho$ when m considers n as a node of security level ρ. If $\rho = \mathsf{bad}$ then m considers n a *bad* (unreliable) node and stops any interaction with it. In this manner, we implement a simple form of revocation. A node can receive new assertions from its neighbours. These assertions will be opportunely stored in the local trust store by the trust manager, according to a local security policy $\mathcal{P}$. A *security policy* $\mathcal{P}$ is a function that evaluates the current information collected by a node and returns a set of assertions consistent with the policy implementation. Formally $\mathcal{P} : \wp(Assertions) \to \wp(Assertions)$. For example, let us consider that the policy of a node m has to evaluate these two assertions: $\langle n, q, \rho \rangle$ and $\langle l, q, \rho' \rangle$. The implementation of the policy of m could be such that $\mathcal{P}(\langle n, q, \rho \rangle \cup \langle l, q, \rho' \rangle) = \langle m, q, \rho \rangle$ if $T(m, n) > T(m, l)$, otherwise $\mathcal{P}(\langle n, q, \rho \rangle \cup \langle l, q, \rho' \rangle) = \langle m, q, \rho' \rangle$, where T is the trust table of m. For simplicity, we assume that all nodes have the same security policy $\mathcal{P}$. Notice that the outcome of the policy function could differ from one node to another as the computation depends on the local knowledge of nodes. The presence of a revocation mechanism implies that our policy function is a non-monotone one. Table 7.1 summarises our trust model.

Table 7.2 Syntax of CTAN

Values			
u	::=	v	closed value
	|	x	variable
Networks			
M, N	::=	$\mathbf{0}$	empty network
	|	$M \mid N$	parallel composition
	|	$n[P]_T$	node
Processes			
P, Q	::=	nil	termination
	|	$\sigma!\langle\tilde{u}\rangle.P$	broadcast sender
	|	$\sigma!\langle\tilde{u}\rangle_u.P$	unicast sender
	|	$\sigma?(\tilde{x}).P$	receiver
	|	$P + Q$	nondeterministic choice
	|	$[\tilde{u}\ \mathsf{op}\ \tilde{u}']P, Q$	matching
	|	$H\langle\tilde{u}\rangle$	recursion

Messages exchanged among nodes are assumed to be encrypted using a hierarchical key generation and distribution protocol [122,211]. The trust manager may determine a key redistribution when a security level is compromised. More generally, re-keying [72] allows to refresh a subset of keys when one or more nodes join or leave the network; in this manner nodes are enable to decrypt past traffic, while evicted nodes are unable to decrypt future traffic. As showed in [211] re-keying may be relatively inexpensive if based on "low-cost" hashing operators.

7.3 The Calculus

7.3.1 Syntax

In Table 7.2, we define the syntax of our calculus in a two-level structure, a lower one for *processes* and a upper one for *networks*. We use letters $k, l, m, n, \ldots$ for node names. The Greek symbol σ ranges over the security levels low and high, the only ones which are directly used by programmers. We use letters x, y, z for *variables*, u for *values*, and v and w for *closed values*, i.e. values that do not contain free variables. We write $\tilde{u}$ to denote a tuple $u_1, \ldots, u_k$ of values.

Networks are collections of nodes (which represent devices) running in parallel and using channels at different security levels to communicate with each other. We use the symbol $\mathbf{0}$ to denote an empty network. We write $n[P]_T$ for a node named n (denoting its network address) executing the sequential process P, with a local trust store T.

Processes are sequential and live within the nodes. We write nil to denote the skip process. The multicast sender process $\sigma!\langle\tilde{v}\rangle.P$ transmits the message $\tilde{v}$ to all trusted nodes at security level σ, and then continues as P. The unicast sender process $\sigma!\langle\tilde{v}\rangle_n.P$ transmits the message $\tilde{v}$ to node n at security level σ, and then

continues as P. As we will see in the last section of this paper, unicast transmissions are quite common in MANET protocols.

Remark 7.1. Messages are assumed to be encrypted using a hierarchical key generation and distribution protocol. Thus, a message transmitted at security level ρ can be decrypted only by nodes at security level ρ or greater, according to the trust store of both sender and receiver. Moreover, we assume that messages are always signed by transmitters. In unicast communication we assume the presence of a *link key*, that is a key shared by two neighbours. Link keys provide protection for unicast messages exchanged between two neighbouring nodes.

The receiver process $\sigma?(\tilde{x}).P$ listens for incoming communications at security level σ. Upon reception, the receiver processes evolves into P, where the variables of $\tilde{x}$ are replaced with the message $\tilde{v}$. We write $\{^{\tilde{v}}/_{\tilde{x}}\}P$ for the substitution of variables $\tilde{x}$ with values $\tilde{v}$ in P. In process $[\tilde{v}\,\mathtt{op}\,\tilde{w}]P, Q$, $\mathtt{op}$ is a binary operator returning a boolean. It may range for example over the symbols $=, <, >, \leq, \geq, \in, \subseteq$. The process $[\tilde{v}\,\mathtt{op}\,\tilde{w}]P, Q$ denotes the "if then else" construct: it behaves as P if $\tilde{v}\,\mathtt{op}\,\tilde{w} = \mathtt{true}$, and as Q otherwise. Process $P+Q$ denotes nondeterministic choice. We write $H\langle\tilde{v}\rangle$ to denote a process defined via a definition $H(\tilde{x}) \stackrel{\text{def}}{=} P$, with $\mid \tilde{x} \mid = \mid \tilde{v} \mid$, where $\tilde{x}$ contains all variables that appear free in P. Defining equations provide *guarded recursion*, since P may contain only guarded occurrences of process identifiers. In process $\sigma?(\tilde{x}).P$ variables $\tilde{x}$ are bound in P. This gives rise to the standard notion of α-conversion and free and bound variables. We assume there are no free variables in our networks. The absence of free variables in networks is trivially maintained as the network evolves. Given a network M, $\mathsf{nds}(M)$ returns the set of the names of those nodes which constitute the network M. Notice that, as networks addresses are unique, we assume that there cannot be two nodes with the same name in the same network. We write $\prod_i M_i$ to denote the parallel composition of all sub-networks M_i. We write $M \mid N$ for the parallel composition of two sub-networks M and N. We write $\sigma!\langle\tilde{v}\rangle$ or $\sigma!\langle\tilde{v}\rangle_n$ to mean $\sigma!\langle\tilde{v}\rangle.\mathsf{nil}$ or $\sigma!\langle\tilde{v}\rangle_n.\mathsf{nil}$, respectively.

As usual in process calculi, in Table 7.3 we define *structural congruence*, written $\equiv$, as the basic congruence between networks that only differ for minor differences in the syntax.

7.3.2 Operational Semantics

We give the operational semantics of our calculus in terms of a *labelled transition system* (LTS). We have divided our LTS in two sets of rules. Table 7.4 contains the rules to model the synchronisation between sender and receivers. Table 7.5 contains the rules to model trust management.

Our transitions are of the form

$$M \xrightarrow{\lambda}_{\rho} M'$$

indicating that the network M can perform the action λ, at security level ρ, evolving into the network M'. By construction, in any transition of this form ρ will be different from bad. More precisely, ρ will be equal to low for low-level security transmissions, and equal to high for high-level security transmissions. If $\rho = \mathsf{trust}$

Table 7.3 Structural Congruence

$m[P+Q]_T \equiv m[Q+P]_T$	(Struct Sum Comm)
$m[P+(Q+Q')]_T \equiv m[(P+Q)+Q']_T$	(Struct Sum Assoc)
$m[P+\mathsf{nil}]_T \equiv m[P]_T$	(Struct Sum Zero)
$m[[\tilde{v}\ \mathtt{op}\ \tilde{v}']P,Q]_T \equiv m[P]_T \quad \text{if} \quad \tilde{v}\ \mathtt{op}\ \tilde{v}' = \mathtt{true}$	(Struct Then)
$m[[\tilde{v}\ \mathtt{op}\ \tilde{v}']P,Q]_T \equiv m[Q]_T \quad \text{if} \quad \tilde{v}\ \mathtt{op}\ \tilde{v}' = \mathtt{false}$	(Struct Else)
$m[A\langle\tilde{v}\rangle]_T \equiv m[\{\tilde{v}/\tilde{x}\}P]_T$ if $A(\tilde{x}) \stackrel{\text{def}}{=} P \wedge \mid \tilde{x} \mid = \mid \tilde{v} \mid$	(Struct Rec)
$M \mid N \equiv N \mid M$	(Struct Par Comm)
$(M \mid N) \mid M' \equiv M \mid (N \mid M')$	(Struct Par Assoc)
$M \mid \mathbf{0} \equiv M$	(Struct Par Zero)
$M \equiv M$	(Struct Refl)
$M \equiv N$ implies $N \equiv M$	(Struct Symm)
$M \equiv N \ \wedge\ N \equiv O$ implies $M \equiv O$	(Struct Trans)
$M \equiv N$ implies $M \mid M' \equiv N \mid M'$, for all M'	(Struct Cxt Par)

Table 7.4 LTS - Synchronisation

$$\text{(MCast)}\ \frac{\mathcal{D} := \{n : T(m,n) \geq \sigma\} \quad \mathcal{D} \neq \emptyset}{m[\sigma!\langle\tilde{v}\rangle.P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma m[P]_T} \qquad \text{(Rcv)}\ \frac{T(n,m) \geq \sigma \quad \mid \tilde{x} \mid = \mid \tilde{v} \mid}{n[\sigma?(\tilde{x}).P]_T \xrightarrow{m?\tilde{v}\triangleright n}_\sigma n[\{\tilde{v}/\tilde{x}\}P]_T}$$

$$\text{(UCast)}\ \frac{T(m,n) \geq \sigma}{m[\sigma!\langle\tilde{v}\rangle_n.P]_T \xrightarrow{m!\tilde{v}\triangleright n}_\sigma m[P]_T}$$

$$\text{(RcvPar)}\ \frac{M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_\rho M' \quad N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_\rho N' \quad \widehat{\mathcal{D}} := \mathcal{D} \cup \mathcal{D}'}{M \mid N \xrightarrow{m?\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho M' \mid N'}$$

$$\text{(Sync)}\ \frac{M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M' \quad N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_\rho N' \quad \mathcal{D}' \subseteq \mathcal{D}}{M \mid N \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M' \mid N'}$$

$$\text{(Par)}\ \frac{M \xrightarrow{\lambda}_\rho M' \quad \mathsf{sender}(\lambda) \notin \mathsf{nds}(N)}{M \mid N \xrightarrow{\lambda}_\rho M' \mid N} \qquad \text{(Sum)}\ \frac{m[P]_T \xrightarrow{\lambda}_\sigma m[P']_T}{m[P+Q]_T \xrightarrow{\lambda}_\sigma m[P']_T}$$

then the transition models trust management and involves all trusted nodes. The variable λ ranges over the labels $m!\tilde{v}\triangleright\mathcal{D}$, $m?\tilde{v}\triangleright\mathcal{D}$, and τ, where $\mathcal{D}$ is a set of nodes. We sometimes write $m!\tilde{v}\triangleright n$ and $m?\tilde{v}\triangleright n$ as an abbreviation for $m!\tilde{v}\triangleright\{n\}$ and $m?\tilde{v}\triangleright\{n\}$, respectively. The label $m!\tilde{v}\triangleright\mathcal{D}$ models the transmission of message $\tilde{v}$, originating from node m, and addressed to the set of nodes in $\mathcal{D}$. The label $m?\tilde{v}\triangleright\mathcal{D}$ represents the reception of a message $\tilde{v}$, sent by m, and received by the nodes in $\mathcal{D}$. The label τ models internal actions, which cannot be observed. The function $\mathsf{sender}(\cdot)$ applies to a label and returns the name of the sender, thus $\mathsf{sender}(m!\tilde{v}\triangleright\mathcal{D}) = \mathsf{sender}(m?\tilde{v}\triangleright\mathcal{D}) = m$, whereas $\mathsf{sender}(\tau) = \perp$ (undefined).

Let us comment on the rules of Table 7.4. Rule (MCast) models a node m which sends a message $\tilde{v}$ at security level σ; the set $\mathcal{D}$ contains the destination nodes with security level at least σ, according to the trust store of m.

Remark 7.2. Rule (MCast) may recall the Directed Diffusion approach of [125], in which a node decides to which neighbours to send the data to, according to a *reinforcement* mechanism. These reinforcements limit the paths of transmissions, as messages only travel down these reinforced paths, and hence increase efficiency. In [72] the Directed Diffusion is used to formalise a mechanism of securing group communication.

Rule (Rcv) models a node n receiving a message $\tilde{v}$, sent by node m, at security level σ. Node n receives the message from m only if it trusts m at security level σ. Here, we abstract on the actual behaviour of receivers as they verify the identity of the sender and discard unauthorised messages. Rule (UCast) models a unicast transmission of message $\tilde{v}$ from node m to node n at security level σ. Rule (RcvPar) serves to put together parallel nodes receiving from the same sender. If sender and receiver(s) trust each other then they may synchronise by one or more applications of rule (Sync). In this rule, the condition $\mathcal{D}' \subseteq \mathcal{D}$ ensures that only authorised recipients receive the transmitted value. Rule (Par) is standard in process calculi. Notice that using rule (Par) we can model situations where potential receivers do not necessarily receive the message, either because they are not in the transmission range of the transmitter or simply because they loose the message. Rule (Sum) is also standard: if one of the process performes an action then the whole process performs that action. Rules (Sync), (RcvPar), (Sum) and (Par) have their symmetric counterparts.

Let us explain the rules in Table 7.4 with an example.

Example 7.3. Let us consider the network:

$$M \stackrel{\text{def}}{=} k[\sigma?(\tilde{x}).P_k]_{T_k} \mid l[\sigma?(\tilde{x}).P_l]_{T_l} \mid m[\sigma!\langle\tilde{v}\rangle.P_m]_{T_m} \mid n[\sigma?(\tilde{x}).P_n]_{T_n}$$

where $T_k(k,m) \geq \sigma, T_l(l,m) < \sigma, T_m(m,n) = T_m(m,l) \geq \sigma, T_m(m,k) < \sigma$ and $T_n(n,m) \geq \sigma$. In this configuration, node m broadcasts message $\tilde{v}$ at security level σ, knowing that the nodes allowed to receive the message at that security level are n and l. However, node l does not trust m at security level σ. Thus, n is the only node that may receive the message. By an application of rules (MCast), (Rcv), (Par), and (Sync) we have:

$$M \xrightarrow{m!\tilde{v}\triangleright\{l,n\}}_{\sigma} k[\sigma?(\tilde{x}).P_k]_{T_k} \mid l[\sigma?(\tilde{x}).P_l]_{T_l} \mid m[P_m]_{T_m} \mid n[\{\tilde{v}/\tilde{x}\}P_n]_{T_n} \ .$$

Now, let us comment on the rules of Table 7.5 modelling trust management. We remember that each node comes with a trust manager component which is not specified in our syntax. We rather model the behaviour of this component through the transition rules of Table 7.5. The transmissions modelled in this table are addressed to all trusted nodes i.e. all nodes at security level trust. Rule (DTrust) models *direct trust*. This happens when the *monitoring module* of a node m, while monitoring the activity of a trusted node n, detects a misbehaviour of n. In this case, node m executes two operations: (i) it implements node revocation updating its trust store, according to its local policy; (ii) it broadcasts the

Table 7.5 LTS - Trust Management

$$\text{(DTrust)}\ \frac{T(m,n) > \mathsf{bad} \quad \tilde{v} := n,\mathsf{bad} \quad T' := \mathcal{P}(T \cup \langle m, \tilde{v}\rangle) \quad \mathcal{D} := \{n : T(m,n) > \mathsf{bad}\}}{m[P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathsf{trust}} m[P]_{T'}}$$

$$\text{(SndRcm)}\ \frac{T(m,n) = \rho \quad \tilde{v} := n,\rho \quad \mathcal{D} := \{n : T(m,n) > \mathsf{bad}\}}{m[P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathsf{trust}} m[P]_T}$$

$$\text{(RcvRcm)}\ \frac{T(n,m) > \mathsf{bad} \quad \tilde{v} := l,\rho \quad T' := \mathcal{P}(T \cup \langle m, \tilde{v}\rangle)}{n[P]_T \xrightarrow{m?\tilde{v}\triangleright n}_{\mathsf{trust}} n[P]_{T'}}$$

$$\text{(Lose)}\ \frac{T' \subseteq T \quad T'' := \mathcal{P}(T')}{n[P]_T \xrightarrow{\tau}_{\mathsf{trust}} n[P]_{T''}}$$

corresponding information to inform all *trusted* nodes about the misbehaviour of n. Rule (SndRcm) models *indirect trust* by sending a recommendation. This may happen, for example, when a node moves and asks for recommendations on new neighbours. Again, recommendations are addressed to all trusted nodes, according to the trust knowledge of the recommender. Rule (RcvRcm) models the reception of a recommendation from a trusted node: a new trust table T' is calculated, applying the local policy to $T \cup \langle m, \tilde{v}\rangle$. Rule (Lose) models loss of trust information. This happens, for instance, when a node moves, changing its neighbourhood. In this case, assertions concerning old neighbours must be deleted as they cannot be directly verified. The consistency of the remaining assertions must be maintained by applying the security policy. Notice that, in this manner, we may also loose information about bad nodes that may be encountered again in the future. However, in order to get in touch with an unknown party, a node needs enough recommendations from trusted neighbours.

Let us explain the rules in Table 7.5 with an example.

Example 7.4. Let us show how direct and indirect trust are modelled in our setting. Let us consider the network:

$$M \stackrel{\text{def}}{=} k[P_k]_{T_k} \mid l[P_l]_{T_l} \mid m[P_m]_{T_m} \mid n[P_n]_{T_n}$$

where $T_k(k,m) \geq \mathsf{trust}$, $T_l(l,m) = \mathsf{bad}, T_m(m,n) = T_m(m,l) = T_m(m,k) \geq \mathsf{trust}$, and $T_n(n,m) \geq \mathsf{trust}$. Now, if node m observes that node k is misbehaving, then (i) it adds an assertion $\langle m, k, \mathsf{bad}\rangle$ to its local knowledge; (ii) it broadcasts the information to its trusted nodes. Thus, by an application of rules (DTrust), (RcvRcm), (Par), and (Sync) we have

$$M \xrightarrow{m!\tilde{v}\triangleright\{k,l,n\}}_{\mathsf{trust}} k[P_k]_{T'_k} \mid l[P_l]_{T_l} \mid m[P_m]_{T'_m} \mid n[P_n]_{T'_n} \ .$$

Notice that since l does not trust m, only node n (but also the bad node k) will receive m's recommendation. Moreover the local knowledge of m and n will change, according to their local policy. This is a case of direct trust for m, and indirect trust for n.

Table 7.6 LTS - Matching and Recursion

$$\text{(Then)}\ \frac{n[P]_T \xrightarrow{\lambda}_\rho n[P']_{T'} \quad \tilde{v_1}\ \mathtt{op}\ \tilde{v_2} = \mathtt{true}}{n[[\tilde{v_1}\ \mathtt{op}\ \tilde{v_2}]P, Q]_T \xrightarrow{\lambda}_\rho n[P']_{T'}}$$

$$\text{(Else)}\ \frac{n[Q]_T \xrightarrow{\lambda}_\rho n[Q']_{T'} \quad \tilde{v_1}\ \mathtt{op}\ \tilde{v_2} = \mathtt{false}}{n[[\tilde{v_1}\ \mathtt{op}\ \tilde{v_2}]P, Q]_T \xrightarrow{\lambda}_\rho n[Q']_{T'}}$$

$$\text{(Rec)}\ \frac{n[\{\tilde{v}/\tilde{x}\}P]_T \xrightarrow{\lambda}_\rho n[P']_{T'} \quad H(\tilde{x}) \stackrel{\text{def}}{=} P}{n[H\langle\tilde{v}\rangle]_T \xrightarrow{\lambda}_\rho n[P']_{T'}}$$

Table 7.7 LTS - Synchronisation with network restrictions

$$\text{(MCastR)}\ \frac{\mathcal{D}{:=}\{n : T(m,n) \geq \sigma\} \quad \mathcal{D}{\neq}\emptyset}{m[\sigma!\langle\tilde{v}\rangle.P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma,\emptyset} m[P]_T}$$

$$\text{(RcvR)}\ \frac{T(n,m){\geq}\sigma \quad |\tilde{x}|{=}|\tilde{v}| \quad P'{:=}\{^{\tilde{v}}/_{\tilde{x}}\}P}{n[\sigma?(\tilde{x}).P]_T \xrightarrow{m?\tilde{v}\triangleright n}_{\sigma,(n,m)} n[P']_T}$$

$$\text{(UCastR)}\ \frac{T(m,n) \geq \sigma}{m[\sigma!\langle\tilde{v}\rangle_n.P]_T \xrightarrow{m!\tilde{v}\triangleright n}_{\sigma,\emptyset} m[P]_T}$$

$$\text{(RcvParR)}\ \frac{M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1} M' \quad N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho,C_2} N' \quad \widehat{\mathcal{D}} := \mathcal{D}\cup\mathcal{D}'}{M \mid N \xrightarrow{m?\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho,C_1\cup C_2} M' \mid N'}$$

$$\text{(SyncR)}\ \frac{M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1} M' \quad N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho,C_2} N' \quad \mathcal{D}'\subseteq\mathcal{D}}{M \mid N \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1\cup C_2} M' \mid N'}$$

$$\text{(ParR)}\ \frac{M \xrightarrow{\lambda}_{\rho,C} M' \quad \mathsf{sender}(\lambda) \notin \mathsf{nds}(N)}{M \mid N \xrightarrow{\lambda}_{\rho,C} M' \mid N}$$

$$\text{(SumR)}\ \frac{m[P]_T \xrightarrow{\lambda}_{\sigma,C} m[P']_T}{m[P+Q]_T \xrightarrow{\lambda}_{\sigma,C} m[P']_T}$$

Finally, Table 7.6 contains the standard rules for matching and recursion.

7.4 Node Mobility

In wireless networks, node mobility is associated with the ability of a node to access telecommunication services at different locations from different nodes. Unlike wired networks, where the main security requirements are addressed by installing firewalls, in mobile ad hoc networks node mobility introduces new issues related

to user credential management, indirect trust establishment and mutual authentication between previously unknown and hence untrusted nodes.

For these reasons, node mobility in ad hoc networks has turned to be a challenge for automated verification and analysis techniques. After the first work on model checking of (stationary) ad hoc networks [39], Nanz and Hankin [178] have proposed a process calculus where topology changes are abstracted into a *fixed* representation. This representation, called *network topology*, is essentially a set of *connectivity graphs* denoting the possible connectivities within the nodes of the network. Thus, in [178] the topology is not part of the syntax, but it is a parameter of the operational semantics. A similar approach is introduced in [92], although the labelled transition systems and the equivalence relations are completely different. In [178] a transition to the next state is examined for all possible valid graphs (those contained in the network topology fixed a priori) whereas in [92] a transition is examined for all graphs containing the connections used in a communication. These connections are called *restrictions*.

As the reader may have noticed, our calculus does not directly model the network topology neither in the syntax nor in the semantics. However, it is very easy to add topology changes at semantics level, so that each state represents a set of valid topologies, and a network can be at any of those topologies at any time [92]. In Table 7.7 we rewrite the rules of Table 7.4 in the style of [92]. Rules are of the form

$$M \xrightarrow{\lambda}_{\rho,C} M'$$

indicating that the network M can perform the action λ, at security level ρ, under the network restriction C, evolving into the network M'. Thus, a network restriction C keeps track of the connections which are necessary for the transition to fire. The rules in Table 7.5 can be rewritten in a similar manner, except for rule (Lose) in which the network restriction is empty. Notice that the rule (Lose) in Table 7.5 affects network topology changes. In fact, if a trust information is lost then certain nodes may not be able of communicating anymore.

Example 7.5. Consider the same network given in the Example 7.3. Then by applying rules (MCastR), (RcvR), (ParR), and (SyncR) we have

$$M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma,\{(n,m)\}} k[\sigma?(\tilde{x}).P_k]_{T_k} \mid l[\sigma?(\tilde{x}).P_l]_{T_l} \mid m[P_m]_{T_m} \mid n[\{^{\tilde{v}}/_{\tilde{x}}\}P_n]_{T_n}.$$

The transition is tagged with the network restriction $\{(n,m)\}$, as only node n has synchronised with node m.

The reader may have noticed that the rules of Table 7.7 do not use network restrictions in the premises. As a consequence, there is a straightforward operational correspondence between a transition $\xrightarrow{\lambda}_{\rho}$ and one of the form $\xrightarrow{\lambda}_{\rho,C}$.

Proposition 7.6.

1. *$M \xrightarrow{\lambda}_{\rho} M'$ with $\lambda \in \{m!\tilde{v}\triangleright\mathcal{D}, m?\tilde{v}\triangleright\mathcal{D}\}$ iff there exists a restriction C such that $M \xrightarrow{\lambda}_{\rho,C} M'$ and $C \subseteq \{(m,n)$ for all $n \in \mathcal{D}\}$.*
2. *$M \xrightarrow{\tau}_{\rho} M'$ iff $M \xrightarrow{\tau}_{\rho,\emptyset} M'$.*

Proof

1. We separately prove that ($\Rightarrow$) if $M \xrightarrow{\lambda}_{\rho} M'$, with $\lambda := \{m!\tilde{v}\triangleright\mathcal{D} \mid m?\tilde{v}\triangleright\mathcal{D}\}$, then there exists a restriction C such that $M \xrightarrow{\lambda}_{\rho,C} M'$ and $C \subseteq \{(m,n) \text{ for all } n \in \mathcal{D}\}$ and ($\Leftarrow$) if $M \xrightarrow{\lambda}_{\rho,C} M'$, with $\lambda := \{m!\tilde{v}\triangleright\mathcal{D} \mid m?\tilde{v}\triangleright\mathcal{D}\}$ and $C \subseteq \{(m,n) \text{ for all } n \in \mathcal{D}\}$ then $M \xrightarrow{\lambda}_{\rho} M'$. It is proved by induction on $M \xrightarrow{\lambda}_{\rho} M'$, in the first case, and on $M \xrightarrow{\lambda}_{\rho,C} M'$, in the second case. We show details only for $\lambda = m!\tilde{v}\triangleright\mathcal{D}$. Similar proof can be exhibited for $\lambda = m?\tilde{v}\triangleright\mathcal{D}$.
   ($\Rightarrow$) The base cases are when the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ is given by rules (MCast), (UCast), (DTrust) or (SndRcm). By rules (MCastR), (UCastR), (DTrustR) or (SndRcmR), respectively, there is $C = \emptyset$ such that $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,\emptyset} M'$. As to the inductive case, let us consider $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ given by rules (Sum), (Sync) or (Par). We show details only for rule (Sync). Let $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ by rule (Sync), with $M = M_1 \mid M_2, M' = M_1' \mid M_2'$ for some $M_1, M_1', M_2,$ and M_2', because $M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M_1'$ and $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho} M_2'$ (the converse is similar) with $\mathcal{D}' \subseteq \mathcal{D}$. By inductive hypothesis, it holds that there are C_1 and C_2 such that $M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1} M_1'$, $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho,C_2} M_2'$, with $C_1 \subseteq \{(m,n) \text{ for all } n \in \mathcal{D}\}$ and $C_2 \subseteq \{(m,n) \text{ for all } n \in \mathcal{D}'\}$. By rule (SyncR) it holds that

$$M_1 \mid M_2 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1\cup C_2} M_1' \mid M_2'.$$

   As $\mathcal{D}' \subseteq \mathcal{D}$, it holds that $C_1 \cup C_2 \subseteq \{(m,n) \text{ for all } n \in \mathcal{D}\}$, as required.
   ($\Leftarrow$) The base cases are when the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C} M'$ is given by rules (MCastR), (UCastR), (DTrustR) or (SndRcmR). Then $C = \emptyset$. By rules (MCast), (UCast), (DTrust) or (SndRcm) we have $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$. As to the inductive case, let us consider $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C} M'$ given by rules (SumR), (SyncR) or (ParR). We show details only for rule (SyncR). Hence, let $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C} M'$ given by rule (SyncR), with $M = M_1 \mid M_2$ and $M' = M_1' \mid M_2'$, for some M_1, M_2, M_1' and M_2', because $M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1} M_1'$ and $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho,C_2} M_2'$ (the converse is similar), with $C = C_1 \cup C_2 \subseteq \{(m,n) \text{ for all } n \in \mathcal{D}\}$ and $\mathcal{D}' \subseteq \mathcal{D}$. Then $C_1 \subseteq \{(m,n) \text{ for all } n \in \mathcal{D}\}$ and by construction it holds that $C_2 \subseteq \{(m,n) \text{ for all } n \in \mathcal{D}'\}$. By inductive hypothesis we have $M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M_1'$ and $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho} M_2'$. By rule (Sync) it holds that $M_1 \mid M_2 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M_1' \mid M_2'$, as required.
2. The proof of this case can be found in Section A.2 at page 178.

□

7.5 Safety Properties

Access control [207] is a well-established technique to provide *safety properties* ensuring that only principals with appropriate access rights can access data.

In general, in distributed systems safety properties ensure that no invalid communications arise [87]. In our setting, safety properties involve security levels, as communications are parameterised on them. Thus, our safety properties aim at guaranteeing that only authorised nodes receive sensible information.

We define a notion of *safety up to a security level* to describe when a communication is safe up to a certain security level.

Definition 7.7 (Safety up to a security level). *A node m transmitting at level ρ may only synchronise with a node n receiving at level ρ or above, according to the local knowledge of m and n, respectively.*

Intuitively, Definition 7.7 says that a synchronisation at a certain security level ρ is safe if the involved parties trust each other at that security level.

The next theorem states that the safety property is preserved at run time: under all possible evolutions, no invalid synchronisation arises. Intuitively, it is proved that if a node n_i receives a message sent by a sender m at security level ρ, it means that m and n_i trust each other at security level at least ρ.

Theorem 7.8 (Safety preservation). *Let* $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ *with*

$$M \equiv m[P]_T \mid \prod_i n_i[P_i]_{T_i} \text{ and } M' \equiv m[P']_{T'} \mid \prod_i n_i[P'_i]_{T'_i} \ .$$

1. *If $P'_i \neq P_i$, for some i, then $T(m, n_i) \geq \rho$ and $T_i(n_i, m) \geq \rho$.*
2. *If $T'_i \neq T_i$, for some i, then $T(m, n_i) \geq \rho$ and $T_i(n_i, m) \geq \rho$.*

Proof

1. This case applies only for transitions at level σ. The proof is by induction on the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$. The case base is when $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ is given by rule (Sync), with $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid n[\sigma?(\tilde{x}).Q]_{T'}$ and $M' \equiv m[P]_T \mid n[\{\tilde{v}/\tilde{x}\}Q]_{T'}$ or $M \equiv m[\sigma!\langle\tilde{v}\rangle_n.P]_T \mid n[\sigma?(\tilde{x}).Q]_{T'}$. We show details only for the first case, the other one is similar. By rule (MCast) we have

$$m[\sigma!\langle\tilde{v}\rangle.P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} m[P]_T,$$

with $\mathcal{D} := \{n : T(m, n) \geq \sigma\}$ and by rule (Rcv)

$$n[\sigma?(\tilde{x}).Q]_{T'} \xrightarrow{m?\tilde{v}\triangleright n}_{\sigma} n[\{\tilde{v}/\tilde{x}\}Q]_{T'},$$

with $T'(n, m) \geq \sigma$. As $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ by (Sync), it holds that $n \in \mathcal{D}$ and then $T(m, n) \geq \sigma$, as required. As to the inductive case, let us consider $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ given by rule (Sync) or (Par), with $M \equiv m[P]_T \mid \prod_i n_i[P_i]_{T_i}$ and $M' \equiv m[P']_T \mid \prod_i n_i[P'_i]_{T_i}$. We show details only for the first case, the other one is similar. Let $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ by (Sync), with $M = M_1 \mid M_2$ and $M' = M'_1 \mid M'_2$, for some M_1, M_2, M'_1 and M'_2, because $M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'_1$

and $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\sigma} M_2'$ (the converse is similar), with $\mathcal{D} := \{n : T(m,n) \geq \sigma\}, \mathcal{D}' \subseteq \mathcal{D}$. More precisely, let

$$M \equiv m[P]_T \mid \prod_i n_i[P_i]_{T_i} = m[P]_T \mid \prod_k n_k[P_k]_{T_k} \mid \prod_j n_j[P_j]_{T_j}$$

and

$$M' \equiv m[P']_T \mid \prod_i n_i[P_i']_{T_i} = m[P']_T \mid \prod_k n_k[P_k']_{T_k} \mid \prod_j n_j[P_j']_{T_j}$$

for appropriate processes and tags, where

$$M_1 = m[P]_T \mid \prod_k n_k[P_k]_{T_k}, M_1' = m[P']_T \mid \prod_k n_k[P_k']_{T_k},$$

$$M_2 = \prod_j n_j[P_j]_{T_j}, M_2' = \prod_j n_j[P_j']_{T_j}.$$

By inductive hypothesis, if $P_k' \neq P_k$, for some k, then $T(m,n_k) \geq \sigma$ and $T_k(n_k,m) \geq \sigma$. By Lemma A.5(1) at page 178, if $P_j' \neq P_j$, for some j, then $T_j(n_j,m) \geq \sigma$. It remains to prove that $T(m,n_j) \geq \sigma$. But, by Lemma A.5(1) we have $n_j \in \mathcal{D}'$. As $\mathcal{D}' \subseteq \mathcal{D}$ and $\mathcal{D} := \{n : T(m,n) \geq \sigma\}$ it holds that $T(m,n_j) \geq \sigma$, as required.

2. This case applies only for transitions at level `trust`. The proof is by induction on the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\texttt{trust}} M'$. The proof is similar to the previous case.

□

In [87] a conformance criterion is defined in terms of *safety despite compromised principals*. According to this criterion, an invalid authorisation decision at an uncompromised node can arise only when nodes on which the decision logically depends are compromised. A node is said *compromised* (or *bad*) when its privileges can be exercised by the attacker. A realistic threat model for a distributed system, and then for hoc networks, should include *partial compromise*, that is the possibility that some of the nodes in the system are compromised. Partial compromise covers deliberate insider attacks as well as external attackers taking ownership of insiders' assets.

In our setting, the safety despite compromise property comes as a consequence of Theorem 7.8, for which trusted nodes never synchronise with untrusted nodes. In this manner, bad nodes (recognised as such) are isolated from the rest of the network and they cannot affect communications.

Corollary 7.9 (Safety despite compromise). *Let* $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ *with*

$$M \equiv m[P]_T \mid \prod_i n_i[P_i]_{T_i} \text{ and } M' \equiv m[P']_{T'} \mid \prod_i n_i[P_i']_{T_i'} \ .$$

If $T(m,n_i)=\mathsf{bad}$ *or* $T_i(n_i,m)=\mathsf{bad}$, *for some* i, *then* $P_i'=P_i$ *and* $T_i'=T_i$.

Proof We proceed by contradiction. We prove that if $P_i' \neq P_i$ or $T_i' \neq T_i$, for some i, then $T(m, n_i) \neq \mathsf{bad}$ and $T_i(n_i, m) \neq \mathsf{bad}$. Indeed, by Theorem 7.8(1) if $P_i' \neq P_i$ it holds that $T(m, n_i) \geq \rho$ and $T_i(n_i, m) \geq \rho$ and by Theorem 7.8(2) if $T_i' \neq T_i$ it holds that $T(m, n_i) \geq \rho$ and $T_i(n_i, m) \geq \rho$. By construction we know that $\rho \neq \mathsf{bad}$. Then this is in contradiction with the hypotheses. □

7.6 Behavioural Semantics

Our main behavioural equivalence is *σ-reduction barbed congruence*, a variant of Milner and Sangiorgi's (weak) barbed congruence [172] which takes into account security levels. Basically, two terms are barbed congruent if they have the same *observables* (called *barbs*) in all possible contexts, under all possible *evolutions*. For the definition of barbed congruence we need two crucial concepts: a *reduction semantics* to describe how a system evolves, and a notion of *observable* which says what the environment can observe in a system.

From the LTS given in Section 7.3.2 it is easy to see that a network may evolve either because there is a transmission at a certain security level or because a node loses some trust information. Thus, we can define the reduction relation $\twoheadrightarrow$ between networks using the following inference rules:

$$\text{(Red1)}\ \frac{M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'}{M \twoheadrightarrow M'} \qquad \text{(Red2)}\ \frac{M \xrightarrow{\tau}_{\mathsf{trust}} M'}{M \twoheadrightarrow M'}$$

We write $\twoheadrightarrow^*$ to denote the reflexive and transitive closure of $\twoheadrightarrow$.

Also in CTAN, as in TCWS and in in CCS [169] and in π-calculus [171], we have both transmission and reception of messages although only transmissions can be observed. In fact, in a broadcasting calculus an observer cannot see whether a given process actually receives a broadcast synchronisation. In particular, if the node $m[\sigma!\langle\tilde{v}\rangle.P]_T$ or the node $m[\sigma!\langle\tilde{v}\rangle_n.P]_T$ evolves into $m[P]_T$ we do not know whether some potential recipient has synchronised with m. On the other hand, if a node $n[\sigma?(\tilde{x}).P]_T$ evolves into $n[\{\tilde{v}/\tilde{x}\}P]_T$, then we can be sure that some trusted node has transmitted a message $\tilde{v}$ to n at security level σ.

Definition 7.10 (σ-Barb). *We write $M \downarrow_n^\sigma$ if either $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$ or $M \equiv m[\sigma!\langle\tilde{v}\rangle_n.P]_T \mid N$, for some $m, N, \tilde{v}, P, T$ such that $n \notin \mathsf{nds}(M)$, and $T(m, n) \geq \sigma$. We write $M \Downarrow_n^\sigma$ if $M \twoheadrightarrow^* M' \downarrow_n^\sigma$ for some network M'.*

The barb $M \Downarrow_n^\sigma$ says that there is a potential transmission at security level σ, originating from M, and that may reach the node n in the environment. In the sequel, we write $\mathcal{R}$ to denote binary relations over networks. As usual, we need to provide some definitions of properties on which our reduction barbed congruence relies. As in TCWS, also in CTAN we are interested in weak behavioural equivalences; then the definition of reduction closure is given in terms of weak reductions.

Definition 7.11 (σ-Barb preserving). *A relation $\mathcal{R}$ is said to be* σ-barb preserving *if whenever $M\ \mathcal{R}\ N$ it holds that $M \downarrow_n^\sigma$ implies $N \Downarrow_n^\sigma$.*

Definition 7.12 (Reduction closure). *A relation $\mathcal{R}$ is said to be* reduction closed *if $M\ \mathcal{R}\ N$ and $M \twoheadrightarrow M'$ imply there is N' such that $N \twoheadrightarrow^* N'$ and $M'\ \mathcal{R}\ N'$.*

Definition 7.13 (Contextuality). *A relation $\mathcal{R}$ is said to be* contextual *if $M\ \mathcal{R}\ N$ implies that $M \mid O\ \mathcal{R}\ N \mid O$, for all networks O.*

Finally, everything is in place to define our σ-reduction barbed congruence.

Definition 7.14 (σ-Reduction barbed congruence). *The* σ-reduction barbed congruence*, written $\cong_\sigma$, is the largest symmetric relation over networks which is σ-barb preserving, reduction closed and contextual.*

7.7 A Bisimulation Proof Method

In the sequel we define an appropriate notion of bisimulation and, as a main result, we prove that our labelled bisimilarity is a proof-technique for our σ-reduction barbed congruence.

In general, a bisimulation describes how two terms (in our case networks) can mimic each other's actions. First of all we have to distinguish between transmissions which can be observed and transmissions which can not be observed by the environment.

$$\text{(Shh)}\ \frac{M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M' \quad \mathcal{D}\subseteq\mathsf{nds}(M) \quad \rho'\neq\mathsf{bad}}{M \xrightarrow{\tau}_{\rho'} M'} \qquad \text{(Obs)}\ \frac{M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M' \quad \mathcal{D}':=\mathcal{D}\setminus\mathsf{nds}(M)\neq\emptyset}{M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_\rho M'}$$

Rule (Shh) models transmissions that cannot be observed because none of the potential receivers are in the environment. Notice that security levels of τ-action are not related to the transmissions they originate from. Rule (Obs) models a transmission, at security level ρ, of a message $\tilde{v}$, from a sender m, that may be received by the nodes of the environment contained in $\widehat{\mathcal{D}}$. Notice that the derivation tree the rule (Obs) can only be applied at top-level. In fact, we cannot use this rule together with rule (Par) of Table 7.4, because λ does not range on the new action.

In the sequel, we use the metavariable α to range over the following actions: τ, $m?\tilde{v}\triangleright\mathcal{D}$ and $m!\tilde{v}\blacktriangleright\mathcal{D}$. Since we are interested in *weak behavioural equivalences*, that abstract over τ-actions, we introduce a standard notion of weak action: we write $\Rightarrow_\rho$ denoting the reflexive and transitive closure of $\xrightarrow{\tau}_\rho$; we also write $\overset{\alpha}{\Longrightarrow}_\rho$ denotes $\Rightarrow_\rho \xrightarrow{\alpha}_\rho \Rightarrow_\rho$; $\overset{\hat{\alpha}}{\Longrightarrow}_\rho$ denotes $\Rightarrow_\rho$ if $\alpha = \tau$ and $\overset{\alpha}{\Longrightarrow}_\rho$ otherwise.

Definition 7.15 (δ-Bisimilarity). *The* δ-bisimilarity*, written $\approx_\delta$, is the largest symmetric relation over networks such that whenever $M \approx_\delta N$ if $M \xrightarrow{\alpha}_\rho M'$, with $\rho \leq \delta$, then there exists a network N' such that $N \overset{\hat{\alpha}}{\Longrightarrow}_\rho N'$ and $M' \approx_\delta N'$.*

This definition is inspired by that proposed in [70]. Intuitively, two networks are δ-bisimilar if they cannot be distinguished by any observer that can perform actions at security level at most δ.

Theorem 7.16 ($\approx_\delta$ is contextual). *Let M and N be two networks such that $M \approx_\delta N$. Then $M \mid O \approx_\delta N \mid O$ for all networks O.*

Proof We prove that the relation

$$\mathcal{S} \stackrel{\text{def}}{=} \{(M \mid O\,,\, N \mid O) \text{ for all } O \text{ such that } M \approx_\delta N\}$$

is a δ-bisimulation. We proceed by case analysis on the transition $M \mid O \xrightarrow{\alpha}_\rho \widehat{M}$, with $\rho \leq \delta$. We show just a few cases. The full proof can be found in Section A.2 at page 180.

- Let $M \mid O \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\rho \widehat{M}$ by an application of rule (Obs), with $\mathcal{D} \neq \emptyset$, because $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$, with $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O) \neq \emptyset$. We have the following possibilities:
  - Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho M'$ and $O \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_\rho O'$, with $\widehat{M} = M' \mid O'$ and $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$. Let $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M)$; as $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O) \neq \emptyset$ then also $\mathcal{D}' \neq \emptyset$. Then we can apply (Obs) and obtain $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_\rho M'$. As $M \approx_\delta N$ then there is N' such that $N \stackrel{m!\tilde{v}\blacktriangleright\mathcal{D}'}{\Longrightarrow}_\rho N'$ with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

$$N \stackrel{\tau}{\Longrightarrow}_\rho N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_\rho N_2 \stackrel{\tau}{\Longrightarrow}_\rho N'$$

    with $\mathcal{D}' := \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \neq \emptyset$. By Lemma A.7 at page 179 we have $M \equiv \widehat{M} \mid m[P]_T$ and $N_1 \equiv \widehat{N} \mid m[Q]_T$, for some $\widehat{M}, \widehat{N}, T, Q$ and P, and $\widehat{\mathcal{D}} := \{n : T(m,n) \geq \rho\}$ and $\widehat{\mathcal{D}'} := \{n : T(m,n) \geq \rho\}$. Then $\widehat{\mathcal{D}'} = \widehat{\mathcal{D}}$. By several applications of Lemma A.6 at page 179 and one application of rule (Sync), as $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$, we have

$$N \mid O \stackrel{\tau}{\Longrightarrow}_\rho N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho N_2 \mid O' \stackrel{\tau}{\Longrightarrow}_\rho N' \mid O'.$$

    It holds that

$$\begin{aligned} \mathcal{D} &:= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) \\ &= \mathcal{D}' \setminus \mathsf{nds}(O) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(N \mid O) \neq \emptyset. \end{aligned}$$

    Then by one application of rule (Obs) we have $N \mid O \stackrel{m!\tilde{v}\blacktriangleright\mathcal{D}}{\Longrightarrow}_\rho N' \mid O'$ and $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.

□

Theorem 7.17 (Soundness). *Let M and N be two networks such that $M \approx_\delta N$. Then $M \cong_\sigma N$, for $\sigma \leq \delta$.*

Proof The σ-barb preserving follows by Lemma A.10 at page 183, the reduction-closure follows by definition and contextuality follows by Theorem 7.16. □

Remark 7.18. For the sake of analysis, we can redefine the δ-bisimilarity using the labelled transition system with network restrictions. However, in Proposition 7.6 we already proved the operational correspondence between the two labelled transition systems. As a consequence, the resulting bisimilarity would not change.

7.8 Non-Interference

We described in Section 5.3 very general schemata for the definition and analysis of security properties. In particular, we dealt with information flow properties. We remember that they are a particular class of security properties for controlling the flow of information among different entities. The seminal idea of *non-interference* proposed in [96] aims at assuring that "*variety in a secret input should not be conveyed to public output*". In a multilevel computer system [28] this property says that information can only flow from low levels to higher ones. The first taxonomy of non-interference-like properties has been uniformly defined and compared in [81,82] in the context of CCS-like process calculus. In particular, processes were divided into high-level and low-level processes, according to the level of actions they can perform. To detect whether an incorrect information flow (i.e. from high-level to low-level) has occurred, a particular non-interference-like property has been defined, the so-called *Non Deducibility on Composition* (NDC). This property basically says that a process is secure with respect to wrong information flows if its low-level behaviour is independent of changes to its high-level behaviour.

Here, we prove a non-interference result using our notion of δ-bisimilarity. Intuitively, high-level behaviours can arbitrarily change without affecting low-level equivalences.

In Definition 7.19 we formalise the concept of high-level behaviour in the terms of high-level networks. We recall that actions at security level trust do not depend on the syntax of the processes as they only depend on the trust manager; thus, these actions can fire at any moment of the computation.

Definition 7.19 (δ-high level network). *A network H is a* δ-high level network*, written $H \in \mathcal{H}_\delta$, if whenever $H \xrightarrow{\lambda}_{\delta'} H'$ then either $\delta' = \mathsf{trust}$ or $\delta' > \delta$. Moreover, $H' \in \mathcal{H}_\delta$.*

The non-interference result is stated in the following theorem. Intuitively, if two δ-bisimilar networks M and N run in parallel with two high-level networks H and K and such that $H \approx_{\mathsf{trust}} K$ then the resulting networks $M \mid H$ and $N \mid K$ are δ-bisimilar as well.

Theorem 7.20 (Non-interference). *Let M and N be two networks such that $M \approx_\delta N$. Let H and K be two networks such that: (i) $H, K \in \mathcal{H}_\delta$, (ii) $H \approx_{\mathsf{trust}} K$, and (iii) $\mathsf{nds}(H) = \mathsf{nds}(K)$. Then, $M \mid H \approx_\delta N \mid K$.*

Proof We prove that the relation

$$\mathcal{S} \stackrel{\text{def}}{=} \{\big(M \mid H\,,\, N \mid K\big) \text{ for all } H, K \in \mathcal{H}_\delta \text{ such that}$$

$$M \approx_\delta N, H \approx_{\mathsf{trust}} K \text{ and } \mathsf{nds}(H) = \mathsf{nds}(K)\}$$

is a δ-bisimulation. We do a case analysis on the transition $M \mid H \xrightarrow{\alpha}_\rho \widehat{M}$, with $\rho \leq \delta$. We show just a few cases. The full proof can be found in Section A.2 at page 184.

- Let $M \mid H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\rho \widehat{M}$ by an application of rule (Obs), with $\rho \neq \texttt{trust}$, because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$. The only possibility is that $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$ by rule (Par) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho M'$ with $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset, m \notin \mathsf{nds}(H)$, as by Lemma A.7 at page 179 $m \in \mathsf{nds}(M)$, and $\widehat{M} = M' \mid H$. Let $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M)$; as $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$ then also $\mathcal{D}' \neq \emptyset$. We can apply rule (Obs) and obtain $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'} M'$. As $M \approx_\delta N$ then there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_\rho N'$ with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

$$N \xRightarrow{\tau}_\rho N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_\rho N_2 \xRightarrow{\tau}_\rho N'$$

with $\mathcal{D}' := \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \neq \emptyset$. As $m \notin \mathsf{nds}(K)$, by several applications of Lemma A.6 at page 179 and by one application of rule (Par) we have:

$$N \mid K \xRightarrow{\tau}_\rho N_1 \mid K \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_\rho N_2 \mid K \xRightarrow{\tau}_\rho N' \mid K.$$

As $\mathsf{nds}(H) = \mathsf{nds}(K)$ it holds that

$$\begin{aligned} \mathcal{D} &:= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ &= \mathcal{D}' \setminus \mathsf{nds}(H) \\ &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) \neq \emptyset. \end{aligned}$$

Thus by one application of rule (Obs) we have $N \mid K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\rho N' \mid K$ and $(M' \mid H, N' \mid K) \in \mathcal{S}$, as required.

- Let $M \mid H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\rho \widehat{M}$ by an application of rule (Obs), with $\rho = \mathsf{trust}$, because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} \widehat{M}$, with $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$. We have the following possibilities:
  - Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} M'$ and $H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} H'$, with $\widehat{M} = M' \mid H', \mathcal{D}'' \subseteq \widehat{\mathcal{D}}$ and $H' \in \mathcal{H}_\delta$. Let $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M)$; as $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$ then also $\mathcal{D}' \neq \emptyset$. We can apply (Obs) and obtain $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathsf{trust}} M'$. As $M \approx_\delta N$ then there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathsf{trust}} N'$ with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

$$N \xRightarrow{\tau}_{\mathsf{trust}} N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\mathsf{trust}} N_2 \xRightarrow{\tau}_{\mathsf{trust}} N'$$

with $\mathcal{D}' := \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \neq \emptyset$. As $H \approx_{\mathsf{trust}} K$ then $K \xRightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} K'$ with $H' \approx_{\mathsf{trust}} K'$ and $K' \in \mathcal{H}_\delta$. Then there are K_1 and K_2 such that

$$K \stackrel{\tau}{\Longrightarrow}_{\mathsf{trust}} K_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} K_2 \stackrel{\tau}{\Longrightarrow}_{\mathsf{trust}} K'$$

By Lemma A.7 at page 179 we have $M \equiv \widehat{M} \mid m[P]_T$ and $N_1 \equiv \widehat{N} \mid m[Q]_T$, for some $\widehat{M}, \widehat{N}, T, Q$ and P and $\widehat{\mathcal{D}} := \{n : T(m,n) \geq \mathsf{trust}\}$ and $\widehat{\mathcal{D}'} := \{n : T(m,n) \geq \mathsf{trust}\}$. Then $\widehat{\mathcal{D}'} = \widehat{\mathcal{D}}$. By several applications of Lemma A.6 at page 179 and one application of rule (Sync), as $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$, we have

$$N \mid K \stackrel{\tau}{\Longrightarrow}_{\mathsf{trust}} N_1 \mid K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} N_2 \mid K_2 \stackrel{\tau}{\Longrightarrow}_{\mathsf{trust}} N' \mid K'.$$

As $\mathsf{nds}(K) = \mathsf{nds}(H)$ it holds that

$$\begin{array}{rcl} \mathcal{D} & := & \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \\ & = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ & = & \mathcal{D}' \setminus \mathsf{nds}(H) \\ & = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) \neq \emptyset. \end{array}$$

Then $N \mid K \stackrel{m!\tilde{v}\blacktriangleright\mathcal{D}}{\Longrightarrow}_{\rho} N' \mid K'$ and $(M' \mid H', N' \mid K') \in \mathcal{S}$, as required. □

7.9 Case Studies

In this section, we use our calculus to formalise and analyse a *secure* version of the leader election algorithm for MANETs [227]. Our encoding is inspired by that given in the ω-calculus [214] for the original leader election algorithm of [227]. We then propose an encoding of the *endairA* routing protocol for ad hoc networks [9]. In our encoding, routing paths are associated with at a certain security level σ as they are composed only by nodes at security level at least σ. This is quite a desirable property in a multilevel network where information at certain security level is supposed to travel along trusted nodes.

7.9.1 A Secure Leader Election Protocol for MANETs

A subnetwork of a network M is said to be a *connected component* of M if all nodes are connected to each other via one or more hops. For simplicity, we assume a total order among node names (also called node-ids). The algorithm proposed in [227] serves to elect the leader node of a connected component with maximum node-id. Nodes periodically use *probe* and *reply* messages to keep track of their neighbours.

The algorithm operates by first "growing" and then "shrinking" a spanning tree rooted at the node that initiates the election algorithm. We refer to this computation-initiating node as the source node. As we will see, after the spanning tree shrinks completely, the source node will have adequate information to determine the most-valued-node and will then broadcast its identity to the rest of the nodes in the network. The algorithm uses three kinds of messages, viz. *Election*, *Ack* and *Leader*. Elections messages are used to grow the spanning tree. When

Figure 7.1 The encoding of the leader election protocol for MANETs

/* *Starting node broadcasts election message and evolves into the* AwaitAckInit *state waiting for ack messages.* */

$$\textsc{Source}(id, elec, lid) \stackrel{\text{def}}{=} \sigma!\langle \texttt{elecMsg}, id\rangle.\textsc{AwaitAckInit}(id, 1, lid)$$

/* *In the* AwaitAckInit *state the initiator node receives ack messages (other messages are ignored) and store the maximum node-id received; eventually the process evolves into the* SendLeader *state.* */

$$\begin{aligned}\textsc{AwaitAckInit}(id, elec, lid) \stackrel{\text{def}}{=}\ & \sigma?(\tilde{x}).[\mathbf{fst}(\tilde{x}) = \texttt{ackMsg}]\\ & \quad [\mathbf{snd}(\tilde{x}) \geq lid]\textsc{AwaitAckInit}(id, elec, \mathbf{snd}(\tilde{x})),\\ & \quad \textsc{AwaitAckInit}(id, elec, lid),\\ & \textsc{AwaitAckInit}(id, elec, lid)\\ & + \textsc{SendLeader}(id, elec, lid)\end{aligned}$$

/* *In the* SendLeader *state the node broadcasts a leader message.* */

$$\textsc{SendLeader}(id, elec, lid) \stackrel{\text{def}}{=} \sigma!\langle \texttt{ldrMsg}, lid\rangle.\textsc{Node}(id, 0, lid)$$

/* *A node which did not initiate the protocol may receive either an election or a leader message, evolving into an* ElectionProcess *or a* LeaderProcess *state, respectively.* */

$$\begin{aligned}\textsc{Node}(id, elec, lid) \stackrel{\text{def}}{=}\ & \sigma?(\tilde{x}).[\mathbf{fst}(\tilde{x}) = \texttt{ldrMsg}]\\ & \quad \textsc{LeaderProcess}(id, elec, lid, \mathbf{snd}(\tilde{x})),\\ & [\mathbf{fst}(\tilde{x}) = \texttt{elecMsg}]\textsc{ElectionProcess}(id, 1, lid, \mathbf{snd}(\tilde{x})), \textsc{Node}(id, elec, lid)\end{aligned}$$

/* *A node in the* LeaderProcess *state basically propagates leader messages containing the maximum between its lid and the maxid received in the leader messages. The most interesting case in when maxid* $<$ *lid. This means that either the node was not part of the election process or the node did not report the ack to its parent nodes, for example because it was disconnected. In both cases, it broadcasts its lid as the maximum node-id.* */

$$\begin{aligned}\textsc{LeaderProcess}(id, elec, lid, maxid) \stackrel{\text{def}}{=}\ & [maxid = lid]\\ & \quad [elec = 0]\textsc{Node}(id, 0, lid), \sigma!\langle \texttt{ldrMsg}, lid\rangle.\textsc{Node}(id, 0, lid),\\ & [maxid{>}lid]\sigma!\langle \texttt{ldrMsg}, maxid\rangle.\textsc{Node}(id, 0, maxid),\\ & [maxid{<}lid]\sigma!\langle \texttt{ldrMsg}, lid\rangle.\textsc{Node}(id, 0, lid)\end{aligned}$$

/* *In the* ElectionProcess *state a node broadcasts the election message received by its parent and evolves into an* AwaitAck *state, waiting for ack messages.* */

$$\textsc{ElectionProcess}(id, elec, lid, id_p) \stackrel{\text{def}}{=} \sigma!\langle \texttt{elecMsg}, id\rangle.\textsc{AwaitAck}(id, elec, lid, id_p)$$

/* *In the* AwaitAck *state the node receives ack messages and update the maximum node-id as in the* AwaitAckInit *state; the only difference is that when acks end, the process evolves into the* SendAck *state.* */

$$\begin{aligned}\textsc{AwaitAck}(id, elec, lid, id_p) \stackrel{\text{def}}{=}\ & \sigma?(\tilde{x}).[\mathbf{fst}(\tilde{x}) = \texttt{ackMsg}]\\ & \quad [\mathbf{snd}(\tilde{x}) \geq lid]\textsc{AwaitAck}(id, elec, \mathbf{snd}(\tilde{x}), id_p),\\ & \quad \textsc{AwaitAck}(id, elec, lid, id_p),\\ & \textsc{AwaitAck}(id, elec, lid)\\ & + \textsc{SendAck}(id, elec, lid, id_p)\end{aligned}$$

/* *In a* SendAck *state a node may either send (unicast transmission) to its parent node an ack message, with the current maximum node-id, or evolve into a* SendLeader *state if the node disconnects from its parent node. In this case, it reports its current leader.* */

$$\textsc{SendAck}(id, elec, lid, id_p) \stackrel{\text{def}}{=} \sigma!\langle \texttt{ackMsg}, lid\rangle_{id_p}.\textsc{Node}(id, elec, lid) + \textsc{SendLeader}(id, elec, lid)$$

election is triggered at a source node s (for instance, upon departure of its current leader), the node broadcasts an *election message*. Each node, i, other than the source s, designates the neighbour from which it first receives an election message as its parent in the spanning tree. These nodes i then broadcast the received election message. After sending an election message, a node awaits acks from its children in the spanning tree, before sending an ack message to its parent. As we will see shortly, the ack messages sent to the parents contains leader-election information based on the ack messages received from children.

Once the spanning tree has completely grown, the spanning tree shrinks back toward the source. Specifically, once all of i's outgoing election messages have been acknowledged, i sends its pending ack message to its parent node. Tree shrinkage begins at the leaves of the spanning tree, which are parents to no other node. Eventually, each leaf receives ack messages for all election messages it has sent. These leaves thus eventually send their pending ack messages to their respective parents, who in turn send their pending ack messages to their own parents, and so on, until the source node receives all of its pending ack messages. In its pending ack message, a node announces to its parent the node-id and the value of the most-valued-node among all its downstream nodes. Hence the source node eventually has sufficient information to determine the most-valued-node from among all nodes in the network, since the spanning tree spans all network nodes. Once the source node for a computation has received acks from all of its children, it then broadcasts a *leader message* to all nodes announcing the node-id of the most-valued-node.

Our version of the encoding elects as leader the node with the maximum node-id in a connected component at a specific security level σ, called *σ-connected component*. In such a component, neighbouring nodes trust each other at security level (at least) σ.

Definition 7.21 (σ-connected component). *Let M be a network. A subnetwork N of M is said to be a* σ-connected component *of M if*

- *for all $m, n \in \mathsf{nds}(N)$ there is a sequence of nodes $m_1, \ldots, m_k \in \mathsf{nds}(N)$, with $N \equiv \widehat{N} \mid m_1[P_1]_{T_1} \mid \ldots \mid m_k[P_k]_{T_k}$, such that $m{=}m_1$, $n{=}m_k$, and $T_i(m_i, m_{i+1}) \geq \sigma$, for $1 \leq i \leq k{-}1$;*
- *whenever $N \equiv \widehat{N} \mid m[P]_{T_m} \mid n[Q]_{T_n}$ with $T_m(m, n) \geq \sigma$ it holds that $T_n(n, m) \geq \sigma$.*

In Figure 7.1 we provide our encoding of a secure version of the leader election protocol for MANETs. We do not consider probe and reply messages as we can model the effect of disconnection between nodes using the choice operator. Let us explain more in detail the encoding of Figure 7.1. At the beginning, each node can be in one of these two states:

- $\textsc{Source}(id, elec, lid)$, if the node initiates the protocol;
- $\textsc{Node}(id, elec, lid)$, otherwise.

While the protocol is executed, nodes may evolve into one of the following states:

- $\textsc{AwaitAckInit}(id, elec, lid)$, a starting node waits for ack messages;
- $\textsc{SendLeader}(id, elec, lid)$, a node broadcasts a leader message;

- ElectionProcess($id, elec, lid, id_p$), a node rebroadcasts the election message previously received by its parent;
- AwaitAck($id, elec, lid, id_p$), a node waits for ack messages;
- SendAck($id, elec, lid, id_p$), a node sends an ack message to its parent;
- LeaderProcess($id, elec, lid, maxid$), a node sets its leader parameter to the value received.

The meaning of the parameters of the above states is the following: id is the name of the node; $elec$ indicates whether the node is part of the election process, thus, $elec = 1$ if the node is participating in the election process, $elec = 0$ otherwise; lid represents the node's knowledge of the leader; id_p is the name of the parent node; $maxid$ is maximum node-id in the spanning tree rooted at id.

A node may send and/or receive election, ack or leader messages. These messages are pairs of the following shape:

- `elecMsg`, id meaning an election message sent by node id;
- `ackMsg`, lid meaning an ack message where lid is the current leader at that stage;
- `ldrMsg`, lid meaning a leader message where lid is the current node's knowledge of the leader.

A starting node begins the protocol in the Source($id, 0, lid$) state, with $lid = id$. In this state, it broadcasts the message $\langle$`elecMsg`$, lid\rangle$ moving into the AwaitAckInit($id, 1, lid$) state, waiting for the ack message. In this state the starting node may receive ack messages of the form $\langle$`ackMsg`$, maxid\rangle$. When this happens, the node checks the $maxid$ variable contained in the ack message. If $maxid \geq lid$ then the node evolves into the AwaitAckInit($id, elec, maxid$) state to record the $maxid$ value, otherwise it remains in AwaitAckInit($id, elec, lid$), waiting for other ack messages. In the state AwaitAckInit($id, elec, lid$) the node may also nondeterministically evolves into the SendLeader($id, elec, lid$) when it has received all acks from its neighbours. In the SendLeader($id, elec, lid$) state a node broadcasts a leader message $\langle$`ldrMsg`$, lid\rangle$ and evolves into the Node($id, 0, lid$) state, waiting for other leader messages sent by its neighbours.

All the other nodes in the network begin the protocol in the Node($id, 0, lid$) state. In this state, they may receive an election message $\langle$`elecMsg`$, id_p\rangle$ or a leader message $\langle$`leaderMsg`$, maxid\rangle$. In the first case they evolve into the state ElectionProcess($id, 1, lid, id_p$), where id_p records the id of their parent node, contained in the election message. In the second case they evolve into the LeaderProcess($id, elec, lid, maxid$) state, where $maxid$ is the leader id contained in the leader message. A node in the ElectionProcess($id, elec, lid, id_p$) state broadcasts an election message containing its node id and then evolves into the state AwaitAck($id, elec, lid, id_p$), waiting for acks. A node in the AwaitAck($id, elec, id, id_p$) state may either receive ack messages of the form $\langle$`ackMsg`$, maxid\rangle$ or evolve into the SendAck($id, elec, lid, id_p$) state to model that all ack messages have been received. When the node receives an ack, it stores the maximum node-id received with the ack, checking if $maxid \geq lid$. A node in the SendAck($id, elec, lid, id_p$) state may send to its parent node id_p an ack message of the form $\langle$`ackMsg`$, lid\rangle$ with the current maximum node-id, and goes into the

NODE$(id, elec, lid)$ state; otherwise the node may evolve into the SENDLEADER-$(id, elec, lid)$ state. This last state models the case when a node is disconnected from its parent node. In this case, the node reports its current leader.

A node in the LEADERPROCESS$(id, elec, lid, maxid)$ basically propagates the received leader message by setting its *lid* parameter to the *maxid* values received in the leader message. The most interesting case is when $maxid < lid$. In this case, either the node was not part of the election process or it did not report the ack message to its parent nodes, for example because it was disconnected. In both case it broadcasts its *lid* as the maximum node-id sending a leader message $\langle \texttt{ldrMsg}, lid \rangle$ and evolves into NODE$(id, 0, lid)$.

Here, we report an example of the protocol.

Example 7.22. Let M be the following network:

$$M \stackrel{\text{def}}{=} l[\text{SOURCE}(l, 0, l)]_{T_l} \mid m[\text{NODE}(m, 0, m)]_{T_m} \mid n[\text{NODE}(n, 0, n)]_{T_n}$$

with $l > m > n$ and $T_l(l, m) = T_m(m, l) = T_m(m, n) = T_n(n, m) \geq \sigma$. Here, we report the evolution M while running the protocol.

$$\begin{aligned} M \xrightarrow{l!\langle \texttt{elecMsg}, l \rangle \triangleright m}_{\sigma} \; & l[\text{AWAITACKINIT}(l, 1, l)]_{T_l} \mid \\ & m[\text{ELECTIONPROCESS}(m, 1, m, l)]_{T_m} \mid \\ & n[\text{NODE}(n, 0, n)]_{T_n} \\ \stackrel{\text{def}}{=} \; & M_1 \; . \end{aligned}$$

Node l starts the protocol broadcasting the election message $\langle \texttt{elecMsg}, l \rangle$, and evolving into the state AWAITACKINIT$(l, 1, l)$. Only node m receives the election message and evolves into the state ELECTIONPROCESS$(m, 1, m, l)$. Node m has marked l as its parent node.

$$\begin{aligned} M_1 \xrightarrow{m!\langle \texttt{elecMsg}, m \rangle \triangleright \{l, n\}}_{\sigma} \; & l[\text{AWAITACKINIT}(l, 1, l)]_{T_l} \mid m[\text{AWAITACK}(m, 1, m, l)]_{T_m} \mid \\ & n[\text{ELECTIONPROCESS}(n, 1, n, m)]_{T_n} \\ \stackrel{\text{def}}{=} \; & M_2 \; . \end{aligned}$$

Node m broadcasts the election message $\langle \texttt{elecMsg}, m \rangle$, and evolves into the state AWAITACK$(m, 1, m, l)$, waiting for acks. Node l ignores the message and remains in the state AWAITACKINIT$(l, 1, l)$, whereas node n receives the message and evolves into the state ELECTIONPROCESS$(n, 1, n, m)$. Again the last parameter m indicates the parent node of n.

$$\begin{aligned} M_2 \xrightarrow{n!\langle \texttt{elecMsg}, n \rangle \triangleright m}_{\sigma} \; & l[\text{AWAITACKINIT}(l, 1, l)]_{T_l} \mid m[\text{AWAITACK}(m, 1, m, l)]_{T_m} \mid \\ & n[\text{AWAITACK}(n, 1, n, m)]_{T_n} \\ \stackrel{\text{def}}{=} \; & M_3 \; . \end{aligned}$$

Node n broadcasts the election message $\langle \texttt{elecMsg}, n\rangle$ and then it evolves into the state $\textsc{AwaitAck}(n, 1, n, m)$, waiting for acks. Node m ignores the message and remains in its state.

$$\begin{aligned} M_3 \xrightarrow{n!\langle \texttt{ackMsg},n\rangle \triangleright m}_{\sigma} \; & l[\textsc{AwaitAcInit}(l,1,l)]_{T_l} \mid m[\textsc{AwaitAck}(m,1,m,l)]_{T_m} \mid \\ & n[\textsc{Node}(n,1,n)]_{T_n} \\ \stackrel{\text{def}}{=} \; & M_4 \ . \end{aligned}$$

As n has no children, it will not receive acks. Thus, it will eventually evolve into the state $\textsc{SendAck}(n, 1, n, m)$ sending the ack message $\langle \texttt{ackMsg}, n\rangle$ to its parent node m. This message contains the current leader of node n, that is n itself. After the sending, n evolves into the state $\textsc{Node}(n, 1, n)$, waiting for leader messages.

$$\begin{aligned} M_4 \xrightarrow{m!\langle \texttt{ackMsg},m\rangle \triangleright l}_{\sigma} \; & l[\textsc{AwaitAckInit}(l,1,l)]_{T_l} \mid m[\textsc{Node}(m,1,m)]_{T_m} \mid \\ & n[\textsc{Node}(n,1,n)]_{T_n} \\ \stackrel{\text{def}}{=} \; & M_5 \ . \end{aligned}$$

When m receives the message $\langle \texttt{ackMsg}, n\rangle$, it checks whether n is greater than its current leader m. As $m > n$, m sends the ack message $\langle \texttt{ackMsg}, m\rangle$ to its parent l and evolves into the state $\textsc{Node}(m, 1, m)$. The message $\langle \texttt{ackMsg}, m\rangle$ contains the current leader of m, that is m itself.

$$\begin{aligned} M_5 \xrightarrow{l!\langle \texttt{ldrMsg},l\rangle \triangleright m}_{\sigma} \; & l[\textsc{Node}(l,0,l)]_{T_l} \mid m[\textsc{LeaderProcess}(m,1,m,l)]_{T_m} \mid \\ & n[\textsc{Node}(n,1,n)]_{T_n} \\ \stackrel{\text{def}}{=} \; & M_6 \ . \end{aligned}$$

When l receives the message $\langle \texttt{ackMsg}, n\rangle$, it checks whether m is greater than its current leader l. As $l > m$, node l broadcasts the leader message $\langle \texttt{ldrMsg}, l\rangle$ and evolves into the state $\textsc{Node}(l, 0, l)$. Node m receives the leader message and evolves into the state $\textsc{LeaderProcess}(m, 1, m, l)$.

$$\begin{aligned} M_6 \xrightarrow{m!\langle \texttt{ldrMsg},l\rangle \triangleright \{l,n\}}_{\sigma} \; & l[\textsc{Node}(l,0,l)]_{T_l} \mid m[\textsc{Node}(m,0,l)]_{T_m} \mid \\ & n[\textsc{LeaderProcess}(n,1,n,l)]_{T_n} \\ \stackrel{\text{def}}{=} \; & M_7 \ . \end{aligned}$$

Node m checks whether l is greater than its current leader m. As $l > m$, then m broadcasts the leader message $\langle \texttt{ldrMsg}, l\rangle$ with l as its current leader and evolves into the state $\textsc{Node}(m, 0, l)$. When l receives this message, as it is in $\textsc{Node}(l, 0, l)$ state, it has simply to check whether the received leader corresponds to its current leader. This is the case, then it remains into $\textsc{Node}(l, 0, l)$ state. When n receives the leader message $\langle \texttt{ldrMsg}, l\rangle$, it evolves into the state $\textsc{LeaderProcess}(n, 1, n, l)$.

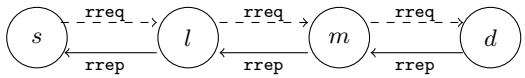
$$M_7 \xrightarrow{n!\langle \texttt{ldrMsg}, l\rangle \triangleright m}_{\sigma} l[\textsc{Node}(l,0,l)]_{T_l} \mid m[\textsc{Node}(m,0,l)]_{T_m} \mid n[\textsc{Node}(n,0,l)]_{T_n} .$$

Finally, node n verifies that l is greater than its current leader n. Thus, l becomes the current leader of n that broadcasts the leader message $\langle \texttt{ldrMsg}, l\rangle$ with current leader l, evolving into the state $\textsc{Node}(n,0,l)$. The leader message sent by n is received by m that verifies that l is already its leader. So, it remains in the state $\textsc{Node}(m,0,l)$.

7.9.2 The endairA Routing Protocol

Figure 7.2 An example of the operation and the messages of endairA

$$\begin{array}{rcll} s & \longrightarrow & * & : \ \texttt{rreq}, s, d, nonce, [\ ] \\ l & \longrightarrow & * & : \ \texttt{rreq}, s, d, nonce, [l] \\ m & \longrightarrow & * & : \ \texttt{rreq}, s, d, nonce, [l,m] \\ d & \longrightarrow & m & : \ \{\texttt{rrep}, s, d, nonce, [l,m]\}_{K_{d-}} \\ m & \longrightarrow & l & : \ \{\{\texttt{rrep}, s, d, nonce, [l,m]\}_{K_{d-}}\}_{K_{m-}} \\ l & \longrightarrow & s & : \ \{\{\{\texttt{rrep}, s, d, nonce, [l,m]\}_{K_{d-}}\}_{K_{m-}}\}_{K_{l-}} \end{array}$$



We described in detail in Section 2.2.3 the routing protocols for wireless systems. We remember that ad hoc networks rely on multi-hop wireless communications where nodes have essentially two roles: (i) acting as end-systems, and (ii) performing routing functions. A routing protocol is used to determine the appropriate paths on which data should be transmitted in a network. Routing protocols for wireless systems can be classified into *topology-based* and *position-based*. Topology-based protocols rely on traditional routing concepts, such as maintaining routing tables or distributing link-state information. Position-based protocols use information about the physical locations of the nodes to route data packets to their destinations. Topology-based protocols can be divided into *proactive* and *reactive* protocols. Proactive routing protocols try to maintain consistent routing information within the system at any time. In reactive routing protocols, a route is established between a source and a destination only when it is needed. For this reason, reactive protocols are also called *on-demand* protocols. Examples of proactive routing protocols for MANETs are OLSR [65] and DSDV [187], while on-demand protocols are DSR [129] and AODV [186].

Initial work on routing in ad hoc networks has considered only the problem of providing efficient mechanisms for finding paths, without considering security issues. However, due to the lack of physical protection, some of the routers could be corrupted affecting the routing paths. Obviously, this can have undesirable

Figure 7.3 The encoding of the endairA protocol

$/^*$ *Starting node broadcasts a request message and evolves into the* AwaitReply *state waiting for reply.* $^*/$

$$\textsc{Source}(id_s, T_s, id_d) \stackrel{\text{def}}{=} \sigma!\langle \mathtt{rreq}, id_s, id_d, [\ ]\rangle.\textsc{AwaitReply}(id_s, T_s, id_d)$$

$/^*$ *The initiator node in the* AwaitReply *state receives a reply message; if the reply is successfully returned, the node accepts the route, evolving into the state* RouteSuccess. $^*/$

$$\begin{array}{l}
\textsc{AwaitReply}(id_s, T_s, id_d) \stackrel{\text{def}}{=} \\
\quad \sigma?(pkt, sigList).[T_s(id_s, sigList) \geq \sigma] \\
\qquad\qquad let\ (y_1, y_2, y_3, y_4) = get(pkt, sigList)\ in \\
\qquad\qquad\quad [y_1 = \mathtt{rrep}] \\
\qquad\qquad\qquad [(\mathbf{tail}(sigList) = y_4) \wedge (\mathbf{head}(sigList) = y_3 = id_d) \wedge (y_2 = id_s)] \\
\qquad\qquad\qquad\quad \textsc{RouteSuccess}(id_s, id_d, y_4), \\
\qquad\qquad\qquad \textsc{Source}(id_s, T_s, id_d), \\
\qquad\qquad\quad \textsc{AwaitReply}(id_s, T_s, id_d) \\
\qquad\qquad else\ \textsc{AwaitReply}(id_s, T_s, id_d), \\
\qquad\quad \textsc{AwaitReply}(id_s, T_s, id_d) \\
\quad + \ \textsc{Source}(id_s, T_s, id_d)
\end{array}$$

$/^*$ *Intermediate nodes may receive a request or a reply message.* $^*/$

$$\begin{array}{l}
\textsc{Node}(id, T) \stackrel{\text{def}}{=} \sigma?(req, id_s, id_d, path).\textsc{RRequest}(id, T, req, id_s, id_d, path) \\
\qquad\qquad + \ \sigma?(pkt, sigList).\textsc{RReply}(id, T, pkt, sigList)
\end{array}$$

$/^*$ *A node receiving a request message controls if it is the intended destination or not.* $^*/$

$$\begin{array}{l}
\textsc{RRequest}(id, T, req, id_s, id_d, path) \stackrel{\text{def}}{=} [req = \mathtt{rreq}] \\
\qquad\qquad\qquad\qquad\qquad [id_d = id] \\
\qquad\qquad\qquad\qquad\qquad\quad \textsc{SendReply}(id, T, path, id_s), \\
\qquad\qquad\qquad\qquad\quad \sigma!\langle \mathtt{rreq}, id_s, id_d, path \uplus id\rangle.\textsc{Node}(id, T) \\
\qquad\qquad\qquad\qquad \textsc{Node}(id, T)
\end{array}$$

$/^*$ *After receiving the request, the destination unicasts a signed reply back along the reverse path to the source.* $^*/$

$$\textsc{SendReply}(id_d, T, path, id_s) \stackrel{\text{def}}{=} \sigma!\langle \{\mathtt{rrep}, id_s, id_d, path\}_{K_{id_d -}}, id_d\rangle_{\mathbf{last}(path)} + \ \textsc{Node}(id_d, T)$$

$/^*$ *A node receiving a reply message controls if the reply is successfully returned.* $^*/$

$$\begin{array}{l}
\textsc{RReply}(id, T, pkt, sigList) \stackrel{\text{def}}{=} \\
\quad [T(id, sigList) \geq \sigma] \\
\qquad let\ (y_1, y_2, y_3, y_4) = get(pkt, sigList)\ in \\
\qquad\quad [y_1 = \mathtt{rrep}] \\
\qquad\qquad [(id \in y_4) \wedge (\mathbf{tail}(sigList) = \mathbf{succ}(y_4, id)) \wedge (\mathbf{head}(sigList) = y_3)] \\
\qquad\qquad\quad \textsc{FwdReply}(id, T, pkt, sigList, \mathbf{pred}(y_4, id, y_2)), \\
\qquad\qquad \textsc{Node}(id, T), \\
\qquad\quad \textsc{Node}(id, T) \\
\qquad else\ \textsc{Node}(id, T), \\
\quad \textsc{Node}(id, T)
\end{array}$$

$/^*$ *If the reply has been successfully returned, a node forwards the reply back to its previous node, adding its signature.* $^*/$

$$\textsc{FwdReply}(id, T, pkt, sigList, id_n) \stackrel{\text{def}}{=} \sigma!\langle \{pkt\}_{K_{id-}}, sigList \uplus id\rangle_{id_n} + \ \textsc{Node}(id, T)$$

effects on the operations of the network. A number of secure routing protocols for MANETs have been proposed such as: SRP [184], Ariadne [121], endairA [9], SAODV [238], and ARAN [209]. In this section, we use our calculus to formalise the on-demand routing protocol *endairA*. In [9] Ács, Buttyán and Vajda have presented a formal framework, in which security of routing is precisely defined, and which can serve as the basis for rigorous security analysis of routing protocols both for ad hoc and sensor networks. They also have developed the endairA protocol for mobile ad hoc networks, and the framework is tested by proving the security of this protocol.

The endairA protocol uses digital signatures. As usual, given a node n, we write K_{n^+} and K_{n-} to mean the public key and the private key of n, respectively. Moreover, given a message $\tilde{v}$, we write $\{\tilde{v}\}_{K_{n-}}$ meaning the message $\tilde{v}$ signed by n. In the following, we refer to *node-id* or to node *identifier* to mean the name of the node.

In Figure 7.2 we report a scheme of the endairA protocol with four nodes: a source s, a destination d and two intermediate nodes l and m. We also provide a graphical representation of the message flow, where the dashed arrows denote the broadcast of *route request messages*, while the continuous arrows denote the unicast sending of *route reply messages*. The protocol works as follows. The source s broadcasts a route request message of the form $\langle \texttt{rreq}, s, d, nonce, [\,] \rangle$. *Nonce* is a randomly generated request identifier, that helps in detecting replay messages, whereas $[\,]$ is the list of intermediate nodes that have successfully received the route request message. At the beginning this list is empty. When the intermediate node l receives the route request for the first time, it broadcasts the message $\langle \texttt{rreq}, s, d, nonce, [l] \rangle$, appending its node-id in the list. When m receives the route request sent by l, it broadcasts a route request message of the form $\langle \texttt{rreq}, s, d, nonce, [l, m] \rangle$. When the request reaches the destination, the node d replies with a unicast message for m (the last node in the route-list) of the form $\{\texttt{rrep}, s, d, nonce, [l, m]\}_{K_{d-}}$. When m receives this message, it verifies that the digital signatures in the reply (the signature of the destination in this case) are valid, and tries to extract the content of the signed message. Then, it verifies that its node-id is in the node list of the reply, and that the previous identifier, l in this case (or that of the source if there is no previous identifier in the node list), and the following identifier, d in this case, belong to neighbouring nodes. Each intermediate node also verifies that the digital signatures of the reply messages received. If these verifications fail, then the reply message is dropped. Otherwise, it is signed by the intermediate node and passed to the next node on the route (towards the source). So, in our case, at the end of the protocol, node l sends to the source s the following unicast message: $\{\{\{\texttt{rrep}, s, d, nonce, [l, m]\}_{K_{d-}}\}_{K_{m-}}\}_{K_{l-}}$. When the source receives the route reply, after it has verified the signatures and extracted the content of the message, it verifies if the first identifier in the route is a neighbour, and that the reply message is the expected one. If some of these verifications fail the message is dropped.

When writing the endairA protocol in our calculus, we consider a few simplifications. Our encoding is actually a secure variant of the original protocol as paths are associated with security levels as they are composed only by trusted nodes. For simplicity, we eliminate the field *nonce* in our messages, as it only helps in

detecting replay attacks and here we do not explicitly deal with attacks. In order to prevent invalid routes, in the original protocol, an intermediate node n_i verifies that the previous node n_{i-1} and the following node n_{i+1} in the accumulated route are neighbouring nodes. In our encoding, we simply verify the nodes in the accumulated route have the appropriate security level. This ensures that also reply messages are forwarded on paths of nodes at security level at least σ. This is guaranteed for free by our operational semantics.

For convenience, we extend the syntax of the matching construct as follows:

$$[(\tilde{u}_1 \,\mathtt{op}\, \tilde{u}'_1)\mathtt{lc}\ldots\mathtt{lc}(\tilde{u}_k \,\mathtt{op}\, \tilde{u}'_k)]P, Q$$

where `lc` is a binary logical operator. We also introduce the following destructor construct

$$\mathit{let}\ \tilde{x}{=}g(u_1, u_2, \ldots, u_k)\ \mathit{in}\ P\ \mathit{else}\ Q$$

where g is a destructor function. A destructor function is defined using a finite set of equations. When the application of a destructor $g(u_1, u_2, \ldots, u_k)$ does not match any of its equations, we write $g(\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_k) = \bot$. We add the following two rules in the operational semantics:

$$\text{(LetIn)}\ \frac{g(u_1, \ldots, u_k) = \tilde{v} \quad n[\{^{\tilde{v}}/_{\tilde{x}}\}P]_T \xrightarrow{\lambda}_\rho n[P']_{T'}}{n[\mathit{let}\ \tilde{x}{=}g(\tilde{u}_1, \ldots, \tilde{u}_k)\ \mathit{in}\ P\ \mathit{else}\ Q]_T \xrightarrow{\lambda}_\rho n[P']_{T'}}$$

$$\text{(LetElse)}\ \frac{g(u_1, \ldots, u_k) = \bot \quad n[Q]_T \xrightarrow{\lambda}_\rho n[Q']_{T'}}{n[\mathit{let}\ \tilde{x}{=}g(\tilde{u}_1, \ldots, \tilde{u}_k)\ \mathit{in}\ P\ \mathit{else}\ Q]_T \xrightarrow{\lambda}_\rho n[Q']_{T'}}\ .$$

For our purposes, we only need a destructor $\mathit{get}(\cdot)$ that takes in input a signed message and a list of node-ids and returns the content of the message if the signatures of the message correspond to the node-ids in the list, $\bot$ otherwise. For instance,

$$\mathit{get}(\{\ldots\{\{\tilde{u}\}_{K_{n_1-}}\}_{K_{n_2-}}\}\ldots\}_{K_{n_k-}}, n_1, n_2, \ldots, n_k) = \tilde{u},$$

whereas $\mathit{get}(\{\ldots\{\{\tilde{u}\}_{K_{n_1-}}\}_{K_{n_2-}}\}\ldots\}_{K_{n_k-}}, m_1, m_2, \ldots, m_k) = \bot$, if $m_i \neq n_i$, for some i. We assume that the destructor *get* is capable to get the corresponding public keys from node-ids. This operation successes only if the list contains exactly all nodes that have signed the message.

In the endairA protocol reply messages need to be signed by all nodes in the route. Thus, for convenience, our reply messages contains the list of nodes that have signed the messages, denoted with the variable *sigList*. We assume that if a node $m[P]_T$ knows a node n, that is $T(m, n)$ is defined, then m knows the public key of n. We remember that in our calculus we assume the presence of a hierarchical key generation and distribution protocol. This implies that a signature is produced with a private key at a specific security level. Thus, if m wants to verify the validity of a signature produced by a node n at security level ρ, it simply has to control whether $T(m, n) \geq \rho$.

We use a number of notations. Let $h = n, l, \ldots, q$ be a list of node-ids and m a node; we sometimes write $T(m, h) \geq \rho$ to mean $T(m, n) \geq \rho \wedge T(m, l) \geq \rho \wedge \ldots \wedge T(m, q) \geq \rho$. Let h be a list and v a value; we write $h \uplus v$ for the list where

v has been appended to h. We assume standard functions on lists such as $\mathbf{head}(\cdot)$ and $\mathbf{tail}(\cdot)$, with the convection that $\mathbf{head}(h) = \mathbf{tail}(h)$, if h is composed by just one value. We also define two ad hoc functions $\mathbf{pred}(\cdot,\cdot,\cdot)$ and $\mathbf{succ}(\cdot,\cdot)$. More precisely, the function $\mathbf{succ}(\cdot,\cdot)$ takes in input a list of node-ids h and a node-id id and returns the list of node-ids that follows id in h; this list can obviously be empty. Whereas $\mathbf{pred}(\cdot,\cdot,\cdot)$ takes in input a list h of node-ids and two node-ids id and id', and returns the node-id that precedes id in h it it exists, otherwise it returns id'. Thus, if $h = n, l, \ldots, q$ is a list of node-ids and s a node-id; then $\mathbf{pred}(h, l, s) = n$, whereas $\mathbf{pred}(h, n, s) = s$.

In Figure 7.3 we provide the encoding of the endairA protocol in our calculus. Let us explain it in some detail.

At the beginning, each node can be in one of these two states:

- $\textsc{Source}(id_s, T_s, id_d)$, when a (source) node initiates the protocol; in this state the node broadcasts a request message;
- $\textsc{Node}(id, T)$, when a node participate receiving a request or reply message.

While the protocol is executed, nodes may evolve into one of the following states:

- $\textsc{AwaitReply}(id_s, T_s, id_d)$, the initiator node waits for the reply message;
- $\textsc{RouteSuccess}(id_s, id_d, path)$, the source node has accepted the route;
- $\textsc{RRequest}(id, T, req, id_s, id_d, path)$, an intermediate node receives a request message;
- $\textsc{SendReply}(id_d, T, path, id_s)$, the destination node sends the reply message;
- $\textsc{RReply}(id, T, pkt, sigList)$, an intermediate node receives a reply message;
- $\textsc{FwdReply}(id, T, pkt, sigList, id_n)$, an intermediate node forwards a reply message.

The source node s begins in the state $\textsc{Source}(id_s, T_s, id_d)$, where id_s and id_d are the ids of the source and the destination, respectively, while T_s is the trust table of s. In this state, the source node broadcasts a route request message of the form $\langle \texttt{rreq}, id_s, id_d, [\,] \rangle$. The last element of the message is the accumulated route so far; at the beginning this list is empty. After this transmission, the node evolves into the state $\textsc{AwaitReply}(id_s, T_s, id_d)$, waiting for a reply message. In particular the source nodes waits for a pair $\langle pkt, sigList \rangle$ composed by a signed reply packet and the list of ids of the nodes that have signed the packet. When the source receives a reply message, it first verifies the signatures. If the verification fails, the source node returns into the state $\textsc{AwaitReply}(id_s, T_s, id_d)$, otherwise it tries to extract the content of pkt. This packet should contain four values: the tag `rrep`, the node-id of the source, the node-id of the destination, and the accumulated route. If the initiator cannot extract these informations it returns into the state $\textsc{AwaitReply}(id_s, T_s, id_d)$, otherwise it starts a number of checks. First, it verifies if it has received a reply packet; then it verifies if the signatures correspond to the ids appearing in the accumulated route; finally it verifies if the ids of the source and the destination are the expected ones. If all these checks are successful the initiator accepts the route, evolving into the state $\textsc{RouteSuccess}(id_s, id_d, path)$, meaning that the route from id_s to id_d contained in $path$ has been accepted, otherwise the reply is dropped and the node returns into the state $\textsc{Source}(id_s, T_s)$. The reply may never arrive at the source; in this case the source can nondeterministically return into the state $\textsc{Source}(id_s, T_s)$, starting again the protocol.

The other nodes taking part in the protocol may be either intermediate nodes or the destination node. In both cases, they begin the protocol in the state $\textsc{Node}(id, T)$, where id is the identity of the node and T is its trust table. In this state, the node expects to receive either a route request message or a reply message.

When a node receive a route request message it first verifies if it is the destination of the route request. If this is the case, the node moves into the state $\textsc{SendReply}(id_d, T, path, id_s)$, otherwise it rebroadcasts the request message, after adding its node-id in the route path $path$, returning into the state $\textsc{Node}(id, T)$. In $\textsc{SendReply}(id_d, T, path, id_s)$, the destination node prepares the reply message. More precisely, it creates the list $sigList$ of signing nodes, containing only its id, and sends the unicast message $\langle \{\mathtt{rrep}, id_s, id_d, path\}_{K_{id_d-}}, [id_d] \rangle$ to the last node in $path$. The choice operator allows to avoid deadlocks in case the last node in $path$ becomes suddenly disconnected.

When a node in the state $\textsc{Node}(id, T)$ receives a reply containing a pair $\langle pkt, sigList \rangle$, it evolves into the state $\textsc{RReply}(id, T, pkt, sigList)$. Here, it carries out the same checks appearing in the state $\textsc{AwaitReply}$ verifying the signatures, and trying to extract the content of pkt. If everything is fine, the node evolves into the state $\textsc{FwdReply}(id, T, pkt, sigList, id_n)$, otherwise the node returns into the state $\textsc{Node}(id, T)$. The parameter pkt records the original signed message previously received, whereas the parameter id_n records the previous node-id in $path$ to which the node has to send back the reply. In the state $\textsc{FwdReply}(id, T, pkt, sigList, id_n)$ the node signs the reply message and adds its id in the list $sigList$. Thus, it sends to id_n the message $\langle \{pkt\}_{K_{id-}}, sigList \uplus id \rangle$. Again, the choice operator in this state allows to avoid deadlocks in case of disconnections of id_n.

Here we report an example of how the protocol works.

Example 7.23. Let M be the following network:

$$M \stackrel{\text{def}}{=} l[\textsc{Source}(l, T_l, n)]_{T_l} \mid m[\textsc{Node}(m, T_m)]_{T_m} \mid n[\textsc{Node}(n, T_n)]_{T_n}$$

with $T_l(l, m) = T_l(l, n) = T_m(m, l) = T_m(m, n) = T_n(n, m) \geq \sigma$. For convenience we define: $v_1 := \{\mathtt{rrep}, l, n, m\}_{K_{n-}}$ and $v_2 := \{\{\mathtt{rrep}, l, n, m\}_{K_{n-}}\}_{K_{m-}}$. Here, we report the evolution of M while running the endairA protocol.

$$\begin{aligned} M \xrightarrow{l!\langle \mathtt{rreq}, l, n, [\,] \rangle \triangleright \{m,n\}}_{\sigma} \; & l[\textsc{AwaitReply}(l, T_l, n)]_{T_l} \mid \\ & m[\textsc{RRequest}(m, T_m, \mathtt{rreq}, l, n, [\,])]_{T_m} \mid \\ & n[\textsc{Node}(n, T_n)]_{T_n} \\ \stackrel{\text{def}}{=} \; & M_1 \ . \end{aligned}$$

Node l starts the protocol in the state $\textsc{Source}(l, T_l, n)$, broadcasting the message $\langle \mathtt{rreq}, l, n, [\,] \rangle$ and evolving into the state $\textsc{AwaitReply}(l, T_l, n)$, waiting for a reply. The accumulated route is still empty. Only node m receives the message and evolves into $\textsc{RRequest}(m, T_m, \mathtt{rreq}, l, n, [\,])$.

$$M_1 \xrightarrow{m!\langle \mathtt{rreq},l,n,[m]\rangle \triangleright \{l,n\}}_{\sigma} l[\textsc{AwaitReply}(l,T_l,n)]_{T_l} \mid m[\textsc{Node}(m,T_m)]_{T_m} \mid n[\textsc{SendReply}(n,T_n,m,l)]_{T_n} \stackrel{\text{def}}{=} M_2 \ .$$

Node m broadcasts the message $\langle \mathtt{rreq}, l, n, [m]\rangle$, where $[m]$ is the accumulated route so far. Then, node m evolves into $\textsc{Node}(m, T_m)$, waiting for a reply. Node l ignores the message sent by m and remains in the state $\textsc{AwaitReply}(l, T_l, n)$, whereas node n receives the message, it verifies to be the destination node and it evolves into the state $\textsc{SendReply}(n, T_n, m, l)$.

$$M_2 \xrightarrow{n!\langle v_1,[n]\rangle \triangleright m}_{\sigma} l[\textsc{AwaitReply}(l,T_l,n)]_{T_l} \mid m[\textsc{FwdReply}(m,T_m,v_1,m,l)]_{T_m} \mid n[\mathsf{nil}]_{T_n} \stackrel{\text{def}}{=} M_3 \ .$$

Node n sends the reply message, together with the list of nodes signing the reply, back to m. Node m receives this message, verifies the signature and extracts the content of v_1. Then, it verifies that its name is in the accumulated route carried by the received message and that the signature corresponds to the destination node. Thus, it evolves into $\textsc{FwdReply}(m, T_m, v_1, m, l)$.

$$M_3 \xrightarrow{m!\langle v_2,[n,m]\rangle \triangleright l}_{\sigma} l[\textsc{RouteSuccess}(l,n,m)]_{T_l} \mid m[\mathsf{nil}]_{T_m} \mid n[\mathsf{nil}]_{T_n}.$$

Node m sends the reply message v_2, together with the list of nodes signing the reply, back to l. Node l receives the message and verifies the signatures of m and n and that they correspond to the node in the path and to the destination, respectively. It also verifies that the node-ids of the source and of the destination are the expected ones. Then it accepts the route and evolves into the state $\textsc{RouteSuccess}(l, n, m)$.

7.10 Related Work

In this section, we describe some related work according to the area in which they have been developed.

Trust Models

Besides the models we described in Section 3.4.1, here we cite the work of Komarova and Riguidel [139]. They have proposed a centralised trust-based access control mechanism for ubiquitous environments. The goal of their work is to give a service provider or a resource holder the opportunity to evaluate the trustworthiness of each potential client, react to the client's activity by adapting access policies to the actual risk level, and derive user's access rights from his previous

behaviour, recommendations from third party and the actual circumstances. It is supposed that the system is able to observe and to log the activity of each client and use this information to estimate correspondent trust values.

In contrast to this approach, we proposed a completely decentralised trust model in which each node is able (by itself and without the support of a service provider or a resource holder) to perform a trust establishment process in order to give a valuation of the neighbours' behaviour in terms of security levels. Thus observation of each user, log of the activities and estimation of trust values are all actions performed by each single node and they are guaranteed by the presence of a local trust manager component. Moreover we assumed a local policy that is "static" because it does not adapted according to actual environment conditions.

Security in Multilevel Systems

The problem of protecting information and resources in multilevel systems [28] has been extensively studied using different approaches. Some methodologies analysed in Chapter 5 have been also successfully applied in this area. Bodei et al. [52] applied flow analysis techniques. Reitman and Andrews [200] have used axiomatic logic. Smith and Volpano in [229], Boudol and Castellani [54] and Heintz and Riecke in [108] have instead focused on type systems for prototypical programming languages.

Information flow properties aim at controlling the way in which information may flow among different entities. They have been first proposed as a means to ensure confidentiality. Excellent surveys are in [83,204]. As we described in Section 5.3, the *Non Deducibility on Composition* (NDC) property was introduced by Focardi and Gorrieri [81] as a non-interference property based on trace semantics: systems are deemed to be interference free if their trace sets, sequences of actions labelled *high* or *low*, satisfy some properties. The notion of *Bisimulation Non Deducibility on Composition* (BNDC) [81] is based on bisimulation rather than trace semantics. More recently, Gorrieri et al. [99] have proposed a framework for the specification of information flow properties for distributed systems, based on the NDC and BNDC properties.

Crafa and Rossi [70] have introduced a notion of *controlled information release* for a typed version of the π-calculus extended with *declassified actions*. The controlled information release property scales to non-interference when downgrading is not allowed. They have provided various characterisations of *controlled release property*, based on typed behavioural equivalence, parameterised on security levels, to model observers at a certain security level. Hennessy [110] has proposed a typed version of the asynchronous π-calculus in which I/O types are associated to security levels. A typed version of *may* and *must* equivalences are used to prove a non-interference result.

On the contrary of the approaches in [110] and [70], we did not consider a type systems for channel names (we remember that we have a unique channel) and for transmitted values. Our definition of δ-high level network is reminiscent of the formalisation of σ-high level source process in [70], with the difference that we do not consider declassified primitives. Moreover in [70] a σ-high level source is not prevented from communicating σ-low values along σ-high channels, as this

is allowed to well-typed processes and the security level of transitions is not established by the process but it can vary according to the security level of the channel. In our calculus a δ-high level network can netheir receive nor transmit δ-low values as the security level of transitions is directly related to the syntax of processes. As in [110] and [70], our formalisation of non-interference result is reminiscent of the non-interference BNDC property of [81]. However in CTAN only the general case $M \mid K \approx_\delta N \mid H$ holds and it is not true that $M \approx_\delta N \mid H$. This happens because in M and N could not still know that some nodes in H are corrupted and then they send them anyway a message. These nodes may be potential observers of the action from M but not of the same action from $N \mid K$.

Process Calculi

We described some calculi for wireless networks [92, 94, 163, 167, 178, 214] and their relation with our work in Section 6.9. In Section 2.2.3 we described how some of them [94, 178, 214] has been applied to the analysis of routing protocols.

Our mobility model is inspired by the one adopted in [92] which is in turn based on the approach of [178]. We thus transfer topology concepts completely to the semantics (similar to CBS$^\sharp$ [178] and RBPT [92]). This means that each state represents a set of valid topologies, and a network can be at any of those topologies at any time. In addition, it is the network behaviour that defines a set of valid topologies under which such behaviour is correct, rather than the underlying topology dictates the network behaviour. A valid topology is a set of connectivity relations between nodes. The advantages of this approach are that it should easily enable the modelling of ad hoc protocol assumptions and the reduction of the number of topologies that should be considered in an eventual verification. The approach of CBS$^\sharp$, in the definition of topology and topology changes, is the same as ours and of RBPT, even if the LTS and equivalence relations are completely different. In CBS$^\sharp$, in any state, a transition to the next state is examined for all possible valid configurations, those satisfying a topology invariant, while in our approach a transition is examined for a subset of valid configurations (only the connections involved in a broadcast are examined).

None of the calculi mentioned above deal with trust. CTAN is the first calculus to contain a trust model for wireless networks then, while the trust model has been developed to accomplish the most important features of trust management systems for MANETs, its integration in a process calculus is a completely novel contributions. In the following, we cite some process calculi using a "pre-established" notion of trust but they do not deal with the development of trust models. Carbone, Nielsen, and Sassone [59] have introduced *ctm*, a process calculus which embodies the notion of trust for ubiquitous systems. In *ctm* each principal is equipped with a *policy*, which determines its legal behaviour, formalised using a Datalog-like logic, and with a *protocol*, in the process algebra style, which allows interactions between principals and the flow of information from principals to policies. Martinelli [157] has defined an integrated framework for the specification and automated analysis for security and trust in complex and dynamic scenarios. The author has used a variant of CCS equipped with an inference construct that permits to the model to handle cryptographic primitives. He has showed how this

calculus, usually used for the formal specification and verification of security protocol, may also be used to analyse a variety of access control approaches based on trust management. Cirillo and Riely [64] have studied the relationship between code identity, static analysis and trust in open distributed systems. Their main result is a robust safety theorem expressed in terms of a distributed higher-order π-calculus with code identity, a primitive for remote attestation, enriched by types for the specification of access control policies.

Safety Properties

Access control [207] is a well-established technique to provide *safety properties* ensuring that only principals with appropriate access rights can access informations. In [53] Boudol have argumented that, in order to develop "security-minded" programming languages, it is necessary to define a safety property based on a notion of a security error, namely that it should not be possible to put in a public location a value elaborated using confidential information. Moreover, the author have showed that this safety property is guaranteed by a standard security type system and that it allows to give natural semantics to various security-minded programming constructs. In [64] Cirillo and Riely have defined an extension to the higher-order π-calculus for analysing protocols that rely on remote attestation. They have also provided a static analysis technique for ensuring *robust safety* in the presence of arbitrary attackers. Their robust safety is defined in terms of runtime errors and it requires that no valid process can lead to a runtime error even in the presence of arbitrary attackers. In [154] Maffeis et al. have introduced in the setting of a higher-order spi calculus an instance of the code-carrying authorisation approach, according to which access-control decisions can partly be delegated to untrusted code obtained at run-time. The dynamic verification of this code ensures the safety of authorisation decisions. They have provided a formal definition of robust safety relying on the operational semantics. They have verified their results in terms of dynamic typechecking. In [87] Fournet, Gordon and Maffeis have considered the problem of statically verifying the conformance of the code of a system to an explicit authorisation policy. They have proposed policy conformance criteria in terms of safety properties, that generalise specifications of authentication protocols. They have formalised these criterions in the setting of a process calculus, and presented a verification technique based on a type system.

In our approach we do not consider neither run-time errors nor type systems to check if a safety property is ensured. Indeed our definitions and formalisations just rely on the labelled transition systems and are expressed in terms of security levels of synchronisations. However the intuitions behind them are the same ones of the cited works: the safety property has to be ensured at run-time, while the system evolves, and also in presence of compromised nodes.

7.11 Chapter Summary

For its potential to provide ubiquitous connectivity without the assistance of any fixed infrastructure, ad hoc networks attract the attention of many researchers. Lack of a fixed infrastructure, node mobility, shared wireless medium, cooperative

behaviour, and physical vulnerability are some of the features that make challenging the design of a security scheme for mobile ad hoc networks. In this context, trust management systems have been developed as an answer to the inadequacy of traditional security systems.

In this chapter, we proposed a process calculus for mobile ad hoc networks which embodies a behaviour-based multilevel decentralised trust model. We called this calculus CTAN, standing for Calculus for Trustworthy Ad hoc Networks. Our trust model supports both direct trust, by monitoring nodes behaviour, and indirect trust, by collecting recommendations and spreading reputations. The operational semantics of the calculus is given in terms of a labelled transition system, where actions are executed at a certain security level. We define a labelled bisimilarity parameterised on security levels. Our bisimilarity is a congruence and an efficient proof method for an appropriate variant of reduction barbed congruence. Communications are proved safe with respect to the security levels of the involved parties. In particular, we ensure safety despite compromised nodes: compromised nodes cannot affect the rest of the network. A non-interference result expressed in terms of information flow is also proved.

In the next chapter we extend this calculus with a very simple notion of time that allows us to expressed lifetime of the assertions. In this manner, more recent trust informations should have more influence on the trust establishment process.

8

Time vs Trust

8.1 Introduction

Time and trust are two very related concepts as trust information may change over time. Indeed, trust relations can remain available only for some periods of time. As a consequence, when considering this aspect in a trust model, it should be useful to know the lifetime of the assertions carrying trust information. The lifetime of an assertion indicates its validity, that is the amount of time during which the assertion can be evaluated before it expires. In our previous trust model of Section 7.2, we abstracted from this topic, considering that our assertions were always valid. As we explained in Section 2.3, besides the concern with trust, in the world of wireless networks time plays a central rôle. Indeed various issues are greatly influenced by temporal relationships.

We now introduce a timed variant of the calculus presented in the previous chapter. We call it *TCTAN*, standing for a *Timed Calculus of Trustworthy Ad hoc Networks*. First of all, we extend the previous trust model with a very simple notion of time to express the validity of trust information. Hence, we use time to add timestamps to assertions. The use of timestamps is common in practice, e.g. in cryptographic protocols, in which it is useful the distinction between long-term and short-term secrets or to explicit lifetime of certificates. By adding timestamps to assertions, more recent trust information should have more influence on the trust establishment process. The policy should be implemented in order to give priority to more recent assertions. Then, we develop appropriate changes in operational semantics in order to distinguish between instantaneous and timed actions. As in Chapter 6 for TCWS, we adopt fictitious clock approach of, e.g., [111]. As in TCWS, time proceeds in *discrete* steps represented by occurrences of a simple action tick, in the style of Hennessy and Regan's TPL [111], to denote idling until the next clock cycle. A global time is supposed to be updated whenever all the processes agree on this, by globally synchronising on the special action, called tick, representing the passing of one time unit. All the other actions are assumed to be instantaneous. As described in Section 2.3, in order to achieve this, wireless devices have to rely on a *common notion of time* among the devices, provided by some *clock synchronisation protocol*.

In TCTAN we assume the presence of a clock synchronisation protocol following the *untethered clock approach* [75, 188, 202], achieving a common notion of time without synchronisation. A global time scale is maintained while letting the local clocks run untethered. The untethered approach is becoming popular, because a considerable amount of energy can be saved by this approach. Protocols of this type are design for mobile wireless networks thus they do not rely on network connectivity. The basic idea of this protocol is *not* to synchronise the local clock of each node but instead generate timestamps using unsynchronised local clocks. When such locally generated timestamps are passed between devices, they are transformed into the local time of the receiving device. More precisely, when a message containing a timestamp in transferred between nodes, the timestamps are first transformed from the local time to UTC (Universal Coordinated Time, which is used as a common time transfer format) and then to the local time of the receiver. In these protocols it is only required that all nodes in a network adhere to a common notion of global time.

We now work with configurations. A *configuration* $t \triangleright M$ is a pair composed by a time indicator and a network.

Message transmissions do not require any amount of time because we assume they are instantaneous. We follow a *two-phase* approach [181] separating the execution of actions from the passage of time. We provide the operational semantics of our calculus in terms of a labelled transition system, where transitions for synchronisation are of the form:

$$t \triangleright M \xrightarrow{\lambda}_{\rho} t \triangleright N$$

indicating that the configuration $t \triangleright M$ can perform the action λ, ranging over sending, reception and silent actions, at security level ρ, evolving into the configuration $t \triangleright N$. For that concerns time passing, we have transitions of the form:

$$t \triangleright M \xrightarrow{\mathsf{tick}}_{\rho} t{+}1 \triangleright N$$

indicating that the configuration $t \triangleright M$ can perform the action tick evolving into the configuration $t{+}1 \triangleright N$, where the global time is updated.

We adapt the behavioural theory of CTAN in order to model it to timing extensions. TCTAN preserves all the properties of CTAN and moreover, as TCWS, it enjoys the basic time properties of (i) time determinism, (ii) patience and (iii) maximal progress.

We end this introduction with an outline of the present chapter. In Section 8.2 we introduce our trust model with timed extensions. In Section 8.3 we describe the timed calculus TCTAN and its operational semantics. In Section 8.4 we adapt to TCTAN the notion of observational equivalence of Section 7.6. In Section 8.5 we propose a timed labelled bisimilarity as a proof method for our observations equivalence. In Section 8.6 we discuss some properties. In Section 8.7 we use our timed calculus to give an encoding of the ARAN protocol. In Section 8.8 we present related work. Finally, in Section 8.9 we give a summary of the chapter.

Table 8.1 Timed trust framework

$m, n \in Nodes$	node name
$\langle \mathcal{S}, < \rangle$	complete lattice
$\mathsf{bad}, \mathsf{trust}, \mathsf{low}, \mathsf{high} \in \mathcal{S}$	security level
$t \in Time$	timestamp
$A \in Assertions = Nodes \times Nodes \times \mathcal{S} \times Time$	timed assertion
$T \subseteq \wp(Assertions)$	trust store
$\mathcal{P} : \wp(Assertions) \to Time \to \wp(Assertions)$	timed policy function

8.2 Timed Trust Model

In order to add time to trust information, we extend the definition of the assertions of our trust model proposed in Section 7.2. We simply provide an assertion with a timing tag. This information will be useful to check its validity in a local trust store. Then the set of assertions is now defined as $Assertions = Nodes \times Nodes \times \mathcal{S} \times Time$, where $Time \subseteq \mathbb{N}$. Here we use $t \in Time$ to mean a timestamp. More precisely, a timestamp indicates the instant of time at which the assertion has been added in the local trust store of a node; its lifetime is calculated from that instant of time. The generation of this timestamp is an abstraction of the mechanism used by the assumed clock synchronisation protocol, for which timestamps are first transformed from the current local time to UTC and then to the final local time of the node. Following the approach of [95], we assume to have a *global time*. The presence of this protocol ensure that the timestamps of the assertions in a local trust store are always coherent with the global time. Now, an assertion $\langle m, n, \rho \rangle_t$ says that *at istant of time t a node m has trusted a node n at security level ρ*. For simplicity, we assume that all the assertions have the same lifetime and that all trust managers know it. The security policy $\mathcal{P}$ is now a function that evaluates the current information collected by a node, the current time, and returns a set of consistent and valid assertions. Formally, $\mathcal{P} : \wp(Assertions) \to Time \to \wp(Assertions)$. We assume that the policy obtains the remaining lifetime, and then the validity, of an assertion by comparing the current time and the timing tag of the assertion. In this manner it is possible to infer whether the lifetime of the assertion is expired or whether the assertion is more or less recent with respect to other ones. We also assume that the policy is implemented in order to give priority to more recent assertions. If the assertion is no more valid, as its lifetime is expired, then the policy should provide to erase such an assertion from the local trust store. The new definition of the policy ensures that a trust table always contains information coherent with the current time.

Table 8.1 summarises our timing extension.

8.3 The Calculus: Syntax and Operational Semantics

The syntax of our calculus with timing extension does not change with respect to that described in Table 7.2. As a consequence, also the structural congruence of Table 7.3 is the same. All the pertaining about free and bound variables and guarded recursion for CTAN are still valid here.

Table 8.2 LTS - Synchronisation

$$\text{(MCastT)}\ \frac{\mathcal{D} := \{n : T(m,n) \geq \sigma\} \quad \mathcal{D} \neq \emptyset}{t \triangleright m[\sigma!\langle \tilde{v}\rangle.P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright m[P]_T}$$

$$\text{(RcvT)}\ \frac{T(n,m) \geq \sigma \quad |\ \tilde{x}\ | = |\ \tilde{v}\ |}{t \triangleright n[\sigma?(\tilde{x}).P]_T \xrightarrow{m?\tilde{v}\triangleright n}_\sigma t \triangleright n[\{\tilde{v}/\tilde{x}\}P]_T}$$

$$\text{(UCastT)}\ \frac{T(m,n) \geq \sigma}{t \triangleright m[\sigma!\langle \tilde{v}\rangle_n.P]_T \xrightarrow{m!\tilde{v}\triangleright n}_\sigma t \triangleright m[P]_T}$$

$$\text{(RcvParT)}\ \frac{t \triangleright M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_\rho t \triangleright M' \quad t \triangleright N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_\rho t \triangleright N' \quad \widehat{\mathcal{D}} := \mathcal{D} \cup \mathcal{D}'}{t \triangleright M \mid N \xrightarrow{m?\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho t \triangleright M' \mid N'}$$

$$\text{(SyncT)}\ \frac{t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho t \triangleright M' \quad t \triangleright N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_\rho t \triangleright N' \quad \mathcal{D}' \subseteq \mathcal{D}}{t \triangleright M \mid N \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho t \triangleright M' \mid N'}$$

$$\text{(SumT)}\ \frac{t \triangleright m[P]_T \xrightarrow{\lambda'}_\sigma t \triangleright m[P']_T}{t \triangleright m[P+Q]_T \xrightarrow{\lambda'}_\sigma t \triangleright m[P']_T}$$

$$\text{(ParT)}\ \frac{t \triangleright M \xrightarrow{\lambda'}_\rho t \triangleright M' \ \ \mathsf{sender}(\lambda') \notin \mathsf{nds}(N)}{t \triangleright M \mid N \xrightarrow{\lambda'}_\rho t \triangleright M' \mid N}$$

Table 8.3 LTS - Trust Management

$$\text{(DTrustT)}\ \frac{T(m,n) > \mathsf{bad} \quad \tilde{v} := \langle n, \mathsf{bad}\rangle_t \quad T' := \mathcal{P}(T \cup \langle m, \tilde{v}\rangle, t) \quad \mathcal{D} := \{n : T(m,n) > \mathsf{bad}\}}{t \triangleright m[P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathsf{trust}} t \triangleright m[P]_{T'}}$$

$$\text{(SndRcmT)}\ \frac{T(m,n) = \rho \quad \tilde{v} := \langle n, \rho\rangle_{t'} \quad \mathcal{D} := \{n : T(m,n) > \mathsf{bad}\}}{t \triangleright m[P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathsf{trust}} t \triangleright m[P]_T}$$

$$\text{(RcvRcmT)}\ \frac{T(n,m) > \mathsf{bad} \quad \tilde{v} := \langle l, \rho\rangle_{t'} \quad T' := \mathcal{P}(T \cup \langle m, \tilde{v}\rangle, t)}{t \triangleright n[P]_T \xrightarrow{m?\tilde{v}\triangleright n}_{\mathsf{trust}} t \triangleright n[P]_{T'}}$$

$$\text{(LoseT)}\ \frac{T' \subseteq T \quad T'' := \mathcal{P}(T', t)}{t \triangleright n[P]_T \xrightarrow{\tau}_{\mathsf{trust}} t \triangleright n[P]_{T''}}$$

As we stated above, we assume the presence of a *global time* [95] $t \in \mathit{Time}$ with $\mathit{Time} \subseteq \mathbb{N}$. Then, for that concerns the rules of our timed LTS, they are now defined over *configurations* of the form $t \triangleright M$, where $t \in \mathit{Time}$ is the global time indicator and M is a network. The rules are of the form

$$t \triangleright M \xrightarrow{\lambda}_\rho t' \triangleright M'$$

Table 8.4 LTS - Time Passing

$$\text{(Tick-}\mathbf{0}\text{)}\ \frac{t' := t+1 \quad \rho \neq \mathsf{bad}}{t \rhd \mathbf{0} \xrightarrow{\mathsf{tick}}_{\rho} t' \rhd \mathbf{0}}$$

$$\text{(Tick)}\ \frac{t' := t+1 \quad T' := \mathcal{P}(T,t') \quad P \neq P+Q \quad t \rhd m[P]_T \overset{m!\tilde{v}\rhd\mathcal{D}}{\not\longrightarrow}_{\sigma} \quad \rho \neq \mathsf{bad}}{t \rhd m[P]_T \xrightarrow{\mathsf{tick}}_{\rho} t' \rhd m[P]_{T'}}$$

$$\text{(SumTick)}\ \frac{t \rhd m[P]_T \xrightarrow{\mathsf{tick}}_{\rho} t' \rhd m[P]_{T'} \quad t \rhd m[Q]_T \xrightarrow{\mathsf{tick}}_{\rho} t' \rhd m[Q]_{T'}}{t \rhd m[P+Q]_T \xrightarrow{\mathsf{tick}}_{\rho} t' \rhd m[P+Q]_{T'}}$$

$$\text{(ParTick)}\ \frac{t \rhd M \xrightarrow{\mathsf{tick}}_{\rho} t' \rhd M' \quad t \rhd N \xrightarrow{\mathsf{tick}}_{\rho} t' \rhd N'}{t \rhd M \mid N \xrightarrow{\mathsf{tick}}_{\rho} t' \rhd M' \mid N'}$$

Table 8.5 LTS - Matching and Recursion

$$\text{(ThenT)}\ \frac{t \rhd n[P]_T \xrightarrow{\lambda}_{\rho} t' \rhd n[P']_{T'} \quad \tilde{v_1}\ \mathtt{op}\ \tilde{v_2} = \mathtt{true}}{t \rhd n[[\tilde{v_1}\ \mathtt{op}\ \tilde{v_2}]P,Q]_T \xrightarrow{\lambda}_{\rho} t' \rhd n[P']_{T'}}$$

$$\text{(ElseT)}\ \frac{t \rhd n[Q]_T \xrightarrow{\lambda}_{\rho} t' \rhd n[Q']_{T'} \quad \tilde{v_1}\ \mathtt{op}\ \tilde{v_2} = \mathtt{false}}{t \rhd n[[\tilde{v_1}\ \mathtt{op}\ \tilde{v_2}]P,Q]_T \xrightarrow{\lambda}_{\rho} t' \rhd n[Q']_{T'}}$$

$$\text{(RecT)}\ \frac{t \rhd n[\{\tilde{v}/\tilde{x}\}P]_T \xrightarrow{\lambda}_{\rho} t' \rhd n[P']_{T'} \quad H(\tilde{x}) \overset{\text{def}}{=} P}{t \rhd n[H\langle\tilde{v}\rangle]_T \xrightarrow{\lambda}_{\rho} t' \rhd n[P']_{T'}}$$

where the metavariable λ ranges now over $\tau, m!\tilde{v}\rhd\mathcal{D}, m?\tilde{v}\rhd\mathcal{D}$ and tick. In the sequel, we write λ' to range over $\tau, m!\tilde{v}\rhd\mathcal{D}$ and $m?\tilde{v}\rhd\mathcal{D}$.

In the sequel, we adopt the following notation; let be $\tilde{v} := \langle n, \rho\rangle_t$, then we write $\langle m, \tilde{v}\rangle$ to mean $\langle m, n, \rho\rangle_t$. Tables 8.2, 8.3 and 8.4 model respectively synchronisation, trust management and time passing. Rules in Tables 8.2 and 8.3 are instantaneous, that is assumed to take no time, whereas rules in Table 8.4 indicates the moving from istant t to the next instant of time $t+1$. The rules of Table 8.2 are similar to the corresponding without time in Table 7.4. For that concerns Table 8.3, in rule (DTrustT) models *direct trust* as rule (DTrust) of Table 7.5; a timestamp t, indicating the current global time, is added to trust information, as we explained above. Thus a new trust table is calculated applying the local policy to $T \cup \langle m, \tilde{v}\rangle$ and t. In general, when it is necessary to calculate a new trust table, the policy is always applied to the current trust table and the current global time. This is due to the changes operate at the trust model, as stated in Section 8.2. Rule (SndRcmT) models *indirect trust* (SndRcm) of Table 7.5; by sending a recommendation that now contains also timing information. Similarly, in rule (RcvRcmT) models the reception of a recommendation with timing information from a trusted node: a new trust table T' is calculated, applying the local policy to $T \cup \langle m, \tilde{v}\rangle$ and t.

Rules in Table 8.4 are quite simple: rule (Tick-**0**) is quite standard, rule (Tick) models the passage of one time unit. Indeed the global time t is increased and a new coherent trust table is calculated. The requirement $t \triangleright m[P]_T \not\xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma$ is necessary to ensure the maximal progress property. The requirement $P \neq P+Q$ is necessary as, with respect to tick actions, the choice operator + behaves different: for a node $m[P+Q]_T$ time passes when both $t \triangleright m[P]_T$ and $t \triangleright m[Q]_T$ are able to perform a tick action, and in such a case by performing tick it is reached a configuration where both the derivatives of the choice can still be chosen. In other words, if both the derivatives are just idling before the environment requests one of them, the choice between them will not be made by the passage of time alone. This is to say that + is not decided by the action tick. This is necessary to ensure that the passage of time is deterministic. Rule (SumTick) models this. Rule (ParTick) ensures that a clock-tick happens simultaneously in every branch of parallel composition. We can notice that the security level of transmissions in Table 8.4 is arbitrary as it does not depend on the syntax of the processes. Thus, tick actions can fire at every valid security level.

Finally, Table 8.5 contains the standard rules for matching and recursion.

8.4 Behavioural Semantics

In this section, as in Section 8.5, we deal with very similar arguments to those provided in Sections 7.6, 7.7 and 7.8. We need only to modify some definitions given in those sections in order take into account the timing aspects recently introduced. The results obtained for the timed calculus TCTAN are similar to those obtained for CTAN. The behavioural semantics is now defined over configurations.

Let us start with our main behavioural equivalence, the *σ-timed reduction barbed congruence*. First of all, we rewrite the rules (Red1) and (Red2) of the previous reduction semantics in terms of configurations:

$$\text{(RedT1)}\ \frac{t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho t \triangleright M'}{t \triangleright M \twoheadrightarrow t \triangleright M'} \qquad \text{(RedT2)}\ \frac{t \triangleright M \xrightarrow{\tau}_{\mathsf{trust}} t \triangleright M'}{t \triangleright M \twoheadrightarrow t \triangleright M'}$$

We use the same notation $\twoheadrightarrow^*$ of Section 7.6 to denote the reflexive and transitive closure of $\twoheadrightarrow$.

The following three definitions are very similar to Definitions 7.10, 7.11 and 7.12 respectively. The only difference is that they are reformulated according to configurations.

Definition 8.1 (σ-timed barb). *We write $t \triangleright M \downarrow_n^\sigma$ if either $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$ or $M \equiv m[\sigma!\langle\tilde{v}\rangle_n.P]_T \mid N$, for some $m, N, \tilde{v}, P, T$ such that $n \notin \mathsf{nds}(M)$, and $T(m,n) \geq \sigma$. We write $t \triangleright M \Downarrow_n^\sigma$ if $t \triangleright M \twoheadrightarrow^* t \triangleright M' \downarrow_n^\sigma$ for some network M'.*

Definition 8.2 (σ-timed barb preserving). *A relation $\mathcal{R}$ is said to be* σ-timed barb preserving *if whenever $t \triangleright M \mathcal{R} t \triangleright N$ it holds that $t \triangleright M \downarrow_n^\sigma$ implies $t \triangleright N \Downarrow_n^\sigma$.*

Definition 8.3 (Reduction closure). *A relation $\mathcal{R}$ is said to be* reduction closed *if $t \triangleright M \mathcal{R} t \triangleright N$ and $t \triangleright M \twoheadrightarrow t \triangleright M'$ imply there is N' such that $t \triangleright N \twoheadrightarrow^* t \triangleright N'$ and $t \triangleright M' \mathcal{R} t \triangleright N'$.*

A new definition is necessary because, when comparing two configurations, time must pass in the same manner for both.

Definition 8.4 (tick-closure). *A relation $\mathcal{R}$ is said to be* tick-closed *if $t \triangleright M \; \mathcal{R} \; t \triangleright N$ and $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ imply there is N' such that $t \triangleright N \twoheadrightarrow^* \xrightarrow{\mathsf{tick}}_\rho \twoheadrightarrow^* t' \triangleright N'$ and $t' \triangleright M' \; \mathcal{R} \; t' \triangleright N'$.*

Definition 8.5 (Contextuality). *A relation $\mathcal{R}$ is said to be* contextual *if $t \triangleright M \; \mathcal{R} \; t \triangleright N$ implies that $t \triangleright M \mid O \; \mathcal{R} \; t \triangleright N \mid O$, for all configurations $t \triangleright O$.*

Finally, everything is in place to define our σ-timed reduction barbed congruence.

Definition 8.6 (σ-Timed reduction barbed congruence). *The* σ-timed reduction barbed congruence*, written $\cong'_\sigma$, is the largest symmetric relation over configurations which is σ-timed barb preserving, reduction closed,* tick*-closed and contextual.*

8.5 A Bisimulation Proof Method

For similar reasons to those in the discussion of Section 7.7, we need to define a proof technique based on labelled bisimilarity. In the sequel we define a new notion of bisimilarity taking into account the timing aspects of our calculus. As a main result, we prove that our new labelled bisimilarity is a proof-technique for our σ-timed reduction barbed congruence.

We rewrite our rule (Shh) and (Obs) of Section 7.7 in terms of configurations.

$$\text{(ShhT)} \quad \frac{t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho t \triangleright M' \quad \mathcal{D} \subseteq \mathsf{nds}(M) \quad \rho' \neq \mathsf{bad}}{t \triangleright M \xrightarrow{\tau}_{\rho'} t \triangleright M'}$$

$$\text{(ObsT)} \quad \frac{t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho t \triangleright M' \quad \mathcal{D}' := \mathcal{D} \setminus \mathsf{nds}(M) \neq \emptyset}{t \triangleright M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_\rho t \triangleright M'}$$

The meanings are the same to those in Section 7.7. Again, in the derivation tree the rule (ObsT) can only be applied at top-level. In fact, we cannot use this rule together with rule (ParT) of Table 8.2, because the label λ' in rule (ParT) does not range on this new action.

In the sequel, we use the metavariable α to range over the following actions: $\tau, \mathsf{tick}, m?\tilde{v}\triangleright\mathcal{D}$, and $m!\tilde{v}\blacktriangleright\mathcal{D}$. Since we are interested in *weak behavioural equivalences*, that abstract over τ-actions, we adopt the same notion of weak action previously introduced: then we write $\Rightarrow_\rho$ denoting the reflexive and transitive closure of $\xrightarrow{\tau}_\rho$; we also write $\overset{\alpha}{\Longrightarrow}_\rho$ denotes $\Rightarrow_\rho \xrightarrow{\alpha}_\rho \Rightarrow_\rho$; $\overset{\hat{\alpha}}{\Longrightarrow}_\rho$ denotes $\Rightarrow_\rho$ if $\alpha = \tau$ and $\overset{\alpha}{\Longrightarrow}_\rho$ otherwise. It is important to notice that time must pass at the same manner for the configurations, as our behavioural relations are defined over configurations with the same global time.

Definition 8.7 (δ-Timed Bisimilarity). *The* δ-timed bisimilarity*, written* $\approx'_\delta$*, is the largest symmetric relation over configurations such that whenever* $t \triangleright M \approx'_\delta t \triangleright N$ *if* $t \triangleright M \xrightarrow{\alpha}_\rho t' \triangleright M'$*, with* $\rho \leq \delta$*, then there exists a configuration* $t \triangleright N'$ *such that* $t \triangleright N \stackrel{\hat{\alpha}}{\Longrightarrow}_\rho t' \triangleright N'$ *and* $t' \triangleright M' \approx'_\delta t' \triangleright N'$*.*

Theorem 8.8 ($\approx'_\delta$ is contextual). *Let* $t \triangleright M$ *and* $t \triangleright N$ *be two configurations such that* $t \triangleright M \approx'_\delta t \triangleright N$*. Then* $t \triangleright M \mid O \approx'_\delta t \triangleright N \mid O$ *for all configurations* $t \triangleright O$*.*

Proof We prove that the relation

$$\mathcal{S} \stackrel{\text{def}}{=} \{\big(t \triangleright M \mid O\,,\, t \triangleright N \mid O\big) \text{ for all } t \triangleright O \text{ such that } t \triangleright M \approx'_\delta t \triangleright N\}$$

is a δ-timed bisimulation. We proceed by case analysis on the transition $t \triangleright M \mid O \xrightarrow{\alpha}_\rho t' \triangleright \widehat{M}$, with $\rho \leq \delta$. The full proof can be found in Section A.3 at page 189. □

Theorem 8.9 (Soundness). *Let* $t \triangleright M$ *and* $t \triangleright N$ *be two configurations such that* $t \triangleright M \approx'_\delta t \triangleright N$*. Then* $t \triangleright M \cong'_\sigma t \triangleright N$*, for* $\sigma \leq \delta$*.*

Proof The σ-timed barb preserving follows by Lemma A.13 at page 189, the reduction-closure and the tick-closure follow by definition and contextuality follows by Theorem 8.8. □

8.6 Properties

For that concerns safety properties discussed in Section 7.5, as they do not involve time, they remain unchanged. For that concern the time properties, TCTAN satisfies the same properties of TCWS described in Section 6.4, that is time determinism, patience, and maximal progress.

Theorem 8.10 formalises the determinism nature of time passing: a configuration can reach at most one new state by executing the action tick.

Theorem 8.10 (Time Determinism). *Let* $t \triangleright M$ *be a configuration. If* $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ *and* $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M''$ *then* M' *and* M'' *are syntactically the same.*

Proof By induction on the length of the proof of $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$. The full proof can be found in Section A.3 at page 190.

□

As in [111, 195], the maximal progress property says that processes communicate as soon as a possibility of communication arises. In TCWS this property says that transmissions cannot be delayed. In this calculus this property says that transmissions *at level* σ cannot be delayed. The transmissions at level trust do not depend on the syntax of the processes, then the actions concerning with trust can fire at any step of the computation and cannot be predicted in advance.

Theorem 8.11 (Maximal Progress). *Let* $t \triangleright M$ *be a configuration. If there is* N *such that* $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright N$ *then* $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ *for no network* M' *and* $\rho \neq \mathsf{bad}$*.*

Proof By induction on the structure of M. The full proof can be found in Section A.3 at page 190.

□

The last time property is patience. In [111,195] and in TCWS patience guarantees that a process will wait indefinitely until it can communicate. In our setting, as in TCWS, this means that if no transmission *at level* σ can start then it must be possible to execute a tick-action to let time pass.

Theorem 8.12 (Patience). *Let $t \triangleright M$ be a configuration. If $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} t \triangleright M'$ for no network M' then there is a network N and $\rho \neq \mathsf{bad}$ such that $t \triangleright M \xrightarrow{\mathsf{tick}}_{\rho} t' \triangleright N$.*

Proof By contradiction and then by induction on the structure of M. The full proof can be found in Section A.3 at page 190. □

It is also possible to prove a very similar non-interference result to that described in Section 7.8 using as process equivalence the notion of δ-time bisimilarity previously defined. As we can notice in Table 8.4, the security levels of tick actions are not related to the syntax of the processes. For this reason, we can say that the non interference property is not related with time. Without loss of generality and in order to simplify the notation of the following Definition 8.13 and Theorem 8.14, we assume that tick actions are executed at security level `trust`. Thus, whenever $t \triangleright M \xrightarrow{\mathsf{tick}}_{\rho} t' \triangleright M'$ we assume that $\rho = \mathtt{trust}$.

High-level configurations of our setting are very similar to high-level networks presented in Definition 7.19, as we can notice from the following definition:

Definition 8.13 (δ-high level configuration). *A configuration $t \triangleright H$ is a* δ-high level configuration*, written $t \triangleright H \in \mathcal{H}'_{\delta}$, if whenever $t \triangleright H \xrightarrow{\lambda}_{\delta'} t' \triangleright H'$ then either $\delta' = \mathtt{trust}$ or $\delta' > \delta$. Moreover, $t' \triangleright H' \in \mathcal{H}'_{\delta}$.*

The new statement of non-interference result differs from that in Theorem 7.20 only for the presence of configurations:

Theorem 8.14 (Non-interference). *Let $t \triangleright M$ and $t \triangleright N$ be two configurations such that $t \triangleright M \approx'_{\delta} t \triangleright N$. Let $t \triangleright H$ and $t \triangleright K$ be two configurations such that (i) $t \triangleright H, t \triangleright K \in \mathcal{H}'_{\delta}$; (ii) $t \triangleright H \approx'_{\mathtt{trust}} t \triangleright K$ and (iii) $\mathsf{nds}(H) = \mathsf{nds}(K)$. Then, $t \triangleright M \mid H \approx'_{\delta} t \triangleright N \mid K$.*

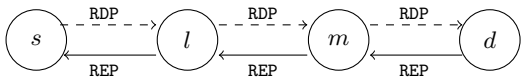
Proof The full proof can be found in Section A.3 at page 191. □

8.7 Time at Work: the ARAN Routing Protocol

In this section, we use our calculus to formalise the secure, on-demand routing protocol *ARAN* [209]. It uses cryptographic certificates to bring authentication, message integrity and non-repudiation to the route discovery process. As usual, given a node n, we write K_{n^+} and K_{n^-} to mean the public key and the private key of n, respectively. Moreover, given a message $\tilde{v}$, we write $\{\tilde{v}\}_{K_{n^-}}$ meaning the message $\tilde{v}$ signed by n. In the following, we refer to *ip address* to mean the name of the node and we use the notation ip_n and n with the same meaning.

Figure 8.1 An example of the operation and the messages of ARAN

$$
\begin{array}{rcll}
s & \rightarrow & * & : \{\texttt{RDP}, d, N_s\}_{K_{s-}}, [cert_s] \\
l & \rightarrow & * & : \{\{\texttt{RDP}, d, N_s\}_{K_{s-}}\}_{K_{l-}}, [cert_s, cert_l] \\
m & \rightarrow & * & : \{\{\texttt{RDP}, d, N_s\}_{K_{s-}}\}_{K_{m-}}, [cert_s, cert_m] \\
d & \rightarrow & m & : \{\texttt{REP}, s, N_s\}_{K_{d-}}, [cert_d] \\
m & \rightarrow & l & : \{\{\texttt{REP}, s, N_s\}_{K_{d-}}\}_{K_{m-}}, [cert_d, cert_m] \\
l & \rightarrow & s & : \{\{\texttt{REP}, s, N_s\}_{K_{d-}}\}_{K_{l-}}, [cert_d, cert_l]
\end{array}
$$



ARAN requires the use of a *trusted certificate server* (*tcs*), whose public key is known to all *valid* nodes. This server sends to each node a certificate, containing the ip address of the node, its public key, a timestamp t of when the certificate was created, and a time e at which the certificate expires, all signed with the private key of *tcs*. For instance, let n be the node receiving a certificate from *tcs*; the certificate $cert_n$ has the form $\{ip_n, K_{n^+}, t, e\}_{K_{tcs^-}}$. Nodes use these certificates to authenticate themselves to other nodes during the exchange of routing messages.

In Figure 8.1 we report a scheme of the ARAN protocol with four nodes: a source s, a destination d and two intermediate nodes l and m. We also provide a graphical representation of the message flow, where the dashed arrows denote the broadcast of *route discovery packet*, while the continuous arrows denote the unicast sending of *reply packet*. We do not report in the figure the preliminary phase in which each node receives a certificate from *tcs*, assuming it has been already performed. The protocol proceeds as follows. The source s begins a route instantiation to the destination d by broadcasting to its neighbours a route discovery packet and its certificate. The packet is of the form $\langle\{\texttt{RDP}, d, N_s\}_{K_{s^-}}, [cert_s]\rangle$, where RDP is the packet identifier, d is the ip address of the destination and N_s is a nonce, all signed with the private key of the source. The purpose of the nonce is to uniquely identify a RDP coming from a source and then it helps in detecting replay messages. Each time s performs a route discovery, it monotonically increases the nonce. When a node receives a RDP message, it sets an entry in its routing table with s as destination, and the neighbour from which it received the RDP for the first time as next hop. This is useful for the reverse path back to the source, in anticipation of eventually receiving a reply message that it will need to forward back to the source. When the intermediate node l receives the request packet sent by s, it verifies that the certificate has not expired. Then it uses the public key of s, which it extracts from $cert_s$, to validate the signature. The receiving node also checks the nonce to verify that it has not already processed this RDP: nodes do not forward messages with already-seen tuples. If some of these verifications fail, the message is dropped, otherwise the intermediate node l signs the received RDP packet and appends its own certificate $cert_l$. Then l rebroadcasts a packet of this form: $\langle\{\{\texttt{RDP}, d, N_s\}_{K_{s^-}}\}_{K_{l^-}}, [cert_s, cert_l]\rangle$. When m receives the route discovery sent by l, it verifies if the certificates are still valid, then it validates the signatures

for both the source s and the node l it received the RDP from, using the public keys extracted by the certificates in the message. If some of these verifications fail, the message is dropped, otherwise m removes the certificate and the signature of l and records it as its parent node. Then it signs the message, originally sent by the source, and appends its own certificate. Then m rebroadcasts a packet of this form: $\langle\{\{\texttt{RDP}, d, N_s\}_{K_{s^-}}\}_{K_{m^-}}, [cert_s, cert_m]\rangle$. Eventually the message is received by the destination d, who replies to the first node from which it has received the RDP for a source and a given nonce. In this case this node is m. Then the destination sends to m a reply packet and its certificate. The message that d sends has this form: $\langle\{\texttt{REP}, s, N_s\}_{K_{d^-}}, [cert_d]\rangle$. It includes the packet type identifier `REP`, the ip address of the source, and the nonce originally sent by the source, all signed with the private key of the destination, and the certificate of d. When m receives the reply packet from d, it verifies if the certificate is not expired and validates its signature. If some of these verifications fail, the message is dropped, otherwise m signs with its private key the message and appends its own certificate before forwarding the packet to its parent node. Then m sends to l a message of this form: $\langle\{\{\texttt{REP}, s, N_s\}_{K_{d^-}}\}_{K_{m^-}}, [cert_d, cert_m]\rangle$. Eventually the message is received by l, that verifies if the certificates are not expired and validates the signatures on the received message. If some of these verifications fail, the message is dropped, otherwise node l removes signature and certificate of m. Then it signs with its private key the message, originally sent by the destination, and appends its own certificate before unicasting the reply to s. Then l sends a message of this form: $\langle\{\{\texttt{REP}, s, N_s\}_{K_{d^-}}\}_{K_{l^-}}, [cert_d, cert_l]\rangle$. When the source receives the reply, it verifies the certificates are still valid, then it validates the signatures and verifies the nonce returned by the destination. If some of these verifications fail, the message is dropped, otherwise the source notices that the intended destination was reached and that l is the first next-hop towards the destination.

When writing the ARAN protocol in our calculus, we consider a few simplifications. For simplicity, we eliminate the nonce field in our messages, as it only helps in detecting replay attacks and here we do not explicitly deal with attacks. The certificates we use in our encoding contain only the ip address of the node, signed with the private key of the trust certificate server. It is not necessary to include the public key and timing factors into certificates as we assume that if a node $m[P]_T$ knows a node n, that is $T(m,n)$ is defined, then m knows the public key of n. According to our timing extension described in Section 8.2, this means that its certificate is still valid. We remember that in our calculus we assume the presence of a hierarchical key generation and distribution protocol. This implies that a signature is produced with a private key at a specific security level. Thus, if m wants to verify the validity of a signature produced by a node n at security level ρ, it simply has to control whether $T(m,n) \geq \rho$.

We adopt the syntax extension for the matching operator

$$[(\tilde{u}_1 \,\texttt{op}\, \tilde{u}'_1)\texttt{lc}\ldots\texttt{lc}(\tilde{u}_k \,\texttt{op}\, \tilde{u}'_k)]P, Q$$

as we explained in Section 7.9.2. We also use the destructor construct

$$let\ \tilde{x}{=}g(u_1, u_2, \ldots, u_k)\ in\ P\ else\ Q$$

introduced in Section 7.9.2.

For our purposes, we need the same destructor $get(\cdot)$ of Section 7.9.2, that takes in input a signed message and a list of ip addresses and returns the content of the message if the signatures of the message correspond to the ip addresses in the list, $\perp$ otherwise. We also need another destructor $ipcert(\cdot)$, that takes in input a list of certificates and the ip address of the trust certificate server *tcs* and returns a list with the ip addresses of the certificates if their signatures correspond to the ip address of the trust certificate server, $\perp$ otherwise. For instance,

$$ipcert(\{ip_{n_1}\}_{K_{tcs}-}, \ldots, \{ip_{n_k}\}_{K_{tcs}-}, ip_{tcs}) = ip_{n_1}, \ldots, ip_{n_k},$$

whereas $ipcert(\{ip_{n_1}\}_{K_{ip_1}-}, \ldots, \{ip_{n_k}\}_{K_{ip_k}}, ip_{tcs}) = \perp$, if $ip_i \neq ip_{tcs}$, for some i. We assume that also the destructor *ipcert*, as well the destructor *get*, is capable to get the corresponding public keys from ip addresses.

We use a number of notations, some of them are the same introduced in Section 7.9.2. Let $h = n, l, \ldots, q$ be a list of ip address and m an ip address; we sometimes write $T(m, h) \geq \rho$ to mean $T(m, n) \geq \rho \wedge T(m, l) \geq \rho \wedge \ldots \wedge T(m, q) \geq \rho$. Let h be a list and v a value; we write $h \uplus v$ for the list where v has been appended to h. We assume standard functions on lists such as $\mathbf{head}(\cdot)$ and $\mathbf{tail}(\cdot)$, with the convection that $\mathbf{head}(h) = \mathbf{tail}(h)$, if h is composed by just one value. We also define an ad hoc function $\mathbf{orig}(\cdot)$ that takes in input a signed message and eliminate all the signatures but the first one. For instance, $\mathbf{orig}(\{\ldots\{u\}_{K_{n_1}-}\ldots\}_{K_{n_k}-}) = \{u\}_{K_{n_1}-}$ and $\mathbf{orig}(\{u\}_{K_{n_1}-}) = \{u\}_{K_{n_1}-}$

In Figure 8.2 we provide the encoding of the ARAN protocol in our calculus. Let us explain it in some detail.

At the beginning, each node can be in one of these two states:

- $\textsc{Source}(ip_s, ip_d, T_s, cert_s, ip_{tcs})$, when a (source) node initiates the protocol. In this state the node broadcasts a request message;
- $\textsc{Node}(ip, cert, T, ip_{tcs}, ip_p)$, when a node participates receiving a request or reply message.

While the protocol is executed, nodes may evolve into one of the following states:

- $\textsc{AwaitReply}(ip_s, ip_d, T_s, cert_s, ip_{tcs})$, the initiator node waits for the reply message;
- $\textsc{RouteSuccess}(ip_s, ip_d, ip_p)$, the source node has noticed that the intended destination can be reached going through ip_p as the first next hop;
- $\textsc{FwdRequest}(ip, cert, T, pkt, ip_{tcs}, cert_s, ip_d, ip_s, ip_p)$, an intermediate node receives a request message;
- $\textsc{SendReply}(ip_d, cert_d, T_d, ip_{tcs}, ip_s, ip_p)$, the destination node sends the reply message;
- $\textsc{FwdReply}(ip, cert, T, ip_{tcs}, pkt, cert_d, ip_p)$, an intermediate node forwards a reply message.

The source node s begins in the state $\textsc{Source}(ip_s, ip_d, T_s, cert_s, ip_{tcs})$, where ip_s, ip_d, ip_{tcs} are the ip addresses of the source, the destination and the trust certificate server, respectively, while $cert_s$ and T_s are the certificate and the trust table of s, respectively. In this state, the source node broadcasts a route discovery packet of the form $\langle\{\texttt{RDP}, ip_d\}_{K_{ip_s}-}, [cert_s]\rangle$. After this transmission, the node evolves into the state $\textsc{AwaitReply}(ip_s, ip_d, T_s, cert_s, ip_{tcs})$, waiting for a reply

Figure 8.2 The encoding of the ARAN protocol

$/^*$ *Starting node broadcasts* RDP *packet and evolves into* AwaitReply *state waiting for reply.*$^*/$

$$\textsc{Source}(ip_s, ip_d, T_s, cert_s, ip_{tcs}) \stackrel{\text{def}}{=} \sigma!\langle\{\texttt{RDP}, ip_d\}_{K_{ip_s-}}, [cert_s]\rangle.\textsc{AwaitReply}(ip_s, ip_d, T_s, cert_s, ip_{tcs})$$

$/^*$ *The initiator node in the* AwaitReply *state receives a reply message; if the reply is successfully returned, the node accepts the route, evolving into the state* RouteSuccess. $^*/$

$$\begin{array}{l}
\textsc{AwaitReply}(ip_s, ip_d, T_s, cert_s, ip_{tcs}) \stackrel{\text{def}}{=} \\
\quad \sigma?(reppkt, certList).let\ x{=}ipcert(certList, ip_{tcs})\ in \\
\qquad\qquad [T_s(ip_s, x){\geq}\sigma] \\
\qquad\qquad\quad let\ (y_1, y_2){=}get(reppkt, x)\ in \\
\qquad\qquad\qquad [y_1 = \texttt{REP}] \\
\qquad\qquad\qquad\quad [(y_2{=}ip_s){\wedge}(\mathbf{head}(x){=}ip_d)] \\
\qquad\qquad\qquad\qquad \textsc{RouteSuccess}(ip_s, ip_d, \mathbf{tail}(x)), \\
\qquad\qquad\qquad\quad \textsc{Source}(ip_s, ip_d, T_s, cert_s, ip_{tcs}), \\
\qquad\qquad\qquad \textsc{AwaitReply}(ip_s, ip_d, T_s, cert_s, ip_{tcs}) \\
\qquad\qquad\quad else\ \textsc{AwaitReply}(ip_s, ip_d, T_s, cert_s, ip_{tcs}), \\
\qquad\qquad \textsc{AwaitReply}(ip_s, ip_d, T_s, cert_s, ip_{tcs}) \\
\qquad\quad else\ \textsc{AwaitReply}(ip_s, ip_d, T_s, cert_s, ip_{tcs}) \\
\quad + \ \textsc{Source}(ip_s, ip_d, T_s, cert_s, ip_{tcs})
\end{array}$$

$/^*$ *Intermediate nodes may receive a request or a reply message.* $^*/$

$$\begin{array}{l}
\textsc{Node}(ip, cert, T, ip_{tcs}, ip_p) \stackrel{\text{def}}{=} \\
\quad \sigma?(rpkt, certList).let\ x{=}ipcert(certList, ip_{tcs})\ in \\
\qquad\qquad [T(ip, x){\geq}\sigma] \\
\qquad\qquad\quad let\ (y_1, y_2){=}get(rpkt, x)\ in \\
\qquad\qquad\qquad [y_1 = \texttt{RDP}] \\
\qquad\qquad\qquad\quad \textsc{FwdRequest}(ip, cert, T, \mathbf{orig}(rpkt), ip_{tcs}, \mathbf{head}(certList), y_2, \\
\qquad\qquad\qquad\qquad\qquad \mathbf{head}(x), \mathbf{tail}(x)), \\
\qquad\qquad\qquad [y_1 = \texttt{REP}] \\
\qquad\qquad\qquad\quad \textsc{FwdReply}(ip, cert, T, ip_{tcs}, \mathbf{orig}(rpkt), \mathbf{head}(certList), ip_p), \\
\qquad\qquad\qquad \textsc{Node}(ip, cert, T, ip_{tcs}, ip_p) \\
\qquad\qquad\quad else\ \textsc{Node}(ip, cert, T, ip_{tcs}, ip_p), \\
\qquad\qquad \textsc{Node}(ip, cert, T, ip_{tcs}, ip_p) \\
\qquad\quad else\ \textsc{Node}(ip, cert, T, ip_{tcs}, ip_p)
\end{array}$$

$/^*$ *A node receiving a request message controls if it is the intended destination or not.* $^*/$

$$\begin{array}{l}
\textsc{FwdRequest}(ip, cert, T, pkt, ip_{tcs}, cert_s, ip_d, ip_s, ip_p) \stackrel{\text{def}}{=} \\
\quad [ip_d = ip] \\
\qquad \textsc{SendReply}(ip, cert, T, ip_{tcs}, ip_s, ip_p), \\
\quad \sigma!\langle\{pkt\}_{K_{ip}-}, [cert_s{\uplus}cert]\rangle.\textsc{Node}(ip, cert, T, ip_{tcs}, ip_p)
\end{array}$$

$/^*$ *After receiving the request, the destination unicasts a signed reply and its certificate back along the reverse path to the source.* $^*/$

$$\begin{array}{l}
\textsc{SendReply}(ip_d, cert_d, T_d, ip_{tcs}, ip_s, ip_p) \stackrel{\text{def}}{=} \\
\quad \sigma!\langle\{\texttt{REP}, ip_s\}_{K_{ip_d-}}, [cert_d]\rangle_{ip_p} + \ \textsc{Node}(ip_d, cert_d, T_d, ip_{tcs}, ip_p)
\end{array}$$

$/^*$ *A node receiving a reply packet rebroadcasts the packet and its certificate*$^*/$

$$\begin{array}{l}
\textsc{FwdReply}(ip, cert, T, ip_{tcs}, pkt, cert_d, ip_p) \stackrel{\text{def}}{=} \\
\quad \sigma!\langle\{pkt\}_{K_{ip}-}, [cert_d{\uplus}cert]\rangle_{ip_p} + \ \textsc{Node}(ip, cert, T, ip_{tcs}, ip_p)
\end{array}$$

message. In particular the source node waits for a pair $\langle reppkt, certList\rangle$ composed by a signed reply packet and a list of certificates. When the source receives a reply message, it extracts the ip addresses from the certificates and then it verifies their validity. If these verifications fail, the source node returns into the state AwaitReply($ip_s, ip_d, T_s, cert_s, ip_{tcs}$), otherwise it tries to extract the content of *reppkt*. This packet should contain two values: the tag `REP` and the ip of the destination. If the initiator cannot extract these information, it returns into the state AwaitReply($ip_s, ip_d, T_s, cert_s, ip_{tcs}$), otherwise it starts a number of checks. First, it verifies if it has received a reply packet; if it is not the case, it returns into the state AwaitReply($ip_s, ip_d, T_s, cert_s, ip_{tcs}$). Otherwise it verifies if the ip addresses of the source and the destination are the expected ones. If all these checks are successful, the initiator accepts the route, evolving into the state RouteSuccess(ip_s, ip_d, ip_p), meaning that the the intended destination ip_d has been reached and ip_p is the first hop in the route towards it, otherwise the reply is dropped and the node returns into the state Source($ip_s, ip_d, T_s, cert_s, ip_{tcs}$). The reply may never arrive at the source; in this case the source can nondeterministically return into the state Source($ip_s, ip_d, T_s, cert_s, ip_{tcs}$), starting again the protocol.

The other nodes taking part in the protocol may be either intermediate nodes or the destination node. In both cases, they begin the protocol in the state Node(ip, $cert$, T, ip_{tcs}, ip_p), where ip, ip_{tcs}, ip_p are the ip addresses of the node, the trust certificate server and the parent node, respectively, whereas $cert$ and T are the certificate and the trust table of the node, respectively. In this state, the node expects to receive either a route request message or a reply message. More precisely, it expects for a pair $\langle rpkt, certList\rangle$ composed by a signed packet and a list of certificates. When a node receives a message of this form, it first extracts the ip addresses from the certificates and then it verifies their validity. If the verification fails, the node returns into the state Node(ip, $cert$, T, ip_{tcs}, ip_p), otherwise it tries to extract the content of *rpkt*. This packet should contain two values: a type identifier and an ip address. If the initiator cannot extract these informations it returns into the state Node(ip, $cert$, T, ip_{tcs}, ip_p), otherwise it starts a number of checks. First, it verifies if the type identifier is `RDP`; if is so, then the node evolves into FwdRequest(ip, $cert$, T, pkt, ip_{tcs}, $cert_s$, ip_d, ip_s, ip_p). If the type identifier is `REP`, then the node evolves into FwdReply(ip, $cert$, T, ip_{tcs}, pkt, $cert_d$, ip_p). If these checks are not successful, the message is dropped and the node returns into the state Node($ip, cert, T, ip_{tcs}, ip_p$). Into the state FwdRequest(ip, $cert$, T, pkt, ip_{tcs}, $cert_s$, ip_d, ip_s, ip_p), the parameter pkt records the original message signed by the source, the parameter $cert_s$ records the certificate of the source, whereas the parameter ip_p records the parent node to which the node has to send back the reply. This is the ip address of the last certificate contained in *certList*. In this state the node first verifies if it is the destination of the route discovery packet. If this is the case, the node moves into the state SendReply($ip_d, cert_d, T_d, ip_{tcs}, ip_s, ip_p$), otherwise it signs the original packet pkt and adds its certificate to $cert_s$. Thus, it broadcasts the message $\langle \{pkt\}_{K_{ip-}}, [cert_s \uplus cert]\rangle$ and evolves into the state Node(ip, $cert$, T, ip_{tcs}, ip_p), waiting for the reply. In SendReply($ip_d, cert_d, T_d, ip_{tcs}$, ip_s, ip_p), the destination node prepares the reply message $\langle \{\texttt{REP}, id_s\}_{K_{ip_d-}}, [cert_d]\rangle$

and sends it to ip_p. The choice operator allows to avoid deadlocks in case the node ip_p becomes suddenly disconnected.

When a node in the state $\textsc{Node}(ip, cert, T, ip_{tcs}, ip_p)$ receives a reply, it evolves into the state $\textsc{FwdReply}(ip, cert, T, ip_{tcs}, pkt, cert_d, ip_p)$. The parameter pkt contains the original reply packet signed by the destination and $cert_d$ the certificate of the destination, whereas ip_p records the parent node to which the node has to send back the reply. In the state $\textsc{FwdReply}(ip, cert, T, ip_{tcs}, pkt, cert_d, ip_p)$ the node signs the reply message pkt and adds to $cert_d$ its certificate $cert$. Thus, it sends to ip_p the message $\langle \{pkt\}_{K_{ip}-}, [cert_d \uplus cert] \rangle$. Again, the choice operator in this state allows to avoid deadlocks in case of disconnections of ip_n.

Here we report an example of how the protocol works.

Example 8.15. Let M be the following network, where l, m and n are all valid nodes, where s is the source and n is the destination:

$$t \triangleright M \stackrel{\text{def}}{=} l[\textsc{Source}(l, n, T_l, cert_l, tcs)]_{T_l} \mid m[\textsc{Node}(m, cert_m, T_m, tcs, [])]_{T_m} \mid n[\textsc{Node}(n, cert_n, T_n, tcs, [])]_{T_n}$$

with $T_l(l, m) = T_l(l, n) = T_m(m, l) = T_m(m, n) = T_n(n, m) = \sigma$. The last parameter of the Node state has to record the parent node to which the reply has to be send. At the beginning it is empty. For convenience we define: $v_1{:=}\{\texttt{RDP}, n\}_{K_l-}$, $v_2{:=}\{\{\texttt{RDP}, n\}_{K_l-}\}_{K_m-}$, $v_3{:=}\{\texttt{REP}, l\}_{K_n-}$, $v_4{:=}\{\{\texttt{REP}, l\}_{K_n-}\}_{K_m-}$ and $w{:=}\langle l, \sigma \rangle_{t'}$ Here, we report the evolution of M while running the ARAN protocol.

$$\begin{aligned} t \triangleright M \xrightarrow{l!\langle v_1, [cert_l] \rangle \triangleright \{m,n\}}_{\sigma} & \; l[\textsc{AwaitReply}(l, n, T_l, cert_l, tcs)]_{T_l} \mid \\ & m[\textsc{FwdRequest}(m, cert_m, T_m, v_1, cert_l, n, l, l)]_{T_m} \mid \\ & n[\textsc{Node}(n, cert_n, T_n, tcs, [])]_{T_n} \\ \stackrel{\text{def}}{=} \; & t \triangleright M_1 \; . \end{aligned}$$

Node l starts the protocol in the state $\textsc{Source}(l, n, T_l, cert_l, tcs)$, broadcasting the message $\langle \{\texttt{RDP}, n\}_{K_l-}, [cert_l] \rangle$ and evolving into the state $\textsc{AwaitReply}(l, n, T_l, cert_l, tcs)$, waiting for a reply. Only node m receives the message and evolves into $\textsc{FwdRequest}(m, cert_m, T_m, v_1, cert_l, n, l, l)$.

$$\begin{aligned} t \triangleright M_1 \xrightarrow{m!w \triangleright \{l,n\}}_{\mathsf{trust}} & \; l[\textsc{AwaitReply}(l, n, T_l, cert_l, tcs)]_{T_l'} \mid \\ & m[\textsc{FwdRequest}(m, cert_m, T_m, v_1, cert_l, n, l, l)]_{T_m} \mid \\ & n[\textsc{Node}(n, cert_n, T_n, tcs, [])]_{T_n'} \\ \stackrel{\text{def}}{=} \; & t \triangleright M_2 \; . \end{aligned}$$

We now consider that n has asked for recommendation on l. We remember that we assume the presence of a trust manager component performing this kind of operations. Node m knows l; then by rule (SndRcmT) of Table 8.3 it broadcasts

the recommendation $\langle l, \sigma \rangle_{t'}$. Then, by rule (RcvRcmT) of Table 8.3, n can receive this information. We assume that the policy is such that it allows to add in the trust store of n the trust relation $\langle n, l, \sigma \rangle_t$, then $T'_n := T_n \cup \langle n, l, \sigma \rangle_t$.

$$\begin{aligned} t \triangleright M_2 \xrightarrow{m!\langle v_2, [cert_l, cert_m]\rangle \triangleright \{l,n\}}_{\sigma} & \; l[\textsc{AwaitReply}(l, n, T_l, cert_l, tcs)]_{T'_l} \mid \\ & \; m[\textsc{Node}(m, cert_m, T_m, tcs, l)]_{T_m} \mid \\ & \; n[\textsc{SendReply}(n, cert_n, T_n, tcs, l, m)]_{T'_n} \\ \stackrel{\text{def}}{=} & \; t \triangleright M_3 \; . \end{aligned}$$

Node m broadcasts the message $\langle \{\{\mathtt{RDP}, n\}_{K_{l-}}\}_{K_{m-}}, [cert_l, cert_m] \rangle$. Then, node m evolves into $\textsc{Node}(m, cert_m, T_m, tcs, l)$, waiting for a reply. Node l ignores the message sent by m and remains in the state $\textsc{AwaitReply}(l, n, T_l, cert_l, tcs)$, whereas node n receives the message, it verifies to be the destination node and it evolves into the state $\textsc{SendReply}(n, cert_n, T_n, tcs, l, m)$.

$$\begin{aligned} t \triangleright M_3 \xrightarrow{n!\langle v_3, [cert_n]\rangle \triangleright m}_{\sigma} & \; l[\textsc{AwaitReply}(l, n, T_l, cert_l, tcs)]_{T'_l} \mid \\ & \; m[\textsc{FwdReply}(m, cert_m, T_m, tcs, v_3, cert_n, l)]_{T_m} \mid \\ & \; n[\mathsf{nil}]_{T'_n} \\ \stackrel{\text{def}}{=} & \; t \triangleright M_4 \; . \end{aligned}$$

Node n prepares the reply $\{\mathtt{REP}, l\}_{K_{n-}}$ and sends it together with its certificate back to m. Node m receives this message, verifies the validity of the certificate, validates the signature and extracts the content of the message. Thus, it evolves into $\textsc{FwdReply}(m, cert_m, T_m, tcs, v_3, cert_n, l)$.

$$t \triangleright M_4 \xrightarrow{m!\langle v_4, [cert_n, cert_m]\rangle \triangleright l}_{\sigma} l[\textsc{RouteSuccess}(l, n, m)]_{T'_l} \mid m[\mathsf{nil}]_{T_m} \mid n[\mathsf{nil}]_{T'_n}.$$

Node m signs the original message received by the destination, adds its certificate and sends to l the message $\langle \{\{\mathtt{REP}, l\}_{K_{n-}}\}_{K_{m-}}, [cert_n, cert_m] \rangle$. Node l receives the message and verifies the validity of the certificates of m and n. It also verifies that the ip addresses of the source and of the destination are the expected ones. Then it accepts the route and evolves into the state $\textsc{RouteSuccess}(l, n, m)$.

8.8 Related Work

Process Calculi

A similar notion of time passing of TCTAN has been adopted by the following calculi. Gorrieri et al. in [100] have showed how the tCryptoSPA [98], a language for the description of cryptographic protocols in a real time setting, may be suitable to formally verify security aspects of wireless protocols. The authors have aimed

at enhancing analysis technique in order to cope with wireless protocols where a time synchronisation is required. Among the properties, they have checked timed secrecy, requiring that a message is kept secret only for a certain amount of time, and timed integrity, requiring that a message sent in a time interval has not been altered during the communication. They have used the approach based on non-interference developed in [86] as a method to express security properties. The effectiveness of their approach has been shown by studying the timed integrity property for μTESLA protocol [188]. They have follow the time synchronisation approach of [188], where synchronisation is required between a base station and the sensors in the network.

Godskesen and Nanz [95] have described how to extend a simple process calculus with realistic mobility models. The semantics of the calculus incorporates a notion of global time passing that allows to express a wide range of mobility models currently used in protocol development practice. The authors have considered a minimal calculus around the idea of time, as it is only used to determine the positions of nodes. Then only mobility part of the calculus is time dependent. They have developed a behavioural equivalence and pre-order that allow to compare the strength of mobility models in the formal setting described in the paper.

Timed Trust Models

Chakraborty and Ray [61] have proposed a memory model taking into consideration both the limitation set by the memory size dedicated to store trust-related data and the timing factor. This model is based on a sliding window. This memory model does not keep track of the user's past behaviour and, in case of dealing with a strategic attacker, such a user will be considered as a trusted one following a time interval greater than the sliding window. A problem is related to the choice of the window size. A small window size will lead to the fast "forgiving" of malicious users, while a large window size increases the amount of data to be processed and slows down the decision making.

An improvement of the memory model of [61] has been proposed by Giang et al. in [93] by introducing a forgetting factor. A feedback or observation value from each party is calculated as a sum of the current feedback and the product of the forgetting factor and the previous feedback. In such a way an actual feedback value contains information about previous interactions. A disadvantage of this model is that the final feedback value does not depend on the time factor and recent information has the same value as old information.

Another approach to the users behaviour history recording, based on fading memories, is proposed in the TrustGuard system [217]. It is assumed that the system stores a limited number of reputation-based trust values instead of results of interactions. Recent past observations are aggregated over time intervals. Using this model, the system needs to recalculate the recorded history after each interaction. Such a model prevents a malicious node from regaining trust quickly by keeping track of the previous bad behaviour but by the same reason the trust value for a malicious node decreases slowly if this node has shown good behaviour in the past.

Gorla, Hennessy and Sassone in [97] have extended the RT approach of [147] to include time validity and boolean guards that control the availability of credentials.

In such an extended framework, credentials are conditional on the availability of supporting credentials in the execution context. In addition to a set-theoretic and a logic-programming semantics, they have developed for the extended language a series of increasingly powerful inference systems for establishing these conditional credentials. Thus they have given a purely operational interpretation of RT credentials, giving a set of inference rules for deriving credentials from a set of RT statements. This inference system is an explicit formalisation of the intuitive meaning of RT statements and provides a convenient way of working with them.

8.9 Chapter Summary

In this chapter we introduced the TCTAN calculus, a timing extension of the CTAN of Chapter 7. We focused on the relationship between time and trust in the setting of MANETs. In trust models for ad hoc networks, the timing factor is important because more recent trust information should have more influence on the trust establishment process. We introduced a timed variant of our trust model proposed in Section 7.2, where a timestamp on assertions allows to reason about their validity. As in TCWS, time proceeds in discrete steps represented by occurrences of a simple action tick. A global clock is supposed to be updated whenever all the nodes agree on this. We worked on configurations $t \triangleright M$, where t indicates the global time and M is a network. Then we provided an operational semantics separating instantaneous actions and actions involving time. We reformulated the reduction barbed congruence of Section 7.6 and the bisimulation of Section 7.7 in terms of configurations. Then we showed that the properties of CTAN are preserved by TCTAN. Moreover we proved that the same time properties of TCWS of Section 6.4 are also satisfied by TCTAN. Finally, we applied the calculus to formalise the ARAN protocol. This protocol uses digital certificates that contain timing information expressing their validity.

9

Conclusions

Wireless communication is a broad and dynamic field that has drawn the attention of many researchers and has stimulated important technological advances over the last few decades. Besides the enormous proliferation of "engineering" papers, in the last years the "theoretical" computer science research community has addressed particular attention to formal specification and analysis of wireless systems. This constitutes a very good starting point to provide a comprehensive understanding of the fundamental principles of wireless communication.

9.1 Contributions

In this dissertation, we dealt with the question of how apply the well-known technique of process calculus to the modelling of important features of wireless systems. Moreover, we dealt with trust-based security to develop security mechanisms specifically tailored for wireless systems. Considering that these systems cover large and complex areas, we proposed three different process calculi and in each of them we focused on some specific aspects.

A Timed Calculus for Wireless Systems

We proposed a timed process calculus for wireless systems called TCWS in which we focused on timing aspects and communication interferences. This is the first calculus dealing with time and interferences for wireless systems. Other calculi rely on the presence of some MAC-level protocol to remove interferences. However, in wireless systems *collisions cannot be avoided* although there are protocols to reduce their occurrences.

TCWS models *local broadcast communications* and *time passing*, operating a distinction in the labelled transition system between transitions for modelling sending and receiving actions and transitions for modelling the passage of time. We showed that our calculus enjoys some desirable *timed properties*. We then proposed a notion of *well-formedness* over the networks and we proved that it is preserved at run-time. We illustrated the usability of the calculus to model the *Carrier Sense Multiple Access* scheme [123], a widely used MAC level protocol in

which a device senses the channel before transmitting. We provided as main *behavioural equivalence* a timed variant of the reduction barbed congruence [172]. As usual, when working with barbed congruence, it is useful to define a labelled bisimilarity as a proof-method. Thus we showed a *soundness* result which states that our labelled bisimilarity implies our timed barbed reduction congruence. Then we proved a number of *algebraic laws* on well-formed networks using our bisimulation proof-technique.

A Calculus of Trustworthy Ad hoc Networks

We then proposed another process calculus for MANETs called CTAN in which we focused on *mobility* of nodes and *trust-based* security. We embodied a *trust model* in the calculus; trust relations among nodes are represented by an association of a *security level* to them. Thus our networks are modelled as *multilevel systems* [28]. The main objective of the model is to isolate bad nodes, i.e. nodes which do not behave as expected. For this reason, we supported *node revocation*. According to our knownledge, this is the first process calculus for wireless networks embodying a trust model.

We proposed a simple *mobility model*, following the approach of [92], for which our calculus does not directly model the network topology neither in the syntax nor in the semantics but topology changes are added at semantics level.

We showed that CTAN enjoys *safety properties*, aiming at guaranteeing that only authorised nodes receive sensible information. We also provided security variants of *reduction barbed congruence* and *labelled bisimilarity*, proving as above a *soundness* theorem. We used our bisimilarity to prove a *non-interference* result, for which high-level behaviours can arbitrarily change without affecting low-level equivalences. Finally, we used our calculus to formalise and analyse a *secure* version of the leader election algorithm for MANETs [227]. We then provided an encoding of the *endairA* routing protocol for ad hoc networks [9].

Time vs Trust

Our last process calculus *TCTAN* is a simple *time extension* of the previous calculus CTAN. We treated timing aspects to express the relationship between trust and time. We used our calculus to formalise the secure, on-demand routing protocol *ARAN* [209] that uses cryptographic certificates with a time validity.

9.2 Future Work

Many aspects could be considered for future directions.

Multiple channels

In TCWS we assumed the presence of a unique channel. However, new techniques have been developed in the last years to provide several virtual channels. The most known techniques [222] are *Frequency Division Multiplexing* (FDM), in which the signal is split into many narrow bands, and *Time Division Multiplexing* (TDM), in which the time domain is divided into several recurrent timeslots of fixed length, one for each sub-channel. A generalisation of TCWS with multiple channels (à la CCS) should be straightforward.

Probabilistic models

Probabilistic models are nowadays widely used in the design and verification of complex systems in order to quantify unreliable or unpredictable behaviour in security, performance and reliability analysis. Probability is taken into account when analysing quantitative security properties (measuring, in a sense, the security level of the protocol) or when dealing with probabilistic protocols. We can find many probabilistic frameworks based on process calculus approach applied to security analysis. Just as an example, we cite [11, 12, 101, 173]. We believe that TCWS represents a solid basis to develop *probabilistic theories* to do quantitative evaluations on collisions, and more generally on node failures.

Security analysis

The research community quite agrees on the belief that formal specification and analysis are good directions for the development of secure systems. The formalism of process calculus has been widely applied in this context, after that it has been extended to cope with the possibility to detect flaws in communication and security protocols. Just as an example of applications to wireless system *security analysis*, we cite [94, 177, 178, 214]. In particular, in [177, 178] the authors have sketched an attacker model that can be expressed in their calculus. Then they have described the problem of communication protocol security as the ability to provide the desired network service in an adversary environment. Hence a security condition, fully integrated into the framework and then derivable automatically, has been expressed via a notion of equivalence. This condition is parametric with respect to the attacker model, the network topology, and the data format used by the protocol. The authors have applied the calculus introduced in their works to the modelling of real-world protocols and they have been able to detect some distinct exploitable flaws and have detected omissions and ambiguities in the protocols they analysed.

We used our calculi CTAN and TCTAN to model routing protocols for MANETs but we did not deal with their security properties. This is could be a possible future direction of our research, along the lines of the papers cited above.

History-based trust

We developed behavioural-based trust models for MANETs. Behaviour-based systems are often called experience-based as in these models an entity A trusts another entity B based on its experience on B's past behaviour. Then, besides the concept of reputation, these systems heavily rely on the concept of *history experience*. Many of the papers cited in Section 3.4 take into account also past evidence during the evaluation process. In the same section we also cited some probabilistic approaches to trust management. In general, probabilistic systems work by assuming a particular probabilistic model, say λ, for the behaviour of principals. The goal is to predict the behaviour of principals in future interactions, given their behaviour in past interactions and the model λ, i.e., computing a probability $P(\text{next} \mid \text{past}, \lambda)$. Thus probability and past behaviour aspects can be treated together. Another possible future direction may be considering these aspects in our trust models.

A

Appendix

A.1 Proofs of Chapter 6

In order to prove Proposition 6.10 on the Exposure consistency property, we use the following auxiliary lemmas.

Lemma A.1. *Let* $M \xrightarrow{\lambda} M'$ *with* $\lambda \in \{m!v, m?v\}$ *such that* $M \equiv \prod_{i\in} n_i[W_i]^{\nu_i}_{t_i}$ *and* $M' \equiv \prod_{i\in I} n_i[W'_i]^{\nu_i}_{t'_i}$.

1. *If* $\lambda = m?v$ *then* $m \neq n_i$, *for all* i.
2. *If* $\lambda = m!v$ *then there is* $i \in I$ *such that* $m = n_i$, $W_i = !\langle v\rangle.P_i$ *and* $W'_i = \langle v\rangle^{\delta_v}.P_i$.
3. *If* $m \notin \nu_i$, *for some* i, *then* $t'_i = t_i$; *if also* $m \neq n_i$, *then* $W'_i = W_i$.
4. *If* $m \in \nu_i$, *for some* i, *then* $t'_i = \max(t_i, \delta_v)$.
5. *If* $m \in \nu_i$ *and* $W'_i{=}(x)_w.P_i$, *for some* i, *and* $w{\neq}\perp$, *then* $w = m{:}v$, $t_i = 0$, $t'_i = \delta_v$, *and* $W_i \ ::= \ ?(x).P_i \ \mid \ \lfloor ?(x).P_i \rfloor Q_i$,
6. *If* $W_i = \langle w\rangle^r.P_i$, *for some* i, *then* $W'_i = W_i$.
7. *If* $m \neq n_i$ *and* $W'_i = \langle w\rangle^r.P_i$, *for some* i, *then* $W_i = W'_i$.

Proof The results easily follow by transition induction. □

Lemma A.2. *Let* $M \xrightarrow{\mathsf{tick}} M'$ *such that* $M \equiv \prod_{i\in} n_i[W_i]^{\nu_i}_{t_i}$ *and* $M' \equiv \prod_{i\in I} n_i[W'_i]^{\nu_i}_{t'_i}$.

1. $t'_i = t_i \ominus 1$, *for all* i.
2. *If* $W'_i = (x)_v.P$, *for some* i, *then*
   - *either* $W_i = W'_i$
   - *or* $W_i \ ::= ?(x).P \ \mid \ \lfloor ?(x).P \rfloor Q$ *and* $v = \perp$
3. *If* $W'_i = \langle w\rangle^r.P$, *for some* i, *then* $W_i = \langle w\rangle^{r+1}.P$.

Proof The results easily follow by transition induction. □

Proof of Proposition 6.10

The proof proceeds by transition induction on the derivation of $M \xrightarrow{\lambda} M'$, for $\lambda \in \{m!v, m?v, \mathsf{tick}\}$. We show the remaining cases. The other ones are at page 90.

- Let $M \xrightarrow{m?v} M'$ by an application of rule (RcvPar) with $M = M_1 \mid M_2$, $M_1 \xrightarrow{m?v} M_1'$, $M_2 \xrightarrow{m?v} M_2'$, and $M' = M_1' \mid M_2'$, where both M_1' and M_2' are exposure consistent by inductive hypothesis. We have to prove that M' respects the clauses of Definition 6.5.
  - Clauses 1-2. We reason as in the case of the the sending action $m!v$ examined above.
  - Clause 3. Let $M' \equiv \prod_i n_i[W_i']^{\nu_i}_{t_i'} \mid h[\langle v\rangle^r.P]^{\nu_h}_{t_h'} \mid n[W']^{\nu_n}_{t_n'}$, with $h \in \nu_n$. We have to prove that $r \leq t_n'$. We only consider the case when $h \in \mathsf{nds}(M_1)$ and $n \in \mathsf{nds}(M_2)$ (or viceversa). The other cases are easier. By Lemma A.1(1) it holds that $m \notin \mathsf{nds}(M')$. By A.1(7) it follows:

$$M \equiv \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid h[\langle v\rangle^r.P]^{\nu_h}_{t_h} \mid n[W]^{\nu_n}_{t_n}$$

    for appropriate processes and tags. Now, if $m \in \nu_n$ by Lemma A.1(4) we have $t_n' = \max(t_n, \delta_v)$. As M is exposure consistent it holds that $r \leq t_n$ and hence also $r \leq t_n'$. On the other hand, if $m \notin \nu_n$ by an application of Lemma A.1(3) we have $t_n' = t_n$; as M is exposure consistent it follows that $r \leq t_n = t_n'$.
  - Clause 4. Let

$$M \equiv N \mid n[W]^{\nu}_{t} = \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid n[W]^{\nu}_{t}$$

    and

$$M' \equiv N' \mid n[W']^{\nu}_{t'} = \prod_i n_i[W_i']^{\nu_i}_{t_i'} \mid n[W']^{\nu}_{t'}$$

    with $t' > 0$ and $\mathsf{active}(k', N') \neq t'$ for all $k' \in \nu \cap \mathsf{actsnd}(N')$. We have to prove that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N')$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i' \geq t'$. There are two cases.
    - Let $m \notin \nu$. By Lemma A.1(3) we have $t' = t$. By Lemma A.1(6), it follows that $\mathsf{actsnd}(N) \subseteq \mathsf{actsnd}(N')$. As a consequence, $\nu \cap \mathsf{actsnd}(N) \subseteq \nu \cap \mathsf{actsnd}(N')$. Since $t' = t$ we can derive that for all $k \in \nu \cap \mathsf{actsnd}(N)$ it holds that $\mathsf{active}(k, N') \neq t$. By Lemma A.1(6) and Lemma A.1(7) if $k \neq m$ then $\mathsf{active}(k, N) = \mathsf{active}(k, N')$. Since $m \notin \nu$ it follows that for all $k \in \nu \cap \mathsf{actsnd}(N)$ it holds $\mathsf{active}(k, N) \neq t$. Since M is exposure consistent it follows that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N)$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i \geq t$. Notice that $\nu \setminus \mathsf{nds}(N) = \nu \setminus \mathsf{nds}(N')$. Moreover, by Lemma A.1(3) and A.1(4) we have $t_i \leq t_i'$, for all i. This allows us to derive that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N')$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i' \geq t_i \geq t = t'$.
    - Let $m \in \nu$. By Lemma A.1(1) we have $m \notin \mathsf{nds}(M)$. By Lemmas A.1(6) and A.1(7) for all $k \in \mathsf{nds}(N)$ it holds that $\mathsf{active}(k, N) = \mathsf{active}(K, N')$.

As a consequence, $\mathsf{actsnd}(N) = \mathsf{actsnd}(N')$, and hence $\nu \cap \mathsf{actsnd}(N) = \nu \cap \mathsf{actsnd}(N')$. By Lemma A.1(4) we have $t' = \max(t, \delta_v)$. So, there are two cases.

- Let $\delta_v \leq t$. Then $t' = t$ and for all $k \in \nu \cap \mathsf{actsnd}(N)$ it holds $\mathsf{active}(k, N) \neq t$. Since M is exposure consistent it follows that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N)$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i \geq t$. Notice that $\nu \setminus \mathsf{nds}(N) = \nu \setminus \mathsf{nds}(N')$. Moreover, by Lemmas A.1(3) and A.1(4) we have $t_i \leq t'_i$, for all i. This allows us to derive that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N')$ such that if $\hat{k} \in \nu_i$, for some i, then $t'_i \geq t_i \geq t = t'$.
- Let $\delta_v > t$. Then $t' = \delta_v$. In this case, there is $m \in \nu \setminus \mathsf{nds}(N')$ such that if $m \in \nu_i$, for some i, then by Lemma A.1(4) it holds that $t'_i = \max(t_i, \delta_v)$. Thus, $t'_i \geq \delta_v = t'$.

- Let $M \xrightarrow{\mathsf{tick}} M'$ by an application of rule (Par-tick) with $M = M_1 \mid M_2$, $M_1 \xrightarrow{\mathsf{tick}} M'_1$ and $M_2 \xrightarrow{\mathsf{tick}} M'_2$, and $M' = M'_1 \mid M'_2$, where both M'_1 and M'_2 are exposure consistent by inductive hypothesis. We have to prove that M' respects the clauses of Definition 6.5.
  - Clauses 1-2. It is easy to show that M' is exposure consistent. The results follow by inductive hypothesis.
  - Clause 3. Let

$$M' \equiv \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid h[\langle v \rangle^r.P]^{\nu_h}_{t'_h} \mid n[W']^{\nu_n}_{t'_n}$$

    with $h \in \nu_n$. We have to prove that $r \leq t'_n$. We only consider the case when $h \in \mathsf{nds}(M_1)$ and $n \in \mathsf{nds}(M_2)$ (or viceversa). The other cases are easier. By Lemma A.2(1) and A.2(3) we have

$$M \equiv \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid h[\langle v \rangle^{r+1}.P]^{\nu_h}_{t_h} \mid n[W]^{\nu_n}_{t'_n+1}$$

    for appropriate processes and tags. As M is exposure consistent, it follows that $r \leq t'_n$.
  - Clause 4. Let

$$M \equiv N \mid n[W]^{\nu}_{t} = \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid n[W]^{\nu}_{t}$$

    and

$$M' \equiv N' \mid n[W']^{\nu}_{t'} = \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid n[W']^{\nu}_{t'}$$

    with $t' > 0$ and $\mathsf{active}(k', N') \neq t'$ for all $k' \in \nu \cap \mathsf{actsnd}(N')$. We have to prove that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N')$ such that if $\hat{k} \in \nu_i$, for some i, then $t'_i \geq t'$. By Lemma A.2(1) we have $t' = t \ominus 1$. Since $t' > 0$ it follows that $t > 1$. Moreover, by Lemma A.2(3), if $W'_i = \langle w \rangle^{r'}.Q$, for some i, then $W_i = \langle w \rangle^r.Q$, with $r' = r \ominus 1$. As a consequence, $\mathsf{actsnd}(N') \subseteq \mathsf{actsnd}(N)$. By Lemma A.2(3) if $\mathsf{active}(k', N') \neq t'$ then $\mathsf{active}(k', N) \neq t' + 1 = t$. Notice also that $\mathsf{active}(k, N) = 1$ for all $k \in \mathsf{actsnd}(N) \setminus \mathsf{actsnd}(N')$. Thus, since $t > 1$ for all $k \in \nu \cap \mathsf{actsnd}(N)$ it holds that $\mathsf{active}(k, N) \neq t$. Since

M is exposure consistent it follows that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N)$ such that if $\hat{k} \in \nu_i$, for some i, then $t_i \geq t$. Notice that $\nu \setminus \mathsf{nds}(M) = \nu \setminus \mathsf{nds}(M')$. Moreover, by Lemma A.2(1) we have $t'_i = t_i \ominus 1$, for all i. This allows us to derive that there is $\hat{k} \in \nu \setminus \mathsf{nds}(N')$ such that if $\hat{k} \in \nu_i$, for some i, then $t'_i \geq t'$.

□

Proof of Proposition 6.11

Let us consider all the possible values of λ. We show remaining cases. The other ones are at page 92.

- Let $M \xrightarrow{m?v} M'$ by an application of rule (RcvPar) with $M = M_1 \mid M_2$, $M_1 \xrightarrow{m?v} M'_1$, $M_2 \xrightarrow{m?v} M'_2$, and $M' = M'_1 \mid M'_2$, where both M'_1 and M'_2 are transmission consistent by inductive hypothesis. We have to prove that M' respect the clauses of Definition 6.6.
  - Clause 1. Let

$$M' \equiv N' \mid n[(x)_w.Q]^{\nu_n}_{t'_n} = \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid n[(x)_w.Q]^{\nu_n}_{t'_n}$$

    with $w \neq \bot$. We have to prove that $\mid \mathsf{actsnd}(N') \cap \nu_n \mid \leq 1$. There are two possibilities.
    - If $m \notin \nu_n$ then by Lemma A.1(3) we have

$$M \equiv N \mid n[(x)_w.Q]^{\nu_n}_{t_n} = \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid n[(x)_w.Q]^{\nu_n}_{t_n} \ .$$

      Since M is transmission consistent, we have $\mid \mathsf{actsnd}(N) \cap \nu_n \mid \leq 1$. By Lemmas A.1(6) and A.1(7) we derive $\mathsf{actsnd}(N') = \mathsf{actsnd}(N)$. This allows us to derive that $\mid \mathsf{actsnd}(N') \cap \nu_n \mid \leq 1$.
    - If $m \in \nu_n$, since $w \neq \bot$, by Lemma A.1(5) we have

$$M \equiv N \mid n[W]^{\nu_n}_0 = \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid n[?(x).Q]^{\nu_n}_0$$

      where $t'_n = \delta_v$ and $W = ?(x).Q$ (the case $W = \lfloor ?(x).Q \rfloor R$ is similar). By Lemmas A.1(6) and A.1(7) we derive $\mathsf{actsnd}(N') = \mathsf{actsnd}(N)$. Since M is exposure consistent, by clause 3 of Definition 6.5 we derive $\mid \mathsf{actsnd}(N) \cap \nu_n \mid = 0$. As a consequence, $\mid \mathsf{actsnd}(N') \cap \nu_n \mid = 0$.
  - Clause 2. Let

$$M' \equiv \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid h[\langle w_1 \rangle^r.P]^{\nu_h}_{t'_h} \mid n[(x)_{w_2}.Q]^{\nu_n}_{t'_n}$$

    with $h \in \nu_n$ and $w_2 \neq \bot$. We have to show that $w_2 = m{:}w_1$ and $r = t'_n$. By Lemma A.1(1) we have $h \neq m$. By Lemmas A.1(7) we have

$$M \equiv \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid h[\langle w_1 \rangle^r.P]^{\nu_h}_{t_h} \mid n[W]^{\nu_n}_{t_n}$$

    for appropriate processes and tags. Now, there are two cases.

  - If $m \notin \nu_n$ then by Lemma A.1(3) we have $W = (x)_{w_2}.Q$ and $t'_n = t_n$. Since M is transmission consistent it follows that $w_2 = m : w_1$ and $t'_n = r$.
  - If $m \in \nu_n$ then by Lemma A.1(5) we have $W = ?(x).Q$ (the case $W = \lfloor ?(x).P \rfloor Q$ is similar) and $t_n = 0$. Since M is exposure consistent, by clause 3 of Definition 6.5 it should be $t_n > 0$. This contradiction shows that this case is not possible.

– Clause 3. Let

$$M' \equiv N' \mid n[(x)_w.P]^{\nu_n}_{t'_n} = \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid n[(x)_w.P]^{\nu_n}_{t'_n}$$

with $\mid \mathsf{actsnd}(N') \cap \nu_n \mid > 1$. We have to show that $w = \bot$. By Lemma A.1 we have

$$M \equiv \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid n[W]^{\nu_n}_{t_n}$$

for appropriate processes and tags. Since $\mid \mathsf{actsnd}(N') \cap \nu_n \mid > 1$ it follows that $W'_j = \langle w_j \rangle^{r_j}.P_j$ and $W'_k = \langle w_k \rangle^{r_k}.P_k$, for some j and k such that $\{n_j, n_k\} \subseteq \nu_n$. By Lemma A.1(1) and Lemma A.1(7) we have $W_j = W'_j$ and $W_k = W'_k$. At this point we reason by contradiction. Suppose $w \neq \bot$. Then, by Lemma A.1(5) we have $W = ?(x).P$ (the case $W = \lfloor ?(x).P \rfloor Q$ is similar) and $t_n = 0$. However, since M is exposure consistent, by clause 3 of Definition 6.5 it must be $t_n > 0$. This contradiction allows us to derive that $w = \bot$.

- Let $M \xrightarrow{\mathsf{tick}} M'$ by an application of rule (Par-tick) with $M = M_1 \mid M_2$, $M_1 \xrightarrow{\mathsf{tick}} M'_1$ and $M_2 \xrightarrow{\mathsf{tick}} M'_2$, and $M' = M'_1 \mid M'_2$, where both M'_1 and M'_2 are transmission consistent by inductive hypothesis. We have to prove that M' respects the clauses of Definition 6.6. Let us examine the three clauses one by one.

  – Clause 1. Let

$$M' \equiv N' \mid n[(x)_w.Q]^{\nu_n}_{t'_n} = \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid n[(x)_w.Q]^{\nu_n}_{t'_n}$$

  with $w \neq \bot$. We have to prove that $\mid \mathsf{actsnd}(N') \cap \nu_n \mid \leq 1$. By Lemma A.2(2), since $w \neq \bot$, it must be

$$M \equiv N \mid n[(x)_w.Q]^{\nu_n}_{t_n} = \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid n[(x)_w.Q]^{\nu_n}_{t_n}$$

  Since M is transmission consistent it follows that $\mid \mathsf{actsnd}(N) \cap \nu \mid \leq 1$. By Lemma A.2(3) it follows that $\mathsf{actsnd}(N') \subseteq \mathsf{actsnd}(N)$. This implies $\mid \mathsf{actsnd}(N') \cap \nu \mid \leq 1$.

  – Clause 2. Let

$$M' \equiv \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid h[\langle w_1 \rangle^r.P]^{\nu_h}_{t'_h} \mid n[(x)_{w_2}.Q]^{\nu_n}_{t'_n}$$

  with $h \in \nu_n$ and $w_2 \neq \bot$. We have to show that $w_2 = m{:}w_1$ and $r = t'_n$. Since $w_2 \neq \bot$, by Lemmas A.2(1), A.2(2), and A.2(3)

$$M \equiv \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid h[\langle w_1\rangle^{r+1}.P]^{\nu_h}_{t_h} \mid n[(x)_{w_2}.Q]^{\nu_n}_{t'_n+1} \ .$$

Since M is transmission consistent we have $w_2 = m{:}w_1$ and $r+1 = t'_n+1$. As a consequence, $r = t'_n$.

– Clause 3. Let

$$M' \equiv N' \mid n[(x)_w.P]^{\nu_n}_{t'_n} = \prod_i n_i[W'_i]^{\nu_i}_{t'_i} \mid n[(x)_w.P]^{\nu_n}_{t'_n}$$

with $\mid \mathsf{actsnd}(N') \cap \nu_n \mid > 1$. We have to show that $w = \bot$. By an application of Lemma A.2(2) there are two possibilities:

· Either

$$M \equiv N \mid n[(x)_w.P]^{\nu_n}_{t_n} = \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid n[(x)_w.P]^{\nu_n}_{t_n} \ .$$

In this case, by Lemma A.2(3) it follows that $\mathsf{actsnd}(N') \subseteq \mathsf{actsnd}(N)$. Thus $\mid \mathsf{actsnd}(N') \cap \nu_n \mid > 1$ implies $\mid \mathsf{actsnd}(N) \cap \nu_n \mid > 1$. Since M is transmission consistent it follows that $w = \bot$.

· Or

$$M \equiv N \mid n[W]^{\nu_n}_{t_n} = \prod_i n_i[W_i]^{\nu_i}_{t_i} \mid n[W]^{\nu_n}_{t_n}$$

where W is either a receiver or a receiver with timeout, and $w = \bot$.

□

In order to prove Theorem 6.15 on the Patience property, we use the following auxiliary lemmas.

Lemma A.3. *Let N be a well-formed network such that $m \notin \mathsf{nds}(N)$. Then $N \xrightarrow{m?v} N'$, for some network N'.*

Proof Let us proceed by induction on the structure of N.

- Let $N = \mathbf{0}$. By an application of rule (Zero) we have $N \xrightarrow{m?v} N$.
- Let $N = n[W]^{\nu}_t$, with $W ::= P \mid A$. Let us proceed by induction on the structure of W.
  - Let $W = \mathsf{nil}$. There are two cases.
    · If $m \notin \nu$ then by an application of rule (OutRng) we have $N \xrightarrow{m?v} N'$.
    · If $m \in \nu$ then by an application of rule (Exp) we have $N \xrightarrow{m?v} N'$ with $N' = n[\mathsf{nil}]^{\nu}_{t'}$, where $t' = \max(t, \delta_v)$.
  - Let $W = !\langle v\rangle.P$. This case is similar to the previous one.
  - $W = \mathsf{tick}.P$. This case is similar to the previous one.
  - $W = \langle v\rangle^r.P$. This case is similar to the previous one.
  - Let $W = ?(x).P$. There are three sub-cases.
    · If $m \notin \nu$ by an application of rule (OutRng) we have $N \xrightarrow{m?v} N$.
    · If $t = 0$ and $m \in \nu$ there are two cases:
      · by an application of rules (RcvP) and (Rcv) we can derive $N \xrightarrow{m?v} N'$, with $N' = n[(x)_v.P]^{\nu}_{\delta_v}$;

  - by an application of rule (Exp) we can derive $N \xrightarrow{m?v} N'$, with $N' = n[?(x).P]^{\nu}_{\delta_v}$.
  - If $t > 0$ and $m \in \nu$ then by an application of rule (Exp) we have $N \xrightarrow{m?v} N'$, with $N' = n[?(x).P]^{\nu}_{t'}$ where $t' = \max(t, \delta_v)$.
- Let $W = \lfloor ?(x).P \rfloor Q$. This case is similar to the previous one.
- Let $W = (x)_w.P$. There are two sub-cases.
  - If $m \in \nu$ then by an application of rule (Coll), it holds that $N \xrightarrow{m?v} N' = n[(x)_{\perp}.P]^{\nu}_{t'}$ with $t' := \mathsf{max}(t,\delta_v)$.
  - If $m \notin \nu$ then by an application of rule (OutRng) we have $N \xrightarrow{m?v} N$.
- Let $W = [v = v]P_1, P_2$. By an application of rule (Then) we can apply the inductive hypothesis to conclude that the statement holds.
- Let $W = [v_1 = v_2]P_1, P_2$, with $v_1 \neq v_2$. By an application of rule (Else), this case is similar to the previous one.
- Let $W = H\langle \tilde{v} \rangle$. The constraint on guarded recursion ensures us that by an application of rule (Rec) we can apply the inductive hypothesis to conclude that the statement holds.

- Let $N = N_1 \mid N_2$. By inductive hypothesis it holds that $N_1 \xrightarrow{m?v} N_1'$ and $N_2 \xrightarrow{m?v} N_2'$, for some N_1', N_2'. By an application of rule (RcvPar) it holds that $N \xrightarrow{m?v} N'$, for $N' = N_1' \mid N_2'$.

□

Lemma A.4. *Let M be a well-formed network. If $M \xrightarrow{m!v} M'$ then for all network N such that $M \mid N$ is a well-formed network it holds that $M \mid N \xrightarrow{m!v} M' \mid N'$ for some network N'.*

Proof The result follows by Lemma A.3 and an application of rule (Sync). □

Proof of Theorem 6.15

We show the remaining cases. The other ones are at page 95.

- Let $M = n[W]^{\nu}_t$. We proceed by induction on the structure of P, showing the remaining case. The other ones are at page 95.
  - If $W = \mathsf{nil}$ and $t > 0$ then $M \xrightarrow{\mathsf{tick}} n[P]^{\nu}_{t-1}$ by an application of rules (Nil-tick) and (Time-t). Thus the statement does not apply.
  - If $W = ?(x).P$ and $t = 1$ then $M \xrightarrow{\mathsf{tick}} n[\{^{\perp}/_{x}\}P]^{\nu}_0$, by an application of rules (RcvP) and (RcvFail). Thus the statement does not apply.
  - If $W = ?(x).P$ and $t > 1$ then $M \xrightarrow{\mathsf{tick}} n[(x)_{\perp}.P]^{\nu}_{t-1}$, by an application of rules (RcvP) and (RcvFail). Thus the statement does not apply.
  - If $W = \mathsf{tick}.P$ and $t > 0$ then $M \xrightarrow{\mathsf{tick}} n[W]^{\nu}_{t-1}$ by an application of rules (Sigma) and (Time-t). Thus the statement does not apply.
  - If $W = \lfloor ?(x).P \rfloor Q$ and $t = 1$ then $M \xrightarrow{\mathsf{tick}} n[\{^{\perp}/_{x}\}P]^{\nu}_0$, by an application of rules (RcvP) and (RcvFail). and the statement does not apply.
  - If $W = \lfloor ?(x).P \rfloor Q$ and $t > 1$ then $M \xrightarrow{\mathsf{tick}} n[(x)_{\perp}.P]^{\nu}_{t-1}$, by an application of rules (RcvP) and (RcvFail), Thus the statement does not apply.

- If $W = [v = v]P_1, P_2$ then by an application of rule (Then) we can apply the inductive hypothesis to conclude that we fall in one of the previous cases.
- If $W = [v_1 = v_2]P_1, P_2$, with $v_1 \neq v_2$, by an application of rule (Else) we can apply the inductive hypothesis to conclude that we fall in one of the previous cases.
- If $W = H\langle\tilde{v}\rangle$ the constraint of guarded recursion ensures us that by an application of rule (Rec) we can apply the inductive hypothesis and we fall in one of the previous cases.
- If $W = \langle v\rangle^r.P$, with $r > 1$, and $t > 0$ then by an application of rules (ActSnd) and (Time-t) we have $M \xrightarrow{\mathsf{tick}} n[\langle v\rangle^{r-1}.P]^{\nu}_{t-1}$ and the statement does not apply.
- If $W = \langle v\rangle.P$ and $t = 0$, then by an application of rules (ActSnd) and (Time-0) we have $M \xrightarrow{\mathsf{tick}} n[P]^{\nu}_0$ and the statement does not apply.
- If $W = \langle v\rangle.P$ and $t > 0$, then by an application of rules (ActSnd) and (Time-t) we have $M \xrightarrow{\mathsf{tick}} n[P]^{\nu}_{t-1}$ and the statement does not apply.
- If $W = (x)_v.P$, with $t = 0$, then by an application of rules (ActRcv) and (Time-0) we have $M \xrightarrow{\mathsf{tick}} n[\{^v/_x\}P]^{\nu}_0$ and the statement does not apply.

□

Proof of Theorem 6.24

We show the remaining cases. The other ones are at page 100.

- Let $M \mid O \xrightarrow{\tau} \widehat{M}$, by an application of rule (Shh), because $M \mid O \xrightarrow{m!v} \widehat{M}$ and $\mathsf{ngh}(m, M \mid O) \subseteq \mathsf{nds}(M \mid O)$.
  There are two possible cases:
  - Let $M \mid O \xrightarrow{m!v} \widehat{M}$, by an application of rule (Sync), because $M \xrightarrow{m!v} M'$ and $O \xrightarrow{m?v} O'$, with $\widehat{M} = M' \mid O'$ and

    $$\mathsf{ngh}(m, M \mid O) \setminus \mathsf{nds}(M \mid O) = (\mathsf{ngh}(m, M) \setminus \mathsf{nds}(M)) \setminus \mathsf{nds}(O) = \emptyset \ .$$

    Again there are two possibilities:
    - Let $\mathsf{ngh}(m, M) \setminus \mathsf{nds}(M) = \emptyset$. Then, by an application of rule (Shh) we have $M \xrightarrow{\tau} M'$. Since $M \approx N$ there is N' such that $N \Rightarrow N'$ and $M' \approx N'$. We know that $O \xrightarrow{m?v} O'$. Let us assume $O \neq \mathbf{0}$ (the case when $O = \mathbf{0}$ is simple). In a network composed by several parallel components the action $m?v$ can be derived only by an application of rule (RcvPar). By definition of our networks there are n_i, W_i, ν_i, and t_i, for $1 \leq i \leq k$, such that $O = \prod_{i=1}^{k} n_i[W_i]^{\nu_i}_{t_i}$. By definition of rule (RcvPar), $O \xrightarrow{m?v} O'$ if and only if for all i, $1 \leq i \leq k$, there are W_i', ν_i', and t_i' such that

      $$n_i[W_i]^{\nu_i}_{t_i} \xrightarrow{m?v} n_i[W_i']^{\nu_i'}_{t_i'}$$

      and $O' = \prod_{i=1}^{k} n_i[W_i']^{\nu_i'}_{t_i'}$. Since $M \mid O$ is well-formed, by node-uniqueness it follows that $n_i \notin \mathsf{nds}(M)$ for all i, $1 \leq i \leq k$. Now, since

· $\mathsf{ngh}(m, M) \setminus \mathsf{nds}(M) = \emptyset$
· $n_i \notin \mathsf{nds}(M)$, for all i
· $M \mid O$ is connected (see clause 1 of Definition 6.4)

it follows that $m \notin \nu_i$, for all i, $1 \leq i \leq k$. This implies that the transitions

$$n_i[W_i]_{t_i}^{\nu_i} \xrightarrow{m?v} n_i[W_i']_{t_i'}^{\nu_i'}$$

can only be derived by applying rule (OutRng) with $W_i' = W_i$, $\nu_i' = \nu_i$, and $t_i' = t_i$. This implies $O' = O$. Now, since $N \Rightarrow N'$, by several applications of Lemma 6.22 it follows that $N \mid O \Rightarrow N' \mid O = N' \mid O'$. By Theorem 6.12, both $M' \mid O'$ and $N' \mid O'$ are well-formed. As $M' \approx N'$ it follows that $\left(M' \mid O' \,,\, N' \mid O'\right) \in \mathcal{S}$.

· Let $\nu' = \mathsf{ngh}(m, M) \setminus \mathsf{nds}(M) \neq \emptyset$. Then, by an application of rule (Obs) we have $M \xrightarrow{m!v \blacktriangleright \nu'} M'$ because $M \xrightarrow{m!v} M'$. Since $M \approx N$ there is N' such that $N \xRightarrow{m!v \blacktriangleright \nu'} N'$, with $M' \approx N'$ and $\nu' = \mathsf{ngh}(m, N) \setminus \mathsf{nds}(N)$. Since the action $m!v \blacktriangleright \nu'$ can be only generated by an application of rule (Obs), there are N_1 and N_2 such that

$$N \xrightarrow{\tau}{}^{*} N_1 \xrightarrow{m!v} N_2 \xrightarrow{\tau}{}^{*} N' \ .$$

Since $O \xrightarrow{m?v} O'$, by several applications of Lemma 6.22 and one application of rule (Sync) we have:

$$N \mid O \xrightarrow{\tau}{}^{*} N_1 \mid O \xrightarrow{m!v} N_2 \mid O' \xrightarrow{\tau}{}^{*} N' \mid O' \ .$$

As $M \mid O$ is well-formed and $m \in \mathsf{nds}(M)$, by node-uniqueness it follows that $m \notin \mathsf{nds}(O)$. Thus,

$$\begin{aligned} \mathsf{ngh}(m, N \mid O) \setminus \mathsf{nds}(N \mid O) &= (\mathsf{ngh}(m, N) \setminus \mathsf{nds}(N)) \setminus \mathsf{nds}(O) \\ &= \nu' \setminus \mathsf{nds}(O) \\ &= (\mathsf{ngh}(m, M) \setminus \mathsf{nds}(M)) \setminus \mathsf{nds}(O) \\ &= \mathsf{ngh}(m, M \mid O) \setminus \mathsf{nds}(M \mid O) \\ &= \emptyset \ . \end{aligned}$$

By an application of rule (Shh) we can derive

$$N \mid O \xrightarrow{\tau}{}^{*} N_1 \mid O \xrightarrow{\tau} N_2 \mid O' \xrightarrow{\tau}{}^{*} N' \mid O' \ .$$

By Theorem 6.12, both $M' \mid O'$ and $N' \mid O'$ are well-formed. As $M' \approx N'$ it follows that $\left(M' \mid O' \,,\, N' \mid O'\right) \in \mathcal{S}$.

– Let $M \mid O \xrightarrow{m!v} \widehat{M}$, by an application of rule (Sync) because $M \xrightarrow{m?v} M'$ and $O \xrightarrow{m!v} O'$, with $\widehat{M} = M' \mid O'$. This case is similar to a previous one.

- Let $M \mid O \xrightarrow{m?v} \widehat{M}$, by an application of rule (RcvPar), because $M \xrightarrow{m?v} M'$, $O \xrightarrow{m?v} O'$, and $\widehat{M} = M' \mid O'$. This case is easy.
- Let $M \mid O \xrightarrow{\mathsf{tick}} \widehat{M}$ by an application of rule (Par-tick) because $M \xrightarrow{\mathsf{tick}} M'$, $O \xrightarrow{\mathsf{tick}} O'$, and $\widehat{M} = M' \mid O'$. This case is easy.

□

Proof of Theorem 6.26

We show details only for Law 7; for the other ones we provide only the bisimulation relation.

1. It is easy to prove that $\mathcal{S}$

$$\mathcal{S} \stackrel{\text{def}}{=} \{n[\mathsf{nil}]^{\nu}_{t}, n[Sleep]^{\nu}_{t} \text{ for all } t\}$$

   where $Sleep \stackrel{\text{def}}{=} \mathsf{tick}.Sleep$ is a bisimulation.
2. It is easy to prove that $\mathcal{S}$

$$\mathcal{S} \stackrel{\text{def}}{=} \{n[\mathsf{nil}]^{\nu}_{t}, n[P]^{\nu}_{t} : \text{ for all } P, t, \text{ and } P \text{ does not contain sender processes}\} \cup Id$$

   where Id is the identity relation between network terms is a bisimulation.
3. It is easy to prove that $\mathcal{S}$

$$\mathcal{S} \stackrel{\text{def}}{=} \{n[\mathsf{tick}^{r}.P]^{\nu}_{s}, n[\mathsf{tick}^{r}.P]^{\nu}_{t} : \text{ for all } r, s, t, s \leq r \text{ and } t \leq r\} \cup Id$$

   where Id is the identity relation between network terms, is a bisimulation.
4. Let us define:
   - $A \stackrel{\text{def}}{=} m[\langle v\rangle^{r}.P]^{\nu}_{t} \mid n[(x)_{m:v}.Q]^{\nu'}_{r} \mid M$, where $m \in \nu'$ and for all $k \in \nu' \setminus m$ it holds that $M \equiv k[\mathsf{tick}^{s}.R]^{\nu_k}_{t_k} \mid M'$, with $s \geq r$.
   - $B \stackrel{\text{def}}{=} m[\langle v\rangle^{r}.P]^{\nu}_{t} \mid n[\mathsf{tick}^{r}.\{^{m:v}/_{x}\}Q]^{\nu'}_{r} \mid M$, where $m \in \nu'$ and for all $k \in \nu' \setminus m$ it holds that $M \equiv k[\mathsf{tick}^{s}.R]^{\nu_k}_{t_k} \mid M'$, with $s \geq r$.

   Now, let

$$\mathcal{S} \stackrel{\text{def}}{=} \{(A, B) : \text{ for all } P, r, t, \nu, \ldots\} \cup Id$$

   where Id is the identity relation between network terms. It is easy to prove that $\mathcal{S}$ is a bisimulation.
5. The proof of this law can be found in Section 6.8 at page 103.
6. Let us define:
   - $A_1 \stackrel{\text{def}}{=} m[\langle v_1\rangle^{r}.!\langle v_2\rangle.P]^{\nu}_{s} \mid n[(x)_{w}.Q]^{\nu'}_{t}$, with $t > r$,
   - $B_1 \stackrel{\text{def}}{=} m[\langle v_1\rangle^{r}.!\langle v_2\rangle.P]^{\nu}_{s} \mid n[(x)_{\perp}.Q]^{\nu'}_{t}$, with $t > r$,
   - $A_2 \stackrel{\text{def}}{=} m[!\langle v_2\rangle.P]^{\nu}_{s} \mid n[(x)_{w}.Q]^{\nu'}_{t}$,
   - $B_2 \stackrel{\text{def}}{=} m[!\langle v_2\rangle.P]^{\nu}_{s} \mid n[(x)_{\perp}.Q]^{\nu'}_{t}$

   where $m \in \nu'$. It is easy to prove that $\mathcal{S}$

$$\mathcal{S} \stackrel{\text{def}}{=} \{(A_1, B_1) : \text{ for all } r, P, s \ldots\} \cup \{(A_2, B_2) : \text{ for all } r, P, s \ldots\} \cup Id$$

   where Id is the identity relation between network terms, is a bisimulation.
7. Let us define:
   - $A_1 \stackrel{\text{def}}{=} m[!\langle v\rangle.P]^{\nu}_{t} \mid N$, where for all $n \in \nu$ it holds that $N \equiv n[W]^{\nu'}_{t'} \mid N'$, with $t' > 0$
   - $B_1 \stackrel{\text{def}}{=} m[!\langle w\rangle.P]^{\nu}_{t} \mid N$, where for all $n \in \nu$ it holds that $N \equiv n[W]^{\nu'}_{t'} \mid N'$, with $t' > 0$
   - $A_2 \stackrel{\text{def}}{=} m[\langle v\rangle^{r}.P]^{\nu}_{t} \mid N$, with $r \leq \delta_{v}$, where for all $n \in \nu$ it holds that $N \equiv n[W]^{\nu'}_{t'} \mid N'$, with $t' \geq r$

- $B_2 \stackrel{\text{def}}{=} m[\langle w\rangle^r.P]^\nu_t \mid N$, with $r \leq \delta_w$, where for all $n \in \nu$ it holds that $N \equiv n[W]^{\nu'}_{t'} \mid N'$, with $t' \geq r$

where $\delta_v = \delta_w$. Now, let

$$\mathcal{S} \stackrel{\text{def}}{=} \{(A_1, B_1) : \text{ for all } P, t, \nu, \ldots\} \cup \{(A_2, B_2) : \text{ for all } P, t, \nu, \ldots\} \cup Id$$

where Id is the identity relation between network terms. We prove that $\mathcal{S}$ is a bisimulation. We proceed by case analysis on the possible transitions.

- Let us examine the most interesting transitions of A_1. The reasoning for the other transitions of A_1 is simpler. Notice that by maximal progress the term A_1 cannot perform tick-actions.
  - Let $A_1 \xrightarrow{\tau} A_2 = m[\langle v\rangle^{\delta_v}.P]^\nu_t \mid \hat{N}$, because $A_1 \xrightarrow{m!v} A_2$ by an application of rule (Shh). This is possible only by an application of rule (Sync) with
    - $m[!\langle v\rangle.P]^\nu_t \xrightarrow{m!v} m[\langle v\rangle^{\delta_v}.P]^\nu_t$
    - $N \xrightarrow{m?v} \hat{N}$, where if $n \in \nu$ then $\hat{N} \equiv n[\hat{W}]^{\nu'}_{\hat{t}} \mid \hat{N}'$, with $\hat{t} \geq \delta_v$ (by definition of rules (Coll) and (Exp)).

    Notice that since all nodes in $\nu \cap \mathsf{nds}(N)$ are exposed, it follows that if $\hat{W}$ is an active receiver then it will be of the form $(x)_\perp.P$, for some P. Now, $A_2 \xrightarrow{\tau} B_2 = m[\langle w\rangle^{\delta_w}.P]^\nu_t \mid \hat{\hat{N}}$, because $A_2 \xrightarrow{m!v} B_2$ by an application of rule (Shh). This is possible only by an application of rule (Sync) with
    - $m[!\langle w\rangle.P]^\nu_t \xrightarrow{m!w} m[\langle w\rangle^{\delta_w}.P]^\nu_t$
    - $N \xrightarrow{m?w} \hat{\hat{N}}$, where if $n \in \nu$ then $\hat{\hat{N}} \equiv n[\hat{\hat{W}}]^{\nu'}_{\hat{\hat{t}}} \mid \hat{\hat{N}}'$, with $\hat{\hat{t}} \geq \delta_v$ (by definition of rules (Coll) and (Exp)).

    Again, since all nodes in $\nu \cap \mathsf{nds}(N)$ are exposed, it follows that if $\hat{\hat{W}}$ is an active receiver then it will be of the form $(x)_\perp.P$, for some P. Moreover, since $\delta_v = \delta_w$ it follows $\hat{t} = \hat{\hat{t}}$. As a consequence, $\hat{N} = \hat{\hat{N}}$ and $(A_2, B_2) \in \mathcal{S}$.
- Let us examine the most interesting transitions of A_2. The reasoning for the other transitions is simpler.
  - Let $A_2 \xrightarrow{\mathsf{tick}} A'_2 = m[\langle v\rangle^{r-1}.P]^\nu_{t\ominus 1} \mid \hat{N}$ by an application of rule (Par-tick) because
    - $m[\langle v\rangle^r.P]^\nu_t \xrightarrow{\mathsf{tick}} m[\langle v\rangle^{r-1}.P]^\nu_{t\ominus 1}$
    - $N \xrightarrow{\mathsf{tick}} \hat{N}$.

    In this case we have $B_2 \xrightarrow{\mathsf{tick}} B'_2 = m[\langle w\rangle^{r-1}.P]^\nu_{t\ominus 1} \mid \hat{N}$. Now, independently whether $r > 1$ or not we have $(A'_2, B'_2) \in \mathcal{S}$.

□

A.2 Proofs of Chapter 7

Proof of Proposition 7.6

1. The proof of this case can be found at page 118.
2. We separately prove that ($\Rightarrow$) if $M \xrightarrow{\tau}_{\rho} M'$ then $M \xrightarrow{\tau}_{\rho,\emptyset} M'$ and ($\Leftarrow$) if $M \xrightarrow{\tau}_{\rho,\emptyset} M'$ then $M \xrightarrow{\tau}_{\rho} M'$. It is proved by induction on $M \xrightarrow{\lambda}_{\rho} M'$, in the first case, and on $M \xrightarrow{\lambda}_{\rho,C} M'$, in the second case
   ($\Rightarrow$) The base case is when $M \xrightarrow{\tau}_{\rho} M'$ is given by rule (Lose). By rule (LoseR) it holds that $M \xrightarrow{\tau}_{\rho,\emptyset} M'$. As to the inductive case, we consider $M \xrightarrow{\tau}_{\rho} M'$ given by rule (Par). Then $M = M_1 \mid M_2$, for some M_1 and M_2, with $M_1 \mid M_2 \xrightarrow{\tau}_{\rho} M_1' \mid M_2$, because $M_1 \xrightarrow{\tau}_{\rho} M_1'$ and $M' = M_1' \mid M_2$, for some M_1' (the converse is similar). By inductive hypothesis, it holds that $M_1 \xrightarrow{\tau}_{\rho,\emptyset} M_1'$. Thus by rule (ParR) we have $M_1 \mid M_2 \xrightarrow{\tau}_{\rho,\emptyset} M_1' \mid M_2$, as required.
   ($\Leftarrow$) The base case is when $M \xrightarrow{\tau}_{\rho,\emptyset} M'$ is given by rule (LoseR). By rule (Lose) it holds that $M \xrightarrow{\tau}_{\rho} M'$, as required. As to the inductive case, we consider $M \xrightarrow{\tau}_{\rho,\emptyset} M'$ given by rule (ParR). Then $M = M_1 \mid M_2$, for some M_1 and M_2, with $M_1 \mid M_2 \xrightarrow{\tau}_{\rho,\emptyset} M_1' \mid M_2$, because $M_1 \xrightarrow{\tau}_{\rho,\emptyset} M_1'$ and $M' = M_1' \mid M_2$, for some M_1', (the converse is similar). By inductive hypothesis, it holds that $M_1 \xrightarrow{\tau}_{\rho} M_1'$. Thus by rule (Par) we have $M_1 \mid M_2 \xrightarrow{\tau}_{\rho} M_1' \mid M_2$, as required.

□

In order to prove Theorem 7.8 of Safety preservation, we need the following auxiliary lemma.

Lemma A.5. *Let* $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ *with* $M \equiv \prod_i n_i[P_i]_{T_i}$ *and* $M' \equiv \prod_i n_i[P_i']_{T_i'}$.

1. *If* $P_i' \neq P_i$, *for some* i, *then* $n_i \in \mathcal{D}$ *and* $T_i(n_i, m) \geq \rho$.
2. *If* $T_i' \neq T_i$, *for some* i, *then* $n_i \in \mathcal{D}$ *and* $T_i(n_i, m) \geq \rho$.

Proof

1. This case applies only for transitions at level σ. It is proved by induction on $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$. The base case is when the transition $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ is given by rule (Rcv); thus $M = n[\sigma?(\tilde{x}).P]_T$, $M' = n[\{\tilde{v}/\tilde{x}\}P]_T, \mathcal{D} = n$ and $T(n, m) \geq \sigma$, as required. As to the inductive case, let us consider the transition $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ given by rules (Sum), (RcvPar) or (Par). We show details only for (RcvPar). Let be $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ given by rule (RcvPar) with $M = M_1 \mid M_2$ and $M' = M_1' \mid M_2'$, for some M_1, M_2, M_1' and M_2', because $M_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\sigma} M_1'$ and $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\sigma} M_2'$, where $\mathcal{D} := \mathcal{D}' \cup \mathcal{D}''$. More precisely, let

$$M \equiv \prod_i n_i[P_i]_{T_i} = \prod_k n_k[P_k]_{T_k} \mid \prod_j n_j[P_j]_{T_j}$$

and

$$M' \equiv \prod_i n_i[P'_i]_{T'_i} = \prod_k n_k[P'_k]_{T_k} \mid \prod_j n_j[P'_j]_{T_j}$$

for appropriate processes and tags, where

$$M_1 = \prod_k n_k[P_k]_{T_k}, M'_1 = \prod_k n_k[P'_k]_{T_k}, M_2 = \prod_j n_j[P_j]_{T_j}, M'_2 = \prod_j n_j[P'_j]_{T_j}.$$

If $P'_k \neq P_k$, for some k or if $P'_j \neq P_j$, for some j, the result follows by inductive hypothesis.

2. This case applies only for transitions at level `trust`. It is proved by induction on $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\texttt{trust}} M'$. The proof is similar to the previous case.

□

All τ-actions propagate through parallel composition; it is easy to prove the following lemma.

Lemma A.6. *If $M \xrightarrow{\tau}_\rho M'$ then $M \mid N \xrightarrow{\tau}_\rho M' \mid N$ and $N \mid M \xrightarrow{\tau}_\rho N \mid M'$.*

In order to prove Theorem 7.16 on the contextuality of δ-bisimilarity, we need the following auxiliary lemmas.

Lemma A.7. *If $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M'$ then there are $\widehat{N}, P$ and T such that $M \equiv \widehat{N} \mid m[P]_T$ and $\mathcal{D} := \{n : T(m,n) \geq \rho\}$.*

Proof The proof proceeds by induction on the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M'$. The case base is when $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M'$ is given by rules (MCast), (UCast), (DTrust), or (SndRcm). We show details only for rule (MCast), the other ones are similar. Then $M = m[\sigma!\langle\tilde{v}\rangle.P]_T$, for some P and T and $\mathcal{D} := \{n : T(m,n) \geq \rho\}$. As to the inductive case, we consider when $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M'$ is given by rules (Sum), (Sync) or (Par). We show details only for rule (Sync). Let $M = M_1 \mid M_2$ and $M' = M'_1 \mid M'_2$, for some M_1, M_2, M'_1 and M'_2, and $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M'$ by rule (Sync) because $M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho M'_1$ and $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_\rho M'_2$ (the converse is similar), with $\mathcal{D}' \subseteq \mathcal{D}$. The result follows by inductive hypothesis, as it holds that $M_1 \equiv \widehat{N} \mid m[P]_T$, for some $\widehat{N}, P, T$ and $\mathcal{D} := \{n : T(m,n) \geq \rho\}$. □

Lemma A.8. *Let M, N be two networks such that $M \approx_\delta N$ and O, O' be two networks such that $O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho O'$ and $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho \widehat{M}$, for some $\widehat{M}$ and $N \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\rho \widehat{N}$, for some $\widehat{N}$. Then $\mathcal{D} \setminus \mathsf{nds}(M) = \mathcal{D} \setminus \mathsf{nds}(N)$.*

Proof The proof proceeds by contradiction. Let us suppose that $\mathcal{D} \setminus \mathsf{nds}(M) \neq \mathcal{D} \setminus \mathsf{nds}(N)$. We can have the following cases:

- $\mathcal{D} \setminus \mathsf{nds}(M) = \emptyset$. This implies that $\mathcal{D} \cap \mathsf{nds}(M) = \mathcal{D}$. The set $\mathcal{D} \cap \mathsf{nds}(M)$ indicates the potential receivers in M. Thus, in this case $\mathcal{D}$ is this set. Then it may be that $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_\rho M'$, for some M'. Now, as $\mathcal{D} \setminus \mathsf{nds}(N) \neq \emptyset$, this implies that there exists a $\widehat{\mathcal{D}} \neq \mathcal{D}$ such that $\mathcal{D} \cap \mathsf{nds}(N) = \widehat{\mathcal{D}}$. Then it may be that $N \xrightarrow{m?\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho N'$, for some N', in contradiction with the hypothesis $M \approx_\delta N$;

- $\mathcal{D} \setminus \mathsf{nds}(M) \neq \emptyset$. This case is similar to the previous one.

□

Proof of Theorem 7.16

We prove that the relation

$$\mathcal{S} \stackrel{\text{def}}{=} \{(M \mid O\,,\, N \mid O) \text{ for all } O \text{ such that } M \approx_\delta N\}$$

is a δ-bisimulation. We proceed by case analysis on the transition $M \mid O \xrightarrow{\alpha}_\rho \widehat{M}$, with $\rho \leq \delta$.

- Let $M \mid O \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\rho \widehat{M}$ by an application of rule (Obs), with $\mathcal{D} \neq \emptyset$, because $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$, with $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O) \neq \emptyset$. We have the following possibilities:
  - Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho M'$ and $O \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_\rho O'$, with $\widehat{M} = M' \mid O'$ and $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$. We showed this case at page 123.
  - Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_\rho M'$ and $O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho O'$, with $\widehat{M} = M' \mid O'$ and $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$. As $M \approx_\delta N$ then there is N' such that $N \stackrel{m?\tilde{v}\triangleright\mathcal{D}''}{\Longrightarrow}_\rho N'$ with $M' \approx_\delta N'$. This implies that there are N_1 and N_2 such that

    $$N \stackrel{\tau}{\Longrightarrow}_\rho N_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_\rho N_2 \stackrel{\tau}{\Longrightarrow}_\rho N'.$$

    Then by several applications of Lemma A.6 and one application of rule (Sync), as $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$, we have

    $$N \mid O \stackrel{\tau}{\Longrightarrow}_\rho N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho N_2 \mid O' \stackrel{\tau}{\Longrightarrow}_\rho N' \mid O'.$$

    By Lemma A.8 it holds that $\widehat{\mathcal{D}} \setminus \mathsf{nds}(M) = \widehat{\mathcal{D}} \setminus \mathsf{nds}(N)$, then

    $$\begin{aligned} \mathcal{D} &:= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(N \mid O) \neq \emptyset. \end{aligned}$$

    Then by one application of rule (Obs) we have $N \mid O \stackrel{m!\tilde{v}\blacktriangleright\mathcal{D}}{\Longrightarrow}_\rho N' \mid O'$ and $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.
  - Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$ by an application of rule (Par) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho M'$, with $\widehat{M} = M' \mid O$ and $m \notin \mathsf{nds}(O)$. The proof is similar to the first case.
  - Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$ by an application of rule (Par) because $O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho O'$, with $\widehat{M} = M \mid O'$ and $m \notin \mathsf{nds}(M)$. By Lemma A.7, $m \in \mathsf{nds}(O)$; as we assume that networks are node unique then $m \notin \mathsf{nds}(N)$. By one application

of rule (Par) we have $N \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}} N \mid O'$. By Lemma A.8 it holds that $\widehat{\mathcal{D}} \setminus \mathsf{nds}(M) = \widehat{\mathcal{D}} \setminus \mathsf{nds}(N)$, then

$$\begin{array}{rcl} \mathcal{D} & := & \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O) \\ & = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) \\ & = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O) \\ & = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(N \mid O) \neq \emptyset. \end{array}$$

Thus by one application of rule (Obs) we have $N \mid O \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} N \mid O'$ and $(M \mid O', N \mid O') \in \mathcal{S}$, as required.

- Let $M \mid O \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\rho} \widehat{M}$. This case is easy to prove.
- $M \mid O \xrightarrow{\tau}_{\rho} \widehat{M}$ by an application of rule (Shh), because $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$, with $\mathcal{D} \subseteq \mathsf{nds}(M \mid O)$ and $\rho' \neq \mathsf{bad} \leq \delta$. We have the following possibilities:
  - Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} M'$ and $O \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho'} O'$, with $\widehat{M} = M' \mid O', \mathcal{D}'' \subseteq \mathcal{D}$ and $\mathcal{D} \setminus \mathsf{nds}(M \mid O) = \emptyset$. Let $\mathcal{D}' := \mathcal{D} \setminus \mathsf{nds}(M)$; as $\mathcal{D} \setminus \mathsf{nds}(M \mid O) = \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) = \emptyset$ but it is easy to prove that $\mathcal{D}'' \subseteq \mathsf{nds}(O) \neq \emptyset$ then $\mathcal{D}' \neq \emptyset$. Then we can apply (Obs) and obtain $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\rho'} M'$. As $M \approx_{\delta} N$ then there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\rho'} N'$ with $M' \approx_{\delta} N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

$$N \xRightarrow{\tau}_{\rho'} N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho'} N_2 \xRightarrow{\tau}_{\rho'} N'$$

with $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \neq \emptyset$. By Lemma A.7 we have $M \equiv \widehat{M} \mid m[P]_T$ and $N_1 \equiv \widehat{N} \mid m[Q]_T$, for some $\widehat{M}, \widehat{N}, T, Q$ and P and $\mathcal{D} := \{n : T(m,n) \geq \rho\}$ and $\widehat{\mathcal{D}} := \{n : T(m,n) \geq \rho\}$. Then $\widehat{\mathcal{D}} = \mathcal{D}$. Then by several applications of Lemma A.6 and one application of rule (Sync), as $\mathcal{D}'' \subseteq \mathcal{D}$, we have

$$N \mid O \xRightarrow{\tau}_{\rho'} N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} N_2 \mid O' \xRightarrow{\tau}_{\rho'} N' \mid O'.$$

It holds that

$$\begin{array}{rcl} & & \mathcal{D} \setminus \mathsf{nds}(M \mid O) \\ & = & \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) \\ & = & \mathcal{D} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O) = \emptyset. \end{array}$$

Then by one application of rule (Shh) we have $N \mid O \xRightarrow{\tau}_{\rho} N' \mid O'$ and $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.

  - Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho'} M'$ and $O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} O'$, with $\widehat{M} = M' \mid O'$ and $\mathcal{D}'' \subseteq \mathcal{D}$. As $M \approx_{\delta} N$ then there is N' such that $N \xRightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho'} N'$ with $M' \approx_{\delta} N'$. This implies that there are N_1 and N_2 such that

$$N \xRightarrow{\tau}_{\rho'} N_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho'} N_2 \xRightarrow{\tau}_{\rho'} N'.$$

Then by several applications of Lemma A.6 and one application of rule (Sync), as $\mathcal{D}'' \subseteq \mathcal{D}$, we have

$$N \mid O \stackrel{\tau}{\Longrightarrow}_{\rho'} N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} N_2 \mid O' \stackrel{\tau}{\Longrightarrow}_{\rho'} N' \mid O'.$$

As by Lemma A.8 $\mathcal{D} \setminus \mathsf{nds}(M) = \mathcal{D} \setminus \mathsf{nds}(N)$ it holds that

$$\begin{array}{ll} & \mathcal{D} \setminus \mathsf{nds}(M \mid O) \\ = & \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) \\ = & \mathcal{D} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O) = \emptyset. \end{array}$$

Then by one application of rule (Shh) we have $N \mid O \stackrel{\tau}{\Longrightarrow}_{\rho} N' \mid O'$ and $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.

– Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} \widehat{M}$ by an application of rule (Par) because $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} M'$ with $\widehat{M} = M' \mid O$. There are two possibilities:
  · Let $\mathcal{D} \subseteq \mathsf{nds}(M)$. Then $M \xrightarrow{\tau}_{\rho} M'$ by an application of (Shh). As $M \approx_\delta N$, there is N' such that $N \stackrel{\tau}{\Longrightarrow}_{\rho} N'$, with $M' \approx_\delta N'$. By several applications of Lemma A.6 we have $N \mid O \stackrel{\tau}{\Longrightarrow}_{\rho} N' \mid O$. Then $(M' \mid O, N' \mid O) \in \mathcal{S}$, as required.
  · Let $\mathcal{D} \not\subseteq \mathsf{nds}(M)$. Then by an application of rule (Obs) we have $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\rho'} M'$, with $\mathcal{D}' := \mathcal{D} \setminus \mathsf{nds}(M) \neq \emptyset$. As $M \approx_\delta N$, there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\rho'} N'$, with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

$$N \stackrel{\tau}{\Longrightarrow}_{\rho'} N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho'} N_2 \stackrel{\tau}{\Longrightarrow}_{\rho'} N'$$

with $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \neq \emptyset$. By several applications of Lemma A.6 and by one application of rule (Par) we have:

$$N \mid O \stackrel{\tau}{\Longrightarrow}_{\rho'} N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho'} N_2 \mid O \stackrel{\tau}{\Longrightarrow}_{\rho'} N' \mid O.$$

It holds that

$$\begin{array}{ll} & \mathcal{D} \setminus \mathsf{nds}(M \mid O) \\ = & \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) \\ = & \mathcal{D}' \setminus \mathsf{nds}(O) \\ = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O) = \emptyset. \end{array}$$

Then by one application of rule (Shh) we have $N \mid O \stackrel{\tau}{\Longrightarrow}_{\rho} N' \mid O$ and $(M' \mid O, N' \mid O) \in \mathcal{S}$, as required.

– Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$ by an application of rule (Par) because $O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} O'$ with $m \notin \mathsf{nds}(M)$ and $\widehat{M} = M \mid O'$. There are two possibilities:
  · Let $\mathcal{D} \subseteq \mathsf{nds}(O)$. Then $O \xrightarrow{\tau}_{\rho} O'$ by an application of (Shh). By one application of rule (Par) we have $N \mid O \stackrel{\tau}{\Longrightarrow}_{\rho} N \mid O'$. Then $(M \mid O', N \mid O') \in \mathcal{S}$, as required.

- Let $\mathcal{D} \not\subseteq \mathsf{nds}(O)$. By Lemma A.7 $m \in \mathsf{nds}(O)$; as we assume that networks are node unique then $m \notin \mathsf{nds}(N)$. By one application of rule (Par) we have $N \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} N \mid O'$. As by Lemma A.8 $\mathcal{D} \setminus \mathsf{nds}(M) = \mathcal{D} \setminus \mathsf{nds}(N)$ it holds that

$$\begin{aligned} & \mathcal{D} \setminus \mathsf{nds}(M \mid O) \\ = \ & \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) \\ = \ & \mathcal{D} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O) = \emptyset. \end{aligned}$$

  Thus by one application of rule (Shh) we have $N \mid O \stackrel{\tau}{\Longrightarrow}_{\rho} N \mid O'$ and $(M \mid O', N \mid O') \in \mathcal{S}$, as required.

- Let $M \mid O \xrightarrow{\tau}_{\rho} \widehat{M}$ by an application of rule (Par). This case is easy to prove.

□

In order to prove Theorem 7.17 of Soundness, we need to prove the following auxiliary lemmas.

Lemma A.9.

1. *If* $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$, *where* $\mathcal{D}$ *contains more than one node, then there are* N, P, T *such that* $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$.
2. *If* $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$, *with* $\mathcal{D} = n$, *for some* n, *then there are* N, P, T *such that* $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$ *or* $M \equiv m[\sigma!\langle\tilde{v}\rangle_n.P]_T \mid N$.

Proof

1. By induction on the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$. The base case is when $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ is given by rule (MCast); thus $M = m[\sigma!\langle\tilde{v}\rangle.P]_T$, for some P and T. As to the inductive case, let us consider $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ given by rules (Sum), (Sync) or (Par). We only examine the case of rule (Sync). Let $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ given by rule (Sync) because $M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M_1'$ and $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\sigma} M_2'$, and $M = M_1 \mid M_2$ and $M' = M_1' \mid M_2'$ for some M_1, M_1', M_2, M_2' and $\mathcal{D}'$. By inductive hypothesis it holds that $M_1 \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$, for some N, P, T. Thus $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N \mid M_2$, as required.
2. By induction on the transition $M \xrightarrow{m!\tilde{v}\triangleright n}_{\sigma} M'$. This case is similar to the previous one.

□

Lemma A.10.

1. *If* $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\sigma} M'$ *then* $M \downarrow_n^{\sigma}$, *for all* $n \in \mathcal{D}$.
2. *If* $M \downarrow_n^{\sigma}$ *then there is a value* $\tilde{v}$ *and a set of nodes* $\mathcal{D}$, *with* $n \in \mathcal{D}$, *such that* $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\sigma} M'$,

Proof By Lemma A.9 and by Definition 7.10. □

Proof of Theorem 7.20

- Let $M \mid H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} \widehat{M}$ by an application of rule (Obs), with $\rho \neq \texttt{trust}$, because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho} \widehat{M}$. We show this case at page 126.
- Let $M \mid H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} \widehat{M}$ by an application of rule (Obs), with $\rho = \texttt{trust}$, because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\texttt{trust}} \widehat{M}$, with $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$. We have the following possibilities:
  - Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\texttt{trust}} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\texttt{trust}} M'$ and $H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\texttt{trust}} H'$, with $\widehat{M} = M' \mid H', \mathcal{D}'' \subseteq \widehat{\mathcal{D}}$ and $H' \in \mathcal{H}_\delta$. We show this case at page 126.
  - Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\texttt{trust}} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\texttt{trust}} M'$ and $H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\texttt{trust}} H'$, with $\widehat{M} = M' \mid H', \mathcal{D}'' \subseteq \widehat{\mathcal{D}}$ and $H' \in \mathcal{H}_\delta$. As $M \approx_\delta N$ then there is N' such that $N \xRightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\texttt{trust}} N'$ with $M' \approx_\delta N'$. This implies that there are N_1 and N_2 such that

$$N \xRightarrow{\tau}_{\texttt{trust}} N_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\texttt{trust}} N_2 \xRightarrow{\tau}_{\texttt{trust}} N'.$$

    Let $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(H)$; as $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$ then also $\mathcal{D}' \neq \emptyset$. By rule (Obs) we have $H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\texttt{trust}} H'$, with $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(H) \neq \emptyset$. As $H \approx_{\texttt{trust}} K$ then $K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\texttt{trust}} K'$, with $H' \approx_{\texttt{trust}} K'$ and $K' \in \mathcal{H}_\delta$. This implies that there are K_1 and K_2 such that

$$K \xRightarrow{\tau}_{\texttt{trust}} K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\texttt{trust}} K_2 \xRightarrow{\tau}_{\texttt{trust}} K'.$$

    By Lemma A.7 we have $H \equiv \widehat{H} \mid m[P]_T$ and $K_1 \equiv \widehat{K} \mid m[Q]_T$, for some $\widehat{H}, \widehat{K}, T, Q$ and P and $\widehat{\mathcal{D}} := \{n : T(m,n) \geq \texttt{trust}\}$ and $\widehat{\mathcal{D}'} := \{n : T(m,n) \geq \texttt{trust}\}$. Then $\widehat{\mathcal{D}'} = \widehat{\mathcal{D}}$. By several applications of Lemma A.6 and one application of rule (Sync), as $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$, we have

$$N \mid K \xRightarrow{\tau}_{\texttt{trust}} N_1 \mid K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\texttt{trust}} N_2 \mid K_2 \xRightarrow{\tau}_{\texttt{trust}} N' \mid K'.$$

    As $\mathsf{nds}(K) = \mathsf{nds}(H)$ and as by Lemma A.8 $\widehat{\mathcal{D}} \setminus \mathsf{nds}(M) = \widehat{\mathcal{D}} \setminus \mathsf{nds}(N)$ it holds that

$$\begin{array}{rcl} \mathcal{D} & := & \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \\ & = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ & = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) \neq \emptyset. \end{array}$$

    Then by one application of rule (Obs) we have $N \mid K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} N' \mid K'$ and $(M' \mid H', N' \mid K') \in \mathcal{S}$, as required.
  - Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\texttt{trust}} \widehat{M}$ by an application of rule (Par) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\texttt{trust}} M'$ with $\widehat{M} = M' \mid H$. The proof is similar to the case of the transition $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho} M' \mid H$, with $\rho \neq \texttt{trust}$.

– Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathtt{trust}} \widehat{M}$ by an application of rule (Par) because $H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathtt{trust}} H'$ with $\widehat{M} = M \mid H', H' \in \mathcal{H}_\delta$ and $m \notin \mathsf{nds}(M)$. Let $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(H)$; as $\mathcal{D} := \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$ then also $\mathcal{D}' \neq \emptyset$. By rule (Obs) we have $H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathtt{trust}} H'$. As $H \approx_{\mathtt{trust}} K$ then $K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathtt{trust}} K'$, with $H' \approx_{\mathtt{trust}} K'$ and $K' \in \mathcal{H}_\delta$. This implies that there are K_1 and K_2 such that

$$K \xRightarrow{\tau}_{\mathtt{trust}} K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\mathtt{trust}} K_2 \xRightarrow{\tau}_{\mathtt{trust}} K'.$$

By Lemma A.7 we have $H \equiv \widehat{H} \mid m[P]_T$ and $K_1 \equiv \widehat{K} \mid m[Q]_T$, for some $\widehat{H}, \widehat{K}, T, Q$ and P and $\widehat{\mathcal{D}} := \{n : T(m,n) \geq \mathtt{trust}\}$ and $\widehat{\mathcal{D}'} := \{n : T(m,n) \geq \mathtt{trust}\}$. Then $\widehat{\mathcal{D}'} = \widehat{\mathcal{D}}$. As $m \in \mathsf{nds}(K)$ and as we assume that networks are node unique, then $m \notin \mathsf{nds}(N)$. By several applications of Lemma A.6 and by one application of rule (Par) we have:

$$N \mid K \xRightarrow{\tau}_\rho N \mid K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}} N \mid K_2 \xRightarrow{\tau}_\rho N \mid K'.$$

As $\mathsf{nds}(H) = \mathsf{nds}(K)$ and by Lemma A.8 $\widehat{\mathcal{D}} \setminus \mathsf{nds}(M) = \widehat{\mathcal{D}} \setminus \mathsf{nds}(N)$ it holds that

$$\begin{aligned} \mathcal{D} &:= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) \neq \emptyset. \end{aligned}$$

Thus by one application of rule (Obs) $N \mid K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\rho N \mid K'$ and $(M \mid H', N \mid K') \in \mathcal{S}$, as required.

- Let $M \mid H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_\rho \widehat{M}$ with $\rho \neq \mathtt{trust}$. The only possibility is that $M \mid H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_\rho M' \mid H$ by rule (Par) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_\rho M'$ with $\widehat{M} = M' \mid H$ and $m \notin \mathsf{nds}(H)$. This case is easy to prove.
- Let $M \mid H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_\rho \widehat{M}$ with $\rho = \mathtt{trust}$. We have the following possibilities:
  – Let $M \mid H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} \widehat{M}$ by rule (RcvPar) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\mathtt{trust}} M'$ and $H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathtt{trust}} H'$, with $\mathcal{D} := \mathcal{D}' \cup \mathcal{D}''$, $\widehat{M} = M' \mid H'$ and $H' \in \mathcal{H}_\delta$. This case is easy to prove.
  – Let $M \mid H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} \widehat{M}$ by rule (Par) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} M'$ with $\widehat{M} = M' \mid H$. The proof is similar to the case of the transition $M \mid H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_\rho M' \mid H$, with $\rho \neq \mathtt{trust}$.
  – Let $M \mid H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} \widehat{M}$ by rule (Par) because $H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} H'$ with $\widehat{M} = M \mid H', m \notin \mathsf{nds}(M)$ and $H' \in \mathcal{H}_\delta$. This case is easy to prove.
- $M \mid H \xrightarrow{\tau}_\rho \widehat{M}$ by an application of rule (Shh), because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$, with $\mathcal{D} \subseteq \mathsf{nds}(M \mid H)$ (and then $\mathcal{D} \setminus \mathsf{nds}(M \mid H) = \emptyset$). Let us consider $\rho' \neq \mathtt{trust}$. The only possibility is that $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$ by an application of rule (Par) because $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} M$ with $\widehat{M} = M' \mid H$ and $m \notin \mathsf{nds}(H)$. There are two possibilities:

- Let $\mathcal{D} \subseteq \mathsf{nds}(M)$. Then $M \xrightarrow{\tau}_\rho M'$ by an application of (Shh). As $M \approx_\delta N$, there is N' such that $N \stackrel{\tau}{\Longrightarrow}_\rho N'$, with $M' \approx_\delta N'$. By several applications of Lemma A.6 we have $N \mid K \stackrel{\tau}{\Longrightarrow}_\rho N' \mid K$. Then $(M' \mid H, N' \mid K) \in \mathcal{S}$, as required.
- Let $\mathcal{D} \not\subseteq \mathsf{nds}(M)$. Then by an application of rule (Obs) we have $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\rho'} M'$, with $\mathcal{D}' := \mathcal{D} \setminus \mathsf{nds}(M) \neq \emptyset$. As $M \approx_\delta N$, there is N' such that $N \stackrel{m!\tilde{v}\blacktriangleright\mathcal{D}'}{\Longrightarrow}_{\rho'} N'$, with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

$$N \stackrel{\tau}{\Longrightarrow}_{\rho'} N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho'} N_2 \stackrel{\tau}{\Longrightarrow}_{\rho'} N'$$

  with $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \neq \emptyset$. By several applications of Lemma A.6 and by one application of rule (Par), as $m \notin \mathsf{nds}(K)$, we have:

$$N \mid K \stackrel{\tau}{\Longrightarrow}_{\rho'} N_1 \mid K \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho'} N_2 \mid K \stackrel{\tau}{\Longrightarrow}_{\rho'} N' \mid K.$$

  As $\mathsf{nds}(H) = \mathsf{nds}(K)$ it holds that

$$\begin{aligned} & \mathcal{D} \setminus \mathsf{nds}(M \mid H) \\ = \ & \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ = \ & \mathcal{D}' \setminus \mathsf{nds}(H) \\ = \ & \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) = \emptyset. \end{aligned}$$

  Thus by one application of rule (Shh) we have $N \mid K \stackrel{\tau}{\Longrightarrow}_\rho N' \mid K$ and $(M' \mid H, N' \mid K) \in \mathcal{S}$, as required.

- $M \mid H \xrightarrow{\tau}_\rho \widehat{M}$ by an application of rule (Shh), because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$, with $\mathcal{D} \subseteq \mathsf{nds}(M \mid H)$ (and then $\mathcal{D} \setminus \mathsf{nds}(M \mid H) = \emptyset$). Let us consider $\rho' = \mathtt{trust}$. We have the following possibilities:
  - Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} \widehat{M}$ by rule (Par) because $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} M'$ with $\widehat{M} = M' \mid H$ and $m \notin \mathsf{nds}(H)$. This case is similar to the previous one.
  - Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} \widehat{M}$ by rule (Par) because $H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} H'$ with $\widehat{M} = M \mid H'$ and $H' \in \mathcal{H}_\delta$ and $m \notin \mathsf{nds}(M)$. This case is similar to the previous one. There are two possibilities:
    - Let $\mathcal{D} \subseteq \mathsf{nds}(H)$. Then $H \xrightarrow{\tau}_{\mathtt{trust}} H'$ by an application of (Shh). As $H \approx_{\mathtt{trust}} K$, there is K' such that $K \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} K'$, with $H' \approx_{\mathtt{trust}} K'$ and $K' \in \mathcal{H}_\delta$. By several applications of Lemma A.6 we have $N \mid K \stackrel{\tau}{\Longrightarrow}_\rho N' \mid K$. Then $(M \mid H', N \mid K') \in \mathcal{S}$, as required.
    - Let $\mathcal{D} \not\subseteq \mathsf{nds}(H)$. Then by an application of rule (Obs) we have $H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathtt{trust}} H'$, with $\mathcal{D}' := \mathcal{D} \setminus \mathsf{nds}(H) \neq \emptyset$. As $H \approx_\delta K$, there is K' such that $K \stackrel{m!\tilde{v}\blacktriangleright\mathcal{D}'}{\Longrightarrow}_{\mathtt{trust}} K'$, with $H' \approx_\delta K'$ and $K' \in \mathcal{H}_\delta$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are K_1 and K_2 such that

$$K \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathtt{trust}} K_2 \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} K'$$

with $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(K) \neq \emptyset$. By Lemma A.7 we have $H \equiv \widehat{H} \mid m[P]_T$ and $K_1 \equiv \widehat{K} \mid m[Q]_T$, for some $\widehat{H}, \widehat{K}, T, Q$ and P and $\widehat{\mathcal{D}} := \{n : T(m,n) \geq \mathtt{trust}\}$ and $\widehat{\mathcal{D}'} := \{n : T(m,n) \geq \mathtt{trust}\}$. Then $\widehat{\mathcal{D}'} = \widehat{\mathcal{D}}$. As we assume that networks are node unique then $m \notin \mathsf{nds}(N)$. By several applications of Lemma A.6 and by one application of rule (Par) we have:

$$N \mid K \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} N \mid K_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} N \mid K_2 \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} N \mid K_2.$$

As $\mathsf{nds}(H) = \mathsf{nds}(K)$ and as by Lemma A.8 $\mathcal{D} \setminus \mathsf{nds}(M) = \mathcal{D} \setminus \mathsf{nds}(N)$ it holds that

$$\begin{array}{ll} & \mathcal{D} \setminus \mathsf{nds}(M \mid H) \\ = & \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ = & \mathcal{D} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) = \emptyset. \end{array}$$

Thus by one application of rule (Shh) we have $N \mid K \stackrel{\tau}{\Longrightarrow}_{\rho} N \mid K'$ and $(M \mid H', N \mid K') \in \mathcal{S}$, as required.

– Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} \widehat{M}$ by rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} M'$ and $H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathtt{trust}} H'$, with $\widehat{M} = M' \mid H', \mathcal{D}'' \subseteq \mathcal{D}$ and $H' \in \mathcal{H}_\delta$. As $\mathcal{D}' := \mathcal{D} \setminus \mathsf{nds}(M) \neq \emptyset$ (because it is easy to prove that $\mathcal{D}'' \subseteq \mathsf{nds}(H) \neq \emptyset$) then we can apply (Obs) and obtain $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathtt{trust}} M'$. As $M \approx_\delta N$ then there is N' such that $N \stackrel{m!\tilde{v}\blacktriangleright\mathcal{D}'}{\Longrightarrow}_{\mathtt{trust}} N'$ with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

$$N \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathtt{trust}} N_2 \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} N'$$

with $\mathcal{D}' := \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \neq \emptyset$. As $H \approx_{\mathtt{trust}} K$ then $K \stackrel{m?\tilde{v}\triangleright\mathcal{D}''}{\Longrightarrow}_{\mathtt{trust}} K'$ with $H' \approx_{\mathtt{trust}} K'$ and $K' \in \mathcal{H}_\delta$. Then there are K_1 and K_2 such that

$$K \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} K_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathtt{trust}} K_2 \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} K'.$$

By Lemma A.7 we have $M \equiv \widehat{M} \mid m[P]_T$ and $N_1 \equiv \widehat{N} \mid m[Q]_T$, for some $\widehat{M}, \widehat{N}, T, Q$ and P and $\widehat{\mathcal{D}} := \{n : T(m,n) \geq \mathtt{trust}\}$ and $\widehat{\mathcal{D}'} := \{n : T(m,n) \geq \mathtt{trust}\}$. Then $\widehat{\mathcal{D}'} = \mathcal{D}$. By several applications of Lemma A.6 and one application of rule (Sync), as $\mathcal{D}'' \subseteq \mathcal{D}$, we have

$$N \mid K \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} N_1 \mid K_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} N_2 \mid K_2 \stackrel{\tau}{\Longrightarrow}_{\mathtt{trust}} N' \mid K'.$$

As $\mathsf{nds}(K) = \mathsf{nds}(H)$ it holds that

$$\begin{array}{ll} & \mathcal{D} \setminus \mathsf{nds}(M \mid H) \\ = & \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ = & \mathcal{D} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) = \emptyset. \end{array}$$

Then by one application of rules(Shh) we have $N \mid K \stackrel{\tau}{\Longrightarrow}_{\rho} N' \mid K'$ and $(M' \mid H', N' \mid K') \in \mathcal{S}$, as required.

– Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} \widehat{M}$ by rule (Sync) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathtt{trust}} M'$ and $H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} H'$, with $\widehat{M} = M' \mid H', \mathcal{D}'' \subseteq \mathcal{D}$ and $H' \in \mathcal{H}_\delta$. As $M \approx_\delta N$ then there is N' such that $N \overset{m?\tilde{v}\triangleright\mathcal{D}''}{\Longrightarrow}_{\mathtt{trust}} N'$ with $M' \approx_\delta N'$. This implies that there are N_1 and N_2 such that

$$N \overset{\tau}{\Longrightarrow}_{\mathtt{trust}} N_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathtt{trust}} N_2 \overset{\tau}{\Longrightarrow}_{\mathtt{trust}} N'.$$

As $\mathcal{D}' := \mathcal{D} \backslash \mathsf{nds}(H) \neq \emptyset$ (because it is easy to prove that $\mathcal{D}'' \subseteq \mathsf{nds}(M) \neq \emptyset$) we can apply rule (Obs) and obtain $H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathtt{trust}} H'$, with $\mathcal{D}' := \mathcal{D} \setminus \mathsf{nds}(H) \neq \emptyset$. As $H \approx_{\mathtt{trust}} K$ then $K \overset{m!\tilde{v}\blacktriangleright\mathcal{D}'}{\Longrightarrow}_{\mathtt{trust}} K'$, with $H' \approx_{\mathtt{trust}} K'$ and $K' \in \mathcal{H}_\delta$. This implies that there are K_1 and K_2 such that

$$K \overset{\tau}{\Longrightarrow}_{\mathtt{trust}} K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\mathtt{trust}} K_2 \overset{\tau}{\Longrightarrow}_{\mathtt{trust}} K'.$$

By Lemma A.7 we have $H \equiv \widehat{H} \mid m[P]_T$ and $K_1 \equiv \widehat{K} \mid m[Q]_T$, for some $\widehat{H}, \widehat{K}, T, Q$ and P and $\mathcal{D} := \{n : T(m,n) \geq \mathtt{trust}\}$ and $\widehat{\mathcal{D}'} := \{n : T(m,n) \geq \mathtt{trust}\}$. Then $\widehat{\mathcal{D}'} = \mathcal{D}$. By several applications of Lemma A.6 and one application of rule (Sync), as $\mathcal{D}'' \subseteq \mathcal{D}$, we have

$$N \mid K \overset{\tau}{\Longrightarrow}_{\mathtt{trust}} N_1 \mid K_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathtt{trust}} N_2 \mid K_2 \overset{\tau}{\Longrightarrow}_{\mathtt{trust}} N' \mid K'.$$

As $\mathsf{nds}(K) = \mathsf{nds}(H)$ and as by Lemma A.8 $\mathcal{D} \setminus \mathsf{nds}(M) = \mathcal{D} \setminus \mathsf{nds}(N)$ it holds that

$$\begin{aligned} & \mathcal{D} \setminus \mathsf{nds}(M \mid H) \\ = \ & \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ = \ & \mathcal{D} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) = \emptyset. \end{aligned}$$

Then by one application of rule (Shh) we have $N \mid K \overset{\tau}{\Longrightarrow}_\rho N' \mid K'$ and $(M' \mid H', N' \mid K') \in \mathcal{S}$, as required.

- Let $M \mid O \xrightarrow{\tau}_\rho \widehat{M}$ by an application of rule (Par). This case is easy to prove. □

A.3 Proofs of Chapter 8

Similarly to Lemma A.6, also in TCTAN all τ-actions propagate through parallel composition.

Lemma A.11. *If* $t \triangleright M \xrightarrow{\tau}_\rho t \triangleright M'$ *then* $t \triangleright M \mid N \xrightarrow{\tau}_\rho t \triangleright M' \mid N$ *and* $t \triangleright N \mid M \xrightarrow{\tau}_\rho t \triangleright N \mid M'$*, for all networks* N*.*

Proof of Theorem 8.8

- Let $t \triangleright M \mid O \xrightarrow{\alpha'}_\rho t \triangleright \widehat{M}$, for $\alpha' \in \{\tau, m!\tilde{v}\blacktriangleright\mathcal{D}, m?\tilde{v}\triangleright\mathcal{D}\}$. The proofs of these cases are similar to the proofs of Theorem 7.16 at page 123.
- Let $t \triangleright M \mid O \xrightarrow{\mathsf{tick}}_\rho t' \triangleright \widehat{M}$ by an application of rule (ParTick) because $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ and $t \triangleright O \xrightarrow{\mathsf{tick}}_\rho t' \triangleright O'$, and $\widehat{M} = M' \mid O'$. As $t \triangleright M \approx'_\delta t \triangleright N$ then $t \triangleright N \stackrel{\mathsf{tick}}{\Longrightarrow}_\rho t' \triangleright N'$, with $t' \triangleright M' \approx'_\delta t' \triangleright N'$. This implies that

$$t \triangleright N \Rightarrow_\rho t \triangleright N_1 \xrightarrow{\mathsf{tick}}_\rho t' \triangleright N_2 \Rightarrow_\rho t' \triangleright N'.$$

  Thus, by several application of Lemma A.11 and an application of rule (ParTick) we have

$$t \triangleright N \mid O \Rightarrow_\rho t \triangleright N_1 \mid O \xrightarrow{\mathsf{tick}}_\rho t' \triangleright N_2 \mid O' \Rightarrow_\rho t' \triangleright N' \mid O'.$$

  Thus $t \triangleright N \mid O \stackrel{\mathsf{tick}}{\Longrightarrow}_\rho t' \triangleright N' \mid O'$ and $(t' \triangleright M' \mid O', t' \triangleright N' \mid O') \in \mathcal{S}$, as required. □

In order to prove Theorem 8.9 of Soundness, we need to prove the following auxiliary lemmas.

Lemma A.12.

1. *If* $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright M'$*, where* $\mathcal{D}$ *contains more than one node, then there are* N, P, T *such that* $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$*.*
2. *If* $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright M'$*, with* $\mathcal{D} = n$*, for some* n*, then there are* N, P, T *such that* $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$ *or* $M \equiv m[\sigma!\langle\tilde{v}\rangle_n.P]_T \mid N$*.*

Proof Similar to the proof of Lemma A.9 at page 183. □

Lemma A.13.

1. *If* $t \triangleright M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\sigma t \triangleright M'$ *then* $t \triangleright M \downarrow^\sigma_n$*, for all* $n \in \mathcal{D}$*.*
2. *If* $t \triangleright M \downarrow^\sigma_n$ *then there is a value* $\tilde{v}$ *and a set of nodes* $\mathcal{D}$*, with* $n \in \mathcal{D}$*, such that* $t \triangleright M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\sigma t \triangleright M'$*,*

Proof By Lemma A.12 and by Definition 8.1. □

Proof of Theorem 8.10

By induction on the length of the proof of $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$. The base cases are when the transitions are derived by the applications of rule (Tick-**0**), (Tick) and (SumTick). It is straightforward to prove that the statement holds for these rules. As to the inductive case, let $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ by an application of rule (ParTick). This implies that $M = M_1 \mid M_2$, for some M_1 and M_2, with $t \triangleright M_1 \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M_1'$, $t \triangleright M_2 \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M_2'$ and $M' = M_1' \mid M_2'$. As $M = M_1 \mid M_2$, the transition $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M''$ can be derived only by applying rule (ParTick) where $t \triangleright M_1 \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M_1''$, $t \triangleright M_2 \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M_2''$ and $M'' = M_1'' \mid M_2''$, for some M_1'', M_2''. By inductive hypothesis it holds that M_i' and M_i'' are syntactically the same, for $i \in \{1, 2\}$. This implies that M' and M'' are syntactically the same. □

Proof of Theorem 8.11

By induction on the structure of M. If $M = \mathbf{0}$ the statement does not apply. If M is composed by only one node and $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright N$, this can be derived only by an application of rule (MCastT), (UCastT), or (SumT). In all cases the possibility that $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$, for some M' and $\rho \neq \mathtt{bad}$, is excluded by the requirement of rule (Tick). As to the inductive case, let M be composed by at least two nodes and $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright N$ by an application of rule (SyncT) or (ParT). We examine only the first case. The other one is similar. If $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright N$ is given by an application of rule (SyncT), then $M = M_1 \mid M_2$ for some M_1 and M_2, with $t \triangleright M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright M_1'$, $t \triangleright M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_\sigma t \triangleright M_2'$, for some $\mathcal{D}'$, and $N = M_1' \mid M_2'$ (the converse is similar). As $M = M_1 \mid M_2$ the only rule for deriving a tick-transition from M is (ParTick). However, the inductive hypothesis guarantees that $t \triangleright M_1 \xrightarrow{\mathsf{tick}}_\rho t' \triangleright \widehat{M}$ for no network $\widehat{M}$ and $\rho \neq \mathtt{bad}$; then $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ for no network M' and $\rho \neq \mathtt{bad}$. □

Proof of Theorem 8.12

By contradiction and then by induction on the structure of M. We prove that if $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright N$ for no network N and $\rho \neq \mathtt{bad}$ then there is a network M' such that $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright M'$. Let us proceed by induction on the structure of M.

- Let $M = \mathbf{0}$. Then $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M$ by an application of rule (Tick-**0**). So, the statement does not apply.
- Let $M = m[P]_T$. We proceed by induction on the structure of P.
  - If $P = \mathsf{nil}$ or $P = \sigma?(\tilde{x}).Q$ then $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ by an application of rules (Tick). Thus the statement does not apply.
  - If $P = \sigma!\langle\tilde{v}\rangle.P$ then $t \triangleright M \not\xrightarrow{\mathsf{tick}}_\rho$ by requirement of rule (Tick). However, $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright M'$, by an application of rule (MCastT), in contradiction with the hypothesis.
  - If $P = \sigma!\langle\tilde{u}\rangle_n.P$ and $T(m, n) \geq \sigma$ then $t \triangleright M \not\xrightarrow{\mathsf{tick}}_\rho$ by requirement of rule (Tick). However, $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright n}_\sigma t \triangleright M'$, by an application of rule (UCastT), in contradiction with the hypothesis.

- If $P = \sigma!\langle\tilde{u}\rangle_n.P$ and $T(m,n) < \sigma$ then $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright n}\!\!\!\!\!\!/\;\;\;_\sigma$ by requirement of rule (UCastT) but $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ by an application of rules (Tick). Thus the statement does not apply.
- If $P = P' + Q$ then $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M'$ by an application of rules (SumTick) because $t \triangleright m[P']_T \xrightarrow{\mathsf{tick}}_\rho t' \triangleright m[P']_{T'}$ and $t \triangleright n[Q]_T \xrightarrow{\mathsf{tick}}_\rho t' \triangleright m[Q]_{T'}$. Thus the statement does not apply.
- If $P = P' + Q$ and $t \triangleright M \xrightarrow{\mathsf{tick}}\!\!\!\!\!\!/\;\;\;_\rho$ because either $t \triangleright m[P']_T \xrightarrow{\mathsf{tick}}\!\!\!\!\!\!/\;\;\;_\rho$ or $t \triangleright m[Q]_T \xrightarrow{\mathsf{tick}}\!\!\!\!\!\!/\;\;\;_\rho$. By inductive hypothesis it means that either $t \triangleright m[P']_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright m[P'']_T$ or $t \triangleright m[Q]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright m[Q']_T$, for some P'', Q', then by rule (SumT) $t \triangleright M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright M'$ in contradiction with the hypothesis.
- If $P = [\tilde{v}\,\mathsf{op}\,\tilde{v}']P', Q$ with $\tilde{v}\,\mathsf{op}\,\tilde{v}' = \mathtt{true}$ then by an application of rule (ThenT) we can apply the inductive hypothesis to conclude that we fall in one of the previous cases.
- If $P = [\tilde{v}\,\mathsf{op}\,\tilde{v}']P', Q$, $\tilde{v}\,\mathsf{op}\,\tilde{v}' = \mathtt{false}$, by an application of rule (ElseT) we can apply the inductive hypothesis to conclude that we fall in one of the previous cases.
- If $P = H\langle\tilde{v}\rangle$ the constraint of guarded recursion ensures us that by an application of rule (RecT) we can apply the inductive hypothesis and we fall in one of the previous cases.

- Let $M = M_1 \mid M_2$. A transition of the form $t \triangleright M \xrightarrow{\mathsf{tick}}_\rho t' \triangleright N$ can be derived only by an application of rule (ParTick). If this action cannot be performed then at least one of the premises of rule (ParTick) does not hold:
  - If $t \triangleright M_1 \xrightarrow{\mathsf{tick}}_\rho t' \triangleright \widehat{M}$ for no network $\widehat{M}$ and $\rho \neq \mathtt{bad}$, then by inductive hypothesis we have $t \triangleright M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright M_1'$, for some M_1'. By an application of rule (ParT) it holds that $t \triangleright M_1 \mid M_2 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma t \triangleright M_1' \mid M_2$, in contradiction with the hypothesis.
  - If $t \triangleright M_2 \xrightarrow{\mathsf{tick}}_\rho t' \triangleright M_2'$ for no network M_2' and $\rho \neq \mathtt{bad}$ then we can reason as in the previous sub-case.

□

Proof of Theorem 8.14

We prove that the relation

$$\mathcal{S} \stackrel{\text{def}}{=} \{\big(t \triangleright M \mid H\,,\, t \triangleright N \mid K\big) :\; t \triangleright H, t \triangleright K \in \mathcal{H}'_\delta,$$

$$t \triangleright M \approx_\delta t \triangleright N,\, t \triangleright H \approx_{\mathsf{trust}} t \triangleright K \text{ and } \mathsf{nds}(H){=}\mathsf{nds}(K)\}$$

is a δ-timed bisimulation. We proceed by case analysis on the transition $t \triangleright M \mid H \xrightarrow{\alpha}_\rho t' \triangleright \widehat{M}$, with $\rho{\leq}\delta$.

- Let $t \triangleright M \mid H \xrightarrow{\alpha'}_\rho t \triangleright \widehat{M}$, for $\alpha' \in \{\tau, m!\tilde{v}\blacktriangleright\mathcal{D}, m?\tilde{v}\triangleright\mathcal{D}\}$ and $\rho \neq \mathsf{trust}$. The proofs of these cases are similar to the proofs of Theorem 7.20 at page 125.
- Let $t \triangleright M \mid H \xrightarrow{\alpha'}_\rho t \triangleright \widehat{M}$, for $\alpha' \in \{\tau, m!\tilde{v}\blacktriangleright\mathcal{D}, m?\tilde{v}\triangleright\mathcal{D}\}$ and $\rho = \mathsf{trust}$. The proofs of these cases are similar to the proofs of Theorem 7.20 at page 125.

- Let $t \rhd M \mid H \xrightarrow{\mathsf{tick}}_{\mathsf{trust}} t' \rhd \widehat{M}$ by an application of rule (ParTick) because $t \rhd M \xrightarrow{\mathsf{tick}}_{\mathsf{trust}} t' \rhd M'$ and $t \rhd H \xrightarrow{\mathsf{tick}}_{\mathsf{trust}} t' \rhd H'$, and $\widehat{M} = M' \mid H'$. As $t \rhd M \approx'_\delta t \rhd N$ then $t \rhd N \stackrel{\mathsf{tick}}{\Longrightarrow}_{\mathsf{trust}} t' \rhd N'$, with $t' \rhd M' \approx'_\delta t' \rhd N'$. This implies that

$$t \rhd N \Rightarrow_{\mathsf{trust}} t \rhd N_1 \xrightarrow{\mathsf{tick}}_{\mathsf{trust}} t' \rhd N_2 \Rightarrow_{\mathsf{trust}} t' \rhd N'.$$

  Moreover, as $t \rhd H \approx_{\mathsf{trust}} t \rhd K$ then $t \rhd K \stackrel{\mathsf{tick}}{\Longrightarrow}_{\mathsf{trust}} t' \rhd K'$, with $t' \rhd H' \approx'_\delta t' \rhd K'$ and $H', K' \in \mathcal{H}'_\delta$. This implies that

$$t \rhd K \Rightarrow_{\mathsf{trust}} t \rhd K_1 \xrightarrow{\mathsf{tick}}_{\mathsf{trust}} t' \rhd K_2 \Rightarrow_{\mathsf{trust}} t' \rhd K'.$$

  Thus, by several applications of Lemma A.11 and an application of rule (ParTick) we have

$$t \rhd N \mid K \Rightarrow_{\mathsf{trust}} t \rhd N_1 \mid K_1 \xrightarrow{\mathsf{tick}}_{\mathsf{trust}} t' \rhd N_2 \mid K_2 \Rightarrow_{\mathsf{trust}} t' \rhd N' \mid K'.$$

  Thus $t \rhd N \mid K \stackrel{\mathsf{tick}}{\Longrightarrow}_{\mathsf{trust}} t' \rhd N' \mid K'$ and $(t' \rhd M' \mid H', t' \rhd N' \mid K') \in \mathcal{S}$, as required.

□

References

1. IEEE 802.15. ANSI/IEEE Standard 802.15: Wireless MAC and PHY Specifications for Low Rate WPAN. IEEE Computer Society, 2005.
2. IEEE 802.16. ANSI/IEEE Standard 802.16: Air Interface for Fixed Broadband Wireless Access Systems. IEEE Computer Society, 2004.
3. Martín Abadi. Secrecy by Typing in Security Protocols. *Journal of the ACM*, 46(5):749–786, 1999.
4. Martín Abadi and Bruno Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. *Journal of the ACM*, 52(1):102–146, 2005.
5. Martin Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A Calculus for Access Control in Distributed Systems. *ACM Transaction on Programming Languages and Systems*, 15(4):706–734, 1993.
6. Martín Abadi and Andrew D. Gordon. A Calculus for Cryptographic Protocols: The spi Calculus. *Information and Compututation*, 148(1):1–70, 1999.
7. Alfarez Abdul-Rahman and Stephen Hailes. Supporting Trust in Virtual Communities. In *HICSS*, volume 6, pages 6007–6016. IEEE Computer Society, 2000.
8. Luca Aceto and Matthew Hennessy. Towards Action-Refinement in Process Algebras. *Information and Computation*, 103(2):204–269, 1993.
9. Gergely Ács, Levente Buttyán, and István Vajda. Provably Secure On-Demand Source Routing in Mobile Ad Hoc Networks. *IEEE Transaction on Mobile Computing*, 5(11):1533–1546, 2006.
10. Efthimia Aivaloglou, Stefanos Gritzalis, and Charalabos Skianis. Trust Establishment in Sensor Networks: Behaviour-based, Certificate-based and a Combinational Approach. *International Journal of System of Systems Engineering*, 1(1–2):128–148, 2008.
11. Alessandro Aldini, Mario Bravetti, and Roberto Gorrieri. A Process-Algebraic Approach for the Analysis of Probabilistic Non-interference. *Journal of Computer Security*, 12(2):191–245, 2004.
12. Alessandro Aldini and Alessandra Di Pierro. A Quantitative Approach to Noninterference for Probabilistic Systems. *Electronic Notes of Theoretical Computer Science*, 99:155–182, 2004.
13. Todd R. Andel and Alec Yasinsac. Automated Evaluation of Secure Route Discovery in MANET Protocols. In *SPIN*, volume 5156 of *Lecture Notes in Computer Science*, pages 26–41. Springer, 2008.
14. Ross J. Anderson. *Security Engineering: a Guide to Building Dependable Distributed Systems*. Willey Edition, 2001.

15. Andrew W. Appel and Edward W. Felten. Proof-Carrying Authentication. In *ACM Conference on Computer and Communications Security*, pages 52–62. ACM Press, 1999.
16. AVISPA. website `http://www.avispa-project.org/`.
17. Jos C. M. Baeten. A Brief History of Process Algebra. *Theoretical Computer Science*, 335(2-3):131–146, 2005.
18. Jos C. M. Baeten and Jan A. Bergstra. Real Time Process Algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.
19. Jos C. M. Baeten and Jan A. Bergstra. Discrete Time Process Algebra. *Formal Aspects of Computing*, 8(2):188–208, 1996.
20. Jos C. M. Baeten, Jan A. Bergstra, and Michel A. Reniers. Discrete Time Process Algebra with Silent Step. In *Proof, Language, and Interaction*, pages 535–570. The MIT Press, 2000.
21. Jos C. M. Baeten and C. A. Middelburg. *Process Algebra with Timing*. EATCS Series. Springer-Verlag, 2002.
22. Jos C. M. Baeten and M. A. Reniers. Timed Process Algebra (With a Focus on Explicit Termination and Relative-Timing). In *SFM*, volume 3185 of *Lecture Notes in Computer Science*, pages 59–97. Springer-Verlag, 2004.
23. Jos C. M. Baeten and Michel A. Reniers. Timed Process Algebra (With a Focus on Explicit Termination and Relative-Timing). In *SFM*, volume 3185 of *Lecture Notes in Computer Science*, pages 59–97. Springer, 2004.
24. Venkat Balakrishnan, Vijay Varadharajan, and Uday Tupakula. *Trust Management in Mobile Ad Hoc Networks*, pages 473–500. Computer Communications and Networks. Springer, 2009.
25. Dirk Balfanz, Drew Dean, and Mike Spreitzer. A Security Infrastructure for Distributed Java Applications. In *IEEE Symposium on Security and Privacy*, pages 15–26. IEEE Computer Society, 2000.
26. David A. Basin, Sebastian Mödersheim, and Luca Viganò. OFMC: A Symbolic Model Checker for Security Protocols. *Int. J. Inf. Sec.*, 4(3):181–208, 2005.
27. Hans Bekič. Towards a Mathematical Theory of Processes. Technical Report TR 25.125, IBM Laboratory Vienna, 19t1.
28. David Elliott Bell and Leonard J. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report MTR-2997, MITRE Corporation, 1976.
29. Jan A. Bergstra and Jan Willem Klop. Fixed Point Semantics in Process Algebra. Technical Report Tech. Report IW 208, Mathematical Centre, Amsterdam, 1982.
30. Jan A. Bergstra and Jan Willem Klop. Process Algebra for Synchronous Communication. *Information and Computation*, 60:109–137, 1984.
31. Jan A. Bergstra and Jan Willem Klop. Algebra for Communicating Processes with Abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.
32. Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science Inc., New York, NY, USA, 2001.
33. Marco Bernardo and Roberto Gorrieri. Extended markovian process algebra. In *CONCUR*, volume 1119 of *Lecture Notes in Computer Science*, pages 315–330. Springer, 1996.
34. Gérard Berry and Gérard Boudol. The Chemical Abstract Machine. *Theoretical Computer Science*, 96(1):217–248, 1992.
35. Gérard Berry and Laurent Cosserat. The ESTEREL Synchronous Programming Language and its Mathematical Semantics. Technical Report 842, INRIA, Sophia-Antipolis, 1988.
36. Gérard Berry and Georges Gonthier. The Esterel Synchronous Programming Language: Design, Semantics, Implementation. *Science of Computer Programming*, 19(2):87–152, 1992.

37. Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of Trust in Open Networks. In *ESORICS*, volume 875 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 1994.
38. Karthikeyan Bhargavan, Davor Obradovic, and Carl A. Gunter. Formal Verification of Standards for Distance Vector Routing Protocols. *Journal of the ACM*, 49(4):538–576, 2002.
39. Karthikeyan Bhargavan, Davor Obradovic, and Carl A. Gunter. Formal Verification of Standards for Distance Vector Routing Protocols. *Journal of the ACM*, 49(4):538–576, 2002.
40. Kenneth J. Biba. Integrity Considerations for Secure Computer Systems. Technical Report MTR-3153, MITRE Corporation, 1975.
41. Bruno Blanchet. From Secrecy to Authenticity in Security Protocols. In *SAS*, volume 2477 of *Lecture Notes in Computer Science*, pages 342–359. Springer, 2002.
42. Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. A Static Analyzer for Large Safety-Critical Software. In *SIGPLAN*, pages 196–207. ACM, 2003.
43. Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The KeyNote Trust-Management System Version 2. RFC 2704, 1999.
44. Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. KeyNote: Trust Management for Public-Key Infrastructures (Position Paper). In *Security Protocols Workshop*, volume 1550 of *Lecture Notes in Computer Science*, pages 59–63. Springer, 1998.
45. Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The Role of Trust Management in Distributed Systems Security. In *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 185–210. Springer, 1999.
46. Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. In *Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society, 1996.
47. Matt Blaze, Joan Feigenbaum, and Martin Strauss. Compliance Checking in the PolicyMaker Trust Management System. In *Financial Cryptography*, pages 254–274, 1998.
48. Matt Blaze, Joan Ioannidis, and Angelos D. Keromytis. Trust Management and Network Layer Security Protocols. In *Security Protocols Workshop*, volume 1796 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 2000.
49. Chiara Bodei, Mikael Buchholtz, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Automatic Validation of Protocol Narration. In *CSFW*, pages 126–140. IEEE Computer Society, 2003.
50. Chiara Bodei, Mikael Buchholtz, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Static Validation of Security Protocols. *Journal of Computer Security*, 13(3):347–390, 2005.
51. Chiara Bodei, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Static Analysis for Secrecy and Non-interference in Networks of Processes. In *PaCT*, volume 2127 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2001.
52. Chiara Bodei, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Static Analysis for the pi-Calculus with Applications to Security. *Information and Computation*, 168(1):68–92, 2001.
53. Gérard Boudol. Secure Information Flow as a Safety Property. In *FAST*, volume 5491 of *Lecture Notes in Computer Science*, pages 20–34. Springer, 2009.
54. Gérard Boudol and Ilaria Castellani. Noninterference for Concurrent Programs and Thread Systems. *Theoretical Computer Science*, 281(1-2):109–130, 2002.
55. Ed Brinksma, Joost-Pieter Katoen, Rom Langerak, and Diego Latella. A stochastic causality-based process algebra. *The Computer Journal*, 38(7):552–565, 1995.

56. Stephen D. Brookes, Charles A. R. Hoare, and A. W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM*, 31(3):560–599, 1984.
57. Levente Buttyán and Jean-Pierre Hubaux. *Security and Cooperation in Wireless Networks*. Cambridge University Press, 2008.
58. Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A Formal Model for Trust in Dynamic Networks. In *SEFM*, pages 54–. IEEE Computer Society, 2003.
59. Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A Calculus for Trust Management. In *FSTTCS*, volume 3328 of *Lecture Notes in Computer Science*, pages 161–173. Springer, 2004.
60. Luca Cardelli and Andrew D. Gordon. Mobile Ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.
61. Sudip Chakraborty and Indrajit Ray. TrustBAC: Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems. In *SACMAT*, pages 49–58. ACM, 2006.
62. Bryan Chess and Gary McGraw. Static Analysis for Security. *Security and Privacy*, 2(6):76–79, 2004.
63. Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, and Martin Strauss. REFEREE: Trust Management for Web Applications. *Computer Networks and ISDN Systems*, 29(8-13):953–964, 1997.
64. Andrew Cirillo and James Riely. Access Control Based on Code Identity for Open Distributed Systems. In *TGC*, volume 4912 of *Lecture Notes in Computer Science*, pages 169–185. Springer, 2007.
65. Tim Clausen and Philipp Jacquet. Optimized Link State Routing Protocol (OLSR), 2003. RFC 3626.
66. COQ. website `http://coq.inria.fr/`.
67. Flavio Corradini, Gian Luigi Ferrari, and Marco Pistore. On the Semantics of Durational Actions. *Theoretical Computer Science*, 269(1-2):47–82, 2001.
68. Flavio Corradini and Marco Pistore. Closed Interval Process Algebra versus Interval Process Algebra. *Acta Informatica*, 37(7):467–509, 2001.
69. Coverity. website `http://coverity.com/`.
70. Silvia Crafa and Sabina Rossi. Controlling Information Release in the π-calculus. *Information and Computation*, 205(8):1235–1273, 2007.
71. Willem P. de Roever and Jozef Hooman. Design and Verification in Real-Time Distributed Computing: an Introduction to Compositional Methods. In *PSTV*, pages 37–56. North-Holland, 1989.
72. Robero Di Pietro, Luigi V. Mancini, Yee W. Law, Sandro Etalle, and Paul J. M. Havinga. LKHW: A Directed Diffusion-Based Secure Multicast Scheme for Wireless Sensor Networks. In *ICPP Workshops*, pages 397–413. IEEE Computer Society, 2003.
73. Danny Dolev and Andrew Chi-Chih Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
74. Carl M. Ellison, Bill Frantz, Butler Lampson, Ronald L. Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI Certificate Theory. RFC 2693, 1999.
75. Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-Grained Network Time Synchronization Using Reference Broadcasts. In *OSDI*, volume 36, pages 147–163. USENIX Association, 2002.
76. Jeremy Elson and Kay Römer. Wireless Sensor Networks: a New Regime for Time Synchronization. *Computer Communication Review*, 33(1):149–154, 2003.
77. Cristian Ene and Traian Muntean. A Broadcast based Calculus for Communicating Systems. In *IPDPS*, page 149. IEEE Computer Society, 2001.
78. Fujun Feng, Chuang Lin, and Junshan Li. An Access Control Model for Trustworthy Network. *Networks Security, Wireless Communications and Trusted Computing, International Conference on*, 1:308–311, 2009.

79. David Ferraiolo and Richard Kuhn. Rôle-Based Access Control. In *NIST-NCSC National Computer Security Conference*, pages 554–563. IEEE Computer Society Press, 1992.
80. Robert W. Floyd. Assigning Meanings to Programs. *Mathematical Aspects of Computer Science*, 19(19-32):1, 1967.
81. Riccardo Focardi and Roberto Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3(1):5–33, 1995.
82. Riccardo Focardi and Roberto Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering*, 27(3):550–571, 1997.
83. Riccardo Focardi and Roberto Gorrieri. Classification of Security Properties (Part I: Information Flow). In *FOSAD*, volume 2171 of *Lecture Notes in Computer Science*, pages 331–396. Springer, 2001.
84. Riccardo Focardi, Roberto Gorrieri, and Fabio Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 354–372. Springer, 2000.
85. Riccardo Focardi, Roberto Gorrieri, and Fabio Martinelli. Classification of Security Properties (Part II: Network Security). In *FOSAD*, volume 2946 of *Lecture Notes in Computer Science*, pages 139–185. Springer, 2002.
86. Riccardo Focardi and Fabio Martinelli. A Uniform Approach for the Definition of Security Properties. In *World Congress on Formal Methods*, volume 1708 of *Lecture Notes in Computer Science*, pages 794–813. Springer, 1999.
87. Cédric Fournet, Andrew D. Gordon, and Sergio Maffeis. A Type Discipline for Authorization in Distributed Systems. In *CSF*, pages 31–48. IEEE computer Society, 2007.
88. Diego Gambetta. *Trust: Making and Breaking Cooperative Relations*. Blackwell Publishers, 1988.
89. Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-Sync Protocol for Sensor Networks. In *SenSys*, pages 138–149. ACM Press, 2003.
90. Matthew Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, 2005.
91. Fatemeh Ghassemi, Wan Fokkink, and Ali Movaghar. Restricted Broadcast Process Theory. In *SEFM*, pages 345–354. IEEE Computer Society, 2008.
92. Fatemeh Ghassemi, Wan Fokkink, and Ali Movaghar. Equational Reasoning on Ad Hoc networks. In *FSEN*, volume 5961 of *Lecture Notes in Computer Science*, pages 113–128. Springer, 2009.
93. Pho Duc Giang, Le Xuan Hung, Sungyoung Lee, Young-Koo Lee, and Heejo Lee. A Flexible Trust-Based Access Control Mechanism for Security and Privacy Enhancement in Ubiquitous Systems. In *MUE*, pages 698–703. IEEE Computer Society, 2007.
94. Jens Chr. Godskesen. A Calculus for Mobile Ad Hoc Networks. In *COORDINATION*, volume 4467 of *Lecture Notes in Computer Science*, pages 132–150. Springer, 2007.
95. Jens Chr. Godskesen and Sebastian Nanz. Mobility Models and Behavioural Equivalence for Wireless Networks. In *COORDINATION*, volume 5521 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2009.
96. Joseph A. Goguen and José Meseguer. Security Policies and Security Models. In *Security and Privacy*, pages 11–20. IEEE Computer Society, 1982.
97. Daniele Gorla, Matthew Hennessy, and Vladimiro Sassone. Inferring Dynamic Credentials for Role-based Trust Management. In *PPDP*, pages 213–224. ACM Press, 2006.

98. Roberto Gorrieri, Enrico Locatelli, and Fabio Martinelli. A Simple Language for Real-Time Cryptographic Protocol Analysis. In *ESOP*, volume 2618 of *Lecture Notes in Computer Science*, pages 114–128. Springer, 2003.
99. Roberto Gorrieri, Fabio Martinelli, and Ilaria Matteucci. Towards Information Flow Properties for Distributed Systems. *Electronic Notes in Theoretical Computer Science*, 236:65–84, 2009.
100. Roberto Gorrieri, Fabio Martinelli, Marinella Petrocchi, and Anna Vaccarelli. Formal Anaylsis of Some Timed Security Properties in Wireless Protocols. In *FMOODS*, volume 2884 of *Lecture Notes in Computer Science*, pages 139–154. Springer, 2003.
101. Jean Goubault-Larrecq, Catuscia Palamidessi, and Angelo Troina. A Probabilistic Applied Pi-Calculus. In *APLAS*, volume 4807 of *Lecture Notes in Computer Science*, pages 175–190. Springer, 2007.
102. Tyrone W. A. Grandison. *Trust Management for Internet Applications*. PhD thesis, Department of Computing, University of London, 2003.
103. Jan Friso Groote. Specification and Verification of Real Time Systems in ACP. In *PSTV*, pages 261–274. North-Holland, 1990.
104. B. S. I. Group. Specification of the Bluetooth System. `www.bluetooth.com`, 2004.
105. Trusted Computing Group. website `http://www.trustedcomputinggroup.org/`.
106. Barbara Guttman, Edward Roback, National Institute of Standards, and Technology (U.S.). *An Introduction to Computer Security: the NIST Handbook*. U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 2006. Special Pubblication 800-12.
107. Zygmunt J. Haas. A New Routing Protocol for the Reconfigurable Wireless Networks. In *ICUPC*, pages 562–566. IEEE Computer Society, 1997.
108. Nevin Heintze and John G. Riecke. The SLam Calculus: Programming with Secrecy and Integrity. In *POPL*, pages 365–377. ACM Press, 1998.
109. Matthew Hennessy. *Algebraic Theory of Processes*. The MIT Press, Cambridge, MA, 1988.
110. Matthew Hennessy. The Security Picalculus and Non-Interference. *Journal of Logic and Algebraic Programming*, 63(1):3–34, 2005.
111. Matthew Hennessy and Tim Regan. A Process Algebra for Timed Systems. *Information and Computation*, 117(2):221–239, 1995.
112. Matthew Hennessy and James Riely. A Typed Language for Distributed Mobile Processes. In *Proc. 25th POPL*. ACM Press, 1998.
113. Amir Herzberg, Yosi Mass, Joris Michaeli, Dalit Naor, and Yiftach Ravid. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. In *IEEE Symposium on Security and Privacy*, pages 2–14. IEEE Computer Society, 2000.
114. Jane Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, University of Edinburgh, 1994.
115. Charles A. R. Hoare. An Axiomatic Basis for Computer Programming. *Communication of the ACM*, 12(10):576–580, 1969.
116. Charles A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, 1978.
117. Charles A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
118. Gerard J. Holzmann. The Model Checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.
119. Yih-Chun Hu, David B. Johnson, and Adrian Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Ad Hoc Networks*, 1(1):175–192, 2003.
120. Yih-Chun Hu and Adrian Perrig. A Survey of Secure Wireless Ad Hoc Routing. *IEEE Security & Privacy*, 2(3):28–39, 2004.

121. Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11(1-2):21–38, 2005.
122. Dijiang Huang and Deep Medhi. A Secure Group Key Management Scheme for Hierarchical Mobile Ad Hoc Networks. *Ad Hoc Networks*, 6(4):560–577, 2008.
123. IEEE 802.11 WG. ANSI/IEEE Standard 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Computer Society, 2007.
124. IEEE 802.11i. ANSI/IEEE Standard 802.11i: Wireless LAN Security Standards. IEEE Computer Society, 2004.
125. Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed Diffusion for Wireless Sensor Networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.
126. Isabelle. website `http://www.cl.cam.ac.uk/research/hvg/Isabelle/index.html`.
127. ISO. ISO/IEC 17799. Information Technology - Security Techniques - Code of Practice for Information Security Management, June 2005.
128. Di Jin, Stamatios Kartalopoulos, and Pramode Verma. *Wireless Ad Hoc Sensor Networks Security*, pages 129–159. Security and Privacy in Mobile and Wireless Networking. Troubador, 2009.
129. David B. Johnson and David A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic Publishers, 1996.
130. Audun Jøsang. A Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–212, 2001.
131. Audun Jøsang. Trust and Reputation Systems. In *FOSAD*, volume 4677 of *Lecture Notes in Computer Science*, pages 209–245. Springer, 2007.
132. Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Simplification and Analysis of Transitive Trust Networks. *Web Intelligence and Agent Systems*, 4(2):139–161, 2006.
133. Audun Jøsang and Roslan Ismail. The Beta Reputation System. In *EC*. ACM, 2002.
134. Audun Jøsang, Roslan Ismail, and Colin Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2):618–644, 2006.
135. Audun Jøsang and Svein J. Knapskog. A Metric for Trusted Systems. In *NIST-NCSC*, 1998.
136. John Felix Charles Joseph, Amitabha Das, and Bu-Sung Lee. *Security Threats in Ad Hoc Routing Protocols*, chapter 18. Computer Communications and Networks. Springer London, 2009.
137. Vladimiro Sassone Julian Rathke and Pawel Sobociński. Semantic Barbs and Biorthogonality. In *FoSSaCS*, volume 4423 of *Lecture Notes in Computer Science*, pages 302–316. Springer, 2007.
138. Krukow Karl. *Towards a Theory of Trust for the Global Ubiquitous Computer*. PhD thesis, BRICS, University of Aarhus, 2006.
139. Maryna Komarova and Miguel Riguidel. Adjustable Trust Model for Access Control. In *ATC*, volume 5060 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2008.
140. Karl Krukow, Mogens Nielsen, and Vladimiro Sassone. A Logical Framework for History-based Access Control and Reputation Systems. *Journal of Computer Security*, 16(1):63–101, 2008.
141. Ajay D. Kshemkalyani. The Power of Logical Clock Abstractions. *Distributed Computing*, 17(2):131–150, 2004.
142. Leslie Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Commununication of the ACM*, 21(7), 1978.

143. Leslie Lamport and P. M. Melliar-Smith. Synchronizing Clocks in the Presence of Faults. *Journal of the ACM*, 32(1):52–78, 1985.
144. Cosimo Laneve and Gianluigi Zavattaro. Foundations of Web Transactions. In *FoSSaCS*, volume 3441 of *Lecture Notes in Computer Science*, pages 282–298. Springer, 2005.
145. Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation Logic: A Logic-Based Approach to Distributed Authorization. *ACM Transactions on Information and System Security*, 6(1):128–171, 2003.
146. Ninghui Li and John C. Mitchell. A Rôle-based Trust-management Framework. In *DISCEX (1)*, pages 201–207. IEEE Computer Society, 2003.
147. Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a Role-Based Trust-Management Framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, 2002.
148. Qun Li and Daniela Rus. Global Clock Synchronization in Sensor Networks. *IEEE Transactions on Computers*, 55(2):214–226, 2006.
149. W. Li, A. Joshi, and T. Finin. Policy-based Malicious Peer Detection in Ad Hoc Networks. In *PASSAT*. IEEE Computer Society, 2009.
150. Javier Lopez, Rodrigo Roman, and Cristina Alcaraz. Analysis of Security Threats, Requirements, Technologies and Standards in Wireless Sensor Networks. In *FOSAD*, volume 5705 of *Lecture Notes in Computer Science*, pages 289–338. Springer, 2009.
151. Gavin Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.
152. Gavin Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. *Software - Concepts and Tools*, 17(3):93–102, 1996.
153. Gavin Lowe and A. W. Roscoe. Using CSP to Detect Errors in the TMN Protocol. *IEEE Transactions on Software Engineering*, 23(10):659–669, 1997.
154. Sergio Maffeis, Martín Abadi, Cédric Fournet, and Andrew D. Gordon. Code-Carrying Authorization. In *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 563–579. Springer, 2008.
155. Stephen Marsh. Trust in Distributed Artificial Intelligence. In *MAAMAW*, volume 830 of *Lecture Notes in Computer Science*, pages 94–112. Springer, 1994.
156. Fabio Martinelli. Analysis of Security Protocols as Open Systems. *Theoretical Computer Science*, 290(1):1057–1106, 2003.
157. Fabio Martinelli. Towards an Integrated Formal Analysis for Security and Trust. In *FMOODS*, volume 3535 of *Lecture Notes in Computer Science*, pages 115–130. Springer, 2005.
158. Fabio Martinelli and Marinenlla Petrocchi. *Formal Techniques for Security Analysis in Wireless Systems*, pages 235–263. Security and Privacy in Mobile and Wireless Networking. Troubador, 2009.
159. Jay McCarthy, Shriram Krishnamurthi, Joshua D. Guttman, and John D. Ramsdell. Compiling Cryptographic Protocols for Deployment on the Web. In *WWW*, pages 687–696. ACM, 2007.
160. John McCarthy. A Basis for a Mathematical Theory of Computation. In *Computer Programming and Formal Systems*, pages 33–70. North-Holland, 1963.
161. D. Harrison McKnight and Norman L. Chervany. The Meaning of Trust. MISRC Working Paper Series Technical Report 96-04, University of Minnesota, Management Information Systems Research Center, 1996.
162. Merrian-Webster. Merrian-Webster Online. website `http://www.merriam-webster.com/`.
163. Massimo Merro. An Observational Theory for Mobile Ad Hoc Networks (full paper). *Information and Computation*, 207(2):194–208, 2009.

164. Massimo Merro and Eleonora Sibilio. A Timed Calculus for Wireless Systems. Technical Report 75/2009, Department of Computer Science - University of Verona, 2009, submitted to a journal.
165. Massimo Merro and Eleonora Sibilio. A Calculus of Trustworthy Ad Hoc Networks. In *FAST*, volume 5983 of *Lecture Notes in Computer Science*, pages 157–172. Springer, 2010.
166. Massimo Merro and Eleonora Sibilio. A Timed Calculus for Wireless Systems. In *FSEN*, volume 5961 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2010.
167. Nicola Mezzetti and Davide Sangiorgi. Towards a Calculus For Wireless Systems. *Electronic Notes in Theoretical Computer Science*, 158:331–353, 2006.
168. Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer Verlag, 1980.
169. Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.
170. Robin Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.
171. Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.
172. Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer Verlag, 1992.
173. John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A Probabilistic Polynomial-Time Process Calculus for the Analysis of Cryptographic Protocols. *Theoretical Computer Science*, 353(1-3):118–164, 2006.
174. Michael Mock, Reiner Frings, Edgar Nett, and Spiro Trikaliotis. Continuous Clock Synchronization in Wireless Real-Time Applications. In *SRDS*, pages 125–133. IEEE Computer Society, 2000.
175. Faron Moller and Chris M. N. Tofts. A Temporal Calculus of Communicating Systems. In *CONCUR*, volume 458 of *Lecture Notes in Computer Science*, pages 401–415. Springer, 1990.
176. Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A Computational Model of Trust and Reputation for E-businesses. In *HICSS*, pages 188–204. IEEE Computer Society.
177. Sebastian Nanz. *Specification and Security Analysis of Mobile Ad-Hoc Networks*. PhD thesis, Imperial College London, 2006.
178. Sebastian Nanz and Chris Hankin. A Framework for Security Analysis of Mobile Wireless Networks. *Theoretical Computer Science*, 367(1-2):203–227, 2006.
179. Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, 1978.
180. James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The Sybil Attack in Sensor Networks: Analysis and Defenses. In *IPSN*, pages 259–268. IEEE Computer Society Press, 2004.
181. Xavier Nicollin and Joseph Sifakis. The Algebra of Timed Processes, ATP: Theory and Application. *Information and Computation*, 114(1):131–178, 1994.
182. Mogens Nielsen and Karl Krukow. On the Formal Modelling of Trust in Reputation-Based Systems. In *Theory Is Forever*, volume 3113 of *Lecture Notes in Computer Science*, pages 192–204. Springer, 2004.
183. Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer, 1999.
184. Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Link State Routing for Mobile Ad Hoc Networks. In *SAINT Workshops*, pages 379–383. ACM, 2003.

185. Xuehai Peng and Chuang Lin. Architecture of Trustworthy Networks. In *DASC*, pages 269–276. IEEE Computer Society, 2006.
186. Charles E. Perkins and Elizabeth M. Belding-Royer. Ad-hoc On-Demand Distance Vector Routing. In *WMCSA*, pages 90–100. IEEE Computer Society, 1999.
187. Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *SIGCOMM*, pages 234–244, 1994.
188. Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. Spins: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521–534, 2002.
189. Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, 1962.
190. Asad A. Pirzada and Chris McDonald. Trust Establishment in Pure Ad-hoc Networks. *Wireless Personal Communications*, 37(1-2):139–168, 2006.
191. Gordon D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.
192. Amir Pnueli. Linear and Branching Structures in the Semantics and Logics of Reactive Systems. In *ICALP*, volume 194 of *LNCS*, pages 15–32. Springer Verlag, 1985.
193. PolySpace. website `http://www.polyspace.com/`.
194. K. V. S. Prasad. A Calculus of Broadcasting Systems. *Science of Computer Programmig*, 25(2-3):285–327, 1995.
195. K.V.S. Prasad. Broadcasting in Time. In *COORDINATION*, volume 1061 of *Lecture Notes in Computer Science*, pages 321–338. Springer Verlag, 1996.
196. PreFast. website `http://www.microsoft.com/whdc/devtools/tools/PREfast.mspx`.
197. Corrado Priami. Stochastic pi-calculus. *The Computer Journal*, 38(7):578–589, 1995.
198. Yacine Rebahi, Vicente Mujica, and Dorgham Sisalem. A Reputation-based Trust Mechanism for Ad Hoc Networks. In *ISCC*, pages 37–42. IEEE Computer Society, 2005.
199. George M. Reed. A Hierarchy of Domains for Real-Time Distributed Computing. Technical Report, Oxford, 1988.
200. Richard P. Reitman and Gregory R. Andrews. An Axiomatic Approach to Information Flow in Programs. *ACM Transactions on Programming Languages and Systems*, 2(1):56–76, 1980.
201. Rodrigo Roman, M.Carmen Fernandez-Zago, Javier Lopez, and Chen Hsiao-Hwa. *Trust and Reputation Systems for Wireless Sensor Networks*, pages 105–127. Security and Privacy in Mobile and Wireless Networking. Troubador, 2009.
202. Kay Römer. Time Synchronization in Ad hoc Networks. In *MobiHoc*, pages 173–182. ACM, 2001.
203. Sini Ruohomaa and Lea Kutvonen. Trust Management Survey. In *iTrust*, volume 3477 of *Lecture Notes in Computer Science*, pages 77–92. Springer, 2005.
204. Peter Y. A. Ryan and Stanley A. Schneider. Process Algebra and Non-Interference. In *CSFW*, pages 214–227. IEEE Computer Society, 1999.
205. Jordi Sabater and Charles Sierra. Regret: A Reputation Model for Gregarious Societies. In *AGENTS*, pages 194–195. ACM Press, 2001.
206. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.
207. Ravi S. Sandhu and Pierangela Samarati. Access Control: Principles and Practice. *IEEE Communications Magazine*, 32:40–48, 1994.
208. Davide Sangiorgi and David Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

209. Kimaya Sanzgiri, Daniel LaFlamme, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. Authenticated Routing for Ad Hoc Networks. *IEEE Journal on Selected Areas in Communication, special issue on Wireless Ad Hoc Networks*, 23(3):598–610, 2005.
210. Dana Scott and Christopher Strachey. Toward a Mathematical Semantics for Computer Languages. In *Proceedings of the Symposium on Computers and Automata*, volume XXI, pages 19–46. Polytechnic Press, 1971.
211. Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Efficient Hierarchical Key Generation and Key Diffusion for Sensor Networks. In *SECON*, pages 76–84. IEEE Communications Society, 2005.
212. Mihail L. Sichitiu and Chanchai Veerarittiphan. Simple, Accurate Time Synchronization for Wireless Sensor Networks. In *WCNC*, pages 1266–1273. IEEE Computer Society, 2003.
213. Barbara Simons, Jennifer L. Welch, and Nancy A. Lynch. An Overview of Clock Synchronization. In *Fault-Tolerant Distributed Computing*, volume 448 of *Lecture Notes in Computer Science*, pages 84–96. Springer, 1990.
214. Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. A Process Calculus for Mobile Ad Hoc Networks. *Science of Computer Programming*, in press, 2009.
215. Elankayer Sithirasenan, Saad Zafar, and Vallipuram Muthukkumarasamy. Formal Verification of the IEEE 802.11i WLAN Security Protocol. In *ASWEC*, pages 181–190. IEEE Computer Society, 2006.
216. Dawn Xiaodong Song, Sergey Berezin, and Adrian Perrig. Athena: A Novel Approach to Efficient Automatic Security Protocol Analysis. *Journal of Computer Security*, 9(1/2):47–74, 2001.
217. Mudhakar Srivatsa, Li Xiong, and Ling Liu. TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In *WWW*, pages 422–431. ACM, 2005.
218. William Stallings. *Cryptography and Network Security Principles and Practices*. Pearson Education Inc., NJ, USA, 2006. 4th edition.
219. Ben Strulo. *Process Algebra for Discrete Event Simulation*. PhD thesis, Imperial College, 1993.
220. Weilian Su and Ian F. Akyildiz. Time-Diffusion Synchronization Protocols for Sensor Networks. *IEEE/ACM Transactions on Networking*, 13(2):384–397, 2005.
221. Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock Synchronization for Wireless Sensor Networks: a Survey. *Ad Hoc Networks*, 3(3):281–323, 2005.
222. Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, 2003.
223. George Theodorakopoulos and John S. Baras. On Trust Models and Trust Evaluation Metrics for Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 24(2):318–328, 2006.
224. M. Llanos Tobarra, Diego Cazorla, Fernando Cuartero, Gregorio Díaz, and María-Emilia Cambronero. Model Checking Wireless Sensor Network Security Protocols: TinySec + LEAP + TinyPK. *Telecommunication Systems*, 40(3-4):91–99, 2009.
225. I. T. Union. ITU-T recommendation x.509 (08/97) - information technology - open systems interconnection - the directory. Authentication framework, 1997.
226. Rob J. van Glabbeek. The Linear Time-Branching Time Spectrum (Extended Abstract). In *CONCUR*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 1990.
227. Sudarshan Vasudevan, Jim Kurose, and Don Towsley. Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks. In *ICNP*, pages 350–360. IEEE Computer Society, 2004.
228. Luca Viganò. Automated Security Protocol Analysis with the AVISPA Tool. *Electronic Notes Theoretical Computer Science*, 155:61–86, 2006.

229. Dennis M. Volpano and Geoffrey Smith. Secure Information Flow in a Multi-Threaded Imperative Language. In *POPL*, pages 355–364. ACM Press, 1998.
230. Stephen Weeks. Understanding Trust Management Systems. In *IEEE Symposium on Security and Privacy*, pages 94–105. IEEE Computer Society, 2001.
231. Edward Wobber, Martin Abadi, and Michael Burrows. Authentication in the Taos Operating System. *ACM Transactions on Computer Systems*, 12(1):3–32, 1994.
232. Shahan Yang and John S. Baras. Modeling Vulnerabilities of Ad Hoc Routing Protocols. In *SASN*, pages 12–20. ACM, 2003.
233. Wang Yi. Real-Time Behaviour of Asynchronous Agents. In *CONCUR*, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer Verlag, 1990.
234. Wang Yi. *A Calculus of Real Time Systems*. Ph.D Thesis, Chalmers University, 1991.
235. Suyoung Yoon, Chanchai Veerarittiphan, and Mihail L. Sichitiu. Tiny-sync: Tight Time Synchronization for Wireless Sensor Networks. *ACM Transactions on Sensor Networks*, 3(2):81–118, 2007.
236. Ender Yuksel, Hanne Riis Nielson, Christoffer Rosenkilde Nielsen, and Mehmet Bulent Orencik. A Secure Simplification of the PKMv2 Protocol in IEEE 802.16e-2005. In *FCS-ARSPA*, pages 149–164.
237. Giorgos Zacharia and Pattie Maes. Trust Management through Reputation Mechanisms. *Applied Artificial Intelligence*, 14(9):881–907, 2000.
238. Manel Guerrero Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *WiSe*, pages 1–10. ACM, 2002.
239. Paul Zimmerman. *The Official PGP User's Guide*. MIT Press, Cambridge, 1995.