Cooperative Intrusion Detection For The Next Generation Carrier Ethernet

Pan Jieke

DI–FCUL TR–2008–10

March 2008

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

Technical reports are available at `http://www.di.fc.ul.pt/tech-reports`. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

COOPERATIVE INTRUSION DETECTION FOR THE NEXT GENERATION CARRIER ETHERNET
(Versão Pública)

Pan Jieke

MESTRADO EM INFORMÁTICA

2007

COOPERATIVE INTRUSION DETECTION FOR THE NEXT GENERATION CARRIER ETHERNET
(Versão Pública)

Pan Jieke

Dissertação submetida para obtenção do grau de
MESTRE EM INFORMÁTICA

pela

Faculdade de Ciências da Universidade de Lisboa

Departamento de Informática

Orientador:

Miguel Nuno Dias Alves Pupo Correia

2007

Resumo

Hoje em dia os elementos de rede (NEs) da camada 2 do modelo OSI, *bridges* ou *switches*, são componentes complexos, com centenas de milhares de linhas de código, que podem ser vulneráveis a ataques, permitindo até a execçuão remota de código. Este trabalho tem como objectivo a criação de um sistema para proteger infra-estruturas de rede Carrier Ethernet de ataques lançados por NEs maliciosos contra o protocolo de gestão de ligações, o Spanning Tree Protocol, e as sua variantes.

Na tese é proposto que os NEs sejam equipados com um componente de detecção de intrusões. Cada um dos detectores utiliza um mecanismo da detecção de intrusões baseada em especificacão e inspecciona o comportamento dos outros NEs através da análise das mensagens recebidas. O comportamento correcto dos NEs é descrito tendo em conta a especificação normalizada do protocolo STP. Se existir um desvio entre um comportamento esperado e o actual, o NE é suspeito de ser malicioso. A especificação é estendida com anotações de padrões temporais, de modo a detectar desvios do protocolo por parte dos NEs localmente. Os resultados da detecção local nos NEs são enviados para os outros, para que todos possam correlacionar a informação da detecção, diagnosticar quais são os NEs maliciosos e logicamente removê-los da rede, desligando todas as portas a eles ligadas.

PALAVRAS-CHAVE: Detecção Cooperativa de Intrusões, Detecção de Intrsões baseada em especificação, Carrier Ethernet, Spanning Tree Protocol, Topologia da Rede, Segurança.

Abstract

Current OSI model layer 2 network elements (NEs, e.g., bridges, switches) are complex hardware and software boxes, often running an operating system, service and administration software, that can be vulnerable to attacks, including to remote code execution inside them. The purpose of this thesis is to present an architecture to protect the Carrier Ethernet network infrastructure from attacks performed by malicious NEs against the link management protocol, Spanning Tree Protocol, and its variations.

This thesis proposes that NEs are equipped with an intrusion detection component. Each detector uses a specification-based intrusion detection mechanism in order to inspect the behaviour of other NEs through the analysis of the received messages. The correct behaviour of the NEs is crafted from the standard specification of the STP protocol. If there is a deviation between current and expected behaviour, then the NE is considered to be malicious. The specification is extended with temporal pattern annotations, in order to detect certain deviations from the protocol. The results of the local detection are then transmitted to the other NEs, in order to cooperatively establish a correlation between all the NEs, so that malicious NEs can be logically removed from the network (disconnecting the ports connected to them).

KEY WORDS: Cooperative Intrusion Detection, Specification-based In-

trusion Detection, Carrier Ethernet, Spanning Tree Protocol, Network Topology, Security.

Acknowledgments

This work has been possible thanks to the generous contribution of ideas, time, support and encouragement of many people.

First and foremost, my sincerest appreciation to my advisor Professor Miguel Pupo Correia (FCUL) who provided technical support, friendship, and inspiration for keeping me from taking myself seriously with his many advisories. I'd also like to thank to my supervisor Engineer João Redol (Nokia Siemens Networks Portugal, S.A.), for his generosity, faith and superb guidance, for his constant attention to details and his encouragement. This thesis would not exist without them.

My gratitude to Nokia Siemens Networks Portugal, S.A., for sponsoring totally the work, trusting me, and let me pursue my dreams. The work in this thesis was developed in the context of the N3TS project. Many people in the project contributed directly and indirectly to this thesis, with suggestions, discussions, etc. With the risk of missing some, they were: Eng. André Pimentel, Eng. Nuno Perpétua, Eng. Hugo Gomes, Eng. João Rebelo, Dr. Mauro Lemos and Dr. Tito Nobre.

Cheers to the rest of the exceptional research department at Nokia Siemens Networks Portugal, S.A. for incredible works they produce and for all the good time over the years. Thank you specially to Eng. Catarina Francisco, Master Rui Luís and Dr. Hugo Simões, for their availability

and thought-provoking discussions. With their help and positive reinforcement, many of my research projects may have never amounted to anything. Thanks also to my ex-colleagues at LaSIGE laboratory (FCUL) and many friends who gave me a constant support and encouragement through these years.

Finally, I must thank all my family. This thesis is specially dedicated to the memory of my grandfather who pushed me to continue doing the research work. My parents and the rest of my family were always a great support.

Lisboa, December 31, 2007

Pan Jieke

To all who never give up.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The telecommunications industry, meeting the needs of an increasingly global commerce environment, has contributed to better productivity and bridged communities globally in almost every industrial segment. The reality that current communication infrastructures are so efficient, is in no small part due to standards developed by several international organizations. The standards that keep current networks efficient also pave the way for next generation networks. However, while standards have continued to meet the end-users and industry needs, the increasing use of open interfaces and protocols, the multiplicity of new actors, the sheer diversity of applications and platforms, and implementations not always sufficiently tested, have increased the opportunity for malicious usage of networks. In recent years, a surge of denial of service attacks caused periods of unavailability in services and parts of the Internet, involving innumerable service providers and users, and often resulting in a major impact in terms of costs [Turner(editor), 2006]. The question then is how does one deploy an open communication network without letting it vulnerable to these attacks. The answer is complex but a first requirement is clearly the

protection of the communication infrastructure itself.

Layered network architectures, like the OSI and TCP/IP models, separate functionality in layers, where the lower layers provide services for the higher layers. These flexible models provide a form of separation of concerns, which allows the layers to be designed and implemented independently. However, from the security point of view, once a lower layer is compromised, the reliability of the higher layers can also be affected. Most existing network security paradigms and models are concerned with the layers from 3 to 7 of the OSI model, i.e., from the Network to the Application layers. In Internet terms, this means security is mostly concerned with protocols like IP, TCP, HTTP and SOAP, and issues like user authentication, data integrity and confidentiality. Less attention has been paid to the *network infrastructure*, i.e., to layer 1 and 2 protocols.

When layer 2 protocols are used on local trusted networks, their security is usually not critical. However, with the use of layer 2 protocols over wide area networks, the assumption of their security is problematic. As more and more broadband service providers deploy access networks based exclusively on layer 2 protocols, attacks focused on the data link layer become more feasible. Particularly, since home users will have access to the network, they could attempt to manipulate the protocols to disrupt service to other customers and the broadband service provider. Therefore, the data link layer could be an attractive target for attackers, being the lowest layer not requiring physical access to manipulate. Attacks against these two layers can "disable" the network, causing unavailability of the higher layer protocols, with a huge impact on a large-scale network, possibly involving a vast number of service providers and users.

1.1 Objectives

The purpose of this work is to present an architecture to protect the *Carrier Ethernet* network infrastructure using a *cooperative distributed intrusion detection and response system*. The motivation for this work is that this kind of network technology is currently being widely adopted around the world by Internet service providers and many other companies[1]. More specifically, the thesis is about the OSI Data Link Layer (layer 2) [ISO, 1994], its protocols and *network elements (NEs)*, i.e., *switches* and *bridges*.

Currently, NEs are complex hardware and software boxes, often running an operating system, service and administration software. Therefore, NEs can have vulnerabilities, which can be exploitable like the vulnerabilities that are hacked everyday in the Internet, e.g., in Web or email servers [Turner(editor), 2006]. For instance, several NEs from different manufactures were disclosed a lot of vulnerabilities:

- The Cisco Catalyst family routers run the Cisco IOS software. A heap buffer overflow vulnerability [Cisco, 2005a] that would allow remote execution of code in some versions of IOS has been recently reported.

- The HP Procurve 4000M is a common, managed switch, which provides low-cost and scalable Ethernet switching. There exists at least one vulnerability in the web administration interface that allows an attacker to reset a switch, allowing the attacker to arbitrarily and repeatedly deny access to all switched ports [HP, 2002].

- The Avaya P330 Stackable Switch [Avaya, 2005] was found to have a default password. An attacker could use this default password to

[1]The adoption of Carrier Ethernet is being pushed by the Metro Ethernet Forum. The site of the forum is at: http://www.metroethernetforum.org

gain remote access to such a switch.

In this thesis the interest focus on the problem of a layer 2 NE being controlled by a hacker, starting to behave maliciously and launching attacks against the network infrastructure. We consider the case of a network formed by a set of NEs, some of which are *malicious*. To deal with this problem, it is proposed that NEs are equipped with a component which provides *network intrusion detection*. Each of these detectors inspects the behavior of other NEs by inspecting the messages received from them and by cooperating with other detectors in other NEs to diagnose the attack and (logically) remove the malicious NE from the network.

1.2 Contributions

The contribution of this thesis is the design of the first distributed network intrusion detection system for carrier Ethernet. It focuses on the original link management protocol of the (switched) *Ethernet*, the *Spanning Tree Protocol (STP)*, and its variations, the *Rapid Spanning Tree Protocol (RSTP)* and the *Multiple Spanning Tree Protocol (MSTP)*, as a case study [Cisco, 1997, IEEE, 1998b, Cisco, 2006b, Cisco, 2006a, IEEE, 1998c].

The detection follows a recent approach dubbed *specification-based intrusion detection*, which relies on a specification of the protocol to detect deviations from it. However, this form of intrusion detection is normally used to detect deviations from sequences of messages or state transitions that should occur, while many attacks against STP have to do with certain patterns in terms of time behavior. Therefore, we have to extend the usual specification-based intrusion detection scheme with time pattern *annotations*, in order to detect all known attacks against xSTP (STP and its

variations).

The intrusion detection scheme works roughly as follows. The detectors in the NEs inspect the STP protocol messages received in the NE in real-time and without interfering with the operation of the network. Whenever a message is received in a NE, the detector checks the behavior of the NEs involved against their expected behavior. The correct behavior of the NEs is described following the STP protocol specification [IEEE, 1998b]. If there is a deviation between expected and actual behavior, the NE is suspected of being malicious and logically removed from the network.

The results of the local detection in the NEs are sent to the other NEs, so that all can correlate the detection information, diagnose which is (or are) the malicious NE, and then disconnected it. A management network currently used with carrier Ethernet networks by service providers (DCN) is used to provide extra integrity and authentication support in the communication of the detection information.

Besides xSTP, we might also have considered other attacks in malicious NEs, like discarding messages or forwarding them to the wrong port/link. However, these attacks are similar to those studied for malicious routers at layer 3, and there is already a considerable literature about how to deal with them, e.g., [Perlman, 1988, Mizrak et al., 2006, Lee et al., 2006]. Although these mechanisms were designed to be applied at a different OSI level, some of them can arguably be used at layer 2.

1.3 Structure of the thesis

The thesis is organized as follows. Chapter 2 gives some insight about carrier Ethernet networks, the STP protocol, the attacks against this protocol, and current protections available against these attacks. Chapter 3 describes the proposed cooperative intrusion detection architecture. Chapter 4 presents experimental results. Finally, Chapter 5 concludes the document and presents some future work.

1.4 Achievement

Until the thesis submission date, this thesis has achieved two scientific publications, one patent acceptance and one patent submission.

Publications:

- Pan Jieke, João Redol and Miguel Correia.
  "Specification-Based Intrusion Detection System for Carrier Ethernet".
  In 3rd International Conference on Web Information Systems and Technologies, Barcelona, Spain, March 2007

- Pan Jieke, João Redol and Miguel Correia.
  "Detecção Cooperativa de Intrusões em Redes Carrier Ethernet".
  In 3ª Conferência Nacional sobre Segurança Informática nas Organizações, Lisboa, Portugal, November 2007

Patent:

- Pan Jieke and João Redol.
  "Method for Protocol Behaviour Detection in Carrier Ethernet",
  Filed for patent office (level 4, 0-5), December 2007

- Pan Jieke and João Redol.
  "Integrated Network Security Architecture Monitoring System",
  Submitted for evaluation, September 2007

Chapter 2

Related work

2.1 Carrier Grade Ethernet

Networking today comprises a wide range of technologies. Different organizations have different requirements and the technologies are constantly evolving, requiring constant adaptation and reconfiguration of the systems. Therefore, it is important to have a reference model which defines rules for the network infrastructure to deal with all this complexity. The standard networking model is the *Open Systems Interconnection (OSI)* [ISO, 1994]. This model provides flexibility for network infrastructure construction and separation of functionality. Basically, it separates the network functionality in components called *layers*, where each layer is as independent of the others as possible.

There are seven layers in the OSI model. A lower layer provides services for higher layers without having to reveal how these services are implemented. This separation of concerns provides flexibility to the model, allowing changes in lower layers (mostly) without affecting the users and applications that rely in upper layers. This layering, however, causes that

unreliability in a lower layer might compromise all layers above it.

The second layer of the OSI model – the Data Link Layer – stays between the Network Layer (layer 3) and the Physical Layer (layer 1). This is the layer that transfers data between adjacent network nodes in a wide area network or between nodes on the same local area network segment, detecting and possibly correcting errors that may occur in the Physical Layer.

Ethernet is currently the dominant layer 2 networking technology in wired local area networks, for a set of reasons that are debatable but that surely include its low cost, simple management and its constant evolution in terms of bandwidth provided [Metcalfe and Boggs, 1988]. Because of the very competitive interface price per Mbps and high bandwidth supported, many carriers are installing Ethernet-based networks as an alternative to legacy ATM/SDH deployments, mainly in the access area. With the advent of the *Internet* and strongly growing data services, packet-switched networks, like IP/Ethernet networks, are better suited to satisfy the needs of the carriers' customers. Ethernet was not originally a *carrier-grade* technology, because it was designed for LAN environments. However there are currently several efforts in standardization organizations, like the IEEE, in order to adapt the Ethernet functionality to the needs of carriers.

2.1.1 The problem

System administrators and developers normally assume that the Data Link Layer is secure or, at least, are not concerned about its security. There are several reasons for this situation: layer 2 bridges/switches have no interface for human interaction, they are physically controlled by an orga-

nization, their protocols (layer 2 only) and software are reasonably simple when compared, for instance, with Web protocols and servers, and, more importantly, the typical Internet hacker is several layer 2 networks away from its target, so it needs layer 3 routing to reach the network. As Ethernet becomes an access network solution in the carrier class, more and more service providers deploy access to the network at layer 2, so attacks focused on the data link layer become more feasible. As one of the most commonly used *Ethernet* standard link management protocols, the *Spanning Tree Protocol* and its variations (RSTP and MSTP) are an obvious target for those attacks, which can compromise the security of all the layers above 2 and the network availability.

2.2 Spanning Tree Protocol Family

LAN bridges/switches were first conceived and developed during the mid-1980s as layer 2 switching devices supporting star-shaped networks, instead of the original *Ethernet* bus architecture. In order to truly supplant SONET/SDH with Ethernet in Metro networks, Ethernet must meet the demanding standards service providers expect and rely upon from circuit-based networks. These include high availability and fault tolerance in the event of network or equipment failure. Given that Ethernet services are most cost-effectively implemented on native Ethernet equipment, a variety of resiliency mechanisms have been introduced to Ethernet to enable the construction of such a network. The result is an Ethernet-based Metro network that is far more efficient and lower cost than SONET-based deployments. The key to achieving high availability with Ethernet in the Metro, however, is that the network is architected on a foundation of re-

siliency.

When switched Ethernet appeared, there was some concern about redundant connections between two or more switches, since they would form loops and would cause the replication of some of the messages flowing in the network. Therefore, the *Spanning Tree Protocol (STP)* was designed with the purpose of managing links, removing existing loops without human intervention (Figure 2.1) [IEEE, 1998b]. Currently, STP is mostly used for fault tolerance. Redundant connections are introduced in the network on purpose and, if a link (i.e., a cable or interface) fails, STP reconfigures the network to use a redundant link, thus ensuring the continuity of service.

A tree topology is always loop-free, since there is one and only one path from a leaf of the tree to any other of the leafs. The purpose of STP is to organize the network in such a tree topology without leaving any segment isolated, i.e., disconnected from the rest of the tree.

2.2.1 Root bridge

A tree has a root from which the remainder of the tree branches out. The NE in the root of the spanning tree is called *root bridge* in STP. The root bridge is the logical center of a network and there is only one root bridge in each instantiation of the tree, i.e., between two reconfigurations of the network performed by STP. Depending on the configuration, any bridge can be the root bridge. The root bridge may change over time if the topology changes, e.g., because a bridge is removed or added to the network.

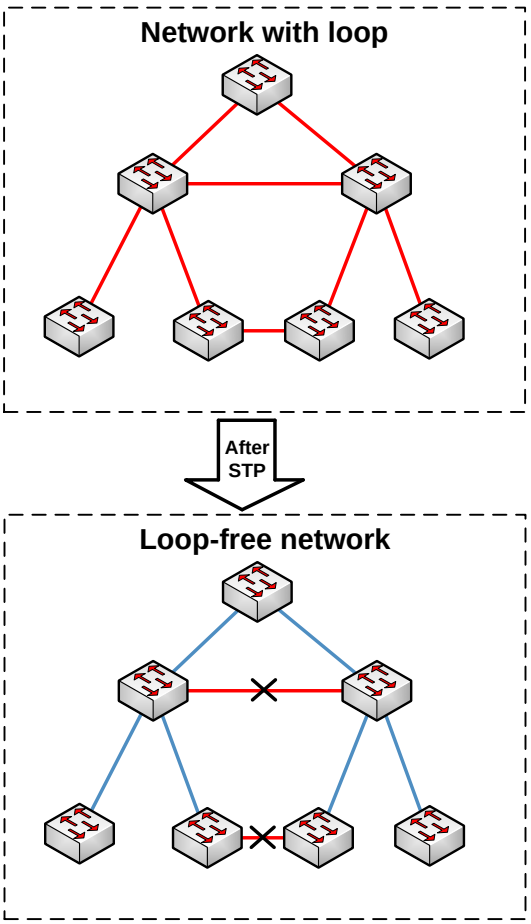

Figure 2.1: Spanning Tree Protocol

2.2.2 Designated bridge

Preventing loops in the network means to ensure that there is one and only one path from one bridge to another. A simple way to obtain this, is by ensuring that only one bridge – the *designated bridge* – is responsible for forwarding traffic from the direction of the root onto a given link. If there is only one active path from the root to a link, then by definition there are no loops in the topology. Each link has exactly one designated bridge, which is one of the two bridges directly connected to that link. The root

bridge is always the designated bridge for all links to which it is directly connected. Non-root bridges may be the designated bridges for no links, one link, or more than one link in the network. For instance, on the bottom rectangle of Figure 2.1, the root bridge (on the top) is the designated bridge for the 2 links to which it is connected; each of the 2 bridges below is the designated bridge for the 2 links below it; the 4 bridges on the bottom are not designated bridges for any link.

2.2.3 Designated Ports and Root Ports

For a given bridge, there are the following types of ports:

- *Designated Port:* is a port in the active topology used to forward traffic away from the Root Bridge onto the link for which this bridge is the Designated Bridge.
- *Root Port:* is a port in the active topology that provides connectivity from the Designated Bridge towards the Root Bridge.
- All other ports of a bridge will be inactive (disabled or blocked) in the steady-state, i.e., when there are no reconfiguration of the topology going on.

2.2.4 Bridge Identifiers and Port Identifiers

To configure, calculate and maintain the spanning tree, each bridge in the network has a unique identifier. This *bridge ID* (Figure 2.2) is a 64-bit value unique to each bridge. It is a concatenation of a 16-bit priority value (high weight 16 bits) and a globally unique 48-bit which is the NE's MAC address (low weight 48 bits).

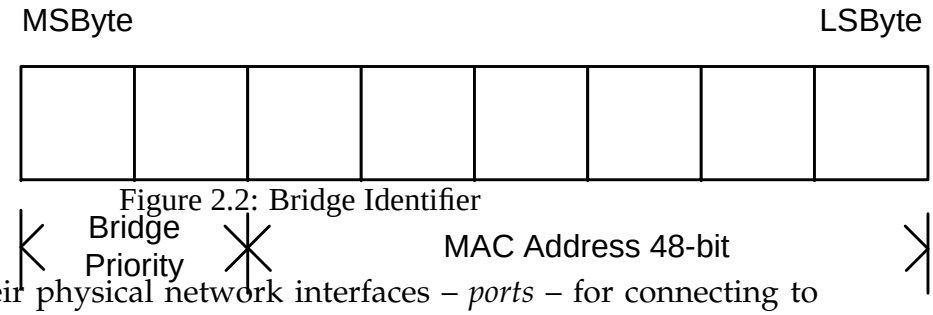

Figure 2.2: Bridge Identifier

NEs use their physical network interfaces – *ports* – for connecting to other NEs. Each port on a NE is assigned a *port identifier* (Figure 2.3), similar to the bridge identifier. A port identifier is a concatenation of a unique 8-bit port number with a configurable priority field. Port numbers are locally unique to the NE, and simply denote the number of the physical attachment on the NE. Each port connects to a link, which may connect to another LAN bridge/switch, a wide area connection, or a computer.

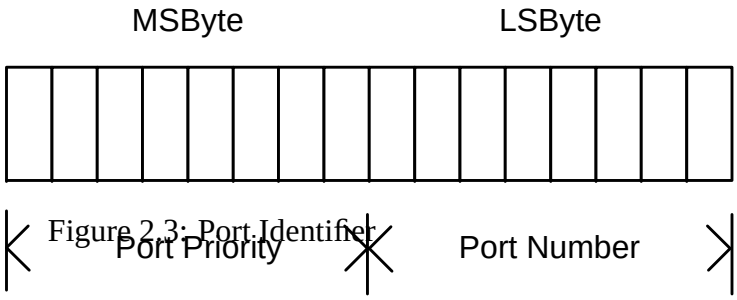

Figure 2.3: Port Identifier

2.2.5 Spanning Tree Calculation and Maintenance

STP attempts to configure the network such that every leaf is reachable from the root through the path/link with the lowest cost determined by

the data rate of each link. Therefore, the spanning tree topology for a given set of links and bridges is determined by the bridge IDs, the link costs, and the port identifiers associated with the bridges in the network. STP has to perform three operations:

- determine a root bridge, which is the bridge with lowest bridge ID;
- determine the designated bridge for each link;
- maintain the topology over time.

In practice, all of these operations are done in parallel, through the spanning tree algorithm operating identically and independently in each bridge.

2.2.5.1 Root bridge election

The election algorithm of root bridge is simple: the bridge with the numerically-lowest Bridge ID becomes the Root Bridge at a given time. A change in a root bridge will cause the spanning tree to reconfigure such that Root Bridge always has the lowest numbered Bridge ID.

The network administrator can control which bridge will be the default root, and the order in which other bridges will assume root responsibility, by manipulating the priority field in the Bridge ID. Since the priority field constitutes the most significant bits of the Bridge ID, it always overrides any effect of the remaining 48 bits for the identification.

2.2.5.2 Designated bridge election

Once there is a Root Bridge, it is necessary to identify, for each and every link in the network, a single bridge responsible for forwarding traffic

from the root to that link. This is the Designated Bridge for the link in question. By definition, the Root Bridge is the Designated Bridge for every link to which it attaches. The Designated Bridge for each other link will be the bridge that offers the lowest cost path back to the root. The path cost is simply the sum of the link costs over the path, with the link costs. Thus, the spanning tree will be such that the highest capacity links are always used, rather than diverting traffic needlessly through slower links.

It is possible that two bridges have the same path cost back to the root. In the event of such a tie, the bridge with the lowest Bridge ID will become the Designated Bridge. Similarly, it is possible that a Designated Bridge can have two ports on the same link. Only one of the ports can be the Designated Port for the link, the port with the lowest numbered Port ID will be chosen.

2.2.5.3 Bridge Protocol Data Units

STP operates on the principle that all Designated Bridges advertise their current understanding of the spanning tree and their internal state by emitting, on a regular basis through their Designated Ports, a configuration message. STP's configuration messages are called *Bridge Protocol Data Units (BPDUs)* (Figure 2.4). BPDUs are used to learn about the existence of other bridges and to obtain the information needed to calculate and maintain the spanning tree. There are four types of BPDUs definable by the TC field in the BPDU:

- Configuration BPDU
- Topology Change Notification BPDU
- Topology Change Notification Acknowledge BPDU

- Topology Change BPDU

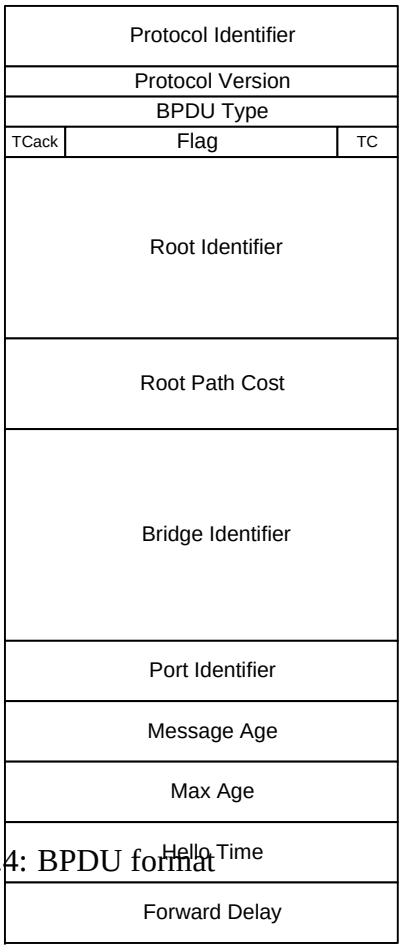

Figure 2.4: BPDU format

2.2.5.4 Network topology maintenance

When the spanning tree has converged to the proper topology, the regular emission of Configuration BPDU maintains the topology, keeping inactive bridges and ports from becoming active and creating undesirable loops. In the event of a link or bridge failure, the lack of regular Configuration BPDU from the now failed link or bridge may cause previously

inactive bridges and ports to become active in order to maintain maximum connectivity. The spanning tree topology will then reconverge to the best topology now available.

In normal (steady state) operation, the protocol operates as follows:

1. Once every period designated by *Hello Time* (typically 2 seconds), the Root Bridge transmits a Configuration BPDU. The message indicates that the sender is the Root Bridge (the Root ID will be the same as the Bridge ID in the BPDU) and that the path cost to the root bridge is zero (since it is being sent by the root itself).

2. All bridges sharing links with the Root Bridge receive the message and pass it to the STP entity within the bridge.

3. Designated Bridges use the information received from the Root Bridge to create a new Configuration BPDU, updating the values of the Bridge ID, path cost, Port ID and other fields appropriately, finally transmitting this message through each of its Designated Ports.

4. Similarly, bridges sharing the links with each of these Designated Bridges receive this second tier message. As a result, the Designated Bridges for the next tier in the tree will transmit new, modified messages through their own Designated Ports.

5. This process continues until there are no more Designated Bridges.

Once a spanning tree is determined and set in place, there is rarely any need to change the topology. Topology changes only occur if:

- A designated or root bridge fails
- A designated or root port fails

- An active link fails
- A bridge internal configuration changes (bridge priority, port priority change, etc)
- A new designated or root bridge is added
- A new link is added

When a bridge that is not the Root Bridge detects changes on the active topology, it transmits a Topology Change BPDU through its Root Port. This is repeated until the bridge receives an acknowledgment from the Designated Bridge for that link. That Designated Bridge similarly transmits a Topology Change Notification BPDU through its root port to the next designated bridge, until the message reaches the Root Bridge.

When the Root Bridge is informed of the change, it sets the Topology Change flag (TC) in all Configuration Messages transmitted for some time, so that all bridges become aware of the topology change.

2.2.6 RSTP and MSTP

In 1998, IEEE 802.1w introduced an evolution of the Spanning Tree Protocol, the Rapid Spanning Tree Protocol (RSTP), which provides faster spanning tree convergence after a topology change. Standard IEEE 802.1D-2004 now defines RSTP and obsoletes STP [Cisco, 2006b].

RSTP is a refinement of STP and therefore shares most of its basic operation characteristics. However there are some notable differences as summarized below:

- Faster aging of information: a bridge considers that it has lost connectivity to its direct neighboring root or designated bridge if it mis-

ses three consecutive BPDUs.

- Introduction of flags in the messages to describe fully port roles and port states.

- After a change is detected, unwanted source address information is purged from forwarding tables [1] without delay.

- Origination of Configuration Message BPDUs on a port by port basis, instead of transmission on Designated ports following reception of information from the Root.

The Multiple Spanning Tree Protocol (MSTP), originally defined in IEEE 802.1s and later merged into IEEE 802.1Q-2003, defines an extension to the RSTP protocol to further apply the loop-free mechanism in virtual LANs (VLANs) [Cisco, 2006a, IEEE, 1998c]. This "Per-VLAN" Multiple Spanning Tree Protocol configures a separate spanning tree for each VLAN group and blocks the links that are redundant within each Spanning Tree.

2.2.7 STP Attacks

STP is a low level network link management protocol. A few characteristics render it vulnerable to several types of attacks from hackers who have direct physical access to the network equipment. These attacks use the legitimate connection with network equipment to inject attacks in the network. Recall that we say that a *NE is malicious* if it is controlled by a hacker, thus can execute arbitrary attacks against the STP protocol.

[1] Every learning bridge keeps a mapping of ports and MAC addresses reachable through those ports in a data structure known as the forwarding table.

A careful analysis of the protocol and the literature on the matter, lead us to conclude that the attacks possible against STP are essentially the follow [Marro, 2003, Artemjev and Myasnyankin, 2003]:

- ID changing attacks

    1. Priority value changing

       Typically, the election of entities who participate in STP (the root bridge or the designated bridge) is based on the bridge IDs, which are composed by the bridge priority value and MAC address. A malicious NE can force its election as Root or designated bridge by modifying (reducing) its priority value that has the highest weight in the bridge ID, so that is becomes the lowest bridge ID.

    2. MAC spoofing

       Although MAC spoofing does not permit a malicious NE to force its election as root since the priority value has low weight in the bridge ID, this attack can cause (undesirable) topology changes.

- Silent attacks

    – As STP is a self-managed protocol, when the network topology is stable every NE continues sending Configuration BPDUs to each other to indicate they are alive. A malicious NE can omit sending these BPDUs to cause an undesirable network reconfiguration. When executed repeatedly, this attack can impair the availability of the network.

- Faked failure attacks

  - In STP, the failure of a NE is detected by its neighbors. When a neighbor NE detects the failure, it sends a Topology Change BPDU to indicate this event. A malicious NE might generate fake Topology Change BPDUs to cause network reconfigurations.

- BPDU flooding attacks

  1. Flood of Topology Change BPDUs

     A flood of bogus Topology Change BPDUs from different addresses is injected in the network, which can cause undesirable network reconfigurations.

  2. Flood of Topology Change Notification BPDUs

     A flood of bogus Topology Change Notification BPDUs from different addresses are injected in the network, causing undesirable network reconfigurations.

  3. Flood of Configuration BPDUs claiming root role

     Configuration BPDU with invalid bridge identifiers, claim to be new NEs in the topology, possibly being elected as the new root.

- Invalid BPDU

  - This attack consists in sending a BPDU that is malformed or that could not be sent in the current state of the NE. This attack can be, for instance, an attempt to find a vulnerability in the target NE, which would allow remote code execution or would crash the NE.

Notice that with exception of the last attack, all these are attacks against the availability of the network infrastructure, i.e., denial of service attacks.

The last attack, *Invalid BPDU*, can be used with other purposes, like doing a buffer overflow attack against the NE software, in case there is corresponding vulnerability.

2.2.8 STP attacks prevention

Several mechanisms can be used to mitigate attacks against STP. Marro proposes the extension of STP with message authentication [Marro, 2003]. He assumes that the attacks are generated by clients (computers, non-NEs) simulating they are legitimate NEs, so it does not consider the case of malicious NEs, capable of generating valid message authentication codes. The idea consists in the addition to each BPDU of a *message authentication code (MAC)* [Menezes et al., 1997], obtained using symmetric encryption keys shared by the NEs. When a legitimate bridge sends a BPDU, it will take a valid MAC and, therefore be accepted by other bridges. If the BPDU is sent by a NE that does not have the key(s), it will take an invalid MAC so it will be refused by other NEs. In this work we assume NEs can be malicious, so they might generate attacks with valid message authentication codes if that mechanism was used. Therefore that mechanism is not enough to deal with some of the attacks considered in this thesis.

Cisco has proposed two features in their bridges/switches called *BPDU Guard* [Cisco, 2005b] and *ROOT Guard* [Cisco, 2005c] that block the reception of BPDUs from non-STP ports, thus preventing attacks from clients. BPDU Guard and ROOT Guard are similar, but their impact is different. BPDU Guard simply disables the port upon a BPDU reception on that port, and ROOT Guard allows the device to participate in STP as long as the device does not try to become the root. Once more, these mechanisms are not enough to deal with some of the attacks considered in the thesis: attacks

coming from malicious NEs.

The standard 802.1x protocol performs the authentication of users in a IEEE 802 LAN, e.g., for dial-up connections [IEEE, 1998a]. A user needs to make a request to a gateway that controls network access and forwards requests to an authentication server, e.g., a RADIUS server. It can prevent attacks from non-authenticated client computers simulating they are NEs, like Marro's scheme.

All of these solutions consider only that the clients connected to the NEs can be malicious and assume NEs are always correct. On the other hand, they have still many limitations. Marro's approach has the limitation of being difficult to implement in real systems, since it requires modifying the STP protocol itself and putting a shared cryptographic key in all NEs. The key distribution is non-trivial, especially because layer 2 NEs are usually deployed without further configuration. Cisco's solutions need an initial static configuration, complicating the network administration. 802.1x has a potential high cost in terms of resources and administrative overhead by using certificate-based authentication.

2.3 Intrusion Detection and STP

Security mechanisms like 802.1x access control [IEEE, 1998a] and firewalls have a fundamental role in network security but can not prevent all possible attacks against a network. For instance, legitimate users are able to perform attacks even if those mechanisms are properly used. Intrusion detection is a second line of defense [Denning and Neumann, 1985, Porras et al., 1998, Kruegel et al., 2005]. The idea is to gather computer or network data and analyze it looking for attacks or intrusions. Classically,

an Intrusion Detection System (IDS) attempts to detect the presence or the likelihood of all kinds intrusions. Intrusion detection can be performed in real-time or off-line.

IDSs come in a variety of "flavors" and approach the goal of detecting suspicious traffic in different ways. There are network based (NIDS) and host based (HIDS) intrusion detection systems. There are IDSs that detect based on looking for specific signatures of known threats, similarly to the way anti-virus software typically detects and protects against malware and there are IDSs that do detection based on comparing traffic patterns against a baseline and looking for anomalies. There are IDSs that simply monitor and alert and there are IDSs that perform an action or actions in response to a detected threat.

From the point of view of where they are placed, IDSs can be:

- NIDS - Network Intrusion Detection Systems are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network. Ideally they would scan all inbound and outbound traffic, however doing so might create a bottleneck that would impair the overall speed of the network.

- HIDS - Host Intrusion Detection Systems are run on individual hosts or devices on the network. A HIDS monitors the inbound and outbound packets from the device only and will alert the user or administrator of suspicious activity is detected. A HIDS can also look for malicious activity in the host itself, e.g., in the syslog file.

From the point of view of their detection strategy, IDSs are classically classified in two classes:

- *Misuse detection* [Ilgun et al., 1995, Lindqvist and Porras, 1999] uses attack signatures to detect known attacks. If properly configured, a misuse intrusion detection system generates a low rate of false alarms but, on the other hand, does not recognize attacks that are unknown when the signatures were generated (or, more precisely, attacks/intrusions not described by the signatures).

- *Anomaly detection* is based on patterns of normal behavior, created (typically) using an automated training process [Kruegel et al., 2005]. Deviations from this "normal behavior" are considered to be caused by intrusions, so an intrusion can be detected without any previous knowledge. Although anomaly detection overcomes misuse detection's weaknesses, a high rate of false alarms is hampering of using this type of detection mechanism.

2.3.1 Specification-based intrusion detection

Specification-based intrusion detection is a hybrid form of intrusion detection that tries to combine the strengths of the previous two techniques: misuse detection and anomaly detection. Instead of relying on machine learning techniques, like anomaly detection, the specification-based approach [Balepin et al., 2003, Sekar et al., 2002, Uppuluri and Sekar, 2001] uses manually developed specifications that craft legitimate system/protocol behaviors. A deviation from the specification is considered to be the symptom of an intrusion. This solution has been applied to a few protocols and applications [Tseng et al., 2003, Orset et al., 2005], but never to Ethernet protocols or STP, to the best of our knowledge.

In this approach, manually developed specifications are used to char-

acterize legitimate protocol behaviour. As this method is based on legitimate behaviour, it does not generate false alarms when usual protocol behaviour is encountered. Thus, its false positive rate can be comparable to that of misuse detection. Since it detects attacks as deviation from legitimate behaviour, it has the potential to detect previously unknown attacks.

[Tseng et al., 2003, Orset et al., 2005] proposed specification-based intrusion detection systems that can detect attacks on *mobile ad-hoc networks (MANET)*. The first paper introduced a specification that traces request-reply flows to detect attacks on the AODV routing protocol, and the second one proposed a specification of backward checking for the OLSR protocol. The major differences comparing to the thesis proposal, are the environment and the protocols studied. On a MANET, the most important problem is blocking the entry of new malicious nodes in the network infrastructure, by using a forward address table to decide if an unknown node is trusted. The specification is developed based on the forward address tables mechanism. DoS attacks were not considered in these works. The problem in this thesis is more complicated since as a legitimate NE is able to behave incorrectly, simply checking the NE's address is not enough to mitigate the threats.

Chapter 3

Cooperative Intrusion Detection System

The contents of this chapter were omitted due to the confidentiality requirements. A brief summary of its contents is the following.

This thesis proposes a distributed intrusion detection system for *Carrier Ethernet* networks. The proposal involves extending each NE with a network intrusion detection software component. It is assumed that a subset (unknown) of the NEs of a network can be compromised by hackers and launch attacks against the network infrastructure. We say those NEs are *malicious*, while the rest are *correct*. To make the assumption weaker, if a NE is malicious, its network intrusion detection component can be compromised too, and give wrong information about detections. Therefore, a subset of the NEs may not follow the correct specification of STP, and this incorrect behavior has to be detected by the correct NEs/detectors.

This thesis uses *specification-based intrusion detection*, which detect attacks as deviations from a norm. By specifying correct behavior, any other behaviors will be classified as anomalous. The detector performs network

intrusion detection in run-time. The specification of the STP protocol is modeled using a *state machine*. The states of this machine are the states of the protocol, and state transitions are caused by the reception of BPDUs or expiration of timeouts.

The specification which represents the correct functionality of the protocol was manually developed. As the specification-based intrusion detection itself did not lead well with certain type of attacks, especially attacks use timeout, therefore the mechanism was extended with *annotations* in order to improve the detection of these particular cases.

A distributed correlation mechanism was also proposed in order to detect cooperatively malicious NEs. The correlation information is exchanged between NEs and correlation done by each NE. To enhance the information integrity, the existing DCN and Network Manager subsystem were used for the authentication process.

Chapter 4

Evaluation

4.1 Implementation

The implementation was not done with real bridges/switches but using an emulator. The reasons against using real equipment were essentially two: (1) using an emulator it is much simpler to test arbitrarily complex networks, with as many NEs as needed; (2) although the insertion of a detector inside a NE is quite feasible for its manufacturer, it is quite challenging for end-users.

The emulator used was the *RSTP simulator* [Rozin, 2002]. Although it is called "simulator", it is really an emulator since it has components (processes) that emulate a NEs, instead of using mathematical models of the network and its components. The reason why we used an emulator of RSTP, instead of an emulator of STP was that RSTP is the current standard, the most used nowadays and its protocol specification does not differ much from STP's. Therefore, it proves the scalability of this solution and its applicability of STP protocol's variations.

The RSTP simulator is a full implementation of 802.1s done in C lan-

guage, as a set of libraries and APIs. There are two types of processes: *bridge* and *mngr*. The first is an emulation of a RSTP bridge and the second is the environment, i.e., it allows to manage the topology, the communication between the NEs, and allows sending messages to the network.

The two programs use a few libraries. The *bridge* process is mostly based on *librstp.a* that contains the implementation of RSTP. The library *libuid.a* contains functions to process BPDUs, e.g., to extract/insert a BPDU from/into a network message. The library *libcli.a* provides commands to manage the network. For instance, it provides commands like *link* and *disconnect* which allow the user to instantiate the network. There are also tools to trace state machine transitions and to get information about network modifications. Both programs show timestamps to allow the user to understand when the commands were executed and when events have taken place.

Each NE has locally a representation of the STP protocol specification, which was manually insert into the original code of the RSTP simulator. Each NE keeps for each of its neighbours information about its current state (state name). Once it receives a message and if it is correct then it updates its state. The time pattern values $Rmax_e$ are configurable parameters of the program.

For the correlation phase, the *mngr*'s source code and the communication channel with *bridges* were modified by adding code for the DCN and the authentication mechanism. The detection information exchanged between NEs has the following format [suspected NE id, last state NE deviated] and the SHA-1 hash function [NIST, 1994] was used to generate ACK information. Additionally, the correlation period is also a configurable parameter. This period defines the duration for which a NE waits

for ACKs from other NEs from the Network Manager and Authenticator.

In summary the configurable parameters are:

- $Rmax_{timeout}$ – timeout in the transition between the state Wait_for_-CONF_BPDU and the state Wait_TCNA_BPDU
- $Rmax_{tcn}$ – timeout for the send_TCN_BPDU events in state Wait_-for_TCNA_BPDU
- $Rmax_{id}$ – timeout for id changing of a NE
- *CooperativePeriod* – timeout for the period of a NE waiting for ACKs from Network Manager and Authenticator

The source code of the emulator was modified to allow the injection of all the attacks described in Section 2.2.7.

The emulation can be executed in one or more computers. Each NE instance is executed as a single process of the operating system and the processes communicated using UDP/IP. In the experiments all the processes were run in the same machine. The test machine was a laptop with a 2.1Ghz Centrino processor and Suse Linux 10.2.

4.2 Emulation of an Attack

This section presents an example of attack detection using the emulation scenario. The scenario has only 3 NEs, for the screenshots to be as simple and understandable as possible. However we still had to clean up the pictures and add some comments for readability.

Figure 4.1 (top) presents a screenshot of the *mngr* process. When it is executed, it becomes aware of the 3 NEs that are running, which are

```
17:29:16 Mngr > Bridge B4323 hello :)
Bridge B4324 hello :)
Bridge B4325 hello :)

17:30:13 Mngr > link B4323 1 B4324 2
connect B4323 port p01 to B4324 port p02
17:30:24 Mngr > link B4324 3 B4325 4
connect B4324 port p03 to B4325 port p04
17:30:51 Mngr > link B4323 5 B4325 6
connect B4323 port p05 to B4325 port p06
Sorry, p06 invalid
17:31:06 Mngr > link B4323 3 B4325 5
connect B4323 port p03 to B4325 port p05
```

```
Before attack
17:31:32 B4323 > show bridge
Bridge:           B4323                  State:enabled
BridgeId:         8000-00e310000001      Bridge Proirity: 32768
(0x8000)
Designated Root:  8000-00e310000001
Root Port:        none
Time Since Topology Change: 9
Max Age:          20    Bridge Max Age:        20
Hello Time:        2    Bridge Hello Time:     2
Forward Delay:    15    Bridge Forward Delay: 15
Hold Time:         3

Start attack
17:31:36 B4323 > sleep
```

Figure 4.1: *mngr* (top) and malicious bridge (bottom)

internally numbered B4323, B4324 and B4325 (no relation with the bridge ID). Then we executed three *link* commands to interconnect the 3 pairs of ports of the NEs, forming a fully connected network with a loop.

On the bottom of the figure is a screenshot of the NE selected to be malicious (*bridge* process). The command *show bridge* gives some information about the NE. This is the NE with the lowest bridge ID so it is elected the root bridge by RSTP. However, sometime later, at instant 17:31:36 the *sleep* command is executed to make a silent attack. After that instant the NE does not execute RSTP or forwards messages.

Figure 4.2 shows screenshots of the other two NEs, which are not ma-

```
Before the attack from the NE B4323
17:31:39 B4324 > show bridge
Bridge:          B4324                State:enabled
BridgeId:        8000-00e410000001    Bridge Proirity:
32768 (0x8000)
Designated Root: 8000-00e310000001
Root Port:       8002 (p02), Root Cost:     20
Time Since Topology Change: 10
Max Age:         20   Bridge Max Age:       20
Hello Time:       2   Bridge Hello Time:    2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3

During the attack from the NE B4323
17:31:41 B4324 > 17:33:24:
brige B4324 became root
17:33:24:
brige B4324 new root port: p03
17:33:24:sttrans (B4324-p02): FORWARDING=>DISCARDING
17:33:25:
brige B4324 became root

show bridge
Bridge:          B4324                State:enabled
BridgeId:        8000-00e410000001    Bridge Proirity:
32768 (0x8000)
Designated Root: 8000-00e410000001
Root Port:       none
Time Since Topology Change: 5
Max Age:         20   Bridge Max Age:       20
Hello Time:       2   Bridge Hello Time:    2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3
```

```
Before the attack from the NE B4323
17:31:44 B4325 > show bridge
Bridge:          B4325                State:enabled
BridgeId:        8000-00e510000001    Bridge Proirity:
32768 (0x8000)
Designated Root: 8000-00e310000001
Root Port:       8005 (p05), Root Cost:     20000
Time Since Topology Change: 30
Max Age:         20   Bridge Max Age:       20
Hello Time:       2   Bridge Hello Time:    2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3

During the attack from the NE B4323
17:31:56 B4325 > 17:33:24:sttrans (B4325-p04):
DISCARDING=>LEARNING
17:33:24:sttrans (B4325-p04): LEARNING=>FORWARDING
17:33:25:
brige B4325 became root
17:33:25:
brige B4325 new root port: p04
show bridge
Bridge:          B4325                State:enabled
BridgeId:        8000-00e510000001    Bridge Proirity:
32768 (0x8000)
Designated Root: 8000-00e410000001
Root Port:       8004 (p04), Root Cost:     20000
Time Since Topology Change: 4
Max Age:         20   Bridge Max Age:       20
Hello Time:       2   Bridge Hello Time:    2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3
```

Figure 4.2: Correct bridges

licious. On these two NEs, only a couple of *show bridge* commands were executed. The timeout for the NEs to send a Configuration BPDU is by default 2s, which is the value recommended by the standard. The malicious NE went silent at 17:31:36, so approximately 2s later (17:33:24) both correct NEs detected it did not send that BPDU, both decide they are the root bridge, exchange some BPDUs, and end up agreeing that the root is NE B4323 (the other writes that its port p04 is connected to the root).

When the timeout expires, the intrusion detectors in both NEs also suspect that the malicious NE is indeed malicious. However the attack at this stage is indistinguishable from a real failure of the NE, like a crash or power disconnection. Therefore, the two correct NEs do nothing about it. However, the malicious NE goes on running again and becoming the root, then becoming silent again repeatedly, causing frequent reconfigurations of the network (this is not displayed in the screenshots, though). Recall that the timeout transition is annotated with a maximum number of repetitions $Rmax_{timeout}$. Therefore, after $Rmax_{timeout}$ repetitions of this cycle both correct NEs decide that the malicious NE is indeed malicious, exchange information about this sending ACKs to the authenticator, disconnect the malicious NE permanently.

4.3 Severity of Attacks

This section evaluates the severity of the attacks using the emulation environment. The two attack injection scenarios, now with more NEs, are shown in Figures 4.3. In the scenario 1, there is only one malicious NE, in the middle of the topology, and in the scenario 2 there are 5 malicious NEs.

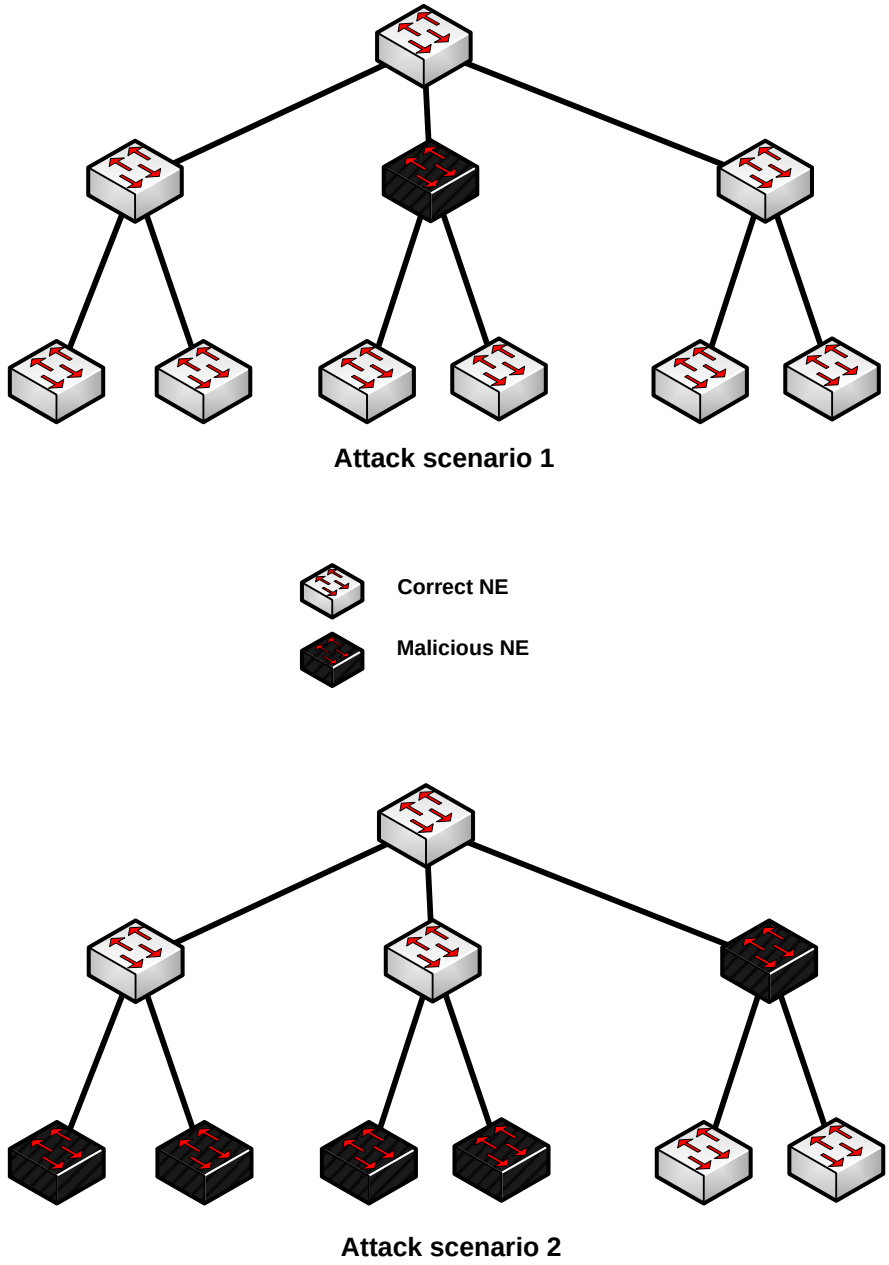

Figure 4.3: Attack Scenarios

Attack	Result	Severity	Effects
ID changing attacks	Successful	CT	network completely unavailable
Silent attacks	Successful	CT	network completely unavailable
Faked failure attacks	Successful	CR	partial network reconfiguration
BPDU flooding attacks	Successful	CR	small amount of traffic flooding
Invalid BPDU	Unsuccessful	MN	no effect

Table 4.1: Evaluation of the attacks in scenario 1 (1 malicious NE)

All the identified attacks were tested in the two scenarios. Tables 4.1 and 4.2 show an evaluation of the severity of these attacks against RSTP. The attacks are categorized into the five ISO categories [ISO, 2002]: CT (catastrophic: $>$0.95), CR (critical: $>$0.75), MG (marginal: $>$0.5) and MN (minor: $>$0.25). The highest value represents the highest severity or impact. The values were given manually by us, watching the consequences to the network of each attack.

The experiments allowed us to conclude that the ID changing and the silent attacks are the two most dangerous threats to the protocol. To achieve a catastrophic denial of service in the network, a malicious NE has to be close to the root of the tree in the topology. However, if the attacker is far from the root, it can first do an ID changing attack to become the root bridge, or at least a designated bridge to some link or set of links. After being the root bridge or a designated bridge, other attacks become easier. At that stage, silent attacks are the most critical, since even a single malicious NE –scenario 1– can force the reconfiguration of the tree periodically, pre-

venting the communication altogether or at least delaying it with constant needs for retransmissions.

In the case of faked failure attacks and BPDU flooding attacks, the effects were not so disruptive. These attack are much more critical when performed by a set of malicious NEs (scenario 2). This is the reason why they are marked CT in Table 4.2 but only CR in Table 4.1.

Interestingly, in the both scenarios the Invalid BPDU attack failed because the NEs discarded the BPDUs with invalid fields. This was true for the NE provided with the emulator, and for the limited number of malformed messages tested, but it is not necessarily true for all NEs currently available or with all possible malformed messages.

Attack	**Result**	**Severity**	**Effects**
ID changing attacks	Successful	CT	network completely unavailable
Silent attacks	Successful	CT	network completely unavailable
Faked failure attacks	Successful	CT	frequent network reconfiguration
BPDU flooding attacks	Successful	CT	network completely unavailable
Invalid BPDU	Unsuccessful	MN	no effect

Table 4.2: Evaluation of the attacks in scenario 2 (5 malicious NE)

4.4 Intrusion Detection Evaluation

Besides the previous experiments, an evaluation of the intrusion detection was also performed. As mentioned in Section 4.1, the code of the

two programs *mngr* and *bridge* was modified to include not only the local detection functionality but also the correlation mechanism, emulating the DCN and the authentication mechanism specified in the thesis.

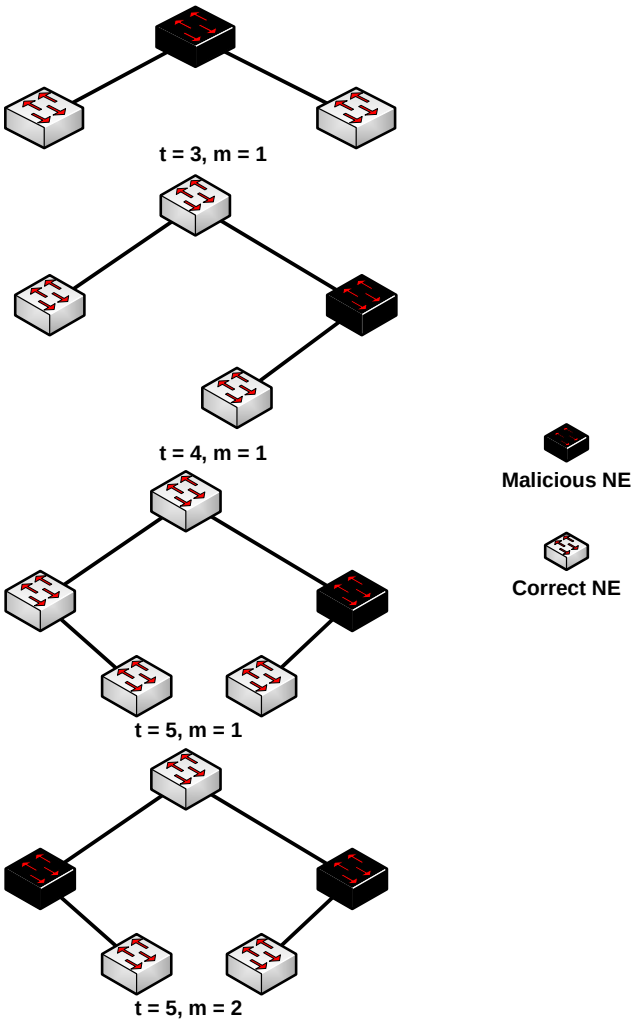

Figure 4.4: IDS Evaluation Scenarios

The evaluation of IDSs is known to be a complex problem, since the number of possible attacks is immense and the performance of each IDS is much affected by too many parameters, like the throughput and the particular variation of some attacks [Lippmann et al., 2000]. However, the

IDS scheme proposed in this thesis is different because the number of possible attacks is small and known (see Section 2.2.7). Therefore, we used the emulation environment to emulate all these attacks and all were effectively detected (no false negatives). There were also no false positives, i.e., nothing detected to be an attack which was not really an attack. There are some cases in which accidental faults like the intermittent failure and recovery of a NE may be mistakingly detected as an attack, but this is not particularly serious since it is indeed a problem that has to be corrected, independently of its origin not being malicious.

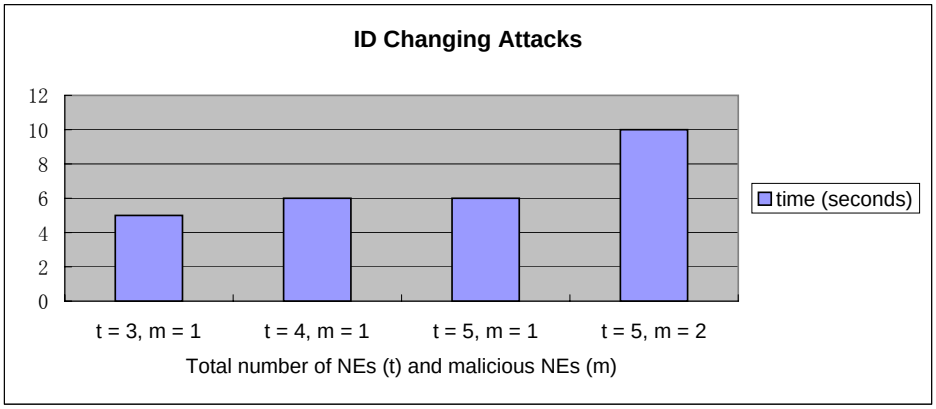

Figure 4.5: Evaluation of the IDS with ID Changing Attacks

After this evaluation in terms of ability to detect attacks, several experiments were made to evaluate the performance of the IDS implemented. Here the results of the evaluation for the two most critical attacks – ID Changing Attack and Silent Attack – are presented.

Several evaluation scenarios were considered (see Figure 4.4). These scenarios were defined in terms of two parameters: t, the total number of NEs in the network, and m, the number of malicious NEs. The *yy* axis indicates the average time for a NE to detect malicious activity, and the *xx* axis gives the different evaluation scenarios. All the NEs tested were run in the

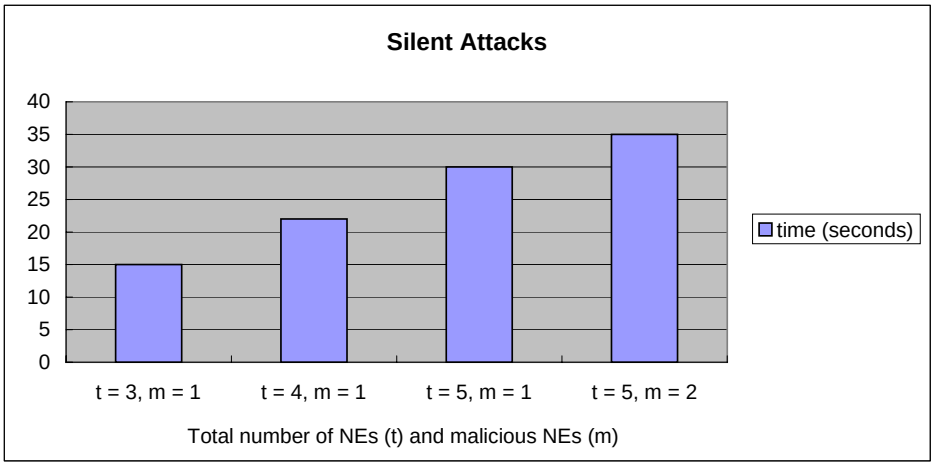

Figure 4.6: Evaluation of the IDS with Silent Attacks

same machine, although the simulator also allows the execution in a LAN. The timeouts were set to 5 seconds, the *CooperativePeriod* parameter was set to 10 seconds, and the parameters $Rmax_e$ were set to 2. Each evaluation process began with a malicious NE starting an attack and ended when a malicious NE was identified. Each NE obtains the evaluation time locally and the time values presented in the graphics are average values of all the measurement obtained by the NEs.

The results of the experiments are shown in Figures 4.6 and 4.5. Some conclusions of the experiments results are the following:

- The detection of ID Changing Attacks was faster than the detection of Silent Attacks. The main reason for this result is that detecting ID Changing Attacks does not require cooperative intrusion detection: each NE can detect locally the attack by using the number of repetition of identification of a NE (configurable parameter, 2 in this case). For detecting Silent Attacks, the cooperative intrusion detection scheme has to wait for ACKs coming from the DCN, and the *CooperativePeriod* parameter was set to 10 seconds, so the times are

larger than this.

- The complexity of network topology has a strong influence on the cooperative intrusion detection stage.
- The position of the malicious NE on the network also has influence on the performance.

Chapter 5

Conclusion

5.1 Conclusion

The thesis presents the design of a novel system for protecting Carrier Ethernet networks from attacks performed by their own NEs. Each NE is equipped with a specification-based intrusion detection system, which locally detects the malicious behaviour of other NEs against STP and its variations. The results from the local detection are later exchanged and correlated between NEs. The correlation is done by each NE locally, permiting the identification of malicious NEs.

The specification which represents the correct operation of the protocol was manually developed based on the standard. After a preliminary study of the attacks and the STP protocol characteristics, it was concluded that the specification-based intrusion detection itself did not deal well with certain type of events (e.g., timeout), therefore the mechanism was extended with *annotations* in order to detect all known attacks against STP.

In the correlation stage, a distributed correlation mechanism was proposed. Instead of the single point of decision, correlation information is

exchanged between NEs and correlation is done by each NE. To enhance the information integrity, the existing DCN and Network Manager subsystems were used. These two components are not an addition to the current Carrier Ethernet architecture, since they are often deployed.

The solution was implemented and tested using an emulator of RSTP. The emulator was modified to allow the injection of attacks on the network and to provide the proposed intrusion detection mechanism. Several evaluations were done, including attack injection tests, and the solution performance evaluation. All analysed attacks were successfully injected in the network, but a subset of them were more critical independently of the network topology. In relation to the solution performance evaluation, the results showed that it did not bring a great impact in terms of network bandwidth consumption and can be done fast enough for practical purposes.

5.2 Future Work

There is a set of attacks that can be performed by malicious NEs that are very similar to those that can be done by malicious routers. Examples include forwarding messages to the wrong link or discarding messages. As future work, we intend to do research on how to deal with these attacks. They were not considered yet because the solutions might be similar to those used to deal with malicious routers, an area about which there is a large literature, since the seminal work by Perlman [Perlman, 1988].

The implementation of the present solution was done using a simulator. It will be interesting to test the system in a real environment, which means inserting the code in existing NEs and evaluate in real-time the per-

formance and impact for the network.

The same detection mechanism can be also adapted for other layer protocols. Until now, the solution only accepts a manually crafted protocol specification. A future work can be an automation of the protocol specification development process and to allow more than one specification in the same detector.

Bibliography

[Artemjev and Myasnyankin, 2003] Artemjev, O. K. and Myasnyankin, V. V. (2003). Fun with the Spanning Tree Protocol. *Phrack*, 11(61).

[Avaya, 2005] Avaya (2005). *Avaya P330 Stackable Switch found with default password*. Avaya Inc.

[Balepin et al., 2003] Balepin, I., Maltsev, S., Rowe, J., and Levitt, K. N. (2003). Using specification-based intrusion detection for automated response. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection*, pages 136–154.

[Cisco, 1997] Cisco (1997). *Using VlanDirector, Appendix C - Understanding Spanning-Tree Protocol*. Cisco Systems Inc.

[Cisco, 2005a] Cisco (2005a). *Cisco Security Advisory: IOS Heap-based Overflow Vulnerability in System Timers*. Cisco Systems Inc. Document ID 68064, Revision 1.2.

[Cisco, 2005b] Cisco (2005b). *Spanning Tree PortFast BPDU Guard Enhancement*. Cisco Systems Inc. Document ID 10586.

[Cisco, 2005c] Cisco (2005c). *Spanning Tree Protocol Root Guard Enhancement*. Cisco Systems Inc. Document ID 10588.

[Cisco, 2006a] Cisco (2006a). *Understanding Multiple Spanning-Tree Protocol (802.1s)*. Cisco Systems Inc. Document ID 24248.

[Cisco, 2006b] Cisco (2006b). *Understanding Rapid Spanning-Tree Protocol (802.1w)*. Cisco Systems Inc. Document ID 24062.

[Denning and Neumann, 1985] Denning, D. E. and Neumann, P. G. (1985). Requirements and model for IDES - a real-time intrusion detection expert system. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA.

[HP, 2002] HP (2002). *HP Procurve 4000M Stacked Switch HTTP Reset Vulnerability*. Hewlett-Packard Co.

[IEEE, 1998a] IEEE (1998a). *802.1X - Port Based Network Access Control*.

[IEEE, 1998b] IEEE (1998b). *ANSI/IEEE 802.1D-2004 standard - Part 3: Media Access Control (MAC) Bridges*.

[IEEE, 1998c] IEEE (1998c). *ANSI/IEEE 802.1Q-2003 standard - Part 3: Media Access Control (MAC) Bridges*.

[Ilgun et al., 1995] Ilgun, K., Kemmerer, R. A., and Porras, P. A. (1995). State transition analysis: A rule-based intrusion detection approach. *Software Engineering*, 21(3):181–199.

[ISO, 1994] ISO (1994). *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*.

[ISO, 2002] ISO (2002). *Risk management vocabulary guidelines for use in standards*. ISO Copyright Office.

[Kruegel et al., 2005] Kruegel, C., Valeur, F., and Vigna, G. (2005). *Intrusion Detection and Correlation: Challenges and Solutions*, volume 14 of *Advances in Information Security*. Springer-Verlag.

[Lee et al., 2006] Lee, S., Wong, T., and Kim, H. S. (2006). Secure split assignment trajectory sampling: A malicious router detection system. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks (DSN'06)*, pages 333–342.

[Lindqvist and Porras, 1999] Lindqvist, U. and Porras, P. A. (1999). Detecting computer and network misuse through the production-based expert system toolset (p-BEST). In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 146–161.

[Lippmann et al., 2000] Lippmann, R., Haines, J., Fried, D., Korba, J., and Das, K. (2000). Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. In Debar, H., Mé, L., and Wu, S. F., editors, *Recent Advances in Intrusion Detection - Third International Workshop*, volume 1907, pages 162–182.

[Marro, 2003] Marro, G. M. (2003). Attacks at the data link layer. Master's thesis, University of California.

[Menezes et al., 1997] Menezes, A. J., Oorschot, P. C. V., and Vanstone, S. A. (1997). *Handbook of Applied Cryptography*. CRC Press.

[Metcalfe and Boggs, 1988] Metcalfe, R. M. and Boggs, D. R. (1988). Ethernet: distributed packet switching for local computer networks. In *Innovations in Internetworking*, pages 25–34. Artech House.

[Mizrak et al., 2006] Mizrak, A. T., Cheng, Y.-C., Marzullo, K., and Savage, S. (2006). Detecting and isolating malicious routers. *IEEE Transactions on Dependable and Secure Computing*, 3(3).

[NIST, 1994] NIST (1994). Announcement of weakness in the secure hash standard.

[Orset et al., 2005] Orset, J.-M., Alcalde, B., and Cavalli, A. R. (2005). An EFSM-based intrusion detection system for ad hoc networks. In *Proceedings of the 3rd International Symposium on Automated Technology for Verification and Analysis*, pages 400–413.

[Perlman, 1988] Perlman, R. (1988). *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Massachussets Institute of Technology.

[Porras et al., 1998] Porras, P. A., Schnackenberg, D., Staniford-Chen, S., Stillman, M., and Wu, F. (1998). The common intrusion detection framework architecture. http://www.isi.edu/gost/cidf/drafts/architecture.txt.

[Rozin, 2002] Rozin, A. (2002). Rapid spanning tree 802.1w simulator. http://sourceforge.net/projects/rtsplib.

[Sekar et al., 2002] Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., and Zhou, S. (2002). Specification-based anomaly detection: a new approach for detecting network intrusions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 265–274.

[Tseng et al., 2003] Tseng, C.-Y., Balasubramanyam, P., Ko, C., Limprasittiporn, R., Rowe, J., and Levitt, K. (2003). A specification-based intru-

sion detection system for aodv. In *SASN '03: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 125–134.

[Turner(editor), 2006] Turner(editor), D. (2006). Symantec Internet security threat report. Trends for January 06–June 06. Volume X.

[Uppuluri and Sekar, 2001] Uppuluri, P. and Sekar, R. (2001). Experiences with specification-based intrusion detection. *Lecture Notes in Computer Science*, 2212:172–189.