System Support for Client-Server Based Interactive Applications in Ad Hoc Networks

Thesis by

Nan Lin

In Partial Fulfilment of the Requirements
for the Degree of
Doctor of Philosophy

School of Computing Science
Newcastle University

2007

To my parents and brother

Abstract

With the emergence of wireless communications and mobile computing, new ways for people to interact with each other and their surrounding environment are emerging. Mobile devices, such as Personal Digital Assistants (PDAs) with wireless communication interfaces are able to communicate directly with each other if they are "close enough". If such devices are also able to act as message relays (routers), then a very powerful facility in the form of a mobile ad hoc network can be made available to applications. Although present day PDAs have sufficient networking and processing capabilities to support interesting interactive applications, as yet, there is hardly any software available for constructing and maintaining ad hoc networks and not many practical interactive applications have been built and tested. Rather, much of the research work is still at simulation and modelling stage. Thus it is difficult at this stage to decide what system support (middleware) for interactive applications a PDA should contain.

This work is a step in the direction of remedying the situation by considering a class of applications where interactions between users can be supported by one of the nodes in the ad hoc network acting as a server. The thesis presents the design, implementation and evaluation of three such applications on PDAs, and based on that experience, describes what system support (middleware) for such applications is required. These applications are: Auction, Bingo game and Chatting that work over ad hoc networks. The work presented here can be used as a basis for deciding how the current generation of PDA operating systems can be extended for supporting ad hoc networking and what additional services are required for interactive applications.

Acknowledgements

First of all, I would like to thank Professor Santosh Shrivastava, my supervisor, who has provided tremendous advice and support during the study of my PhD. His insight helped my work go in the right direction, and he taught me the principles of research work. I am extremely grateful for this. I also appreciate the assistance and help from Dr. Jennie Palmer, Doug Palmer, Simon Woodman, and other staffs in the distributed group of Newcastle University.

I would like to thank my parents and brother for their financial support and moral encouragement throughout the years, which made it all possible.

Finally, I would like to thank Mark E. Townson family and Miss Ruyan Chen, who bring much pleasure to my life in the UK.

Contents

List of figures

Chapter 1 Introduction

An ad hoc network is "a transitory association of mobile nodes which do not depend upon any fixed support infrastructure. ... Connection and disconnection is controlled by the distance[1] among nodes and by willingness to collaborate in the formation of cohesive, albeit transitory community. [1]"

1.1 Background

Ad hoc network is a new term for an old technology. Since the early 1970s, not long after the initial development of the packet switching technology that grew into what people now know as the Internet, the U.S. Department of Defence sponsored research to enable packet switch technology to operate without the restrictions of fixed or wired infrastructure [2]. One of the original motivations is found in the military need for battlefield survivability. Under battlefield conditions, soldiers and their mobile platforms must be able to move about freely without any of the restrictions imposed by wired communications devices. Thus, they need a mobile wireless communications system for coordinating group actions which operates in a distributed manner, avoiding single points of failure such as centralized control stations.

The key components of ad hoc networks are the portable devices and the wireless capability that connects those devices with each other. In its earlier stage due to the limitations on both aspects the applications were confined mainly in military usages. In recent years ad hoc networking is re-emerging amid evolving wireless

[1]Under the circumstance that two nodes are not directly reachable, packets will be routed node-by-node, or hop-by-hop, until these packets arrive at the destination. Hence, the hop count is usually chosen as a measure of the distance between two nodes.

communication technologies and continued miniaturization of novel devices with the extraordinary rise of processing power available. Such devices could be laptops, cellular phones, personal digital assistants (PDAs), global positioning system (GPS) receivers, household appliances, wearable computers, vehicles, watches, MP3 players, gaming consoles or hosts of sensors. Some of them have already become indispensable gadgets in people's daily life, regardless of their computing capabilities that vary from high to low; at least they are able to implement one or some specific functions. For example, PDAs help people manage their personal contacts, to-do-lists, and appointments. Or, consider an intelligent and tiny tag close to a painting in a museum which is able to store details of the exhibit's information.

The rise of wireless communication technology has also changed what it means to be "in touch". Intuitively, people could relate wireless communication with the cellular phone systems. Since AT&T and Bell Labs constructed the first analogue cellular system in 1977, over one and a half billion people globally now have digital mobile phones after a short span of 30 years [3]. These global networks have covered great parts of the world, which will soon be upgraded to the evolving third-generation (3G) cellular system. The service provision has expanded from the very beginning of voice-only communication to many kinds of data service such as Internet surfing, multimedia-messages (MMS), video calling and so on.

However, deployment of global cellular telephony networks needs investment of much money and time to set up the infrastructure. Sometimes the mechanism of such one-hop connectivity to the wired infrastructure is neither convenient nor desirable. For instance, two persons who are only several meters away in a mall making conversation will have to get help from the base station which may be several kilometres away. There should be some mechanism which helps them contact face-to-

face in a short range without the need for any infrastructure. Possible candidates are wireless Piconetwork systems [4] (as exemplified by the Bluetooth [5] radio-on-a-chip), wireless personal area network (WPAN) systems (IEEE 802.15 family of standards [6]), wireless local area network systems (IEEE 802.11 [7] and HiperLAN/2 [8] families of standards). Actually, new generation PDAs or mobile phones already have one or more wireless interfaces, which enable them either to connect to the Internet or equipments nearby. However this kind of working model still could be treated as the extension of the existing infrastructure; new features such as the IEEE 802.11 ad hoc mode capability is not exploited to full potential. The combination of mobile devices and ad hoc networking technologies opens up exciting possibilities for new forms of collaboration involving interactions among co-located people carrying mobile devices.

1.2 MANET

Within the last few years there has been a surge of interest in mobile ad hoc networks (MANET). Generally speaking, a mobile ad hoc network is a set of mobile platforms or nodes where each one is free to move about arbitrarily [9]. Each node logically acts as a router that may connect to other nodes and that also may have multiple wireless communication interfaces. A mobile ad hoc network may operate in isolation, or may expand the present Internet vision in which nodes on the edge of the network are typically connected and supported by a single wireless hop to the fixed, wired infrastructure [10]. Besides traditional military applications in battlefields, MANET can provide an extremely flexible method for establishing communications for fire/safety/rescue operations, disaster recovery scenarios or undersea operation requiring rapidly-deployable communications with survivable, efficient dynamic networking. Other likely additional applications are described in section 1.3.

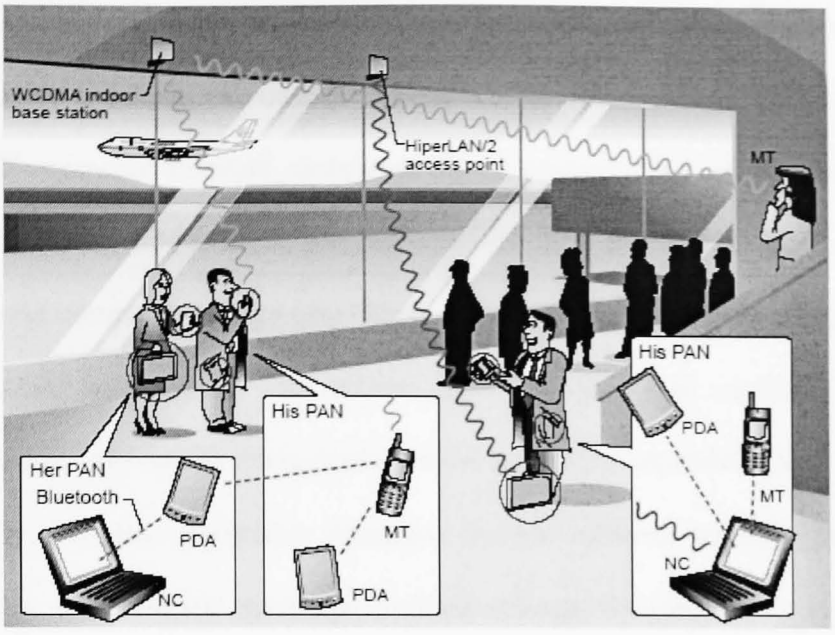


Figure 1.1: At an airport, where people can access the Internet, ad hoc Bluetooth connections that are used to interconnect personal devices, such as PDAs, 3G phones and laptops. For instance, a user might retrieve email via a HiperLAN/2 interface to the laptop in a briefcase, but read messages and reply to them via his or her PDA [11].

1.3 Spontaneous interaction in self-organized MANET

Two directions may be envisioned for emerging appliances that are wireless-networked enabled. On the one hand people can envisage the availability of Internet connections anywhere at any time. Figure 1.1 shows that passengers' portable devices can access the Internet via the 3G base station or HiperLAN/2 access point. On the other hand it would also be interesting to investigate how people can interact with their surroundings using their mobile devices in order to accomplish some tasks. With the unique characteristic of being totally independent from any authority and infrastructure, there is a great potential for users to exploit their neighbourhood. Also shown in figure 1.1, with a Bluetooth ad hoc connection passengers could use PDAs to read their own emails in the laptops. If security policy permits, furthermore, users could also share some files in their PDAs via these ad hoc connections.

It is this small but practically relevant subset of MANET space that is the focus of this thesis. Roughly speaking, two or more ad-hoc-networking-enabled entities can simply form a self-organized network spontaneously as soon as they are within reach. If every device is also able to act as message relay (router), the network can be extended to cover some comparably large area. Then a powerful facility can be made available for supporting applications. The entities have the ability to establish multi-hop communication, and can exchange information with their interested peers implicitly or explicitly. Possible application scenarios include informal social interactions in public places, opportunistic meetings in office settings, data-sharing in a conference hall, polling on high streets, interactive multi-user games, and educational multi-user application for use in classrooms.

1.4 Mobile computing middleware

Traditionally, middleware deployed upon operating systems and underneath the application layer provides application designers with a higher level of abstraction, hiding the complexity introduced by distribution. Specifically, mobile middleware aims at facilitating communication and coordination of distributed components, concealing complexity raised by mobility as much as possible. Research in the field of mobile middleware system has proliferated. However, currently available mobile middleware addresses only a few aspects of mobile applications like communication [12], data-sharing [13], or location awareness [14]. To some extent the mobile middleware design is based on assumption, speculation, and simulation, since there are not many applications proposed to help researchers clearly identify what the concrete requirements are for mobile middleware.

1.5 Research statement

In the field of MANET, much of the recent work has been concentrated on the foundation: routing protocols. Many papers, such as [15, 16, 17] (unicast, multicast, and manycast) have appeared in the literature. However, most of these studies have been investigated under extensive simulation and not many have been implemented, with the result: none of the prototype versions are available on popular operating systems. In addition, mainstream development tools do not support mobile platform well on different aspects: some on hardware; some on graphic user interface (GUI); some APIs have bugs within specific functions.

The motivation of this research stems from the current lack of availability of mobile applications that make use of MANET. Especially, there is a lack of applications which can support mobile device users interacting with each other in a limited area (such kinds of applications will be termed interactive multi-user applications). Therefore there is insufficient understanding of the requirements for mobile middleware for MANET. Consequently it is necessary to design, implement and evaluate various applications in order to understand what concrete, suitable functionalities are required for mobile middleware. This is the approach taken in this thesis. The main research objectives are as follows:

i. Investigating and understanding the salient characteristics of a class of interactive multi-user applications in MANET, and studying the concrete requirements for such applications.

ii. Studying different types of routing protocols for MANET. Analyzing their advantages and drawbacks under different conditions. Pinpointing the suitable ones capable of supporting proposed interactive multi-user applications.

iii. By taking account of proposed applications, coming up with an initial design of relevant components of mobile middleware.

iv. Implementing a few interactive multi-user applications, starting from routing protocol upwards.

v. Evaluating the interactive multi-user applications.

vi. Revisiting the original middleware design and proposing a refined design.

This thesis broadly investigates state-of-the-art technologies related with ad hoc networking and pays much attention to the needs of interactive multi-user applications in self-organized MANET. The main contribution is to implement a modular, general-purpose, set of middleware components capable of supporting representative applications in ad hoc networks. Three applications have been selected and implemented. They are "Auction", "Bingo game", and "Chatting". This thesis describes how an application specific ad hoc network can be formed and maintained in an economic manner: the routing information is acquired together with application information reactively. There is no need to send route discovery, route request or route error packets separately. A node can recover lost packets by inquiring from its direct neighbour nodes. These two features substantially reduce network traffic. The system is adaptive to its operational surroundings: some performance-related parameters can be adjusted automatically when the system senses any variation of its surrounding neighbourhood. Field tests on HP Jornada and iPAQ PDAs have been conducted together with systematic simulation studies. The work has revealed valuable data and results. In particular, a novel, adaptive, and efficient broadcast protocol has been developed. The thesis provides a sound foundation for building future mobile middleware. The development tools are standard packages of J2SE (Java 2 Standard Edition) without any extra enhancement.

1.6 Thesis outline

The rest of this thesis is organized as follows: with regard to the OSI network reference model, chapter 2 reviews related work in ad hoc networking covering most design issues from MAC layer to application layer. In addition, this chapter introduces simulation tools available and relevant simulation study results. Chapter 3 describes the interactive multi-user application scenarios in a subset of MANET that is called "Convenience Networks". The characteristics of the system and related applications are presented. Chapter 4 identifies general design requirements and specific design issues for interactive multi-user applications in Convenience Networks based on the three example applications (Auction, Bingo game, Chatting) discussed in chapter 3. The initial design and implementation of a generalized data process module that is capable of supporting the three applications are presented. Chapter 5 presents detailed evaluation from two sources: field tests in a small-scale test bed and dedicated simulation studies. Based on the results from the evaluation, chapter 6 describes how mobile middleware can be made responsive to changes in its working environment. The design of a prototype of such adaptable middleware, the revised data process module with adaptive broadcasting features, is presented. The functionalities of the revised components and how the system performance can be enhanced are described. Simulation studies presented confirm such improvements. Chapter 7 concludes the thesis and points out the direction for future work.

Chapter 2 Related Work

As an investigation of a system for supporting mobile applications, this chapter covers design issues from layer 2 to layer 7 of the OSI network reference model. There is much demand for systems that can support new requirements imposed by users' mobility coping at the same time with complex heterogeneity of hardware and software. This chapter investigates some of the most relevant technologies of mobile computing to date: MAC protocols, routing protocols and related security issues, service discovery standards, mobile middleware, as well as describes some proposed applications in mobile environments. Towards the end of this chapter, network simulators and their use in simulation studies are also introduced as they are useful tools for evaluating research results in mobile computing.

2.1 Wireless communication technologies for ad hoc networks

Wireless networks are transforming the way people use computers and other personal electronic devices at work, home, and when travelling. This section surveys two leading wireless communication technologies that support ad hoc networking.

2.1.1 Bluetooth

Bluetooth [5, 18, 19] provides a short-range (no more than 10 meters) wireless connectivity, aimed at replacing cables between linked electronic devices like PDAs with GPS receivers or mobile phones with hand-free kits. A piconet is formed by two or more Bluetooth devices that share the same radio channel. Within in a piconet, a Bluetooth device can be either the master or the slave. Each piconet always contains

one master and up to seven active slaves. Any Bluetooth device can become the master in a piconet. A Bluetooth device can be active in more than one piconet, but can be the master into only one. Communication in a piconet is controlled by the master according to a polling scheme. The master polls each slave; a slave is only allowed to send a packet after having been polled by the master. Furthermore two or more piconets can be interconnected to form a scatternet. However scatternet for constructing multi-hop ad hoc network is not specified enough to enable implementation.

2.1.2 IEEE 802.11 ad hoc mode

The IEEE 802.11 standard [7, 19] defines an operational mode "infrastructure-less" or "ad hoc" so that it makes this technology an enabler for ad hoc networks. When operating in this mode, nodes are said to form an Independent Basic Service Set (IBSS). Each IBSS is uniquely assigned an identifier: "IBSSID". The locally administrated IBSSID will be used by any other nodes that join an IBSS. Actually the IEEE 802.11 standard is a good platform to implement single-hop ad hoc networks as nodes must be within the same transmission range to be able to communicate. With regard to multi-hop ad hoc networks, this requires the addition of routing mechanisms at nodes (see section 2.2) so that nodes can forward packets toward the intended destination, thus extend the scope of the ad hoc network beyond the transmission range of the source node.

2.1.2.1 IEEE 802.11 MAC layer

The Distributed Coordination Function (DCF) contention-free service provided by the IEEE 802.11 MAC layer supports nodes working in the "ad hoc" mode, as another

service Point Coordination Function (PCF) cannot. The DCF service is made available on top of a variety of physical layers. Three different technologies have been specified in the standard: Infrared (IF), Frequency Hopping Spread Spectrum (FHSS), and Direct Sequence Spread Spectrum (DSSS).

The DCF service provides the basic access method of the IEEE 802.11 MAC protocol, which is based on a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) scheme. According to the DCF service, before transmitting a data frame, a node must sense the channel to determine whether any other node is transmitting. If the medium is found to be idle for an interval longer than the Distributed InterFrame Space (DIFS), the node continues transmitting. On the other hand if the medium is busy, the transmission is deferred until the end of the ongoing transmission. A random interval, referred to as the back-off time, is then selected to initialize the back-off timer. The back-off timer is decreased for as long as the channel is sensed as idle, stopped when a transmission is detected on the channel, and reactivated when the channel is sensed as idle again for more than a DIFS.

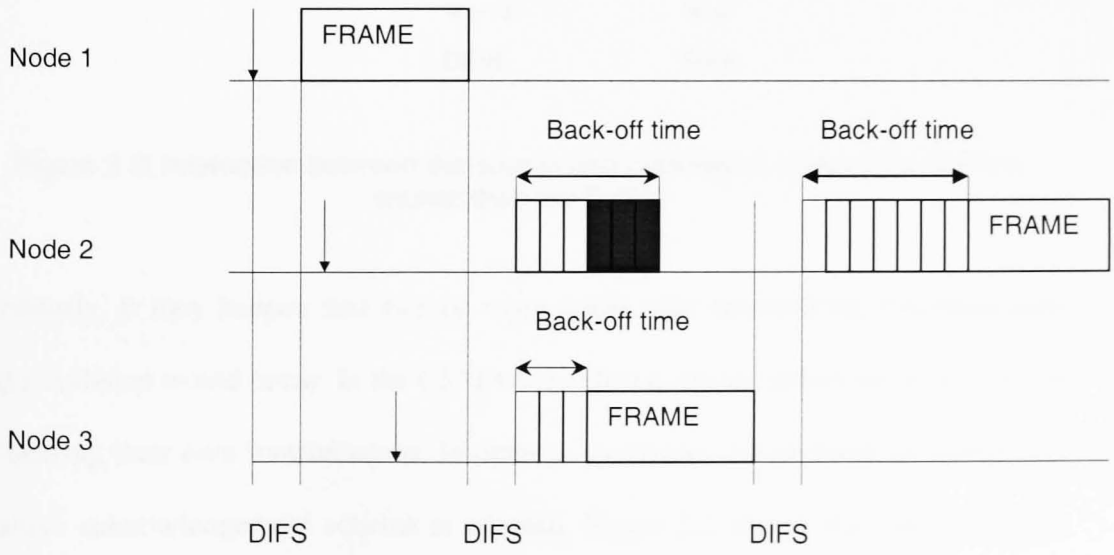


Figure 2.1: The IEEE 802.11 basic access mechanism

The back-off time is slotted. Specifically, the back-off time is an integer number of slots uniformly chosen in the interval (0, CW-1). CW is referred to as the Contention Window. At the first transmission attempt, CW $=CW_{min}$, and it is doubled at each retransmission up to CW_{max}. (The values of CW_{max} and CW_{min} depend on the physical layer adopted, for example, for the FHSS physical layer $CW_{max}=1024$ and $CW_{min}=16$, respectively [7].) Figure 2.1 shows that the slotted back-off times of node 2 and node 3, which at the beginning have the value of six and three respectively. The back-off timer of node 2 is disabled (shown in black colour) while node 3 is transmitting its frame; the timer is reactivated (setting the new value of seven) a DIFS after node 3 has completed its transmission.

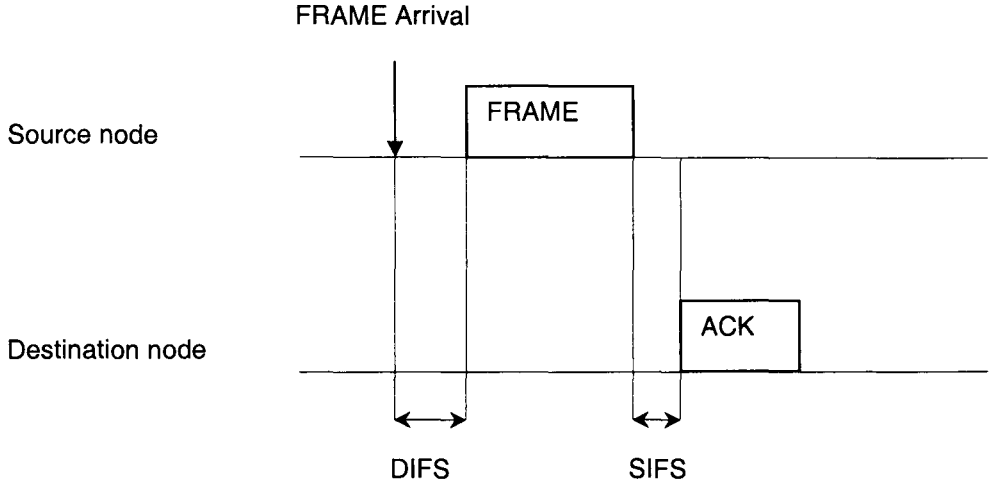


Figure 2.2: Interaction between the source and destination nodes (The SIFS is shorter than the DIFS.)

Obviously, it may happen that two or more nodes start transmitting simultaneously and a collision would occur. In the CSMA/CA scheme, nodes cannot detect a collision by hearing their own transmissions. In order to overcome this problem an immediate, positive acknowledgement scheme is adopted. Figure 2.2 shows that the destination node initiates the transmission of an acknowledgement frame (ACK) after a time

interval called the Short InterFrame Space (SIFS) upon receiving a data frame. The SIFS is shorter than the DIFS in order to give priority to the receiving node over other possible nodes waiting for transmissions. If the ACK is not received by the source node, the data frame is presumed to have been lost, and a retransmission is scheduled. The ACK is not transmitted if the received frame is corrupted. A Cyclic Redundancy Check (CRC) algorithm is used for error detection. After an erroneous frame is detected, a node must remain idle for at least an Extended InterFrame Space (EIFS) interval before it reactivates the back-off algorithm. Reception of an error-free frame during the EIFS resynchronizes the node to the actual busy/idle state of the medium, so the EIFS is terminated and normal medium access continues following the reception of that frame.

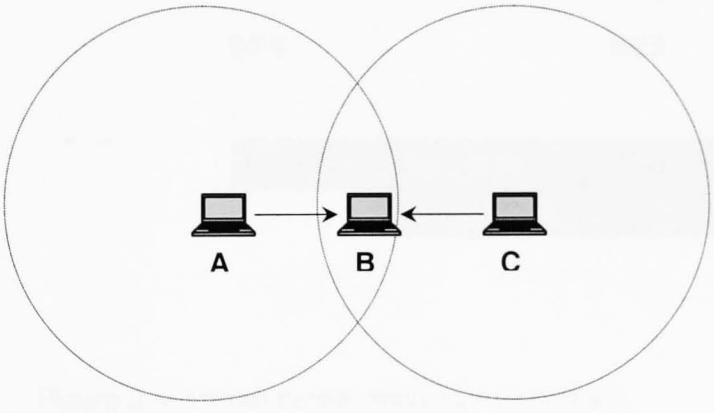


Figure 2.3: The "hidden terminal" problem

2.1.2.2 Common problems in wireless ad hoc networks

Under the IEEE 802.11 DCF protocol, the characteristics of wireless medium generate complex phenomena like the "hidden terminal" [20] and "exposed terminal" [20] problems.

Figure 2.3 shows a typical "hidden terminal" scenario. Assume that node B is in the transmitting range of both node A and node C, but A and C cannot hear each other. A is transmitting to B. According to the IEEE 802.11 DCF protocol, C senses the channel first if it wants to transmit a frame to B. C finds the medium free because it is not able to hear A's transmission. Therefore it starts transmitting the frame but it will result in a collision at destination B.

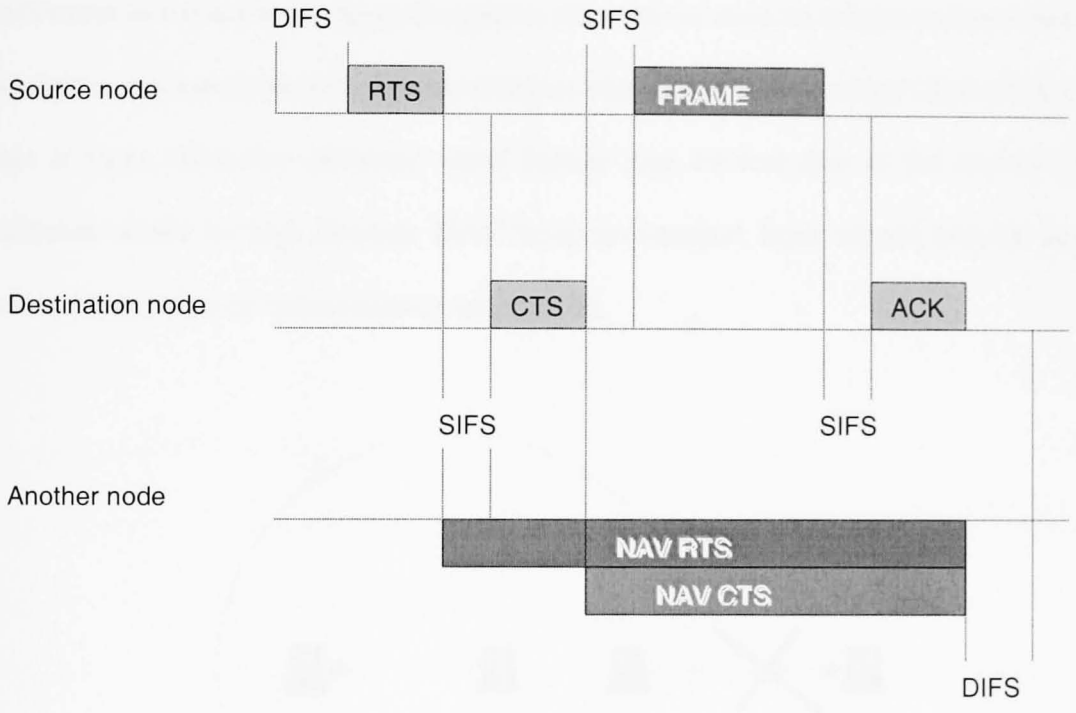


Figure 2.4: Virtual career sensing mechanism

The "hidden terminal" problem can be alleviated by a virtual carrier sensing mechanism that is based on two control frames: Request To Send (*RTS*) and Clear To Send (*CTS*). According to this mechanism, before transmitting a data frame, the source node sends a *RTS* frame to the destination node announcing the upcoming frame transmission (see figure 2.4). Upon receiving the *RTS* frame, the destination node replies by sending a *CTS* frame to indicate that it is ready to receive the data

frame. Both the *RTS* and *CTS* frames contain the total duration of the transmission, that is, the overall time interval needed to transmit the data frame and the related ACK. This information can be heard by any listening node that uses this information to set up a timer called the Network Allocation Vector (NAV). When the NAV is greater than zero, the node must refrain from accessing the medium. By adopting the *RTS/CTS* mechanism, nodes may become aware of transmissions from hidden nodes and learn how long the channel will be used for these transmissions. The same restriction as for acknowledgement applies: the receiver must be unique so that it only works for unicasting packets. Broadcasting or multicasting has to adopt CSMA/CA so that it faces reliability problem: some frames may be lost due to the chance of collision would be high because MAC layer or transport layer cannot provide any acknowledgement or retransmission mechanism.

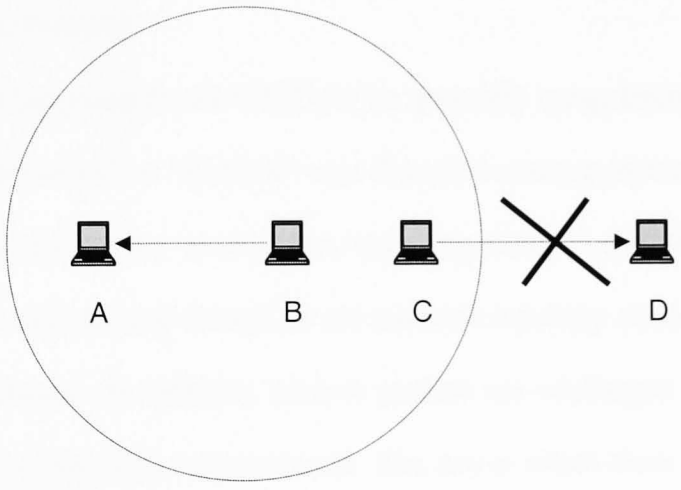


Figure 2.5: The "exposed terminal" problem

Figure 2.5 depicts a typical scenario in which the "exposed terminal" problem may occur. Assume that both node A and node C can hear transmissions from node B, but A cannot hear from C. B is transmitting to A and C wants to send frames to node D.

According to the IEEE 802.11 DCF protocol, C senses the medium and finds it busy for B's transmission. Therefore it refrains from transmitting to D, though its transmission will not result in a collision taking place at A. The "exposed terminal" problem may thus result in a throughput reduction, as C could have sent frames to D.

2.2 Routing protocols for MANET

For mobile ad hoc networks (MANET), the issue of routing packets between any pair of nodes is a central challenge because of node mobility. Development of dynamic routing protocols that can efficiently establish and maintain routes between two communication entities that may be multi-hop away has been the most active research area in ad hoc networking by far. Considering the fact that MANET could work as a self-organized system, or an extension of the wide deployment of the Internet, or a combination of the two, the proposed routing protocols should be compatible with traditional Internet Protocol.

Routing protocols proposed for MANET can be generally categorized by the routing strategy, either "proactive" or "reactive" way. Proactive routing protocols work in the way that they maintain routes to all nodes, including those to which no packets are being sent. They react to any change in the network topology even if no traffic is affected by the change. In addition, control packets are exchanged periodically to maintain routes to every node in the network. The rate at which these control packets are sent must reflect the dynamics of the network in order to maintain valid routes. Hence scarce resources like power or link bandwidth will be used more frequently for control traffic as node mobility increases. An alternative approach is to establish routes reactively. A route between a pair of nodes is determined only when they are explicitly needed to send packets. This allows network nodes to focus on either the

routes that are being used, or the routes that are being set up, which prevents the unnecessary route updating packets.

2.2.1 Destination-sequenced distance-vector routing

"Destination-sequenced distance-vector (DSDV) [21] is a proactive hop-by-hop distance vector routing protocol [11]."

In a network, each node maintains a route table containing the next hop to any available destination with the corresponding hops count that will be required. Each node needs to advertise its route table to each of its direct neighbours. The frequency of the advertisement must be adapted to node mobility, ensuring that every node can always locate other nodes in the network. Each node agrees to relay data packets to others upon request. Besides these a significant characteristic is that each route table entry is tagged with a sequence number that is originated by the destination node to indicate how new, or fresh, a given route is [21]. Altogether with its own increasing sequence number each node broadcasts or multicasts routing packets periodically and incrementally as topological changes are detected.

Receiving a new routing packet a node compares with those which are already available from previous received routing information. Routes with more recent sequence numbers are always preferred; routes with older sequence numbers are discarded [21]. Of the routes with the same sequence number, those with a "better" metric (usually choose small "hop count") will be used; the existing route is discarded or stores as less preferable [21]. Before the node advertises its route table, the metrics for routes chosen from the newly received routing information are each incremented by one hop, as incoming packets will require one more hop to reach the destinations. Because there are two ways for new route to be chosen, regarding that timing skews

between the various nodes are expected, it may turn out that a particular node receives new routing information in a pattern that causes it to consistently change route from one next to another, even when the destination node has not moved [21]. Hence routes that show an improved metric may be scheduled for advertisement at a later time, which time depends on the average setting time for routes to the particular destination under consideration [21].

The broken link to a former neighbour may be detected by the MAC protocol, or it may be inferred if no any broadcast has been received for a while from the node. When a link to the node has broken, any route through it is immediately assigned an infinite metric with an updated sequence number. Then an updating routing packet is broadcast announcing the broken link; this is the only situation in which the sequence number is generated by any mobile node other than the destination.

2.2.2 Dynamic source routing

Dynamic source routing (DSR) [15, 22, 23] is a reactive routing protocol that uses source routing instead of hop-by-hop to deliver data packets. The "source route" field in packets' headers carry the addresses of the nodes through which the packets must pass. Thus the main benefit is that intermediate nodes need not keep route information since it has been explicitly encapsulated in the packets. Nor any kind of periodic packets need to be sent. DSR supports unidirectional and asymmetric links, setting up routes based on the source's demand absolutely. Two mechanisms that work together to allow the discovery and maintenance of source routes in MANET.

2.2.2.1 Route discovery

When a source needs to send a data packet but has little routing information to the destination, it initiates a route discovery. To establish a route, the source floods a

Route Request packet with a unique request ID. When this request packet arrives at the destination or a node has the route to the destination, it responds a Route Reply packet containing route information back to the source along with the addresses accumulated in the request packet's header. The "route cache" maintained at each node records routes the node has learned and overheard over time to reduce overhead generated by a route discovery phase [24].

When a node receives a Route Request packet, this packet is forwarded only if all of the following conditions are met:

i. The node is not the target (destination) of the Route Request packet.

ii. The node is not listed in the "source route" field of the Route Request packet.

iii. The Route Request packet is not a duplicate.

iv. No route information to the target is available in its route cache.

If all are satisfied, the node appends its identification (address) to the "source route" field in the packet's header and broadcasts the packet. If condition (ii) or (iii) is not met, it simply discards the packet. If a node is the destination of the Route Request packet or has route information to the destination, it builds and sends a Route Reply packet to the source, as described above. Figure 2.6 shows the procedure when node A is sending a packet to the target node D.

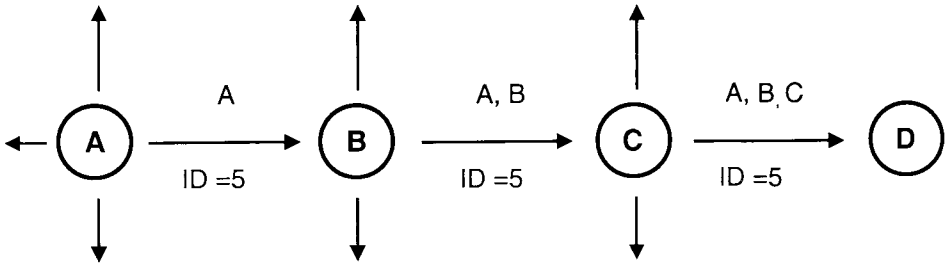


Figure 2.6: Route discovery example with node A as the initiator and D as the target

2.2.2.2 Route maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that the packet has been received by the next hop along the source route; the packet is retransmitted (up to a maximum number of attempts) until this confirmation of receipt is received [23]. It is achieved by passively listening for the transmission of the next-hop neighbour or by setting a bit in a packet's header to request an explicit acknowledgment. When a node fails to receive an acknowledgment, it sends a Route Error packet to the originator to invoke a new route discovery phase, identifying the link over which the packet could not be forwarded, see figure 2.7. Nodes that receive a Route Error packet delete any route entry (from their route cache) which uses the broken link [23].

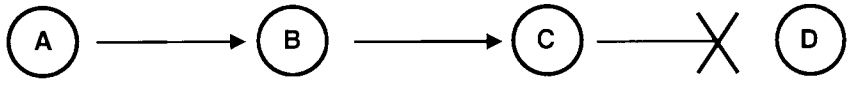


Figure 2.7 Route maintenance example: node C is unable to forward a packet from A to D over its link to the next hop, D.

2.2.2.3 Additional features

A number of additional features like the use of "route cache", preventing route reply storms, packet salvaging, automatic route shortening, and so on is possible to optimize the two basic operations of route discovery and route maintenance. Therefore, these features can reduce the number of overhead packets and improve the efficiency of the routes used in data packets.

2.2.3 Ad hoc on-demand distance-vector protocol

Ad hoc On-demand distance-vector (AODV) protocol [25, 26] is essentially a combination of DSR and DSDV. It borrows the basic on-demand mechanism of Route Discovery and Route Maintenance from DSR, plus the use of hop-by-hop routing, sequence numbers for each destination, and periodic beacon from loop-free DSDV.

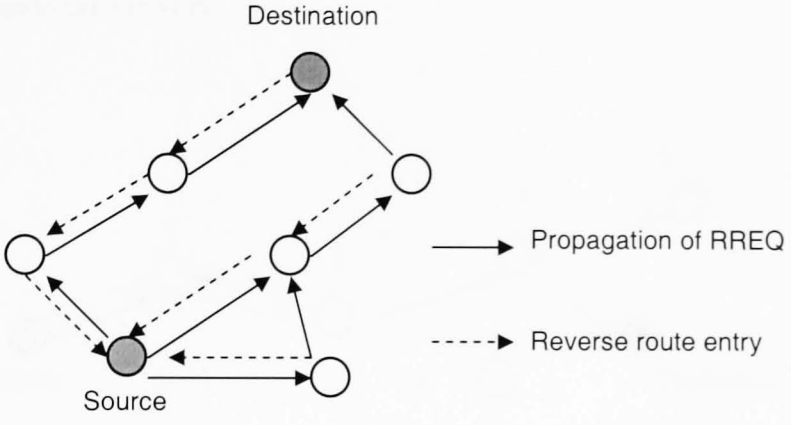


Figure 2.8: Propagation of RREQs throughout the network

When a node needs to send a data packet to some destination without a valid route to that node, it initiates broadcasting a Route Request packet (RREQ) to its neighbours, including the last known sequence number for the destination and other mandatory address information [26]. The RREQ is flooded in a controlled manner same as DSR through the network until it reaches the destination or a node having a route to the destination. Each node that forwards the RREQ records in its own route table the address of the neighbour from which the first copy of the RREQ is received, thereby creates the reverse route; in this way, the node knows how to forward a Route Reply packet (RREP) to the source if one is received later [26]. When the RREQ reaches a node having a fresh enough route to the destination, a RREP is created then unicast back to the neighbour along the reverse route; nodes along this route set up forward

route entries in their route tables which point to the node from which the RREP came. Associated with each route entry is a lifetime. If this route entry is not used within the specified lifetime, the route information is deleted to prevent stale routing information from lingering in the route table [26]. If the route request is lost, the source is allowed to retry the broadcast route discovery mechanism. Figure 2.8 shows the propagation of RREQs across the network as well as the formation of the reverse route entries at each node inside the network.

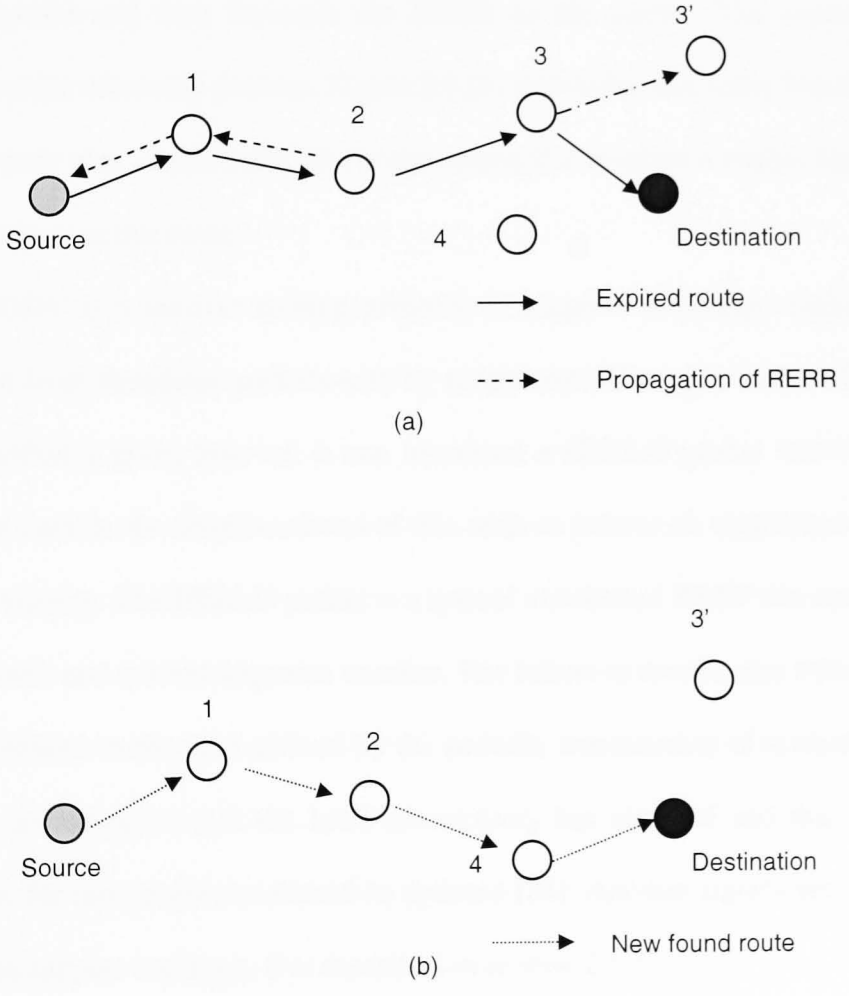


Figure 2.9: Route maintenance in AODV

Routes are maintained as long as needed by the source. If a source moves, it is able to reinitiate the route discovery process to find a new route to the destination. If either the destination or some intermediate node moves, the upstream neighbour of the node notices the move and propagates a Route Error packet (RERR) to each of its active upstream neighbours to inform them of the erasure of that part of the route. Figure 2.9 illustrates the procedure. In figure 2.9 (a) the original route from the source to the destination is coming through nodes 1, 2, and 3. Node 3 then moves to the new location. Node 2 notices the break and sends a RERR to node 1. Node 1 marks this route as invalid and then forwards the RERR to the source. The source has to reinitiate a route discovery process. Figure 2.9 (b) shows the new route found through node 4. A node also creates a RERR for the source if it receives a packet for which it does not have an active route.

Though AODV is a reactive routing protocol, it supports obtaining neighbourhood information from broadcast packets sent by neighbours. If a node has not broadcast anything within a given interval, it can broadcast a HELLO packet that cannot be rebroadcast outside the neighbourhood of this node to inform its neighbours that it is still in the vicinity. The HELLO packet is a special unsolicited RREP that contains the node's address and current sequence number. The failure to receive any transmissions from a neighbour in the time defined by the periodic transmission of several HELLO packets is an indication that the local connectivity has changed and that the route information for this neighbour should be updated [26]. Another significant feature of AODV is to support multicast. It is described in section 2.3.1.

2.2.4 Other routing protocols

Many other protocols have been proposed for MANET including: Temporally-Ordered Routing Algorithm (TORA) [27], Optimized Link State Routing (OLSR)

[28], and so forth. If GPS can be used to help nodes gather enough location information, Location-Aided Routing (LAR) [29], Distance Routing Effect Algorithm for Mobility (DREAM) [30] may be the suitable candidates. Zone Routing Protocol (ZRP) [31] is a hybrid protocol: it uses proactive approach to nodes within the zone and reactive approach to nodes outside the zone. Core-Extraction Distributed Ad hoc Routing (CEDAR) algorithm [32] selects a minimum set of nodes (core) to perform Quality of Service route computations. Readers are referred to [33, 34] for survey of routing protocols.

2.3 Multicast routing protocols for MANET

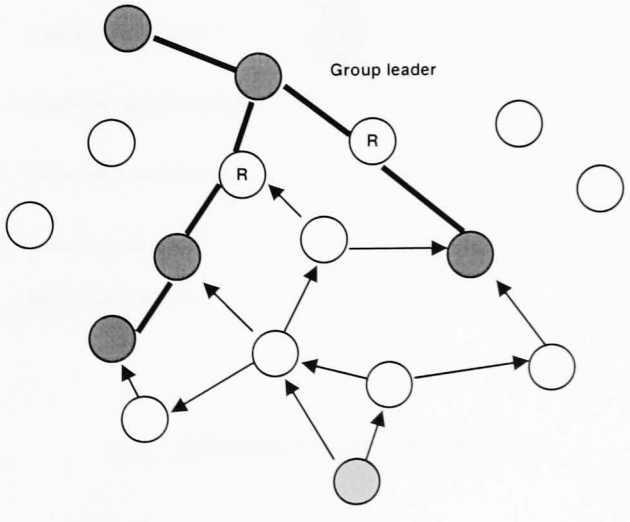
Multicast plays an important role for collaborative and, more generally, group communication applications in MANET. Many multicast routing protocols have been proposed in recent years, like Core-Assisted Mesh Protocol (CAMP) [35], Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) [36], and so forth. In this section, two outstanding on-demand multicast protocols are introduced.

2.3.1 A multicast extension of AODV

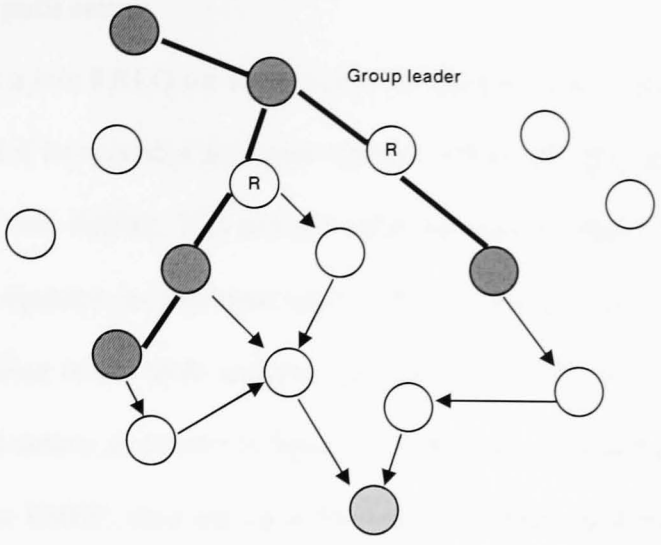
The algorithm [37] uses the same two packet types (RREQ and RREP) for the multicast route request and route reply as previously described in section 2.2.3. Only one new packet type, the Multicast Activation packet (MACT), is needed. When nodes join the multicast group (or group for short), a multicast tree composed of group members and nodes connecting the group members is created. Each group has associated with it a group leader. The group leader is responsible for maintaining and disseminating the group sequence number indicating the freshness of the routing information for the group.

2.3.1.1 Route discovery

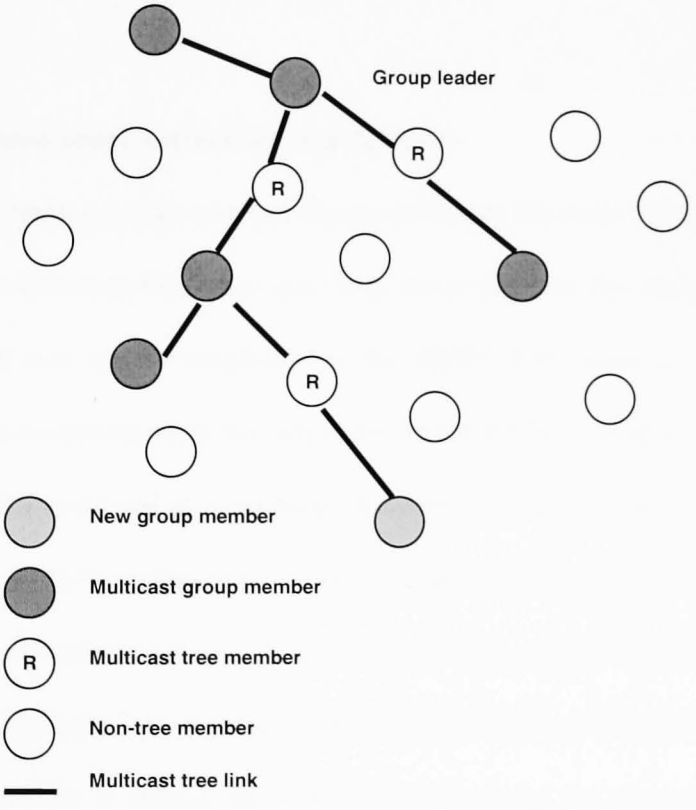
The source broadcasts a RREQ setting a *join* flag in the packet's header if it wishes to join the group. If the RREQ is a *join* request, only a node that is a member of the desired multicast tree may respond. Otherwise a node receives the *join* RREQ for a group of which it is not a member, or if it does not have a route to that group, it creates a reverse route entry to the source and then broadcasts the packet. Figure 2.10 (a) shows the propagation of a join RREQ. The source waits a fixed period to receive a reply. If it does not so, it rebroadcasts its request with the new broadcast ID increased by one. If not receiving any reply after the maximum number of broadcasts, this node can assume that no other group members exist in the connected portion of the network. If the node was attempting to join the group, it becomes that group's leader. When a node receives a *join* RREQ, it adds an un-activated entry for the source in its multicast route table. Each next-hop entry in the route table has an associated Activated flag. If the flag is false, the node does not forward any data packet for the group along that link. Only after the link is activated explicitly can it be used to send data packets.



(a): RREQ propagation



(b): RREP sent back to the source



(c): Multicast tree branch addition

Figure 2.10: Multicast join operation

2.3.1.2 Forward path setup

If a node receives a *join* RREQ for a group, it may reply if it is a router for the group's multicast tree and if its recorded sequence number for the group is at least as great as that contained in the RREQ. The group leader can always reply this packet. The responding node updates its multicast route table by placing the requesting nodes' next-hop information in the table and then generates a RREP. The node unicasts the RREP back to the source as shown in figure 2.10 (b). As nodes along the route to the source receive the RREP, they set up a forward path entry in their multicast route table by adding the node from which they received the RREP as a next hop; the value of the Hops Count field in the entry is increased. Then they forward the RREP to the next node.

2.3.1.3 Multicast route activation/ deactivation

At the end of discovery interval, the source activates the route by unicasting a MACT to its selected next hop. Once this next-hop mode receives this packet, it activates the route and, if it was not the originator of the RREP, then sends its own MACT to its next hop. This continues until the originator of the RREP is reached. At that point the new route to the multicast tree has been determined. The procedure is shown in figure 2.10 (c). The MACT is also used when a node wishes to leave the group. If a leaf node on the multicast tree wants to leave, it unicasts a MACT to its next hop. It then deletes the group information from its multicast route table. When the node's next hop receives the packet, it deletes the next-hop information for the sending node. If the process makes this node a leaf, and if the node is not a group member, it may similarly prune itself from the tree by unicasting a MACT to its next hop. Otherwise it keeps its status unchanged.

2.3.1.4 Multicast tree maintenance

The group members should be always reachable; unlike the unicast scenario the destination does not need to be reachable unless another node is currently sending packets to it. When a link is broken, the node downstream of the break is responsible for repairing it. This node initiates the repair by broadcasting a *join* RREQ for the group which includes an extension field indicating the sending node's distance from the group leader. Only nodes on the multicast tree that are at least this close to the multicast group leader may reply to the RREQ. Any nodes can respond to a RREQ by sending a RREP as long as the following conditions are met:

i. It is part of the multicast tree.

ii. It has a fresh enough group sequence number.

iii. Its hops count to the group leader is smaller than that indicated by the extension field of the RREQ.

Then forward path setup, subsequent route deletions and unicasting MACT occur as described before. The MACT should include the information that the distance to the group leader may have changed.

2.3.2 On-demand multicast routing protocol

"On-demand Multicast routing protocol (ODRMP) is a mesh-based, instead of tree-based, multicast protocol that provides richer connectivity among multicast members [16]". A mesh for each multicast group relates with a forwarding group, which is made up of a set of nodes responsible for forwarding multicast packets on the shortest route between any pair in the multicast group. Comparing with trees, the mesh provides richer connectivity among multicast members helping overcome node displacements and channel fading.

A request phase and a reply phase, like on-demand unicast routing protocols, comprise the protocol. When a multicast source needs to send packets, it floods an advertising packet, named JOIN QUERY, with data payload piggybacked. The packet is periodically broadcast into the whole network. The process helps nodes refresh their membership information as well as update the routes as follows.

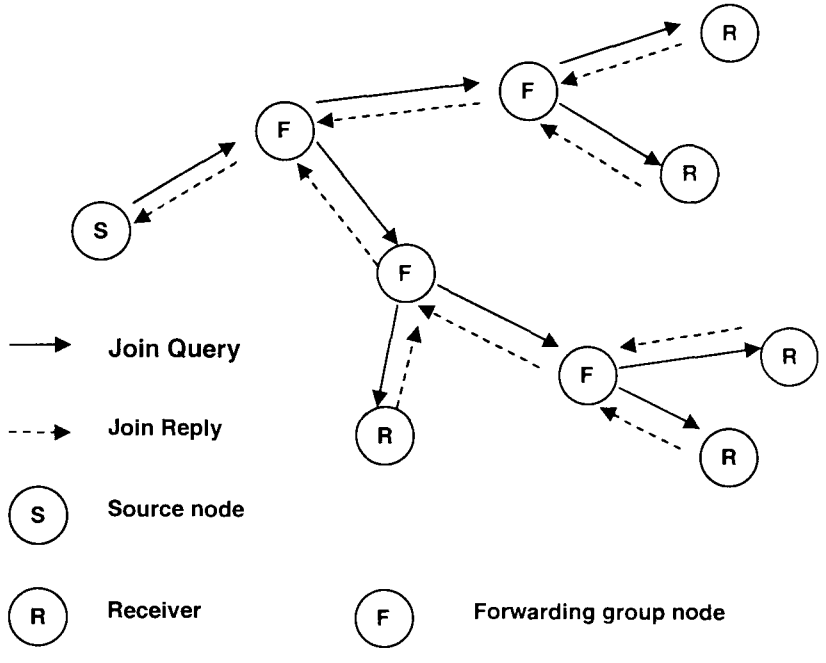


Figure 2.11: On-demand procedure for membership setup and maintenance [16]

When a node receives the first arrived JOIN QUERY, it stores the upstream neighbour ID into its multicast route table then rebroadcasts the packet. When the JOIN QUERY reaches a multicast receiver, the receiver responds a packet, named JOIN REPLY, by broadcast. The JOIN REPLY comprises a list of entries; each entry comprises a sender ID and the related next-hop ID. When a node receives a JOIN REPLY, it checks if the next-hop ID of one entry in the packet matches its own ID. If it does, the node realizes that it is on the route to the source and thus is part of the forwarding group. "It then sets the Forwarding Group Flag up and broadcasts its own

JOIN REPLY built upon matched entries. The JOIN REPLY is thus propagated by each forwarding group member until it reaches the multicast source via the shortest path [16]". Figure 2.11 shows the whole procedure: the process constructs the routes form sources to receivers and builds a mesh of nodes, the "forwarding group" [16]. In ODRMP, no explicit control packets need to be sent to join or leave the group [16]. If a multicast source wants to leave it just stops sending JOIN QUERY while receivers do not send JOIN REPLY if they no longer want to receive from a particular multicast group. Another major strength of ODRMP is its unicast routing capability [38, 39].

2.4 Security in ad hoc network routing

Security in ad hoc networks is an essential component for packet routing; network operation can be easily jeopardized if countermeasures are not embedded into basic network functions at the early stages of their design [40]. The nodes in an ad hoc network cannot be trusted for the correct execution of critical network functions. Therefore cooperative security schemes seem to offer the only reasonable solution: node misbehaviour can be detected through the collaboration among a number of nodes assuming that a majority of nodes do not misbehave [40]. The article [41] reviews attacks on ad hoc networks and discusses approaches for establishing cryptographic keys in ad hoc networks. The state of research in secure ad hoc network routing protocols and related research challenges are also described.

2.4.1 SEAD

Secure Efficient Ad hoc Distance vector routing protocol (SEAD) [42] is a proactive secure routing protocol based on DSDV. SEAD is to authenticate the sequence number and metric in each entry of a routing update packet (see table 2.1) using one-

way hash chains. In addition, the receiver of SEAD routing update packets also authenticates the sender, ensuring that the packet originates from the correct source. In each routing update packet, each node uses a specific element from its hash chain about itself (the metric is always zero). "Based on this initial element, the one-way hash chain provides authentication for the metric's low bound in other routing update packets for this destination; attackers can increase the metric or claim the same metric but cannot decrease it [42]".

Destination	Metric	Next hop	Seq. No.	Hash value
A	3	C	3	A36D492F
B	2	D	2	1AA498B4

Table: 2.1: Two entries in a SEAD routing update packet

A node uses a hash function $H(x)$. To generate its one-way hash chain, the node chooses an arbitrary value x and computes the sequence values $h_0, h_1, h_2, \ldots, h_n$, where $h_0 = x$ and $h_i = H(h_{i-1})$ for $0 < i \leq n$. Therefore, given h_i it can verify the authenticity of h_j if $j < i$. However, the distribution of h_n is beyond the scope of SEAD. While the maximum hops count between any pair of nodes in the ad hoc network is $m-1$ (n is divided by m), SEAD uses the sequence number i in some entry of a routing update packet to determine a contiguous group of m elements from the node's hash chain. These elements are $h_{km}, h_{km+1}, \ldots, h_{km+m-1}$ for $k = n/m - i$; one of them is chosen to authenticate the entry. Specifically, if the metric for this entry is j, $0 \leq j < m$, then h_{km+j} is used to authenticate the entry for that sequence number i, given any early authentic hash element from the same hash chain.

2.4.2 Ariadne

"Aridane is an on-demand secure ad hoc routing protocol based on DSR that withstands node compromise and relies on highly efficient symmetric cryptography [43]". It is more secure and general; every filed in the packet's header can be authenticated rather than the sequence number and metric in SEAD. Three authentication schemes can be supported whereas the use of Ariadne with Tesla [44] is the main interest. Tesla achieves the asymmetry authentication from loose clock synchronization and delayed key disclosure [44]. For each message Tesla adds only message authentication code (MAC) for broadcast authentication. In the following discussion, the initiator *S* performs a route discovery for the target *D* assuming they share the secret key pair of K_{SD} and K_{DS} for message authentication in each direction.

2.4.2.1 Ariadne route discovery

Figure 2.12 shows an example of Ariadne route discovery and reply. The route discovery packet (RREQ) is enhanced by eight additional fields used to provide authentication and integrity to the routing protocol as follows:

<ROUTE REQUEST, initiator, target, ID, time interval, hash chain, node list, MAC list>

As in DSR, the "initiator" and "target" are set to the addresses of *S* and *D*. In addition *S* sets the "ID" to an exclusive value in initiating each route discovery. The value of "time interval" is the Tesla time interval at the expected arrival time of this request at *D*, taking account of clock skew. Then *S* initializes the "hash chain" leaving the "node list" and "MAC list" empty.

When any node *A* receives a RREQ for which it is not the target, same as in DSR, it checks its local pair of <initiator, ID> values recording the latest request it has

received, to determine if it has already received the request. If it has, *A* discards the packet. In addition *A* also checks if the value of "time interval" in the request is valid: this value must not be too far in the future, and the key corresponding to it must not have been disclosed yet [44]. If this value is not valid, *A* discards the packet. Otherwise, *A* modifies the RREQ by appending its own address to the "node list", replacing the "hash chain" with the hashed value, and appending a MAC of the entire RREQ to the "MAC list". *A* uses the Tesla key to compute the MAC, where the key is related with the "time interval" specified in the request. Afterwards *A* rebroadcasts the modified RREQ.

S: $h_0 = \mathrm{MAC}_{K_{SD}}(\mathbf{REQUEST}, S, D, id, ti)$

$S \rightarrow *$: $\mathbf{REQUEST}, S, D, id, ti, h_0, (), ()\rangle$

A: $h_1 = H[A, h_0]$

$M_A = \mathrm{MAC}_{K_{A_{ti}}}\ \mathbf{REQUEST}, S, D, id, ti, h_1, (A), ()$

$A \rightarrow *$: $\mathbf{REQUEST}, S, D, id, ti, \mathbf{h_1}, (A), \mathbf{M_A})$

B: $h_2 = H[B, h_1]$

$M_B = \mathrm{MAC}_{K_{B_{ti}}}(\mathbf{REQUEST}, S, D, id, ti, h_2, (A, B), (M_A))$

$B \rightarrow *$: $\mathbf{REQUEST}, S, D, id, ti, \mathbf{h_2}, (A, B), (M_A, \mathbf{M_B})\rangle$

C: $h_3 = H[C, h_2]$

$M_C = \mathrm{MAC}_{K_{C_{ti}}}(\mathbf{REQUEST}, S, D, id, ti, h_3, (A, B, C), (M_A, M_B))$

$C \rightarrow *$: $\mathbf{REQUEST}, S, D, id, ti, \mathbf{h_3}, (A, B, \mathbf{C}), (M_A, M_B, \mathbf{M_C})$

D: $M_D = \mathrm{MAC}_{K_{DS}}(\mathbf{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C))$

$D \rightarrow C$: $\mathbf{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), \mathbf{M_D}, ()$

$C \rightarrow B$: $\mathbf{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (\mathbf{K_{C_{ti}}})$

$B \rightarrow A$: $\mathbf{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, \mathbf{K_{B_{ti}}})$

$A \rightarrow S$: $\mathbf{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, K_{B_{ti}}, \mathbf{K_{A_{ti}}})$

Figure 2.12: Route discovery and reply example in Ariadne. The bold font indicates changed packet fields, relative to the previous packet of that type. [43]

When *D* receives the RREQ eventually, it also checks the validity of the packet. The keys from the specified value of "time interval" must not been disclosed and the "hash chain" filed be equal the value *D* computes. *D* computes its own hashed value in the following way. *D* can calculate the initial value h_0 because it shares the key with the initiator *S*. Then *D* hashes h_0 k times to get the final value, where k is *D*'s position in the node list.

2.4.2.2 Ariadne route reply

If the target *D* determines that the request is valid, it returns a ROUTE REPLY packet (RREP) to the initiator *S*, containing eight fields:

<ROUTE REPLY, target, initiator, time interval, node list, MAC list, target MAC, key list>

The fields of target, initiator, time interval, node list, and MAC list are set to the corresponding values from the RREQ, the "target MAC" is set to a MAC computed on the preceding fields in the reply with the key K_{DS}, and the "key list" is initialized to be empty [43]. Then the RREP is returned to *S* along the reserved source route in the "node list" of the packet. After receiving the RREP, the node waits the time specified in the "time interval" of the packet to disclose its key. Then it appends its key to the "key list" and forwards the RREP according to the source route indicated in the packet. When *S* receives the RREP, it verifies that each key in the "key list", the "target MAC", and each MAC in the "MAC list" are valid. If every test succeeds, *S* accepts the RREP; otherwise, *S* discards it.

2.4.2.3 Ariadne route maintenance

In order to prevent the injection of invalid ROUTE ERROR packets (RRER) into the network fabricated by any node other than the one that encounters a broken link, the Tesla authentication information is added into the RRER. Therefore, any node on the return route of a RRER can authenticate the error. However, all the nodes on the return route have to buffer the packet first because of the delayed disclosure of the Tesla key. Later, when the node that encountered the broken link discloses the key and sends it over the return route, these nodes on that route can authenticate their buffered packets.

2.5 Service advertisement and discovery

A routing protocol helps a source find a route to some destination. The assumption for each protocol is that the source knows the destination name or address correctly. However in ad hoc networks when nodes meet accidentally as well as transiently this assumption is not realistic. Without some help it is difficult to acquire such useful information in advance. Service discovery technologies are therefore developed to reduce the difficulty when dealing with the configuration and interaction of a large number of nodes. Service discovery technologies commit themselves to simplify the use of mobile devices in a network by allowing them to be discovered, configured, and used by other devices with a minimum of manual effort [45].

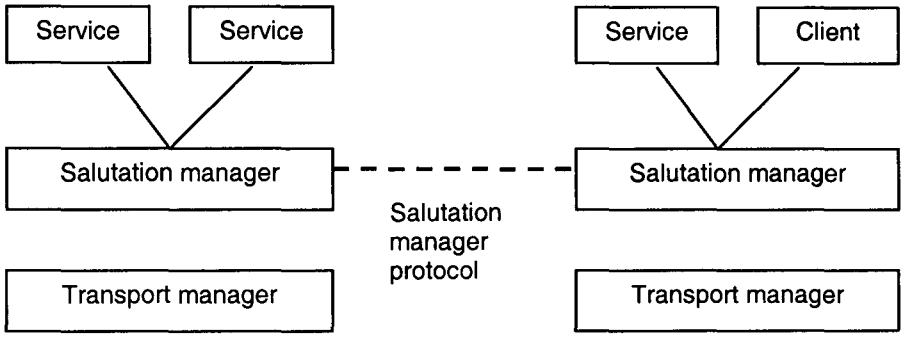


Figure 2.13: Salutation architecture

2.5.1 Salutation

Salutation [46] is a coordination framework. It comprises three fundamental components: functional units, salutation managers, and transport managers (see figure 2.13). The functional unit defines a service such as printing, faxing, or document storage. The Salutation manager is the core of the architecture functioning as a service broker. The transport manager providing reliable communication channels isolates the

Salutation manager from the underlying network transports. The interface between the Salutation manager and the transport manager achieves transport protocol independence, through which the Salutation manager performs the following tasks including service registration, service discovery, service availability, and service session management. The Salutation manager provides an interface to both server and client applications. A service provider registers its service in a local Salutation manager. When a client asks its local Salutation manager for a particular service, all the Salutation managers cooperate to perform the search. Then the result is returned to the client.

2.5.2 Universal plug and play

"Universal Plug and Play (UPnP) [47] is a suite of protocols and system services for device discovery and control in small to medium size IP networks [48]". The key components are service, device, and control point (see figure 2.14), which eXtensible Markup language (XML) is used to describe device's properties and services. UPnP applies Simple Service Discovery Protocol (SSDP) through which a device is able to announce its presence to the network or discover other available devices. SSDP uses Hyper Text Transfer Protocol (HTTP) over both multicast and unicast UDP. The service announcement message contains a Uniform Resource Identifier (URI) that identifies the resource as well as a Uniform Resource Locator (URL) pointing to the XML file that describes the announcing device. Once a control point has discovered a device and obtained the related URL for the service, it learns more about how to use, control, and coordinate with the device by retrieving its XML description file. Specifically, "control is expressed as a collection of Simple Object Access Protocol (SOAP) objects and their URLs are stored in the XML file. The description for a

service includes a list of actions to which the service responds and a list of state variables that model the service at runtime [49]". When these variables change, the service publishes updates by sending event messages to a control point which has subscribed to this type of information. "These messages are also expressed in XML and formatted using the General Event Notification Architecture [49]".

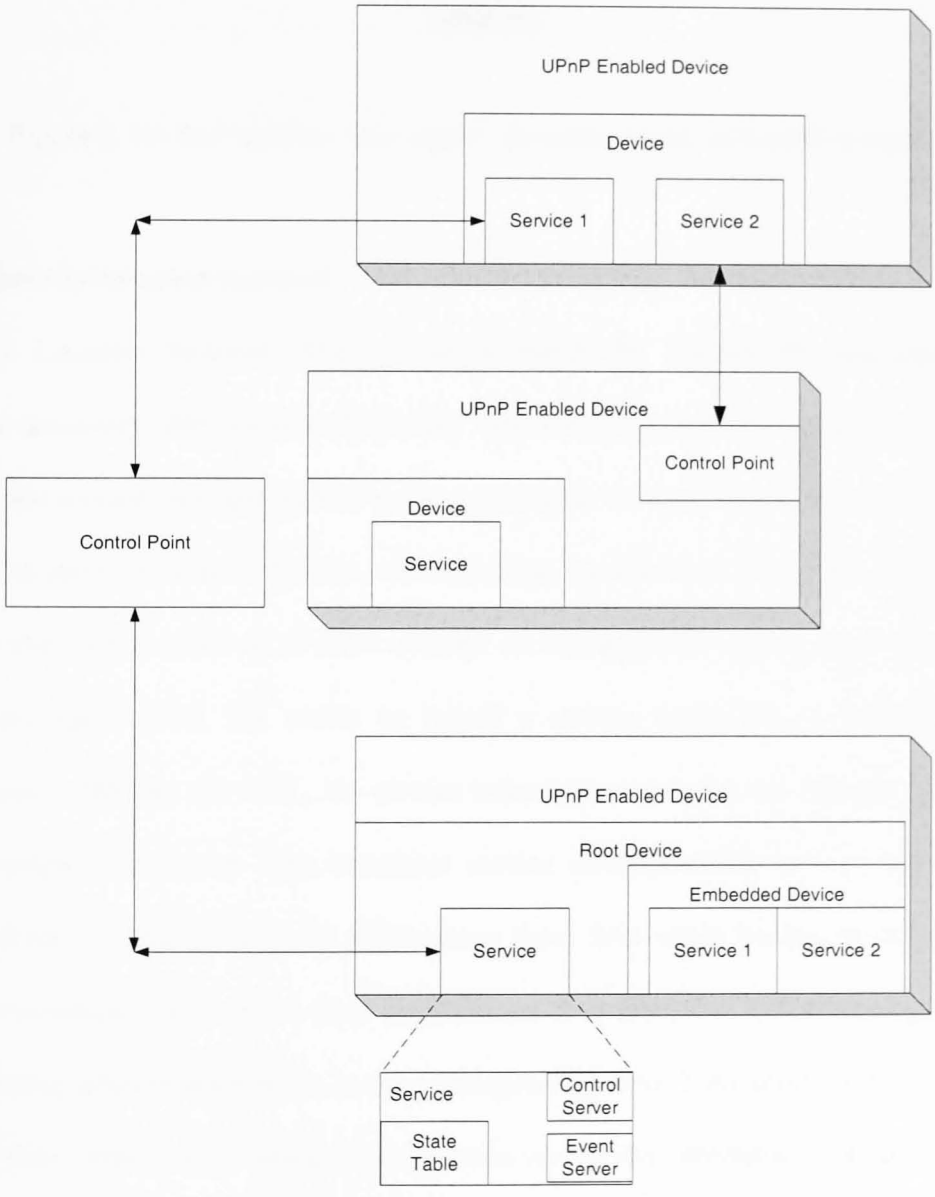


Figure 2.14: UPnP control point, device, and service

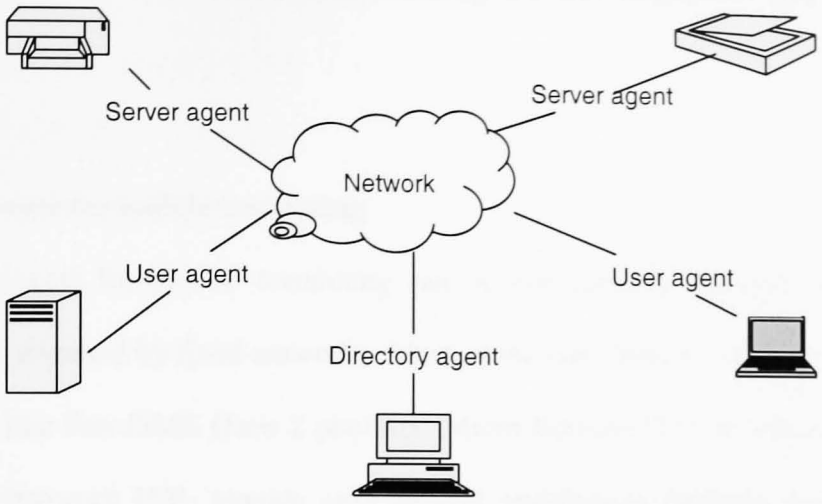


Figure 2.15: SLP entities: user agent, directory agent, and service agent

2.5.3 Service location protocol

Service Location Protocol (SLP) is for decentralized, lightweight, and extensible service discovery [50]. Service URLs like "service:printer:lpr://hostname" are used to define the service type and addresses for a particular service. Hence, users can browse available services in their domain, select and use the one they want [49]. Figure 2.15 shows that SLP is made up of three entities: service agent (SA), user agent (UA), and directory agent (DA). SA works on behalf a service while UA on behalf a user application. Besides the URL, the service information includes the lifetime and a set of descriptive attributes. SAs broadcast service advertisements and try registering themselves to DAs. DAs cache information from SAs while process requests from UAs. Periodically SAs renew their registrations with the DAs, which are responsible for sending acknowledgements to the corresponding SAs. UAs send service requests to the DAs, which apply string-based queries for service attributes. It leads the DAs reply the most appropriate services from available services in the network. Then a UA is able to access the service pointed to by the returned URL. In SLP, DA is optional.

Without the DAs, the SAs that control matching services respond to UAs' requests directly.

2.6 Middleware for mobile computing

The requirements for mobile computing can be considerably different from their counterparts imposed by fixed networks. Most of the commercial systems for mobile computing, like Sun J2ME (Java 2 platform, Micro Edition) [51] or Microsoft .NET compact Framework [52], provide only general middleware facilities leaving it to applications themselves to handle all the services and deal with the non-functional requirements. On the other hand, more and more research middleware targeting the needs of mobile computing have been devised. This section sheds some light on the different types of mobile middleware technologies.

2.6.1 J2ME

J2ME is a development package targeting a broad range of consumer devices and embedded devices [51]. The architecture of J2ME is designed to be modular and scalable in order to support the inherent diversity of these devices. In the term of J2ME, the "configuration" is chosen to specify a Java virtual machine as well as a minimum set of Java classes for each class of devices. Currently, there are two J2ME configurations: Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). The smaller CLDC is designed for low-end devices with less than 512 kilobytes memory while CDC targets more powerful devices normally having at least two megabytes memory. A configuration must work together with a "profile" which specifies a narrower category of devices within the chosen configuration. The profile is implemented on top of the configuration. For example,

the combination of CLDC with Mobile Information Device Profile (MIDP) is to provide a complete Java application environment for mobile phones and other devices with similar capabilities. In addition, optional packages can be added to the J2ME platform so that developers are able to support more technologies as they wish.

2.6.2 Request/reply middleware for mobile computing

Traditional middleware implementing synchronous semantics, like RPC or object-oriented CORBA, has been adapted to mobile environments mainly supporting mobile devices interoperating with existing fixed networks.

ALICE is an architecture that allows "CORBA objects running on mobile devices to transparently interact with objects hosted by off-the-shelf CORBA implementations [53]". In ALICE, a mobile device needs to connect to a mobile gateway, which redirects the connection to a fixed host. From the system view, it requires mobility layer and swizzling IIOP (S/IIOP) layer running on mobile devices and mobile gateways to be implemented between TCP/IP layer and IIOP layer. The mobility layer, on behalf of the upper layer, mainly handles with TCP issues such as performing reconnection if a TCP link is broken or allocating a TCP port on a mobile gateway. The S/IIOP layer performs operations which are affected by mobility, especially publication and encoding of object references [53]. Therefore server as well as client objects can work on mobile devices without relying on any centralized administration mechanism. However the main challenge in this direction is "in terms of software size and protocol suitability" [54], as it originated from the distributed systems for fixed networks and depends on synchronous communication mechanisms.

More attempts have been made in the direction of applying traditional middleware in a sort of semi-asynchronous paradigm [54]. Rover is such an example: it combines

"re-locatable dynamic objects" and "queued remote procedure calls" to provide unique services for "mobile-transparent" or "mobile-aware" applications [55].

2.6.3 Message-oriented middleware for mobile computing

Message-oriented middleware (MOM) supports communication between distributed components via message-passing and works asynchronously. Message senders and receivers are loosely coupled through identified message queues. A sender sends a message to a queue that acts as a temporary storage. Once the middleware has confirmed the acceptance of the message the sender is able to continue processing its local tasks. The receiver retrieves the message from the queue at some different time and may send a reply applying the same queuing mechanism. Sun Java Message Service (JMS) [56], IBM WebSphere MQ [57], and so forth are examples of successful MOM for traditional distributed systems.

Pronto [58], a JMS messaging system, is specifically designed for mobile applications working in both centralized and decentralized forms. Like ALICE, JMS servers are assumed running in a wired network and applications on mobile devices (clients) may connect the servers through gateways. The optimized, lightweight mobile JMS client is an adaptation of JMS specification, which supports the necessary functions in mobile environments. The gateway controls message traffic; each plug-in component like SmartCaching implements a specific task in order to improve the transport efficiency with respect to varied application scenarios. The publish-subscribe paradigm is applied providing greater flexibility and scalability as the decoupling of publisher and subscriber components are able to exchange messages under complex conditions like frequent disconnection or changing numbers of recipients. If the system works in the decentralized way, the mobile JMS client will be replaced by the

severless JMS, using IP multicast as the transport mechanism. A publisher acts as a temporary server and keeps a subscription list [58].

Pronto claims that it could be deployed in MANET, however "it does not adequately target ad hoc settings, since mobile nodes rely on fixed servers for the exchange of message" [59]; or from programming perspective, Java Remote Method Invocation (RMI) does not work well if a client connects to a server via some relay nodes. Actually most MOM implementations for mobile environment are usually straightforward extensions of existing middleware, the very first specifically designed and implemented for MANET is [60]. E. Vollset et al comprehensively identify and address the design requirements encountered in practical settings. *Jomp*, an implementation of ODRMP in application layer, a transport module focusing on group communication, is integrated into the existing Arjuna JMS [61] service. The publish-subscribe semantics is implemented by mapping JMS topics to multicast addresses. However it does not consider the situation when a destination is out of the network temporarily. Then it leaves supporting persistent messages delivery as the future work.

EMMA (Epidemic Messaging Middleware for Ad hoc networks) is one of the latest adaptations of JMS for MANET [62, 63]. Based on the context-aware routing protocol [64] (an instance of epidemic routing protocol [65]), the embedded persistence mechanism in EMMA ensures the information dissemination in a network in which partition happens frequently. Each node maintains a limited size buffer containing the messages that it generates and receives from others. When two nodes encounter, the messages needed to be sent are duplicated on each side through a so called anti-entropy session [62]. The process is like an infection. Persistent messages have high priority over non-persistent ones; low-priority messages will be deleted if

there is insufficient room in the buffer. In addition, EMMA provides an efficient acknowledgement mechanism. If the acknowledgment is not received within a given timeout, the failure will be notified to the sender. If the acknowledgment is for a persistent message, the success will be notified to the sender. At the same time, the acknowledgement is also used to signal to delete the delivered messages that could be still stored in some nodes' buffers in the network.

2.6.4 Tuple space based middleware for mobile computing

Tuple space based middleware is a completely asynchronous and decoupled message-passing paradigm: a sender need not know the location of the receiver and its availability. It has been proved effective in mobile settings, though not initially designed for this purpose (its origin goes back to Linda [66]). Generally, tuples, the elementary data structures, are vector of typed values or fields. A tuple space works as a repository of tuples. It is globally shared, associatively addressed and persistent memory space that is used by processes to communicate [54]. Processes create tuples and put them in the tuple space using *write* primitive; also they can concurrently access tuples using *read* or *take* primitive [67]. There are two tuple retrieval primitives: *blocking* and *non-blocking*. During a retrieval operation, the results are selected through pattern matching on the tuple space. Tuples' lifetime is independent from the processes that create them.

LIME [68] is a typical tuple space based middleware. It inherits and adapts the communication model proposed by Linda. The shift from a fixed context to a mobile one is accomplished by dividing Linda tuple space into many so-called interface tuple spaces (ITSs). The ITS contains tuples that a mobile unit wishes to share with other mobile units; the ITS is permanently and exclusively bound to the mobile unit. Each

mobile unit can only access the global context via its own ITS that travels with it when movement happens. Access to ITS takes place using Linda primitives. Upon a new mobile unit arrives, a sequence of operations, named "engagement", is performed to merge the tuples in new unit's ITS with those, already shared, belonging to other units. In addition, LIME extends Linda operations by introducing the "tuple location" parameter to *take* primitive and a notion of "reaction".

2.6.5 Data-sharing middleware in mobile computing

One of the major issues in mobile computing is the support for disconnected operations and provision of data-sharing. Many proposed systems, like Coda [13], Odyssey [69], and Bayou [70], have made data-sharing their sole task and tried to maximise availability of data, giving users access to replicas; they differ in the way that they ensure replicas move towards eventual consistency. Despite of many proprietary data synchronisation protocols for mobile devices, it still lacks a single standard so that it represents a limitation for both users and developers.

XMIDDLE [71] is an example of data-sharing middleware. Data are stored in a tree structured form; data-sharing is through sharing of trees. "Linking" to a tree is similar to mounting network file systems in distributed operating systems. Hosts can duplicate the trees and continue modifying the off-line data when disconnections happen. When two hosts reconnect, XMIDDLE exploits tree differencing techniques to detect the differences between the possibly conflicting replicas. Because different operations may have been performed on the same node of the tree, XMIDDLE cannot complete the reconciliation task alone. Then the task is handed over to the application developer who defines application-specific conflict resolution policies to each node of the tree. When a conflict is detected, XMIDDLE will apply the proper policy in order

to successfully complete the merging procedure. The significant contribution of XMIDDLE is to address working in pure ad hoc networks, "as no assumption is made about the existence of more powerful and trusted hosts which should play the role of servers and on which a collection of data should be replicated in full [54]".

2.6.6 Other middleware solutions

Other distinguishing classes of middleware for mobile computing, for example, reflective middleware [72], context-aware middleware [14], and event-based middleware [73], each of which attempts to address mobile architectural requirements using different approaches, have been identified, illustrated and comparatively discussed in [54, 67]. They bring more heuristic ideas to researchers, like the aim of reflective middleware is "to make the middleware more adaptable to its environment and better able to cope with changes [67]".

The latest development of reflective middleware is Web Services based ReMMoC (Reflective Middleware for Mobile Computing) [74]. ReMMoC is independent from particular discovery and interaction protocols; it provides a minimal set of functionalities and extra components each for a specific service type implementation can be added when required. Services are inspected and adapted at run-time: the system monitors the environment and the service types in use and reconfigures itself to mirror the current setup.

The event-based communication paradigm is a possible alternative for dealing with large-scale systems [75]. Events contain data that describe a request or message. In order to receive events, clients (subscribers) have to express (subscribe) their interests in receiving particular events. Servers (publishers) publish events that will be sent to all interested subscribers. A typical example of event-based middleware is HERMES

[73], in which a content-based algorithm [76] is applied to deliver events from publishers to subscribers.

2.7 Mobile applications

In this section, the evolution of wireless computing paradigm is reviewed and the examples in various settings are surveyed.

2.7.1 Nomadic computing

Nomadic computing refers to a transparent, virtual networking environment where users can access computing and communication facilities not only from their home bases but also while they are in transit and when they reach their new destinations [77]. The notion was presented when wires or cables dominated the whole network settings but users became familiar with wireless communication: it could help users get rid of controls from the cable-enabling networking. However the level of wireless connectivity often included extended periods of low bandwidth or no communication at all. The goal of the research is precisely to permit users and programmes to be as effective as possible in this environment of uncertain connectivity without changes to their manners in which they operate [77]. One of basic solution to support nomadic computing is Mobile IP [78].

2.7.2 Ubiquitous computing

Mark Weiser envisioned the era of ubiquitous computing [79], which forces computers to be effectively invisible and computers with communication interfaces are embedded in devices of everyday use. Human beings are enveloped in such an immersive environment; they are able to adapt to the environment by interacting with

these ever-present devices that are at background for anything at anytime. Consequently, information is integrated with human beings' everyday physical world. Work on ubiquitous computing is still in the very beginning stage. Ad hoc networking is just one of its characteristic; spontaneous interaction can take place transparently to human users and automatically help them fulfil some tasks.

The DEAPspace project proposes and implements a framework applying ad hoc networking technologies in order to support a class of pervasive application, which features hidden computing and spontaneous interactions between proximity-based mobile devices [80]. Some new application scenarios are described in detail so that the requirements for the framework can be derived. The authors elaborate the key concepts of the system: the service model, compact and efficient service description and matchmaking. The core contribution is a prompt, efficient and push-based service discovery protocol in transient, limited size ad hoc networks. The work could be regarded as a case study for general service discovery protocols described in section 2.4. Interestingly, the framework could work in wired networks as it provides the abstract interfaces shielding applications from the concrete networks.

The Cooltown project [81] targets the intersection of Web technologies, wireless communication technologies, handheld devices, and small electronic appliances. An infrastructure supporting location-aware but ubiquitous applications for nomadic users without a central point has been set up. According to Cooltown, every physical entity in the world could fall into three categories: people, things, and places. The Cooltown team argues that Web technologies are the best middleware for building the connections between people and things or places in their everyday environments. URLs (Universal Resource Locators) and/or web servers are embedded into things and places. Using URLs for addressing, physical URL beaconing and sensing of

URLs for discovery [81], people's mobile devices can interact with things and places. Therefore, "Web presence" for people, things, and places can be implemented.

2.7.3 Applications in pure ad hoc networks

In the past few years a great member of novel applications have emerged as the use of wireless communication becomes a part of people's daily life.

The Swiss watch-maker Swatch has put "synchro.beat" [82] on the market, which allows the bearer to interact with some other people in the real world via sound transmission. Or they could communicate with each other through the virtual world: World Wide Web. The watch becomes the key to enter the company's specially designed entertaining web site. After logging in the web site a bearer should find out how compatible s/he is with others by taking the compatible test (c-test). Then the bearer could find his/her compatible pals when s/he is shopping in a mall, walking on street or anywhere in which another bearer is close to him/her. Two watches are able to implicitly exchange c-tests with each other and display results on the screens. By this way the bearers build up their buddy lists and could play games or have other fun when they meet at that particular web site simultaneously.

The TunA project proposes an application supporting music-sharing when people close to each other form a social network [83]. The music is on the handheld device like PDA or Apple iPod with wireless interface. It creates the opportunity to enjoy what others around are listening and synchronize the favourite music. It allows users to share music in many locations, fostering a sense of awareness of the surrounding physical environment while they move around [83].

CaféTrek is an ad hoc network game for supporting face-to-face interaction in public places [84]. The project uses the mobile device as a trigger to notify its user who

should interact face-to-face with another one close by. Computer Mediated Communication (CMC) technique is applied in the design. CaféTrek is specifically targeted for public places where most people do not know each other. The implemented tools can encourage a social encounter to a peer-to-peer (P2P) CMC session then facilitate face-to-face interaction based on information entered in the CMC session.

The applications mentioned above are only working in one hop range. It means that mobile nodes are close each other so that their wireless interface can connect directly. Thus the applications cannot cover large space and attract more potential users. Researchers have addressed the challenge and proposed some prototypes supporting multi-hop communication among nodes in an ad hoc network.

P. Poupyrev et al report the performance of a real-world application "*Whizbe*" running on PDAs [85], where packets are broadcast in a large scale two hops ad hoc network. The main goal of *Whizbe* is to inform users when their friends or acquaintances are in close physical proximity. Users have no other contact except broadcasting to exchange the personal profile. The empirical study setting in a static one hop ad hoc network involves one laptop as the master and up to nine PDAs as clients. The experimental data shows that "there is significant intrinsic delay and fluctuation in transmission times even for rebroadcast by a single PDA [85]". The performance of the application is strongly affected when many PDAs are contending for rebroadcast on the wireless channel. Carefully analysing the test data the authors deduce that probability-based control should allow adequate performance. Based on these results and comprehensive analysis, the time to complete broadcasting of one packet in a large scale two hops network (1500 nodes) can be estimated: it varies from two minutes to nine minutes depending on the algorithm used. The result may be adequate

for this specific application but would be too slow for many other applications. In the worst situation a user may get response after 18 minutes when s/he sent off an urgent request.

H. Ritter et al present a system with the help of "roaming infrastructure" to support a multiplayer game in MANET [86]. The tiny hardware is named "Embedded Web Server" (EWS) and could work as a web server, which equips Bluetooth, 433 MHz Radio Frequency (RF) and Ethernet interface. Bluetooth enables players' devices to form an ad hoc network. Subsequently a multiplayer game is set up spontaneously and gaming data are transported. The RF technology is adopted to maintain a consistent global view of the game, track players' movement and position (by measuring the link quality) and exchanges packets between different piconets, since its working range is about 20 meters that is about one time more than Bluetooth. EWS could be set up quickly in ad hoc fashion and helps overcome strong restrictions of Bluetooth, such as its transmission range, the size of one piconet and the formation of a scatternet. EWSs are distributed into the network to act as the master of each piconet, from which players' devices could access the whole network. On the other hand EWSs are roughly acting as roaming service-providers or special access points, which are connected by specific wireless communication instead of reliable wired cable. Nonetheless for the limitation of Bluetooth the formation of an ad hoc network is time-consuming inquiry process and it leads to long handover time of about 1.43 seconds (the time that EWSs need to detect a player roaming from one piconet to another). This handover times restrict the types of games that the architecture is suitable for.

Dawn IM system brings the functionality inherent in wired network Instant Messaging (IM) system to pure ad hoc environments: facilitating spontaneous chats between a

pair or a group of mobile hosts in a small-scale network without any centralized server's help [87]. It adopts a decentralized P2P approach to locate the session peer. Every host manages its own status information and is responsible for informing others of its present state. The status information is distributed following the "fetching" model described in RFC-2778 [88]. Each host uses a "group-based" approach to manage chatting sessions; one protocol supports both unicast-style and multicast-style message exchange. A similar but simplified work could be found here [89].

2.8 Network simulators and simulation studies

Analytically quantifying the performance and complex behaviour of even a simple mobile application often yields inaccurate results due to a large number of simplifying assumptions that need to be made. Furthermore, performing actual experiments is onerous: acquiring hundreds of devices, managing their software and configuration, controlling a distributed experiment and aggregating the data, and so forth. These difficulties make empirical endeavours daunting. Consequently, a vast majority of research in the area of ad hoc network performance evaluation is based entirely on simulation, a fact that underscores the critical role of efficient simulators. Currently the two popular "discrete event" simulators in the wireless research area are ns-2 [90] and GloMoSim [91], of which the latter will be introduced in chapter 5 since it has been chosen to conduct simulation studies for this work.

2.8.1 Ns-2

The widely used ns-2 simulator was originally developed for simulation of TCP and routing protocols over wired networks. Then the simulator was extended in order to support accurate simulation of the physical layer aspects with the related MAC

protocols of multi-hop wireless ad hoc networks. These extensions can simulate the physical and link layer behaviour of a wireless network and allow arbitrary movement of nodes within the network. Some of the proposed routing protocols, for example, DSDV, TORA, DSR and AODV, have been integrated into ns-2. Each run of the simulator accepts as an input scenario file that describes the exact motion of each mobile node together with the sequence of packets originated by each node as time progresses [92].

2.8.2 Simulation studies

Simulators are adopted to focus on a detailed simulation of the OSI reference model from layer 1 to 4.

J. Broch et al present the results of a detailed packet-level simulation comparing four unicast routing protocols: DSDV, TORA, DSR, and AODV [93]. The performance of each protocol is measured under a series of changing conditions while the direct and fair comparison between the protocols are applied the identical traffic payloads and environmental conditions. Each ns-2 simulation run involves 50 mobile nodes to form the ad hoc network. These nodes move around in a rectangular area that is 1500 meters long by 300 meters wide ([1500 m * 300 m]) for 900 seconds of simulated time. Three metrics are chosen to evaluate the protocols: packet delivery ratio, routing overhead, and path optimality. The simulation results show that all the four protocols successfully deliver a great percentage of the data packets to their destinations when node mobility is very low. DSR and AODV perform particularly well, delivering over 95% of the data packets regardless of mobility rate [93]. With regard to overhead, the difference between the best and poorest one, which is DSR and TORA respectively, is almost an order of magnitude. For the three on-demand protocols, TORA, DSR, and

AODV, their overhead drops as their mobility rate drops. However, the overhead of DSDV is not affected by node mobility rate and is constant, as the protocol periodically broadcasts the routing updates. About path optimality, both DSDV and DSR take routes very close to optimal. In contrast, TORA and AODV show a trend of taking longer paths than optimal for some packets. Generally, "the performance of DSR was very good at all mobility rates and movement speeds, although its use of source routing increases the number of routing overhead bytes required by the protocol [93]".

Applying the similar simulator setting and methods as described above, S. R. Das et al focus on the performance comparison of two prominent on-demand routing protocols: DSR and AODV [94]. A new field configuration is introduced into the simulation study: [2200m * 600m] field with 100 nodes. Under such circumstance, however, the simulated time has to be reduced to 500 seconds because of resource constraint on hardware. With regard to application oriented metrics like end-to-end delay and throughput, the simulation results show that DSR outperforms AODV when node mobility is low, or there is smaller number of nodes sending data packets; AODV outperforms DSR under the "stress" situation, for instance, high mobility or more traffic payload. The performance gap becomes wide when the stress increases. However, DSR consistently generates less routing traffic than AODV. Furthermore, the simulation study demonstrates that the interoperation between the network layer and MAC layer could also significantly affect the protocol's performance. For example, more unicast routing packets like route request or route reply generated by DSR are expensive in the IEEE 802.11 MAC layer. Therefore, DSR's "apparent" savings on routing overhead does not represent an expected reduction of the traffic payload into the network.

T. Kunz et al present simulation results regarding the comparison of two on-demand multicast routing protocols: AODV and ODRMP [95]. The simulator is configured as follow: 50 nodes move around in a [1000 m x 1000 m] square space for 900 seconds of simulated time. The performance metrics collected are delivery ratio and routing overhead. A series of experiments are performed with regard to varying parameters: number of senders, node mobility, and multicast group size. Overall, AODV shows a poor packet delivery ratio compared to ODMRP. On the other hand, ODMRP suffers from scalability issues as there are more members or senders in the multicast group. It verifies similar results that "mesh-based protocols outperformed tree-based protocols and the availability of alternate routes provided robustness [96]".

2.9 Summary

This chapter has broadly reviewed ad hoc networking technologies and supporting software systems covering MAC protocols, routing protocols and security issues, service discovery standards, middleware, and wireless network simulators. The main aim of the survey is to help readers evaluate the strength and weakness of various approaches and to gain enough background knowledge to understand the requirements for designing middleware for mobile systems.

Chapter 3 Convenience Network: concepts and applications

With the soaring popularity of mobile devices and evolving ad hoc networking technologies, new ways for people to interact with their neighbourhood environments are emerging [97, 98]. These enabling technologies have potentials to facilitate as well as extend human interaction capabilities with ease, flexibility, and spontaneity. This chapter begins with a brief review of applications in MANET. This is followed by the motivation behind the notion of "Convenience Network" (CN) which is able to facilitate spontaneous interactions among a group of people in a specific area. The "local shopping centre" scenario is detailed and then the characteristics of CN are described. Three example applications, namely "Auction" [99], "Bingo game", and "Chatting", are introduced. These will be chosen as the basis of empirical studies for determining system requirements for CNs in chapter 4. This chapter concludes with the description of characteristics of applications in CNs.

3.1 Applications in MANET

Early MANET applications and deployments have been military oriented [100]. Since then, especially in the past a few years, interest in MANET applications in other areas has grown due to the soaring popularity of mobile devices. MANET applications have advanced substantially and expanded into specialized fields such as education [101], entertainment [102], vehicle service [103], and so forth. Among them, a sensor network is a unique ad hoc network, which consists of hundreds or thousands of tiny nodes. These low-cost, low-power, multi-functional devices may be spread across a geographical area to monitor changing conditions at different locations. A wide range

of public and commercial applications adopting sensor network technology have been deployed, including environment monitoring [104], traffic control [105], safety surveillance [106], health care [107], and so forth. Based on various application scenarios, I. Chlamtac et al present a categorization of present and possible future applications in MANET as well as the services they may provide in each area [100], which is shown in table 3.1.

Applications	Descriptions/services
Tactical Networks	• Military communication, operations • Automated Battlefields
Sensor Networks	• Home applications: smart sensor nodes and actuators can be buried in Appliances to allow end users to manage home devices locally and remotely • Environmental applications include tracking the movements of animals (e.g., birds and insects), chemical/ biological detection, precision agriculture, etc. • Tracking data highly correlated in time and space, e.g., remote sensors for weather, earth activities
Emergency Services	• Search and rescue operations, as well as disaster recovery; e.g., early retrieval and transmission of patient data (record, status, diagnosis) from/to the hospital • Replacement of a fixed infrastructure in case of earthquakes, hurricanes, fire etc.
Commercial Environments	• E-Commerce: e.g., Electronic payments from anywhere (i.e., taxi) • Business: o dynamic access to customer files stored in a central location on the fly o provide consistent databases for all agents o mobile office • Vehicular Services: o transmission of news, road condition, weather, music o local ad hoc network with nearby vehicles for road/accident guidance
Home and Enterprise Networking	• Home/Office Wireless Networking (WLAN) e.g., shared whiteboard application; use PDA to print anywhere; trade shows • Personal Area Network (PAN)
Educational applications	• Setup virtual classrooms or conference rooms • Setup ad hoc communication during conferences, meetings, or lectures
Entertainment	• Multi-user games • Robotic pets • Outdoor Internet access
Location aware services	• Follow-on services, e.g., automatic call-forwarding, transmission of the actual workspace to the current location • Information services o push, e.g., advertise location specific service, like gas stations o pull, e.g., location dependent travel guide; services (printer, fax, phone, server, gas stations) availability information

Table 3.1: Applications in MANET [100]

A class of applications or services that falls under home and enterprise networking, educational applications, and entertainment applications is considered in this thesis. Roughly, these applications provide services in populous public places and are

entertainment-oriented. These applications reflect the aspirations of mobile device users to probe their vicinities in order to look for user-specific information expediently; a facility needs to be formed to help them accomplish the goal.

3.2 Motivation for Convenience Network

The idea of CN has its origin in the notion of "Spontaneous Network" [97] or "Spontaneous Information System (SIS)" [98]. "Spontaneous networking is the integration of services and devices into network environments with the objective of instantaneous service availability without any manual intervention [108]". R. Liscano et al claim "spontaneous networking can be considered as an application approach to ad hoc networking [109]". The advantage of spontaneous networking is to facilitate knowledge sharing through informal interactions since two mobile entities can exchange information at least if they are close enough. For instance, figure 3.1 shows SNiF (Social Networking in Fur), a system which allows pet owners to interact through their pets' social networks [110]. There have been many workplace studies that justify this claim [111].

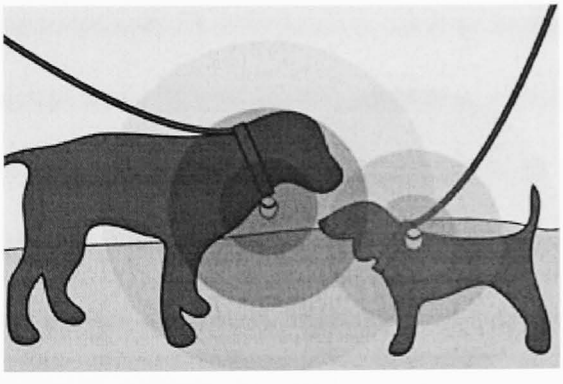


Figure 3.1: Mood ring meets messaging: A new digital dog collar aims to help owners match with their dogs' favourite canine pals. [110]

The name "Convenience Network" captures the activities of wireless-ready mobile device users who form an "informal" system when they gather in public places such as supermarket, airport lounge, and so forth. The following local shopping centre scenario describes the motivation behind CN.

The management of local shopping centre has introduced a Customer Message Board (CMB) service, which allows customers to advertise free for some time. These advertisements may cover many subjects including: second-hand goods to sell, flat letting or house-sharing, part-time job required, professional service provided, local events, and so forth. Everyday a great number of people shop in a local shopping centre at all time of day. They have the common sense of knowing that they as customers are welcome to post unsolicited messages on the CMB. There is always a large amount of various kinds of information advertised on the CMB. Everyday new items could be posted while some old ones are withdrawn. Actually, people could withdraw their items at any time before they expire, for instance, the goods have been sold. Besides shopping, sometimes customers go to a local shopping centre in order to look for some useful information on the CMB. Under such circumstances, information providers and customers are connected by the CMB which operates as the information rendezvous. Regarding some specific information, such as looking for a flat in a local area, it is more convenient and efficient for flat-seekers to check the CMB first rather than look for information in newspaper classified fields or yellow pages, since landlords understand that advertising on the CMB is very effective.

However, there is an inherent limitation with regard to advertising or finding out "convenient" information in a local shopping centre. Firstly, the main purpose of customers is shopping. During their shopping periods they may advertise or look for useful information since they know of the existence of CMB, although they may not

exactly know its location. However, shoppers have to pass by a CMB so that they are able to post or browse announcements on it. Shoppers are unable to obtain any detail even if they are fairly close to the CMB, for example, they are in the next row of the CMB. Considering the convenience aspect, shoppers would not like to knowingly approach CMB and then stop browsing for a while. Instead, they prefer to advertise or search information during their shopping intervals. Under some particular circumstances, for instance, when shoppers have a break in a café or wait in long queues, it is then an appropriate occasion for them to make use of the CMB. Unfortunately, at these moments, shoppers are actually in an inconvenient situation because they are unable to reach any CMB. Secondly, shoppers could forget to browse the CMB although they intended to do so. On returning home, they would suddenly realize that they have forgotten one of the important tasks on their calendars. Finally, and the most unfortunately, shoppers could still miss some interesting information when they browse the CMB, because they may be in a rush to leave, or the scripts on posters are too small or scratchy to be readable, or they ignore the potential posters that are on the very top or bottom of the CMB.

3.3 Enabling Convenience Network

The inconvenience described above could be overcome if ad hoc networking is enabled and the process of advertising or browsing on the CMB is computerized. It is feasible if "the Spontaneous Network is created as soon as, at least, two mobile entities approach each other [98]". Afterwards, two entities have the ability to establish communication, and exchange information directly if they are close enough. If every device is also able to act as a message relay (router) then the service provision can be extend to a larger area. Given the context of local shopping centre, if

shoppers carry wireless-ready mobile devices (at least one of network interfaces works in "ad hoc" mode), a powerful facility, CN, can be made available. CN will help the serving participants accomplish their favourite tasks comfortably.

The local shopping centre could replace the CMB with an electronic message board (EMB) as the information rendezvous (server), on which various kinds of information may be categorized and advertised periodically. A CN can be formed when passing-by shoppers' mobile devices (clients) are interconnected. Some clients should directly reach the server. Hence, others are able to access to the server so long as packets are relayed between them. The information stored in the server is sourced either from the server or clients, since the clients are permitted to publish their announcements on the server. Initially, clients look for information by finding the server that bears a well-known name; the procedure is equivalent to shoppers' approaching the CMB. Once the server receives the clients' queries successfully, it responds to them with their demanded information. Actually, the server periodically advertises information to the whole network. The clients could just switch on their devices then wait for the information. However, it is not necessary that the server stores all detailed information. Some of information could be held by reference, which means the server knows from where it can be retrieved. In any case, the server can supply the clients with the detailed information. Therefore, CN can bring much flexibility to their users. For instance, given local shopping centre scenario, at least users do not have to stand exactly in front of the CMB to browse. Instead, these users could sit on chairs and browse their PDAs in a café at ease inside the shopping centre. Moreover, CN is an ideal platform that can help users spontaneously exchange any useful information with the help of the server. The server could be a fixed machine which is set up by business operator or a mobile device whose owner is roaming inside the network. In

the latter situation, such a person would like to voluntarily share his/her time and services resources to others.

3.4 Characteristics of Convenience Network

In this section, the key characteristics of CN are summarized.

3.4.1 Network is not planned

In a given period, a CN could suffer arbitrary partitions and merges. Therefore, during this period, it is difficult to know both the physical boundary of the CN and the accurate number of users. (In MANET, packets may be relayed from a source to the destination. Under the circumstance, in other words, it could so say that these packets are delivered hop-by-hop.) Hop count could be roughly chosen as unit to measure the distance between two nodes in an ad hoc network or the diameter of an ad hoc network (the longest distance between any pair of nodes apart in the network). The physical distance of one hop is no longer than a node's wireless transmission range, which depends on the type of wireless communication and physical environment conditions. Normally, one hop is between 10 to 150 meters. Hence, within a few hops, a CN could cover a large enough area, like a meeting zone in a hotel, or an open waiting area in an airport. This would enable people who intend to explore their surroundings inside this area to interact with each other.

3.4.2 Mobile devices cannot be configured in advance

CN could be spontaneously formed anywhere, at any time, with any number of participants. Functional or non-functional information such as software environment on mobile device, mobile device's name, address and type of network interface, user's

profile, available service, cannot be determined ahead of time. As a result, it is not possible to configure users' devices in advance.

3.4.3 Users walk around in a Convenience Network

Users gather in public places forming a CN; in the network, they walk around in different ways under changing conditions. For instance, if a CN is formed in a mall, some users could be strolling aimlessly (their speed would be around 0.5 meters per second (m/s)) as well as some others hurrying about (their speed would be around 2 m/s). However, the users would not continue to walk aimlessly when they knew they were isolated from the network, for instance, if nothing was displayed on their mobile devices for a long while. After that moment, these users' walking behaviours will be related to the application; they will try particular paths purposefully until they reconnect to the network and continue to participate in the application. In contrast, users move around only occasionally if a CN is formed in an airport waiting lounge. Most of the time, these users keep relatively still. During these waiting periods, they could operate their devices interacting with some unknown people nearby from time to time. A user may move to a new place, such as a seat close to a kiosk after buying some snacks.

3.4.4 Users lack technical knowledge

Normally, users can operate intuitively. For instance, they switch on or off their devices, or follow the clear guidance displayed on the devices' screens. But users in general do not have a full understanding of technical details of their devices or computing jargon. Hence, many users are not able to configure their devices to the needs of an application.

3.5 Empirical studies

The local shopping centre scenario puts forward rich opportunities for various kinds of applications for the convenience of shoppers. Examples of such opportunities may include: shoppers could obtain the latest sale information; shoppers could sell their spare tickets during their shopping; sometimes shoppers could find some people to chat while they are resting in a café; sometimes shoppers could play games with others while waiting in queues. Not limited to shopping centre, CN could be actually formed in many public places, such as pubs, shopping malls, high street, lounges in airports or train stations. Applications, such as shared whiteboards, polling, chatting or multi-user gaming are ideal candidates for operation in such an informal environment.

In the following sub-sections, three representative applications, "Auction", "Bingo game", and "Chatting", and the related application scenarios will be described in order to illustrate practical values of CN. For instance, given the context of the local shopping centre scenario, for some shoppers, the auction is an effective and fast way selling their spare tickets; for some shoppers, it is easy to pass queuing time when playing a Bingo game; for some shoppers, it is fun to chat with others unknown but share same interests. Basically, the three applications are the extension of their counterparts in the wired world, as people like to run their favourite applications everywhere at any time. They provide similar functions as their popular counterparts but work purely in ad hoc fashion. The three application scenarios share similar communication patterns; the flow of information is determined by the specific interests. Additionally, the design issues of the three applications will also be summarized. (The details are described in chapter 4.) These issues are used to study the application-oriented requirements for CNs.

3.5.1 Auction

Internet auctions [112, 113] are a new form for the classic type of transaction (auction), which introduces variability into nearly all aspects of an exchange. The process is mystifying: venue is virtual, price is unpredictable, and auctioneers and bidders are strangers. Yet interest in this form of transaction has been intense. Some bidders become addicted to interested auctions. As someone anonymous stated: "it (the Internet auction) is fun, it is easy and, well, you get hooked [114]".

Given the local shopping centre scenario, a lady wanting to sell a spare concert ticket could choose to advertise on the EMB when she does shopping and then wait for calls at home. An alternative is to have a quick sale by auction when she is in the shopping centre, since she is confident that some other shoppers are interested purchasing the ticket. A spontaneous auction could be created when shoppers who carry ad-hoc-networking-enabled PDAs are close together; their devices could automatically explore the surroundings and directly establish communication, form a CN and exchange information implicitly and explicitly. The lady auctioneer (her device is the auction server) pushes auction information into the network, while others (their devices are auction clients) after receiving them relay until everyone knows what will be happening. Interested buyers inform the auctioneer so that they are entitled to bid when the auction starts. This is an instance of open cry auction, defined by the mode of communication in which all status information is conveyed globally to all participants. Buyers submit their bidding within a short duration after receiving the latest auction information; the process advances round by round until a bidder wins the auction. This type of auction may also happen in a social club. Consider a group of football fans who always meet there for drinking, chatting and betting. Someone has a spare home ticket and wants to sell it. Some others without the ticket would like

to get one. If their PDAs are all turned on, the seller's announcement should reach every PDA wherever in the club. Then an auction will start after a short while, which will be similar to the lady selling her concert ticket in the shopping centre.

3.5.1.1 Auction procedure

i. Advertising. The auctioneer periodically broadcasts auction information to the network. Anyone who receives the message relays it by rebroadcasting until the message reaches the maximum hop count. Relay nodes may receive the same message from their different neighbours but it is relayed only once.

ii. Registration. Interested buyers register the auction by sending a pseudonym to the auctioneer. When the first bidder has registered, the auctioneer begins counting down until the auction starts. During this period other buyers may continue to join in the auction; the auctioneer periodically broadcasts an incremental permission-bidder-list with the auction information. Some buyers may have to register again if their names are not found on the permission-bidder-list. The auctioneer builds up a bidder list.

iii. Bidding. Under rules the auction starts when the countdown is over. The auctioneer broadcasts progress packets to the network. Bidders may respond for all or some messages. During this period some bidders may leave.

iv. Ending. Under rules the auction ends. The auctioneer is responsible for broadcasting the result. Both sides of the deal contact each other in private.

3.5.1.2 Design issues for Auction

There are many issues relating to designing electronic auctions [115]. With respect to auctions in CNs, the main interest is *"fairness"*, which roughly means that all participants should be treated equally [116].

3.5.2 Bingo game

A survey by Nokia indicates that nearly 50 percent of the people asked like to play games while waiting [117]. The expectation is that CN will be of particular interest for many game scenarios; the users' mobile devices which are in sufficiently close proximity could be potential game devices, since these devices are able to communicate directly or via a limited number of relays without the need for further externally provided infrastructure. Imagine some travellers waiting for their trains at a railway station. They may sit in a lounge reading books or stroll around along the platform stretching their legs. The wireless-ready mobile devices of these travellers could form a CN. During their spare time a person (his/her device acting as the gaming server) volunteers to call for a Bingo game. The Bingo game model is described below.

B	I	N	G	O
2	17	44	48	74
3	23	33	47	68
7	27	**ANY**	59	66
10	30	39	53	75
12	18	43	55	70
	BINGO!			

Figure 3.2: A Bingo card

A Bingo card interface is shown in figure 3.2. Each card has five rows and five columns thus providing 25 spaces. The columns are labelled from left to right with the letters: "B", "I", "N", "G", and "O". With one exception (the centre space is "any") the spaces in the card are assigned values as follows: "B" column contains a number

from 1-15; "I" column contains a number from 16-30; "N" column contains a number from 31-45; "G" column contains a number from 46-60; "O" column contains a number from 61-75. Thus there are 75 possible Bingo ball numbers: B1,...B15, I16,...I30, N31,...O75; the number of unique Bingo cards is very large. In a short while gaming information from the server's device is put through to reach all persons in the network. Interested players send register messages back to the server and get permission to play. After a short time, the game starts and ends when someone wins the game. Players do not need to identify other participants or configure their devices in advance. During the gaming time they may leave without notifying anybody. The Bingo game model has three aspects that make it attractive in CN. These aspects are:

i. There is no limit to the number of players: as many as possible can join the game.

ii. The game does not last for a very long time. This is because announcing Bingo balls after a maximum of 75 times there will be a winner (however, normally it does not require 75 times).

iii. If some players leave during the game this does not affect any other players or the final result.

3.5.2.1 Bingo game procedure

i. Advertising. A Bingo server periodically broadcasts gaming information to the network. Anyone who receives the message relays it by rebroadcasting until the message reaches the maximum hop count. Relay nodes may receive the same message from their different neighbours but it is relayed only once.

ii. Registration. Interested players send registration information (player name and the Bingo card) to the server. When the first player has registered, the server begins a countdown until the game starts. During this period other players may

continue to join in the game and the server periodically broadcasts an incremental permission-player-list with gaming information. Some players may again send the registration information to the server if their names are not found on the permission-player-list. The server also maintains a local copy of player list.

iii. Playing Bingo. Under rules the game starts by announcing the Bingo ball. The server periodically broadcasts Bingo balls to the network. Players check their own cards against the balls announced and look for a BINGO (five in a row, column, or diagonal). During this period players may leave. When a player notices a BINGO, s/he declares a win by sending the player name and the winning card.

iv. Ending the game. The server pauses the game (stops announcing balls) while it verifies the player name with the card. If indeed it is a valid "BINGO" card belonging to the player, the game is over and the server announces the result. Otherwise the player is given a warning. After three warnings a player will be kicked out of the game. The server continues announcing the new Bingo balls.

v. After the game is over, the server may automatically clear its status display waiting for the first player to join the next game. Players can clear their application and join the next game, or choose to quit.

3.5.2.2 Design issues for Bingo game

The main concern is "trust". Players need to trust the gaming server: the server announces the Bingo balls randomly and should not favour any particular player. The gaming server needs to verify the winner: the winning card should be the validly registered card coming from the actual owner.

3.5.3 Chatting

The use of instant messaging (IM) on the Internet has gained much popularity in recent years. IM, like AOL Instant Messenger [118], Yahoo Messenger [119], ICQ [120], or MSN Messenger [121] makes use of the traditional client/server model. IM provides nearly real-time message delivery and allows users to maintain a contact list of people with whom they wish to interact. IM is able to inform users that their contacts are online. Therefore, users can type in and send short messages (chatting) to any of the people in the list as long as those people are online. The core functionality of IM is synchronously exchanging textual messages between two or more users. It is necessary for every user to set up and maintain a contact list or "buddy list". IM monitors the status of users' buddies on the network. In this respect, the concept of presence and availability of a user is relevant. Presence is the state, for example, "online" or "offline", characterizing the network status of the software or hardware device through which a user communicates. Availability is an attribute of a user, for example, "available" or "busy", denoting their willingness to communicate with other users based on the preferences and policies that are associated with the user.

In CN, if a chatting server which is able to maintain the whole network member list exists and a searching mechanism can be introduced to help users set up their contact lists, then CN could also be a good platform for chatting. Given the local shopping centre scenario, the EMB is capable of undertaking such a responsibility. Potential chatting users (clients) could register themselves by leaving their names, interests, presence and availability information on the EMB when they enter the shopping centre. Based on the interest, the EMB informs each user (client) of a name list containing related presence and availability information. Afterwards, shoppers who join the chatting application could have a chat with anybody on the list when they are

free. For example, they may be having a break in a café or they need to pass the time whilst waiting in a long queue. If shoppers failed to find the EMB initially, a person who stays on for a relatively long time could use his/her powerful device as the EMB. This procedure appears transparent for other shoppers.

3.5.3.1 Chatting procedure

i. Registration. A user sets his/her presence and availability status and then starts the client program bearing a username unique to the Chatting application. The client programme at once broadcasts the user's name with associated presence and availability information to the server. Anyone who receives the message relays it by rebroadcasting until the message reaches the maximum hop count or arrives at the server. Relay nodes may receive the same message from their different neighbours but it is relayed only once.

ii. Advertising. When the chatting server receives a client's register message or notification message (see procedure (iii) below), it will update the maintained clients' presence and availability information. The chatting server periodically broadcasts this to the network. Anyone who receives the message relays it by rebroadcasting until the message reaches the maximum hop count. Relay nodes may receive the same message from their different neighbours but it is relayed only once.

iii. Notification. When a client wants to change its presence and availability status, or finds out the received information about itself is wrong, or estimates another's status may have changed, it will immediately inform the server so that the server can modify the related information accordingly.

iv. Chatting. After receiving packets containing all members' presence and availability information, a chatting session takes place when a client sends and receives text messages with another on the member list. Procedures (iv) and (ii) or (iii) could happen simultaneously.

3.5.3.2 Design issues for Chatting

Participating in ad hoc networking environments without any knowledge about people around them, chatting users need to acquire and maintain their contact lists efficiently and promptly. Therefore, they could choose someone with whom to chat from time to time. In addition, the presence and availability information needs to reflect the real situation of the changing and complex environment.

3.6 Characteristics of applications in Convenience Networks

Since direct wireless connectivity is based on physical proximity, this reflects and facilitates the way that human beings seek convenience from their surroundings when they are on the move, since they are always constrained by the availability or affordability of existing infrastructure.

3.6.1 Application server is present

L. M. Feeney et al argue that "servers are problematic because nodes that become partitioned form the server must agree to either promote a backup server or reinitialize the service [98]". Some other scholars also believe that there should be no dependence on central servers in ad hoc networks: from the point of the view of the application, all nodes' roles inside the network are suggested to be equal. In other words, the application design would follow peer-to-peer (P2P) model.

However, in CN, this is probably an unrealistic view; the traditional client/server relationship can be defined according to an application logic. For instance, in a shopping mall, an EMB is required to be set up which then serves customers' mobile devices, although the connection between the EMB and a mobile device is ad hoc in manner. This means that the connecting route may change from time to time as the shopper walks around. In CN, server's responsibilities will be represented in two aspects. The fundamental aspect is the server working as the service and naming directories, which could include availability and presence information. However, depending on concrete applications, not all application clients need such information. Then the server can work as a bridge helping two clients set up a connection on demand. The server keeps polling its clients periodically so that it can always update the hosted information. The role of a Chatting server could belong to this category. Another responsibility of the server is working as a traditional application server like an Auction server or a Chatting server; it manages the whole procedure of the application from start to end. Obviously, in CN server's lifetime must be longer than that of any clients.

3.6.2 Limited extent in both space and time

CN is formed spontaneously; however, this does not imply that the running applications would vanish instantaneously. Informality in CN makes it an ideal platform for non mission-critical applications. By their very nature, these applications have a limited extent in both space and time. The life cycle of the network, in addition to being constrained by the battery life of mobile devices, mainly rests on the nature of the running application or the number of people who share the same interest. The application-dependant autonomous system dissolves only if all potential users leave

the network. Normally, CN could last for any length of time from a few minutes to an hour. For example, customers could spend half of an hour shopping in their local shopping centre. During this period, users would join or leave the network, as sometimes they intend to do so or sometimes they accidentally walk out of the network. Under the latter circumstance, these users would probably try to find new locations in order to rejoin the CN. Such a partition taking place in a limited area does not affect the application seriously. Similarly in an auction application, if some bidders are isolated temporarily from the network, the whole auction procedure may at most extend a few rounds.

3.6.3 Interaction via mobile devices

To some extent, CN works as social networks; users are connected and organized by some common interests when they walk into public places. Users are expected to interact with others by interacting with their own devices in a manner similar to reading and sending messages on mobile phones. Ring tone could alert users when messages arrive and are being displayed on their screens. Users could ignore some messages for a while as they may not be available at that moment. Or, users could respond after reading them. The responding interval really depends on the users' behaviours; this could vary from a few seconds to minutes. The messages are short because users are generally reluctant to enter large amounts of data due to the difficulties in keypad usage [122]. In addition, it is likely that users will become impatient when they read a great amount of information on small screens. The concepts of "intentionality", "common sense" and "cooperation with willingness" among moving people utilizing the properties of mobile devices are being leveraged in order to create an ordered method for initializing the network configuration and

conducting the smooth running of applications. Consequently, CN, as a supplement and enhancement of human interaction, are able to bring much convenience to users.

3.7 Summary

In this chapter, the notion of CN was presented after reviewing the development of applications in MANET. How the ad hoc networking capability of mobile devices can be leveraged to form an ordered system was described. The system is capable of enabling users to seek convenience from their neighbourhood. Specifically, the characteristics of the system and some related applications were described. Finally, three example applications were explored in detail. These three applications will be chosen to study both the system requirements for CNs and the necessary system support which should be incorporated in mobile middleware.

Chapter 4 Design and implementation of interactive applications

This chapter identifies general design requirements for multi-user interactive applications in Convenience Networks (CNs). Then the design and implementation of a generalized data process module is presented. The module is able to support the three representative applications in CNs as described in chapter 3. The module will serve as a foundation of mobile middleware; the ultimate goal of this empirical study is to find out what system support should be incorporated in the middleware for MANET-based applications as currently no such support is provided.

4.1 Design requirements

In order to help developers easily implement multi-user interactive applications in CNs, the application-oriented requirements are comprehensively analyzed and the common communication pattern is abstracted.

4.1.1 Application-oriented requirements

Based on the description given in sections 3.4 to 3.6, the following requirements for applications in CNs are identified.

i. Service utilization. Different types of services hosted on a server need to be effectively advertised. By effectiveness it means that the server should be able to disseminate messages to interested clients at the right time. On the other hand, clients need proper mechanisms to discover services.

ii. Communication (routing packets). The process of service advertisement, discovery, provision and use is through exchange of packets between servers

and clients. A sender needs to dynamically find a route to deliver packets to its destination.

iii. Connectivity. Given the uncertainty introduced by node mobility, there may well be a need for more than one route connecting a sender and the receiver. Ideally, the sender needs to choose the route over which the shortest transmission latency occurs.

iv. Adaptability. Applications need to deal with various changes dynamically and responsively during their lifetimes. Besides user mobility, there may be unnoticeable changes that take place around users and affect the application unexpectedly. The application should be adaptable to these changes and the procedure transparent to users.

v. Lightweight and portability. Given the heterogeneity of mobile devices, the overall application design needs to be lightweight and portable in order to gain acceptable performance across different hardware platforms.

vi. Security. Given the complex situations in ad hoc networks [122] and requirement (v), the main concern is on "trust": server should be convinced that each received packet comes from the right client with the correct information; each client is confident to be treated impartially.

vii. Reliability. Given requirement (v), this goal needs to be achieved in an economical way. If packets are lost, the proposed recovery mechanism should not significantly consume computing resources. It should not have negative impact on members' activities inside the network.

viii. Simplicity. Users expect simple interfaces with plain language description; available choices should be clear.

4.1.2 Common aspects of applications in Convenience Networks

Applications in CNs are running on users' mobile devices and monitoring events happening in the networks. Based on the description of the three representative applications in section 3.5, the following common communication pattern can be observed.

i. Dissemination by a server. The server periodically disseminates information to interested clients. The information could be anything related to current applications such as an auction will happen in 10 minutes time, the currently held highest bid value, calling a Bingo ball, the result of an auction, and so forth. Or it could be help information like the presence and availability information of users in the network.

ii. User response. When packets from the server arrive at users' mobile devices, a suitable method like using ring tone can be applied to alert users. Reading messages on devices' screens, users could operate their devices to respond to the server or another participant depending on the application logic. Users could leave at anytime without necessarily informing anyone. Also users could ignore some messages by taking no further action.

4.2 Design overview

The extension of the three popular wired-network applications to CN offers considerable potentials. However, decentralized nature of CN and dynamic topology certainly pose challenges to designers. The approach taken is to keep the simplicity of the traditional client/server architecture as the nature of applications requires. Clients look for the server, inquire, and respond to messages from the server; the server monitors the network situation and informs all its clients periodically. Clients need not

communicate with each other in Auction and Bingo game. However that is the compulsory function in Chatting so that the Chatting server also acts the mediator for communication. The difficulties exist in:

- How does the server maintain and disseminate the application information to clients? What do clients do if they suspect some packets are lost?
- How do clients reach the server?
- How do client and the server or two chatting entities find a route between them and maintain a reliable connection?
- Does it affect the application if some clients leave the application without notice?

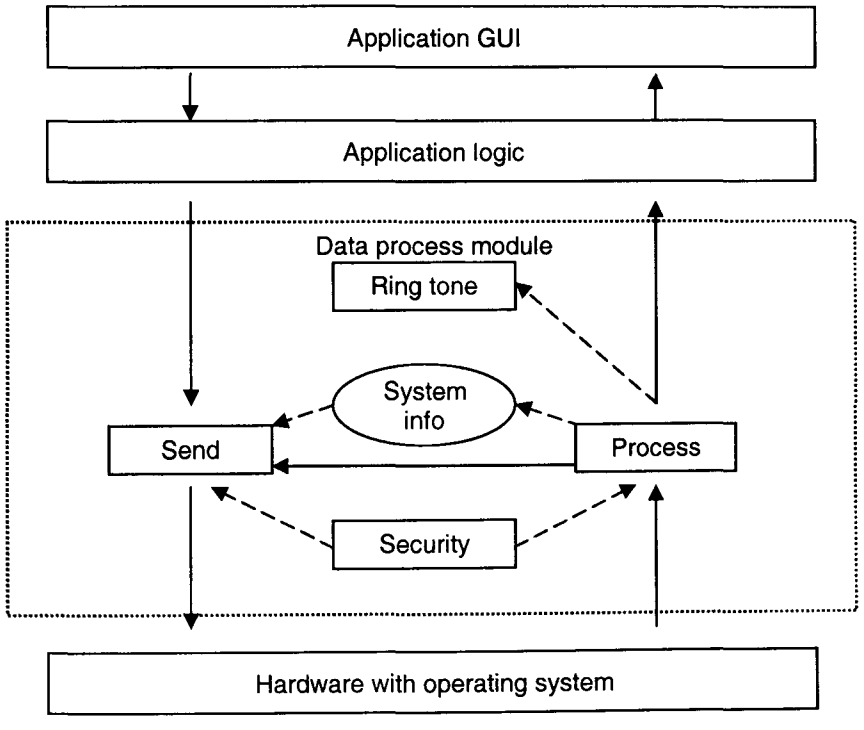


Figure 4.1: System architecture

There are two modes of communication required by the applications: broadcast (one-to-all interaction) when the server disseminates necessary information; unicast (one-to-one interaction) when a client corresponds with the server or a client chats with

another. The system architecture is shown in figure 4.1. The general data process module is to support various types of multi-user interactive applications in CNs.

The design is based on the following simple ideas. Broadcast is implemented by flooding (diffusion): a node receiving a broadcast packet diffuses it by rebroadcasting; it discards duplicate packets. A node rebroadcasting a packet also adds its identity in the route field kept in the header part of the packet. A node receiving a broadcast packet can find out the route to the source taken by that packet from the information in the header, and uses that route for unicasting any response packet to the source; a node receiving such a packet forwards (relays) it to the next node on the route. The protocol is a simplified version of Dynamic Source Routing (DSR) protocol [15]. As many simulation studies point out: in a small-scale ad hoc network with low mobility (that is one of the assumptions of the three proposed applications), DSR outperforms other routing protocols [93, 94]. The proposed communication protocol design keeps the "source route" idea of DSR: a node broadcasts packets in the same way as if it broadcasts "route inquiry" packets of DSR, but without implementing "route reply" or "route maintenance". These simplifications are possible because of the nature of the applications.

4.3 Definitions

In a CN, a sender, node S needs to send packets to the receiver, node D. If D and S are out of their transmission ranges, the packets are delivered node-by-node (hop-by-hop) following a source route like (S->A->B->C->D) decided by the proposed communication protocol. Given this context, the following terminology will be used to describe the protocol design in section 4.5, testing scenarios, and testing results in chapter 5 and chapter 6.

- *Source route length*, denoted as L_s, is defined as the hop count of source route. For example, given a source route (S->A->B->C->D), the source route length is four hops.
- *Radius* of the network, denoted as H_{max}, is defined as the maximum hop count that a packet can pass over the network after it leaves from the server and the source route the packet used must not have repeated nodes. In a CN, if the actual value of H_{max} is four hops, the network is called a 4-hop network.
- *Distance* between two nodes is defined as the length of the shortest source route connecting the two nodes theoretically. The length of the source route a sender used may be longer than the sender's distance to the receiver. For example, the distance between server S and client D is four hops means there exists a 4-hop source route connecting S and D. Under this circumstance, D could be called a 4-hop client.
- *Neighbours* of a node are defined as the collection of nodes which their distance to this node is one hop.

4.4 Application assumptions

Different instances of an application can be in progress at different stages in a CN. The following additional assumptions have been made:

i. All wireless interfaces work well in both directions. The transmission ranges of all wireless interfaces are equal.

ii. An application is enacted in some limited geographical place (say, the radius of the formed network is four hops) and the number of participants is relatively small (say, less than 50).

iii. Participants will run applications on their mobile devices when they arrive at some public place.

iv. Participants can be far enough apart so that not all of the wireless interfaces on their mobile devices are within the transmission range of each other (for short, the participants' devices are out of the range of each other).

v. Participants are able to move. Two participants' devices within range at one point in time may be out of range moments later. Participants may move to find a suitable location to run the application.

vi. Participants' devices assist each other in the process of relaying packets.

vii. Mobility is low, especially when participants are playing the game or chatting with another people. It is expected that significant changes will be relatively infrequent during the lifetime of an application.

viii. The name of the application server is well-known to all participants in the network. Each device bearing the same, exclusive network name has been configured to listen to or send packets to a well-known port.

ix. IP addresses of devices do not conflict with each other.

4.5 Protocol design

Nodes exchange data through formatted packets (details are given in section 4.9). Clients maintain a pair of variables recording the server address and the highest sequence number of the received packets from the server (S_h). Clients discard any packets with sequence numbers less than S_h or stop rebroadcasting if the source route length the packets used reaches a given H_{max} (radius of the network). Otherwise clients append their own address to the source route in the control part of a packet, update S_h and rebroadcast to the network. This procedure is illustrated in figure 4.2.

Packets in figure 4.2 show a simplified structure of packet from server: row 1 records sequence number of this packet and the server's address, row 2 records source route the packet taken, and row 3 shows payload data. A client learns its source route to the server by reversing the source route in the control part of the packet; it stores this route information in its route table. Clients update their route tables whenever they receive a new packet from the server. Clients use their source routes when they want to unicast packets to the server. Every client keeps only one source route to the server in order to minimise its memory requirements.

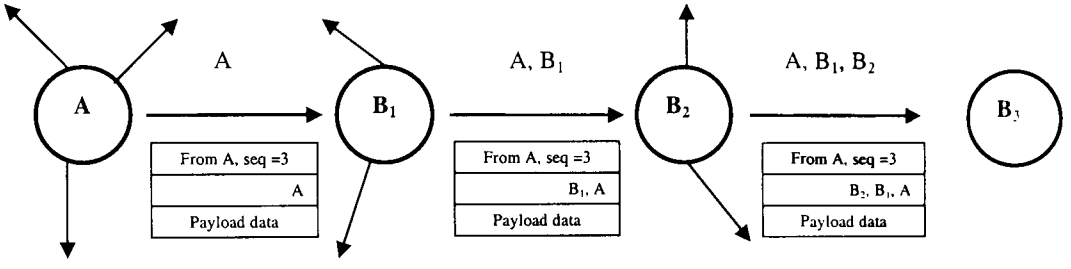


Figure 4.2: Source route building in a packet from server A. B_1, B_2, and B_3 are clients. (A sets H_{max} to three hops.)

Due to node mobility in the network, a client's source route sometimes will not work. This could happen if one suddenly moves out of its neighbour's range just after it updated the route table. The result is that the server may not receive one response from this node. The applications are not mission-critical; the lost packets do not seriously affect the applications. The node could learn the source route to the server again when it receives the new broadcast packet from the server.

4.5.1 Broadcast storm problem

Straightforward flooding without any control mechanisms is simple and effective for small-scale ad hoc networks. However when density of nodes becomes higher, radio

signals are highly likely to overlap with others in a limited area, flooding is usually very costly and will result in serious redundancy, contention, and collision; far too many redundant packets will be rebroadcast leading to a "broadcast storm" [123, 124]. S.Y. Ni et al address this problem in depth for MANET; the phenomenon where the transmission of a packet by broadcasting will trigger other surrounding neighbours to transmit the same or modified packet [123]. The authors argue that the broadcast is spontaneous and unreliable in MANET. Without knowing the concrete amount of the broadcast recipients, no acknowledgement will be used for solicited broadcast packets though "attempts should be made to distribute such packets to as many hosts as possible without paying too much effort [123]". Setting in a CSMA/CA network, drawbacks of blind flooding include [123]:

i. Redundant rebroadcasts. When a mobile node decides to rebroadcast a just-in broadcast packet, all its neighbours have already received it.

ii. Contention. Upon receiving a broadcast packet from the same source, a mobile node could decide to rebroadcast the packet as its neighbours do so. These transmissions which are all from nearby nodes may severely contend with each other.

iii. Collision. Because of the deficiency of CSMA/CA algorithm, the lack of *RTS/CTS* exchange, and the absence of contention detection, collisions are highly likely to happen and cause damages.

4.5.2 Additional area reached by rebroadcasting

Through simulation study, S.Y. Ni et al present the result on the additional area that can be reached by rebroadcasting after a node receives the same broadcast packet [123]. The simplest scenario is shown in figure 4.3, in which there are two nodes A

and B. Assuming the source node A and the relay node B have same wireless transmission range r, the additional area that can benefit from B's rebroadcasting is the shaded region. The result shows that the maximum additional area can be reached is around 0.61 πr^2 and the average value is 0.41 πr^2. The results become worse when there are more relay nodes around A: when there are two nodes B and C around A, the benefit from C's rebroadcasting after it heard A and B's broadcast packets decreases on an average to 0.19 πr^2. Generally when a node hears the same broadcast packet four times or more, the expected additional coverage is below 0.05% [123].

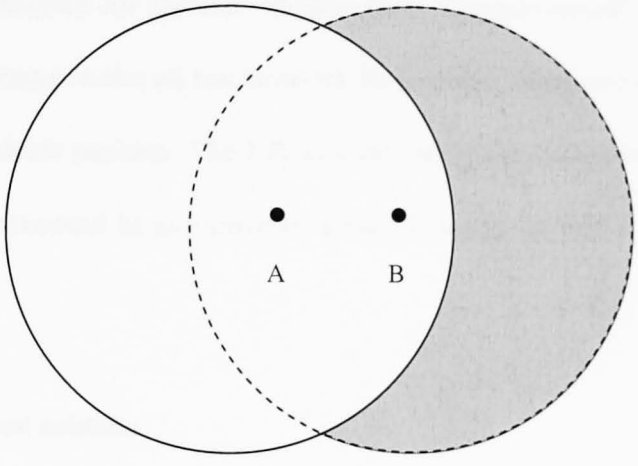


Figure 4.3: The additional area can be reached after node B rebroadcast the received broadcast packet from node A [123].

4.5.3 Solutions to overcome the broadcast storm problem

In order to alleviate this problem it should be possible to prohibit some nodes from rebroadcasting to reduce redundancy, and thus contention and collision. Five schemes are proposed to do so. They are "Probabilistic" scheme, "Counter-based" (CB) scheme, "Distance-based" scheme, "Location-based" scheme, and "Cluster-based" scheme [123, 124]. These schemes differ in how a mobile node estimates redundancy and how it accumulates knowledge to assist its decision. Except the last scheme,

which relies on some local connectivity information, all of them operate in a fully distributed manner [123].

"Probabilistic" scheme works in an intuitive way. Upon receiving a broadcast packet for the first time, a node will rebroadcast it with probability P (when P =1, the scheme is equivalent of simple flooding) [123]. However there is no practical way to decide the value of P that should be used in the application. "Distance-based" scheme needs the relative distance between nodes to make decision. The distance could be acquired by measuring the signal strength on which a message is received. "Location-based" scheme needs help from positioning devices like GPS receivers. Therefore these two schemes are not suitable for general applications. "Cluster-based" scheme assumes that clusters are formed in the ad hoc network first before using any of one of the four schemes to rebroadcast packets. The CB scheme (to be described below) is practical and could be implemented in any programming language so that it is adopted in the current design.

4.5.4 Counter-based scheme

The CB scheme maintains an inverse relationship between the number of times a redundant packet is received at a node within a short interval and the probability of that node rebroadcasting. There are two constants involved: maximum waiting interval (T_{max}) and maximum number of redundant packets received (C_{max}). Upon receiving of a previously unseen packet, the node initialises a counter with a value of zero and sets a random waiting time (RWT), which is between zero and T_{max}. During RWT, the counter is incremented by one for each redundant packet received. After RWT expires, if the counter value is less than C_{max}, the redundant packet is rebroadcast. Otherwise the packet is discarded.

4.6 Application design issues

This section presents design issues from two aspects: general ones that apply to all applications and specific ones that relate to each of the three applications described in section 3.5.

4.6.1 General design issues

Due to node mobility or other interference on radio channel, packets could get lost during the transmissions. Generally, broadcast packets encapsulate the network and application information; unicast packets contain the responses from clients. From application's perspective, server is keen to find out why it did not receive responses from some clients after disseminating a packet. These clients did not receive the packet, or their response packets were lost, or they did not respond at all. On the other hand, participating clients need to know if their response packets had arrived at the server. Actually, some clients cannot receive broadcast packets if they are out of the network temporarily. Hence the server will lose the chance to attract more potential users. A client's source route sometimes might not work. This could happen if one suddenly moves out of its neighbour's wireless transmission range just after it updated the route table. The result is that the server may not receive a response from this client. On all accounts, applications could do with a mechanism that can help them retrieve lost packets. In addition, regarding different nature of applications in CN, the lost packets will have different negative impact on them. Therefore, system design should target those concrete issues in application specific manner as well.

The prototype design described here does not provide any mechanism to salvage lost packets. However, based on the simulation study that is described in section 5.6.1, a mechanism is presented in section 6.3.4. It is observed that applications in CNs could

tolerate packet loss to some extent. Like in an auction, for some specific rounds, suppose the server did not receive packets from some bidders. Hence, the currently held highest bid value may be incorrect. Possibly during this period some of bidders could not receive the server's packets. A few rounds later, all bidders could receive the latest auction information again. If some bidders find out that the currently held highest bid value is lower than what they had bid, they could realize that their previous bids had lost and would bid again. Lost packets could not affect the auction if those bids were lower than the currently held highest bid. The result is that at most the auction lasts a few rounds more.

Secondly, users are rational. They could realize that packets are getting lost as nothing is displayed on their mobile devices for some time. They would move, try to find a niche, reconnect to the server, and resume participating in the applications. They will learn the source routes when they receive new broadcast packets from the server again. If these users are really isolated from the network, there is no way to guarantee that they receive any messages.

Finally, these "informal" applications in CNs are not mission-critical. Users join applications for useful information or fun. If users realize that some packets are lost, they could try other sources later or they may not care about the results at all.

In a word, the overall result is that lost packets cannot affect applications seriously.

4.6.2 Design issues for Auction

In auction *"fairness"* is very important to both sides [125]; it means that all bidders have an equally fair chance of submitting a successful bid, and all sellers have an equally fair chance of selling their items at a price that reflects market demand. *Fairness* could be compromised due to uneven communication delays; bidders that

have faster connections could gain advantages over their competitors with slower connections, as these bidders are several hops away from the auctioneer.

To address the "*fairness*" issue, one commonly used solution is to implement a simple variation of open cry auction. The difference between auctions in CNs and physical open cry auctions is that as bidders operate their mobile devices to submit a bid, then they could bid almost simultaneously; at that moment, the highest bid may have increased but they do not know. Therefore, auction server need not broadcast the result about every received new bid if the bid is higher than the current highest bid. Instead, the server assumes that each new round will last for a fixed period before announcing the latest result. During the period, the server collects bids and can decide the highest bid from the received bids. Once the auction starts, the server will periodically broadcast the latest result until the auction ends. Certainly, for a specific round, some bidding packets may arrive late at the server, just as the server has started the new round. If the bids inside these packets are lower than the current highest bid, they will be ignored. Otherwise the server will update the value of the highest bid for the current round. Consequently, the auctioneer will receive the highest bid that actually reflects the real situation, though the auction may take a few rounds more than the expectation.

An alternative could be to introduce a short interval D_h into the data process module over the network layer if bidders use agents (proxy) to submit a bid. On the server side, when the data process module receives the first new bid for a specific round, it does not send the bid to the upper application logic layer but waits D_h for other possible bids. After D_h, the data process module sends all received bids to the upper layer. The upper layer would decide to broadcast the highest bid to the whole network. Form the view of the application, all the biding information comes at the same time. It

means to all the bidders the chance is equal and fair. The value of D_h is suggested to set to $[(H_{max}-1) *2 +1] * T_D$, where H_{max} is radius of the network and T_D is average packet transmission and processing latency over one hop source route. (The simulation study in section 5.1.6 shows that the value of T_D depends on the computing ability of hardware. In a conservative way, the value of T_D has to be chosen the maximum value for such latency over each hop.) The same method could be applied on clients' side too, so that messages are displayed on everyone's screen at about the same time.

4.6.3 Design issues for Bingo game

The chance to win a game relates to the number of the Bingo cards that a player has. One game has one and only one winner; every announced Bingo ball is only needed by a unique player to declare BINGO. Players are not likely to trust the server to generate cards and also announce Bingo balls randomly. If the server knows each player's cards, players may also suspect if the server is announcing Bingo balls randomly. When a player declares BINGO, the server needs to verify if the claimed winning card is a valid card belonging to this player.

To address the issues described above, in a game each player will have only one Bingo card. Players generate and encrypt their own cards that will be sent to the server. When the game starts, the server broadcasts Bingo balls; players decide to accept or deny the current ball. When a player announces BINGO, the player will send his/her card in plain mode to the server as well as inform the server how to decrypt the card that s/he has sent before. After verifying the claimed winning card the server announces the winner, stops the game or informs all that a player was cheating. In the latter situation the game continues until real winner emerges.

4.6.4 Design issues for Chatting

Clients only know the names of other clients for chatting, as the server maintains name to address mapping. Similarly, presence information only provides clients' names without any address details. Thus, clients need not reveal their true identities. This makes sense as the application is essentially arranging conversation between strangers. Bearing this in mind, the name list disseminated to clients is not really a buddy list, but just a list of names with which users may set up communications. In Chatting clients need to communicate with each other instead of only with the server. Hence clients should find and maintain the routes to the recipients. They also should acquire the presence and availability information from the server globally and promptly.

From the perspective of the application, the server's role is subsidiary; although it maintains routes to all clients with all nodes' presence and availability information that can facilitate each chatting session.

Clients keep the route to the server. In addition, if acting as relay nodes, clients could passively learn source routes to some addresses, although these clients have little knowledge about the corresponding names. It is unnecessary for clients to keep routes to all others as they may be only interested chatting with a few. Upon receiving the latest broadcast packet from the server (containing the latest presence and availability information), clients always refresh the source routes to the server. If a client finds its source route to the server has changed it informs the server the new one. This triggers the server updating its route table.

When a client wants to chat with another (starting the invitation session), first the packet is delivered to the server as the client has no address and route knowledge about the recipient. The packet may be relayed by some other clients and finally

arrives at the server. The server locates the route to the final destination and forwards the packet to the destination. If the recipient accepts the invitation, it will take the reverse route that it just learned and send the acknowledgement back to the initiator (in practice it piggybacks real chatting contents). This procedure is shown in figure 4.4. Having received the presence and availability information, if node D wants to chat with node A, D sends an invitation packet to server S following the source route that it just learned (D->B->S). After checking its route table, S forwards the packet to A. If A accepts the invitation, it sends an acknowledge packet back to S taking the reverse route (S->A) that it just learned. S forwards the packet to the real recipient, D.

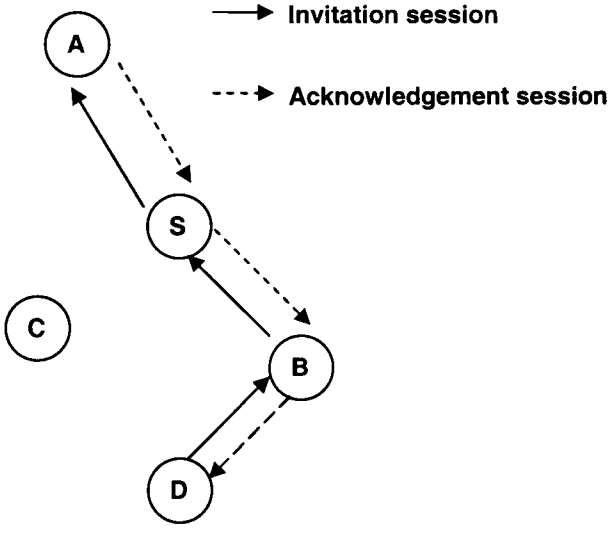


Figure 4.4: Node D initiates chatting with node A. Node S is the Chatting server.

There is possibility to optimize the algorithm. Same as shown in figure 4.4, if node B wants to chat with node D it sends the invitation packet to server S. Then S sends the packet back to B and B transmits it to D. D decides to have a chat with B and sends acknowledgement packet to B following the source route (D->B->S). B, as the real recipient, stops forwarding to the next hop and learns a new source route (B->D). B will use it for the coming session if it continues to chat with D.

In conclusion, when a client receives a unicast packet not having the full addresses of both the originator and recipient in the filed of source route (see tables 4.2 and 4.3), the client should check if it is the recipient. If so, it shortens the source route in the packet and uses it for the coming session. Otherwise it relays the packet to the next hop following the source route.

Another new type of user availability attribute, "unknown" has to be introduced to take account of effects of node mobility (the other three are: "available", "busy", and "off"). When users start the application, the default availability setting is "available". Sometimes a client is not sure why there is no response from the recipient. There could be several reasons: the recipient did not respond promptly, the recipient has left, or the recipient has moved. Then the client informs the server that the availability status of the recipient node should be changed to "unknown". This particular node must inform the server of its actual status after it checks the latest updating packet from the server otherwise the server will set it to "off". If a client wants to change its own availability status it informs the server too. Then the server modifies its record and disseminates by next updating time.

4.7 Implementation

Nowadays native operating platforms for mobile devices mainly are Palm, Windows mobile version, and Symbian. Java, being a software-only platform running on top of other hardware-based platforms and available to address applications development for and deployment to those different environments, displays much advantage over proprietary development tools. Lightweight J2ME CLDC (Connected Limited Device Configuration) with MIDP (Mobile Information Device Profile) has been supported by most of latest mobile devices though they provide no support on address mechanism or socket programming. Regarding designing interactive applications in

CNs, more powerful APIs on graphics, thread, IO, and socket support are required. Therefore mobile devices should at least run J2ME CDC (Connected Device Configuration) with PBP (Personal Basis Profile); it will be better if they support J2SE. In addition, as target platform is different from development platform (Windows XP professional with J2SE 1.3), sometimes applications work well on development platform but fail on target platform, especially when the classes involve methods in Abstract Windows Toolkit (AWT) package. The classes and associated methods (all of them must be supported by different target devices) are had to choose very carefully. What is more, much time is spent repeating modifying design and testing in order to make application look well on mobile devices.

There is much activity on developing routing protocols for MANET. However, most of proposed routing algorithms are still being researched, and are not available on any of the operating systems on mobile devices. The communication protocol described earlier is not implemented in the network layer. It works as a component of middleware. Transport protocol chosen is UDP for its simplicity.

4.7.1 Application framework overview

The architecture of the application framework consists of the following components: application graphic user interface (GUI), application logic, and data process module (see figure 4.1, a solid line depicts message flow while a dashed line shows cooperation between two components.). Application GUI and application logic closely work linking users and an application. The specific application logic extracts and then reflects detail application information to the application GUI from the underlying data process module, by which users make their decisions. Afterwards, the application GUI traps input from users feeding information into the application logic.

Data process module is the core of an application: it resolves the formatted packets from UDP socket and filters out different targets; or generates the formatted packets encapsulating the payload with source route. The data process module is composed of a number of components. "Process" and "send" components, as their names show, are responsible for proceeding received packets from the network and sending packets into the network. Other components are described as follows.

4.7.2 System information component

System information component records the important system variables like the server address, source routes, or the highest sequence number of received packets from an address. For Auction or Bingo game, the route table on client's side only contains the source route to the server; on server's side it stores source routes to all clients (however server may not use them). For Chatting, the route table contains the source route to the server or to some other clients (if this node is a client); or source routes to all clients and mappings between user's name and address (if this node is the server).

4.7.3 Security component

Given the structure of data packet described in section 4.9, security component supports use of "public key" to authenticate fixed length information *M* in the application data part of packet (by decryption). Table 4.1 lists the authentication type. Due to small size of *M*, application logic can send *M* with the encrypted *M*, *E(M)*, in one packet. Given that *E(M)* is next to *M* in a packet, except the public key, the security component just needs to know *index*, the starting position from which *E(M)* is stored in order to authenticate *M*. Specifically, application logic uses DSA (Digital Signature Algorithm) to generate the pair of keys. As a result, the length of *E(M)* is 46

bytes, which is 18 bytes shorter than the result if another popular algorithm, RSA (The Rivest, Shamir, and Adleman algorithm), is chosen.

type	Description
0	Authentication service not required.
1	Real recipient performs authentication.

Table 4.1: Authentication type

4.7.4 Ring tone component

Ring tone component is for practical and adaptive purposes. The component provides one type of ring tone with five levels of volume to alert users to what is happening around them. The level of volume is the mark of the importance of received packet; the higher of the volume, the more important is the packet. The importance of packet is decided by application logic.

4.8 Screen shots

Figures 4.5 to 4.13 show some screen shots of implemented applications.

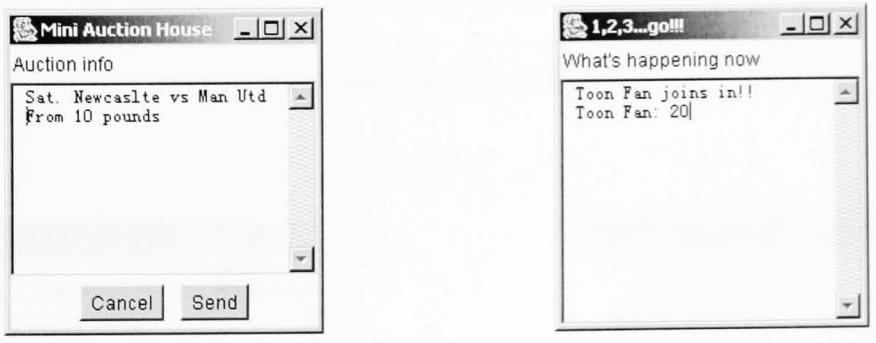


Figure 4.5: Screen shots on Auction server

Figure 4.6: Screen shots on bidder

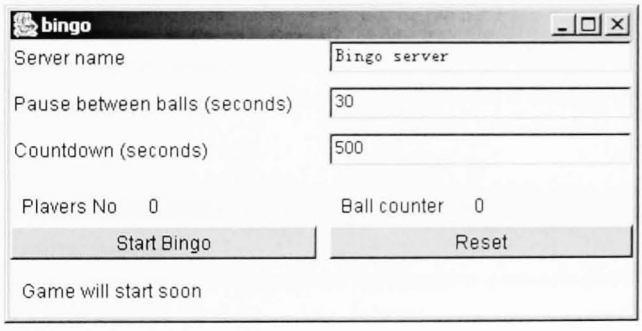


Figure 4.7: Initial situation on Bingo server

Figure 4.8: Screen shots on Bingo client during the registration period

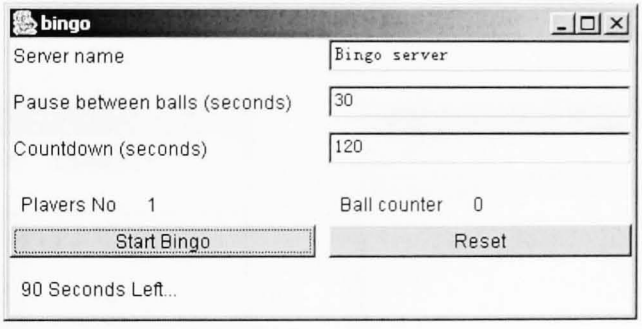


Figure 4.9: Screen shot on Bingo server during the registration period

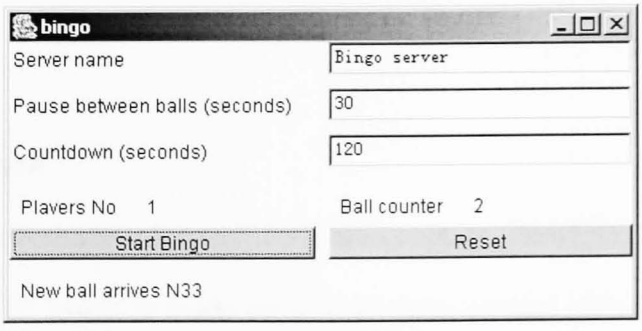


Figure 4.10: Screen shot on Bingo server during the game

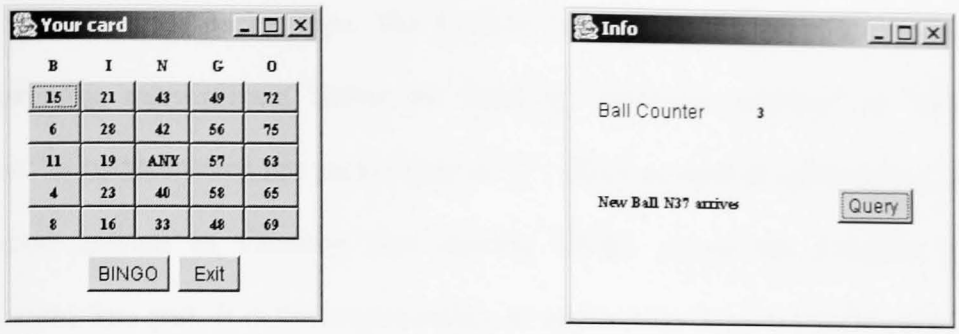


Figure 4.11: Screen shots on Bingo game client during the game

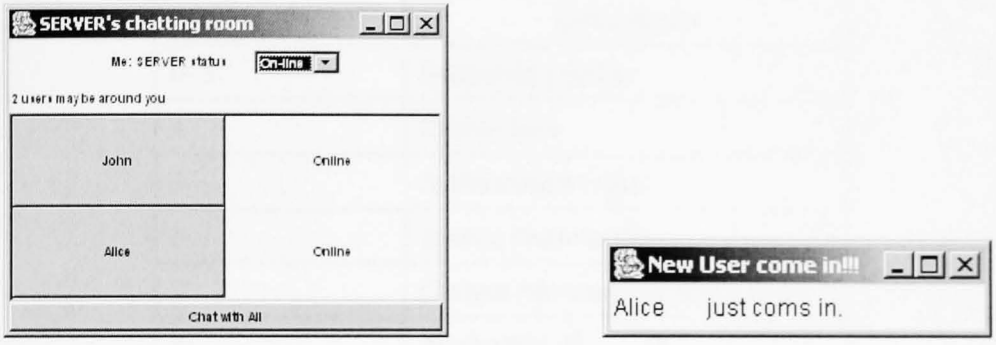


Figure 4.12: Screen shot on Chatting server

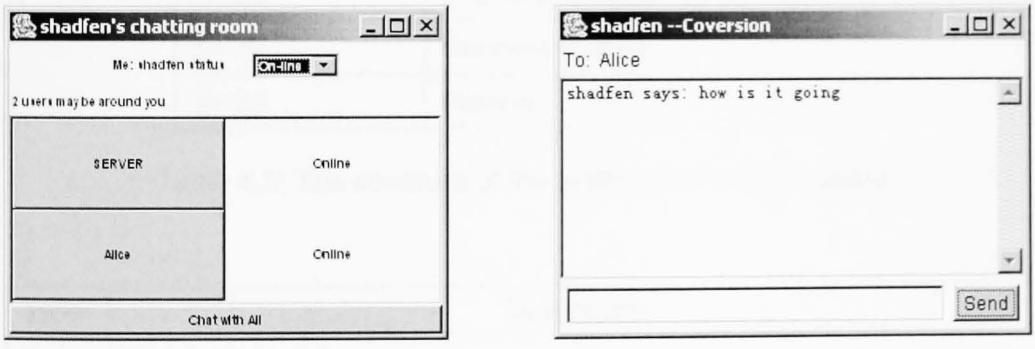


Figure 4.13: Screen shots on Chatting client

4.9 Packet structure

Nodes exchange information through formatted packets. Fixed length of UDP packet (payload size is 512 bytes) is made up of two parts: control part (the first 64 bytes) and application data part. Table 4.2 describes the structure of the control part and

table 4.3 shows the packet type. The field of "sending direction" is for the sake of programming convenience. Either the field of "originator address" or "recipient address" is padded when the packet type is "2", which is used in applications like the invitation session in Chatting (see section 4.6.4). About the structure of the application data part, it is the responsibility of application logic to specifically define it.

Index (Bytes)	Description
0~3	Sequence number
4	Packet type
5	Authentication type
6	Source route length
7	Current hop count in source route
8	Application ID
9	Sending direction
10~29	Source route
30~33	Originator address
34~37	Recipient address
38~63	Reserve

Table 4.2: The structure of the control part in data packet

Type	Description
0	Broadcast packet
1	Unicast packet (source route has the addresses of originator and recipient)
2	Unicast packet (source route has the address of either originator or recipient)

Table 4.3: Packet type

4.10 Summary of APIs

The summary of data process module APIs is shown in table 4.4. (For simplicity, exceptions raised by calls in response to error conditions are not shown.)

Return type	Method name
void	start (boolean role)
	Initialize the system; *role* specifies the host is the server or a client.
void	InitKey(PublicKey public_key, PrivateKey private_key)
	Generate the key pair to encrypt and authenticate information of the host.
byte [ ]	getAppLayerData()
	Return the application data as array of byte. If authentication fails, an exception is raised.
InetAddress	getOriginatorAddress()
	Return the originator's IP address.
void	setMapping(InetAddress originator, String originatorName)
	Set the mapping between the address *originator* and the name *originatorName*.
void	setPublicKey(PublicKey public_key, InetAddress originator)
	Set the public key *public_key* for the address *originator*.
void	forward(byte [ ] AppLayerData, InetAddress recipient)
	Forward the array of byte *ApplayerData* to the address *recipient*.
void	send(byte [ ] AppLayerData)
	Send the array of byte *AppLayerData* to the default IP address[1].
void	send(byte [ ] AppLayerData, InetAddress recipient)
	Send the array of byte *AppLayerData* to the address *recipient*.
void	send (byte [ ] AppLayerData, Authtype a, int index)
	Send the array of byte *AppLayerData* to the default IP address using the authentication type *a*; the encrypted information starts at the position *index*[2].
void	send (byte [ ] AppLayerData, InetAddress recipient, Authtype a, int index)
	Send the array of byte *AppLayerData* to the address *recipient* using the authentication type *a*; the encrypted information starts at the position *index*.
void	applyCB(int C_{max}, int T_{max})
	The host uses the CB scheme with the associated parameters C_{max} and T_{max} to rebroadcast the packet.
void	applyRingTone(int level)
	The ring tone with the volume *level* is used when the packet is received.

Table 4.4: Summary of data process module APIs

[1] For server, it is the broadcast address (255.255.255.255); for client, it is the server's address.
[2] See section 4.7.3.

4.11 Summary

In this chapter the requirements for supporting multi-user interactive applications in CNs were identified. A foundation of middleware design and implementation were presented. The importance of designing such system is to study what functionalities should be supported in mobile middleware for MANET-based applications. A dedicated evaluation of the design is presented in chapter 5.

Chapter 5 Evaluation

In order to evaluate the performance of the designed system for supporting multi-user interactive applications in Convenience Networks, dedicated testing was conducted in two forms. Small scale field testing in laboratory environment provided first-hand data outlining the general understanding of the applications. These results also showed some potentials or possibilities that would happen in larger scale ad hoc networks. Simulation study was to verify such possibility and demonstrated how it affected the functioning of applications. Various quite subtle application scenarios have been simulated and results analyzed comprehensively. Overall conclusion is that the design is capable of supporting the proposed applications working reasonably well in real environments but there is scope for improvements. The results build up sound foundations for designing mobile middleware with adaptation features (that is presented in chapter 6). In addition, previous simulation study results and the latest progress of wireless network simulators are presented.

5.1 Field testing

The testing is to evaluate the performance of two flooding methods, simple flooding and the counter-based (CB) scheme, in field environment. In addition, it reports the impact of hardware heterogeneity on the application's performance.

5.1.1 Testing environment

The testing involved one IBM ThinkPad T21 laptop acting as the server and five HP Jornada 720 PDAs as clients. The hardware configuration and software running

environment is shown in table 5.1. The IEEE 802.11b *WiFi* cards from two vendors (Orinoco and Cisco AirNet 340) were installed, which were set in ad hoc mode and worked on a designated channel. The fastest transmission rate of the *WiFi* card for the laptop was 11 mega bits per second (Mbps) but 2 Mbps for the PDAs.

Unit	CPU	Memory size	Operating system	Software specification
Laptop	Intel Pentium III 800 MHz	384M Bytes	Windows XP Professional	J2SE 1.3
PDA	Intel StrongARM SA1110 206 MHz	20M Bytes	Windows for Handheld PC 2000	HP Chai VM. Compatible with J2SE 1.2

Table 5.1: Hardware configuration and software specification for field testing environment

5.1.2 Performance metrics

Three performance metrics are of interest at each client:

i. *Packet loss*: the number of broadcast packets from the server that are not received by the client divided by the total number of sent packets from the server.

ii. *Round-trip broadcast latency* (*latency* for short): during flooding, the round-trip interval from the time the server initiates broadcasting a packet to the time the rebroadcast packet from the client returns to the server. (Without synchronizing the clocks on the laptop and the PDAs, it is best to measure the round trip latency.)

iii. *Saved ratio*: (r -t) / r, where r is the number of received packets by the client and t is the number of packets rebroadcast by the client. This metric is meaningful in schemes that try to reduce the number of rebroadcast packets, like the CB scheme.

5.1.3 Simple flooding test under no mobility

The first goal is testing the performance of simple flooding as it is the main technique that the application server uses to disseminate information to the whole network. A

static 1-hop ad hoc network (refer to section 4.3 "definitions") was constructed in the topology shown in figure 5.1. The figure depicts the conceptual view of the place where the experiments were conducted. In order to examine the efficiency of simple flooding three configurations were tried: one, two and five clients were put close to the server respectively, all within the wireless transmission range of each other.

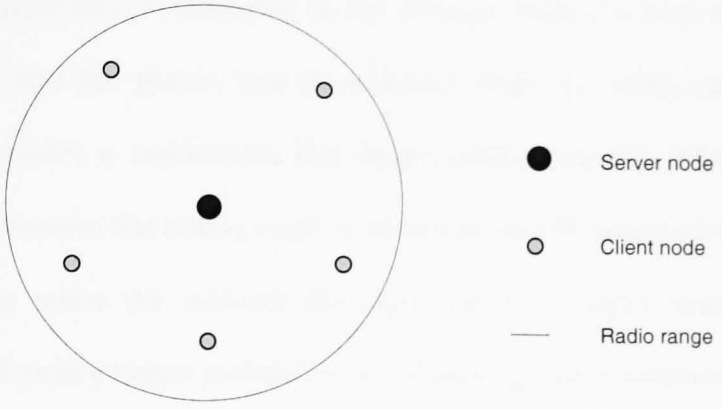


Figure 5.1: Test bed topology of 1-hop ad hoc network for simple flooding

Number of clients	1	2	5
Average latency (ms)	20.7	22.7	73
Average packet loss	< 0.1%	< 0.1%	2%

Table5.2: Round trip latency

Every 500 milliseconds (ms) the server broadcast a UDP packet containing 512 bytes payload and recorded the sending time. Upon receiving a broadcast packet from the server each client rebroadcast to the network following the communication protocol design (see section 4.5). The packet would again come back the server. Then the server recorded the time of reception so that it was able to measure the round-trip latency of 2-hop route for each client. For each session the server broadcast 500

packets and the same experiment was repeated three times. The results are shown in table 5.2.

The ideal situation for communication is when there is only one client with the server since there is no contention when each node broadcasts a packet. With regard to two clients with the server, there is slight contention in the network; on average the latency is a little longer than the ideal situation. When the number of clients reaches five, there is fairly heavy contention as the average latency is over three times the ideal situation and the packet loss is relatively high. To some extent, it shows CSMA/CA provides a mechanism that helps reduce collision when nodes start broadcasting. However the results imply a trend that the efficiency of simple flooding would be poor when the network becomes dense; a higher number of nodes contending will yield a higher probability of collision so that it means that CSMA/CA mechanism will not be able to guarantee reliable broadcasting.

T_{max} (ms)	25	50	100
Average latency (ms)	46.8	57.4	75.7
Average packet loss	0	0	0
Average saved ratio (%)	32.1	36.7	46.2

Table 5.3: Five clients using the CB scheme

5.1.4 Testing the counter-based scheme under no mobility

Same topology was used as shown in figure 5.1. The experiments were conducted with the server and five clients. The server's behaviour was same as described in section 5.1.3 but clients complied with the algorithm described in section 4.5.4. C_{max} (maximum number of redundant packets received in the CB scheme) was fixed to two; T_{max} (maximum waiting interval in the CB scheme) was set to 25 ms, 50 ms and 100 ms. Table 5.3 shows the results. The improved results indicate the benefit gained from

the CB scheme, which reduces the number of redundant broadcasts dramatically. The cost for such a reduction is reasonable as the average latency does not increase too much. Another benefit is the packet loss drops as each client reduces the number of broadcasts, which reduces the contention.

5.1.5 Mobility test

There are two goals of testing under node mobility. First is to measure the average disseminating latency from the server to the farthest client (assuming in real application, the farthest client is at most four hops away from the server), given that only the server is mobile. Second is to understand to what degree various patterns of mobility affect applications in multi-hop ad hoc networks and to find suitable solutions to overcome the problems encountered.

5.1.5.1 Network topology

Figure 5.2 shows the network topology used. One PDA acted as the source; other PDAs acted as relay nodes (routers). The laptop acted as the farthest client (destination). Routers 1 and 2 were about one and half meters apart while the gap between routers 3 and 4 was also around this distance. The thin dotted circles depict the transmission ranges of the source and the destination (assumed to be equal). Initially the routers were neighbours of both the source and the destination; the source was two hops away from the destination (via router 1, or 2, or 3, or 4). However, the pair of routers 1 and 2 was outside the transmission range of the pair of routers 3 and 4. During the test, the source moved randomly within an area, where it could be the neighbour of the destination and was always in the transmission range of routers 1, 2, 3, and 4.

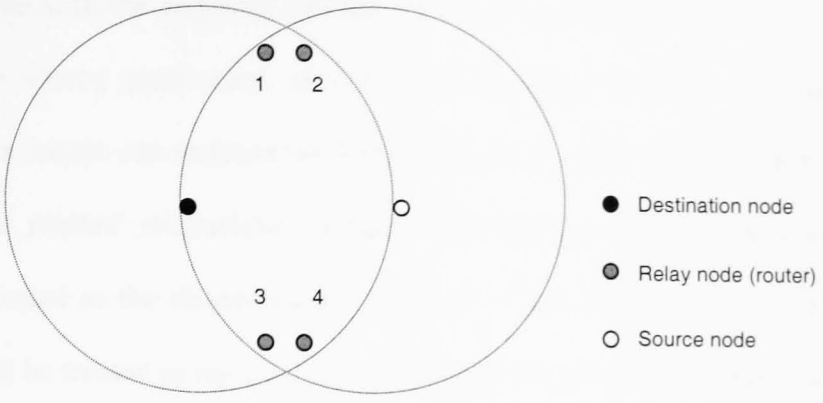


Figure 5.2: Initial nodes' positions for source node mobility test

5.1.5.2 Testing method

In each test, the source broadcast 500 UDP packets at the rate of one packet every 500 ms; each packet contained 512 bytes payload. The source's moving speed was about one meter per second (1 m/s). The destination followed the communication protocol design to rebroadcast new packets from the source. Given this specific test, the destination needed to modify the value of "packet type", making these packets as the acknowledgements from the destination. Hence the routers can distinguish received broadcast packets, as they needed to rebroadcast the new packets from the source as well as new acknowledgement packets from the destination. The routers used the field of "source route length" (see table 4.2) in packets to determine if a received packet was required to rebroadcast: the value of source route length in new packets from the source was "one" while the value was "three" for new packets from the destination. The routers discarded any packets for which the value of source route length was "two" or "four". Upon receiving these new packets, the routers appended their identities into the field of "source route" in the packets then rebroadcast. Naturally, the destination discarded its own acknowledge packets rebroadcast by the routers.

During the whole procedure, the source logged the following information (i) the departure time with the sequence number for each sent packet and (ii) the sequence number, the source route used, and the arrival time for every received packet. Therefore the source can measure the broadcast latency on 2-hop and 4-hop routes by receiving the routers' rebroadcast packets. Such latency on 2-hop or 4-hop routes could be mocked as the disseminating latency to 2-hop or 4-hop clients. The 4-hop latency could be treated as the time interval required to reach the farthest client in the network. The routers and the destination also logged the sequence number and the source route used for each of their received packets. The experiment was conducted three times.

5.1.5.3 Results under normal situation

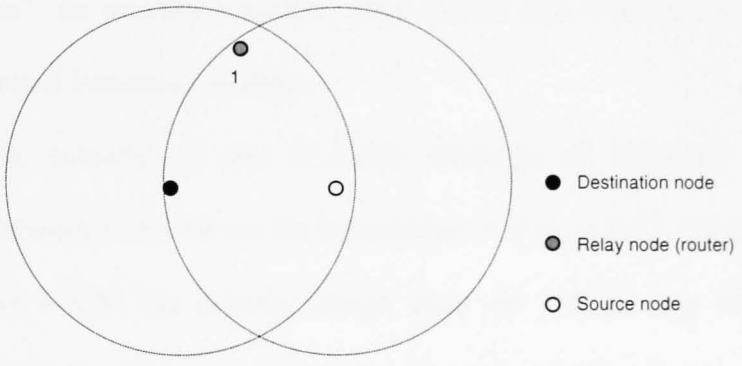
Normal situation was taken to mean that the source was outside the transmission range of the destination, requiring a 2-hop source route to the destination, such as (source->router 1 ->destination). The source was able to measure the average latency on 2-hop source routes was 22.5 ms and 57.8 ms on 4-hop source routes. The packet loss on the source, the routers, and the destination was almost zero. Therefore, it is inferred that in practice under such circumstance the latency is not a significant factor, given that users' response time is of the order of at least a few seconds.

5.1.5.4 Odd situation: source at the border of destination's transmission range

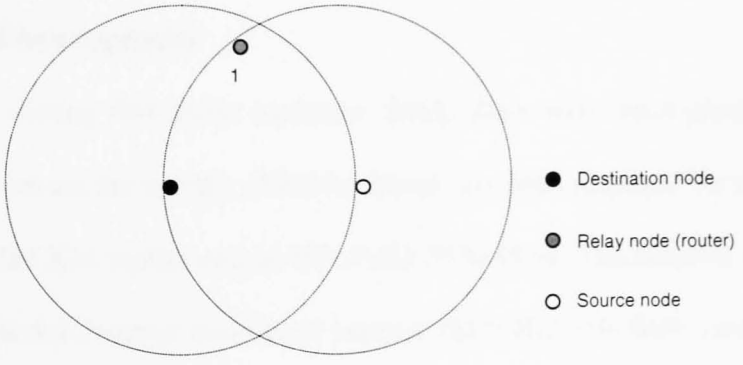
Surprisingly, sometimes the results were poor: the server lost over 30% of acknowledgement packets from the destination. By analyzing the logs at the source, the routers, and the destination, it was found that the destination received almost all the packets form the source, the routers received all the acknowledgement packets

from the destination, but the source did not receive all acknowledgements from the destination. Analysis of the logs showed that the lost acknowledgement packets all used the 1-hop source route (destination->source).

This could be explained in the following way. Whenever the destination received packets from the source for which the corresponding acknowledgement packets were lost, it was noted that the physical distance between the source and the destination was very close to their transmission range. Because of source mobility, the source route that the destination just learned was out of date quickly.



(a): Topology when the destination received a broadcast packet from the source



(b): Topology when the destination acknowledged the packet from the source

Figure 5.3: Source node mobility test

Figure 5.3 shows the simplified network topology: it only includes the source, the destination, and router 1. As shown in figure 5.3(a): at one moment the destination learned the 1-hop source route (destination->source). However, when the destination started to respond, this source route suddenly failed after the source moved out of the transmission range of the destination, as shown in figure 5.3(b). Nevertheless another available source route (destination->router 1->source) worked at that moment. During a test, coincidentally, the source moved out of and entered into the transmission range of the destination frequently. Then it led to the high packet loss of the destination's acknowledgement packets at the server for this test. The phenomenon is named "border problem": an unstable situation when sender and receiver are close to the border of their equal transmission range.

Border problem, actually, is one of many instances of reliability problem in Convenience Network (CN). Given the description in section 4.6.1, about the general design issues in a CN, the current design does not provide any mechanism to overcome this problem. What is required is a general solution whereby a client or a server can retrieve lost packets. Such a general solution is presented in chapter 6.

5.1.6 Impact of heterogeneity

At the time of writing this thesis (summer, 2005) there have been plenty of mobile computing devices on the market. (New hardware has been acquired for this research: an IBM ThinkPad X31 laptop and an HP iPAQ 5550 PDA. The detailed specification is shown in table 5.4. Both of them have internal IEEE 802.11b *WiFi* cards.) However, their computing abilities vary dramatically given that they use different types of CPU, have different memory size, and install different operating systems and software running environments. In such heterogeneous computing environment, it is necessary to acknowledge such differences especially when designing time-sensitive

applications, for example, the auction; otherwise it is unfair for the auctioneer and some bidders who carry compute-poor devices.

Unit	CPU	Memory size	Operating system	Software specification
Laptop	Intel Pentium M 1.4 GHz	512 M Bytes	Windows XP Professional	J2SE 1.3
PDA	Intel PXA 255 400 MHz	64M Bytes	Windows for Pocket PC 2003	IBM J9. Compatible with J2ME CVM & PP

Table 5.4 New hardware configuration and software specification

Testing was conducted to evaluate the impact of UDP packet payload size. Although an IP packet can contain up to 65534 bytes theoretically, the Winsock specification points out "it is inadvisable to attempt to send a broadcast datagram which is larger than the Maximum Transmission Unit (MTU) for the network. [126]" The value of MTU for wireless Ethernet is 1500 bytes. Regarding the IP header size is 24 bytes and UDP header size is 4 bytes, in application layer the maximum payload size is 1472 bytes. The testing method is straightforward: the identical application was running on different working platforms so that the differences could be drawn by comparing the results. The application and testing procedure is same as section 5.1.3. But another two new combinations were introduced into the working platform described as follows:

- Combination 1: The old model laptop (IBM T21) worked as the server; the old model PDA (HP Jornada 720) worked as the client (same as the working platform in section 5.1.1).
- Combination 2: The new model laptop (IBM X31) worked as the server; the old model laptop worked as the client.
- Combination 3: The new model laptop worked as the server; the new model PDA (HP iPAQ 5550) worked as the client.

The aim of this test is to show quantitative difference if users use different computing facilities and the best computing performance in the most ideal situation under various scenarios. Such results should be referred to as an important index (upper bound of performance) to evaluate the proposed applications. The results are shown in table 5.5. (About combination 2, if the roles of the new and old laptops are exchanged, the result remains unchanged. Thus the computing capabilities on the old and new model laptops are equivalent for this specific application.)

Payload size	512 bytes		1472 bytes	
Configuration	Average packet loss	Average latency (ms)	Average packet loss	Average latency (ms)
Combination 1	< 0.1%	20.7	< 0.1%	97.7
Combination 2	< 1%	10.1	< 1%	23
Combination 3	< 1%	11.7	< 1%	24

Table 5.5: Round trip latency between two nodes

5.1.7 Performance of TCP

The proposed applications work purely based on UDP mainly for its simplicity. In order to have a clear performance comparison between UDP and TCP, tests where the PDA uses TCP to deliver a packet to the laptop are conducted (configuration used is combination 1). Under such circumstance, the PDA is able to measure one-way 1-hop latency for TCP is connection-oriented communication. On average it takes the PDA 90 ms to send 512 bytes payload to the laptop. Most of the time is taken up to set up a reliable connection between the PDA and the laptop, as the latency for delivery of 1472 bytes payload is almost same. Obviously UDP outperforms TCP; it could be deduced that complex mechanisms of TCP almost certainly will lead to performance problems in ad hoc networks.

5.2 Simulation study: GloMoSim

Further evaluation of the proposed applications in real world environments will require more mobile devices than can be provided at the moment. In addition, it is difficult to organize a large number of volunteers operating mobile devices for the sake of reproducible scientific results. Hence simulation studies become invaluable as they allow extensive exploration of the design space. The simulator, GloMoSim [91] is chosen for this work.

5.2.1 Overview of GloMoSim

GloMoSim is a sequential and parallel library for simulating wireless networks. The layered communication protocol stack for wireless networks is designed to be extensible and modular, where each layer has its own APIs. Each module specifically simulates a wireless communication protocol in the protocol stack. A proposed protocol for one layer interacts with its direct layers via their APIs. PARSEC, a C-based parallel simulation language, is used to develop the library as well as programme new protocols and modules that can be added to the library.

5.2.2 Understanding GloMoSim

In order to understand the behaviour of GloMoSim, before reporting the simulation study results, what happened during a simulation run of simple flooding when two clients that are directly reachable rebroadcast received packets from the server is to be described first. The strange result is that the collision took place every time the two clients rebroadcast packets. Supposedly, it demonstrates the fact that GloMoSim accurately simulates the IEEE 802.11 MAC protocol from the theoretical view.

The clients in the simulator are identical: they have same wireless interface, same computing power and run same application. It concludes that each client should have

the same performance. In the simulation run as described above, each client received broadcast packets from the server at the same time, each client took an equal time to process the packet and would rebroadcast at the same time. According to the IEEE 802.11 MAC protocol, before the two clients rebroadcast a packet they waited for a short interval of DIFS. The medium had been idle for some time until the end of DIFS. Therefore each client supposed it can send the packet at once after DIFS. The two clients broadcast the packet simultaneously so that the collision happened right at that time. Furthermore the IEEE 802.11 MAC protocol does not recommend any acknowledgement or retransmission mechanism for broadcasting packet. Hence, all of rebroadcasting packets were lost in the simulation run.

Under the circumstance described above, a very short random delay has to be applied before clients rebroadcast packets in simulation studies. However in practice each mobile device, even if identical, most likely will take slightly different processing time for the same packet. Thus, in practice each mobile device is unlikely to send the packet exactly at the same time. Then the mechanism of CSMA/CA will continue to work to guarantee delivery of broadcast packets, avoiding collision.

5.2.3 Setting the simulator

Simulation study is to simulate the proposed application and not just the communication protocol. A separate file in the application layer implements the proposed application combining routing function. The new header files are written and some related ones are modified. The file that implements Internet Protocol is modified in order to inform the simulator to bypass routing function in the network layer and send all packets directly to the application layer. At last these modified and new files are compiled and linked to the simulation platform which works as an independent module in the application layer. Before simulating the application, the

configuration files need to be modified so that the simulator knows exactly where it can find the proper module and what it will do.

Key parameters that were fixed in the simulation studies were transmission range (113 meters), payload size (512 bytes), transport protocol (UDP), transmission rate (2 Mbps), MAC protocol (the IEEE 802.11 DCF protocol). Each test was repeated three times with varied seeds. Considering the simulator behaviour for rebroadcast described in section 5.2.2, the threshold for a very short random delay (W_{max}) on clients in order to simulate rebroadcast at slightly different times, is set to 200 microseconds.

5.3 Simulation study: preliminary testing

In this section the testing was to compare the results from the field testing with the simulation studies under the same conditions. The performance metrics, the testing method and the network topology are same as sections 5.1.2 to 5.1.4. Tables 5.6 and 5.7 show the results for simple flooding and the CB scheme.

Number of clients	1	2	5
Average latency (ms)	5	6.5	10.2
Average packet loss (%)	0	2.7	10.4

Table 5.6: Simulation of simple flooding in 1-hop network

T_{max} (ms)	25	50	100
Average latency (ms)	20.4	19.1	30.5
Average packet loss (%)	0.3	0.3	0
Average saved ratio (%)	53.7	54.7	57.6

Table 5.7: Simulation of the CB scheme in 1-hop network, five client nodes

5.3.1 Average latency

Comparing table 5.2 with table 5.6, readers can see that the simulation results are much optimistic regarding the metric latency. The simulator neglects at least one important factor: it does not take into account the processing delay on each node as the data pass through the application layer to the physical layer, as shown in figure 5.4.

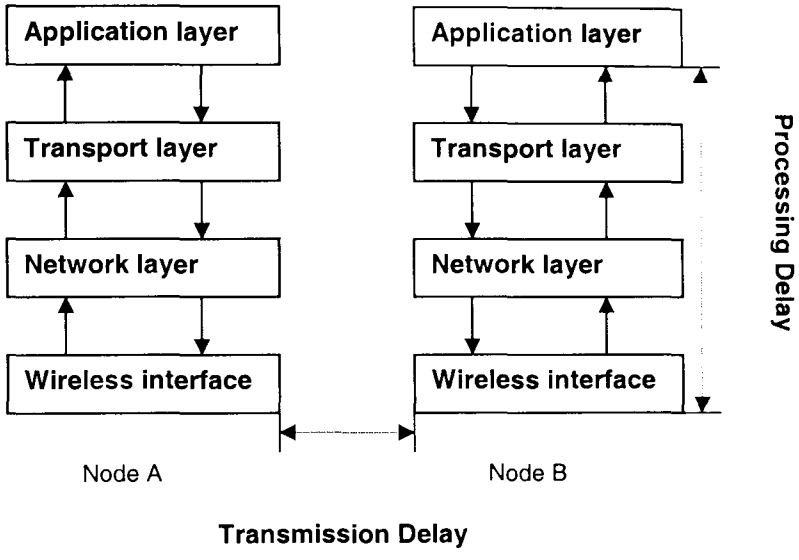


Figure 5.4: Conceptual view of data flow

The processing delay happens in two directions (down from the application layer to the wireless interface; up from the wireless interface to the application layer) and the delay could be assumed to be equal in each direction. P_A and P_B are chosen to denote the overall processing delay on node A and B; D_t is used to denote the round-trip transmission delay between A and B. Then latency $D = P_A + P_B + D_t$. Given the results in table 5.5 (for 512 bytes payload), the theoretical value of D_t is 5.3 ms[1]. Under this circumstance of ideal network situation (only a sender and its receiver in the network), the processing delay for the laptop, for the new model PDA, and for the

[1] The formula is: Round-trip latency =2 * Packet size / Bandwidth. Here the wireless Ethernet packet size is 540 bytes (UDP payload size is 512 bytes; headers' size is 28 bytes), Bandwidth is 2 Mbps.

old model PDA can be calculated as 2.4 ms, 4 ms, and 13 ms respectively. Even so in the real world the processing delay is affected by the density of the network (see table 5.2). The denser the network, longer is the processing delay.

5.3.2 Packet loss

However, comparing table 5.2 with table 5.6, the packet loss in simulation study is much higher than field test. The simulation result closely relates with the value of W_{max} (threshold for a very short random delay on clients in order to avoid rebroadcasting at exactly the same time, see section 5.2.3). If the value of W_{max} was set to 13 ms, the simulation result was very close to the field test. However this particular value of W_{max} is too long to be true in the real world. Therefore the original value is kept in order to simulate the real scenario as closely as possible.

5.4 Extensive simulation study: 1-hop ad hoc network

With the help of simulator it is feasible to estimate hop-by-hop latency under various network conditions and have reproducible results. Therefore the efficiency and the performance of unicast and broadcast according to varying network members and traffic load can be examined. In this section the testing focuses on the performance of unicast, broadcast, and the CB scheme in 1-hop ad hoc network.

The network topology and the testing method were same as section 5.1.3: various numbers of nodes were put close together (all of them were in the transmission range of each other); two, three, six, ten, twenty, thirty, and forty nodes were put into the network. The packet payload size was fixed to 512 bytes or 1472 bytes. Here the end-to-end latency was from a client to the server, rather than the round-trip delay from (the server->the client->the server).

Payload size	512 bytes		1472 bytes	
Number of nodes	Average packet loss	Average latency (ms)	Average packet loss	Average latency (ms)
2	0	3.0	0	6.9
3	0	5.9	0	11.7
6	0	11	0	22.6
10	0	17.9	0	37.1
20	0	35.4	0	73.8
30	0	53.5	0	111.1
40	0	71.9	0	148.7

Table 5.8: Performance of unicast in 1-hop ad hoc network

Payload size	512 bytes		1472 bytes	
Number of nodes	Average packet loss	Average latency (ms)	Average packet loss	Average latency (ms)
2	0	2.5	0	6.5
3	2.7%	4	2.7%	9.7
6	10.4%	7.7	10.4%	18.8
10	19.4%	12	19.4%	29.8
20	35.3%	21.5	35.3%	53.3
30	44.7%	29.7	44.7%	73.9
40	50.5%	37.4	50.5%	93

Table 5.9: Performance of broadcast in 1-hop ad hoc network

5.4.1 Performance of unicast

The results are shown in table 5.8. It shows that unicast is able to provide a reliable mechanism for delivering packets to the destination, given the cost of exchanging *RTS*, *CTS*, and *ACK* control frames. The cost is expensive when the density of network becomes higher. P. Poupyrev et al estimate that the latency is of the order of minutes in a filed environment [85].

5.4.2 Performance of broadcast

The results in table 5.9 clearly show the trend of broadcast storm described in section 4.5.1: when the network becomes dense, the high contention to access the medium will lead to high congestion and high collision. The packet loss is close to 20% when there are only ten nodes in the network. The unreliability exists as in the IEEE 802.11 specification *RTS/CTS* control frames are not exchanged before a node broadcasts packets. The exchanging of *RTS/CTS* frames guarantees reliable transmission of unicast packets though it leads to low efficiency; under the same network conditions the end-to-end latency by unicast is longer than by broadcast. In section 5.3, the interval for which the server broadcast a packet (500 ms) is much longer than the processing delay on each client (less than 200 ms). However, if the interval is close to or shorter than clients' processing delay, the performance will be much poorer. (It is discussed in section 5.5.)

T_{max}(ms)	25				50				100			
Number of nodes	10	20	30	40	10	20	30	40	10	20	30	40
Average latency(ms)	9.3	9.6	11.6	13.9	11.9	9.3	9.3	10	18.3	11.9	9.9	9.2
Average packet loss (%)	1.9	5.7	9.2	12.8	0.2	1.9	3.9	5.1	0.3	0.8	1.4	2
Average saved ratio (%)	64.1	75	77.8	79.3	72	82.8	86	87.3	75.1	86.3	89.9	91.4

Table 5.10: CB scheme in 1-hop ad hoc network, payload size is 512 bytes

5.4.3 Effectiveness of the CB scheme

Table 5.10 shows the results when nodes in 1-hop network used the CB scheme with the varying values of T_{max} (maximum waiting interval in the CB scheme), in which the value of C_{max} (maximum number of redundant packets received in the CB scheme) was set to two and the packet payload size was 512 bytes.

If C_{max} is set to a fixed value the overall performance directly relates with the density of the network and the value of T_{max}. Generally the CB scheme effectively reduces the number of rebroadcast packets. Especially when T_{max} is set to a large value, the packet loss drops sharply and the saved ratio is close to the theoretical value (N- C_{max} +1) / N (N stands for total number of nodes in the network). When the network has not many members, for example, six nodes, a short T_{max} is enough to guarantee good performance.

Actually random waiting time (RWT) in the CB scheme could be thought as the lengthened contention window. The larger the value of T_{max} is set, the less is the probability of collision. Each node's RWT is evenly distributed along the interval of T_{max}; the nodes that have long RWT will at last stop rebroadcasting because they have received enough redundant packets during their RWT. The real latency is determined theoretically by the first (C_{max} -1) nodes' performance. Thus setting T_{max} to a large value does not necessarily lead to prolonged latency and the simulation results verify it. Hence when network becomes dense each node should apply a large value of T_{max} in order to avoid collisions.

In any case, flooding in denser network will generate more packets. For example, when there are 40 nodes inside the network, the packet loss still keeps at two per cent even if T_{max} is set to a large value of 100 ms. This may not be good enough for some applications. Another way that will have the same impact on the network is increasing the packet payload size.

Tables 5.11 and 5.12 show the results when 10 or 40 nodes using the CB scheme in a 1-hop ad hoc network (packet payload size is 1472 bytes). Comparing table 5.11 with table 5.10, in a marginally dense network (10 nodes), it shows: under the same network conditions when the packet payload size significantly increases, T_{max} must be

set a large value in order that the application's performance could keep at reasonable level.

T_{max}(ms)	25	50	100
Average latency (ms)	24.8	22.4	26.2
Average packet loss (%)	6.2	2	0.9
Average saved ratio (%)	40.4	59.7	70

Table 5.11: CB scheme in 1-hop ad hoc network, 10 nodes, payload size is 1472 bytes

T_{max}(ms)	25	50	100
Average latency (ms)	N/A	N/A	26.9
Average packet loss (%)	N/A	N/A	7.6
Average saved ratio (%)	N/A	N/A	83.9

Table 5.12: CB scheme in 1-hop ad hoc network, 40 nodes, payload size is 1472 bytes

5.5 Extensive simulation study: multi-hop ad hoc network

In this section, the testing focuses on the performance of unicast, broadcast, and the CB scheme in more complex environment: a multi-hop ad hoc network. Besides, factors that significantly affect the performance are pinpointed so that they could give guidance on how the system could choose a better way to rebroadcast packets in real applications.

5.5.1 Impact of parameter: radius of the network (H_{max})

It is commonly thought that a client closer to the server has advantages, like always receiving packets from the server or being able to send packets to the server quickly. However the simulation will show that it is not always true and it really depends on the density of network and how the server broadcasts packets.

A network topology shown in figure 5.5 was used to conduct the test in a static mode. 50 nodes were "uniformly" placed in an area which was 200 metres long by 200 metres wide (denoted as [200m *200m]). A node near the centre of the area was designated as the server and others as clients. Most of clients were in the transmission range of the server; only four clients were two hops away.

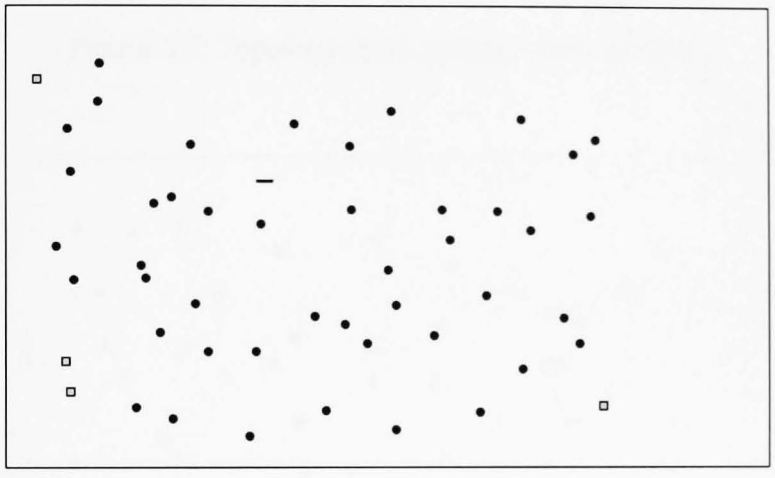


Figure 5.5: Topology of 50 nodes in [200 m*200m]

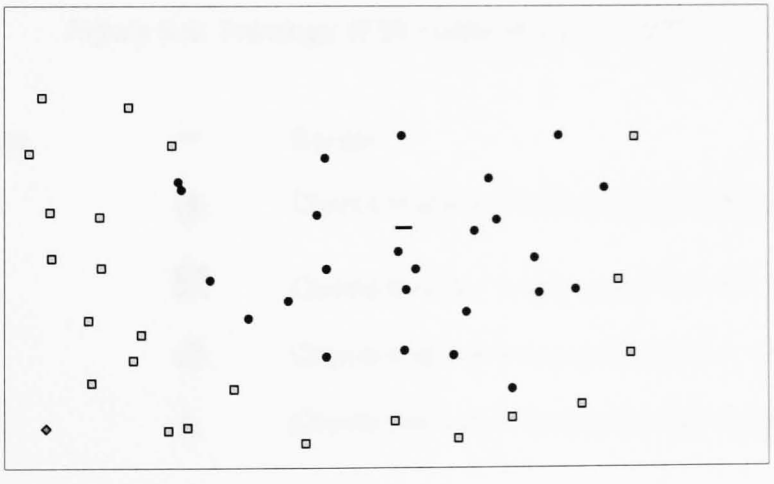


Figure 5.6: Topology of 50 nodes in [300 m*300m]

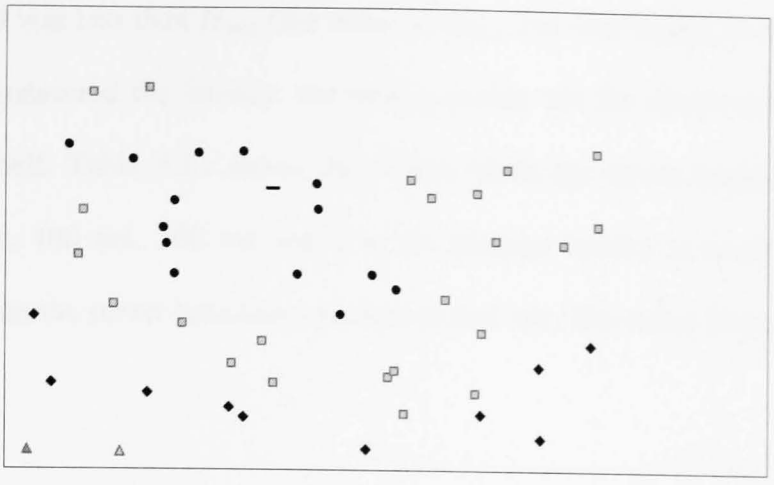


Figure 5.7: Topology of 50 nodes in [400 m*400m]

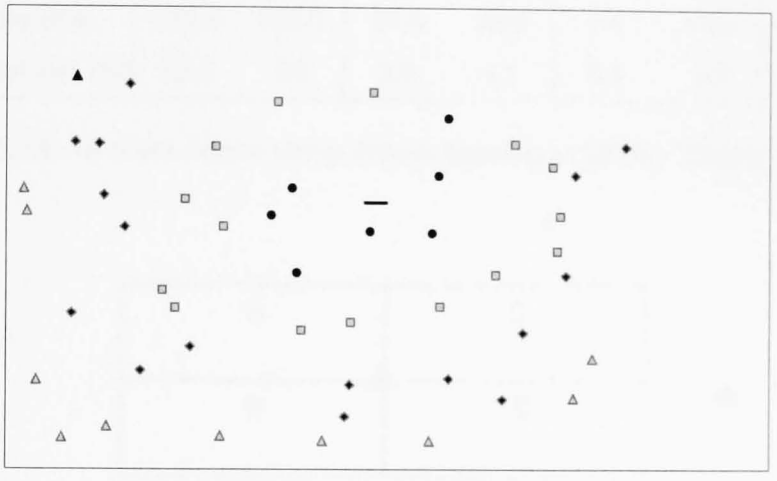


Figure 5.8: Topology of 50 nodes in [500 m*500m]

Key

- Server
- Clients that are 1-hop away from the server
- Clients that are 2-hop away from the server
- Clients that are 3-hop away from the server
- Clients that are 4-hop away from the server

The server broadcast 100 packets including time-stamp information to the network. Clients rebroadcast the new packets from the server and the source route length the

packets used was less than H_{max} (the value of H_{max} was four hops.) At the same time each client measured the latency: the time a packet left the server until the packet arrived at itself. Table 5.13 shows the results while the server broadcast a packet every 50 ms, 100 ms, 200 ms and 2 s; the average latency is much longer than expected when the server broadcasts packets at fast rate, like every 50 ms or 100 ms a packet.

Broadcast interval	50 ms		100 ms		200 ms		2 s	
Distance to server (hop)	1	2	1	2	1	2	1	2
Average latency (ms)	579.6	602.3	24.4	33.9	7.4	13.9	2.5	8.8
Average packet loss (%)	39.4	60	5.9	6.1	0.3	0.3	0	0

Table 5.13: 49 client nodes using simple flooding in [200m *200m], H_{max}=4

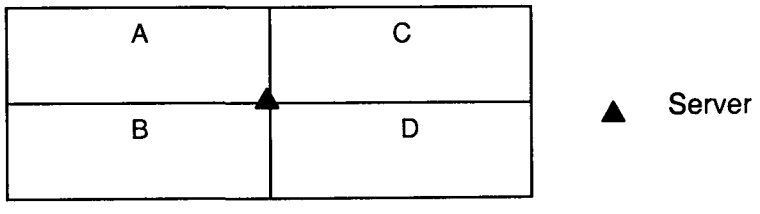


Figure 5.9: Conceptual view of [200m * 200m]

Naturally, most people think all clients that are in the transmission range of the server should receive the identical broadcast packet at the same time. But this is not always true. Figure 5.9 shows the conceptual view of the area which has size [200m *200m]. It is divided into four parts and the server is in the centre. While the server kept promptly broadcasting packets, clients had to follow the same rate by processing and rebroadcasting received packets coming from different sources. Therefore collisions frequently happened at some parts of the network but not all. Consider a specific scenario, at a given time nodes in part C did not receive the broadcast packet from the

server because collisions happened at that time when they were rebroadcasting previously received packets. Eventually these nodes would receive the packet from their neighbours that had successfully finished rebroadcasting in part B or D, though nodes in part C were in the transmission range of the server. The worst observed situation is that a packet reached a node in part C via three hops (Server–>a node in part A–>a node in part B–>a node in part C).

The fact indicates that if the server broadcasts packets in a rapid rate like every 50 ms a packet it will cause a chain effect on clients' side: when clients cannot process received packets fast enough (processing time on clients should be shorter than the interval of which the server broadcasts a packet), they will process the new incoming packets even slowly. In addition, unnecessarily setting H_{max} to four hops in a 2-hop network is another factor that degraded the performance; it generated too many avoidable broadcast packets. Table 5.14 shows the improved results when H_{max} was set to two hops.

Broadcast interval	50 ms		100 ms		200 ms		2 s	
Distance to server(hop)	1	2	1	2	1	2	1	2
Average latency (ms)	403.5	427.3	12.1	18.7	6.6	13.1	2.5	8.6
Average packet loss (%)	35.3	38	2.6	2.9	1.3	1.4	0	0

Table 5.14: 49 client nodes using simple flooding in [200m *200m], H_{max}=2

5.5.2 Effectiveness of counter-based scheme

Table 5.15 shows the result when clients used the CB scheme to rebroadcast packets. The performance is poor for 2-hop clients: their packet loss is quite high. It demonstrates a drawback of the CB scheme: the scheme is based on probability view not on neighbourhood information. As shown in figure 5.5, the density was not very high around 2-hop clients. But these 2-hop clients' upstream neighbours may be in the

dense environment; these nodes stopped rebroadcasting as they used the CB scheme and had received enough redundant packets. These nodes were not able to know that some of their downstream neighbours were solely depending on their forwarding packets. Under such circumstances, these 2-hop clients should inform their upstream neighbours of using simple flooding rather than the CB scheme.

Distance to server	Average latency (ms)	Average packet loss	Average saved ratio
1-hop	2.5	0	50%
2-hop	9.1	49.4%	2.4%

Table 5.15: 49 client nodes using the CB scheme in [200*200], server's broadcast interval is 6 s, H_{max}=4, T_{max}=25 ms

5.5.3 Impact of geographical size on broadcast and unicast

The system design heavily exploits unicast and broadcast through which the network members communicate with each other. Therefore the performance of these two methods in various conditions directly affects the performance of applications.

Depending on the nature of applications, when clients receive a new broadcast packet form the server, their behaviours can fall in three modes. Mode 1: rebroadcasting the packet then unicasting the response to the server at once. Mode 2: rebroadcasting the packet, waiting some time then unicasting the response to the server (assuming all clients will unicast at the same time). Mode 3: rebroadcasting the packet, randomly waiting some time then unicasting the response to the server. Mode 1 represents the case where an application runs as a demon in the background, which automatically implements the application logic. Mode 3 is closer to the situation when users receive a packet in the network, they think for a short while then give the response. Mode 2 does not represent the real world. However it stands for a theoretical situation in which all clients respond at the same time.

The simulation area size ranged from [200m *200m] (almost a 1-hop network, as shown in figure 5.5), [300m *300m] (almost a 2-hop network, only one node was three hops away from the server, as shown in figure 5.6), [400m *400m] (almost a 3-hop network, two nodes are four hops away from the server, as shown in figure 5.7), and [500m *500m] (a 4-hop network, as shown in figure 5.8). The server broadcast 100 packets at the rate of one packet every six seconds. The results are presented from tables 5.16 to table 5.19. (The default value of H_{max} was four hops. Note that in the following tables, packet loss figures are for broadcast only.)

	Mode 1		Mode 2		Mode 3	
Distance to server (hop)	1	2	1	2	1	2
Average packet loss (%)	0	0	0	0	0	0
Average broadcast latency(ms)	2.6	9.1	2.5	8.8	2.6	8.9
Average unicast latency(ms)	167.3	233.8	92.8	143.3	3.6	7.3

Table 5.16: Performance of unicast and broadcast, 50 nodes in [200 m *200m]

	Mode 1			Mode 2			Mode 3		
Distance to server (hop)	1	2	3	1	2	3	1	2	3
Average packet loss (%)	0	0	0	0	0	0	0	0	0
Average broadcast latency (ms)	2.6	14.8	42.4	2.5	16.2	33.5	2.5	15.7	36.1
Average unicast latency (ms)	169	284	305	102	205	248	3.7	8.7	12.8

Table 5.17: Performance of unicast and broadcast, 50 nodes in [300 m *300m]

	Mode 1				Mode 2				Mode 3			
Distance to server (hop)	1	2	3	4	1	2	3	4	1	2	3	4
Average packet loss (%)	0	0	0.2	18	0	0	0.2	12	0	0	0.2	13
Average broadcast latency (ms)	2.6	19.1	43.9	84.5	2.6	17	34.5	58.1	2.6	17.5	35	61.4
Average unicast latency (ms)	159	310	361	350	108	240	309	334	4	9.6	13.7	161

Table 5.18: Performance of unicast and broadcast, 50 nodes in [400 m *400m]

	Mode 1				Mode 2				Mode 3			
Distance to server (hop)	1	2	3	4	1	2	3	4	1	2	3	4
Average packet loss (%)	0	0.7	2.5	18.5	0	0.1	1.4	15.7	0	0	1	13.1
Average broadcast latency (ms)	2.5	17.6	44.4	82.8	2.6	15.7	33.7	54	2.6	15.8	34.2	54.6
Average unicast latency (ms)	121	285	364	384	111	241	335	386	4.3	9.7	14.5	17.9

Table 5.19: Performance of unicast and broadcast, 50 nodes in [500 m *500m]

The results again show low efficiency of unicast when nodes send packets at or almost at the same time (under mode 1 and mode 2) due to the requirement of exchanging *RTS*, *CTS*, and *ACK* control frames. But the performance of unicast is reasonable under mode 3 when transmission takes place sparsely. The performance of broadcast is similar under various modes and area sizes. The conclusion is that the proposed applications will run well as users' responses are much slower than the aggregation of transmission delay and processing delay.

5.5.4 Impact of source route length

Regarding the metric of packet loss, tables 5.18 and 5.19 show that clients for which distance to the server is more than two hops are at a disadvantaged situation than those closer to the server. Especially, packet loss of 4-hop clients jumps greatly. This could be explained as follows. Under the current communication protocol, to some extent, the source route length of packets that a client rebroadcasts is longer than the client's real distance to the server. Given the value of H_{max} (radius of the network) is four hops, some of 4-hop clients' upstream nodes which are in reality three hops or even two hops away from the server, stop rebroadcasting as the source route length of their received packets have reached the limit, H_{max}.

For example, considering the results as shown in table 5.19 (mode 3), table 5.20

shows the corresponding probability that a client learns source routes to the server that have different lengths. After the server broadcasts a packet, the first received packet at each 1-hop client is almost directly from the server. However, with regard to 2-hop clients, the chance that the first received packets use a 2-hop source route reduces to 85.7%; 12.7% of these packets use a 3-hop source route; to make things worse, the remaining 1.6% packets even use a 4-hop source route. As for 3-hop clients, 80.8% of their first received packets use a 3-hop source route and the remaining 19.2% use a 4-hop route. (Due to the value of H_{max} is four hops, the source route length of received packets at 4-hop clients has to be four hops. In this test, the number of 3-hop clients was more than the number of 4-hop clients; otherwise, these 4-hop clients could have even higher packet loss.)

	Probability to learn a *m* hops source route			
Distance to server (hop)	*m* =1	*m* =2	*m* =3	*m* =4
1	0.998	0.002	0	0
2	N/A	0.857	0.127	0.016
3	N/A	N/A	0.808	0.192
4	N/A	N/A	N/A	1

Table 5.20: Probability of clients to learn source routes to the server that have different lengths (clients choose the source route the first received packet used, H_{max} =4)

Under the same scenario used in table 5.20, whether clients use simple flooding or the CB scheme to rebroadcast packets, most of them will receive redundant packets. Among these redundant packets, it turns out that a packet taking a longer route sometimes arrives at a client before a packet taking a shorter route. If clients rebroadcast the packets that have taken the shortest source route rather than the first received packet, table 5.21 shows the corresponding probability that a client learns source routes to the server that have different lengths. Comparing table 5.20 with table

5.21, for example, about 3-hop clients, they could rebroadcast extra 11% more of received packets so that the packet loss of 4-hop clients can be reduced. The revised communication protocol will make use of this method and the detail is presented in chapter 6.

	Probability to learn a m hops source route			
Distance to server (hop)	m =1	m =2	m =3	m =4
1	0.998	0.002	0	0
2	N/A	0.973	0.017	0.01
3	N/A	N/A	0.92	0.08
4	N/A	N/A	N/A	1

Table 5.21: Probability of clients to learn source routes to the server that have different lengths (clients choose the shortest source route among all arrived packets used, H_{max} =4)

5.5.5 Miscellaneous

Table 5.18 (mode 1) shows an interesting result: on average the unicast latency of 3-hop clients is slightly longer than that of 4-hop clients. Statistically, the data collected for measuring unicast latency of 4-hop clients were based on the inadequate samples with an uneven distribution, since there were only two 4-hop clients in a corner of the network. Next, as described above, a 3-hop client could learn and use a 4-hop source route to respond to the server. At last, it may be due to high level of network congestion and multiple access interferences along some of 3-hop clients' source routes which 4-hop clients did not take.

5.6 Simulation study: proposed applications

In this section simulation studies were conducted in mobile mode in order to assess the performance of the proposed applications. (However, due to limited functions provided by GloMoSim and the peer-to-peer nature of each chatting session, Chatting application cannot be simulated properly.)

5.6.1 Simulation of Auction

Terrain size was set to [500m *500m] and 50 nodes were set to be mobile, uniformly distributed in the area. Each node followed "random-waypoint (WP)" mobile model coming with GloMoSim. The WP model introduces "pause time" after which a node changes its moving direction and/or speed. Initially, a node stays in one location for a certain time (pause time). Once this time expires, the node randomly chooses a destination in the simulation area and a speed that is uniformly distributed between Minimum Speed and Maximum Speed. Then the node moves toward the chosen destination at the selected speed. Upon arrival, the node pauses for a specified time before starting the process again. Table 5.22 lists the low and high mobility parameters for the test.

	Low mobility	High mobility
WP-Pause period	3 Minutes	2 seconds
Minimum speed	0	1 m/s
Maximum speed	1 m/s	2 m/s

Table 5.22: Mobility parameters

A node near the centre of the simulation area was designated as the server. The server broadcast 20 packets in 12.5 minutes to the network, which the first five packets mocked as auction information packets (this period was the preparing period) and others as bidding information packets (this period was the bidding period and the auction would end after 15 rounds). During the preparing period every one minute the server broadcast a packet and clients should respond in 30 seconds if they received this packet. During the bidding period the server broadcast a packet every 30 seconds and clients should respond in 20 seconds if they received the packet. How many clients can successfully join the auction and how many times they had to try during

the preparing period under high and low mobility scenarios are main of interest. Results are shown in table 5.23: most of the interested participants did manage to join the auction (assuming all nodes were interested in the auction).

Round number	Nodes join (low mobility)	Nodes join (high mobility)
1	27	26
2	4	11
3	4	5
4	3	2
5	2	1
Fail	9 (3 nodes have been in isolation)	4 (3 nodes have been in isolation)

Table 5.23: Successful registration under low and high mobility

Some of nodes failed to join the auction because:

i. The server did not receive responses from interested clients due to the failure of source routes.

ii. Clients did not receive the auction information packets from the server due to (i) network partition happened so that three clients were isolated and (ii) collisions happened or the clients' upstream nodes stopped rebroadcasting.

It is commonly thought that high mobility possibly has negative effect on finding a route for a node; the just-learned-route failed because some nodes along the route moved away suddenly. But from the application perspective high mobility creates a chance to attract more potential users. Table 5.23 shows that under high mobility more users joined the application.

About what happened during the bidding procedure the results were quite "ad hoc". Though 40 out of 49 nodes joined the auction, for each bidding round, the number of responses the server received varied from 5 to 38. In the real application this could happen, bidders may not always respond every round or most of them bid at the last

moment. However, by checking the nodes' trace in the simulator, it showed that some nodes had been walking away from the server until they became isolated. Before the auction started, potential bidders' movement could be random in the network. But if they knew they had joined the auction their movement was unlikely to be aimless. At least if some ones reckoned they were out of contact with the server they should try walking for better connection, for example, the bidder moved toward the centre of the place from a corner. Another heuristic finding is worth noting: except isolated nodes, if a node failed to receive a broadcast packet, at least one of its neighbours had received it. This will be exploited to improve the system design.

5.6.2 Simulation of Bingo

The preparing period of Bingo game is same as Auction but the server will broadcast 75 packets during the gaming period as there are 75 Bingo balls needed to be announced during the real game. Therefore the gaming period should be treated as the prolonged bidding period except only a winner would end the game early as the server confirms it really had "BINGO". Considering the simulation result of what happened during the bidding period of an auction reported above, the simulation of Bingo game was not fully conducted. However it should assume the mobility is very low and players only move occasionally for trying reconnecting to the network if they think they are lost.

5.7 Further discussion

In this section, simulation study results about the performance of TCP and the complex system behaviours of IEEE 802.11 ad hoc networks are reported. In addition, research issues and the latest progress in the field of wireless network simulators are presented.

5.7.1 Performance of TCP in ad hoc networks

Many simulation studies have looked at the performance of TCP in multi-hop ad hoc networks. Several performance problems have been pointed out. Such phenomena do not appear, or appear with less intensity, when UDP is used.

5.7.1.1 Influence of mobility

Node mobility may severely degrade the performance of TCP in MANET [127, 128, 129, 130] because TCP lacks the ability to efficiently manage mobility effects like route failures, routing overheads, and route changes. Hence, packets will be lost or delayed. However, TCP adversely reacts to these events, misinterprets them as congestion signals and then activates the congestion control mechanism. This leads to unnecessary retransmissions and throughput degradation.

5.7.1.2 Influence of network topology

Even in static environment, the performance of ad hoc network is strongly limited by the interaction between neighbourhood nodes [131]. An example string topology is shown in figure 5.10. Node 1 is the sender and node 6 is the receiver. A solid circle denotes a node's transmission range; the dotted circle denotes node 4's physical carrier sensing range within which the other nodes detect a transmission. Hence, node 4's data packet transmission will interfere with *RTS* frames sent from node 1 to node 2, preventing node 2 from receiving nodes 1's *RTS* transmissions or sending the corresponding *CTS* frames. Therefore the expected maximum bandwidth utilization of a chain of ad hoc nodes is 1/4 [131]. However, the simulation shows an even worse result: the bandwidth utilization reaches only 1/7 [131]. This discrepancy is due to the inability of the IEEE 802.11 MAC protocol to find out the optimum schedule of

transmission by itself. The sender, node 1, actually could inject more packets into the chain than the subsequent nodes can forward.

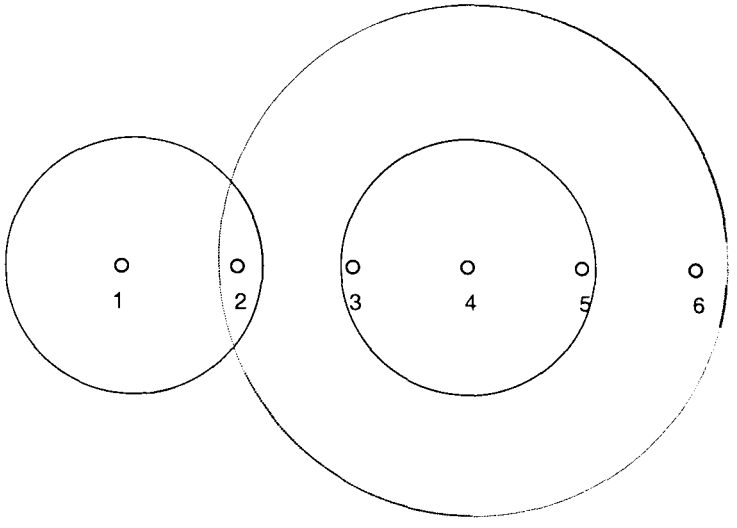


Figure 5.10: A string topology

5.7.1.3 Effects of the interaction between MAC protocol and TCP mechanisms

The interaction of some features of the IEEE 802.11 MAC protocol with the TCP mechanisms could lead to several unexpected and serious problems. Through simulation studies, the problems that may affect the TCP performance in a multi-hop ad hoc network are listed as follows [132, 133]:

i. The instantaneous throughput of a TCP connection may be very unstable even when this is the only active connection in the network (instability problem).

ii. In case of two simultaneous TCP connections, it may happen that the two connections cannot coexist: when one connection develops, the other one is shut down (incompatibility problem).

iii. With two simultaneous TCP connections, if one connection is single hop and the other one is multi-hop, it may happen that the instantaneous throughput of

the multi-hop connection is shut down as soon as the other connection becomes active (even if the multi-hop connection starts first). There is no chance for the multi-hop connection once the single-hop connection has started (single-hop unfairness problem).

The above problems have been revealed in a chain topology as shown in figure 5.10.

5.7.2 Influence of IEEE 802.11b standard

In addition to transmission rates at 1 Mbps and 2 Mbps, the IEEE 802.11b standard enables faster transmission at 5.5 Mbps and 11 Mbps while still guaranteeing the interoperability with the IEEE 802.11 standard. "A basic rate set" containing the data transmission rates that all nodes within a network will be capable of using is defined; this ensures coexistence and interoperability among these nodes which could use different rate to receive and transmit.

Since different transmission rates are used for control and data frames, different transmission ranges and carrier-sensing ranges may exist at the same time in an IEEE 802.11b network, thus the system behaviour becomes complex. For example, *RTS/CTS* frames are transmitted at 2 Mbps with related transmission range around 90 meters; data frames are transmitted at 11 Mbps thus its range limits at around 30 meters [134]. In practice, these two transmission ranges are much shorter than assumed in simulation studies, not constant but variable and dependent on several factors [134]. The physical carrier sensing range, which is almost independent from the data rate and is approximately 200 meters, is important to understand the behaviour of an IEEE 802.11b ad hoc network: even though nodes are outside their respective physical carrier sensing range, they may still interfere if the physical distance between them is lower than 350 meters [134].

5.7.2.1 Grey zone problem in IEEE 802.11b ad hoc networks

An important problem named the "communication grey zone" [135], which relates to the different transmission ranges of control and data frames, is reported. When H. Lundgren et al measure the performance of their implementation of AODV, they observe an unexpectedly large amount of packet losses, especially during route changes. They find that this coincided with specific geographic areas that are named communication grey zones. In such areas, data packets cannot be exchanged though the HELLO packets indicate that neighbours are reachable. This leads to a systematic mismatch between the route state and the real work connectivity [135]. The properties of HELLO packets that differentiate them from data packets and contribute to the occurrence of the problem are identified. These properties and their effects are summarized below [135]:

i. Transmission rate. HELLO packets are broadcast at the basic rate (2Mbps). But data packets are unicast at high speed (5.5 Mbps or 11 Mbps). Therefore HELLO packets have a larger transmission range than data packets.

ii. No acknowledgement. Broadcast in the IEEE 802.11b is transmitted without acknowledgement. Therefore HELLO packets are not guaranteed to be sent over bidirectional links.

iii. Small packet size. The smaller size of HELLO packets result in a lower probability of these being affected by transmission errors, and minor chances of colliding with other packets. This makes it more likely for a HELLO packet to reach a receiver than data packets, especially over poor links.

iv. Fluctuating link. At the transmission border, communication tends to be unreliable due to fluctuating quality of links. It may occur that stable and longer routes can be replaced by shorter but unreliable ones.

5.7.3 Scalability of wireless network simulators

Discrete event simulators fundamentally are useful tools for exploring the design space of wireless networks. However they suffer scalability problems. On the other side, the emerging applications, like a military exercise involving a division of soldiers with numerous wireless devices, call for even larger scale simulation studies. This sub-section reports the latest progress in the field of large scale simulation studies.

5.7.3.1 Scalability problems

Most published simulation results are on a small scale (no more than 100 nodes), for a short duration (up to 1000 seconds of simulation time) and over a small geographical area (limited to a few square kilometres). For example, S. R. Das et al complain that "slow simulation speed and large memory requirement of the ns-2 model prevented us from using larger networks at this point [94]". They have to reduce the simulation time to 500 seconds when they simulated the new filed configuration: 100 nodes in a [2200m * 600m] area. Independently other researchers report the same poor performance of ns-2: it does not scale well.

Regarding GloMoSim, due to PARSEC's large per-entity memory requirements, a technique named "node aggregation" is implemented, wherein the states of multiple simulation nodes are multiplexed within a single PARSEC entity. While the technique effectively reduces memory consumption, it brings on performance overhead as well as increases code complexity.

5.7.3.2 An improved version of ns-2

An improved version of ns-2 [136] is presented. The modified ns-2 simulator is based on the idea of cutting the computation by accurately locating the nodes that are

affected by a transmission. The simulation area is divided into cells to form a grid. A function is implemented to calculate the position of a node in the grid to determine the cell that the node belongs to. Another function is able to get the list of nodes affected by the current transmission. Therefore not all nodes are involved in the computation whereas the original ns-2 regards all nodes on a channel. In addition, nodes can be stored in a double-linked list ordered by their X-coordinates to improve the performance. The list is updated when nodes move to new positions. An algorithm can determine the set of nodes that are affected by the current transmission. The modified ns-2 simulator can have simulated populations of up to 3000 nodes and work up to 30 times faster than the original one [136].

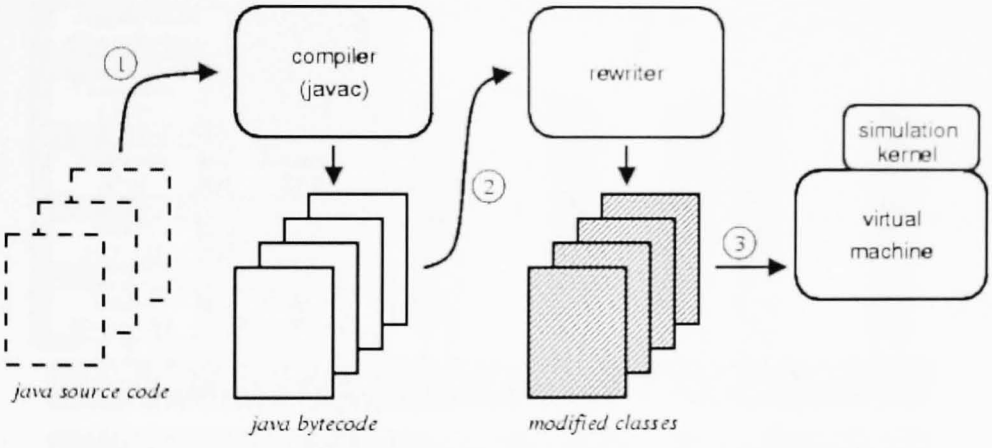


Figure 5.11: JiST system architecture [137]

5.7.3.3 Java in Simulation Time

A recent development in simulation studies is JiST (Java in Simulation Time), which transcends existing highly optimized simulation runtimes both in space and time [137]. JiST, a Java-based simulation platform, takes the advantages of the traditional systems-based and language-based simulator designs executing discrete event simulations efficiently by implementing simulation semantics directly at the bytecode

level. The JiST system architecture is shown in figure 5.11. It consists of four components: a compiler, a virtual machine, a rewriter, and a simulation time kernel. The compiler and virtual machine can be any standard Java products. The remaining components implement simulation time execution semantics. The rewriter, a dynamic class loader, intercepts all class load requests, then verifies and modifies the requested classes subsequently. This step transforms the JiST instructions embedded within the compiled simulation program into code with appropriate simulation time semantics, but otherwise completely keeps the existing program logic. In addition, it performs various static analyses that help drive runtime optimizations. At runtime, the modified classes interact with simulation time kernel through various injected operations [137].

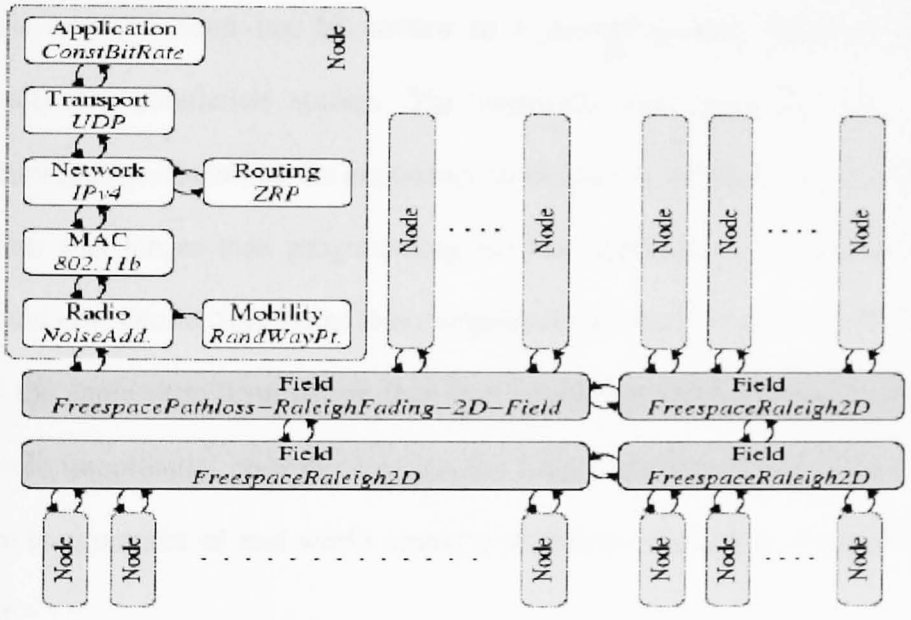


Figure 5.12: The SWANS simulator consists of event-driven components that can be configured and composed to form the desired wireless network simulation. Different classes of components are shown in a typical arrangement together with specific instances of component implementations in italics [138].

5.7.3.4 Scalable Wireless Ad hoc Network Simulator

SWANS (Scalable Wireless Ad hoc Network Simulator) [138] is built on the top of JiST platform. It is organized as independent software components that can be

composed to form complete wireless network or sensor network simulations, as shown in figure 5.12 [138]. Its capabilities are similar to ns-2 and GloMoSim but it is surprisingly efficient and scalable: it supports up to one million nodes and outperforms existing highly optimized simulation runtimes both in time and memory consumption [138]. The detailed performance comparison between SWANS, ns-2, and GloMoSim can be found here [138].

5.7.4 Transparency for ad hoc network simulations

Transparency implies that simulation programs can be transformed to run efficiently without the insertion of simulation-specific library calls or other manual program modifications [138]. JiST differs from previous simulation platforms in that the code that runs on JiST need not be written in a domain-specific language invented specifically for simulation studies. The improved code reuse reduces repeated programming time. Recalling the experience in this study, the time spent on coding in simulation was longer than programming the real application. Developers have to understand how GloMoSim works to accomplish the simulation task from the network layer to the application layer before they start simulation programming. On the other hand, more importantly, code reuse makes the simulation closely match the expected software environment of real world scenario so that the simulation is possibly more accurate.

5.7.5 Mobility model for MANET

In order to thoroughly simulate a proposed protocol for MANET, it is imperative to use a mobility model that accurately represents the mobility of nodes that will eventually utilize the given protocol. T. Camp et al survey the mobile models that are used in the simulations of MANET and presented heuristic conclusions [139]. These

models represent mobile nodes whose movement is either independent or dependent of each other. The simulation results are presented in order to illustrate the importance of choosing a mobile model with proper values of the related parameters in the simulation of a protocol for MANET: the performance drastically changes as a result of changing the simulated mobile model or setting different values of parameters with the same model. Clearly, the authors advocate that the performance of a protocol for MANET should be evaluated with the mobility model that most closely matches the expected real world scenario [139]. As the "random-waypoint (WP)" model is widely adopted in simulation studies, it is discussed in more depth.

The mobile nodes are initially distributed randomly around the simulation area in most of the performance simulations that adopt the WP model. This initial distribution is irrelative to the way that nodes distribute themselves when moving. The average neighbour numbers (ANN) of a mobile node is a key factor affecting the performance of proposed applications. The WP model shows high variability of ANN during a very long initial period of simulation [139]. This high variability will produce corresponding high variability in performance results unless the simulation results are calculated from long simulation runs [140]. In the WP model, the relationship between node speed and pause time is complicated. For example, a scenario with fast moving nodes and long pause times actually produces a more stable network than a scenario with slow moving speed and shorter pause times [139]. Therefore the appropriate parameters need to be evaluated. The WP model is also inclined to produce the phenomenon "density waves": the clustering of nodes in one particular part of simulation area [139]. In the WP model, "the probability of a node choosing a new destination that is located in the centre of the simulation area, or a destination which requires travel through the middle of the simulation area is high [139]".

The WP model is flexible, and possibly creates realistic mobility patterns for the way people might move in a conference or a museum. However, consider the "border problem" described earlier, it is rare for GloMoSim applying the WP model to simulate successfully. A predefined mobility file could be provided to the simulator so that the algorithm can be demonstrated. However, this would contradict with the unpredictable nature of MANET. Regarding the three proposed applications, the mobile model could be: "if something like this happens users should move quickly (slowly) to new locations or keep still to finish tasks". This kind of mobility implies strong human intention and it is hardly supported by any mobile models.

5.7.6 Simulating applications

Simulators focus on detailed simulation of the OSI network reference model from layer 1 to 4, aiming to simulate the lower network layers as exactly as possible. As described in section 5.2.2, if two adjacent nodes rebroadcast packets immediately after they received these packets from the source, the collision must happen. These simulators are useful tools for investigating the design issues of protocols below the network layer. However, they do not supply enough functions (mobile models and programming APIs) to simulate applications.

Evaluating applications for multi-hop ad hoc networks requires more facilities. In addition, it is difficult to organize such evaluations for the sake of reproducible scientific results. "A promising approach is to provide a uniform platform following a development principle composed of simulation, emulation and deployment on real mobile devices [141]". H. Frey et al present a Java-based implementation of such a uniform workbench [141].

The simulation environment mentioned above is a three layered architecture. The layer at the bottom abstracts the layer 1 to 4 of the OSI network reference model,

which includes operating system of mobile device, positioning method (the real application needs help of GPS support), communication protocols, and wireless interface. In addition it deals with other important issues associated with simulation: mobile model, connectivity computation and other dynamic aspects. The middle layer is the simulated application. On the top is the simulated user behaviour that controls the application. (The simulated application can also trigger actions of the simulated user behaviour.) The significant characteristic of this design is the separation of the simulation scenario from the simulated application and the user behaviour. Hence, the use of pre-computed connectivity and mobility data can speed up the simulation. The high abstraction level is based on an object-oriented design; it not only enables developers to concentrate on application design without worrying about technical details of the simulator but also makes applications developed and tested in the simulation environment easily transferred to a real hardware platform. Besides, the system provides the "hybrid simulation" mode, which allows real mobile devices to be connected to a running simulation by using Java remote method invocation (RMI) over a network interface.

The proposed platform is scalable and comfortable for evaluating applications. However, in order to reduce the computational complexity and allow the simulation of a large number of mobile nodes while maintaining short simulation times, it focuses on the simulation of the topological properties of the ad hoc network instead of simulating the physical layer and MAC layer as exactly as possible. Neither does it provide broad mobile models.

5.8 Summary

This chapter described field testing and simulation based testing to evaluate the performance of the designed system for multi-user interactive applications in

Convenience Networks. The testing environments, methods, and settings were described in detail for each part. It started with the real field testing that provided a general understanding of the work. The results reflected interesting directions that were elaborately tested in simulation studies. Overall conclusion is that the design is satisfactory to support applications working reasonably well in real environment. There is scope for improvements that is investigated in chapter 6.

Chapter 6 Adaptable mobile middleware

Applications in Convenience Networks (CNs) are subject to a variety of changes that frequently happen around server and the clients. The simulation studies reported in chapter 5 reveal that these changes have significant impact on the application's overall performance. In order to respond to these changes, the middleware supporting these applications should be made adaptive. In this chapter, a prototype of such middleware, the revision of data process module, is presented to provide extra support for adaptability, connectivity, and reliability requirements, which were not fully supported by the design discussed in chapter 4. The revised data process module is able to adapt to changes in its working environment by automatically choosing the proper broadcast method and setting the values of related parameters for clients on behalf of the application. In addition, the module can retrieve lost packets and help application server maintain connectivity information to clients. Simulation studies demonstrate the effectiveness and efficiency of these new functions provided by the enhanced data process module.

6.1 Need for adaptation

The main goal of this research is to study middleware support required for various types of multi-user interactive applications in CNs. The approach taken in this thesis is to develop and evaluate a variety of such interactive applications exploiting CNs in order to understand what specific functionalities can suitably be incorporated in the middleware, bearing in mind that middleware must handle heterogeneity and mobility issues. Currently, mobile application services are implemented across many styles of

middleware. Thus "middleware heterogeneity" [142] is displayed in mobile computing domain. Since flooding using broadcast is widely used in most phases of communication in MANET and the work reported here, this work has chosen to investigate design of adaptation components that are capable of optimizing diffusion implemented within the data process module.

In general, these adaptation components should incorporate dynamic awareness of their operating environment. This is similar to "context-awareness" [143] which is frequently discussed in the literature. There the researchers pay much attention to location or relative location context, like in office, home; or proximity to resource, such as printers; or user's activity context, like sitting in a lecture theatre or visiting a museum. However in CN, once an operating environment has been established, such as in an auction, most of the time there are no further changes in location and users' behaviour patterns do not change a lot; changes that do occur frequently are in users' physical surroundings. These changes include users joining or leaving applications as they wish, variations in the number of neighbours, or the distance to a specific user. The test results show that these changes, by themselves or in combination can have significant impact on the overall performance of the application. With respect to optimizing diffusion, adaptation components on each client therefore should adapt to these changes by automatically adjusting the working mechanism within data process module on behalf of the application. This would require choosing a proper broadcast method and values for related parameters for diffusing packets at right time. In addition, adaptation components should provide other capabilities. For instance, they can help application server check if each client has responded to its latest packet, or help hosts retrieve lost packets. Hence, all application requirements stated in section 4.1.1 can be implemented in the revised data process module.

If adaptation components, under some circumstances, are unable to improve the situation, they should alert the user hinting at the problem. For example, display a message on the screen: you may be out of network reach. Hopefully, adaptation components would wait for the user's response which will help them fulfil their task eventually. Before discussing the detailed design, the main results from the experiments reported in chapter 5 are summarised first.

6.2 Summary of main findings

Naturally, end-to-end latency is an important metric with regard to the performance of applications in MANET. In addition, packet loss and packet saved ratio can be chosen to evaluate the effectiveness and efficiency of communication protocols that use broadcast. The dedicated field tests and simulation studies have shown these three performance metrics are affected by the way that a host communicates with others under varying network environments.

i. Simple flooding is the easiest way to disseminate packets to the whole network, though it generates a great number of redundant packets, therefore, can cause the "broadcast storm" problem. Under the IEEE 802.11 MAC protocol, sending packets by broadcast is faster than by unicast under the same network conditions. However the latter is more reliable. When server disseminates packets to the whole network, three factors, packet size, frequency of dissemination, and H_{max} (radius of the network), affect the overall performance. The frequency of dissemination should not overrun the processing delays at clients. A large packet size has the same effect as having high frequency of dissemination, both of which consume more computing resources.

ii. The counter-based (CB) scheme is a suitable choice that is easy to implement and could effectively overcome the "broadcast storm" problem. The decision whether to use the scheme while disseminating packets to the whole network mainly depends on the value of one key parameter: the number of neighbours the disseminating node has. If a node has two or more neighbours, it could consider using this scheme. Once a node decides to use the CB scheme, it should apply the proper value of T_{max} (maximum waiting interval in the CB scheme). The denser the surrounding, larger is the value of T_{max} required. Given a medium density (20 to 30 neighbours), 50 milliseconds (ms) is enough; setting a longer delay will degrade the system performance instead (causing longer end-to-end latency). For a very high density (over 40 neighbours), the upper bound value of 100 ms should be applied. Each client should carefully choose simple flooding or the CB scheme to rebroadcast packets according to its surrounding situation. For instance, despite having more than two neighbours, node A should stick to using simple flooding, if A knows itself to be the only intermediate node for node B, which is one of its downstream neighbours. Otherwise, B could miss some packets from the server as A would stop rebroadcasting (see section 5.5.2).

iii. Under the current scheme for determining source route to the server (for a given broadcast from the server, each client uses the source route in the first arrived packet as its source route to the server), some clients may learn the source route of a length that is longer than their actual distance to the server. This phenomenon is explained in section 5.5.4. Simulation studies show that if a client rebroadcasts a packet that has taken a one hop or two hops source route, this length is almost certainly equal to the client's real distance to the

server. However, if the packet has taken a three hops or even longer source route, a redundant packet using a shorter route may well arrive moments later. Furthermore, with respect to the metric of packet loss, clients closer to the server gain an advantage over those that are farther away from the server. For example, given the value of H_{max} is four, clients that are four hops away from the server experience greater packet loss. This is not because collisions happen around these 4-hop clients during transmissions; instead, some of their upstream nodes which are in reality three hops or even two hops away from the server, stopped rebroadcasting as the length of the source route in their accepted packets had reached the limit, H_{max}.

6.3 Design of revised data process module

In this section the design details of the revised data process module are presented. The approach can be summarized as follow. Clients regularly broadcast heart-beat (HB) packets to enable the server and clients to construct information about their neighbours and the network. Using this information, improvements regarding the following design requirements are achieved:

i. Reliability. Clients and the server can retrieve lost packets.

ii. Connectivity. Server can gather connectivity information to each client.

iii. Adaptive flooding. Clients can use better ways of deciding when to use simple flooding or the CB scheme. This enables clients to use an improved communication protocol to diffuse application packets. Server can also adjust the values of parameters used to diffuse application packets.

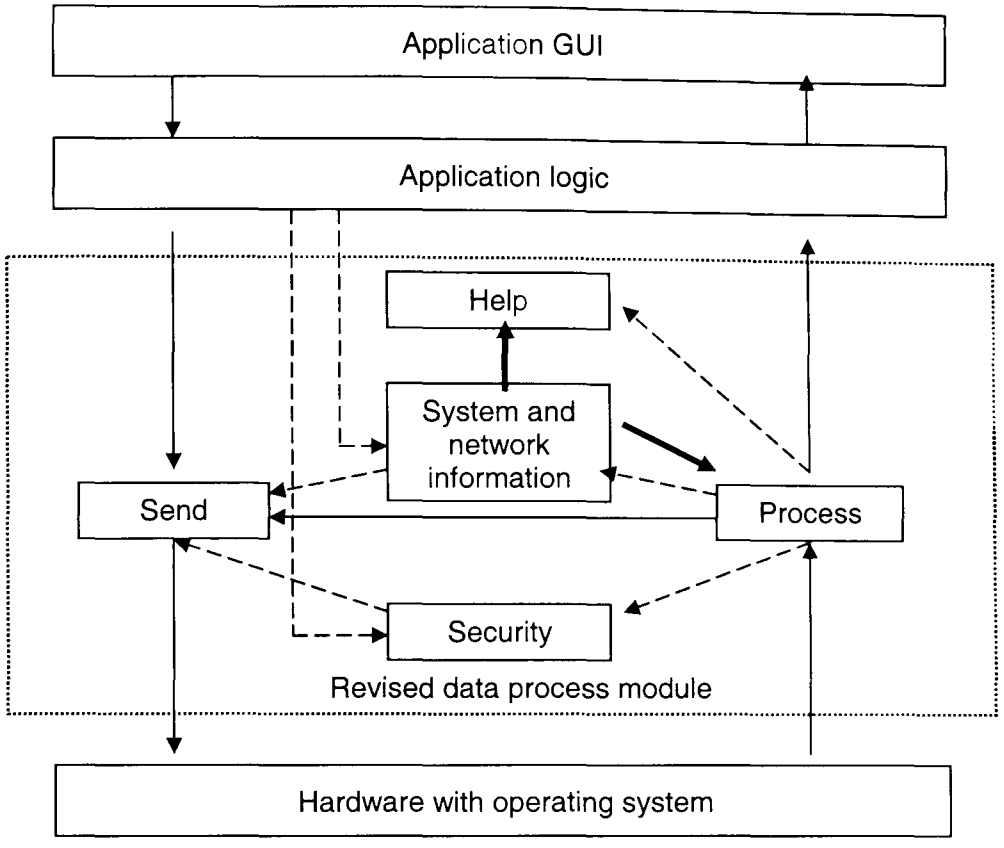


Figure 6.1: Architecture of revised data process module

6.3.1 Revised data process module overview

The architecture of the revised data process module is shown in figure 6.1. Comparing with the original design (see figure 4.1), the "system and network information" (SNIC) component is an extension of the "system information" (SI) component while the "help" component (HC) is an extension of the "ring tone" component. The "process" component is modified to implement the revised communication protocol and process new packet types. Other two components' functions remain unchanged.

SNIC is the core component in the revised design. Comparing with the original SI, SNIC stores a richer set of information, which is used for the purposes of making adaptation decisions and retrieving lost packets. With regard to HC, in addition to adjusting the volume or changing the type of ring tone, it can display alarm messages

on screens hinting at problems to users. SNIC and HC are the adaptation components in the revised design. As in figure 4.1, a solid thin line depicts message flow while a dashed line shows cooperation between two components. A thick solid line shows new cooperation for adaptation. (In order to make the figure clear and simple, the adaptation operation from HC to application GUI is not marked.)

6.3.2 Revised communication pattern

In the original design, only application packets are transported in the network: periodic diffusion from the server to clients followed by unicast responses from clients. These packets by themselves are unable to collect enough neighbourhood information and the state of the whole network. Hence there is insufficient basis for improving the performance of applications.

Under above communication pattern, most of the time clients are idle. In other words, the network is idle most of the time. Clients can make use of the idle period by informing neighbours of network related information through broadcasting HB packets. Such information could include a request for retrieving lost packets, a suggestion for choosing a broadcast method, the sender's system information, and so forth. Based on such information, clients and the server can be made adaptive to their environment.

In the revised system, clients initiate regular broadcasting of HB packets after receiving their very first application packet from the server. Clients stop broadcasting HB packets when the application ends or they leave. The manner that clients use to broadcast HB packets is opposite to the way the server disseminates information. This way a client can collect the information about its neighbours and downstream nodes. In particular, the server can gather the information about the whole network, as every

client is one of its downstream nodes.

In a network, clients can be divided into a few groups; such that in each group, the clients have the same distance to the server. Clients should broadcast HB packets group by group. The group whose members are farthest away from the server should broadcast HB packets first; the group whose members are immediate neighbours of the server should broadcast HB packets at last.

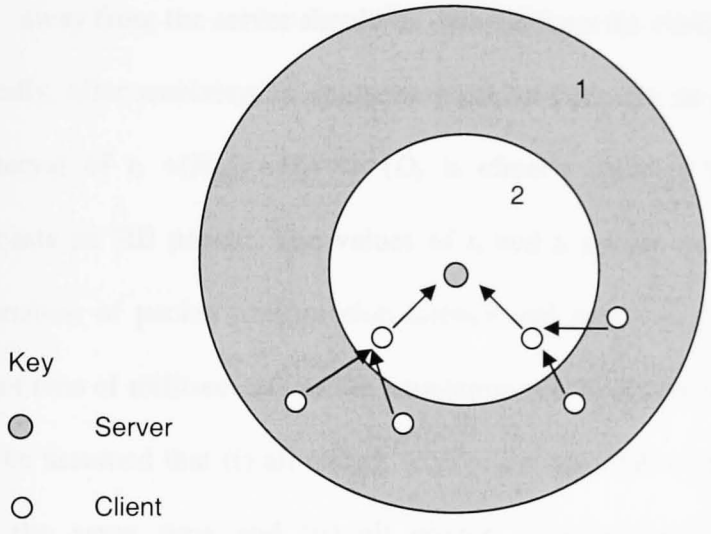


Figure 6.2: Clients broadcast HB packets from outside to inside of the network.

For example, figure 6.2 shows the conceptual view of a 2-hop network. The server is in the centre. The inner ring shows the server's 1-hop range; it stands for the group of the server's 1-hop clients. The outside shaded ring covers the server's 2-hop range. The number in each ring is the order that members of the group should broadcast HB packets: from outside to inside of the network. A solid line shows client's broadcast of an HB packet; the arrow in the line shows the packet can be received by one of the client's upstream neighbours. In a group, each client only broadcasts an HB packet once after the server disseminates an application packet; the client does not rebroadcast received HB packets.

Every application packet from the server contains the following timing information that is used by clients to broadcast their HB packet:

i. The dissemination period T.

ii. After receiving the packet, the interval t_s ($t_s < T$) that clients that are H_{max} (radius of the network) hops away from the server should wait before broadcasting their HB packets.

iii. The interval t_i by which the HB broadcast by a client that is k ($k < H_{max}$) hops away from the server should be delayed from the client that is $k+1$ hops away.

Generally, after receiving an application packet from the server each client waits for an interval of $t_s + (H_{max} - D_s) * t_i$ (D_s is client's distance to the server) and then broadcasts its HB packet. The values of t_s and t_i are set very much longer than the combination of packet transmission latency and processing latency which is of the order of tens of milliseconds as the simulation results shown in table 5.20 (mode 3). It could be assumed that (i) all clients receive the application packet from the server at about the same time and (ii) all clients have responded to the server before broadcasting an HB packet. For example, in an application the server sets T, t_s and t_i to 30 seconds (s), 20 s and 1 s respectively in an application packet. Assuming that H_{max} is four hops, in the network, 4-hop clients, 3-hops clients, 2-hop clients and 1-hop clients will broadcast their HB packets after 20 s, 21 s, 22 s, and 23 s respectively after they receive the application packet from the server.

After broadcasting an HB packet, a client sets a temporary interval to schedule broadcasting of the next HB packet, as it may not receive the next application packet from the server. Assuming that it will move to a position that is H_{max} hops away from the server later, the client sets the temporary interval to $T - (H_{max} - D_s) * t_i$. The temporary interval will be cancelled if clients receive the next application packet from

the server. Then the real waiting interval will be set as described above.

Figure 6.3 shows the new communication sequence in a period T:

i. The server disseminates an application packet and then clients help diffuse the packet to the whole network.

ii. Clients unicast a response packet to the server.

iii. Clients broadcast an HB packet to their neighbours.

iv. Optionally, the server broadcasts an "inquiry" packet in order to retrieve lost response packets from clients. The details of how the server retrieves these packets are described in section 6.3.5.2.

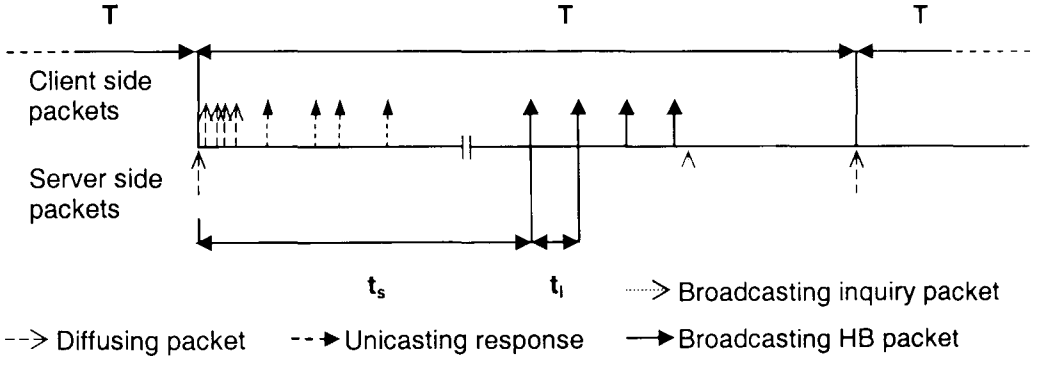


Figure 6.3: Revised communication pattern

6.3.3 Maintaining information in SNIC

As described earlier, clients broadcast HB packets group by group based on their distance to the server. There is an interval (t_i) between two adjacent groups to broadcast HB packets. This interval should be sufficiently large to enable a client to receive all possible HB packets from its downstream neighbours. The client gathers network information about their downstream nodes and then pads the aggregated information into its HB packets. This procedure is illustrated in figure 6.4.

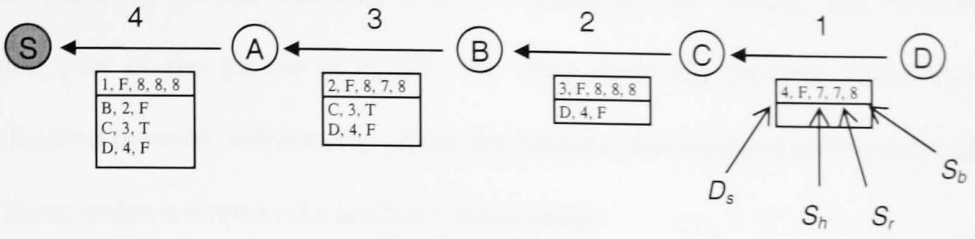


Figure 6.4: Clients pad aggregated information about their downstream nodes into their HB packets.

In figure 6.4, a solid line shows client's broadcast of an HB packet; the number over the line is the order that the client broadcasts its HB packet. Packets in figure 6.4 show a simplified structure of HB packet. Row 1, from left, records system information about the sender of the HB packet: distance to the server (D_s), flag for help (T stands for TRUE meaning the node needs their neighbours' help to retrieve lost packets from the server; F stands for FALSE; the details of how a client can retrieve lost packets is described in section 6.3.4.1.), the highest sequence number of the received packet from the server (S_h), the sequence number of the last received packet from the server to which this sender responded (S_r), and the sequence number of this HB packet (S_b).

The initial value of S_b is set to the value of S_h in the client's first received packet from the server. For each new application packet from the server or new HB packet from a client, the sequence number of the packet is always increased by one. At a client if $S_h = S_b$, then it indicates that the client did not miss any packets from the server; on the other hand, $S_h < S_b$ would indicate that the client did miss some.

Row 2 records network information that this sender collects into a list of entries: each entry comprises a downstream node's address, its distance to the server (D_s) and response status (T stands for TRUE meaning the node responded to the latest packet from the server; F stands for FALSE.). Except for client D's HB packet (as D had

little knowledge about the network when it broadcast the packet, the network information part of the packet is blank), the other three HB packets record their senders' learned network information about the senders' downstream nodes, although some of these nodes are not in the senders' direct range.

Specifically, the receiver (the server or a client) of an HB packet can deduce and then store the following information in its SNIC:

i. Connectivity. The nodes that the receiver can reach and the distance between any of these nodes and the receiver itself. For example, referring to figure 6.4, A knows it could reach B, C, and D. The distance from A to B, A to C, and A to D is one hop, two hops, and three hops respectively.

ii. Response status. Whether the sender of the HB packet responded to the latest packet from the server. For example, referring to figure 6.4, C can deduce that D did not respond as D lost this packet indicated by $S_h < S_b$ in D's HB packet; A can deduce that B did not respond although B received this packet, as in B's HB packet $S_b = S_h$ but $S_h > S_r$.

As the final recipient of the aggregated information, the server learns a panorama of its served clients mastering the connectivity detail to each client, the distance between each client and itself, and each client's response status to its latest packet. The usage of this information is described in sections 6.3.4.2.

6.3.4 Increasing reliability: retrieving lost packets

In its SNIC, each client also stores a limited number of packets, which the client has received from and responded to the server. This way a node can help its neighbours retrieve lost packets from the server; also if the server fails to receive important response packets like bids from specific bidders, it can request them to send these packets again.

6.3.4.1 How clients retrieve lost packets

In a Bingo game, players may lose the chance to win the game if they miss Bingo ball announcement. In order to avoid this happening, a given application can be configured to run with flag for help in HB packet set to TRUE (refer to row 1 in figure 6.4). A client (say B) checks the values of S_b and S_h in the received HB packet (from say C). If $S_h < S_b$, B knows C has lost packets from the server. If B has these packets, B unicasts a "salvage" packet containing the real payload in these packets back to C. Then C may respond to the server. C would unicast its response packet to the neighbour that is closest to the server. This neighbour will forward C's response packet to the server taking its own source route.

Figure 6.5 illustrates the procedure described above. Both A and B respond by unicasting a salvage packet to C, as shown by the dashed lines in figure 6.5. Packets in figure 6.5 show a simplified structure of salvage packet: row 1, from left, records distance to the server (D_s) about the sender of this packet and the sequence number of the packet from the server that this sender can provide for; row 2 records the real payload in the lost packet. Retrieving the lost packet C responds to server S. C unicasts its response packet to A, as A is closer to S than B. Then A forwards C's response packet to S. The solid lines in figure 6.5 show the procedure that C's response packet is delivered to S.

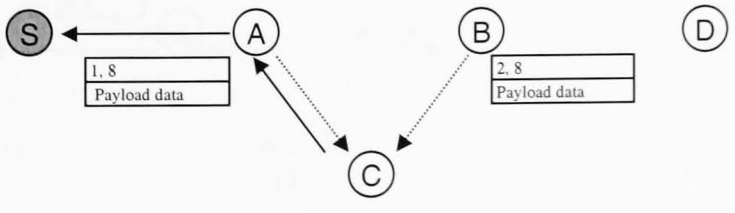


Figure 6.5: Clients A and B help C retrieve a lost packet from server S. Then C responds to S.

6.3.4.2 How a server retrieves lost response packets

Knowing each client's response status, the server can detect that some clients' response packets to its latest diffused application packet are really lost, if it fails to receive these packets nevertheless these clients' response status shows that they have responded. For instance, in figure 6.4, the server should receive the response packet from C.

In order to retrieve lost response packets from clients, the server broadcasts an "inquiry" packet, in which the server specifies the addresses of the clients whose response packets are lost with the sequence numbers of these lost packets. Receiving such an inquiry packet for the first time, a client checks its information in SNIC. If the following two conditions are met, the client rebroadcasts the inquiry packet; otherwise the inquiry packet is discarded. (Naturally clients discard redundant inquiry packets.) The conditions are (i) the client can reach some of the addresses specified in the inquiry packet and (ii) the distance between the client and one of these reachable addresses plus the source route length that the inquiry packet took is less than H_{max} (radius of the network). Finding its address appears in the inquiry packet, the client knows its response packets to the server with the sequence numbers have been lost. Then the client sends these packets back to the server again taking the source route the inquiry packet used, as the client's original source route may have been out of date.

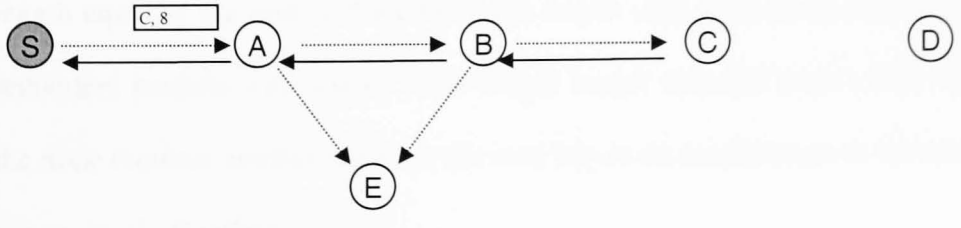


Figure 6.6: Server S disseminates an inquiry packet. Then client C sends the response packet to S again.

For example, as the dashed lines shown in figure 6.6, an inquiry packet is diffused in order that server S needs to retrieve a lost response packet from client C. The packet in figure 6.6 shows a simplified structure of inquiry packet: it records the client address with the sequence number of the response packet from this client that the server fails to receive. Both client A and B rebroadcast the inquiry packet, as they can reach C and the source route length that the inquiry packet will take to arrive at C is less than H_{max}, given that H_{max} is set to four hops. Client E discards the inquiry packet as it has no knowledge about C. Receiving the inquiry packet, C unicasts its response packet with the sequence number specified in the inquiry packet to S taking the reverse route of the inquiry packet used, as the solid lines shown in figure 6.6.

6.3.5 Adaptive flooding

As the main finding (ii) shows that a client should use the CB scheme to rebroadcast a packet from the server if it has two or more neighbours. However care is needed: the client should still use simple flooding if it is the only intermediate node for one of its downstream neighbours. Under such a circumstance, the client should be informed by this neighbour to using simple flooding. Specifically, in its HB packet, a node explicitly requests a particular upstream-neighbour to use simple flooding if one of the following two conditions is met:

i. The node does not receive redundant packets from the server with source route length equal to the node's distance to the server (D_s). (The node may receive redundant packets with source route length longer than the node's D_s.) Then the node requires another node, on the next hop in its source route to the server, to use simple flooding.

ii. If a node is required to use simple flooding, then this node requires the next node, the next hop in its source route to the server, to use simple flooding as well.

Condition (ii) reflects a transitivity relationship: on a multi-hop source route, each intermediate node should use simple flooding wherever the far end node requires its neighbour (on this source route) to use simple flooding.

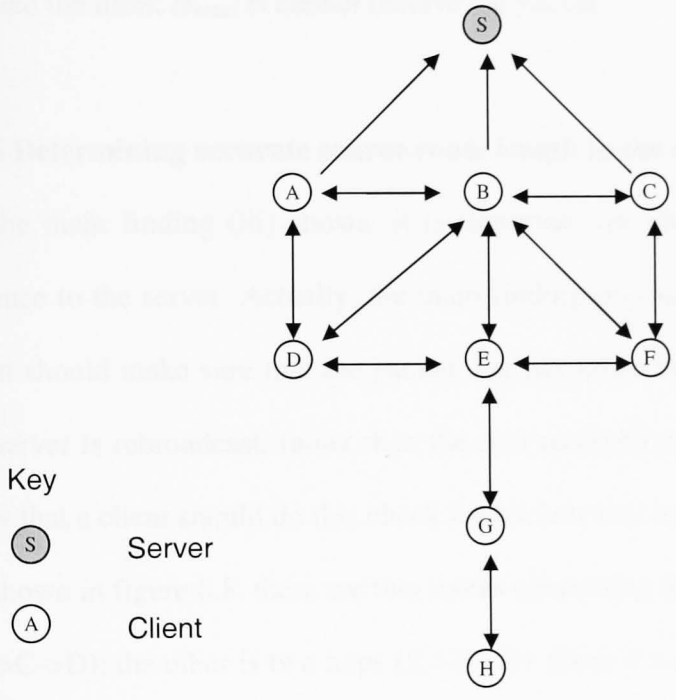


Figure 6.7: A sample network topology

For example, consider the network topology shown in figure 6.7, where H_{max} has been set to four hops. The arrowed lines show nodes that are one hop away from each other, by which they receive their neighbours' HB packets. According to condition (i) described above, node H will require G to use simple flooding, since H can only receive the packet from S via one route (S->B->E->G->H). According to condition (ii), node G and E will require their next hop, E and B, to use simple flooding. Using

the neighbourhood information in their SNIC, node A, D, C, and F will use the CB scheme (A and C have two neighbours while D and F have three). Node B, E, and G will stick to simple flooding although B has five neighbours, E has four, and G has two respectively. If any of node B, E or G were to use the CB scheme, H may not receive the packet from S. For instance, if B uses the CB scheme and stops rebroadcasting, the packet has to arrive at E via a 3-hop source route like (S->A->D->E). Then the packet will not be rebroadcast at G as the source route length has reached the limit, H_{max}. H cannot receive the packet.

6.3.6 Determining accurate source route length to the server

As the main finding (iii) shows: it is important for clients to know their accurate distance to the server. Actually, the main finding (iii) suggests a possible solution: a client should make sure that the packet that has taken the shortest source route from the server is rebroadcast, rather than the first received packet. The simulation results show that a client should do this check for packets that have taken three or more hops. As shown in figure 6.8, there are two routes connecting S and D. One is three hops (S->A->C->D); the other is two hops (S->B->D). Even if a packet taking the 3-hop route arrives first, D should wait a short while before rebroadcasting in case a packet taking the 2-hop route arrives later.

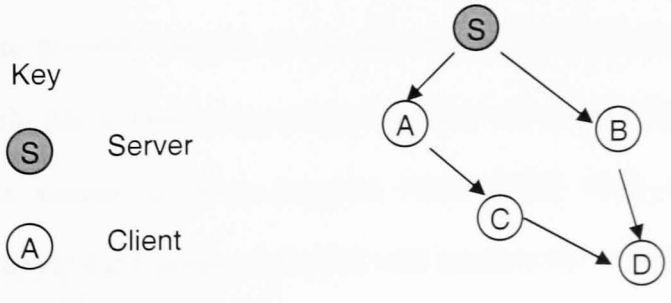


Figure 6.8: Client D chooses a shorter route.

6.3.7 Revised communication protocol

Each client maintains a triple of variables for recording server address (S), the highest sequence number of the received packet from the server (S_h), and distance to the server (D_s). Clients discard any packets with sequence numbers less than S_h or stop rebroadcasting if $L_s = H_{max}$, where L_s is source route length of packet and H_{max} is radius of the network. Otherwise clients update S_h and set D_s to L_s. If the value of L_s is one hop or two hops, the packet is rebroadcast subject to decision on use of the CB scheme. However, if the value of L_s is over two hops, clients wait for a very short interval, up to a fixed value of T_{wait}, for a duplicate packet that might arrive via a shorter route. During the short interval, if a client really receives a packet that has taken a shorter source route, the client will set D_s to L_s and rebroadcast this packet subject to decision on use of the CB scheme. Otherwise, after T_{wait}, the first received packet is rebroadcast subject to decision on use of the CB scheme.

6.3.8 Help component

In this sub-section, some ideas on how help components (HC) can be designed to help users are presented.

For example, the middleware can detect that there are no neighbours around users for a while. If the information is exposed to the users in some form, they could take steps to remedy the situation. An alert window would pop up on the screens informing the users that they are possibly outside of the network range. Having been alerted, if rational users would like to continue participating, they will move and try to rejoin the network. Another similar situation happens when SNIC finds users have not responded to the server for a given round. HC will increase the volume or change the type of ring tone in order to remind the users the arrival of server's packet.

6.4 Evaluation

To assess the advantages bought out by the HB packet mechanism, simulation study is conducted focusing on the performance of the revised communication protocol. A straightforward way of reducing packet loss experienced by clients that are farthest away from the server is simply to use the original protocol with H_{max} set to larger than strictly necessary (this will cause more flooding). The revised communication protocol is compared with such a simple scheme. Two aspects are evaluated:

i. Effectiveness. Does the packet loss experienced by clients that are farthest away from the server drop without sacrificing the performance of unicast and broadcast?

ii. Efficiency. Is there a reduction in the total amount of rebroadcast packets (the network traffic load) while clients use the revised communication protocol?

6.4.1 Setting GloMoSim

In order to allow for fair comparisons with the existing simulation results, GloMoSim uses the same configuration (see section 5.2.3) and topology (see figure 5.8) as before. 50 nodes are uniformly distributed in a 4-hop network. The server is almost at the centre of the network; around it, there are seven 1-hop clients, fifteen 2-hop clients, sixteen 3-hop clients, and eleven 4-hop clients. An enhanced application layer implements the functions of regularly broadcasting HB packets and using the revised communication protocol to rebroadcast packets from the server. The server broadcasts 100 packets at the rate of one packet every 30 seconds. Clients' behaviour is mode 3 (see section 5.5.3: random responding time is between 8 to 14 seconds). If clients fail to receive packets from the server, they do not want to retrieve these lost packets. So does the server; it does not broadcast inquiry packets to retrieve lost response packets

from clients. Other parameters related to broadcasting HB packets and the CB scheme are listed in table 6.1. The default value of H_{max} (radius of the network) is four hops.

Parameter	Explanation and value
T	Server's dissemination period: 30 seconds.
t_s	After receiving a new packet from the server, the interval that clients that are H_{max} hops away from the server should wait before broadcasting their HB packets: 20 seconds.
t_l	The interval by which the HB broadcast by a client that is k (k <H_{max}) hops away from the server should be delayed from a client that is $k+1$ hops away: 1 second.
T_{wait}	The interval for a client waiting for a duplicate packet with a shorter source route if the source route length of the first received packet is over two hops: 23 ms.
C_{max}	Maximum number of redundant packet received in the CB scheme: 2.
T_{max}	Maximum waiting interval in the CB scheme: 25 ms if a client has less than ten neighbours; Otherwise is 50 ms.

Table 6.1: Parameters used in simulation study of revised communication protocol

The following testing scenarios are chosen:

i. Clients use simple flooding to rebroadcast packets from the server.

ii. Clients use the CB scheme to rebroadcast packets from the server.

iii. Clients use simple flooding to rebroadcast packets from the server while the value of H_{max} is set to five hops (though the actual value is four hops).

iv. Clients use the CB scheme to rebroadcast packets from the server while the value of H_{max} is set to five hops (though the actual value is four hops).

v. Clients use the revised communication protocol to rebroadcast packets from the server while they regularly broadcast HB packets.

vi. Clients use the revised communication protocol to rebroadcast packets from the server while they regularly broadcast HB packets. However, they do not apply condition (ii) (see section 6.3.5) to explicitly require one of their upstream neighbours to use simple flooding in their HB packets.

For each testing scenario the simulation was conducted three times with varied seeds.

6.4.2 Simulation study results

Tables 6.2 to 6.7 present the results under each testing scenario (packet loss figures are for broadcast only). Table 6.2 is the exact copy from table 5.19 (mode 3), which is under testing scenario (i) and chosen as the comparison benchmark. Generally, unicast latency is almost same under different testing scenarios whilst the differences with the broadcast latency are not significant.

Distance to server (hop)	1	2	3	4
Average packet loss (%)	0	0	1	13.1
Average broadcast latency (ms)	2.6	15.8	34.2	54.6
Average unicast latency (ms)	4.3	9.7	14.5	17.9

Table 6.2: Performance of unicast and broadcast when clients use simple flooding

Distance to server (hop)	1	2	3	4
Average packet loss (%)	0	0.1	2.2	20.5
Average saved ratio (%)	29.5	22.8	40.4	N/A
Average broadcast latency (ms)	2.6	17.4	32.9	49.9
Average unicast latency (ms)	3.8	9.0	13.2	16.6

Table 6.3: Performance of unicast and broadcast when clients use the CB scheme

Distance to server (hop)	1	2	3	4
Average packet loss (%)	0	0	0	1.5
Average broadcast latency (ms)	2.5	19.6	39	64
Average unicast latency (ms)	3.8	9	13.6	17.6

Table 6.4: Performance of unicast and broadcast when clients use simple flooding (H_{max} =5)

Distance to server (hop)	1	2	3	4
Average packet loss (%)	0	0.2	0.4	3.3
Average saved ratio (%)	28	19.1	23.6	34
Average broadcast latency (ms)	2.5	18.6	34.7	54.8
Average unicast latency (ms)	3.8	9.3	13.6	17.7

Table 6.5: Performance of unicast and broadcast when clients use the CB scheme (H_{max} =5)

Distance to server (hop)	1	2	3	4
Average packet loss (%)	0	0	0.5	4.5
Average saved ratio (%)	17.6	15.1	51.8	N/A
Average broadcast latency (ms)	2.6	17.9	34.5	65.7
Average unicast latency (ms)	3.8	8.6	12.7	16.5

Table 6.6: Performance of unicast and broadcast when clients use the revised communication protocol based on HB packet support

Distance to server (hop)	1	2	3	4
Average packet loss (%)	0	0.2	0.9	13.5
Average saved ratio (%)	27	19.1	54.4	N/A
Average broadcast latency (ms)	2.6	18	34.1	63.6
Average unicast latency (ms)	3.8	8.8	12.8	16.6

Table 6.7: Performance of unicast and broadcast when clients use the revised communication protocol based on incomplete HB packet support

Comparing table 6.3 with table 6.2, it again shows the poor performance of using the CB scheme without discretion; packet loss of 4-hop clients is higher although saved ratio on their upstream nodes is satisfactory.

One solution to alleviate the problem described above could be to increase the value of H_{max}. The corresponding results are shown in tables 6.4 and 6.5. Apparently, packet loss of 4-hop clients drops sharply. But the cost is expensive since these 4-hop clients unnecessarily rebroadcast packets so that the network traffic load is increased (see

table 6.8). In such an IEEE 802.11 ad hoc networks, these extra broadcasts will potentially have negative impact on any other nodes that are within two hops range of these 4-hop clients.

Table 6.6 shows the improved results when clients use the revised communication protocol with HB packet support. Comparing table 6.2 with table 6.6, the packet loss of 4-hop clients drops as well as the saved ratio is reasonable on their upstream nodes. In addition, the cost is acceptable as clients broadcast HB packets when the network is idle; broadcasting of HB packets does not interfere with the clients' rebroadcasting of packets from the server. However, if clients do not fully apply the designated criteria to actively request one of their upstream neighbours to use simple flooding, for instance, neglecting condition (ii) specified in section 6.3.5.2, performance is poor as shown in table 6.7. It demonstrates the importance of the transitivity relationship, which reflects a comprehensive cooperation among nodes on a multi-hop route: every intermediate node along the route should use simple flooding once the node at the far end of the route explicitly does so.

Testing scenario	Total number of packets clients received / out of 4900	Total number of packets clients rebroadcast
i	4740	3456
ii	4637	2582
iii	4883	4600
iv	4856	3636
v	4840	2616

Table 6.8: Effectiveness and efficiency comparison of broadcast methods

Finally, from the perspective of effectiveness and efficiency, table 6.8 presents a clear comparison for different broadcast methods conducted in this simulation study. Total number of packets clients received (much is better) and total number of packets

clients rebroadcast (less is better) stand for the metrics of effectiveness and efficiency for the broadcast methods respectively. The results show the revised communication protocol performs well.

Though table 6.6 shows the improved performance, these 4-hop clients still lose a small number of packets from the server. In another independent test, under same conditions, all clients are set to request lost packets from the server in their HB packets. With their neighbours' help, packet loss of 3-hop and 4-hop clients drops to almost zero.

6.5 Summary

In this chapter, the design of a prototype of adaptable middleware, the enhanced data process module implementing all requirements stated in chapter 4, was presented. Generally, with extra HB packet support, the middleware can select the proper broadcast method with values of related parameters to rebroadcast packets for clients. In particular, the middleware can help server efficiently run an application by gathering connectivity and other kinds of useful information to all clients. Additionally, the middleware can help retrieve lost packets. The simulation studies show that improvements can be achieved: packet loss of clients that are farthest away from the server drops as well as the network traffic load is reduced.

Chapter 7 Conclusions

Mobile devices with wireless network interfaces have gained much popularity in people's daily life. They have created rich opportunities to develop various applications that exploit MANET. This research focuses on a class of interactive multi-user applications in a subset of MANET, termed "Convenience Network" (CN), as hardly any such type of applications are currently available. Therefore concrete requirements of middleware for mobile device using MANET are not well understood and it is not yet clear what system support the underlying middleware should provide. This is the problem addressed in the thesis.

7.1 Summary of the work

The thesis began by extensively investigating related technologies in ad hoc networking: from MAC layer to application layer regarding the OSI network reference model. Interactive multi-user applications in CNs and their requirements are the focus of this thesis. The aim is to study what suitable functionalities should be incorporated in middleware to support various such applications.

The thesis presented the initial design, implementation, and evaluation of a generalized data process module which is capable of supporting three representative applications in CNs, which are "Auction", "Bingo game", and "Chatting" respectively. For each of the three applications, this thesis described how an application specific ad hoc network can be formed and maintained in an economical manner. The application server periodically broadcasts application packets to the whole network; clients append their identities into the header of the fresh packets that they just received and

rebroadcast. Clients can construct the source route to the server by reversing the source route in the header of a received packet, and are able to communicate with the server by unicasting. If clients want to communicate with each other (as in Chatting), packets first are sent to the server who then forwards the packets to the final destinations. Under such circumstances optimizations could be applied. There is no extra overhead for the maintenance of CN. Server to client packet dissemination by applying simple flooding or the "counter-based" (CB) scheme have been implemented and tested on some PDAs. They worked well on the target platform. At the same time the field tests indicated the performance trend when network becomes dense; it also exposed the "border problem" which greatly degrades the system performance due to node mobility. Then testing was turned to simulation study as simulator can provide powerful facilities and rich scenarios helping designers overcome difficulties in field tests and explore the design space in detail. The main results from field tests and simulation studies were summarised in section 6.2 and are described again below.

Naturally, end-to-end latency is an important metric with regard to the performance of applications in MANET. In addition, packet loss and packet saved ratio can be chosen to evaluate the effectiveness and efficiency of communication protocols that use broadcast. These three performance metrics are affected by the way that a node communicates with others under varying network environments.

i. Simple flooding is the easiest way to disseminate packets to the whole network, though it generates a great number of redundant packets, therefore, can cause the "broadcast storm" problem. Under the IEEE 802.11 MAC protocol, sending packets by broadcast is faster than by unicast under the same network conditions. However the latter is more reliable. When server disseminates packets to the whole network, three factors, packet size,

frequency of dissemination, and H_{max} (radius of the network), affect the overall performance. The frequency of dissemination should not overrun the processing delays at clients. A large packet size has the same effect as having high frequency of dissemination, both of which consume more computing resources.

ii. The CB scheme is a suitable choice that is easy to implement and could effectively overcome the "broadcast storm" problem. The decision whether to use the scheme while disseminating packets to the whole network mainly depends on the value of one key parameter: the number of neighbours the disseminating node has. If a node has two or more neighbours, it could consider using this scheme. Once a node decides to use the CB scheme, it should apply the proper value of T_{max} (maximum waiting interval in the CB scheme). The denser the surrounding, larger is the value of T_{max} required. Given a medium density (20 to 30 neighbours), 50 milliseconds (ms) is enough; setting a longer delay will degrade the system performance instead (causing longer end-to-end latency). For a very high density (over 40 neighbours), the upper bound value of 100 ms should be applied. Each client should carefully choose simple flooding or the CB scheme to rebroadcast packets according to its surrounding situation. For instance, despite having more than two neighbours, node A should stick to using simple flooding, if A knows itself to be the only intermediate node for node B, which is one of its downstream neighbours. Otherwise, B could miss some packets from the server as A would stop rebroadcasting.

iii. The simple scheme for determining source route to the server (for a given broadcast from the server, each client uses the source route in the first received

packet as its source route to the server), some clients may learn the source route of a length that is longer than their actual distance to the server. Simulation studies show that if a client rebroadcasts a packet that has taken a one hop or two hops source route, this length is almost certainly equal to the client's real distance to the server. However, if the packet has taken a three hops or even longer source route, a redundant packet using a shorter route may well arrive moments later. Furthermore, with respect to the metric of packet loss, clients closer to the server gain an advantage over those that are farther away from the server. For example, given the value of H_{max} is four hops, clients that are four hops away from the server experience greater packet loss. This is not because collisions happen around these 4-hop clients during transmissions; instead, some of their upstream nodes which are in reality three hops or even two hops away from the server, stopped rebroadcasting as the length of the source route in their accepted packets had reached the limit, H_{max}.

In short, simulation studies indicated that the initial design has the capability of supporting interactive multi-user applications in CNs that meet most of design requirements under the assumed working conditions. But there is room from improvements through dynamic adaptation.

For adaptation to be effective, it is necessary that clients have information about their neighbourhood and a server has "the global view" of the network. In the revised design, this is achieved by clients regularly broadcasting heart-beat (HB) packets to enable the server and clients to construct information about their neighbours and the network. The HB packet scheme has been carefully engineered to enable information to be collected starting from the edges of the network to the centre. Using this

enhancement, the following design requirements that were not fully supported in the initial design have been achieved:

i. Reliability. Clients and the server can retrieve lost packets.

ii. Connectivity. Server can gather connectivity information to each client.

iii. Adaptive flooding. Clients can use better ways of deciding when to use simple flooding or the CB scheme. This enables clients to use an improved communication protocol to diffuse application packets. Server can also adjust the values of parameters used to diffuse application packets.

Hence, all application requirements stated in section 4.1.1 can be implemented in the revised data process module.

7.2 Further work

This research focused on a class of interactive applications exemplified by mobile device users interacting with each other in a limited public place. Typical interaction pattern in such applications consists of a server sending out information regularly to clients using broadcasts and individual clients replying using unicasts. This work highlighted the design of a specific middleware which is capable of supporting this specific type of interactive applications.

The work presented here can be extended in a number of directions. It would be interesting to extend the design to support applications where interactions are not regular or applications that are peer-to-peer, rather than client-server. The Chatting application did indicate that this design can support irregular peer-to-peer interactions via a server. It is believed that this design can be extended to support hybrid peer-to-peer systems that make use of a collection of servers. Another area of investigation would be to investigate support for multimedia.

It is noted that in an IEEE 802.11 ad hoc network, a node actively keeps exchanging MAC layer management beacons with its neighbours even if no application layer packets exchange is taking place. Thus at MAC layer, a node has knowledge of the exact number of its neighbours. If this valuable information could be sent through or shared with the upper layers, it will greatly help system design with respect to adaptation. Cross-layering design [144] follows this idea: to let protocols that belong to different layers cooperate in sharing network-status information. Application of cross-layering principles to the middleware design presented here would be another area of investigation.

References

[1] A. L. Murphy, G. C. Roman, and G. Varghese, "An Exercise in Formal Reasoning about Mobile Communications", International Workshop on Software Specifications and Design, Proceedings of the 9th International Workshop on Software Specification and Design, Iseshima, Japan, April 16-18, 1998, pp. 25-33. Can be found at: http://citeseer.ist.psu.edu/murphy98exercise.html.

[2] J. Jubin, and J. D. Tornow, "The DARPA Packet Radio Project", Proceedings of IEEE, 75 (1), 1987, pp21-32.

[3] Youngsik Lim, "The Limit of Ad Hoc Networks to Commercial Success", can be found at: http://www.cs.rutgers.edu/~rmartin/teaching/fall04/cs552/papers/017.pdf.

[4] Frazer Bennett, David Clarke, Joseph B. Evans, Andy Hopper, Alan Jones, and David Leask, "Piconet: Embedded Mobile Networking", IEEE Personal Communications, 4(5), 1997, pp 8-15.

[5] The Bluetooth specification. Can be found at: http://www.bluetooth.org.

[6] IEEE 802.15 Working Group for Wireless Personal Area Networks. Can be found at: http://ieee802.org/15/index.html.

[7] IEEE Standard 802.11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", August, 1999. Can be found at: http://standards.ieee.org/getieee802/download/802.11-1999.pdf.

[8] Janne Korhonen, "HIPERLAN/2", can be found at: http://www.tml.tkk.fi/Studies/Tik-110.300/1999/Essays/hiperlan2.html.

[9] Mobile Ad-hoc Networks (MANET) Charter. Can be found at: http://www.ietf.org/html.charters/manet-charter.html.

[10] J. Macker, and M. S. Corson, "Mobile Ad Hoc NetWorking and the IETF", ACM Mobile Computing and Communication Review, 2(1), 1998, pp9-14.

[11] Mangus Frodigh, Per Johansson, and Peter Larsson, "Wireless Ad Hoc Networking – The Art of Networking without a Network", Ericsson Review, No.4, 2000.

[12] Ural Mutlu, and Reuben Edwards, "CORBA in Mobile Communications", Proceedings of PostGraduate Networking Conference (PGNET 2003), Liverpool, UK, June 16-17, 2003. Can be found at:
http://www.cms.livjm.ac.uk/pgnet2003/submissions/Paper-48.pdf.

[13] M. Satyanarayanan, J. Kistler, P. Kumar, M. Okasaki, E. Siegel, and D. Steere, "Coda: A Highly Available File System for a Distributed Workstation Environment", IEEE Transactions on Computers, 39(4), 1990, pp 447–459.

[14] S. Long, R. Kooper, G. Abowd, and C. Atkenson, "Rapid Prototyping of Mobile Context-aware Applications: the Cyberguide Case Study", Proceedings of the Second Annual International Conference on Mobile Computing and Networking, White Plains, NY, USA, November, 1996, pp 97–107. ACM Press.

[15] David B. Johnson, David A. Maltz, Yih-Chun Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", Internet draft, can be found at: http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt, work in progress.

[16] Sung-Ju Lee, William Su, and Mario Gerla, "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks", Mobile Networks and Applications, 7(6), 2002, pp 441-453.

[17] C. Carter, S. Yi, P. Ratanchandani, and R. Kravets. "Manycast: Exploring the Space between Anycast and Multicast in Ad Hoc Networks". Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobioCom 2003), San Diego, CA, USA, September 14-19, 2003, pp 273–285. ACM Press.

[18] Chatschik Bisdikian, "An Overview of the Bluetooth Wireless Technology", IEEE Communication, 39(12), 2001, pp 86-94.

[19] R.Bruno, M.Conti, E.Gregori, "WLAN Technologies for Mobile Ad Hoc Networks", Proceedings of the 34th Hawaii International Conference on System Sciences, volume 9, Maui, Hawaii, January 3-6, 2001.

[20] F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II - the Hidden Terminal Problem in Carrier Sense Multiple-access Modes and the Busy-tone Solution," IEEE Transaction on Communications, 23(12), 1975, pp 1417–1433.

[21] Charles E. Perkins, Pravin Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers". Proceedings of the conference on Communication architectures, protocols and applications, London, UK, August 31-September 2, 1994, pp 234-244.

[22] D.B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts", Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, December, 1994, pp 158-163.

[23] D.B. Johnson, and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", Mobile Computing, T.Imielinski, and H.Korth, eds, Kluwer Publishers, Norwell, Mass., 1996, pp 153-181.

[24] Lee, S.-J., Gerla, M., Toh, C.-K, "A Simulation Study of Table-driven and On-demand Routing Protocols for Mobile Ad Hoc Networks", Network magazine IEEE, 13(4), July/August, 1999, pp 48-54.

[25] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing". Internet draft, can be found at: http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt, work in progress.

[26] Charles E. Perkins, Elizabeth M. Belding-Royer, "Ad-hoc On-Demand Distance Vector Routing", Proceedings of 2nd Annual IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, USA, February 25-26, 1999, pp 90-100.

[27] V.D. Park and M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," Proceedings of INFOCOM'97, Kobe, Japan, April 7-11, 1997, pp 1405-1413.

[28] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot. "Optimized Link State Routing Protocol", IETF Internet Draft, draft-ietf-manet-olsr-07.txt. can be found at: http://ietf.org/rfc/rfc3626.txt.

[29] Young-Bae Ko and Nitin H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks", Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Dallas, Texas, USA, October 25-30, 1998, pp 66-75.

[30] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, Barry A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)", Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Dallas, Texas, USA, October 25-30, 1998, pp76-84.

[31] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", Internet draft, draft-ietf-manet-zone-zrp-04.txt, July, 2002. Can be found at: http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-zrp-04.txt.

[32] Raghupathy Sivakumar, Prasun Sinha, and Vaduvur Bharghavan, "CEDAR: a Core-Extraction Distributed Ad hoc Routing Algorithm", IEEE Journal on Selected Areas in Communication, 17(8), 1999, pp1454-1465.

[33] Elizabeth M. Royer, and Chai-Keong Tou, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks", IEEE Personal Communications, 6(2), 1999, pp 46-55.

[34] Xiaoyan Hong, Kaixin Xu, and Mario Gerla, "Scalable Routing Protocols for Mobile Ad Hoc Networks", IEEE Network Magazine, 16(4), 2002, pp 11-21. Can be found at: http://www.cs.ucla.edu/NRL/wireless/uploads/ntwkmgz02-hxy.pdf.

[35] J.J. Garcia-Luna-Aceves, and J. Crowcroft, "The Core-Assisted Mesh Protocol", IEEE Journal on Selected Areas in Communication, 17(8), 1999, pp 1380-1394.

[36] C.W. Wu, Y.C. Tay, and C.-K. Toh, "Ad hoc Multicast Routing Protocol Utilizing Increasing Id-numberS (AMRIS) Functional Specification", Internet-Draft, draft-itef-manet-amris-spec-00.txt, November, 1998, http://www.ietf.org/proceedings/99jul/I-D/draft-ietf-manet-amris-spec-00.txt, suspend.

[37] Elizabeth M. Royer, Charles E. Perkins, "Multicast Operation of the Ad-hoc On-demand Distance Vector Routing Protocol", Proceedings of Mobicom'99, Seattle, Washington, USA, August 15-19, 1999, pp. 207-218.

[38] S. H. Bae, S.-J. Lee, and M.Gerla, "Unicast Performance Analysis of the ODRMP in a Mobile Ad hoc Network Testbed", Proceeding of IEEE ICCCN 2000, Las Vegas, NV, USA, October 16-18, 2000, pp 148-153.

[39] S.-J. Lee, W. Su, and M.Gerla, "Exploiting the Unicast Functionality of the On-Demand Multicast Routing Protocol," Proceedings of IEEE WCNC 2000, volume 3, Chicago, IL, USA, September 23-28, 2000, pp 1317-1322.

[40] Refik Molva, Pietro Michiardi, "Security in Ad hoc Networks", (invited paper), Personal Wireless Communications, September 23-25, 2003, Venice, Italy.

[41] Yin-Chun Hu, Adrian Perrig, "A Survey of Secure Wireless Ad Hoc Routing", IEEE Security and Privacy, 2(3), 2004, pp28-39.

[42] Y.-C. Hu, D.B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks," Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 02), Callicoon, NY, USA, June 20-21, 2002, pp 3–13.

[43] Y.-C. Hu, A. Perrig, and D.B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom 2002), Atlanta, Georgia, USA, September 23-28, 2002, pp 12–23.

[44] A. Perrig et al., "Efficient Authentication and Signing of Multicast Streams over Lossy Channels", Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 14-17, 2000, pp 56–73.

[45] Golden G. Richard III, "Service Advertisement and Discovery: Enabling Universal Device Cooperation", IEEE Internet Computing, September October 2000, pp 18-26.

[46] Salutation Architecture Specification; available online at: http://www.salutation.org/.

[47] Universal Plug and Play Specification; available online at: http://www.upnp.org/.

[48] Adrian Friday, Nigel Davies, and Elaine Catterall, "Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments", Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access, May 2001.

[49] Sumi Helal, "Standards for Service Discovery and Delivery", Pervasive Computing, 2002, pp 95-100.

[50] E.Guttman, "Service Location Protocol: Automatic Discovery of IP Network Services," IEEE Internet Computing, 3(4), 1999, pp 71-80.

[51] Sun Microsystem, Inc. Java Micro Edition. http://java.sun.com/j2me/index.jsp.

[52] Microsoft .NET Compact Framework.
http://msdn.microsoft.com/vstudio/device/compactfx.asp.

[53] M. Haahr, R. Cunningham, and V. Cahill, "Supporting CORBA Applications in a Mobile Environment (ALICE)", Proceedings of 5th International Conference on Mobile Computing and Networking, Seattle, Washington, USA, August 1999, pp39-47.

[54] Cecilia Mascolo, Licia Capra, and Wolfgang Emmerich, "Mobile Computing Middleware", Lecture Notes In Computer Science, Advance Lectures on Networking, Springer-Verlag New York Inc, 2002, pp 20-58. ISBN: 3540001654.

[55] A. D. Joseph, J. A. Tauber, and M. F. Kaashoek, "Mobile Computing with the Rover Toolkit," IEEE Transactions on Computers, 46(3), 1997, pp337-352.

[56] R. Monson-Haefel, D. A. Chappell, "Java Message Service", O'Reilly & Associates, December 2000, ISBN: 0-596-00068-5.

[57] IBM Redbooks, "MQSeries Version 5.1 Administration and Programming Examples", IBM, 1999, ISBN 0738415235.

[58] Eiko Yoneki. "Mobile Applications with a Middleware System in Publish-Subscribe Paradigm", Proceedings of 3rd Workshop on Applications and Services in Wireless Networks (ASWN'03), Bern, Switzerland, July 2-4, 2003.

[59] M.Musolesi, C. Mascolo, and S. Hailes. "Adapting Asynchronous Messaging Middleware to Ad Hoc Networking", ACM International Conference Proceeding Series; volume 77; Proceedings of 2nd International Workshop on Middleware for Pervasive and Ad-hoc Computing , Toronto, Canada, October 18, 2004, pp121-126.

[60] E. Vollset, D. Ingham, and P. Ezhilchelvan, "JMS on Mobile Ad-Hoc Networks", Proceedings of Personal Wireless Communications (PWC), Venice, Italy, September 23-25, 2003, pp 40-52.

[61] Arjuna website: http://www.arjuna.com.

[62] M.Musolesi, C. Mascolo, and S. Hailes. "Adapting Asynchronous Messaging Middleware to Ad Hoc Networking", ACM International Conference Proceeding Series; volume 77; Proceedings of 2nd International Workshop on Middleware for Pervasive and Ad-hoc Computing , Toronto, Canada, October 18, 2004, pp121-126.

[63] Micro Musolesi, "Designing a Context-aware Middleware for Asynchronous Communication in Mobile Ad Hoc Environments", ACM International Conference Proceeding Series volume 79; Proceedings of the 1st International Doctoral Symposium on Middleware, Toronto, Ontario, Canada, October, 19, 2004, pp 304-308.

[64] M.Musolesi, S. Hailes, and C.Mascolo, "Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks", Technical Report, UCL-CS Research Note, July 2004.

[65] A.Vahdat, and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," Technical Report CS-2000-06, Department of Computing Science, Duke University, 2000.

[66] D. Gelernter, "Generative Communication in Linda". ACM Transactions on Programming Languages and Systems, 7(1), 1985, pp 80–112.

[67] Abdulbaset Gaddah, and Thomas Kunz, "A Survey of Middleware Paradigms for Mobile Computing", Carleton University Systems and Computing Engineering Technical Report SCE-03-16, July, 2003. Available online at:
http://www.sce.carleton.ca/wmc/middleware/middleware.pdf.

[68] A. L. Murphy, G. P. Picco, and G. C. Roman, "Lime: A Middleware for Physical and Logical Mobility". Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21), Phoenix, Arizona, USA, April 16-19, 2001, pp 524-536.

[69] M. Satyanarayanan, "Mobile Information Access", IEEE Personal Communications, 3(1), 1996, pp26-33.

[70] A. Demers, K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and B. Welch, "The Bayou Architecture: Support for Data Sharing among Mobile Users", Proceedings of IEEE workshop on Mobile Computing Systems and Applications, Santa Cruz, CA., USA, December, 1994, pp2-7.

[71] C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich, "XMIDDLE: A Data-Sharing Middleware for Mobile Computing", Wireless Personal Communications. 21(1). 2002, pp 77-103.

[72] G. Blair, G. Coulson, A. Andersen, L. Blair, M. Clarke, F. Costa, H. Duran-Limon, T. Fitzpatrick, L. Johnston, R. Moreira, N. Parlavantzas, and K. Saikoski, "The Design and Implemantation of OpenORB 2", IEEE Distributed System Online, 2(6). September, 2001.

[73] Peter R. Pietzuch, "Hermes: A Scalable event-based middleware", Technical Report No. 590, Computer Laboratory, University of Cambridge. Can be found at: http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-590.pdf.

[74] P. Grace, G. Blair, and S. Samuel, "A Reflective Framework for Discovery and Interaction in Heterogeneous Mobile Environments", ACM SIGMOBILE Mobile Computing and Communications Review, 9(1), special section on Discovery and Interaction of Mobile Services, January, 2005, pp2-14.

[75] Jean Bacon, Ken Moody, John Bates, Richard Hayton, Chaoying Ma, Andrew McNeil, Oliver Seidel, and Mark Spiteri, "Generic Support for Distributed Applications", IEEE Computer, 33(3), 2000, pp 68-77.

[76] Antonio Carzaniga, and Alexander L. Wolf, "Content-Based Networking: A New Communication Infrastructure", In NSF Workshop on an Infrastructure for Mobile and Wireless Systems, Scottsdale, AZ, USA, October, 2001.

[77] L. Kleinrock, "Nomadic Computing –An Opportunity", ACM SIGCOMM Computer Communications Review, 25(1), 1995, pp 36-40.

[78] Charles E. Perkins. "Mobile IP, Ad-Hoc Networking, and Nomadicity", COMPSAC, Proceedings of the 20th Conference on Computer Software and Applications, Seoul, Korea, August 19-23, 1996, pp 472-476.

[79] Mark Weiser, "The Computer for the Twenty-First Century", Scientific American, 265(3), 1991, pp 94-105.

[80] Reto Hermann, Dirk Husemann, Michael Moser, Michael Nidd, Christian Rohner, Andreas Schade, "DEAPspace – Transient ad hoc networking of pervasive devices", Computer networks, 35(4), 2001, pp 411-428.

[81] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirjana Spasojevic, "People, places, things: Web presence for the real world", Technical Report HPL-2001-279, Hewlett Packard Laboratories, December, 2001.

[82] Web page at: http://www.swatch.com/synchro/index2.html.

[83] Arianna Bassoli, Cian Cullinan, Julian Moore, Stefan Agamanolis, "TunA: A mobile Music Experience to Foster Local Interactions (poster)", presented at 5th International Conference on Ubiquitous Computing, Seattle, Washington, October 12-15, 2003. Available online at:
http://www.medialabeurope.org/research/library/Bassoli_Tuna_2003.pdf.

[84] J.Sanneblad, L.E. Holmquist, "Using Ad Hoc Network Games to support Face-to-Face Interaction in Public Places", Demonstration, UIST 2002 Conference Companion, Paris, France, October 27-30, 2002. Available online at:
http://www.viktoria.se/fal/publications/2002/uist2002-demo-cafetrek.pdf.

[85] P. Poupyrev, M.Kosuga, and P. Davis, "Analysis of Wireless Message Broadcast in Large Ad Hoc Networks of PDAs", Proceedings of the Fourth IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002), Stockholm, Sweden, September 9-11, 2002, pp 299-303.

[86] H. Ritter, T. Voigt, M. Tian, J. Schiller, "Experiences Using a Dual Wireless Technology Infrastructure to Support Ad-hoc Multiplayer Games", Proceedings of the 2nd Workshop on Network and System Support for Games, Redwood City, CA., May 22-23, 2003, pp101-105.

[87] Derek Greene, Donal O'Mahony. "Instant Messaging & Presence Management in Mobile Ad-Hoc Networks", Proceedings of 2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2004), Orlando, FL, USA, March 14-17, 2004, pp 55-59.

[88] M. Day, J. Rosenberg, and H. Sugano, "A Model for Presence and Instant Messaging" (RFC 2778)," IETF IMPP Working Group, 2000.

[89] Jin-Hee Cho, and Ing-Ray Chen, "Design, Implementation, and Analysis of Wireless Ad Hoc Messenger: Using Microsoft .Net Compact Framework and Pocket PCs", available at: http://people.cs.vt.edu/~irchen/microsoft-grant/wireless-ad-hoc-messenger/.

[90] The network simulator- ns-2, available at: http://www.isi.edu/nsnam/ns/index.html.

[91] Zeng X., R.Bagrodia, M. Gerla, "GloMoSim: a Library for the Parallel Network Simulation Environment", Proceedings of the 12th Workshop on Parallel and Distributed Systems, Banff, Alberta, Canada, May 26-29, 1998, pp154-161.

[92] Valeri Naoumov, Thomas Gross, "Simulation of Large Ad Hoc Networks", Proceedings of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2003), San Diego, CA, USA, September 19, 2003, pp50-57.

[93] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, Jorjeta Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols".

Proceedings of the 4th Annual ACM/IEEE International conference on Mobile Computing and Networking (MobiCom'98), Dallas, Texas, USA, October 25-30, 1998, pp85-97.

[94] Samir R. Das, Charles E. Perkins, Elizabeth M. Royer, Mahesh K. Marina, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks", IEEE INFOCOM 2000, volume 1, Tel Aviv, Israel, March 26-30, 2000, pp3-12.

[95] Thomas Kunz, Ed Cheng, "On-demand Multicasting in Ad-Hoc Networks: Comparing AODV and ODRMP", Proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, July 2-5, 2002, pp453-454.

[96] Sung-Ju Lee, William Su, Julian Hsu, Mario Gerla, and Rajive Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols", INFOCOM 2000, volume 2, Tel Aviv, Israel, March 26-30, 2000, pp565-574.

[97] M. Banatre and F. Weis, "SIS: A New Paradigm for Mobile Communication Systems", Proceedings of IST'99, Helsinki, Finland, November 22-24, 1999.

[98] Laura Marie Feeney, Bengt Ahlgren, Assar Westerlund, "Spontaneous Networking: An Application-oriented Approach to Ad Hoc Networking", IEEE Communications (Special issue on Ad Hoc Networking), 39(6), June 2001, pp176-181.

[99] N. Lin, and S. K. Shrivastava, "System Support for Small-scale Auctions", In Proceedings of the 2nd Mediterranean Workshop on Ad-Hoc Networks MED-HOC NET 2003 (IFIP-TC6-WG6.8), Mahdia, Tunisia, June, 2003, pp 143-149.

[100] Imrich Chlamtac, Marco Conti, Jennifer J.-N. Liu, "Mobile Ad Hoc Networking: Imperatives and challenges", Ad Hoc Networks (1), 2003, pp 13-64.

[101] Chih-Yung Chang and Jang-Ping Sheu, "Design and Implementation of Ad Hoc Classroom and eSchoolbag Systems for Ubiquitous Learning", Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE'02), pp 8-14.

[102] Johan Sanneblad and Lars Erik Holmquist, "Prototyping Mobile Game Applications", Proceedings of First International Workshop on Entertainment Computing (IWEC2002), Makuhari, Japan, May 14-17, 2002, pp 117-124.

[103] Bo Xu, Aris Ouksel, and Ouri Wolfson, "Opportunistic Resource Exchange in Inter-Vehicle Ad-Hoc Networks", Proceedings of IEEE International Conference on Mobile Data Management (MDM'04), January 19-22, 2004, Berkeley, CA, USA, pp 4-12.

[104] Matthew Britton, Lionel Sacks, "The SECOAS Project—Development of a Self-Organising, Wireless Sensor Network for Environmental Monitoring", Proceedings of

Second International Workshop on Sensor and Actor Network Protocols and Application (SANPA 04), August 22, Boston, MA, USA.

[105] Wenjie Chen, Lifeng Chen, Zhang-long Chen, Shi-liang Tu, "A Realtime Dynamic Traffic Control System Based on Wireless Sensor Network", 34th International Conference on Parallel Processing Workshops (ICPP Workshops 2005), June 14-17, 2005, Oslo, Norway, pp 258-264.

[106] Martin Strohbach, Hans-Werner Gellersen, Gred Kortuem, and Christian Kray, "Cooperative Artefacts: Assessing Real World Sitautions with Embedded Technology", Proceedings of the 6th International Conference on Ubiquitous Computing (Ubicomp), September 11-14, 2004, Tokyo, Japan, pp 250-267.

[107] Tia Gao, Dan Greenspan, and Matt Welsh, "Improving Patient Monitoring and Tracking in Emergency Response", Proceedings of 3rd International Conference on Information Communication Technologies in Health, July 7-9, 2005, Samos, Greece.

[108] S. Preuss, H. Clemens, "Spontaneous Networking", Workshop on Future Services for Networked Device (FuSeNetD 1999), In conjunction with Eurescom 1999, November 8-9, 1999, Heidelberg, Germany.

[109] R. Liscano, A. ghavam, "Context Awareness and Service Discovery for Spontaneous Networking", Tutorial in AdhocNow'03

[110] Noah Fields, Jonathan Gips, Philip Liang, Arnaud Pilpre, "SNIF: Social Networking In Fur", Conference on Human Factors in Computing Systems (CHI 2005), April 2-7, 2005, Portland, Oregon, USA. http://sniflabs.com/snif.pdf.

[111] B. Nardi, and S. Whittakser, "The Role of Face-to-face Communication in Distributed Work", In P. Hinds and S. Kiesler (Eds), "Distribute Work", Cambridge, MA, MIT Press, 2002, pp 83-112.

[112] M. Kumar, S.J. Feldman, "Internet Auctions", Proceedings of 3rd USENIX Workshop on Electronic Commerce, Boston, MA, USA, August 31 - September 3, 1998, pp. 49-59.

[113] Ching-Shan Peng, Jose Miguel Pulido, Kwei-Jay Lin and Douglas M. Blough, "The Design of an Internet based Real-Time Auction System", IEEE workshop on dependable and real time e-commerce systems (DARE-98), Denver, June 2,1998.

[114] Miriam Herschlag, and Rami Zwick, "Internet Auctions – Popular and Professional Literature Review", International Journal of Electronic Commerce, 1(2), 2002, pp 161-186.

[115] Jarrod Trevathan and Wayne Read, "Design Issues for Electronic Auctions", School of Information Technology, James Cook University Tech. Rep. (2004).

[116] C. Boyd and W. Mao, "Security Issues for Electronic Auctions," Tech. Rep., Hewlett Packard, TR-HPL-2000, 2000.

[117] Graham, T., "The Future of Mobile Gaming", SmartPhone Conference 2000. Can be found at: http://www.gsmworld.com/technology/applications/entertainment.shtml.

[118] AOL Instant Messenger. http://www.aol.co.uk/aim/

[119] Yahoo messenger. http://messenger.yahoo.com/

[120] ICQ. http://www.icq.com.

[121] MSN Messenger. http://get.live.com/messenger/overview

[122] Shuk Ying Ho and Sai Ho Kwok, "The Attraction of Personalized Service for Users in Mobile Commerce: An Empirical Study", ACM SIGecom Exchanges, 3(4), January 2003, pp 10-18.

[123] S.Y. Ni, Y.C. Tseng, Y.S. Chen and J. P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network", Proceedings of Mobile computing (MobiCom'99), Seattle, Washington, August 15-20, 1999, pp 151-162.

[124] B.Willams and T.Camp, "Comparison of broadcasting techniques for Mobile ad hoc networks", Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002), Lausanne, Switzerland, June 9-11, 2002, pp 194 -205.

[125] M. P. Wellman and P.R. Wurman, "Real Time Issues for Internet Auctions", IEEE Workshop on Dependable and Real Time E-commerce Systems (DARE-98), Denver, USA, June 2, 1998.

[126] Windows Sockets. http://www.sockets.com/winsock.htm.

[127] A. Ahuja et al., "Performance of TCP over Different Routing Protocols in Mobile Ad-Hoc Networks", Proceedings of the IEEE 51st Vehicular Technology Conference (VTC 2000), volume 3, Tokyo, Japan, May 15-18, 2000, pp2315-2319.

[128] T.D. Dyer and R.V. Boppana, "A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks", Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001), Long Beach, CA, USA, October 4-5, 2001, pp56-66.

[129] Z. Fu, X. Meng, and S. Lu, "How Bad TCP can Perform in Mobile Ad Hoc Networks", Proceedings of the IEEE Symposium on Computers and Communications (ISCC 2002), Taormina/Giardini Naxos, Italy, July 1-4, 2002, pp 298-303.

[130] G. Holland, and N.Vaidya, "Analysis of TCP performance over Mobile Ad Hoc Networks", Wireless Networks, 8(2-3), 2002, pp 275-288. ACM Press.

[131] J.Li, C.Blake, D.De Couto, H. Lee, and R. Morris, "Capacity of Wireless Ad Hoc Wireless Networks," Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom'01), Rome, Italy, July 16-21, 2001, pp61-69.

[132] S. Xu, and T. Saadawi, "Does the IEEE 802.11 MAC protocol Work Well in Multi-hop Wireless Ad Hoc Networks?" IEEE Communication Magazine, 39(6), June 2001, pp130-137

[133] S. Xu, and T. Saadawi, "Revealing the Problems with 802.11 MAC Protocol in Multi-hop Wireless Networks", Computer Networks, 38(4), 2002, pp531-548.

[134] Giuseppe Anastasi, Eleonora Borgia, Marco Conti, Enrico Gregori, "IEEE 802.11b Ad Hoc Networks: Performance Measurements", Cluster Computing, 8(2-3), 2005, pp 135-145.

[135] Henrik Lundgren, Erik Nordström, and Christian Tschudin, "The Gray Zone Problem in IEEE 802.11b based Ad hoc Networks", Mobile Computing and Communications Review, 6(3), pp104-105.

[136] Valeri Naoumov, Thomas Gross, "Simulation of Large Ad Hoc Networks", Proceedings of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2003), San Diego, CA, USA, September 19, 2003, pp50-57.

[137] Rimon Barr, Zygmunt J. Haas, Robbert van Renesse, "JiST: Embedding Simulation Time into a Virtual Machine", Proceedings of the 5th EuroSim Congress on Modelling and Simulation, Paris, France, September 6-10, 2004.

[138] Rimon Barr, "SWANS- Scalable Wireless Ad hoc Network Simulator User Guide", available online at: http://jist.ece.cornell.edu/docs/040319-swans-user.pdf.

[139] Tracy Camp, Jeff Boleng, Vanessa Davies, "A Survey of Mobility Models for Ad Hoc Network Research", Wireless Communication & Mobile Computing (Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications), 2(5), 2002, pp 483-502.

[140] J. Boleng, "Normalizing Mobility Characteristics and Enabling Adaptive Protocols for Ad Hoc Networks", Proceedings of the Local and Metropolitan Area Networks Workshop (LANMAN), Boulder Co., USA, March 2001, pp 9-12.

[141] Hannes Frey, Daniel Görgen, Johannes K. Lehnert and Peter Sturm, "A Java-based uniform workbench for simulating and executing distributed mobile applications", FIDJI 2003 International Workshop on Scientific Engineering of Distributed Java Applications,

Luxembourg, Luxembourg, November 27-28, 2003. Available online at: http://www.syssoft.uni-trier.de/soul/download/paper/Frey03.pdf.

[142] Paul Grace, Gordon S. Blair, Sam Samuel, "Middleware Awareness in Mobile Computing", Proceedings of 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), Providence, Rhode Island, USA, May 19-23, 2003, pp382-387.

[143] B. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications", Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, December, 1994, pp85-90.

[144] Marco Conti, Gaia Maselli, Giovanni Turi, Silvia Giordano, "Cross-layering in Mobile Ad Hoc Network Design", Computer, 37(2), 2004, pp48-51.