ABC:

Anonymous routing Based on Characteristics protocol

by

Yang Guoxian

A dissertation submitted to the University of Dublin,
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science

September 2007

DECLARATION

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

____

Yang Guoxian

September 14, 2007

PERMISSION TO LEND AND/OR COPY

ACKNOWLEDGEMENTS

I would like to thank my project supervisor, Stefan Weber, for his enormous help and for our enjoyable meetings during this year. To all my family and friends for support everyday. Finally, to the UbiCom people, for making this a most memorable year.

YANG GUOXIAN

.

ABSTRACT

Ad hoc networks are a type of wireless networks that are characterized through the absence of infrastructure. The nodes utilise radio signals for one-hop communication and leverage routing algorithms to achieve multi-hop communication within a network. For the last few years, wireless communication devices have seen a sharp development in both capacity and popularization. Correspondingly, hardware development catalyzed the rise of ad hoc applications. Wireless internet access, ad hoc data sharing, wireless sensor network and etc are very good examples.

However, the nature of ad hoc networks introduces new challenges for end-to-end communications. Because of the high dynamism of ad hoc networks, information of a node may be lost in a short time. Therefore, a routing protocol that does not rely on definite knowledge of destinations is needed.

In this project, we proposed a novel routing protocol called "**a**nonymous routing **b**ased on **c**haracteristics" (ABC) that does not employ IP addresses or any other type of ID but uses characteristics of destinations to route packets. ABC leverages an analogy of coloured water stream in real world and forwards packets to the direction of a node that contains requested characteristics.

We have implemented ABC in OPNET modeler and performed evaluation on the features and performance of ABC to demonstrate the feasibility our protocol.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

Anonymous routing **b**ased on **c**haracteristics protocol (ABC) is a proactive distance vector routing protocol without any assumption of underlying protocol. ABC does not employ IP address or any other type of ID to route packets, though an ID, in this paper a MAC address, is employed in one-hop communication between neighbours. In this chapter, we will first introduce the general challenges and issues in ad hoc networks and analyse the problems of current routing protocols. Based on this analysis, we introduce the basic concept and philosophy of ABC.

1.1 Background

Ad hoc networks are a type of wireless networks that are characterized through the absence of infrastructure. The nodes utilise radio signals for one-hop communication and leverage routing algorithms to achieve multi-hop communication within a network. Usually ad hoc networks connect to wired networks (i.e. internet) through gateway nodes.

For the last few years, wireless communication devices have seen a sharp development in both capacity and popularization. Correspondingly, hardware development catalyzed the rise of ad hoc applications. Wireless internet access, ad hoc data sharing, wireless sensor network and etc are very good examples.

However, the nature of ad hoc networks and the requirements of applications impose new challenges which traditional communication solutions for wired networks can not meet.

- **Resources constrains**: In ad hoc networks, the transmission range and bandwidth of individual nodes are very limited. The capacity of mobile device is very limited compared with that of traditional workstations.

- **High diversity of mobile devices**: Compared with wired communication, wireless ad hoc communication highly depends on the workstation itself. However, heterogeneous devices differ in characteristics such as storage capacity, internet access, processing ability, mobility, and power capacity, etc.

- **Dynamic topology**: The mobility of devices leads to changes in network topology. Compared to fixed wired communication, frequent path breaks and stale routing information are major issues for end-to-end communication in ad hoc networks.

- **Wireless communication**: Due to the nature of the wireless medium, one-hop communication experiences a high rate of packet collisions and loss. Exposed terminal and hidden terminal problems are also known to be a challenge in a wireless environment.

- **Security and privacy**: High distributed topology and wireless communication of ad hoc networks impose security and privacy challenge to ad hoc applications. Limited capacity of mobile devices accents the difficulty of protection

The requirement for a routing protocol in ad hoc network is to acquire the route information to a given destination node or a set of destination nodes efficiently. Generally, "efficiency" here means the following:

- **Minimum route acquisition delay**: The delay for the acquisition of a route should be as small as possible because the topology of a network may change quickly and short delays provide precise information which lead to better performance for applications.

- **Optimum route**: The route acquired should be a valid route toward the destination and achieve an optimization in one of the following aspects: Hop counts, end to end delay or bandwidth. Further more, the routing protocol should provide certain level Quality of Service (QoS) support.

- **Minimum control overhead**: To satisfy the first two criterions, a routing protocol needs to exchange information between network nodes in addition to application data. This additional information is called control overhead. However, due the scarce bandwidth of ad hoc networks, the control overhead should be as small as possible and achieve a best balance with the requirement above.

- **Autonomous and distributed**: Central control in ad hoc network introduces large amount of control overhead and lack of flexibility. So, nodes should be able to operate in a fully distributed manner.

- **Application-oriented**: "One size doesn't fit all". Even there are requirements for each layer of ISO-OSI model, taking specific application requirements into the consideration of routing protocol will benefit.

- **Scalability**: A routing protocol in ad hoc network should be able work extend to large network as the density of mobile devices in an ad hoc network may be large.

1.2 Motivations

Current routing protocols utilize IP addresses to distinguish nodes in the network. However, because of the high dynamism of ad hoc networks: Nodes may join or leave the network very frequently, it makes not so much sense to look for a specific IP address but a node that satisfies some requirements or provides certain services. Moreover, the assignment of IP address in ad hoc network is not completely distributed. IP address of a recently joined node may be assigned through a DHCP server. For stateless address configuration in IPv6, where assignment is distributed, extra overhead is introduced.

Current research that can retrieve knowledge about nodes in an ad hoc network includes service discovery, which is built upon IP layer, and content-based network, which can be independent from underlying protocol. Most proposed protocols in both approaches are implemented at middleware or application layer. They create large amount of overhead in ad hoc network where bandwidth is very limited.

A network routing protocol that addresses the problem is needed. This protocol should be light weight compared with service discovery and content-based routing protocol and should not depend on IP layer. Instead, it should able to cooperate with IP without extra configuration.

1.3 Key points of ABC

To meet the aforementioned challenge and requirement, we introduce a proactive distance vector routing protocol based on characteristics rather than IP address. Each node in ABC network periodically spread information about its characteristics and the characteristics it hears from other nodes to its neighbours. The characteristics here could be:

- Service from application layer: Such as internet access, printing.
- Underlying characteristics of ad hoc devices: Such as CPU capacity, storage.
- IP address: This makes cooperation between IP and ABC easily.

A characteristic in an ad hoc network does not have to be unique. A node could possess multiple characteristics while same characteristic may originate from multiple source nodes. To achieve this, we introduce concepts of characteristic abstraction and sequence number synchronization in basic design. In advanced design, we propose a numerical approach to improve the performance of ABC.

With characteristics information disseminated throughout a network, packets that are destined for a characteristic are forwarded to direction biased to one or multiple source nodes.

Chapter 2: Related Work

2.1 Routing Protocol

General routing protocols retrieve route information and forward data packet along the retrieved path to a specific node. They can be broken into three major categories:

- Proactive: Topology information is disseminated through an ad hoc network periodically. All possible paths are maintained.

- Reactive: A route between a source and a destination is established on-demand in reactive routing protocol. It creates small control overhead but experiences more route acquisition delay.

- Hybrid: Combining the characteristics of proactive and reactive protocols, a hybrid routing protocol performs proactive routing within multiple small regions and performs reactive routing in a whole network.

There is no Holy Grail in ad hoc routing. Depending on node density, mobility and applications, a routing protocol performs different. Generally, proactive routing protocols experience a short route acquisition delay and react on topology changes well. However, a relatively large amount of control information is introduced. Existing proactive routing protocols focus on minimizing the control overhead as well as retaining the performance advantage against reactive routing protocols. In this section, we examine two proactive routing protocols as ABC is a proactive routing protocol.

2.1.1 Optimized Link State Routing Protocol:

OLSR is a proactive routing protocol. Due to the proactive nature, OLSR excels in route acquisition delay and thus is change tolerant of network topology. Scalability and reduced control overhead are achieved by introducing an optimized flooding control message.

Generally, routing protocols, link state or distance vector, use control messages to collect and distribute topology information of the network in order to compute the optimized path and to forward the data packets between source and destination nodes. OLSR, on the other hand, uses control messages (HELLO message) for the efficiency of distribution of Topology Control messages (TC messages), which are then used for forwarding of data packets. This is like "second derivative" in mathematics term, if we compare control information of a packet distribution to derivative of distribution function. Using the "second derivative" control message, OLSR distribute the topology information in an optimized way.

In OLSR, neighbour nodes exchange HELLO messages, thus node can gather information about its one-hop neighbours and two-hop neighbours. A set of symmetric one-hop neighbour nodes, note as Multiple Point Relay (MPRs), are then selected such that all the two hop neighbours are covered by neighbours of MPRs. The node itself is called MPRs Selector. Node is only responsible for forwarding packets from its MPRs Selectors, and will periodically spread the existence of its MPRs Selectors. This neighbour sensing and topology information flooding scheme eliminates the duplicate of control information in the network and forms the core function of OLSR. In optional design, jittering and piggybacking will improve the performance of OLSR.

OLSR is very well suited for large and dense network. It generates more overhead compared with general reactive routing protocol, which in turn gains robustness and reacts quickly against the topology transformation.

2.1.2 Termite

Termite is a biologically inspired ad hoc routing protocol leveraging swarm intelligence theory. The algorithm is analogous to the behaviours of social insects: Stigmergy, which refers to indirect communication between individuals through the environment [3]. The packets of Termite are considered to route themselves and are able to influence the path of other packets.

To achieve the analogy above, Termite introduces the concept of pheromones. Each node keeps a pheromone value for a known destination at a possible next-hop links towards the destination. Packets destined for a certain node are biased to forward toward links with strong pheromone gradient. At the same time, all the packets increase pheromone value for their source nodes on their incoming links. Throughout this procedure, three fundamental elements are used:

- **Positive Feedback**: Arrival of a packet, which is not necessarily a control packet, increases the pheromone value of the source node on the incoming link by a constant. The packet could either destine for the node or overheard by the node. The more packets from a destination come from a link, the higher pheromone value the link accounts for the destination. Positive feedback emphasized the likelihood of selection on active next-hop link.

- **Negative Feedback**: In nature world, pheromone evaporates and weakens over time. In Termite, all pheromone value decays at a const percentage periodically. When a pheromone of a destination at a next-hop link is smaller than a predefined value, this entry is removed from pheromone table. This is called negative feedback. Negative feedback acts the role of timeout purge of stale route information in general routing protocol, but in a more nature way. Without new arrival positive feedback, route becomes less selected gradually before finally becoming stale.

- **Randomized Route Selection**: Termite imposes randomness based on pheromone value onto route selection. Routes with high pheromone value are more likely to be selected.

In Termite, all the packets a node sends out accumulate pheromone along their paths. Packets destined for this node will go along these reverse paths. It works well in a network with high node density and activity, because data packets also transmit topology information. If there is not enough activity in a network, Termite allows a node emitting a seed packet to walk through the network randomly. To prevent stale routes information, a Time-To-Live (TTL) field is leveraged in Termite packets. Compared with sequence number, TTL requires control overhead but can not guarantee a quick remove of stale routes.

Given the assumption of certain amount data throughput and the fact that seed packets can be generated to spread existence information of a node, Termite can be treated as a proactive routing protocol. It retrieves routing decision quickly, even though the routes acquired is usually “acceptable” but optimal.

Utilizing swarm intelligence and probability techniques, Termite is a simple, robust routing protocol. The arrival of data packet implies the topology information. Therefore, no extra control overhead is needed to spread the existence of the node in a network with certain amount of data communication. Control overhead in much decreased.

2.2 Data-oriented Protocols

As mentioned above, routing protocols based on IP address suffers from lack of knowledge about destination. Service discovery and content-based networks are developed to meet this requirement. They provide a pattern to directly communicate with nodes based on the data rather than IP address of destinations. Here, “data” means either content of information flow or services that a node provides. We use the term “data-oriented” to differentiate this type of protocols from general IP-based routing protocols. In this section, we examine two data-oriented protocols in ad hoc network.

2.2.1 Service Discovery for Mobile Ad hoc Networks

Service discovery is built upon the IP layer to search services in a network. There are three important concepts service discovery that should be announced beforehand. A *client* is a node that requests for a certain service. A *server* is a node that provides certain service(s). *Directory* is a node that stores the service descriptions of the network. Thus, service discovery protocols use a Client/Server (C/S) communication model. Generally, service discovery in mobile ad hoc networks uses two major approaches:

- Directory-less approach: No directory is held in the network. The *server* floods the network with its service advertisement or the *client* floods the network with request for a service. In ad hoc networks, simple flooding schemes are not acceptable because of the overhead that will be generated. Optimized schemes, such as Multi-Point Relays (MPRs), which are often seen in IP routing protocol have been introduced to improve flooding. Examples of this approach are UPnP and DEAPspace.

- Distributed directory approach: *Directories* are dynamically distributed through the network. Instead of searching for a service provider, the *client* tries to locate a directory first, retrieves the service information, and then communicates with the *server*. The major task for distributed directory approaches is cooperation between directories because both client and server move frequently.

In most service discovery systems, IP addresses of servers that provide requested services are returned to a client. The client then has to resort to IP routing protocols to communicate with server.

2.2.2 Content-Based Routing Protocol

A content-based network is a virtual infrastructure where nodes' predictions are used as addresses to substitute traditional IP address.

In content-based routing protocols, a flow of information is routed based on the content of the data and the specific interests of destination nodes rather than according to IP addresses as in traditional IP routing protocols. Content-based routing uses Publisher/Subscriber model with three components: Publishers, subscribers and brokers. Publishers are producers of information. Subscribers are consumers of data. Brokers are nodes between subscribers and publishers. Information flows from a publisher is filtered at a broker according to subscriptions. Data units in a flow are then forwarded to nodes which has subscribed to this information. In such models, subscribers and publishers are decoupled because both can be anonymous to each other [4].

Content-based routing protocol performs a type of multicast. Compared with traditional multicast, multicast groups are created dynamically based on the content of information of interests instead of IP address [4]. A content-based routing protocol is not necessarily built upon IP and IP routing protocols, which makes it a lighter weight compared to service discovery, even though some uses underlying routing protocol to achieve better reliability.

Similar to service discovery, content-based network has initially been designed for wired networks. In ad hoc networks, where there is no infrastructure and the topology is highly dynamic, traditional centralized brokers are replaced with a network of distributed brokers. The efficiency of the cooperation between these brokers and how to maintain an up to date are important issues.

2.3 OPNET: A network simulator

OPNET is a network simulator for the study of communication based on any OSI or user-specific layers. It provides a comprehensive development environment which consists of model design, simulation, statistic collection and analysis.

OPNET is a hierarchical object-oriented simulator. Modelling spaces are divided into four domains: Network, Node, Process and External System. These domains describe a communication system from external to internal perspective. Models are defined in corresponding domain, for example a MANET Node model is defined in Node domain. An instance of this MANET Node model then can be configured and constitute an ad hoc network. Instances inherited from a model are called objects. A communication system, which itself is an object of Network model, is constitutes by a group of objects. OPENT models are each associated with a set of attributes. Model attributes can be predefined before execution, self-configured during execution, or configured through attribute interfaces. Attribute interfaces connect cross-domain models.

The Network domain specifies a network scenario. The communicating entities in a network scenario are instantiated from Node models and referred to as Node instances or Node objects. The Node domain provides modelling of communication devices deployed and interconnected in a Network scenario. A Node model describes functionalities of a class of Node instances in term of building blocks called modules. For example, each layer in OSI model can be implemented in a module. A possible OPNET Node model representing TCP/IP stack is shown as follows [25]:

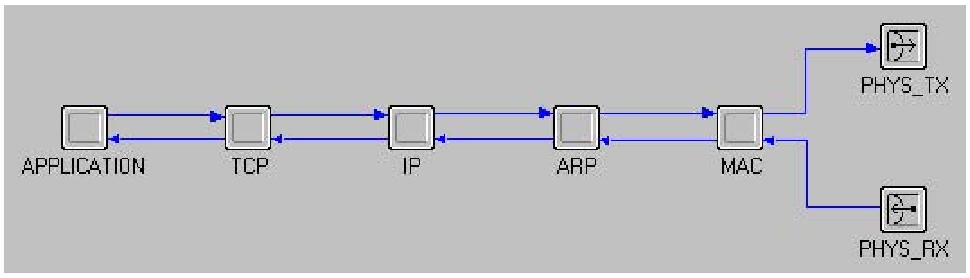


Figure 2-1: TCP/IP stack implemented in OPNET

There are several types of modules: Data sources/sinks, transmitters/receivers and processors/queues. Data sources/sinks generate and destroy packets. Transmitters/receivers simulate behaviours of physical devices by sending and receiving packets. These modules have predefined behaviours, while processors/queues are highly programmable. They are linked through packet streams, statistic wires, and logical associations. Modules are specified by processes. A process is similar to an executing software program and forms tasks for a module. The Process model which a process instantiated from is defined by process editor in the Process domain. There is at least one process called root process in each module. A process can create and invoke other processes. Only one process can be executing at a time.

Processes are described through finite state machines called State Transition Diagrams (STDs). Behaviours of STDs are defined by a general C/C++ programming language in addition to a library of high-level API known as Kernel Procedure provided by OPNET. Processes are driven by interrupts. They are designed to respond to interrupts and to invoke interrupts. The communication between individual processes is done through interrupts. Interface Control Information (ICI) is a formal interface between processes. It is hooked with an interrupt and passed to next process by the invoking process.

OPNET provides large amount of standard process models, from underlying transmission devices, general applications to common protocols. Users can incorporate these models into user-defined systems or evaluate them against user defined models. These models form a uniform environment for all simulations.

Statistics collection and analysis are needed in order to evaluate a communication system. General statistics have already been collected in standard models in OPNET. Meanwhile, specific data of interest can be record via statistic variables in a process. Data analysis in OPNET is a graphing numerical process. In addition, statistics in different simulations can be studied in the same panel.

Chapter 3: Design

In this chapter, we first introduce the scenario of how characteristics spread in ABC, which is an analogy of a coloured water stream. Terminology for this protocol is then explained. Basic design describes the general operations and principles of ABC, followed by an analysis of shortcomings of this design. Then, a numerical approach is applied to characteristic modelling, which we will see an improvement on performance in later chapter.

3.1 Overview

In ABC, we replace IP addresses with characteristics of a node. There may be multiple sources of a characteristic in a network, and a node can have multiple characteristics. Now, we give a general overview of ABC.

In the case of an ad hoc network with a single characteristic, there can be multiple sources of the characteristic in the network. Each source spreads out the characteristic through hello packets to its neighbours. Its neighbours inherit this characteristic and distributed further. This procedure could be treated as a characteristic flow. Different flows of a characteristic can meet and merge at an intermediate node and form a new flow. After an initiation phase, all flows should finally reach a stable state. As stated in Chapter 1, to meet the challenge of mobile ad hoc networks, the initiation time duration should be as small as possible. In the case of an ad hoc network with multiple characteristics, each characteristic follows the same steps above to flow over the network. On receiving multiple characteristics, a node fuses them before rebroadcast.

The propagation of characteristic flow and the fusion of different characteristics are inspired by the color stream blending and propagation. Consider the following scenario: Node B is the originator of a stream with the color red, while node C is the originator of a stream with the color yellow. Their output streams merge A, and the colours blend there into orange. A now carries orange and distributes the color stream further to node X. The density of a color reduces as the color stream propagates, which means a color is blurred at a remote observer. So the downstream node, X, contains less orange.

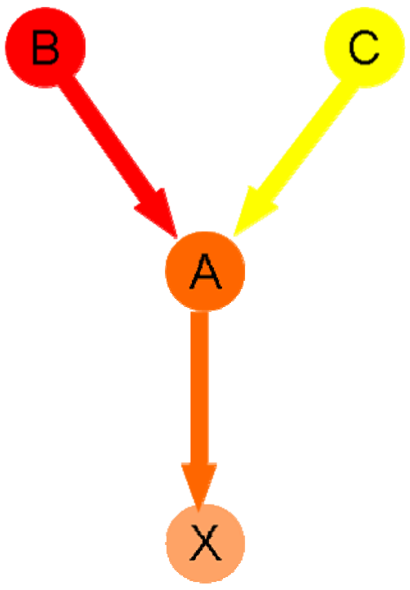


Figure 3-1: Color stream

Assume that node X issues a request for red. Since node A is the source for the color orange it implies a possibility for the color red. Node X will send a request for red to A. A is not the originator of red, but it has information about the real source, node B. Thus, node A will forward the request to node B.

The basic problem for design of ABC is: How do multiple instances co-exist, while not incurring overhead explosion? We propose an approach called “characteristic abstraction” in the basic design and an approach called “characteristic fusion” in the advanced design. Sequence number synchronization is used in each design to manage the pace of each instance, since we use sequence number to prevent stale route information.

3.2 Terminology

- **Characteristic**: In ABC, a node is represented with a group of characteristics. As described in Chapter 1, a characteristic can be encoded from either underlying features of ad hoc devices or services provided at the application layer of an ad hoc device.

- **Characteristic set**: In basic design of ABC, a characteristic set contains characteristics that describe the same feature or services (FoS). For example, internet access set contains characteristics that represent internet access services. Instances of characteristic set vary in capacities of FoS.

- **Weight**: In advanced design of ABC, weight of a characteristic is an integer number that describes the capacity of the characteristic. This implies in advanced design, each characteristic set has only one characteristic.

- **Internal characteristic**: A characteristic that a node generates is called internal characteristic of the node. Internal characteristic can be configured by applications. For the purpose of this paper, we will not discuss the procedure of configuring internal characteristics.

- **External characteristic**: A characteristic that a node inherits from its neighbours is called external characteristic.

- **Local characteristics**: All characteristics, both internal and external, stored in a node are called local characteristics of a node.

- **Characteristic instance**: A node can receive same characteristic that come from different last hop, which means they are in different flows and may originated from different source. In ABC, we call them characteristic instance.

- **Active characteristic**: An active characteristic is a characteristic that a node will spread out through hello messages. In the basic design, internal characteristics - or if there is none, most detailed characteristics of a characteristic set - are selected as active characteristics. In advanced design, an active characteristic is computed from all instances of local characteristics.

3.3 Basic Design

The objective of the basic design is to show the feasibility of characteristic-based routing. We first compose a sample model of characteristics. According to this model, we discuss concepts of two fundamental functions of basic design. Then we describe the routing procedure. And at last we analyze the shortcomings of this design.

3.3.1 Characteristics Modelling

The following graph shows a sample model of characteristics. The level of the node in the tree view shows the detail of the characteristic. The lower level of a characteristic is, the more detailed it is. Initially, all internal characteristics should be set at leaf node level because a characteristic is the most detailed at its source.

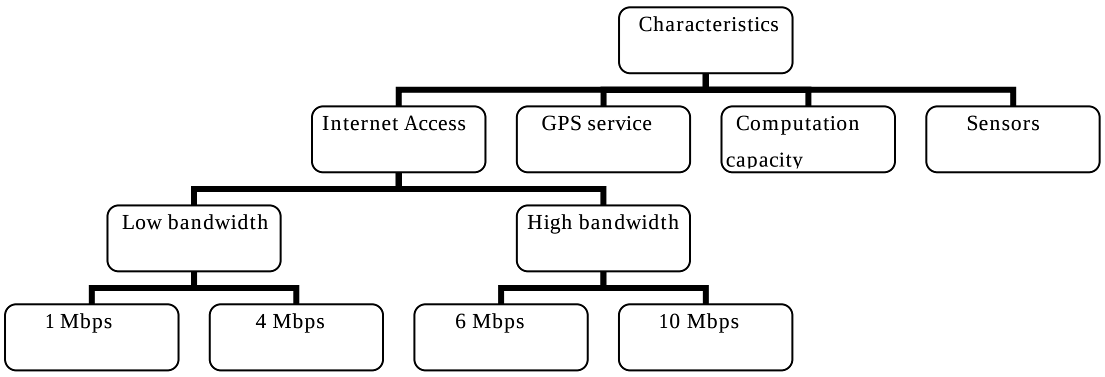


Figure 3-2: Characteristics model

In this project, we use terms “dominance” and “precedence” to denote the relations between nodes in the model above and use kinship terminology to described related nodes in the tree view. The characteristics are encoded based on the n-ary tree structure. An encoding scheme for the example above could be the following:

Characteristics	**Code**
Characteristics	1
Sensors	100
Computation capacity	101
GPS	110
Internet Access	111
1 Mbps	11111
4 Mbps	11110
6 Mbps	11101
10 Mbps	11100

Table 3-1: Characteristics encoding

In this encoding scheme, every code begins with 1 and binary numbers are appended as suffix. This implies that the length of a code indicates the level of the characteristic that it represents in the tree structure. All characteristic codes are brought to the same length by inserting prefix 0. This scheme simplifies the operations on the characteristics. See Appendix for an optional encoding scheme.

3.3.2 Characteristic abstraction and synchronization

Characteristics abstraction describes the procedure in which a characteristic is abstracted to higher level and becomes less detailed along the propagation. Scalability of the basic design comes from characteristic abstraction, as multiple characteristics in the same characteristic set can merge and fuse into an abstracted one at a remote node. In the encoding scheme we use in this project, characteristic abstraction is to left-shift the code by 1 bit.

In ad hoc network routing protocols, sequence numbers are usually used to prevent stale or broken routes information. Since our protocol is essentially a distance vector routing protocol, sequence numbers are also leveraged. However, there can be multiple sources of a characteristic, and different instances of a characteristic may merge at an intermediate node. In this situation, there is no easy way to keep a separate sequence number for each node as in general routing protocol. To address this issue, we introduce sequence number synchronization in our protocol: A sequence number is maintained for each characteristic at all nodes in a network. On receiving a characteristic instance of the same characteristic that is associated with a higher sequence number, a node synchronizes the internal sequence number with the external one. In this way, multiple sources of a characteristic keep the same pace as time goes on. However, it creates large amount of overhead to keep an accurate pace for all the sources. A predefined maximum discrepancy of sequence numbers, indicated as *sd*, is used in synchronization to tune the accuracy of sequence number and overhead it creates. Sequence number synchronization is also used in advanced design, which we will discuss later.

Because of characteristic abstraction, different characteristics in the same characteristic set can be abstracted into the same characteristic at higher level in intermediate nodes. This means that not only instances of a characteristic should keep sequence number synchronized but characteristics in the same characteristic set should also keep sequence number synchronized.

3.3.3 Header Design and Characteristic Table

In this project, we define three types of packets: Hello packet (HELLO), request packet (RREQ), and reply packet (RREP). They share a fixed header as shown below:

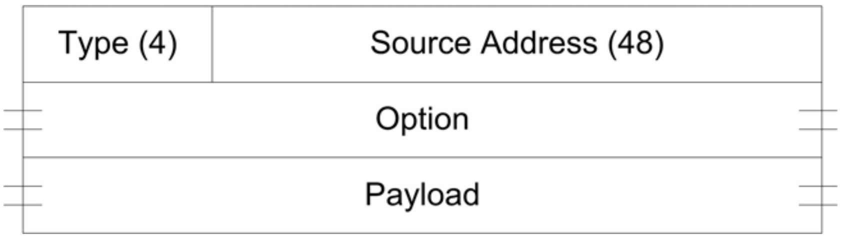


Figure 3-3: Fixed header of ABC packet

The "Type" field specifies the type of optional header, which is appended in "Option" field. Since we use MAC address for communication between neighbours, the "Source Address" field is set 48 bit. Optional headers are given below. The "Data Payload" field in request and reply packets indicate whether the packets carry data content.

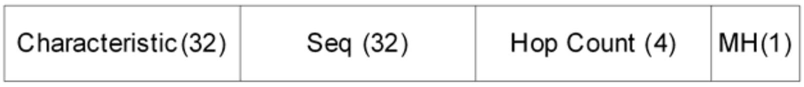


Figure 3-4: Optional header – Hello packet

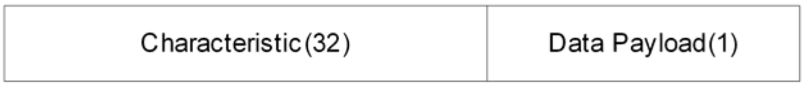


Figure 3-5: Optional header – Request packet

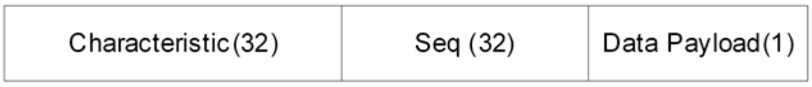


Figure 3-6: Optional header – Reply packet

Characteristic table essentially plays a similar role as routing table in general routing protocol. It consists of two tables:

- Local characteristic table: Same as forwarding table, it contains local characteristic entries. Fields are <characteristic, last hop address, sequence number, hop count> and <characteristic> is the primary key.

- Stale characteristic table: Time expired entry in local characteristic table is removed and inserted into stale characteristic table.

3.3.4 Characteristics Flow

Generated from its source node, a characteristic spreads in partial or full of a network, which can be treated as a characteristic flow. The procedure of a characteristic flow in ABC contains four major steps:

1) Active characteristic selection: First a node evaluates and selects active characteristics from local characteristics using the following criterions.

- It is an internal characteristic

- The most detailed characteristic in the set is selected active.

2) Characteristic spread: The node then performs the following operations on the selected active characteristics and spreads them out through hello packet to all its neighbours: The node increases the hop count of each active characteristic by one. If the hop count reaches a predefined number, *c*, the characteristic is abstracted to a higher level. If an abstracted characteristic is the root node of the characteristic model tree, which carries no valuable information, the propagation of this information should stop. In this way, the propagation range of a characteristic is limited. If a local characteristic is internal, sequence number is increased by one. Then the processed active characteristics are disseminated out to all the neighbours by hello message.

3) Characteristic update: On receiving characteristics from a neighbour, a node performs validation on both the incoming characteristic and the local characteristics. According to the validation result, the local characteristic table is updated. Internal characteristics are synchronized if necessary. The following flow chart shows the detail operation a node follows on receiving hello packet.

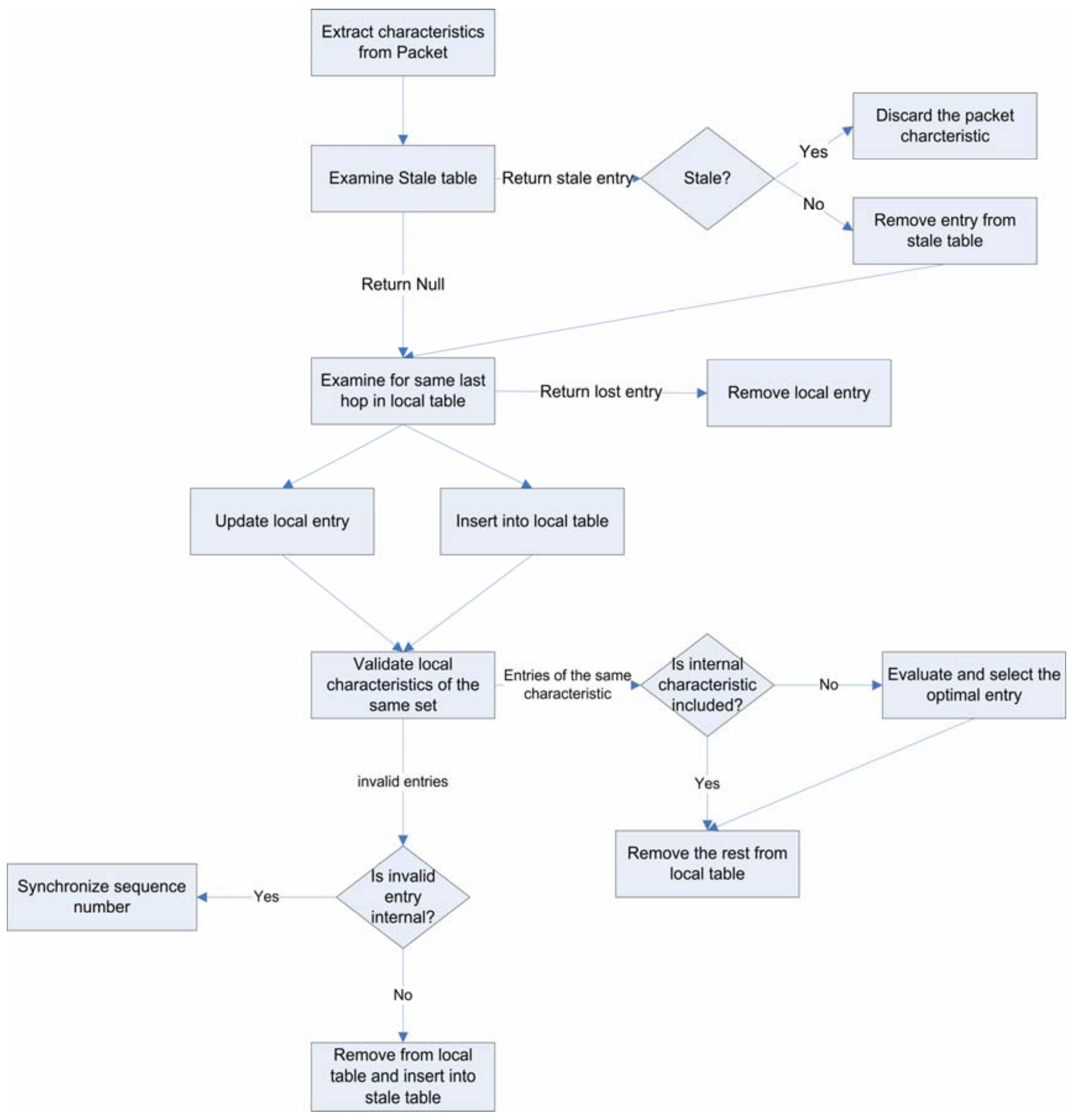

Extract characteristics from Packet
Examine Stale table
Return stale entry
Stale?
Yes
Discard the packet charcteristic
No
Remove entry from stale table
Return Null
Examine for same last hop in local table
Return lost entry
Remove local entry
Update local entry
Insert into local table
Validate local characteristics of the same set
Entries of the same characteristic
Is internal characteristic included?
No
Evaluate and select the optimal entry
Yes
Remove the rest from local table
invalid entries
Is invalid entry internal?
Yes
Synchronize sequence number
No
Remove from local table and insert into stale table

Figure 3-7: Characteristics update in basic design

On receiving a hello packet from a neighbour, a node first extracts characteristics from the packet, in this thesis we call these incoming characteristics packet characteristics. The node then examines the stale characteristic table. For a packet characteristic that matches entry in the stale table, if the sequence number of the packet characteristic is larger than that of the characteristic entry in the stale table, then remove the stale entry. Otherwise, the node discards the packet characteristic.

Local characteristics that come from the same last hop with the hello packet are selected and compared with remaining packet characteristics. Local characteristics which are not in hello packet received from the last-hop node are treated as lost in the source, thus are eliminated from the local table and insert into the stale table. Packet characteristics that are not in the local characteristic table are treated as new characteristics from the source, thus are inserted into the local characteristic table. Local characteristics that also appear in hello packet are updated with packet characteristics.

The local characteristic table contains only one entry for each characteristic and all flows of characteristics in the same set keep the same sequence pace. Now the local characteristic table may contain duplicate or invalid entries. For local characteristics in each characteristic set, the node first examines and retrieves the largest sequence number, and filter invalid entries using sd. Sequence number synchronization is then performed on invalid internal characteristics. Invalid external characteristics are removed from the local characteristic table and inserted into the stale table. Finally duplicated entries are detected. If internal characteristic is included, keep it and eliminates the rest. Otherwise evaluate duplicated entries and keep the optimal one. Optimization means shortest route or if the hop counts of multiple entries are the same, the largest sequence number.

4) Purge stale characteristic: Removed from the local characteristic table, an entry increases its sequence number by one and is inserted into the stale table. We have already seen two cases in which local entries may be purged: Lost at the source node and sequence number invalidation. The third case is timer expires. In the case that a neighbour moves out of the transmission range, a node will not receive explicit information of lost characteristics. A timer is installed for each entry in the local characteristic table which is reset upon update. If timer expires, local entry is purged and insert into the stale table.

Characteristic flows through the network following the steps above. All nodes in the propagation range of a characteristic instance will sense its information. Granularity of this information really depends on the minimum distance to the source.

3.3.5 Characteristics Request and Reply

Characteristic request in ABC is also based on one-hop communication. A node has no knowledge about who is the real source of the requested characteristic or the originator of the request, but the next hop towards that destination. A selected neighbour handles the request without distinction between internal and external.

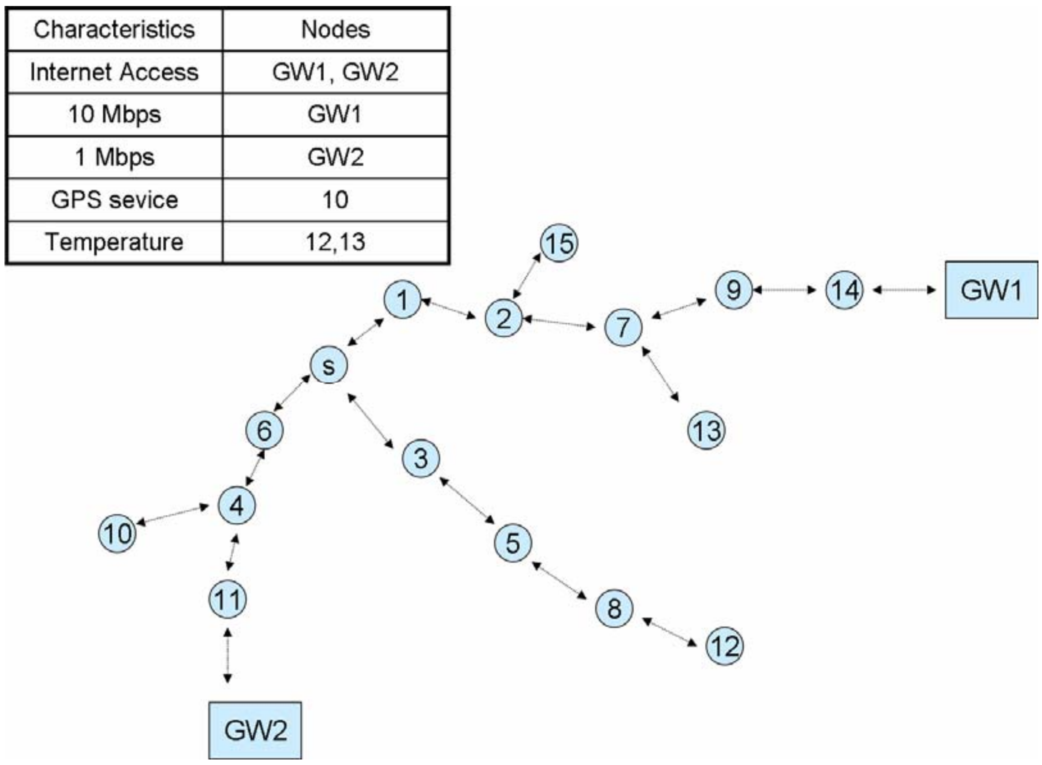
If a node wants to request for a certain characteristic, it searches local characteristic table for a suitable neighbour, and sends request to it. A neighbour is selected if it is the last-hop node of the requested characteristic. If no entry is found that match the characteristic, since abstracted characteristic contains possibility of routes to child characteristics, last hop of ancestor of the characteristic is selected. If no such entry is found, depends on the request configuration, entry that is in the same set with the requested characteristic can be selected as a compromise choice. If there is no aforementioned entry in the local table, the node then hold the request for a certain period waiting for appearance of path information before it drops the request and reports a failure.

On receiving the request packet, the selected neighbour records the last-hop address into the request table for further conversation. If the requested characteristic is internal for the node, it initiates a reply message and send to the last-hop node of the request. Otherwise it handles the request using the same pattern stated above.

For the purpose of this project, we only focus on the routing procedure that is how to forward the request to the destination and how to forward the reply back. We leave the maintenance of the session for future development. See future work for detail.

3.3.6 An Example of Characteristic Flow and Request

To better understand the procedure of characteristic flows and request, consider a static ad hoc network as follows:



Characteristics	Nodes
Internet Access	GW1, GW2
10 Mbps	GW1
1 Mbps	GW2
GPS sevice	10
Temperature	12,13

Figure 3-8: DNS Query in ABC

Assume that the hop count limit, c, is 4, which means characteristics are abstracted to higher level every 4 hops. All the nodes will spread their characteristics periodically. Take GW1 for example. GW1 broadcast an ABC control packet including “bandwidth 10Mbps – TTL 4”. At node 14, the characteristics table would be {bandwidth 10 Mbps, 1, GW1}. Node 14 will also spread its characteristics information. So at node 9 the characteristics table is {bandwidth 10 Mbps, 2, 14}. At node 7, it would be {{bandwidth 10 Mbps, 3, 9}, {temperature sensor, 1, 13}}. At node 2, the table is {{bandwidth 10 Mbps, 4, 7}, {temperature sensor, 2, 7}}. However as the TTL of the characteristic “bandwidth 10 Mbps” reaches 4, node 2 will broadcast information “High Bandwidth -- 4” and “temperature sensor – TTL 2”. Then at node 1, the table is {{High Bandwidth, 1, 2}, {temperature sensor, 3, 2}}. Finally the information would reach node S.

Received all the spread control packets, S would have a characteristics table:

Characteristics	Distance(hops)	Next-hop
High bandwidth	2	1
Bandwidth 1Mbps	4	6
GPS service	3	6
Temperature	4	12
Temperature	4	13

Figure 3-9: Local table of S

S would like to browse the website www.tcd.ie. The local resolver has no information of www.tcd.ie. So a DNS query to find the mail server of www.tcd.ie is placed into ABC packet. The DNS server address may or may not be included in the packet. (Depends on whether application specifies the DNS address).

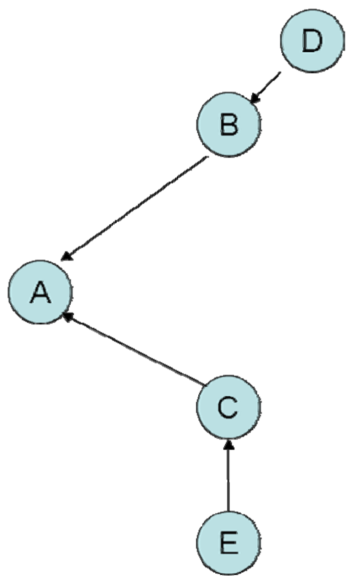
As DNS query doesn’t require a high bandwidth but a shorter response time, so node 6 is selected to forward ABC packet. Node 6 would then select next-hop node according to its characteristics table. The procedure goes on until the ABC packet reaches GW2. GW2 then will translate the ABC packet into UPD/IP packet and send it to DNS server 134.226.36.87 and port 53. A reverse path is also set up from ABC packet. The feedback of DNS is sent through the reverse path to S.

Brower of node S will then send RREQ to 134.226.36.23, which is the IP address of www.tcd.ie. As website browsing may request high bandwidth, so this time node 1 is selected as the next hop, following a similar procedure.

3.3.7 Shortcoming of Basic Design

The basic design focuses on the feasibility of characteristic flow. Performance has not been really taken into considerations. Remote characteristics, after being abstracted and lose details, can be shut down by a different characteristic which is originated nearby and has not been abstracted yet. In this case, compromise has to be made if a node request for the remote characteristic. To a certain extend, this compromise is acceptable. After all, those characteristics are in the set. The real problem of basic design is the lack of balance between capacity of a characteristic set, indicated by different characteristics within the set, and the distance to that characteristic. This problem makes it different for an application to select a characteristic to request for a given requirement.

Consider the scenario below, where hop count limit is also 4. Node A requires an internet access. It difficult for A to evaluate whether B, which may provide a better performance but may also contain uncertainty in the path towards it, or C, which is very detailed but provides limited capacity. For example, certainty of path is more valuable in ad hoc network, which means C should be selected. However, in this case, two characteristic flows really have quite similar length but flow from B provides capacity 10 times as flow from C.



Node A:

Characteristic	Hop Count	Last Hop
High bandwidth	1	B
1Mbps	4	C

Node B:

Characteristic	Hop Count	Last Hop
10Mbps	4	D

Node C:

Characteristic	Hop Count	Last Hop
1Mbps	3	D

Figure 3-10: Which to choose, distance or capacity?

So we need to take consideration of both the capacity of a characteristic and the distance to the source of that characteristic. But there is not easy way to meet this requirement in basic design, as it is difficult to evaluate characteristics at different level. That is why we develop an advanced design and an improved characteristic modelling method, which combines both elements for route selection.

3.4 Advanced Design

3.4.1 Initiatives and Characteristic Modelling

Uncertainty of a characteristic is added as the distance to its source increases. This uncertainty lies in two aspects: The possibility of route break and the possible capacity consumption along a path. A good example for this is that a route that consists of multiple hops will not achieve the same throughput as single hop communication. The bandwidth is reduced in a multiple-hop route.

In this advanced design, we assign a value called *weight* to describe the capacity of a characteristic. In the case of the example in basic design, the characteristic model can be defined as follows. Instead of assigning a separate characteristic for 1Mbps, 4Mbps, 6 Mbps, and 10 Mbps, they are assigned a same characteristic "Internet Access" and associated with different weights.

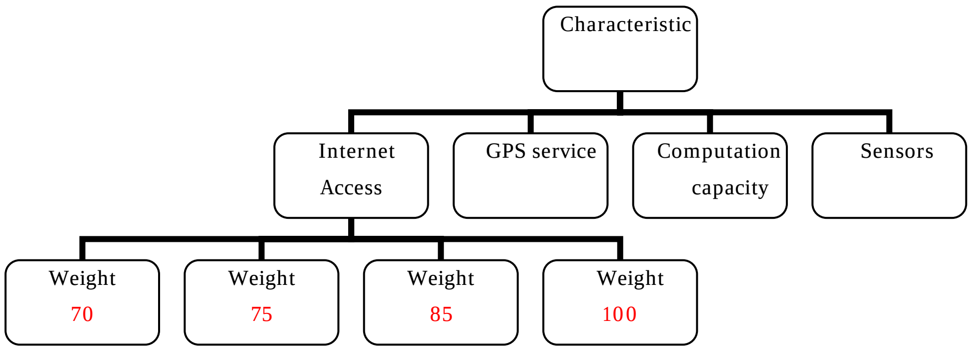
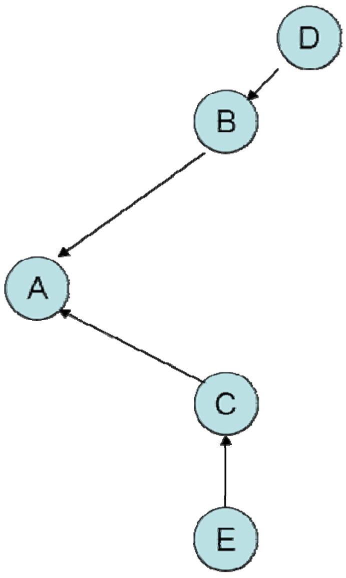


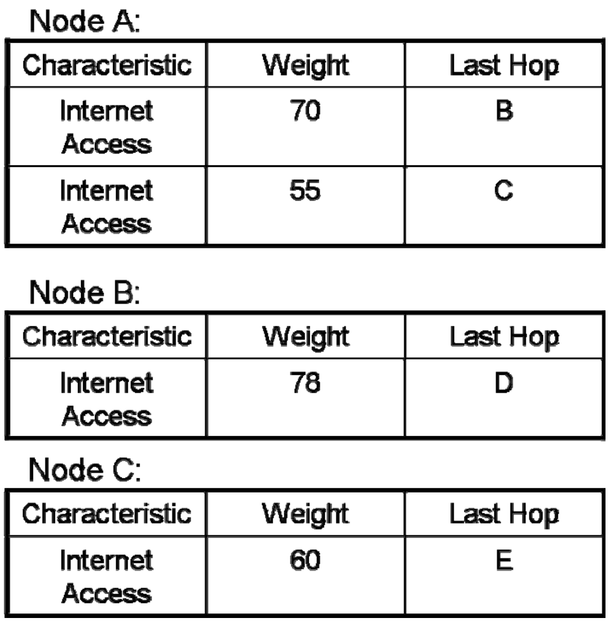
Figure 3-11: Advanced characteristic model

We use a Weight Cost Function to describe how the weight of a characteristic decreases as the flow propagates, and use Weight Fuse Function to describe how weights are merged when multiple flows meet in an intermediate node. In certain conditions, Weight Compensation Function to compensate the weight cost along the path. These numerical functions substitute the characteristic abstraction and describe how a characteristic flows in a network.

3.4.2 Scenario of Advanced Design

To illustrate how characteristic flows propagate, consider the same scenario as stated in section 3.3.7. However, since weight is introduced, flows from node B and node C indicate the same characteristic but differ in weight. And internet request from A can easily select B as next hop. Further more, to distribute workload on single link, a probability scheme can be applied to route selection.





Node A:

Characteristic	Weight	Last Hop
Internet Access	70	B
Internet Access	55	C

Node B:

Characteristic	Weight	Last Hop
Internet Access	78	D

Node C:

Characteristic	Weight	Last Hop
Internet Access	60	E

Figure 3-12: Characteristic Flow in Advanced Design

3.4.3 Characteristic Table

There are three characteristic tables in the advanced design:

- Local characteristic table: The local table stores all valid characteristic instances. The construct of this table is <characteristic, weight, last hop address, sequence number>. A noticeable difference with previous design is that the primary key is <characteristic, last hop address>. That means there can be multiple entries in local table for a same characteristic, which is different in basic design. This change increases the robustness of the protocol, as all necessary flow information is kept in the table.

- Active characteristic table: It stores active characteristics which are computed from the local table. The active characteristic table is like the appearance of a node. The construct of this table is <characteristic, weight, sequence number> and the primary key is<characteristic>. Each entry in active table is linked to a group of entries in the local table that have the same characteristic as it.

- Stale characteristic table: In advanced design, entries removed from the local characteristic table are not inserted into stale table, but those removed from the active table. If an active entry loses its entire links to local table, which means there is no incoming flow anymore, this entry is removed from active table and is inserted into stale table.

3.4.4 Details of Functionality

Core of ABC advanced design consists of four major functionalities: Characteristic spread, characteristic update, characteristic request and reply, and purge stale characteristic. Weight Cost Function, Weight Fuse Function and Weight Compensation Function which control flows in a network are included within these functionalities.

1) Characteristic spread: A node which contains local characteristics periodically inserts its active characteristics into hello packet, which is broadcasted to all one-hop neighbours. If an active characteristic is internal, sequence number is increased by one.

Then, Weight Cost Function is used to simulate the weight lost in a link. This function should be an increasing function of weight because loss of a strong flow should not be smaller than that of a weak flow. In this project, we select

$$c(w) = e^{\frac{w}{\tau}} + const \qquad (1)$$

to describe the weight cost. τ and *const* are parameters for tuning the result. Further more, we can use Taylor Expansion Equation to simplify the formula:

$$c(w) = e^{\frac{w}{\tau}} + const' = 1 + \frac{w}{\tau} + \frac{w^2}{2\tau^2} + const \qquad (2)$$

Since w is between 0 and 100 in this project, and we want the cost in a link for any flow is approximately between 0 and 10, τ is set as 40 and the value of *const* is discussed later.

An active characteristic that is fused from multiple flows contains more reliability and certainty than those come from a single flow. Such active characteristics need to be emphasized so that further route selection can be biased to more reliable links. Weight Compensation Function takes number of advertisement of a characteristic and increases weights of such active characteristics before broadcasting hello packets. So the Compensation function is

$$c'(n) \tag{3}$$

2) Characteristic update: On receiving hello packet from neighbours, a node performs validation on both incoming and local characteristics. To prevent adverse flow which forms a "loop" and may result in "Count to Infinite" problem, incoming characteristics are validated on not only sequence number but also weights. A maximum discrepancy, *wd*, for weight validation is used. Because there can be multiple entries for the same characteristic in the local table, no duplication detection is needed. Then entries are either updated or inserted in local table.

After updating the local characteristic table, active characteristics are computed from corresponding entries in local table. This computation includes both sequence number and weight. The sequence number of each active characteristic is selected as the largest sequence number among all instances. The weight of each active characteristic is computed from all instances using Weight Fusion Function. For an intermediate node where n flows meet, each associated with a weight w_i, the Weight Fusion Function is

$$F(W_n) = F(w_1, w_2, ..., w_n) = \frac{\sum w_i - \sum c(w_i)}{n} \tag{4}$$

The rest of the procedure is quite similar as stated in basic design. The following flow chart shows the detail of characteristic update.

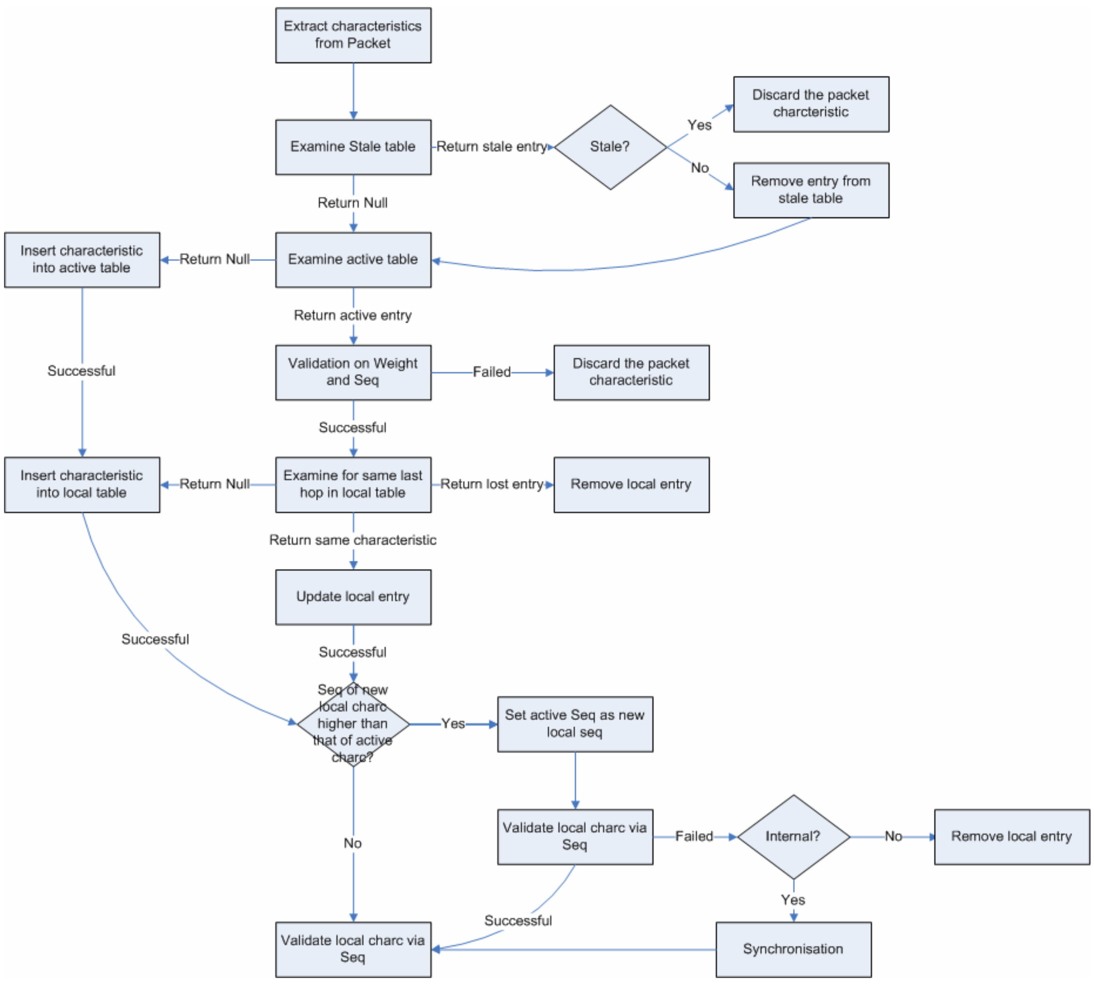

Extract characteristics from Packet
Examine Stale table
Return stale entry
Stale?
Yes
No
Discard the packet charcteristic
Remove entry from stale table
Return Null
Insert characteristic into active table
Return Null
Examine active table
Return active entry
Validation on Weight and Seq
Failed
Discard the packet characteristic
Successful
Successful
Insert characteristic into local table
Return Null
Examine for same last hop in local table
Return lost entry
Remove local entry
Return same characteristic
Update local entry
Successful
Successful
Seq of new local charc higher than that of active charc?
Yes
Set active Seq as new local seq
Validate local charc via Seq
Failed
Internal?
No
Remove local entry
Yes
No
Successful
Validate local charc via Seq
Synchronisation

Figure 3-13: Characteristic update in advanced design

3) Characteristic request and reply: In previous design, local characteristic table contains only the optimal instance of a known characteristic; while in advanced design, all valid instances of each know characteristic are kept for route selection. In other words, there has to be a criterion to select routes to handle request. Several design options have to be made given a context.

The first design option is: If there are more than one advertisements of a request, should single or multiple next-hop nodes be selected to handle the request?

The more instances probed, the higher possibility that one of them leads to a real source of the flow. It is straightforward that if a node forwards request to multiple neighbours, the success rate of request will be higher than single forward. On the other hand, more control overhead is introduced in multiple forward of request. The more important a request is, the more next hops should be selected. For a trivial general request single forward is highly recommended. In this project, we select single forward to implement.

Another question raised is: How to select the next-hop neighbours? A simple way is to select the routes associated with highest weight. But to distribute the workload on each link, and increase the robustness of the protocol, we use a probability based route selection which also appears in other routing protocols.

Given n instances of a requested characteristic, of which instance j is associated with weight, the possibility of selecting route i as next hop is

$$P(w_i) = \frac{(w_i + K)^F}{\sum_{1 \le j \le n} (w_j + K)^F} \qquad (5)$$

, where K and F are parameters to tune the shape of distribution. In this project, we select K = 0 and F = 2.

4) Purge stale characteristic: A timer is set up for each entry in the local table. On receiving characteristics carried in hello packet, corresponding entry is reset. If a timer expires, the entry it associated with is removed from local table. If an entry in the active table loses its entire links to local table, which means there is no incoming flow anymore, this entry is removed from active table and is inserted into stale table.

3.4.5 Flow Computation

After an initialization period, characteristic flows should reach a stable state in an ad hoc network whose topology is fixed. This requirement stems from the fact that in reality, if the environment does not change, the river course stays where it is. The meaning of this is twofold: Variable flow in a network brings uncertainty of control information distribution; frequent table operation add extra computation load to ad hoc devices which possess scarce resources.

Flows in a network are affected by weight functions and weight discrepancy for validation as stated above. All possible situations where characteristic flows are stable can be concluded as the following scenario:

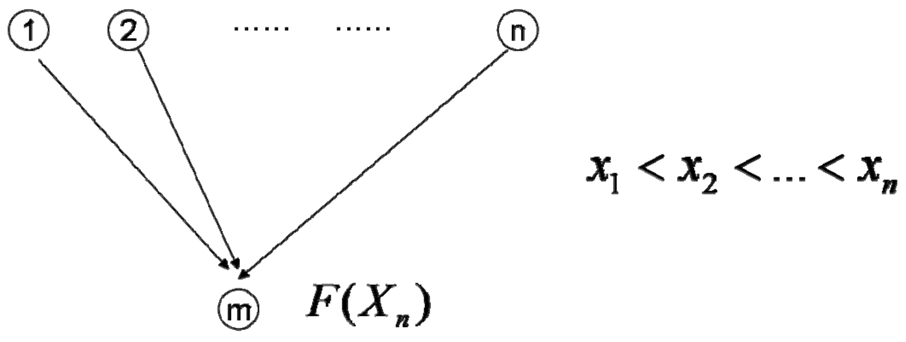


Figure 3-14: Stable Flow Computation

Assume that node 1...n each associated with weight x_1, $x_2 \ldots x_n$. In an intermediate node where these flows meet, the weight is $F(X_n)$. If all the flows are stable, each should be able to enter the intermediate node again. So we have:

$$x_1 - c(x_1) + wd \geq F(X_n) \tag{6}$$

Given the rest of the flows, assume the smallest weight that flow 1 can have is x, which means if flow 1 has weight less than x, it will not be able to enter the intermediate node again. Thus the flow is not stable. Weight x should satisfy that:

$$x - c(x) + wd = F(X_n) \tag{7}$$

On the other hand, flow from node m should not be able to enter any of nodes 1...n and forms an adverse current. So we have:

$$F(X_n) - c(F(X_n)) + wd + c'(n) < x \leq x_1 \tag{8}$$

Combine with previous condition, the requirement of flow function and wd are: The following system of inequation is true.

$$\begin{cases} F(X_n) - c(F(X_n)) + wd + c'(n) < x \\ x - c(x) + wd = F(X_n) \end{cases}$$

Change the form of the system, we have

$$-c(F(X_n)) + wd + c'(n) < c(x) - wd \tag{9}$$

Then

$$wd < \frac{c(F(X_n)) + c(x) - c'(n)}{2} \tag{10}$$

According to equation (7), (10) can be changed into

$$wd < \frac{c(x - c(x) + wd) + c(x) - c'(n)}{2} \tag{11}$$

Given the definition of $c(w)$, it is an increasing function. Now, consider function R(w) as follows:

$$R(w) = w - c(w) = w - \frac{w}{\tau} - \frac{w^2}{2\tau^2} - const' \tag{12}$$

R(w) is also an increasing function if $w \in [0,100]$ and $\tau = 40$, because its first derivative:

$$R'(w) = 1 - \frac{1}{\tau} - \frac{w}{\tau^2} > 0 \text{ for } \forall w \in [0,100], \tau \in [11,+\infty)$$

That means $c(x - c(x) + wd) + c(x)$ is increasing. Inequation (11) has to be true for any possible x and all possible n. We now examine the minimum value of x and maximum value for n. For $x = 0, c'(n) = c'_{\max}$, inequation (11) becomes

$$1 + const > wd + \frac{c'_{\max}}{2} - \frac{wd}{2\tau} - \frac{wd^2}{4\tau^2} \tag{13}$$

In other words, form (13) is an equivalence of (6) and (8). It has to be true to guarantee flows in a network keep stable. An easy way is to select *const* in Cost Function as follows:

$$const = wd + \frac{c'_{\max}}{2} \tag{14}$$

, which makes Weight Cost Function as:

$$c(w) = 1 + \frac{w}{\tau} + \frac{w^2}{2\tau^2} + wd + \frac{c'_{\max}}{2} \tag{15}$$

Function (14) guarantees the stable flows in a network even though it is not the only form. We will show how these functions are implemented in next Chapter and analysis how to select optimal value for each parameter in Chapter 5.

Chapter 4: Implementation

We choose OPNET modeler to implement and evaluate our protocol. In this chapter, we will discuss the details of the implementation. Starting with general structures and node model, we focus on the core implementation of the project: ABC module in the network layer. Data structure and the operation on them affect the functionality and performance of a process. State Transition Diagram (STD), representing state machine of ABC process model, describes the key functionality of our protocol. And finally, interfaces need to be defined between each domain and each module.

4.1 Overview

To perform a realistic simulation, our implementation follows OSI model. We build ABC upon a physical layer and a data link layer. The physical layer includes a radio transmitter module and a radio receiver module. At the MAC layer, we use the IEEE 802.11 module, provided as a standard model by OPNET. To communicate with the routing layer, an interface layer is inserted on top of this standard MAC module. Together, they form a data link layer and provide an OPNET standard point to point communication for the ABC routing protocol. Built on this stack, the implementation of our protocol incorporates the accuracy of OPNET modeler in the first place. The standard source module and sink module are modified to simulate the behaviour of the transport layer and applications above. The node model of our implementation is shown in Figure 4-1.

The ABC network layer module provides the core functionality of the routing procedure. It is set up initially with predefined internal characteristics and other preferences from Network domain. Then it receives data/request packet from source module periodically and generates reply back to sink module after the routing procedure. We have implemented separate routing modules according to both basic design and advanced design. They share similar interfaces and operate in the same stack. In this way, we are able set up scenarios to evaluate them against each other.

We will focus here on the implementation of the advanced design. To some extend, implementations for basic design and advanced design share some similarity. In next chapter, we will explain these similarities.

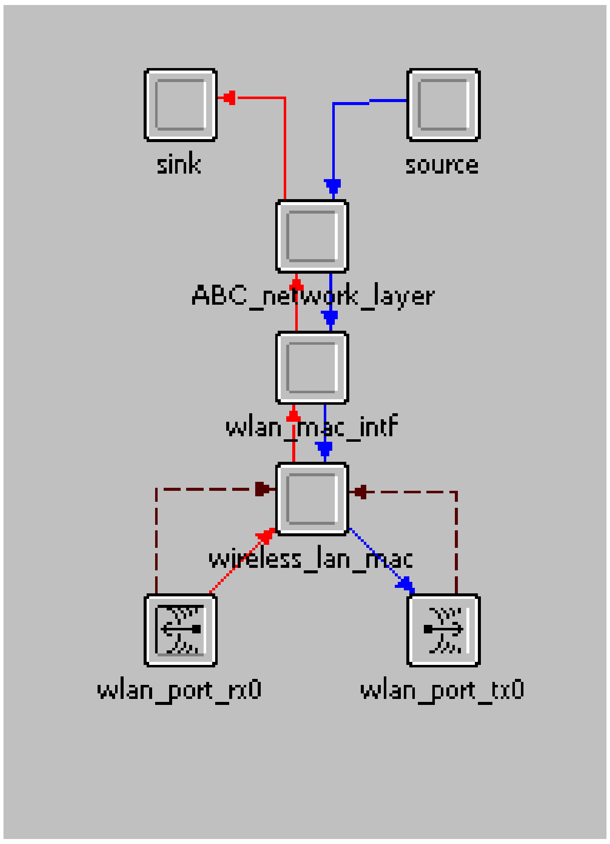


Figure 4-1: Node model of ABC

4.2 Data Structure

In this section, we discuss the implementations of characteristic tables and packets of our protocol.

4.2.1 Tables in ABC

As mentioned in previous chapter, ABC contains a requests table and three characteristic tables, including local table, active table and stale table. One feasible way to implement these tables is to create a compound attribute for each table, of which sub-attributes represent fields of the table. However, operations on a compound attribute are not flexible enough and are usually used in static routing protocol. We use a global list variable to represent a characteristic table. All elements in a list are of the same struct type.

1. The following code is the structure defined for stale table. It indicates two fields in stale table: Characteristic and sequence number.

```
typedef struct stale_charc_entry
  {
  int charc;
  int seq;
  } *lp_stale_charc;
```

2. The following structure definition is for the local characteristic table. Except for the fields we have shown in the design, we need some extra fields for the purpose of efficient implementation. The “related_act_charc” field stores the pointer to the entry in active table it associated with. To set up a timer for each entry in local table requires much computation overhead. The “status” field provides a trade-off between functionality and performance of protocol. We only implement one timer for all entries in local table. When the timer expires, all entries in local table are examined. If the “status” is false, set it as true, otherwise remove the entry. On receiving an update of the entry through hello packet, “status” is reset. In this way, given a timer duration t, the actual time to live for a local entry floats in [t, 2t]. The “int_charc” field represent whether a local characteristic is internal.

```
typedef struct local_charc_entry
  {
  int charc;
  int weight;
  int next_hop;
  int seq;
  lp_act_charc related_act_charc;
  Boolean status;
  Boolean int_charc;
  } *lp_local_charc;
```

3. The following definition is the structure for active table. Similar as the local table, entries in active table also store associated local entries. However, since the relation between active characteristic and local characteristics is one-to-many, we use a list store pointers of all related local entries.

```
typedef struct act_charc_entry
  {
  int charc;
  int weight;
  int seq;
  List* original_charc;
  } *lp_act_charc;
```

4. The following show the definition of request table entry. The field “charc” is the requested characteristic and “addr” records the last-hop address of the request. And

“status” is also used here to purge stale request. The timers for request table and stale characteristic table have separate value.

```
typedef struct request_entry
  {
  int charc;
  int addr;
  Boolean status;
  } *lp_request_entry;
```

Figure 4-2 shows the operations on characteristic tables. Receiving hello packet from neighbours, a node first updates the local table. And then active table is computed from local table, lost characteristics are removed to stale table. Finally, information in active table is inserted into hello packet and broadcast to its neighbours.

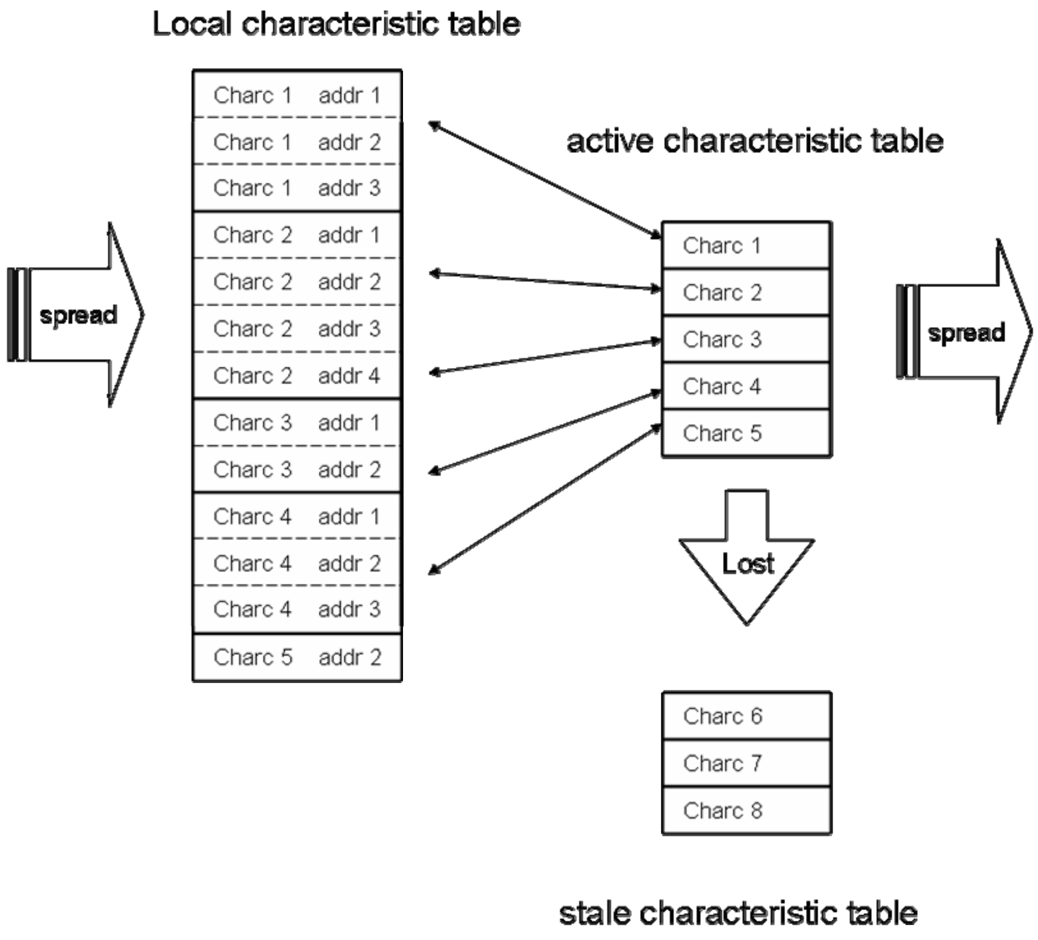


Figure 4-2: Characteristic table operation

The link between local table and active table is bidirectional: Each contains pointers to the other. In original_charc list of an active entry, the pointers of the local entries associated are sorted based on sequence number of local characteristics. This pattern benefits in two aspects: Running time for searching an element in a sorted list with n element is only O(lgn). Compared to simple searching approach, it is time efficient; The sequence number of an active characteristic is set as the highest sequence number among all local entries. This approach simplifies the operations on tables. The maintenance of the sorted list is easy. We use an improved insertion sort approach for new entries. Running time to insert a new entry is also O(lgn).

```
int low = 0;
int high = size of sorted list – 1;
int mid;
lp_local_charc local_charc;
while(low<=high)
{
mid = (low+high)/2;
local_charc = list_access(mid);
if (new_charc->seq > local_charc->seq)
  {low = mid+1;}
else if(new_charc->seq < local_charc->seq)
  {high = mid-1;}
else
  {
   insert new_charc to position mid of the list
   return;
  }
}
Insert new_charc to position low+1 of the list
```

4.2.2 ABC Packets

OPNET provides Packet package to implement packet format. In ABC, RREQ packet and RREP packets have fixed length. We can simply use op_pk_nfd_set() and op_pk_nfd_get() for operations on named fields in those packets. HELLO packet, on the other hand, contains variable number of characteristic information. A dynamic length field is needed.

We define a structure as follows to represent the content format of characteristic flows. Using dynamic memory allocation API op_prg_mem_alloc(), we allocate a memory block for an array of pkt_charc_entry, which is retrieved from the active table. The array is then inserted into a HELLO packet using op_prg_nfd_set_ptr(). However, this API uses an “unnatural” pattern: It takes the address of the pointer of the structure as its parameter. Unawareness of this pattern may lead to data chaos at a receiving end. Because no memory of the structure is actually carried in a HELLO packet, the size of this variable field is 0 bit. We then use op_prg_bulk_size_set() to set the overall size of HELLO packet.

```
typedef struct pkt_charc_entry
 {
 int charc;
 int weight;
 int seq;
 } *lp_pkt_charc;
```

4.3 State Transition of ABC

As we mentioned above, a process in OPNET is described by a finite state machine. The following diagram shows the state transition of ABC. In this diagram, a red state is unforced and green state is forced. Leaving an unforced state is driven by a certain interrupt, while a process leaves a forced state as soon as execution of the state is complete.

The ABC process begins at state “init”. Model attributes are loaded and all state variables are initialized inside this state. The process then waits and receives two self interrupts to enter normal state “idle”. The reason for waiting two self interrupts is that OPNET executes parallel interrupts for modules in the same node model. The underlying 802.11 MAC process requires two interrupts for initialization. ABC network process has to wait for the MAC module to be prepared before sending HELLO packets to it. After two self interrupts, the ABC process has now synchronized itself with all other modules in the node model and enters the “idle” state.

Before entering “idle” state, three timers are registered: Characteristic spreading timer, stale characteristic timer and stale request timer. A timer is in fact a self interrupt, which is scheduled at a specified simulation time. When a timer expires, the corresponding forced state is entered. After execution, timer is reset and starts again. The process comes back to “idle” state.

When a packet is received from the application layer, the ABC process enters the “send packets” state. In the execution of this state, data or request packets are composed and sent to the next-hop node whose address is retrieved from the local table.

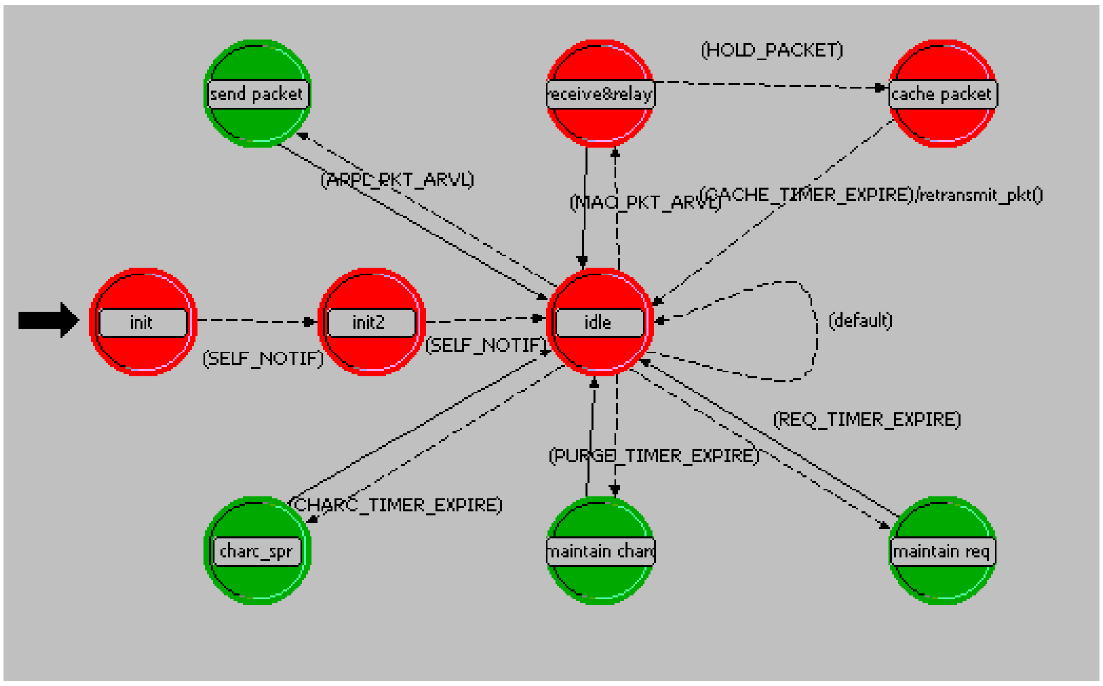


Figure 4-3: STD of ABC routing process

When a packet is received from the MAC layer, the ABC process enters the “receive & relay” state. It first examines the type of the packet. If the packet is a HELLO packet, local table and active table are then updated accordingly. If it is a RREQ packet, the process extracts the characteristic requested. If destined characteristic is an internal one of this node, the process generates and sends out RREP packet. Otherwise, the node will lookup the local table and will compute the next hop to which to forward this packet. If no entry is returned from the local table, a self interrupt is invoked and the process enters the forced state “cache packet”. A timer is registered in the state. When the timer expires, the process tries to retransmit the packet.

4.4 Implementation of Key Functions

We will show briefly how the functions that control the flow are implemented in our project. These functions are not very complex but they are the key to ABC’s performance.

- Weight Cost Function: It adds computation overheads if we actually compute the cost of a given weight. For a mobile ad hoc device, especially a sensor, e^w is not an easy workload. Computing $c(w)=1+\frac{w}{\tau}+\frac{w^2}{2\tau^2}+wd+2c'_{\max}$, on the other hand, is acceptable. However, in this project, we simply use an “if else” clause to assign a predefined integer value to a cost on an inputting weight.

- Weight Compensate Function: Similar with Weight Cost Function, we use “if else” to achieve this function. The only considerations for this function are the return value should not higher than $c'_{\max}$ as we computed before and should be an increasing function. We leave the optimal form of this function for further development.

- Characteristic Abstraction: This function execute in basic design. As explained in Chapter 3, characteristic is abstracted into a higher level when it reaches the edge of propagation. Because of the encoding scheme we selected, a characteristic is easily abstracted by shifting the binary code left by 1 bit.

- Relation Determination: In basic design, it is important to determine the relationship between two characteristics. Similar with previous function, we leverage bit operations here. For a characteristic code, the bit length of the code indicates the level of the node representing the characteristic in current scheme. The decimal value of the code indicates its position in its level. Using Abstraction function based on position in encoding tree, we can determine whether two characteristics have sibling or dominance relations.

- Table operations: Entries in the local characteristic table and entries in the active characteristic table are linked to each other. Further more, an active entry keeps the pointers of its related local entries in a list. The list is sorted based on sequence number. Table operations we have implemented use a “divide and conquer” approach to reduce the running time.

- Request Holding: When failed to send out a request, a node hold the request for a time to wait destination information to appear again. To simplify the implementation, the request holding time is shorter than the characteristic holding time. In this way, new characteristic information that comes during a request is held will not disappear when the request timer expires. This implies that a node does not have to retransmit a holding request before its timer expires.

4.5 Interfaces

We have defined the following interfaces in the ABC routing process communicate and cooperate with other models:

- Packet streams
- ICIs
- Model interface
- Statistics collection
- Built-in trace code

Packet streams transmit packets between modules linked. They are unidirectional and are specified in Node domain. They are divided into outgoing streams and incoming streams. Each stream is assigned an index. Kernel API op_pk_send() uses this index to send out packet to module linked by the stream. Incoming packet causes an interrupt upon which STD receives and reacts.

As mentioned above, ICIs are used to pass control information between modules. ABC installs the destination MAC address on its ICI, and associates with outgoing data stream to MAC module.

Packet streams and ICIs are interfaces between ABC and modules in the same node model. Model interface is really cross-domain interface. Model attributes of ABC include internal characteristics and other predefined parameters. These attributes are configured as model interface, and are passed to Node domain and Network domain. We can simple assign protocol preference in a scenario before the execution. The routing process then reads these attributes during process initialization.

Built-in trace code in OPNET provides examination on the value of process variables trace throughout the simulation. In ABC we insert trace code block wrapped by kernel API op_prg_odb_ltrace_active for values of interest. There are mainly three types of value we have traced in the project: Content of three characteristic tables and request table for each node; Result of all characteristic flow functions to see how flows are computed; Content of control packets to see how flows disseminate. See the following table for the traced values. We can "watch" these values discretely in OPNET console during simulation using "ltrace + name" command. "Discretely" here means the values are showed when there is an action operating on them. Some trace blocks are used for self-test, that is submitting a warning if an unexpected value occurs during simulation.

Trace name	**Value of interest**
local charc	Full content of local characteristic table at each node.
active charc	Full content of active table at each node
lost charc	Entries lost in both local table and active table, indicating also the reason of the loss: Timeout or lost from source
self-test	Perform self testing on range of each value and fault actions, such as weight value and loss of internal characteristics
seq syc	Shows the detail of sequence number synchronization: Node, characteristic, external sequence value, and local sequence value.
flow	Content of HELLO packet in a network: Source node, characteristics carried, weights, and sequence number. Also display all nodes that receive this packet.
*charc abstraction**	Show the detail of the characteristic abstraction.

Table 4-1: Trace block in ABC

* This is trace block for basic design

In a similar way, statistics are record during simulation. Statistics are set as model attributes OPNET. And a global variable, termed as state variable, is used to store or account certain values. Then kernel API op_stat_write() is used to write a statistics variable into corresponding statistics model attribute. After simulation, the selected statistics can be shown and analysed.

Chapter 5: Evaluation

In this chapter, we will evaluate ABC from two aspects: Features and performance. First we will discuss the features of ABC. Then we present evaluations of the performance of ABC. At the end of this chapter, we will give examples of possible ABC applications according to its features and performance.

5.1 Features of ABC

Based on the discussion of the design we have given in previous chapters, we can conclude the features of ABC as follows:

- Characteristic-based routing: ABC replaces IP addresses used in traditional routing protocols with characteristics. A destination is described by a group of characteristics. Packets are routed toward strong gradient of a specified characteristic.

- Light weight: Compared with other data-oriented communication systems, ABC is implemented at network layer. It doses not have to resort to other routing protocols and is independent from IP. The computation load and control overhead are reduced. Therefore, light weight is an intrinsic advantage of ABC. This can also be observed from control overhead and latency of communication, which we will show next section.

- Moderate description: Currently, one of the research trends in service discovery and content-based communications is to describe and to filter information flow in a more semantic pattern. These approaches offer powerful application-oriented information description, yet produce more overhead. ABC provides 32-bit encoding of characteristics. Together with the abstraction method in basic design and the weight associated in the advanced design, ABC supports a moderate characteristic description.

- Fully distributed: ABC is fully distributed. Unlike service discovery and content-based networks, no broker is needed. A node can join, leave or fail without affecting other nodes.

- Anycast/multicast: If the protocol is configured to forward requests to only one of the possible next hops, the protocol will behave like an “anycast” protocol; if we configure nodes in ABC to forward request to all possible next hops, a multicast is then achieved. However, as discussed in chapter 3, a node can be configured to forward its neighbours’ requests to certain number of next hops. In this case, the routing in ABC is between anycast and multicast: Multiple but not all source nodes of a characteristic can receive packets destined for this characteristic.

- Scalability: The analogy of color stream propagation and fusion introduces good scalability to ABC. Two functions actually affect the scalability: A characteristic becomes blurring during propagation; multiple flows can merge at an intermediate node. They function together, but in detail the first determines the scalability on number of nodes and the second one determines the scalability on number of characteristic sources in a network.

  - Characteristic abstraction in basic design and weight cost function in advance design fade off information flows. The former function takes hop limit as parameter, which determines the fading ratio of a characteristic. τ and *const* in the latter function have a similar role. While hop limit simply affecting fading ratio, we can also configure τ and const to change the shape of cost curve. Given a certain cost value we want to assign for maximum weight, in the case of this project it is 100, if both τ and *const* are large, the cost varies little for all possible weights; if both τ and *const* are small, the cost is much less for a small weight. This means that a data flow can fading off fast at a small area and then disseminate into a large area with small weight, which can easily be taken over by a strong flow at remote end.

  - Characteristic fusion, together with sequence number synchronization, affects the scalability on number of flows in a network. For a characteristic, number of source nodes in a network does not affect control overhead. Each node exports only one flow of a characteristic group in basic design and one flow of a characteristic in advanced design.

- Request/Reply: The current version of ABC supports a request/reply communication model. It means that no communication session is supported. Both RREQ and RREP packets can carry data or instructions.

5.1.1 Comparison to Data-oriented protocols

Service discovery, content-based routing protocols and ABC share some similarities in the problems they address while using different approaches. The following table shows a comparison of the differences between them.

	Service discovery	Content-based routing	ABC
Implementation level	Middleware layer	Middleware layer	Network layer
Underlying Network protocol	IP	None or IP multicast routing	-
Communication type	Unicast	Multicast	Anycast/Multicast
Communication model	Client/Server	Publisher/Subscriber	-
Detail of data description*	High	Medium / High	Medium

Table 5-1: Comparison of data-oriented protocols

- ❖ Service discovery uses service description from server which is usually semantic and specified by application layer. Content-based routing protocol uses subscriptions from a subscriber to filter data flow. It is also specified by application layers and can be semantic. ABC is at network layer, and uses encoded characteristic which is supposed to be similar length as IP address.

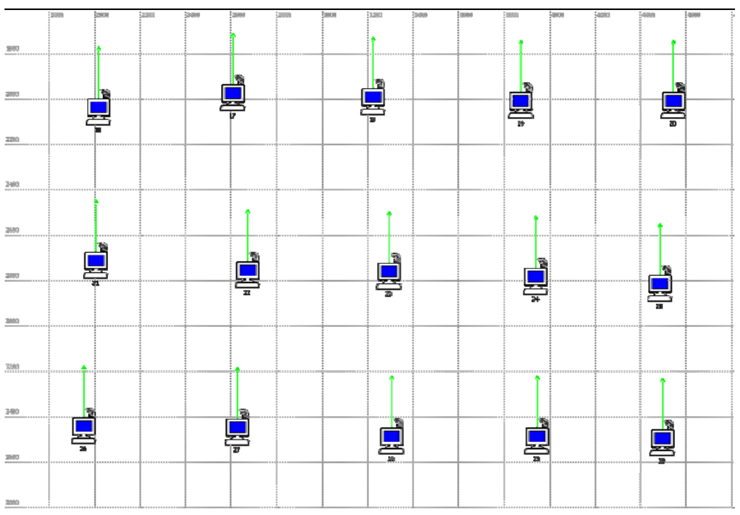
5.2 Numerical Evaluation

In this section, we will analyze the performance of our protocol. First we will state value of our interest and design the experiments accordingly. In second sub-section, we discuss the data that is collected in each experiment.

5.2.1 Experiments Design

The objective of the experiments we have performed is to demonstrate the feasibility of our protocol. The values of interest include control overhead, latency, and success rate of request. Control overhead is defined as the data throughput at when there is no data delivered from application layer. The success rate is the percentage of successful requests among all requests generated in a network.

All the experiments were performed using two topologies, as shown in Figure 5-1. The distance between 2 orthogonal neighbours is between 600m to 700m. The distance between diagonal neighbours is around 900m which is approximately the edge of transmission for the ad hoc devices we configured for this experiment. A small movement may result in the loss of connectivity to diagonal neighbours, and a large movement may result in loss of connectivity to any neighbours.


(a) Mesh with 15 nodes

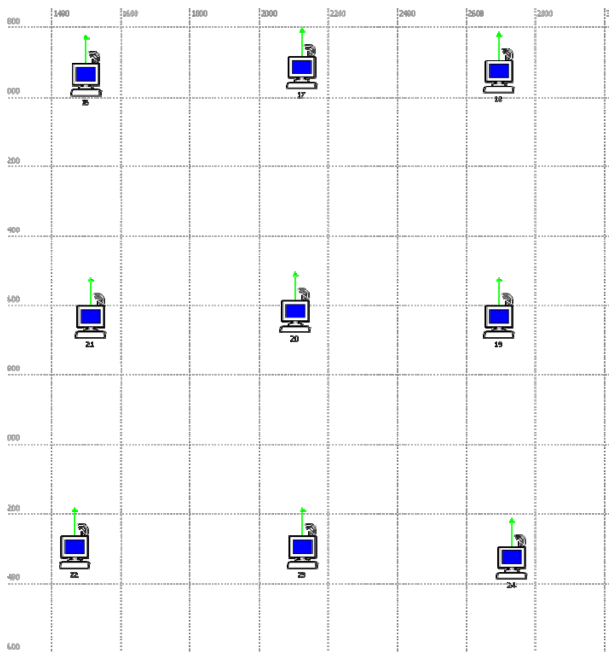

(b) Mesh with 9 nodes

Figure 5-1: Topology used in experiments

We use IEEE 802.11g as underlying MAC protocol. The transmission speed is configured as 54 Mbps and the transmission power is set as 5 mW. This configuration leads to an 800m to 1000m transmission range for ad hoc devices. The hello interval is randomly selected from 1.0s to 3.0s, so that the transmission collision can be reduced. Each node in the network initiates a request to a random characteristic every 10 seconds. The fixed parameters for the simulations are given below:

Parameter	**Value**
Transmission Power	0.005w
τ	40
Characteristic Holding Time	10.0s
Hello Interval	uniform(1.0s, 3.0s)
Request Frequency	6 requests/ min
wd	3
sd	10
$C'_{\max}$	4

Table 5-2: Protocol configuration

In this project, we focus on the demonstration of ABC routing procedure. Therefore, we apply random mobility model to all the nodes. Three types of mobility are examined: Static, medium mobility and high mobility. The medium mobility here is set randomly between 0 to 10 m/s. And the high mobility here is set randomly between 0 to 20 m/s. For the evaluation of scalability, we compare control overhead in different number of nodes and number of characteristics.

Note that the duration for which a node keeps a request affects the success rate and latency as well. We examine how they are balanced according to request holding time. Also we evaluate how stable flows affect the performance by assigning value to Cost Function which can not satisfy system (7) and (8).

Dimension	**Value of interest**
Mobility	Static , uniform(0,10), uniform(0,20)
Request Holding Time	0.1s, 1.0s, 10s
Number of Nodes	6, 8, 15
Stable Flow	TRUE, FALSE
Design Version	Basic, advanced

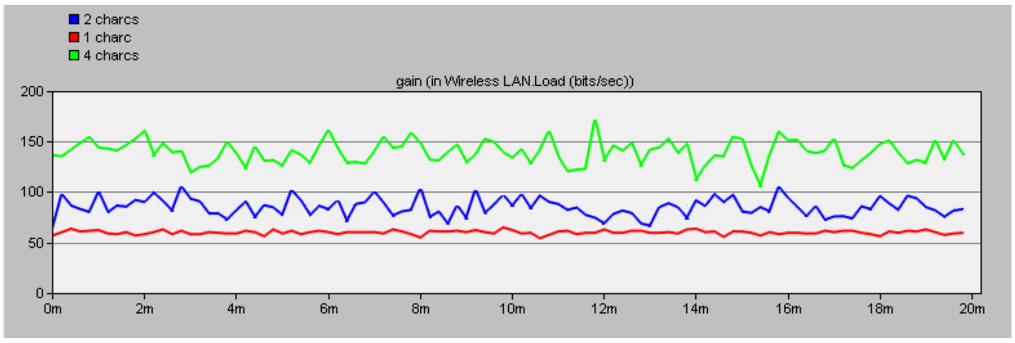
Table 5-3: Evaluation Matrix

5.2.2 Performance Analysis

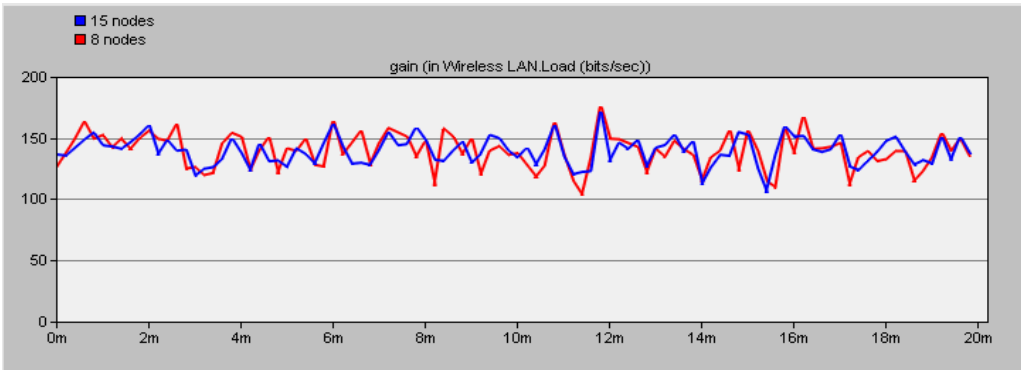
At first of the numerical evaluation, we perform a control experiment where all nodes are static. The success rate for characteristic requests is always 100% in this scenario, which shows the correctness of ABC at a static and reachable network. Then we evaluate the performance of ABC as follows:

Scalability

We have presented a formal analysis of scalability earlier in this chapter. To support this analysis, we examine a mesh topology with 15 nodes and a mesh topology with 9 nodes. Each node has one internal characteristic initially. Figure 5-2 shows that the control overhead increases according to the number of characteristics in a network, but is independent from the number of nodes. Further more, since each node generates a characteristic instance, Figure 5-2 (b) also illustrates that the control overhead is independent from the number of flows in a network.



(a) 15 nodes with different number of characteristics



(b) 4 characteristics with different number of nodes

Figure 5-2: Average control overhead

The control overhead results from HELLO packets used by each node. There are two factors affecting this control overhead: The size of HELLO packets and the frequency of HELLO packet transmission. The frequency of HELLO packet transmissions also determines the propagation speed of a characteristic flow.

Basic and Advanced Design

We have presented the shortcomings of the basic design in Chapter 3, and based on the analysis proposed an advanced design. The following figure shows the comparison of the advanced design and the basic design in a dynamic topology. The speed of each node in this experiment is 10 m/s. The comparison shows that while the success rate for the advanced design remains constant, the success rate for the basic design reduces heavily at initial stage. The advanced design excels the basic design through the 20 minutes simulation.

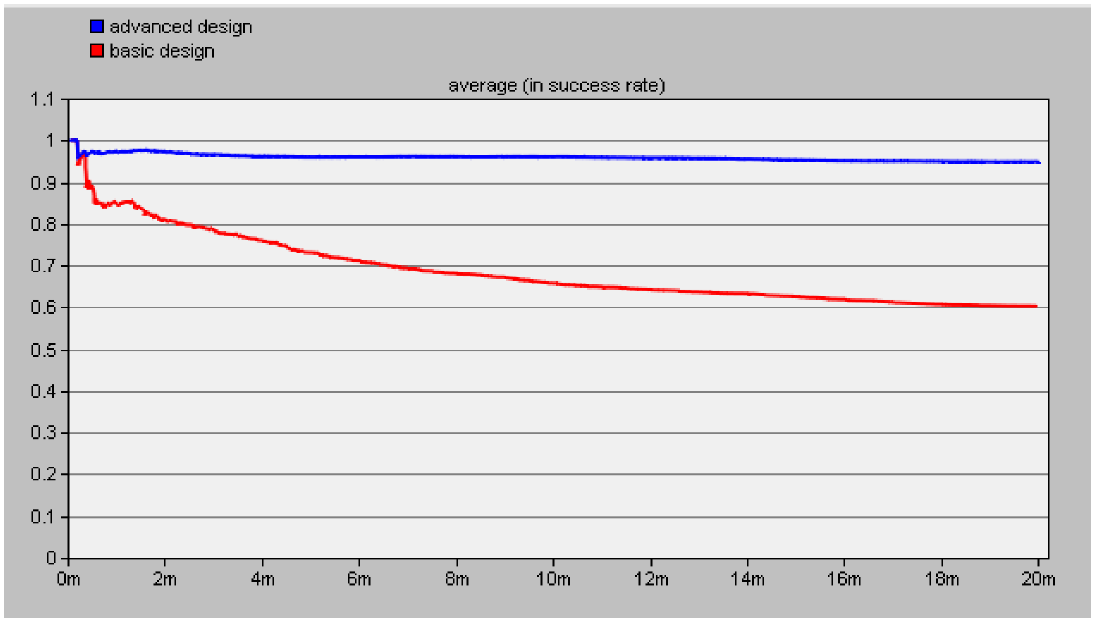
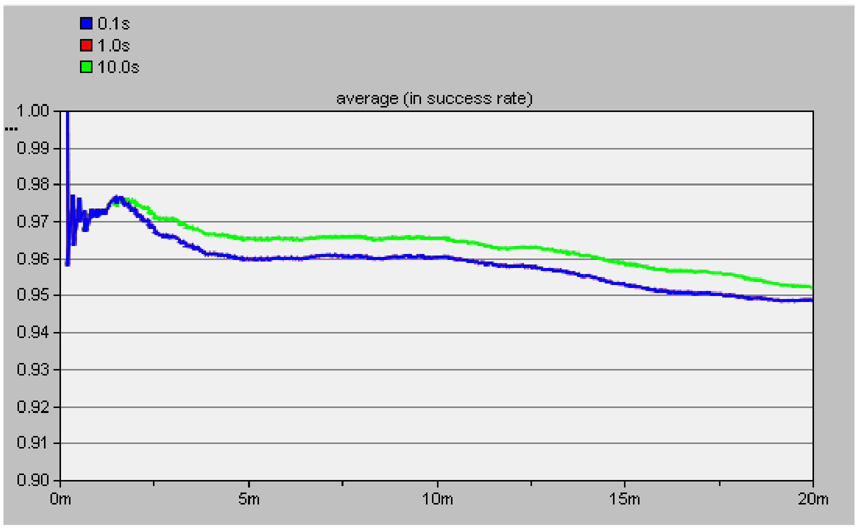


Figure 5-3: Advanced vs. basic design

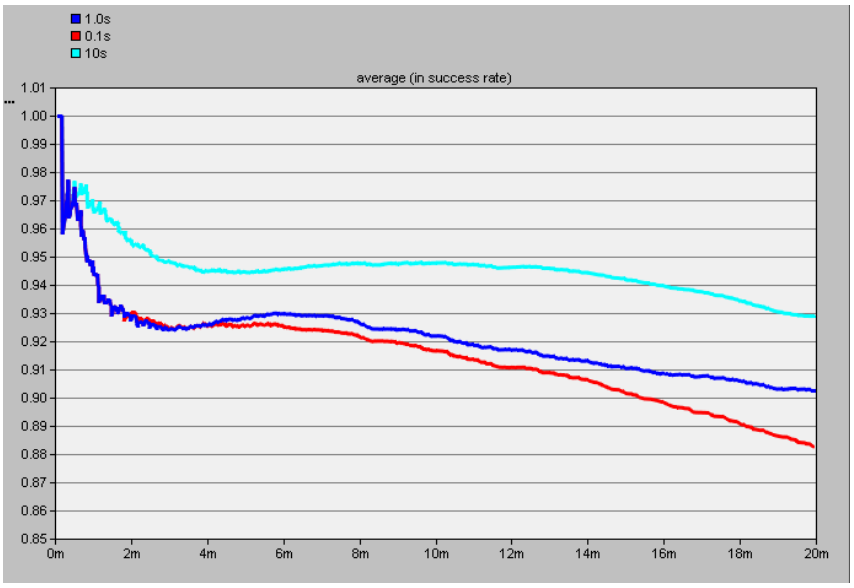
Success Rate and Latency

On receiving a request, either from its application layer or from its neighbour, a node examines local characteristic table for next hop to forward the request. If no valid entry found, the request is hold for a small period and then try to search for valid entry again before discard the request. If holding duration is long enough, ideally the node can retrieve information about the requested destination. Obviously, the Request Holding Time (RHT) affects latency of request severely.

In a network with medium mobility, we can see that holding a request for 10 second gains an additional 0.5% of success rate. No obvious difference is observed between 0.1 s and 1 s RHT. The reason for this is that in such a network, it takes more time to wait for lost destinations to appear again. A small holding time will not lead to a drastic improvement. The second experiment, which used a high mobility, produces a much different result. The success rates for all three RHT values are smaller than in the previous experiment. But compared with 0.1 s RHT, 1 s RHT increase the success rate by 2% and 10 s RHT increase it by 5%.



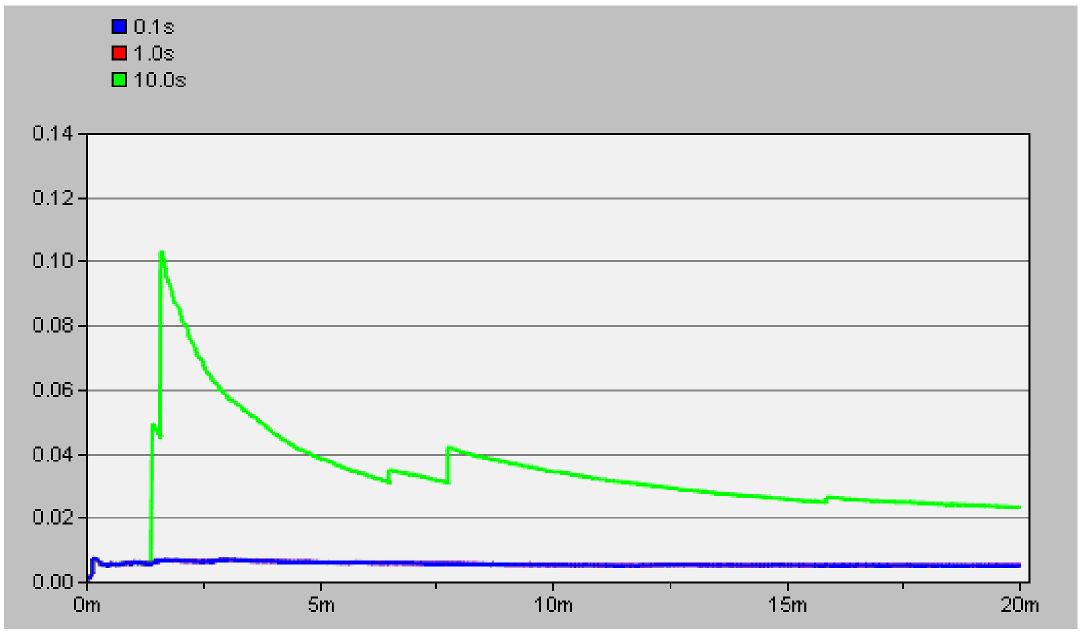
(a) Uniform 10 m/s



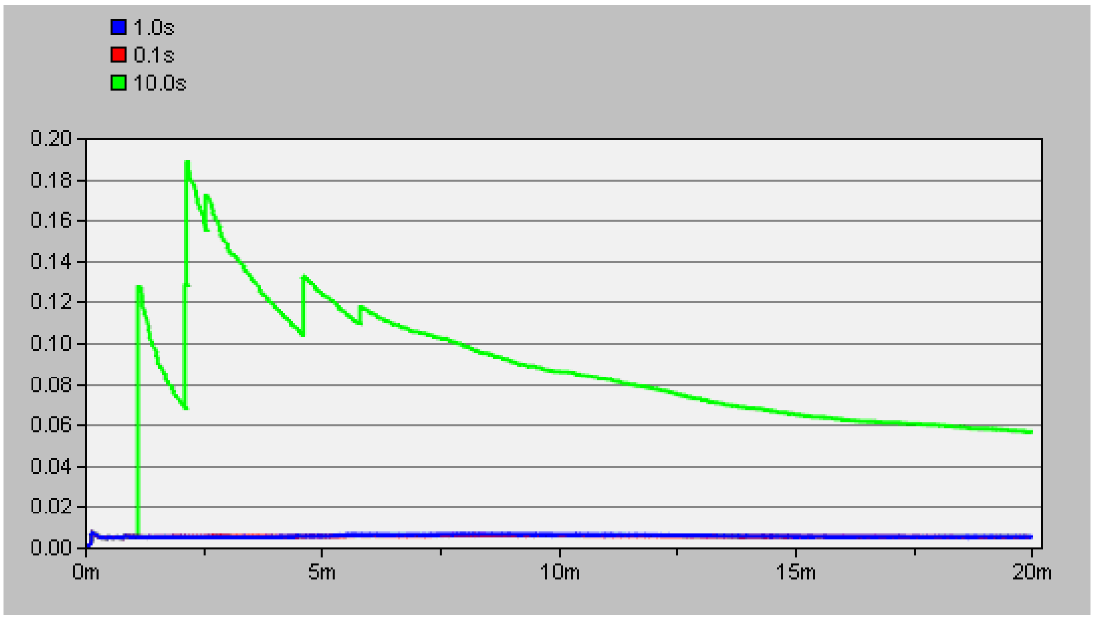
(b) Uniform 20 m/s

Figure 5-4: Success rate on RHT

Thus, increasing the RHT does not have a lot of influence on the success rate. The request latency, however, experiences a strong influence. For a 10 s RHT, the latency introduced is much higher than the others, and grows because of increasing mobility. On the other hand, in both mobility configurations, latencies for 0.1 s and 1 s RHT have no obvious distinction. The latency stays at same level - around 5 ms - when the mobility changes.



(a) Uniform 10m/s



(b) Uniform 20 m/s

Figure 5-5: Latency on RHT

According to this experiment, 1 s RHT is an advisable configuration. We use this value in all other experiments. However, depending on the requirement of application and the topology of a network, different RHT can be select to balance the reliability of transmission or the performance of the protocol.

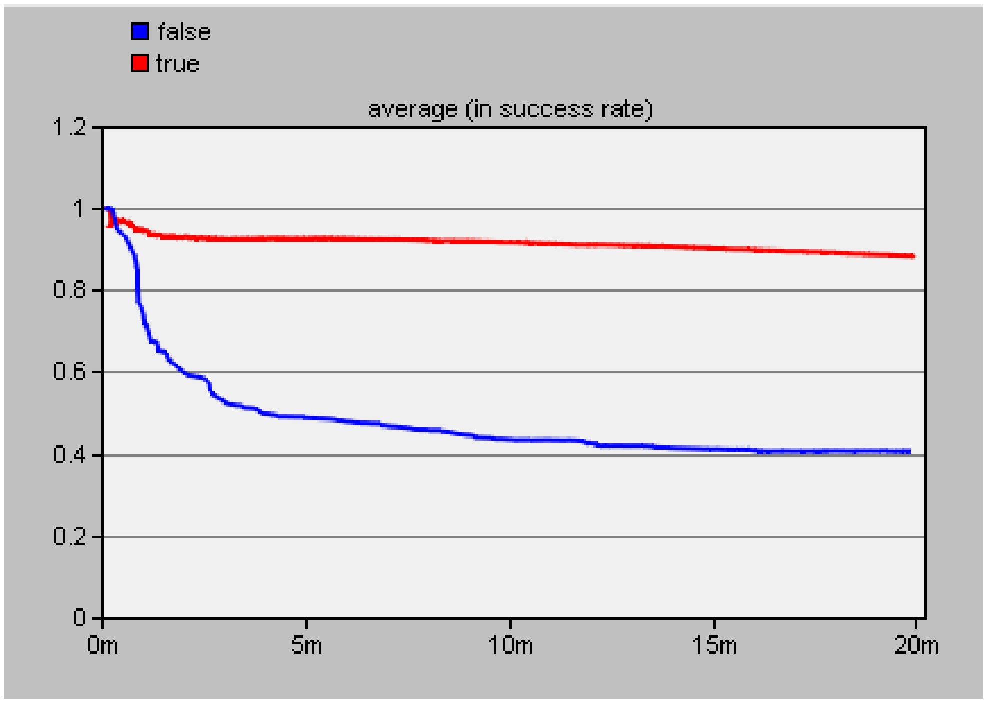
Stableness of Characteristic Flow

We have given a computation for the condition of stable flows. Based on the computation, we compose a Weight Cost Function that will satisfy requirement of stable flows. We would like to demonstrate here why the flow of characteristics in a network should be stable.

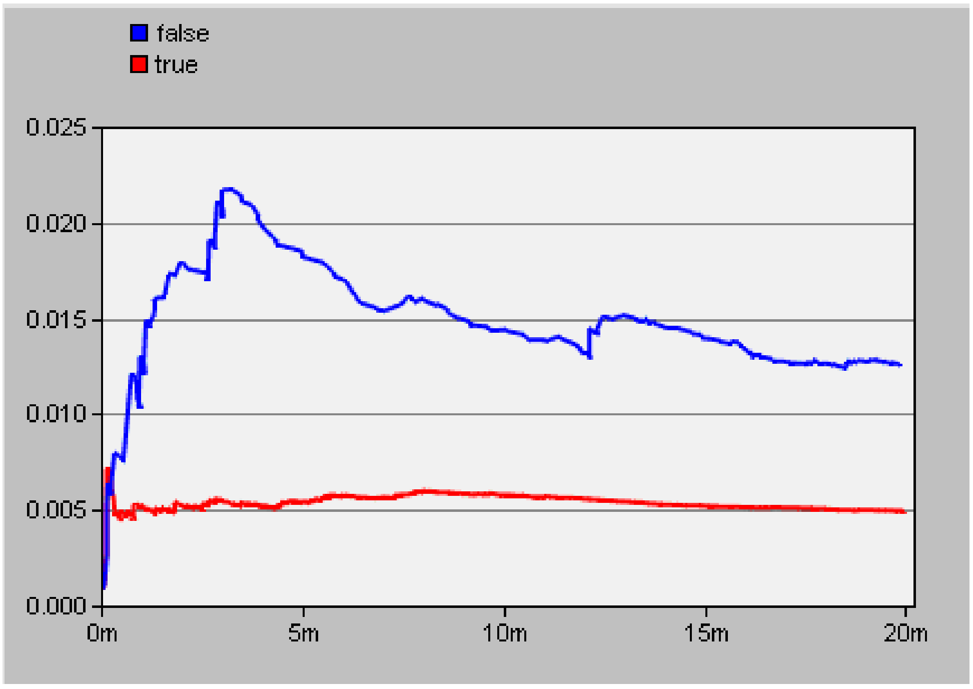
According to the color stream analogy of our design, a stream should not be able to flow backwards. In the advanced design, no hop count is used. If the stable condition formula (8) is not satisfied, a characteristic flow may form a loop, which represents incorrect route information. If formula (7) can not be satisfied, a valid characteristic flow may be able to enter an intermediate node, and lose it effects on a network.

Note that there is no unique way to satisfy stable condition. The form of Weight Cost Function we use in ABC is only one of correct approaches. To illustrate the effect of stableness, we compose a Weight Cost Function that can not make formula (7) and (8) true: Minus a const number 8. Again, this is not the unique way.

We use mesh topology with 15 nodes, 4 characteristics and medium mobility to evaluate unstable flow. From the figures above, we can see that a stable flow gains overwhelming advantage against unstable flow in both success rate and latency. Also active weights at each node changes frequently, much more computation overhead is introduced if stable condition can not be satisfied. In OPNET, 20 minute simulation for unstable flow generates 212277004 events while only 1081745 events generates for stable flow.



(a) Success rate



(b) Latency

Figure 5-6: Stable vs. unstable

5.3 Sample Scenarios

We have discussed the features of ABC and presented a performance evaluation of ABC. The next issue we would like to address are possible applications that can be based on ABC.

Previously, we have shown a DNS query application. Current version of ABC provides powerful and efficient routing for Request/Reply communication. In this section, we will show some other sample applications for ABC.

5.3.1 Sensor Query

Because of the light weight and anonymous nature, ABC suits sensor network applications. And it scales well in a network with large amount of nodes. Consider a sensor network, types of sensor, such as temperature or light, are encoded as different characteristics. Depending on the orientation, battery, computation ability, sensor accuracy or combination of them can be mapped into weight values. The Weight Cost Function can also be configured to suit the application. For example, in an energy efficient network, weight cost at an intermediate node can take battery capacity of the node into consideration. The only requirement for the Weight Cost Function is to satisfy the stable condition. A query for a certain sensor transmits through an optimal path to one or multiple optimal destinations. A sensor node can then follow instructions in the query to perform some actions, i.e. start monitoring, or insert data into RREP packet and send it back.

5.3.2 Battle Field Coordination

One of the most presented motivations of ad hoc networks research comes from military application. Usually ad hoc devices in the military field do not support IP addresses and application in such field requires a small latency. According to our evaluation, ABC satisfies such requirement.

Consider the following scenario: In a large battle field some troops are experiencing a severe fight with the enemy, while others remain relatively idle. A command may want to send information or instructions to either type of troops. The severity and idleness can be set as two characteristics. Weight indicates the level of such status. Packets can be forwarded to different types of troops along a path following the characteristics.

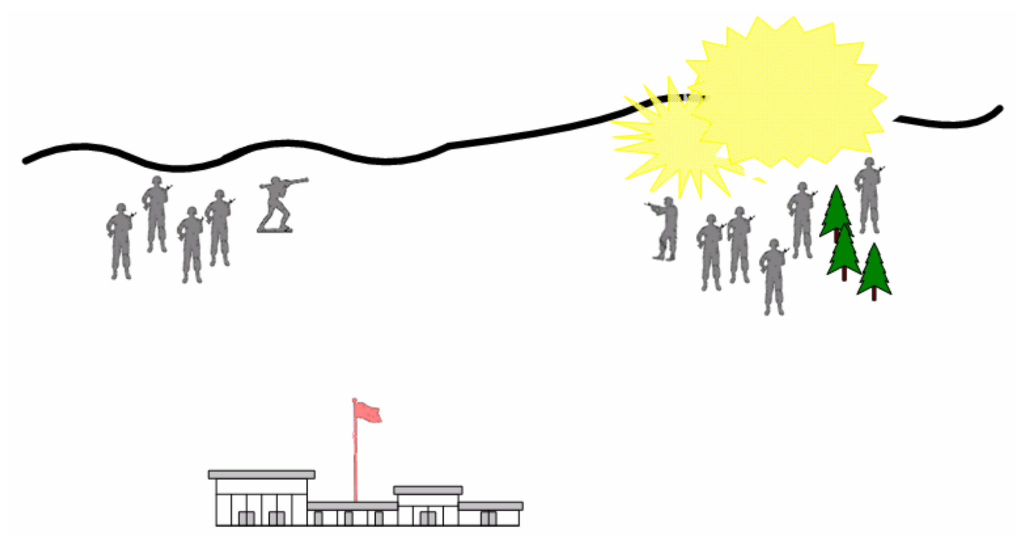


Figure 5-7: Battle field coordination

Chapter 6: Conclusions

6.1 Contributions

In this project, a novel ad hoc routing protocol called ABC has been proposed. To analyse the feasibility and performance of ABC, we have implemented and simulated our protocol in OPNET modeler. ABC is based on the characteristics of nodes, and the use of IP address has been avoided. Therefore, ABC provides a semantic routing service to applications. Compared with other data-oriented protocols, ABC is light weight but powerful enough to supported ad hoc applications. "Light weight" here means it experience a good performance according to our simulation. "Powerful enough" here refers to 32 bit semantic description of nodes.

At the beginning of this thesis, we introduced briefly the idea of ad hoc networks, and explained the problem that we would like to address in this project: Nodes in ad hoc networks are lacking the knowledge of destinations under current routing protocols. ABC is designed to address this problem following the coloured stream analogy. After presenting the orientation of our protocol, we explained a basic design for ABC, which demonstrates the feasibility characteristic-based routing but takes no consideration of performance. Based on the analysis of the basic design, we proposed a numerical approach and discussed some critical design options for this approach. According to the initiatives and design options, we discussed the features of ABC and compared it with other data-oriented communication systems to demonstrate the advantages of our protocol. In order to support our design choices, we presented the results from a number of experiments.

The results of the evaluation show that ABC is able to achieve a characteristic-based routing. It scales well with the number of ad hoc nodes. The latency of requests is acceptable and can be configured to achieve a balance with success rate.

6.2 Future work

ABC is like a young child: It is very promising but can be improved in many aspects. The project we have done has just opened a door. Future work includes the following:

- Multiple characteristics requests: The current version of ABC supports only request for single characteristic. We assume that conjunct characteristics requests will lead to improved flexibility. A numerical characteristic flow used in this project may leverage conclusions from operations research.

- Session management: The communication between two nodes in ABC network is stateless at the moment. A simple improvement is to assign a session id. At an intermediate node, next-hop addresses for both direction of a session can be recorded. Moreover, a session could be encoded as a new characteristic. In this way, a node could perform routing for a conversation if the session is disrupted along the path.

- Incorporate IP addresses: An IP address may be treated as a characteristic and allow the routing for IP address.

- Improved characteristic encoding: The encoding scheme presented here is very simplistic. Further research may focus on a more efficient encoding or a more semantic encoding scheme.

Chapter 7: References

[1] Comer D. E. (2001). *Internetworking With TCP/IP,* vol 1. I Prentice Hall.

[2] Murthy C. Siva Ram, Manoj B.S. (2004). *Ad Hoc Wireless Networks.* Prentice Hall Communications Engineering and Emerging Technologies Series.

[3] Roth M., Wicker S. (2003, December) *Termite: ad-hoc networking with stigmergy.* Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE

[4] Petrovic M., Muthusamy V., and Jacobsen H.A. (2005, August). *Content-Based Routing in Mobile Ad Hoc Networks.* Proceedings of IEEE MobiQuitous *2005.*

[5] Lenders V., May M., and Plattner B.. (2005, June). *Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach.* World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium.

[6] Marin-Perianu, R.S. and Hartel, P.H. and Scholten, J. (2005) *A Classification of Service Discovery Protocols.* Technical Report TR-CTIT-05-25 Centre for Telematics and Information Technology, University of Twente, Enschede. ISSN 1381-3625

[7] Musolesi, M., C. Mascolo, S. Hailes, *EMMA: Epidemic Messaging Middleware for Ad hoc networks.* Personal Ubiquitous Computing, 10(1) 2005, pages 28- 36.

[8] Jun Liu, Jiejun Kong, Xiaoyan Hong, Mario Gerla *,"Performance Evaluation of Anonymous Routing Protocols in MANETs"*, IEEE Wireless Communications and Networking Conference 2006, April 2006.

[9] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massouli. *Epidemic Information Dissemination in Distributed Systems.* IEEE Computer, May 2004.

[10] A. Khelil, C. Becker, J. Tian, and K. Rothermel. *An Epidemic Model for Information Diffusion in MANETs.* In Proceedings of ACM MSWiM'02, September 2002.

[11] Musolesi M., Mascolo C.(2006). *Controlled Epidemic-style Dissemination Middleware for Mobile Ad Hoc Networks,* Mobile and Ubiquitous Systems - Workshops, 2006. 3rd Annual International Conference page 136.

[12] Nedos A., Singh K., Clarke S. (2006). Mobile Ad Hoc Services: Semantic Service Discovery in Mobile Ad Hoc Networks, PhD Thesis, Distributed System Group, Faculty of Computer Science, Trinity, 2006

[13] Cho, C. and Lee, D., *Survey of Service Discovery Architectures for Mobile Ad hoc Networks* Term paper, Mobile Computing, CEN 5531, Department of Computer and Information Science and Engineering (CICE), University of Florida, Fall, 2005.

[14] Li, L and Lamont, L., *Service Discovery for Support of Real-time Multimedia SIP Applications over OLSR MANETs* OLSR Interop & Workshop 2004, San Diego, USA, August 6-7, 2004

[15] Clausen T., Jacquet P., *Optimized Link State Routing Protocol (OLSR) RFC 3626*

[16] M. Kalantari and M. Shayman, *Routing in wireless ad hoc networks by analogy to electrostatic theory*, in Proceedings of IEEE International Communications Conference (ICC-04). Paris, France, June 2004.

[17] C.P. Hall, A. Carzaniga, J. Rose, and A.L. Wolf, *A Content-Based Networking Protocol For Sensor Networks*. Technical Report CU-CS-979-04, Department of Computer Science, University of Colorado, August, 2004.

[18] A. Carzaniga, M.J. Rutherford, and A.L. Wolf, *A Routing Scheme for Content-Based Networking*. Proceedings of IEEE INFOCOM 2004. Hong Kong, China. March, 2004.

[19] Jeannot E., Knutsson B. (2002). Adaptive Online Data Compression. *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing 2002*. page 372.

[20] Van Mieghem P, Vandenberghe L(2006, April), *Trade-Off Curves for QoS Routing*. INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings 2006

[21] Cormen T.H, Leiserson C.E., Rivest R. L., Stein C. *Introduction to Algorithms*, 2nd ed. MIT press 2001

[22] Bulusu N., Jha S. (2005) *Wireless Sensor Networks: A Systems Perspective*, ARTTECH HOUSE 2005

[23] Comer D., Stevens D. (1999). *Internetworking With TCP/IP Volume II: Design, Implementation, and Internals*. Prentice Hall.

[24] *The OPNET website*. (n.d.). Retrieved May 2006: http://www.opnet.com

[25] *The OPNET documentation.* (n.d.). Release 11.0

[26] *The OPNET documentation.* (n.d.). Release 12.0

[27] *Perkins C. E., Royer E. M. Ad-hoc On-Demand Distance Vector Routing RFC3561*