Continuous TCP Connection to the Internet on Fast-changing Mobile Ad-hoc Network

by

Tianhang Lan

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science in Computer Science

Auburn, Alabama
August 3, 2019

Keywords:
MANET, OLSR, Gateway Switching

Approved by

Alvin Lim, Chair, Professor of Computer Science and Software Engineering
Kai Chang, Professor of Computer Science and Software Engineering
Saad Biaz, Professor of Computer Science and Software Engineering

Abstract

As more and more sensors are being used in all industries, especially the Internet of Things, Mobile Ad-hoc Networks (MANET) are also becoming more popular as good ways to organize wireless devices in the same area. Since devices in MANET can be highly mobile, wireless connection to other networks might be very unstable. There are a set of protocols that allows nodes in a MANET to organize themselves by exchanging routing information with other nodes, such as Optimized Link State Routing Protocol (OLSR), Ad hoc On-demand Distance Vector (AODV), and Dynamic Source Routing (DSR). When connecting to another network, however, there lacks an effective way to maintain a stable connection to another network due to the high mobility of nodes and thus causing problems with TCP connections to the destination network. This thesis provides a design that maintains a stable TCP connection to the Internet when the network environment changes in MANET by letting nodes send out messages and inform the Internet router the route to the MANET. We conducted a set of experiments that evaluate the time taken to recover from a network change and the overhead of the design. The result shows that this design can help MANET recover from a network change in a short time and with minimum overhead.

Acknowledgment

I would like to thank Professor Lim for the advice and feedback he has provided in my studies at Auburn University. Additionally, I would like to thank the advisory committee members, Dr. Kai Chang, and Dr. Saad Biaz.

I would also like to thank Yue Cui, Yufei Yan, John David Sprunger and Abhishek Kulkarni for their help.

I would like to thank my parents for their support and encouragement to pursue graduate studies.

Table of Contents

CHAPTER 1

INTRODUCTION

Beginning with The Advanced Research Projects Agency Network (ARPANET) in 1966, the Internet, which is based on TCP/IP protocol, has been growing very fast. According to a survey from Netcraft in March 2019, there are over 1.4 billion Active websites over the Internet [1]. The size of the Internet has grown 100 times compared to 14 million websites in April 2000. Besides the growth in size, the format of the Internet has also been developed in the past decades. The concept of the Internet of Things (IoT) was first named in 1999 by Kevin Ashton, who worked on a standard for tagging objects using RFID for logistics applications [2]. Thanks to smart sensors that can not only collect data but also connect to wireless networks, IoT becomes a massive network connecting all kinds of sensors and objects. The feature that differs IoT from the current Internet is that nodes in IoT can transfer data without requiring interaction with human beings. There are more than 6.3 billion IoT devices in 2016 and will exceed 20 billion in 2020 [3]. Since there are many kinds of devices and networks in IoT, different architectures are needed in different situations. And as massive data is produced by sensors in IoT and sent to central servers to be processed, IoT is closely connected to the Internet. Thus, connections between networks in IoT and to the Internet is very difficult. Among all kinds of networks in IoT, Wireless Sensor Network (WSN) is one of the most important parts of the IoT.

WSN is a network connecting sensors in the same area. With only a small number of nodes in one network, WSN is a low-cost, small-scale network. But as there is no router

or gateway in WSN, an architecture that can properly manage the nodes is needed. One of the most widely used architecture in WSN is Mobile Ad-hoc Network, or MANET. MANET is a continuous self-ordered, infrastructure-less network of mobile devices connected wirelessly. It does not require any infrastructure support for carrying data packets between two nodes because every node in the networks acts as a router. MANET also has a dynamic topology architecture due to the high mobility of nodes. These features bring a huge challenge to the routing protocol that runs in the network.

There are many kinds of routing protocols being developed to manage routing information between nodes in MANET. In this thesis, we do research based on Optimized Link State Routing Protocol (OLSR) [4] because compared to other routing protocols, OLSR can discover links faster but with less overhead, and provide a mechanism to select proper gateways to establish connections to other networks. However, when current gateway node lost connection and a new node get connected with the destination, it confuses the router or gateway of the outside network because the route to MANET has changed and then lead to the termination of the current connection. The goal of this article is to find a solution to this problem that provides a stable connection when the network topology changes based on the OLSR protocol.

Chapter 2 gives background information about related research and tools. Chapter 3 sites the problem that will be discussed. Chapter 4 provides designs of solutions to the problem. Chapter 5 and 6 include experiments done with the designs and the result collected. We conclude the result and discuss future improvements in chapter 8.

CHAPTER 2

BACKGROUND AND RELATED WORKS

2.1 Mobile Ad-hoc Network

Mobile Ad-hoc Network, or MANET, is an autonomous transitory association of mobile nodes that communicate with each other over wireless links [5]. The history of ad hoc networks can be traced back to 1972 and the DoD-sponsored Packet Radio Network (PRNET), which evolved into the Survivable Adaptive Radio Networks (SURAN) program in the early 1980s [6]. Compared to common wireless network (Wi-Fi), MANET does not rely on a wireless access point (AP). MANET can be and is being used in many different situations. For example, MANET forms an important part of Wireless Sensor Networks, where distributed sensors are connected with each other in a small area. WSN plays an important role in the Internet of Things (IoT). Sensors are sensing and collecting data all the time, the data collected is transmitted through WSN to a server and then analyzed. MANET is a very good way to form a network between sensors since it is a wireless, multi-hop, infrastructure less and self-configuring network.

Nodes in a MANET can communicate directly with other nodes that are in its wireless covering range and can discover new nodes that move into the range. For two nodes that are not directly connected, there are intermediate nodes which will relay packets sent by them. Since nodes in a MANET are mobile and there may be nodes joining or leaving the network at any time, the network is not stable, and route to other

nodes may change rapidly. Many protocols have been developed to organize route information of nodes in a MANET and they can be divided into 4 different categories. First is the table-driven routing, or proactive routing, where nodes maintain fresh lists of destinations and their routes by periodically distributing routing tables throughout the network. This includes Optimized Link State Routing Protocol (OLSR), Distance Routing Effect Algorithm for Mobility (DREAM) [7], Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) [8] and so on. There are two main disadvantage, the high overhead for maintenance, and slow reaction on restructuring and failures. The second is On-demand routing, or reactive routing. This kind of routing protocol finds a route on demand by flooding the network with Route Request packets. Ad hoc On-demand Distance Vector (AODV) [9], Dynamic Source Routing [10] belongs to this category. The main disadvantages are high latency time in route finding and probability of network clogging due to flooding algorithm. There are also hybrid routing and hierarchical routing which combines both proactive and reactive routing to get the best performance.

2.2 Optimized Link State Routing Protocol

OLSR protocol is an optimization of the classical link state algorithm designed for MANET. Compared to other link-state protocols, only nodes that are selected as multipoint relays (MPR) by other nodes in OLSR flood routing information, so it greatly reduces routing overhead, especially in large scale MANET [11]. OLSR uses Hello messages to discover its one hop and two hop neighbors. The format of Hello message is

as in Figure 2.1. In a hello message, a node transmits information about all known links and neighbors, including MPRs selected by the node. Thus it is simple to discover new neighbors when they join the network: First node A broadcast a Hello message, and when node B receives the message it will register A as an asymmetric neighbor since B cannot find itself in A's hello message. Then node B will send a Hello message declaring A as an asymmetric neighbor. A will find itself in the hello message from B, so it will set B as a symmetric neighbor. This time A will send a new Hello message containing the address of B, and B will set A as symmetric neighbor upon receiving. And since the Hello message contains all neighbor information of the sender, A and B can know their 2 hop neighbors from Hello messages.

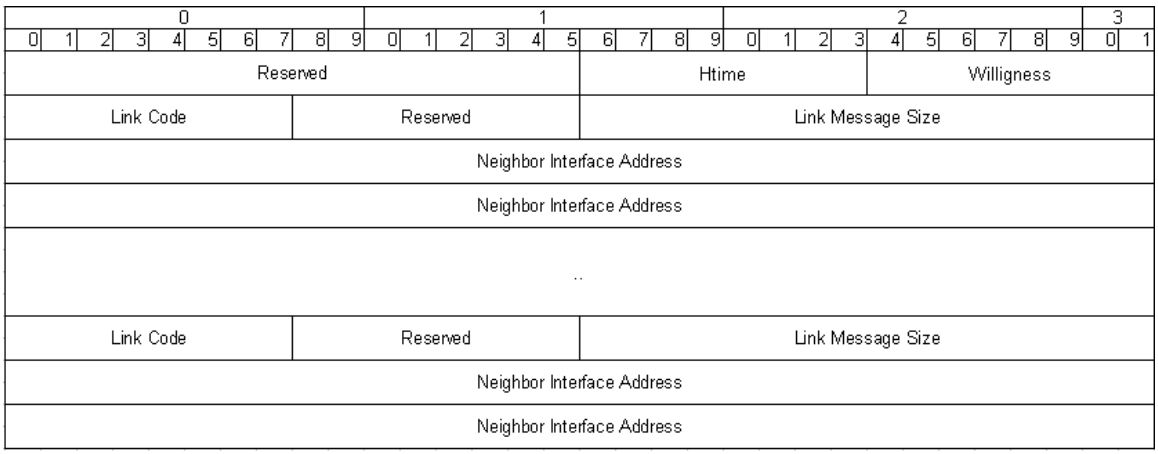

Figure 2.1: Format of Hello Message in OLSR Protocol

To get to know the link state and route to other nodes, a node will select a set of MPRs independently from its symmetric one-hop neighbors. The requirement is that through neighbors in a valid MPR set, the node can reach all symmetric strict two-hop

neighbors. Once a node selects its MPR set, it will mark neighbors in the set as MPR neighbors so nodes selected as MPR will know about this. MPR nodes will flood the Topology Control message, or TC message, that contains network topology information. A node must at least disseminate links between itself and the nodes in its MPR-selector set, in order to provide sufficient information to enable routing. The format of the TC message is shown in Figure 2.2.

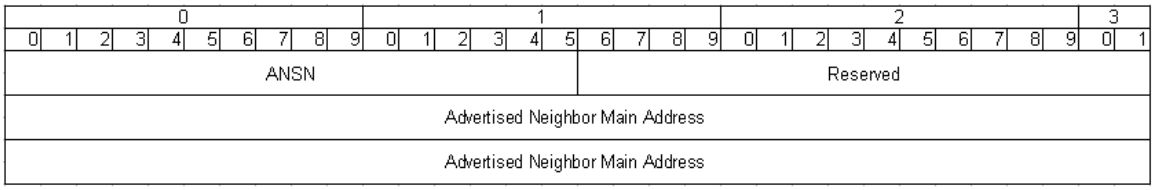

Figure 2.2: Format of TC Message in OLSR Protocol

With the help of routing protocols, nodes can communicate with each other in a MANET, but as we discussed above, there are many situations where nodes need to communicate with hosts that are not in the current network, for example, Internet hosts, and thus needs a route to the destination network. Fortunately, OLSR protocol provides a solution to this problem, a node will try to detect if it is connected to the destination network, and will broadcast a message in the MANET if the destination is reachable. Other nodes receiving this message will select it as the gateway to the destination network. This message is the Host and network association (HNA) message. An HNA message can be generated when a non-OLSR interface of a node is connected to another network, indicating that this node can be a gateway to another network. HNA message contains network address and netmask of outside networks. Nodes trying to connect to

another network will select the node sending corresponding HNA message as the gateway to the destination, and will select the one with best link quality if there are multiple available gateways. The format of the HNA message is shown in Figure 2.3.

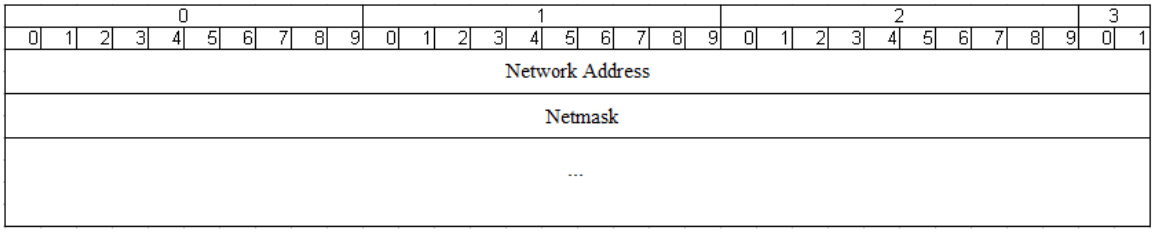

Figure 2.3: Format of HNA Message in OLSR Protocol

Usually, Connection to another network is written in OLSR's configure file with field *Hna4*, and is not changeable during the running of OLSR. However, OLSR provides a plugin named Dynamic Gateway. Dynamic Gateway plugin provides a mechanism that can dynamically detect network connection and send HNA messages. The plugin will execute a ping command to a destination given as a parameter, and if the ping command is executed successfully, which means that there is Internet connection, the plugin will broadcast an HNA message to the OLSR MANET indicating the connection to the Internet, and other nodes in the network can select it as gateway.

2.3 Multipath TCP

Multipath TCP (MPTCP) is an effort towards enabling the simultaneous use of several IP-addresses/interfaces by a modification of TCP that presents a regular TCP

interface to applications, while in fact spreading data across several subflows [12]. By creating multiple subflows, the TCP connection will not be terminated when one of the links is cut off. Also, utilizing multiple links at the same time will increase the throughput to the sum of all available links

MPTCP has 3 key components, master subsocket, multipath control block (MPCB), and slave subsocket. Master socket is a standard socket for TCP connection; MPCB provides function like starting or terminating a subflow, choosing which subflow to send data, and reassemble MPTCP packets; slave subsocket is not known to applications, controlled by MPCB to send data. The structure of MPTCP is shown in Figure 2.4.

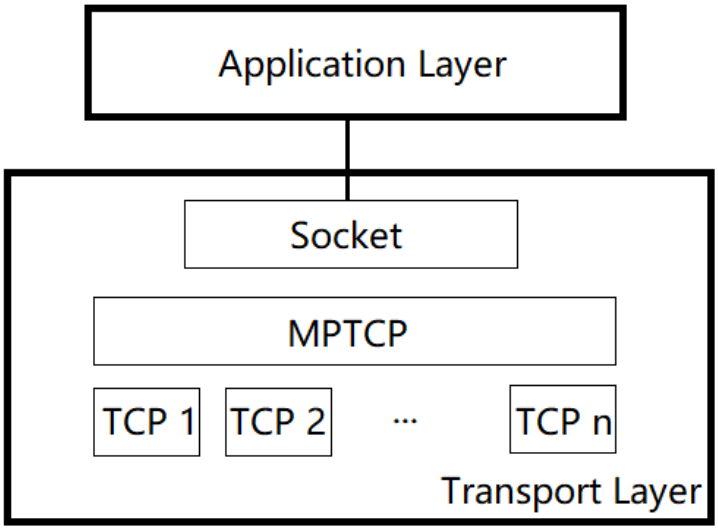

Figure 2.4: Structure of Multipath TCP

MPTCP is compatible with application layer protocols since it provides services just as regular TCP. Starting with a 3-way handshake, only a few fields are being added by MPTCP. Each subflow also works same as a regular TCP connection. The sequence

number is continuous in each subflow, and there is a data sequence number (DSN) to let MPTCP reassemble packets from different subflows.

2.4 Scapy

Scapy is a Python program that enables the user to send, sniff and dissect and forge network packets [13]. This capability allows construction of tools that can probe, scan or attack networks. Scapy is powerful when coming to packet manipulating. It can forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. When users want to send messages on layers below the transport layer, like network layer, link layer and physical layer, Scapy can be very helpful.

CHAPTER 3

PROBLEM STATEMENT

Although OLSR protocol provides a solution to set up a MANET, connection to another network can be frustrating due to the mobility of nodes in the network. Gateway information provided by HNA message might be changed during an on-going TCP connection. There are several different situations.

The first situation is shown in Figure 3.1. Client C is connected to host H through link L1. The packets are forwarded to gateway G1 and go through Network 1. Router R1 in Network 1 will route packets from G1 to host H. If G1 loses connection to R1, and is then connected to R2, it can still reach host H and C will not change it routing information, the TCP connection from C to H is not affected.

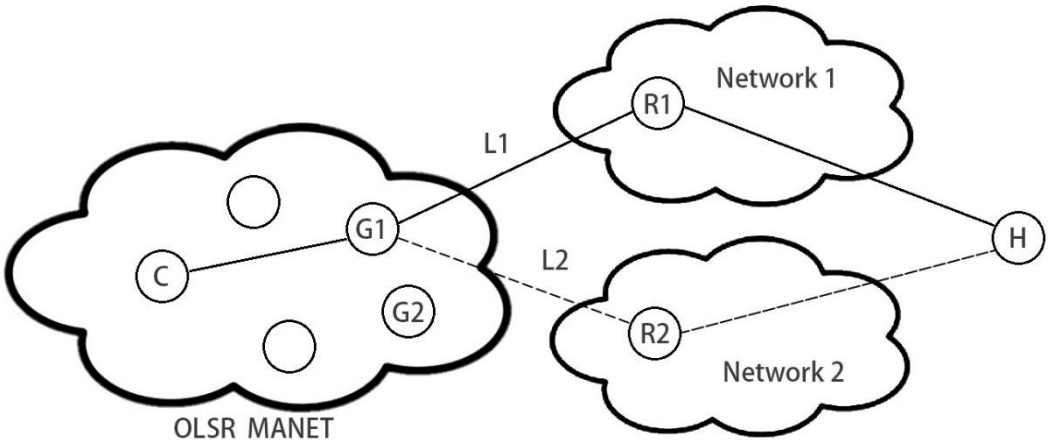

Figure 3.1: Gateway Switching Situation 1

In the second situation, as shown in Figure 3.2, client C is connected to host H through link L1 the same way in situation 1. If G1 loses its connection to R1, C will select G2 as gateway according to latest HNA message from G2 indicating its connection to H. Under this situation, nodes can switch gateway normally and the TCP connection will continue.

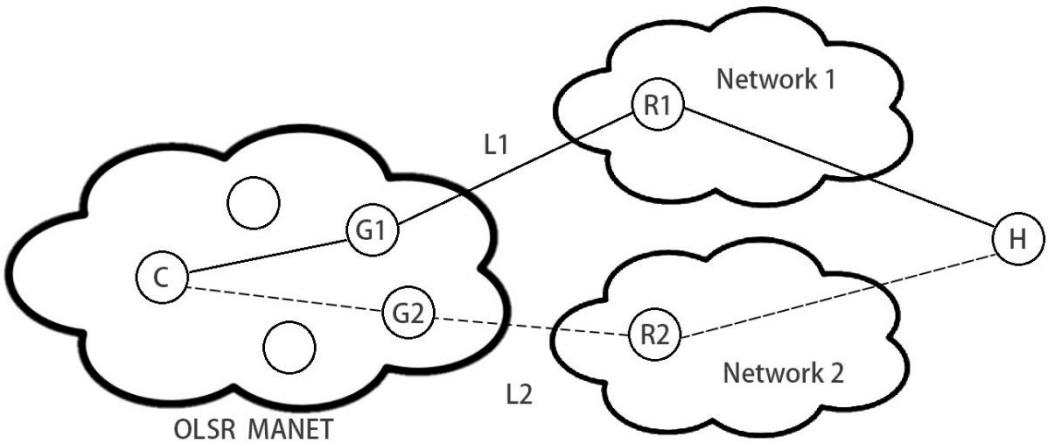

Figure 3.2: Gateway Switching Situation 2

In the third situation which is shown in Figure 3.3, client C is connected to H via link L1. When C selects G2 as its new gateway based on latest HNA message, the router R is not aware of the change in the MANET. If R receives packets from G2, it will forward them to H normally, however the packets from H to C will not be forwarded correctly because the routing table of R indicates that packets should be forwarded to G1 in order to reach C. If G1 is already disconnected, packets from H will not be able to reach C correctly and lead to disconnection of TCP connection between C and H. If G1 is

still connected, but the link quality between G1 and C is very bad, which a very possible reason for C to switch its gateway to G2, packet loss, retransmission or high latency might be observed.

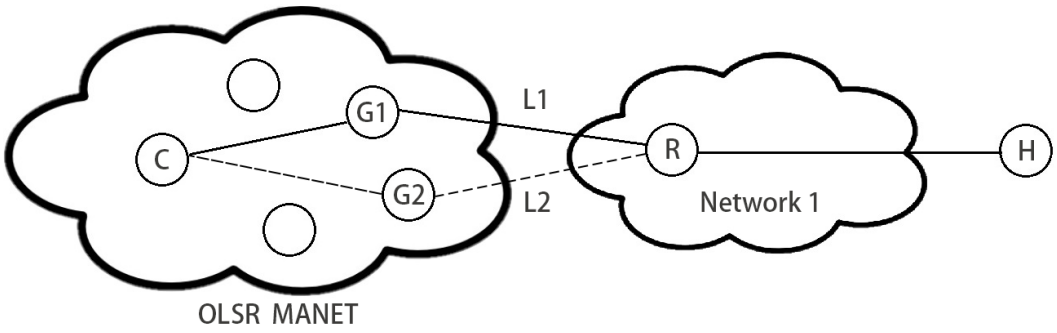

Figure 3.3: Gateway Switching Situation 3

In order to keep the TCP connection alive and efficient when switching gateways in situation 3, router R needs to know which gateway can be used as the route back to OLSR MANET.

CHAPTER 4

PROTOCOL DESIGN

4.1 Multipath TCP

Since there are 2 different links in Figure 3.3, utilizing both links at the same time seems to be a doable way, and based on this idea, we look into currently implemented protocols that can use multiple links at the same time.

The first design is to establish multiple subflows of TCP connection from the client to host using MPTCP. For example, in Figure 3.3, According to the idea of MPTCP, if there are two subflows, L1 and L2, exist at the same time, and when one of the subflow is cut down, another subflow can continue to transmit packets without being affected. This will not only solve the problem of gateway switching but also improve throughput since it makes use of multiple connections at the same time.

However, when trying to apply MPTCP to the MANET, we find out that MPTCP only works when subflows are established on different interfaces. While clients in the MANET only have one active interface, establishing subflows will not succeed since it will be recognized as the same flow due to having the same IP address and port number. So having multiple subflow on one node is not working, and MPTCP is not applicable to the OLSR MANET.

Since protocols that already exists does not meet our requirement, a new kind of protocol is designed.

4.2 Heartbeat Message from Gateways

Using multiple subflows proved to be not working under this situation, we tried to find another approach. Since utilizing both links at the same time is not an option, we need to find a way to let router R correctly route the packets to MANET. The key to select a proper gateway to MANET is that router R knows which gateway is being used by MANET nodes. To select the right gateway, R needs more information about which gateway is active and which is not. Since active gateways are one-hop neighbors to R, we can let gateways sending heartbeat message to router R. R will know a gateway is active when it receives packets from the gateway, and if there is no packet coming from the gateway for a period of time, it is safe to assume the gateway turned inactive, and inactive gateway can turn back to active state when it is connected to R by starting to send new heartbeat message to router R.

We came up with a second design to implement ideas above. In this design, all gateways send a heartbeat signal to the router at a fixed frequency. As shown in figure 4.1, gateway G1 and G2 will send a heartbeat message to router R, when R receives the message, it will register the sender as an alive gateway to the MANET. The message is valid for a certain amount of time and the expire time is refreshed upon arrival of a new heartbeat message, and if there is no new heartbeat message received and the old one expires, R will unregister the gateway, and switch to another valid gateway.

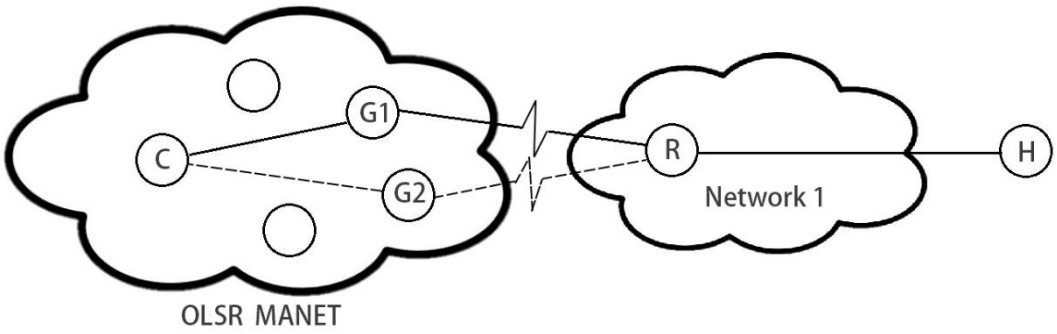

Figure 4.1: Heart-beat Message from Gateways

4.3 Gateway-switch Message

4.3.1 Gateway-switch Message from Clients

In section 4.2, we discussed sending heart-beat from gateways with a fixed interval. However, heartbeat packets are useless when there is no gateway switching happens, causing waste of network resources. There are also situations that this design may not work properly. As shown in Figure 4,2, normally gateway G1 and G2 are connected to router R, and client C1 is sending TCP packets through link L1. When C1 switches its gateway to G2, the connection may not be affected since the outgoing packets go through L1, and incoming packets go through link L2 from R to G2 and back to C1. But if the connection between C1 and G1 is unstable, which might be the reason for C1 to switch its gateway to G2, not changing the route back to C1 will cause packet loss and retransmission. And if gateway G1 is disconnected to the MANET, but still connected to

the router R, router R will not change the route to C1 and cause TCP connection to be terminated. Also, this model may not work when there are multiple clients connecting to different hosts through different gateways since router R reaches back to MANET only through a fixed gateway. For example, when 2 different clients that selected different gateways are connecting to the Internet at the same time, router R will only forward packets from the Internet to one gateway. So, we come up with another design that let the router knows the route to every node that has established a TCP connection to the outer network.

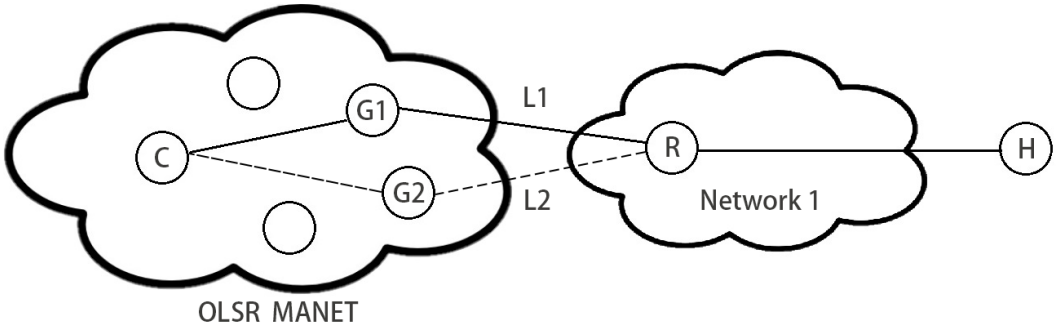

Figure 4.2: Income and Outgoing Traffic Through Different Gateways

In this design, as shown in Figure 4.3, when a client starts a connection to another network, it will send a gateway switch message indicating the gateway it selects. Upon receiving the message, router R will register the client and its gateway as the route to the

client. When the client switches gateway, it will also send a gateway switching message and when R receives the new message, it will change the route to the client correspondingly.

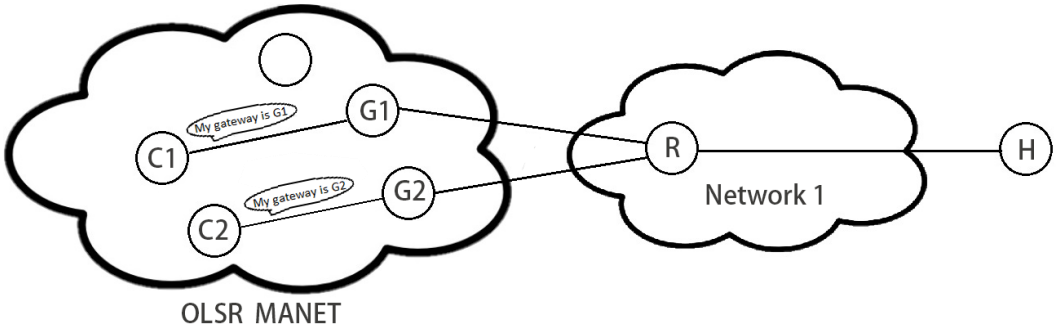

Figure 4.3 Gateway-switching Message from Clients

4.3.2 Gateway-switch Message from Gateways

In the third design, clients need to send a gateway-switch message to let the router know the gateway it selects. But clients may not know the IP addresses of the current Internet router, and unable to set the destination of the gateway-switch message. But gateway nodes always know the IP address of the Internet router it selected, so let the gateway to send the gateway-switch message can be a better approach, and that forms the fourth design.

In the fourth design, clients will send its gateway-switch message to the gateway it selected to let the gateway know, just like in Hello message a node notifies its neighbors about MPR node selecting. When a gateway receives the message, it will send a message

to the Internet router saying it is the correct next hop to reach the client, and router R will set the route to clients with the message from gateways. As shown in Figure 4.4, gateway G1 and G2 receive messages from client C1 and C2 indicating their gateway selection, then they will send to router R gateway-switch messages that contain clients select them as gateways. R will select the gateway to C1 and C2 correspondingly.

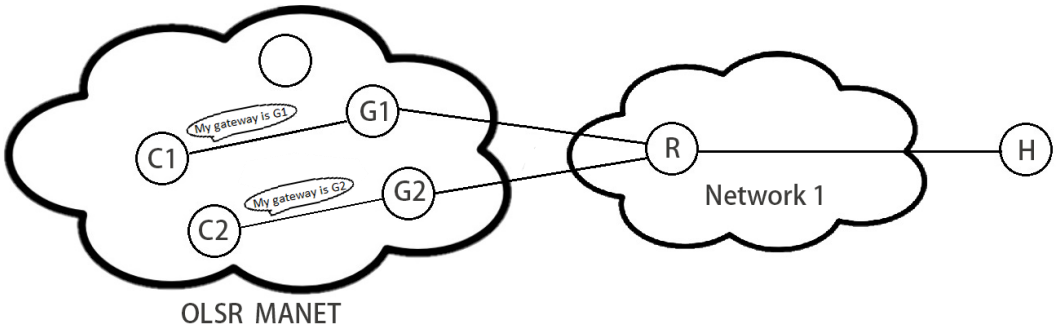

Figure 4.4 Gateway-switching Message from Clients

CHAPTER 5

IMPLEMENTATIONS

5.1 Implementation of Heartbeat Design

5.1.1 First Implementation

For the heartbeat message, there are many protocols to choose from including ICMP, UDP, TCP and a self-designed protocol. TCP keeps a live connection between nodes and has a larger overhead compared to other protocols, so it does not suit our needs. UDP is one of the most used protocols for carrying messages, but since heartbeat message does not carry any actual information other than being a signal of active state, ICMP packets can be a better choice since it is shorter than UDP packets and not like UDP which is part of transport layer, ICMP is part of the network layer as OLSR protocol. ICMP also has another advantage. By setting the TTL to 1, only message sent by nodes that are directly connected to the outer network, which are actually the active gateway nodes, will be delivered to the outside router. Message from other nodes will not flood the message all over the MANET since packets are being dropped after only one hop. This prevents flooding of ICMP packets. With all these advantages, ICMP perfectly fulfills our demand, there is no need to design a protocol from scratch. We choose the ICMP echo packet, which is commonly known as the ping request, as the heartbeat message format. Since ICMP packets are usually short, and thus will not increase the overhead too much, the performance of MANET will not be heavily affected.

To keep track of alive gateways, the router of the outer network maintains a list of active gateways as in Figure 5.1. The timestamp is the time when the last heartbeat message is received. Upon receiving a heartbeat message, the router will first check if the sender is one of the active gateways. If it is then refreshing the timestamp, and if it is not, add a new row to the list. If the gateway turns inactive is the currently selected gateway to MANET, the router will go through the list, compare the timestamp with the current time, and select the next gateway in the list that is still active as the new gateway to MANET.

Gateway 1 IP address	Timestamp 1
Gateway 2 IP address	Timestamp 2
Gateway 3 IP address	Timestamp 3
……	……

Figure 5.1: List of Gateways

5.1.2 Improved Implementation

After implementing this design, we think of ways to reduce the impact on performance caused by heartbeat messages. When looking through OLSR's dynamic gateway plugin, we realize that the plugin detects network connection by sending ICMP

packet the Internet router. So instead of letting nodes send separate ICMP packets to the router, we modified the configuration of Dynamic Gateway plugin, including setting TTL to 1 by adding *-t 1* option to the ping command and shorten the interval and validity time of HNA messages. With these modifications, there is no extra data generated and transmitted between the gateway and Internet router.

5.2 Implementation of Gateway-switch Design

5.2.1 First Implementation of Gateway-switch Message from Clients

For gateway-switch message, since actual route information is sent from the client, ICMP alone cannot satisfy the need. Thus we choose UDP instead. Since the client does not know the IP address of the Internet router, it will set the destination as a remote host on the Internet. Router R will sniff any frame containing a UDP packet forwarded to it, when the gateway-switch message packet is forwarded to router R, it will record the client's address in the IP header, and gateway address in UDP payload.

Router R maintains a dictionary contains all clients and their corresponding gateway, the structure is shown in Figure 5.3. Once it finishes analyzing the UDP packet, it will update its dictionary, adding a new pair of key and value or updating the current key pair based on the two addresses from the UDP packet.

Key	Value
Client 1 IP address	Gateway 1 IP address
Client 2 IP address	Gateway 2 IP address
Client 3 IP address	Gateway 3 IP address
……	……

Figure 5.3: Dictionary of Clients and Gateways

But during the coding phase, we found a serious problem that client C does not actually know the IP address of gateway G's interface connected with router R. So we need to change the design.

Since client C does not know about the network outside the OLSR MANET, we tried to let router R to discover the gateway. Instead of putting the gateway information in UDP packet's payload, we utilize the MAC address of packet forwarded to router R. When a UDP packet from nodes in the OLSR MANET is forwarded to router R, the source MAC address is the MAC address of the gateway since it is the one-hop neighbor of router R. Router R will look up its ARP table, and find the corresponding IP address of that MAC address. Then router R can form a pair of IP addresses and update its dictionary. Since the payload of UDP packets is no longer used, we use ICMP packet instead of the UDP packet.

5.2.3 Implementation of Gateway-switch Message from Gateways

Sending UDP or ICMP packet to a remote host is not a very good move since the packet travels a long distance to reach the remote host but does not contains any useful information and might affect programs run on the remote host. So, we changed the strategy of sending gateway-switch message.

In this design, the client will send the gateway-switch message to its gateway instead of a remote host. When the gateway receives a message from the client, it will send another data packet to the Internet router with the IP address of the client in payload field. When the router receives the message from the gateway, it will parse the packet. The client IP address is in payload, and the gateway IP address is the source IP address of the data packet. Then the router can update its route information dictionary.

By splitting forwarding of routing information into two-part, the Internet router in this design no longer needs to look up ARP table, which is a part of the link layer,

With UDP packet the program runs on both network layer and transport layer. The logic and structure are better and clearer after switching to ICMP because the whole program only runs on the network layer. To achieve this, we need an IP layer protocol that can carry a small piece of data to a specific IP address. By looking deeper into network layer protocols, we find that ICMP structure, as shown in Figure 5.4, actually has a payload field which can be used to hold a few bytes. So, we make an improvement to the prototype. The client sends an ICMP packet to the gateway and the payload is set

to be a fixed string indicating this is a gateway-switch message, and when the gateway received this packet, it can put the IP address of the client into the payload of the message that will be sent to the Internet router. Then the router can update its route information dictionary with the key-value pair.

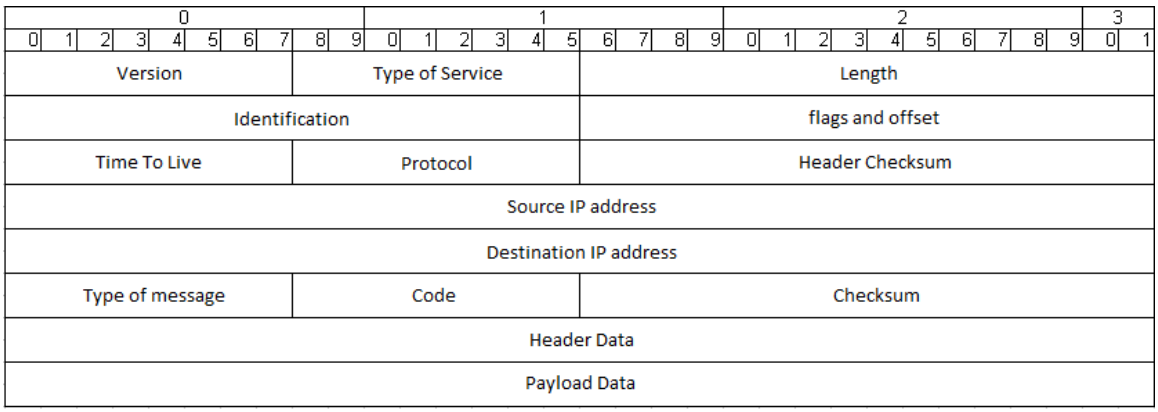

Figure 5.4: Structure of ICMP Packet

This change not only canceled the transport layer part and makes all parts of the program runs on the network layer, but also decreases overall overhead.

CHAPTER 6

EXPERIMENT AND RESULT

6.1 Experiment of Heartbeat Design

6.1.1 Experiment of the Prototype Implementation

To test if the heartbeat design works, we set up a small network as shown in Figure 5.1. We use 3 laptops to form an OLSR MANET, one of them act as client C that communicates with remote server and 2 act as gateways G1 and G2 which route packets from client C to router R. We used a laptop running Ubuntu system instead of an actual router as router R by enabling packet forwarding so we can run programs on it.

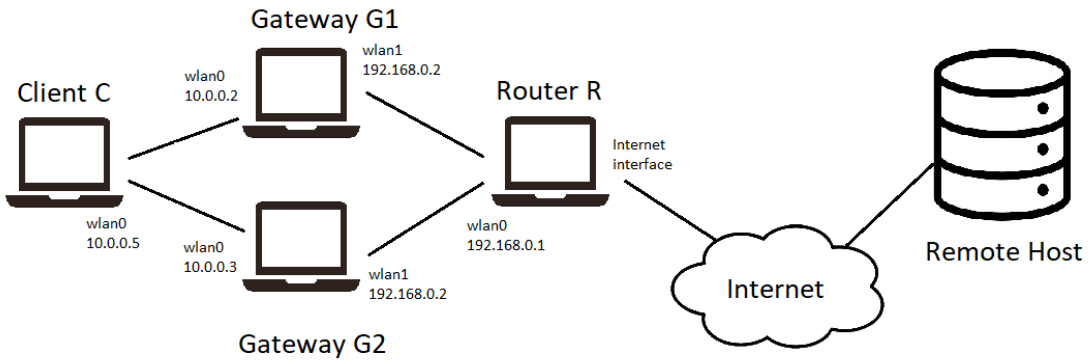

Figure 6.1: Heart-beat Design Experiment Setup

There are 3 different local area networks. The first one is 10.0.0.0/24 network. It is an OLSR MANET consist of client C, gateway G1 and G2. The second is 192.168.0.0/24

network. It is the network in which gateways and Internet router are connected. The third network is the network that the routers Internet interface belongs to. During the test we use the Internet interface as the remote host because it can reduce the impact from the Internet to the lowest level. In the beginning, Client C and Router R both have G1 as its gateway to another network. When the experiment starts, two gateways start to send heartbeat messages to wlan0 interface of R. At a certain time, we cut G1 off to simulate the situation when G1 lost the connection, and force C to switch to G2. When the network returns to a stable state, we reconnect G1 and disconnect G2 to let C switch back to G1.

During the test, we used different configurations, and tested the time takes to switch, the behavior during the test, and bandwidth occupied by heartbeat message. There are two configurable fields:

- Heartbeat Message Interval (Time between 2 heartbeat message)
- Heartbeat Message Validity Time (Time a heartbeat message is valid for)

We did the experiment with 2 sets of configurations.

Configuration 1:

- Heartbeat Message Interval = 1 second
- Heartbeat Message Validity Time = 2 second

Configuration 2:

- Heartbeat Message Interval = 0.5 second
- Heartbeat Message Validity Time = 1 second

First, we observed the behavior of the network when switching gateway using configuration 1.

Time	Packet
0.63	G1 Sends HNA message
2.16	C sends packet 39
2.17	Ack of packet 39
2.26	C sends packet 40
2.27	Ack of packet 40
2.36	G2 Sends HNA message
2.36	C sends packet 41
2.57	C sends packet 42 43
2.79	Retransmit packet 41 42 43
3.23	Retransmit packet 41 42 43
4.01	G2 Sends HNA message
4.11	Retransmit packet 41 42 43 Through G2
5.87	Retransmit packet 41 42 43
5.88	Ack of packet 43
5.88	C sends packet 44-76
5.89	Ack of packet 76
5.97	C sends packet 77
6.00	Ack of packet 77
6.07	C sends packet 78
6.08	Ack of packet 78
6.17	C sends packet 79
6.19	Ack of packet 79

Table 6.1: Configuration 1 Network behavior during Gateway Switching

From Table 6.1 we can see that in the beginning, C is sending a message to R through G1. When gateway G1 is cut off at 2.3 seconds, the packet cannot reach R, and retransmission happened at 4.1 seconds. The HNA message from G1 expires at about 3.6 seconds, and C switches to G2. The router R changed its gateway to the OLSR MANET to G2 at about 5 seconds due to the ICMP packet from G1 expires, and the missed packets is received by R at 4.88 seconds. C then sends all delayed packets from 44 to 74 to R, and the TCP connection resumes to normal state. The time for the network to switch gateway is about 3.2 seconds, and total time between G1 being cut off and the network resumes to normal state is 3.5 seconds.

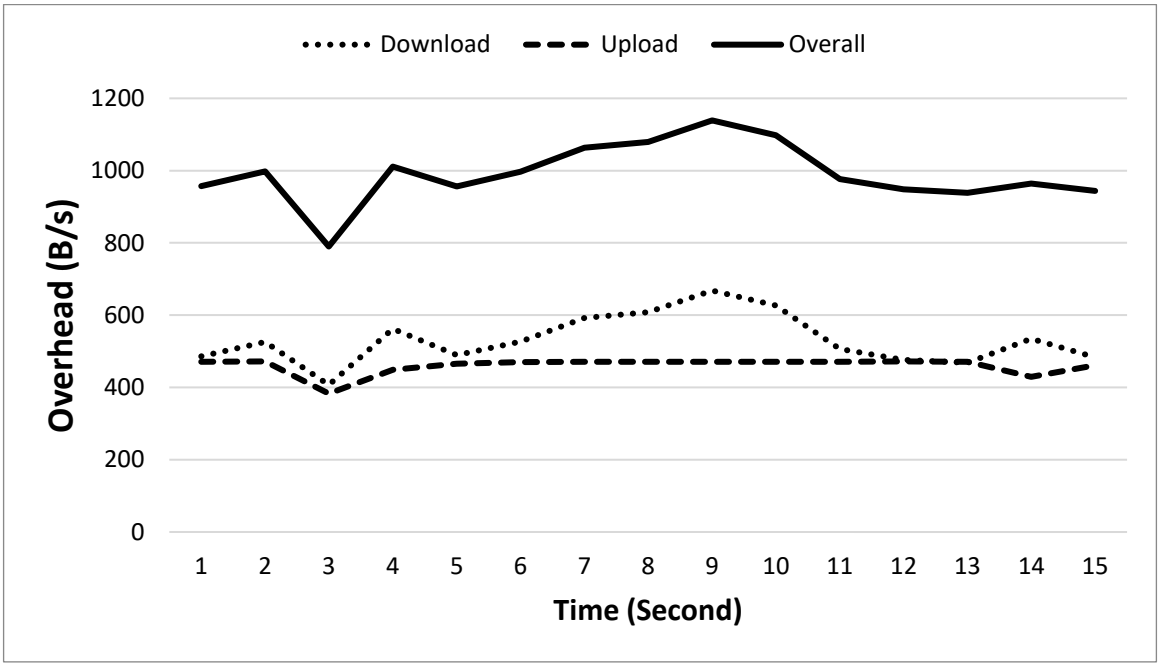

Figure 6.2: Overhead of Configuration 1

Then we measured the overhead brought by OLSR and the design. The result is as Figure 6.2. It is the bandwidth taken when there is no data transmission between the gateway and the Internet router. The download and upload bandwidth are all round 500 Byte/second, and the overall average bandwidth occupation is 990 Byte/second.

Time	Packet	Time	Packet
0.63	C sends packet 91	4.82	G2 Sends HNA message
0.64	Ack of packet 91	4.98	Retransmit packet 94 95 96 Through G2
0.73	C sends packet 92	4.98	Ack of packet 96
0.74	Ack of packet 92	4.98	C sends packet 97-139
0.76	G1 Sends HNA message	4.98	C sends packet 140-164
0.83	C sends packet 93	4.99	Ack of packet 139
0.85	Ack of packet 93	5.10	Ack of packet 164
0.93	C sends packet 94	5.11	C sends packet 165
0.95	G2 Sends HNA message	5.11	Ack of packet 165
1.14	C sends packet 95 96	5.15	C sends packet 166
1.37	Retransmit packet 94 95 96	5.16	Ack of packet 166
1.80	Retransmit packet 94 95 96	5.25	C sends packet 167
2.69	Retransmit packet 94 95 96	5.26	Ack of packet 167
2.70	G2 Sends HNA message	5.35	C sends packet 168
4.45	Retransmit packet 94 95 96	5.36	Ack of packet 168

Table 6.2: Configuration 2 Network behavior during Gateway Switching

Then we run the test with configuration 2. The behavior is in Table 6.2. The whole procedure is very similar to the first run, but the time taken for the network to recover is 4

seconds, which is slightly longer. The reason for this is high frequency and short validity time of heartbeat message makes the system more vulnerable to packet loss. This is proven by gateway switch happening frequently before G1 is cut off.

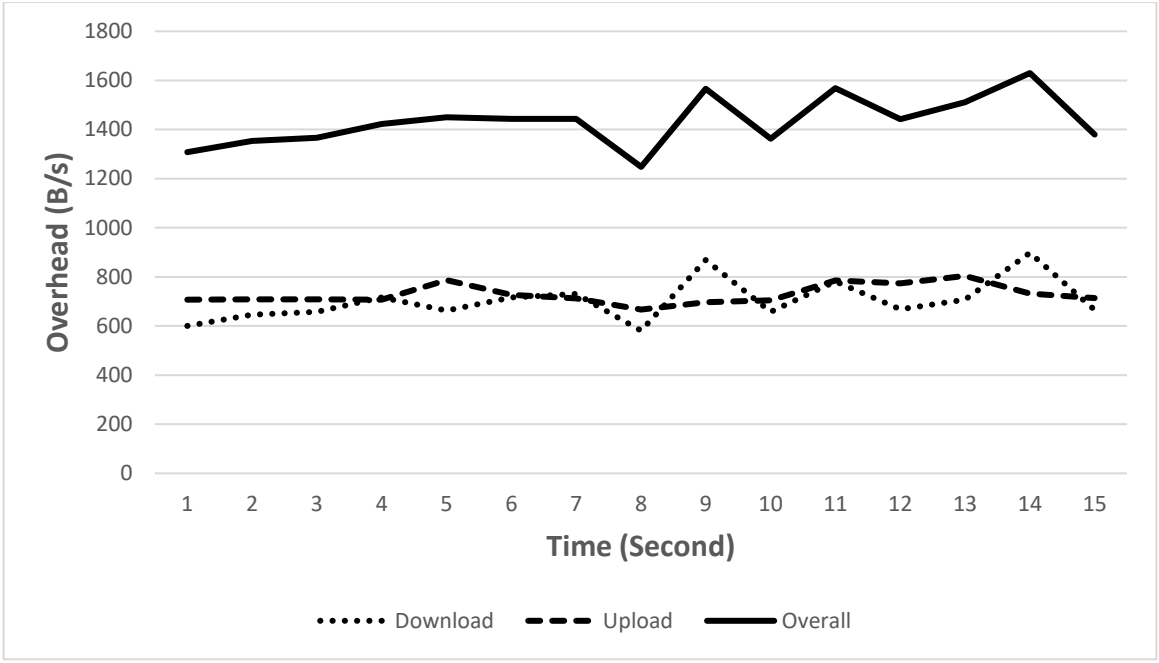

Figure 6.3: Overhead of Configuration 2

The overhead of the second configuration is shown in Figure 6.3. The Download and Upload bandwidth are around 700 Byte/second, and overall takes 1433 Byte/second. With double ICMP packet sent per second, the increase of overhead is as expect.

6.1.2 Experiment of the Improved Implementation

Since the heartbeat message is combined with ping packets from the gateway, we need to directly adjust the configuration of OLSR's Dynamic Gateway plugin. The following are configuration used to run the experiment:

- ping (the destination of ping command) = 192.168.0.1
- pingcmd (the command executed) = ping -c 1 -t 1 -w 1 %s
- pinginterval (time between 2 ping packets) = 0.5 second
- HnaInterval (time between 2 HNA messages) = 1.6 seconds
- HnaValidityTime (time a HNA message is valid) = 3 seconds

In pingcmd the %s will be replaced with ping parameter. During the experiment, we found out that although you can set the HNA interval lower, the OLSR will only send one packet every 1.7 seconds. The difference is when the interval is lower than 1.7s, there are sometimes more than 1 HNA message in one OLSR packet.

Time	Packet
0.60	G2 Sends HNA message
0.61	C sends packet 91
0.61	Ack of packet 91
0.67	G1 Sends HNA message
0.71	C sends packet 91
0.72	Ack of packet 92
0.81	C sends packet 93
0.82	Ack of packet 93
0.91	C sends packet 94
0.92	Ack of packet 94
1.01	C sends packet 95
1.21	C sends packet 96 97
1.42	Retransmit packet 95 96 97
1.84	Retransmit packet 95 96 97
2.56	G2 Sends HNA message
2.68	Retransmit packet 95 96 97
4.36	Retransmit packet 95 96 97 Trough G2
4.36	G2 Sends HNA message
4.37	Ack of packet 97
4.37	C sends packet 98-128
4.37	Ack of packet 128
4.42	C sends packet 129
4.42	Ack of packet 129

Table 6.3: Improved design Network behavior during Gateway Switching

The procedure is still quite similar to the original design, and the time taken to switch gateway is about 3.4 seconds, almost the same compared to the original design

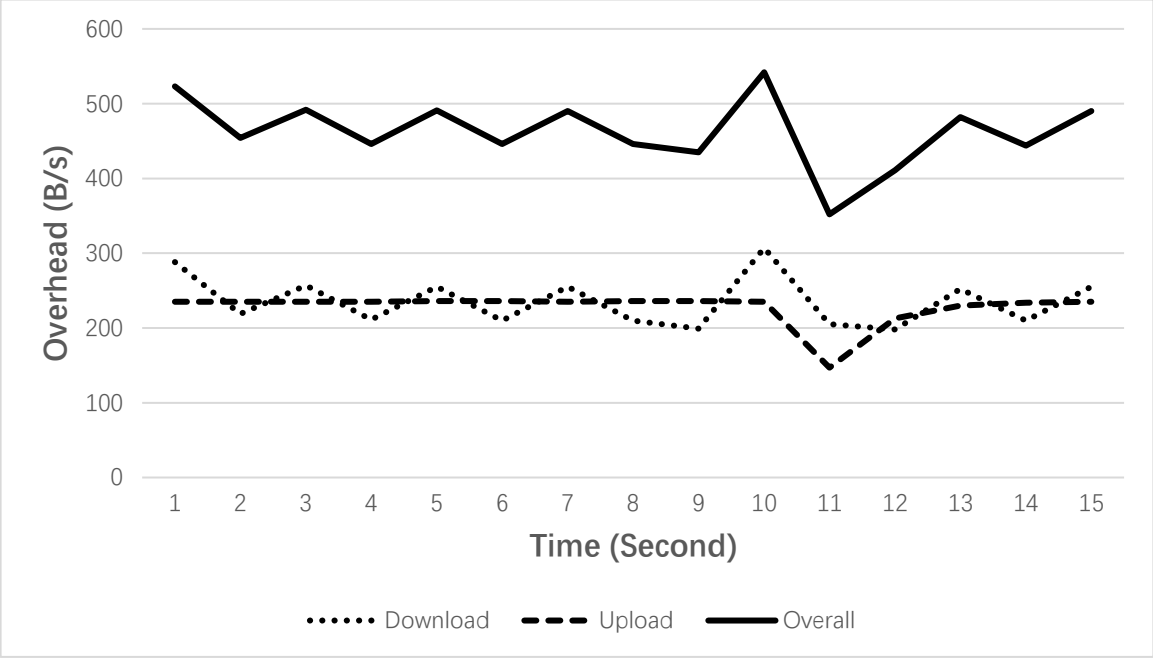

Figure 6.4: Overhead of Improved Heartbeat Design

The overhead of the improved heartbeat design is shown in Figure 6.4. The Download and Upload bandwidth are around 250 Byte/second, and 466 Byte/sec overall. The overhead is very small compared to the original design, only about 50% of configuration 1. Integrating heartbeat message into HNA sensing prove to be effective.

6.2 Experiment of Gateway-switch Design

At the start, we used the same network as Figure 6.1 to test gateway-switch design. The client C will send a gateway-switch message when its gateway changes so different clients can choose different gateways.

From Table 6.4 we can see that the major process is the same. Gateway G1 is cut off, C switches gateway based on HNA message, router R senses the change and select a proper route to C. But notice that at 3.23 seconds, there is a gateway-switch message sent by C. Unlike in the previous experiment where it took the client 2 retransmission after gateway switching to receive Ack packet from router R, after sending the message, retransmission is immediately acknowledged by R. This greatly reduced the time needed to recover the loss of current gateway node for the client, and is shown in the table. The switch time is only 1.7 seconds which is only half compared to 3.4 seconds in heartbeat design.

Time	Packet	Time	Packet
0.08	G1 Sends HNA message	2.19	Retransmit packet 102 103 104
0.12	G2 Sends HNA message	2.61	Retransmit packet 102 103 104
1.27	C sends packet 97	3.23	C sends Gateway-switch message
1.28	Ack of packet 97	3.45	Retransmit packet 102 103 104 Through G2
1.38	C sends packet 98	3.46	Ack of packet 104
1.39	Ack of packet 98	3.46	C sends packet 105-118
1.48	C sends packet 99	3.47	Ack of packet 118
1.48	Ack of packet 99	3.48	C sends packet 119
1.58	C sends packet 100	3.48	Ack of packet 119
1.60	Ack of packet 100	3.48	C sends packet 120
1.68	C sends packet 101	3.49	Ack of packet 120
1.68	Ack of packet 101	3.58	C sends packet 121
1.78	C sends packet 102	3.59	Ack of packet 121
1.98	C sends packet 103 104	3.68	C sends packet 122
2.07	G2 Sends HNA message	3.69	Ack of packet 122

Table 6.4: Network behavior of the Gateway-switch Design during Gateway Switching

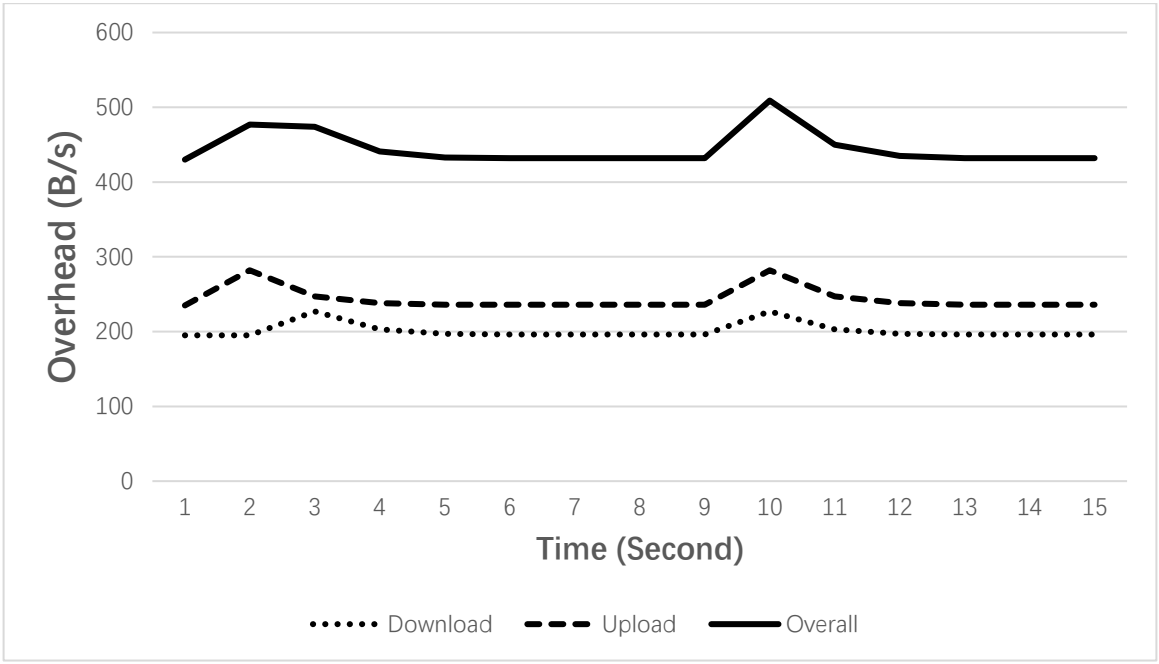

Figure 6.5: Overhead of Gateway-switch Design

According to Figure 6.5, the download overhead is about 200 B/s, the upload overhead is about 240 B/s and the total overhead is 445 B/s. Although the new design no longer utilizes the HNA message, the overhead is still about the same compared to the improved heartbeat design. This is because compared to the improved heartbeat design, messages sent at a fixed frequency in gateway switch design is exactly same, which mainly includes the ICMP packets to sense if the node is connected to the outer network. Although there are also gateway-switch messages being sent, they are very short and is sent only when there is a gateway switch.

After testing the performance of gateway-switch design under the same environment as heartbeat design, we established another small network to test how the design works with multiple clients. The network is simple: 4 laptops connected with each other form an OLSR MANET, 2 of them act as client C1 and C2, other 2 are gateway G1 and G2; G1 and G2 are connected to router R, and R is connected to the Internet.

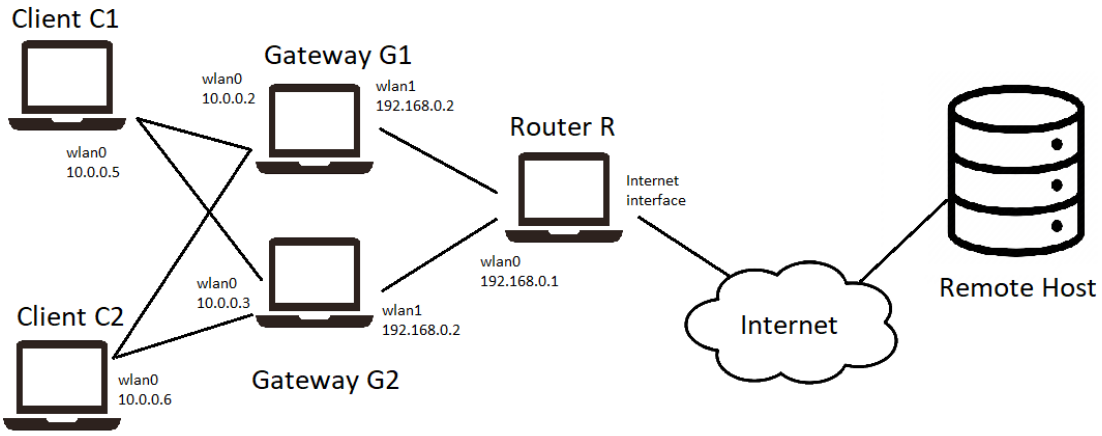

Figure 6.6: Gateway-switching Design Experiment Setup with 2 Clients

In the beginning, C1 and C2 start two different TCP connection to the remote host using two gateways. C1 uses G1 as the gateway, and C2 uses G2 as the gateway. During the experiment, C1 and C2 will switch to another gateway and we will record the behavior and overhead of the system. Since there are two gateways, the data is collected on the router R to avoid unsynchronization of time between two clients.

Time	Packet	Time	Packet
0.33	C1 sends packet 26	5.75	C1 sends packet 80
0.33	Ack of packet 26	5.75	Ack of packet 80
0.43	C1 sends packet 27	5.85	C2 sends packet 72
0.43	Ack of packet 27	5.85	Ack of packet 72
0.47	C2 sends packet 19	5.89	C1 sends packet 81
0.47	Ack of packet 19	5.89	Ack of packet 81
0.52	C2 sends gateway-switch through G1	5.94	C2 sends packet 73
0.55	C1 sends packet28	5.94	Ack of packet 73
0.55	Ack of packet 28	5.97	C1 sends Gateway-switch through G2
0.65	C2 sends packet 20 through G1	6.01	C2 sends packet 74
0.65	Ack of packet 20 through G1	6.01	Ack of packet 74
0.67	C1 sends packet 29	6.07	C1 sends 82 83 from G2
0.67	Ack of packet 29	6.07	Ack of packet 83 From G2
0.68	C2 sends packet 21	6.11	C2 sends packet 75
0.68	Ack of packet 21	6.11	Ack of packet 75
		6.12	C1 sends packet 84
		6.12	Ack of packet 84

Table 6.5: Network behavior of the Gateway-switch Design with 2 Clients

As we can see from Table 6.5, the gateway switch happened in a different way as before. C2 switched from G2 to G1 at 0.52 second and C1 was not affected, and when C1 switched from G1 to G2, C2 continues to run on G1. This proves that the gateway-switch

design works well with multiple clients and multiple gateways, and solves the problem brought by heartbeat design. As G1 and G2 are not cut off during the experiment, the connection is not much affected because packets can go through G1 to R and get back to the client through G2. Since the data is collected on the router side, it is hard to know when the clients disconnect from their original gateways, but we measured the time for the Ack packets to go through the same link as messages sent by clients after gateway switch. For C1 it is 0.13 second, and for C2 it is 0.1 second.

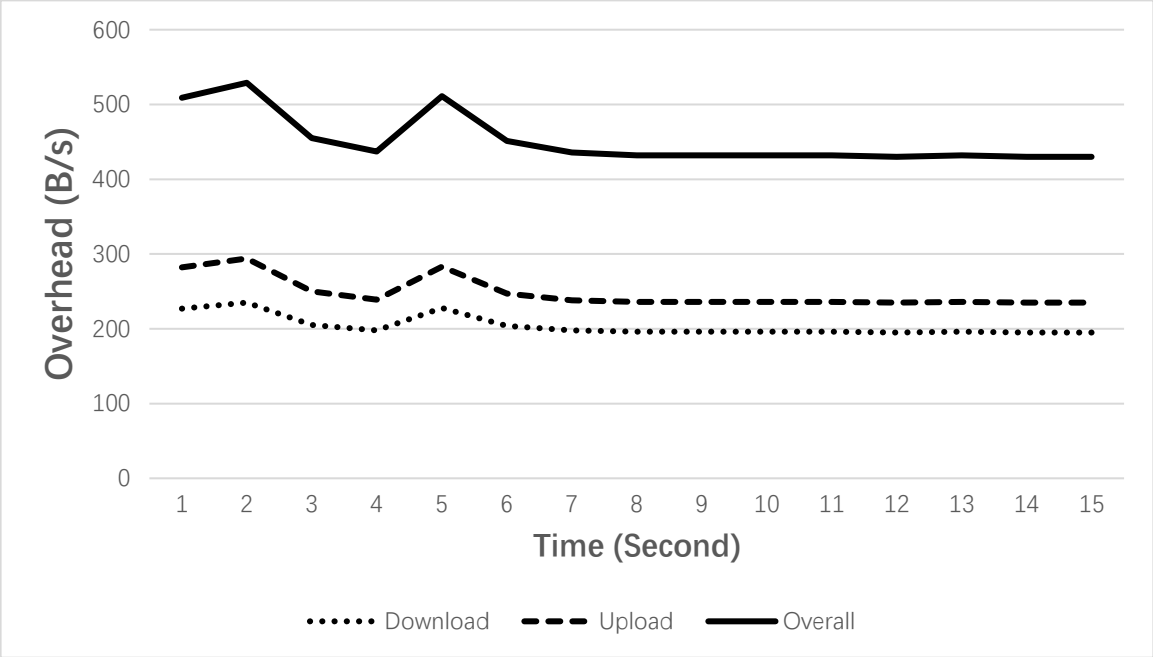

Figure 6.7: Overhead of the Gateway-switch Design with 2 Clients

Then we measured the overhead of the network. It is not a surprise that the overhead remains at the same level as before as more gateways do not add more overhead other than occasion gateway-switch messages. The download overhead is 204 B/s, the upload overhead is 248 B/s and overall overhead is 452 B/s.

Then we ran another test with the improved gateway-switch message design that clients will send gateway-switch message to gateways, and gateway will forward them to the Internet router. Like the last experiment, the data is collected on the router side.

The result is in Table 6.6. It is very interesting that although it is mostly similar to the last experiment, the gateway switch happens before the message is captured. The main reason is that when the client sends a gateway-switch message, it is received by the gateway, the gateway needs to analyze the payload, and generate another message and send to the router. This could take a few hundred milliseconds, and when the message is received by the router, some Ack packets may already be forwarded to the old gateway. The time between the client switches the gateway and the router set the correct route is still short though, which is 0.3 second for C1, and 0.08 seconds for C2.

Time	Packet	Time	Packet
0.03	C1 sends packet 45	5.65	C1 sends packet 101
0.03	Ack of packet 45	5.65	Ack of packet 101
0.04	C2 sends packet 40	5.66	C2 sends packet 96
0.04	Ack of packet 40	5.66	Ack of packet 96
0.13	C1 sends packet 46	5.76	C2 sends packet 97
0.13	Ack of packet 46	5.76	Ack of packet 97
0.14	C2 sends packet 41 through G1	5.85	C2 sends packet 98
0.14	Ack of packet 41 from G2	5.85	Ack of packet 98
0.24	C1 sends packet 47	5.87	C1 sends packet 102 103 from G2
0.24	Ack of packet 47	5.87	Ack of packet 102 103 from G1
0.24	C2 sends packet 42 through G1	5.93	G2 sends Gateway Switch message to R
0.24	Ack of packet 42 from G2	5.95	C1 sends packet 104 from G2
0.33	C1 sends packet 48	5.95	Ack of packet 104 from G2
0.33	Ack of packet 48 from G2	5.94	C2 sends packet 99
0.34	C2 sends packet 43 through G1	5.94	Ack of packet 99
0.34	Ack of packet 43 from G2		
0.38	G1 sends gateway-switch message to R		
0.43	C1 sends packet 49		
0.43	Ack of packet 49		
0.44	C2 sends packet 44 through G1		
0.44	Ack from G1		

Table 6.6: Behavior of Improved gateway-switch Design

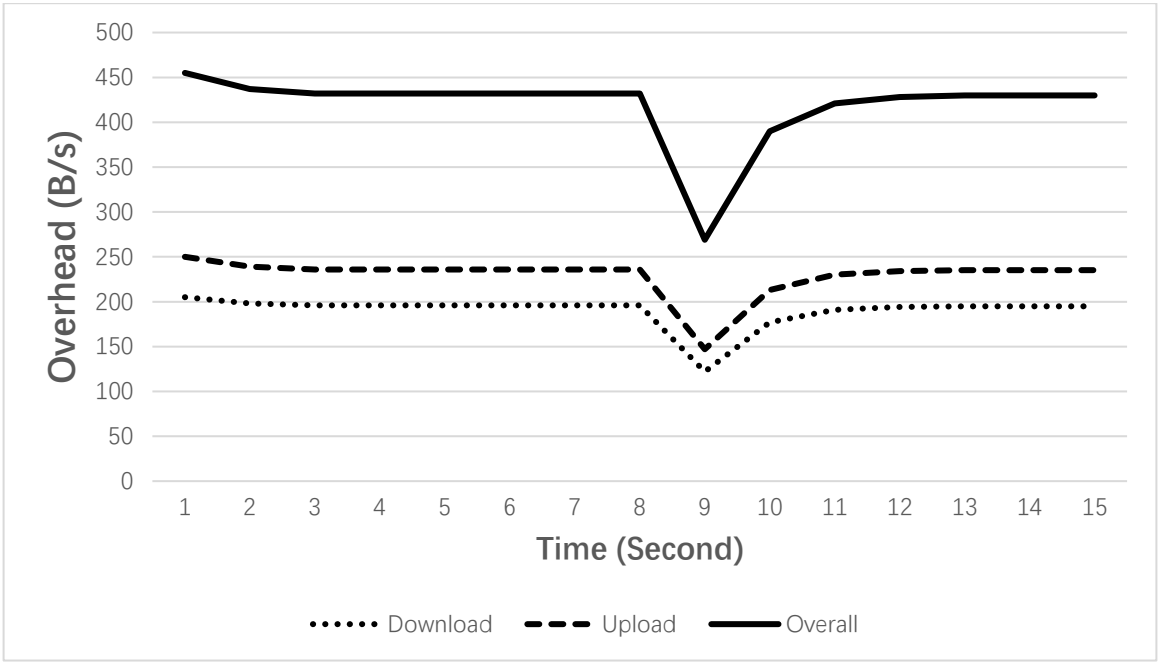

Figure 6.8: Overhead of Improved gateway-switch Design

The overhead remains the same as expected, Collisions may happen at 9 seconds and caused the sudden drop over overhead. The download overhead is 195 B/s, the upload overhead is 235 B/s, and the overall overhead is 430 B/s, discarding the data at 9 seconds.

CHAPTER 7

DISCUSSION

7.1 Initial State

Before running the program of either design, clients will lose TCP connection like in Figure 7.1.

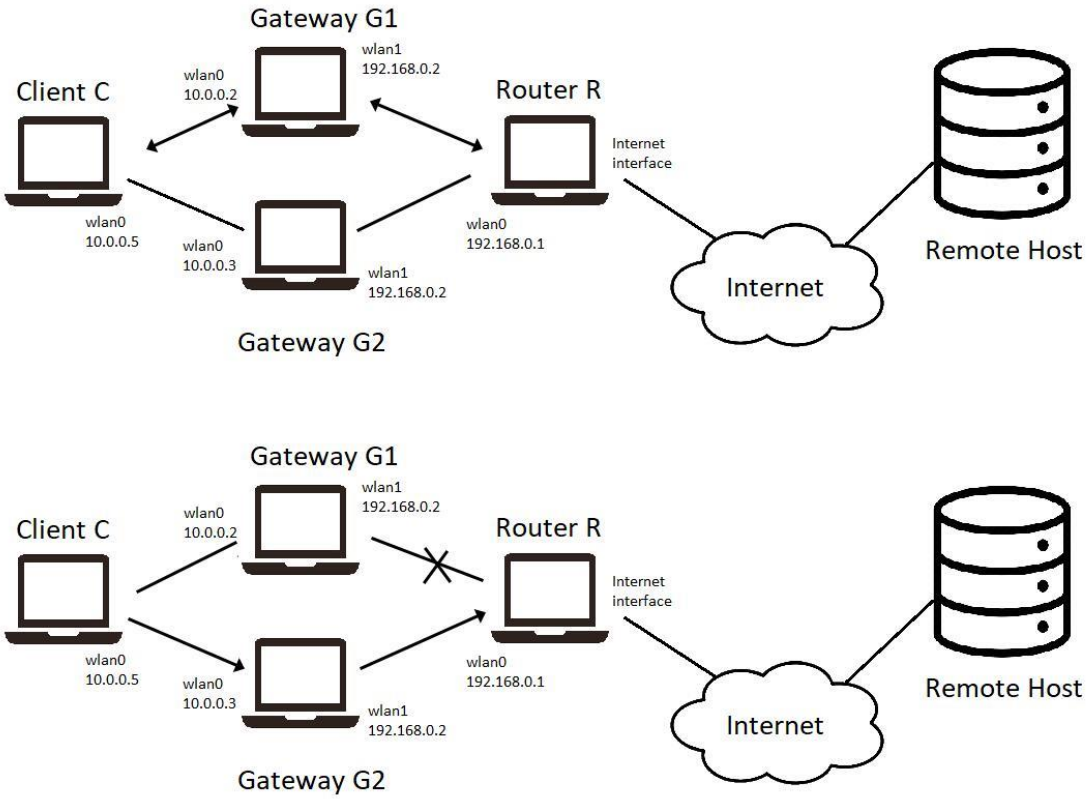

Figure 7.1: The Initial State

In the initial state, when a client C switches to gateway G2 due to the current gateway G1 losing connection to the Internet router R, R will still try to reach back to C from G1. This will terminate any current TCP connection from C to the Internet.

7.2 Heartbeat Message Design

Figure 7.2 shows how heartbeat design works.

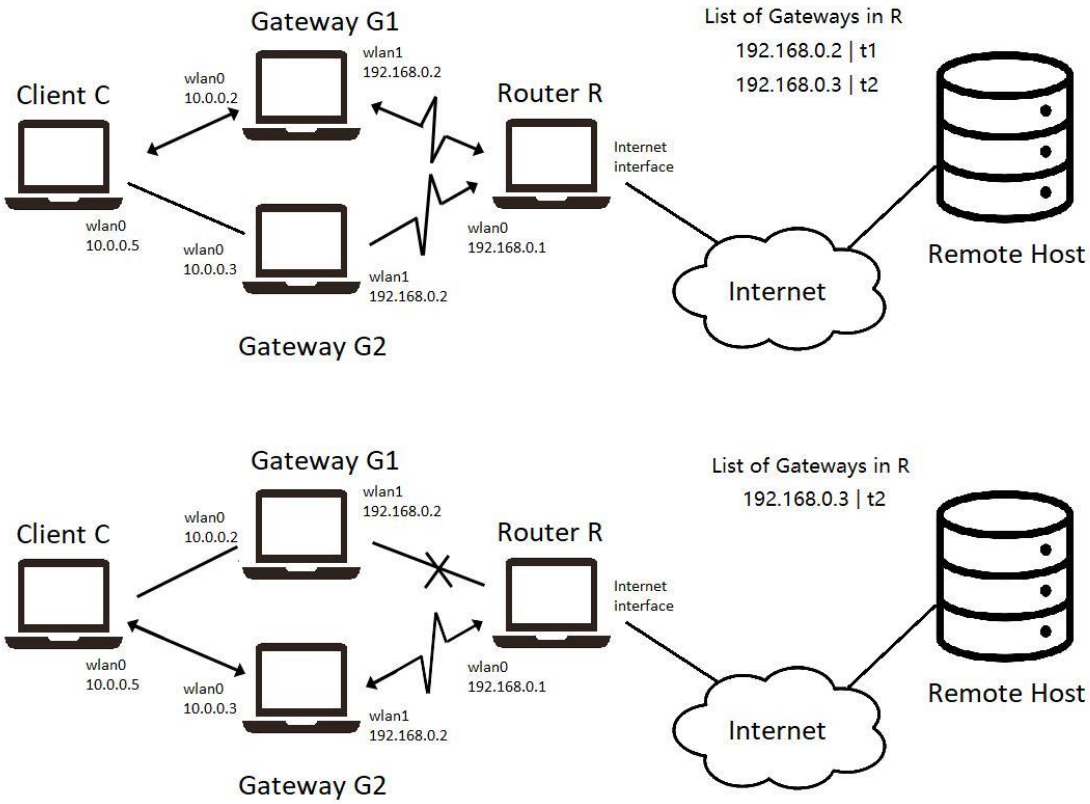

Figure 7.2: The Heartbeat Message Design

At first, gateway G1 and G2 are all sending heartbeat messages to the Internet router R. In R's memory the list containing all active gateways has two records right now: The current gateway G1, and an active but not used gateway G2. Shortly after G1 lose connection to R, the first record in the list will become invalid, and R will choose the next record in the list, and in this case G2, as the new gateway. Client C on the other hand, will also switch the gateway after the last HNA message from G1 expires. So now the data of client C goes through G2 and the TCP connection is still alive.

Based on the result of the test, the switch takes about 3.5 seconds, and the connection before and after the gateway switch is basically the same. The overhead running this design is relatively low and takes only about 1 to 1.4 KB/s of 120 KB/s total bandwidth depending on different configurations.

The improved design has basically the same performance as the prototype design. But the overhead is much lower, about 0.45 KB/s since the improved design does not create any new messages but use the ICMP packets sends by OLSR as heartbeat message. However, there are two weaknesses with the heartbeat message design.

First is that the time to switch the gateway is long. It takes about 3.4 seconds to switch to another gateway. Some TCP connection may timeout during the long wait. The second is that as we discussed before, this design is not working when the current gateway loses connection to the OLSR MANET but still connected to the Internet router, the ICMP message from the gateway will keep the Internet router to stick to it and do not change the route to OLSR MANET as shown in Figure 7.3. Also, if there are more than one gateway connected to the Internet router, only one of them will be used as gateways

to the OLSR MANET. This means that clients selecting other nodes as gateways will not be able to connect to the Internet.

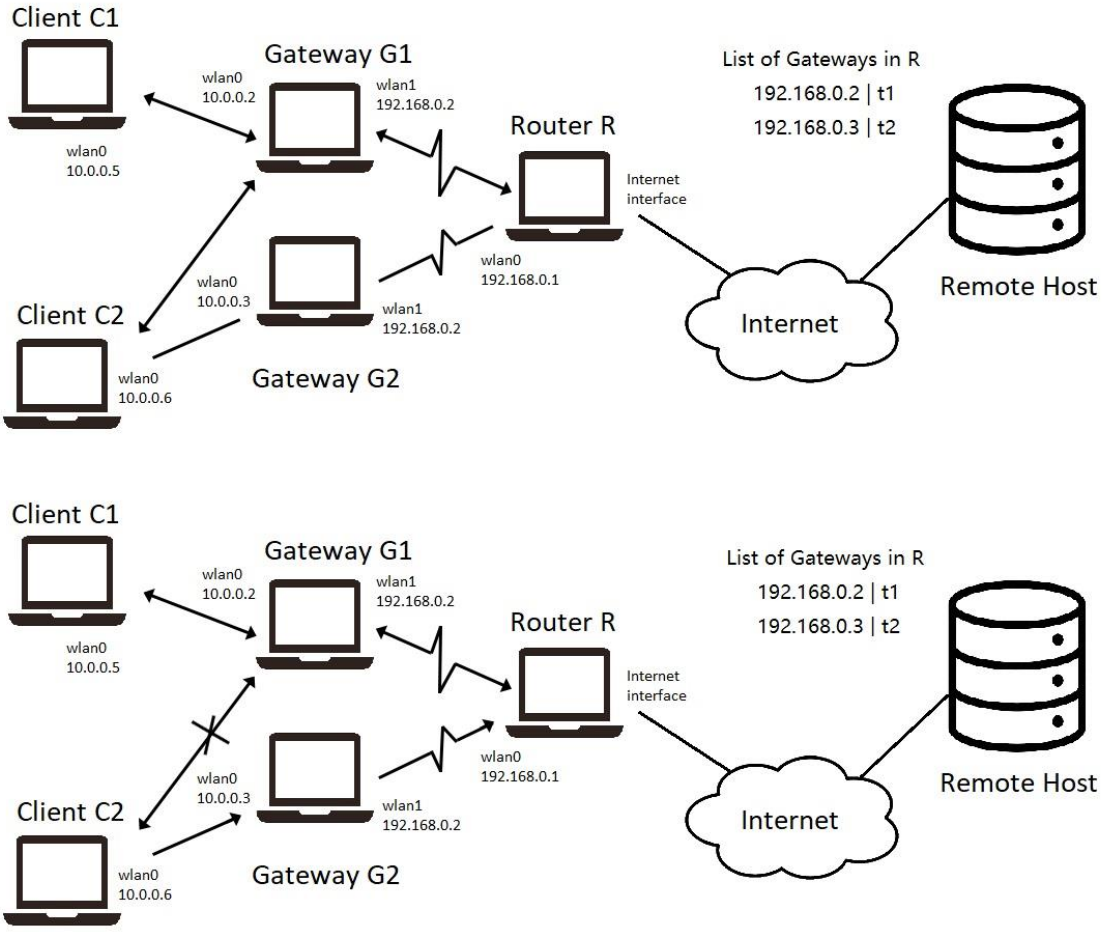

Figure 7.3: Problem with Heartbeat Message Design

In the beginning, client C1 and C2 all communicate with the remote host through gateway G1. But when C2 lose connection to G1 and switches to gateway G2, the Internet router R will not discover this change and remain using G1 and gateway back to

OLSR MANET. In this case, C1's TCP connection is not affected, but C2 will lose connection to the Internet.

7.3 Gateway-switch Message Design

The gateway-switch message design is here to solve this problem. The procedure is shown in Figure 7.4

Client C1 connects to the Internet through gateway G1, and when G1 is down, C1 will switch to gateway G2. After the switch, C1 will send out a gateway-switch message to the remote host, and as router R sniffs the gateway-switch message, it will change the value of key “10.0.0.5” from “192.168.0.2” to “192.168.0.3” in the dictionary. Then C will be able to continue its TCP connection on a new link.

We tested the gateway-switch message design to see if it performs better and solves the problem we talked above. The result is satisfying. The switch time is about 3 to 5 seconds, which is much lower than the heartbeat design. This is mainly because unlike the heartbeat message, which the Internet router switches only after the current gateway expires, the gateway switch message is sent to the Internet router in a very short time after gateway switch happens. The overhead is about the same as the improved heartbeat design since in gateway-switch message design, there is no other message sent periodically except the ICMP message generated by dynamic gateway plugin of OLSR. The gateway-switch message is only sent when there is a gateway switch happening.

The experiment ran on the network with 2 clients and 2 gateways is to test if this design can work under the situation shown in Figure 7.3. The procedure is as in Figure 7.5.

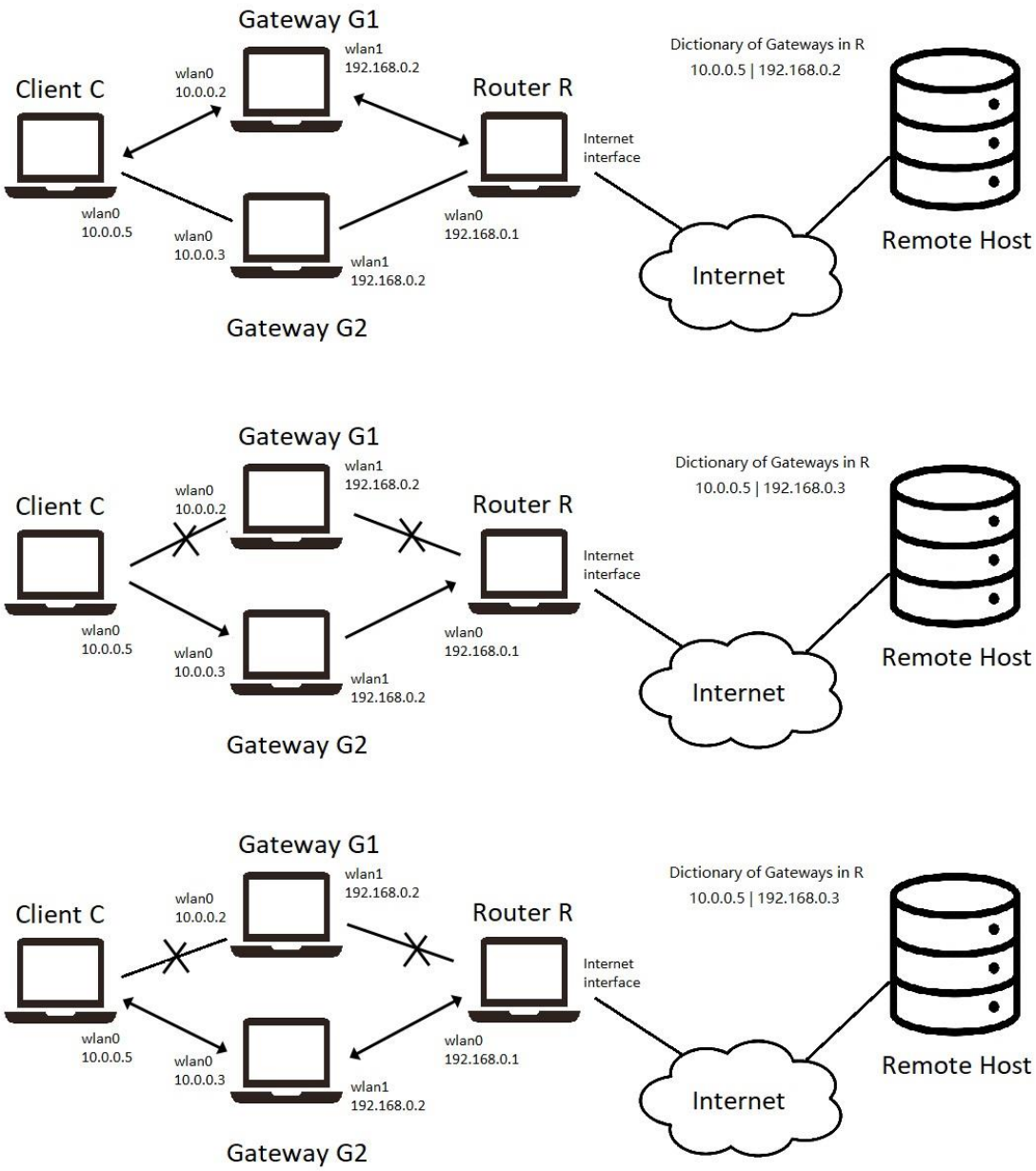

Figure 7.4: Procedure of Gateway-Switch Message Design

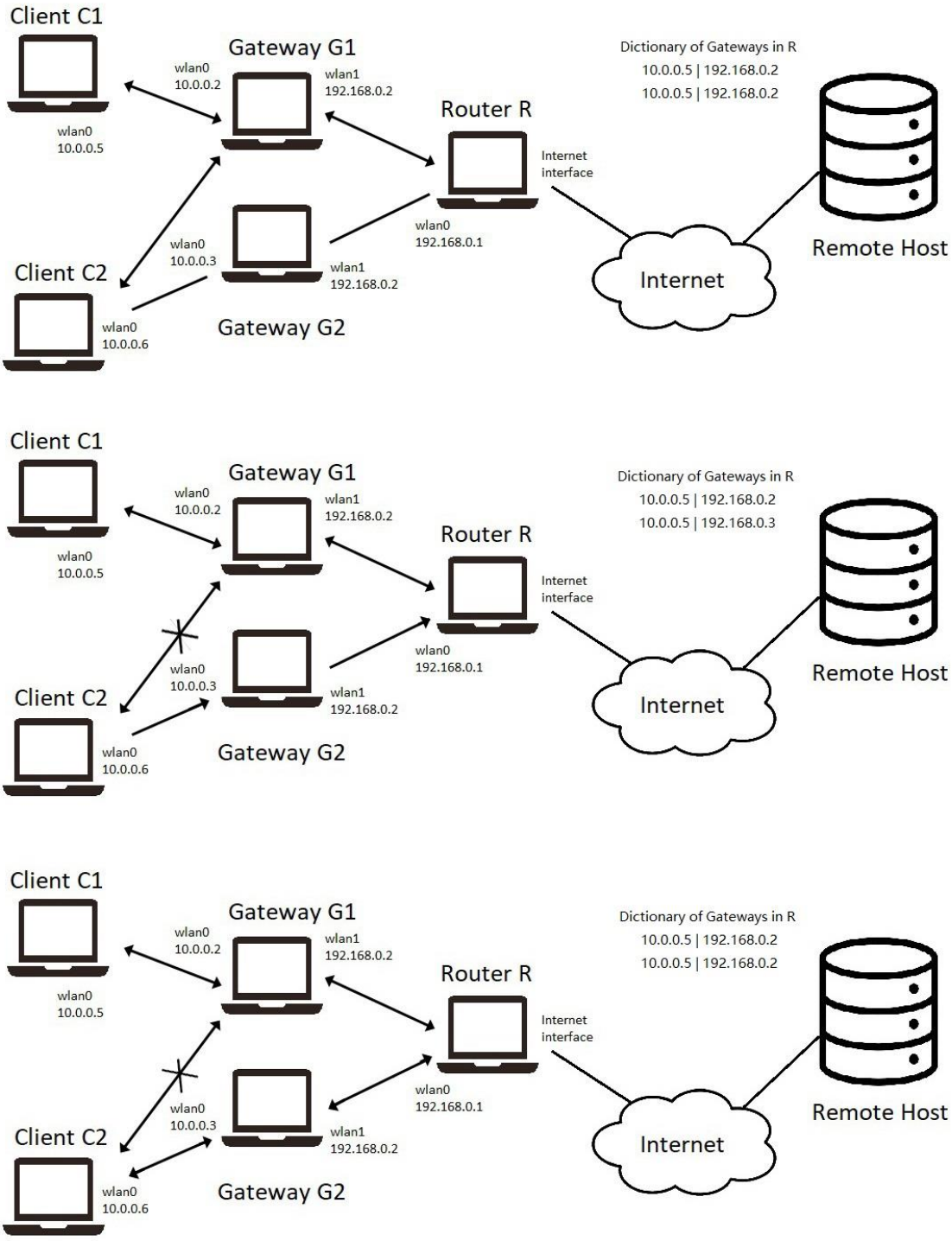

Figure 7.5: Gateway Switching with 2 Clients

In the beginning, router R reaches back to the two clients through gateway G1. When client C2 loses connection to G1, it will switch to gateway G2 and send a gateway-switch message. When router R receives the message, it will change the value of key "10.0.0.6" from "192.168.0.2" to "192.168.0.3" and change the route to C2. Thus, C1 and C2 will maintain 2 TCP connections on 2 different links.

Based on the result of the test, the switch takes about 4 seconds, and the throughput does not change after the gateway switching. The overhead is almost the same comparing to improved heart-beat design.

When it comes to multiple clients, the gateway-switch design works very well. The router reacts very quickly to the gateway switch and the TCP connection is only slightly affected.

CHAPTER 8

CONCLUSION AND SUMMARY

8.1 Conclusion

In conclusion, the gateway-switch message design can maintain continuous TCP connections between multiple clients in an OLSR MANET and the Internet host by helping the Internet router to change its routing table based on the state of OLSR MANET. The time of reconnecting is about 1.7 seconds, and most of the time is spent by the client to switch to a new gateway based on OLSR HNA message and TCP retransmission. The Internet router reacts very fast after the gateway switch and set the correct route in about 0.3 second, the overhead can be as low as 0.45 KB/s, and almost all packets are messages of OLSR protocol. The packets added by the design is minimum because it is only sent when needed, not at a fixed frequency.

8.2 Future Improvement

The first thing we can improve is stability. During the test, disconnection still happens when there is no gateway switching. Some are because of unstable wireless network, but sometimes disconnection happens because the TCP connection takes most of the bandwidth, and message packets to be sent to gateways and routers are either late with a latency of more than 1 second or being lost during the transmission. Limiting the

Throughput can be a good solution to this problem, and we can also set the gateway and routers to send an acknowledgment packet back to the sender to indicate they received it.

We can also improve the time of switching by integrating the program into the OLSR protocol. Currently the gateway switching program is running independently from OLSR, and the HNA message sent every 2 seconds means the time of switching will not drop below 2 seconds. By integrating into OLSR, we can have the program to send HNA message faster and thus improve the time cost.

CHAPTER 9

REFERENCES

[1] https://news.netcraft.com/archives/2019/03/28/march-2019-web-server-survey.html

[2] M. Weyrich and C. Ebert, “Reference architectures for the internet of things,” IEEE Software, vol. 33, no. 1, pp. 112–116, 2016.

[3] https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016

[4] https://www.researchgate.net/profile/Piet_Demeester/publication/235941882_An_overview_of_mobile_ad_hoc_networks_Applications_and_challenges/links/0c960528522f51ad75000000.pdf

[5] Freebersyser and B. Leiner, "A DoD Perspective on Mobile Ad Hoc Networks," Ad Hoc Networking, ed. C. E. Perkins, Addison-Wesley, 2001, pp. 29-51.

[6] http://www.olsr.org/mediawiki/index.php/Main_Page

[7] "A distance routing effect algorithm for mobility (DREAM)" in Basagni, Stefano; Imrich Chlamtac; Violet R. Syrotiuk; Barry A. Woodward (1998). *International Conference on Mobile Computing and Networking Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. New York: ACM Press. pp. 76–84. doi:10.1145/288235.288254. ISBN 978-1-58113-035-5.

[8] https://www.open-mesh.org/projects/open-mesh/wiki

[9] https://tools.ietf.org/html/rfc3561

[10] https://tools.ietf.org/html/rfc4728

[11] https://tools.ietf.org/html/rfc3626

[12] https://www.multipath-tcp.org/

[13] https://scapy.net/