工學碩士學位論文

단방향 IP 성능 측정 도구의 설계 및 구현

Design and Implementation of One-way IP Performance Measurement Tool

2001年 2月

서울大學校 大學院

電氣컴퓨터工學部

鄭 載 勳

단방향 IP 성능 측정 도구의 설계 및 구현

Design and Implementation of One-way IP Performance Measurement Tool

指導敎授 崔 陽 熙

이 論文을 工學碩士 學位論文으로 提出함

2000年 10月

서울大學校 大學院
電氣컴퓨터工學部
鄭 載 勳

李憲洙의 工學碩士 學位論文을 認准함

2000年 12月

委 員 長 _____ 印

副委員長 _____ 印

委 員 _____ 印

Design and Implementation of
One-way IP Performance Measurement Tool

Jaehoon Jeong
School of Computer Science and Engineering
Seoul National University, Seoul, Korea
paul@mmlab.snu.ac.kr

Abstract

In this thesis, we propose an architecture of measurement system which can measure IEFT's IP Performance Metrics(IPPM) such as one-way delay, variation of one-way delay and loss rate in the Internet, which is asymmetric. We also evaluate the measurement results. The synchronization among measurement systems is very important in one-way delay measurement. The proposed architecture uses the Global Positioning System(GPS) to synchronize the measurement systems and provides the precision up to micro-second. To improve the accuracy of measuring one-way delay, the proposed system stamps the time information in the payload which is one of fields in the Ethernet frame just before transmitting the Ethernet frame to the network interface card as well as just after receiving the Ethernet frame from the network interface card. In this thesis, we have tested the proposed system in the Internet and have obtained one-way IP Performance Metrics. Through the result of this measurement, we also present need of one-way measurement.

Key words: IETF's IP Performance Metrics(IPPM), One-way Delay, Global Positioning System(GPS), Active Measurement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The fact that the Internet is asymmetric[1] requires better one-way measurement of the Internet. The IETF's IP Performance Metrics(IPPM) Working Group suggested one-way metrics and architecture for one-way measurement[2]. Metrics are one-way delay, delay variation, packet loss rate, loss pattern, etc[3-5]. One-way measurement is a kind of active measurement where measurement packet is injected in the path to measure and how for the packet to be served is observed. With measurement data through active measurement, the network management can be performed effectively. For example, we can conjecture that some problems happened in the network if we had observed the network for a long time and had found that one-way delay increased much more than in ordinary times. Through the analysis of measurement result that we have obtained through long observation of network, we can find problem of the network(i.e., bottleneck path). We can solve the problem by relocating resources, increasing link capacity, changing network configuration(i.e., routing configuration) and upgrade routers. In the result, we can improve the performance of the entire network.

The system for the active measurement that provides us with the necessary functions for effective network management must provide operator with user

interface with which operator can control the system easily and efficiently. It must also be able to measure the network in stable state for a long time and have functions of self-troubleshooting ,which are finding problem in the measurement system and solving the problem automatically without operator.

In this thesis, we suggest an architecture for measurement system(AMT: Active Measurement Tool) that can perform one-way measurement efficiently; AMT has been designed and implemented so that it may measure one-way metrics stably for a long time and be easy to be expanded. We present the analysis of results that we have measured in the Internet with AMT. This thesis is organized as follows. Chapter 2 introduces Internet Performance Measurement. Chapter 3 presents related work. In Chapter 4, the architecture of suggested system, components of the system, procedures related to measurement and state transition of each component of AMT are explained. Chapter 5 presents performance evaluation. Finally, in Chapter 6, we conclude the thesis and present future work.

Chapter 2

Internet Performance Measurement

There are two kinds of measurement to measure the Internet Performance. One is Passive Measurement and the other is Active Measurement. Figure 1 shows a infrastructure for Internet Performance Measurement.

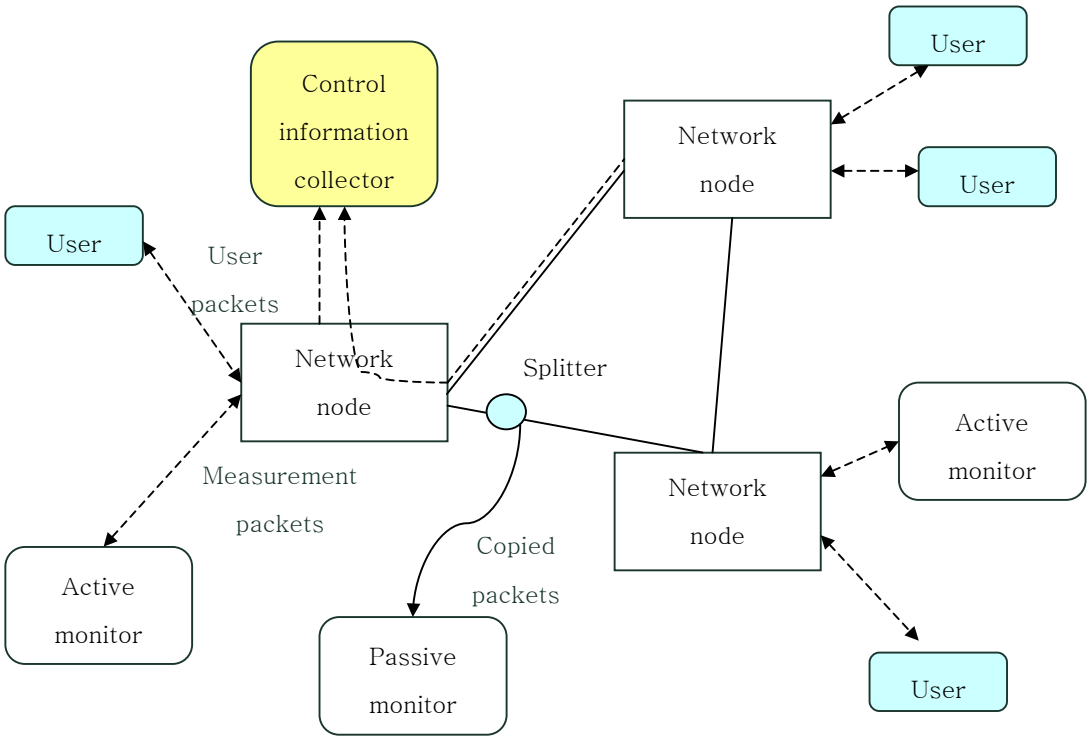

Figure 1. Infrastructure for Internet Performance Measurement

2.1 Passive Measurement(PM)

PM is a way to collect traffic information at links between network nodes with Passive monitor as shown in Figure 1 and analyze the information[16]. With PM, We can visualize network traffic in real-time and analyze the measurement data in offline in use of various analyzer tools. As example tool, there is tcpdump. By analyzing traffic that we collect, we can grasp characteristics of traffic(e.g., bit rate) in respect of user, time, protocol, and application. Through this analysis, we can see volume and pattern of traffic flowing in the Internet Backbone and perform network management efficiently in that we find problem of network and can take step in solving the problem.

2.2 Active Measurement(AM)

AM is a way to inject measurement packets in network with Active monitor as shown as in Figure 1 and observe how the packets are served[9]. With AM, We can visualize network characteristics presented in performance metrics. As example tools, there are ping and traceroute. Performance Metrics are as follows; (a) RTT, (b) one-way delay (c) delay variation (d) packet loss rate, (e) loss pattern, (f) path that packet is transferred, etc[3,4,5]. By analyzing the Performance Metrics that have been obtained through AM between two Active monitors as showed in Figure 1, we can manage network efficiently in that we diagnose the network and can take step in solving problems through resource relocation, increase of link capacity and upgrade of routers.

Chapter 3

Related Work

Many measurement systems were implemented for active measurement. We introduce two representatives among the systems; (a) Skitter and (b) Surveyor.

3.1 Skitter

Skitter is a measurement system that Cooperative Association for Internet Data Analysis(CAIDA) Group have implemented[7]. Skitter was made for analysis of Internet's topology and performance. It injects measurement packets in Internet and observes how the packets are served. Main functions are as follows; (a) Measurement of Forward IP Path, (b) Measurement of RTT, (c) Trace of Routing Change, and (d) Visualization of Network Topology. Skitter provides users with easy and convenient user interface but has a big demerit that it can not measure one-way metrics.

3.2 Surveyor

Surveyor is a measurement system that Advanced Network & Services Group has implemented that can measure one-way metrics[8,9]. The one-way metrics are based on IETF's IPPM. Surveyor consists of two systems; (a) Measurement System and (b) Central Control System. Two systems use One-Way Delay and Packet loss protocol (OWDP)[10,11]. Measurement Systems are synchronized with one another by GPS and perform the actual measurement. Central Control System controls Measurement Systems and gathers measurement data from the Measurement Systems. To improve the accuracy of one-way measurement, Surveyor stamps the time information in Ethernet device driver. It is one of the most popular systems for one-way measurement.

Chapter 4

Active Measurement Tool(AMT)

AMT is an infrastructure that can measure various one-way metrics suggested by IETF's IPPM Working Group. AMT is a PC-based system that uses FreeBSD UNIX and MySQL for operating system and database management system, respectively [12].

4.1 Consideration for Implementation of One-way Measurement

4.1.1 How to synchronize Measurement Systems

There is no need to synchronize measurement systems in order to measure RTTs which are known to be two-way delays. However, when it comes to measurement of one-way delay, we should synchronize measurement systems. Figure 2 shows how to synchronize systems by using GPS satellites. Through GPS satellites, the exact time information can be gathered, which results in maintaining system time

correctly and stably. Hardwares that are used to receive time information from GPS are as follows; Oncore Remote Antenna and Oncore GPS UT Receiver which are the products of Motorola [13]. Network Time Protocol(NTP) Daemon(i.e., ntpd[14]) modifies the kernel time with time information received from GPS. The time information encoded in Pulse Per Second (PPS) format is put into both serial port and parallel. Device driver of each port transforms the PPS into binary format and provides ntpd with the time information formatted as binary. The ntpd updates the kernel time periodically with the time information formatted as binary. In this mechanism, all measurement systems are synchronized with GPS.

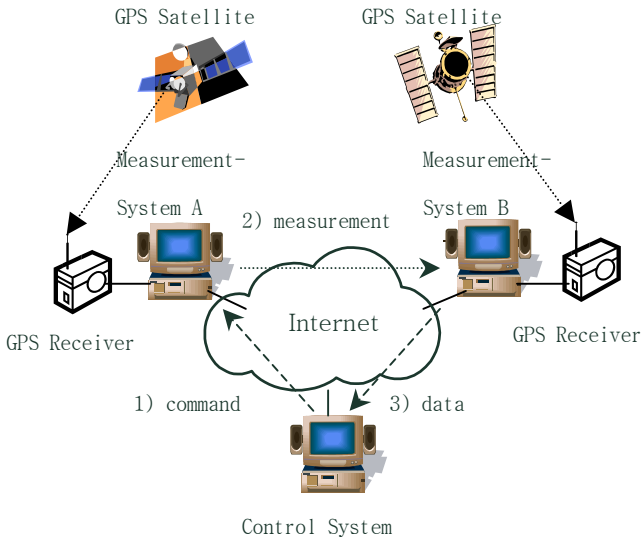

Figure 2. Synchronization among Measurement Systems and Measurement Procedure

4.1.2 Timestamp

To improve the accuracy of measuring one-way delay, the measurement system has to stamp the time information in the payload which is one of fields in the Ethernet frame just before transmitting the Ethernet frame to the network interface card as well as just after receiving the Ethernet frame from the network interface card. In this way, we are capable of reducing the delays occurred through the protocol stack at end hosts. Figure 3 describes the time-stamping procedure[9,15].

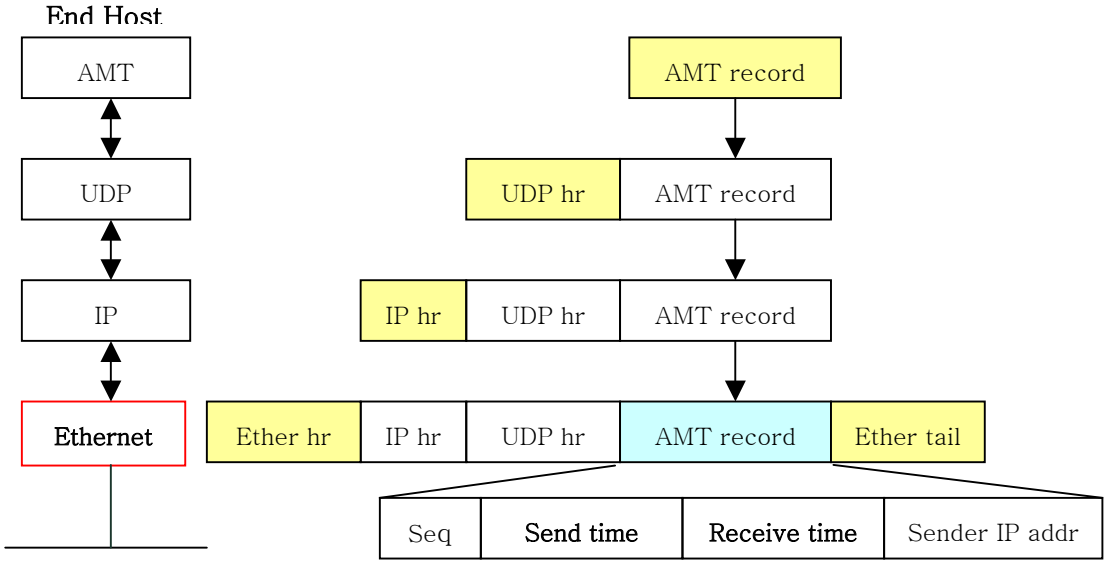

Figure 3. Time-stamping Procedure

4.2 Architecture of AMT System(AMT)

AMT consists of two kinds of systems ; (a) Control System(CS) and (b) Measurement System(MS). While MS performs one-way measurement, CS controls and manages the MS's as well. Figure 4 describes the architecture of AMT.

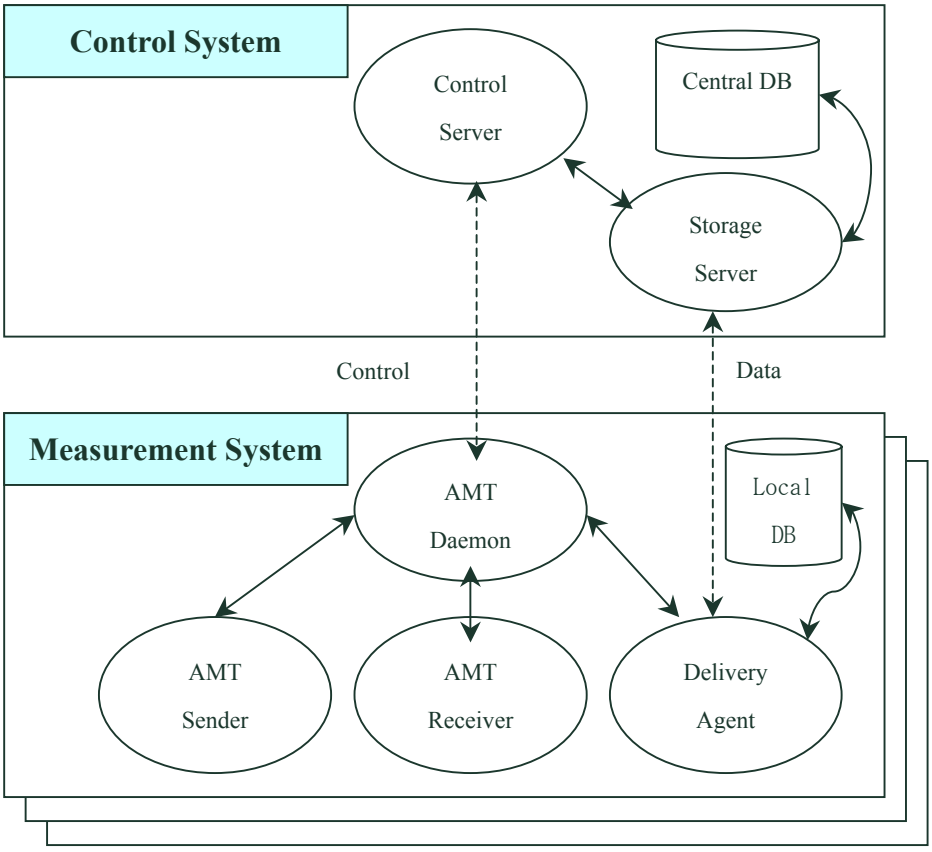

Figure 4. Architecture of AMT System(AMT)

4.2.1 Control System(CS)

CS, main system of AMT, receives the commands from operator and lets operator be able to control MS's by control shell shown as Figure 5 so that one-way measurements may be performed. Figure 5 shows when Control Shell has started

and Figure 6 shows a function of help as an example. CS has two processes; (a) Control Server(CSV) and (b) Storage Server(SSV).

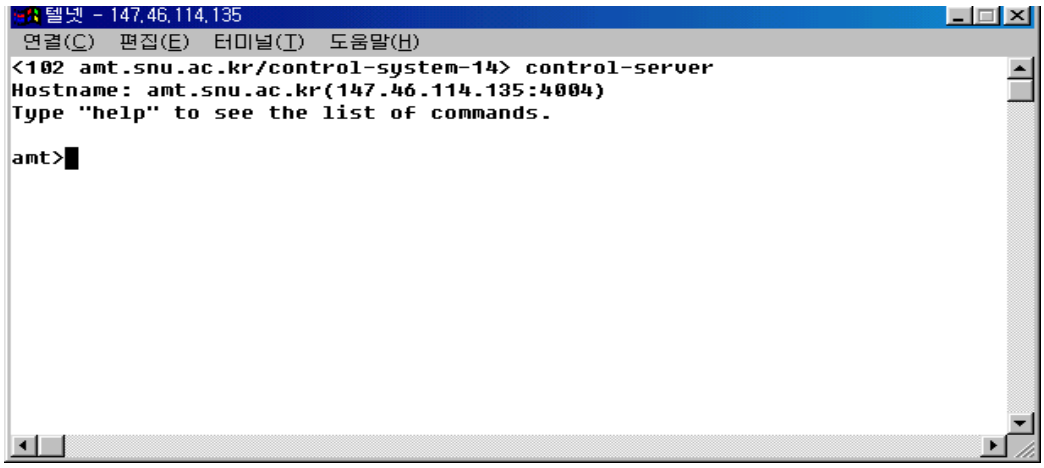

Figure 5. Control Shell

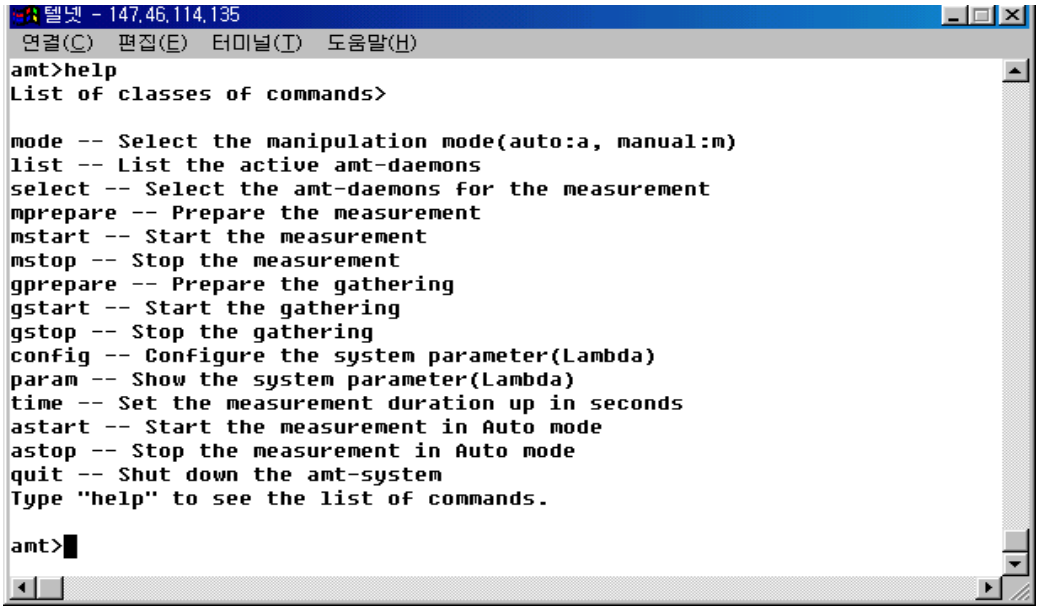

Figure 6. Function of Help provided by Control Shell

1) Control Server(CSV)

CSV receives commands from operator, parses the commands, and then processes the commands. CSV consists of two threads; (a) Main Thread(MT) and

(b) User Interface Thread(UT). UT receives the commands from operator and delivers them to MT. MT parses the commands that have been delivered by UT and processes them by the appropriate function. Figure 7 shows the process of executing a command from operator.

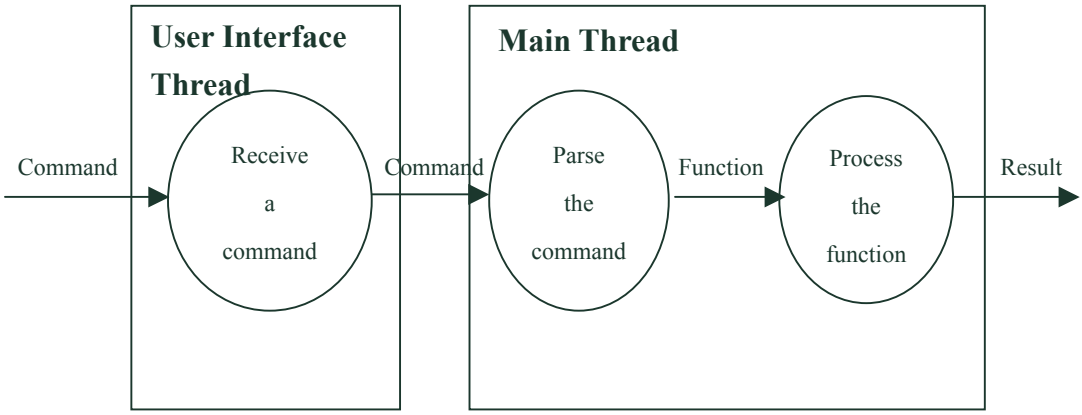

Figure 7. Structure of Control Server(CSV)

2) Storage Server(SSV)

SSV collects measurement data from Local database of each MS after the measurement and stores the data in the Central database. It is forked by CSV when preparing the collection. The collection is performed in the aid of Delivery Agent(DA) in each MS.

4.2.2 Measurement System(MS)

MS has four processes; (a) AMT Daemon(AMTD), (b) AMT Sender(AMTS), (c) AMT Receiver(AMTR) and (d) Delivery Agent(DA).

1) AMT Daemon(AMTD)

After AMTD as main process of MS first registers itself in CS, it receives all the control messages from CSV, processes them and sends the result to CSV. For example, when CSV sends the measurement preparation message to the registered

AMTD in each MS, AMTD receives the message to prepare measurement. It forks AMT Sender(AMTS) and AMT Receiver(AMTR) which will perform actual measurement. All the control messages from CSV to AMTS or AMTR in an MS are sent to AMTD in the MS. The reason that we designed AMT system for all the control message messages between CSV and AMTS or AMTR to go via AMTD is that we tried to make AMTS and AMTR be light-weighted processes that can run in good order for long. Figure 8 shows the structure of AMTD.

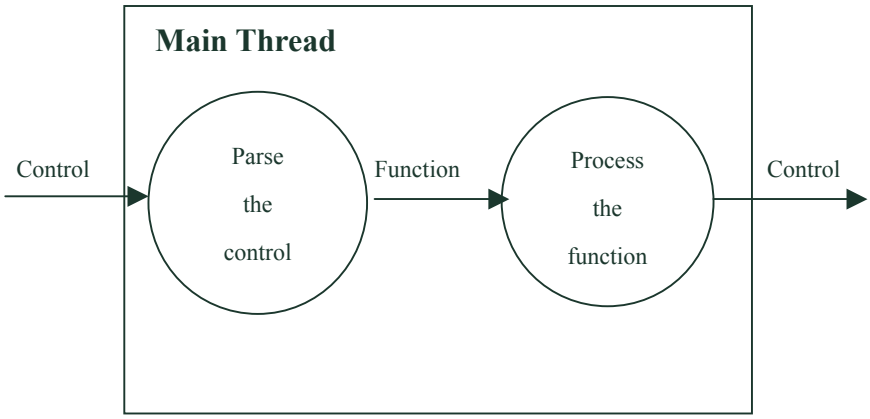

Figure 8. Structure of AMT Daemon(AMTD)

2) AMT Sender(AMTS)

AMTS is forked by AMTD when CS starts measurement. After AMTS receives a measurement start message, it generates measurement packets. The packets are generated in Poisson process by a pseudo-random number generator. AMTS sends every packet to all the AMTRs which are joining in measurement.

3) AMT receiver(AMTR)

AMTR is forked by AMTD when CS starts measurement. After AMTR receives a measurement start message, it opens a local database file to be ready to receive measurement packets. When it receives a measurement packet, it stores the format like Figure 9 in its local database file.

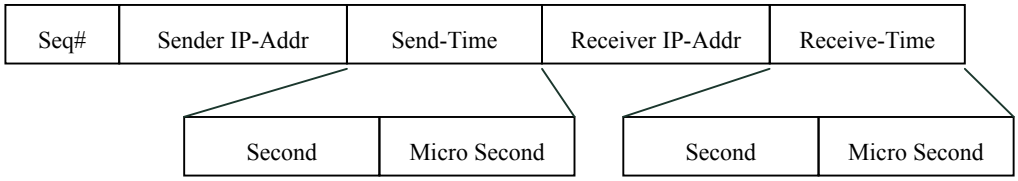

Figure 9. Record Format stored in Local Database(DB)

'Seq#' is 4-byte sequential number field. 'Sender IP-Addr' is 4-byte IP address field of AMT sender that sent the packet. 'Receiver IP-Addr' is also 4-byte IP address field of AMT receiver that received the packet. 'Send-Time' is 8-byte timestamp field where the packet was timestamped in Ethernet device driver just before being sent into network interface card. The type of this field is struct timeval { u_long tv_sec; u_long tv_usec }. 'Receive-Time' is also 8-byte timestamp field where the packet was timestamped in Ethernet device driver just after being received from network interface card.

4) Delivery Agent(DA)

DA is forked by AMTD when CS gathers measurement data from each MS. After DA receives a gather start message, it opens local database file and delivers the measurement data stored in the file to SSV in CS.

4.3 One-way Measurement

4.3.1 Procedure of Initialization

The procedure of initialization for measuring one-way delay is described as shown in Figure 10.

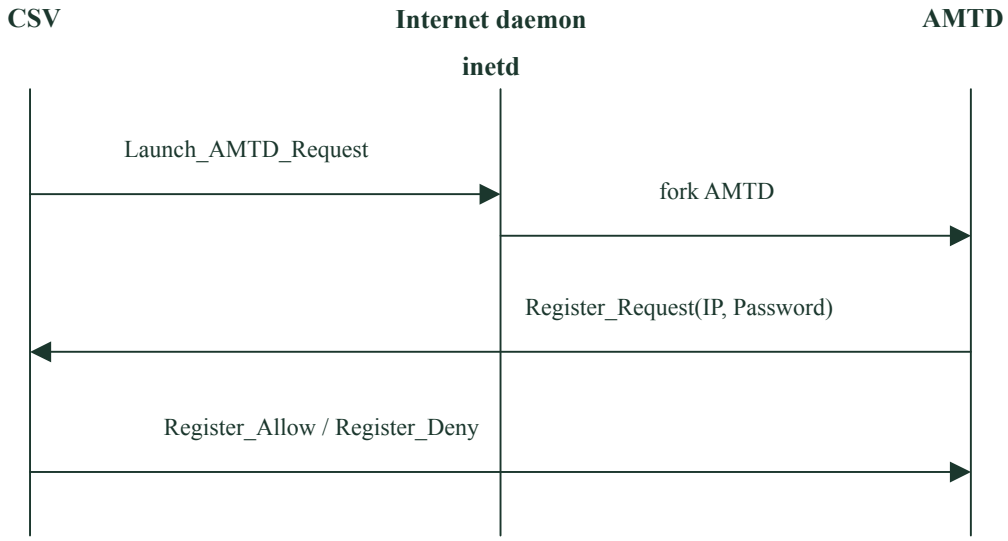

Figure 10. Procedure of Initialization

CSV launches AMTD through Internet Daemon(i.e., inetd) in MS that will take part in measurement. CSV sends inetd a 'Launch_AMTD_Request' message indicating that inetd has to fork an AMTD. Just after AMTD has been forked, AMTD tried to register itself in CSV in a 'Register_Request' message with its IP address and password. If CSV identifies AMTD with IP address and password that AMTD is valid, CSV registers AMTD in a list of MS that will take part in measurement and sends AMTD a 'Register_Allow' message. Otherwise, CSV sends AMTD a 'Register_Deny' message.

4.3.2 Procedure of Measurement

The procedure of measuring one-way delay is described as shown in Figure 11.

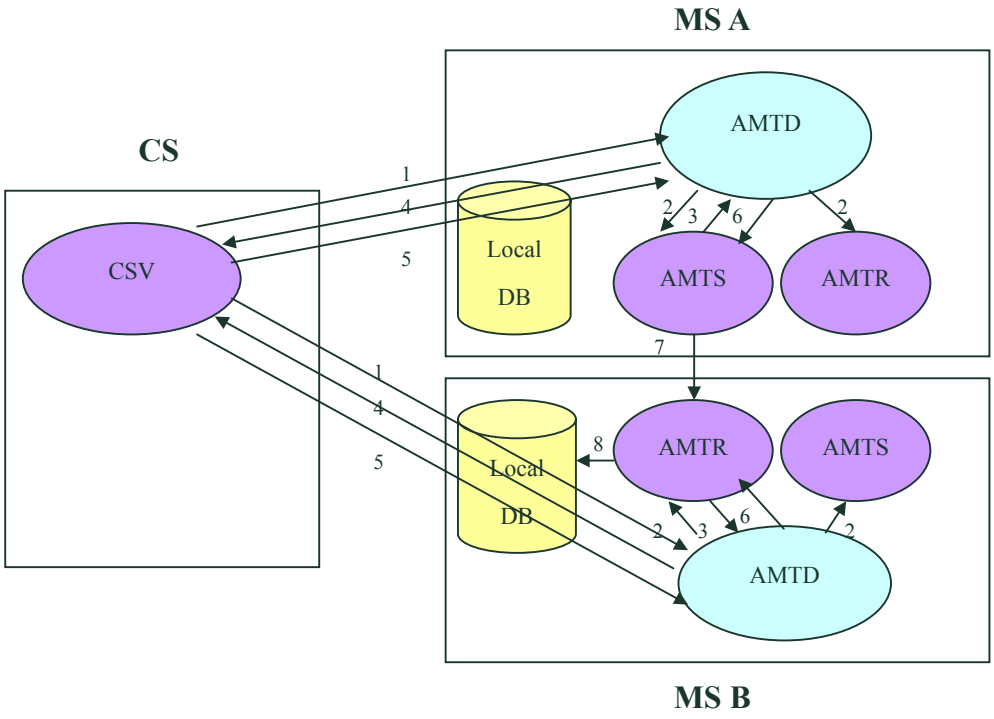

Figure 11. Procedure of Measurement

Step 1. Initialization of AMTD for measurement

CSV sends all the AMTDs that take part in measurement a 'measure-ready' message indicating that they have to prepare a measurement. The control packet including the message provides them with a system parameter(i.e., lambda value for Poisson process) and a list of IP addresses of all the participating AMTDs together with the message.

Step 2. Fork of measurement processes

When AMTD of MS receives the 'measure-ready' message, it makes control channels that will be used to communicate with AMTS and AMTR that are implemented in UNIX domain stream socket. It forks AMTS and AMTR and sends the 'measure-ready' message to them.

Step 3. Establishment of control channel

Just after AMTS and AMTR have been forked by AMTD, they establish control channel that is used to communicate with AMTD. AMTS and AMTR obtain the system parameter of the list of IP addresses from control packet including the 'measure-ready' message. When AMTS and AMTR are ready to measure, they report the readiness to AMTD through control channel implemented in UNIX domain stream socket.

Step 4. Confirmation about readiness from AMTD

When AMTD receives the report from AMTS and AMTR, AMTD sends CSV a 'measure-ready-ack' message indicating that MS is ready to measure.

Step 5. Start of measurement

When CSV has received the report from all participating AMTDs, CSV sends them a 'measure-start' message indicating that they have to start measurement.

Step 6. Start of actual measurement

When AMTD receives the 'measure-start' message, it sends the message to its child processes; AMTS and AMTR.

Step 7. Injection of measurement packets

AMTS generates measurement packets in Poisson process at random. The packets are sent to all participating AMTRs except AMTR in the same host through UDP socket.

Step 8. Storing of measurement records

When AMTR receives a measurement packet, it stores into the local database a record that consists of the following fields; (a) Sequence number, (b) Sender IP address, (c) Send-time, (d) Receiver IP address, and (e) Receive-time. The format of the local database file is binary to reduce the size of record.

4.3.3 Procedure of Gathering of Measurement Data

The procedure of measuring one-way delay is described as shown in Figure 12.

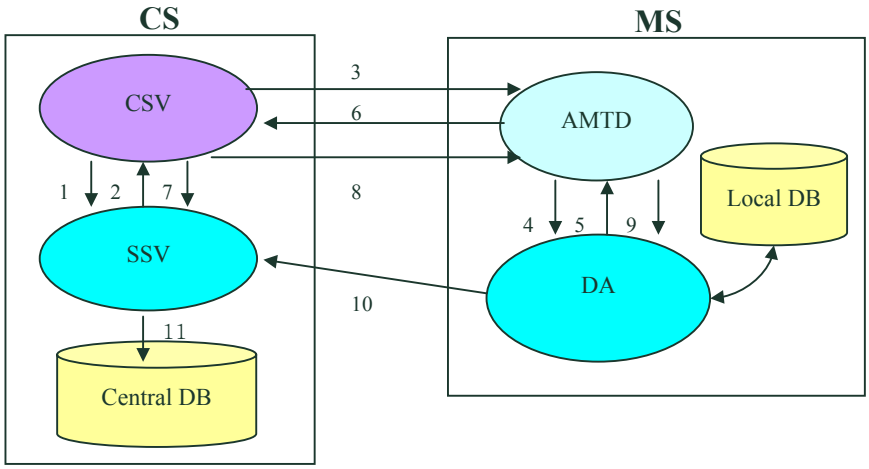

Figure 12. Procedure of Gathering of Measurement Data

Step 1. Initialization of SSV for gathering

CSV makes control channel that is used to communicate with SSV that is implemented in UNIX domain socket. It forks SSV and sends SSV a 'gather-ready' message indicating that SSV has to be prepared to gather measurement data from all participating MS's.

Step 2. Confirmation about readiness from SSV

Just after SSV establishes control channel with CSV, SSV sends CSV a 'gather-ready-ack' message indicating that SSV is prepared to gather measurement data.

Step 3. Initialization of AMTD for gathering

CSV sends all the AMTDs one by one that have taken part in measurement a 'gather-ready' message indicating that they have to prepare the delivery of measurement data stored in their local database file. The control packet including the message provides IP address of SSV for AMTDs.

Step 4. Fork of DA

AMTD makes control channel that is used to communicate with DA. It forks DA and sends DA a ‘gather-ready’ message indicating that DA has to be prepared to deliver measurement data to SSV.

Step 5. Establishment of control channel

Just after DA establishes control channel with AMTD, DA sends AMTD a ‘gather-ready-ack’ message indicating that DA is prepared to deliver measurement data to SSV.

Step 6. Confirmation about readiness from AMTD

When AMTD receives the report from DA, AMTD sends CSV a ‘gather-ready-ack’ message indicating that MS is ready to measure.

Step 7. Start of SSV

When CSV has received the report from AMTD, CSV sends SSV a ‘gather-start’ message indicating that SSV has to start measurement. When SSV receives the message, it opens its central database and waits for DA in an MS to connect.

Step 8. Start of gathering

CSV sends AMTD a ‘gather-start’ message indicating that MS has to deliver measurement data to SSV.

Step 9. Start of actual gathering

When AMTD receives the ‘gather-start’ message, AMTD sends DA the message. DA tries to establish a data channel that will be used to transfer measurement data to SSV. The data channel is TCP.

Step 10. Transmission of measurement data

DA reads measurement data from its local database and delivers the data to SSV.

Step 11. Storing of measurement data

Whenever SSV receives a data packet that has measurement data, it stores the fields of the packet in its central database.

The procedure from Step 3 to Step 11 is repeated until SSV completes collection measurement data from all participating MS's.

4.3.4 Procedure of Visualization of Measurement Result

Figure 13 shows AMT Visualizer(AMTV). AMTV receives the following inputs; (a) Sender IP, (b) Receiver IP, and (c) Date. The meaning of combination of three input fields is that MS with 'Sender IP' address generated measurement packets and sent them to MS with 'Receiver IP' address on 'Date'. We present an example from Figure 13. The meaning of combination of input fields is that we want to get the result from measurement packets that MS with IP address 147.46.14.69 sent MS with IP address 203.232.127.20 on November 28, 2000. The output of the result is graphs of one-way delay and loss rate during a day.

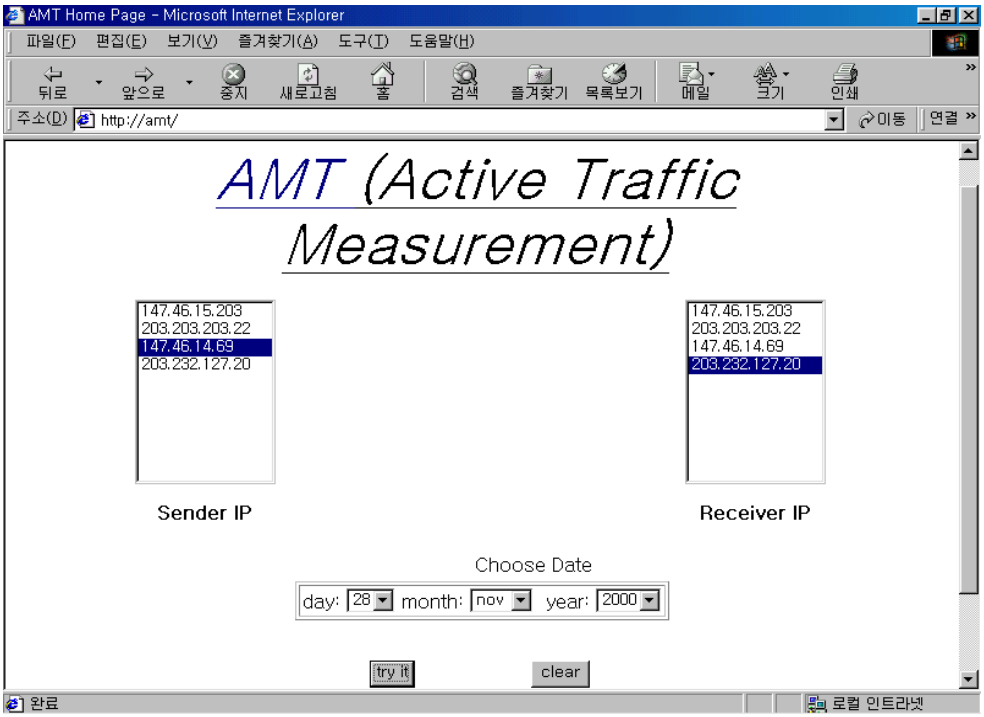

Figure 13. AMT Visualizer(AMTV)

Figure 14 shows the result of the query of Figure 13. The result is one-way delay on November 20, 2000.

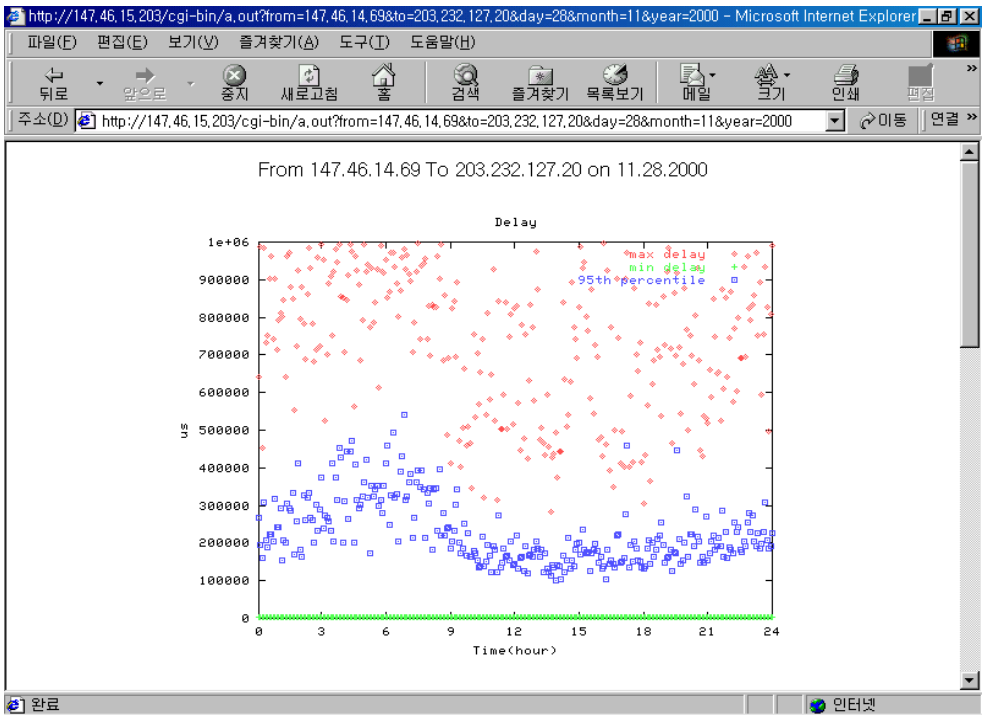

Figure 14. Result of a Query at AMTV

Figure 15 describes the procedure of visualization. When operator sends a query requesting measurement result between two end hosts on a specific day with AMTV, the query is transferred to CGI program called as Measurement-Analysis Agent(MAA) via HTTP Daemon. MAA searches the result of the query in Central Database(DB) and returns the result to AMTV via HTTP Daemon.

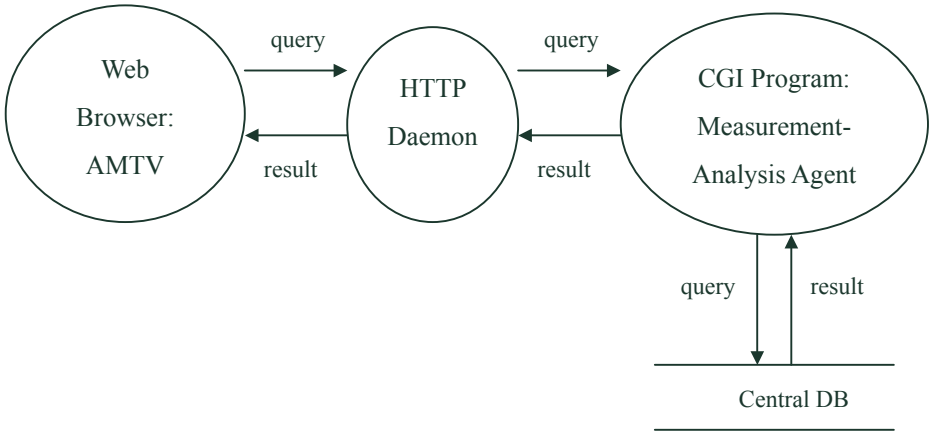

Figure 15. Procedure of Visualization

4.4 State Transition Diagram of AMT

We describe state transition of each process of AMT. AMT process can be described as a State Machine in use of Automata. Table 1 summarizes the states in which AMT process can stay.

State	Description
Init	Initial state of Control Server(CSV)
No exist	State when a process does not exist
Register	State when a process except CSV is registered in CSV
Measure-ready	State when a measurement is prepared to start
Measure-start	State when a measurement is proceeding
Measure-stop	State when a measurement is stopped
Gather-ready	State when a gathering is prepared to start
Gather-start	State when a gathering is proceeding
Gather-complete	State when a gathering is completed
Gather-stop	State when a gathering is stopped

Table 1. Process States

4.4.1 Control Server(CSV)

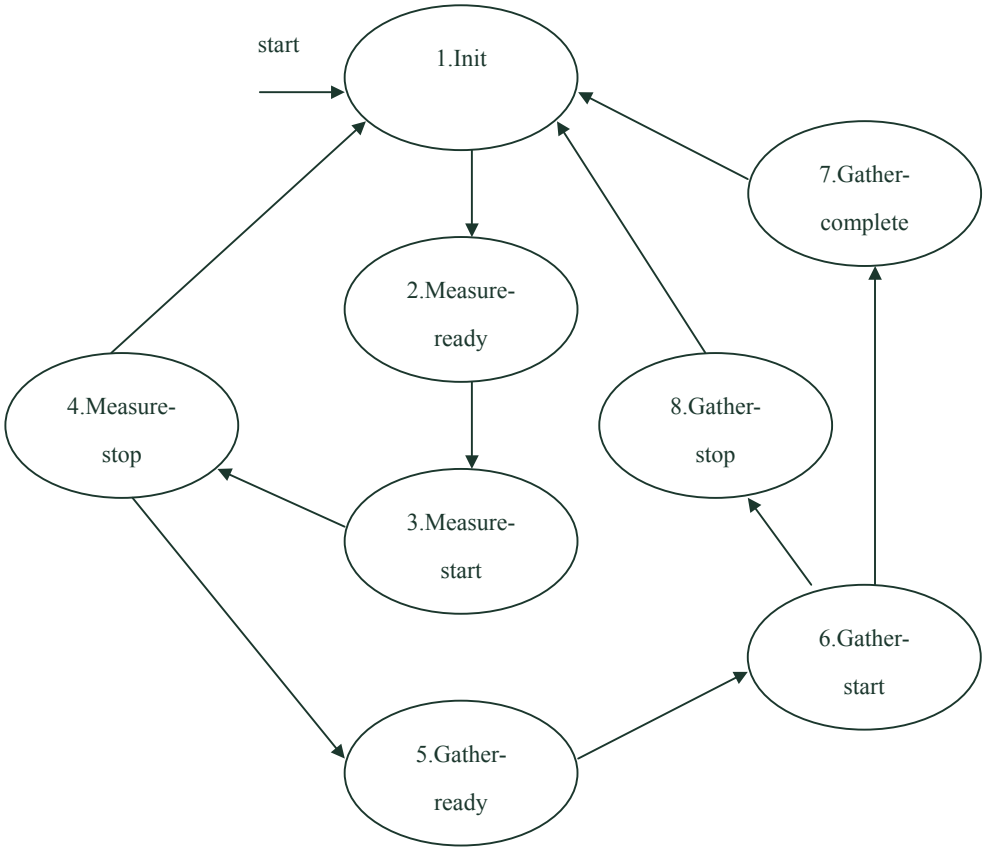

Figure 16. State Transition of Control Server

CSV changes its state according to operator's command and sends a control message to AMT processes such as AMTD and SSV. For CSV to go to next state, it has to receive feedback message from AMT processes. CSV changes its state as follows to process a measurement;

State 1. Init

When CSV is launched by operator, it stays in 'Init' state.

State 2. Measure-ready

When CSV gets a command that it has to start a measurement from operator, it sends a 'measure-ready' message to each AMTD that takes part in measurement. When CSV has received a 'measure-ready-ack' feedback

message from all AMTDs, it goes to ‘Measure-ready’ state and sends a ‘measure-start’ message to each AMTD.

State 3. Measure-start

When CSV has received a ‘measure-start-ack’ feedback message from all AMTDs, it goes to ‘Measure-start’ state. Now AMT is under a one-way measurement.

State 4. Measure-stop

When a measurement timer expires, this event indicates that the measurement has to be stopped. CSV sends a message ‘measure-stop’ message to each AMTD and SSV. When CSV has received from each AMTD a ‘measure-stop-ack’ feedback message, it goes to ‘Measure-stop’ state. If CSV has to gather measurement data by configuration that operator has set, it sends a ‘gather-ready’ message to each AMTD and SSV. From this point, gathering of measurement data from MS’s has started. Otherwise, it goes to initial state, ‘Init’.

State 5. Gather-ready

When CSV has received a ‘gather-ready-ack’ feedback message from all AMTDs and SSV, it goes to ‘Gather-ready’ state and sends a ‘gather-start’ message to SSV. When CSV has received a ‘gather-start-ack’ feedback message from SSV, it sends a ‘gather-start’ message to an AMTD that scheduler for gathering selects.

State 6. Gather-start

When CSV has received a ‘gather-start-ack’ feedback message from the AMTD, it goes to ‘Gather-start’ state. Now AMT is under gathering of measurement data from one MS.

State 7. Gather-complete

When CSV has received a ‘gather-complete’ message from SSV and AMTD, it goes to ‘Gather-complete’ state. From this point, Unless CSV

has gathered measurement data from all MS's, CSV again sends a 'gather-start' message to SSV. When CSV has received a 'gather-start-ack' feedback message from SSV, it sends a 'gather-start' message to the next AMTD that scheduler for gathering selects. When CSV has received a 'gather-start-ack' feedback message from the AMTD, it goes to 'Gather-start' state. When CSV has completed the gathering from all MS's, it goes to initial state, 'Init'. Because SSV has completed its role and is not needed longer, it is killed by its parent process, CSV. CSV removes the zombie of SSV by wait system call.

State 8. Gather-stop

When CSV is under gathering and the following two events happens, it goes to 'Gather-stop' state. When operator sent a 'gather-stop' command or when a problem during gathering from an MS happened, it goes to initial state, 'Init'.

4.4.2 AMT Daemon(AMTD)

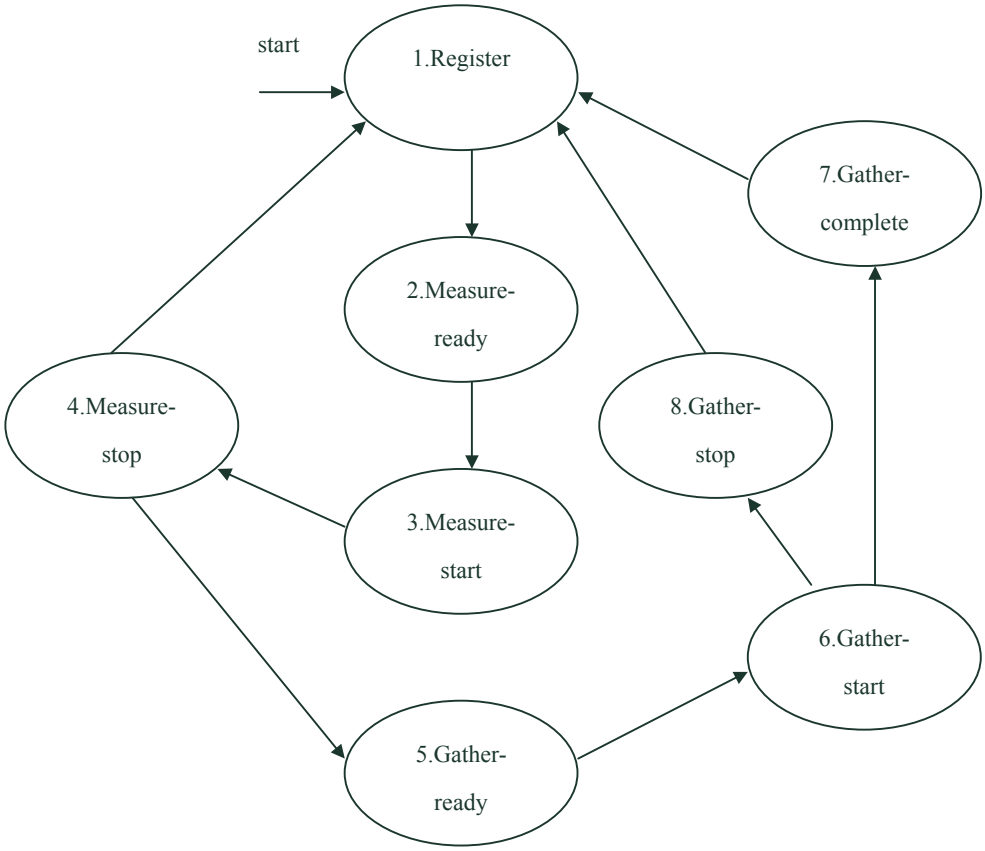

Figure 17. State Transition of AMT Daemon(AMTD)

AMTD is under control of CSV. So AMTD changes its state with processing a message from CSV. AMTD also relays a message from CSV to its child processes; AMTS, AMTR and DA. AMTD sends a feedback message from them to CSV. AMTD changes its state as follows to process a measurement and a delivery of measurement data;

State 1. Register

When AMTD is launched by Internet Daemon(i.e., inetd), it tries to register itself in CSV. AMTD sends CSV a ‘Register-request’ message indicating that AMTD wants to join in measurement activity as shown in

Figure 10. When AMTD receives a ‘Register-allow’ feedback message from CSV, it goes ‘Register’ state.

State 2. Measure-ready

When AMTD gets a ‘measure-ready’ message indicating that it has to prepare measurement, it forks measurement processes; AMTS and AMTR. When AMTD has received a ‘measure-ready-ack’ feedback message from both AMTS and AMTR, it goes to ‘Measure-ready’ state and send CSV a ‘measure-ready-ack’ feedback message.

State 3. Measure-start

When AMTD has received a ‘measure-start’ message indicating that it has to start a measurement, it relays the ‘measure-start’ message to both AMTS and AMTR. When AMTD has received a ‘measure-start-ack’ feedback message from both AMTS and AMTR, it goes to ‘Measure-start’ state and a ‘measure-start-ack’ feedback message to CSV. Now AMT is under a one-way measurement.

State 4. Measure-stop

When AMTD has received a ‘measure-stop’ message indication that it has to stop the measurement, it relays the ‘measure-stop’ message to both AMTS and AMTR. When AMTD has received a ‘measure-stop-ack’ feedback message from both AMTS and AMTR, it goes to ‘Measure-stop’ state and a ‘measure-stop-ack’ feedback message to CSV. Because AMTS and AMTR have completed their roles and are not needed longer, they are killed by its parent process, AMTD. AMTD removes the zombies of AMTS and AMTR by wait system call.

State 5. Gather-ready

When AMTD gets a ‘gather-ready’ message indicating that it has to prepare delivery of measurement data, it forks a delivery process, DA. When AMTD has received a ‘gather-ready-ack’ feedback message from DA, it goes to ‘Gather-ready’ state and send CSV a ‘gather-ready-ack’

feedback message.

State 6. Gather-start

When AMTD has received from CSV a ‘gather-start’ message indicating that it has to start a delivery of measurement data, it relays the ‘gather-start’ message to DA. When AMTD has received a ‘gather-start-ack’ feedback message from DA, it goes to ‘Gather-start’ state and a ‘gather-start-ack’ feedback message to CSV. Now AMT is under a gathering of measurement data.

State 7. Gather-complete

When AMTD has received from DA a ‘gather-complete’ message indicating that DA has delivered measurement data stored in Local DB to SSV, AMTD goes to ‘Gather-complete’ state and sends a ‘gather-complete’ message to CSV. Because DA has completed its role and is not needed longer, it is killed by its parent process, AMTD. AMTD removes the zombie of DA by wait system call. AMTD goes to initial state, ‘Register’.

State 8. Gather-stop

When AMTD has received from CSV ‘gather-stop’ message indicating that DA has to stop delivering measurement data to SSV, AMTD sends a ‘gather-stop’ message to DA. When AMTD has received a ‘gather-stop-ack’ feedback message, it goes to ‘Gather-stop’ state and sends CSV a ‘gather-stop-ack’ feedback message. Because DA is not needed longer, it is killed by its parent process, AMTD. AMTD removes the zombie of DA by wait system call. AMTD goes to initial state, ‘Register’.

4.4.3 AMT Sender(AMTS)

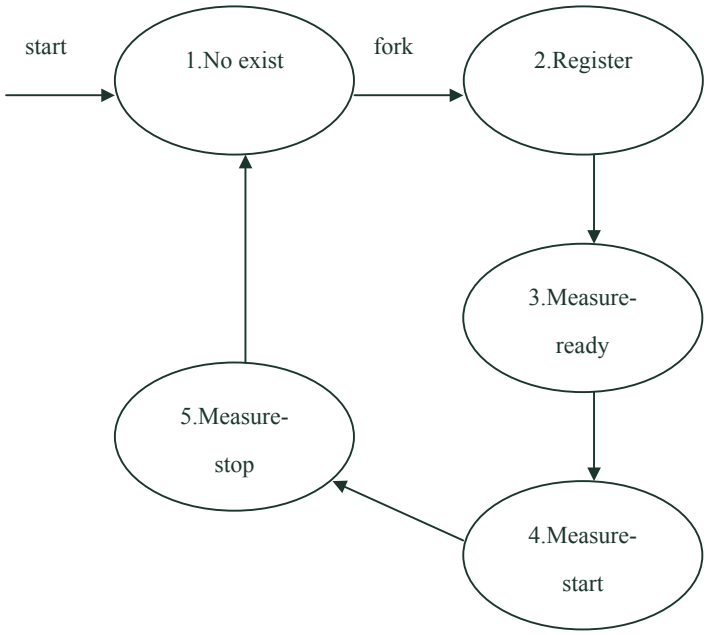

Figure 18. State Transition of AMT Sender(AMTS)

AMTS is under control of AMTD. So AMTS changes its state with processing a message from AMTD. After processing a message, AMTS sends a feedback message to AMTD.

State 1. No exist

This state indicates that AMTS does not exist in MS.

State 2. Register

When AMTS has been forked by its parent process(AMTD), it goes to 'Register' state.

State 3. Measure-ready

Just after AMTS has prepared a measurement, it sends a 'measure-ready-ack' message to AMTD and goes to 'Measure-ready' state.

State 4. Measure-start

When AMTS has received a ‘measure-start’ message from AMTD, it sends a ‘measure-start-ack’ feedback message to AMTD and goes to ‘Measure-start’ state. In this state, AMTS starts to generate measurement packets and send them to AMTRs except AMTR of its own MS.

State 5. Measure-stop

When AMTS has received a ‘measure-stop’ message from AMTD, it sends a ‘measure-stop-ack’ feedback message to AMTD and goes to ‘Measure-stop’ state. When AMTS has been killed by its parent process(AMTD), it goes to ‘No exist’ state.

4.4.4 AMT Receiver(AMTR)

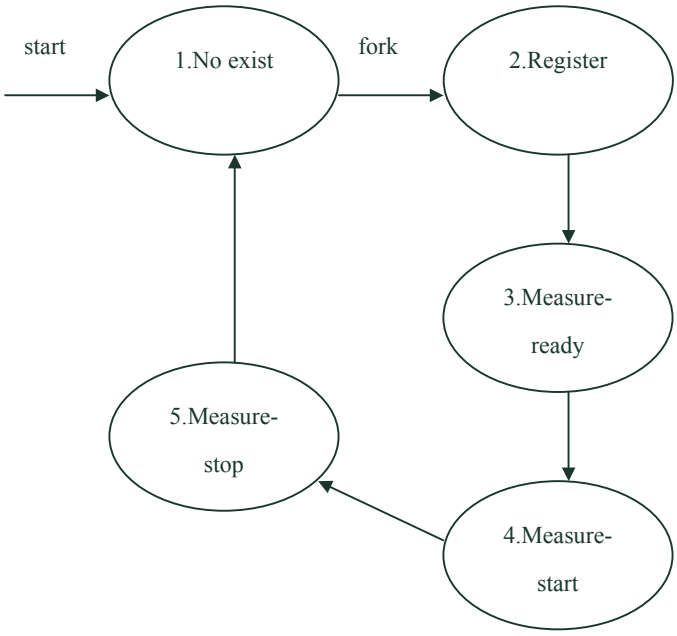

Figure 19. State Transition of AMT Receiver(AMTR)

AMTR is under control of AMTD. So AMTR changes its state with processing a message from AMTD. After processing a message, AMTR sends a feedback message to AMTD.

State 1. No exist

This state indicates that AMTR does not exist in MS.

State 2. Register

When AMTR has been forked by its parent process(AMTD), it goes to 'Register' state.

State 3. Measure-ready

Just after AMTR has prepared a measurement, it sends a 'measure-ready-ack' message to AMTD and goes to 'Measure-ready' state.

State 4. Measure-start

When AMTR has received a ‘measure-start’ message from AMTD, it sends a ‘measure-start-ack’ feedback message to AMTD and goes to ‘Measure-start’ state. In this state, AMTS starts to receive measurement packets and store them in Local DB.

State 5. Measure-stop

When AMTR has received a ‘measure-stop’ message from AMTD, it sends a ‘measure-stop-ack’ feedback message to AMTD and goes to ‘Measure-stop’ state. When AMTR has been killed by its parent process(AMTD), it goes to ‘No exist’ state.

4.4.5 Storage Server(SSV)

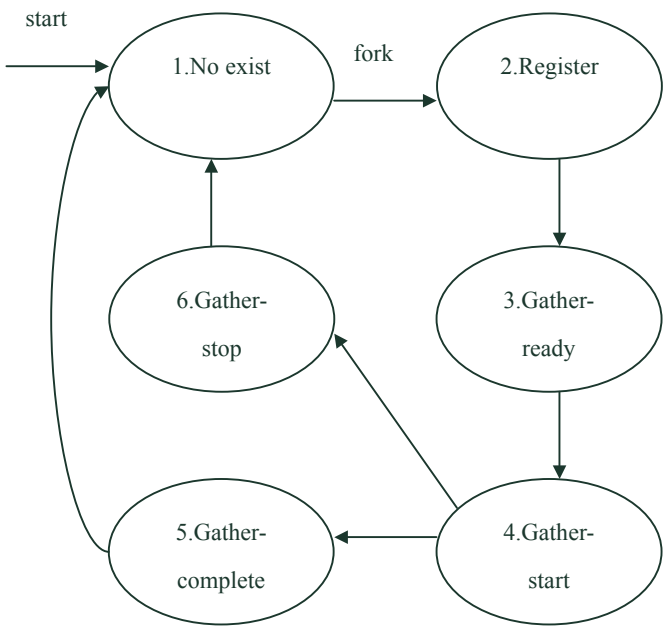

Figure 20. State Transition of Storage Server(SSV)

SSV is under control of CSV. So SSV changes its state with processing a message from SSV. After processing a message, SSV sends a feedback message to CSV.

State 1. No exist

This state indicates that SSV does not exist in CS.

State 2. Register

When SSV has been forked by its parent process(CSV), it goes to 'Register' state.

State 3. Gather-ready

Just after SSV has prepared a gathering, it sends a 'gather-ready-ack' message to CSV and goes to 'Gather-ready' state.

State 4. Gather-start

When SSV has received a ‘gather-start’ message from CSV, it sends a ‘gather-start-ack’ feedback message to CSV and goes to ‘Gather-start’ state. In this state, SSV starts to gather measurement data and store them in Central DB.

State 5. Gather-complete

When SSV has completed gathering of measurement data from all MS’s, it sends a ‘gather-complete’ message to CSV and goes to ‘Gather-complete’ state. When SSV has been killed by its parent process(CSV), it goes to ‘No exist’ state.

State 6. Gather-stop

When SSV has received a ‘gather-stop’ message from CSV, it stops gathering of measurement data. SSV sends a ‘gather-stop-ack’ feedback message to CSV and goes to ‘Gather-stop’ state. When SSV has been killed by its parent process(CSV), it goes to ‘No exist’ state.

4.4.6 Delivery Agent(DA)

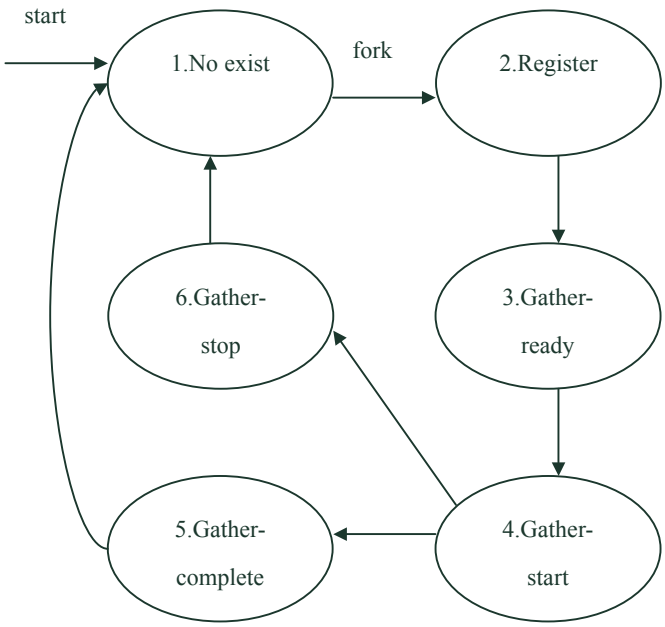

Figure 21. State Transition of Delivery Agent(DA)

DA is under control of AMTD. So DA changes its state with processing a message from AMTD. After processing a message, DA sends a feedback message to AMTD.

State 1. No exist

This state indicates that DA does not exist in MS.

State 2. Register

When DA has been forked by its parent process(AMTD), it goes to 'Register' state.

State 3. Gather-ready

Just after DA has prepared a gathering, it sends a 'gather-ready-ack' message to AMTD and goes to 'Gather-ready' state.

State 4. Gather-start

When DA has received a ‘gather-start’ message from AMTD, it sends a ‘gather-start-ack’ feedback message to AMTD and goes to ‘Gather-start’ state. In this state, DA starts to deliver measurement data stored in Local DB to SSV in CS.

State 5. Gather-complete

When DA has completed delivery of measurement data, it sends a ‘gather-complete’ message to SSV and AMTD. DA goes to ‘Gather-complete’ state. When DA has been killed by its parent process(AMTD), it goes to ‘No exist’ state.

State 6. Gather-stop

When DA has received a ‘gather-stop’ message from AMTD, it stops delivery of measurement data. DA sends a ‘gather-stop-ack’ feedback message to AMTD and goes to ‘Gather-stop’ state. When DA has been killed by its parent process(AMTD), it goes to ‘No exist’ state.

Chapter 5

Performance Evaluation

We measured one-way delay and loss-rate in Internet including Korea commercial Network(KORNET) and evaluate the result of measurement.

5.1 Test Environment

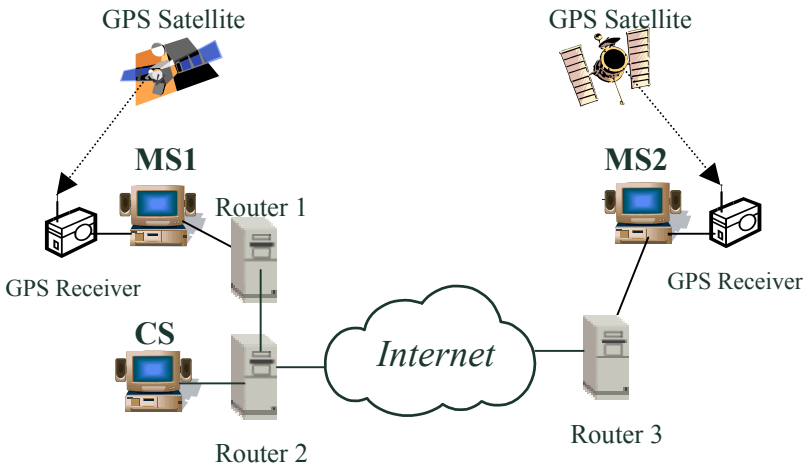

Figure 22. Test Network

Figure 22 shows the topology of test network. MS1 of which IP address is 147.46.14.69 is located in a subnet in Seoul National University and MS2 of which IP address is 203.232.127.20 is located in a subnet of KORNET. CS of which IP address is 147.46.15.203 is located in a subnet in Seoul National University. Router 1's IP address is 147.46.14.65, Router 2's IP address is 147.46.15.2 and Router 3's IP address is 203.232.127.14. Router 1 is adjacent to Router 2 as shown in Figure 22. Table 2 is the result of traceroute from MS 1 to MS2 and Table 3 is is the result of traceroute from MS2 to MS1. From Table 2 and Table 3 we can see that path from MS 1 to MS2 is the reverse of path from MS 2 to MS 1.

TTL	Host or Router	RTT [ms]
1	147.46.14.65	0.314 0.300 0.302
2	147.46.15.2	3.241 3.089 3.707
3	147.46.50.1	9.799 3.702 13.738
4	seoulgw.snu.ac.kr(147.46.80.99)	19.017 10.674 13.084
5	168.126.233.37	7.502 5.721 5.506
6	168.126.228.101	5.941 6.667 5.713
7	211.193.39.65	5.703 3.355 3.743
8	168.126.109.1	4.890 11.054 18.531
9	hh-r1-ge1.kornet.net(211.192.46.31)	27.146 23.769
10	168.126.104.43	22.908 164.230 152.018
11	168.126.12.18	58.623 16.821 15.419
12	203.232.127.20	8.326 8.965 8.374

Table 2. Traceroute from MS 1 to MS 2

TTL	Host or Router	RTT [ms]
1	203.232.127.14	2.000 1.974 1.693
2	168.126.12.17	2.987 2.892 2.871
3	168.126.104.2	3.069 2.942 2.872
4	hh-c1-ge3.kornet.net(211.192.47.1)	3.039 2.989 2.900
5	168.126.109.2	3.456 3.348 3.248
6	211.193.39.84	3.316 3.288 3.310
7	168.126.228.231	3.352 3.291 3.231
8	168.126.233.38	3.964 4.125 4.351
9	147.46.80.101	9.535 14.049 8.442
10	147.46.50.5	18.195 9.378 9.177
11	147.46.15.55	8.230 6.488 7.220
12	147.46.14.69	6.946 8.716 11.368

Table 3. Traceroute from MS 2 to MS 1

5.2 Evaluation of Result of Measurement

We measured one-way delay during a day from 0 AM on 2000/11/28 to 12 PM on 2000/11/28. We generated measurement packets in the frequency that Lambda of Poisson process is 2.

Figure 23 shows one-way delay from MS1 to MS2(Delay1) and Figure 24 shows one-way delay from MS2 to MS1(Delay2). X-axis of graph is time. The unit is 5 minutes. Y-axis is one-way delay. The unit is 1 microsecond(us). As representative values, we selected (a) Minimum delay, (b) 95^{th} percentile and (c) Maximum delay in the period of 5 minutes. Because Percentile is the most reasonable among three representatives, we compare two figures(Figure 23 and Figure 24) by 95^{th} percentile. We can see that 95^{th} percentile of Delay1 is from 100148[us] to 539724[us] and that 95^{th} percentile of Delay2 is from 7923[us] to 16344[us]. As a result, we can see that the one-way path from MS1 to MS2(Path1) has bigger and more variable one-way delay than that from MS2 to MS1(Path2).

Figure 25 shows loss-rate from MS1 to MS2(Rate1) and Figure 26 shows loss-rate from MS2 to MS1(Rate2). X-axis of graph is time. The unit is 5 minutes. Y-axis is loss-rate. The unit is number of loss-packet. We computed loss-rate by RFC 2680[5]. We decided loss-threshold as 1[sec]. We consider a packet that has bigger one-way delay than loss-threshold as a loss. We can see that Rate1 is from 0 to 295 and that Rate2 is from 26 to 180. As a result, we can see that the one-way path from MS1 to MS2(Path1) has bigger and more variable loss-rate than that from MS2 to MS1(Path2).

Figure 27 shows delay-jitter from MS1 to MS2(Jitter1) and Figure 28 shows delay-jitter from MS2 to MS1(Jitter2). X-axis of graph is time. The unit is 5 minutes. Y-axis is delay-jitter. The unit is 1 microsecond(us). We computed delay-jitter by difference of Maximum delay and Minimum delay in a unit time(5 minutes). As a result, we can see that the one-way path from MS1 to MS2(Path1) has bigger and more variable delay-jitter than that from MS2 to MS1(Path2).

Through the above measurement, we can infer that Path1 is more loaded than Path2 or that Path1 has some problems(e.g., Problem of routing configuration). We can not find the above fact with a ‘Ping’ that measures RTT between two end hosts.

Like this, with one-way measurement we can get much useful information in order to manage network.

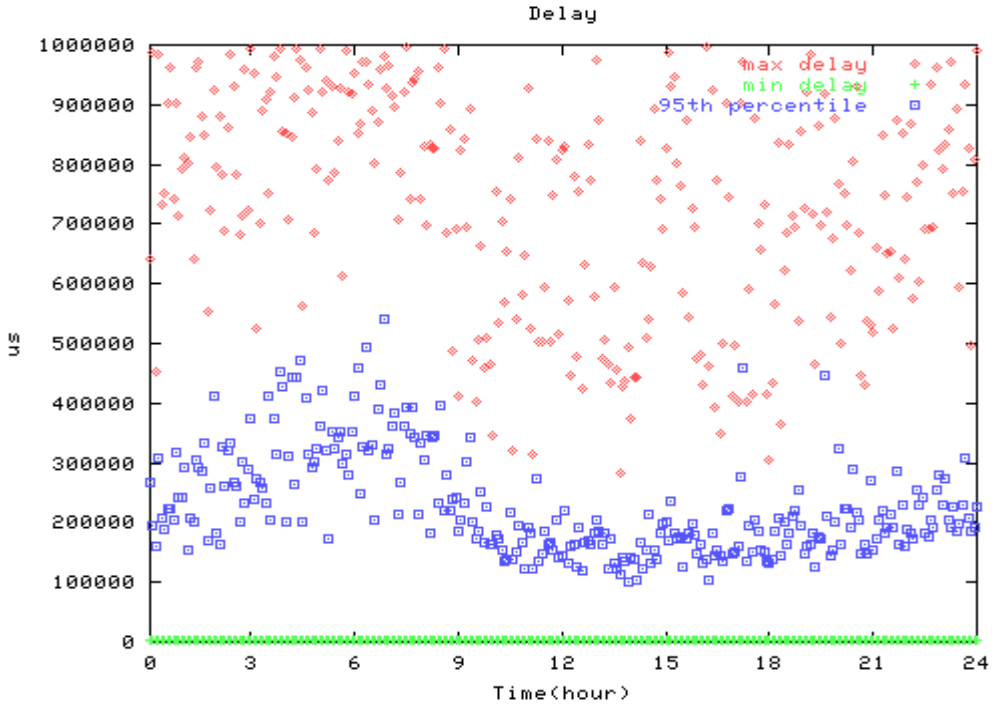

Figure 23. One-way Delay from MS1 to MS2

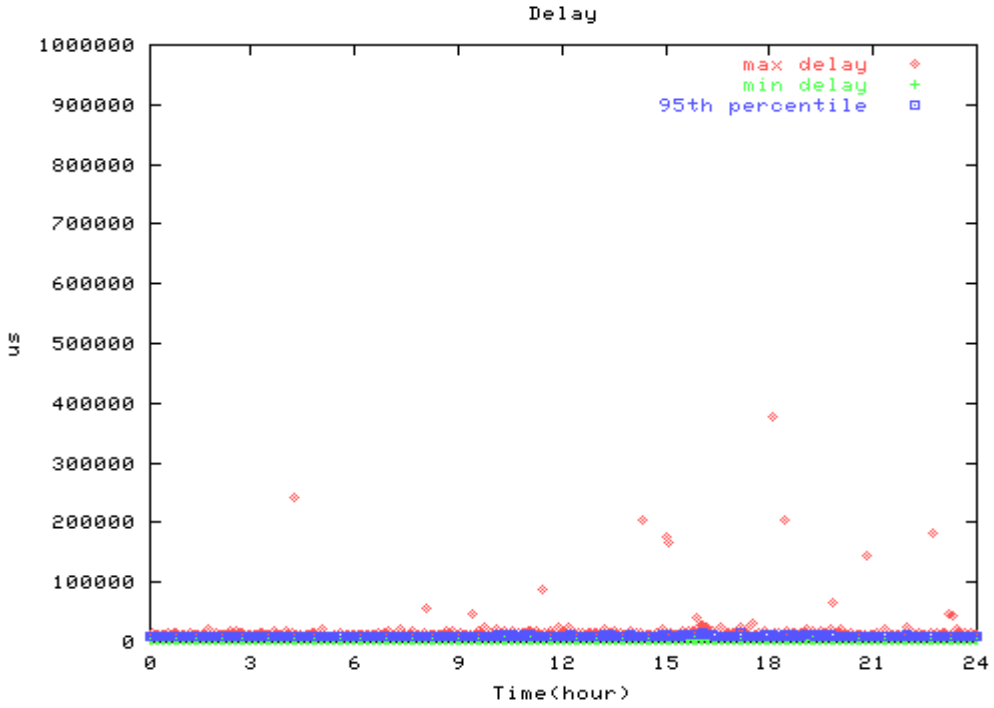

Figure 24. One-way Delay from MS2 to MS1

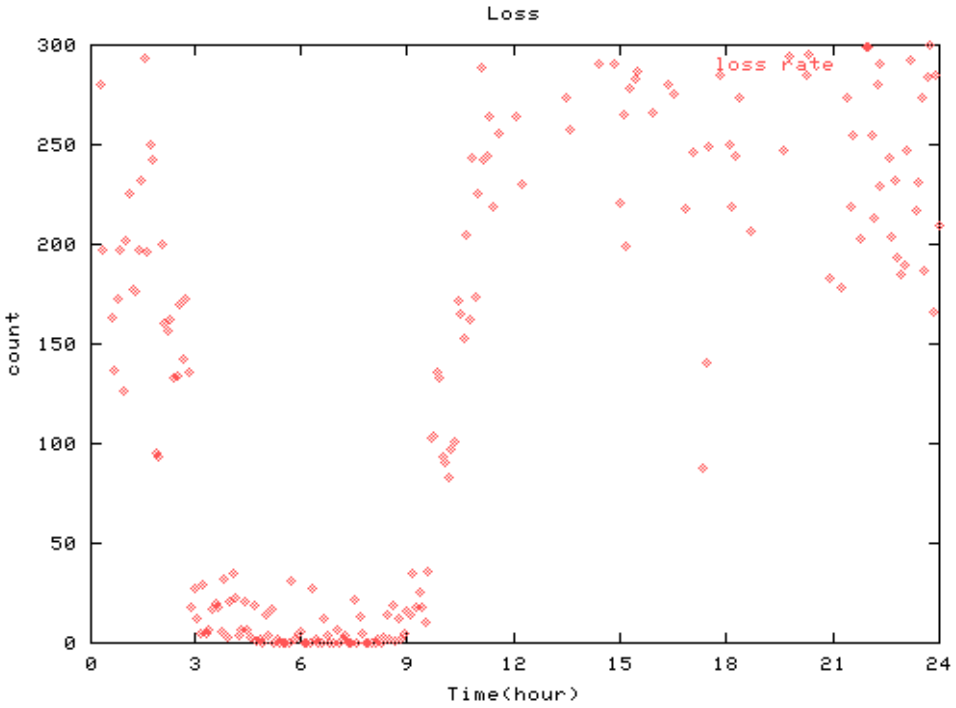

Figure 25. Loss Rate from MS1 to MS2

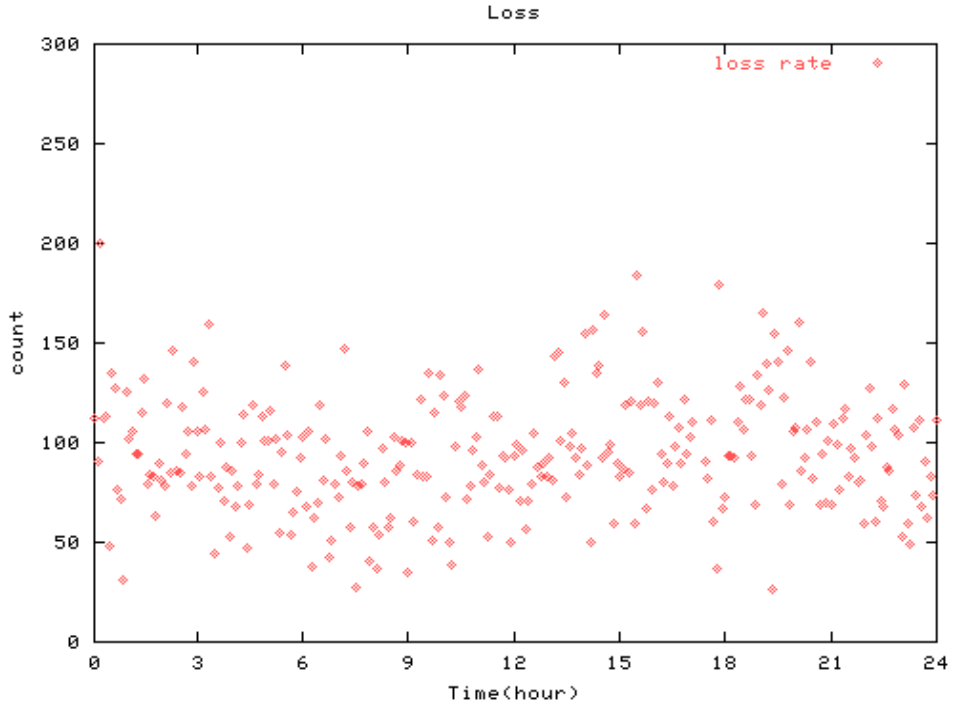

Figure 26. Loss Rate from MS2 to MS1

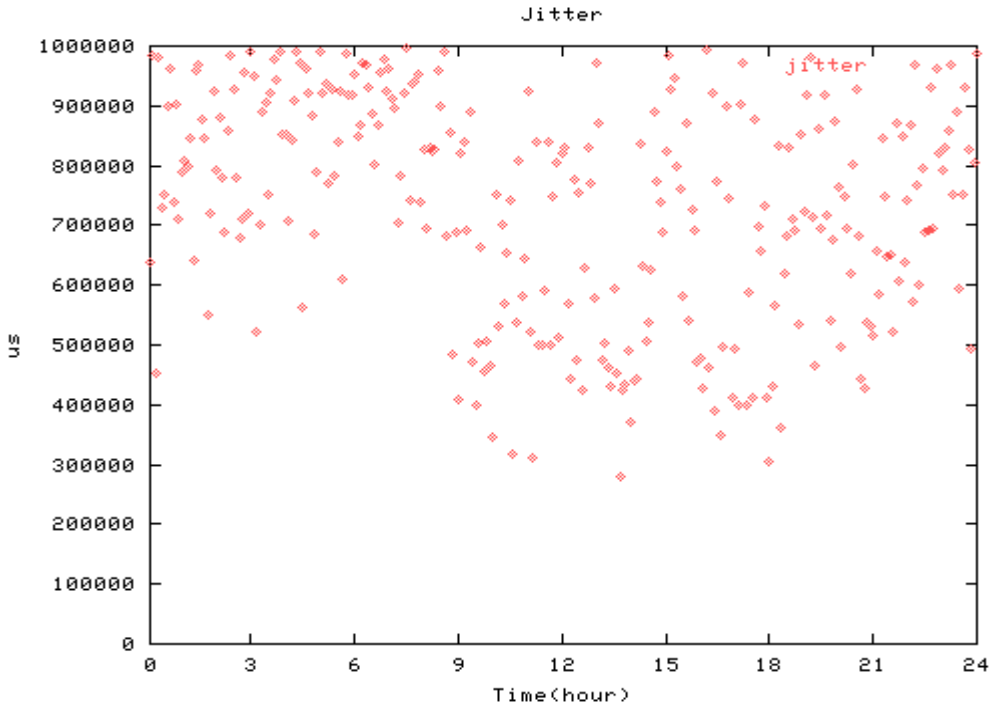

Figure 27. Delay Jitter from MS1 to MS2

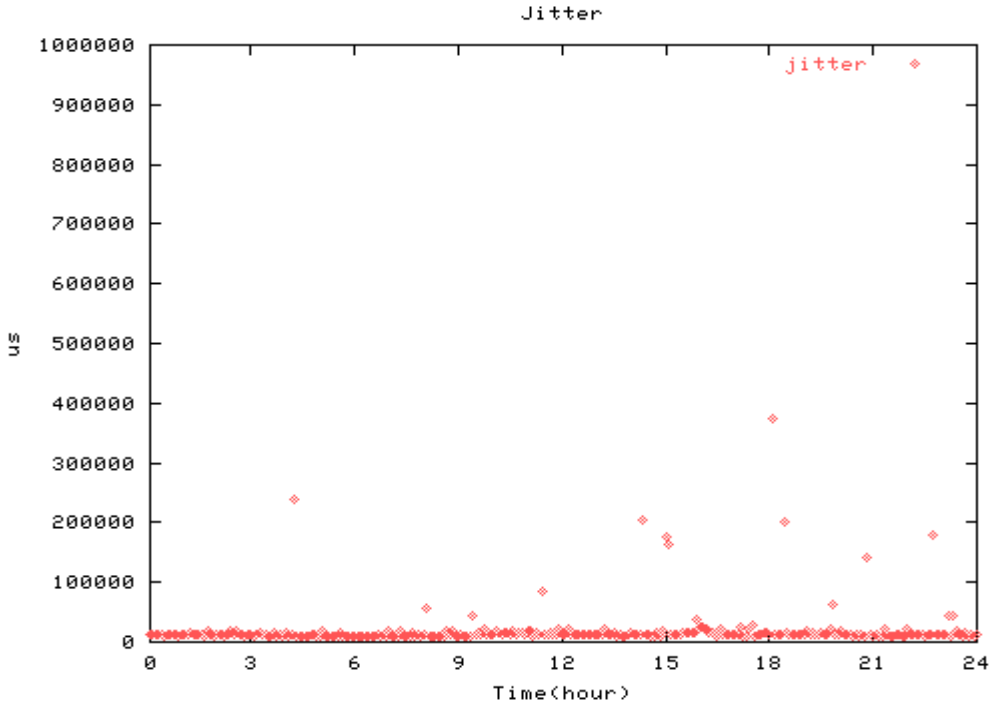

Figure 28. Delay Jitter from MS2 to MS1

Chapter 6

Conclusion and Future Work

One-way measurement is a kind of active measurement where measurement packet is injected in the path to measure and how for the packet to be served is observed. With measurement data through active measurement, the network management can be performed effectively. When we consider that the Internet is asymmetric, we can do active measurement better with one-way measurement in order to grasp the state of network accurately. One-way metrics that the IETF's IP Performance Metrics(IPPM) Working Group suggested are popularly used in one-way measurement.

There are many cases where one-way measurement is useful. For example, if we had observed the network for a long time and had found that one-way delay increased much more than in ordinary times. We can guess that some problems have happened in the network. We can cope with the problems by resource relocation, load balancing through modification of routing path and so on.

In this thesis, we suggested an architecture for measurement system(AMT: Active Measurement Tool) that can perform the one-way measurement efficiently. We also described the procedures related to measurement. We presented the analysis of result that we have obtained through measurement in the Internet

including Korea Commercial Network(KORNET).

AMT is expected to be deployed in Korea Commercial Network(KORNET) and Asia Pacific Advanced Network(APAN) for network management and experiment related to QoS(i.e., DiffServ). More functions will be added; (a) Function for self-troubleshooting (b) Control of Control Server through Web and (c) Authentication of control messages used in AMT.

Bibliography

[1] V. Paxson, “End-to-End Internet Packet Dynamics”, IEEE/ACM Transactions on Networking, Vol.7, No.3, pp.277 -292, June 1999.

[2] V. Paxson, “Framework for IP Performance Metrics”, RFC 2330, May 1998.

[3] G. Almes et al., “A One-way Delay Metric for IPPM”, RFC 2679, September 1999.

[4] C. Demichelis et al., “Instantaneous Packet Delay Variation Metric for IPPM”, Internet-Draft, October 1999.

[5] G. Almes et al., “A One-way Packet Loss Metric for IPPM”, RFC 2680, September 1999.

[6] Tony McGregor et al., “The NLANR Network Analysis Infrastructure”, IEEE Communications Magazine, May 2000.

[7] skitter Website, http://www.caida.org/tools/measurement/skitter/

[8] surveyor Website, http://www.advanced.org/surveyor/

[9] Sunil Kalidindi et al., “Surveyor: An Infrastructure for Internet Performance Measurements”, presented at INET'99, San Jose, June 1999.

[10] Sunil Kalidindi,“OWDP:A Protocol to measure One-Way Delay and Packet Loss”, Surveyor Technical Report 001.

[11] Sunil Kalidindi, “OWDP Implementation, v1.0”, Surveyor Technical Report 002.

[12] MySQL Website, http://www.mysql.com

[13] NTP Website, http://www.eecis.udel.edu/~ntp/

[14] David L. Mills, “Network Time Protocol (Version 3): Specification, Implementation and Analysis”, RFC 1305, March 1992.

[15] Gary R. Wright, W. Richard Stevens, “TCP/IP Illustrated, Volume 2: Implementation”, Addison Wesley.

[16] Measurement Specification of CAIDA Group, http://www.caida.org/tools/measurement/measurementspec/

초 록

본 논문에서는 비대칭적인 인터넷에서의 단방향 지연시간, 지연시간 편차 및 손실률을 측정할 수 있는 시스템구조를 제시하고, 구현된 측정도구를 이용하여 측정된 결과를 분석한다. 제안된 구조에서는 단방향성 IP 성능 인자의 측정에 반드시 필요한 측정기계들간의 시간 동기화를 위해서 Global Positioning System(GPS)을 이용하여 Micro-second 수준의 정밀도를 제공한다. 그리고, 측정의 정확도를 향상시키기 위하여 송신자 및 수신자가 이더넷 프레임 처리 직전에 시간 정보를 기록하여 프로토콜 계층간의 이동에 의한 지연시간을 줄이고 있다. 본 논문에서는 구현한 측정 도구를 인터넷에 적용하여 단방향성 IP 성능 인자를 얻는다. 또한 이 측정결과를 통해 단방향 측정의 필요성을 제시한다.

주요어: IETF's IP Performance Metrics(IPPM), 단방향 지연, Global Positioning System(GPS), 능동적 측정.

감사의 글

대학원을 마칠 즈음에서 아쉬운 점이 많습니다. 지난 2년동안의 대학원생활을 돌이켜보면 잘 한 것보다 잘 못한 것이 더 많이 남아 있는 것 같습니다. 가장 아쉬운 것은 늘 의욕만 앞섰지, 그 의욕만큼 실행에 옮기지 못했던 것입니다. 다음에 다시 공부할 기회가 주어지면 후회없는 대학원 생활을 하고 싶습니다.

처음에 연구실에 들어왔을 때는 연구실 사람들이 모두 낯설었지만, 졸업을 앞둔 지금은 모두 정든 사람들이 되었습니다. 학부때부터 전공하고 싶었던 컴퓨터 네트워크를 2년간 공부한 것은 제 생애의 큰 행운이 아니었나 싶습니다. 그러나 주위의 여러 고마운 분들의 도움이 없었다면 지금의 저는 존재할 수 없었다고 알고 있습니다. 가장 고마운 분은 어머니입니다. 4남 5녀의 막내로 절 낳아주시고 지금까지 많은 사랑으로 건강하고 올바른 젊음이로 키워주신 어머니께 먼저 고마움을 전하고 싶습니다. 아버지께도 고마움을 전합니다. 국민학교 5학년때 돌아가시고 지금은 천국에 계시는 아버지는 어린 자식인 저를 엄하게 키우시면서 숨은 사랑을 많이 베풀어 주셨습니다. 형들과 누나들에게도 감사를 전하고 싶습니다. 컴퓨터 네트워크를 전공할 수 있게 저를 선택해 주시고, 2년간 많은 사랑으로 지도해주신 최양희 교수님, 제가 대학원 생활을 잘 못하고 방황할 때 친동생처럼 충고해주고 이끌어주신 정승훈 선배님, 공부뿐만 아니라 인생의 선배로서 많은 도움을 주신 박장연 선배님, 때론 친구처럼 허물없이 지내며 연구하는 방법을 아낌없이 나누어 주시고 형처럼 많은 사랑을 주신 김동균 선배님, 학위 논문을 쓰는데 많은 조언을 주신 이영석 선배님, 이 논문의 주제인 측정도구를 같이 구현하며 많은 고생을 함께 한 박재우 후배 그리고 2년간 저와 기쁨과 슬픔을 같이 한 연구실의 저의 단짝인 정환석 학우와 동기들을 비롯한 다른 모든 선배님들과 후배들에게 진심으로 고마움을 전하고 싶습니다.

대학원생활에 못지 않게 저에게 중요한 신앙생활의 터전인 성당 주일학교에서 함께 일한 선생님들과 많은 배려와 사랑을 주신 오베드로 신부님과 최 다니엘 수녀님께도 감사를 전하고 싶습니다. 그리고 형제와 같은 벗들인 신정훈, 박우동, 최진호 그리고 정현주와 기쁨을 함께 나누고 싶습니다.

끝으로 저의 인생을 이끌어 주시고 사랑과 지혜 그리고 용기를 주시는 하느님께 진심으로 감사를 전하고 싶습니다.

2001년 1월 5일,
관악산에 있는 컴퓨터신기술공동연구소,
정재훈 (Jaehoon Jeong, Paul).