Streaming Media and Multimedia Conferencing Traffic Analysis Using Payload Examination

Hun-Jeong Kang, Myung-Sup Kim, and James W. Hong

This paper presents a method and architecture to analyze streaming media and multimedia conferencing traffic. Our method is based on detecting the transport protocol and port numbers that are dynamically assigned during the setup between communicating parties. We then apply such information to analyze traffic generated by the most popular streaming media and multimedia conferencing applications, namely, Windows Media, Real Networks, QuickTime, SIP and H.323. We also describe a prototype implementation of a traffic monitoring and analysis system that uses our method and architecture.

Keywords: Internet management, network traffic monitoring and analysis, multimedia traffic analysis.

I. Introduction

The use of streaming media and multimedia conferencing applications is growing rapidly. Many Internet sites provide various rich media broadcasts such as movies, video clips, and music. Also, video conferencing applications are gaining in popularity. This trend will accelerate because of an increasing number of Internet users, high bandwidth connections, and improved PC performance. These applications are generating a huge volume of network traffic and thus causing Internet Service Providers (ISPs) and enterprises to provide more bandwidth in their networks.

For various purposes, including network planning, most network administrators wish to have a good understanding of the traffic usage on their network. Unfortunately, traditional analysis methods based on well-known port numbers [1] cannot be used to analyze traffic generated by streaming media and multimedia conferencing applications. Because these applications make use of dynamically allocated port numbers that are assigned during set up sessions, most traditional methods cannot determine the application of such traffic. As a result, this traffic is misidentified as unknown traffic in these methods [2], [3].

To solve this problem, we have developed a method and real-time system for monitoring and analyzing streaming media and multimedia conferencing traffic, which are collectively called *multimedia service traffic* in this paper. We have developed a dynamic session analyzer that parses control protocols. This module analyzes the payload of a packet associated with a control protocol and extracts information about transport protocols and port numbers used for transferring multimedia service data. We use this information to determine whether or not unknown traffic is a particular type

Manuscript received Apr. 28, 2003; revised Jan. 12, 2004.

This work was supported by the Electrical and Computer Engineering Division at POSTECH under the BK21 program of Ministry of Education, the HY-SDR Research Center at Hanyang University under the ITRC program of Ministry of Information and Communication, and the Program for the Training of Graduate Students in Regional Innovation which was conducted by the Ministry of Commerce, Industry and Energy, Korea.

Hun-Jeong Kang (phone: +82 54 279 5641, email:bluewind@etri.re.kr), Myung-Sup Kim (email: mount@postech.ac.kr), and James W. Hong (email: jwkhong@postech.ac.kr) are with the DPNM Laboratory, POSTECH, Pohang, Korea.

of multimedia service traffic. With this approach, we can analyze multimedia service traffic and obtain information about network usage.

This paper is organized as follows. Section II provides an overview of several popular multimedia service protocols. Section III presents our analysis method for multimedia service traffic. Section IV describes the architecture of our multimedia service traffic monitoring and analysis. Section V describes our implementation of the monitoring and analysis system. Section VI discusses related work on traffic monitoring and analysis. Finally, Section VII summarizes our work and discusses possible future work.

II. Overview of Multimedia Service Protocols

Streaming media services transfer data as if it is a continuous flow, that is, an Internet user need not wait to download a large file before seeing the video or hearing the audio. Streaming media services use two sessions, control and data, for the communication and data transfer between a client and server. Control sessions are used to negotiate the protocol, bandwidth and data port to be used in a data transfer between a client and a server. Data sessions are used to stream the multimedia data from the server to the client. Multimedia data can be transported over HTTP, but in this paper we do not consider it as streaming traffic. Many streaming media services have been introduced to the marketplace, but only three have achieved a substantial user base [4]: Windows Media Technology (WMT) [5], Real Networks [6], and QuickTime [7]. These services differ in their protocols, as illustrated in Table 1.

Most applications of multimedia conferencing are based on Session Initiation Protocol (SIP) [8] or H.323 [9]. Table 2 describes protocols used in these applications.

During a multimedia service, two types of sessions are created between a client and a server: a *control session* and a *data session*. The control session is responsible for setting up a connection and controlling navigation, such as play and pause. This session uses control protocols such as Real Time Streaming Protocol (RTSP) [10] and MMS [4]. These control protocols usually use TCP with well-known ports. The data session sends the multimedia service contents to the client over the data session protocol, including Real Networks Data Transfer [6], Real time Transfer Protocol [11], and MMST/MMSU (MMS over TCP/UDP) [4]. Both TCP and UDP are used in these data protocols. We designate each packet related to the control session and data session as a control packet and a data packet.

Figure 1 illustrates a process in which control sessions and a data session are constructed, and then multimedia data is transferred. To begin, the control session is constructed through a well-known port number. As described in Fig. 1(a), streaming media services (e.g., Real Networks, QuickTime) or applications based on SIP have one control session. On the other hand, H.323 applications have two control sessions: Q.931 and H.245. Next, the control sessions create a new data session by negotiating a transport protocol and port numbers. Then, the data session transfers multimedia data through the dynamically assigned transport protocol and port numbers. In this paper, we introduce a new term, *dynamic session*, which makes use of the transport protocol and port numbers that are dynamically negotiated by the control session, such as the data session and second control session, as shown in Fig. 1(b).

When a control session negotiates a dynamic session, the packet payload of the control session contains the negotiation results, such as a transport protocol and port numbers used in the dynamic session. By selecting and analyzing the control packet, we can discover information about the dynamical session, which is called *dynamic session information* in this paper.

Table 1. Streaming media service protocols.

	QuickTime	Real Networks	WMT
Control session protocol	RTSP	RTSP	MMS
Data session protocol	RTP	RTP, RDT	MMST/MMSU

Table 2. Multimedia conferencing protocols.

	SIP-based application	H.323-based application
Control session protocol	SIP	RTP
Data session protocol	Q.931, H.245	RTP

III. An Analysis Method for Multimedia Service Traffic

In this section, we present our proposed method for analyzing multimedia service traffic.

Figure 2 is a flowchart to illustrate packets being captured and processed. The overall procedure consists of three major parts: flow generation, dynamic session analysis, and traffic analysis.

Flow generation captures packets and analyzes their header. By collecting and associating related packets, this part generates flows [14]-[16], which represent a series of packets traveling between "interesting" end points. By associating packets that belong to an identical flow, the system overhead of the processing data can be reduced. Based on this flow

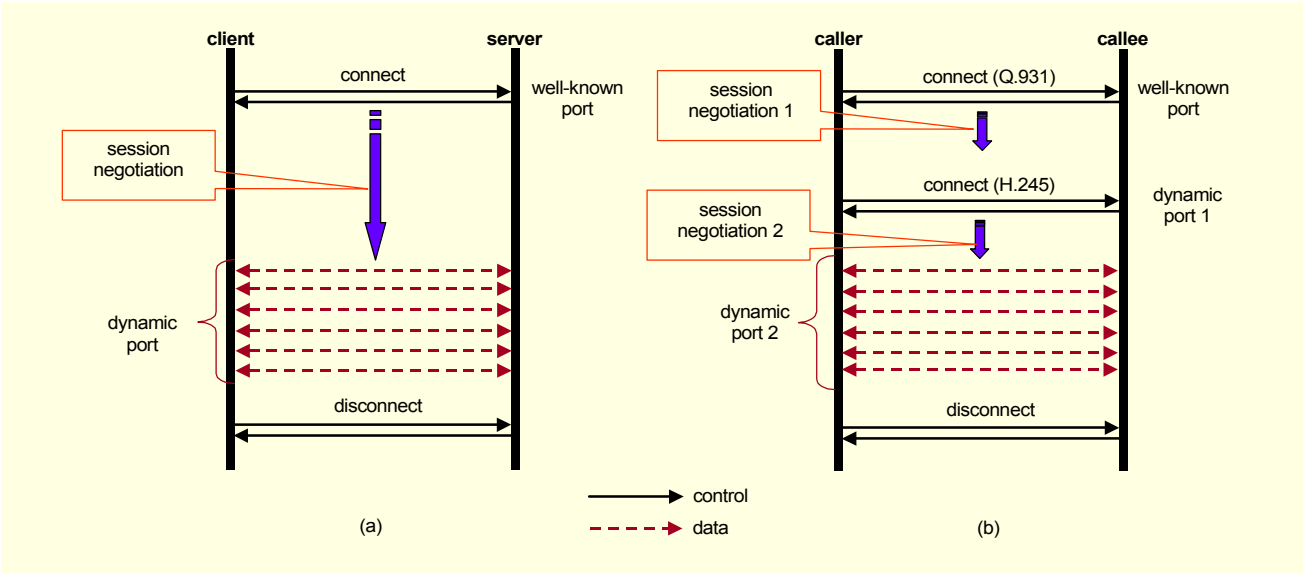

Fig. 1. A multimedia service control and data session: (a) RealMedia, QuickTime, WMT, SIP, (b) H.323.

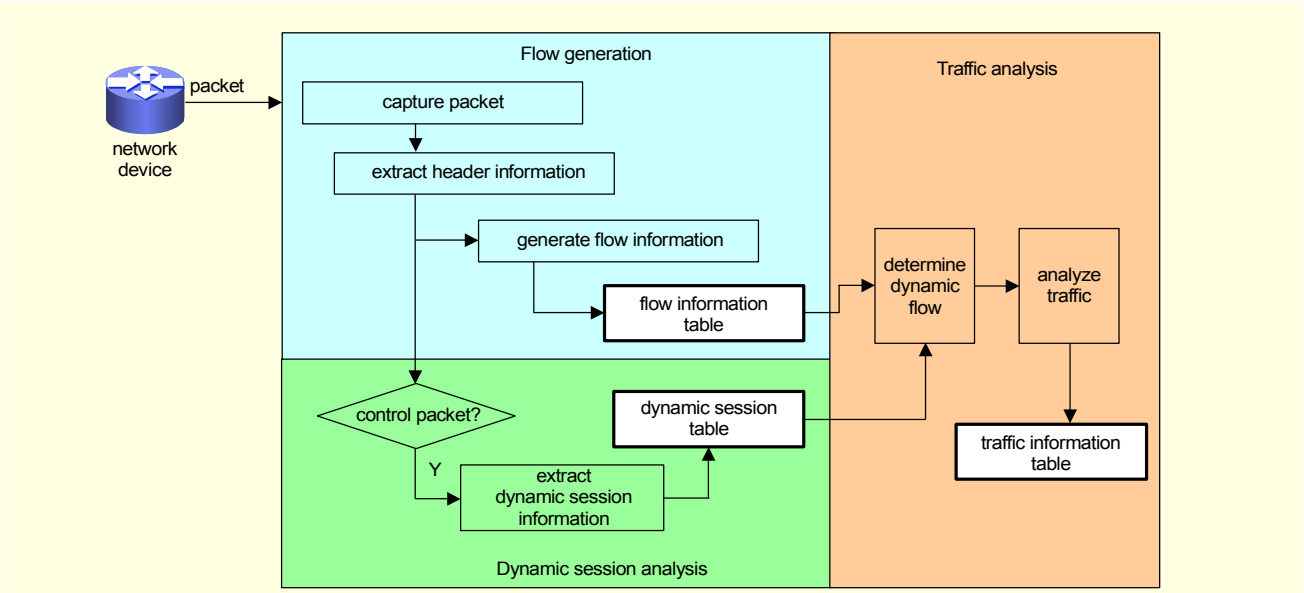

Fig. 2. A flowchart for multimedia service traffic monitoring and analysis.

information, the traffic analyzer generates various traffic information into the traffic information table.

However, it is insufficient to identify dynamic session traffic with only a port number. The reason for this is that dynamic sessions do not use well-known ports. Therefore, we need dynamic session information to decide whether or not a flow with an unknown port number is related to multimedia service traffic. This information can be extracted in the dynamic session analyzer. When a packet is analyzed by the flow generation part, the control packet is sent to the dynamic session analysis part. Then, the packet is analyzed to determine whether or not it contains dynamic session information. Figure 3 describes our algorithm to discover dynamic session information from the control packet.

The dynamic session analysis part receives a control message, including packet header information and a payload of the transport layer. First, the procedure determines if the FIN flag is set to identify it as a session disconnect request. If not, the module analyzes whether the packet contains dynamic session information. The analysis of the control message is performed

```
1  Procedure DynamicSessionAnalyzer ( Msg )
2    if FIN Flag in Msg is NOT set {
3      switch (protocol in Msg) {
4        case RTSP:
5          if SourcePort in Msg = RTSP server port number
6          then result ← ParseRTSP (payload of Msg);
7        case MMS:
8          if DestinationPort in Msg = MMS server port number
9          then result ← ParseMMS (payload of Msg);
10       case SIP:
11         result ← ParseSIP (payload of Msg ) ;
12       case Q.931:
13         if SourcePort in Msg = Q.931 receiver port
14         then result ← ParseQ931 (payload of Msg ) ;
15       default: {
16         Match Msg with information in dynamic session table;
17         if Msg is registered in the table and protocol in Msg = H.245
18         then result ← ParseH245 (payload of Msg);
19       }
20     }
21     if result= TRUE then {
22       create new dynamic session information;
23       insert dynamic session information into dynamic session table;
24     }
25   }
26   else   delete session information from dynamic session table;
```

Fig. 3. A dynamic session analysis algorithm.

according to the application-layer protocol that is determined by the well-known port numbers used by these protocols. Also, we can reduce the analysis overhead by selecting a packet that is likely to contain dynamic session information.

Figure 4 illustrates message exchanges of control sessions during negotiation of the dynamic session. In the case of RTSP, a client sends a SETUP request to a server along with the candidates for a data transport protocol and port number (or a range of port numbers) to be used for receiving multimedia service data. Next, the RESPONSE contains the protocol and port numbers chosen by the server. Accordingly, the procedure ascertains whether the source port of the packet is an RTSP server port (i.e., 554) (line 5), in order to select the server response packet. When parsing the payload of a control packet, the parser receives the information on media tracks. Thus, we can handle multiple, separate media tracks. However, we do not include information related to multiple, separate media tracks because our main purpose is to identify multimedia traffic applications.

Next, MMS is a proprietary control protocol. Although its specification is not publicly open, we have discovered by observing and analyzing packets that the client's request in MMS contains the transport protocol and port numbers used for transferring multimedia service data. Therefore, the destination port number is checked to verify that it is the MMS server port number (i.e., 1755) (line 8) for the purpose of choosing the client request packet. If the server rejects the request, the client again sends the request with another port number or does not send any request. In these cases, we do not misinterpret traffic, including the rejected port as a dynamic session port. We can identify the active port among the two, as the rejected port will not have any traffic. We can also identify the application's traffic by discerning the server address that is stored with the rejected port in the dynamic session table. In SIP, dynamic session information is contained in the *invite* message of a client or in a response message from a server. For this reason, the procedure selects packets with 5060, SIP server port (line 10), and verifies if they are *invite* or response messages.

However, services based on H.323 have two dynamic sessions, as illustrated in Fig. 5. First, Q.931 uses the well-known receiver port, 1720. Because the information about H.245 is contained in the connect message, the procedure determines if the source port is the receiver port (line 13 in Fig. 3). In the case of H.245, this session uses a port number that is dynamically allocated by a Q.931 session. Accordingly, we need to look up the dynamic session table and determine whether a captured packet is related to a H.245 session by matching the dynamic session information generated by the

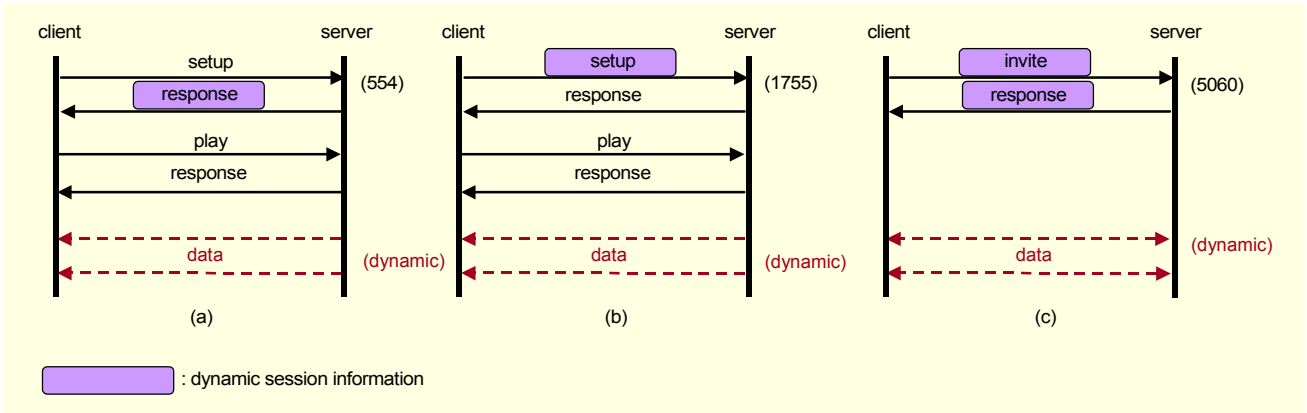

Fig. 4. A dynamic session negotiation: (a) RTSP, (b) MMS, and (c) SIP.

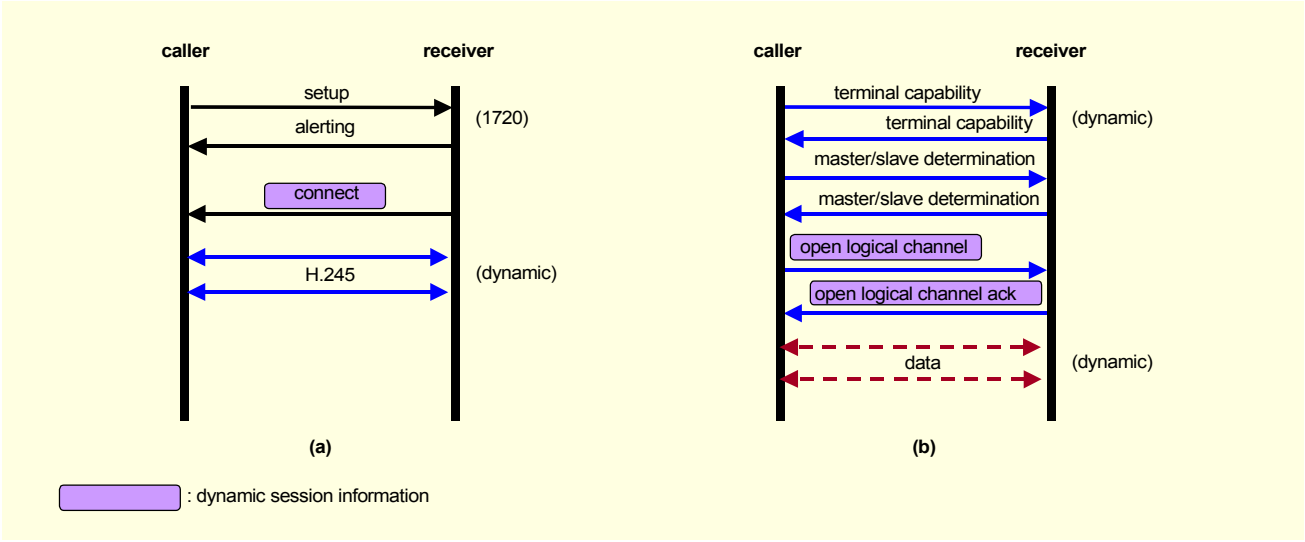

Fig. 5. The dynamic session negotiation in H.323: (a) Q.931, (b) H.245.

Q.931 session (lines 16 and 17 in Fig. 3). Then, the procedure selects an open logical channel or open logical channel ack message in the H.245 session.

The payload of the selected packet is parsed according to each control protocol (lines 6, 9, 11, 14, or 16 in Fig. 3). According to the protocol specification, the procedure matches the payload with a regular expression, as described in Table 3. In RTSP, the analyzer parses the payload and searches for the string components: "Transport:", "; client_port=", the number or range of numbers, and ";". These string components appear in the payload of the SETUP request's response. Although we have not ascertained the specification of MMS, we can analyze by searching for strings: "MMS", 'URL-string format', "TCP" or "UDP," and the port number. In SIP, the procedure extracts dynamic session information from a Session Description Protocol (SDP) [17] part. It finds a media header, which consists of components such as "M=", media type, port number, transport protocol, and payload type.

Contrary to the above text-based protocols such as RTSP, MMS, and SIP, the procedure searches for the locations of the port number in Q.931 or H.245 packets. Figure 6 shows parts of the payload that contains the dynamic session information in the Q.931 and H.245 packets. In Q.931, the dynamic session analyzer extracts a dynamically assigned port number from a port in the User-User info Element information. It can find the port number of H.245 in the tsap Identifier of a forwardLogicalChannel, reverseLogicalChannel, or a network access parameter.

After parsing, the procedure confirms whether or not the dynamic session information is discovered (line 19 in Fig. 3). If so, this information is stored into the dynamic session table that contains information on active dynamic sessions (line 20 and 21 in Fig. 3).

When a multimedia service is completed, information on the dynamic session must be removed. This information is usually deleted from the dynamic session table (line 23 in Fig. 3) when the TCP FIN flag is set. A packet with a FIN flag is generated when a control session is disconnected by operations such as TEARDOWN in RTSP in order to terminate a service. Most terminated services are detected by using the FIN flag. However, the FIN packet may never be captured because of such effects as packet losses [3]. In such cases, the information

Table 3. A regular expression of dynamic session information.

Control protocol	Regular expression
RTSP	[Tt][Rr][Aa][Nn][Ss][Pp][Oo][Rr][Tt]:[a-zA-Z]+[/-][a-zA-Z])*.*; client_port=[1-9][0-9]{3,4}(-[1-9][0-9]{3,4}){0,1};.*
MMs	\\\\[1-9][0-9]{1,2}\.\\[1-9][0-9]{1,2}\.\\[1-9][0-9]{1,2}\.\\[1-9][0-9]{1,2}\\ [([Tt][Cc][Pp])|([Uu][Dd][Pp])]\\[1-9][0-9]{1,2}[0-9][0-9].*
SIP	[Mm]=[a-zA-Z]+[1-9][0-9]{3,4}[a-zA-Z]+([/-][a-zA-Z])*. *

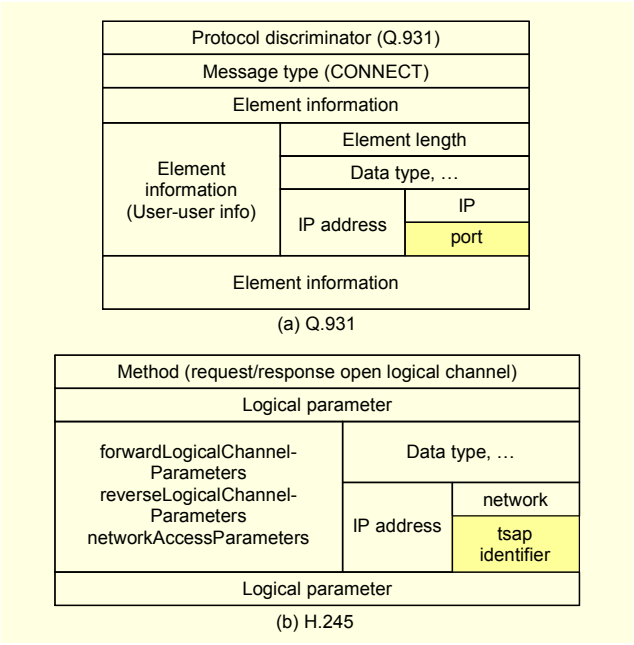

Fig. 6. The payloads of Q.931 and H.245.

is removed from the table by selecting a session, which shows no activity for a certain period of time. Since some packets are transferred to hold a connection even when a service is in a pause state, this method is used only if the timeout period is longer than the time required to send the packets to hold the session.

In this paper, we consider only well-known ports used for a TCP control session establishment. In the future, we want to incorporate the use of unknown ports for establishing a TCP control session, although such situations are rare.

IV. An Architecture for Multimedia Service Traffic Monitoring and Analysis

We have developed a system for monitoring and analyzing multimedia service traffic. The proposed method is designed as modules and is added to Next Generation MONitoring (NG- MON) [12], which is a network traffic monitoring and analysis system based on flows. As illustrated in Fig. 7, tasks are divided into several phases, which are serially interconnected using a pipelined architecture. Each phase can be executed on separate computer systems and cooperates with adjacent phases using pipeline processing. These phases can be composed of a cluster of computers wherever the system load of the phase is higher than the performance of a single computer system. Each phase can be replaced with more optimized modules as long as they provide and use the same interfaces.

Because of this flexible architecture, we could easily integrate our modules with NG-MON. As illustrated in Fig. 8, a dynamic session analyzer has been added between a packet capturer and a flow generator to perform an analysis on the dynamic sessions. Also, a dynamic relation mapper is appended before a traffic analyzer. The other modules, such as the flow generator and traffic analyzer, have the same functionality as that of NG-MON. Below, we describe each phase in detail.

1. Flow Generation

Flow generation consists of a packet capturer, a flow generator, and a flow store. The packet capturer collects the packets passing through a probing point. Previously, NG-MON has captured only the header of the packets. However, NG-MON for multimedia traffic analysis captures the payload as well as the header of the packets. Another function of the packet capturer is to extract information from the packet header and send it to the flow generator. The format of the packet header information is also shown in Fig. 9. The time stamp represents the time when the packet is captured.

By collecting a series of packets, the flow generator creates a flow which is defined in our system as a sequence of packets with the same 5-tuple: source IP address, destination IP address, source port, destination port, and protocol number. Figure 9

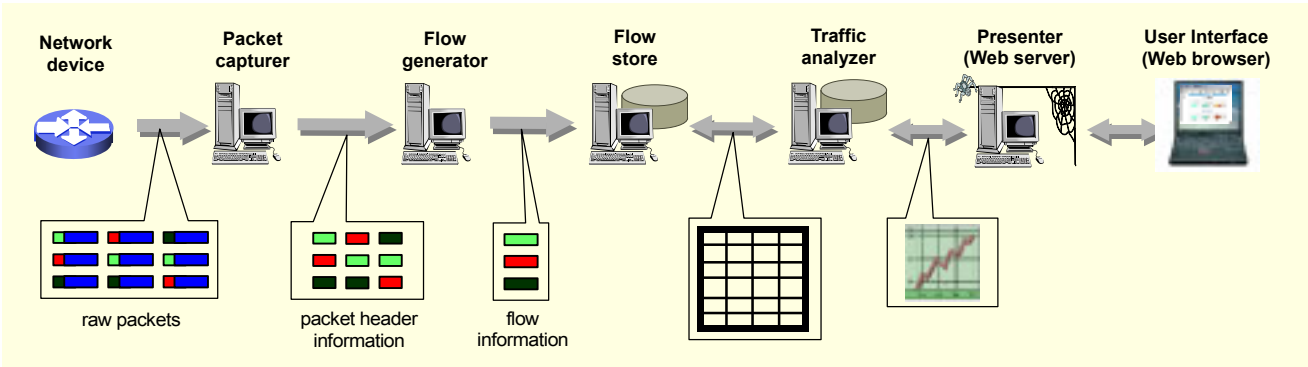

Fig. 7. The architecture of NG-MON.

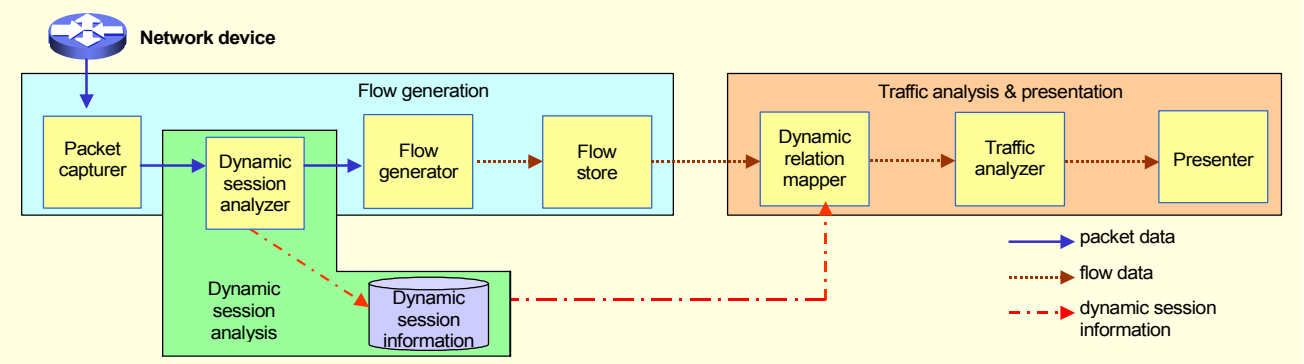

Fig. 8. A multimedia service traffic monitoring architecture.

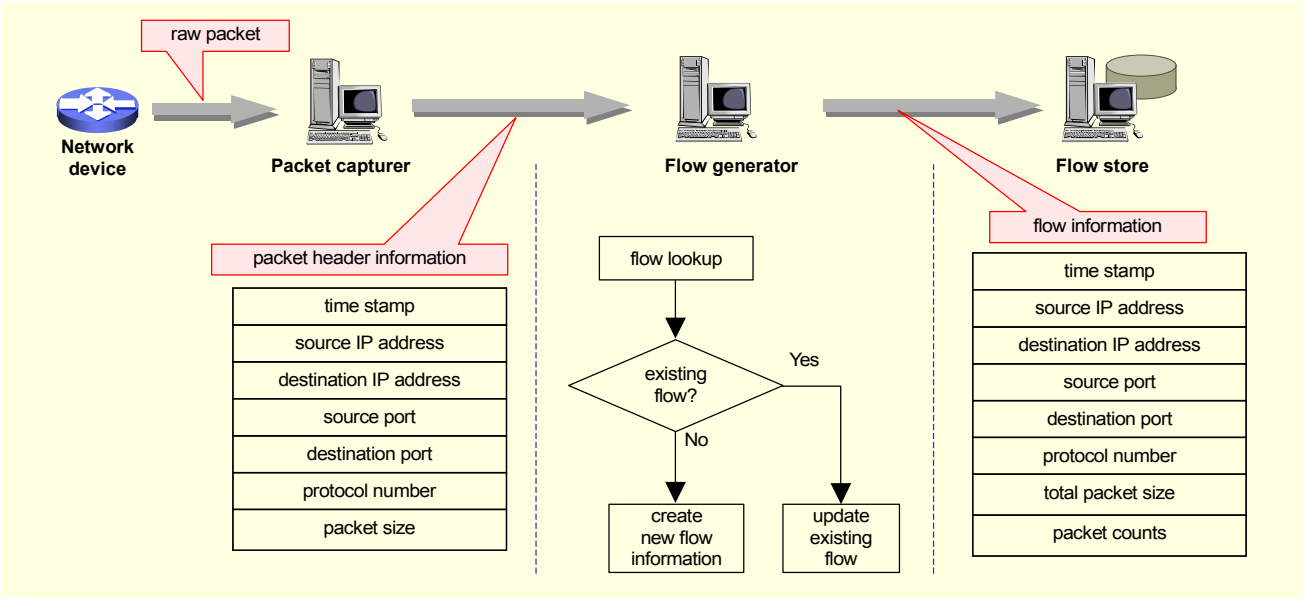

Fig. 9. A flow generation module.

illustrates the function and parameter of the flow generator. The flow generator stores the flow data in its memory area for processing. When receiving the packet header information, the flow generator searches the corresponding flow data from its flow table and then updates it, or creates a new flow if one does not already exist. Packets in the same flow are associated into the same entry of the table by increasing the packet count and adding the length to the total packet size. While being collected and converted to a flow, fragmented packets are reassembled in this phase. The number of packets or flows to be created depends on the distribution of packets, how much volume is generated, and how many packets are in the flow. The flow generator periodically exports the flow data to the flow store, and the flows are then stored in a database. Here, the period can be configured according to the flow time-out in order to associate the flow information during a predetermined time, such as one minute. After being sent, the flows are deleted immediately from the flow generator. However, flow information is stored for three hours in the flow store for data backup.

Our system mainly analyzes traffic distribution such as traffic volume, packet counts, and application. Therefore, we only capture and measure packets passing through the measuring points. We are not concerned with packets that do not pass through this probing point (i.e., lost packets). We count only the number of packets and bytes. Therefore, the out of order packets do not affect our analysis.

2. Dynamic Session Analysis

Dynamic session analysis provides information for identifying multimedia service traffic. As shown in Fig. 10, a dynamic session analyzer is composed of two modules: a control decider and dynamic parser. The control decider determines whether a captured packet is a control packet by

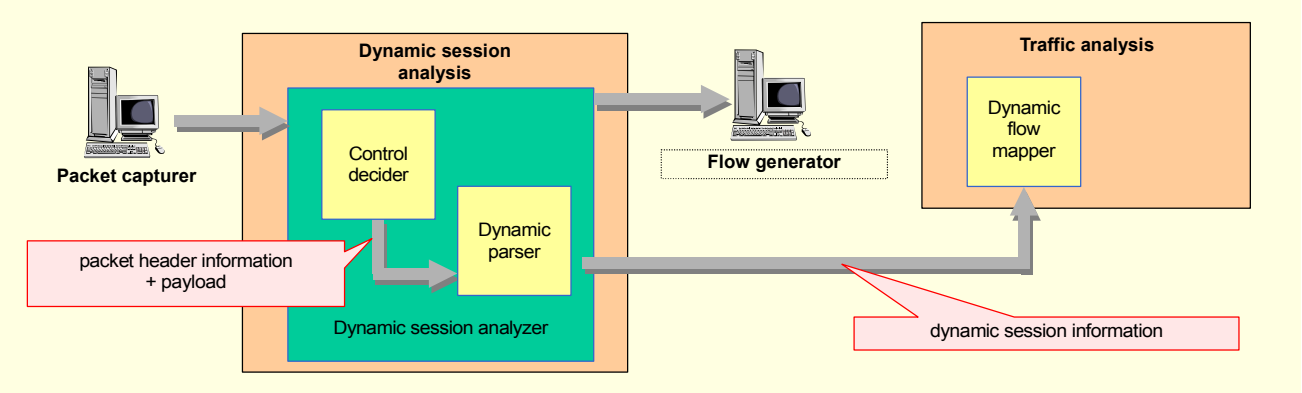

Fig. 10. A dynamic session analyzer.

checking the port numbers (RTSP, MMS, SIP and Q.931) or by looking up the dynamic session table (H.245). If so, this module sends the packet to the dynamic parser. By using the method for dynamic session analysis described in Section III, the dynamic parser discovers the information on the dynamic session. This information is stored into the dynamic session table and referred by the dynamic relation mapper during traffic analysis.

Table 4 shows the format of the dynamic session information. In this format, the control server address and control server port are the IP address and port number of the server in the control session that created the dynamic session. Similarly, the control client address and control client port are the IP address and port number of the client in the control session. With this information, the system is aware of the relationship between the control and dynamic sessions. The session start time is the time when the dynamic session information is newly created, and the session end time is the time when the control session is disconnected. The session end time is set either when a TCP FIN flag of a control packet is set, or when no packet in the same session is captured during a threshold time. These session time fields are used to determine whether a dynamic session is active or inactive. The threshold time is determined by our observation. After measuring the interval periods in which several applications send packets to hold the connections, we used the time that is much longer than the observed interval times.

It may happen that a client first sends a SETUP request to establish a session with the server and then terminates it without a DISCONNECT message. That is, if the TCP FIN flag is not set for some session, then the session is observed by the system for a predetermined time (currently 60 minutes). If no packet in the same session is captured during this time, the session is considered to be inactive, and the session end time is set in the dynamic session information. Thus, our system can handle this situation without any inconsistency in port numbers or increase in the database size. This situation can also be handled by our system because we observe the session for a predetermined threshold time to decide whether the session is active or inactive.

Table 4. The dynamic session information format.

control server address	control server port	control client address	control client port
data client address		data client port	
transport protocol		session start time	session end time

3. Traffic Analysis and Presentation

To identify the multimedia service application of traffic, a dynamic session mapper is added between a flow store and traffic analyzer of NG-MON. Only multimedia service traffic is selected by this mapper and sent to the traffic analyzer.

The dynamic relation mapper decides the relation between a dynamic flow and a control flow. This module identifies whether a flow with an unknown port number is related to a dynamic session. As illustrated in Fig. 11, this module matches the flow information with the dynamic session information. The tuples to be compared are as follows: destination (or source) IP address and dynamic client address; destination (or source) port and dynamic client port; and protocol number and transport protocol. If the compared tuples are equal, some fields are added to flow information, such as the IP address and port number of the control session. By adding these fields, we can map the dynamic flow and the control flow that creates the dynamic flow. In the case of control flows, the control server and client information are filled up with its own IP address and port number. Otherwise, the flow information is sent again to the mapper to be compared again in the next period.

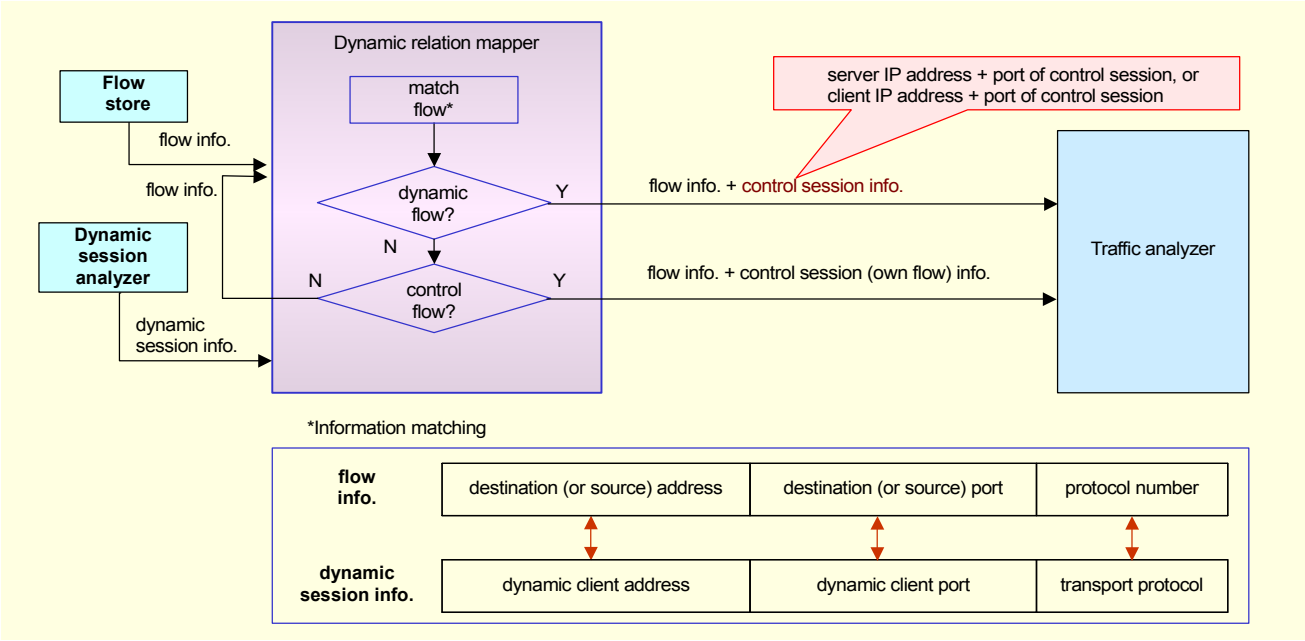

Fig. 11. A dynamic relation mapper.

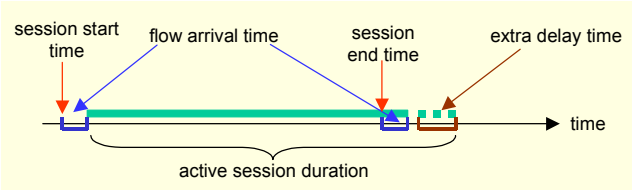

Fig. 12. An active session duration.

Because dynamic sessions are frequently created and destroyed, we need to validate whether the dynamic session is active. Figure 12 illustrates the times used to denote the session duration. The session start and session end times can be obtained from the dynamic session information. The flow arrival time is the time it takes for a flow to travel from the packet capturer to the relation mapper, which is the summation of the following two periods: the time required in the flow generator (one minute for our system) and the time required in the flow store (one minute for our system). The flow arrival time is not changed unless the system configuration is changed. Generally, the traffic analyzer determines that a session is active when the present time lies between two time fields, (session start time + flow arrival time) and (session end time+ flow arrival time).

Although the probability is small, it is possible to commit a false-rejecting error, where the real data packet is misidentified as being unassociated with the multimedia service session. This problem is caused by delays when the information in the dynamic session table is applied too early or too late. For example, some data packets may pass a probing point before a control packet that contains the dynamic session information because of the different routing paths. Conversely, data packers may follow the control packet that disconnects the sessions after terminating the service.

We have developed two schemes to solve this problem. For the prevention of a too-late application, we leave the unknown traffic in the flow table, although it was previously judged as not being multimedia service traffic. This flow is again matched with the dynamic session information in the next database update time. As a result, the flow has the opportunity to be double-checked to verify that it is a dynamic session of the multimedia service traffic. Because the false-rejecting phenomenon occurs in a short delay relative to the period of updating the database, this scheme gives sufficient time to apply the information in the streaming session table. For the prevention of a too-early application, we leave the information in the dynamic session table during the extra delay time after the point when the session end time is set. We have observed that some packets pass the probing point even though the session has been terminated. This phenomenon may occur because of different routing paths that the packets travel through. While maintaining the dynamic session information for a while (e.g., 10 seconds), we apply it to the packets that arrive late. As this delay is short, the port might not be assigned to another application. Even if the port is assigned to another application, there is no confusion about the application using the port because we check the port number with the other parameters such as the server address.

The traffic analyzer performs an analysis of the traffic by querying the flow data stored in the database. It can analyze

multimedia service traffic at the session level. Multimedia services are likely to open several sessions. The traffic analyzer can discover and analyze sessions separately. By identifying the type of the session, such as a control or data session, it generates such information per session as the transferred packet count. In addition, it integrates the information of the sessions that belong to the same multimedia service. For example, it can analyze the traffic volume exchanged in the control and data sessions related to the same multimedia service.

The presenter shows the analyzed data corresponding to the request of a user through the web server. Employing traffic prepared by the traffic analyzer, it can provide a user with information in a fast access time.

V. Implementation

In this section, we present system implementation and performance tests of the proposed system. The deployed system and environments are illustrated in Fig. 13.

Network traffic is injected into a packet capturer in our gigabit campus backbone network through a network splitter, which can copy network traffic passively. The packet headers are analyzed and converted into flows every minute. In the case of a control packet, the dynamic session analyzer parses its payload and extracts the dynamic session information from it. Using this information, the dynamic relation mapper identifies the dynamic flows and stores the traffic information into the database. According to the user requests, the presenter shows this traffic information through the Web. The Network Time Protocol [18] is used to synchronize the time of the machines involved.

By applying our proposed method, we can monitor and analyze multimedia service traffic. Our method can identify dynamic flows that were formerly identified as unknown traffic. As an example of our analysis results, Fig. 14 illustrates the information on WMT traffic that appeared during a one-minute period. The table on the left summarizes information such as total packets and bytes. The graph plots WMT traffic every minute during a one hour span. The graph shows 0 to 33 min only, as the output was taken at 3 hr 33 min. This presenter collects and shows several flows generated by the identical WMT service between two hosts. The below table discerns the flows as control and data sessions. As shown in the table, the majority of multimedia service traffic is the data flow which is classified previously as unknown traffic.

While working with our gigabit campus backbone network, the system handles 15,000 flows, 2,500,000 packets, and 1.5 GB per minute on average. It can also deal with a maximum of 50,000 flows, 5,000,000 packets, and 3.5 GB per minute. Additionally, we have verified our system performance, particularly of the dynamic session analyzer, by measuring the resource utilization in each phase, as shown in Fig. 15. The test environment is as follows. The packet capturer, dynamic session analyzer, and flow generator are executed in the same machine, which has a 2.4-GHz CPU, 1 GB of memory, and a 1 gigabit network interface card with Red Hat Linux 8.0 as an operating system. This machine is connected to a traffic generator by a gigabit network link. We transfer 50,000 packets per second,

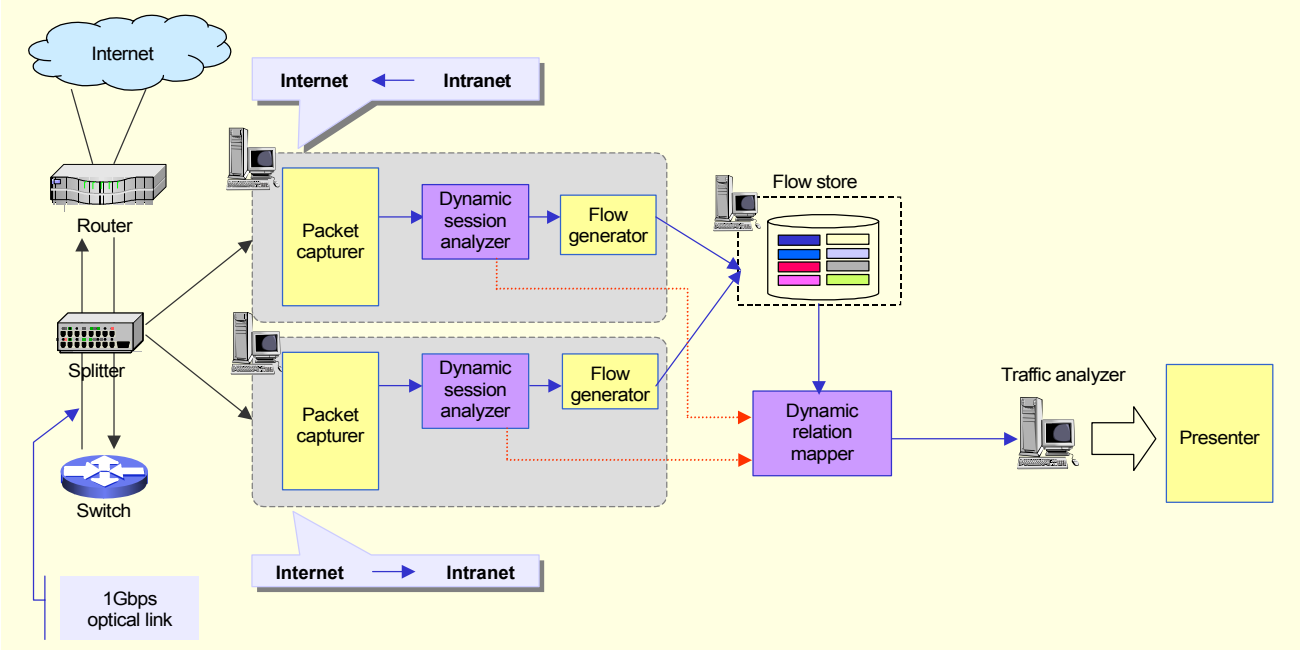

Fig. 13. The system implementation architecture.

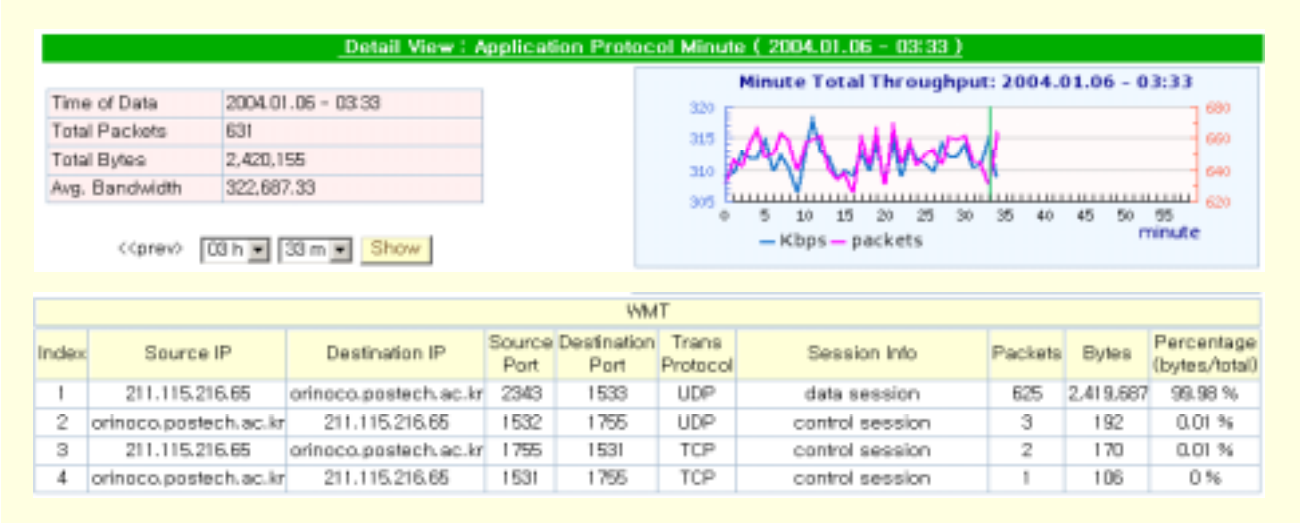

Fig. 14. An analysis on WMT streaming service.

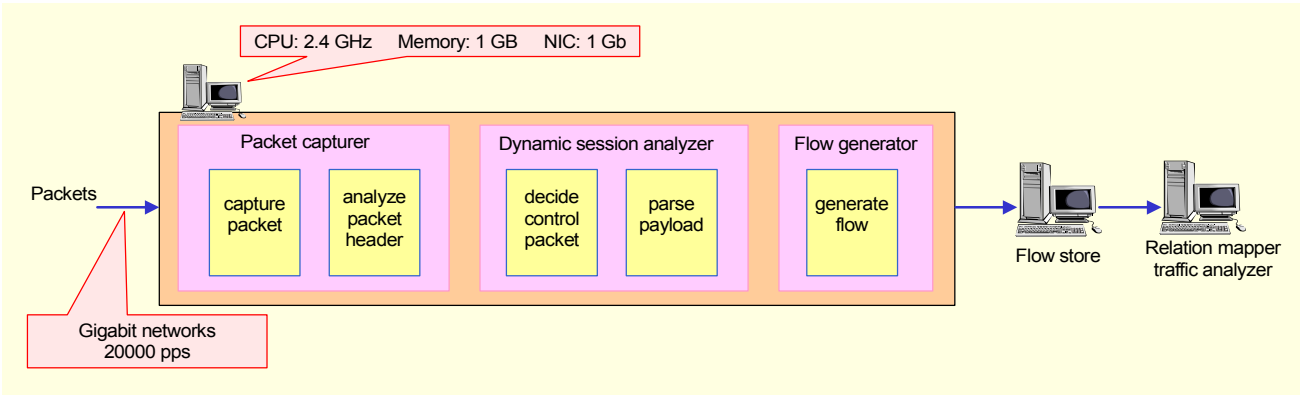

Fig. 15. Phases in testing performance.

generated by 15,000 flows by using SmartBits [20].

First, we measure the additional system overhead caused by capturing packets with payloads. This overhead depends on the performance of the capture filter, libpcap [19]. We generate packets of 1,500 B in size and capture packets of varying sizes. Figure 16 presents the changes in CPU utilization according to changes in the captured packet size. The memory utilization has a constant value of 0.6% for all captured packet sizes. As the captured packet size increases, the system overhead (in terms of CPU utilization) also increases. Thus we wish to capture packets with the smallest size possible. In most cases, the dynamic session information is contained in the first 120 B of payload in a control packet. Therefore, we capture packets of 200 B in size, which is the summation of the packet header length (64 B) and payload (120 B).

Next, we create the following test environment to measure performance of each phase in identical conditions. We transmit 50,000 packets of 1,500 B in size per second, utilizing a 600 Mbps bandwidth. As SmartBits cannot generate packets with a payload that contains dynamic session information, we should simulate an invocation of payload parsing modules. A parsing module is executed for every N packets, where N depends on the percentage of control protocol packets. For example, if 1,000 packets are captured and N for RTSP is 20, the RTSP parsing module will be executed 50 times (1,000/20 = 50). The CPU overhead due to changes in the number of executions of the parsing module is shown in Fig. 17. In every test, each protocol is parsed at the following rates: RTSP at 30%, MMS at 50%, SIP at 5%, Q.931 at 10%, and H.245 at 5%, which reflects our monitoring results of the POSTECH campus network. The CPU overhead gradually increases with an increase in the percentage of control packets. But when all the packets are control packets, the maximum overhead is not more than 20%. Memory overhead by this parsing module is close to zero. In fact, we have observed that the POSTECH campus network has a small number of control packets, which

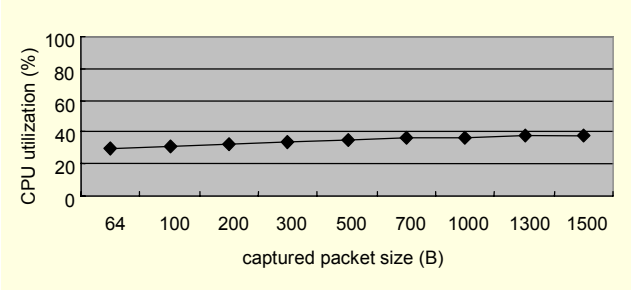

Fig. 16. CPU utilization in packet capturing.

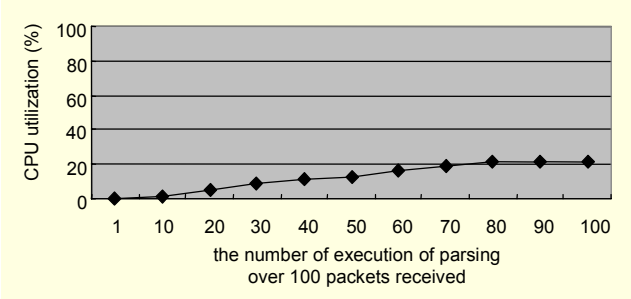

Fig. 17. CPU overhead due to parsing execution.

is less than 0.1% out of all the packets. Therefore, the parsing payload does not largely affect the overall system overhead.

Table 5 indicates the system overhead in each phase. The second row represents the memory utilization of each phase. Memory is used significantly in the packet capture and flow generation phases. The remaining phases have negligible memory consumption. The CPU utilization at each phase is presented in the third row. The CPU is consumed mostly during the packet capture (30 %), packet header analysis (8 %), and flow generation (35 %) phases. CPU utilization increases when we capture packets with a payload (200 B) due to the dynamic session analysis.

If an application does not use a well-known port number as a control channel, we cannot monitor multimedia traffic. Therefore, we considered parsing N bytes of each packet. However, this resulted in a severe system overhead because we had to parse N bytes of all the captured packets according to the specifications of all the control protocols. The results of our performance test indicates that this method suffers from a severe system overhead. It consumed 100% of the CPU utilization and lost processing data. As a result, we determined to parse only the control packets by checking the packet header information. At the control decision phase, the decision is made by only checking the port number of the control packets that use well known-port numbers or by validating the hashed values in the dynamic session table. As this phase involves only a few operations, it does not lead to a large CPU overhead. Similarly, payload parsing does not cause significant system overhead because of the small number of control packets. The fourth row shows additional CPU overhead due to the dynamic session analyzer, which includes the overhead of capturing payloads of the packets, the control decision, and payload parsing. These results show that these modules do not induce a large system overhead.

We deployed our system in the campus network and collected traffic over a period of one day. Table 6 demonstrates the overall statistics of an application layer traffic analysis. The first column describes the total number of flows captured by the system from the campus network over one day. Columns 2 and 3 show the total number of packets and total number of bytes captured.

Table 7 describes the percentage distribution of applications detected by our system over the one day period. The first row represents the distribution for applications (Real Networks and QuickTime) that use the RTSP protocol. The total number of

Table 5. The system overhead in each phase.

Utilization	Packet capture	Packet header analysis	Control decision	Payload parsing	Flow generation
Memory (%)	0.6	≅ 0	≅ 0	≅ 0	2
CPU (%)	30 (64 bytes)	8	1	≅ 0	35
	33 (200 bytes)				
CPU (%) (Additional)	3	N.A.	1	≅ 0	N.A.

N.A. : not available

Table 6. The overall statistics of an application layer traffic analysis.

Total flows	Total packets	Total bytes
151,059,362	3,225,390,089	2,044,744,537,573

Table 7. The percentage distribution of applications detected.

Application	Flows		Packets		Bytes	
RTSP	23,409	0.02%	9,021,694	0.28%	9,525,109,414	0.47%
H.323	217,615	0.14%	851,592	0.03%	58,943,437	0%
WMedia	141,316	0.09%	34,774,078	1.08%	37,075,008,767	1.81%

flows captured for RTSP are 23,409, which is 0.02% of the total flows captured. Similarly, the total number of packets captured for RTSP is 9,021,694. RTSP packets are 0.28% out of the total packets captured, and the total number of bytes for RTSP is 9,525,109,414, which makes up 0.47% of the total bytes captured. Similarly, the second and third rows represent the distribution for H.323 and Windows Media player. This analysis shows that the users in the campus network use Windows media more than RTSP and H.323, which is four times the byte percentage of RTSP.

VI. Related Work

In this section, we describe two multimedia service traffic analysis methods, namely a heuristic method and a selective capturing method, and two traffic identification methods. As mmdump [3] is proprietary, we could not get the source code from the authors. Therefore, we have compared it with the method used by mmdump. We tested it by changing a capture filter whenever new dynamic session information was found in our system.

A heuristic method works as follows. First, it records ongoing control sessions. When a flow is seen with an unknown port number on two hosts, it checks to verify whether an active control connection exists between the same hosts. If so, it assumes that the flow corresponds to a dynamic session. This method is adopted by Flowscan [13], a flow-based traffic analysis system.

A selective capturing method is used by mmdump [3], a tool for monitoring multimedia traffic on the Internet. This method parses the control messages to extract the dynamically assigned port numbers. The parsing module then dynamically changes a packet filter to allow packets associated with these ports to be captured.

Table 8 presents the comparison results of different methods of a multimedia service analysis. The heuristic method can analyze traffic by capturing only the packet header, but it must capture all the packets passing through a probing point. This analysis may provide inaccurate information. In a false-positive problem, traffic seen with an unknown port number may not be related to the active control connection that exists between two connected hosts. For example, in scanning, many data sessions are created between the two hosts, but none of these relates to the active control connection between them. In a false-negative problem, some multimedia service data can be transferred from another source that does not participate in the active control connection. In this case, the heuristic method misidentifies the multimedia service traffic as unknown-traffic because no active control connection exists between the hosts transferring multimedia service data.

The selective capturing method suffers from a system overhead because it captures an entire packet including the payload. On the other hand, it can reduce the resource requirements and capture overhead by capturing only packets that contain listed port numbers. Because this module parses the payload and extracts the exact port numbers to be used in a dynamic session, it does not cause false-positive problems. However, it may generate false-rejecting errors. As mentioned in mmdump [3], this method misses packets while changing a capture filter. The results are even worse if a missed packet contains dynamic session information. As a result, many packets which belong to a data session negotiated by the missed packet may pass the probing point without being captured. This method also cannot hold dynamic port information after changing a filter. Once a filter is changed, the previous port number is useless. Thus, it cannot capture packets that arrive late after deleting the dynamic port information. Furthermore, it cannot handle IP-fragmentation. During our tests we observed that about 40 to 70% of WMT packets are fragmented. Similarly, some applications send large fragmented data streams into the network. However, capturing libraries such as libpcap [19] used in this method cannot identify the port numbers of fragmented packets because the port numbers are contained in other packets. As a result, this method misses fragmented packets even though they are multimedia service packets.

Another related study is the Signature Mapping-Based Method [21], which performs traffic identification by considering all IP level traffic and does not concentrate on streaming traffic alone. In a Signature Mapping-Based Method,

Table 8. Comparisons of multimedia service analysis methods.

Method	Heuristic method	Selective capturing method	Proposed method
Capturing overhead (packet size)	High	Low	High
Capturing overhead (packet number)	Low	High	High
Accuracy	Inaccurate	Accurate	Accurate
Packet loss	No	Yes	No
Fragment handling	Can handle	Cannot handle	Can handle

the portion of the payload data that is static, unique, and distinguishable from other packets is examined for all applications regardless of the protocol they are using, standard or proprietary, and is marked as a signature for an application. By comparing every packet payload with pre-determined signatures, this method can identify application traffic more accurately than the traditional method.

However, this method requires a lot of offline work to discover the signatures of individual applications. This method can be used to identify streaming media traffic. However, some difficulties exist in using it. While identifying streaming media traffic, this method causes severe system overhead in the process of packet capture and payload examination because we need to compare the pre-determined signature with the payload of all the packets. Also, to apply this method to the on-line and real-time traffic analysis system, it is necessary to refine and fine-tune the basic algorithm for the target environment.

The proposed payload examination method can be applied to P2P traffic identification [22], because most of the P2P applications use dynamically generated port numbers for content delivery. However, difficulty arises due to the use of proprietary protocols. Also, some P2P applications transfer encrypted data which makes the payload examination impossible. Recent research [23] on the identification of traffic type gave up identifying the traffic according to application. Instead, they proposed a new traffic type, called TCP-Big, which is the aggregation of unknown flows that transmit more than 100kB in less than 30 minutes, and showed that the TCP-Big traffic has almost the same properties as P2P traffic (2003). Our proposed method suffers from capturing overhead since it captures all packets with each packet's payload. However, our method provides accurate information by extracting the exact dynamic session information through a parsing of the payload. It does not stop the capturing process because it does not change the packet filter. Further, the method considers IP-fragmentation. Because the method reassembles fragmented packets in a flow generator, it can identify the port numbers of fragmented packets.

VII. Concluding Remarks

In this paper, we presented a method and system for monitoring and analyzing multimedia service traffic. This method analyzes control protocol messages and extracts information on dynamic sessions. The extracted information includes dynamically selected protocol and port numbers, which are used to determine whether or not the unknown traffic is multimedia traffic. This approach makes it practical to monitor previously unknown multimedia service traffic and other services.

This method boosts the analysis of traffic from the packet level to the session level. It does not simply extract the header information of a packet but also makes it possible to analyze traffic per session by acquiring session information. In addition, it overcomes problems with existing approaches that use only well-known port numbers of TCP or UDP for identifying the application of traffic. By analyzing application messages, this method discovers the status of the application and raises the analysis to the application level.

Our multimedia service traffic analysis method is integrated with NG-MON, through which we are currently monitoring Internet traffic between our campus and the Internet. Although we have not yet applied our system to ISP networks, we leave this as future work.

The packet payload examination becomes difficult in secure channel streaming due to the use of encrypted data. In the future, we want to extend the proposed analysis method to support streaming over secure channels as well. Also, we are planning to provide support for other types of traffic that create and use dynamic sessions.

References

[1] Internet Assigned Numbers Authority, http://www.iana.org/.

[2] James W. Hong, Soon-Sun Kwon, and Jae-Young Kim, "WebTrafMon: Web-Based Internet/Intranet Network Traffic Monitoring and Analysis System," *Computer Comm., Elsevier Science*, vol. 22, no. 14, Sept. 1999, pp. 1333-1342.

[3] Jacobus van der Merwe, Ramon Caceres, Yang-hua Chu, and Cormac Sreenan "mmdump–A Tool for Monitoring Internet Multimedia Traffic," *ACM Computer Comm. Review*, vol. 30, no. 4, Oct. 2000.
[4] J.Craig Lowery, "Using Dell PowerApp.cache for Caching and Splitting Media Streams," http://www.dell.com/us/en/esg/topics/power_ps3q01-lowery.htm, May 2001.
[5] Microsoft, Windows Media Technology, http://www.microsoft.com/windows/windowsmedia/default.asp.
[6] Real Networks, Real Media Technology, http://www.realnetworks.com/.
[7] Apple, QuickTime, http://www.apple.com/quicktime.
[8] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," *RFC 2543*, Mar. 1999.
[9] ITU-T, "Recommendation H.323: Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service," 1996.
[10] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," *RFC 2336*, Apr. 1998.
[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," *RFC1889*, Jan. 1996.
[12] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju, and James W. Hong, "The Architecture of NG-MON: A Passive Network Monitoring System", *Lecture Notes in Computer Science 2506, 13th IFIP/IEEE Int'l Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, Montreal, Canada, Oct. 2002, pp. 16-27.
[13] Dave Plonka, FlowScan, http://net.doit.wisc.edu/~plonka/FlowScan/.
[14] Siegfried Lifler, "Using Flows for Analysis and Measurement of Internet Traffic," *Diploma Thesis*, Institute of Comm. Networks and Computer Engineering, University of Stuttgart, 1997.
[15] J.Quittek, T. Zseby, B. Claise, and K.C. Norsth, "IPFIX Requirements," *Internet Draft*, http://norseth.org/ietf/ipfix/draft-ietf-ipfix-architecture-00.txt.
[16] CAIDA, "Preliminary Measurement Spec for Internet Routers," http://www.caida.org/tools/measurement/measurementspec/.
[17] M. Handley and V. Jacobson, "SDP: Session Description Protocol," *RFC 2327*, Apr. 1998.
[18] NTP, http://www.ntp.org/.
[19] libpcap, http://www.tcpdump.org/.
[20] SmartBits, http://www.spirentcom.com/.
[21] T. S. Choi et. al., "Rate-Based Internet Accounting System Using Application-aware Traffic Measurement," *Proc. of 2003 Asia-Pacific Network Operations and Management Symp.(APNOMS 2003)*, Fukuoka, Japan, Oct. 1-3, 2003, pp.404-415.
[22] Subhabrata Sen and Jia Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. of the second ACM SIGCOMM Workshop on Internet Measurement Workshop*, Nov. 2002.
[23] Alexandre Gerber, Joseph Houle, Han Nguyen, Matthew Roughan, and Subhabrata Sen, "P2P The Gorilla in the Cable," *National Cable & Telecommunications Association (NCTA) 2003 National Show*, Chicago, IL, June 8-11, 2003.

Hun-Jeong Kang received his BS and MS degrees in computer science and engineering from Pohang University of Science and Technology (POSTECH) in 2001 and 2003, respectively. Currently, he is a Researcher in the Department of Computer Science and Engineering, POSTECH. His research interests include network traffic monitoring and network security. He is a Member of KNOM.

Myung-Sup Kim received his BS and MS degrees in computer science and engineering from POSTECH in 1998 and 2000, respectively. Currently, he is a PhD candidate in the Department of Computer Science and Engineering, POSTECH. His research interests include Internet traffic monitoring and analysis, application-lay traffic analysis, and network security attack detection and prevention. He is a Member of IEEE and KNOM.

James W. Hong is an Associate Professor in the Department of Computer Science and Engineering, POSTECH, Pohang, Korea. He has been with POSTECH since May 1995. Prior to joining POSTECH, he was a Research Professor in the Department of Computer Science, University of Western Ontario, London, Canada. Dr. Hong received the BS and MS degrees from the University of Western Ontario in 1983 and 1985, respectively, and the PhD degree from the University of Waterloo, Waterloo, Canada in 1991. He has been very active as a Participant, Program Committee Member and Organizing Committee Member for IEEE CNOM sponsored symposiums such as NOMS, IM, DSOM and APNOMS. For the last few years, he has been working on various research projects on network and systems management, which utilize Web, Java, CORBA and XML technologies. He is IEEE Comsoc CNOM Vice Chair and KICS KNOM Chair. His research interests include network and systems management, traffic monitoring and analysis, and security management. He is a Member of IEEE, KICS, KNOM, and KISS.