멀티미디어 서비스 트래픽 모니터링 및 분석 방법

강 훈 정
bluewind@postech.ac.kr

2002년 12월 13일

분산처리 및 네트워크관리 연구실
포항공과대학교 컴퓨터공학과

Contents

- Introduction
- Related Work
- Proposition of Methods
  - Analysis of dynamic session
    - RTSP, MMS, H.323, SIP
  - Identification and analysis of multimedia service flow
- System Design and Implementation
  - Requirements analysis and system architecture
  - Implementation and results
- Conclusion and Future Work

Introduction

- Needs of Traffic Monitoring
  - To understand the behavior of networks and usage pattern
  - To make policy of using network or billing
  - To provide information for network planning

- Problems
  - Trends: streaming and multimedia conferencing traffic has increased significantly
  - Difficulty in monitoring of streaming and multimedia conferencing traffic
    - Applications use dynamically allocated port numbers
    - Protocols are various, and some of them are proprietary

- Approach
  - Research multimedia service traffic characteristics and related protocols
  - Propose methods of monitoring and analysis applied to flow-based system
    - to extract dynamically assigned port numbers and protocol information
    - to identify and analyze multimedia service traffic
  - Implement system adopting proposed methods

Multimedia Service Protocols - Related Work

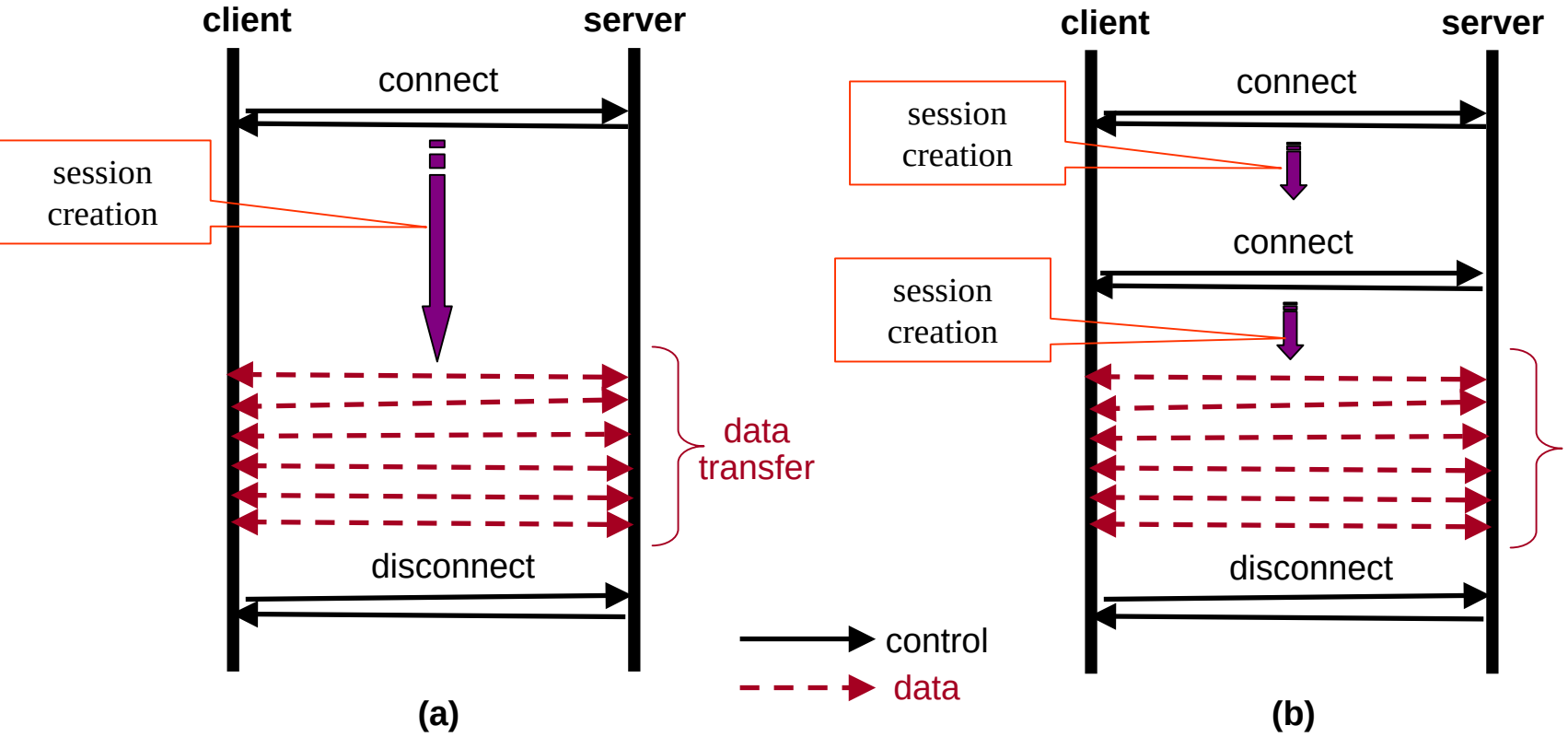
- Streaming services

Streaming service platform	Control Protocol	Data transfer protocol	Proposer
RealMedia	RTSP	RDT	RealNetworks
QuickTime	RTSP	RTP	Apple
WiMT	MMS	MMST or MMSU	Microsoft

- Multimedia conferencing services
  - H.323
    - Standards for multimedia communications over LAN
    - Signaling protocol : Q.931
    - Connection control protocol: H.245
  - SIP
    - Standard protocol for initiating session that involves multimedia elements
    - Establish, modify, or terminate multimedia sessions or Internet telephony calls
    - Use SDP for describing multimedia transfer

Multimedia Service Sessions - Related Work

- Two kinds of session are used during multimedia services
  - Control session: set up connection, control navigation
  - Data session: transfer streaming media data
  - Control session negotiate and create other session (Dynamic Session)



Related Monitoring Methods (1/2) - Related Work

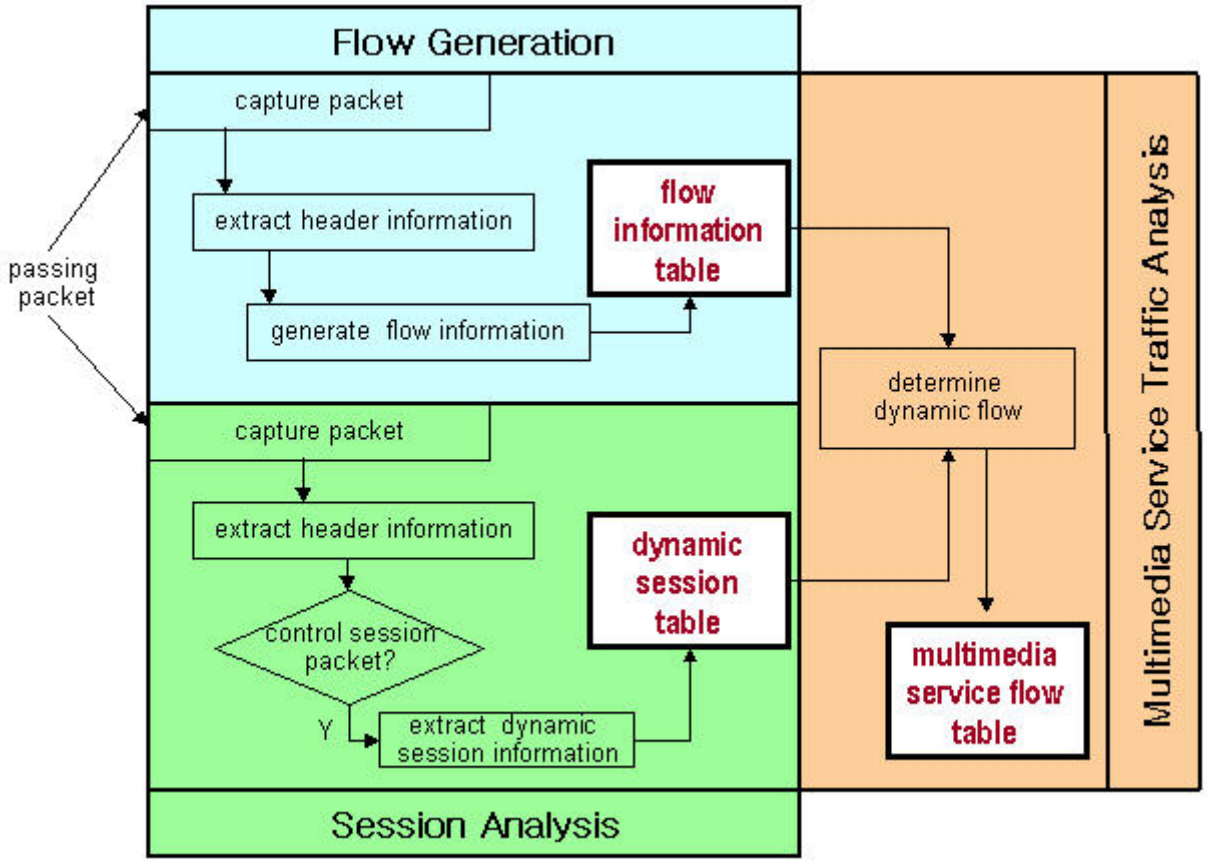
- Heuristic method
  - Used by FlowScan, FluxoScope
  - Monitored multimedia service traffic
    - RTSP
  - Operation
    - to remember ongoing control sessions
    - when a flow is seen with an unknown port number on both ends
    - to assume that the flow corresponds to a data session
  - Problems
    - Inaccuracy in assumption
      - Other types of traffic may exist even though control session exists on both ends
      - Data may be transferred via other multimedia server

Related Monitoring Methods (2/2) - Related Work

- Selective capturing method
  - Used by mmdump
  - Monitored multimedia service traffic
    - RTSP, H.323
  - Operation
    - to parse streaming control protocol to get dynamically assigned port number
    - changes the packet capture filter
  - Problems
    - IP fragmentation
      - limitation of monitored traffic (MMS is not analyzed)
      - cannot identify port number-->discard fragmented packet
    - inaccuracy
      - false reject
        - » cannot capture packet while changing filter
        - » data session packet may arrive after control session ends
    - difficulty in applying to other system
      - provide system overhead capturing payload of packet

Overall Procedure - Methods

- Methods of monitoring and analysis for multimedia service traffic
  - To be applied to flow-based traffic monitoring system
  - Flow Generation, Session Analysis, Multimedia Service Traffic Analysis



Session Analysis - Methods

- **To extract dynamic session information**
  - Capture packet with payload
  - Extract header information
  - Select control session packet
  - Parsing payload of the packet
    - RTSP, MMS, H.323, SIP
  - Find dynamic session information and store it

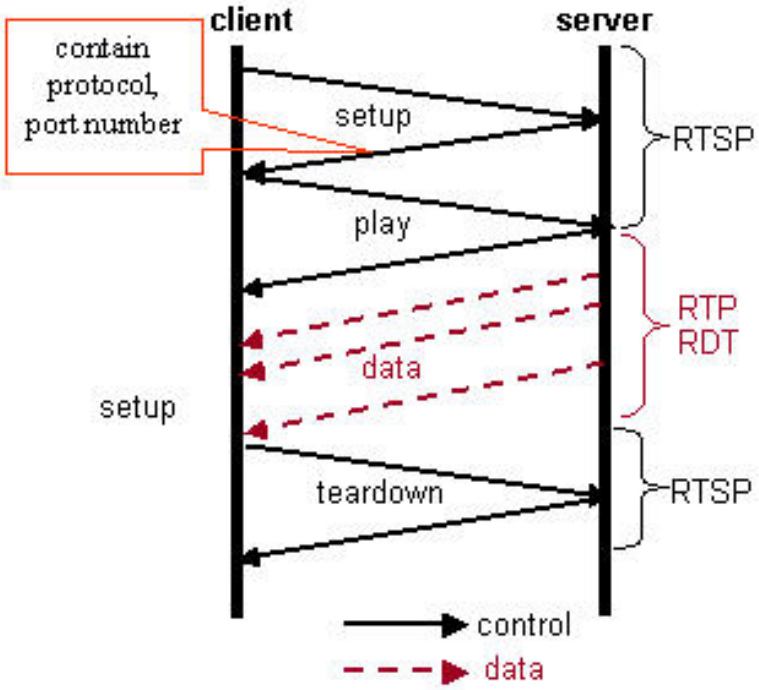
```
procedure SessionAnalysis ( var Msg : CONTROL SESSION MESSAGE,
                            var Table : SET of DINAMIC SESSION RECORD )
var
result : boolean ;
dsRec : DYNAMIC SESSION RECORD ;
Q931Rec : DYNAMIC SESSION RECORD ;
begin
if protocol of Msg =RTSP then
if TCP Flag.FIN of Msg is set then
goto DELETE_SESS;
 else result := ParseRTSP (Msg, dsRec);
else if protocol of Msg =MMS then
if TCP Flag.FIN of Msg is set then
goto DELETE_SESS;
else result := ParseMMS (Msg, dsRec);
else if protocol of Msg =Q.931 then
if. TCP Flag.FIN of Msg is set then
goto DELETE_SESS;
else result := ParseQ931 (Msg, dsRec);
else
if DYNAMIC SESSION RECORD from Msg ∈ Table then
Q931Rec := DYNAMIC SESSION RECORD in Table;
if TCP Flag.FIN of Msg is not set then
goto DELETE_SESS;
else
Q931Rec := DYNAMIC SESSION RECORD ;
result := ParseH245 (Msg, dsRec, Q931Rec);
else goto END_ALGO;                          { not control session packet}
if result = TRUE then
 add to dsRec to Table ;
goto END_ALGO;

DELETE_SESS:                                 { packet disconnects session}
set FIN time of DYNAMIC SESSION RECORD from Msg in Table;

END_ALGO:
 end ; {SessionAnalysis }
```

Session Analysis - RTSP (1/2) - Methods

- Select setup response message from server



Connection setup

```
procedure ParseRTSP( var Msg : CONTROL SESSION MESSAGE,
                     var Table : SET of DINAMIC SESSION RECORD )
var
msgHeader : array[1..256] of character ;
result : boolean ;
begin
if source port of Msg = RTSP server port number then      {response message}
msgHeader := read a line from Msg;
if msgHeader = RESPONSE MESSAGE then
while (there is a line to read in Msg ) do begin
msgHeader := read a next line from Msg;
if msgHeader = TRANSPORT MESSAGE then {transport parameter}
data client port of dsRec := PARSED PORT;
transport protocol of dsRec := PARSED PROTOCOL;
result : =TRUE;
else if msgHeader = SERVER MESSAGE then    {server application}
data application of dsRec := PARSED APPLICATION;
end
if result = TRUE then
data client address of dsRec := destination address of Msg;
control server address of dsRec := source address of Msg;
control server port of dsRec := source port of Msg;
control client address of dsRec := destination address of Msg;
control client port of dsRec := destination port of Msg;
   end ; {ParseRTSP}
```

Analysis control packet

Session Analysis - RTSP (2/2) - Methods

- Find ports from Transport Message

```
TSP/1.0 200 OK
erver: DSS/4.0 [v410]-Win32
seq: 3
ession: 36223754198775
ast-Modified: Thu, 18 Oct 2001 01:56:08 GMT
ache-Control: must-revalidate
ate: Thu, 07 Nov 2002 02:30:12 GMT
xpires: Thu, 07 Nov 2002 02:30:12 GMT
ransport: RTP/AVP;unicast;client_port=6972-6973;source=141.223.82.51;server_por
6970-6971;ssrc=00007F42
-Transport-Options: late-tolerance=1.500000
```

RTSP RESPONSE MESSAGE
Example payload of setup response
[Rr][Tr][Ss][Pp]/[0-9]+ \ .[0-9]+ 200 OK.*
SERVER MESSAGE
[Ss][Ee][Rr][Vv][Ee][Rr]: [a-zA-Z]+/]/[0-9]+ \ .[0-9]+
TRANSPORT MESSAGE
[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};.* \ n

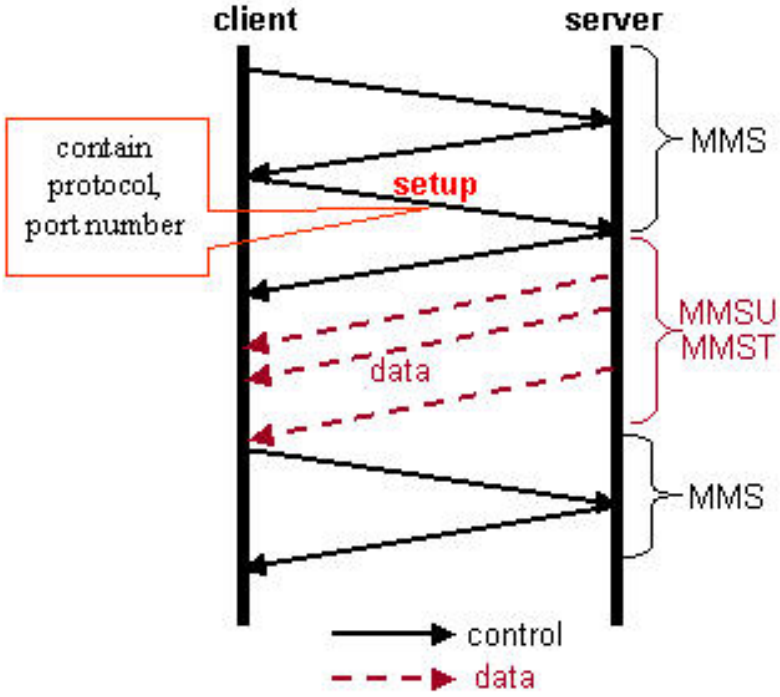
Regular expression

```
procedure ParseRTSP( var Msg : CONTROL SESSION MESSAGE,
                     var Table : SET of DINAMIC SESSION RECORD )
var
msgHeader : array[1..256] of character ;
result : boolean ;
begin
if source port of Msg = RTSP server port number then      {response message}
msgHeader := read a line from Msg;
if msgHeader = RESPONSE MESSAGE then
while (there is a line to read in Msg ) do begin
msgHeader := read a next line from Msg;
if msgHeader = TRANSPORT MESSAGE then {transport parameter}
data client port of dsRec := PARSED PORT;
transport protocol of dsRec := PARSED PROTOCOL;
result : =TRUE;
else if msgHeader = SERVER MESSAGE then     {server application}
data application of dsRec := PARSED APPLICATION;
end
if result = TRUE then
data client address of dsRec := destination address of Msg;
control server address of dsRec := source address of Msg;
control server port of dsRec := source port of Msg;
control client address of dsRec := destination address of Msg;
control client port of dsRec := destination port of Msg;
   end ; {ParseRTSP}
```

Analysis control packet

Session Analysis - MMS (1/2) - Methods

- Select setup request message to server

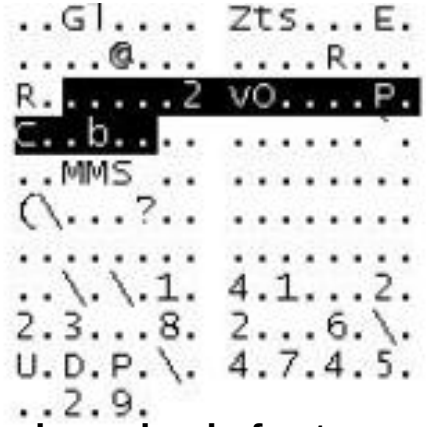


Connection setup

```
procedure ParseMMS ( var Msg : CONTROL SESSION MESSAGE,
                     var Table : SET of DYNAMIC SESSION RECORD )
var
msgHeader : array[1..256] of character ;
url : array[1..256] of character ;
begin
if destination port of Msg = MMS server port number then      {request message}
msgHeader := read 8 bytes from Msg;
if msgHeader = MMS START MESSAGE then
   msgHeader := read next 2 bytes from Msg;
if msgHeader = MMS SETUP MESSAGE then
while (there is a byte to read in Msg ) do begin
msgHeader := next byte from Msg;
if msgHeader := URL CHARACTER then
add msgHeader to url ;
end
if (URL FORM MESSAGE is in url ) then
if PARSED ADDRESS = source address of dsRec then
data port of dsRec := PARSED PORT;
data client address of dsRec := source address of Msg;
control server address of dsRec := destination address of Msg;
control server port of dsRec := destination port of Msg;
control client address of dsRec := source address of Msg;
control client port of dsRec := source port of Msg;
end ; {ParseMMS}
```

Analysis control packet

Session Analysis - MMS (2/2) - Methods



Example payload of setup request

MMS START MESSAGE
Hexa decimal form (01 00 00 00 CE FA 0B B0)
MMS SETUP MESSAGE
Hexa decimal form (60 00)
URL CHARACTER
Regular expression of "[\ \ \ .0-9a-zA-Z]"
URL FORM MESSAGE
Regular expression of "\ \ \ \ [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].*"

Message format

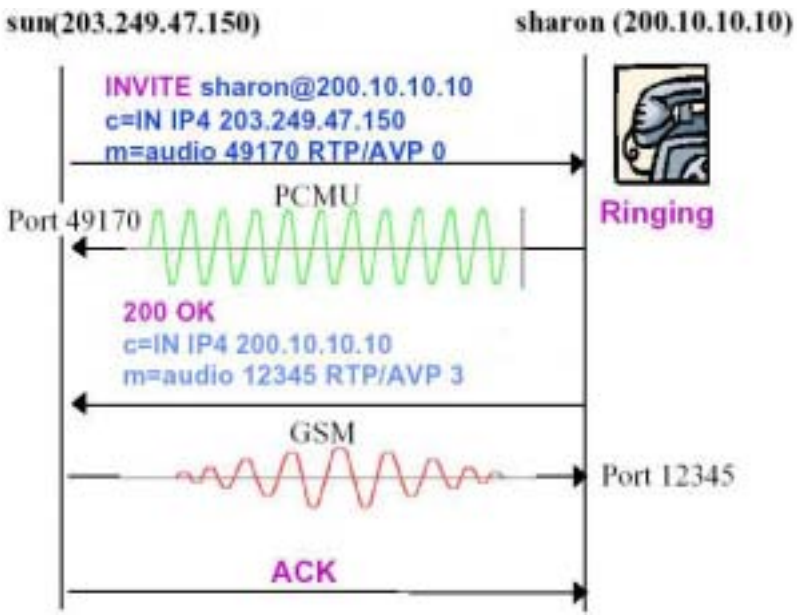
- Find port from URL FORM

```
procedure ParseMMS ( var Msg : CONTROL SESSION MESSAGE,
                     var Table : SET of DYNAMIC SESSION RECORD )
var
msgHeader : array[1..256] of character ;
url : array[1..256] of character ;
begin
if destination port of Msg = MMS server port number then      {request message}
msgHeader := read 8 bytes from Msg;
if msgHeader = MMS START MESSAGE then
   msgHeader := read next 2 bytes from Msg;
if msgHeader = MMS SETUP MESSAGE then
while (there is a byte to read in Msg ) do begin
msgHeader := next byte from Msg;
if msgHeader := URL CHARACTER then
add msgHeader to url ;
end
if (URL FORM MESSAGE is in url ) then
if PARSED ADDRESS = source address of dsRec then
data port of dsRec := PARSED PORT;
data client address of dsRec := source address of Msg;
control server address of dsRec := destination address of Msg;
control server port of dsRec := destination port of Msg;
control client address of dsRec := source address of Msg;
control client port of dsRec := source port of Msg;
end ; {ParseMMS}
```

Analysis control packet

Session Analysis - SIP (1/2) - Methods

- Select invite or response message



Example of connection setup

```
procedure ParseSIP ( var Msg : CONTROL SESSION MESSAGE,
                     var Table : SET of DYNAMIC SESSION RECORD )
var
msgHeader : array[1..256] of character ;
begin
if destination port of Msg = SIP server port number then
msgHeader := read a line from Msg;
if (msgHeader = INVITE MESSAGE) or                    {request message}
(msgHeader = RESPONSE MESSAGE) then                   {response message}
while (there is a line to read in Msg ) do begin
msgHeader := read a next line from Msg;
if msgHeader = TRANSPORT MESSAGE then                 {transport parameter}
data client port of dsRec := PARSED PORT;
data port of dsRec := PARSED PORT;
data client address of dsRec := source address of Msg;
control server address of dsRec := destination address of Msg;
control server port of dsRec := destination port of Msg;
control client address of dsRec := source address of Msg;
control client port of dsRec := source port of Msg;
end
 end ; {ParseSIP}
```

Analysis control packet

Session Analysis - SIP (2/2) - Methods

- Find port from media transmission message

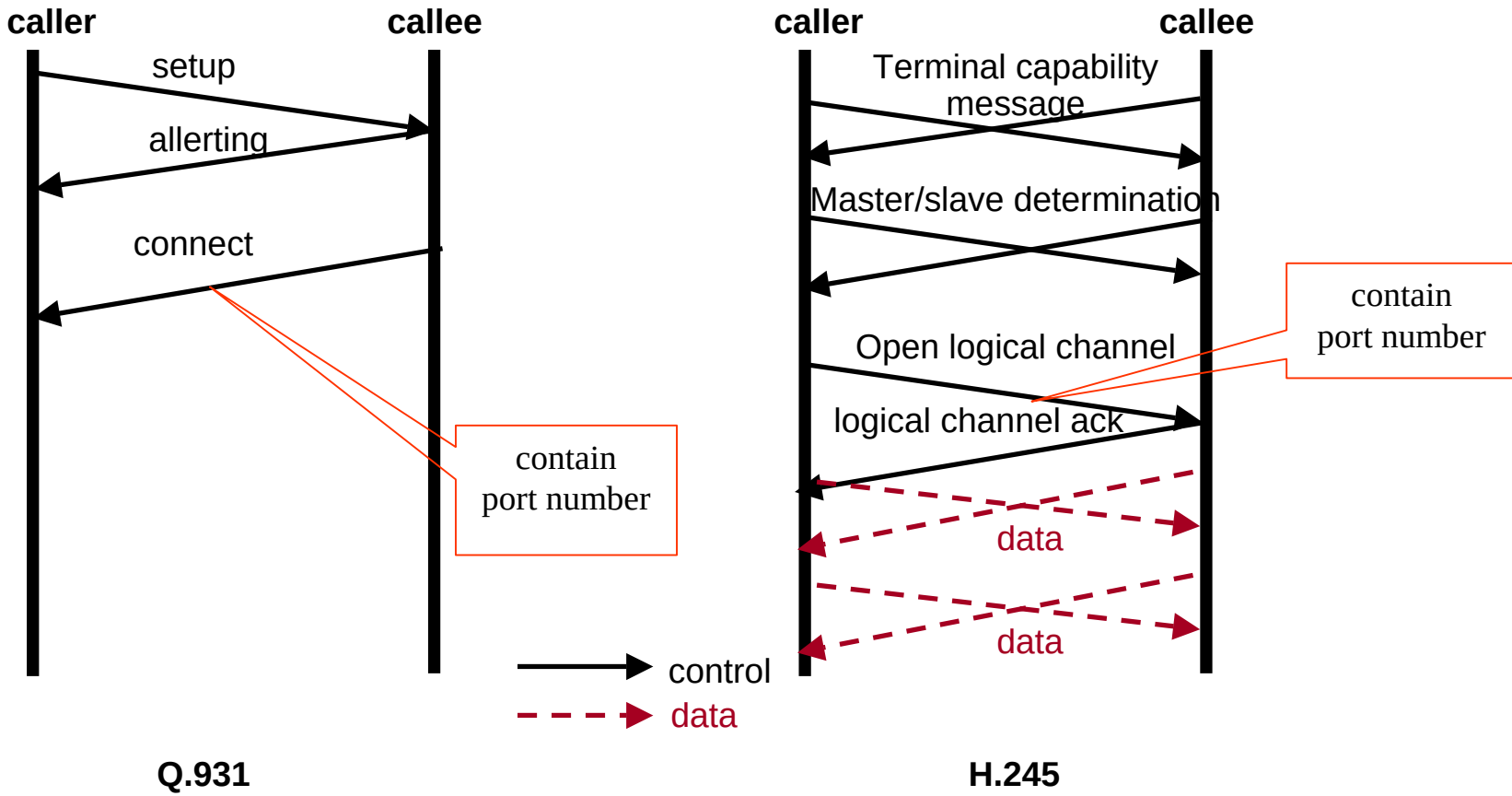
INVITE MESSAGE
[Ii][Nn][Vv][Ii][Tt][Ee] [0-9a-zA-Z]+@ [0-9a-zA-Z]+ [Ss][Ii][Pp].*
RESPONSE MESSAGERTSP
[Ss][Ii][Pp]/[0-9]+ \.[0-9]+ 200 OK.*
TRANSPORT MESSAGE
[Mm]= [a-zA-Z]+ [1-9][0-9]{3,4} [a-zA-Z]+([/-][a-zA-Z])*.*

Regular expression

```
procedure ParseSIP ( var Msg : CONTROL SESSION MESSAGE,
                     var Table : SET of DYNAMIC SESSION RECORD )
var
msgHeader : array[1..256] of character ;
begin
if destination port of Msg = SIP server port number then
msgHeader := read a line from Msg;
if (msgHeader = INVITE MESSAGE) or            //request message
(msgHeader = RESPONSE MESSAGE) then           //response message
while (there is a line to read in Msg ) do begin
msgHeader := read a next line from Msg;
if msgHeader = TRANSPORT MESSAGE then   //transport parameter
data client port of dsRec := PARSED PORT;
data port of dsRec := PARSED PORT;
data client address of dsRec := source address of Msg;
control server address of dsRec := destination address of Msg;
control server port of dsRec := destination port of Msg;
control client address of dsRec := source address of Msg;
control client port of dsRec := source port of Msg;
end
 end ; {ParseSIP}
```

Analysis control packet

Session Analysis - H.323 (1/2) - Methods



Connection setup

Session Analysis - H.323 (2/2) - Methods

procedure $ParseQ931$ (**var** Msg : CONTROL SESSION MESSAGE,
var $Table$: SET of DYNAMIC SESSION RECORD)
var
$msgHeader$: array[1..256] of character ;
begin
if protocol discriminator of Msg = Q.931 **then**
if message type = CONNECT **then**
while (there is a byte to read in Msg) **do begin**
$msgHeader$:= next element information from Msg;
if $msgHeader$ = user-user info **then**
if IP address of H.245 address = $source\ \ address$ of $dsRec$ **then**
$data\ port$ of $dsRec$:= port of H.245 address;
$transport\ protocol$ of $dsRec$:= UDP;
$data\ client\ address$ of $dsRec$:= $source\ address$ of Msg;
$control\ server\ address$ of $dsRec$:= $source\ address$ of Msg;
$control\ server\ port$ of $dsRec$:= $source\ port$ of Msg;
$control\ client\ address$ of $dsRec$:= $destination\ address$ of Msg;
$control\ client\ port$ of $dsRec$:= $destination\ port$ of Msg;
end
end ; $\{ParseQ931\}$

procedure $ParseH245$ (**var** Msg : CONTROL SESSION MESSAGE,
var $Table$: SET of DYNAMIC SESSION RECORD
var $Q931Rec$: DYNAMIC SESSION RECORD)
var
$msgHeader$: array[1..256] of character ;
begin
if. (method of Msg = request of H.245 open logical channel or
method of Msg = response of H.245 open logical channel) **then**
while (there is a byte to read in Msg) **do begin**
$msgHeader$:= next logical parameter from Msg;
if ($msgHeader$ = forwardLogicalChannelParameters or
$msgHeader$ = reverseLogicalChannelParameters or
$msgHeader$ = networkAcessParameters) **then**
$data\ port$ of $dsRec$:= port of H.245 address;
$transport\ protocol$ of $dsRec$:= UDP;
$data\ port$ of $dsRec$:= PARSED PORT;
$data\ client\ address$ of $dsRec$:= $source\ address$ of Msg;
$control\ server\ address$ of $dsRec$:= $control\ server\ address$ of $Q931Rec$;
$control\ server\ port$ of $dsRec$:= $control\ server\ port$ of $Q931Rec$;
$control\ client\ address$ of $dsRec$:= $control\ client\ address$ of $Q931Rec$;
$control\ client\ port$ of $dsRec$:= $control\ client\ port$ of $Q931Rec$;
end
end ; $\{ParseH245\}$

<table>
<tr><td>Protocol discriminator</td><td>…….</td><td>……</td></tr>
<tr><td>Message type</td><td>…….</td><td>……</td></tr>
<tr><td>Element information1</td><td>Element length</td><td>……</td></tr>
<tr><td rowspan="3">Element information (User-User Info)</td><td>Element length</td><td>……</td></tr>
<tr><td rowspan="2">Ip address</td><td>IP</td></tr>
<tr><td>port</td></tr>
</table>

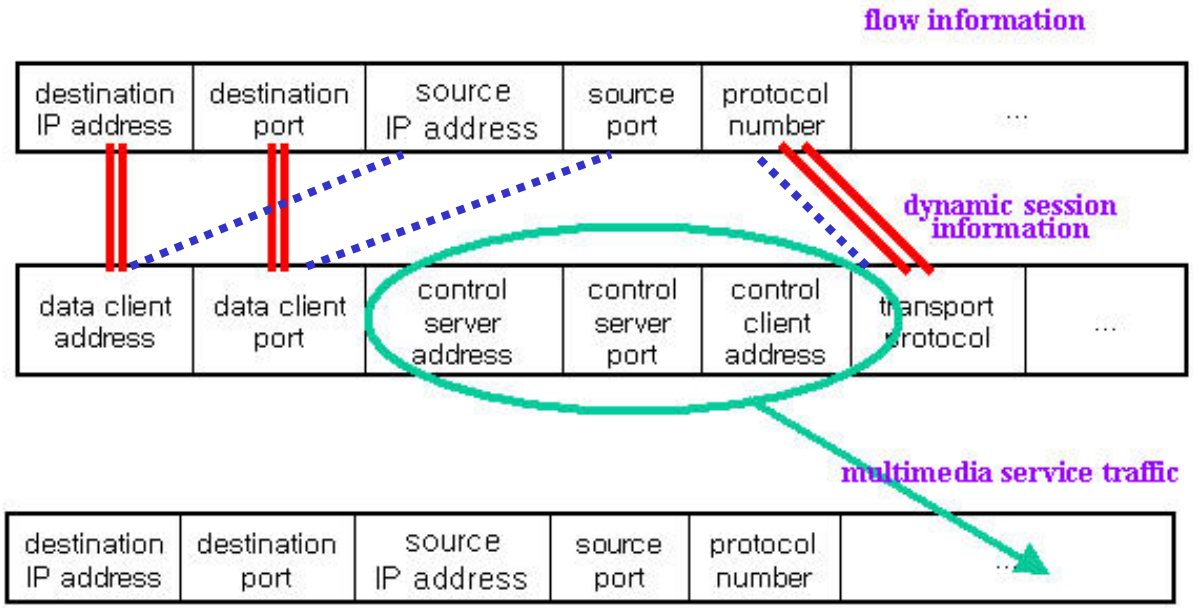
<table>
<tr><td>Method</td><td>…….</td><td>……</td></tr>
<tr><td>Logical parameter1</td><td>…….</td><td>……</td></tr>
<tr><td rowspan="3">forwardLogicalChannelParameters

reverseLogicalChannelParameters

networkAcessParameters</td><td>Data type</td><td>……</td></tr>
<tr><td>…..</td><td>……</td></tr>
<tr><td rowspan="2">Ip address</td><td>network</td></tr>
<tr><td>tsapIdentifier</td></tr>
</table>

Traffic Analysis (1/2) - Methods

- Determine and analyze multimedia service traffic flow
  - Match dynamic session information with flow information



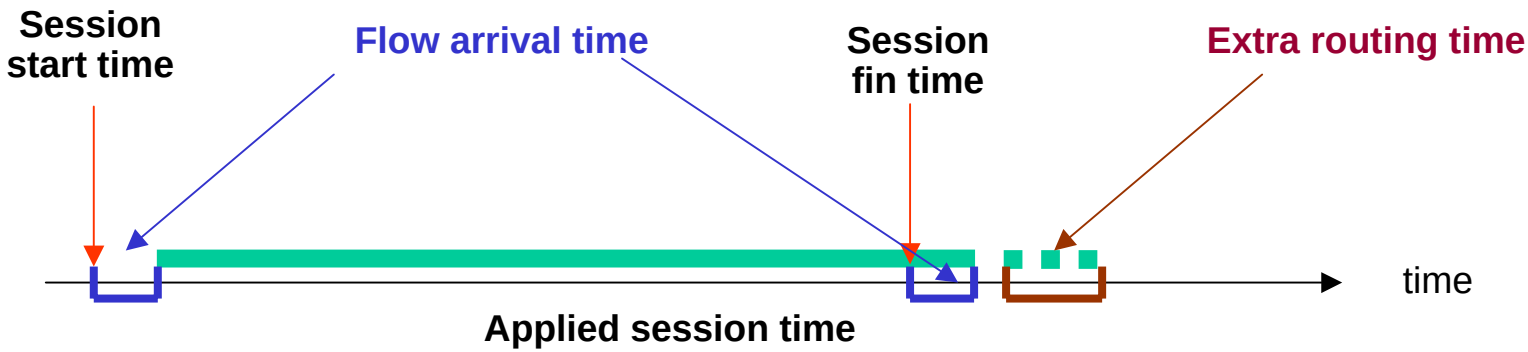
Field matching

Traffic Analysis (2/2) - Methods

- Manage dynamic session information
  - delete inactivated dynamic session
  - apply valid session start and fin time

dynamic session information

data client address	data client port	transport protocol	session start time	session fin time		...



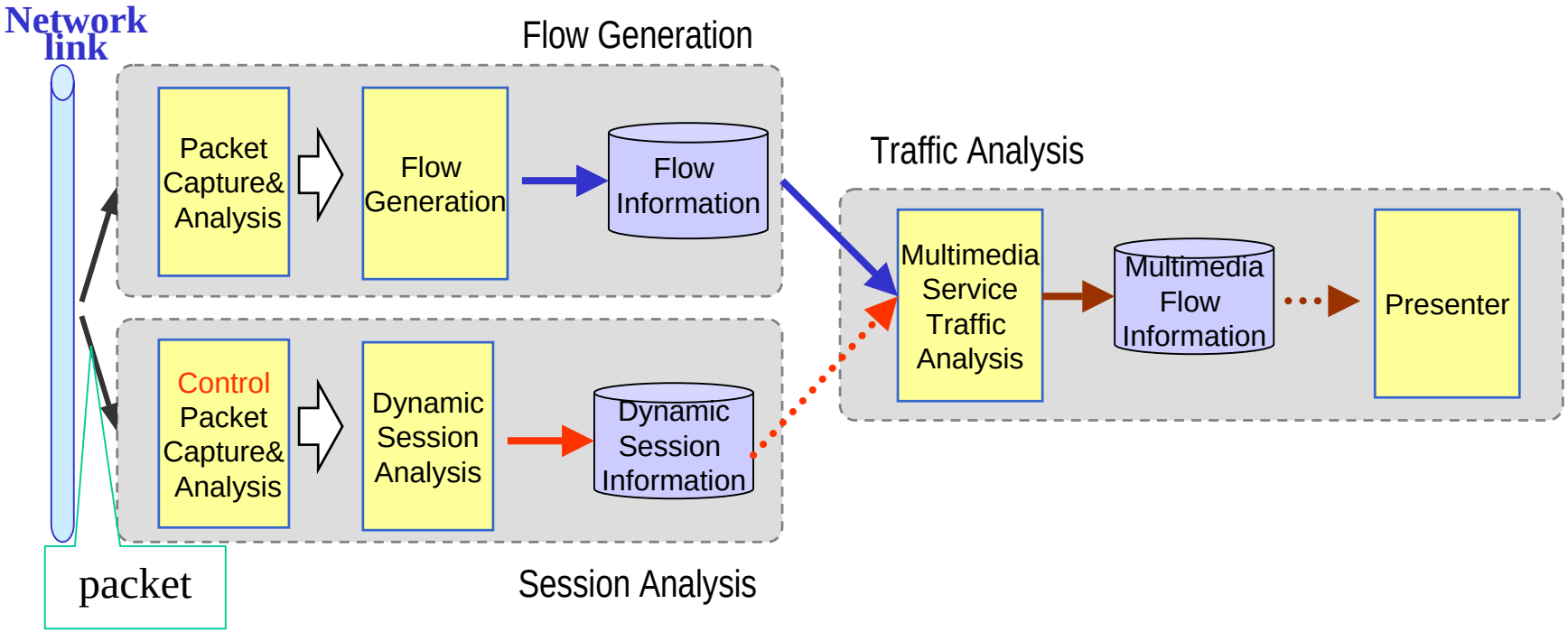
MST-MON Requirements - Design

- Multimedia Service Traffic Monitoring System
  - to apply proposed methods to NG-MON

- Requirements
  - Monitoring and analysis for multimedia service traffic
    - Object traffic
      - Streaming service traffic : Windows Media Technology, RealMedia, QuickTime
      - Multimedia conferencing service :applications based on H.323, SIP
    - Function
      - to get flow information
      - to extract dynamic information
      - to identify and analyze multimedia service flow
  - Cooperation with NG-MON
    - Real time online analysis
      - load distributed
      - reduce response time
    - Consideration of limited storage

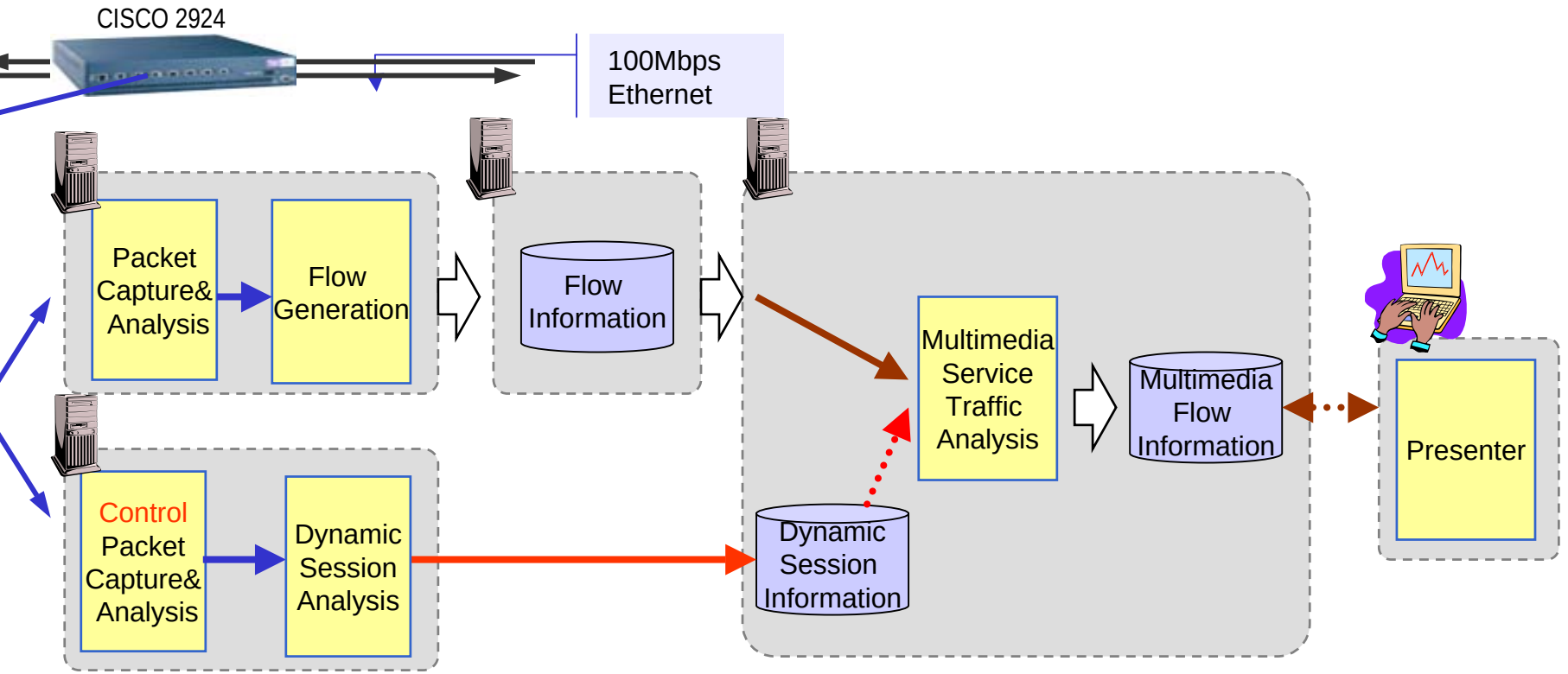
MST-MON Architecture - Design

- Design issues
  - to adopt NG-MON system architecture
  - to apply the proposed method without modifying existing MG-MON system
  - 3 components
    - Flow Generation, Session Analysis, Traffic Analysis



Implementation Environments - Implementation

- Environments
  - Monitoring multimedia service traffic in DPNM Lab.
  - 100Mbps Ethernet
  - Using mirroring ports in switch



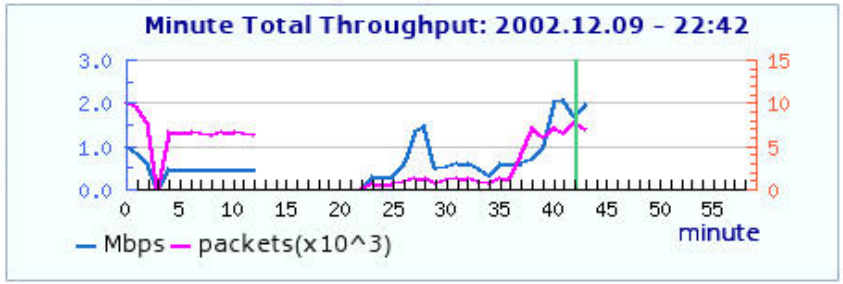
Application Minute View(1/2) - Implementation

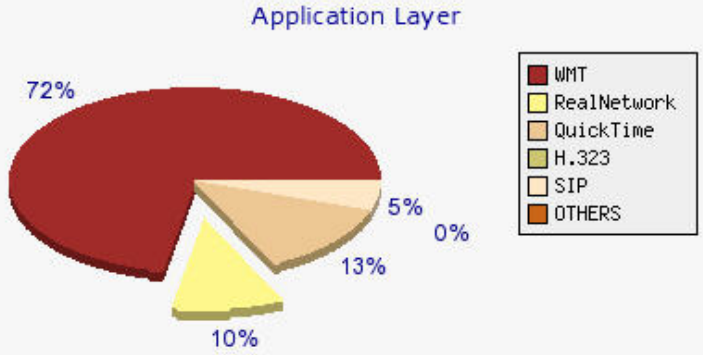
Application Protocol Minute View

Minute Hour Day Month

Time of Data	2002.12.09 - 22:42
Total Packets	7,790
Total Bytes	13,225,280
Avg. Bandwidth	1,763,370.67

<<prev> 22 h 42 m Show





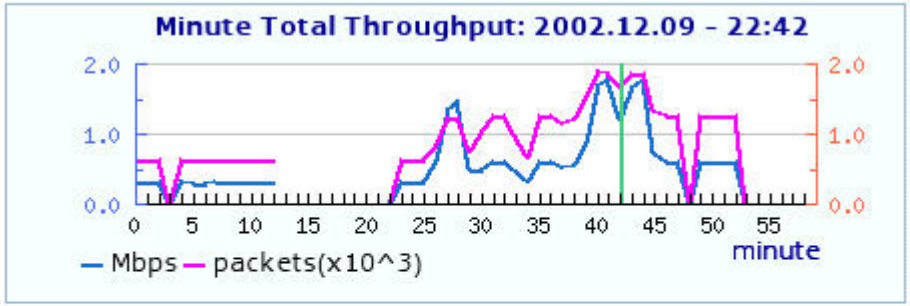
Applicaton

Index	Application	Packets	Bytes	Percentage(bytes/total)	detail
1	WMT	1,666	9,534,087	72.09 %	
2	QuickTime	1,666	1,705,312	12.89 %	
3	RealNetWork	1,348	1,316,339	9.95 %	
4	SIP	3,050	664,900	5.03 %	
5	H.323	60	4,642	0.04 %	

Application Minute View(2/2) - Implementation

Detail View : Application Protocol Minute (2002.12.09 - 22:42)

Time of Data	2002.12.09 - 22:42
Total Packets	1,666
Total Bytes	9,534,087
Avg. Bandwidth	1,271,211.60



<<prev> 22 h 42 m Show <next>>

WMT

Index	Client IP	Sever IP	Packets	Bytes	Percentage (bytes/total)	detail
1	danube.postech.ac.kr	mail.ex.onkino.co.kr	422	4,798,866	50.33 %	
2	danube.postech.ac.kr	211.115.216.65	622	2,373,717	24.9 %	
3	kwai.postech.ac.kr	211.115.216.65	622	2,361,504	24.77 %	

WMT

Index	Source IP	Destination IP	Source Port	Destination Port	Trans Protocol	Session Info	Packets	Bytes	Percentage (bytes/total)
1	mail.ex.onkino.co.kr	danube.postech.ac.kr	3388	4044	UDP	data sesstion	411	4,796,364	99.95 %
2	danube.postech.ac.kr	mail.ex.onkino.co.kr	4040	1755	TCP	control session	6	2,074	0.04 %
3	mail.ex.onkino.co.kr	danube.postech.ac.kr	1755	4040	TCP	control session	5	428	0.01 %

Results Comparison - Implementation

Applicaton					
Index	Source Port	Destination Port	Packets	Bytes	Percentage(bytes/total)
1	20	0	7,494	10,666,460	43.67 %
2	3388	4044	411	4,796,364	19.64 %
3	4403	4009	619	2,373,441	9.72 %
4	4377	4751	619	2,361,228	9.67 %
5	6970	6974	1,234	1,552,516	6.36 %

WMT									
Index	Source IP	Destination IP	Source Port	Destination Port	Trans Protocol	Session Info	Packets	Bytes	Percentage (bytes/total)
1	mail.ex.onkino.co.kr	danube.postech.ac.kr	3388	4044	UDP	data sesstion	411	4,796,364	99.95 %
2	danube.postech.ac.kr	mail.ex.onkino.co.kr	4040	1755	TCP	control session	6	2,074	0.04 %
3	mail.ex.onkino.co.kr	danube.postech.ac.kr	1755	4040	TCP	control session	5	428	0.01 %

12	80	0	103	19,525	0.08 %
13	55579	6000	64	15,664	0.06 %
14	1029	4379	9	13,062	0.05 %
15	55580	6000	26	12,552	0.05 %
16	10000	5004	50	10,900	0.04 %
17	138	138	41	10,287	0.04 %
18	6977	6971	117	8,974	0.04 %
19	6975	6971	115	8,962	0.04 %
20	6000	55579	48	4,204	0.02 %

28	4040	1755	6	2,074	0.01 %
43	1755	4040	5	428	0 %

Conclusion and Future Work

- Proposed methods and system design for monitoring and analysis of multimedia service traffic
  - Research multimedia service traffic characteristics and related protocols
  - Extract dynamically assigned port numbers and protocol information
  - Identify and analyze multimedia service traffic
  - Implement system adopting proposed methods
  - Raise analysis from transport level to application level

- Future Work
  - Complete accuracy of parsing of MMS protocol
  - Apply MST-MON Enterprise network
  - Find multimedia service traffic distribution, characteristics, usage pattern
  - Research and extend the proposed analysis methods to traffic using dynamic session
    - File transfer in instant messaging
    - Peer to Peer application, etc.