스마트 디바이스 응용 트래픽 분석

김 명 섭

2011년 11월 11일

고려대학교 과학기술대학 컴퓨터정보학과
tmskim@korea.ac.kr, http://nmlab.korea.ac.kr

Contents

Introduction

- Traffic classification is one of the hottest issue in modern networks.
- Accurate, Complete, and Real-time classification of Internet traffic is necessary for efficient …
  - Network Operations and Management
  - Capacity Planning
  - Provisioning
  - Service Differentiation
  - Cost Reduction
- So far, lots of methods have been introduced.
  - However, there is no single method that satisfies the three requirements
- This talk focuses on
  - KU-MON real-time traffic classification system we have developed
  - Smart-Phone Traffic Classification on a campus network

Definition of Traffic Classification

Definition of Traffic Classification

➢ General process of traffic classification → per-flow process

- Capture packets from network link in passive mode.
  - Using the mirroring function of network devices
  - By tapping directly from a target network link.
- Generate flows
  - Aggregation of packets with same 5-tuple packet header information (src_ip, src_port, dst_ip, dst_port, portocol)
  - A flow could be uni-directional or bi-directional
  - A flow contains 5-tuple header info. and other additional data (packet count, byte size, timing info. statistical info. etc.)

Definition of Traffic Classification

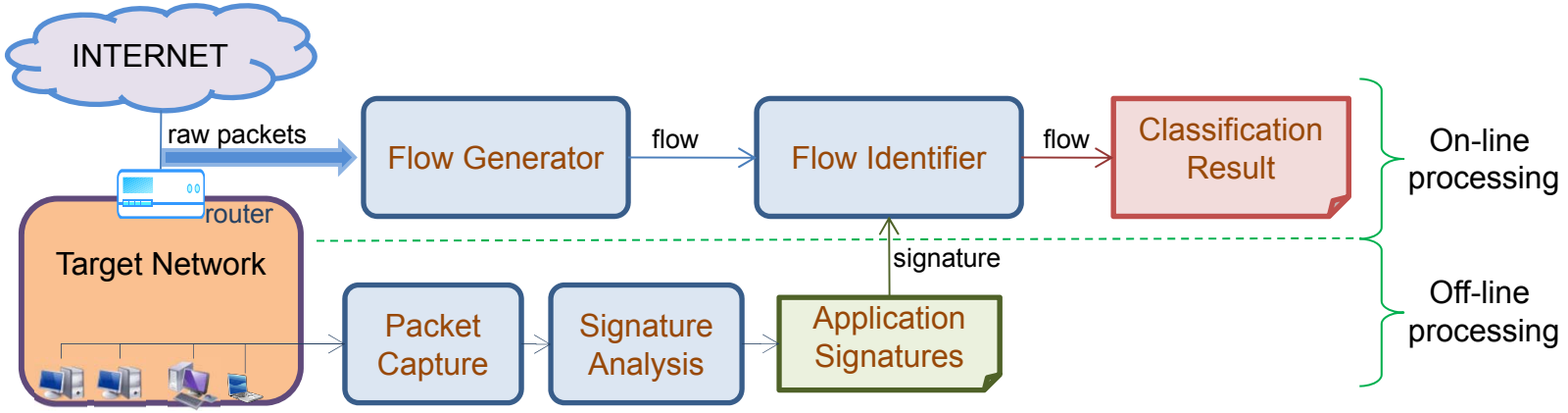
➢ General process of traffic classification → per-flow process

- Extract Features from a flow
  - Header Info: IP address, port number, TTL, etc
  - Payload Info: payload data of first n packets in a flow
  - Statistics Info: distribution of packet size, time, jitter, etc.
- Determine the identity of each flow
  - Protocol name or application name
  - Payload, Header, Statistics, Protocol Semantics, Correlation
- Classify flows into groups according to their identity.

Definition of Traffic Classification

➢ General process of traffic classification → per-flow process

➢ Typical system for traffic classification



Application traffic classification

- Traffic Classification in the perspective of network layers
  - Classification in network layer
    - IP, ARP, RARP, ……
  - Classification in transport layer
    - TCP, UDP, ICMP, ……
  - Classification in application layer
    - HTTP, SMTP, FTP, SSH, NTP, SIP, RTP, RTCP, ……
- Is that all?
  - Application-layer protocol based classification is not enough
  - Application based classification is more desirable.

Application traffic classification

- As a result
  - The application traffic classification is so complex that it is not easy to determine what is the identity of a traffic flow

- Our approach is …
  - 4 level classification for each flow
    - Application
    - Process
    - Protocol
    - Function

Flow	Application	Process	Protocol	Function
flow1	NateOn	nateonmain.exe	NateOn	Login
flow2	NateOn	nateonmain.exe	NateOn	Text chatting
flow3	NateOn	nateonmain.exe	NateOn	File Transfer
flow4	NateOn	nateonmain.exe	RTP/RTCP	Voice chatting
flow5	Fileguri	filegurimain.exe	Fileguri	Search
flow6	Fileguri	filegurimain.exe	Fileguri	File Transfer
flow6	Fileguri	fscagent.exe	HTTP	update

  - Network managers are interested in application and function level classification more than the protocol level classification

Methods in Traffic Classification

Methods in Traffic Classification

- Port-based
- Payload signature based
- Protocol semantics based
- Statistical analysis based
- Machine learning based
- Correlation based

Methods in Traffic Classification

- Comparison of several classification methods
  - Accuracy and completeness
  - Cost of signature generation and maintenance
  - Processing overhead
  - Dependence on location and time

Method	Accuracy	Cost of Signature (Rule) Extract	Cost of Operation	Signature (Rule) Maintenance	Locality Dependence
	Completeness				Time Dependence
Well-Known Port	Low	Low	Low	Easy	Independent
	Low				dependent
Payload Signature	High	High	High	Difficult	Independent
	High				dependent
Machine Leaning	High	High	High	Difficult	dependent
	Mid				dependent
Flow Correlation	High	High	High	Difficult	dependent
	Low				dependent
Application Behavior	High	High	High	Difficult	Independent
	Low				dependent

Evaluation of Traffic Classification

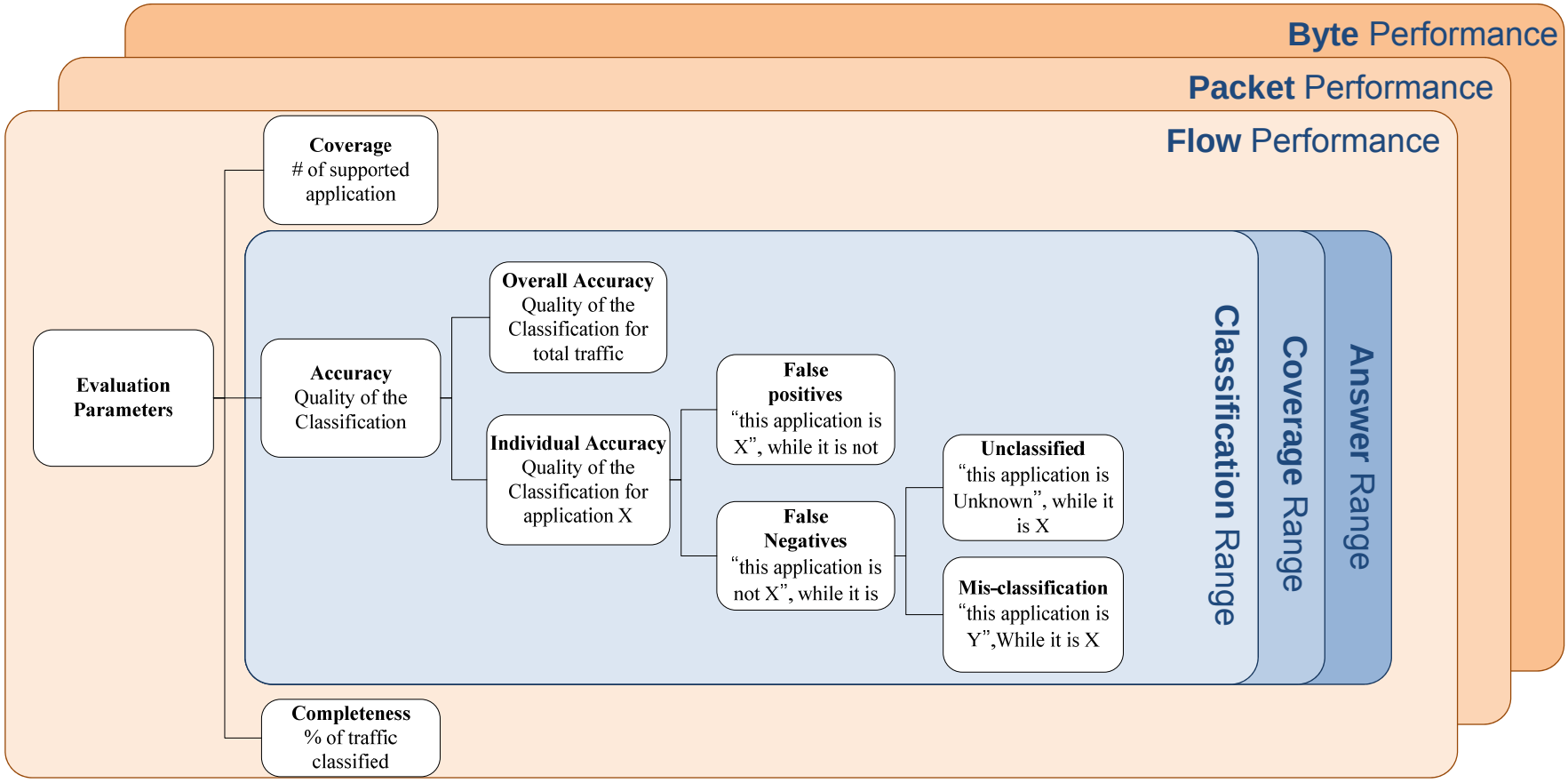
Performance Criteria

- Performance Criteria for Traffic Classification Methodology

- Coverage
  - Number of Applications supported by the given method.
- Completeness
  - Rate(%) of traffic classified by the given method
- Accuracy
  - Quality of the classification result by the given method.
- Scalability
  - Number traffic volumes processed at a time by the given method
- Robustness
  - Ability to deal with the effects of asymmetric routing, packet losses, and reordering by the given method.

Accuracy and Completeness

- Focus on the accuracy and completeness of a given method.
  - Accuracy and completeness highly depend on the converge

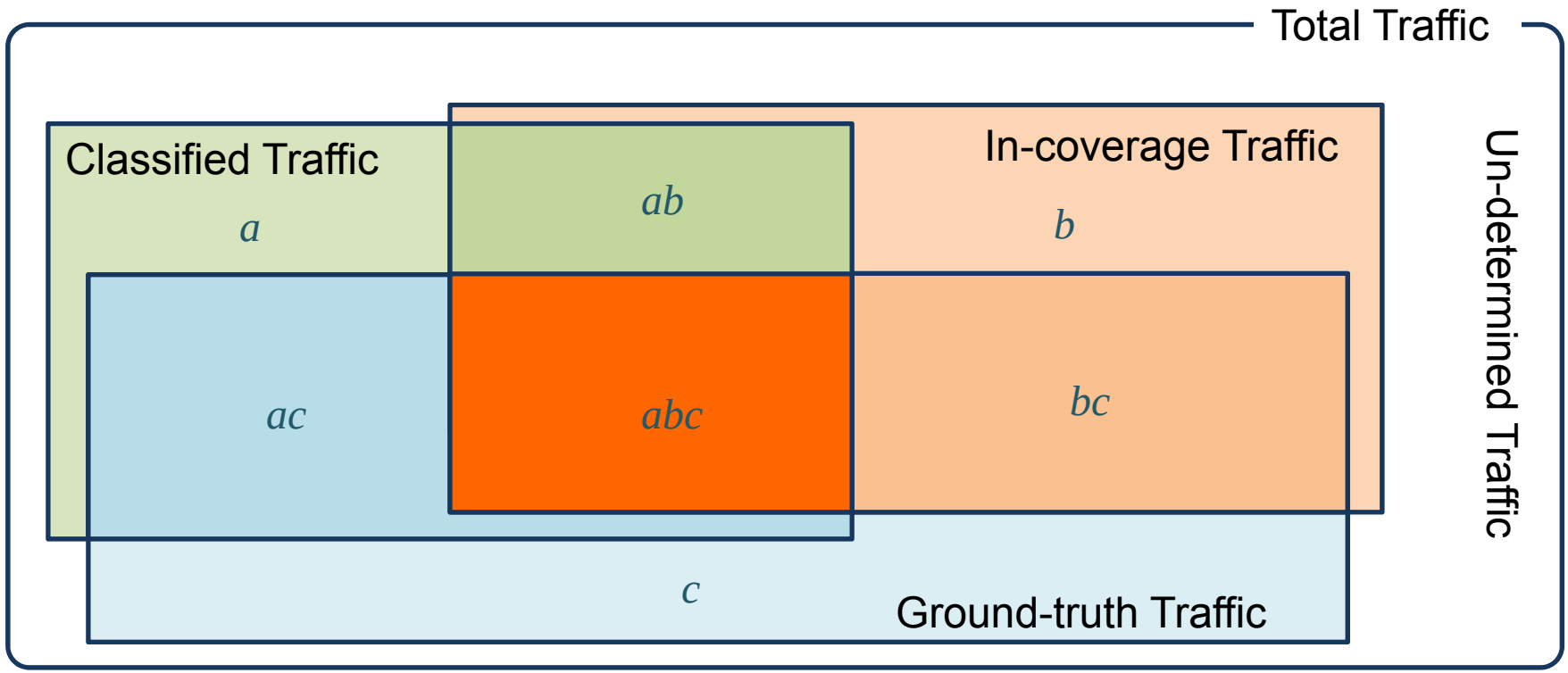


Ground-truth traffic data

- How to get the test traffic data?
- The scope of test traffic data
  - Whole traffic from a target link or network ?
  - Subset of traffic ?
- The ground-truth answer
  - How to get the ground-truth answer to the test traffic?
    - Manual collection
    - Port-based collection
    - Signature-based collection
    - Agent-based collection

Test traffic combination

➢ When a classification method is applied to a traffic data



Set(a) is traffic classified by a given method
Set(b) is all traffic belong to the application in coverage
Set(c) is all traffic whose answer is found.

➢ Need to determine the range of traffic data for evaluation.

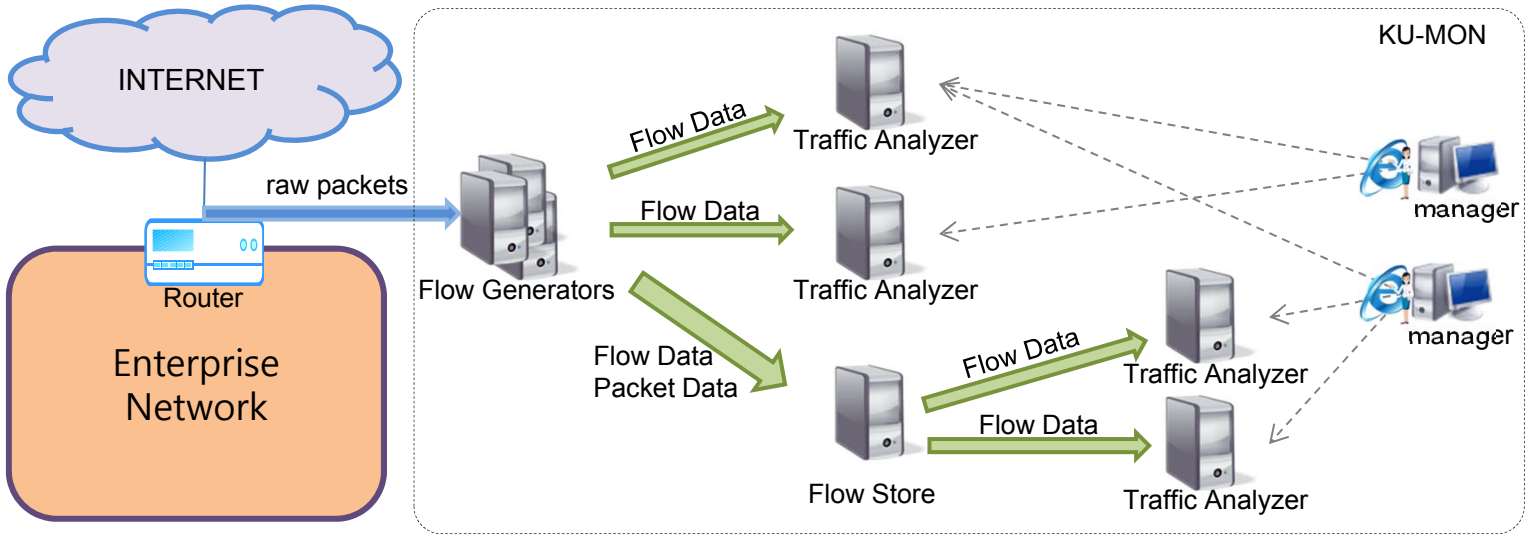
Traffic Classification in KU-MON

What is KU-MON?

- KU-MON(**K**orea **U**niversity - **MON**itoring)
  - A real-time traffic monitoring and analysis system for enterprise network as well as backbone network
  - Developed from NM Lab. Dept. of CISE, Korea Univ.
  - Deployed at Korea Univ. Sejong campus network.

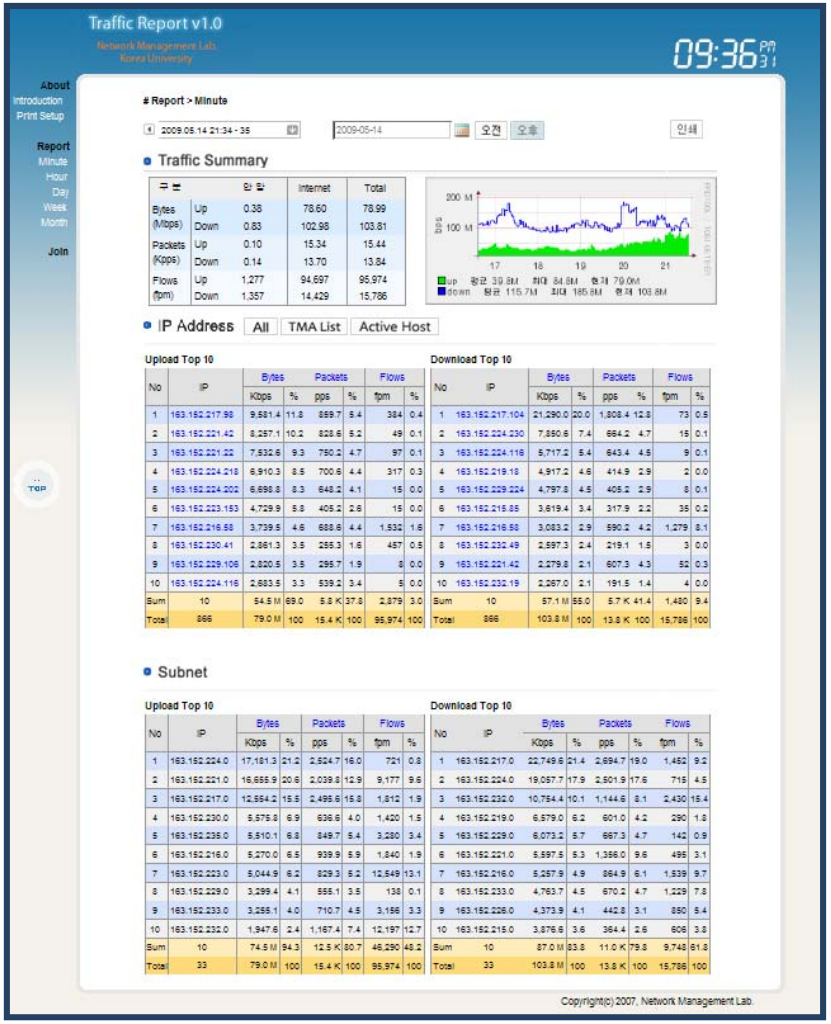
- KU-MON system at Korea Univ.
  - Captures all of the packets at the internet link of our campus.
  - Generates flow and packet data
  - Distributes the flow and packet data to multiple analysis servers
  - Analyzes traffic data to get useful information
    - Classification and evaluation system
  - Displays the analysis results on the web.
    - http://kumon.korea.ac.kr

KU-MON - system architecture

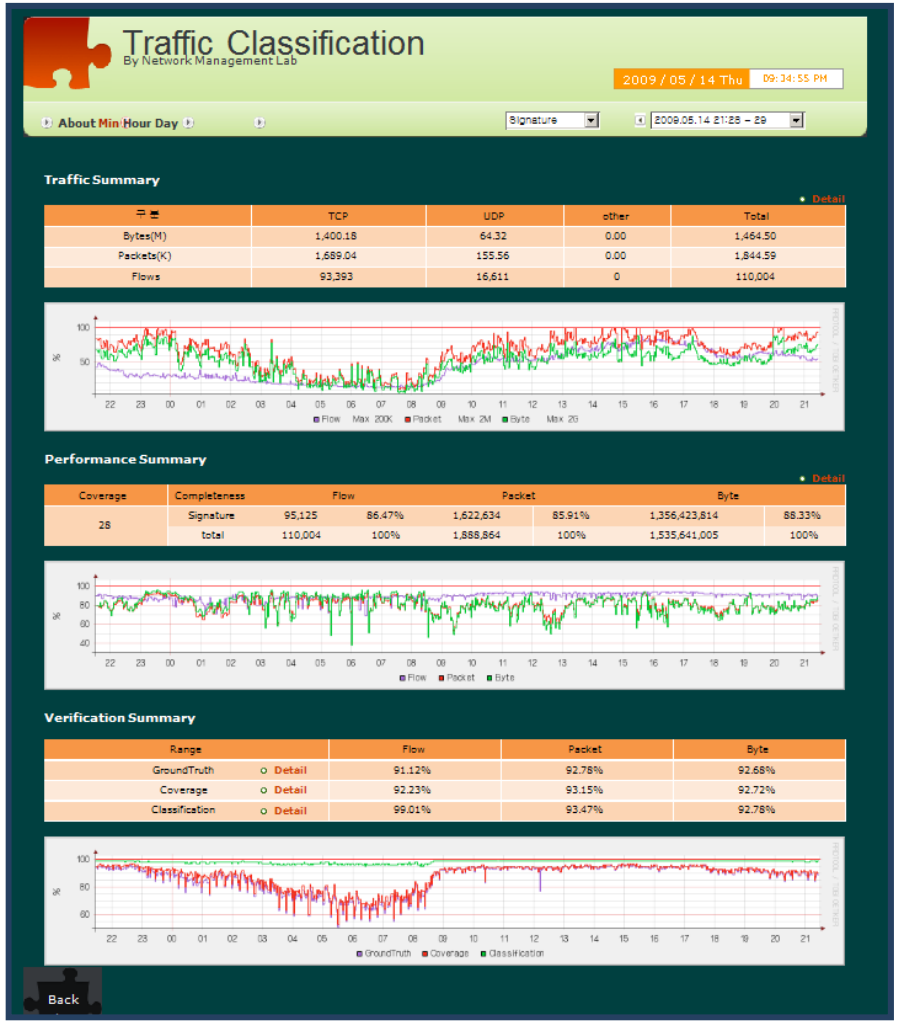


- Flow Generator generates flow and packet data with our own format
- Flow Analyzer performs various analysis tasks
- Flow Store stores flow and packet data for on/off-line in-detail analysis
  - 100 TBytes of disk raid for 1year traffic data
- Manager accesses the analysis results through web.

KU-MON – web-based user interfaces



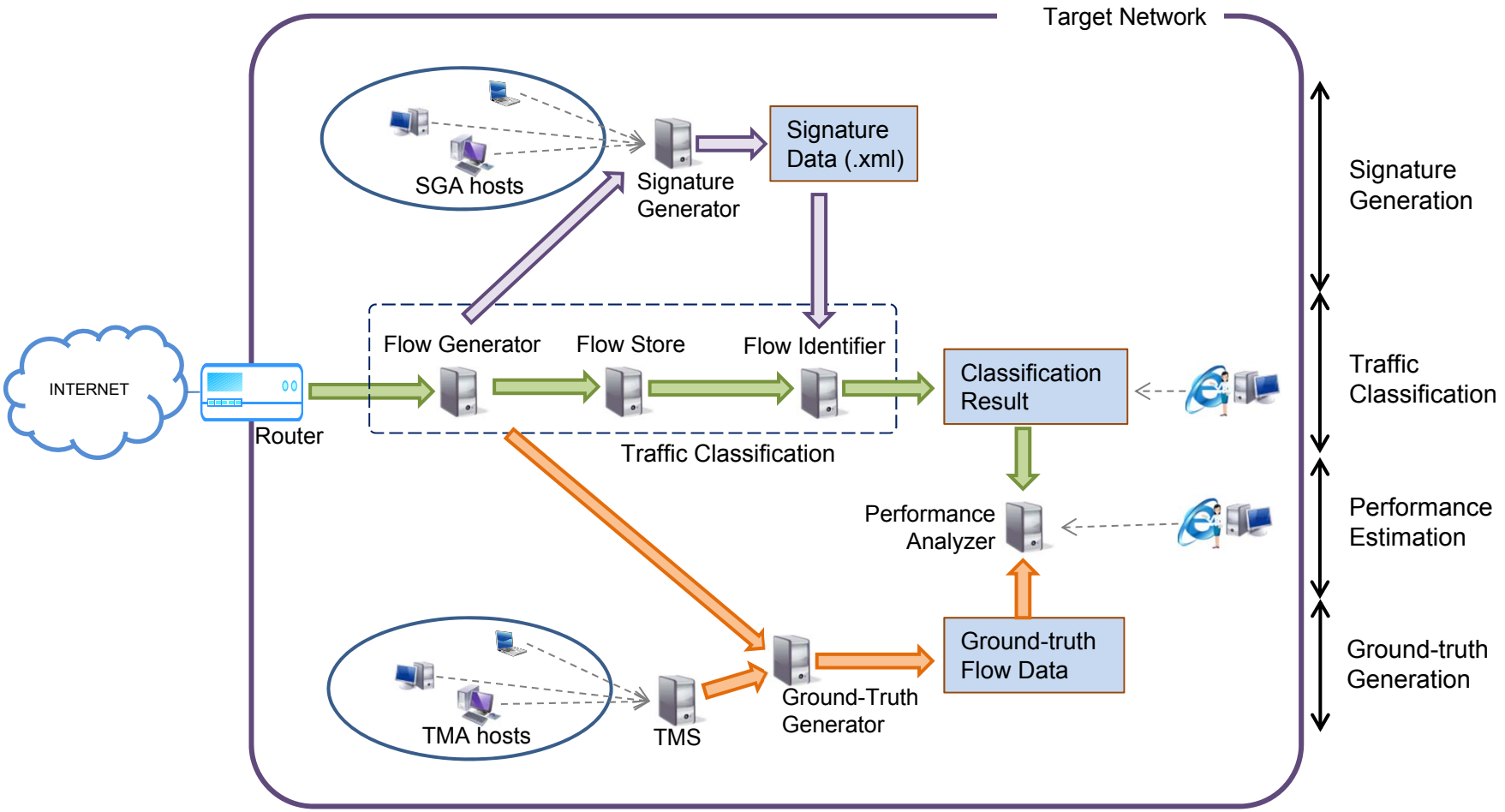
Web UI of Host Analysis System



Web UI of Traffic Classification System

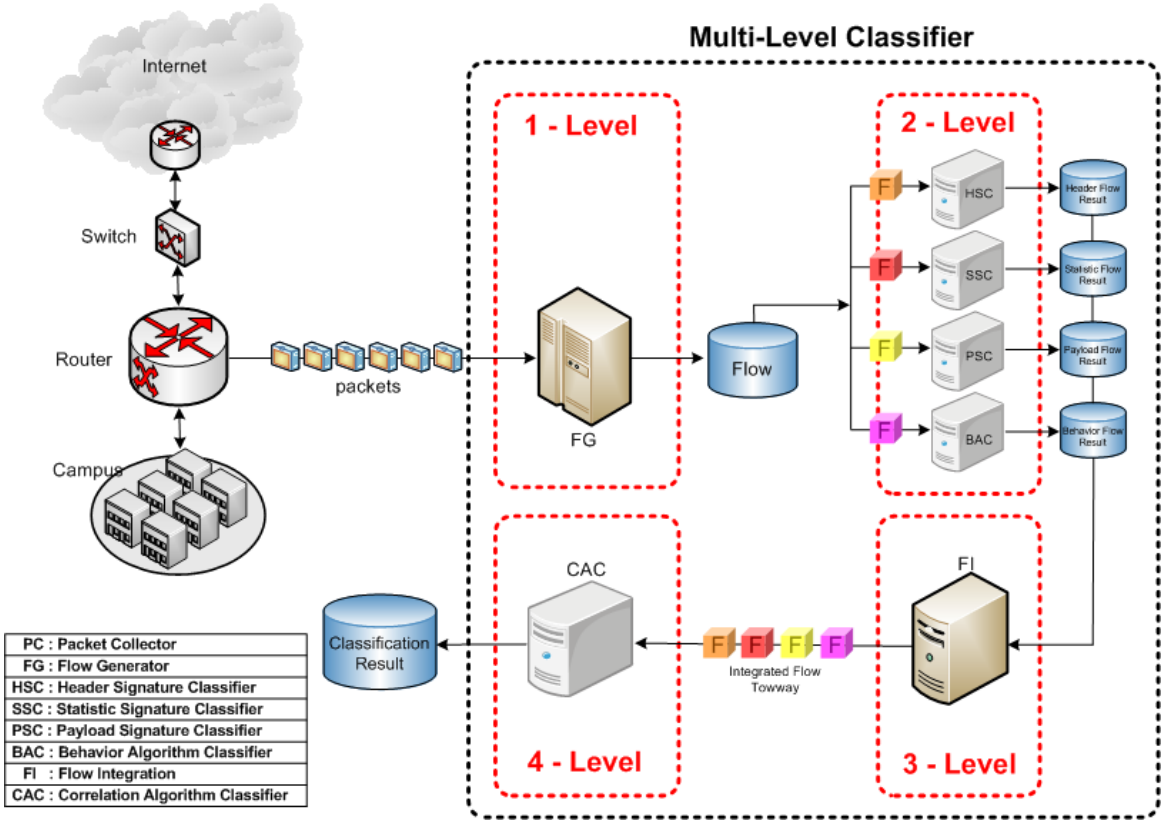
Traffic Classification System in KU-MON

➢ Overall structure for KU-MON traffic classification system



Multi-level Traffic Classification

- 1-Level
  - FG(Flow Generator)
- 2-Level
  - Header sig.-based
  - Statistic sig.-based
  - Payload sig.-based
  - Behavior based
- 3-Level
  - FI(Flow Integration)
- 4-Level
  - CAC (Correlation Algorithm Classifier)



Ground-truth Generation

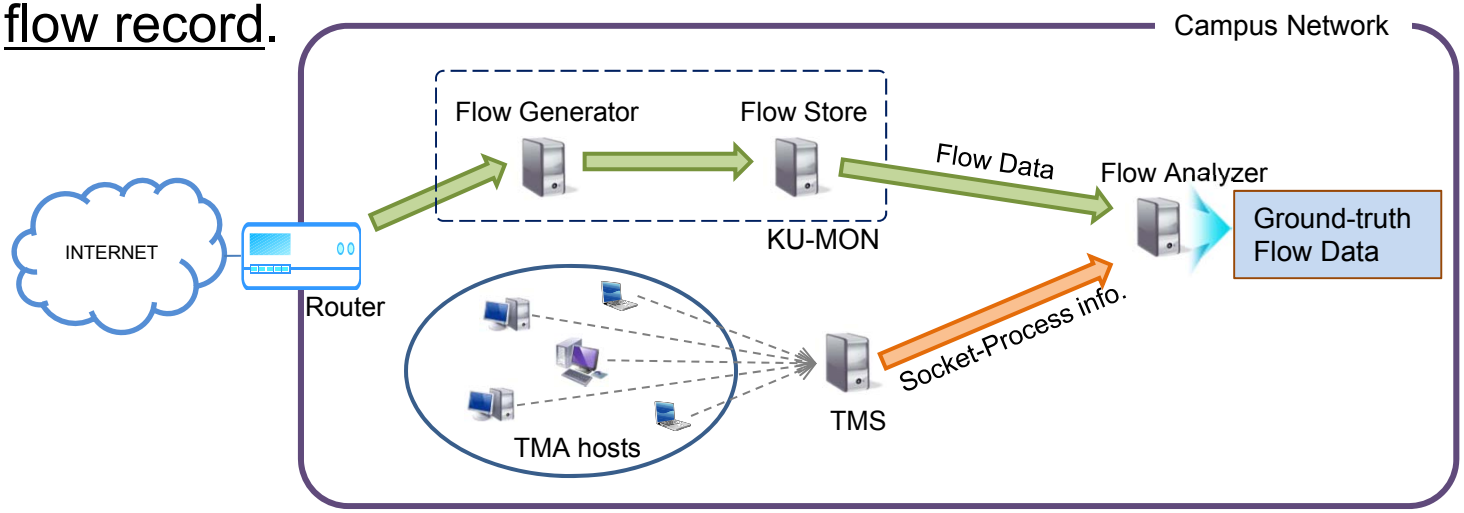
Ground-truth Generation

➢ TMA for ground-truth traffic data

- TMA is a tiny agent program installed at the end hosts.
- TMA gathers socket and process information and send them to the central server called TMS.
- TMA is installed at over 300 end hosts in our campus network.

➢ Assessment Network

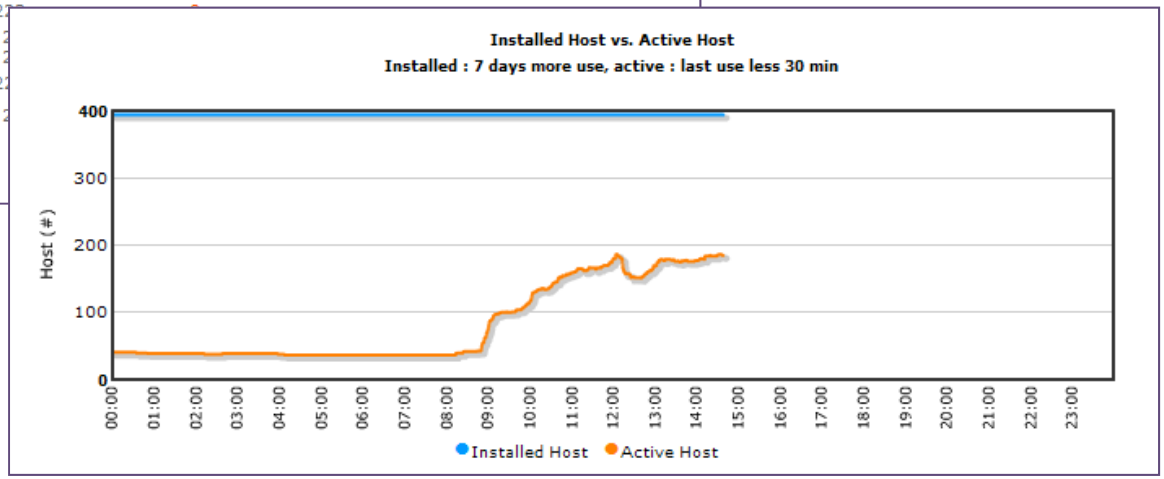
- The ground truth traffic data is created from the TMA record and flow record.



Ground-truth Generation

➢ TMA Monitoring System(TMAM)

Location	Code	IP Range	Usage	call	Etc.
과기대 크림슨	11	230.201 ~ 230.251	35	1396	every vacation
이미지실	12	216.165 ~ 216.229	32		
PC실	13	216.141 ~ 216.164	13		
NM Lab.	14	230.33 ~ 230.80	4		
Etc.	15	215.x~218.x 224.x 229.x~230.x	3		
농심관 전산1	21	211.21 ~ 211.88	68		
전산2	22	211.90 ~ 211.150	57		
209호	23	211.152 ~ 211.193	41		
Etc.	24	210.x , 211.x	1		
도서관 전자정보실	31	221.125 ~ 221.196 213.51 ~ 213.92	82		011-1780-7901
3층	32	221.101 ~ 221.122	20		
1층	33	213.161 ~ 213.180	15		
Etc.	34	213.x 221.x	1		
경상대 PC실	41	219.101 ~ 219.141	1		
Etc.	42	219.x 21			
인문대 PC실	51	214.101 ~ 2 222.181 ~ 2			
Etc.	52	214.x 21			
기타 교내	0	210.x ~ 2			
기타 교외	2				
Total	1				



Flow Format in KU-MON

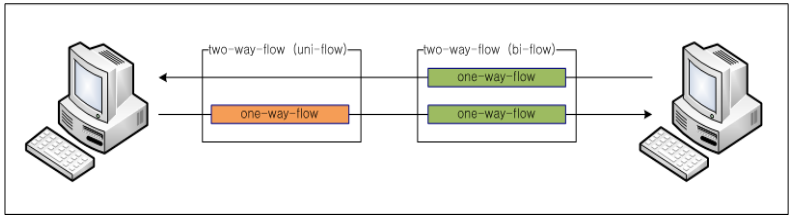
Flow format in KU-MON

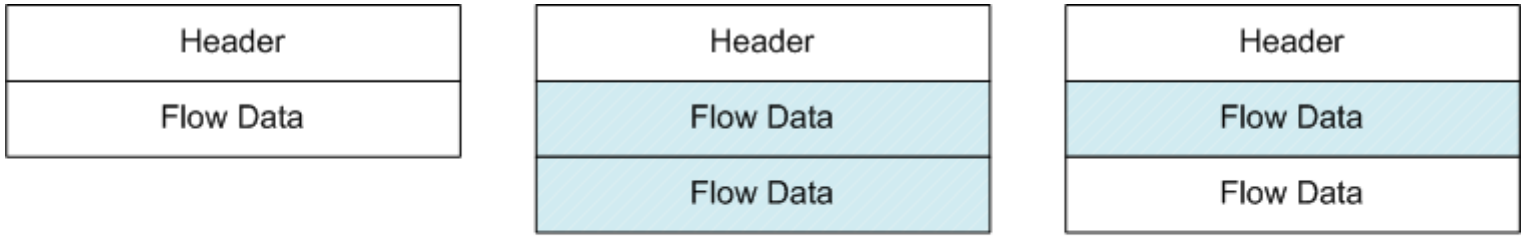
➢ One-way flow

- a uni-directional sequence of packets that share the 5-tuple (according to Cisco)
- 5-tuple (SrcIP, DstIP, SrcPort, DstProt, Protocol)

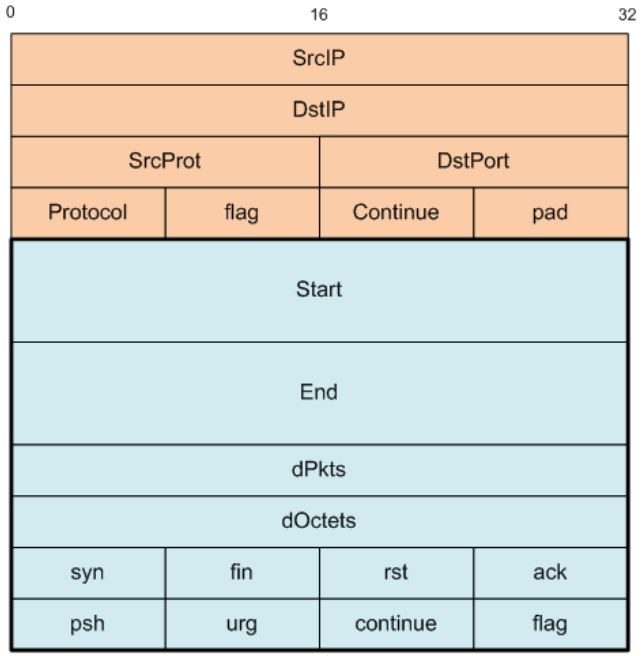
➢ Two-way flow

- one-way flow + reverse one-way flow
  - ◆ Bi-directional flow
    - – 쌍이 있는 Two-way flow format
  - ◆ Uni-directional flow
    - – 쌍이 없는 Two-way flow format

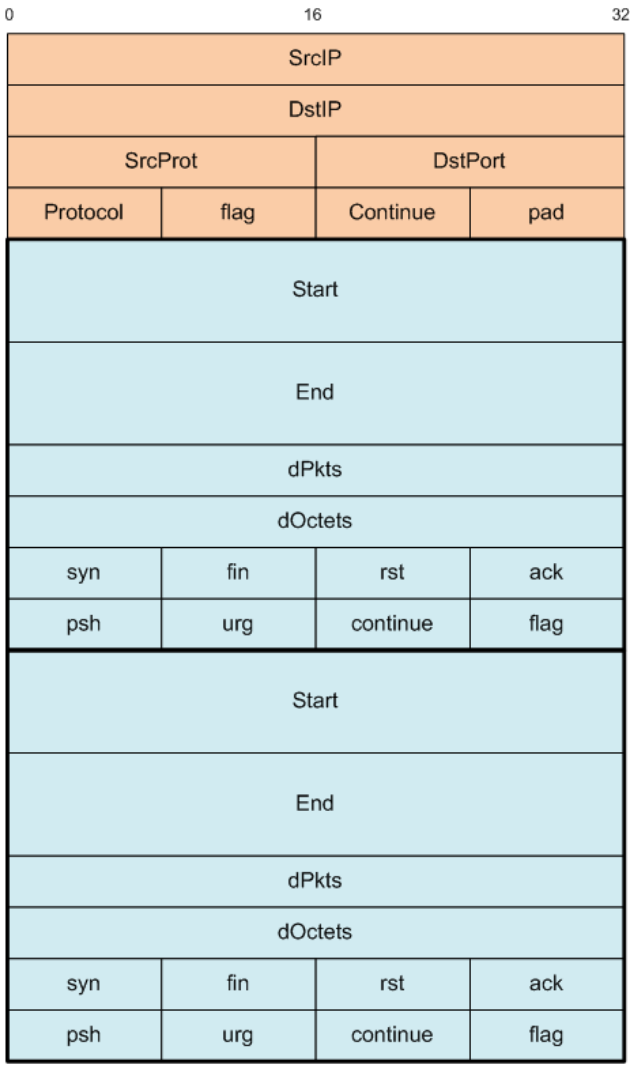




Flow format in KU-MON



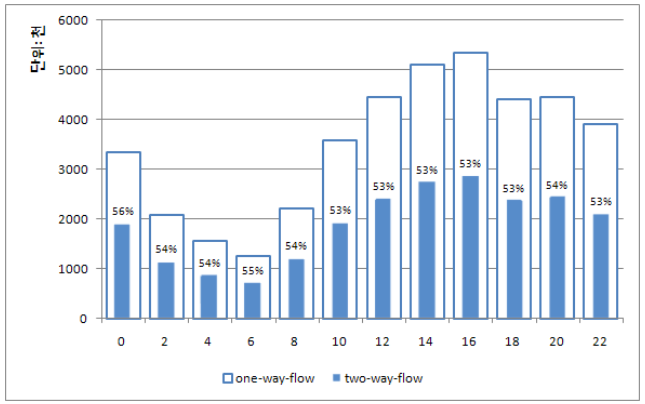
One-Way Flow Format



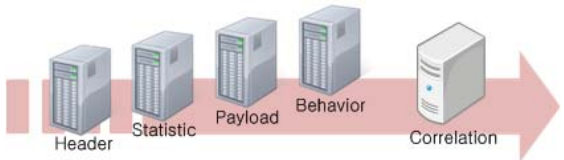
Two-Way Flow Format

Flow format in KU-MON

- One-way → Two-way
- 장점
  - Flow 개수 감소로 인한 분석 부하 감소
  - Reverse Flow 처리용이
  - 패킷 손실에 대한 유연한 대처
  - Session 단위의 분석 가능 → 분석 정확성 상승
    - 기존 One-way 기반 유지
    - Asymmetric-routing 단점 보완



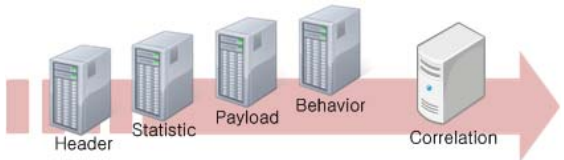
One-way-flow vs. Two-way-flow



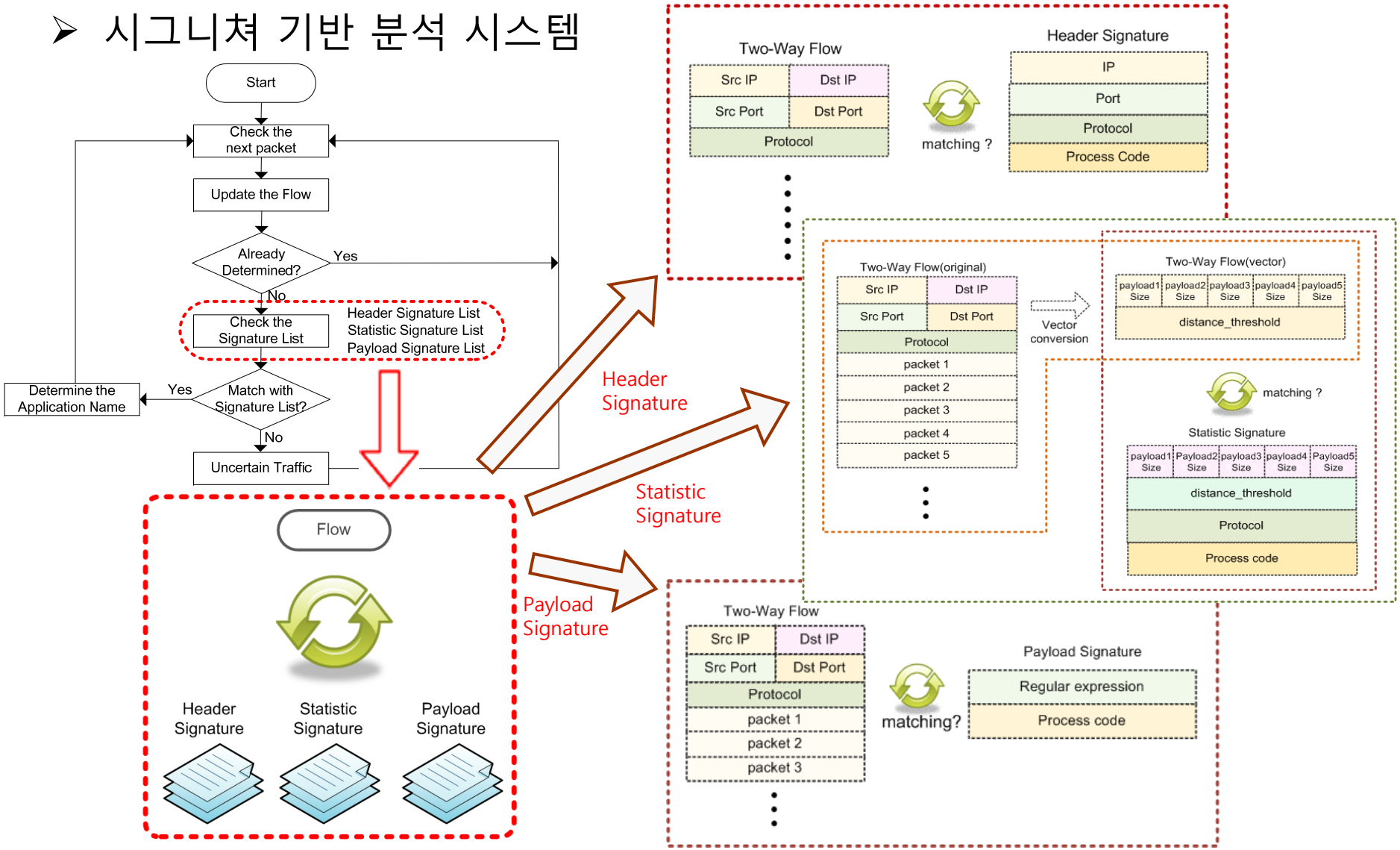
Multi-level Traffic Classification System

I. Signature-based 분석 시스템
   a. Header 시그니쳐 분석 시스템
   b. Statistic 시그니쳐 분석 시스템
   c. Payload 시그니쳐 분석 시스템
II. Behavior 분석 시스템
III. Flow Integration 시스템
IV. Correlation 기반 분석 시스템

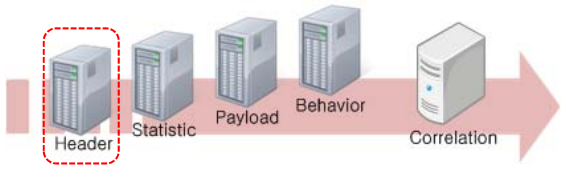
Level 2 – Signature based



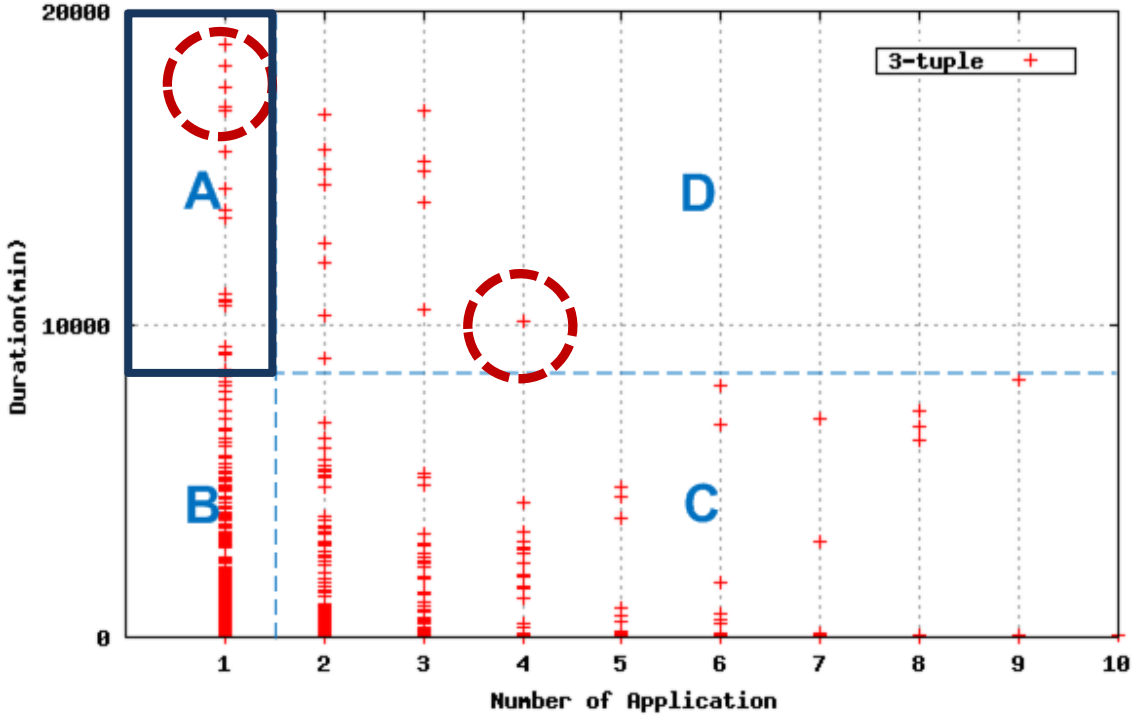
➢ 시그니쳐 기반 분석 시스템



Header Signature 기반



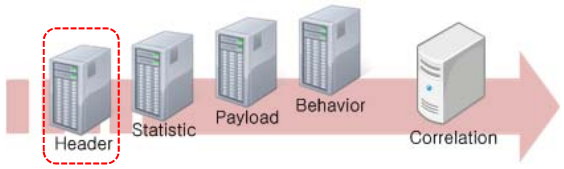
1. Header Signature based 트래픽 분석



13일 동안
TCP/UDP 포함
Server/Client 모두

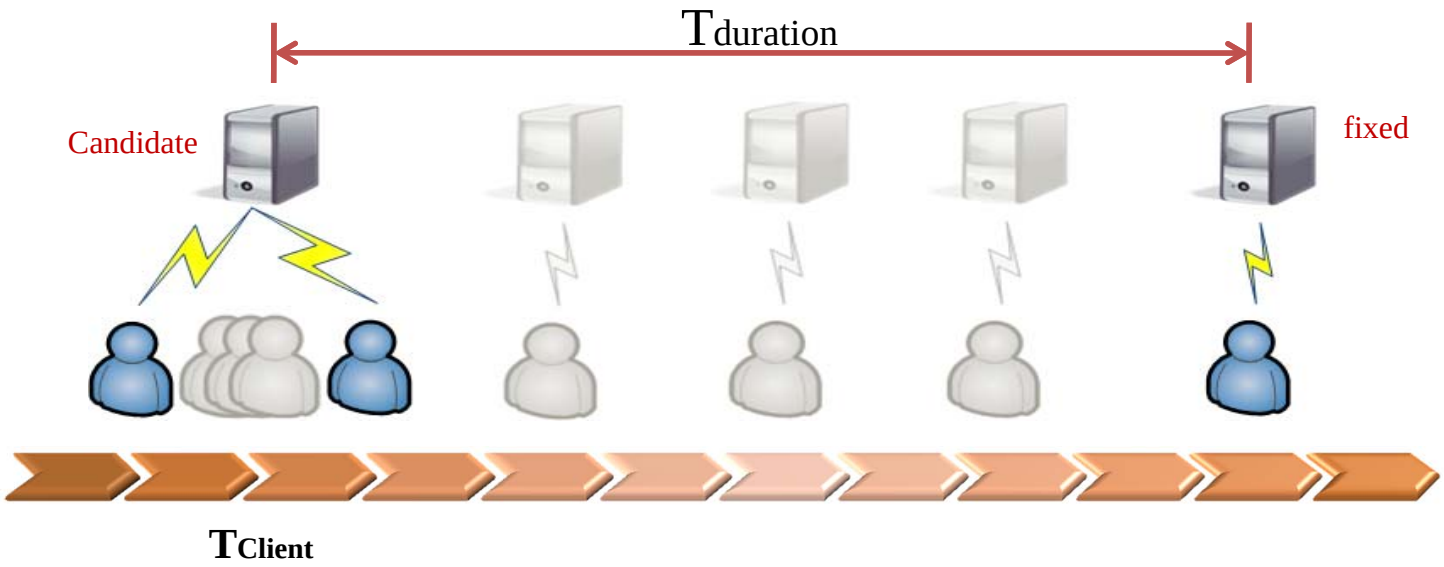
- IP-port 3-tuple의 구분 (IP address, port number, transport protocol)
  - 고정 IP-port (Group A)
  - 동적 IP-port (Group B)
  - 임시 IP-port (Group C)
  - 충돌 IP-port (Group D)

Header Signature 기반

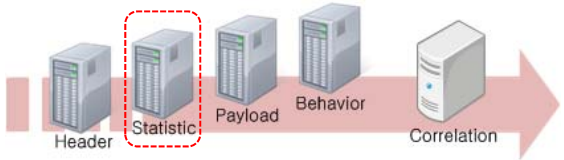


- 정의
  - 하나의 응용에 고정적으로 사용되는 **IP-port 3-tuple (IP address, port, protocol)**
- 장점
  - Transport Layer 헤더 정보만 이용
  - 낮은 오버헤드

- **추출방법**

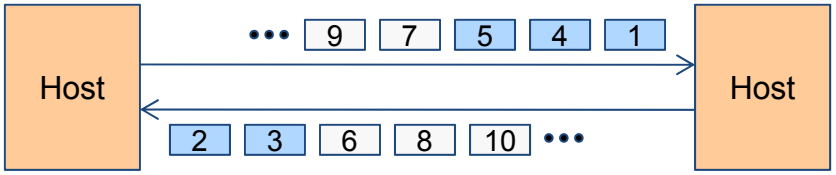


Statistic Signature 기반



- Statistic Signature의 정의
  - 패킷의 헤더나 캡쳐 정보에 기반한 응용 프로그램의 고유한 통계적 특징
    - packet size distribution(PSD), inter-arrival time, window size …

- Packet Size Distribution (PSD)
  - Packet Size : 패킷의 페이로드 크기
  - 첫 5개 이내의 페이로드가 존재하는 패킷(TCP 컨트롤 패킷 제외)
  - 양방향 Flow

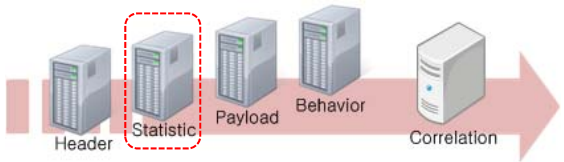


- PSD 시그니쳐 표현

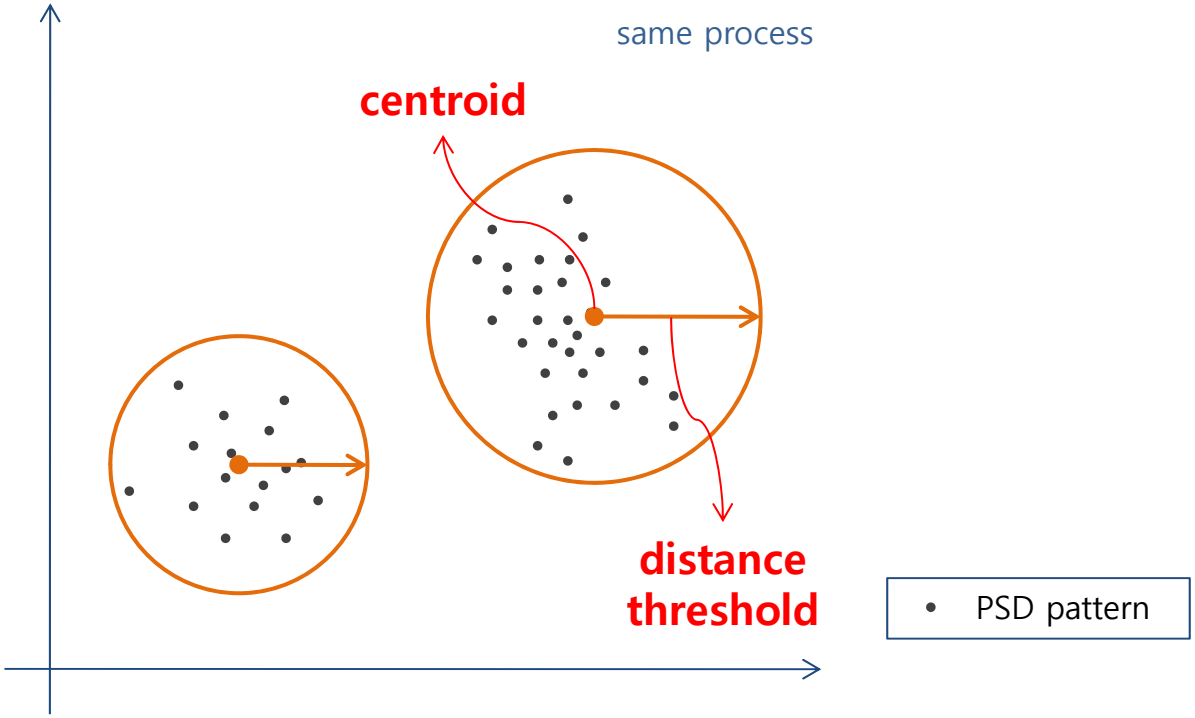
```
<application name="afreeca">
    <process name="afreecaplayer.exe">
        <signature code="0">
            <header trans_prot="tcp"/>
            <statistic p1="606" p2="-16" p3="20" p4="-1460" p5="-1460" distance_threshold="10"/>
        </signature>
```

(606, -16, 20, -1460, -1460)

Statistic Signature 기반

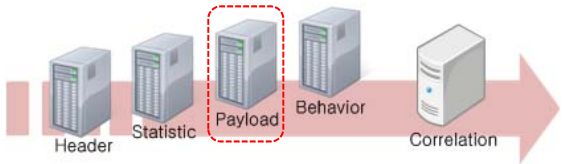


➢ PSD 시그니쳐 생성 방법

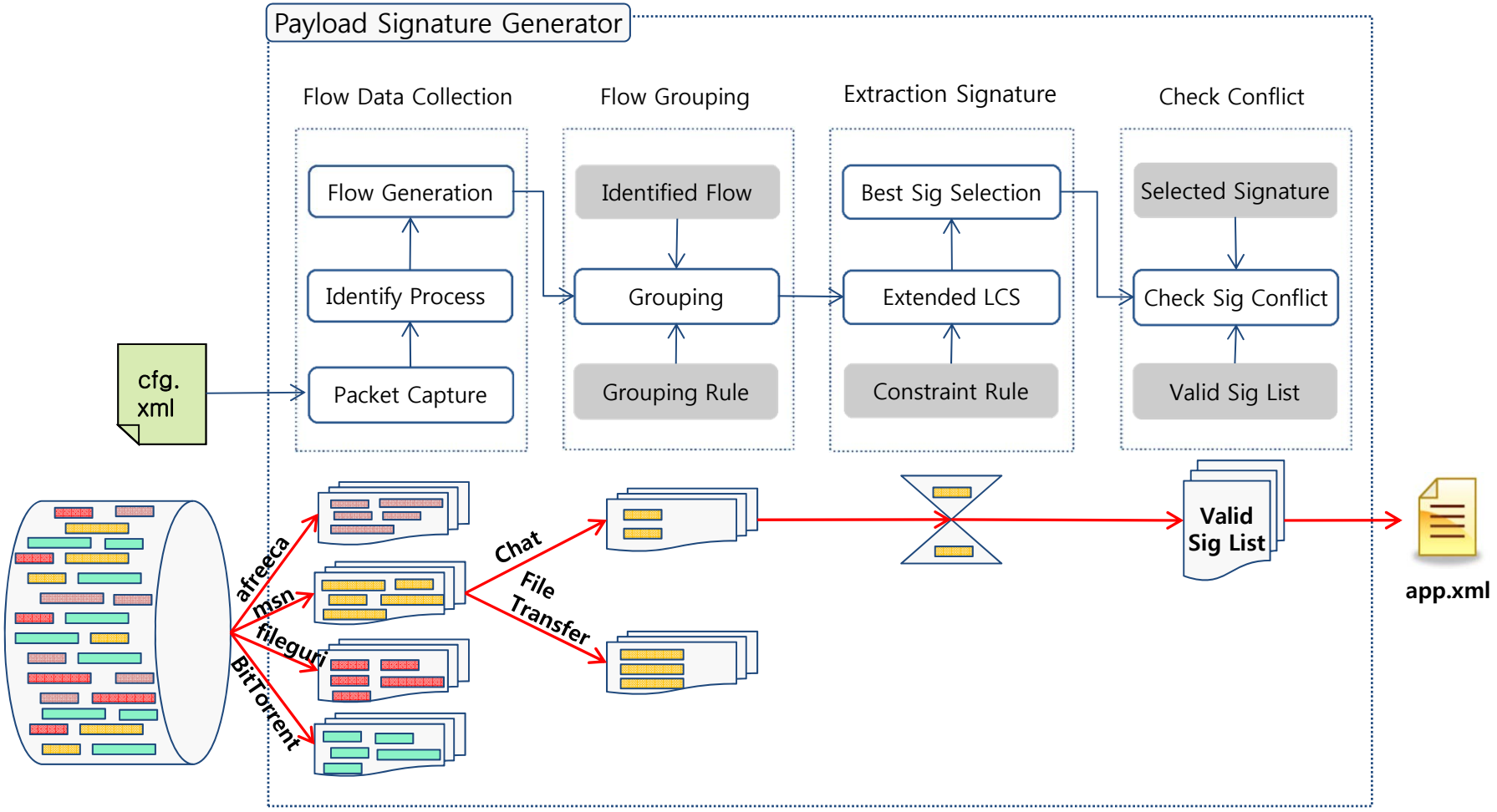


- PSD를 그룹핑하여 그룹의 중심값(centroid)과 거리값(distance)로 표현
- 시그니쳐 수의 감소

Payload Signature 기반

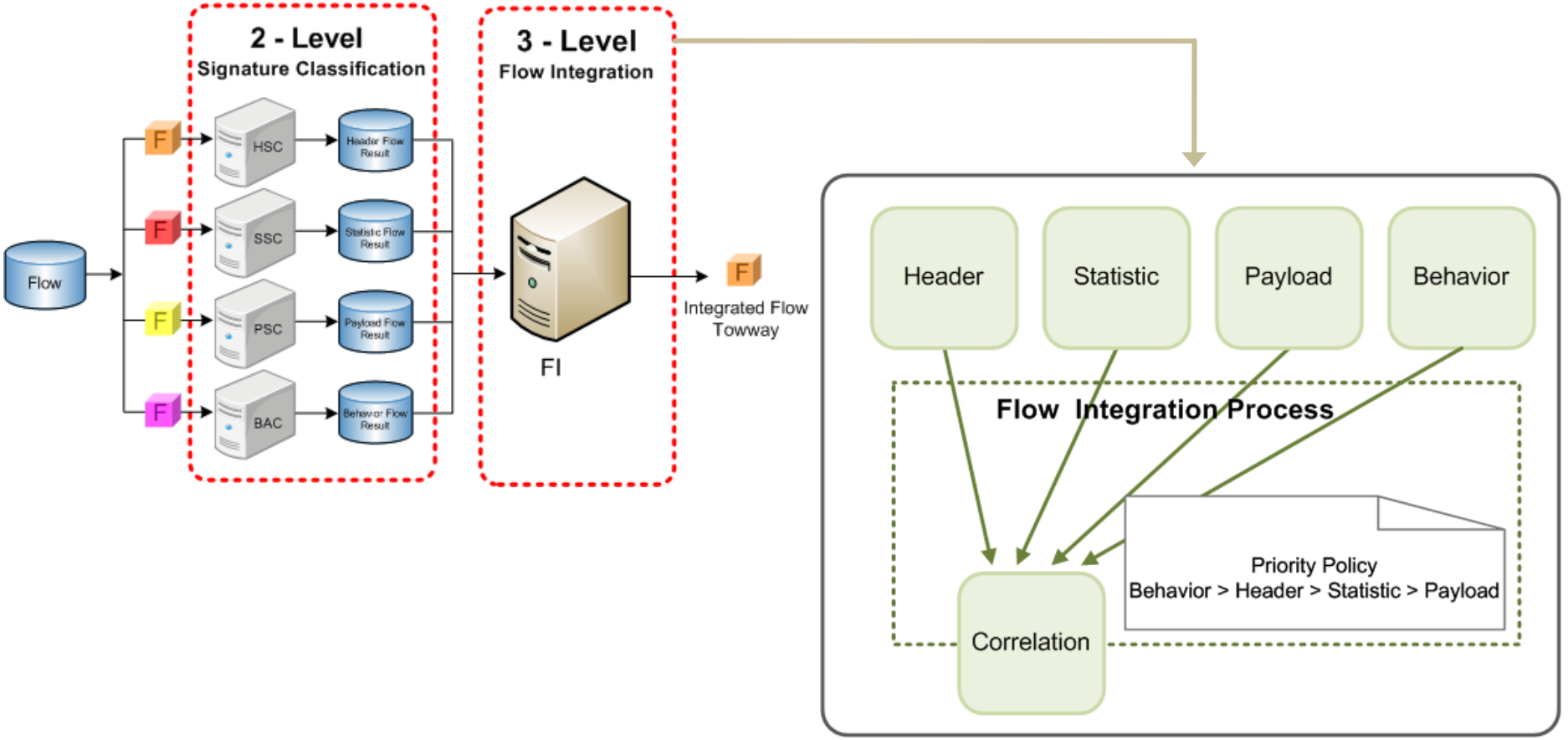


➢ 페이로드 시그니쳐 추출 시스템



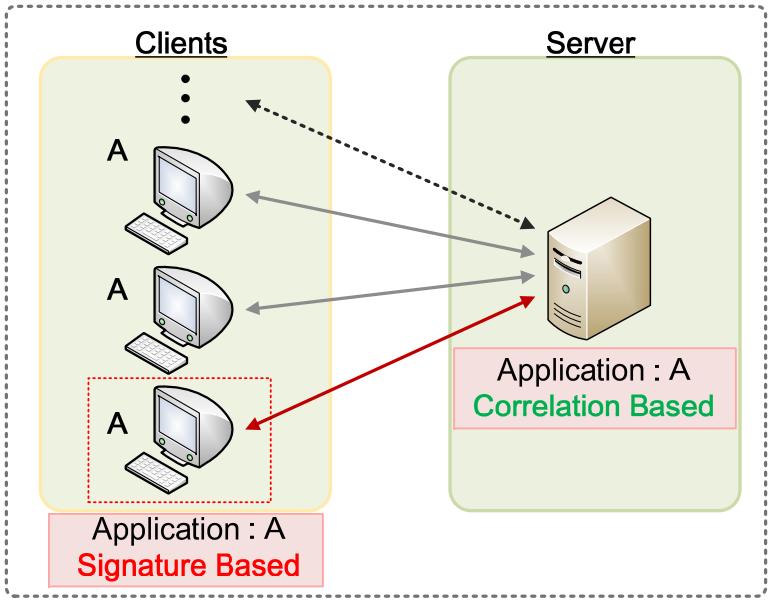
Flow Integration

- Flow Integration(FI)
  - 하나의 Flow에 대한 네 가지 시스템의 분석결과
    - 우선순위 정책(Priority Policy) 적용

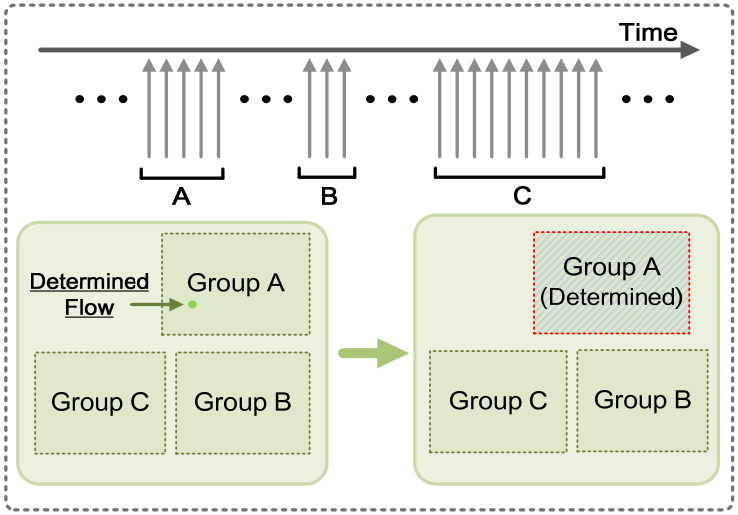


Flow Correlation

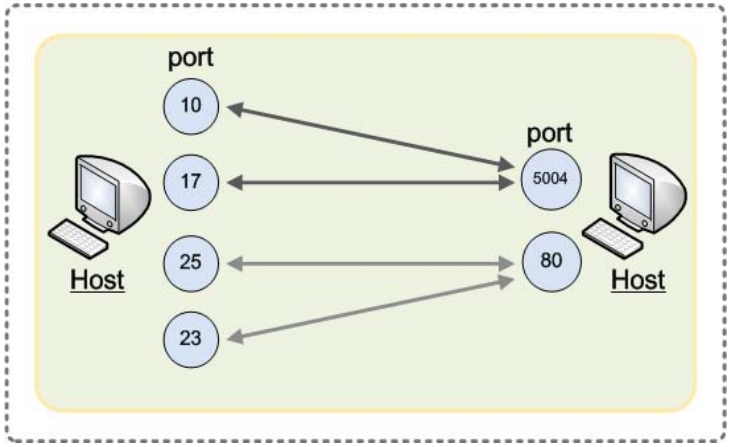
- Correlation 기반 분석 방법론
  - 서버-클라이언트 기반 상관관계 (Server-Client Correlation)
  - 발생시간 기반 상관관계 (Time-based Correlation)
  - 호스트-호스트 기반 상관관계 (Host-Host Correlation)



<Server-Client Correlation>



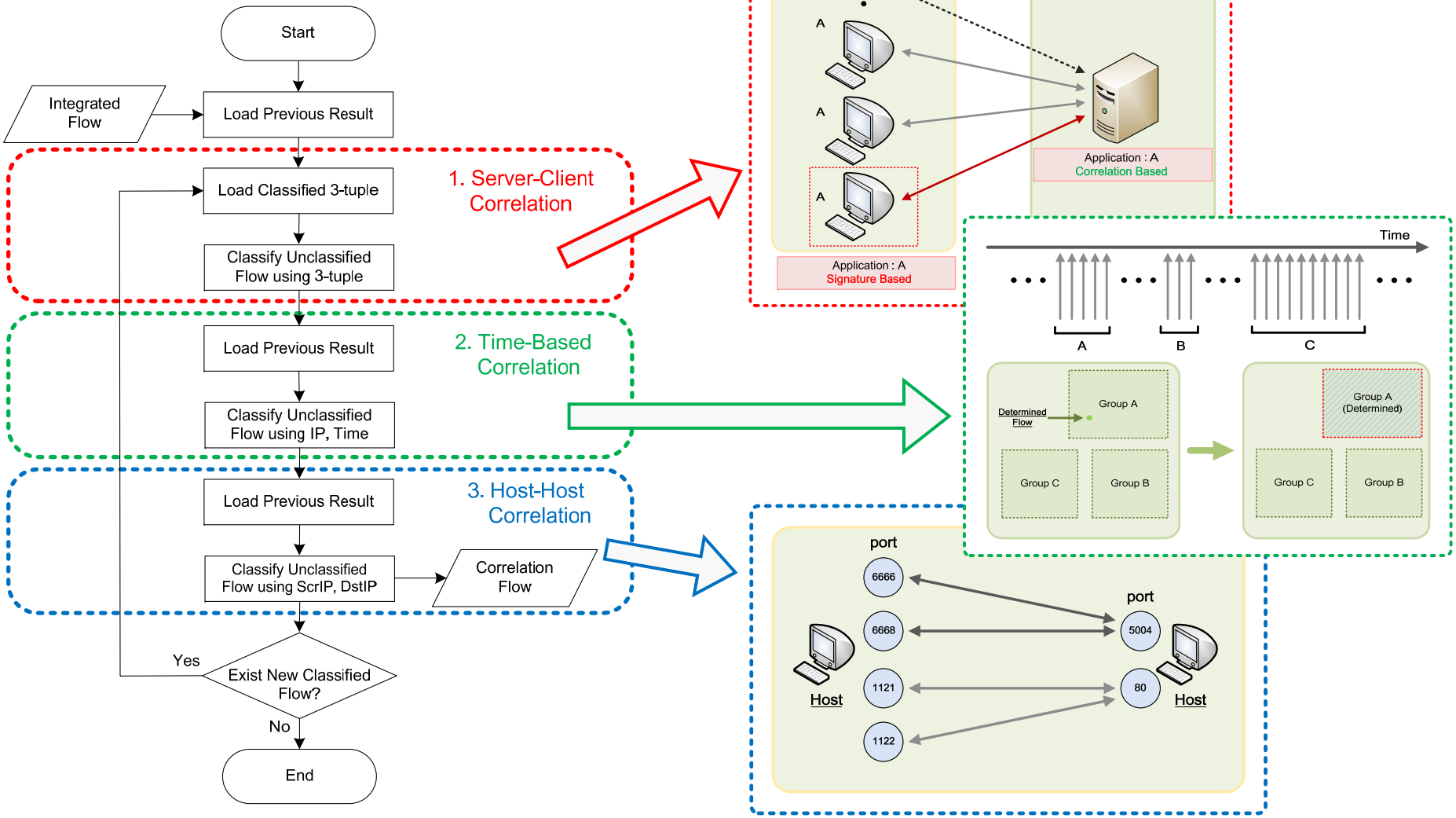
<Time-based Correlation>



<Host-Host Correlation>

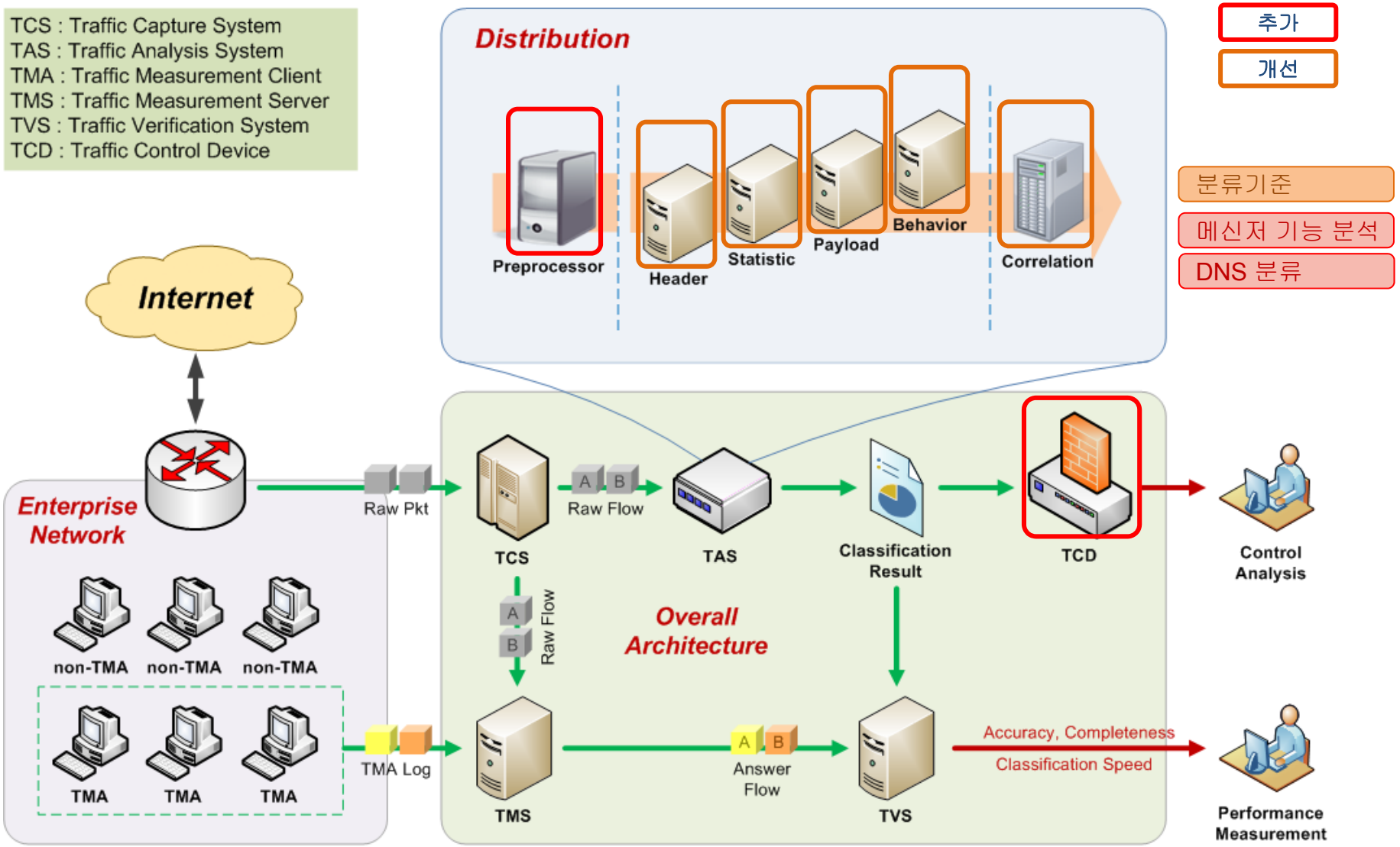
Flow Correlation

➢ Correlation 분석 순서도

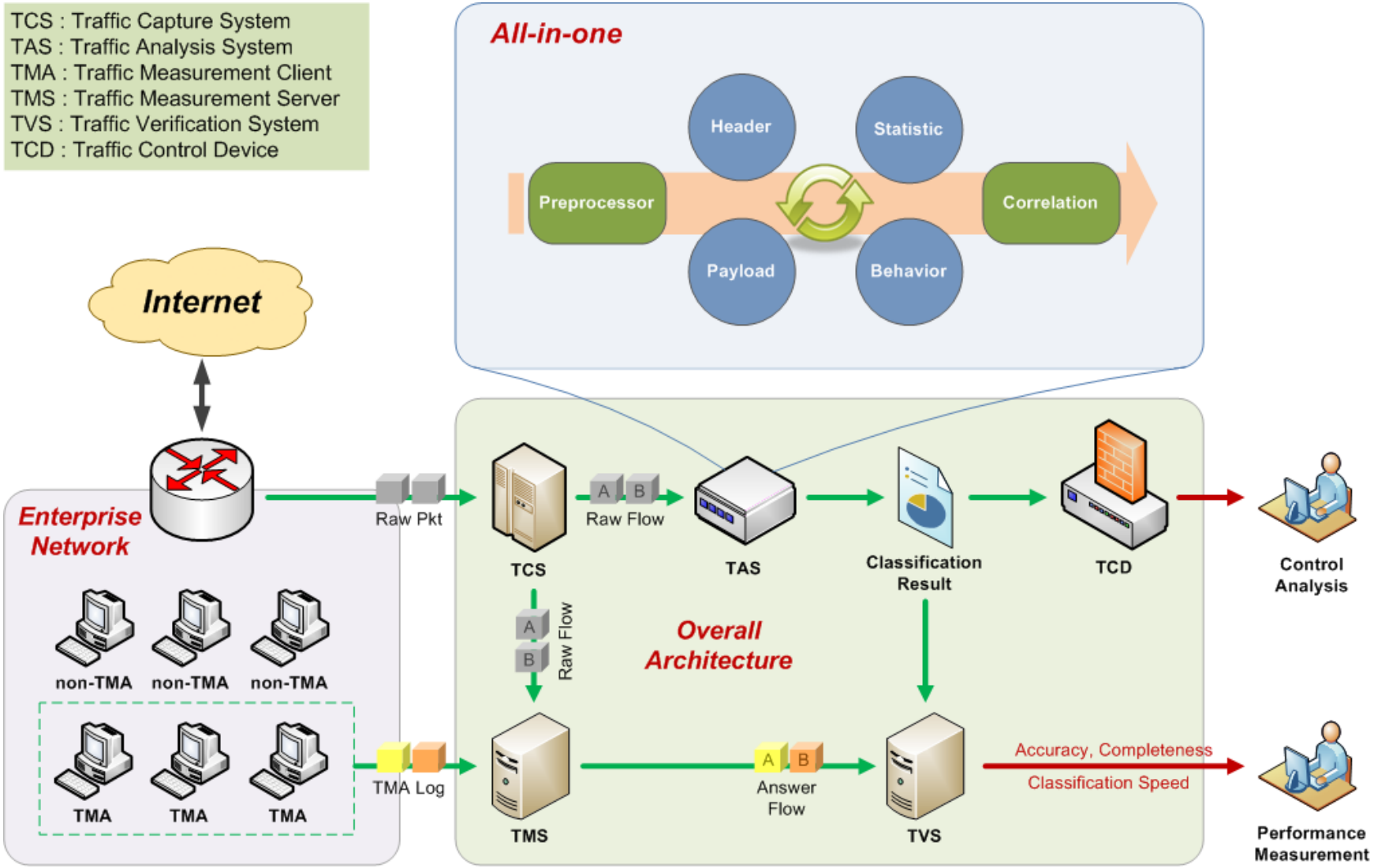


시스템 전체 구성도

전체 구성도(Distributed-to-min)



전체 구성도(All-in-one)



트래픽 분류 결과 검증

- 시그니쳐(Signature)
- All-in-one 시스템의 성능 측정
- 분석기 별 분류 결과

실험목적

- 실험목적
  - All-in-one 시스템의 정상적인 동작 확인
  - 지난 연구의 분류 결과와 비교를 통한 알고리즘 성능 향상 검증

- 실험 방법
  - All-in-one 시스템의 부하 측정
    - 학내 망과 기숙사 망
    - 온라인 테스트
      - 하루 동안의 CPU 사용량, Bandwidth, Session 측정
    - 오프라인 테스트
      - 분석 시간 측정
  - 각 분석기 별 분석률 및 정확도 측정
    - 학내 망
    - Header, Statistic, Payload, Behavior, Correlation
    - 지난 연구와의 비교

Signature

➢ 시그니쳐 개수

	Application	Process	Signature
Header	55	77	44,812
Statistic	114	181	3,714
Payload	149	239	835

➢ 시그니쳐 응용 별 분포

Rank	Header			Statistic			Payload		
	App.	#	Dist.	App.	#	Dist.	App.	#	Dist.
1	torrent	33,510	74.78%	melon player	366	9.85%	donkey	40	4.79%
2	fileguri	7,388	16.49%	nateon	307	8.27%	afreeca	31	3.71%
3	fasoo drm	1,226	2.74%	windows	224	6.03%	windows live	29	3.47%
4	windows	1,176	2.62%	cymini	187	5.04%	windows	27	3.23%
5	internet explorer	770	1.72%	gretech_gomplayer	181	4.87%	estsoft_alsong	25	2.99%
6	skype	277	0.62%	alwil software	165	4.44%	nateon	24	2.87%
7	nateon	100	0.22%	internet explorer	147	3.96%	gretech_gomplayer	20	2.40%
…	…	…	…	…	…	…	…	…	…
Total		44,812	100%		3,714	100%		835	100%

네트워크 및 시스템

➢ 테스트베드 네트워크

No.	Location	Bandwidth (average)	Bandwidth (peak)	Verification
1	고려대	100Mbps	350Mbps	Yes
2	고려대(기숙사)	250Mbps	1,400Mbps	No
3	강원대	22 Mbps	84 Mbps	No
4	포항공대	250 Mbps	485 Mbps	No

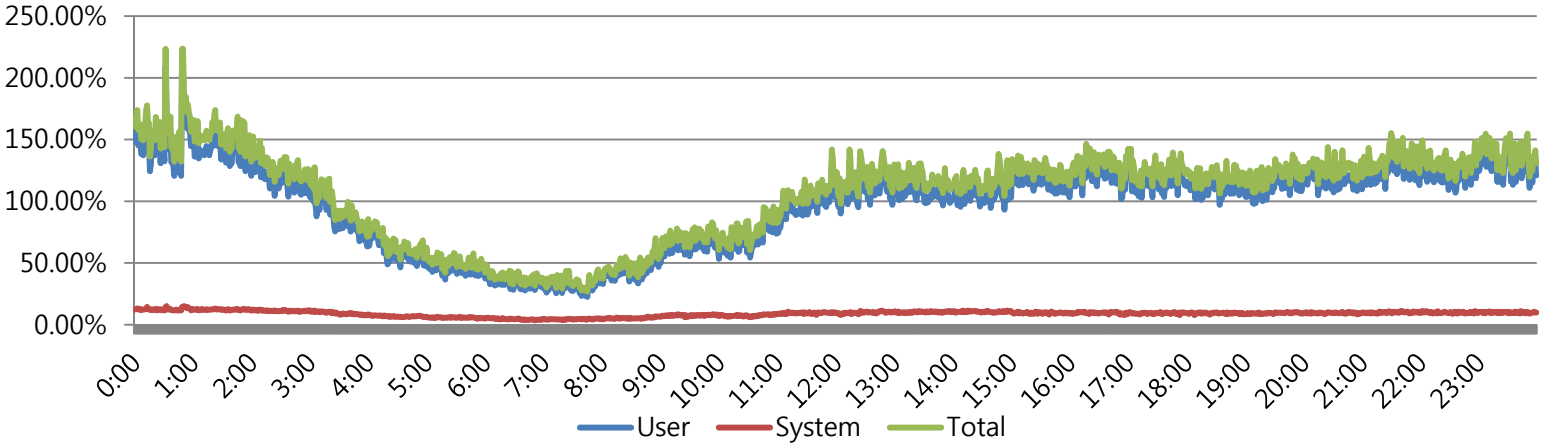
- **All-in-one 시스템**의 최종 테스트 베드 = **고려대(기숙사)**

➢ 시스템 사양

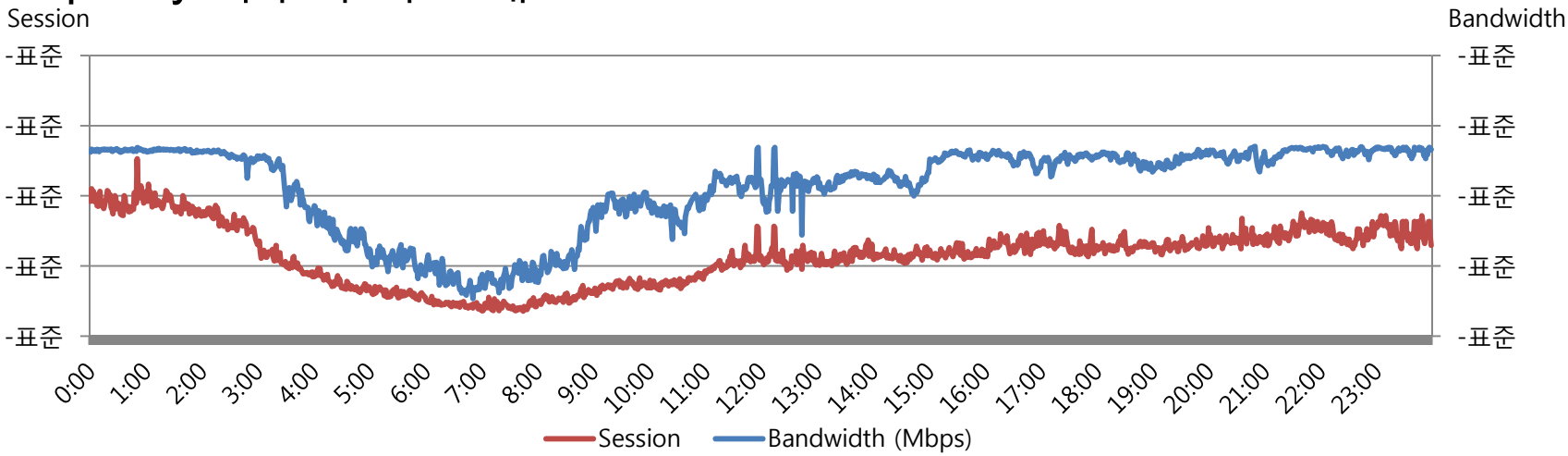
Location	고려대 학내망	고려대 기숙사망
CPU	Intel Core2 Quad Q6600 2.4G	Intel Core i7 930 2.8G
Memory	4.0G(ramdisk – 2.0G)	12.0G (ramdisk – 8.0G)
Linux	CentOS 5.5, 2.6.18	CentOS 5.5, 2.6.18

All-in-one (기숙사)

- CPU Usage 하루 추이 그래프 (8 cores by hyper-threading)



- Capacity 하루 추이 그래프



All-in-one (기숙사)

➢ 성능 테스트 – CPU Usage & Capacity

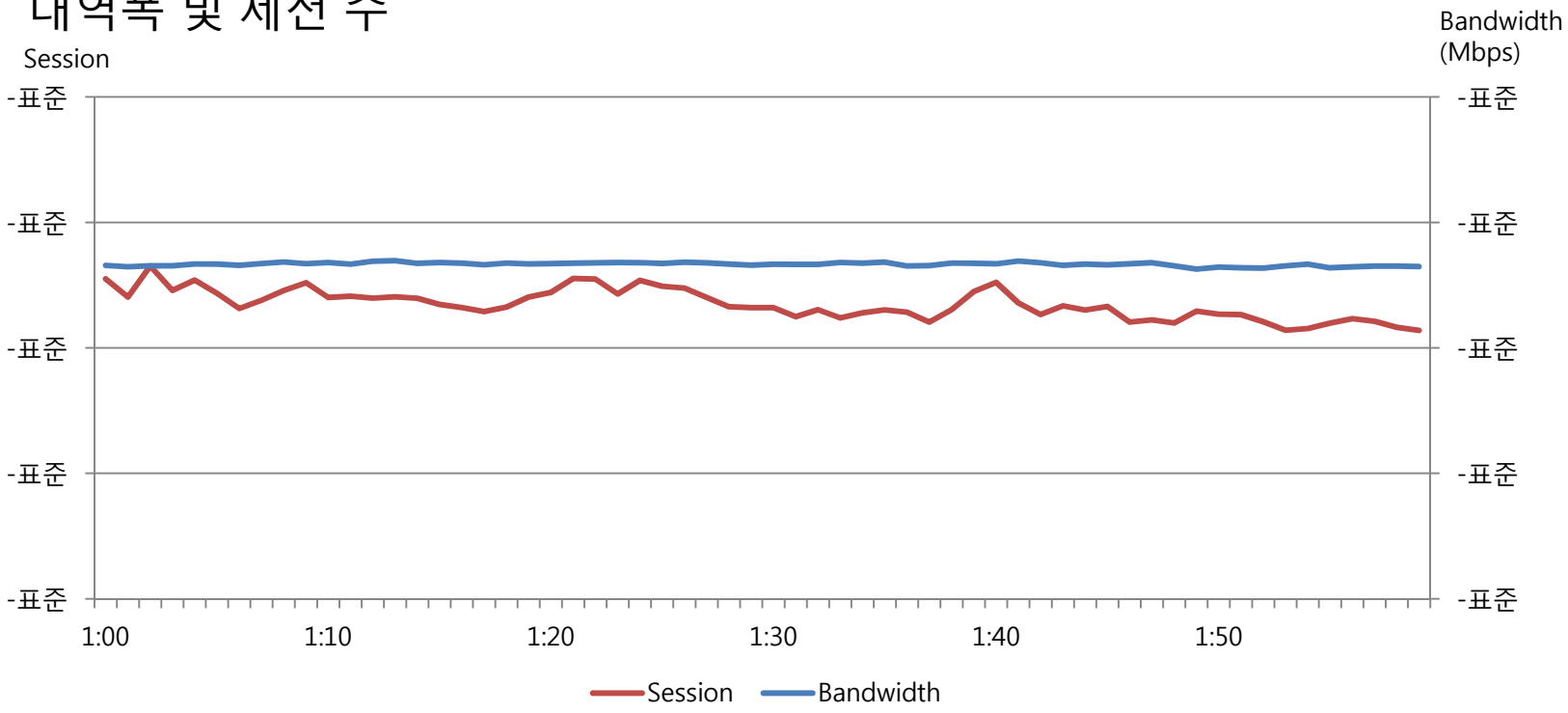
	CPU Usage			Bandwidth	Session /min
	User	System	Total		
Min	20.30%	3.51%	26.87%	267.85 Mbps	17,799
Max	209.03%	96.18%	223.42%	1351.70 Mbps	126,353
Average	96.48%	9.08%	105.56%	1061.93 Mbps	56,971

➢ CPU별 Usage (for total system)

	CPU0	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6	CPU7
Min	1.50%	1.28%	1.15%	7.98%	0.97%	8.93%	1.88%	3.60%
Max	32.42%	26.43%	26.07%	37.65%	25.53%	37.98%	23.43%	35.08%
Average	11.73%	11.36%	12.50%	24.16%	12.47%	24.38%	11.48%	17.32%

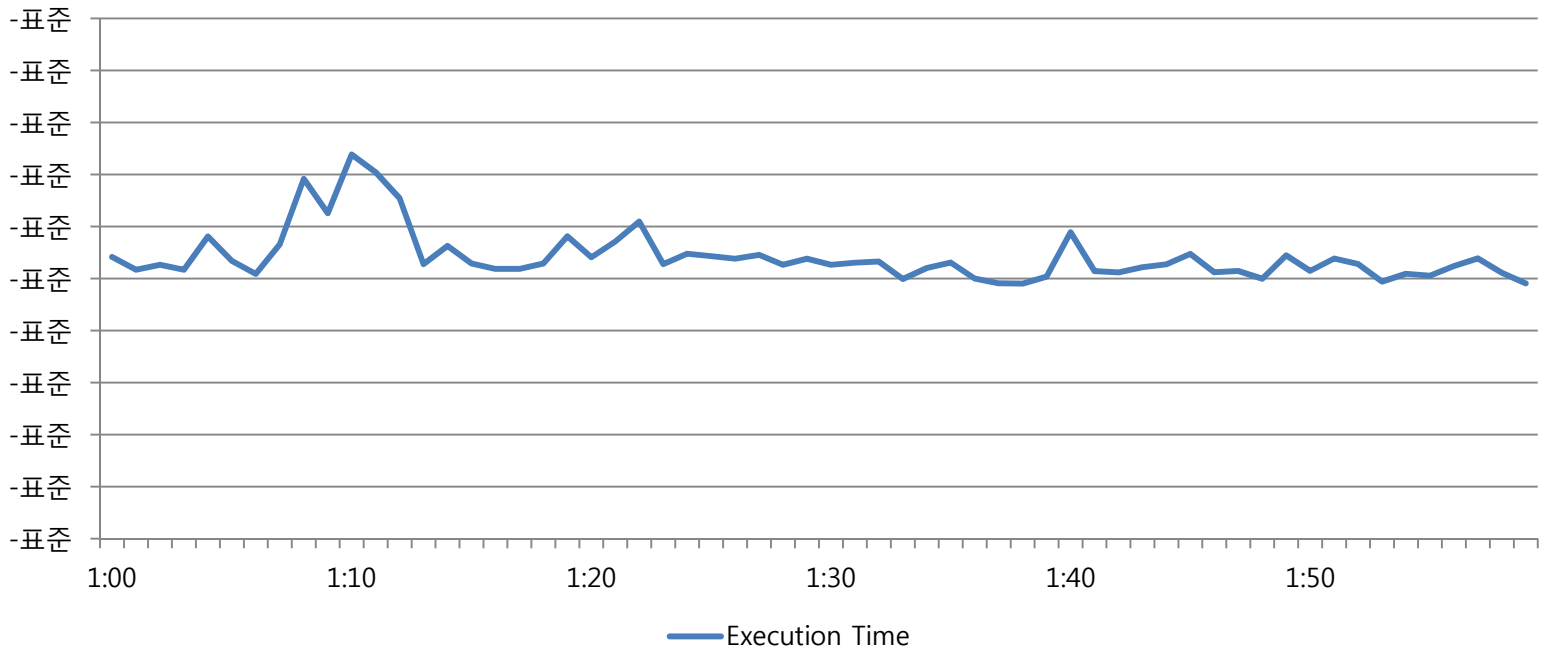
All-in-one (기숙사)

- 오프라인 성능 테스트
  - 대상 : 기숙사 트래픽(pcap 파일)
  - 기간 : 2010-12-13 01:00 ~ 2010-12-13 01:59 (1시간, 하루 중 가장 높은 대역폭 차지)
  - 실험방법 : 분 단위 60개의 파일을 분석하는 시간 측정
  - 대역폭 및 세션 수



All-in-one (기숙사)

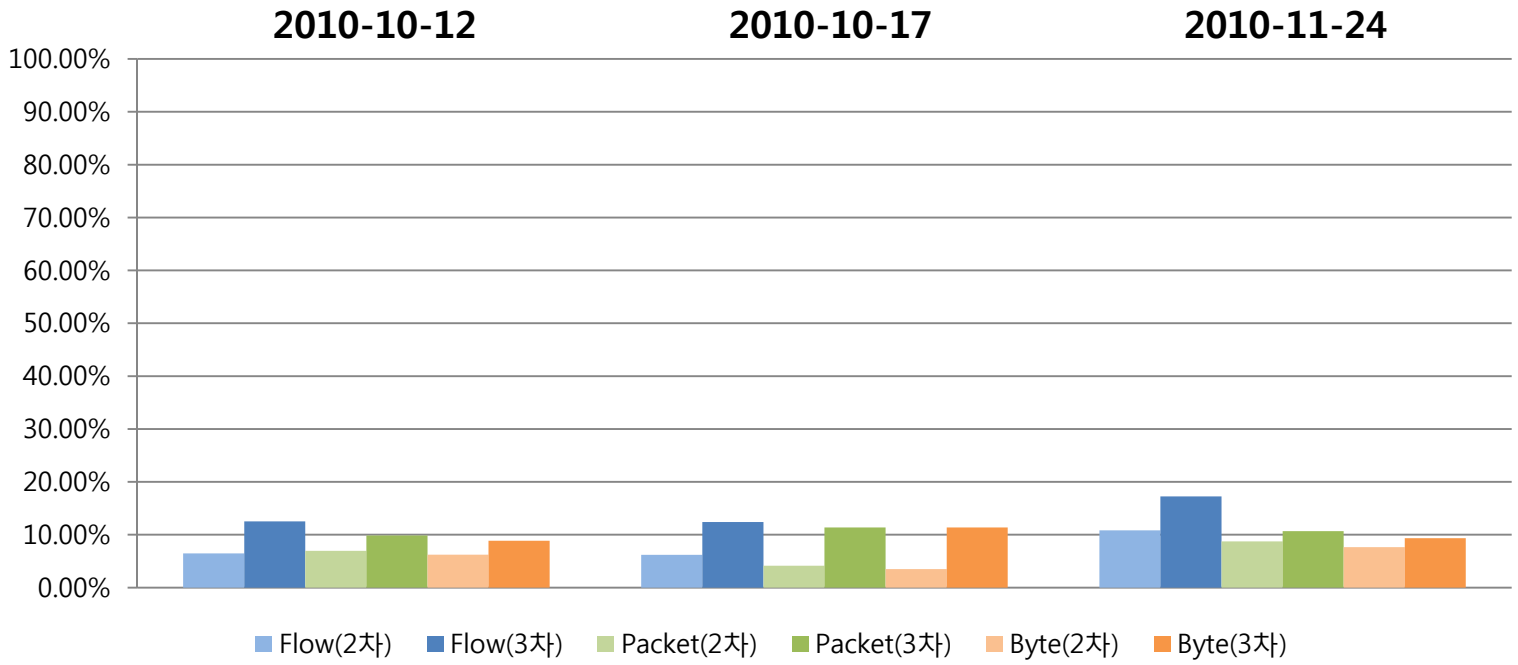
- 오프라인 성능 테스트
  - 대상 : 기숙사 트래픽(pcap 파일)
  - 기간 : 2010-12-13 01:00 ~ 2010-12-13 01:59 (1시간, 하루 중 가장 높은 대역폭 차지)
  - 실험방법 : 분 단위 60개의 파일을 분석하는 시간 측정
  - 분석 시간(s)



Header

➢ Completeness

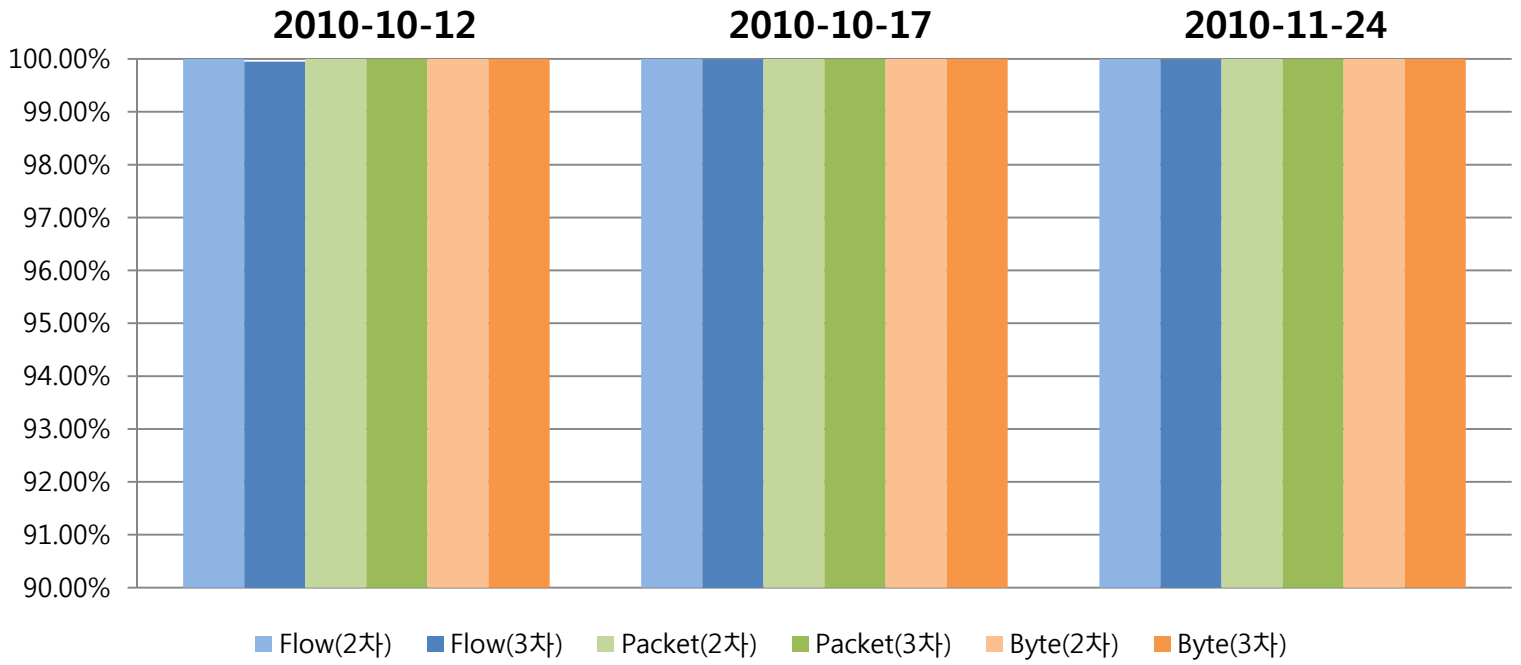
Date	Flow		Packet		Byte	
	2차	3차	2차	3차	2차	3차
2010-10-12	6.47%	12.53%	6.96%	9.86%	6.23%	8.86%
2010-10-17	6.20%	12.41%	4.13%	11.36%	3.51%	11.36%
2010-11-24	10.84%	17.24%	8.74%	10.68%	7.66%	9.36%



Header

➢ Accuracy

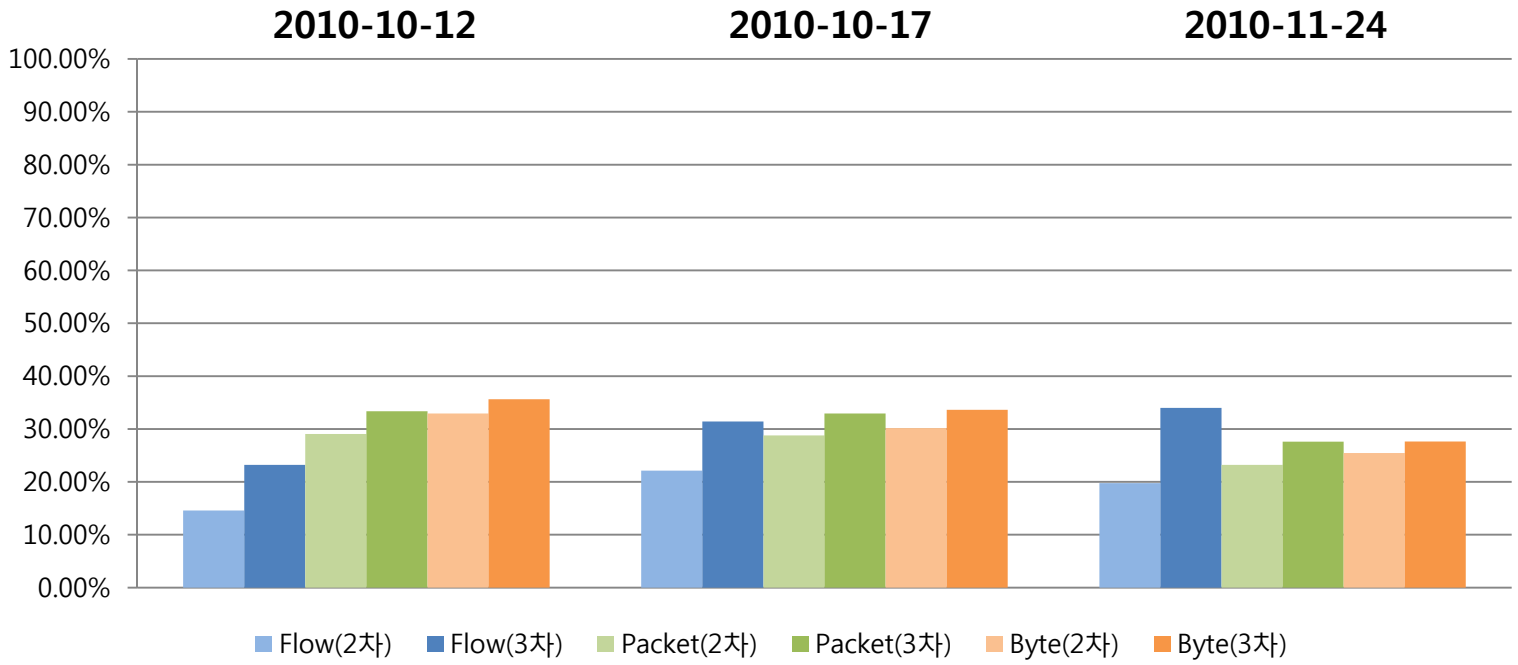
Date	Flow		Packet		Byte	
	2차	3차	2차	3차	2차	3차
2010-10-12	100.00%	99.95%	100.00%	100.00%	100.00%	100.00%
2010-10-17	100.00%	99.99%	100.00%	100.00%	100.00%	100.00%
2010-11-24	100.00%	99.99%	100.00%	100.00%	100.00%	100.00%



Statistic

➢ Completeness

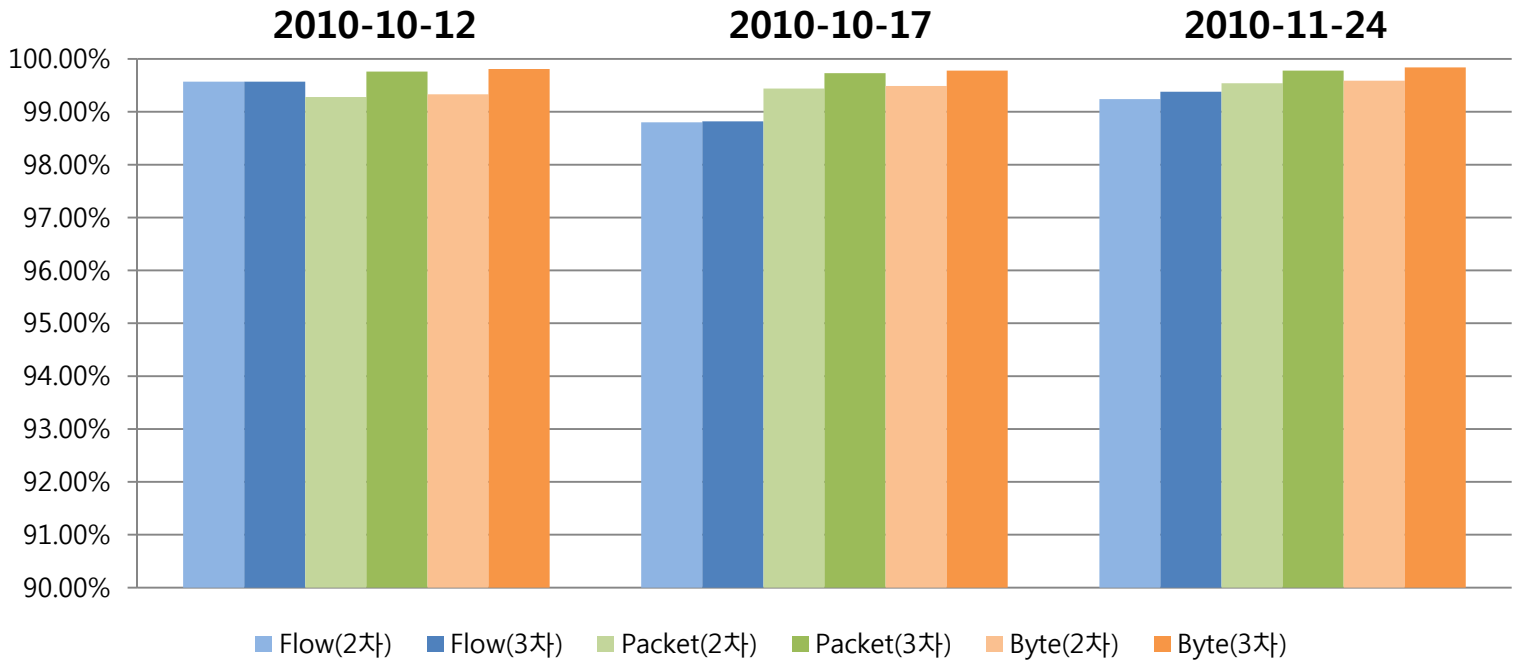
Date	Flow		Packet		Byte	
	2차	3차	2차	3차	2차	3차
2010-10-12	14.59%	23.20%	29.05%	33.37%	32.93%	35.63%
2010-10-17	22.12%	31.43%	28.80%	32.92%	30.19%	33.63%
2010-11-24	19.77%	33.99%	23.22%	27.59%	25.44%	27.62%



Statistic

➢ Accuracy

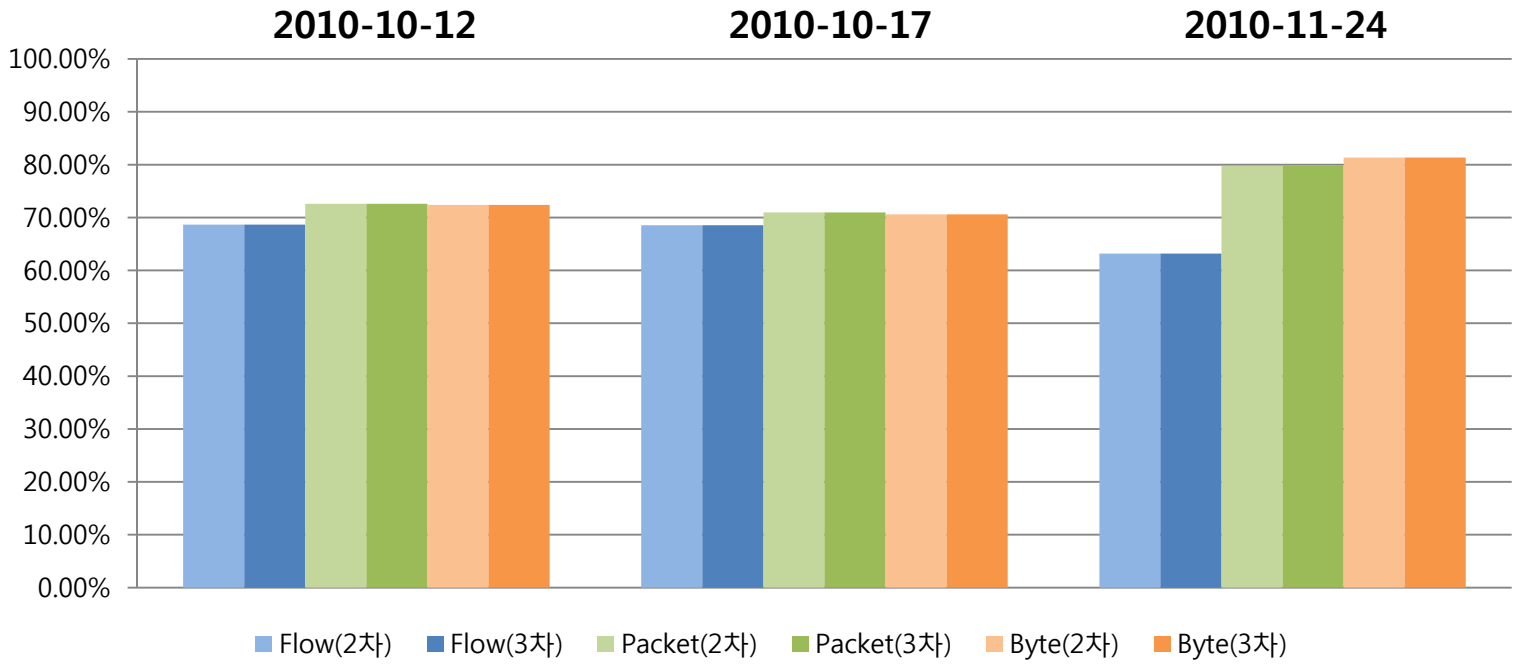
Date	Flow		Packet		Byte	
	2차	3차	2차	3차	2차	3차
2010-10-12	99.57%	99.57%	99.28%	99.76%	99.33%	99.81%
2010-10-17	98.80%	98.82%	99.44%	99.73%	99.49%	99.78%
2010-11-24	99.24%	99.38%	99.54%	99.78%	99.59%	99.84%



Payload

➢ Completeness

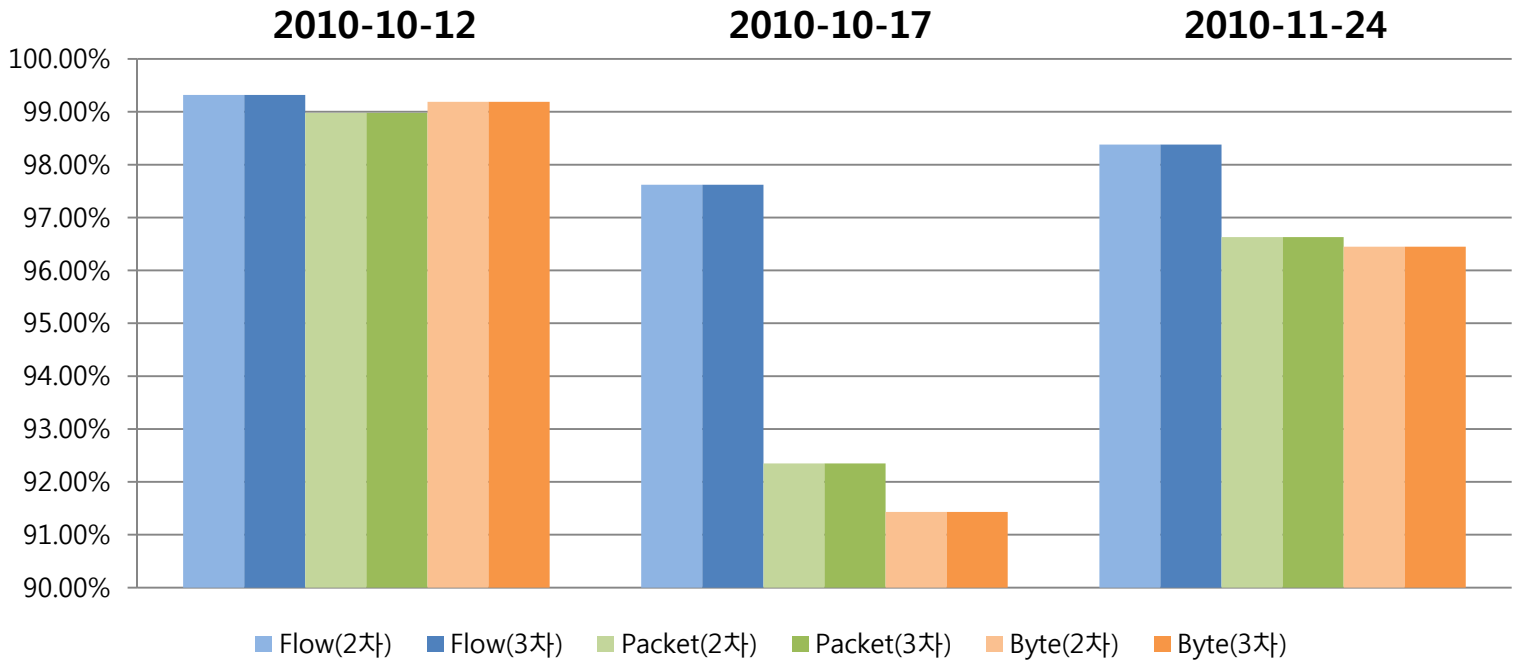
Date	Flow		Packet		Byte	
	2차	3차	2차	3차	2차	3차
2010-10-12	68.67%	68.67%	72.60%	72.60%	72.39%	72.39%
2010-10-17	68.55%	68.55%	70.98%	70.98%	70.59%	70.59%
2010-11-24	63.17%	63.17%	79.78%	79.78%	81.35%	81.35%



Payload

➢ Accuracy

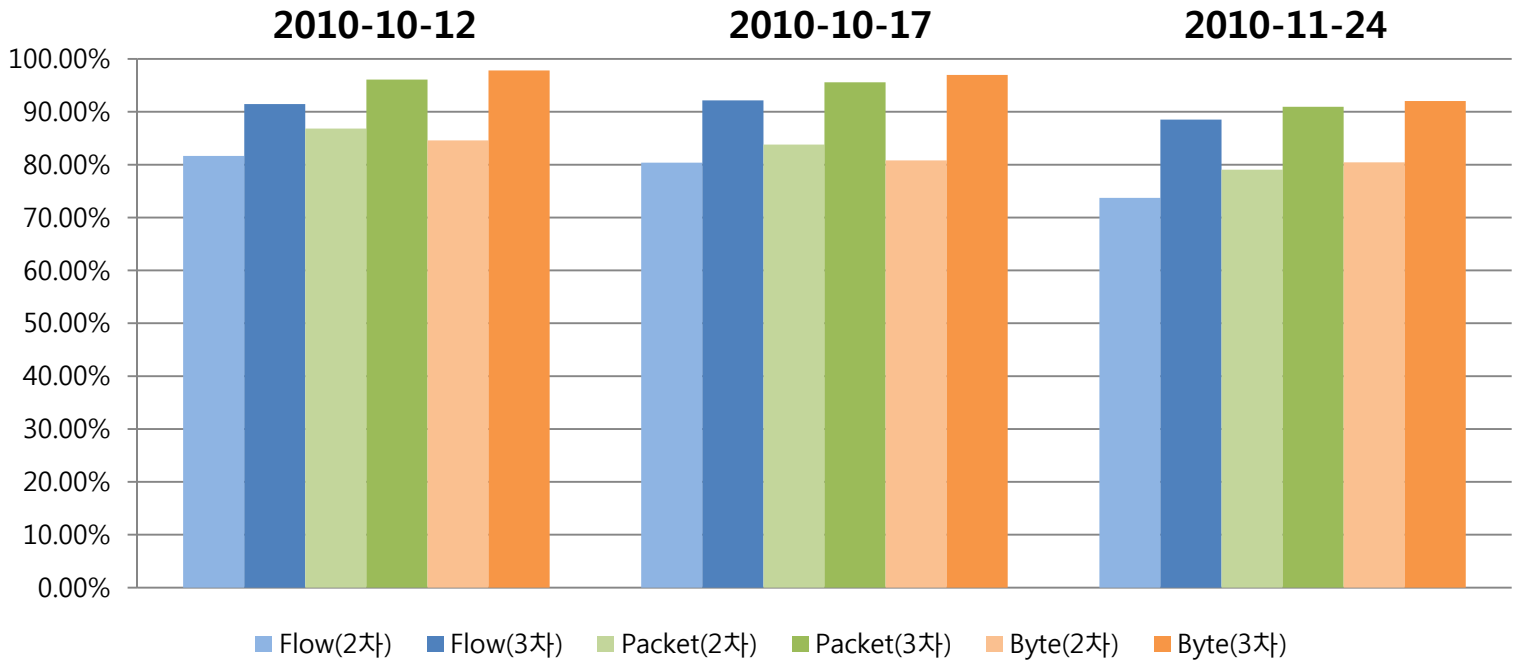
Date	Flow		Packet		Byte	
	2차	3차	2차	3차	2차	3차
2010-10-12	99.32%	99.32%	98.98%	98.98%	99.19%	99.19%
2010-10-17	97.62%	97.62%	92.35%	92.35%	91.43%	91.43%
2010-11-24	98.38%	98.38%	96.63%	96.63%	96.45%	96.45%



Correlation

➢ Completeness

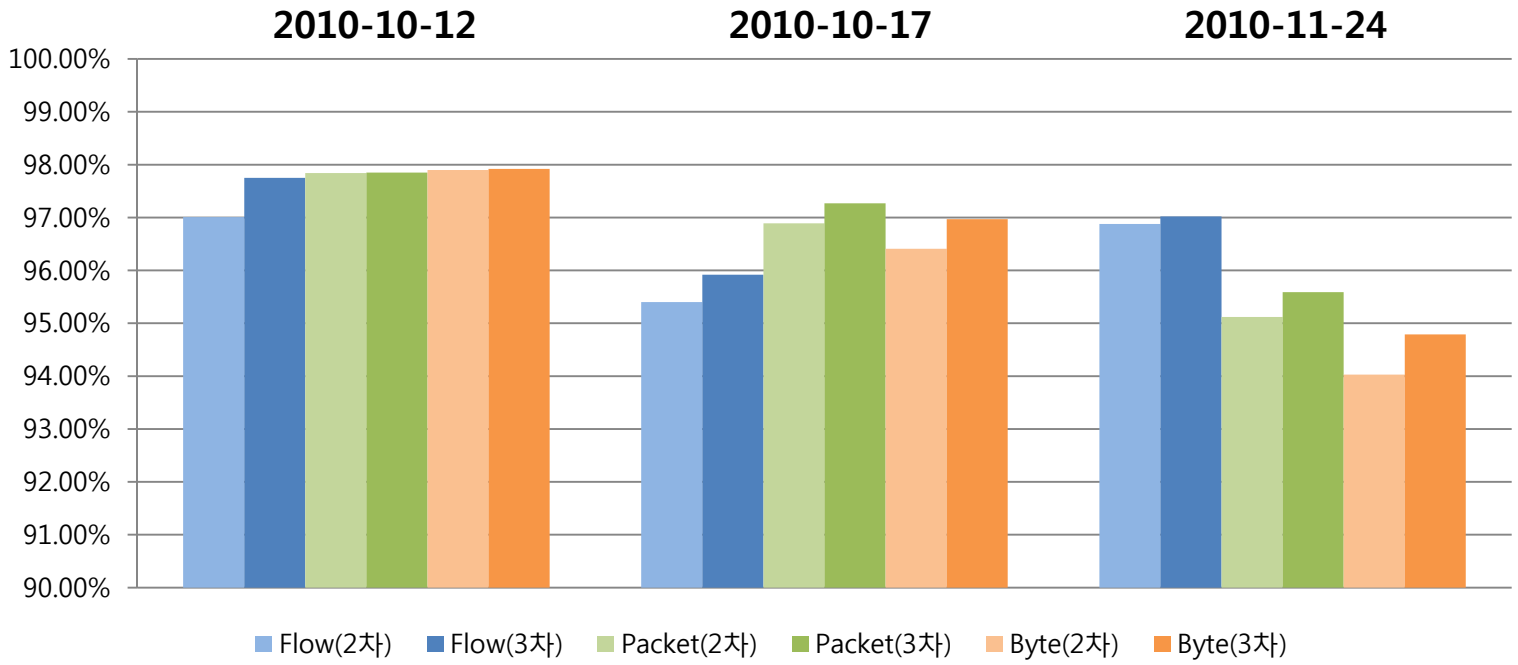
Date	Flow		Packet		Byte	
	2차	3차	2차	3차	2차	3차
2010-10-12	81.64%	91.48%	86.83%	96.10%	84.58%	97.81%
2010-10-17	80.37%	92.15%	83.81%	95.57%	80.82%	96.98%
2010-11-24	73.71%	88.53%	79.04%	90.94%	80.45%	92.04%



Correlation

➢ Accuracy

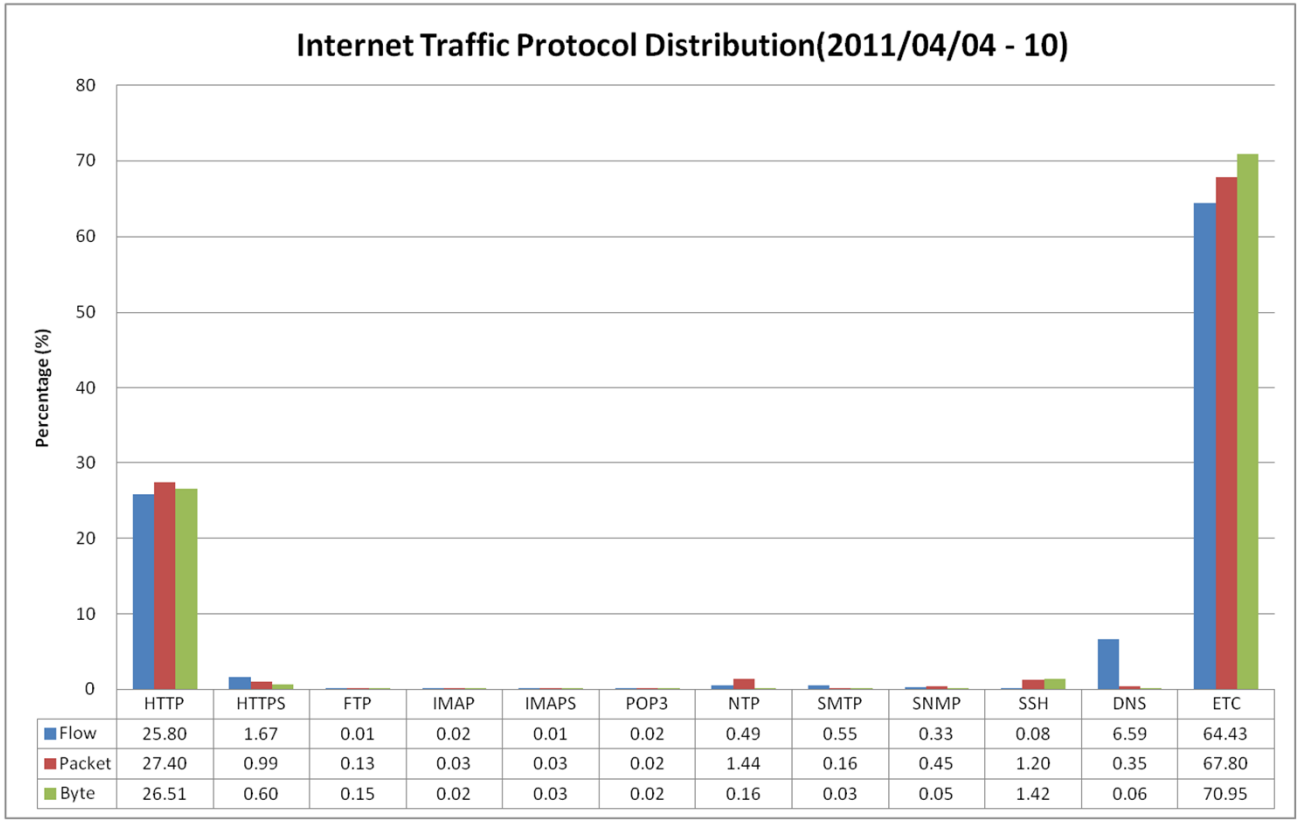
Date	Flow		Packet		Byte	
	2차	3차	2차	3차	2차	3차
2010-10-12	97.01%	97.75%	97.84%	97.85%	97.90%	97.92%
2010-10-17	95.40%	95.92%	96.89%	97.27%	96.41%	96.97%
2010-11-24	96.88%	97.02%	95.12%	95.59%	94.03%	94.79%



스마트 디바이스 트래픽 분류

스마트폰 트래픽의 특징

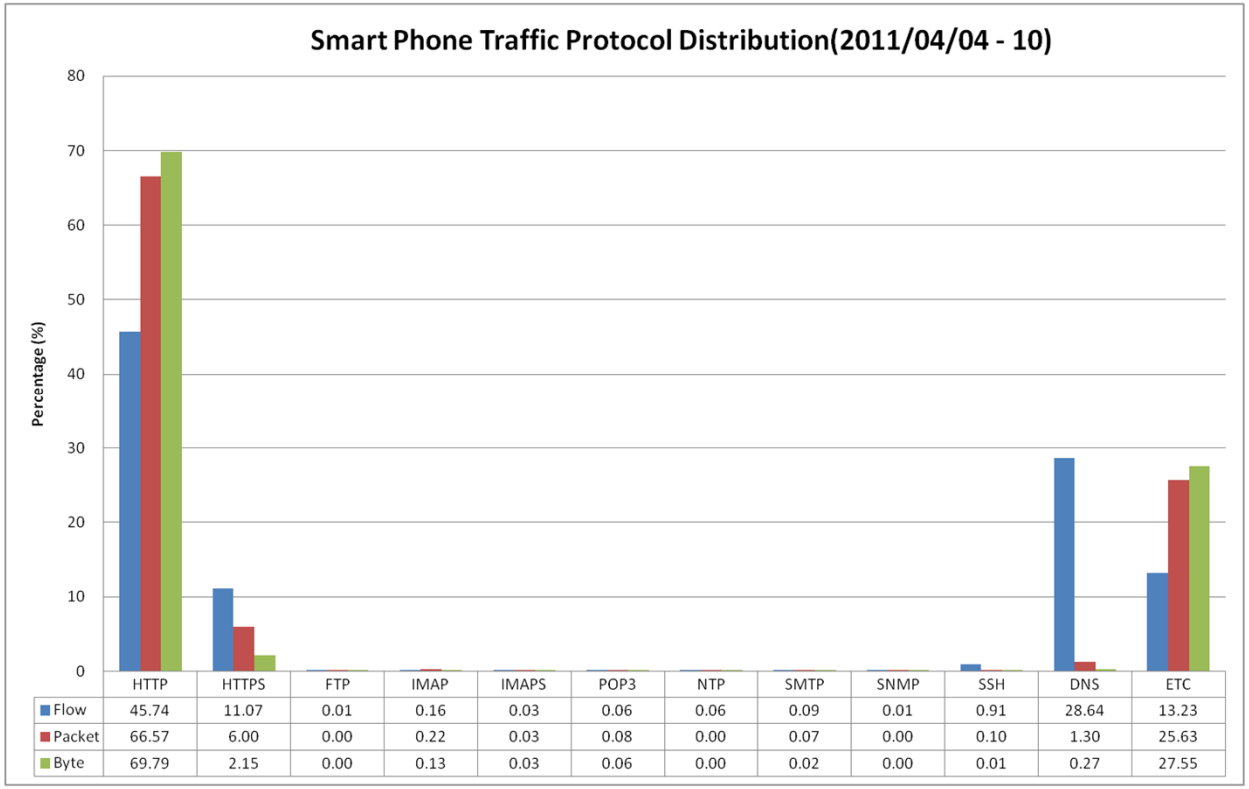
- 기존의 인터넷 트래픽
  - **HTTP** : 25%, **ETC** : 70 %
    - ETC → 응용 고유의 프로토콜
      - 각 응용 고유의 프로토콜의 분석 필요



	HTTP	HTTPS	FTP	IMAP	IMAPS	POP3	NTP	SMTP	SNMP	SSH	DNS	ETC
Flow	25.80	1.67	0.01	0.02	0.01	0.02	0.49	0.55	0.33	0.08	6.59	64.43
Packet	27.40	0.99	0.13	0.03	0.03	0.02	1.44	0.16	0.45	1.20	0.35	67.80
Byte	26.51	0.60	0.15	0.02	0.03	0.02	0.16	0.03	0.05	1.42	0.06	70.95

스마트폰 트래픽의 특징

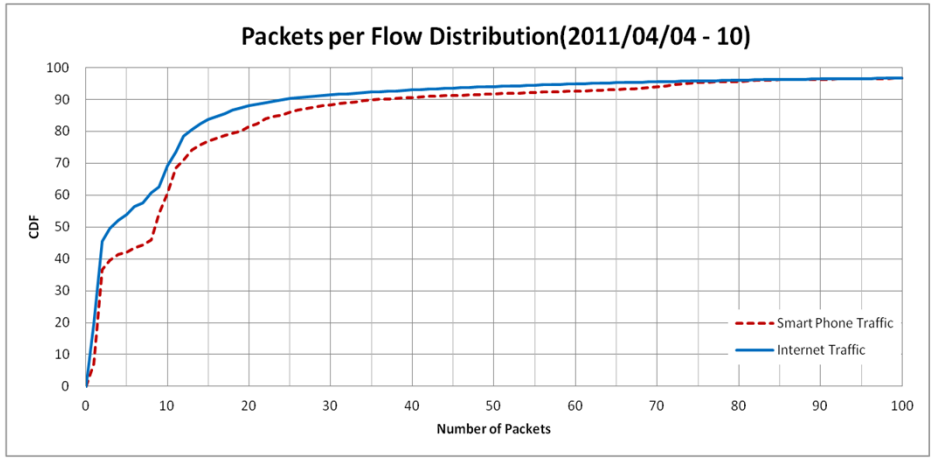
- 스마트폰 트래픽
  - **HTTP** : 70%, **ETC** : 27%
    - 스마트폰의 응용이 HTTP프로토콜을 통해 서비스
      - Ex : HTTP를 통한 응용의 XML, JSON 전송
    - HTTP의 분석이 필요



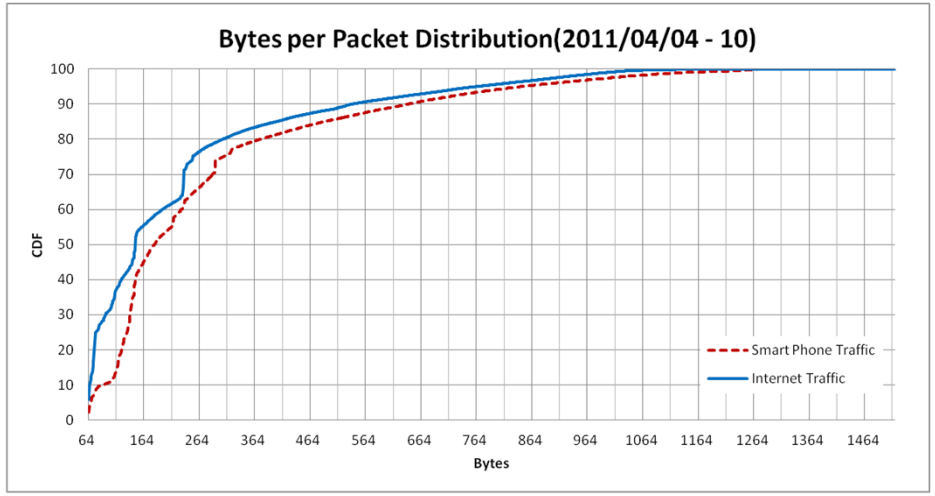
	HTTP	HTTPS	FTP	IMAP	IMAPS	POP3	NTP	SMTP	SNMP	SSH	DNS	ETC
Flow	45.74	11.07	0.01	0.16	0.03	0.06	0.06	0.09	0.01	0.91	28.64	13.23
Packet	66.57	6.00	0.00	0.22	0.03	0.08	0.00	0.07	0.00	0.10	1.30	25.63
Byte	69.79	2.15	0.00	0.13	0.03	0.06	0.00	0.02	0.00	0.01	0.27	27.55

스마트폰 트래픽의 특징

- 플로우당 패킷 분포
  - 스마트폰
    - 플로우 당 패킷 ↑
    - HTTP에 기반 이유
    - 10개 이하의 패킷
      - Small Web contents
      - Application Data
  - 기존의 인터넷
    - 10개 이하
      - P2P



- 패킷당 바이트량 분포
  - 스마트폰
    - 패킷당 바이트 ↑
    - HTTP에 기반 이유
    - 128바이트 이하
      - 메시지전송 기능
  - 기존의 인터넷
    - 128바이트 이하
      - P2P



서비스별 트래픽 분류 방법

- HTTP User-Agent의 활용방안
  - HTTP의 User-Agent 항목
    - 서비스 제공입장에서 중요한 정보
      - 사용자 정보 판별 목적
        - » 사용자의 기기 판별
        - » 브라우저 정보 습득
        - » 사용 Webkit 파악
    - 서버 측 : 디바이스 환경에 최적화된 웹 페이지 제공 가능
      - 단말에서의 서비스 제공
        - » 단말 별 해상도
        - » 화면크기 및 DPI(Dots per Inch)
  - 스마트폰에서의 User-Agent
    - 사용하는 응용의 정보를 기술
    - 아이폰 Facebook 응용의 User-Agent

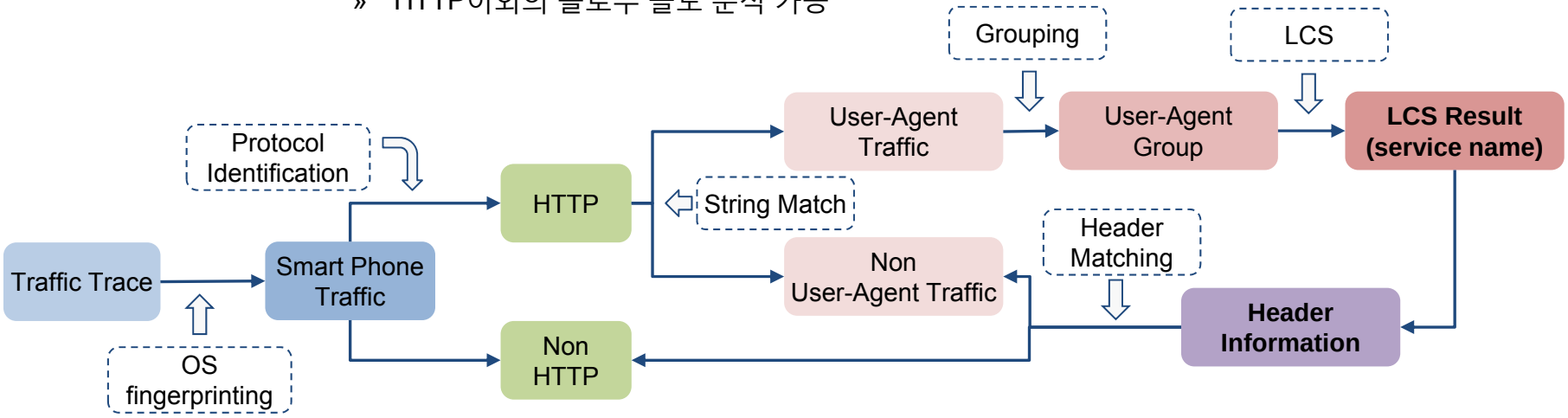
User-Agent: FacebookTouch3.4 OS/4.2.1 ko_KR Carrier/KT Device/iPhone3,1 Safari/iPhone

    - 안드로이드 Safari 엔진을 사용하는 웹 브라우저

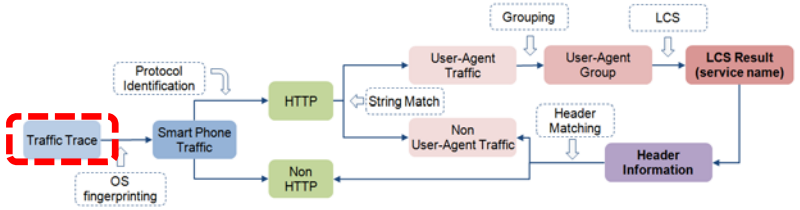
User-Agent: Mozilla/5.0 (Linux; U; Android 2.2.1; ko-kr; SHW-M180K Build/FROYO) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1

서비스별 트래픽 분류 방법

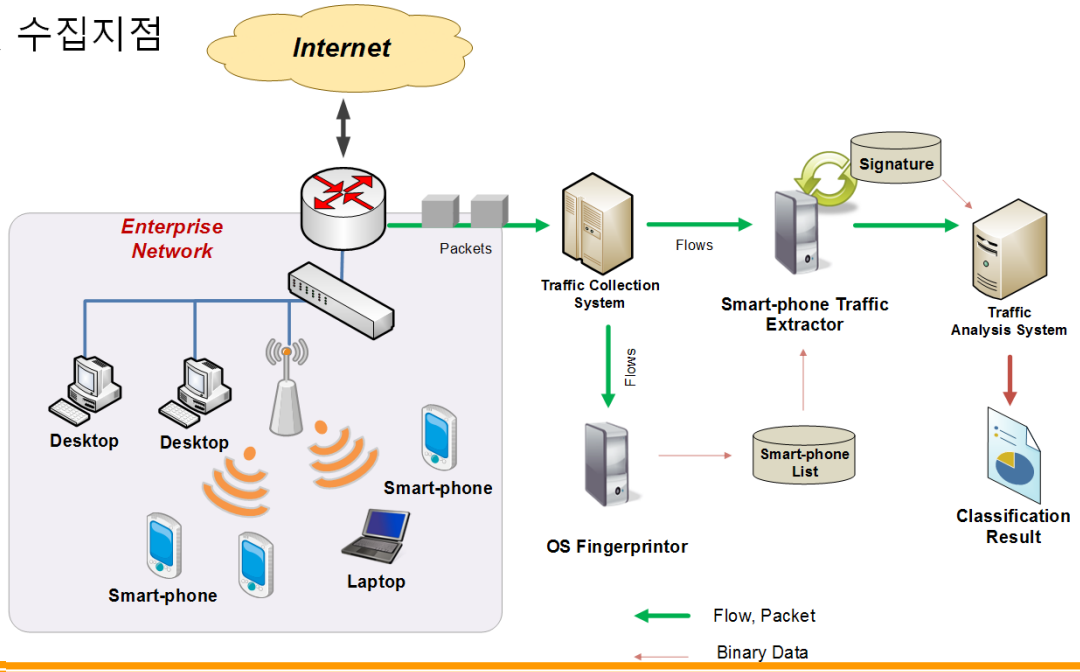
- 트래픽 분류방법
  - 스마트폰 트래픽 추출
    - OS fingerprinting
  - HTTP 트래픽 추출
    - HTTP User-Agent를 통한 분류
      - 서버 IP, Port, String Size를 통한 그룹핑
      - LCS(Longest Common Substring)알고리즘을 통한 공통 string(키워드) 추출
        - » 공통 string → 서비스 Identification
  - Non HTTP 트래픽
    - 헤더 정보를 이용한 추가 분석
      - 헤더 시그니쳐 구성
        - » HTTP이외의 플로우 들도 분석 가능



서비스별 트래픽 분류 방법

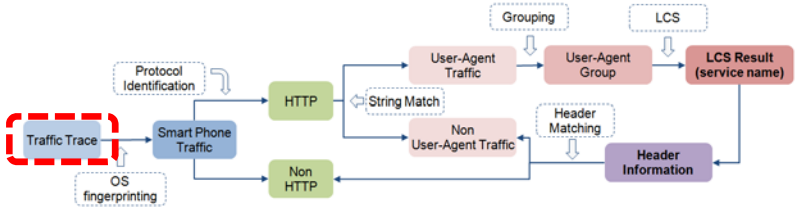


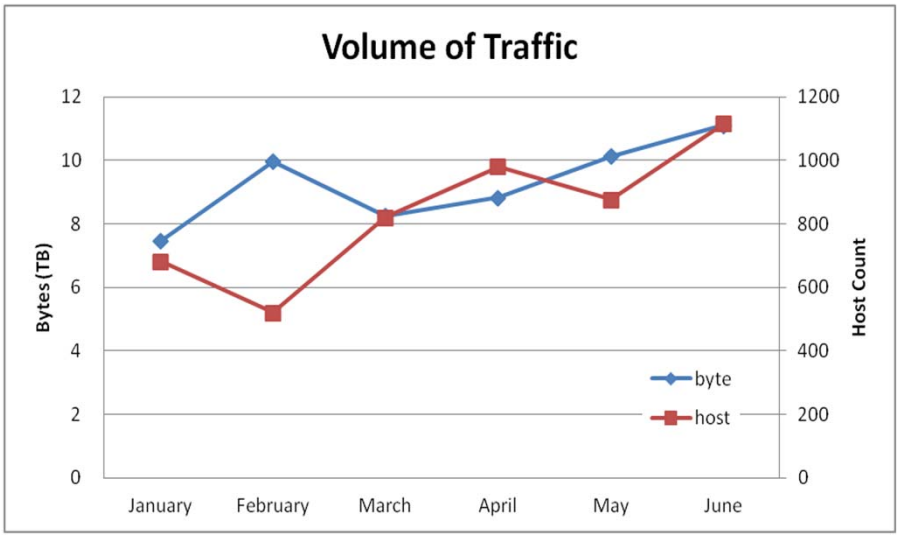
- 트래픽 트레이스
  - 트래픽 수집환경
    - 고려대학교 네트워크
    - 유, 무선, 서버 등의 약 3천 여대 호스트
      - 공용 PC, Laptop, 연구실, 강의실, etc
    - 최대 450Mbps
    - 하나의 라우터를 통해 인터넷과 연결
      - » 패킷 수집지점



서비스별 트래픽 분류 방법

- 트래픽 트레이스(cont.)
  - 스마트폰을 포함 모든 호스트
    - 공용 PC, 노트북, 서버, 스마트폰, Etc.
  - 각 월별 7일, 6개월 트래픽 수집
    - 총 53 TB의 트레이스

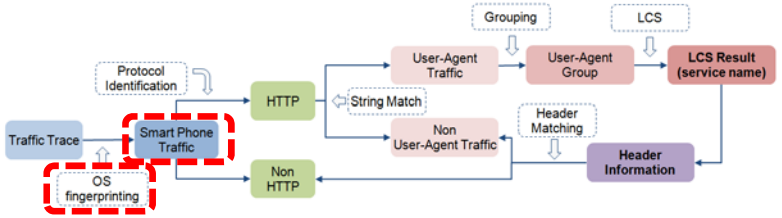


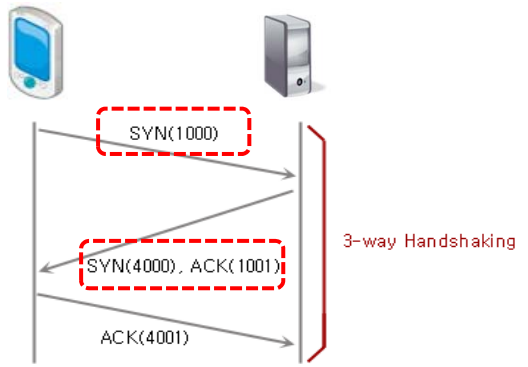


Month (duration)	Flows	Packet	Bytes	Host (/min)
January (7 days, 10~16)	271 x 10^6	9.04 x 10^9	7.5 TB	682.1
February (〃 , 21~27)	195 x 10^6	11.23 x 10^9	9.9 TB	520.2
March (〃 , 7~13)	341 x 10^6	10.07 x 10^9	8.2 TB	820.7
April (〃 , 4~10)	234 x 10^6	10.45 x 10^9	8.8 TB	980.5
May (〃 , 25~31)	303 x 10^6	12.18 x 10^9	10.1 TB	876.8
June (〃 , 6~12)	282 x 10^6	13.33 x 10^9	11.1 TB	1115.3
TOTAL				
2011-01 ~ 2011-06	1339 x 10^6	65.76 x 10^9	53.84 TB	825.5

서비스별 트래픽 분류 방법

- 스마트폰 트래픽 추출
  - OS Fingerprinting
    - TCP/IP의 헤더 정보 이용
      - 초기 통신을 위한 SYN or SYN/ACK 패킷
      - 운영체제 별 상이한 TCP/ IP 헤더 존재





Smart Phone	iPhone	Android	Windows Mobile	Symbian	BlackBerry
OS Version	iOS 3.x iOS 4.x	Cupcake Éclair Froyo Gingerbread	WinMo 6.1 WinMo 6.5	SymbianOS	BlackBerry

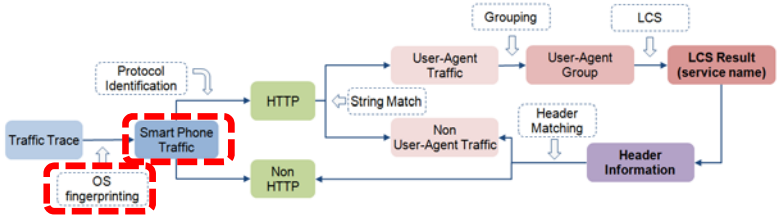
OS fingerprintor가 판별 가능한 스마트폰 운영체제

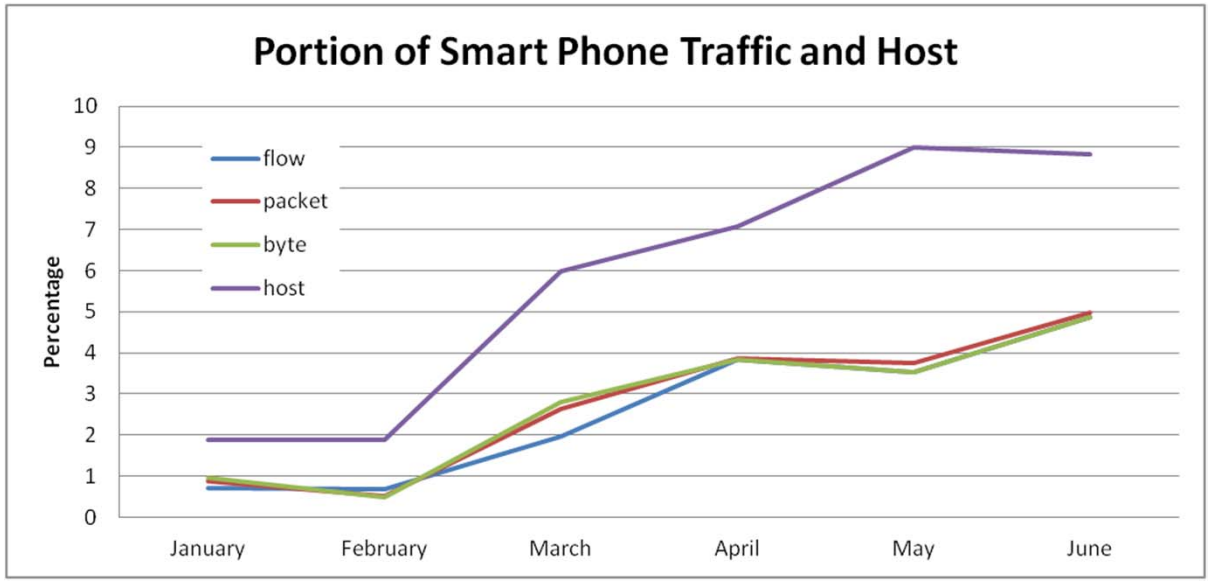
	구분	WinXP	Win7	WinMO	Android Eclair	iPhone iOS	Linux
IP Header	Initial TTL	128	128	128	64	64	64
TCP Header	Nop Count	2	3	3	1	3	1
	Timestamp	X	X	X	O	O	O
	SACK	O	O	O	O	O	O
	Window Size Scale	X	O	O	O	O	O
	Window Scale Factor	-	2	1	1	2	-
	Window Size	65535	8192	65535	5840	65535	5840
	Maximum Segment Size	1460	1460	1460	1460	1460	1460
	EOL	X	X	X	X	O	X

운영체제 별 SYN 패킷 패턴

서비스별 트래픽 분류 방법

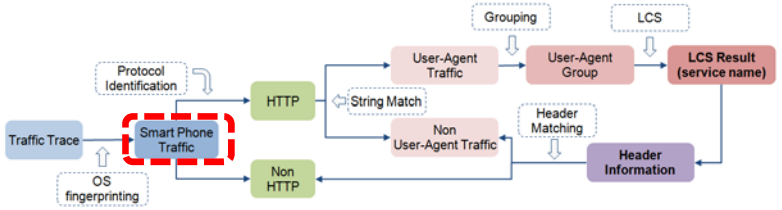
- 스마트폰 트래픽 추출(cont.)
  - 판별을 통한 스마트폰 트래픽 분포
    - 스마트폰 트래픽
      - Flow (43.5 x 106)
      - Bytes (1.59 TB)
      - Host (53 /min)
    - 전체대비 평균 발생 비율
      - Flow (2.6%)
      - Bytes (2.7%)
      - Host (5.8%)

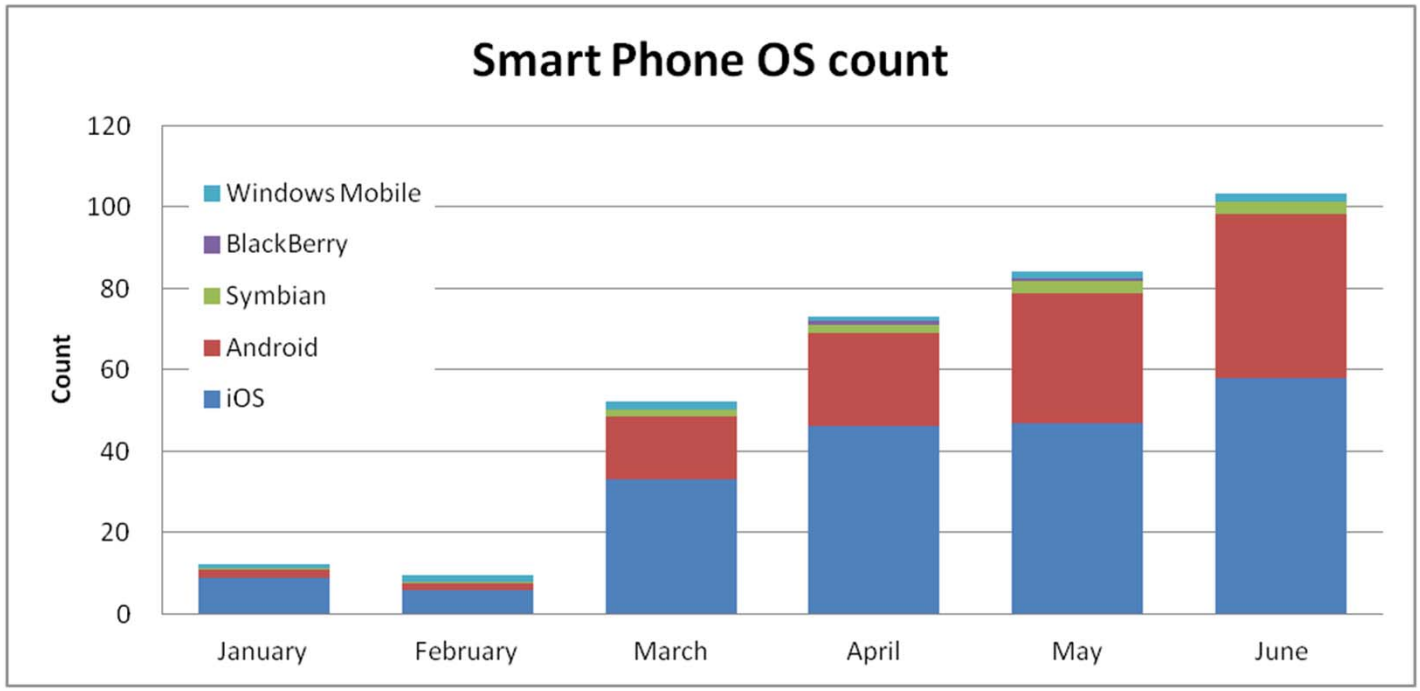




서비스별 트래픽 분류 방법

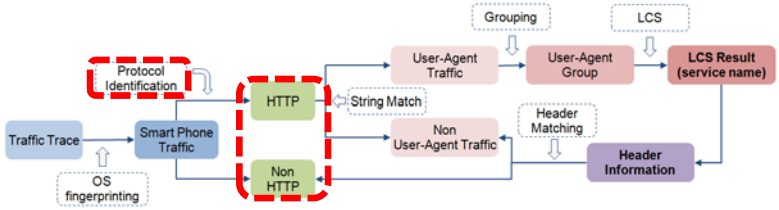
- 스마트폰 트래픽 추출(cont.)
  - 스마트폰 운영체제 분포
    - iOS, Android ↑
    - Android기반 운영체제의 증가량이 iOS 보다 ↑
      - 스마트폰 시장과의 밀접한 관계





서비스별 트래픽 분류 방법

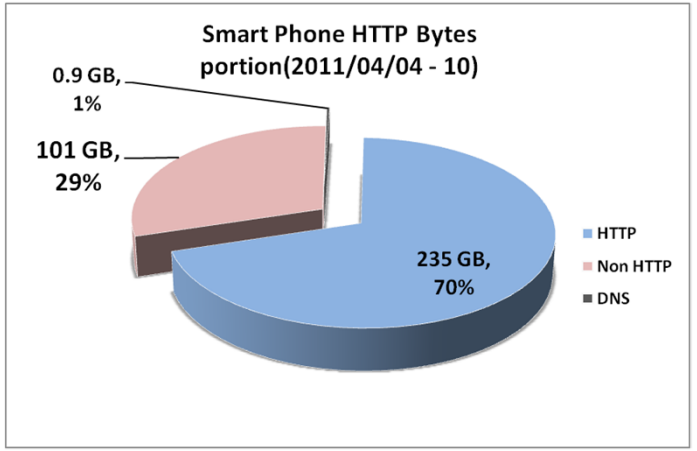
- 스마트폰 HTTP User-Agent
  - HTTP 트래픽 구분
    - Payload 기반의 Protocol Identification
      - HTTP 시그니쳐를 정규표현 형태로 기술
      - 시그니쳐 매칭을통한 HTTP 트래픽 구분
    - HTTP, Non HTTP 분포
      - 스마트폰 트래픽의 70%가 HTTP로 구성



Regular Expression
^HTTP/1\.
^GET@HTTP/1\.
^POST@HTTP/1\.
^.GET@HTTP/1\.
^content-length

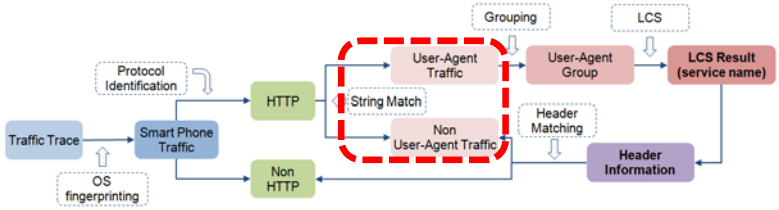
2011/04/04~10 HTTP 분포

	Flow	Packet	Bytes
Smart Phone	8.98 x 10^6 (100)	402 x 10^6 (100)	337 GB (100)
Non HTTP	4.77 x 10^6 (55.3)	135 x 10^6 (33.7)	102 GB (30.3)
HTTP	4.11 x 10^6 **(45.7)**	267 x 10^6 **(66.3)**	235 GB **(69.7)**
HTTP + DNS	6.68 x 10^6 **(74.4)**	273 x 10^6 **(67.9)**	236 GB **(70.0)**



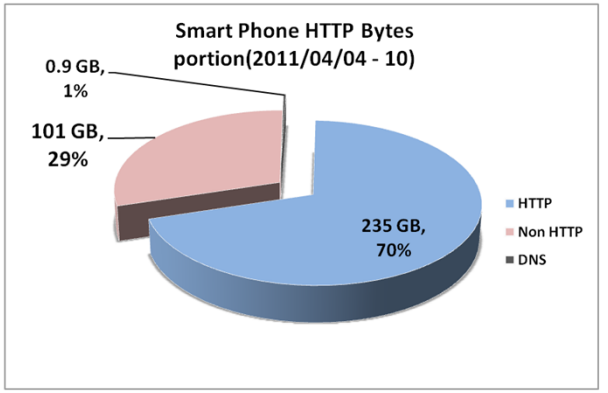
서비스별 트래픽 분류 방법

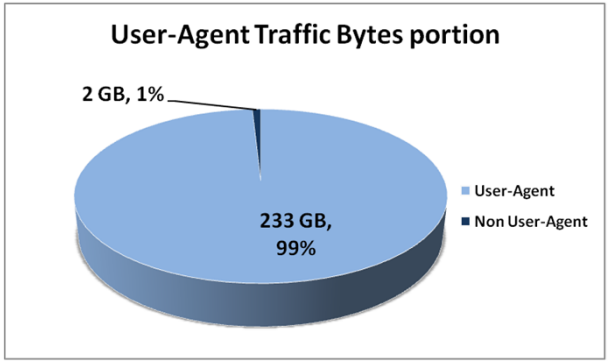
- 스마트폰 HTTP User-Agent(Cont.)
  - User-Agent, Non User-Agent 트래픽
    - HTTP트래픽중 User-Agent 분포
      - HTTP트래픽의 99%가 User-Agent필드를 명시



2011/04/04~10 HTTP User-Agent 분포

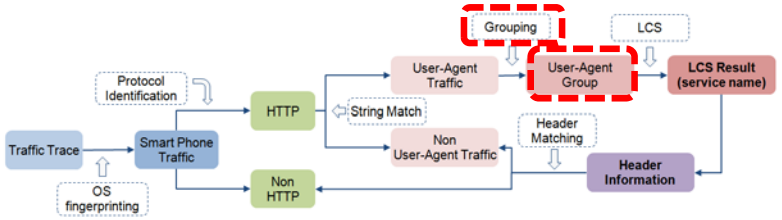
	Flow	Packet	Bytes
Smart Phone	8.98 x 10^6 (100)	402 x 10^6 (100)	337 GB (100)
Non HTTP	4.77 x 10^6 (55.3)	135 x 10^6 (33.7)	102 GB (30.3)
HTTP	4.11 x 10^6 (45.7)	267 x 10^6 (66.3)	235 GB (69.7)
User-Agent Traffic			
User-Agent	4.03 x 10^6 (44.8) **(HTTP portion:98)**	264 x 10^6 (65) **(HTTP portion:99)**	233 GB (69) **(HTTP portion:99)**

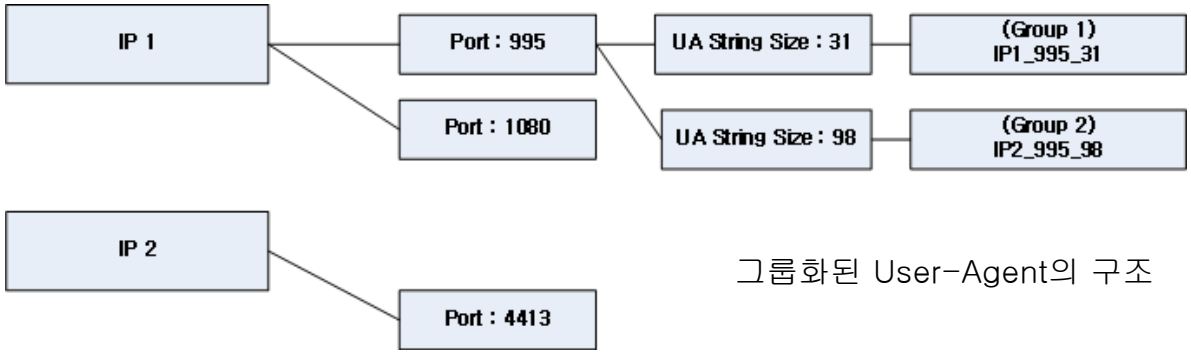




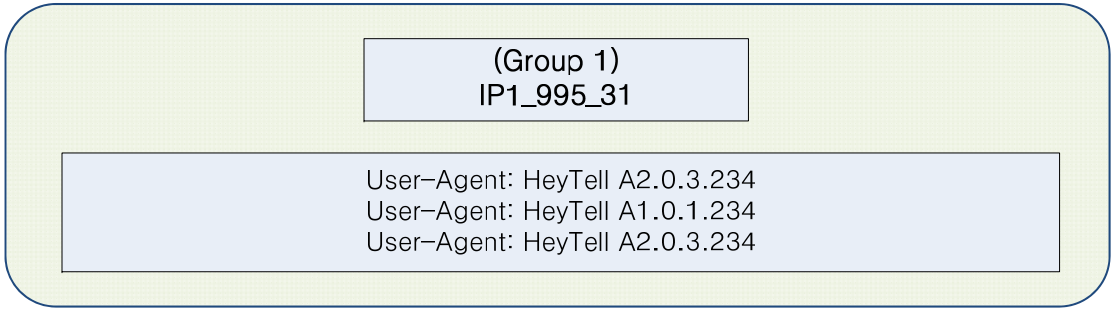
서비스별 트래픽 분류 방법

- User-Agent 그룹핑
  - LCS를 통한 공통 String의 원활한 추출을 위한 선행과정
  - 그룹핑 기준
    - Server Port, Server IP, String Size
  - 245,390개의 Group 생성





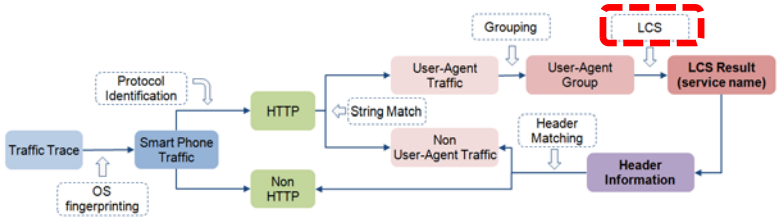
그룹화된 User-Agent의 구조



IP, Port, User-Agent Size로 그룹화된 Group 1

서비스별 트래픽 분류 방법

- LCS String Extraction
  - LCS알고리즘



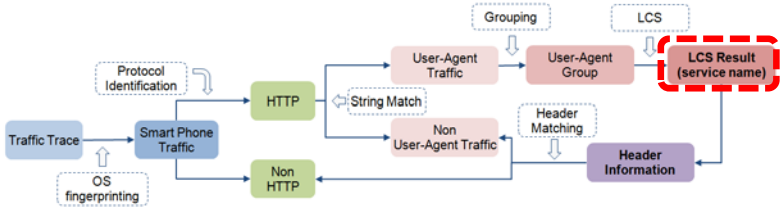
- LCS problem
  - 주어진 두 sequence X = <x1, x2,…,xm>와 Y = <y1, y2,…,yn>
  - X와 Y의 Longest Common Subsequence를 찾는 문제
  - Dynamic Programming 적용

- Example
  - sequence X = <A,B,C,B,D,A,B>
  - sequence Y = <B,D,C,A,B,A>
  - ANSWER : "BCBA"

i	Yj	j=0 B	1 D	2 C	3 A	4 B	5 A	6
0	Xi	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

서비스별 트래픽 분류 방법

- LCS String Extraction(cont.)
  - 각 User-Agent 그룹 → LCS 적용
    - 각각의 LCS결과를 서비스로 분류 가능



(Group 1)
IP1_995_31

User-Agent: HeyTell A2.0.3.234
User-Agent: HeyTell A1.0.1.234
User-Agent: HeyTell A2.0.3.234

LCS

HeyTell A.0.234

HeyTell 응용의 LCS 결과

- 네트워크 내에 여러 Version의 응용이 사용된 경우
  - User-Agent의 다양화
    - » 결과 : Version정보의 삭제 후 추출

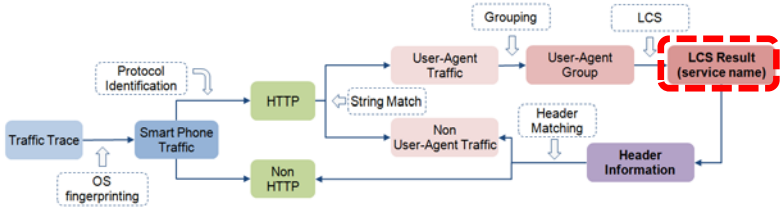
HeyTell A.0.234

- 네트워크 내에 하나의 Version의 응용이 사용된 경우
  - User-Agent의 단일화
  - Version 정보를 포함한 모든 결과가 추출

FaceWorldcup/2.0.0 CFNetwork/485.12.7 Darwin/10.4.0

서비스별 트래픽 분류 방법

- LCS String Extraction(cont.)
  - LCS 결과
    - 245,390개의 Group 에서 서로 다른 서비스 추출
      - 2,325개의 서로 다른 서비스 추출
      - Service Identification 가능
      - 페이로드 시그니쳐 로써 활용가능

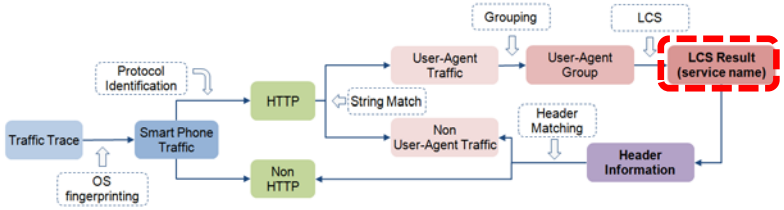


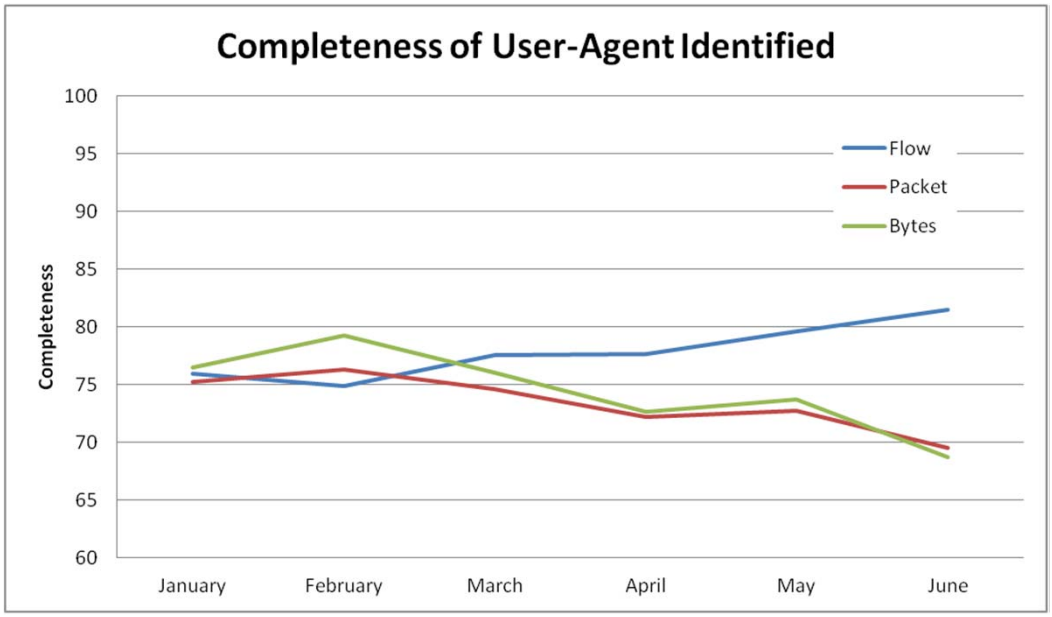
No.	User-Agent
1	MozilgeToeic
2	SpeedTest
3	AndroidSpeedtest
4	ElephantMarket
5	moreauFree
6	%ED%95%A0%EC%9D%B8%EC%9D%98%20%EB%8B%AC%EC%9D%B8/
7	Do.*Premium
8	PinkAge
9	%ED%8F%AC%EC%BC%93%EC%8A%A4%ED%83%80%EC%9D%BC/
10	DaumCloud
11	WooriBudong
12	TET4
13	StockToday
14	Trend_Final
15	humorUniv
…	…
2325	iRollerCoastr

각 응용의 이름을 구글링을 통해 확인

서비스별 트래픽 분류 방법

- LCS String Extraction(cont.)
  - Service Identified 결과
    - Byte 75% 의 분석률(dns 포함)
      - Flow(78 %)
      - Packet(73 %)
      - Byte(75 %)

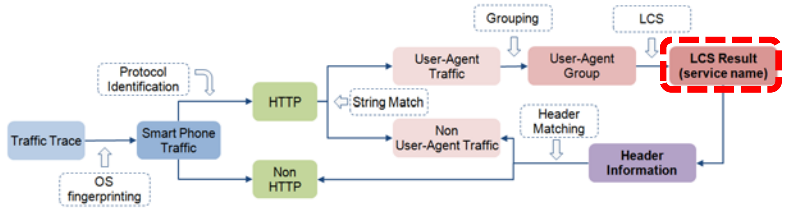




Completeness	Flow	Packet	Bytes
January	75.95	75.21	76.52
February	74.83	76.28	79.28
March	77.58	74.63	76.05
April	77.60	72.21	72.64
May	79.58	72.77	73.72
June	81.46	69.50	68.69
Average	77.83	73.43	74.48

서비스별 트래픽 분류 방법

- Non User-Agent 트래픽
  - User-Agent 명시되지 않은 트래픽
    - 일반적인 운영체제의 정보만 표현
      - 추가분석의 필요성



- Android hiThere 응용의 플로우

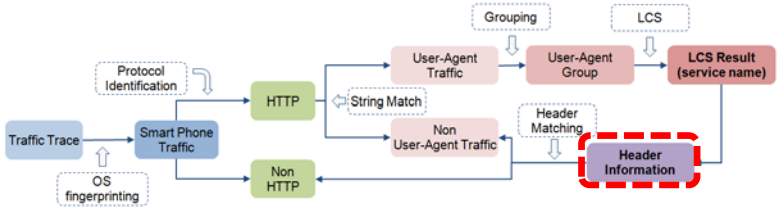
```
192.168.102.91   : 56203 --   6 -- 93.186.          :     80 [   ]
User-Agent: Dalvik/1.2.0 (Linux; U; Android 2.2.1; Desire HD Build/FRG83D)
Host: cs5029.vkontakte.ru
Connection: Keep-Alive
倉뿧流♪}(♪█b??믍zN?U?枋?젏回?‼?╁<D楠n%dF쀡┬9c뱷?0?J朱█*←}kㄸ?@?┬v┣?→鶩%젴m?5?챙<┴
POST /hiThere/MLoverXmlController.jsp HTTP/1.1
Content-Length: 637
Content-Type: application/x-www-form-urlencoded
Host: app.hithere.co.kr
Connection: Keep-Alive
```

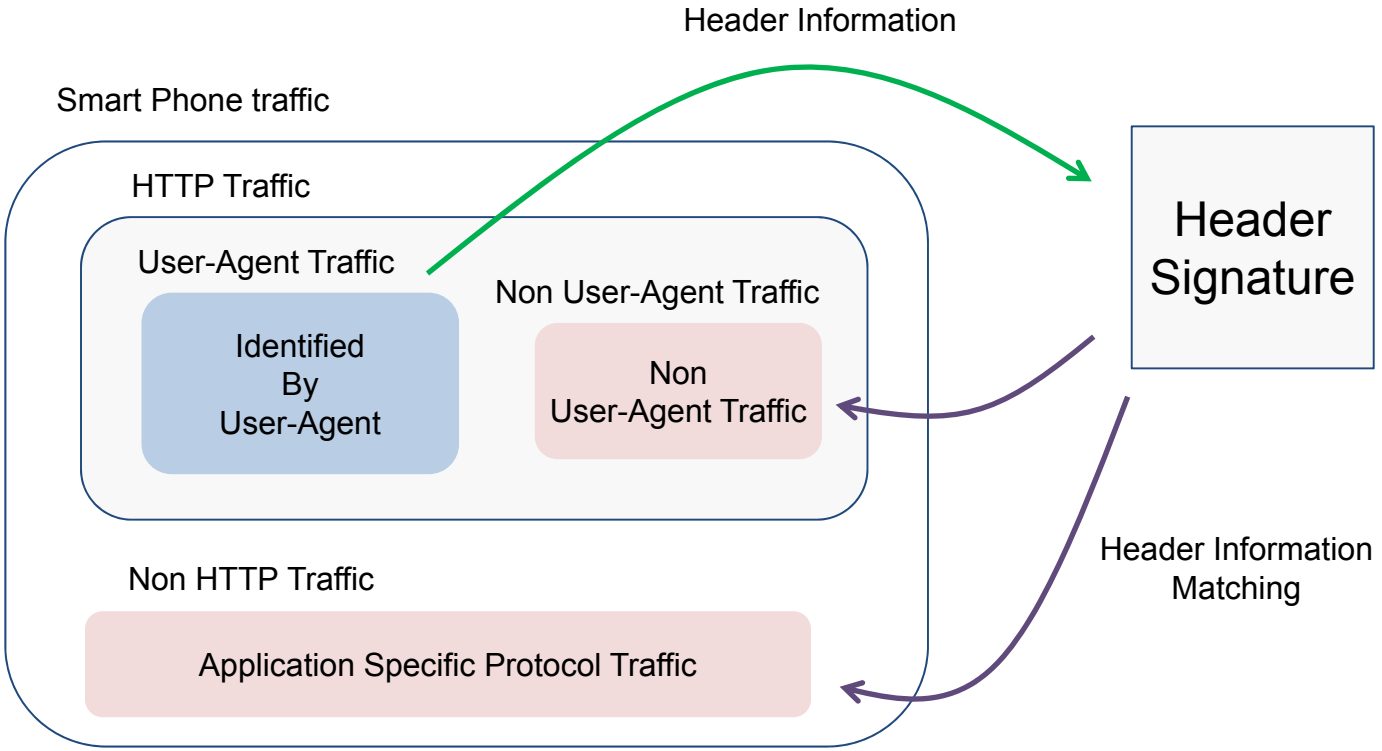
- HTC Streaming Play 응용의 플로우

```
192.168.102.91   : 56203 --   6 -- 93.186.          :     80 [   ]
GET /u33696362/audio/5f1c30e969c2.mp3 HTTP/1.1
User-Agent: Dalvik/1.2.0 (Linux; U; Android 2.2.1; Desire HD Build/FRG83D)
Host: cs5029.vkontakte.ru
Connection: Keep-Alive
```

서비스별 트래픽 분류 방법

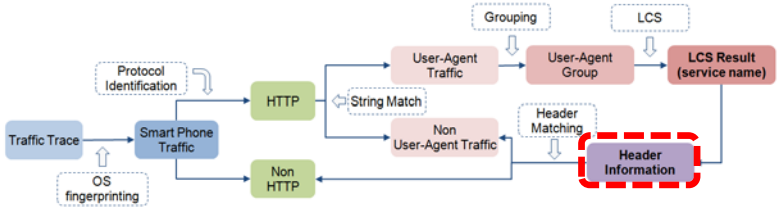
- 헤더정보를 이용한 추가분석
  - User-Agent트래픽에서 헤더정보 추출
    - 추출된 헤더 정보이용
      - Non HTTP, Non User-Agent 분석 가능

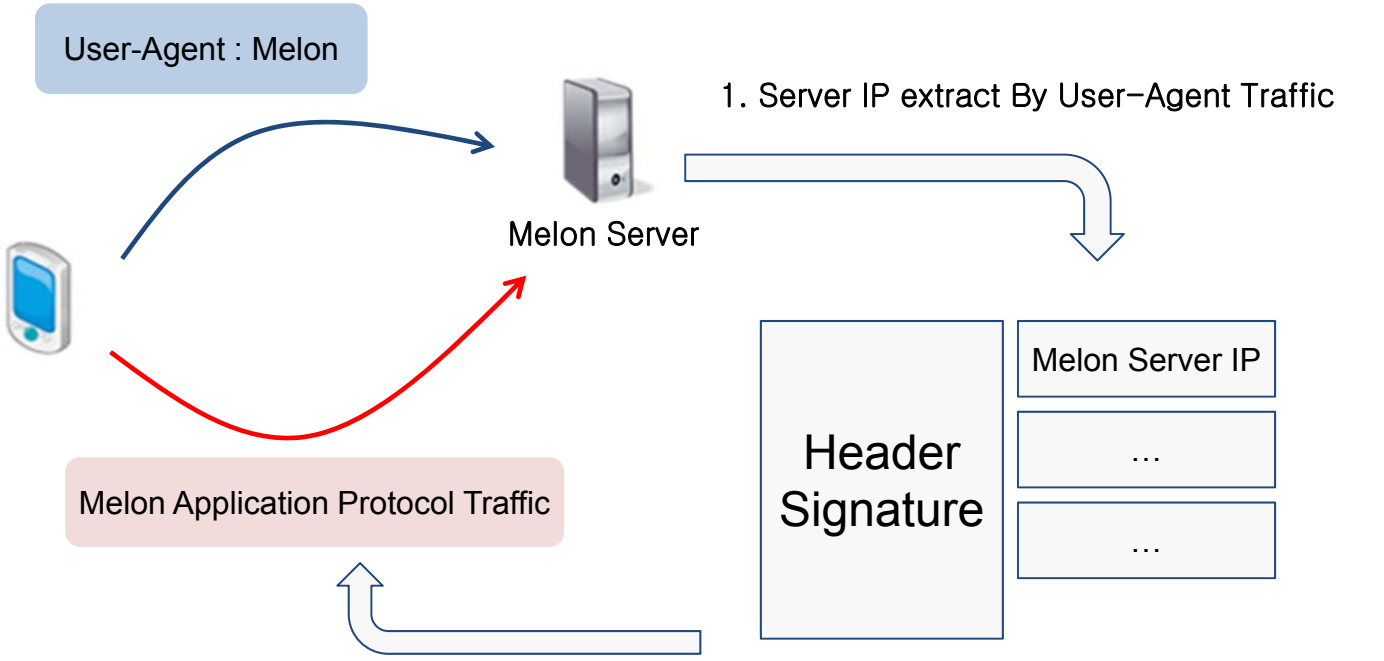




서비스별 트래픽 분류 방법

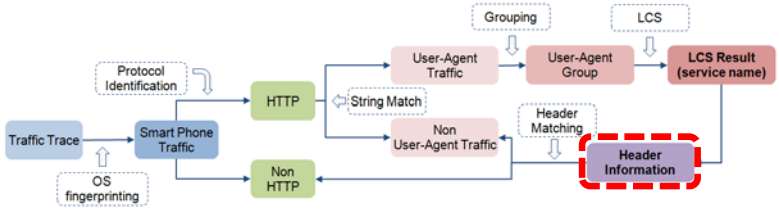
- 헤더정보를 이용한 추가분석(cont.)
  - User-Agent트래픽에서 헤더정보 추출
    - Example
      - Melon : Music Streaming Service Application

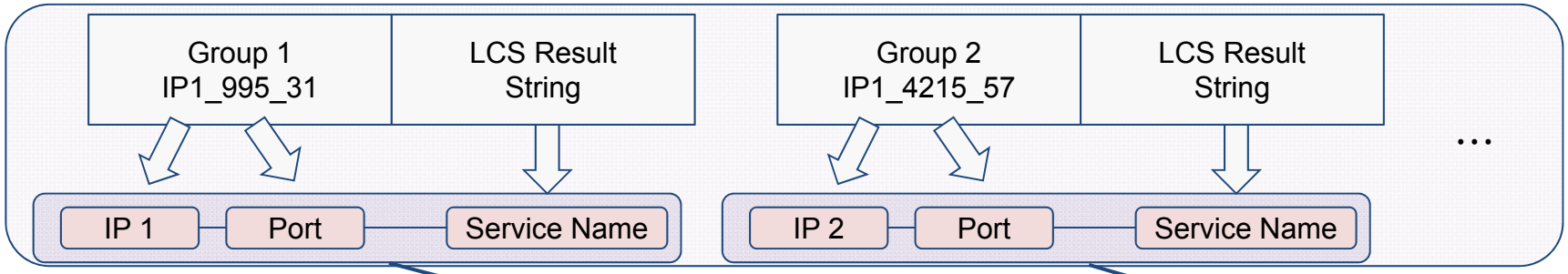




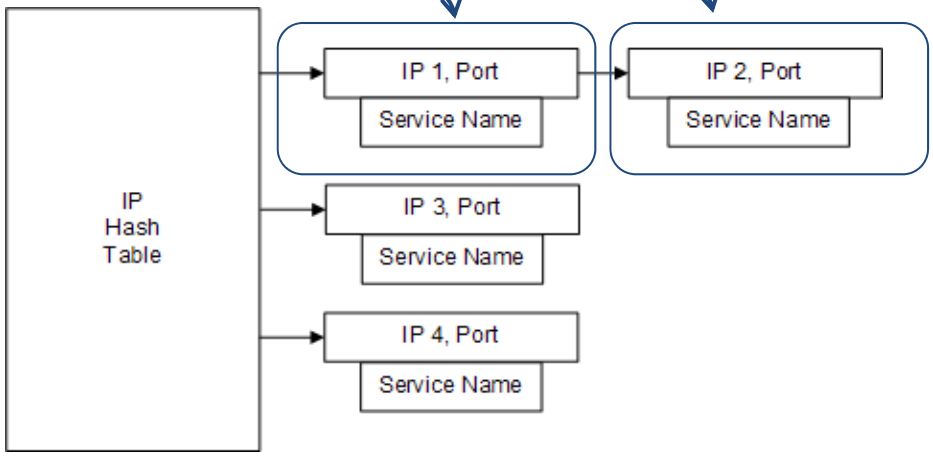
서비스별 트래픽 분류 방법

- 헤더정보를 이용한 추가분석(cont.)
  - 헤더 시그니쳐 구성
    - 생성된 LCS 결과의 각 그룹의 정보를 기반으로 생성



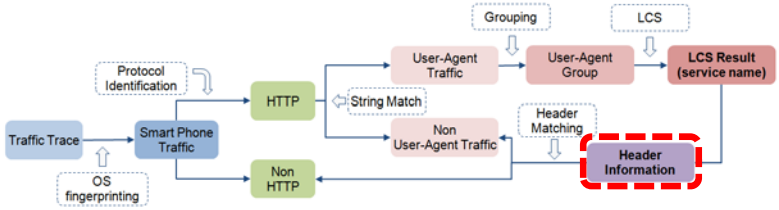


- 해쉬 형태의 접근 구조
  - 링크드 리스트
  - 각 요소
    - » IP
    - » Port
    - » Service Name

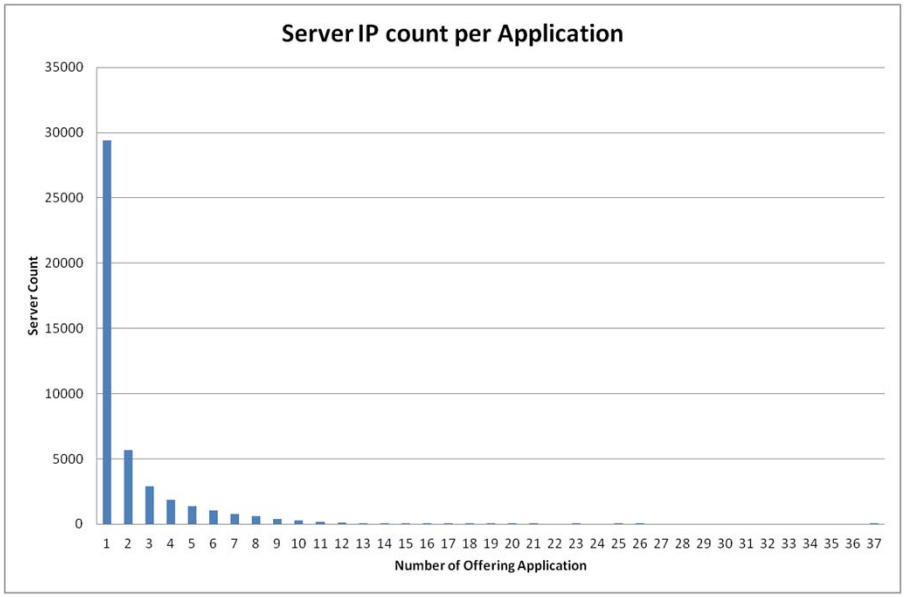


서비스별 트래픽 분류 방법

- 헤더정보를 이용한 추가분석(cont.)
  - 헤더 시그니쳐 구성 결과
    - 총 4만 5천 여개의 헤더 시그니쳐 생성
    - 시그니쳐의 유효성
      - 한 IP에서 하나의 서비스만 제공하는 IP 를 헤더 시그니쳐로 사용

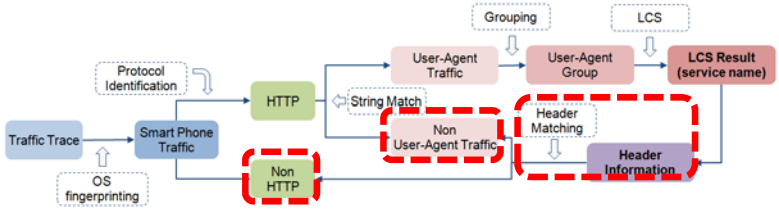


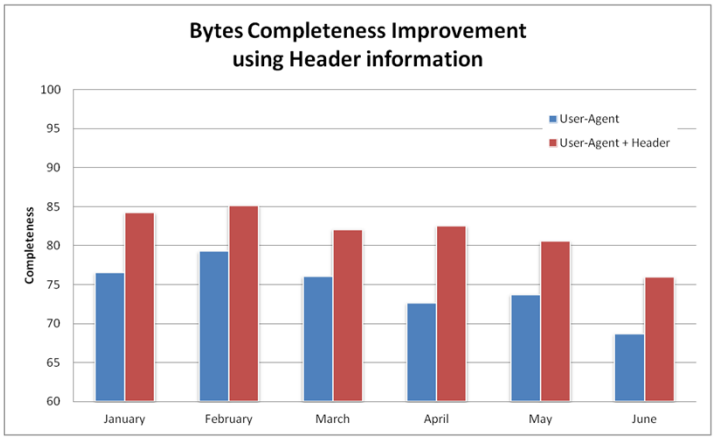
Number of Offering Application	Server IP count	Distribution	Cumulative Distribution
1	**29397**	**65.25**	**65.25**
2	5666	12.57	77.83
3	2919	6.48	84.31
4	1878	4.16	88.48
5	1371	3.04	91.53
6	1054	2.33	93.87
7	801	1.77	95.64
8	610	1.35	97.00
9	420	0.93	97.93
10	274	0.60	98.54
11	188	0.41	98.96
	...		
37	1	0.002	100

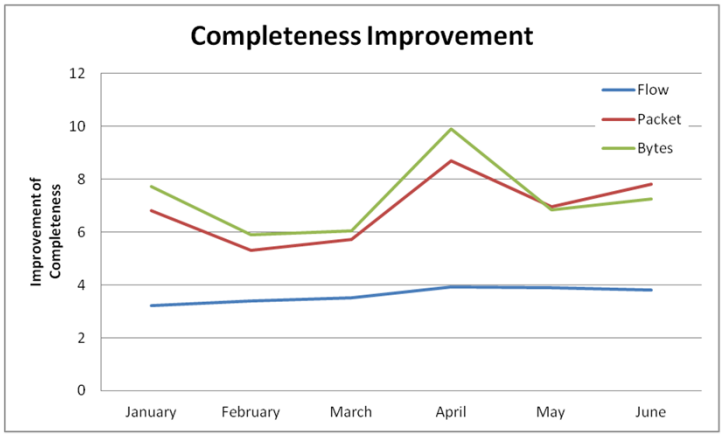


서비스별 트래픽 분류 방법

- 헤더정보를 이용한 추가분석(cont.)
  - 생성된 헤더 시그니쳐의 적용
    - 기존의 User-Agent + 헤더 시그니쳐
      - 평균 바이트 7.3%의 분석률 향상





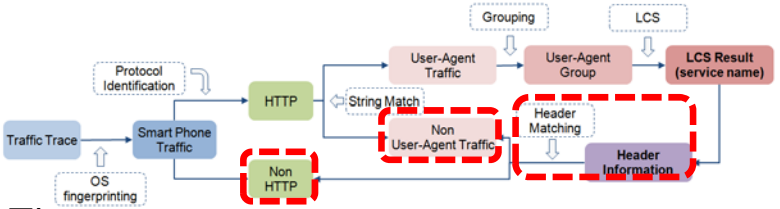


Completeness	Flow	Packet	Bytes
January	79.17 (+3.2)	82.02 (+6.8)	84.23 (+7.71)
February	78.22 (+3.3)	81.58 (+5.3)	85.18 (+5.8)
March	81.07 (+3.4)	80.35 (+5.7)	82.09 (+6)
April	81.52 (+3.9)	80.92 (+8.7)	82.52 (+9.8)
May	83.47 (+3.9)	79.72 (+6.9)	80.55 (+6.8)
June	85.25 (+3.8)	77.29 (+7.8)	75.94 (+7.2)
Average	81.5 (+3.7)	80.3 (+6.9)	81.8 (7.3)

서비스별 트래픽 분류 방법

➢ 헤더정보를 이용한 추가분석(cont.)

■ 추가 분석된 트래픽 형태



◆ User-Agent에 명시되지 않은 형태의 트래픽

– NateVideo

```
################################################################################
163.152.232.121 : 57425 --  6 -- 211.115.        :    80 [ D]
== ip:211.115.11.217  ==          == reg:NateVideo/
GET /OApi/RestApi/CY/200110/wap_pims_diary_mydiary_write_cyphone/v1?folder_
Authorization: OAuth
Host: openapi.nate.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

le-Connection-Type: WiFi
X-Dsid: 219858543
X-Apple-Client-Application: Software
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

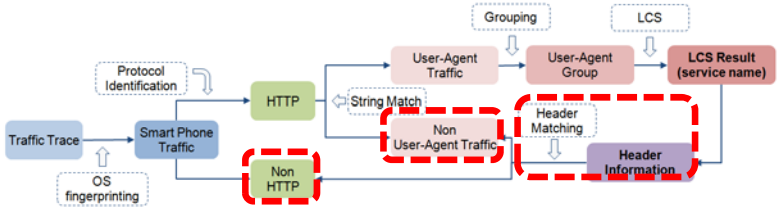
◆ ACK, FIN 패킷만 존재하는 형태의 트래픽

– kakaotalk

```
163.152.234.167 : 57674 --  6 -- 203.246.        :    80 [  ]
                            > 11.28->11.28 [ 0.00sec] [    1p         70b] => [  0][ A F  ] [   ]
                            < 11.28->11.28 [ 0.00sec] [    1p         70b] => [  0][ A    ] [   ]
== ip:203.246.172.34  ==          == reg:kakao.*talk
```

서비스별 트래픽 분류 방법

- 헤더정보를 이용한 추가분석(cont.)
  - 분석하지 못한 트래픽
    - 트레이스내 비정상 트래픽
      - 어떠한 서비스의 트래픽인지 판별 불가
        - » 비정상 탐지 알고리즘에 의해 해결 가능



    - 응용 고유의 트래픽
      - 해당서비스 고유의 프로토콜 분석 필요
        - » 모바일 Torrent

```
163.152.233.179 : 61363 --  6 -- 121.96.146.59   : 44466 [ D]
‼BitTorrent protocol‼BitTorrent protocol
```

    - 새로운 서비스의 트래픽
      - User-Agent를 명시하지 않은 트래픽
        - » HTTP 필드 이외의 다른 필드 조사 필요

실험 및 결과 분석

- 트래픽 발생 순위
  - 바이트 Top 15
    - Heavy Traffic
      - Graphical 처리의 응용
      - Streaming, Application Download, Web, 기타 응용

	User-Agent(Regular Expression)	Flow	Packet	Byte (GB)	Byte Portion
1	Safari	10630677	418372759	**297.19**	27.17
2	iTunes-iPhone	655350	202191638	**208.48**	19.06
3	AppleCoreMedia	571087	156647675	**152.85**	13.97
4	NaverWebtoon	65660	64050415	**69.05**	6.31
5	iTunes-iPod	138100	33675748	**36.16**	3.30
6	Stagefright	9776	22801266	**22.28**	2.03
7	OpenCORE	12032	20740790	**21.13**	1.93
8	NHN iPhoneApp	13747	16261237	**16.15**	1.47
9	Firefox	267076	17535726	**14.68**	1.34
10	AndroidDownloadManager	22992	13631963	**14.17**	1.29
11	Android update	109890	11971080	**11.68**	1.06
12	Smart Daily	12542	7602744	**7.67**	0.70
13	MelOn	2818	7096457	**7.45**	0.68
14	iBooks	27577	5664006	**5.64**	0.51
15	GQ	115427	6107661	**4.81**	0.44

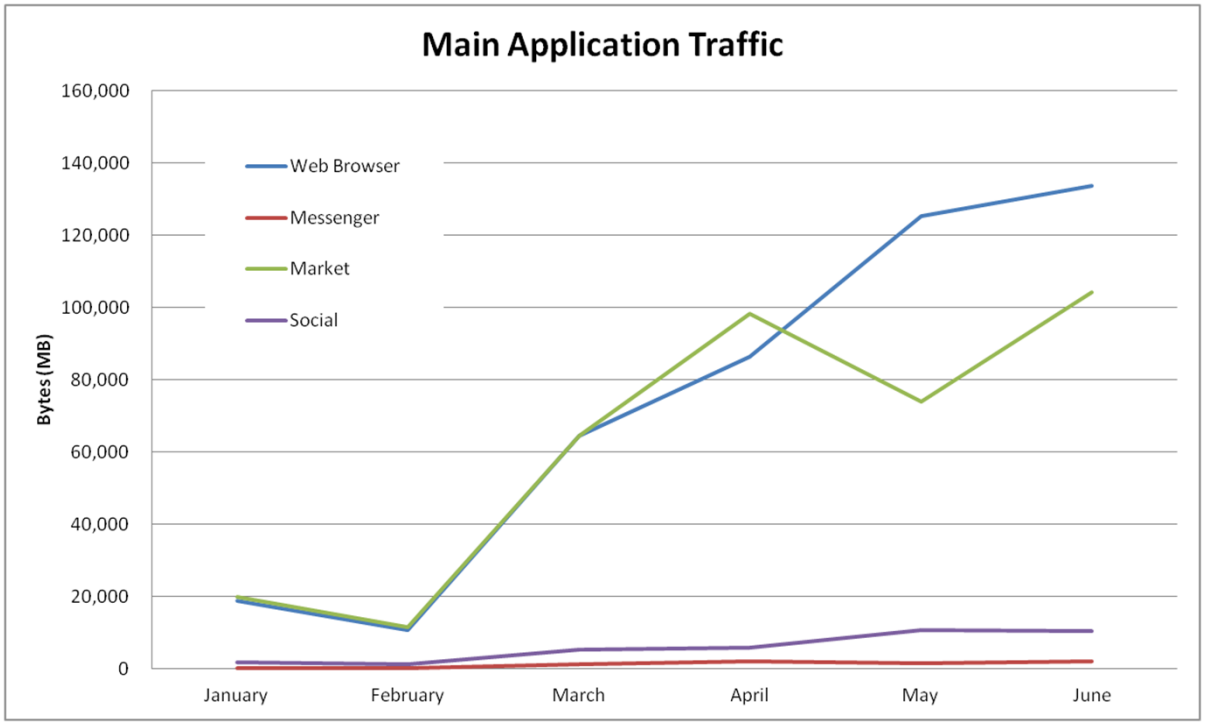
실험 및 결과 분석

- 트래픽 발생 순위(cont.)
  - 플로우 Top 15
    - Heavy Session
      - 단발적 메시지 전송으로 인한 다수의 세션(카카오토크) 순위권 진입
        - » 소셜네트워크 계열의 응용

	User-Agent(Regular Expression)	Flow	Packet	Byte (GB)	Flow Portion
1	Safari	**10630677**	418372759	297.19	49.71
2	CaptiveNetworkSupport	**1196403**	13223980	1.73	6.02
3	iTunes-iPhone	**655350**	202191638	208.48	3.29
4	AppleCoreMedia	**571087**	156647675	152.85	2.87
5	Kakaotalk	**488620**	7353698	2.91	2.46
6	Chrome	**334471**	14663486	11.43	1.68
7	Firefox	**267076**	17535726	14.68	1.34
8	freeapps	**182909**	4244111	2.76	0.92
9	iTunes-iPod	**138100**	33675748	36.16	0.69
10	Wit	**129293**	2436841	1.2	0.65
11	GQ	**115427**	6107661	4.80	0.58
12	Nhnsearch	**96889**	975586	0.12	0.48
13	HiThere	**78168**	1628387	0.85	0.39
14	Installous	**77269**	3659408	3.01	0.38
15	iPhoneMinihompy	**76924**	1220431	0.48	0.38

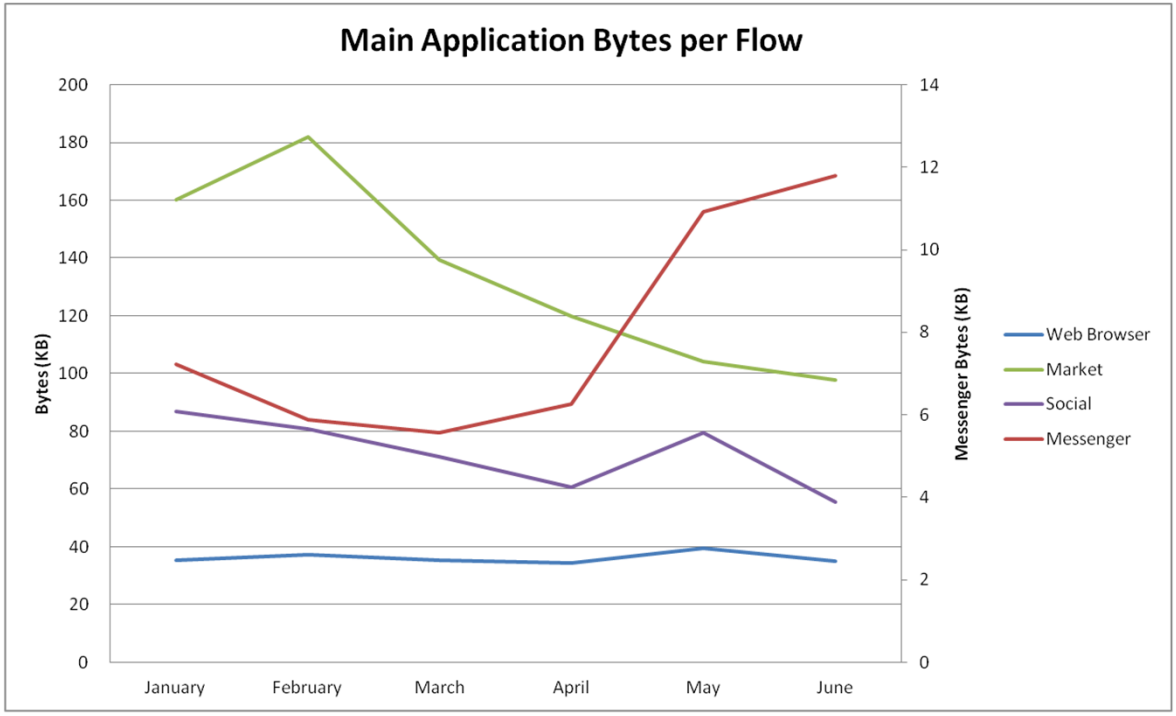
실험 및 결과 분석

- 주요 응용분석
  - Web Browser(safari, opera, skyfire, dorothy)
  - Messenger(kakaotalk, navertalk, daum, ebuddy)
  - Social(twitter, facebook)
  - Market(appstore, android market)



실험 및 결과 분석

- 주요 응용분석(cont.)
  - 플로우당 바이트 발생량
    - Messenger 응용의 플로우당 바이트 증가
      - 해당 응용을 통한 다양한 기능의 사진, 동영상 전송이 원인
    - Market 응용의 플로우당 바이트량 감소
      - 스마트폰 사용의 안정화(새로운 응용의 다운 및 설치 감소)



실험 및 결과 분석

- 주요 응용분석(cont.)
  - 10초당 주요 응용의 호스트 수
    - 스마트폰으로 인해 웹 검색의 대중화, 새로운 응용의 관심
      - web > market > messenger > social

	Total	Web Browser	Messenger	Market	Social
January	2.1	1.5	0.2	0.5	0.1
February	1.6	1.1	0.2	0.3	0.1
March	8.2	5.0	1.4	1.5	0.3
April	11.5	5.3	1.9	4.6	0.5
May	13.1	7.6	1.6	3.1	0.7
June	16.4	8.1	1.9	5.8	0.9
Average	**8.8**	**4.8**	**1.2**	**2.6**	**0.4**

결론 및 향후 연구 – 스마트폰 트래픽 분석

- 결론
  - 스마트폰의 트래픽 특징 분석
    - 주요 트래픽인 HTTP 트래픽의 분포
    - 기존의 트래픽과의 차이점 분석
  - HTTP의 User-Agent를 통한 서비스별 트래픽 분류
    - 스마트폰의 HTTP트래픽 특성을 이용
      - User-Agent 그룹핑
      - LCS알고리즘을 통한 서비스 Identification
      - 헤더 시그니쳐 구성을 통한 분석률 향상
  - 서비스별 트래픽 분석 결과 제시
    - 스마트폰 트래픽의 발생 비율
    - 운영체제별 분포
    - 트래픽 분류 결과 제시

- 향후 연구
  - User-Agent필드 이외의 HTTP필드에서의 시그니쳐 추출 연구
  - 다양한 네트워크에서의 방법론 적용

경청해 주셔서 감사합니다