Automated Application Signature Generation for Traffic Identification

Young J. Won, Seong-Chul Hong, Byung-Chul Park, and James W. Hong

Distributed Processing and Network Management Lab.
Dept. of Computer Science and Engineering
POSTECH, Korea
{yjwon, jwkhong}@postech.ac.kr

Aug. 16, 2008
Outline

- Introduction on DPNM, POSTECH
- Our Experience on Measurement
- Automated Signature Generation
- Conclusion
POSTECH Since 1986

- Founded by POSCO – 2nd largest iron and steel manufacturer in the world
  - 3000 students, 230 faculty members, 800 researchers

- Distributed Processing and Network Management Lab. ([http://dpnm.postech.ac.kr](http://dpnm.postech.ac.kr)) since 1995
  - 6 PhD students, 3 MS students, 1 researcher as of 2008
Recent Industry Projects

Projects Regarding Traffic Measurement & Analysis Only

- **Korea Telecom (KT)**
  - BGP threats & ISP relations (2008~)
  - Bundled service traffic analysis (2007)
  - Application-level traffic classification (2006)
  - High-speed network monitoring system (2005)

- **POSCO**
  - Industrial control networks fault detection & prediction (2008~)

- **Government**
  - CASFI (2008)
  - High-speed traffic monitoring & audit systems (2004~2005)

- **Others**
POSTECH’s Experiences in Traffic Measurement & Analysis

- Traffic Monitoring Systems
- Enterprise Networks
- Mobile Data Networks
- Industrial Control Networks
- IPTV Traffic
Traffic Monitoring Systems

- **MRTG+ (1997)**
  - Extension of MRTG, LIVE visualization of traffic

  - Passive traffic monitoring system (up to 100 Mbps)
  - Distributed architecture

- **NGMON (2002~)**
  - Next Generation Network MONitoring and Analysis System
  - Targeting 1-10 Gbps or higher networks
  - Traffic classification, security attack detection & host analysis
Enterprise Networks

❖ Campus Networks
  ● Characteristics analysis of Internet traffic from the perspective of flows [ComCom ‘06]
  ● Application-level traffic monitoring & analysis [ETRI ‘05]

❖ Korea Internet eXchange (2004)


❖ Analysis Categories
  ● Flow size / duration / packet distribution / size distribution / flash flows / volume pattern / flow occurrence period / port number distribution and more
  ● Flow & Packet-based analysis
  ● Focusing on traffic classification & its applications
Mobile Data Networks

- Investigating the unique and unusual traffic characteristics reflecting the user and data service patterns [PAM ‘07]
  - Previous works are limited to small scale measurement study between the selected end hosts
  - They focused on TCP or performance factors rather than understanding the user behavior and the root cause for such phenomenon
Industrial Control Networks

- **Industrial Control Networks (ICN)?**
  - Robust communications between controlling and controlled devices in a manufacturing environment
    - Building, Factory, and Process Automation
  - Mission critical process & Non-fault tolerable networks
  - Emergence of Industrial Ethernet → Ethernet/IP-based
    - EtherNet/IP, PROFINET, TCnet, Vnet/IP, EPA, RAPIEnet
  - Real-world ICN test bed: POSCO

- **Problems?**
  - The cost of network malfunctioning is severe.
  - ICN fault diagnosis techniques require different standards.
    - due to differences of traffic nature

- **Papers**
  - Traffic characteristics [APNOMS ‘07]
  - Fault detection and analysis system [ComMag ‘08]
IPTV Traffic

- Investigation of combinational traffic models for TPS components
  - Bandwidth demand models, Traffic impact analysis

- Commercial IPTV traffic measurements [ComMag ‘08]
  - End-user IPTV traffic measurements of residential broadband access networks
    - IPTV STB over ADSL, Cable, FTTB, and FTTH

\[\text{Diagram: Throughput - D&P via Broadband Access}\]
\[\text{Diagram: Throughput - D&P vs. Streaming via xDSL}\]
Automated Signature Generation for Traffic Identification
Traffic Classification

- Classification has been done based on: [Szabo '08]
  - Port
  - Signature
  - Connection pattern
  - Statistics
  - Information theory
  - Combined classification method

- Signature-based method often is used as ground truth for validation
  - We focus on obtaining accurate signatures
Motivation

- Desire for obtaining accurate, non-bias, and less time-consuming signatures
  - No systematic approach for signature extraction
  - Avoiding tedious and exhaustive search for signatures
  - Dealing with thousands of applications (e.g., P2P)

- Validation requirements
  - Cross validation with classification algorithms themselves
  - Relying on signature eventually for ground truth

- No concrete set of signatures
  - Proposing a sharing data set for signature list
  - Industry: Ipoque, Sandvine, Procera, and etc.

- An extra question in mind
  - What about encrypted traffic applications?
Related Work

- **POSTECH’s work on classification**
  - Flow Relationship Mapping (FRM) [M.Kim, ‘04]
  - Hybrid approach between flow relations and signature matching [Won ‘06]
  - ML-based attempts - papers in Korean

- **P2P traffic identification using signature**
  - Packet inspection [Gummandi ‘03, Karagiannis ‘04]
  - Protocol analysis [Sen ‘04]
    - Accurate but only for open protocols

- **Automated worm signature generation**
  [Kim ‘04, Singh ’04, Singh ’05]
  - Sliding-window algorithms [Scheirer ’05]
We proposed a **LCS-based Application Signature Extraction** technique - **LASE**

- Longest Common Subsequence algorithm
  - [Cormen ’01]
- Avoiding exhaustive search for signatures
- Extracting candidate signature for later analysis
Constraints of LASER (1/2)

- **Number of packets per flow**
  - A concrete signature exists in the initial few packets of the flow [Sen ’04]
  - Tentative packet grouping

- **Minimum substring length**
  - Signature is simply a sequence of substrings
  - Length of substring reflect the significance as a signature
  - To avoid trivial signatures
    - e.g. ‘/’ in HTTP protocol

- **Packet size**
  - Size differs due to purpose of the packets (signaling or download)
  - Packet size in a close range infers higher chance for valid signatures
Constraints of LASER (2/2)

- Example: LimeWire
  - Signaling - avg. 390 bytes, Downloading - 1460 bytes
  - Avoiding unnecessary packet comparisons
  - Reducing garbage characters from the generated signature
LASER Pseudocode

1: procedure Signature_Generation ()
2:   F1_Pool {F1[], ..., Fk[]} ← Sanitized_packet_collector
3:   F1[] ← Iterate, packet dump for Flow 1
4:   F2[] ← Iterate, packet dump for Flow 2
5:   while i from 0 to #packet_constraint do
6:     while j from 0 to #packet_constraint do
7:       if |F1[i].packet_size - F2[j].packet_size| < threshold
8:         result_LCS ← LASER (F1[i], F2[j])
9:         LCS_Pool [] ← Append result_LCS, end if
10:      j++. end while
11:   i++. end while
12:   S ← select the longest from LCS_Pool
13:   while i from 0 to number of rest flows of Flow_Pool do
14:     F1 ← select one from the rest of Flow_Pool
15:     result_LCS ← LASER (S, F1)
16:     S ← select the longest from result_LCS
17:     i++. end while, end while
18: return S

19: procedure LASER (PacketA[1...m], PacketB[1...n])
20:   PacketA[m...1] ← Reverse byte stream
21:   PacketB[n...1] ← Reverse byte stream
22:   Matrix [m][n]
23:   while i from 0 to m do
24:     while j from 0 to n do
25:       if i = 0 or j = 0, then Matrix[i][j] ← 0
26:       else if PacketA[i] = PacketB[j], then
27:         Matrix[i][j] ← 'Diagonal'
28:       else if Matrix[i][j] = p[i][j-1], then
29:         Matrix[i][j] ← 'Up'
30:       else Matrix[i][j] ← 'Left', end if
31:      end while
32:     i++. end while
33:   if Matrix[0][0] = Left, then j--
34:      if Matrix[0][0] = 'Up', then i--
35:      else if Matrix[0][0] = 'Diagonal', then do
36:        Substring ← Append PacketA[i]
37:        if Matrix[i-1][j-1] = 'Diagonal', then
38:          Substring ← Append special break point character (e.g., '?)
39:        end if
40:      i--; j--; end while
41:      while tokenizing substring based on break point do
42:        if token_length > minimum_substring_length_constraint
43:        then, result_LCS ← Append token_substring, end while
44: return result_LCS
Applying Constraints

- Number of packets per flow constraint
- Packet size constraint
- F1 and F2 are used as input to LASER

```
3:     F1[] ← Iterate, packet dump for Flow 1
4:     F2[] ← Iterate, packet dump for Flow 2
5:     while i from 0 to #_packet_constraint do
6:       while j from 0 to #_packet_constraint do
7:         if |F1[i].packet_size - F2[j].packet_size| < threshold
8:           result_LCS ← LASER (F1[i], F2[j])
```
Refining Process

12: \( S \leftarrow \) select the longest from LCS_Pool
13: \textbf{while} \( i \) from 0 to number of rest flows of Flow_Pool \textbf{do}
14: \( F_i \leftarrow \) select one from the rest of Flow_Pool
15: \( \text{result}_LCS \leftarrow \text{LASER} (S, F_i) \)
16: \( S \leftarrow \) select the longest from result_LCS
17: \( i++ \), \textbf{end while}, \textbf{end while}

\[\text{Simply put,}\]

\begin{align*}
\text{Candidate}\_\text{signature}\_1 &= \text{Signature} (\text{Flow 1, Flow 2}) \\
\text{Candidate}\_\text{signature}\_2 &= \text{Signature} (\text{Flow 3, Candidate}\_\text{signature}\_1) \\
&\ldots \\
\text{Candidate}\_\text{signature}\_n &= \text{Signature} (\text{Flow n+1, Candidate}\_\text{signature}\_n-1) \\
\text{If} \ \text{Candidate}\_\text{signature}\_n &= \text{Candidate}\_\text{signature}\_n-1 \\
\text{For} \ \text{the certain iteration counts} \text{ then} \\
\text{Candidate}\_\text{signature}\_n \text{ is the final signature}
\end{align*}
## Signatures by LASER

<table>
<thead>
<tr>
<th>Application</th>
<th>Sequence of substrings</th>
</tr>
</thead>
<tbody>
<tr>
<td>LimeWire</td>
<td>&quot;LimeWire&quot;, &quot;Content-Type:&quot;, &quot;Content-Length:&quot;, &quot;X-Gnutella-Content-URN&quot;, &quot;run:sha:1&quot;, &quot;XAlt&quot;, &quot;X-Falt&quot;, &quot;X-Create-Time:&quot;, &quot;X-Features:&quot;, &quot;X-Thex-URI&quot;</td>
</tr>
<tr>
<td>BitTorrent</td>
<td>&quot;0x13BitTorrent protocol&quot;</td>
</tr>
<tr>
<td>Fileguri</td>
<td>&quot;HTTP&quot;, &quot;Freechal P2P&quot;, &quot;User-Type:&quot;, &quot;P2PErrorCode:&quot;, &quot;Content-Length:&quot;, &quot;Content-Type:&quot;, &quot;Last-Modified&quot;</td>
</tr>
</tbody>
</table>

- **Choice of P2P applications for early evaluation**
- **Signature extraction from encrypted traffic:**
  - Skype v3.0
    - No signature was found yet
    - The signatures of v1.5 and v2.0 [Ehlert ’06] were not valid anymore
Classification with Absolute Ground Truth

- **Validation approaches**
  - Cross match with known signatures
  - Cross validation with other classification method
  - Cross validation with ground truth set

- **Agent-based log collection**
  - Traffic Measurement Agent (TMA)

- **Choice of Classification Algorithm**

- **Absolute Ground Truth**
  - Text log data with process name, timestamp, IP, port information
  - 100% accuracy
Automated Signature Generation System

**LASER agent**
- Signature extraction of on-going application in PC
- Reporting to the collecting server periodically
- MSDN functions for process id and name look up
- Winpcap for packet dump
- Low CPU load (<5%) and memory consumption

**Collection server**
- Aggregating signatures according to process name
- **Filtering process** – Applying the LASER algorithm among the collected signatures
  - Removing garbage characters/terms
  - Finding common set among possible candidates

**Open Signature List:** [http://dpnm.postech.ac.kr/signature](http://dpnm.postech.ac.kr/signature)
- LASER agent program is available.
- Providing over 80 pre-searched signatures by exhaustive search and in related literatures
- Providing a list of automatically generated signatures for comparison
Concluding Remarks

- **We have shown**
  - POSTECH’s efforts on traffic monitoring and analysis
  - Automated signature generation algorithm

- **We propose a open repository for signatures**

- **Future Work**
  - Automated rule discovery system
    - Containing not just signatures, but pattern information
  - A new approach to cope with encryption or tunneling traffic
  - Signatures for WiMAX applications (Wibro in Pohang)
  - Certifying signatures
**Ground Truth vs. LASER**

<table>
<thead>
<tr>
<th>Application</th>
<th>TMA Log (MB)</th>
<th>Classification Result (MB)</th>
<th>False Negative (%)</th>
<th>False Positive (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LimeWire</td>
<td>1223.36</td>
<td>1120.35</td>
<td>8.42</td>
<td>0</td>
</tr>
<tr>
<td>BitTorrent</td>
<td>4190.07</td>
<td>3754.30</td>
<td>10.40</td>
<td>0</td>
</tr>
<tr>
<td>Fileguri</td>
<td>3189.61</td>
<td>3177.17</td>
<td>0.39</td>
<td>0</td>
</tr>
<tr>
<td>Others</td>
<td>12482.69</td>
<td>13033.91</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Accuracy</strong></td>
<td></td>
<td>97.39 %</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Accuracy analysis against signature-based classification algorithms**
  - LASER algorithm achieves 97% accuracy

- **0% FP: Restricted signature format**
  - HTTP traffic was not classified as LimeWire or Fileguri
  - Cause of FN: HTTP traffic, packets containing flags only
Automated Signature Generation for Traffic Identification

Traditionally, Internet applications have been identified by using predefined well-known ports with questionable accuracy. An alternative approach, application-layer signature mapping, involves the exhaustive search of reliable signatures but with more promising accuracy. With a prior protocol knowledge, the signature generation can guarantee a high accuracy. As more applications use proprietary protocols, it becomes increasingly difficult to obtain an accurate signature while avoiding time-consuming and manual signature generation processes.

We propose a LCS-based (Longest common subsequence) Application Signature Extraction algorithm (LASER), which can automatically determine a trustworthy pattern in the packet’s payload without a prior knowledge of protocol formats. Although there have been a few research works on worm signature generation, it is difficult to adopt the popular sliding window algorithm that has been applied to worm signature generation due to the differences in traffic nature between autonomous network-based applications and worms. To our knowledge, no other research has attempted to automatically generate signatures for non-threatening Internet applications.

REFERENCES

### Automated Signature Generation for Traffic Identification

<table>
<thead>
<tr>
<th>Process</th>
<th>In/Out</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>MsnMsg.exe</td>
<td>in</td>
<td>2008-08-08 02:24:23</td>
</tr>
<tr>
<td>NATEONMain.exe</td>
<td>in</td>
<td>2008-08-07 17:39:34</td>
</tr>
<tr>
<td>Outlook.exe</td>
<td>in</td>
<td>2008-03-03 19:12:08</td>
</tr>
<tr>
<td>Skype.exe</td>
<td>in</td>
<td>2008-08-09 02:51:20</td>
</tr>
<tr>
<td>Tor.exe</td>
<td>in</td>
<td>2008-08-12 11:18:49</td>
</tr>
<tr>
<td>Wow.exe</td>
<td>in</td>
<td>2008-08-11 04:05:22</td>
</tr>
<tr>
<td>Zulu.exe</td>
<td>in</td>
<td>2008-08-07 00:37:10</td>
</tr>
</tbody>
</table>
Automated Signature Generation for Traffic Identification

* Signatures from Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amarena</td>
<td>&quot;POST rpc/config HTTP/&lt;version&gt;&quot; &quot;User-Agent Amarena&quot; &quot;Host:</td>
</tr>
<tr>
<td>GigaTribe</td>
<td>&quot;GET&quot; ≤ &amp;reqd=OpenSession&quot; HTTP/1.1&quot; &quot;User-Agent GigaTribe&quot; HTTP/1.1&quot; 200 OK</td>
</tr>
<tr>
<td>Zbuff</td>
<td>&quot;ZEPP 19.29 (port)&quot;-offset(0) 0x00000000a, ZEPP OK (number 12,28,29) (my IP address)offset(0) 0x00000000a</td>
</tr>
<tr>
<td>Bitlord</td>
<td>&quot;GET HTTP&quot; User-Agent:BitTorrent&quot; <a href="http://www.bitlord.com">www.bitlord.com</a></td>
</tr>
<tr>
<td>DC++</td>
<td>&quot;HTTP&quot; &quot;User-Agent:DC++&quot;</td>
</tr>
<tr>
<td>Tor</td>
<td>&quot;GET to server&quot; &quot;Gettor&quot; status</td>
</tr>
<tr>
<td>Gtalk</td>
<td>streamto:=&quot;gmail.com&quot; username=&quot;jabber&quot; client</td>
</tr>
<tr>
<td>Auto2P2P</td>
<td>&quot;NOTIFY * HTTP&quot; USN: mail:Auto2P2P</td>
</tr>
<tr>
<td>KCEasy</td>
<td>&quot;GET HTTP&quot; offset(0) cookie:KCEasy</td>
</tr>
<tr>
<td>Linewire</td>
<td>&quot;GET&quot; &quot;User-Agent: LineWire&quot; &quot; Java &quot;</td>
</tr>
<tr>
<td>Stealth</td>
<td>&quot;POST value&quot; HTTP/1.1&quot;</td>
</tr>
<tr>
<td>TrustShare</td>
<td>&quot;LARS REGENSBERGER'S FILE SHARING PROTOCOL 0.2&quot; offset(0)</td>
</tr>
<tr>
<td>iMesh</td>
<td>&quot;POST offset(0) function=login&quot; Host: login.iMesh.com</td>
</tr>
<tr>
<td>Mate</td>
<td>client=MUTE&quot;version=offset(12)</td>
</tr>
<tr>
<td>Soulseek</td>
<td>&quot;GET&quot; offset(0) &quot;User-Agent: SoulSeek&quot;</td>
</tr>
<tr>
<td>Skype</td>
<td>&quot;GET&quot; offset(0) &quot;HTTP&quot; &quot;User-Agent: skype&quot;</td>
</tr>
</tbody>
</table>

* Signatures from Snort P2P

<table>
<thead>
<tr>
<th>Application/Type:</th>
<th>Content</th>
<th>Offset</th>
<th>Depth</th>
<th>Distance</th>
<th>Within</th>
<th>Direction</th>
<th>Home Port</th>
<th>External Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2P master login</td>
<td>&quot;00 02 00&quot;</td>
<td>1</td>
<td>3</td>
<td>out</td>
<td>any</td>
<td>any</td>
<td>SSHSS</td>
<td>any</td>
</tr>
<tr>
<td>P2P master new user login</td>
<td>&quot;00 06 00&quot;</td>
<td>1</td>
<td>3</td>
<td>out</td>
<td>any</td>
<td>any</td>
<td>$SSHSS</td>
<td>any</td>
</tr>
<tr>
<td>P2P master download attempt</td>
<td>&quot;00 CB 00&quot;</td>
<td>1</td>
<td>3</td>
<td>in</td>
<td>any</td>
<td>any</td>
<td>$SSHSS</td>
<td>any</td>
</tr>
</tbody>
</table>