Network monitoring in high-speed links

Algorithms and challenges

Pere Barlet-Ros

Advanced Broadband Communications Center (CCABA)
Universitat Politècnica de Catalunya (UPC)
http://monitoring.ccaba.upc.edu
pbarlet@ac.upc.edu

NGI course, 30 Nov. 2010
Outline

1. Network monitoring
2. Addressing the technological barriers
3. Use cases
4. Addressing the social barriers
Outline

1. Network monitoring
   - Introduction
   - Active monitoring
   - Passive monitoring
   - Technological and social barriers

2. Addressing the technological barriers

3. Use cases

4. Addressing the social barriers
Introduction to network monitoring

- Process of measuring network systems and traffic
  - Routers, switches, servers, ... 
  - Network traffic volume, type, topology, ... 

- Monitoring is crucial for network operation and management
  - Traffic engineering, capacity planning, BW management 
  - Fault diagnosis, troubleshooting, performance evaluation 
  - Accounting, billing, security, ... 

- Network measurements are important for networking research
  - Design and evaluation of protocols, applications, ... 
  - Traffic modeling and characterization 

- Network monitoring is very challenging and datasets scarce
  - From technological and “social” standpoint
Classification of network monitoring tools and methods

- Hardware vs. software
- Online vs. offline
- LAN vs. WAN
- Protocol level
- **Active vs. passive**
Active monitoring

- Active tools are based on traffic injection
  - Probe traffic generated by a measurement device
  - Response to probe traffic is measured

- Pros: Flexibility
  - Devices can be deployed at the edge (e.g., end-hosts)
  - No instrumentation at the core is needed
  - Measurement does not directly rely on existing traffic

- Cons: Intrusiveness
  - Probe traffic can degrade network performance
  - Probe traffic can impact on the measurement itself

- Main usages
  - Performance evaluation (e.g., ping)
  - Bandwidth estimation (e.g., pathload)
  - Topology discovery (e.g., traceroute)
Passive monitoring

- Traffic collection from inside the network
  - Routers and switches (e.g., Cisco NetFlow)
  - Passive devices (e.g., libpcap, DAG cards, optical taps)

- Pros: Transparency
  - Network performance is not affected
  - No additional traffic is injected
  - Useful even with a single measurement point

- Cons: Complexity
  - Requires administrative access to network devices
  - Requires explicit presence of traffic under study
  - Online collection and analysis is hard (e.g., sampling)
  - Privacy concerns

- Multiple and diverse usages
  - Traffic analysis and classification, . . .
  - Anomaly and intrusion detection, . . .
Technological and social barriers

- **Datasets and platforms for research purposes are limited**
  - Outdated datasets
  - Academic traffic only
  - Anonymized and without payloads

- **Technological barriers**
  - Internet was designed without monitoring in mind
  - Collection of Gb/s and storage of TB/day
  - Links speeds increase at faster pace than processing speeds
  - Building monitoring apps is error-prone and time-consuming

- **Social barriers**
  - Lack of coordination between projects
  - ISPs have no incentive to share information
  - Privacy and competition concerns
  - Monitoring hardware is expensive and difficult to manage
Outline

1. Network monitoring

2. Addressing the technological barriers
   - Bloom filters
   - Bitmap algorithms
   - Direct bitmaps and variants
   - Bitmaps over sliding windows

3. Use cases

4. Addressing the social barriers
Technological challenges

- Few $ns$ per packet
  - Interarrivals $8ns$ (40Gb/s), $32ns$ (10Gb/s)
  - Memory access times $< 10ns$ (SRAM), tens of $ns$ (DRAM)

- Obtaining simple metrics becomes extremely challenging
  - Approaches based on hash tables do not scale
  - Core of most monitoring algorithms
  - E.g., Active flows, flow size distribution, heavy hitter detection, delay, entropy, sophisticated sampling, ...

- Probabilistic approach: trade accuracy for speed
  - Extremely efficient compared to compute exact answer
  - Fit in SRAM, 1 access/pkt
  - Probabilistic guarantees (bounded error)
Bloom filters¹

- Space-efficient data structure to test set membership
  - Based on hashing (e.g., pseudo-random hash functions)

- Examples of usage in network monitoring
  - Replace hash tables to check if a flow has already been seen
  - Definition of flow is flexible

- Advantages
  - Small memory (SRAM) is needed compared to hash tables

- Limitations
  - False positives are possible
  - Removals are not possible (counting variants can support them)

---

Bloom filters

- Parameters
  - $k$: #hash functions
  - $m$: size of the bitmap
  - $p$: false positive rate
  - $n$: #elements in the filter (max)

Figure: Example of a bloom filter\(^2\)

---

Direct bitmaps (linear counting)\(^3\)

- Space-efficient algorithms to count the number of unique items
  - E.g., useful to count the number of flows over a fixed time interval

- Basic idea
  - Each flow hashes to one position (and all its packets)
  - Counting the number of 1’s is inaccurate due to collisions
  - Count the number of unset positions instead
  - E.g., 20KB to count 1M flows with 1% error

- Estimate formulae
  - Flow hashes to a given bit: \( p = 1/b \)
  - No flow hashes to a given bit: \( p_z = (1 - p)^n \approx (1/e)^{n/b} \)
  - Expected non-set bits: \( E[z] = bp_z \approx b(1/e)^{n/b} \)
  - Estimated number of flows: \( \hat{n} = b \ln(b/z) \)

---

Bitmap variants

- Direct bitmaps scale linearly with the number of flows
- Variants: Virtual, multiresolution, adaptive, triggered bitmaps, ...

---

Bitmaps over sliding windows

- **Timestamp Vector (TSV)**\(^5\)
  - Vector of timestamps (instead of bits)
  - \(O(n)\) query cost

- **Countdown Vector (CDV)**\(^6\)
  - Vector of small timeout counters (instead of full timestamps)
  - Independent query and update processes
  - \(O(1)\) query cost

---


CDV and TSV performance

- 30-min trace, 271 Mbps, 1 query/s, 50K/10s-1.8M/min flows\(^7\)

\(^7\) A hash table would require several MBs with these settings
Outline

1. Network monitoring
2. Addressing the technological barriers
3. Use cases
   - Load shedding
   - Lossy difference aggregator
4. Addressing the social barriers
Overload problem

- Previous solutions focus on a particular metric
  - They are not valid for any (arbitrary) monitoring application

- Monitoring systems are prone to dramatic overload situations
  - Link speeds, anomalous traffic, bursty traffic nature . . .
  - Complexity of traffic analysis methods

- Overload situations lead to uncontrolled packet loss
  - Severe and unpredictable impact on the accuracy of applications
  - . . . when results are most valuable!!
Overload problem

- Previous solutions focus on a particular metric
  - They are not valid for any (arbitrary) monitoring application

- Monitoring systems are prone to dramatic overload situations
  - Link speeds, anomalous traffic, bursty traffic nature . . .
  - Complexity of traffic analysis methods

- Overload situations lead to uncontrolled packet loss
  - Severe and unpredictable impact on the accuracy of applications
  - . . . when results are most valuable!!

Load Shedding Scheme

- Efficiently handle extreme overload situations
- Over-provisioning is not feasible
Case study: Intel CoMo

- CoMo (Continuous Monitoring)\(^8\)
  - Open-source passive monitoring system
  - Framework to develop and execute network monitoring applications
  - Open (shared) network monitoring platform

- Traffic queries are defined as *plug-in* modules written in C
  - Contain complex computations

\(^8\) http://como.sourceforge.net
Case study: Intel CoMo

- CoMo (Continuous Monitoring)\textsuperscript{8}
  - Open-source passive monitoring system
  - Framework to develop and execute network monitoring applications
  - Open (shared) network monitoring platform

- Traffic queries are defined as \textit{plug-in} modules written in C
  - Contain complex computations

Traffic queries are \textbf{black boxes}

- Arbitrary computations and data structures
- Load shedding cannot use knowledge of the queries

\textsuperscript{8} http://como.sourceforge.net
Load shedding approach\textsuperscript{9}

**Working scenario**
- Monitoring system supporting multiple arbitrary queries
- Single resource: CPU cycles

**Approach:** Real-time modeling of the queries’ CPU usage

1. **Find correlation between traffic features and CPU usage**
   - Features are query agnostic with deterministic worst case cost
2. Exploit the correlation to predict CPU load
3. Use the prediction to decide the sampling rate

---

System overview

Figure: Prediction and Load Shedding Subsystem
Traffic features vs CPU usage

Figure: CPU usage compared to the number of packets, bytes and flows
Traffic features vs CPU usage

Figure: CPU usage versus the number of packets and flows
### Prediction methodology\(^{10}\)

#### Multiple Linear Regression (MLR)

\[
Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_p X_{pi} + \varepsilon_i, \quad i = 1, 2, \ldots, n.
\]

- **\(Y_i\)** = \(n\) observations of the response variable (measured cycles)
- **\(X_{ji}\)** = \(n\) observations of the \(p\) predictors (traffic features)
- **\(\beta_j\)** = \(p\) regression coefficients (unknown parameters to estimate)
- **\(\varepsilon_i\)** = \(n\) residuals (OLS minimizes SSE)

---

Prediction methodology\textsuperscript{10}

\textbf{Multiple Linear Regression (MLR)}

\[ Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_p X_{pi} + \epsilon_i, \quad i = 1, 2, \ldots, n. \]

- \( Y_i \) = \( n \) observations of the response variable (measured cycles)
- \( X_{ji} \) = \( n \) observations of the \( p \) predictors (traffic features)
- \( \beta_j \) = \( p \) regression coefficients (unknown parameters to estimate)
- \( \epsilon_i \) = \( n \) residuals (OLS minimizes SSE)

\textbf{Feature selection}

- Variant of the Fast Correlation-Based Filter (FCBF)
- Removes \textit{irrelevant} and \textit{redundant} predictors
- Reduces significantly the cost and improves accuracy of the MLR

Load shedding scheme\textsuperscript{11}

Prediction and Load Shedding subsystem

1. Each 100\textit{ms} of traffic is grouped into a \textit{batch} of packets
2. The traffic features are efficiently extracted from the batch (multi-resolution bitmaps)
3. The most relevant features are selected (using FCBF) to be used by the MLR
4. MLR predicts the CPU cycles required by each query to run
5. Load shedding is performed to discard a portion of the batch
6. CPU usage is measured (using TSC) and fed back to the prediction system

Results: Load shedding performance

Figure: Stacked CPU usage (Predictive Load Shedding)
Results: Load shedding performance

Figure: CDF of the CPU usage per batch
Results: Packet loss

(a) Original CoMo
(b) Reactive Load Shedding
(c) Predictive Load Shedding

Figure: Link load and packet drops
## Results: Error of the queries

<table>
<thead>
<tr>
<th>Query</th>
<th>original</th>
<th>reactive</th>
<th>predictive</th>
</tr>
</thead>
<tbody>
<tr>
<td>application (pkts)</td>
<td>55.38% ±11.80</td>
<td>10.61% ±7.78</td>
<td>1.03% ±0.65</td>
</tr>
<tr>
<td>application (bytes)</td>
<td>55.39% ±11.80</td>
<td>11.90% ±8.22</td>
<td>1.17% ±0.76</td>
</tr>
<tr>
<td>counter (pkts)</td>
<td>55.03% ±11.45</td>
<td>9.71% ±8.41</td>
<td>0.54% ±0.50</td>
</tr>
<tr>
<td>counter (bytes)</td>
<td>55.06% ±11.45</td>
<td>10.24% ±8.39</td>
<td>0.66% ±0.60</td>
</tr>
<tr>
<td>flows</td>
<td>38.48% ±902.13</td>
<td>12.46% ±7.28</td>
<td>2.88% ±3.34</td>
</tr>
<tr>
<td>high-watermark</td>
<td>8.68% ±8.13</td>
<td>8.94% ±9.46</td>
<td>2.19% ±2.30</td>
</tr>
<tr>
<td>top-k destinations</td>
<td>21.63 ±31.94</td>
<td>41.86 ±44.64</td>
<td>1.41 ±3.32</td>
</tr>
</tbody>
</table>
One-way delay measurement

- Traditional approaches are expensive
  - Probing traffic (intrusive)
  - Trajectory sampling
  - Constant overhead per sample

- Alternative: LDA (Lossy Difference Aggregator)$^{12}$
  - Send only sums of timestamps
  - Deal with packet loss (sampling + partition input stream)

---

Lossy Difference Aggregator (LDA)

1,2,...,5

(delay = 1)

2,3,...,6

network
Lossy Difference Aggregator (LDA)

1, 2, ..., 5
a

(network delay = 1)

2, 3, ..., 6
b

\[
\frac{(2-1)+(3-2)+\ldots}{n} = 1
\]

(send \(n\) timestamps)
Lossy Difference Aggregator (LDA)

\[
\frac{(2-1) + (3-2) + \ldots}{n} = 1 \quad \text{(send } n \text{ timestamps)}
\]

\[
\frac{(2+3+\ldots)-(1+2+\ldots)}{4} = \frac{\sum t_b - \sum t_a}{n} = 1 \quad \text{(send } \text{just one } \sum t_s\text{)}
\]
Lossy Difference Aggregator (LDA)

Network monitoring

Addressing the technological barriers

Use cases

Addressing the social barriers

- Use cases

1, 2, ..., 5

(delay = 1)

2, 3, ..., 6

15 5

20 5
Lossy Difference Aggregator (LDA)

1, 2, ..., 5

(delay = 1)

2, 3, ..., 6

\( \sum_{1}^{5} \quad \# 
\)

\( \sum_{1}^{5} \quad \# 
\)

\( \sum_{1}^{5} \quad \#
\)

\( \sum_{1}^{5} \quad \#
\)

result: \( \frac{20 - 15}{5} = 1 \)
Lossy Difference Aggregator (LDA)

packet loss: mismatch in number of packets!
must protect against packet loss
Lossy Difference Aggregator (LDA)

\[ \begin{array}{c|c}
\sum & \# \\
\hline
1 & 1 \\
2 & 1 \\
7 & 2 \\
5 & 1 \\
\end{array} \]

\[ \begin{array}{c|c}
\sum & \# \\
\hline
2 & 1 \\
3 & 1 \\
9 & 2 \\
6 & 1 \\
\end{array} \]

total: \(< 15, 5 >\)

total: \(< 20, 5 >\)

\[ \frac{20 - 15}{5} = 1 \]
Lossy Difference Aggregator (LDA)

1,2,\ldots,5 \quad \text{(delay = 1)} \quad 2,\ldots,6

\begin{align*}
\sum &\quad \# \\
1 &\quad 1 \quad \checkmark \\
2 &\quad 1 \quad \times \\
7 &\quad 2 \quad \checkmark \\
5 &\quad 1 \quad \checkmark \\
\end{align*}

\text{total: } < 13, 4 >

\begin{align*}
\sum &\quad \# \\
2 &\quad 1 \quad \checkmark \\
0 &\quad 0 \quad \times \\
9 &\quad 2 \quad \checkmark \\
6 &\quad 1 \quad \checkmark \\
\end{align*}

\text{total: } < 17, 4 >

\frac{17-13}{4} = 1 \quad \checkmark
Lossy Difference Aggregator (LDA)

1,2,…,5 (delay = 1)

2,3,…,6

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
Dimensioning the packet sampling rate\textsuperscript{13}

- PSR presents a classical tradeoff. The higher,
  - the more packets it aggregates, but . . .
  - . . . the higher the chance of counter invalidation

- We wish to maximize the exp. effective sample size:

\[
E[S] = \frac{(1 - r) pn}{e^{n rp/b}} \quad \text{which is max. with} \quad p = \frac{b}{nr}
\]

- Original analysis obtained \( p \approx 0.5 \frac{b}{nr} \)
  - our analysis doubles the sampling rate
  - we improve sample size by \( \approx 20\% \)

- Best algorithm in terms of network overhead up to \( \approx 25\% \) loss

\textsuperscript{13} J. Sanjuàs-Cuxart \textit{et al.} Validation and improvement of the Lossy Difference Aggregator to measure packet delays. TMA, 2010.
Outline

1. Network monitoring
2. Addressing the technological barriers
3. Use cases
4. Addressing the social barriers
   - CoMo-UPC initiative
The problem

- Several researchers working on:
  - Anomaly Detection (AD)
  - Traffic Classification (TC)

- Real packet traces are needed to test novel AD and TC methods

- Several AD and TC algorithms require:
  - Unanonymized IP addresses (or prefix-preserving)
  - Payload inspection (e.g., IDS, DPI, ...)

- Traditional solution: Anonymized traffic traces
  - Examples: NLANR, CAIDA, CRAWDAD, ...
The problem

- Several researchers working on:
  - Anomaly Detection (AD)
  - Traffic Classification (TC)

- Real packet traces are needed to test novel AD and TC methods

- Several AD and TC algorithms require:
  - Unanonymized IP addresses (or prefix-preserving)
  - Payload inspection (e.g., IDS, DPI, ...)

- Traditional solution: Anonymized traffic traces
  - Examples: NLANR, CAIDA, CRAWDAD, ...

Anonymization is not the right solution!

- Data owners: Privacy concerns
- Researchers: Not enough data
- Lack of recent publicly available traces
CoMo-UPC

- Move the code to the data
  - Instead of publishing anonymized data traces

- Significantly lowers the privacy concerns
  - Traffic data do not leave provider premises
  - Data providers keep the ownership of the data
  - The source code can be inspected by the data owner

- Researchers have (blind) access to unanonymized traffic
  - IP addresses and payloads can be processed...
  - ... but not stored or exported

- The CoMo system is based on this model
  - Implement AD and TC methods as CoMo modules

http://monitoring.ccaba.upc.edu/como-upc
UPC traffic

1 GE full-duplex (≈ 900 Mbps, 60K flows/s)

Connects 40 departments and 25 faculties (10 campuses)

12 traffic traces are also available