Packet-Level Analytics in Software without Compromises

HotCloud ’18, July 9th, 2018, Boston, MA

Oliver Michel
John Sonchack
Eric Keller
Jonathan M. Smith
Network monitoring is important

- Security issues
- Performance issues
- Equipment failure
- Misconfiguration
Challenging environment

- more traffic
- more threats
- encrypted traffic

**Total Ransomware Samples**

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Collected Samples [M]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q4 2015</td>
<td>4.4</td>
</tr>
<tr>
<td>Q1 2016</td>
<td>6</td>
</tr>
<tr>
<td>Q2 2016</td>
<td>7.8</td>
</tr>
<tr>
<td>Q3 2016</td>
<td>8.4</td>
</tr>
<tr>
<td>Q4 2016</td>
<td>8.9</td>
</tr>
<tr>
<td>Q1 2017</td>
<td>9.5</td>
</tr>
<tr>
<td>Q2 2017</td>
<td>10.8</td>
</tr>
<tr>
<td>Q3 2017</td>
<td>12.1</td>
</tr>
</tbody>
</table>

[McAfee Labs Thread Report Dec. 2017]

**Fraction of encrypted HTTP traffic in Google Chrome**

<table>
<thead>
<tr>
<th>Date</th>
<th>% encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 6 2015</td>
<td>47</td>
</tr>
<tr>
<td>Jun 4 2016</td>
<td>54</td>
</tr>
<tr>
<td>Jun 3 2017</td>
<td>67</td>
</tr>
<tr>
<td>Jun 2 2018</td>
<td>82</td>
</tr>
</tbody>
</table>

[Google Transparency Report 2018]
Existing systems make compromises

Excerpt from Oliver Michel's work: 

Packet-Level Analytics in Software without Compromises — Oliver Michel

Diagram showing stages of data processing with functions: filter(), analytics, groupby(), zip().

Key points:
- Loss of information
- Loss of capability
Programmable Forwarding Engines

- Programmable Forwarding Engines
  - Marple [SIGCOMM 2017]
  - *flow [ATC 2018]

High-Performance Network Telemetry

~ 131 M packet records/s
*flow technology

?
The ideal network analytics system

Is it possible to perform packet-level analytics on cloud-scale infrastructures without compromises?

- per-packet records
- x86 / general purpose programming language
- ~5M pps per core
Leveraging parallel architectures
Leveraging parallel architectures

Packet-Level Analytics in Software without Compromises — Oliver Michel
Characteristics of packet record workloads

Can we use properties of packet analytics workloads to our advantage?

- Network attached input
- Partitionability/aggregation
- High rates, small, well-formed records
Network attached input
Many small records

- Array vs. linked list
- Lock-free design
- Wait-free design
- Zero-copy operations
```cpp
int main(int argc, char** argv)
{
    jetstream::app app;
    auto source = app.add_stage<source>(1, "enp6s0f0");
    auto sink = app.add_stage<sink>(1, std::cout);
    app.connect<jetstream::pkt_t>(source, sink);
    app();
    return 0;
}
```
Performance

Parallel operators:
- Source
- Intermediate processors
- Sink

Graph:
- Throughput [M packets/s]
- Intermediate processors
- Passthrough
- Packets per source
Performance

- Facebook web cluster: ~ 91M egress pps
- ~32 cores for basic packet-level insight
- 176 web servers — 1 analytics server: ~0.5% of cluster capacity

Is it possible to perform packet-level analytics on cloud-scale infrastructures without compromises?

jetstream → high-performance, software network analytics platform
Q&A / Discussion

Oliver Michel

oliver.michel@colorado.edu
http://nsr.colorado.edu/oliver
The *right* approach for network monitoring and analytics?

What data do we *need* for monitoring/debugging?
PANEL OPENING SLIDE
Packet-Level Analytics in Software without Compromises

Oliver Michel, John Sonchack, Eric Keller, Jonathan M. Smith
University of Colorado Boulder, University of Pennsylvania

encrypted traffic

behavioral analysis

programmable forwarding engines

packet level record generation

complex applications

software processing

source

parallel operators

sink
Backup Slides
class source : public jetstream::proc {
    [...]
};

explicit source(const std::string& iface_name_) : proc() {
    add_out_port<jetstream::pkt_t>(0);
    [...]
}

jetstream::signal operator()() override {
    out_port<pkt_t>(0)->enqueue(read_from_nic(_pkt),
                jetstream::signal::continue);
    return jetstream::signal::continue;
}
Jetstream architecture

NUMA awareness

pipeline 1 → CPU socket 1
pipeline 2 → CPU socket 2

Backend
(e.g., time series DB)
Packet-Level Analytics without Compromises — Oliver Michel

Stream Processing

Filter only TCP

Parallelize group by IP Destination

ip_dst % 2 == 0

ip_dst % 2 == 1

Filter

by time (e.g., 10sec)

> n Bytes per 10 sec

Alert

Packet Packet

TCP Packet TCP Packet
Reducing copy operations
Reducing copy operations

```
1 packet p;
2 p.ip_proto = 6;
3 q.enqueue(p);
```

```
1 auto p = q.enqueue();
2 p->ip_proto = 6;
```
Technologies

• Programmable switches and PISA: Protocol Independent Switch Architecture
  • Reconfigurable match-action tables in hardware
  • multiple stages with TCAM/ALU pair, fixed processing time, guarantees line rate