Trumpet: Timely and Precise Triggers in Data Centers
Long failure repair times in large networks

Human-in-the-loop failure assessment and repair
Examine ongoing network events to find possible root-causes

Operators examine dashboards
A framework for **programmed** detection of **events** in large datacenters
- Link failure
- Middlebox failure
- Loop
- Packet delay
- Packet burst
- DDoS
- Congestion
- Burst Loss
- Lost packet
- Incast
- Switch failure
- Blackhole
- Load imbalance
- Traffic hijack

- Availability
- Performance
- Security
Detecting Transient Congestion

40 ms burst

Timeouts lasting several 100 ms
Dashboard data insufficient

*Aggregated, often sampled* measures of network health
Did this tenant see a sudden increase in traffic over the last few milliseconds?
Inspect every packet.

- Link failure
- Middlebox failure
- DDoS
- Traffic surge
- Congestion
- Switch failure
- Burst Loss
- Incast
- Packet delay
- Lost packet
- Loop
- Packet burst
- Blackhole
- Load imbalance
- Traffic hijack

Some events may require inspecting every packet.
Eventing Framework Requirements

Expressivity
★ Set of possible events not known a priori

Fine timescale eventing
★ Capture transient and onset events

Per-packet processing
★ Precise event determination

Because data centers will require high availability and high utilization
Where do we place eventing functionality?

- **Switches**
- **NICs**
- **Hosts**

- Are programmable
- Have processing power for fine-time scale eventing
- Already inspect every packet
Trumpet is a host-based eventing framework
Research Questions

What eventing architecture permits programmability and visibility?

How can we achieve precise eventing at fine timescales?

What is the performance envelope of such an eventing framework?
Research Questions

What eventing architecture permits programmability and visibility?

How can we achieve precise eventing at fine timescales?

What is the performance envelope of such an eventing framework?

*Trumpet* has a logically centralized event manager that aggregates *local events* from per-host *packet monitors*. 
For each packet matching **Filter**

group by **Flow-granularity**

and report every **Time-interval**

each group that satisfies **Predicate**

Flow volumes, loss rate, loss pattern (bursts), delay
Is there any flow sourced by a service that sees a burst of losses in a small interval?

For each packet matching Service IP Prefix

group by 5-tuple

and report every 10ms

any flow whose \( \text{sum (is\_lost & is\_burst)} > 10\% \)
Is there a job in a cluster that sees abnormal traffic volumes in a small interval?

For each packet matching Cluster IP Prefix and Port group by Job IP Prefix and report every 10ms any job whose sum (volume) > 100MB.
Congestion?

Trumpet Event Manager

Congestion Triggers

Contains event attributes, detects local events
Trumpet can be used by programs to drill-down to potential root causes.
Research Questions

What eventing architecture permits programmability and visibility?

How can we achieve precise eventing at fine timescales?

What is the performance envelope of such an eventing framework?

The monitor optimizes packet processing to inspect every packet and evaluate predicates at fine timescales.
Run monitor on CPU core used by software switch

❖ Conserves CPU resources
❖ Avoids inter-core synchronization
Can a single CPU core monitor thousands of triggers at full packet rate (14.8 Mpps) on a 10G NIC?
Two Obvious Tricks

- Use kernel bypass
  - Avoid kernel stack overhead
- Use polling to have tighter scheduling
  - Trigger time intervals at 10ms

Necessary, but far from sufficient....
Packet → Match filters → Update statistics at flow granularity → Check predicate at time-interval

With 1000s of triggers

<table>
<thead>
<tr>
<th>Filter</th>
<th>Flow granularity</th>
<th>Time interval</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IP = 10.1.1.0/24</td>
<td>5-tuple</td>
<td>10ms</td>
<td>Sum(loss) &gt; 10%</td>
</tr>
<tr>
<td>Source IP = 20.2.2.0/24</td>
<td>Service IP prefix</td>
<td>100ms</td>
<td>Sum(size) &lt; 10MB</td>
</tr>
</tbody>
</table>
Which of these should be performed

❖ *On-path*
❖ *Off-path*
Which operations to do on-path?

❖ 70ns to forward and inspect packet
How to schedule off-path operations?

- Off-path on same core, can delay packets
- Bound delay to a few μs
Packet History

Match filters

Update statistics at flow granularity

Check predicate at time-interval

Doesn’t scale to large numbers of triggers
Still cannot reach goal

- Memory subsystem becomes a bottleneck
Packet

Match filters

Update statistics at 5-tuple granularity

Can be different from flow-granularity

Gather statistics at flow granularity

Check predicate at time-interval

On-Path

Off-Path
Packet ➔ Match filters ➔ Update statistics at 5-tuple granularity

- Use tuple-space search for matching
- Match on first packet, cache match
- Lay out tables to enable cache prefetch
- Use TLB huge pages for tables
Optimizations

- Lazy cleanup of statistics across intervals
- Lay out tables to enable cache prefetch
- *Bounded-delay cooperative scheduling*
Bound delay to a few μs
Research Questions

What eventing architecture permits programmability and visibility?

How can we achieve precise eventing at fine timescales?

What is the performance envelope of such an eventing framework?

Trumpet can monitor thousands of triggers at full packet rate on a 10G NIC
Trumpet is expressive
- Transient congestion
- Burst loss
- Attack onset

Trumpet scales to thousands of triggers

Trumpet is DoS-Resilient
Detecting Transient Congestion

Trumpet can detect millisecond scale congestion events.
Trumpet can process *14.8 Mpps*

- 64 byte packets at 10G
- 650 byte packets at 4x10G

... while evaluating 4K triggers at 10ms granularity

*Xeon ES-2650, 10-core 2.3 Ghz, Intel 82599 10G NIC*
Above this rate, Trumpet would **miss events**.

Triggers matched by each flow:
- **4**
- **8**
- **16**

How often each predicate is checked:

- **Rate (Mpps)**
- **Check interval (ms)**
At moderate packet rates, can detect events at 1ms

Number of <trigger, flow> pairs increases statistics gathering overhead
Need to profile and provision Trumpet deployment.

Above 10ms, CPU can sustain full packet rate.
Conclusion

Future datacenters will need fast and precise eventing

- Trumpet is an expressive system for host-based eventing

Trumpet can process 16K triggers at full packet rate

- ... without delaying packets by more than 10 μs

Future work: scale to 40G NICs

- ... perhaps with NIC or switch support

https://github.com/USC-NSL/Trumpet
Outage budget for **five 9s** availability

24 seconds per month

Long failure durations due to time to **root-cause** failures
Every optimization is necessary*

*Details in the paper
Humans in the Loop

Detect
Locate
Inspect
Fix
Programs in the Loop

Detect

Locate

Inspect

Fix

Programs in the loop