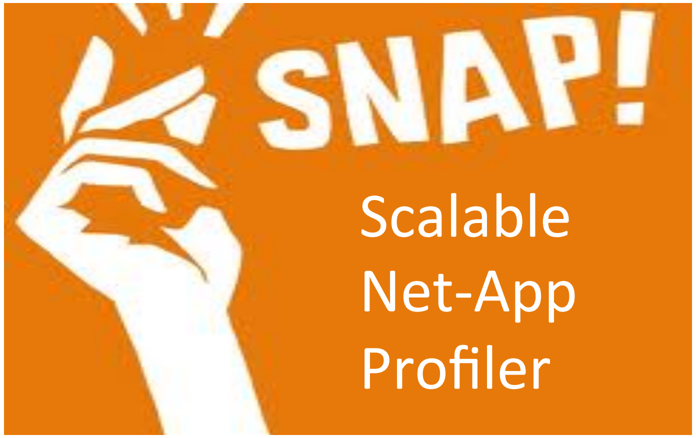
Profiling Network Performance in Multi-tier Datacenter Applications

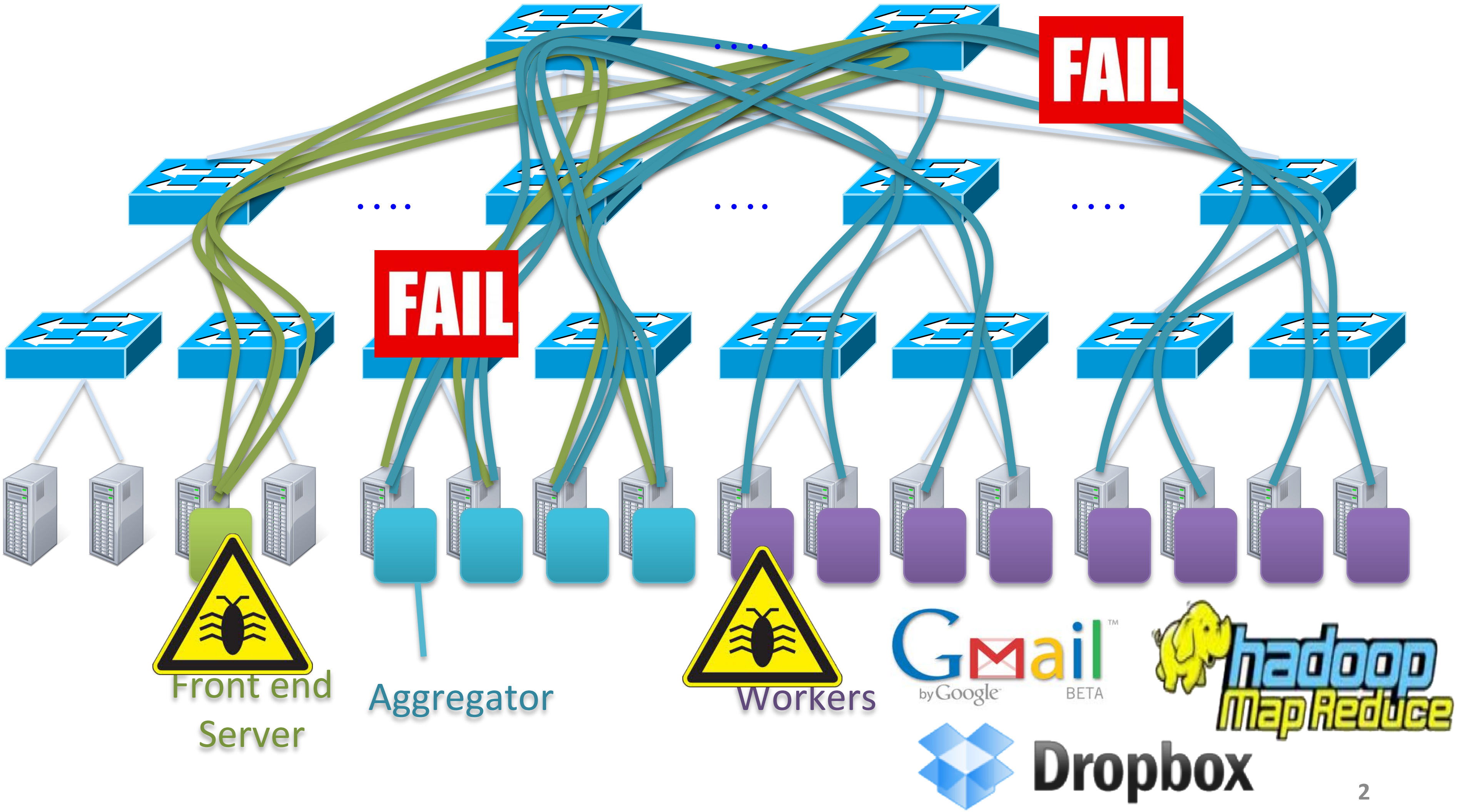


Scalable
Net-App
Profiler

Minlan Yu
Princeton University

Joint work with Albert Greenberg, Dave Maltz, Jennifer Rexford, Lihua Yuan, Srikanth Kandula, Changhoon Kim

Applications inside Data Centers

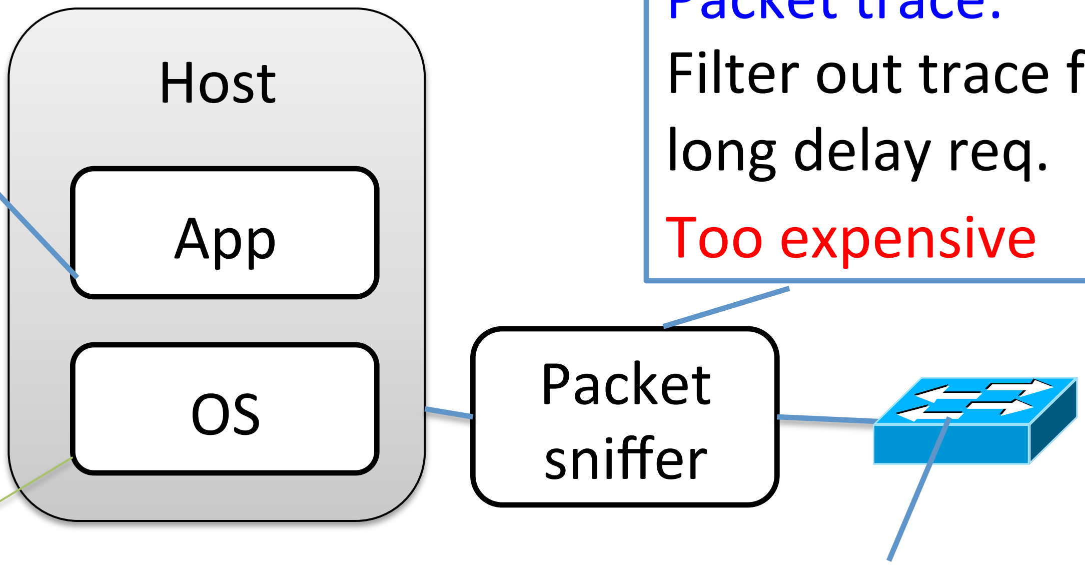


Challenges of Datacenter Diagnosis

- Large complex applications
  - Hundreds of application components
  - Tens of thousands of servers
- New performance problems
  - Update code to add features or fix bugs
  - Change components while app is still in operation
- Old performance problems (Human factors)
  - Developers may not understand network well
  - Nagle's algorithm, delayed ACK, etc.

Diagnosis in Today's Data Center

App logs:
#Reqs/sec
Response time
1% req. >200ms delay
Application-specific



Packet trace:
Filter out trace for long delay req.
Too expensive

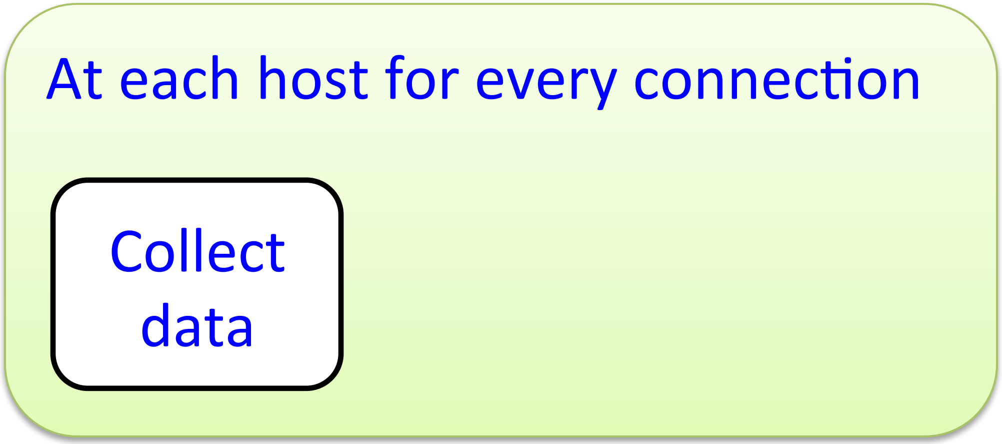
SNAP:
Diagnose net-app interactions
Generic, fine-grained, and lightweight

Switch logs:
#bytes/pkts per minute
Too coarse-grained

SNAP: A Scalable Net-App Profiler

that runs everywhere, all the time

SNAP Architecture



At each host for every connection

Collect data

Collect Data in TCP Stack

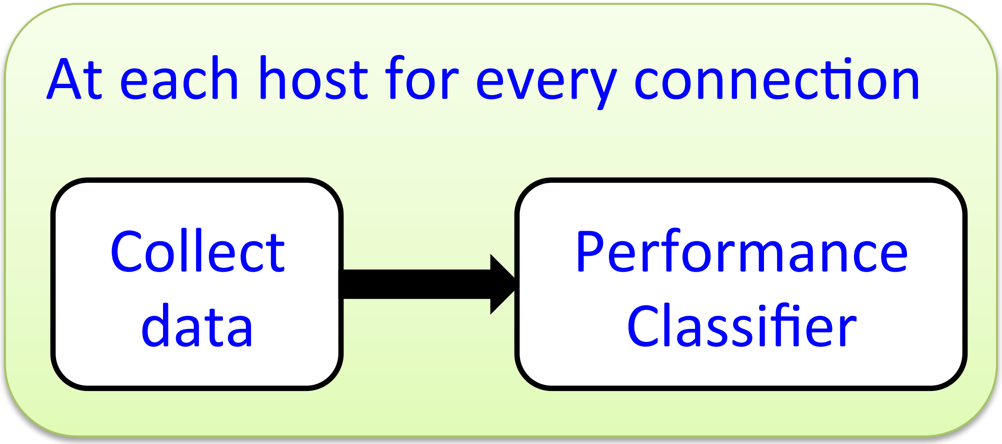
- TCP understands net-app interactions
  - Flow control: How much data *apps* want to read/write
  - Congestion control: *Network* delay and congestion

- Collect TCP-level statistics
  - Defined by RFC 4898
  - Already exists in today's Linux and Windows OSes

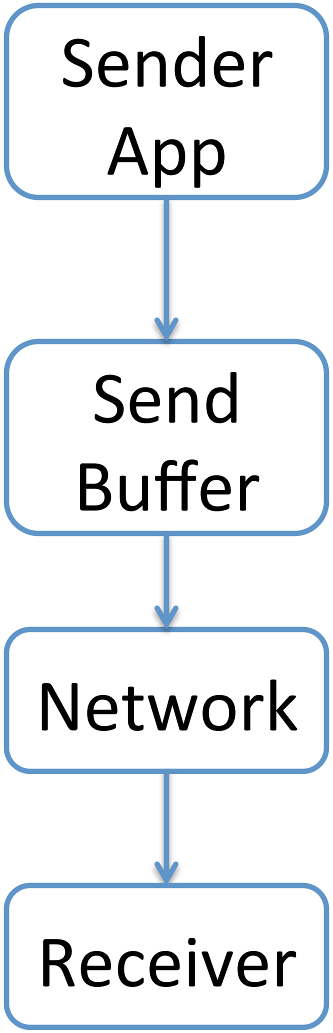
TCP-level Statistics

- Cumulative counters
  - Packet loss: #FastRetrans, #Timeout
  - RTT estimation: #SampleRTT, #SumRTT
  - Receiver: RwinLimitTime
  - Calculate the difference between two polls
- Instantaneous snapshots
  - #Bytes in the send buffer
  - Congestion window size, receiver window size
  - Representative snapshots based on Poisson sampling

SNAP Architecture



Life of Data Transfer



Sender App

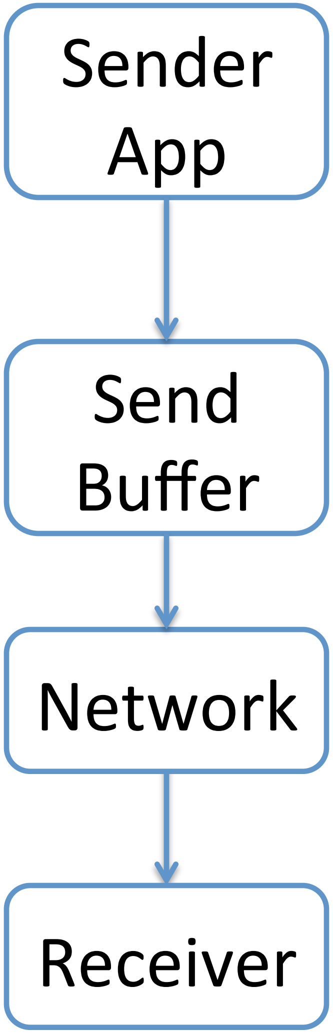
Send Buffer

Network

Receiver

- Application generates the data
- Copy data to send buffer
- TCP sends data to the network
- Receiver receives the data and ACK

Taxonomy of Network Performance



Sender App

– No network problem

Send Buffer

– Send buffer not large enough

Network

– Fast retransmission

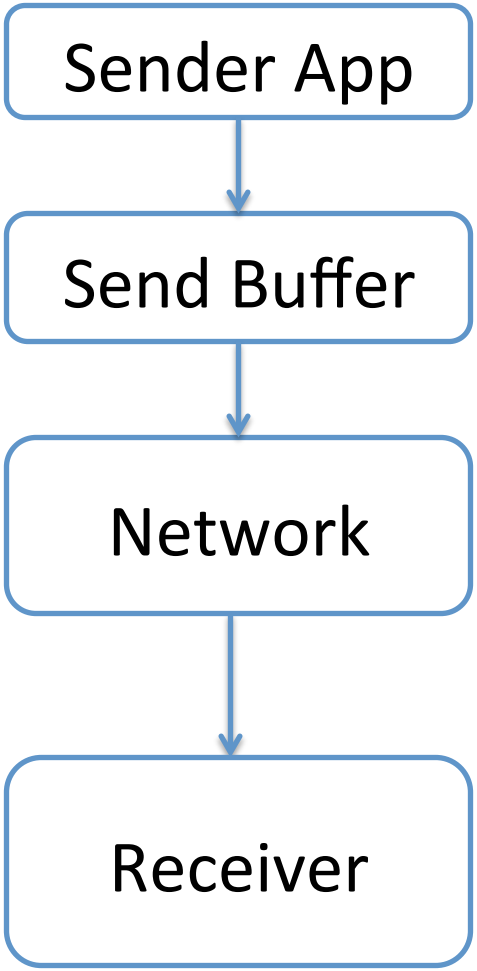
– Timeout

Receiver

– Not reading fast enough (CPU, disk, etc.)

– Not ACKing fast enough (Delayed ACK)

Identifying Performance Problems



– Not any other problems

– #bytes in send buffer — Sampling

– #Fast retransmission

– #Timeout

Direct measure

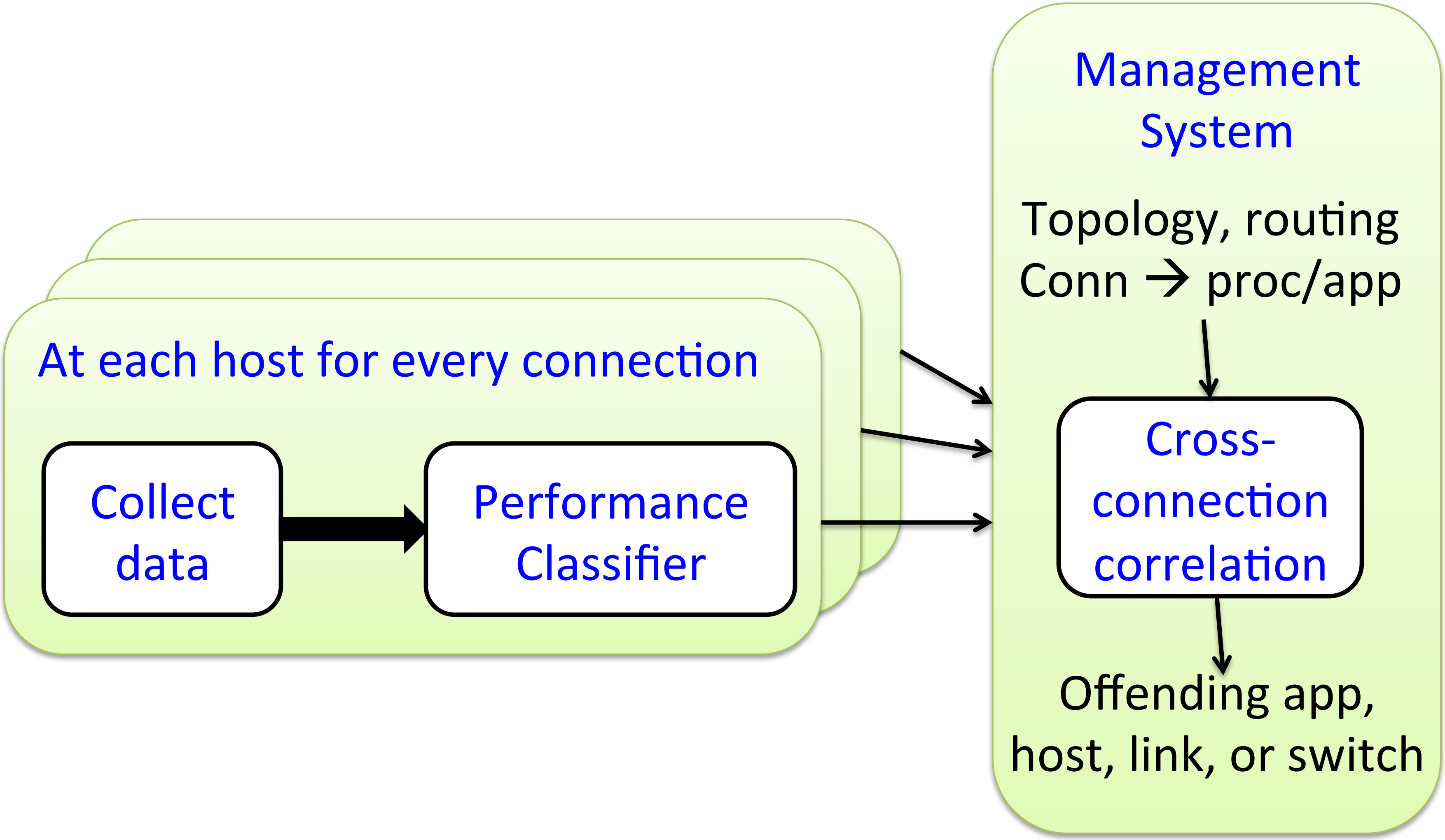
– RwinLimitTime

– Delayed ACK

Inference

*diff(SumRTT) > diff(SampleRTT)*MaxQueuingDelay*

SNAP Architecture



At each host for every connection

Collect data

Performance Classifier

Management System

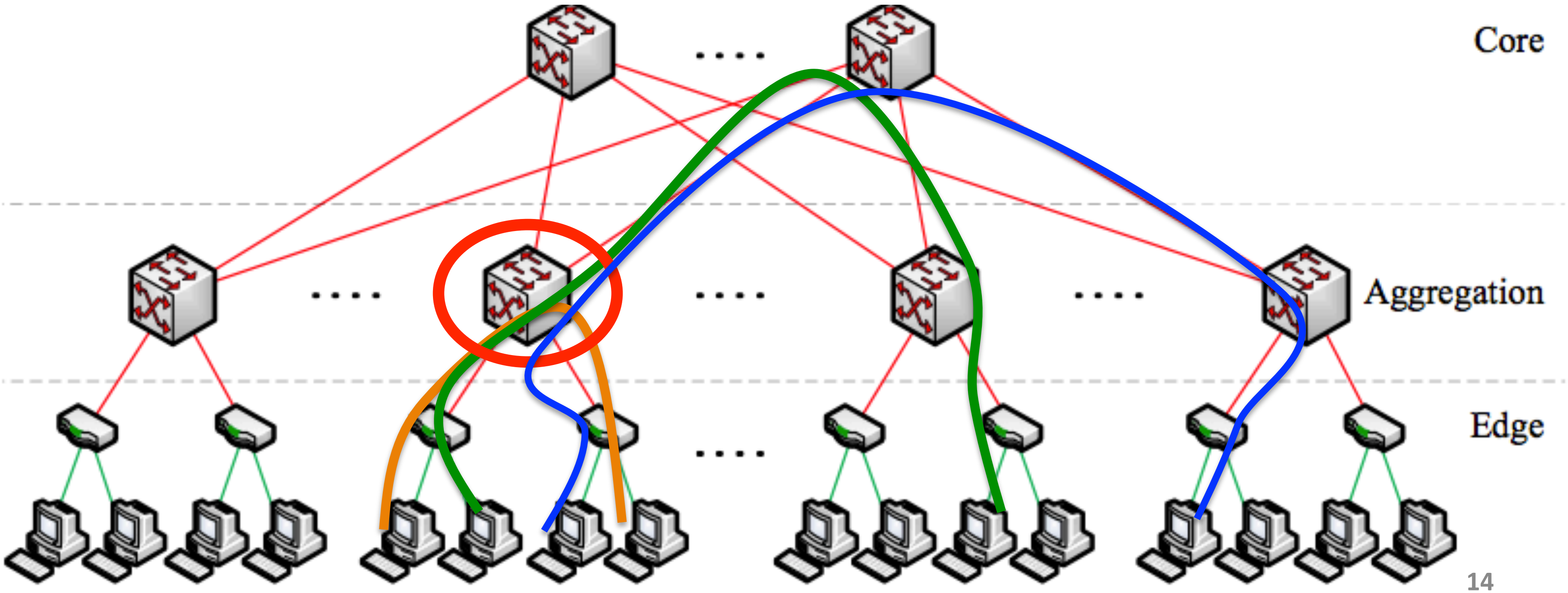
Topology, routing
Conn → proc/app

Cross-connection correlation

Offending app, host, link, or switch

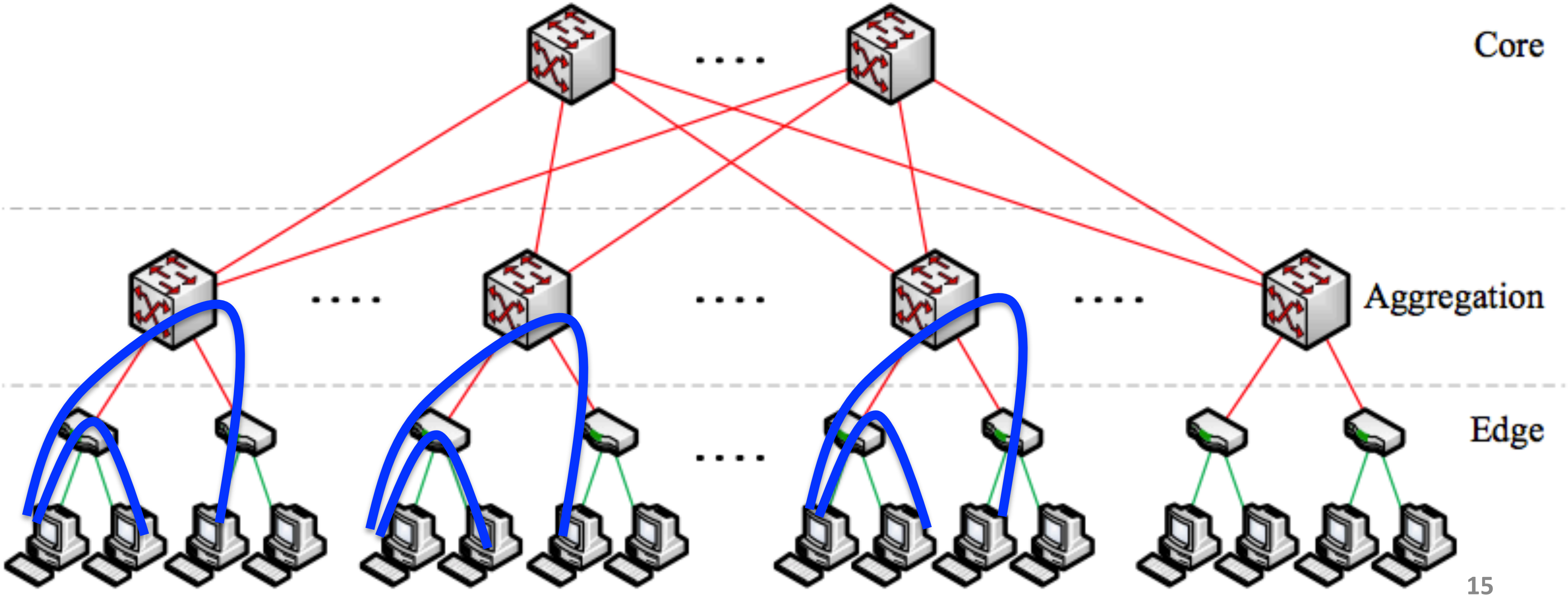
Pinpoint Problems via Correlation

- Correlation over shared switch/link/host
  - Packet loss for all the connections going through one switch/host
  - Pinpoint the problematic switch

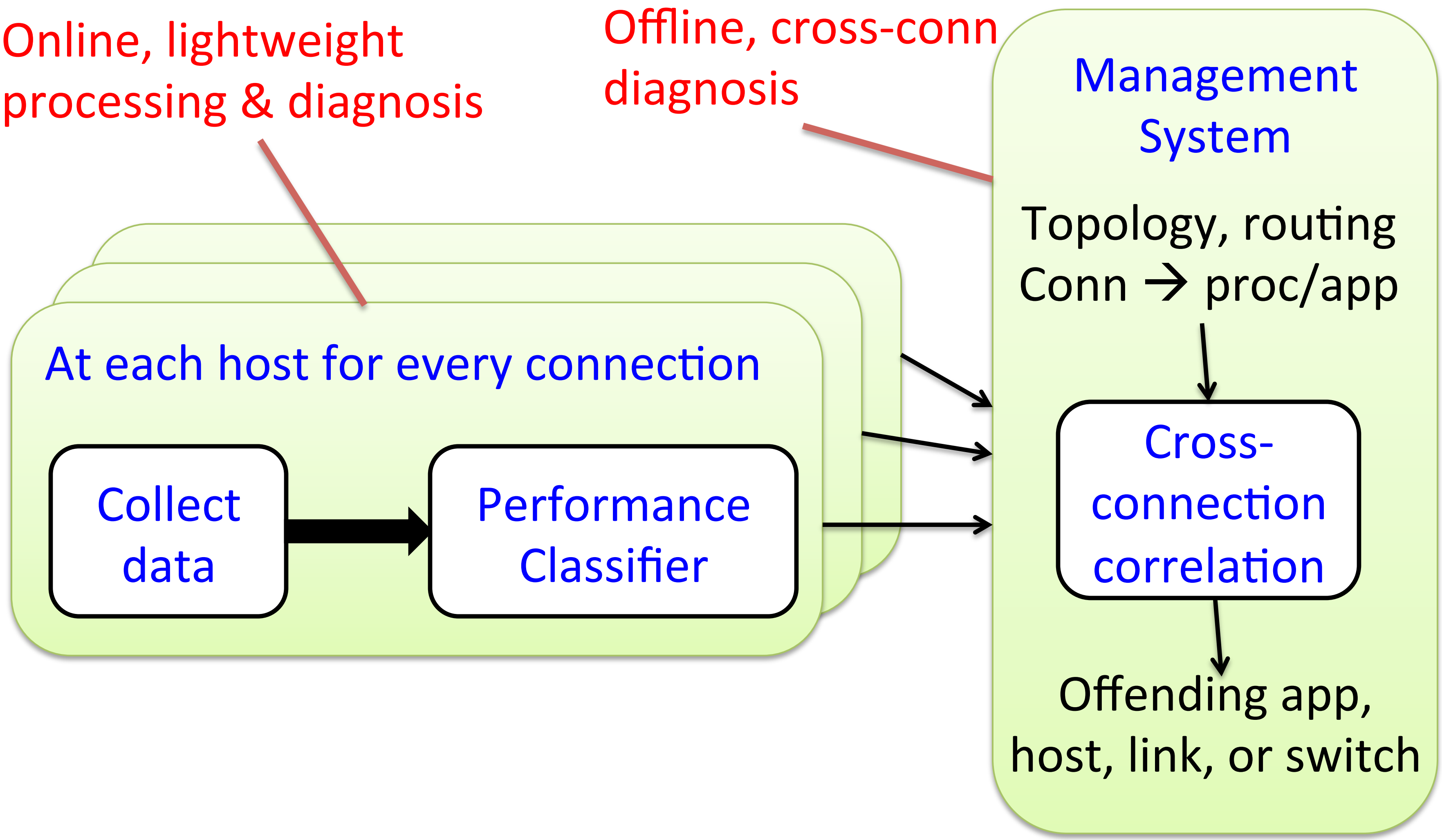


Pinpoint Problems via Correlation

- Correlation over application
  - – Same application has problem on all machines
  - – Report aggregated application behavior



SNAP Architecture



Online, lightweight processing & diagnosis

Offline, cross-conn diagnosis

Management System

Topology, routing
Conn → proc/app

Cross-connection correlation

Offending app, host, link, or switch

At each host for every connection

Collect data

Performance Classifier

Reducing SNAP Overhead

- SNAP overhead
  - – Data volume: 120 Bytes per connection per poll
  - – CPU overhead:
    - 5% for polling 1K connections with 500 ms interval
    - Increases with #connections and polling freq.

- Solution: Adaptive tuning of polling frequency
  - – Reduce polling frequency to stay within a target CPU
  - – Devote more polling to more problematic connections

SNAP in the Real World

Key Diagnosis Steps

- Identify performance problems
  - Correlate across connections
  - Identify applications with severe problems
- Expose simple, useful information to developers
  - Filter important statistics and classification results
- Identify root cause and propose solutions
  - Work with operators and developers
  - Tune TCP stack or change application code

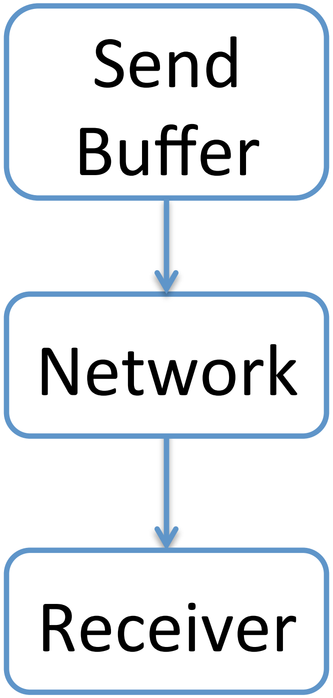
SNAP Deployment

- Deployed in a production data center
  - *8K* machines, *700* applications
  - Ran SNAP for a week, collected terabytes of data

- Diagnosis results
  - Identified *15* major performance problems
  - *21%* applications have network performance problems

Characterizing Perf. Limitations

#Apps that are limited
for > 50% of the time



Send Buffer

1 App – Send buffer not large enough

Network

6 Apps
– Fast retransmission
– Timeout

Receiver

8 Apps – Not reading fast enough (CPU, disk, etc.)

144 Apps – Not ACKing fast enough (Delayed ACK)

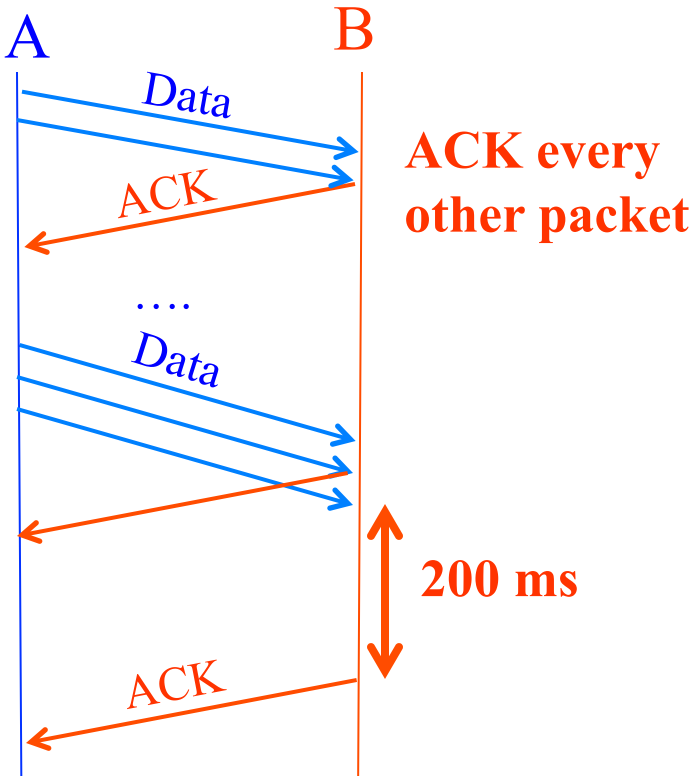
Three Example Problems

- Delayed ACK affects delay sensitive apps
- Congestion window allows sudden burst
- Significant timeouts for low-rate flows

Problem 1: Delayed ACK

- Delayed ACK affected many delay sensitive apps
  - *even* #pkts per record → 1,000 records/sec

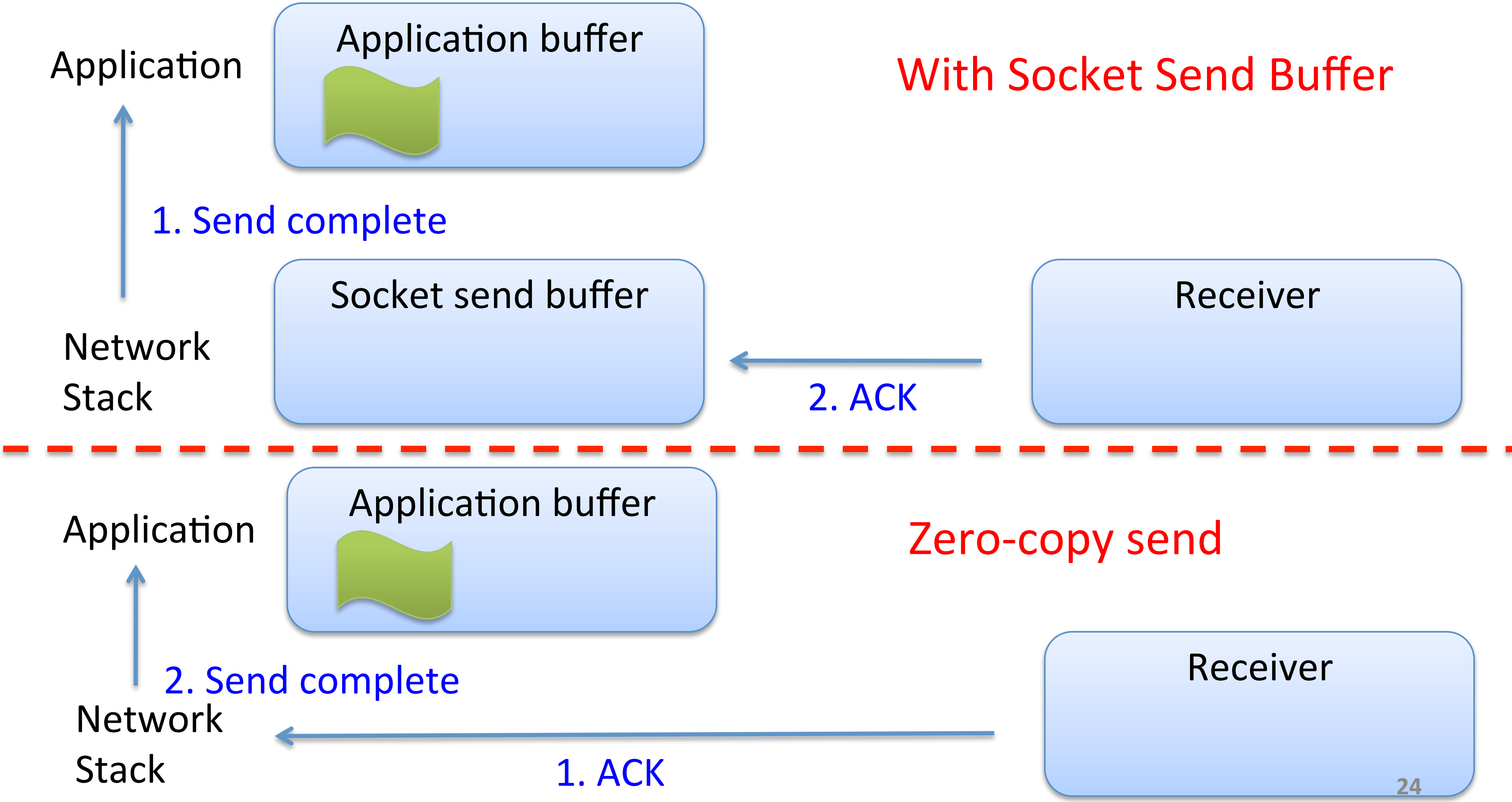
    odd #pkts per record → 5 records/sec
  - Delayed ACK was used to reduce bandwidth usage and server interrupts



Proposed solutions:
Delayed ACK should be disabled in data centers

Send Buffer and Delayed ACK

- SNAP diagnosis: Delayed ACK and zero-copy send



Application

Application buffer

With Socket Send Buffer

1. Send complete

Socket send buffer

Receiver

Network Stack

2. ACK

Application

Application buffer

Zero-copy send

2. Send complete

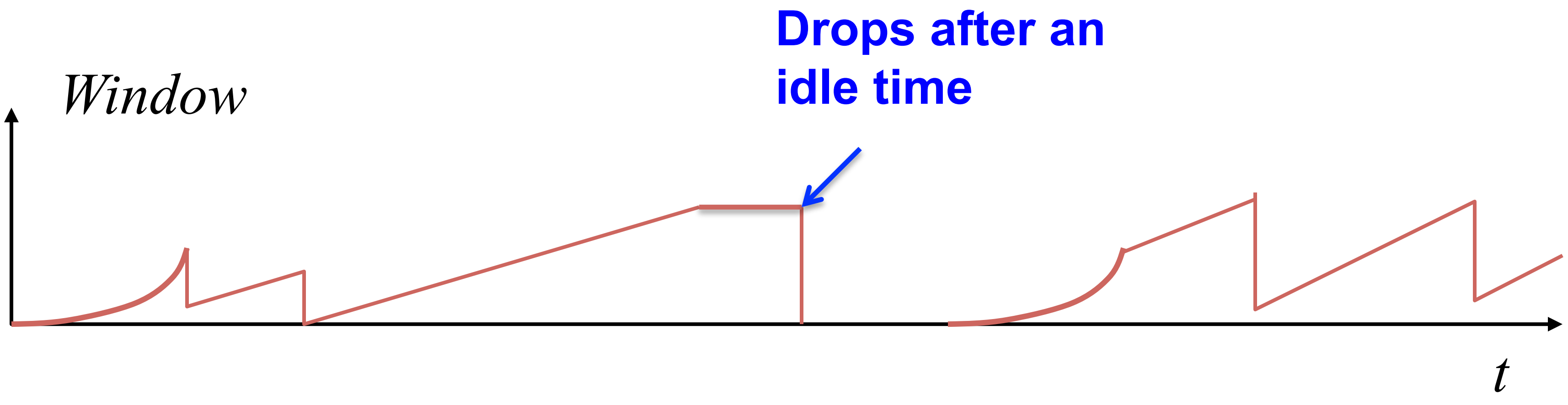
Receiver

Network Stack

1. ACK

Problem 2: Congestion Window Allows Sudden Bursts

- Increase congestion window to reduce delay
  - To send *64* KB data with *1* RTT
  - Developers intentionally keep congestion window large
  - Disable slow start restart in TCP



Slow Start Restart

- SNAP diagnosis
  - Significant packet loss
  - Congestion window is too large after an idle period

- Proposed solutions
  - Change apps to send less data during congestion
  - New transport protocols that consider both congestion and delay

Problem 3: Timeouts for Low-rate Flows

- SNAP diagnosis
  - More fast retrans. for high-rate flows (1-10MB/s)
  - More timeouts with low-rate flows (10-100KB/s)

- Proposed solutions
  - Reduce timeout time in TCP stack
  - New ways to handle packet loss for small flows

Conclusion

- A simple, efficient way to profile data centers
  - Passively measure real-time network stack information
  - Systematically identify problematic stages
  - Correlate problems across connections
- Deploying SNAP in production data center
  - Diagnose net-app interactions
  - A quick way to identify them when problems happen
- Future work
  - Extend SNAP to diagnose wide-area transfers

Thanks!