Distributed Network Monitoring and Debugging with SwitchPointer



Praveen Tammana[†] Rachit Agarwal[‡] Myungjin Lee[†]

[†]University of Edinburgh, [‡]Cornell University

Network monitoring and debugging is complex

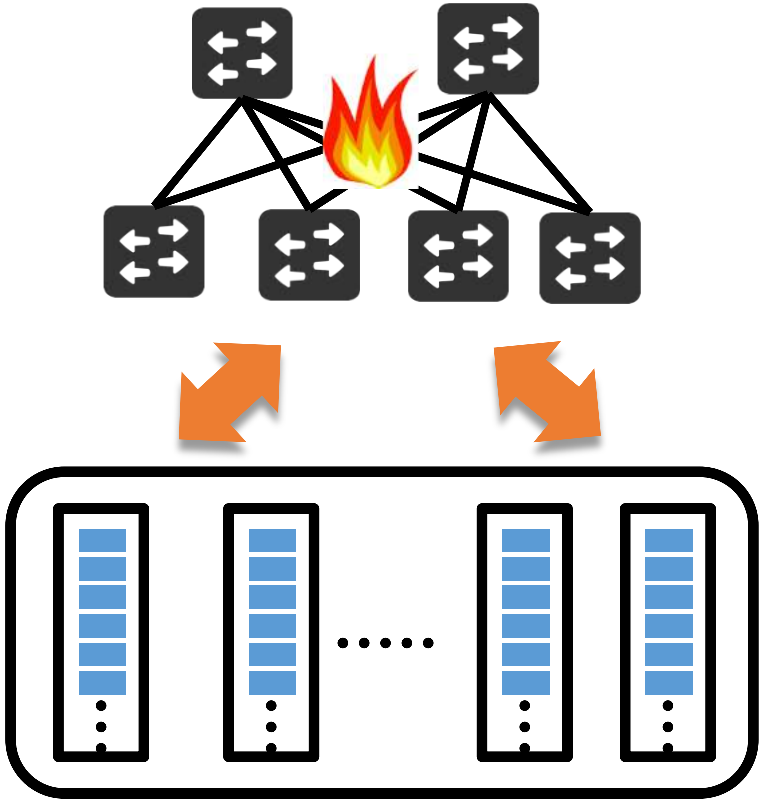
Network monitoring and debugging is complex

Marple [SIGCOMM'17], PathDump [OSDI'16], FlowRadar [NSDI'16],
EverFlow [SIGCOMM'15], Trumpet [SIGCOMM'16], Opensketch [NSDI'13]

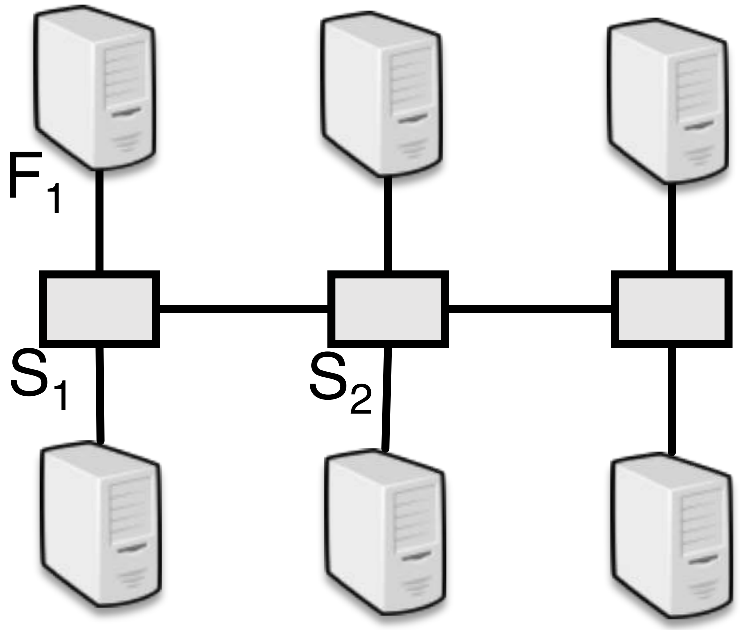
Network monitoring and debugging is complex

Marple [SIGCOMM'17], PathDump [OSDI'16], FlowRadar [NSDI'16], EverFlow [SIGCOMM'15], Trumpet [SIGCOMM'16], Opensketch [NSDI'13]

- Increasingly larger scale
  - Over 100k endpoints
  - 10/40/100 GE
  - Aggregate traffic > 100 Tbps



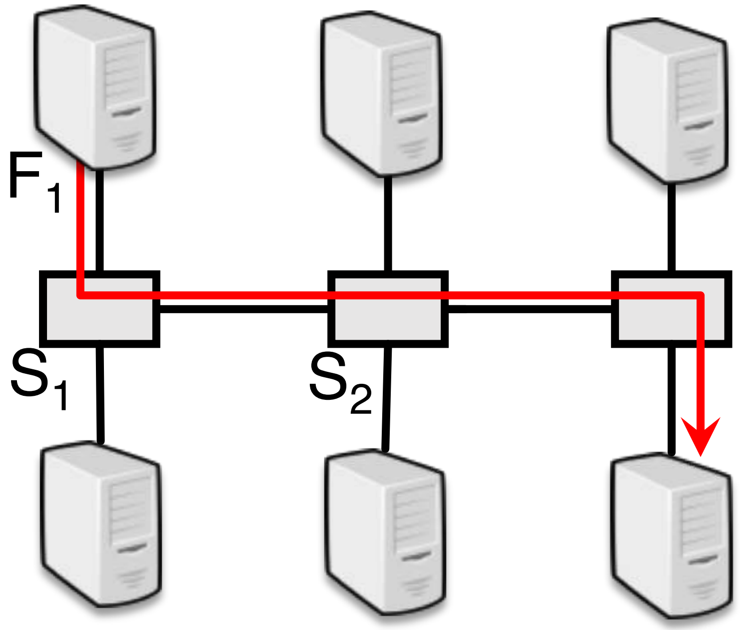
An example: Too many red lights



F_1: Low priority

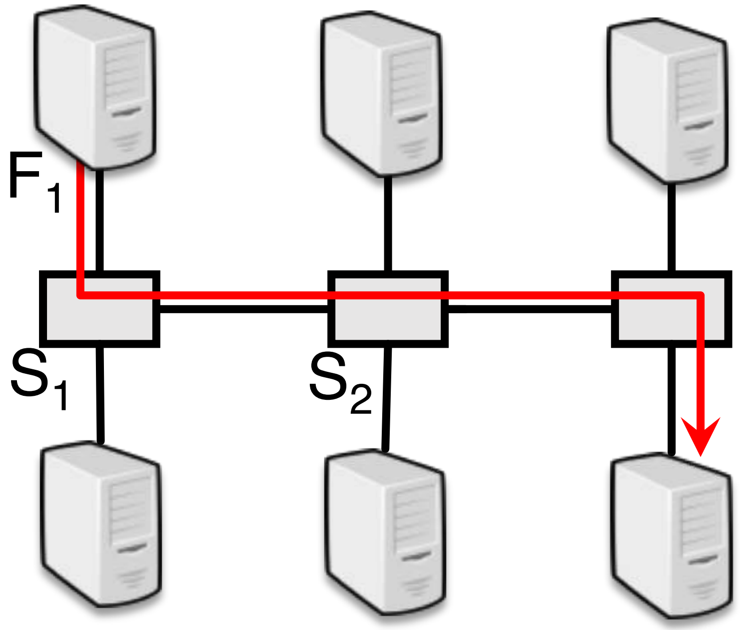
F_2, F_3: High priority

An example: Too many red lights

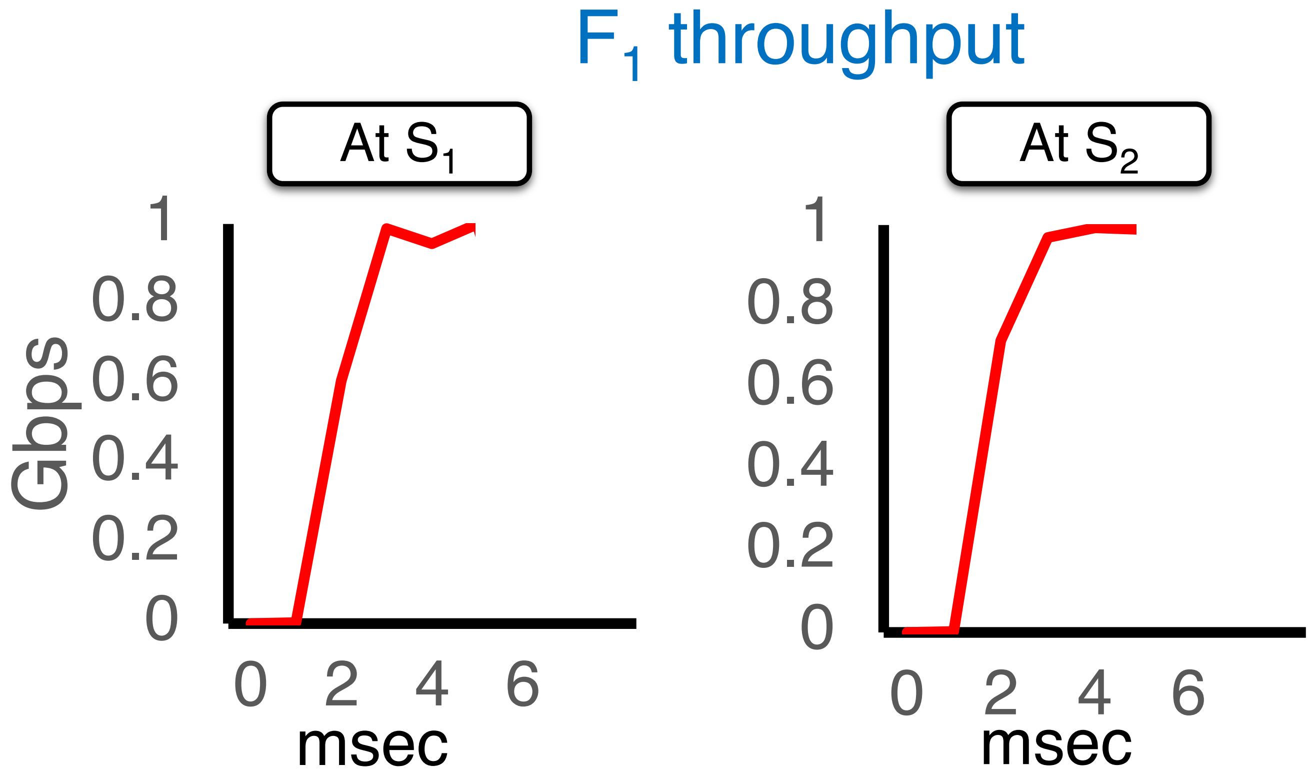


F_1: Low priority
F_2, F_3: High priority

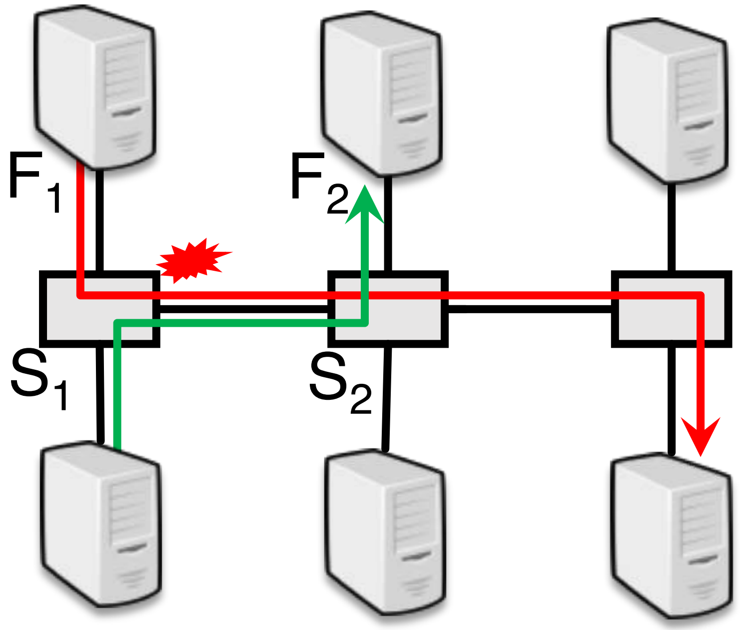
An example: Too many red lights



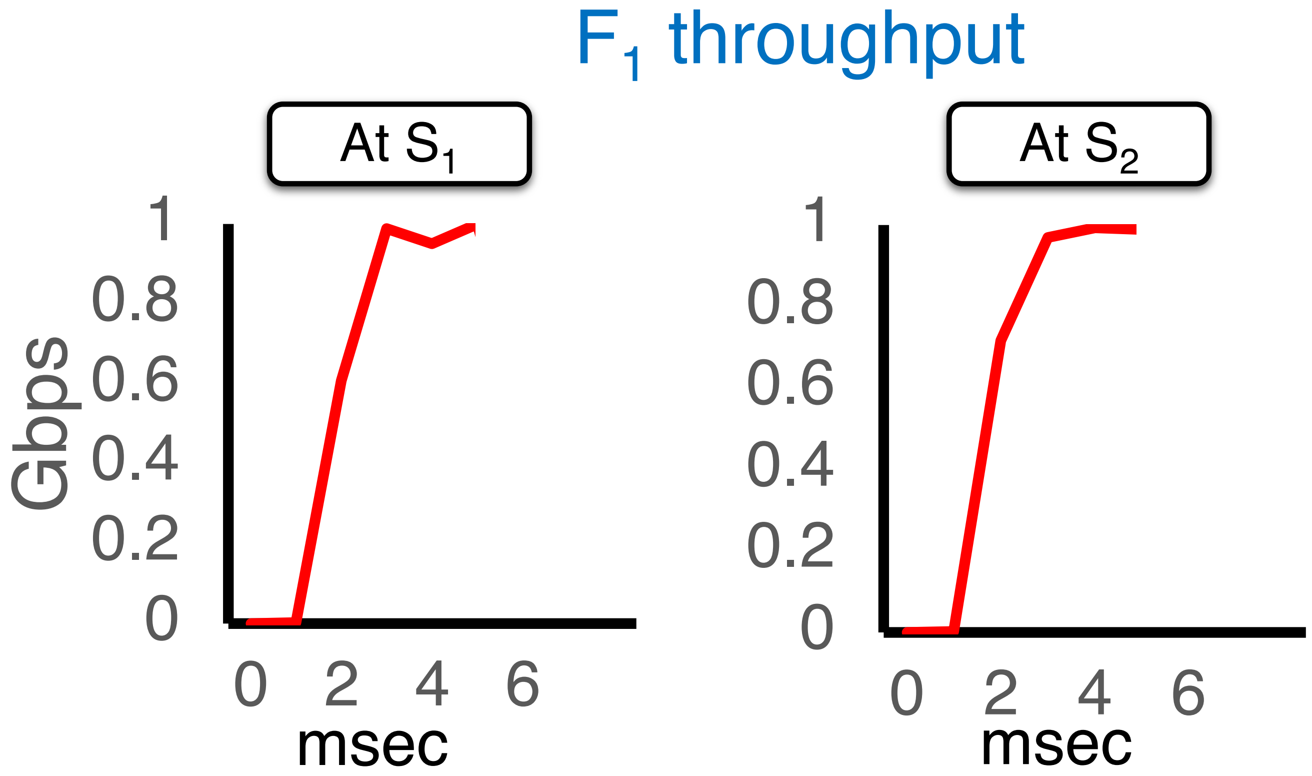
F$_1$: Low priority
F$_2$, F$_3$: High priority



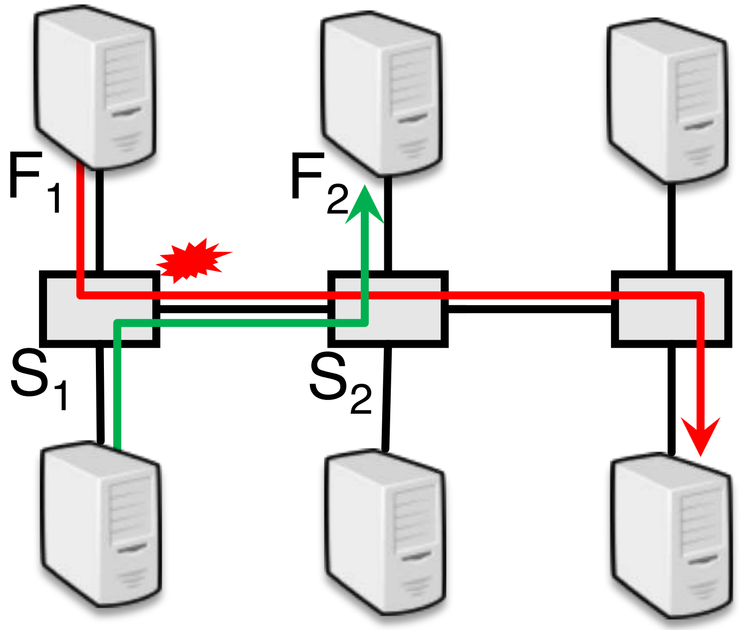
An example: Too many red lights



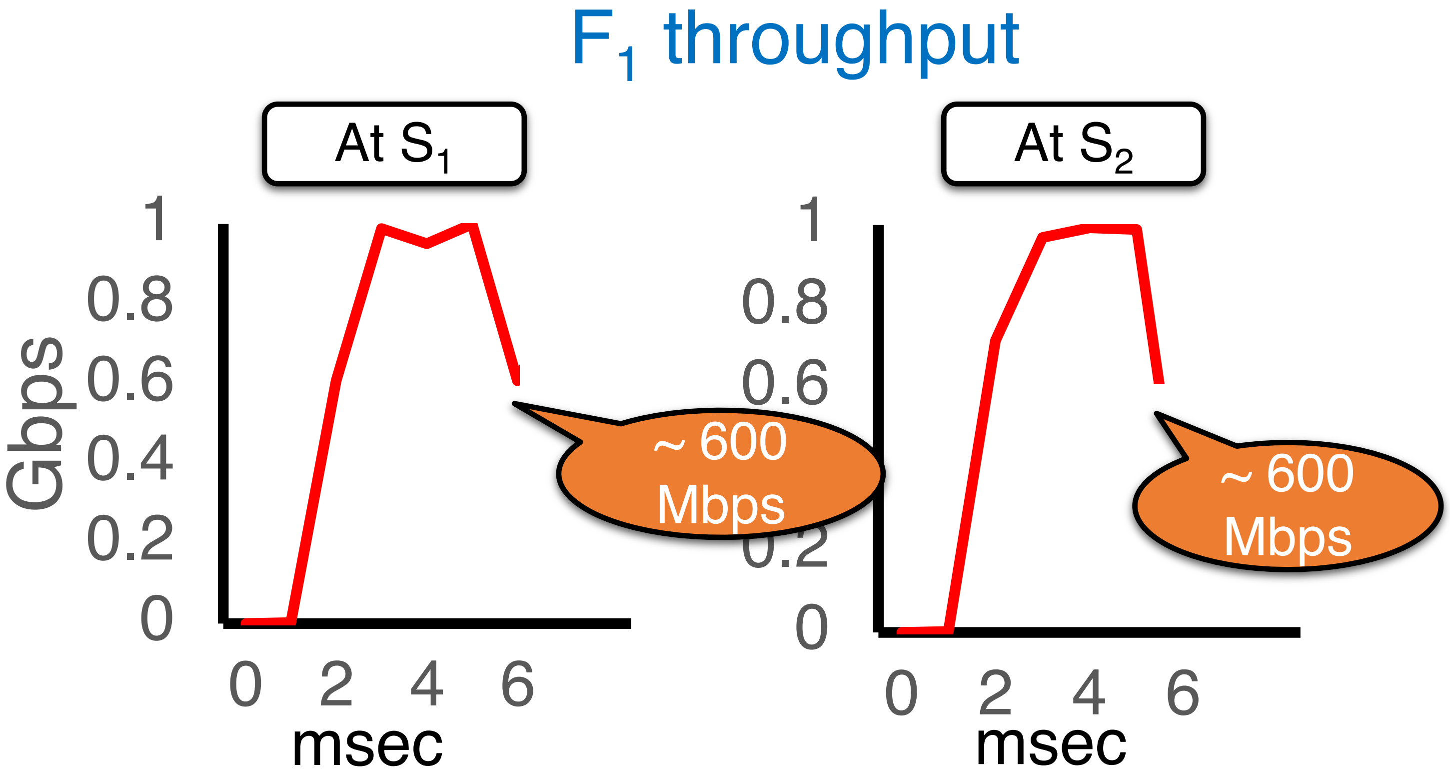
F_1: Low priority
F_2, F_3: High priority



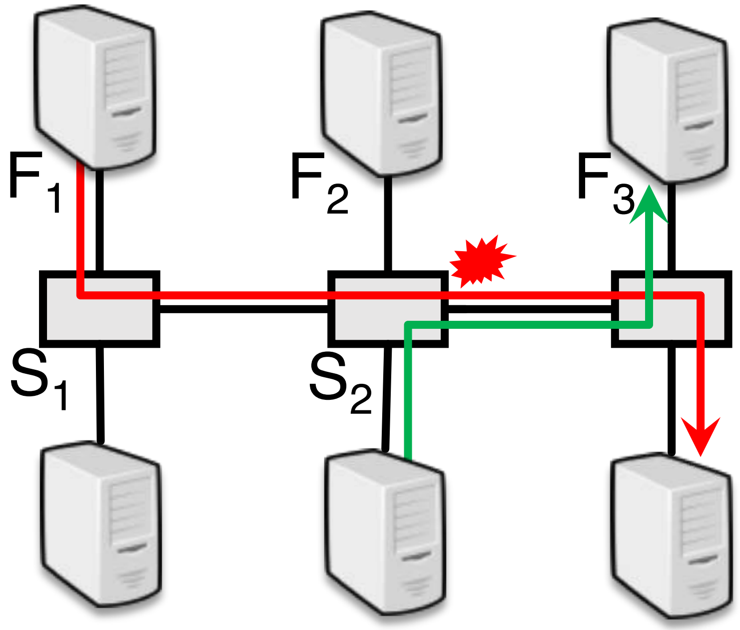
An example: Too many red lights



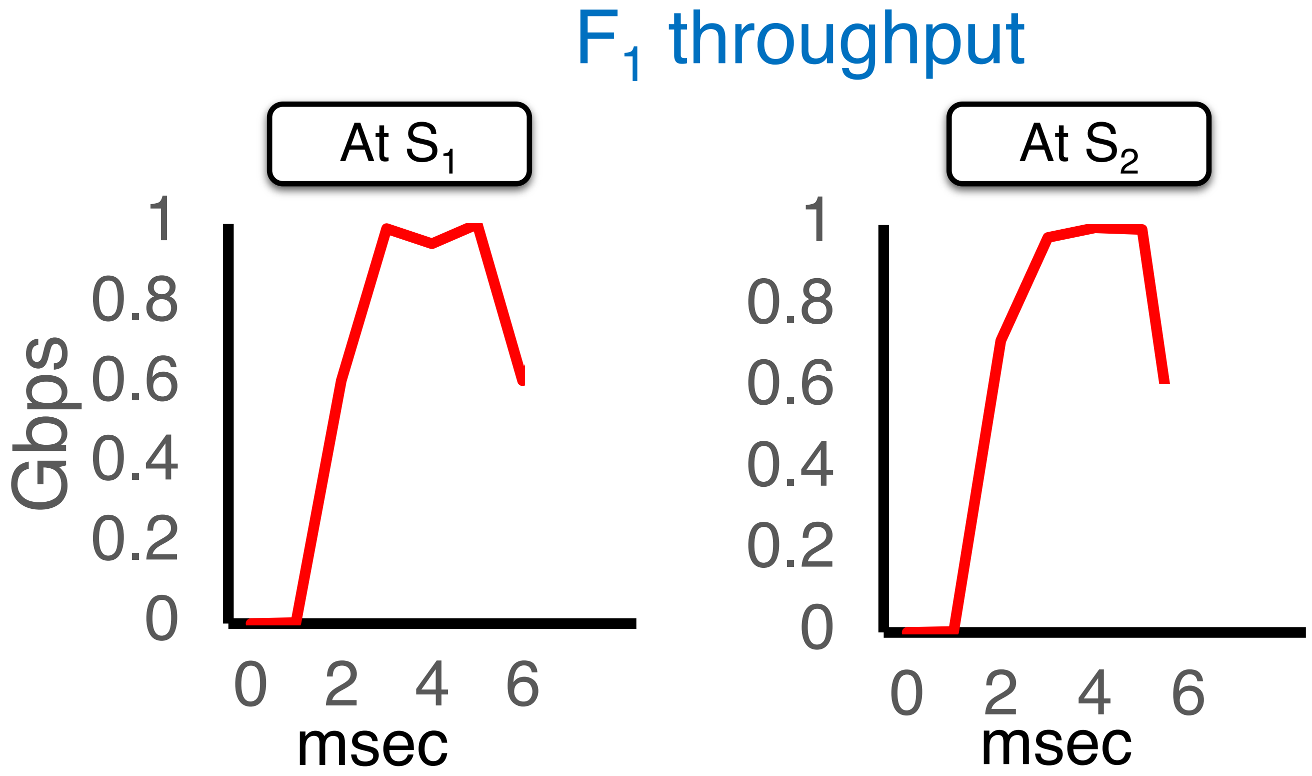
F_1: Low priority
F_2, F_3: High priority



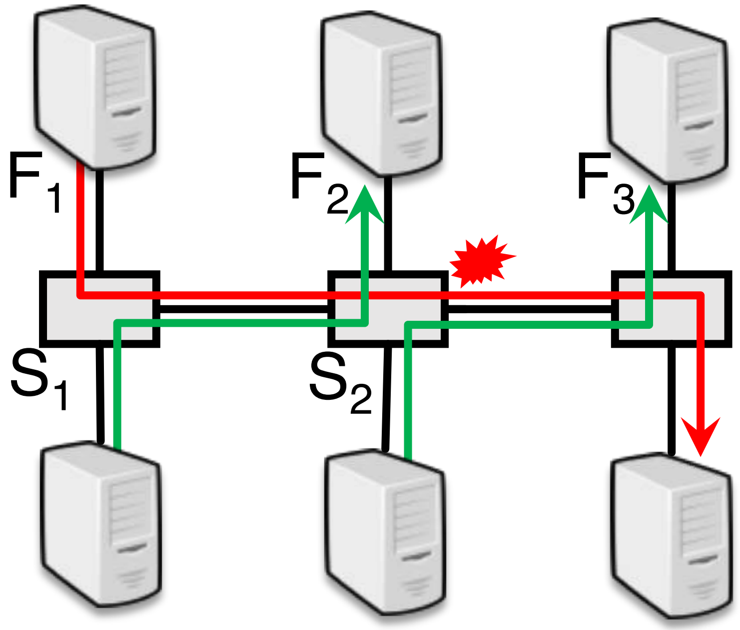
An example: Too many red lights



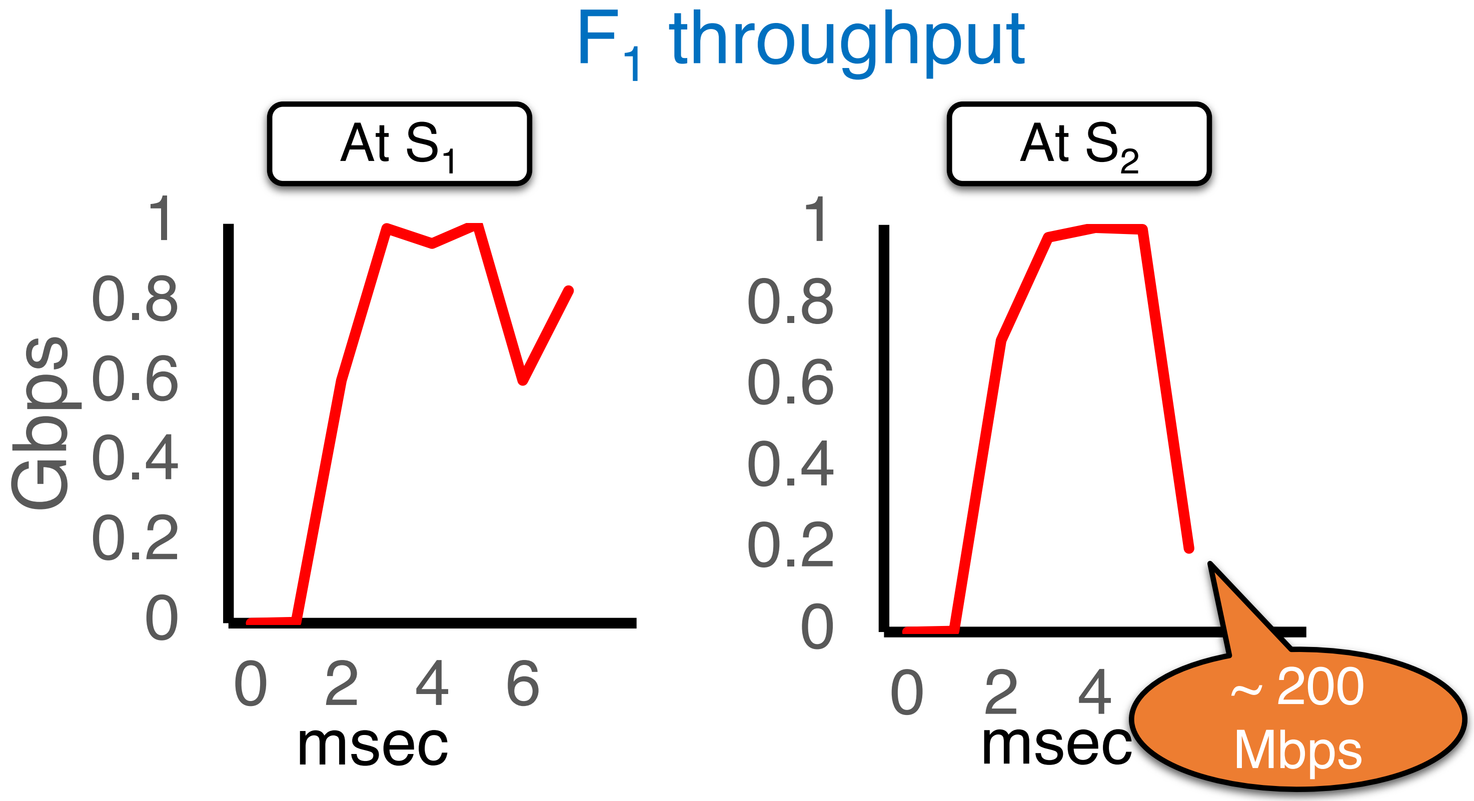
F_1: Low priority
F_2, F_3: High priority



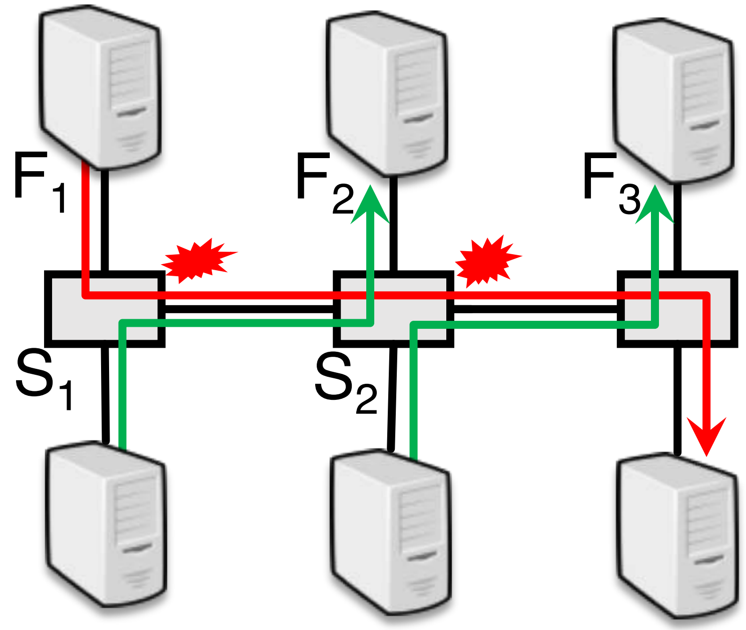
An example: Too many red lights



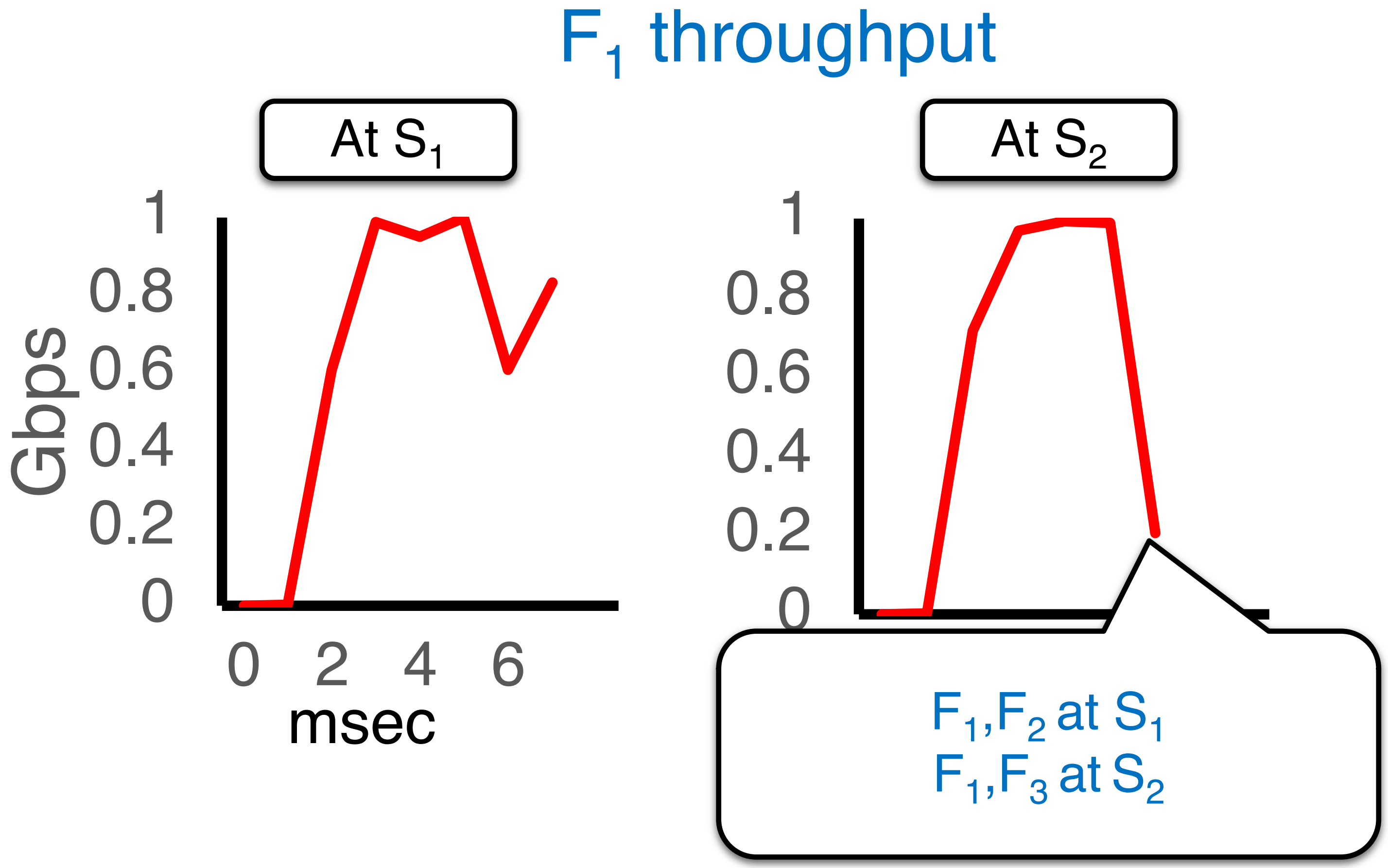
F_1: Low priority
F_2, F_3: High priority



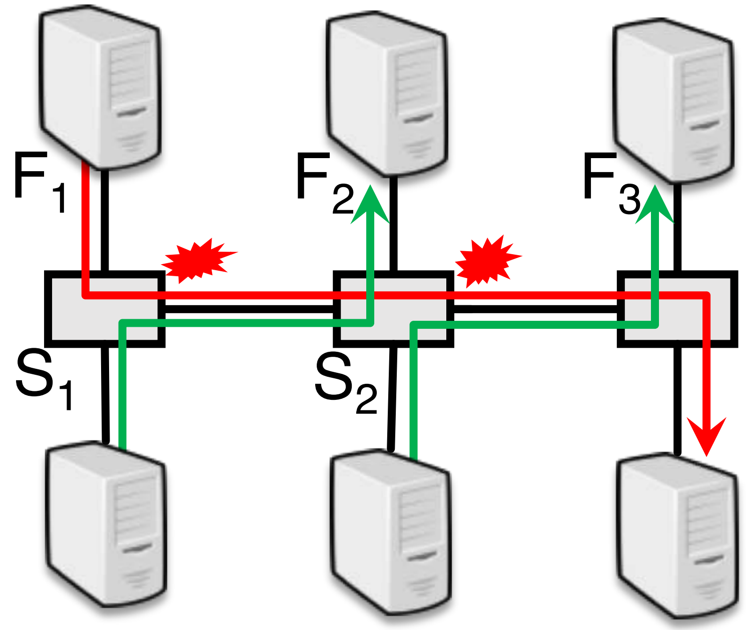
An example: Too many red lights



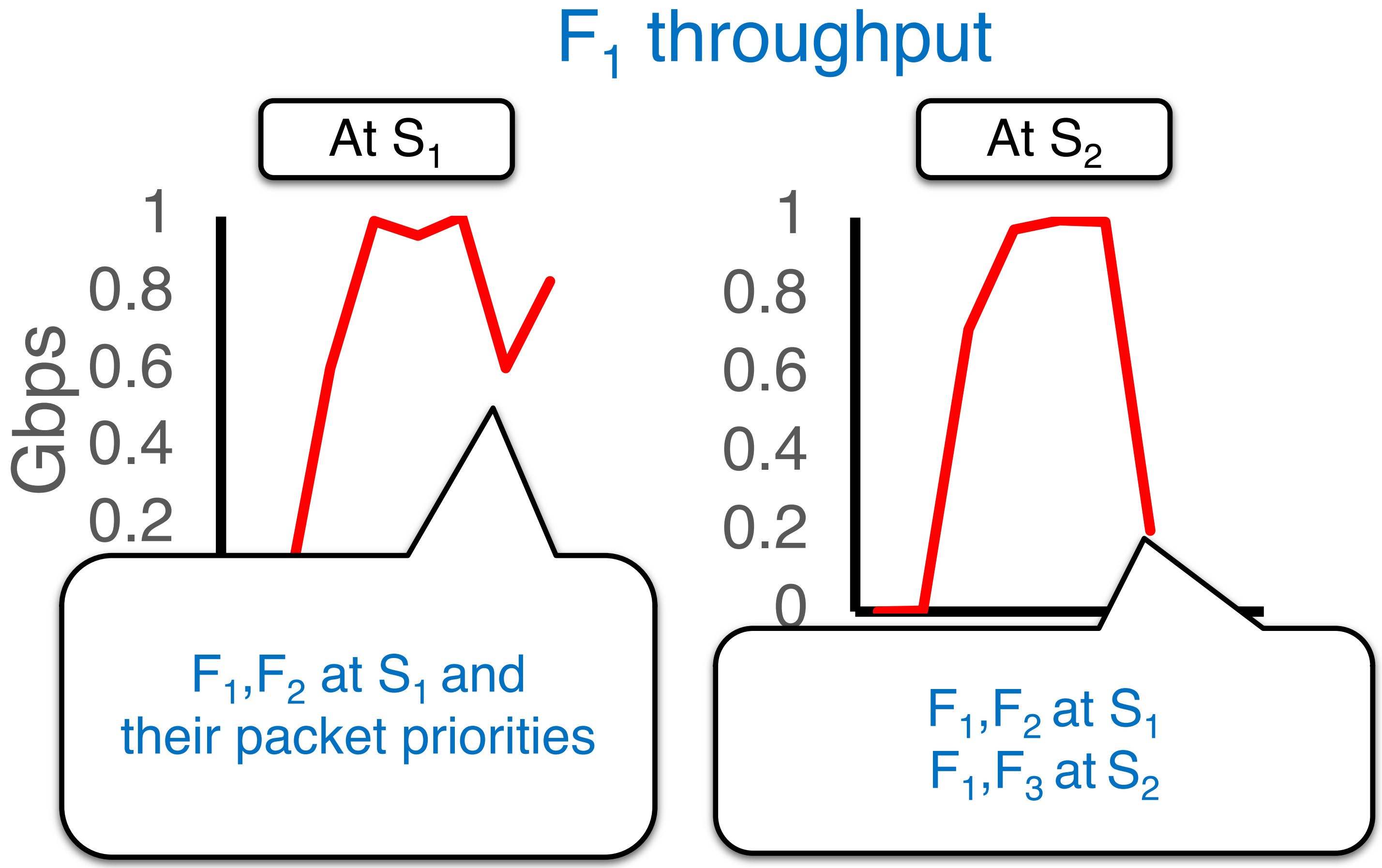
F_1: Low priority
F_2, F_3: High priority



An example: Too many red lights

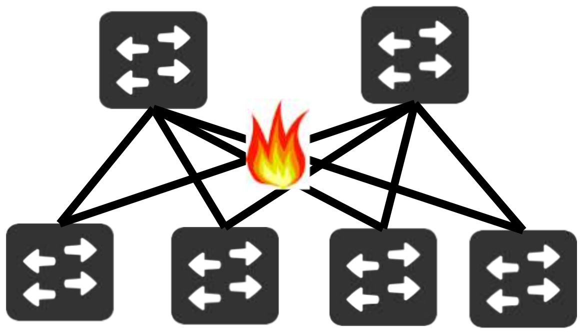


F_1: Low priority
F_2, F_3: High priority



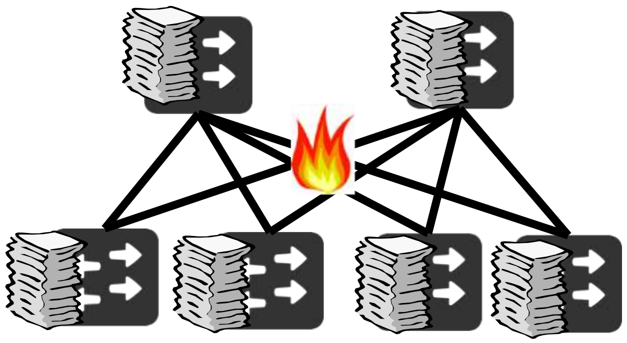
Existing Approaches

In-network techniques



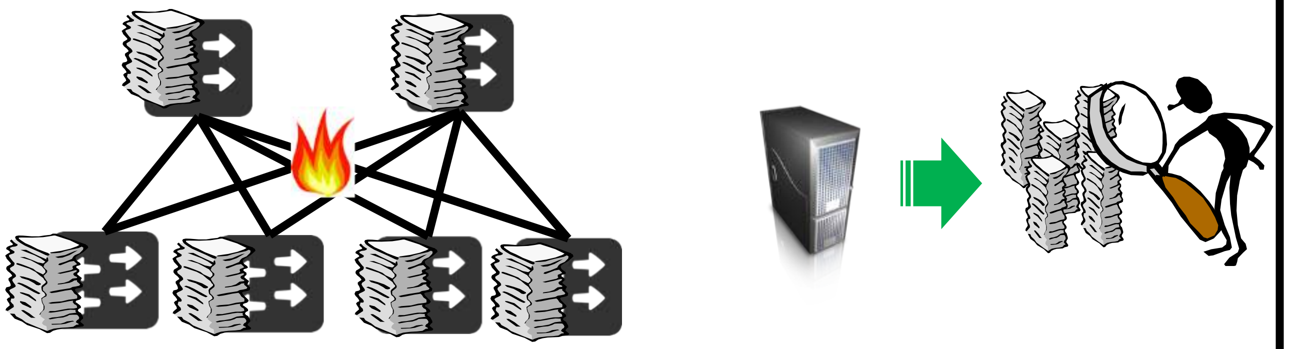
Existing Approaches

In-network techniques



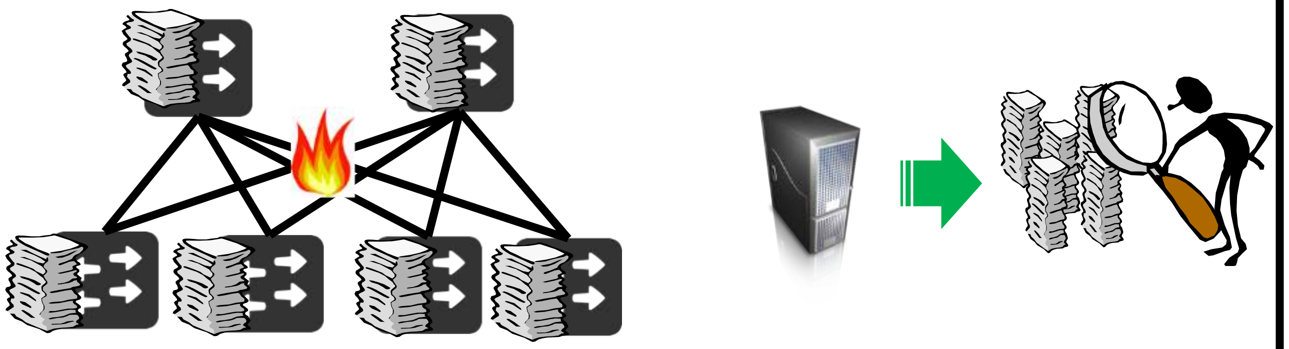
Existing Approaches

In-network techniques



Existing Approaches

In-network techniques



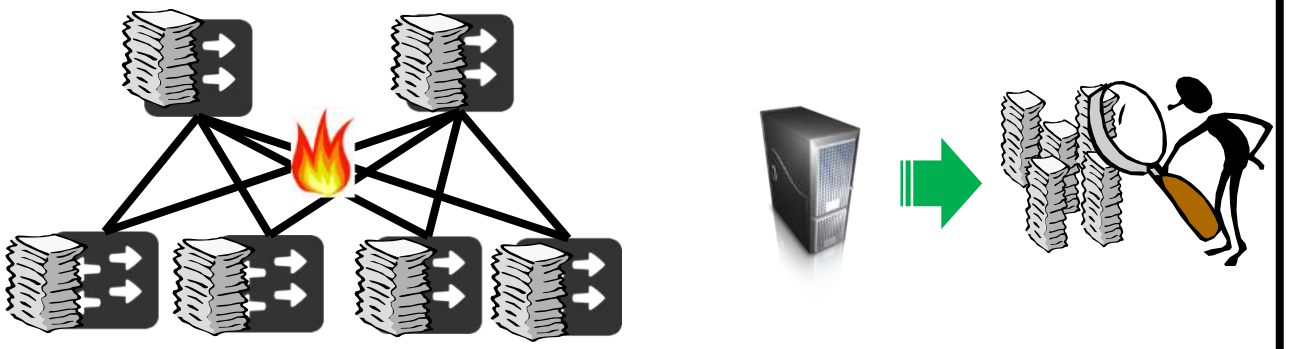
High in-network visibility

Requires more data plane resources

E.g.: Marple, EverFlow, FlowRadar

Existing Approaches

In-network techniques

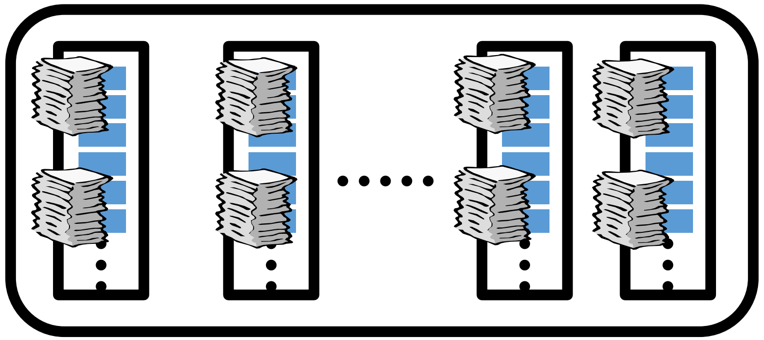


High in-network visibility

Requires more data plane resources

E.g.: Marple, EverFlow, FlowRadar

End-host based techniques



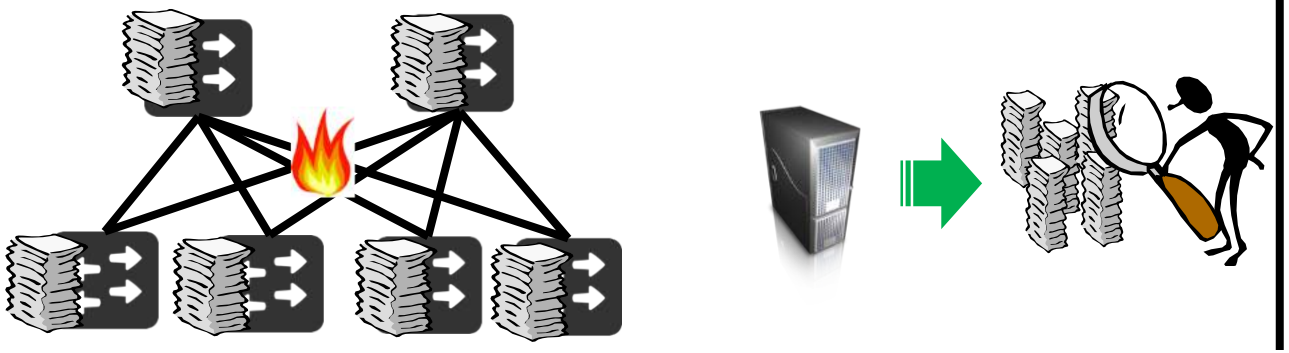
More resources & programmability

Lose network visibility

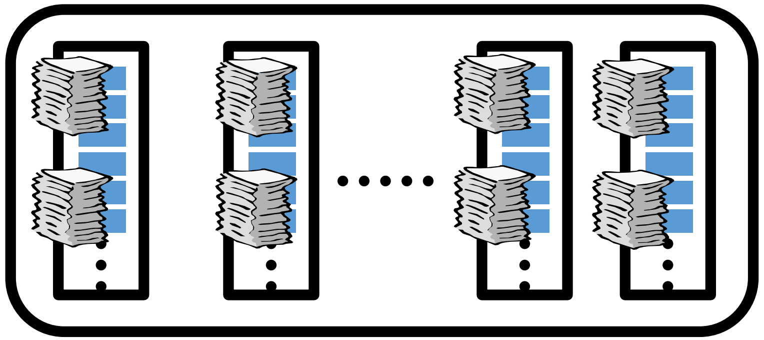
E.g.: PathDump, Trumpet

Existing Approaches

In-network techniques



End-host based techniques



SwitchPointer

High in-network visibility + More resources & programmability

Requires more data plane resources

E.g.: Marple, EverFlow, FlowRadar

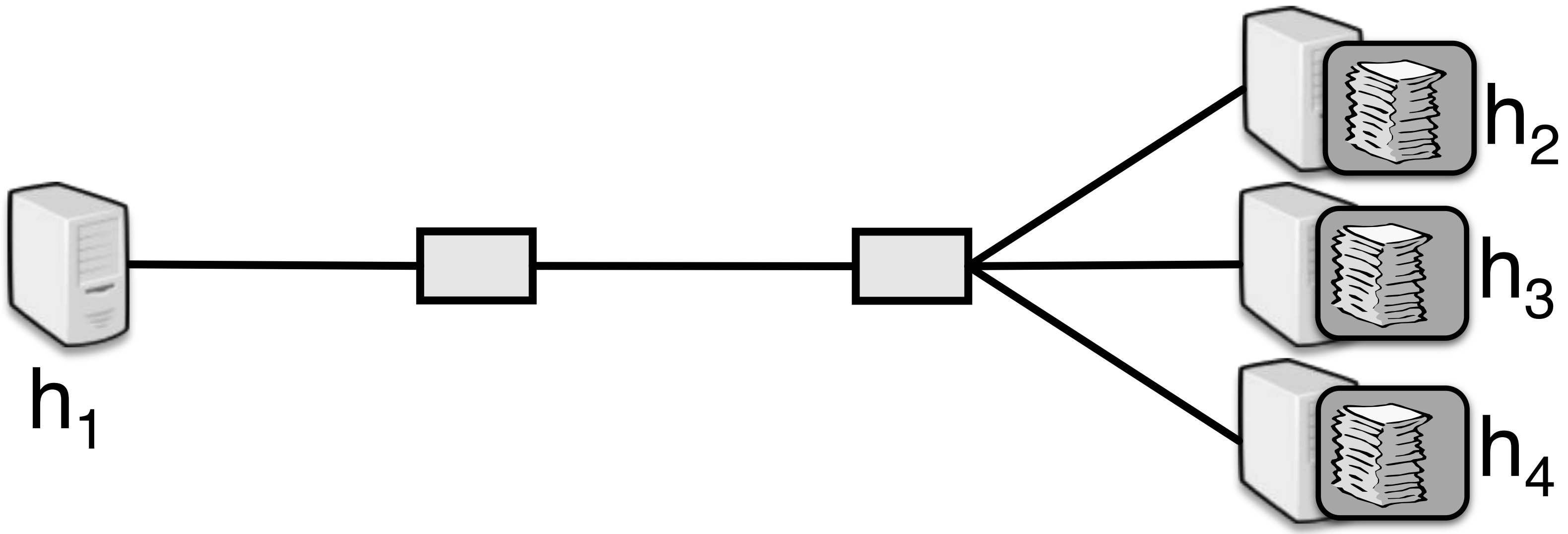
Lose network visibility

E.g.: PathDump, Trumpet

SwitchPointer

Integrates the best of two worlds

Insight: End-hosts collect and monitor telemetry data

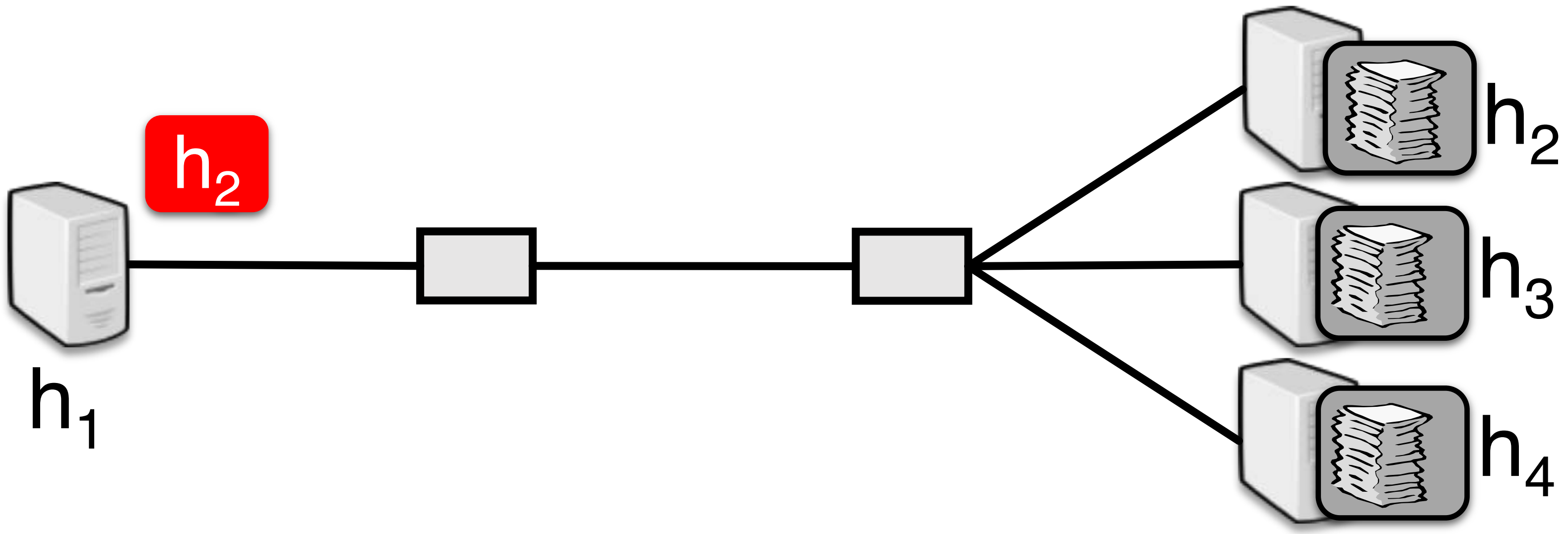


SwitchPointer

Integrates the best of two worlds

Insight: End-hosts collect and monitor telemetry data

New insight: Switch stores the telemetry data

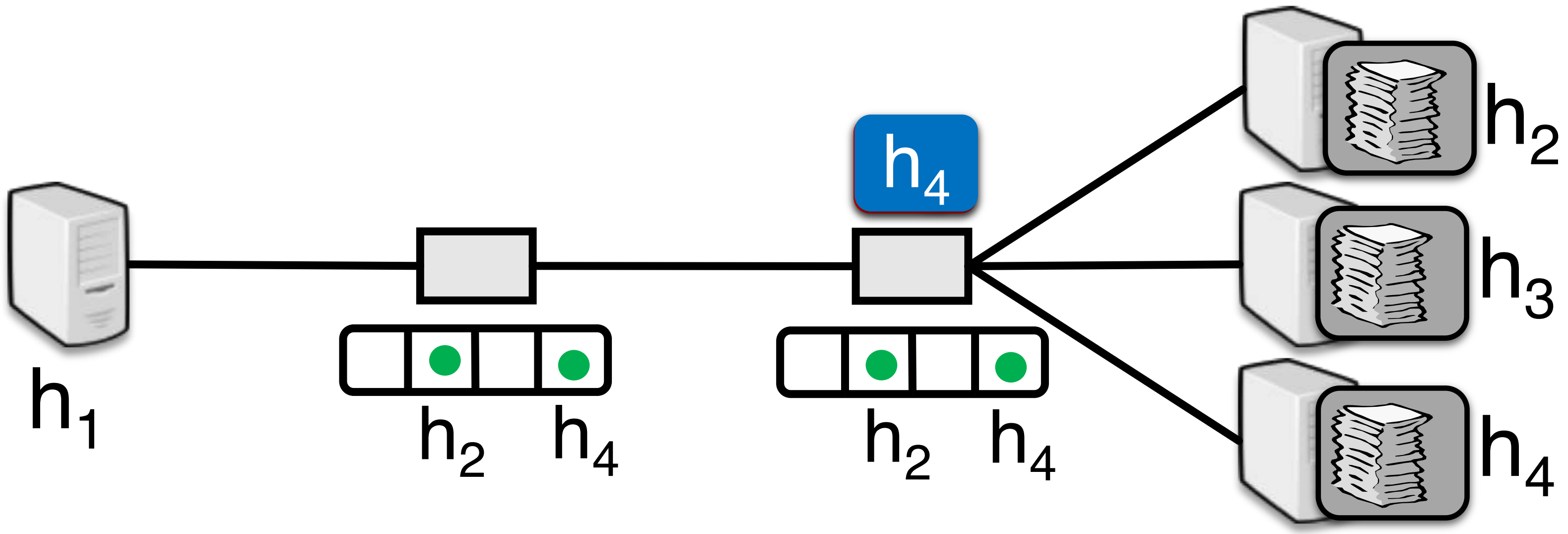


SwitchPointer

Integrates the best of two worlds

Insight: End-hosts collect and monitor telemetry data

New insight: Switch stores the ~~telemetry data~~ ***pointers to end-hosts***

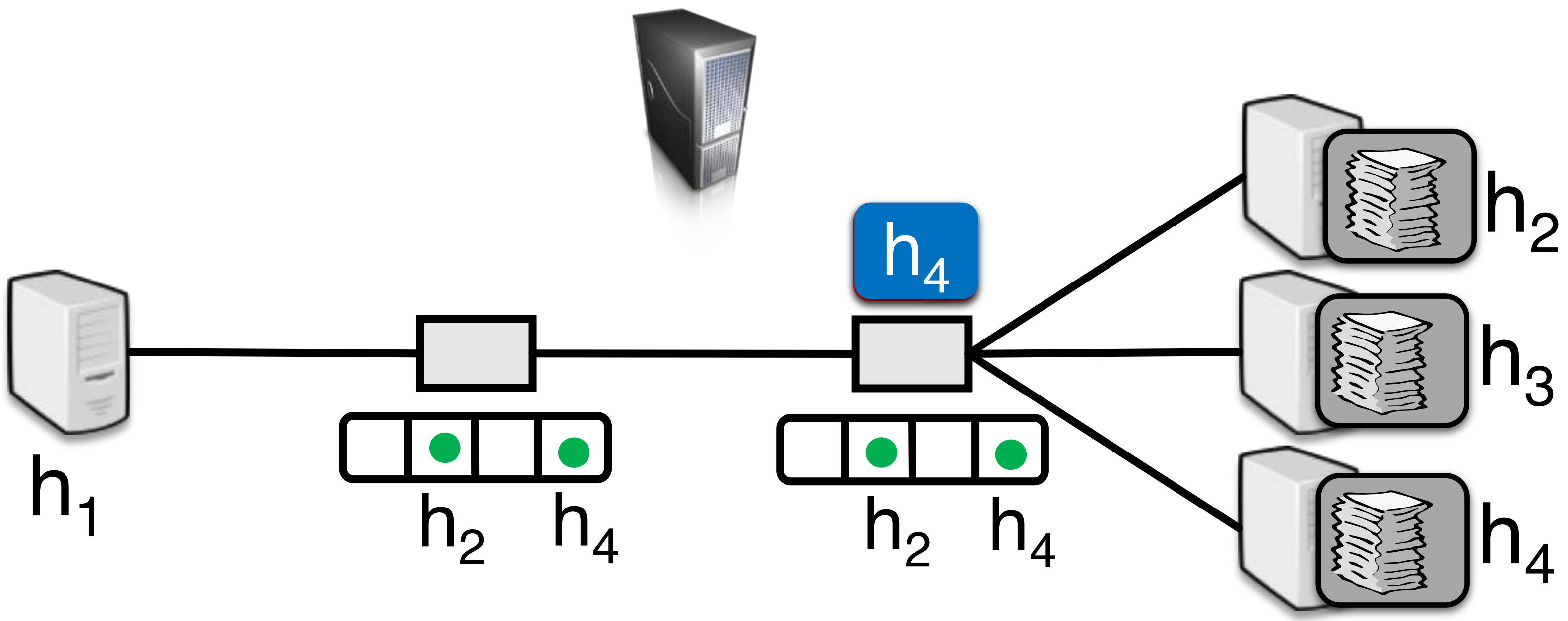


SwitchPointer

Integrates the best of two worlds

Insight: End-hosts collect and monitor telemetry data

New insight: Switch stores the ~~telemetry data~~ ***pointers to end-hosts***

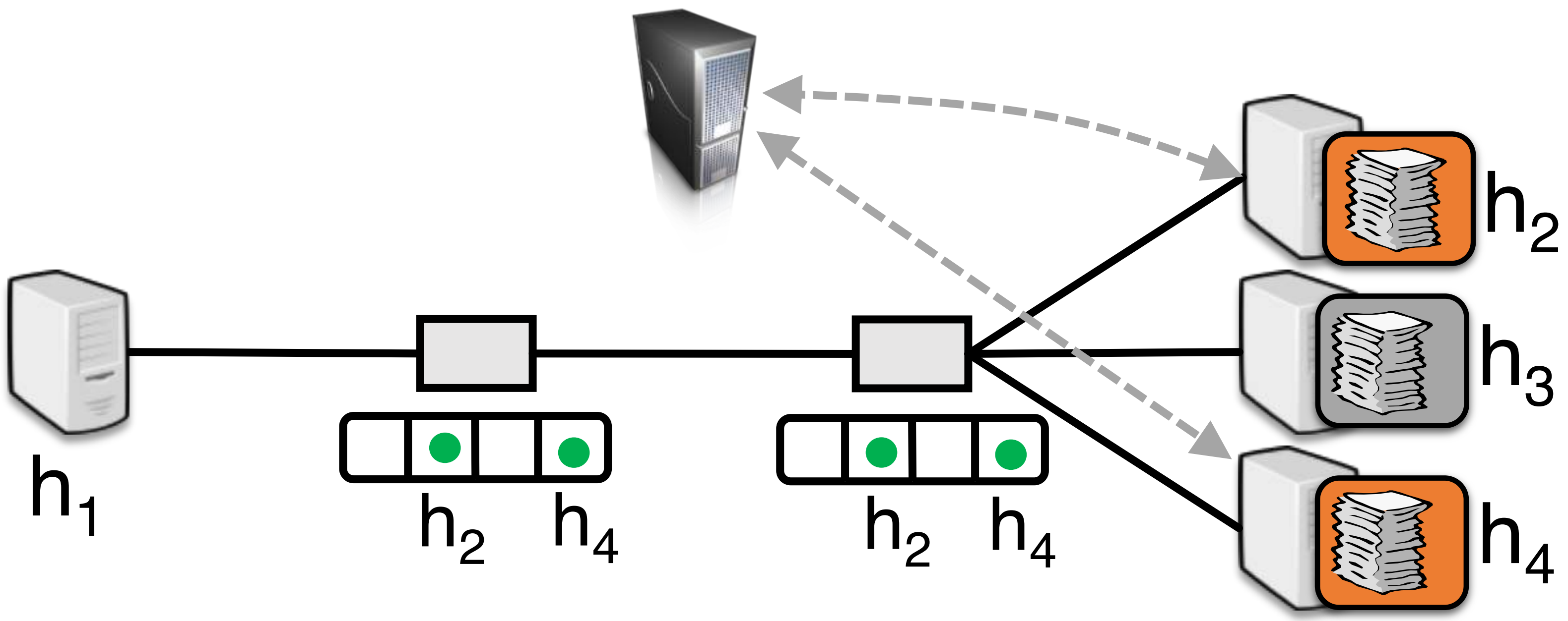


SwitchPointer

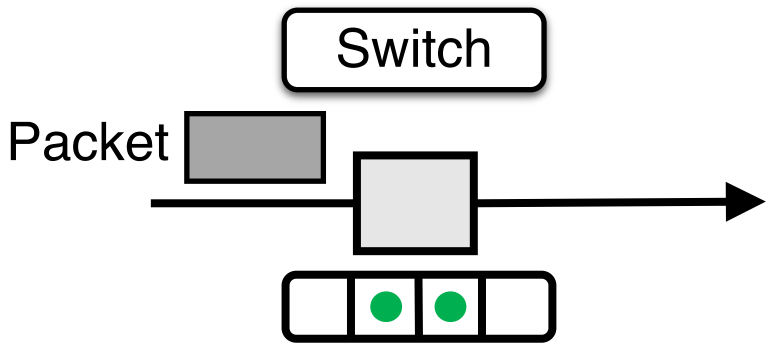
Integrates the best of two worlds

Insight: End-hosts collect and monitor telemetry data

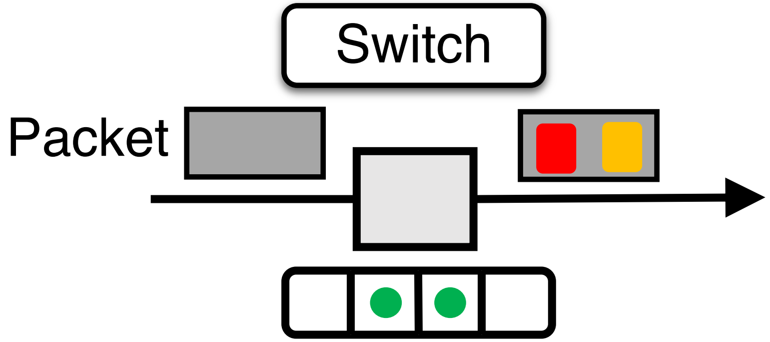
New insight: Switch stores the ~~telemetry data~~ ***pointers to end-hosts***



SwitchPointer in a nutshell

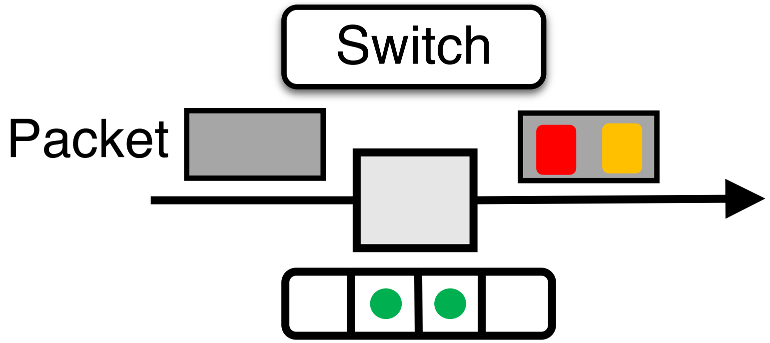


SwitchPointer in a nutshell

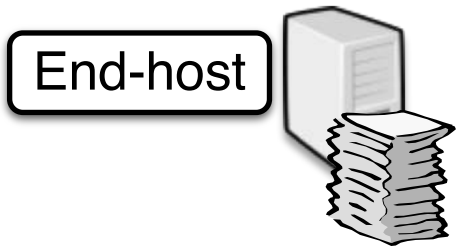


- Divides time into epochs
- Maintains per-epoch pointer to all end-hosts
- Embeds linkID(■) and epochID (■)

SwitchPointer in a nutshell

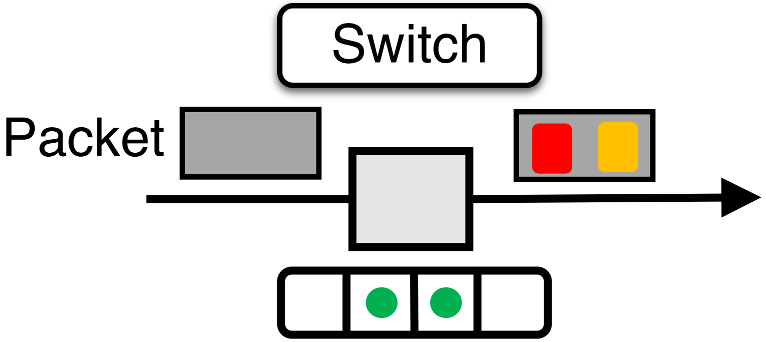


- Divides time into epochs
- Maintains per-epoch pointer to all end-hosts
- Embeds linkID(■) and epochID (■)

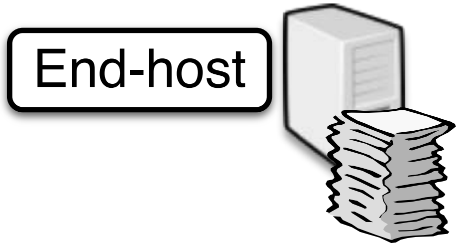


- Collect and monitor telemetry data
- Provides query service to filter telemetry data

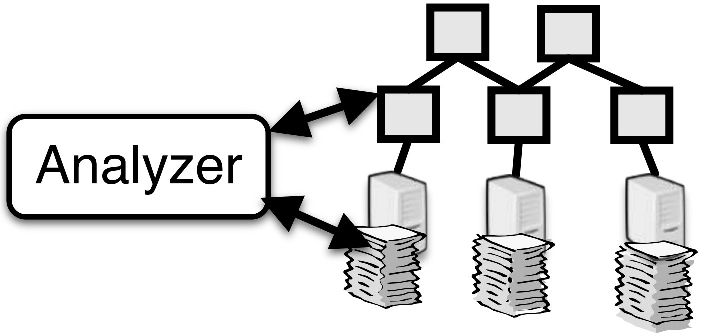
SwitchPointer in a nutshell



- Divides time into epochs
- Maintains per-epoch pointer to all end-hosts
- Embeds linkID(■) and epochID (■)

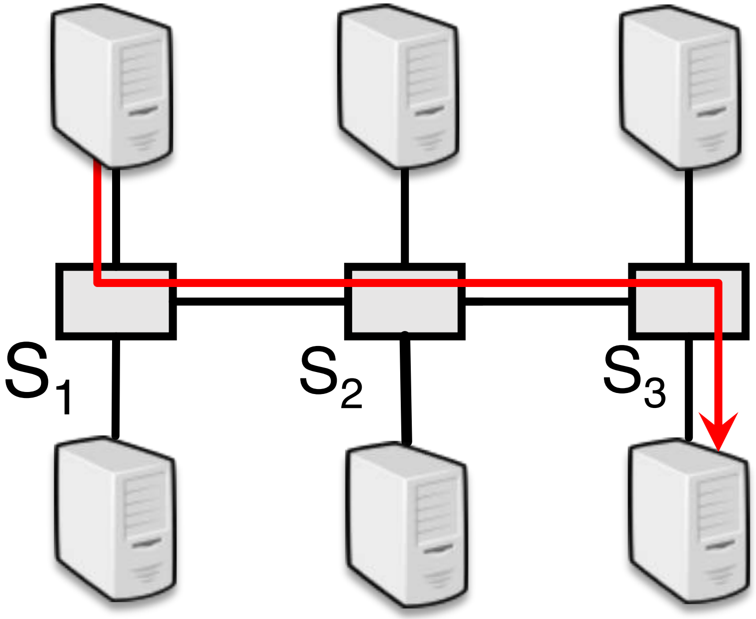


- Collect and monitor telemetry data
- Provides query service to filter telemetry data

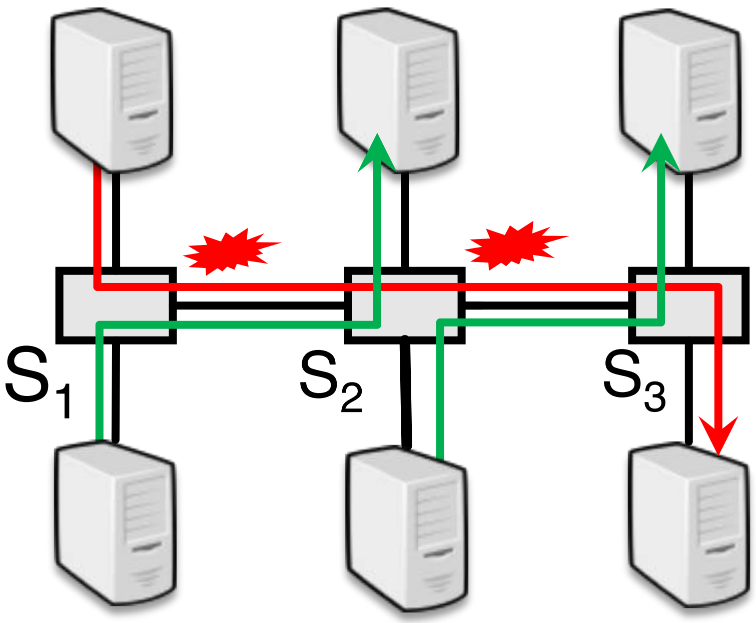


- Uses pointers at switches
- Locates the data necessary for debugging

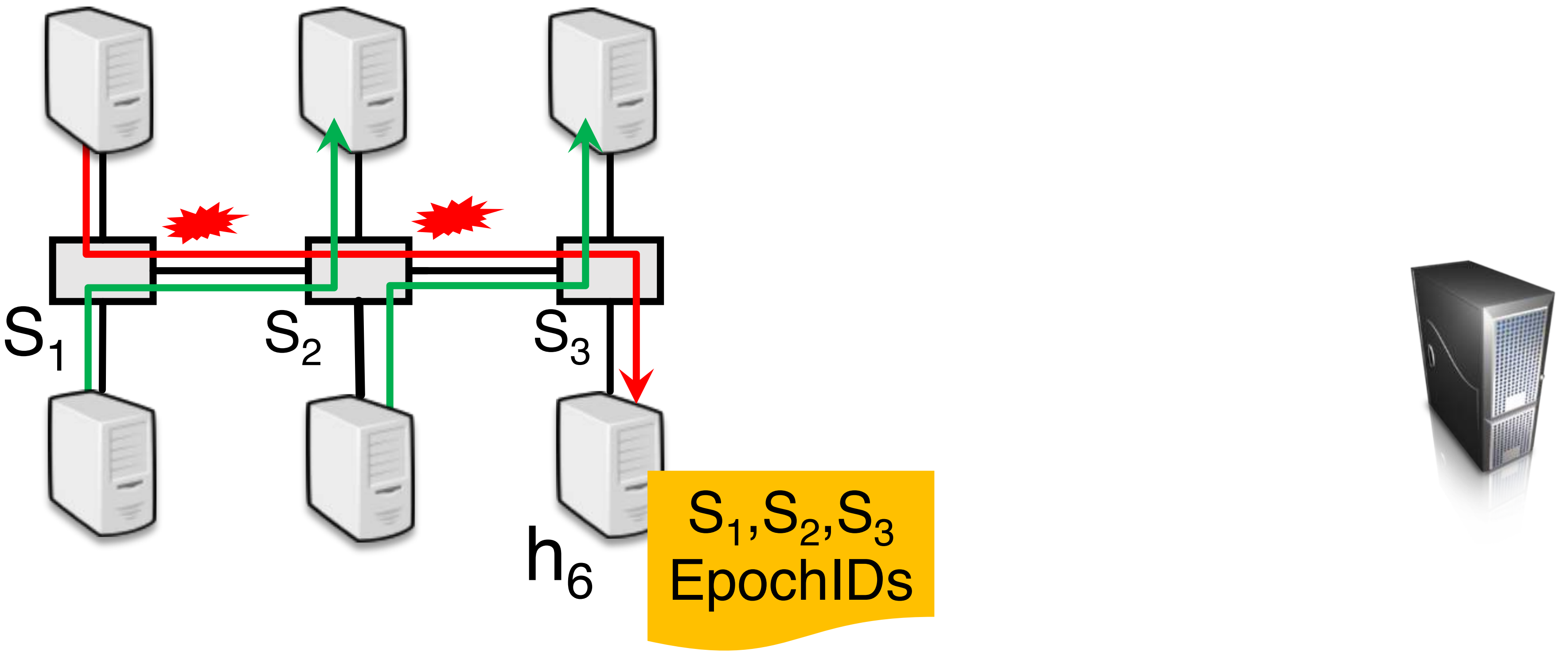
Let’s revisit “Too many red lights” problem



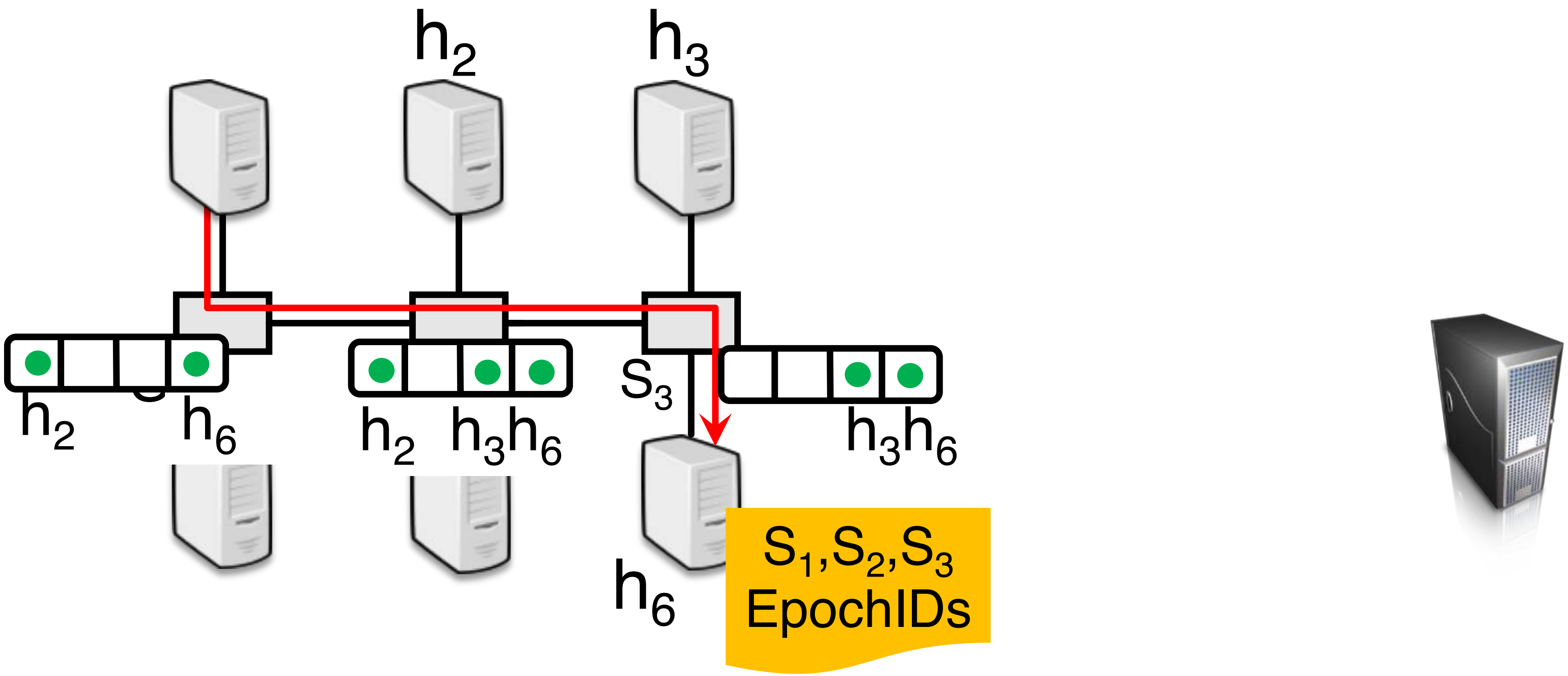
Let's revisit "Too many red lights" problem



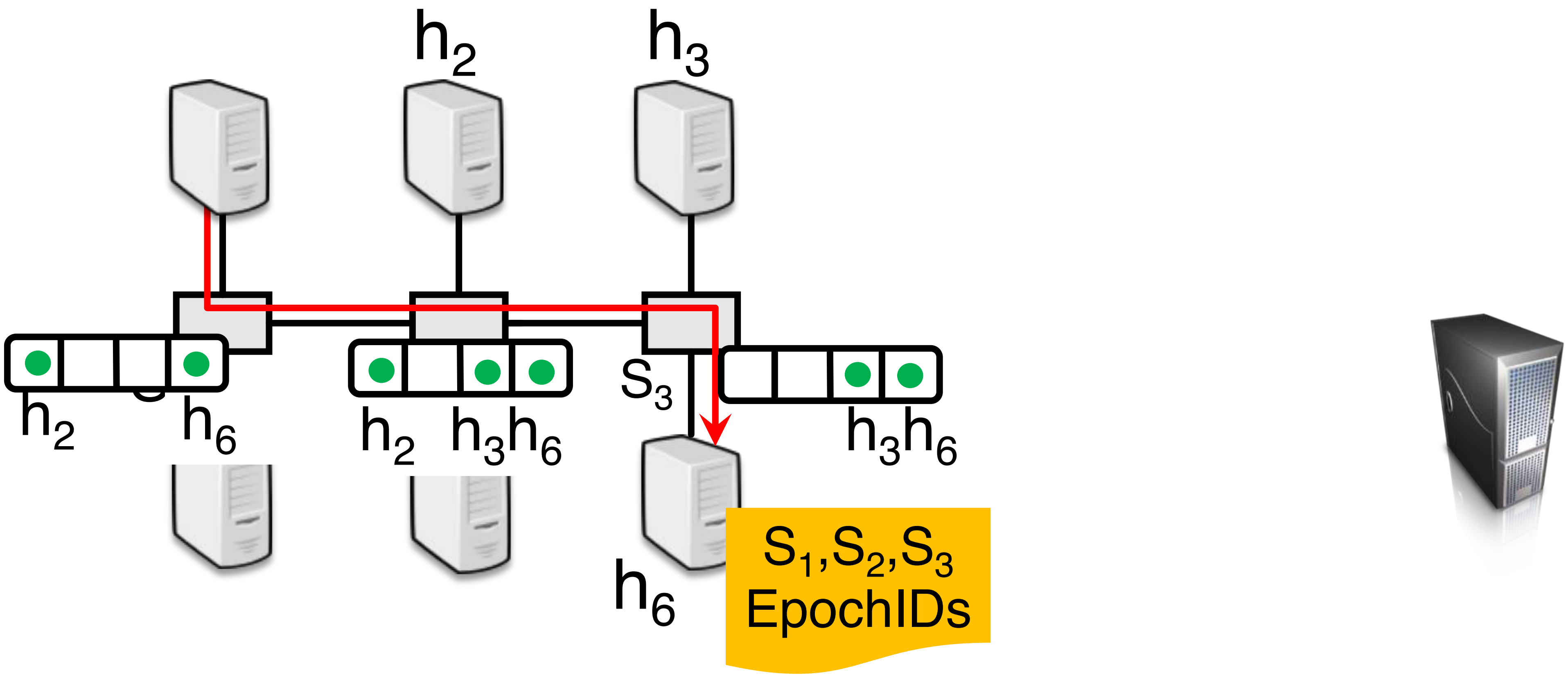
Let's revisit "Too many red lights" problem



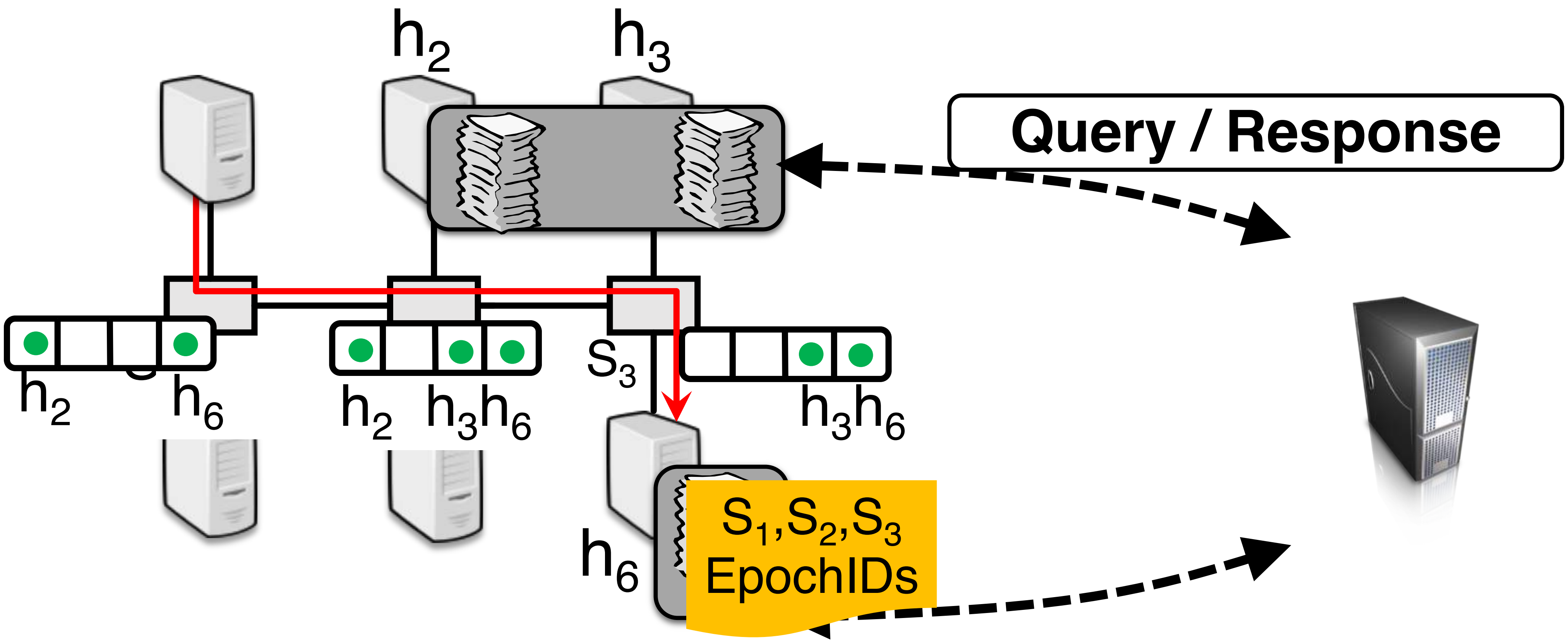
Let's revisit "Too many red lights" problem



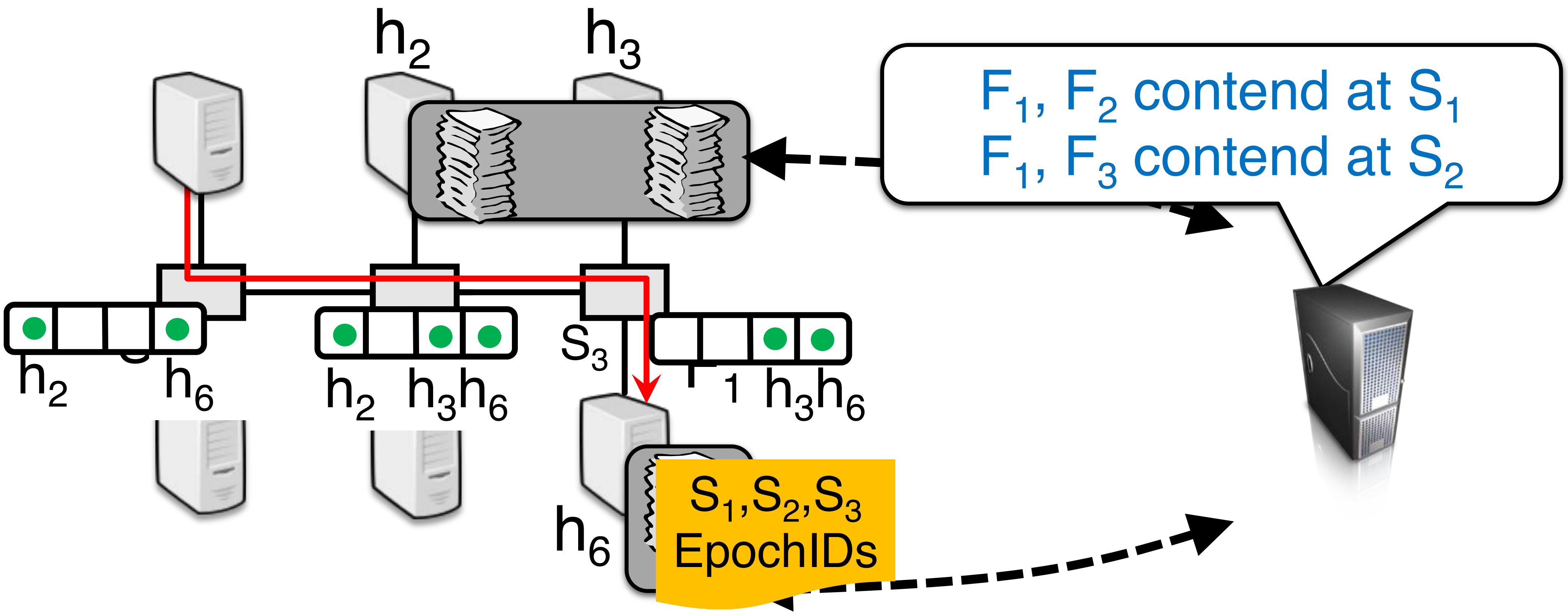
Let's revisit "Too many red lights" problem



Let’s revisit “Too many red lights” problem



Let's revisit "Too many red lights" problem



SwitchPointer: Four technical challenges

- How to decide the right epoch size?
- How to efficiently maintain pointers?
- How to efficiently embed telemetry data?
- How to handle asynchronous clocks?

SwitchPointer design

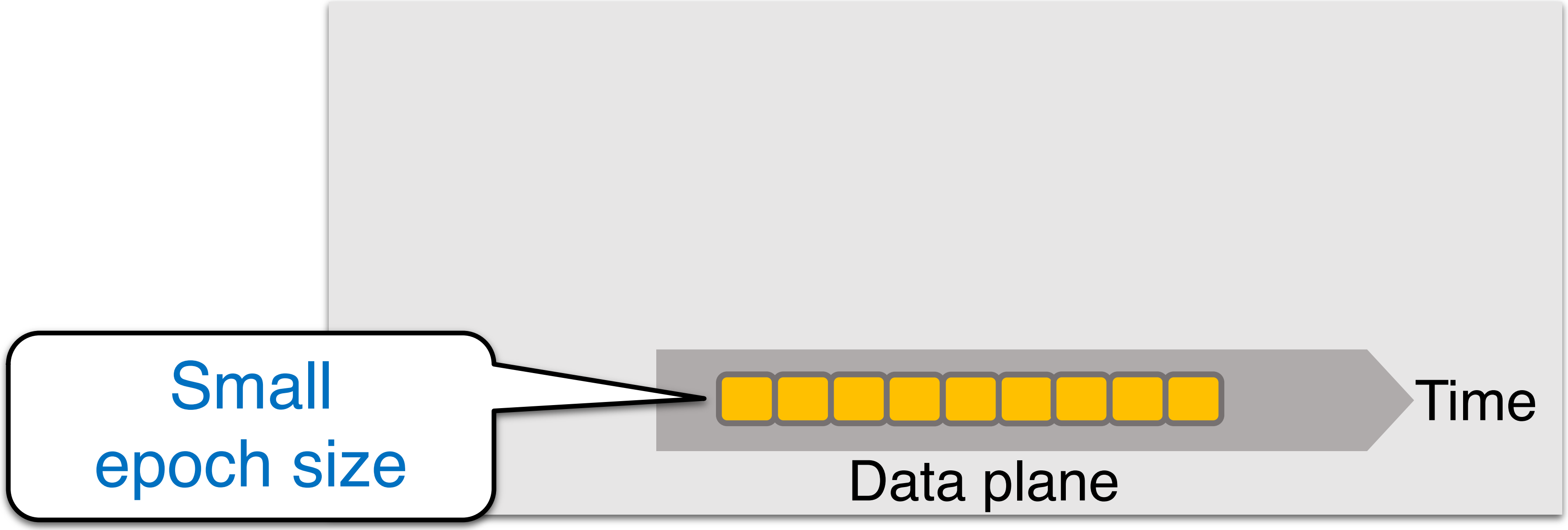
Data structure for pointers

Tradeoff between memory and bandwidth, and system efficiency

SwitchPointer design

Data structure for pointers

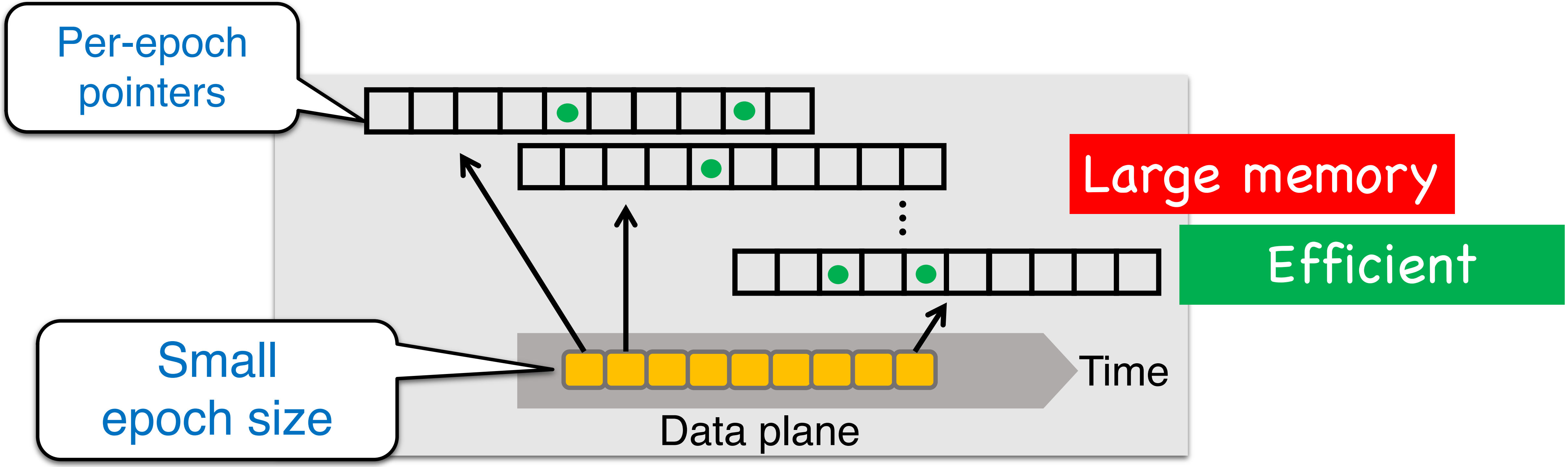
Tradeoff between memory and bandwidth, and system efficiency



SwitchPointer design

Data structure for pointers

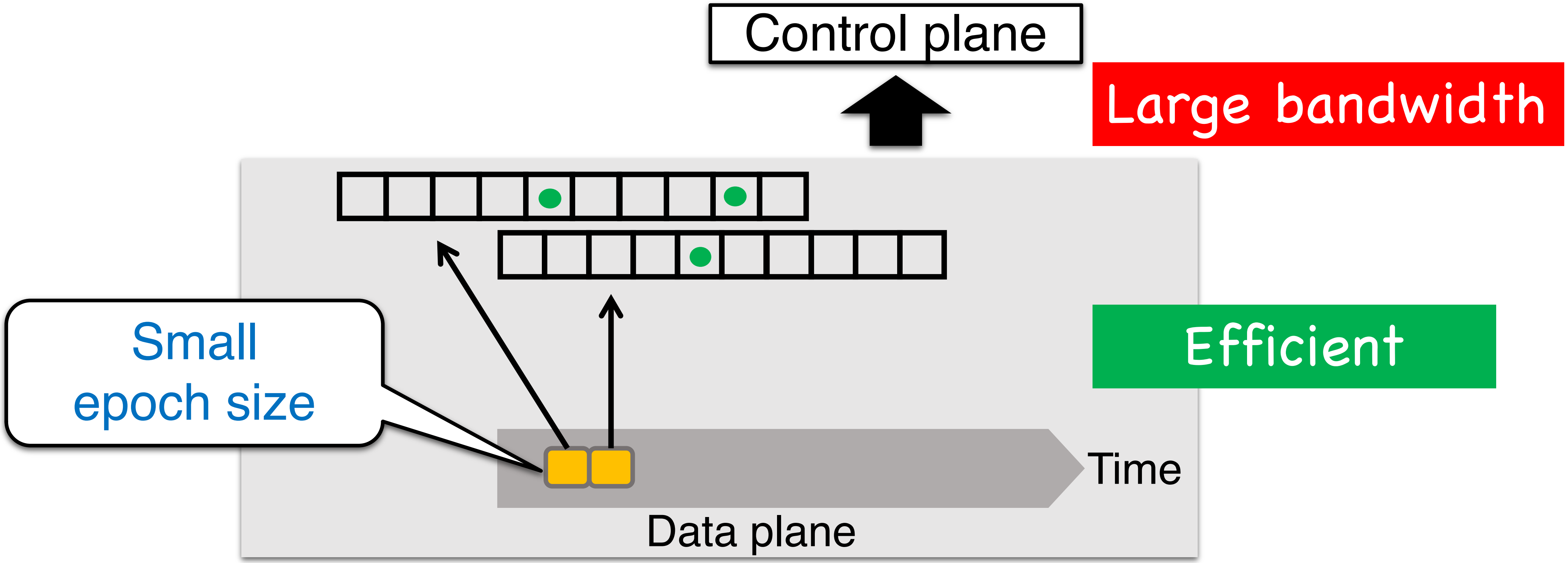
Tradeoff between memory and bandwidth, and system efficiency



SwitchPointer design

Data structure for pointers

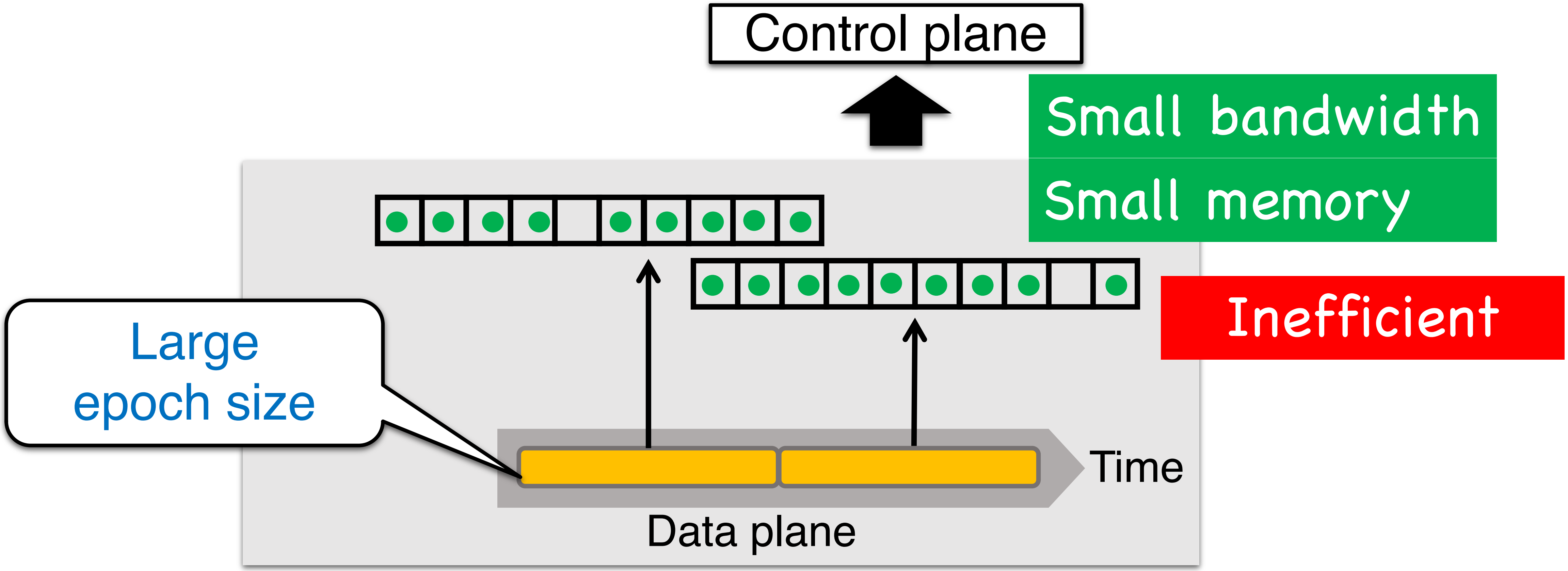
Tradeoff between memory and bandwidth, and system efficiency



SwitchPointer design

Data structure for pointers

Tradeoff between memory and bandwidth, and system efficiency



SwitchPointer design

Our solution: Hierarchical data structure for pointers

Each subsequent level has epochs with exponentially larger time scales

SwitchPointer design

Our solution: Hierarchical data structure for pointers

Each subsequent level has epochs with exponentially larger time scales

Epoch size = α

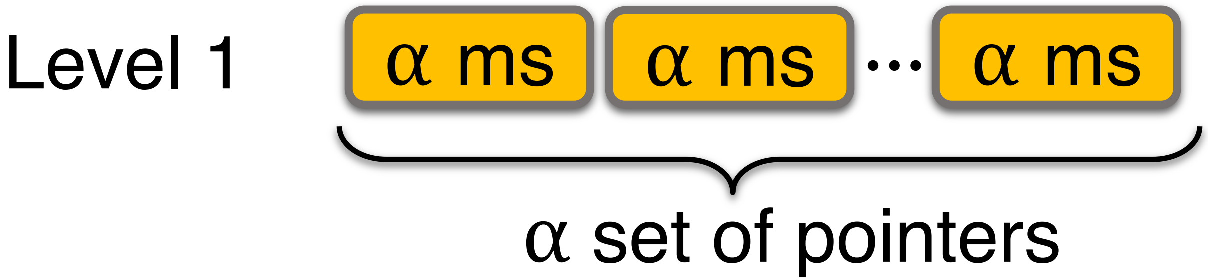
Level 1

SwitchPointer design

Our solution: Hierarchical data structure for pointers

Each subsequent level has epochs with exponentially larger time scales

Epoch size = α

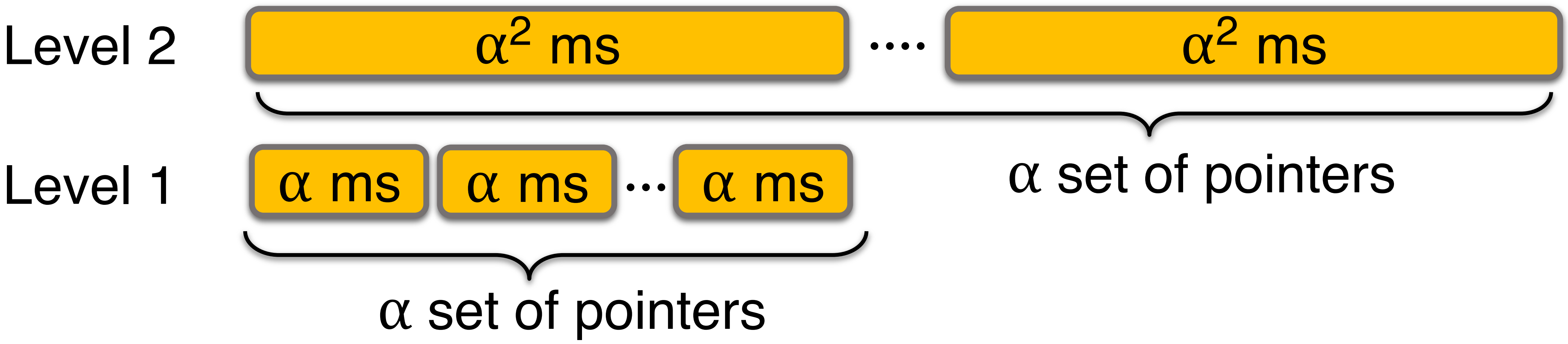


SwitchPointer design

Our solution: Hierarchical data structure for pointers

Each subsequent level has epochs with exponentially larger time scales

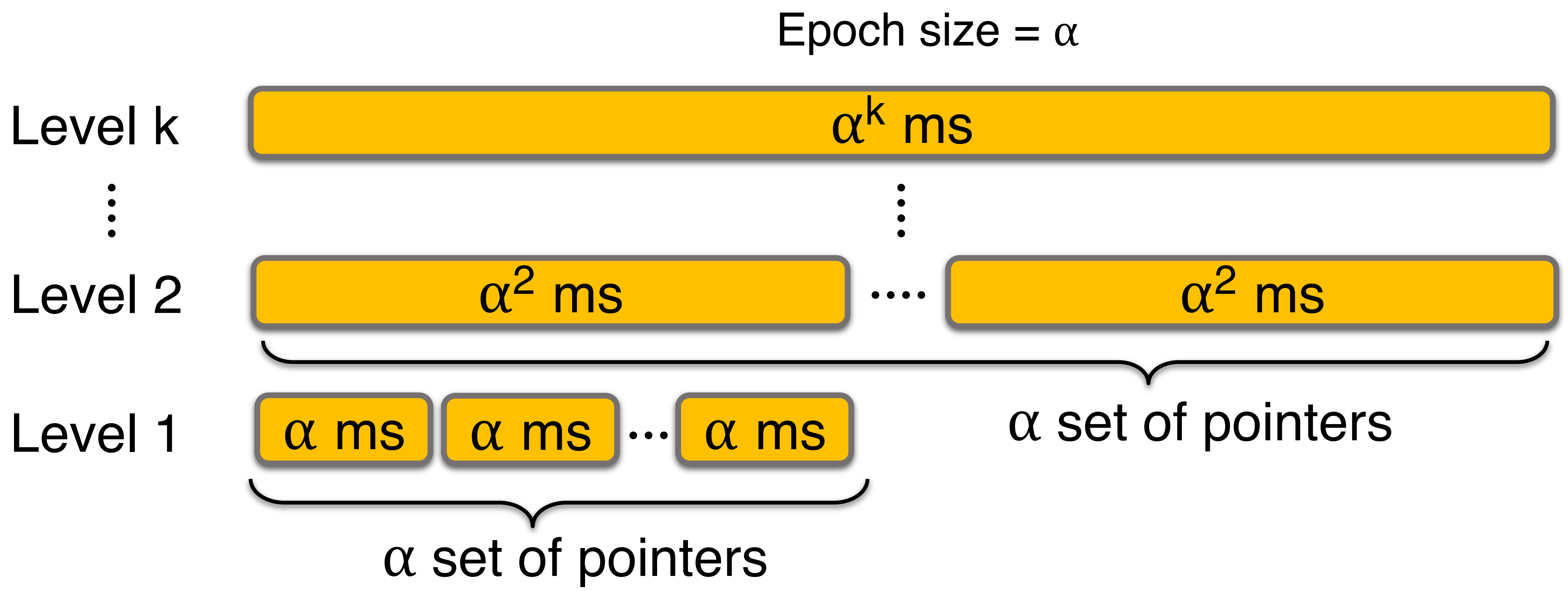
Epoch size = α



SwitchPointer design

Our solution: Hierarchical data structure for pointers

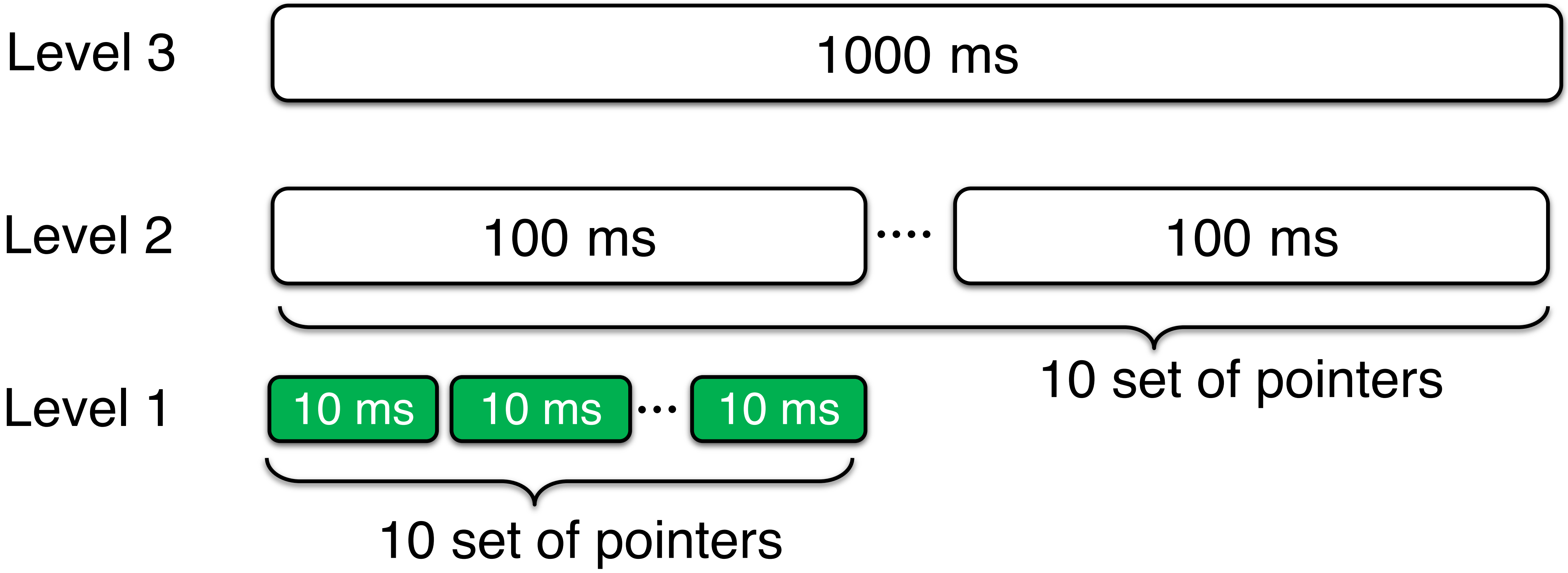
Each subsequent level has epochs with exponentially larger time scales



SwitchPointer design

Our solution: Hierarchical data structure for pointers

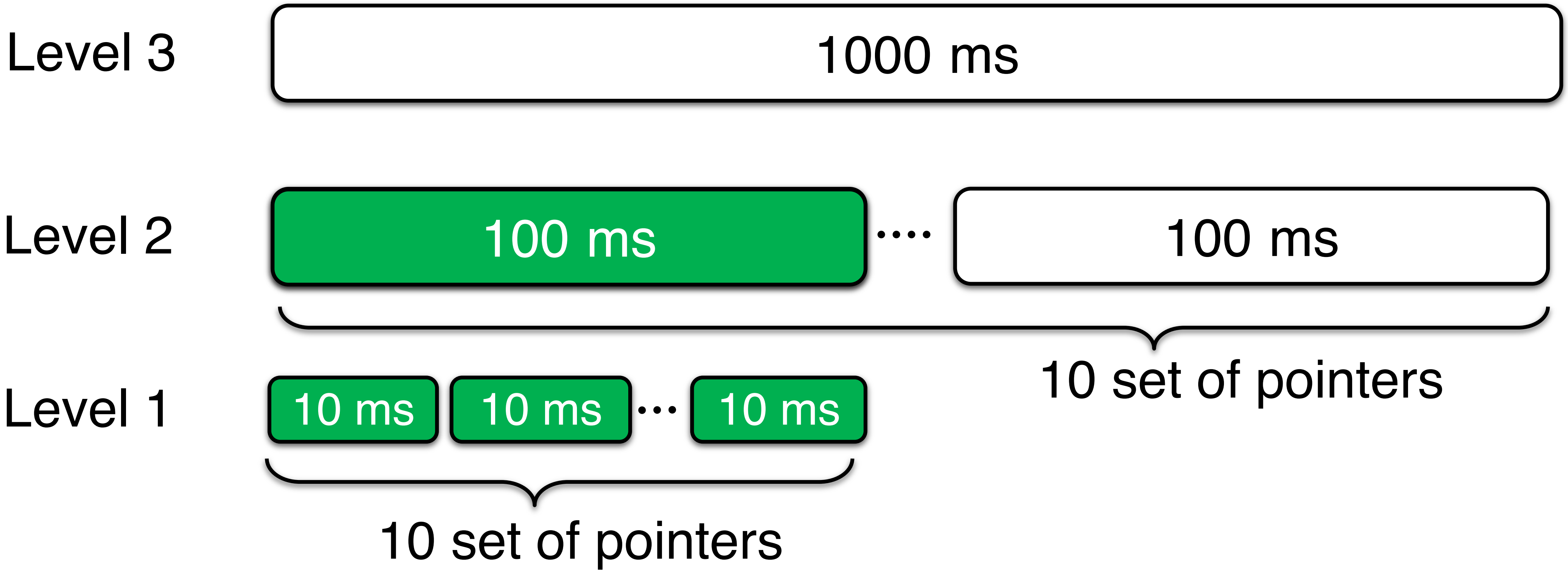
$\alpha = 10$ ms $\quad k = 3$



SwitchPointer design

Our solution: Hierarchical data structure for pointers

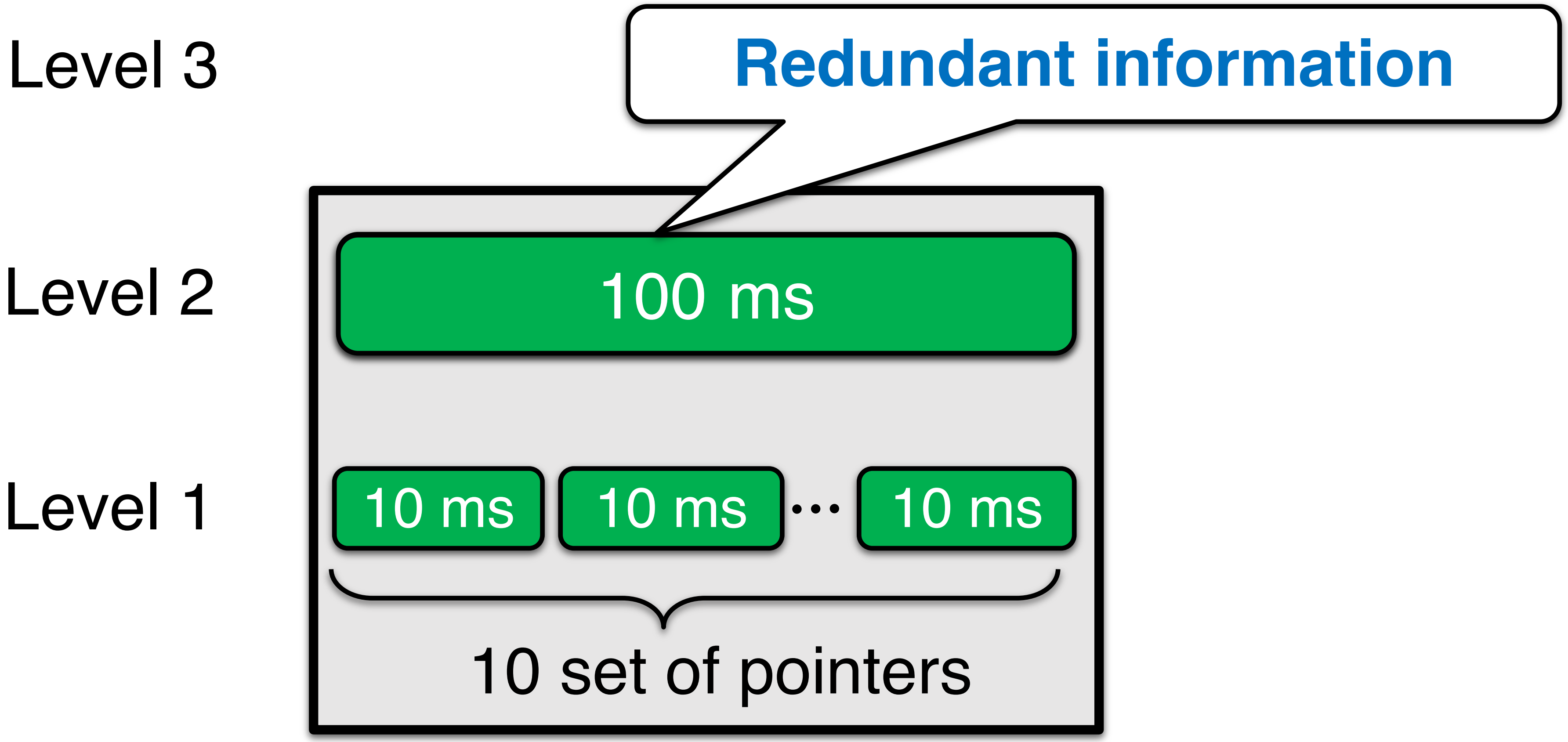
$\alpha = 10$ ms $k = 3$



SwitchPointer design

Our solution: Hierarchical data structure for pointers

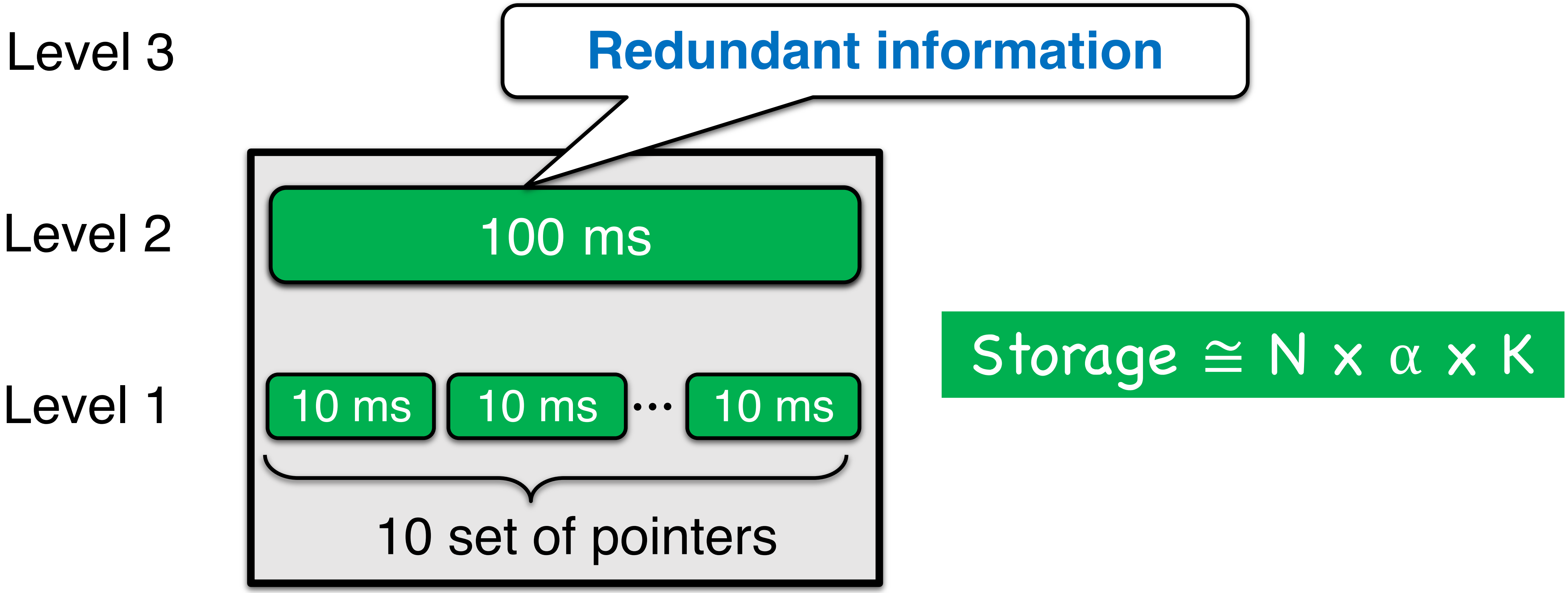
$\alpha = 10$ ms $k = 3$



SwitchPointer design

Our solution: Hierarchical data structure for pointers

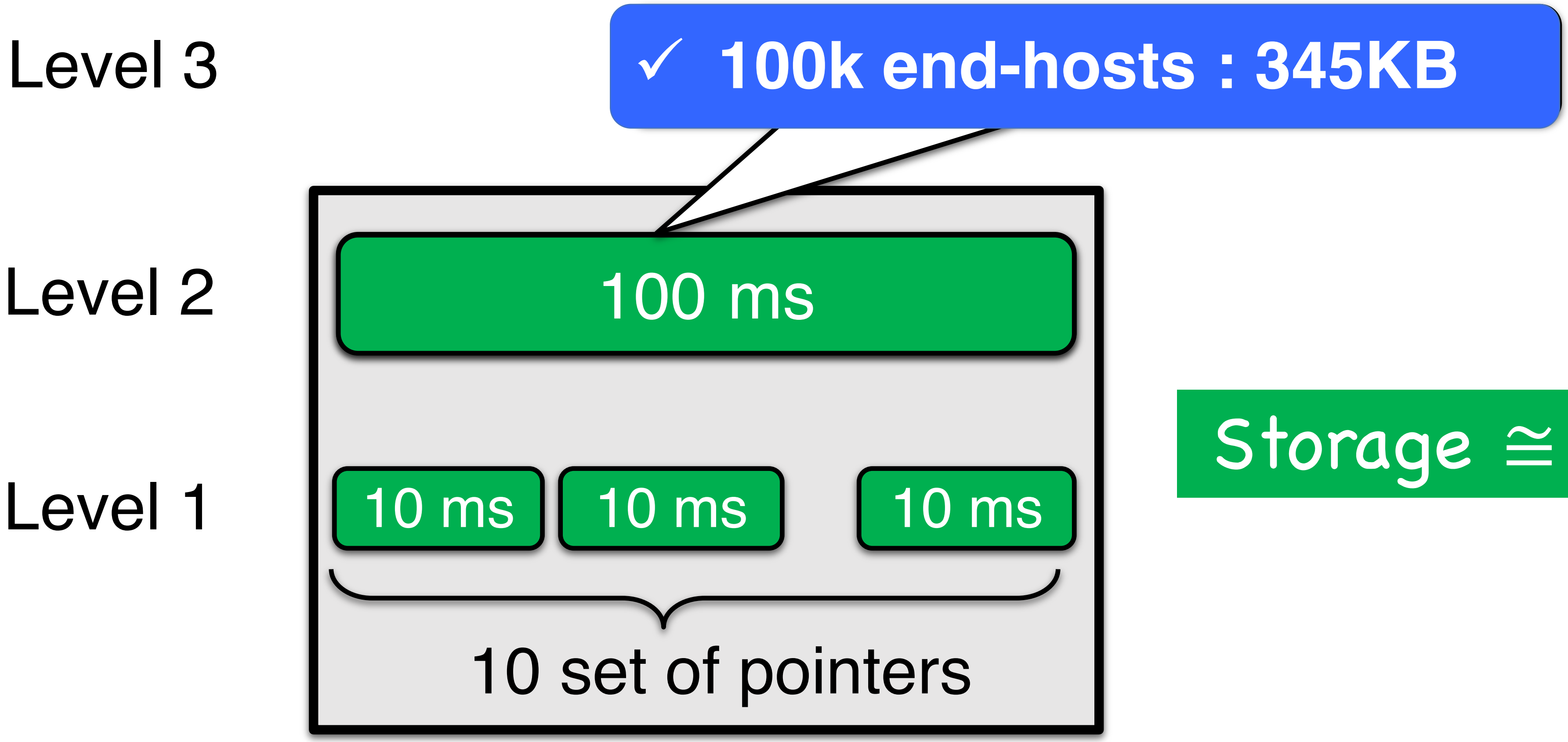
α = 10 ms   k = 3



SwitchPointer design

Our solution: Hierarchical data structure for pointers

α = 10 ms   k = 3

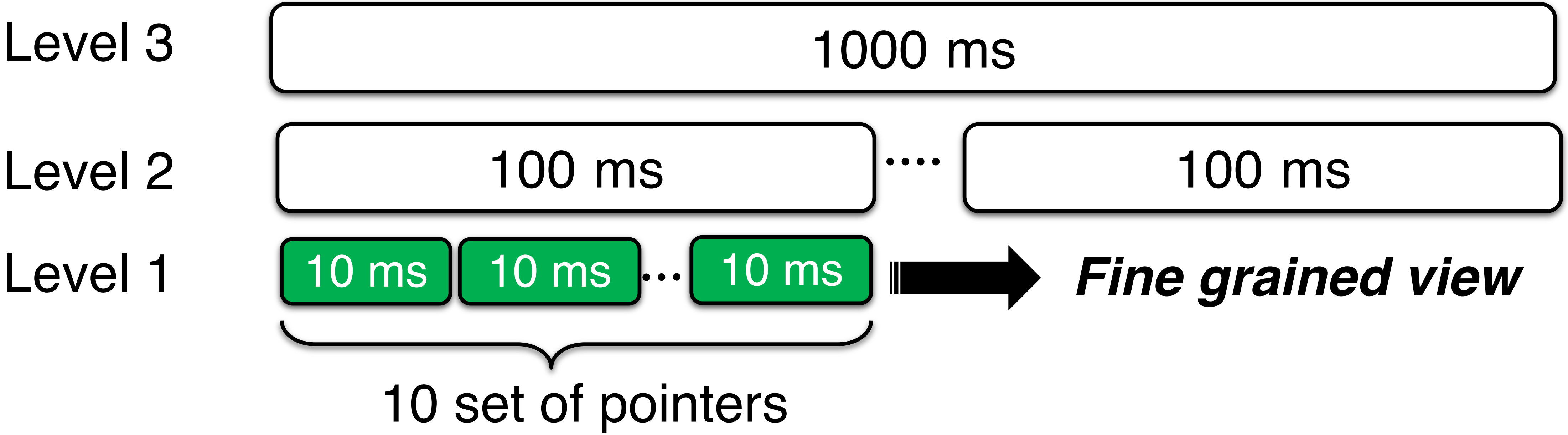


Storage ≅ N x α x K

SwitchPointer design

Our solution: Hierarchical data structure for pointers

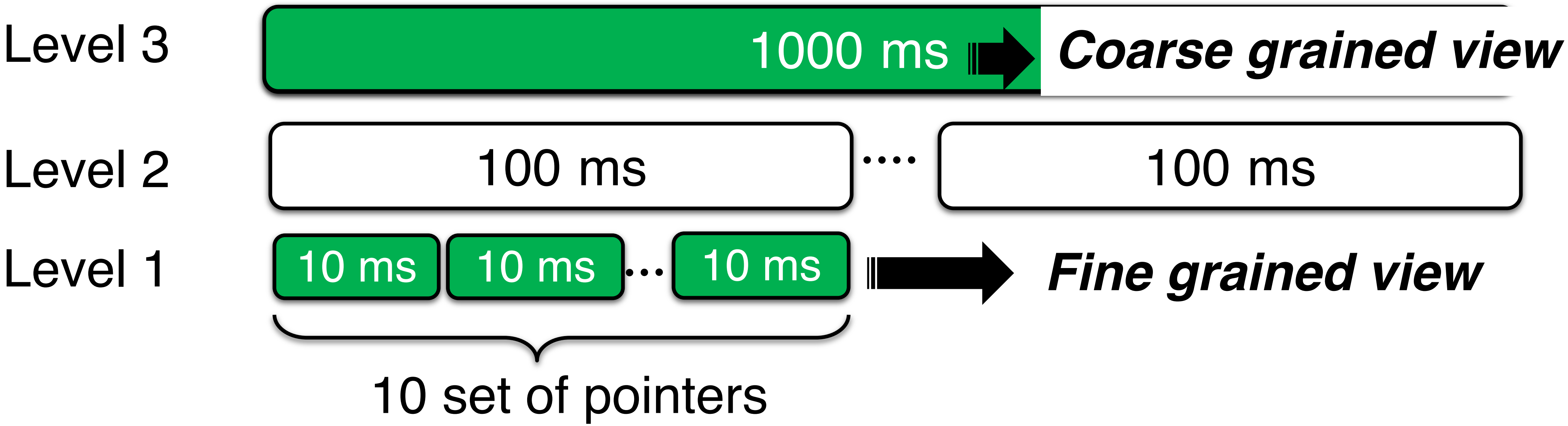
$\alpha = 10$ ms $\quad k = 3$



SwitchPointer design

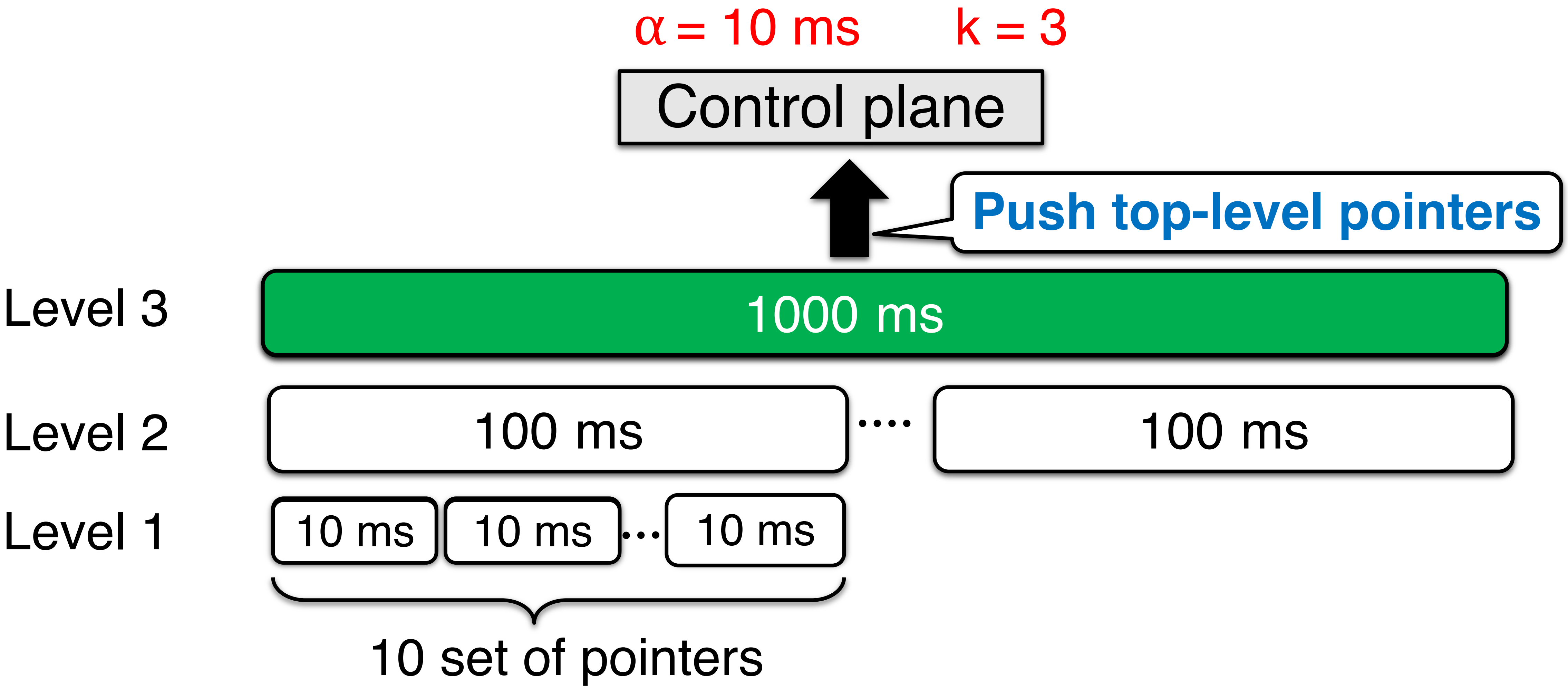
Our solution: Hierarchical data structure for pointers

$\alpha = 10$ ms $k = 3$



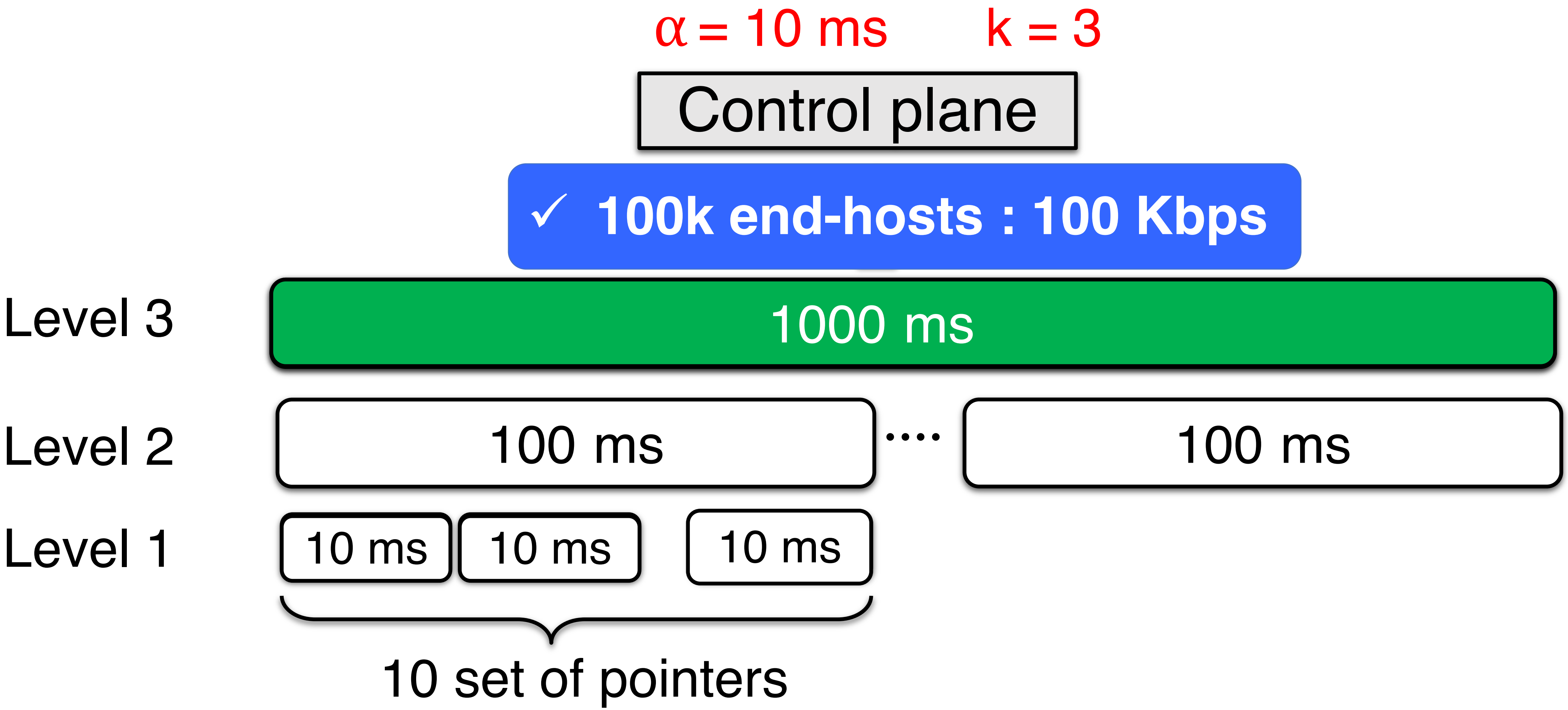
SwitchPointer design

Our solution: Hierarchical data structure for pointers



SwitchPointer design

Our solution: Hierarchical data structure for pointers



SwitchPointer: Four technical challenges

- How to decide the right epoch size?
- How to efficiently maintain pointers?
- How to efficiently embed telemetry data?
- How to handle asynchronous clocks?

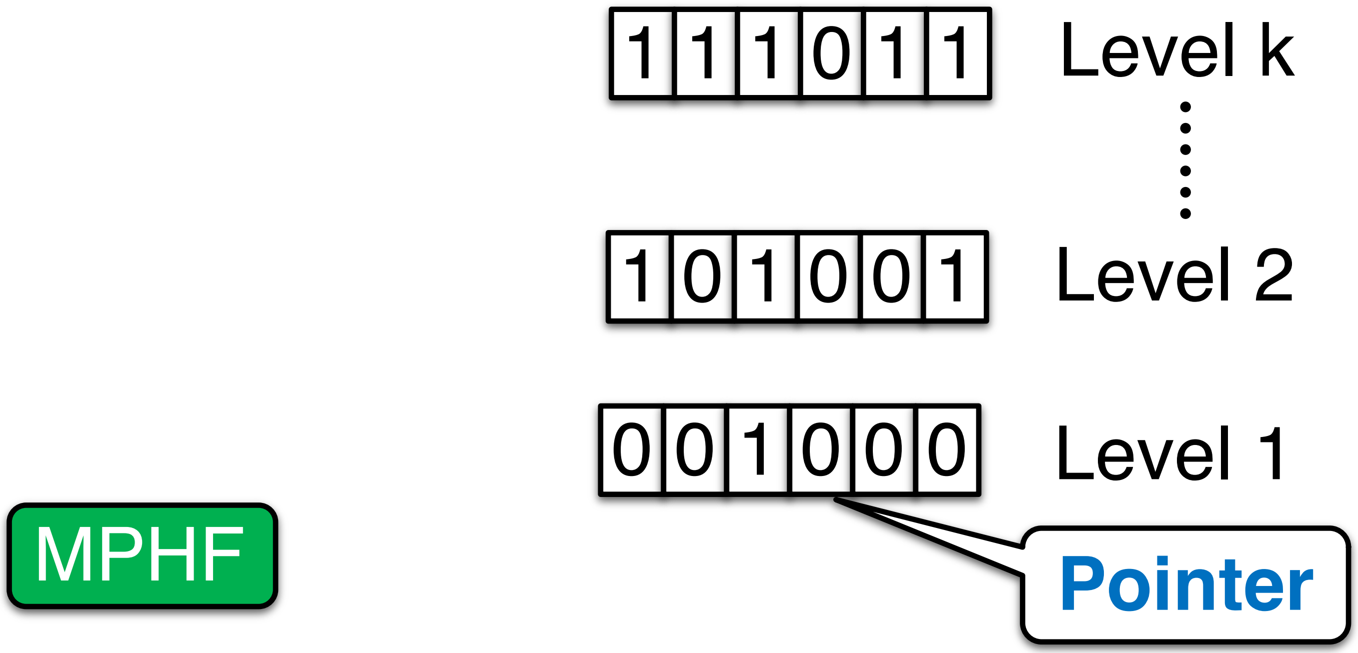
Minimal Perfect Hash Functions

- Maps distinct keys (dest IPs) to a set of integers
- No hash collisions
- 2.1 bits of storage per end-host
- Construction time is large

SwitchPointer design

Maintaining pointers in the hierarchical data structure

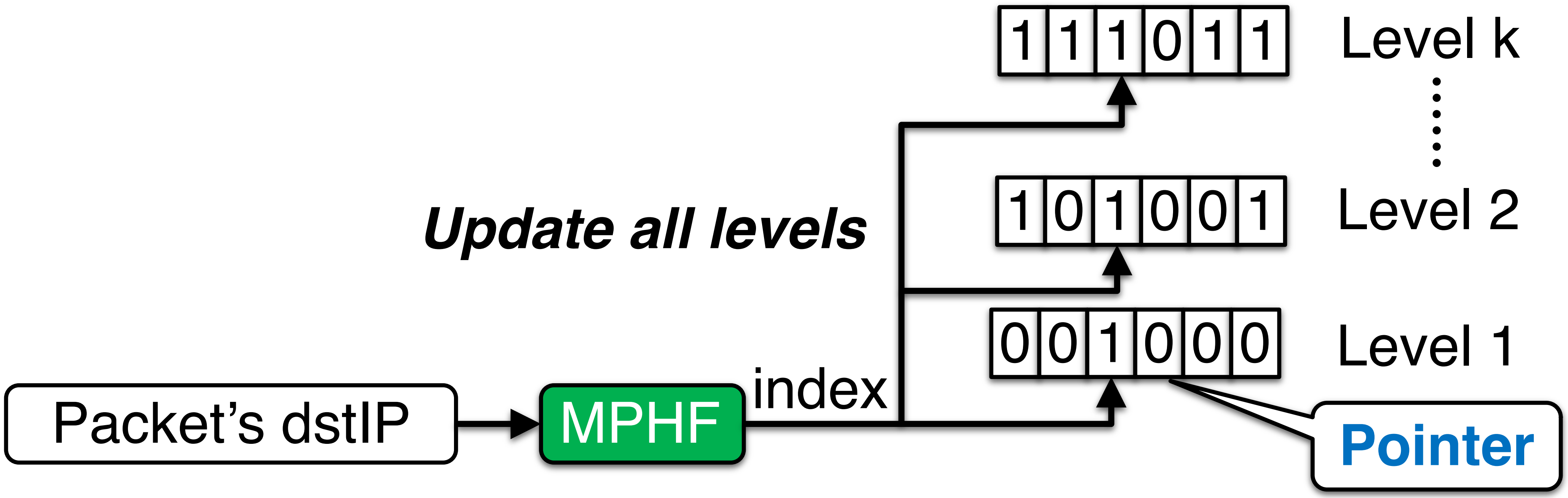
Minimal perfect hash function (MPHF)



SwitchPointer design

Maintaining pointers in the hierarchical data structure

Minimal perfect hash function (MPHF)

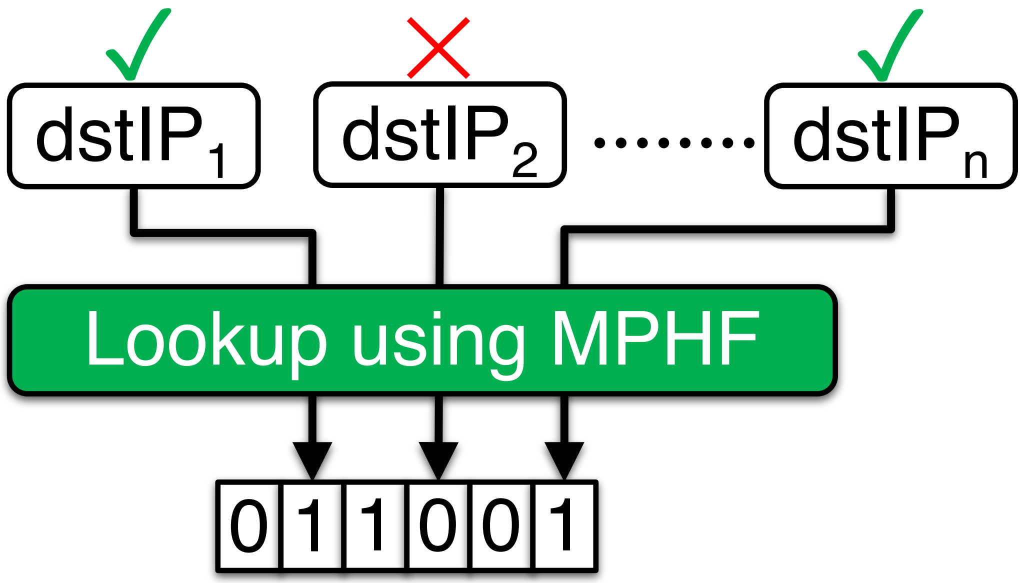


- **Single operation** to find the index to set in all levels

SwitchPointer design

Maintaining updated pointers in the hierarchical data structure

Minimal perfect hash function (MPHF)



Checks dstIP's corresponding bit in the bit array

SwitchPointer: Four technical challenges

- How to decide the right epoch size?
- How to efficiently maintain pointers?
- How to efficiently embed telemetry data?
- How to handle asynchronous clocks?

SwitchPointer design

Switch embeds telemetry data (e.g., linkID, epochID)

SwitchPointer design

Switch embeds telemetry data (e.g., linkID, epochID)

- INT: Packet header space limitation
- Cherrypick [SOSR'15] for current deployments

SwitchPointer design

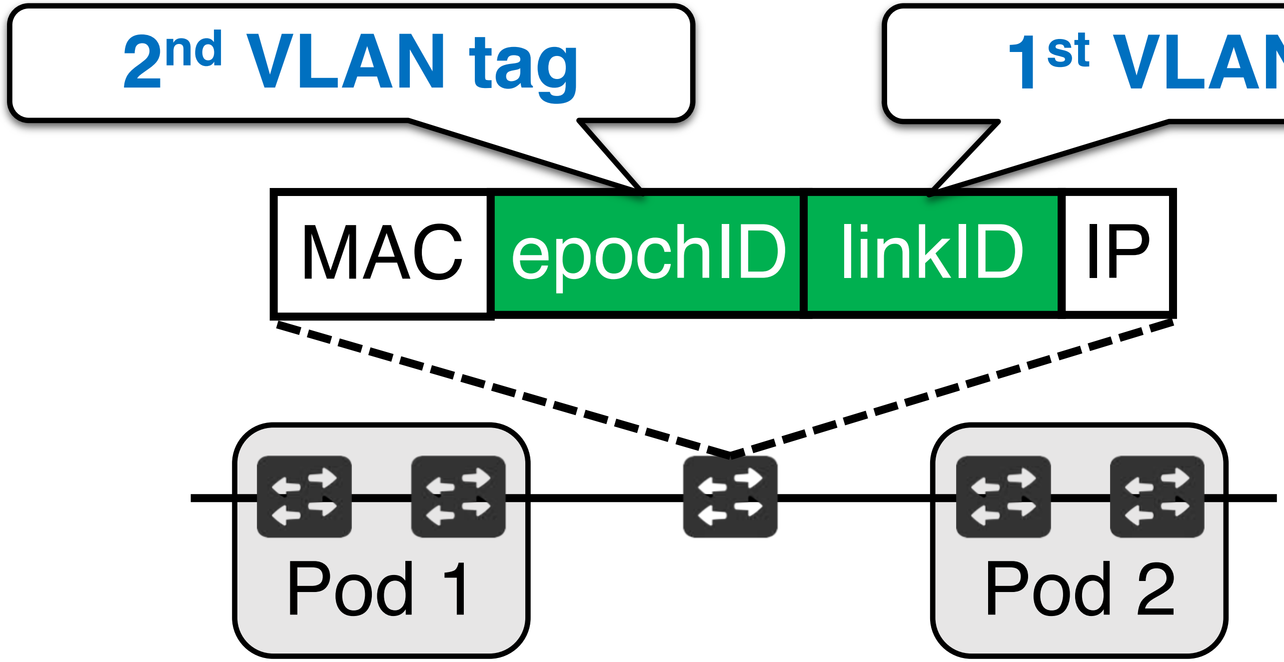
Switch embeds telemetry data (e.g., linkID, epochID)

- INT: Packet header space limitation
- Cherrypick [SOSR'15] for current deployments

SwitchPointer design

Switch embeds telemetry data (e.g., linkID, epochID)

- INT: Packet header space limitation
- Cherrypick [SOSR'15] for current deployments



SwitchPointer design

- Switch embeds telemetry data (e.g., linkID, epochID)
  - Packet header space limitation
  - Cherrypick [SOSR'15] for current deployments
- End-host collect and monitor telemetry data (E.g., PathDump [OSDI'16])

SwitchPointer design

- Switch embeds telemetry data (e.g., linkID, epochID)
  - Packet header space limitation
  - Cherrypick [SOSR'15] for current deployments
- End-host collect and monitor telemetry data (E.g., PathDump [OSDI'16])
  - Reconstructs the path
  - Computes a range of epochs for pod switches

SwitchPointer design

- Switch embeds telemetry data (e.g., linkID, epochID)
  - Packet header space limitation
  - Cherrypick [SOSR'15] for current deployments
- End-host collect and monitor telemetry data (E.g., PathDump [OSDI'16])
  - Reconstructs the path
  - Computes a range of epochs for pod switches

More details in our paper

SwitchPointer design

- Switch embeds telemetry data (e.g., linkID, epochID)
  - Packet header space limitation
  - Cherrypick [SOSR'15] for current deployments

- End-host collect and monitor telemetry data (E.g., PathDump [OSDI'16])
  - Reconstructs the path
  - Computes a range of epochs for pod switches

INT simplifies embedding and decoding telemetry data

More details in our paper

SwitchPointer: Four technical challenges

- How to decide the right epoch size?
- How to efficiently maintain pointers?
- How to efficiently embed telemetry data?
- How to handle asynchronous clocks?

Set bound on clock difference between any pair of devices

SwitchPointer: Four technical challenges

- How to decide the right epoch size?
- How to efficiently maintain pointers?
- How to efficiently embed telemetry data?
- How to handle asynchronous clocks?

Set bound on clock difference between any pair of devices

More details in our paper

SwitchPointer - Coverage

SwitchPointer - Coverage

In-network techniques

- TCP in-cast diagnosis
- Heavy hitter
- ECMP load imbalance diagnosis
- Silent random packet drops
- Traffic matrix
- DDoS

⋮

SwitchPointer - Coverage

In-network techniques

- TCP in-cast diagnosis
- Heavy hitter
- ECMP load imbalance diagnosis
- Silent random packet drops
- Traffic matrix
- DDoS

⋮

End-host based techniques

- TCP out of order packet delivery
- TCP non-monotonic
- Traffic bursts
- SYN flood attacks
- New TCP connections
- TCP in-complete flows

⋮

SwitchPointer - Coverage

In-network techniques

- TCP in-cast diagnosis
- Heavy hitter
- ECMP load imbalance diagnosis
- Silent random packet drops
- Traffic matrix
- DDoS

⋮

End-host based techniques

- TCP out of order packet delivery
- TCP non-monotonic
- Traffic bursts
- SYN flood attacks
- New TCP connections
- TCP in-complete flows

⋮

+

Spatially and temporally correlated problems
E.g.: Too many red lights, Traffic cascades

SwitchPointer - Coverage

In-network techniques

- TCP in-cast diagnosis
- Heavy hitter
- ECMP load imbalance diagnosis
- Silent random packet drops
- Traffic matrix
- DDoS

⋮

End-host based techniques

- TCP out of order packet delivery
- TCP non-monotonic
- Traffic bursts
- SYN flood attacks
- New TCP connections
- TCP in-complete flows

⋮

\+

Spatially and temporally correlated problems

E.g.: Too many red lights, Traffic cascades

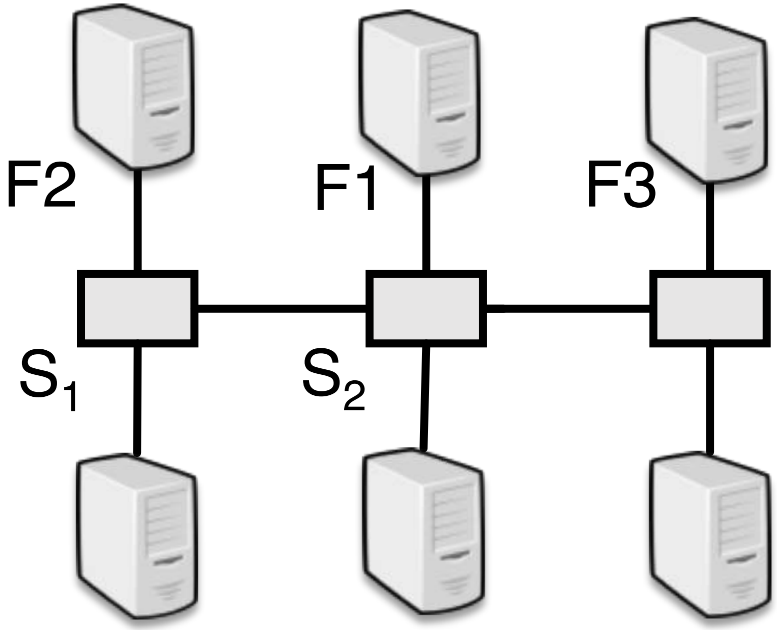
https://github.com/PathDump/Applications

Problems SwitchPointer cannot debug

- Instantaneous queue sizes
- Overlay loop detection
- Incorrect packet modification
- Packet properties at a switch

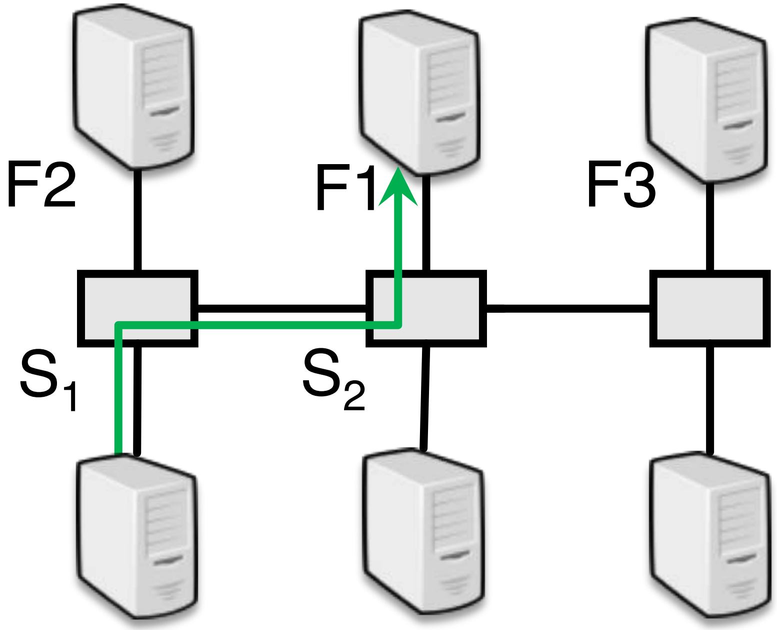
https://github.com/PathDump/Applications

A more complex example: Traffic cascades

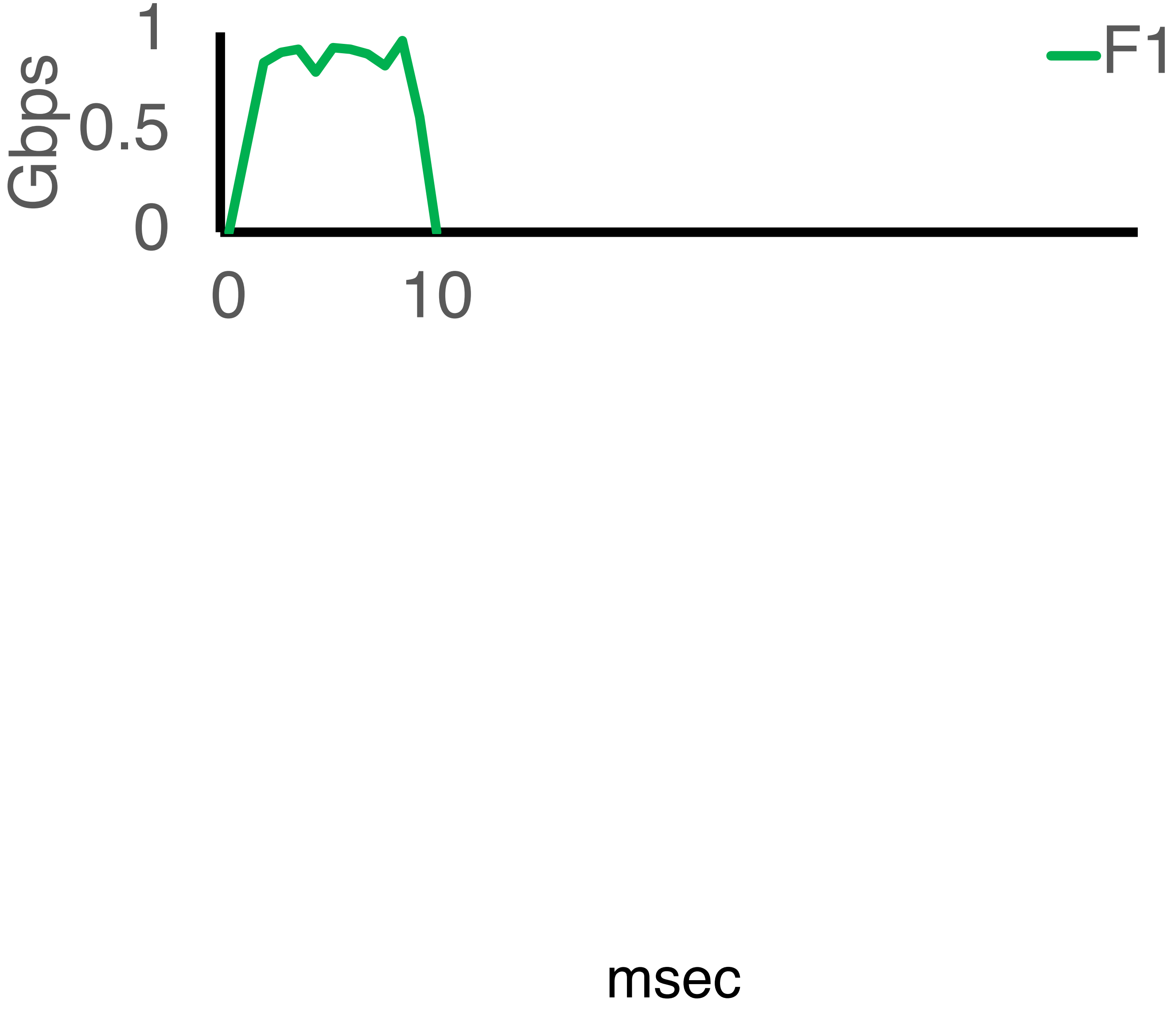


F_1: High priority
F_2: Middle priority
F_3: Low priority

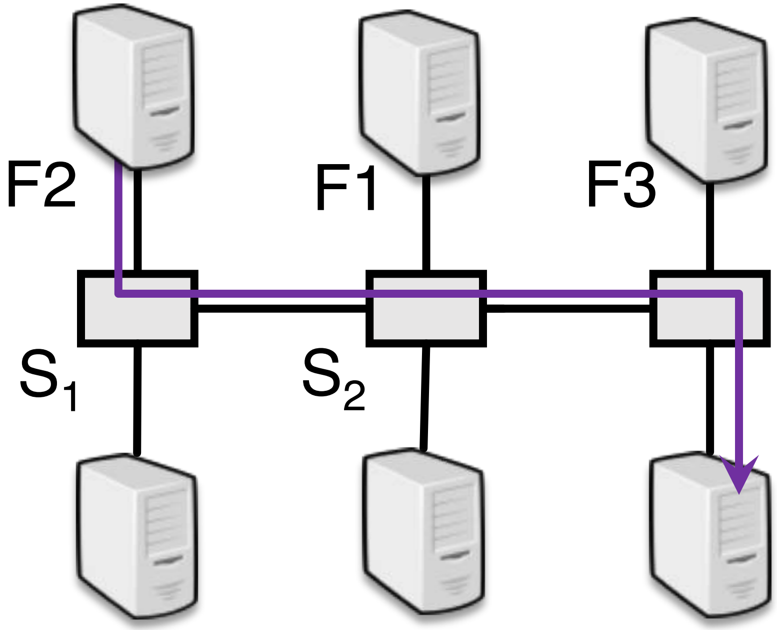
A more complex example: Traffic cascades



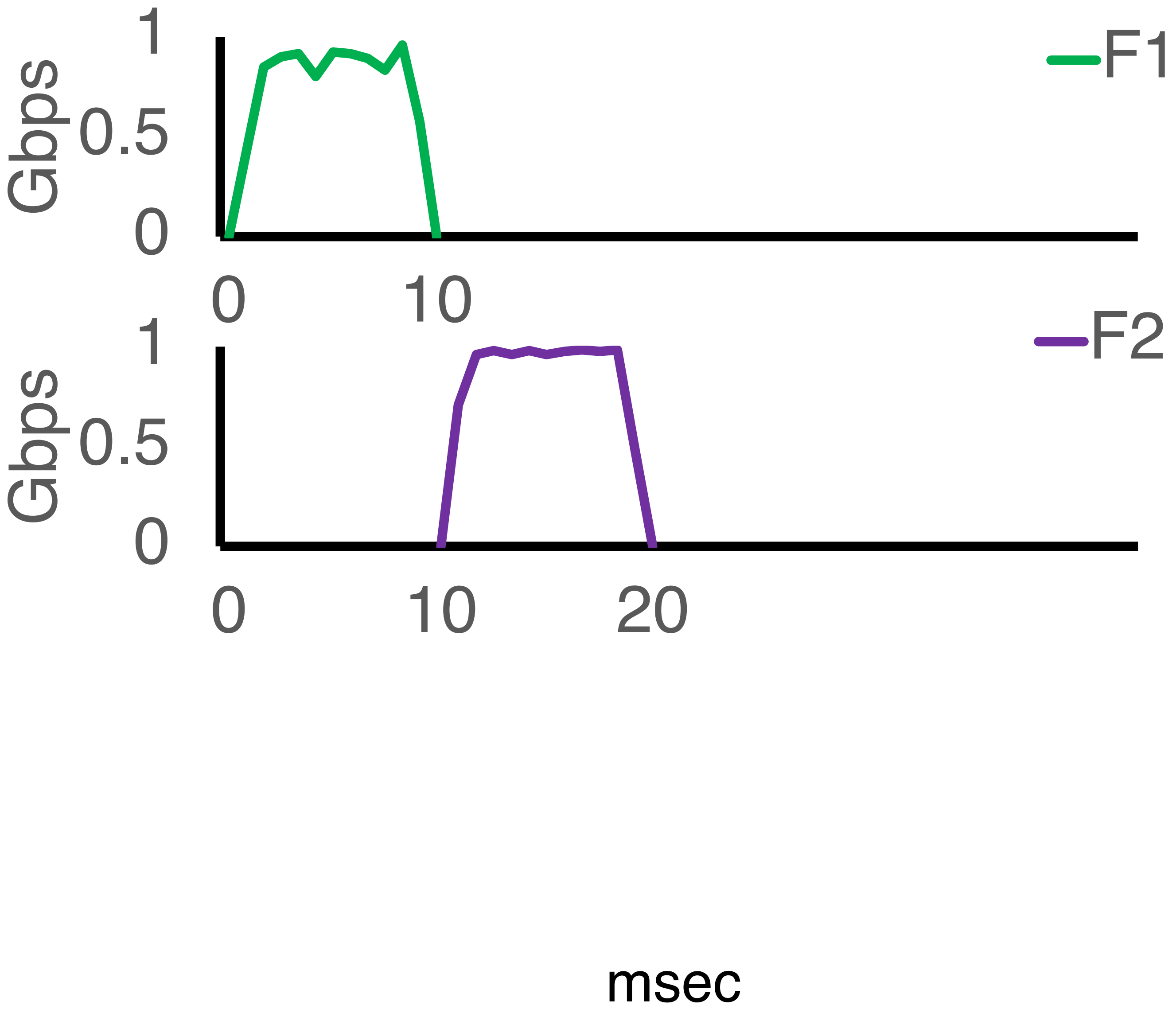
F_1: High priority
F_2: Middle priority
F_3: Low priority



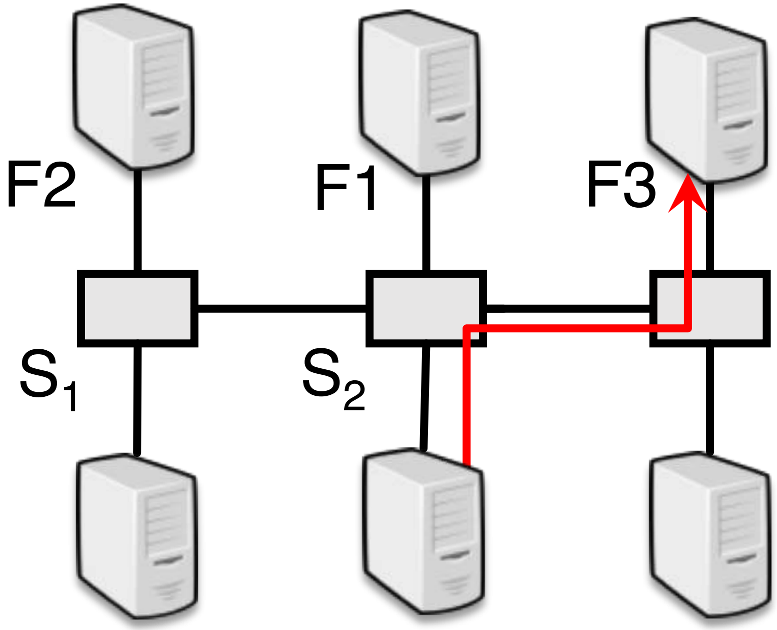
A more complex example: Traffic cascades



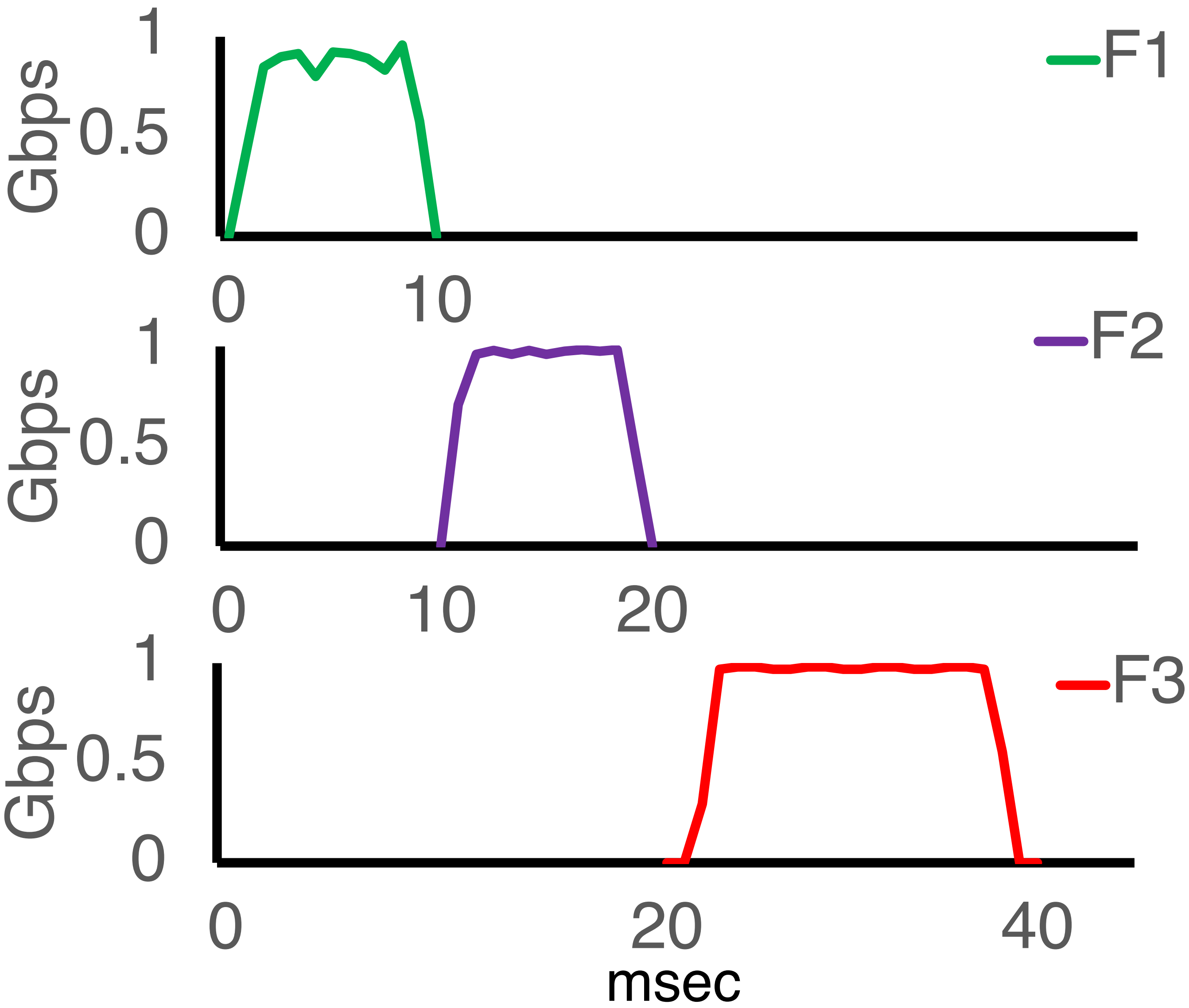
F_1: High priority
F_2: Middle priority
F_3: Low priority



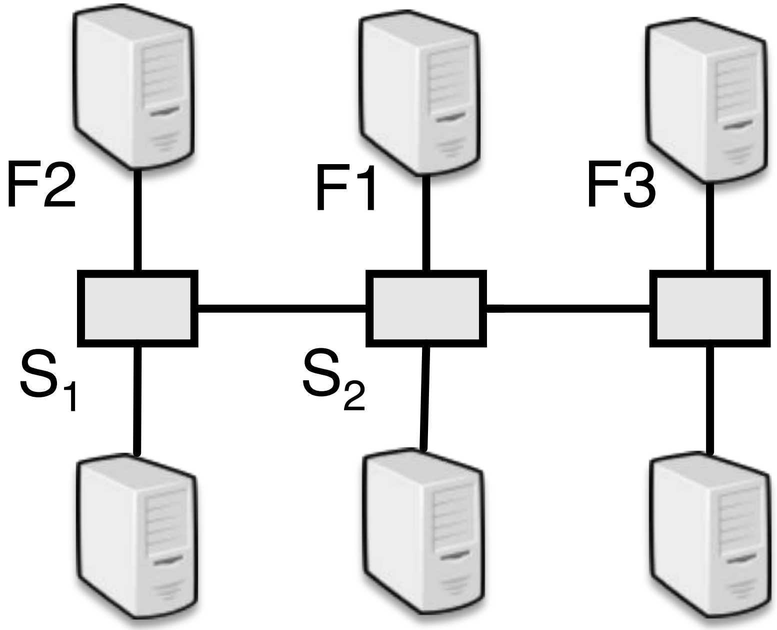
A more complex example: Traffic cascades



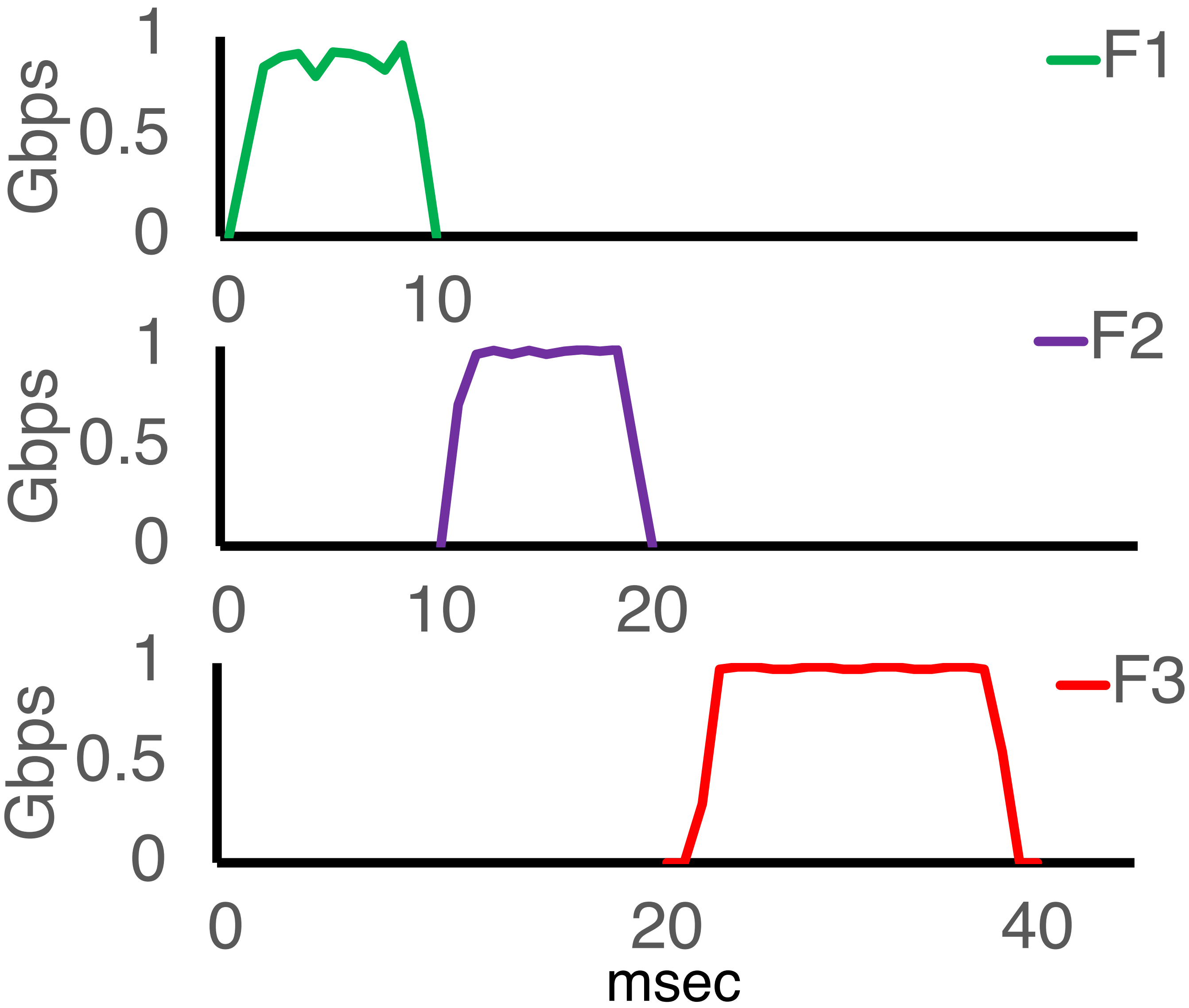
F_1: High priority
F_2: Middle priority
F_3: Low priority



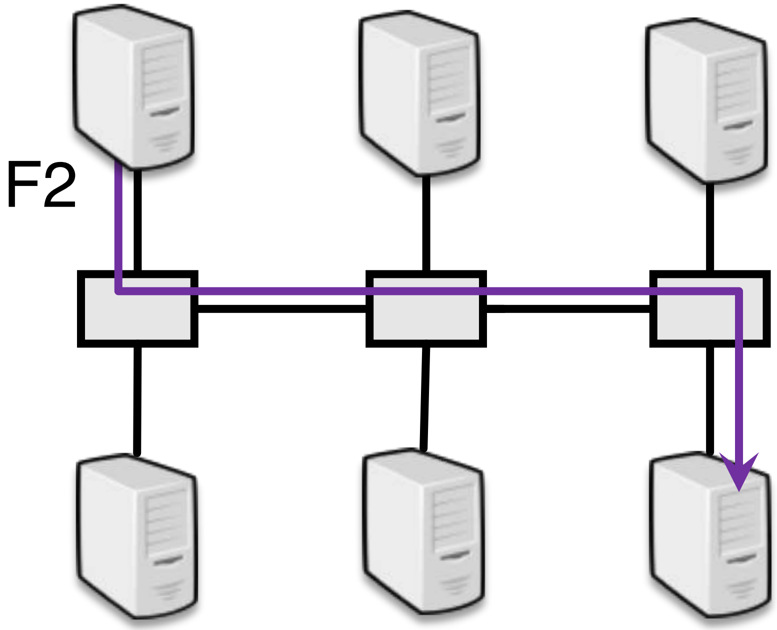
A more complex example: Traffic cascades



F_1: High priority
F_2: Middle priority
F_3: Low priority

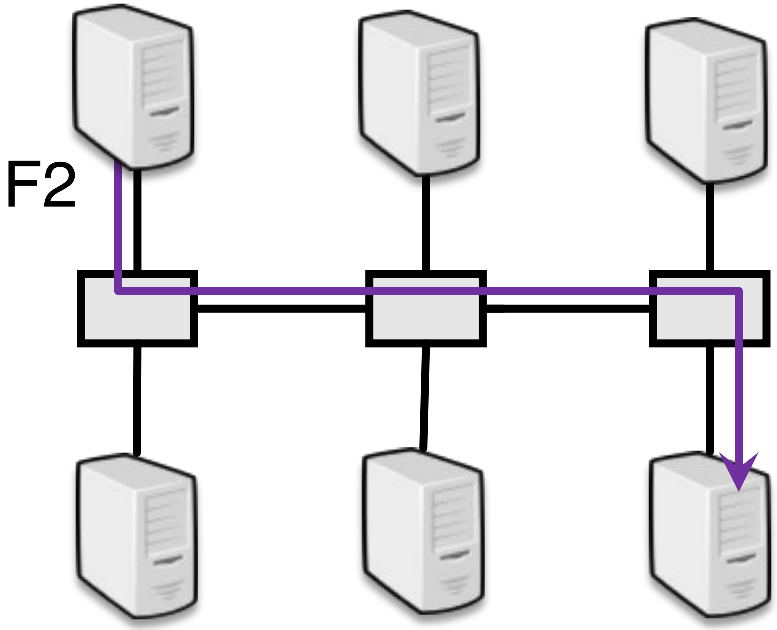


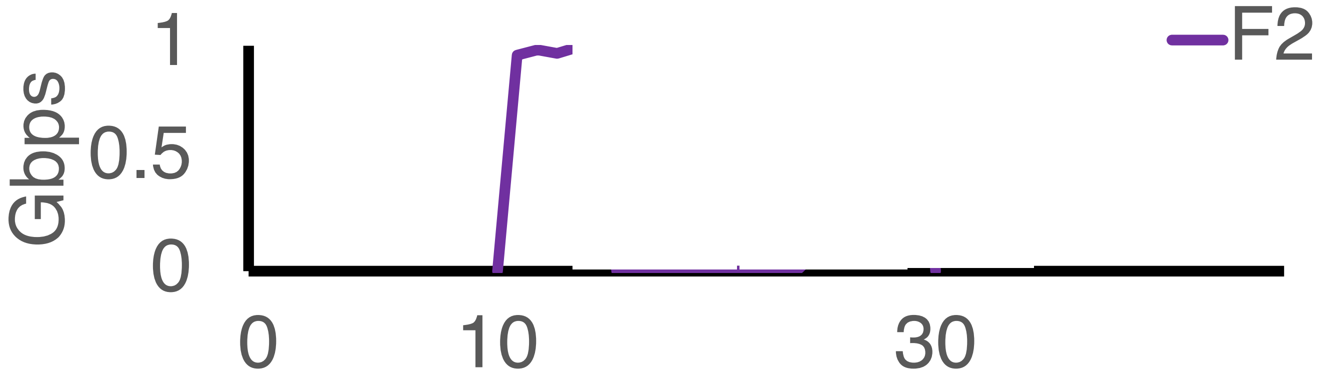
A more complex example: Traffic cascades



F_1: High priority
F_2: Middle priority
F_3: Low priority

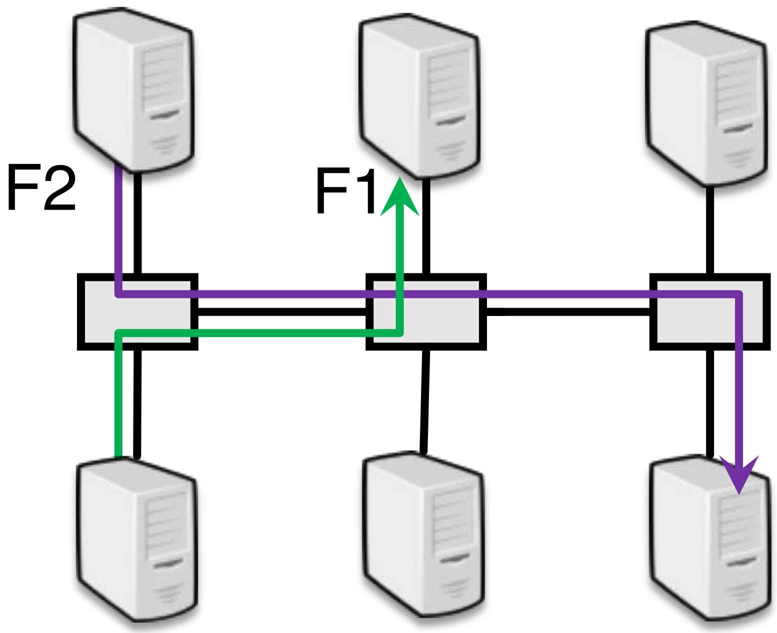
A more complex example: Traffic cascades

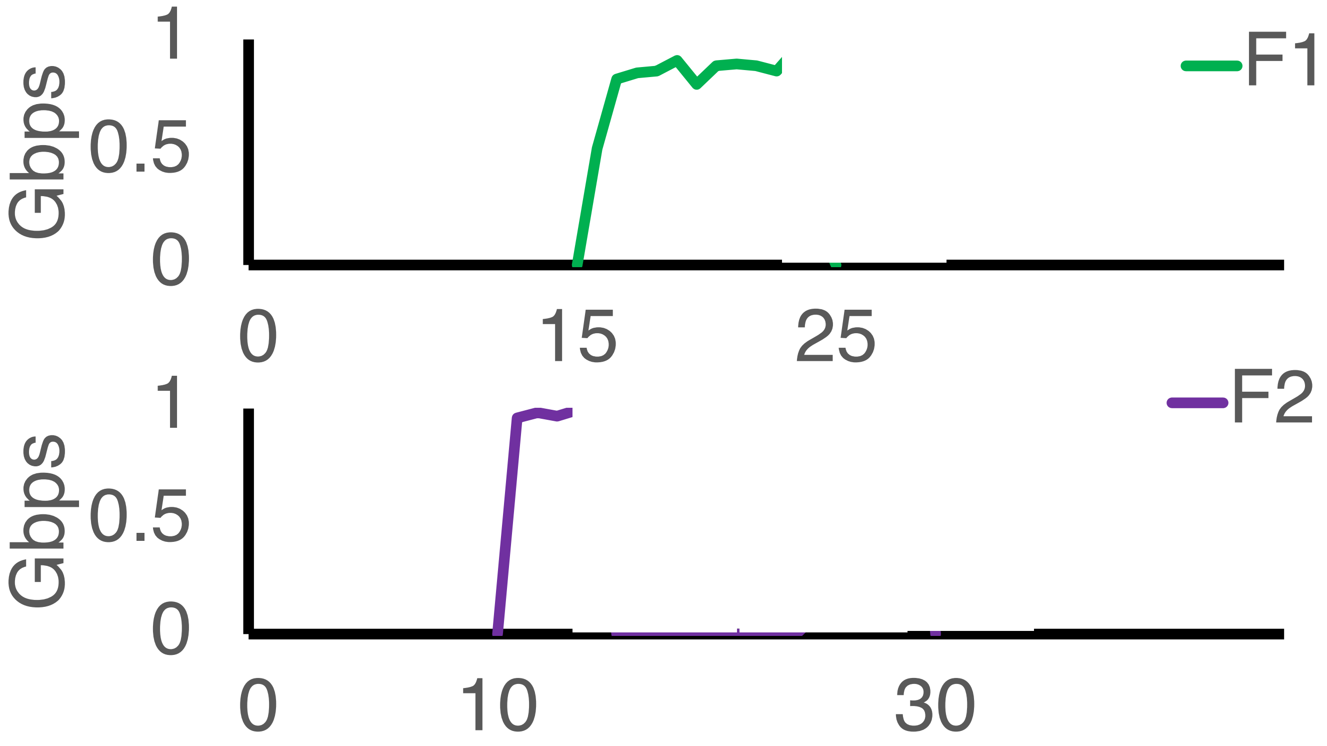




F_1: High priority
F_2: Middle priority
F_3: Low priority

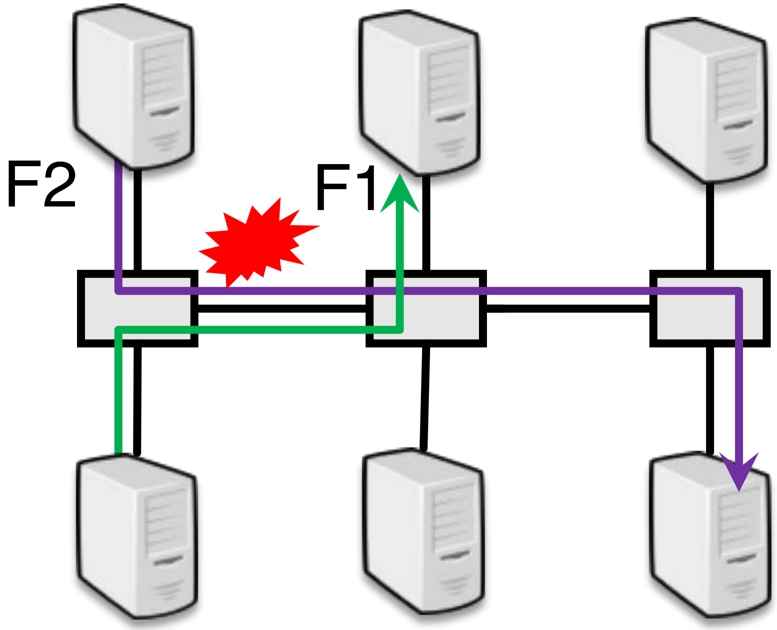
A more complex example: Traffic cascades



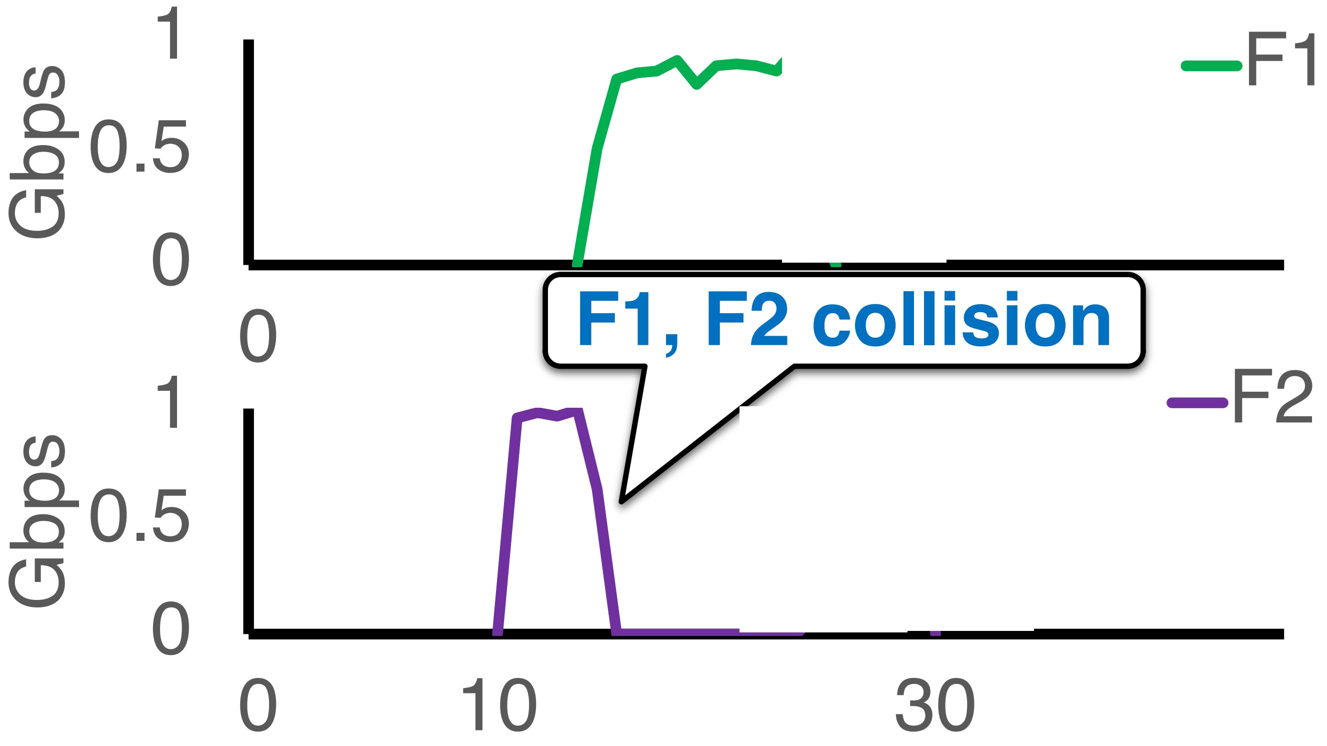


F_1: High priority
F_2: Middle priority
F_3: Low priority

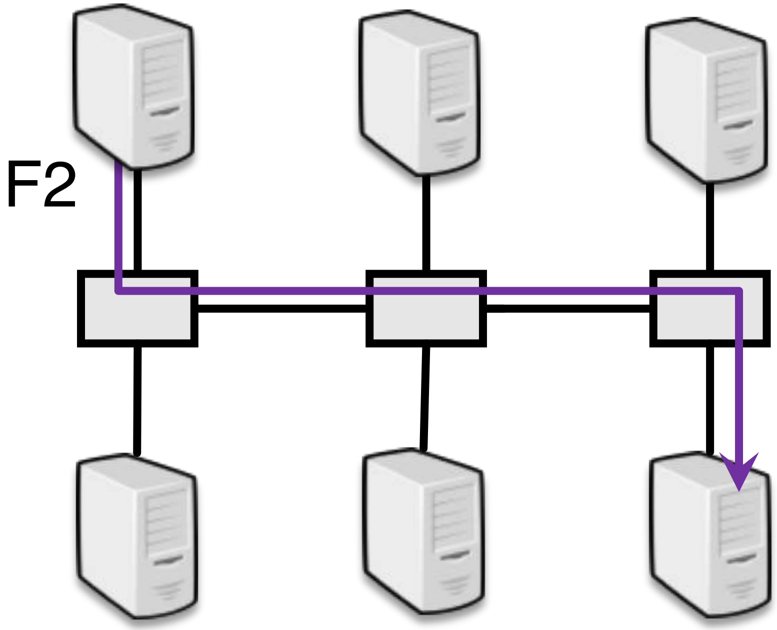
A more complex example: Traffic cascades

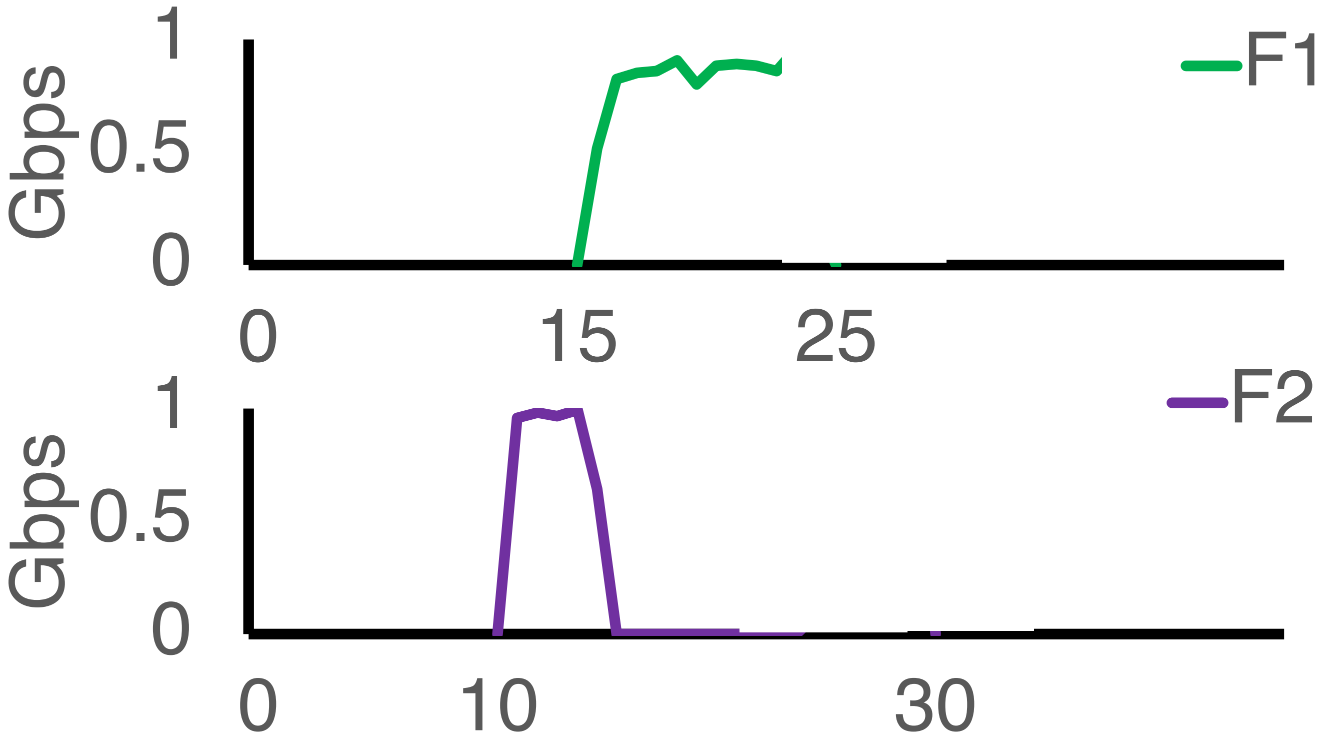


F_1: High priority
F_2: Middle priority
F_3: Low priority



A more complex example: Traffic cascades



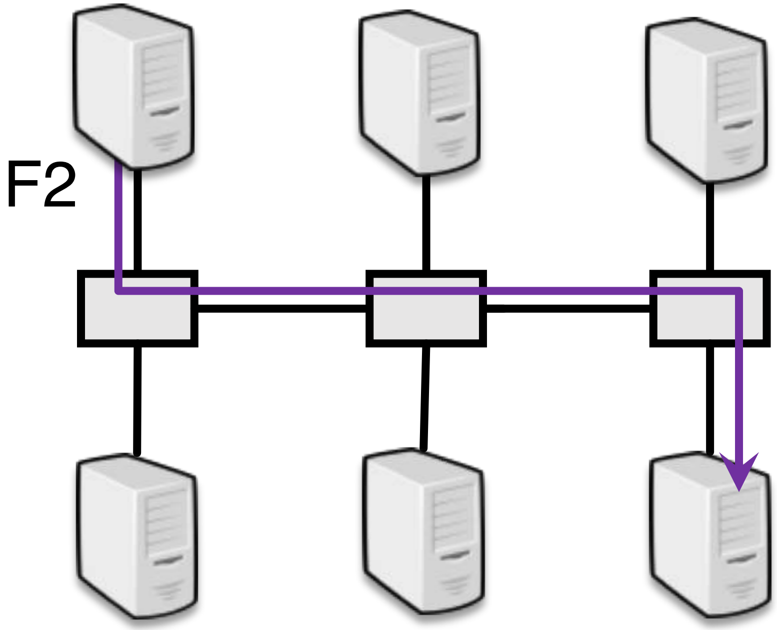


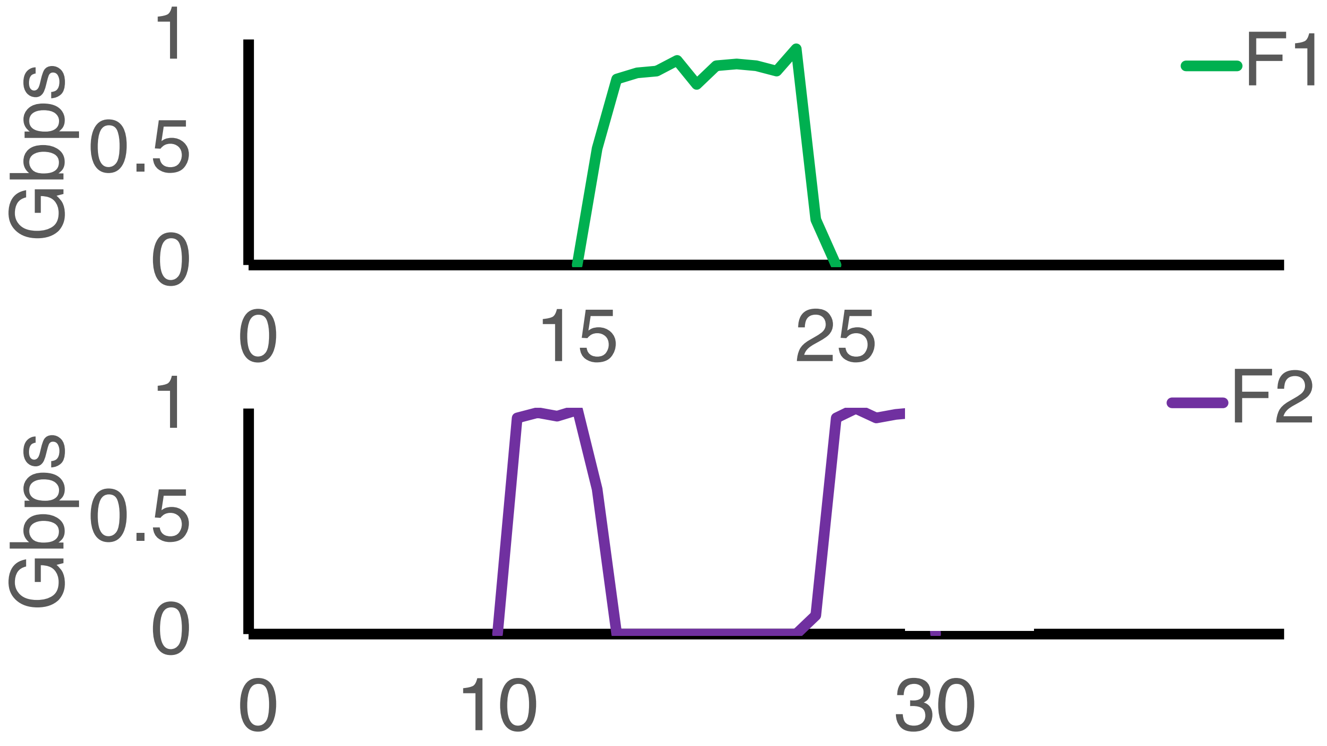
F_1: High priority

F_2: Middle priority

F_3: Low priority

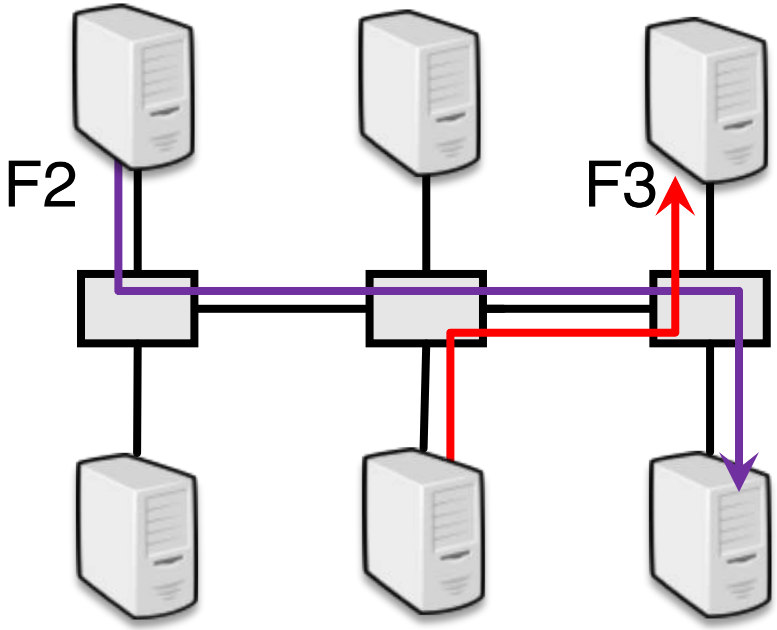
A more complex example: Traffic cascades

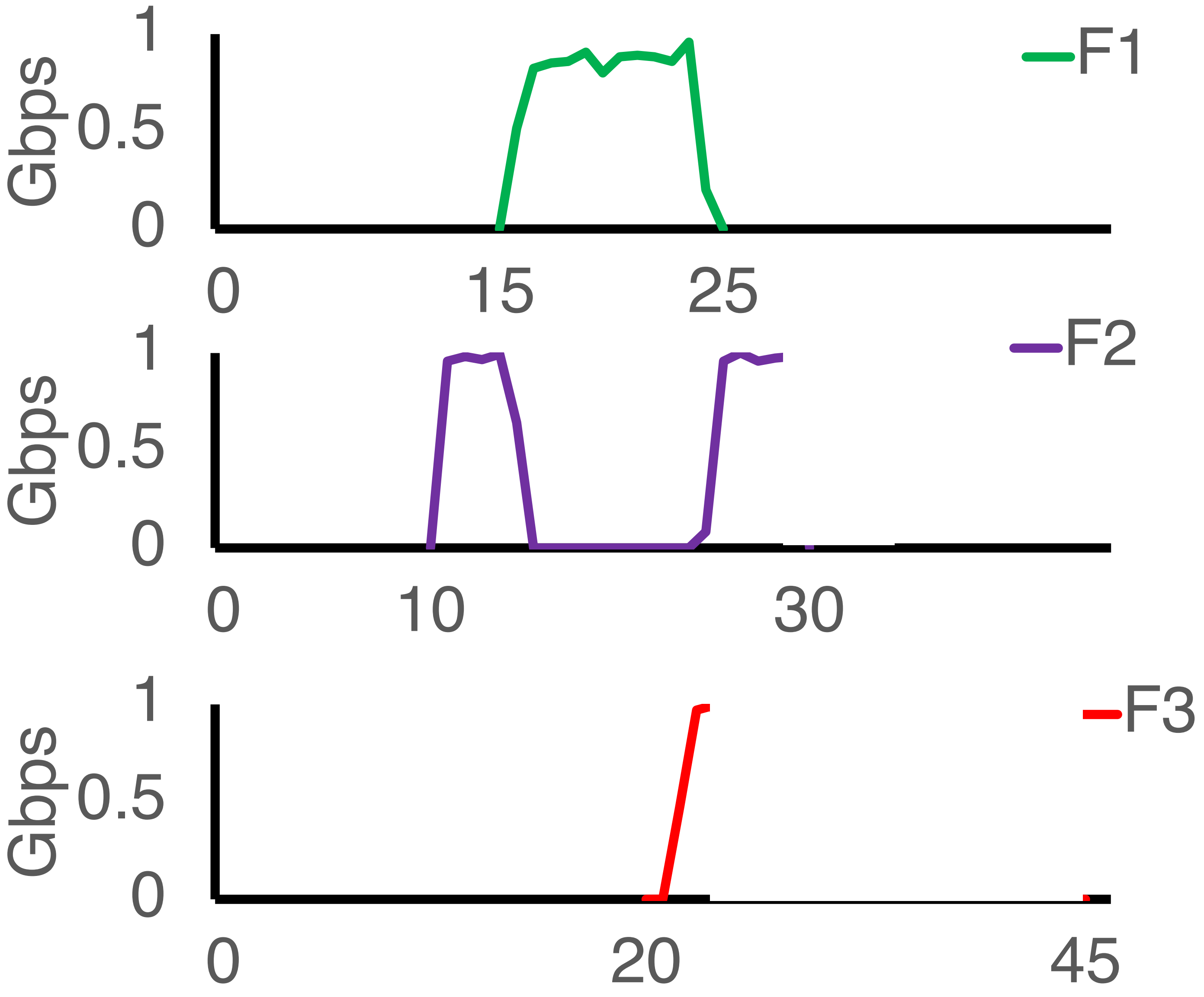




F_1: High priority
F_2: Middle priority
F_3: Low priority

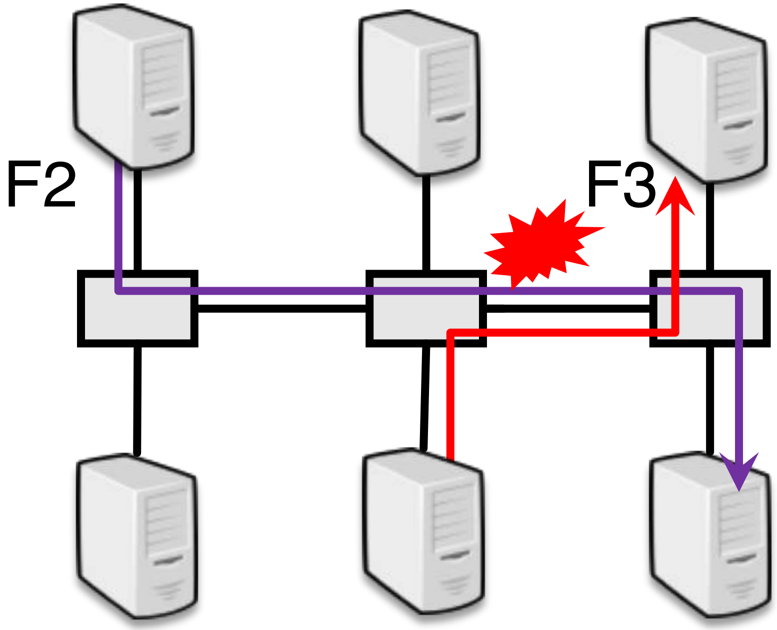
A more complex example: Traffic cascades

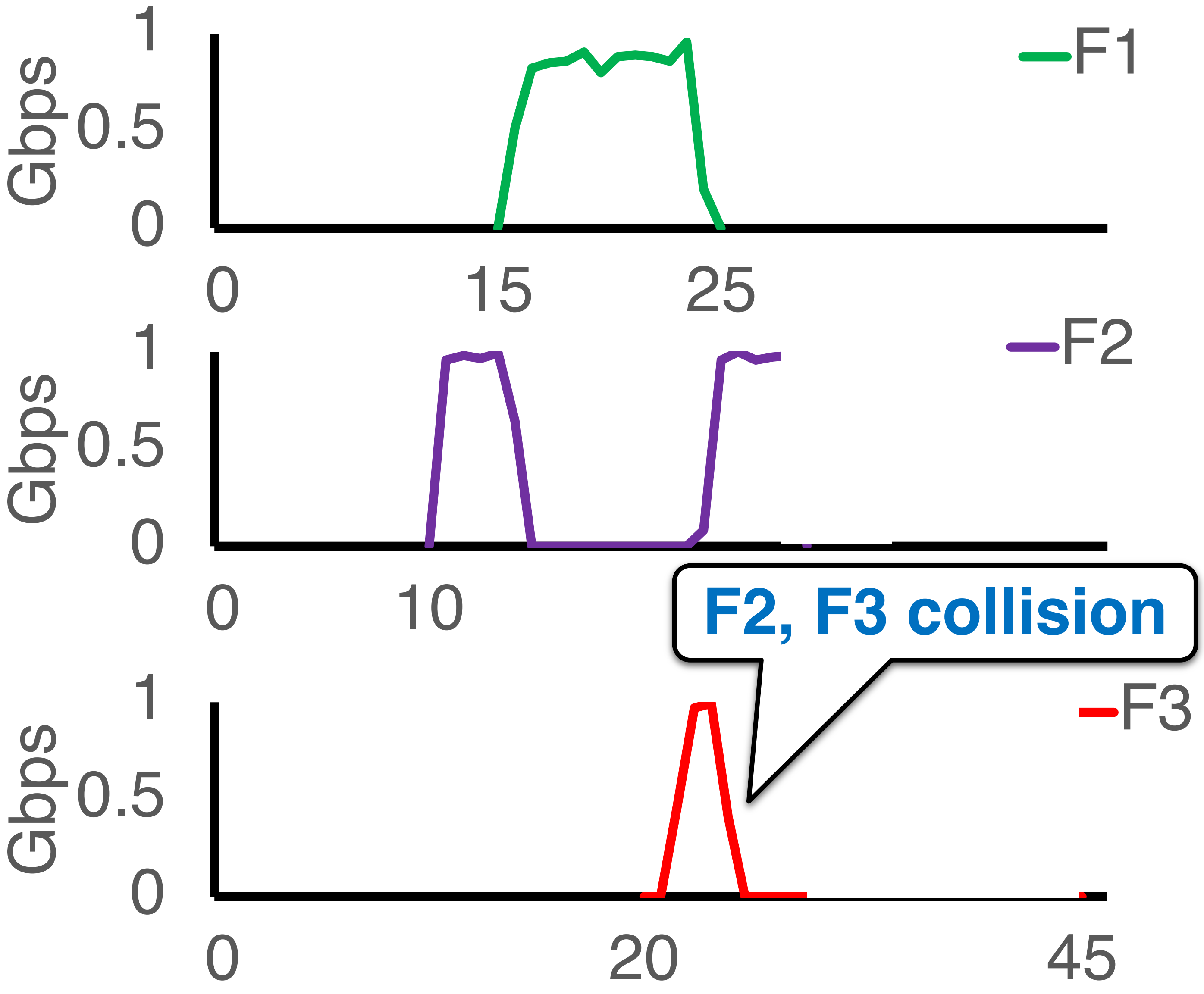




F_1: High priority
F_2: Middle priority
F_3: Low priority

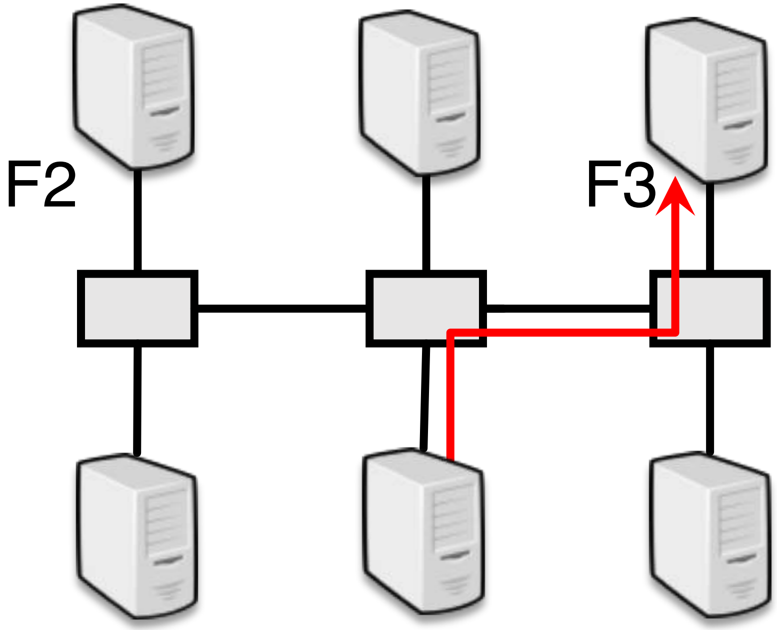
A more complex example: Traffic cascades



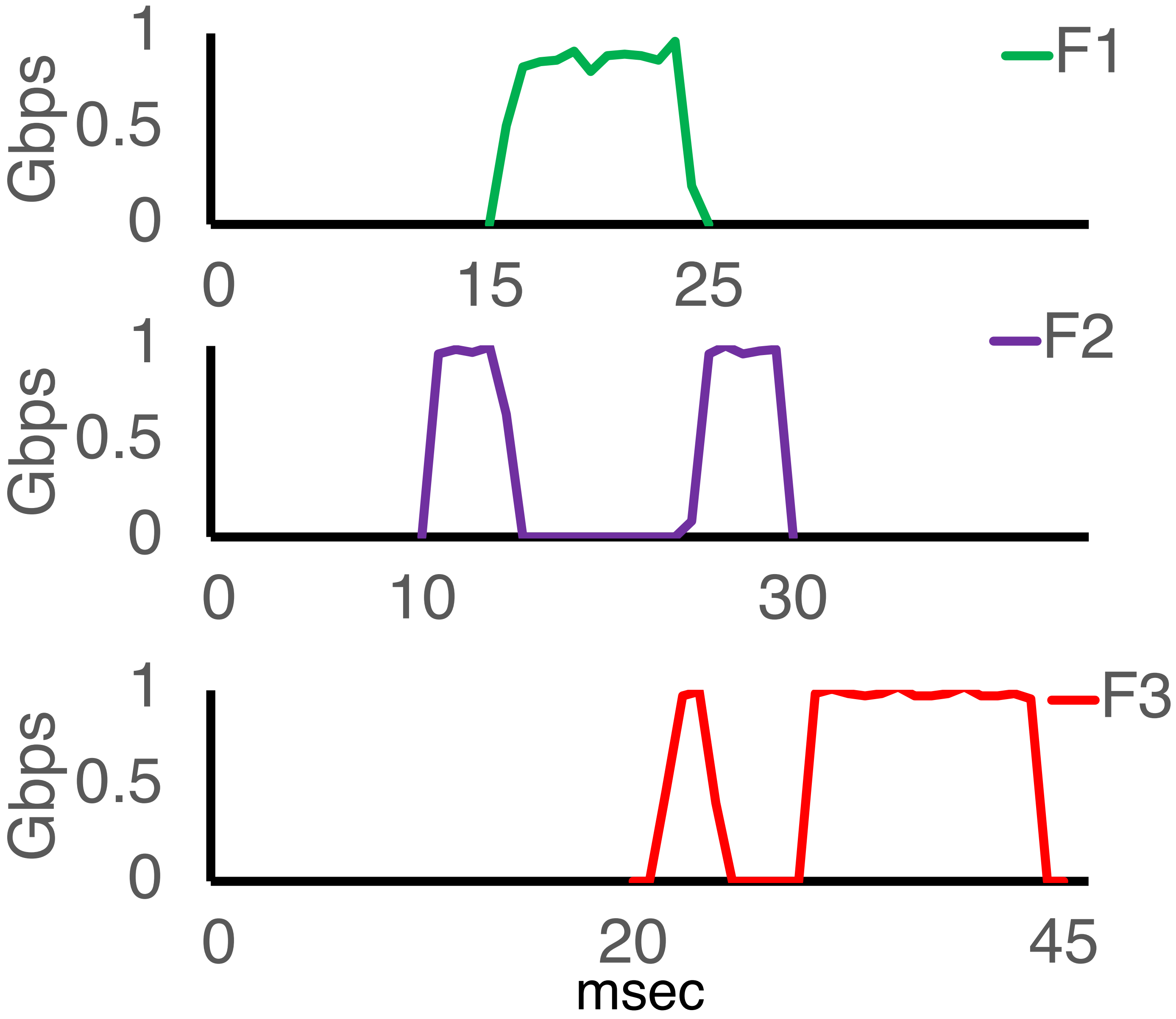


F_1: High priority
F_2: Middle priority
F_3: Low priority

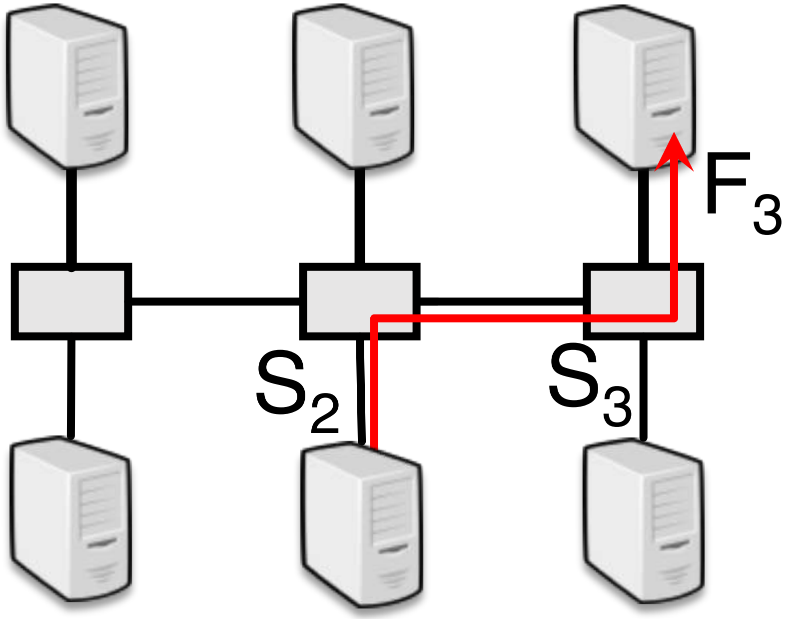
A more complex example: Traffic cascades



F_1: High priority
F_2: Middle priority
F_3: Low priority

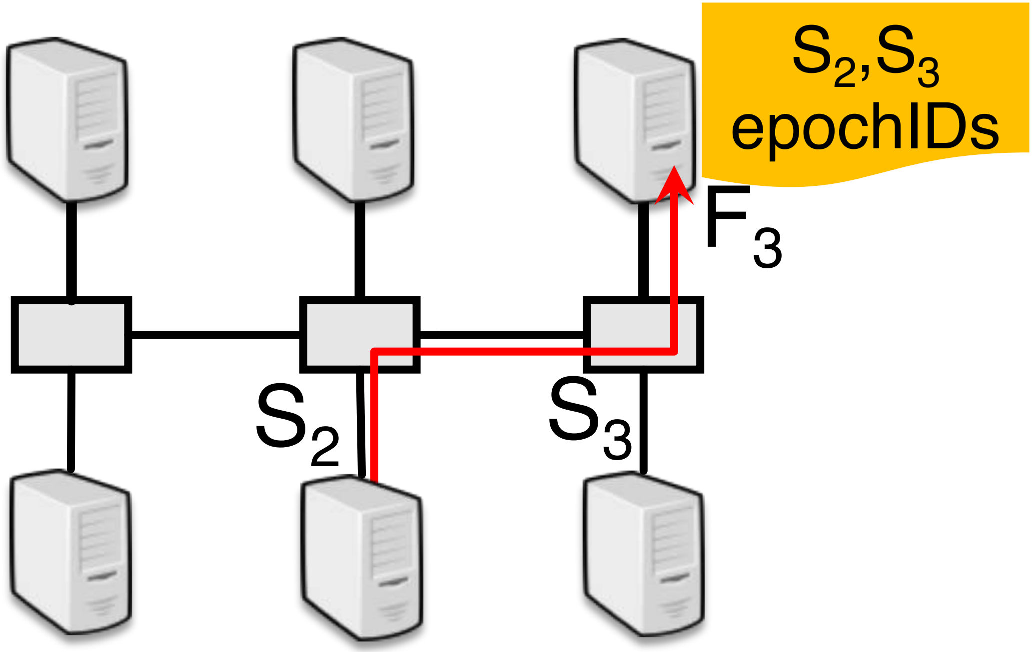


A more complex example: Traffic cascades



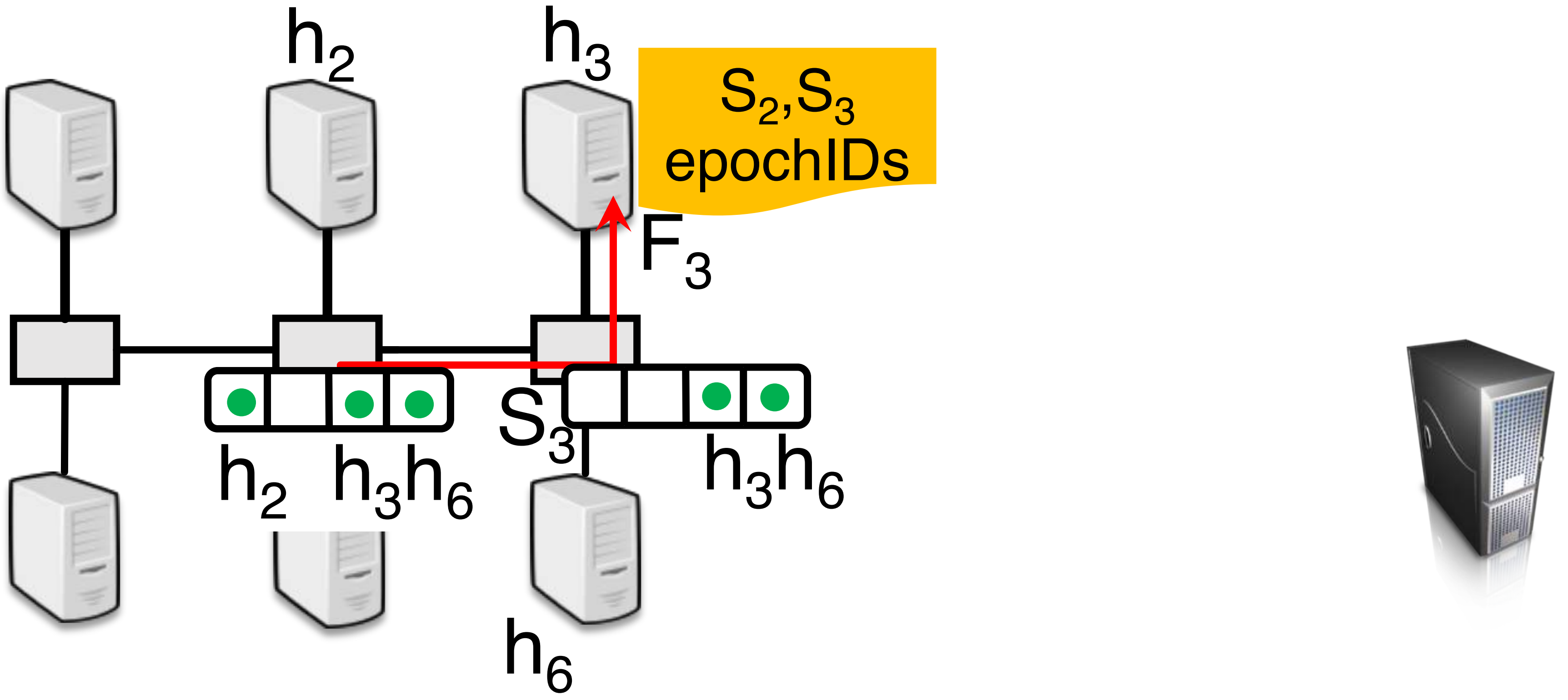
F_1: High priority
F_2: Middle priority
F_3: Low priority

A more complex example: Traffic cascades



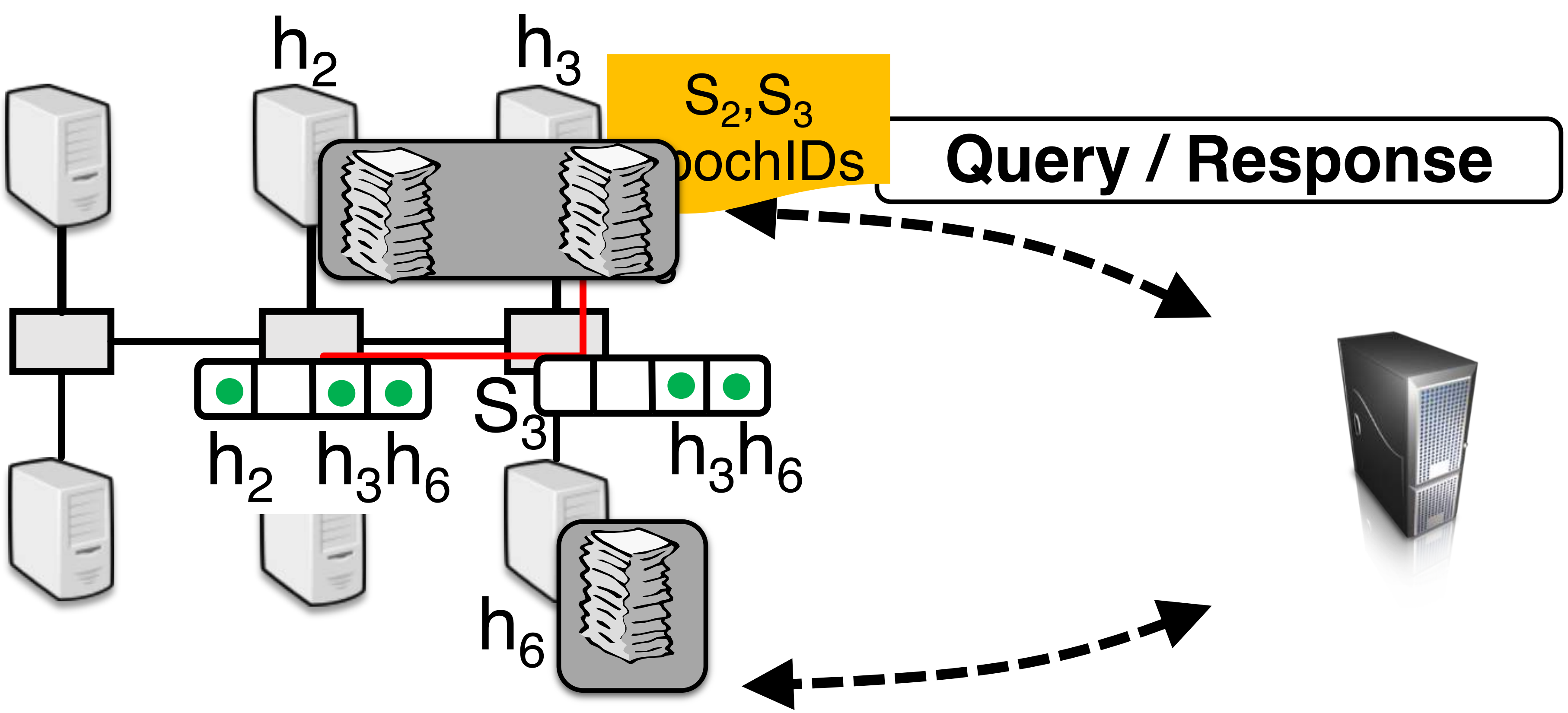
F_1: High priority
F_2: Middle priority
F_3: Low priority

A more complex example: Traffic cascades



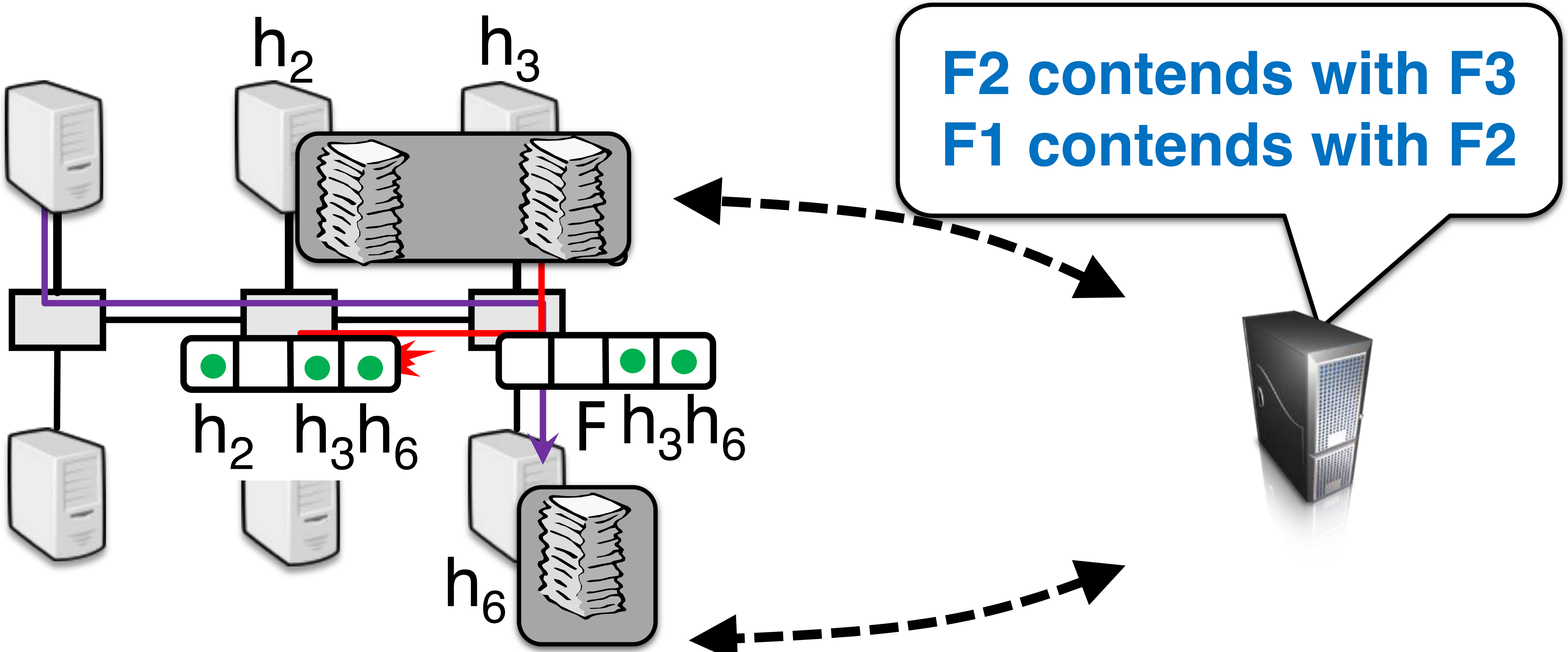
F_1: High priority
F_2: Middle priority
F_3: Low priority

A more complex example: Traffic cascades



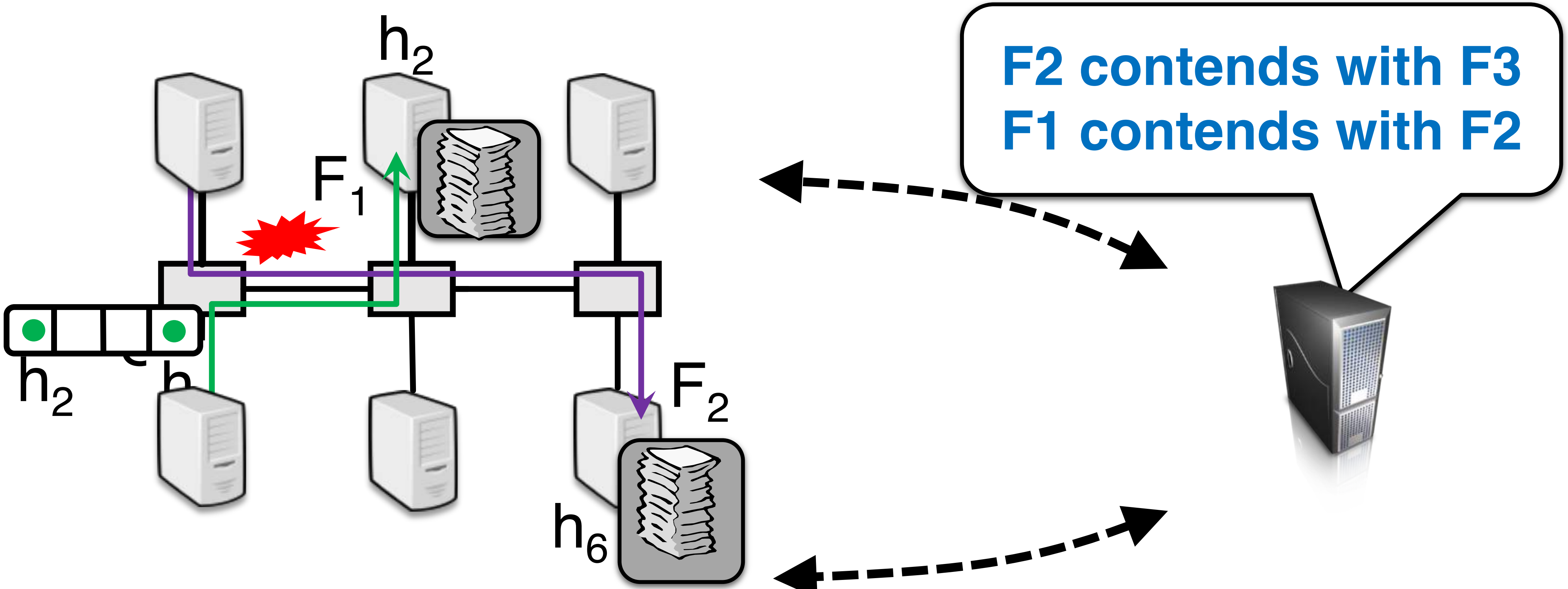
F_1: High priority
F_2: Middle priority
F_3: Low priority

A more complex example: Traffic cascades



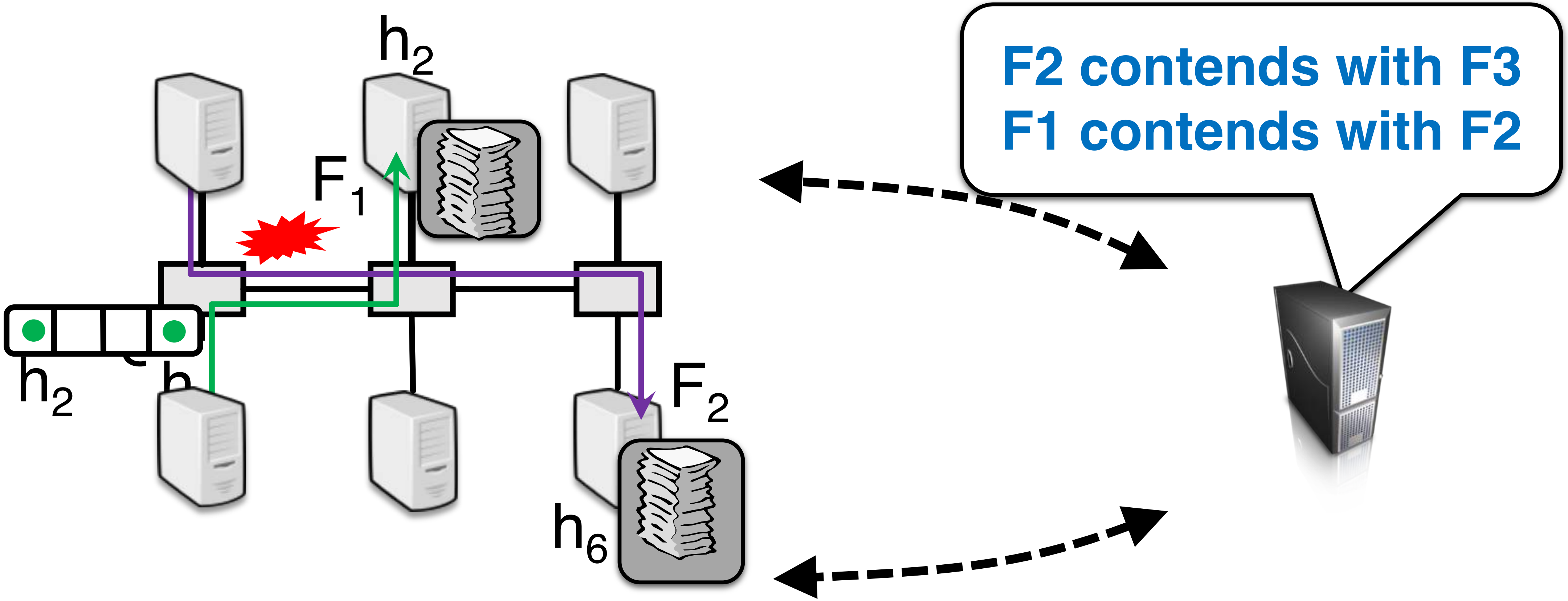
F_1: High priority
F_2: Middle priority
F_3: Low priority

A more complex example: Traffic cascades



F_1: High priority
F_2: Middle priority
F_3: Low priority

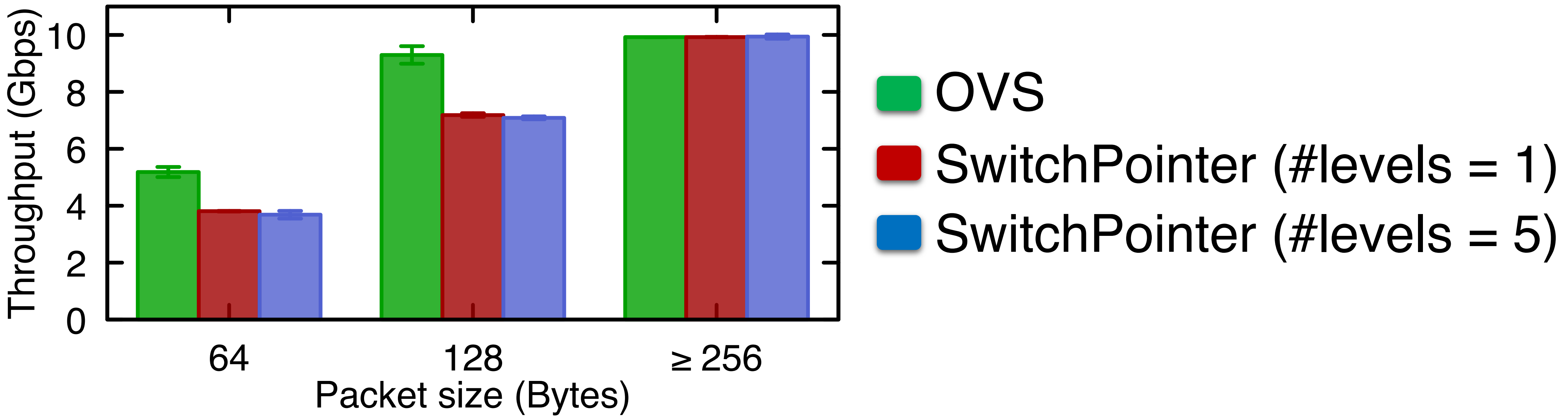
A more complex example: Traffic cascades



F_1: High priority
F_2: Middle priority
F_3: Low priority

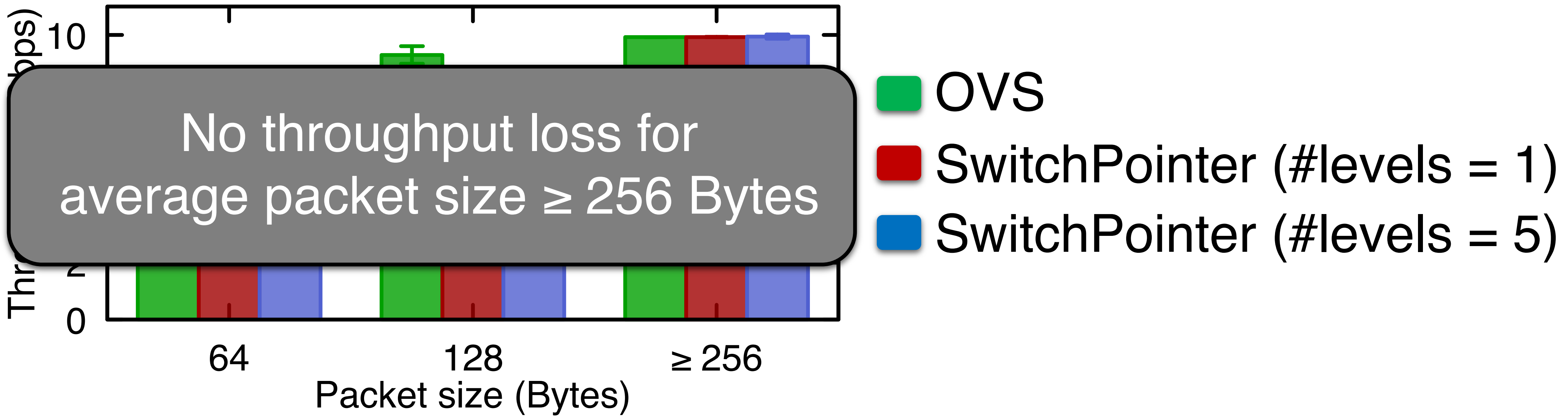
SwitchPointer overhead (software implementation)

- Prototype
  - ✓ Implemented on top of OVS-DPDK version
  - ✓ Build minimal perfect hash function using CMPH library



SwitchPointer overhead (software implementation)

- Prototype
  - ✓ Implemented on top of OVS-DPDK version
  - ✓ Build minimal perfect hash function using CMPH library



No throughput loss for average packet size ≥ 256 Bytes

Conclusion

- Achieves benefits of both end-host and in-network approaches
- Switch acts as a “directory service”
- Uses end-host resources to collect and monitor telemetry data
- Debugs a large class problems
- Ongoing work: Hardware implementation using P4 and NetFPGA