Memory-Efficient Performance Monitoring on Programmable Switches with Lean Algorithms

Alan (Zaoxing) Liu

Joint work with Samson Zhou, Ori Rottenstreich, Vladimir Braverman, Jennifer Rexford

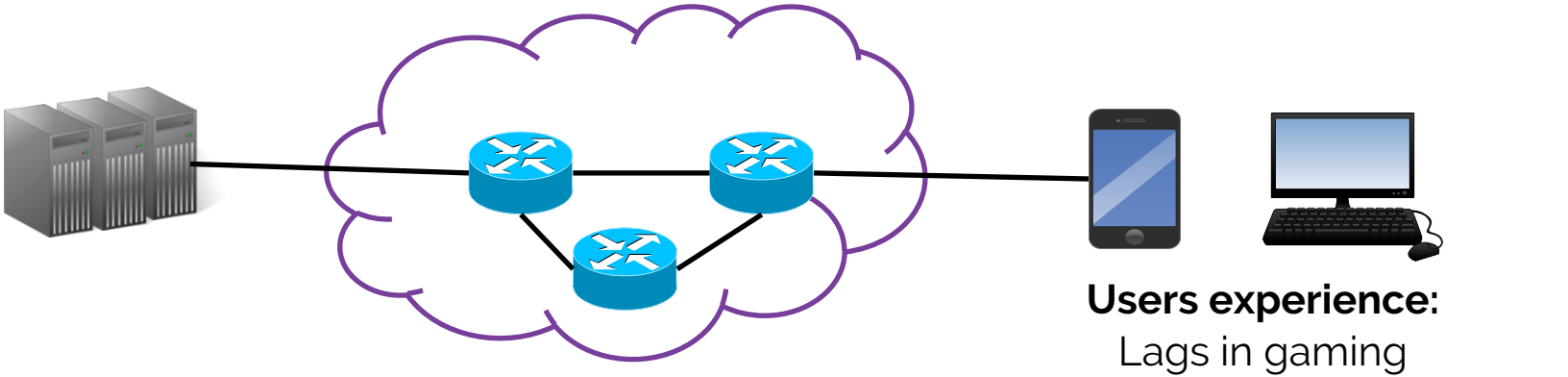
Carnegie Mellon University

TECHNION Israel Institute of Technology

JOHNS HOPKINS UNIVERSITY

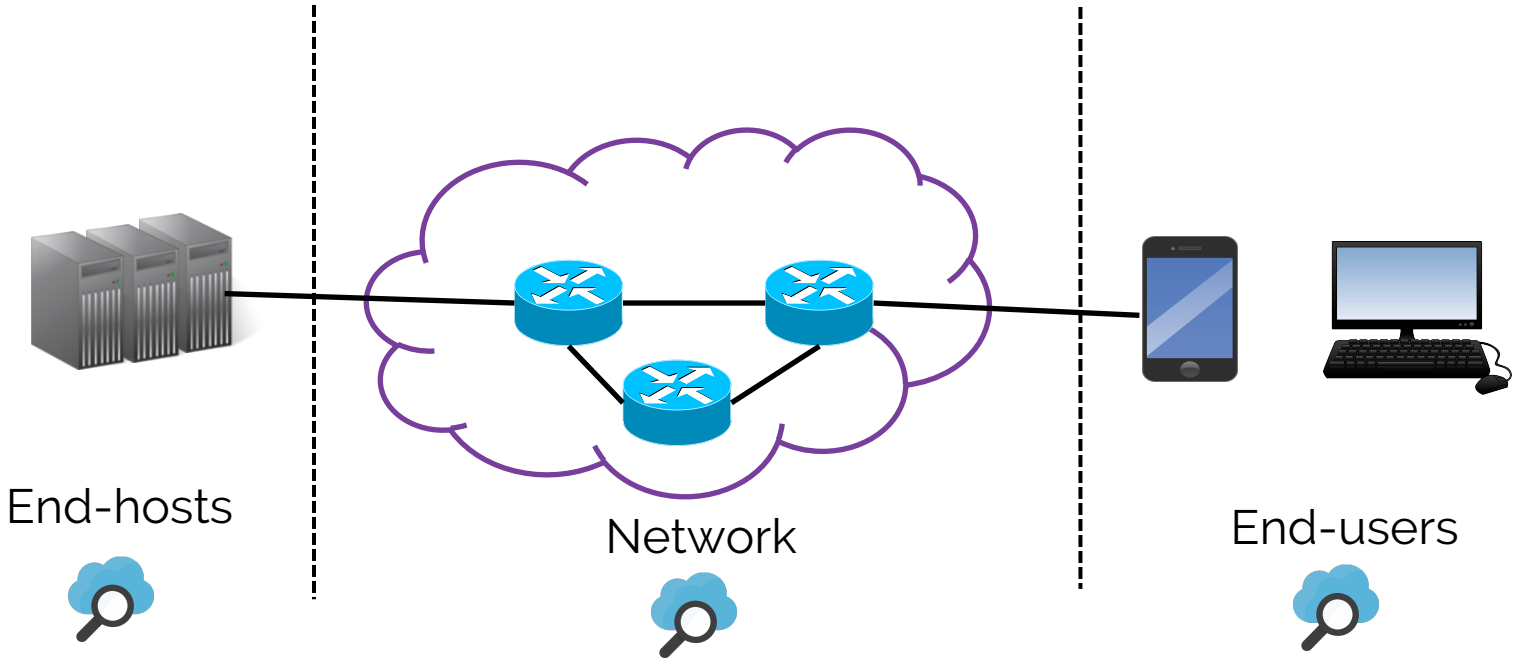
PRINCETON UNIVERSITY

Network performance issues are pervasive



Observations: Network performance issues can happen anytime, anywhere in the network

Where to monitor network performance?



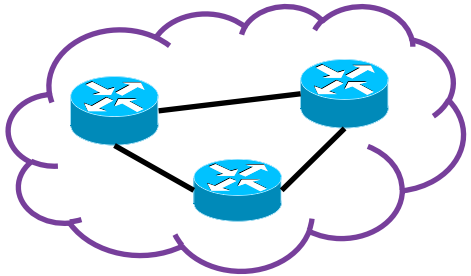
- **End-hosts:** Need to modify the OS' network stack.
- **In the network:** No end-host access

What performance statistics we care about?

Packet latency (TCP)

Packets sent

Sender



Latency:
|time(■) – time(■)|

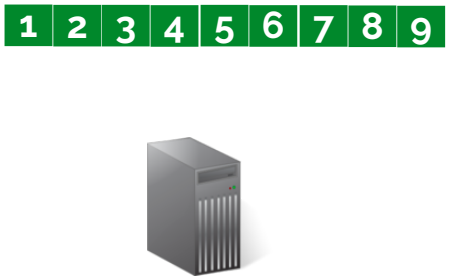
Packets received

Acknowledgements sent

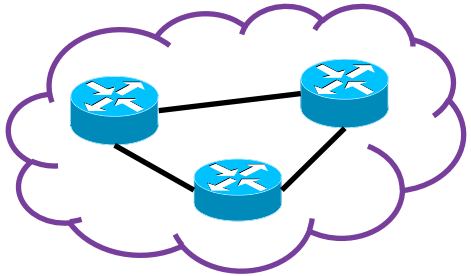
Receiver

What performance statistics we care about?

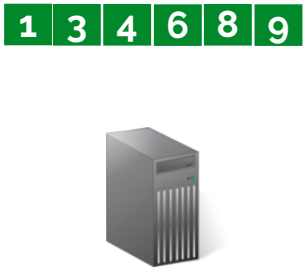
Packet loss



Sender



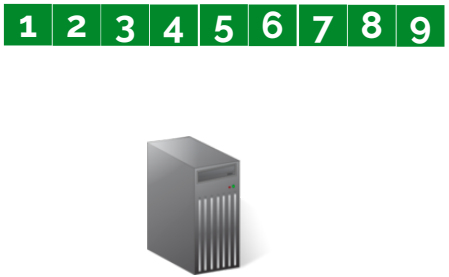
Lost packets: 3



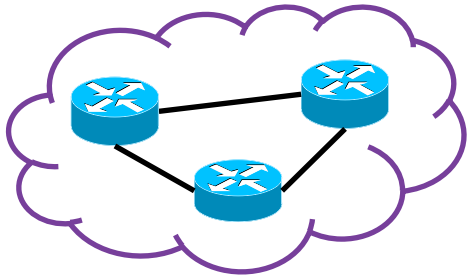
Receiver

What performance statistics we care about?

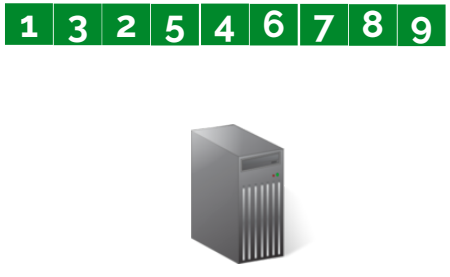
Out-of-order packets



Sender



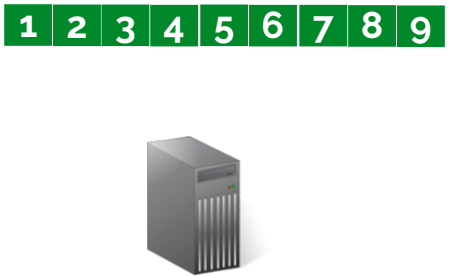
Out-of-order packets: 4



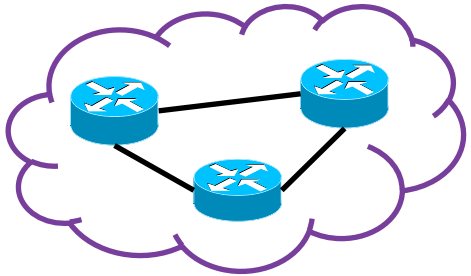
Receiver

What performance statistics we care about?

Retransmitted packets



Sender



Retransmitted packets: 2

Receiver

Performance statistics are useful in network diagnosis

Faulty network links

"Flows with high packet loss"

Slow TCP rates

"Flows with high latency"

Incorrect Quality of Service

"Flows with high out-of-order packets"

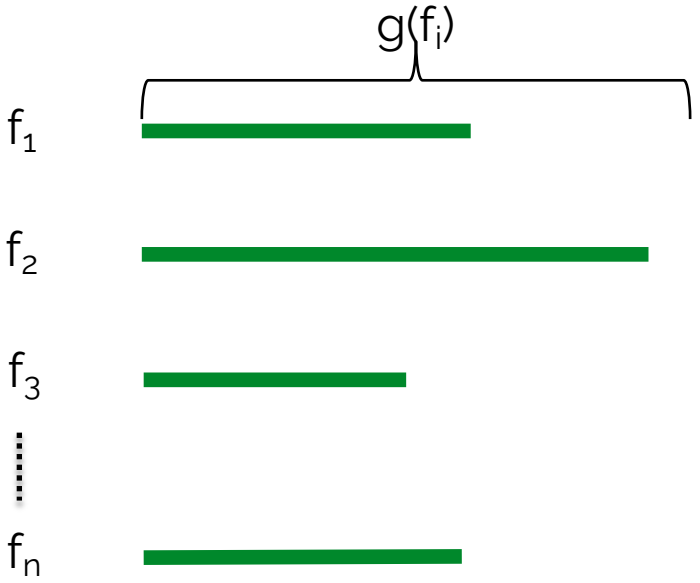
Congested network

"Lost, retransmitted packets"

Network diagnosis needs various performance statistics

Performance monitoring: A streaming problem

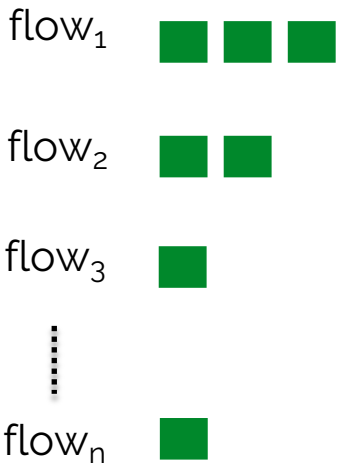
Network packets



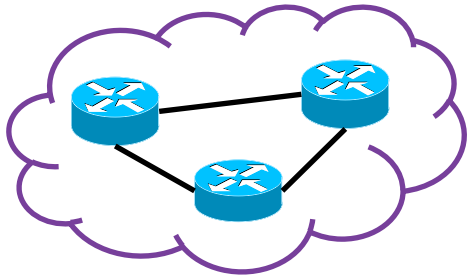
g-heavy hitters: Identify flows f_i that have large $g(f_i)$ values
g(·): Packet loss, round-trip latency, etc.

Existing solutions require per-flow information

Packet latency, loss, out-of-order, etc.



Sender

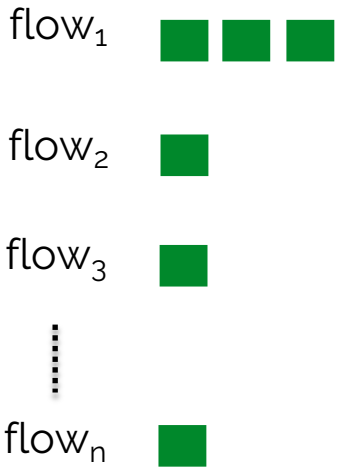


Flow$_1$: timestamps, counters

Flow$_2$: timestamps, counters

Flow$_3$: timestamps, counters

Per-flow storage: Marple [SIGCOMM'16], Dapper [SOSR'17]



Receiver

Keeping per-flow information is not scalable

Data center network with >30 Million flows [MACCDC'2012]

- **Track latency: ~ 240MB memory**
- **Track packet loss: >> 240MB memory**
- **Track out-of-order packets: ~ 240MB memory**

Hardware switch: 10s of MB memory

What we want: Lean algorithms

hundreds of millions flows

w/ several MBs of SRAM space

- **Memory usage:** Sublinear in the input size and the number of flows.
- **Practicality:** A few memory accesses to process each packet.

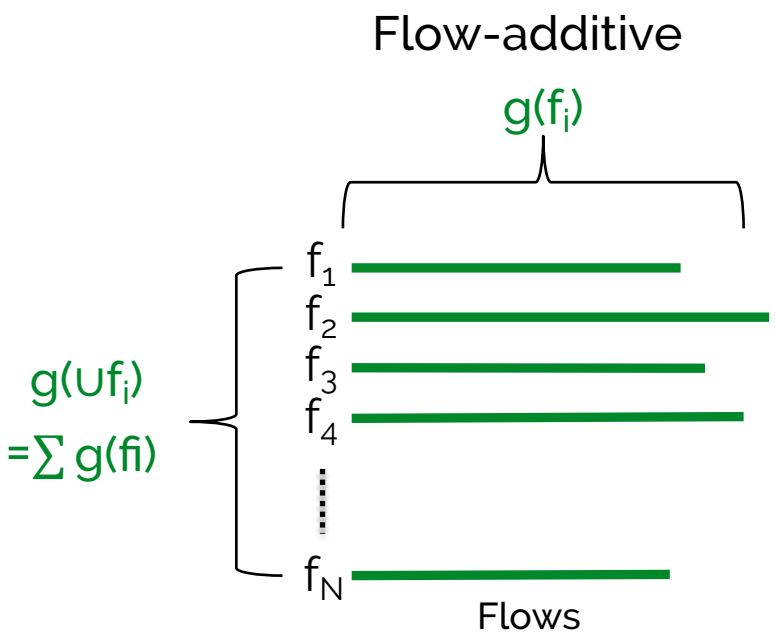
So: Are lean algorithms achievable?

There exists a lean algorithm for a performance metric $g(\cdot)$ if and only if:

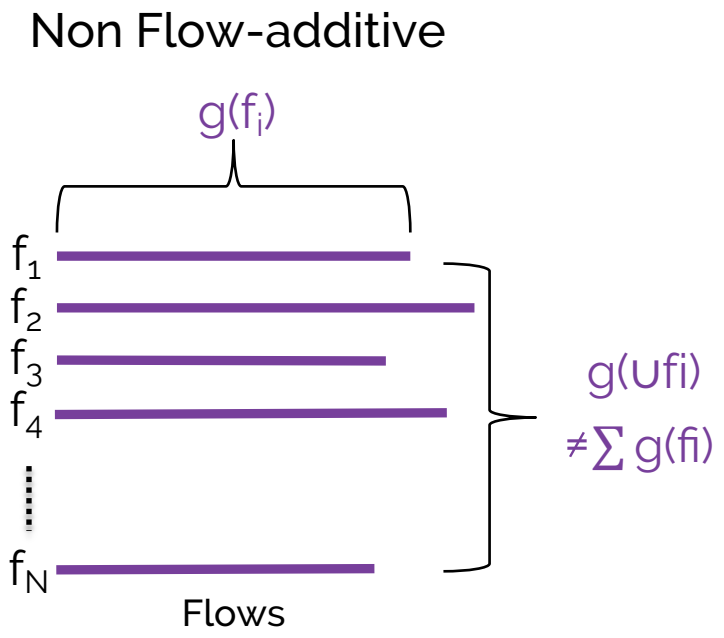
- g is a **flow-additive** function
- g is **flow-sublinear**

Check out our paper for more details!

Requirement 1: Flow-additive functions



g(·): Total Packet Latency, Total Packet Loss, Total Retransmitted Packets.



g(·): Max Packet Latency, Min Packet Loss Rate.

Requirement 2: Flow sublinear

For a network flow f_i ,

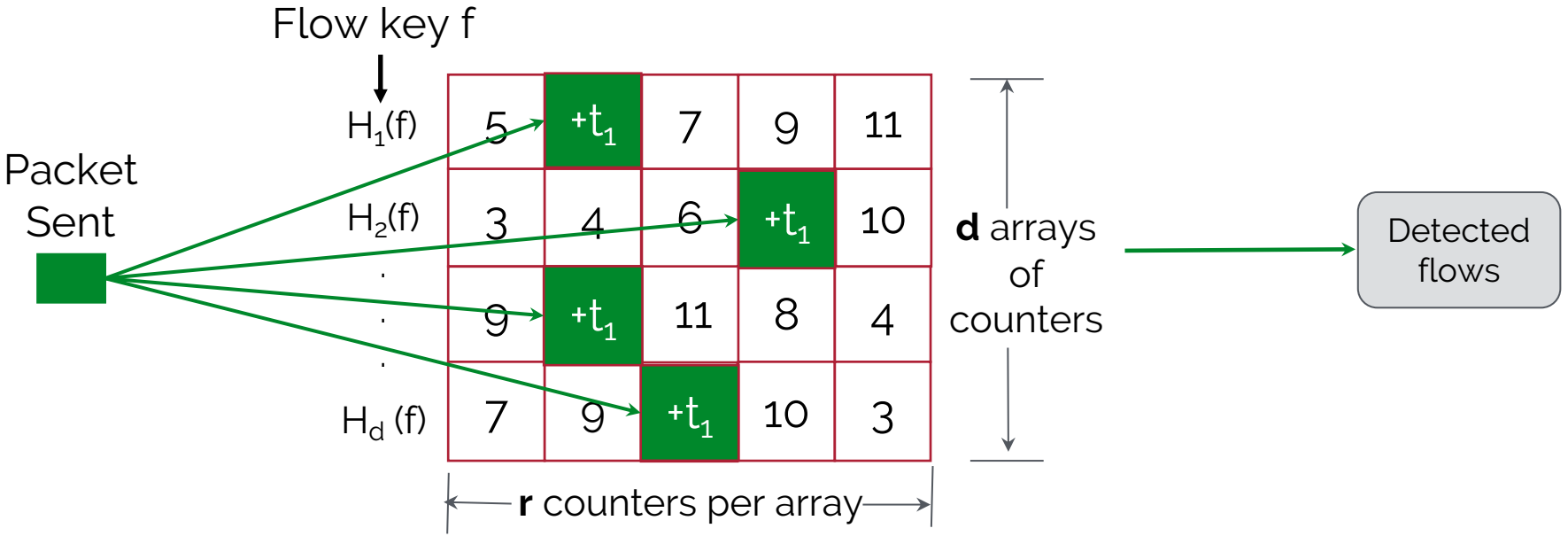
- if $g(f_i)$ can be estimated using space sublinear in $M(f_i)$, where $M(f_i)$ is the size of flow f_i ,

  Then g is **flow-sublinear**

Take-away: if g is not flow-sublinear, there is little hope g can be estimated across all flows.

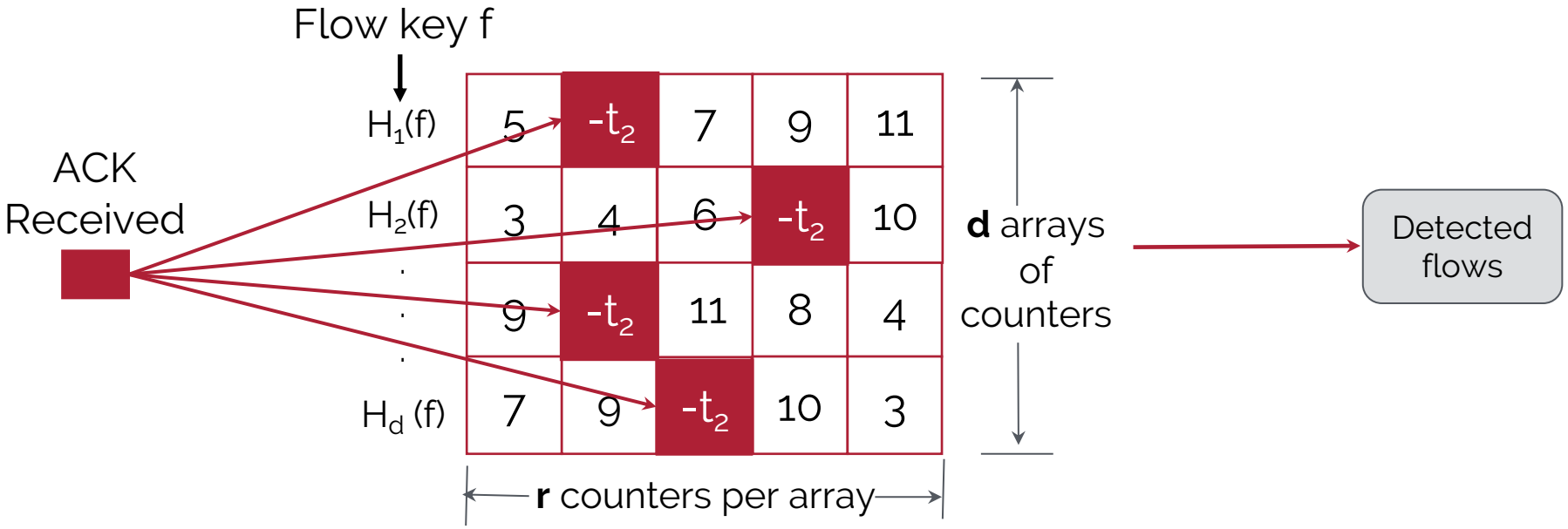
Lean algorithm example I: Round-trip latency

- RTT = total round trip latency for every packet in the traffic
- **Objective:** Detect flows whose total round trip time exceeds $\varepsilon \cdot$ RTT



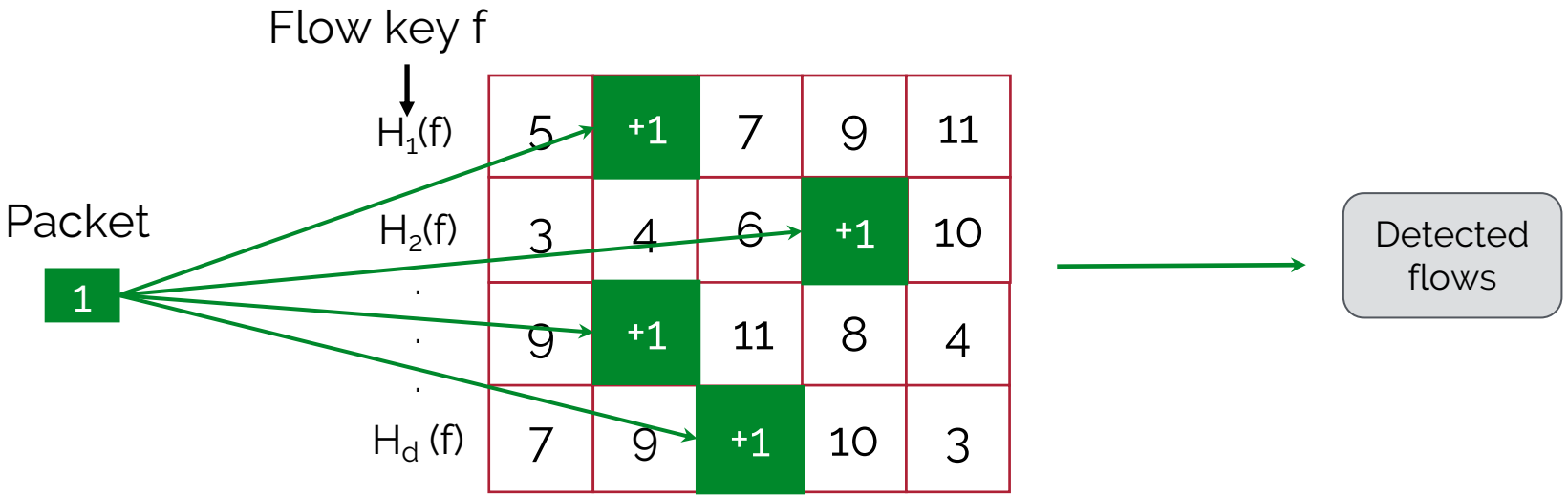
Lean algorithm example I: Round-trip latency

- RTT = total round trip latency for every packet in the traffic
- **Objective**: Detect flows whose total round trip time exceeds $\varepsilon \cdot$ RTT



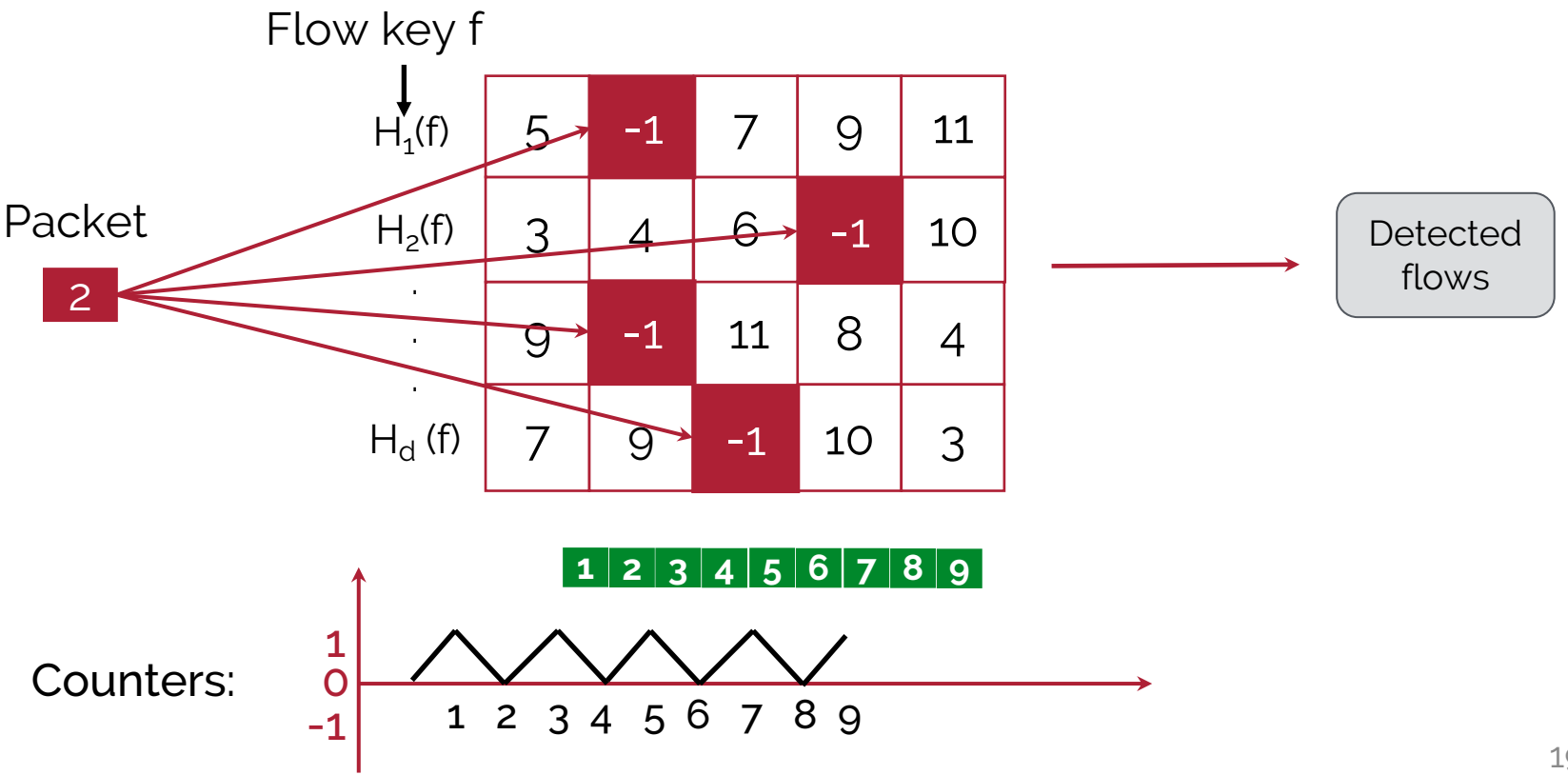
Lean algorithm example II: Packet loss

- **Objective:** Detect flows with high packet loss
- **Assumption:** Random packet loss



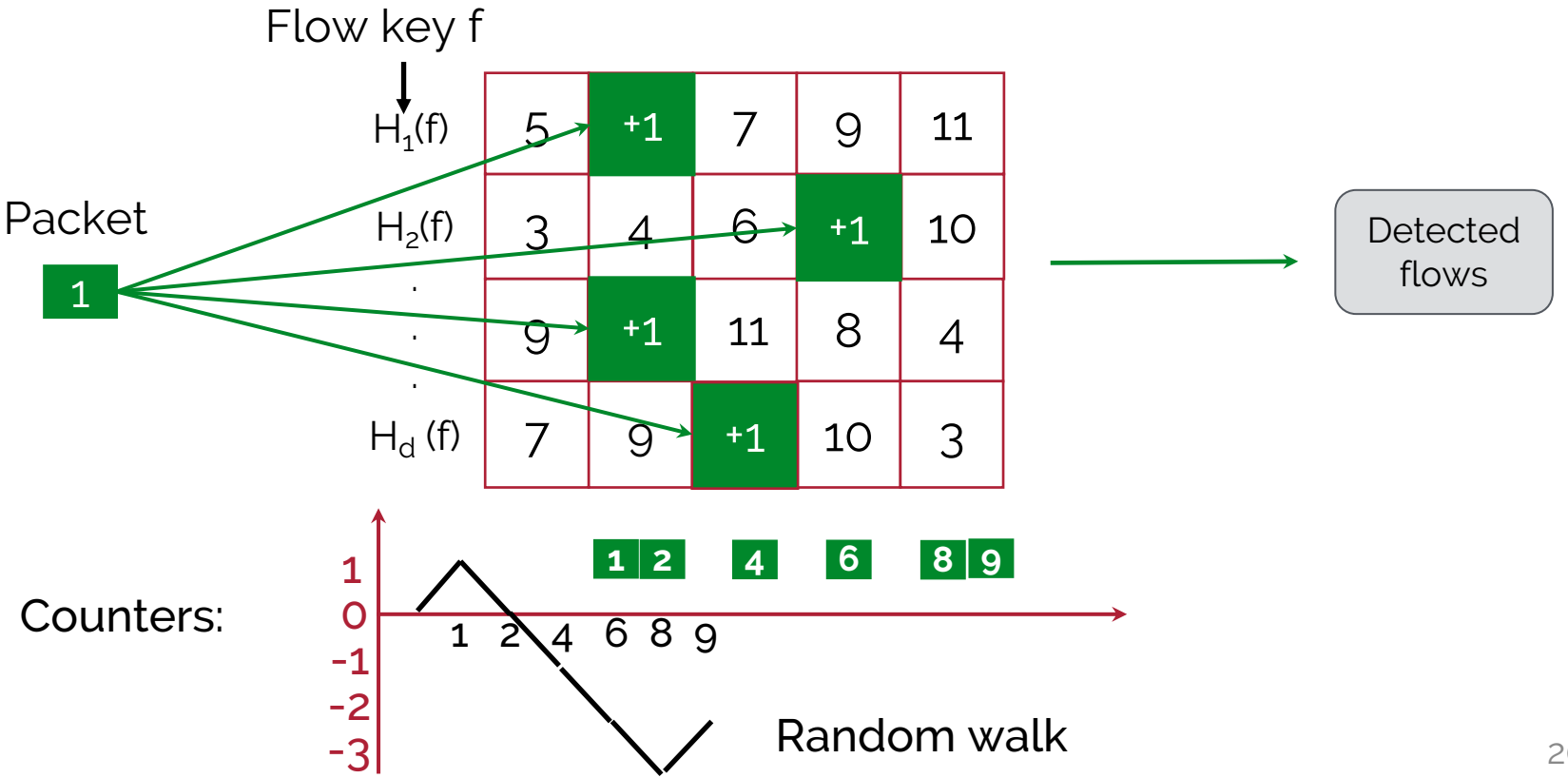
Lean algorithm example II: Packet loss

- If there is no packet loss



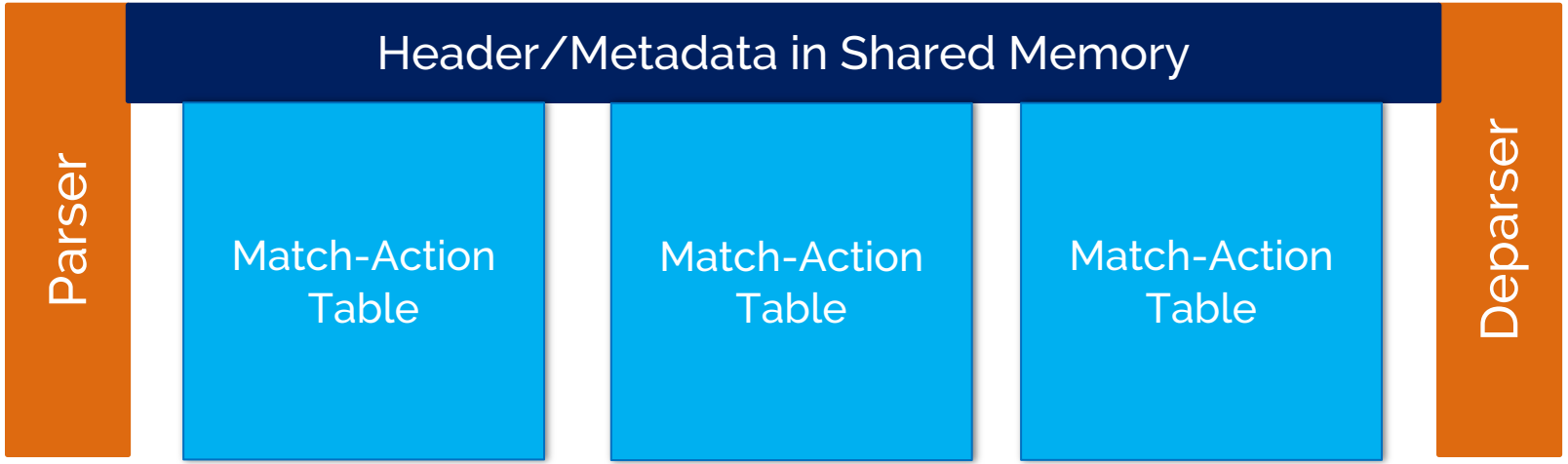
Flow-additive example II: Packet loss

- If there is random packet loss



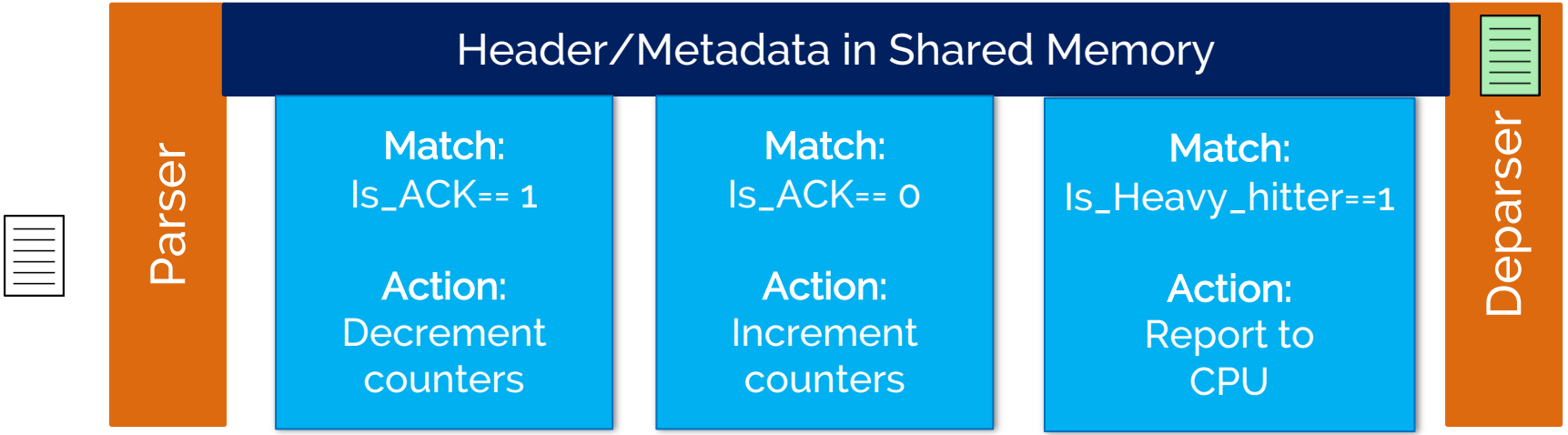
Implementation on programmable switches

TCP Packet Format:



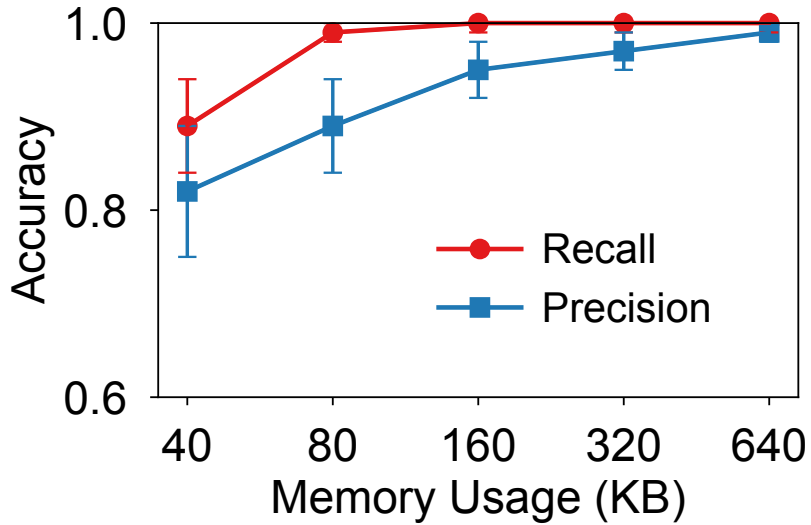
Implementation on programmable switches

TCP Packet Format:



Lean algorithms on detecting high latency flows

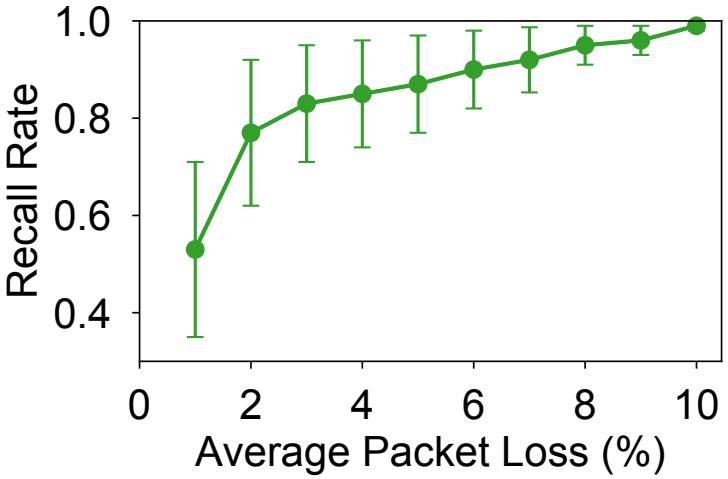
- 3,700,000 flows in a public Internet trace.
- Start with 5 rows of 2000 32-bit counters (40KB).

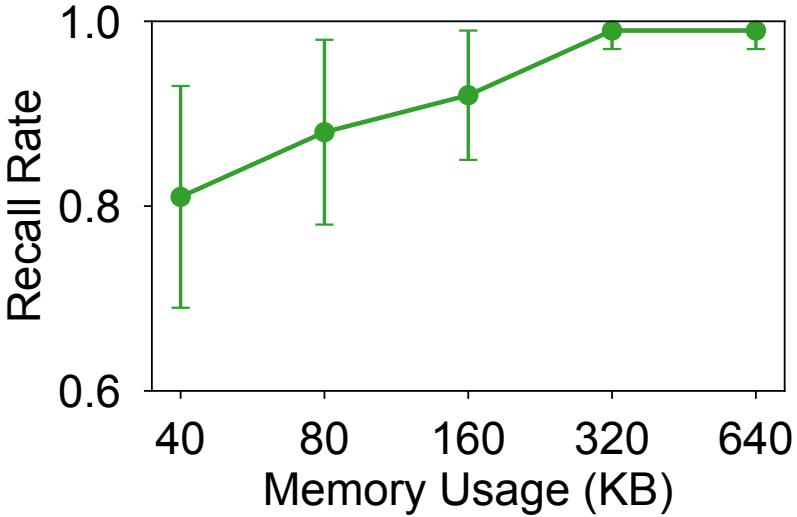


Approximately detect top 100 high latency flows

Lean algorithms on detecting lossy flows

- 3,700,000 flows in a public Internet trace.
- Start with 5 rows of 2000 32-bit counters (40KB).





Approximately detect top 100 lossy flows

Conclusions

- Network performance statistics are important for network diagnosis.
- Existing solutions require per-flow information to detect problematic flows.
- Lean algorithms can be designed for a set of performance statistics
- **Future directions:**
- Other performance statistics: high and low TCP sending/receiving windows.
- Explore real-world characteristics: Are packet loss random?
  Are retransmitted packets and packet losses correlated?
- Build diagnosis tools using lean algorithms.

Thank you!