Network traffic monitoring

- Security Issues
- Performance Issues
- Accounting
- Misconfiguration

Get insights to perform management actions

Efficient Network Monitoring Applications in the Kernel with eBPF and XDP

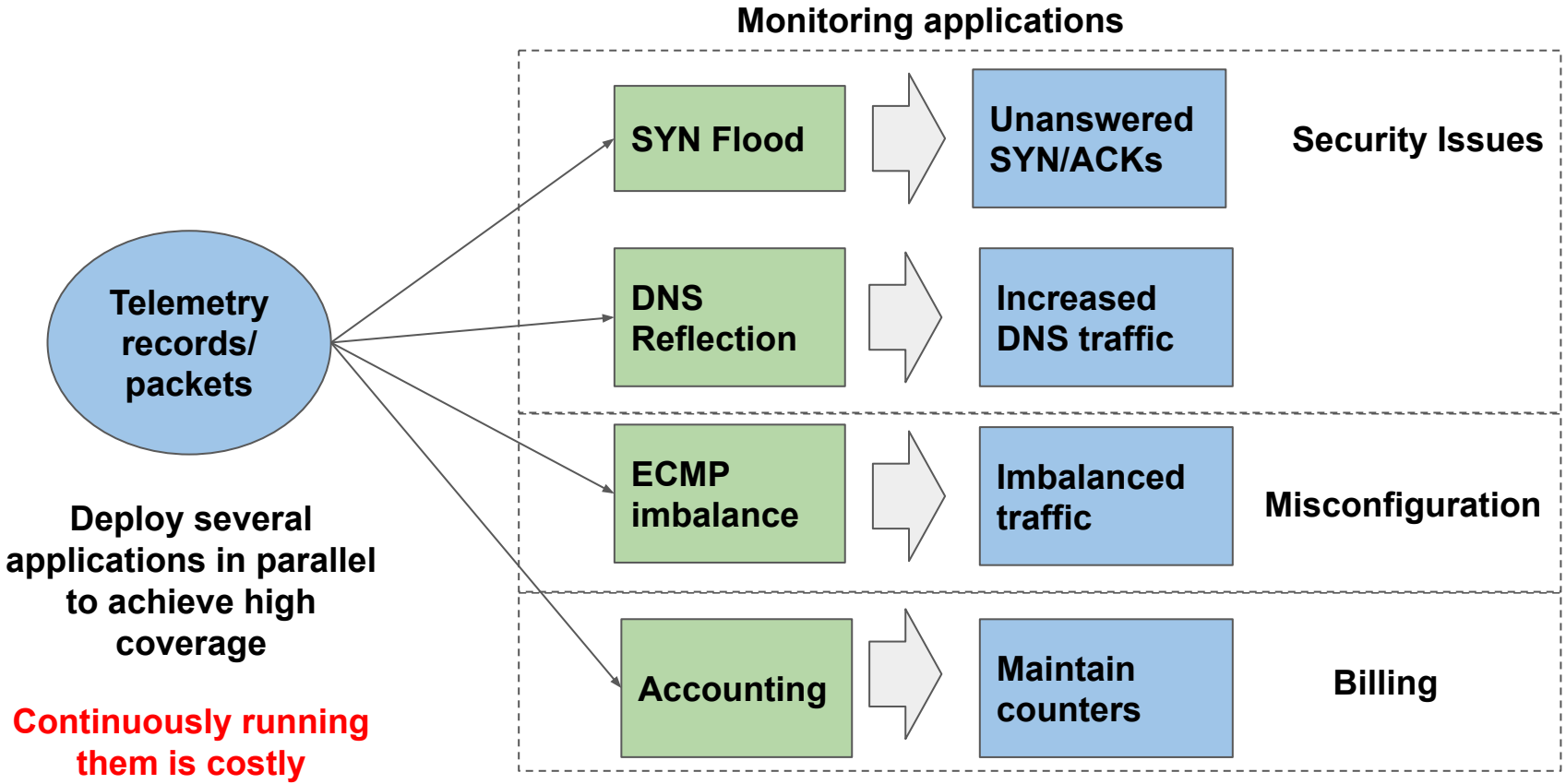
Marcelo Abranches, Oliver Michel, Eric Keller, and Stefan Schmid

University of Colorado Boulder

PRINCETON UNIVERSITY

Technische Universität Berlin

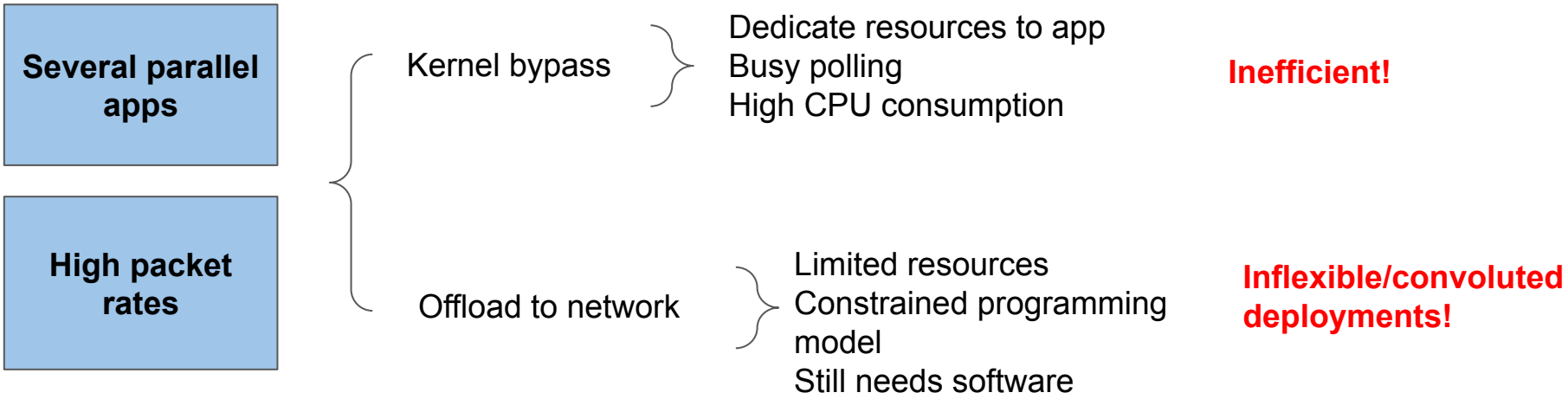
We need continuous high coverage



Deploy several applications in parallel to achieve high coverage

Continuously running them is costly

Network monitoring can be resource inefficient and inflexible



What can we do to address those challenges?

Support efficiently deployments of parallel applications

Provide better resource footprint and flexibility

How can we do it?

Avoid redundant processing

Provide shared high-level statistics

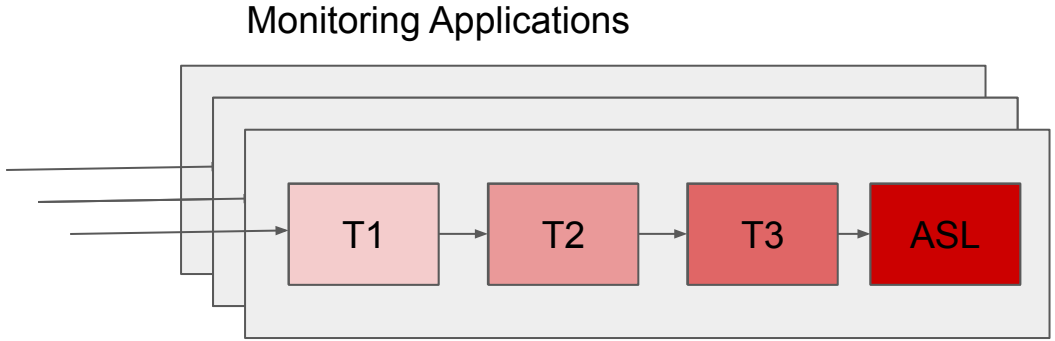
Execute applications only when needed

Build on top of an efficient and flexible data plane

How can we address the challenges of current monitoring systems?

Key opportunities

Despite differences, monitoring applications share similarities



Redundant logic!
All apps process all packets!

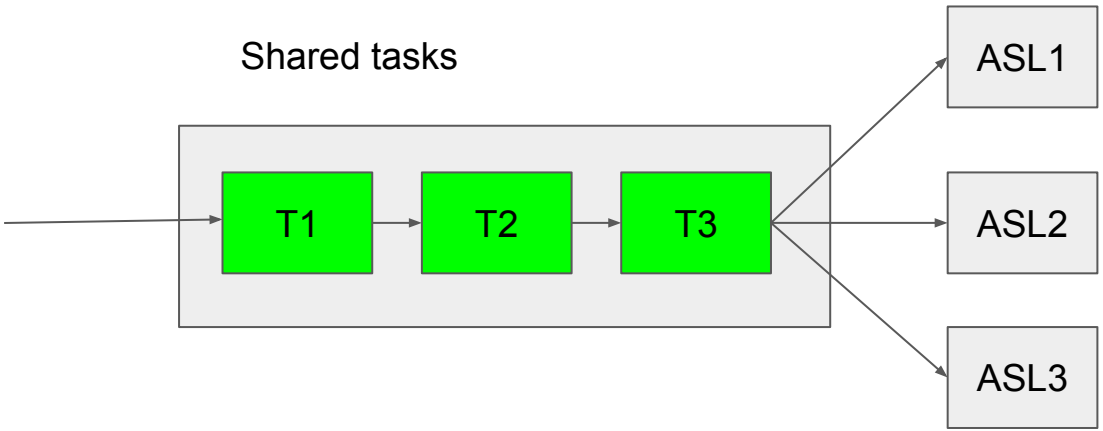
T1 -> Ingest, parse and select packets of interest

T2 -> Compute relevant metrics for the application

T3 -> If conditions are met, send packet for application processing

ASL -> Execute application specific logic, detect conditions and generate events

We can avoid redundant processing

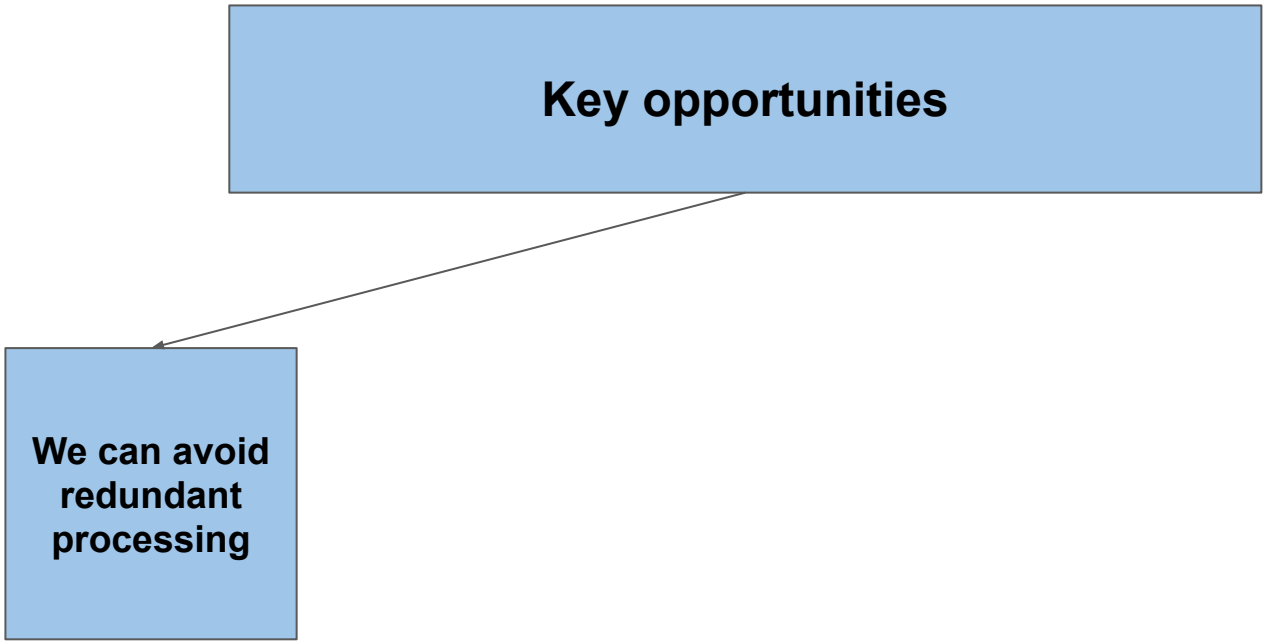


No redundant processing!

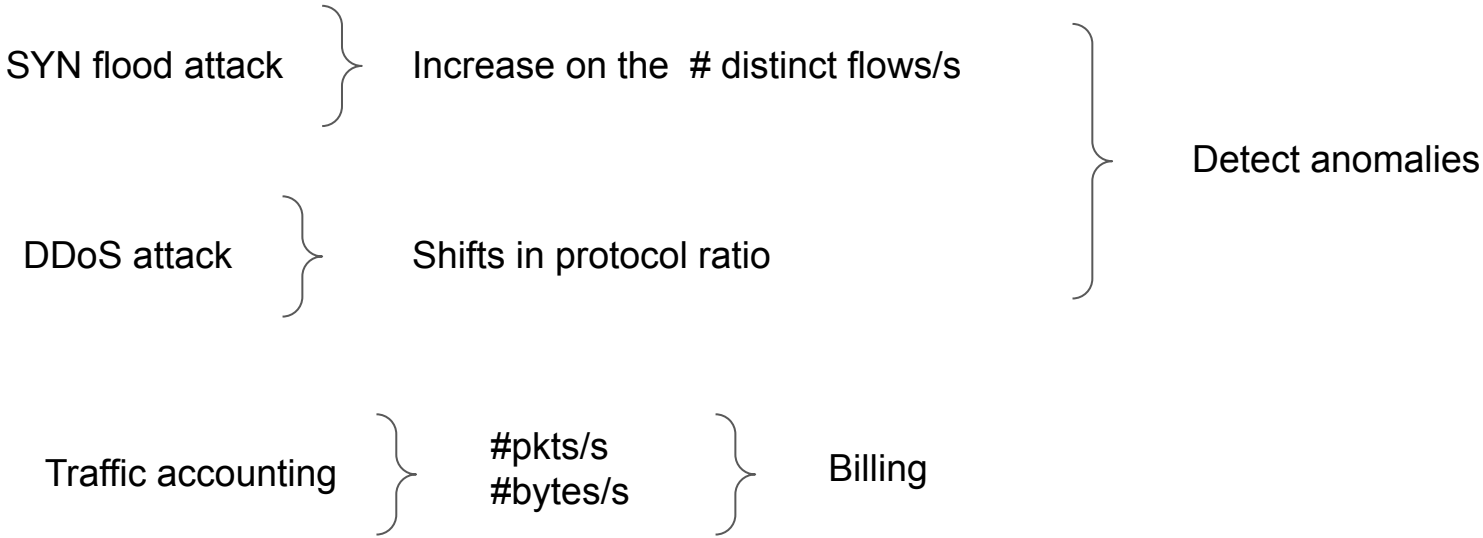
Small resource footprint!

Developers can write slimmer applications!

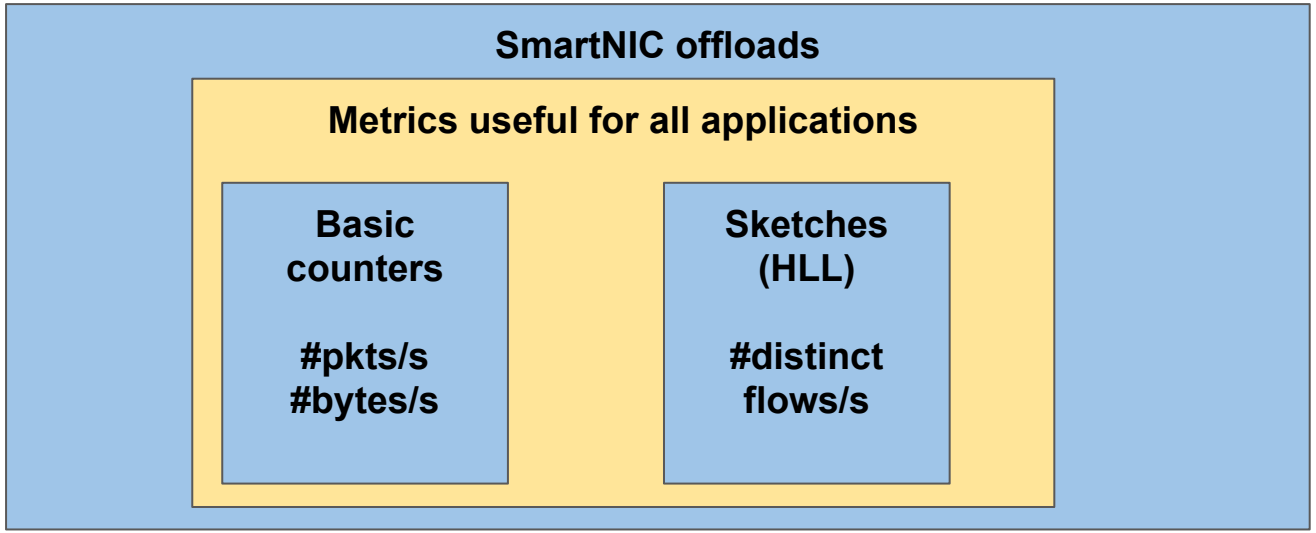
How can we address the challenges of current monitoring systems?



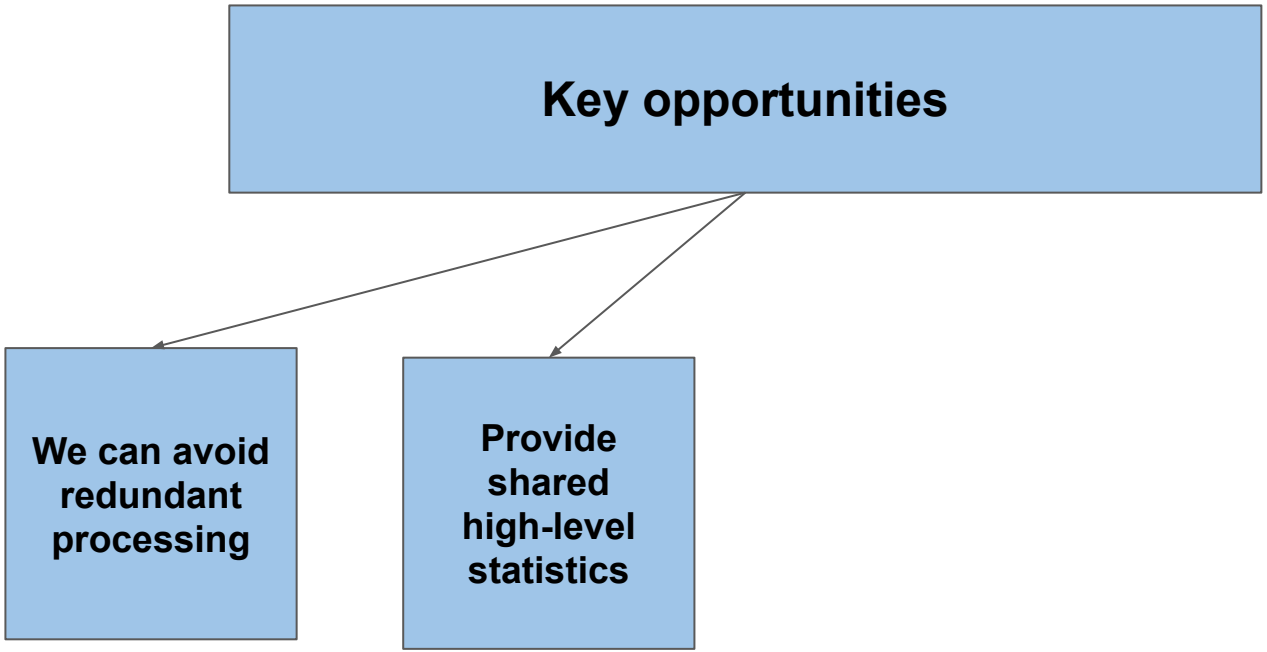
Provide shared high-level statistics



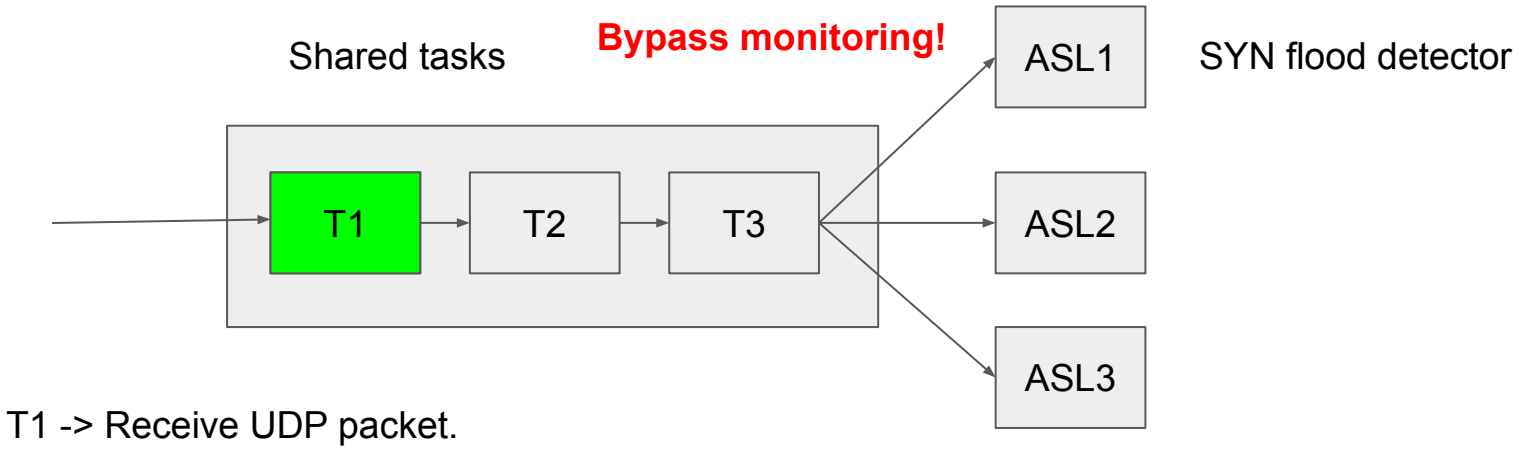
Provide shared high-level statistics



How can we address the challenges of current monitoring systems?

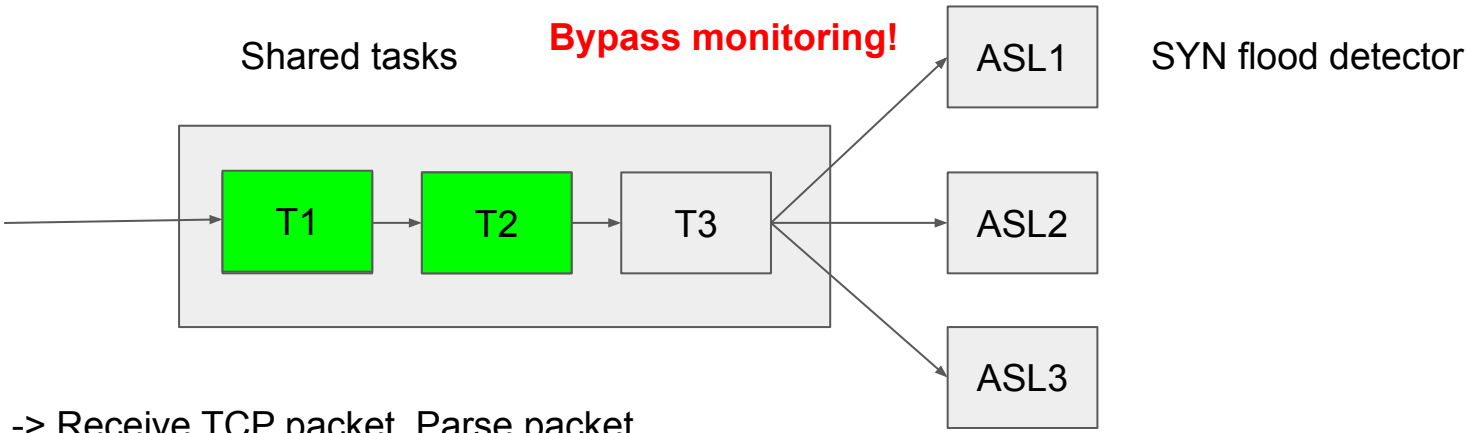


Execute applications only when needed



T1 -> Receive UDP packet.

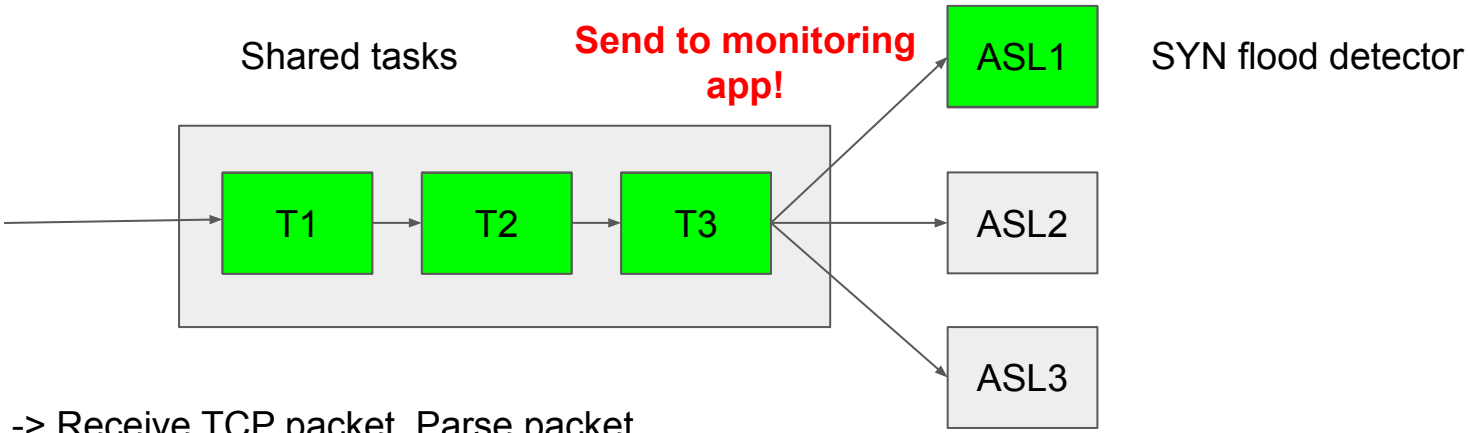
Execute applications only when needed



T1 -> Receive TCP packet. Parse packet.

T2 -> Compute # of distinct flows. **Verify # flows/s < threshold**.

Execute applications only when needed



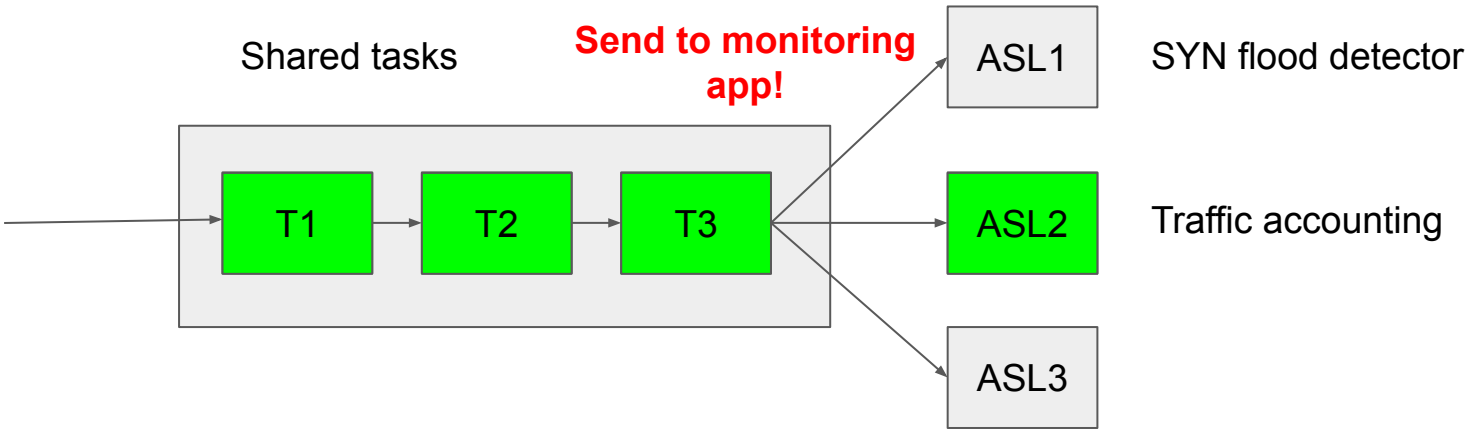
T1 -> Receive TCP packet. Parse packet.

T2 -> Compute # of distinct flows. **Verify # flows/s > threshold.**

T3 -> Send to ASL1 (SYN flood detector)

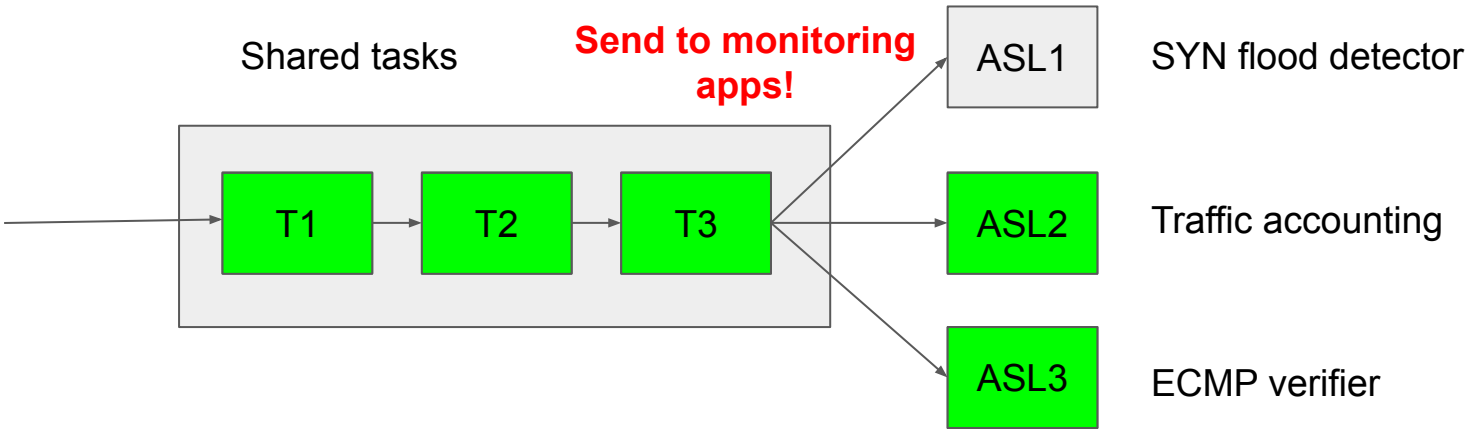
ASL1 -> Identify attack src and dst. Generate event. Apply filter.

Execute applications only when needed



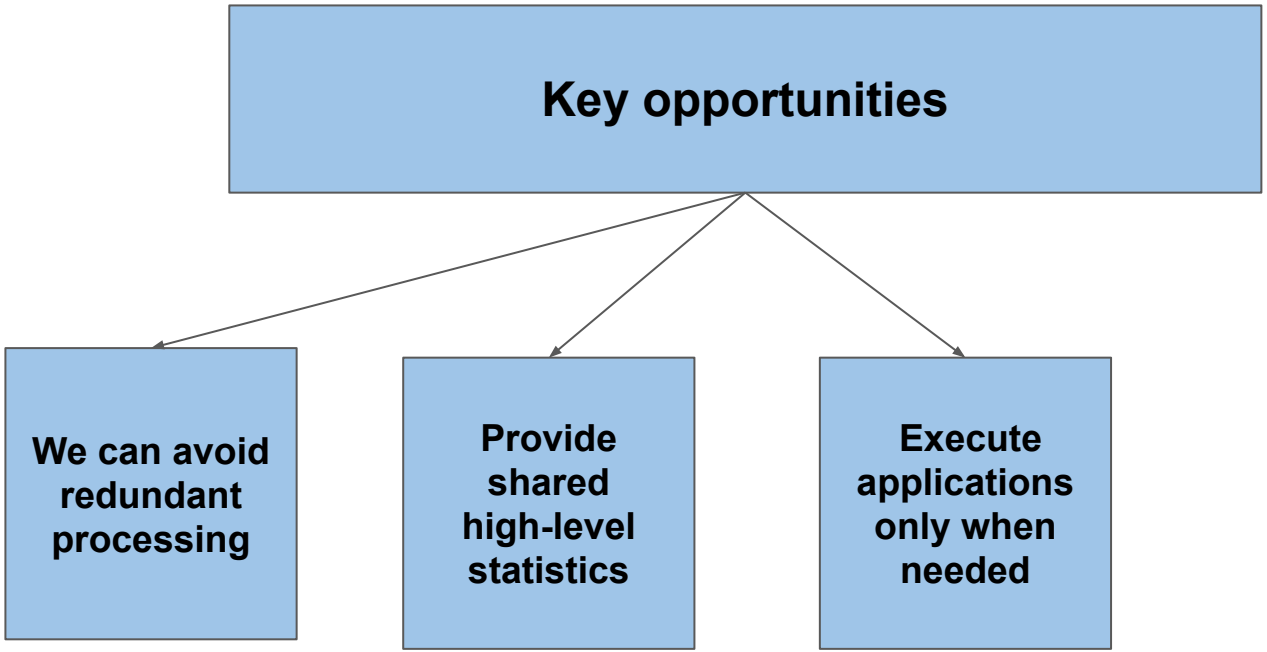
Some applications need to be always on

Execute applications only when needed

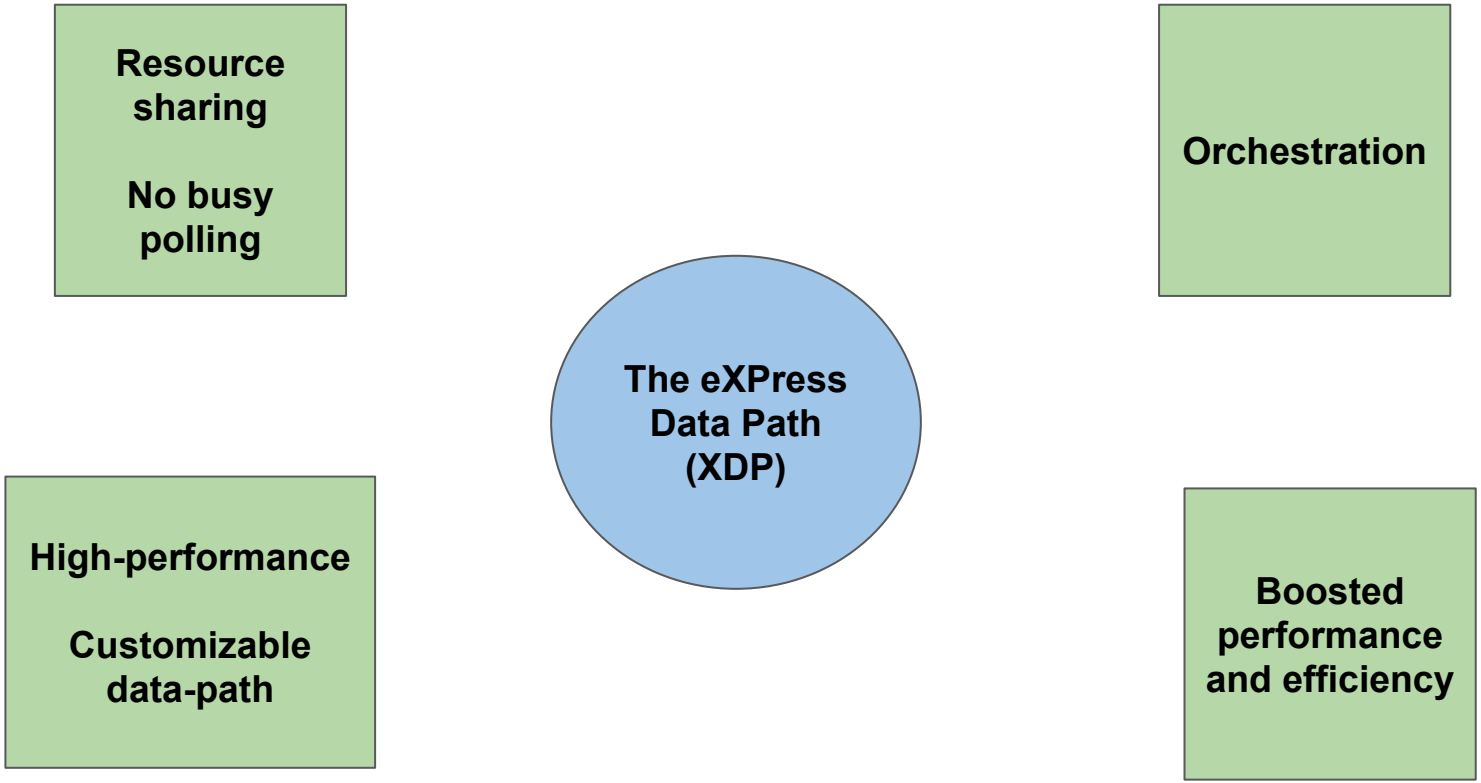


Operators may want to deploy applications on demand

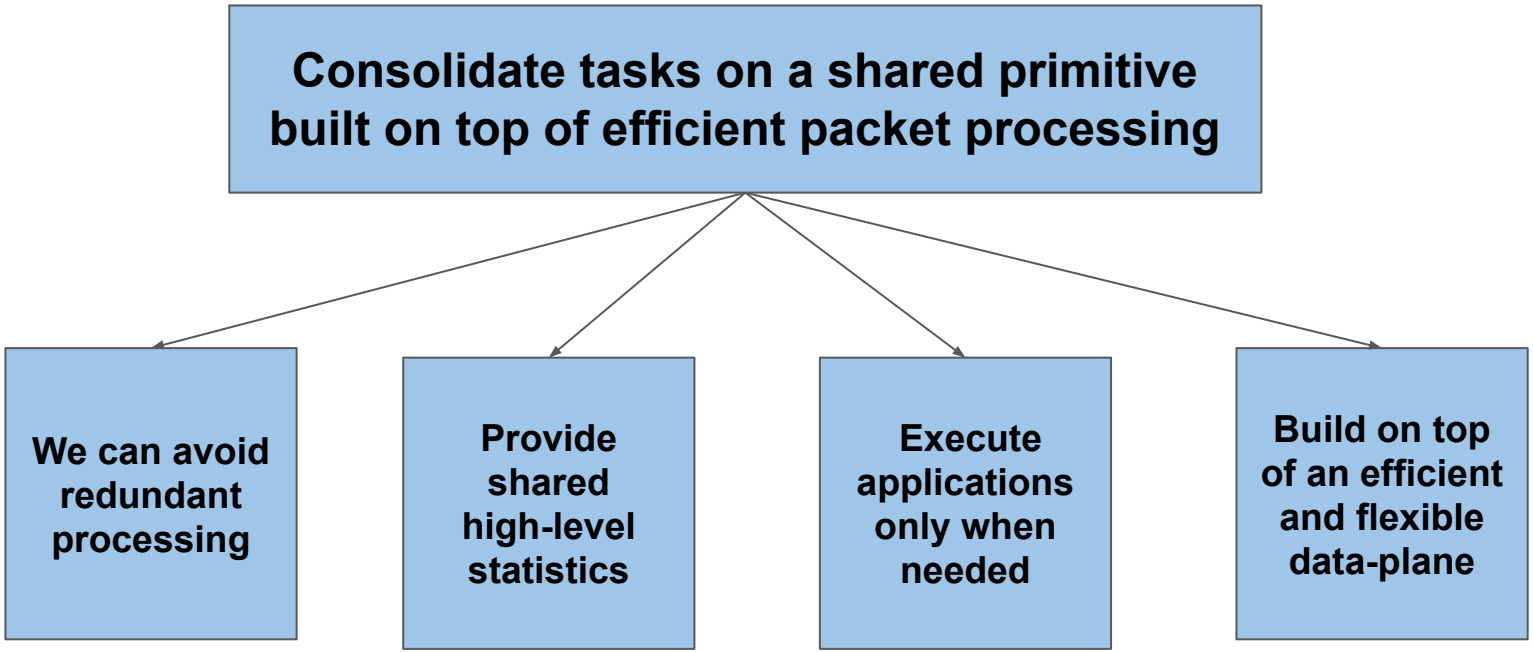
How can we address the challenges of current monitoring systems?



Build on top of an efficient and flexible data-plane

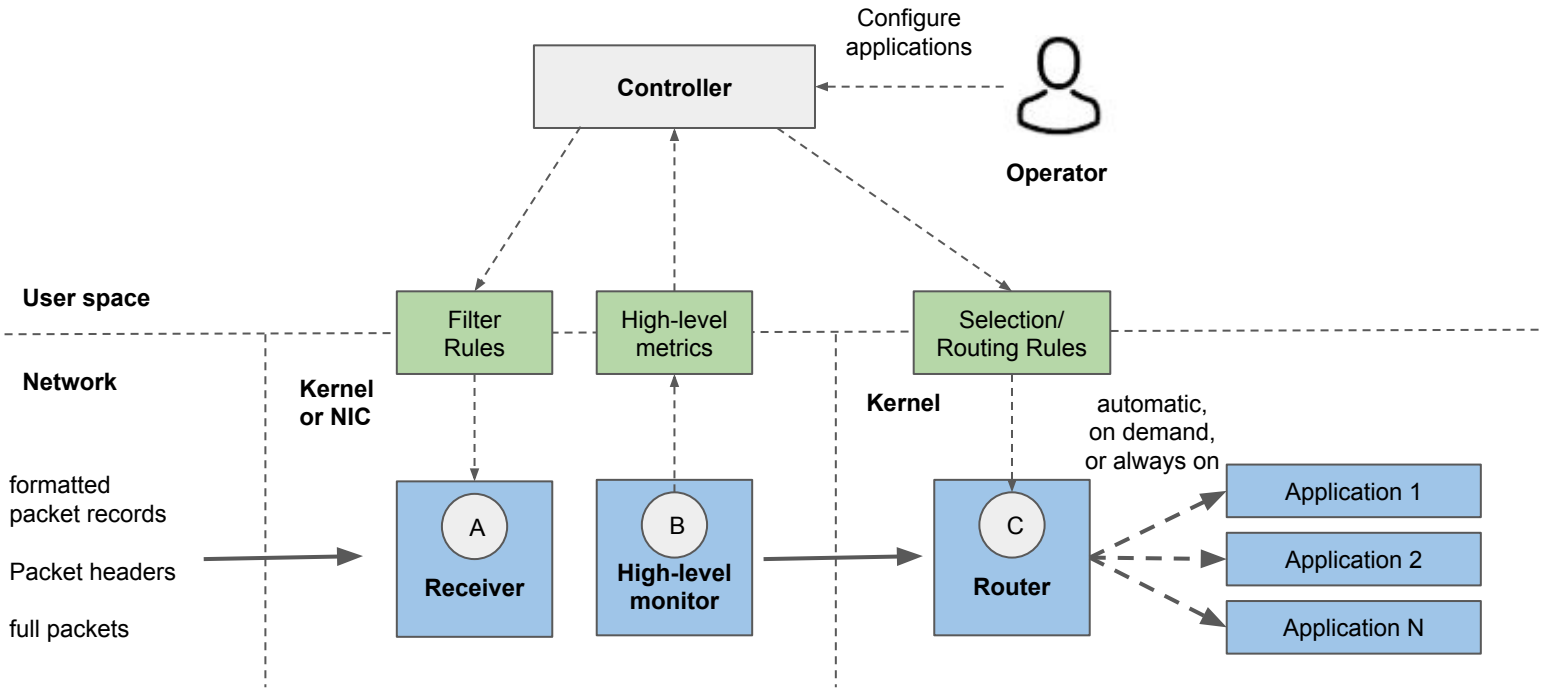


How can we address the challenges of current monitoring systems?

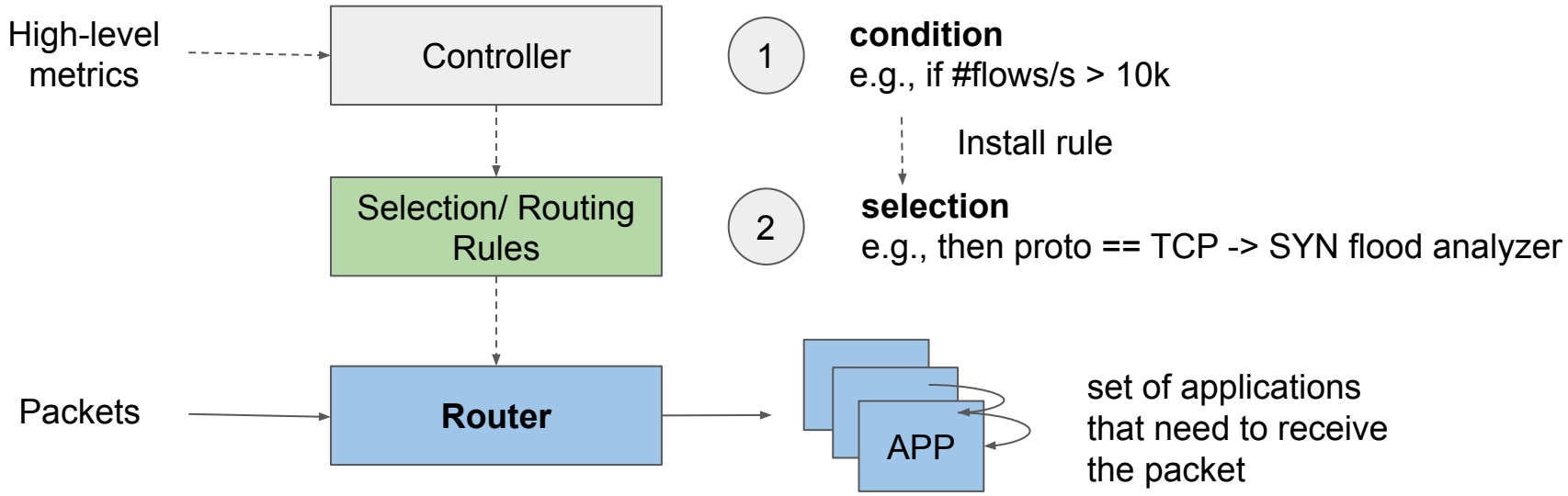


Efficient high-performance high-coverage monitoring system

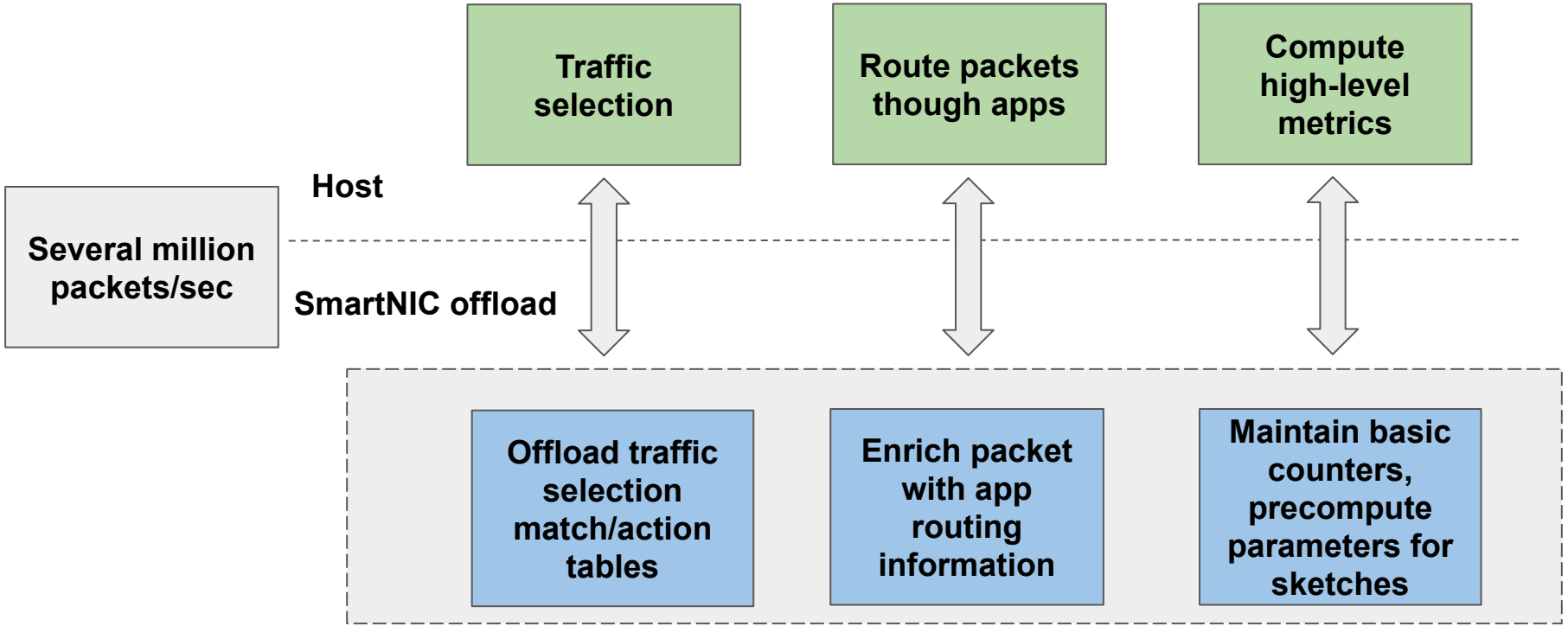
Our system architecture



Router overview



Boosting efficiency and performance



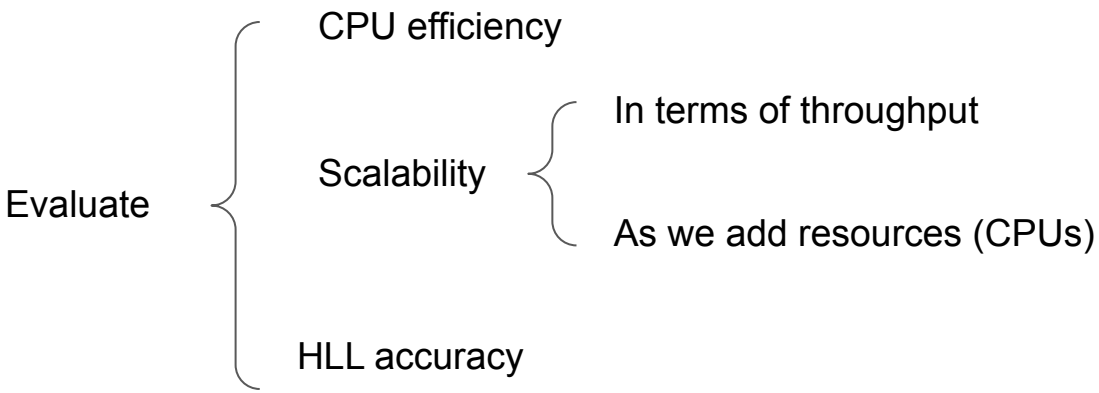
Example applications

Traffic accounting { Counts # of packets and bytes per IP destination

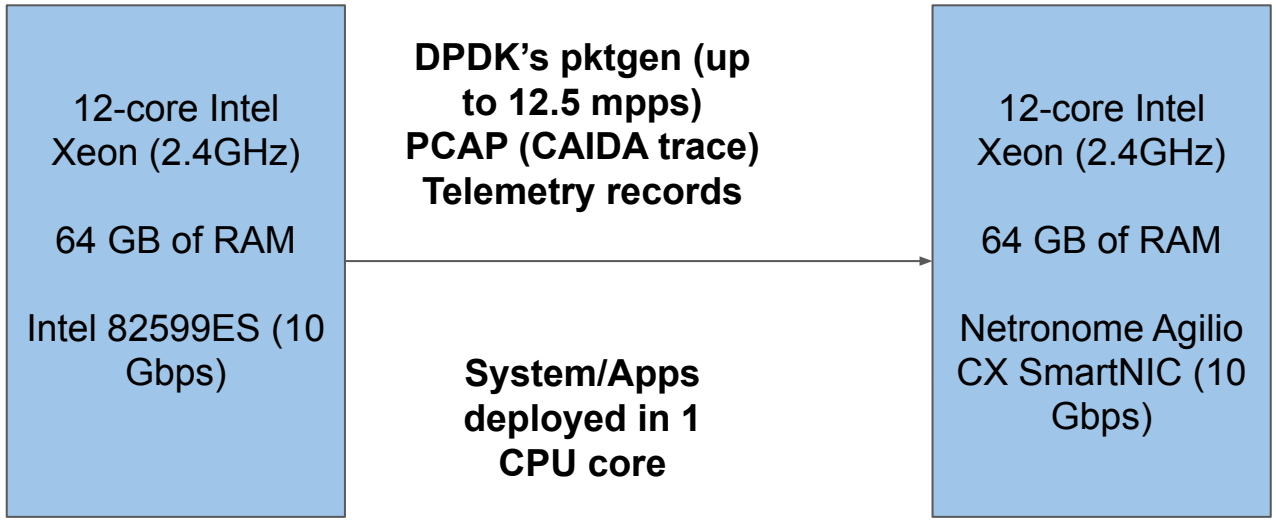
Half open TCP connections analyzer { Tracks incomplete TCP handshakes

DNS flow analyzer { Maintains per flow statistics of DNS traffic

Evaluation

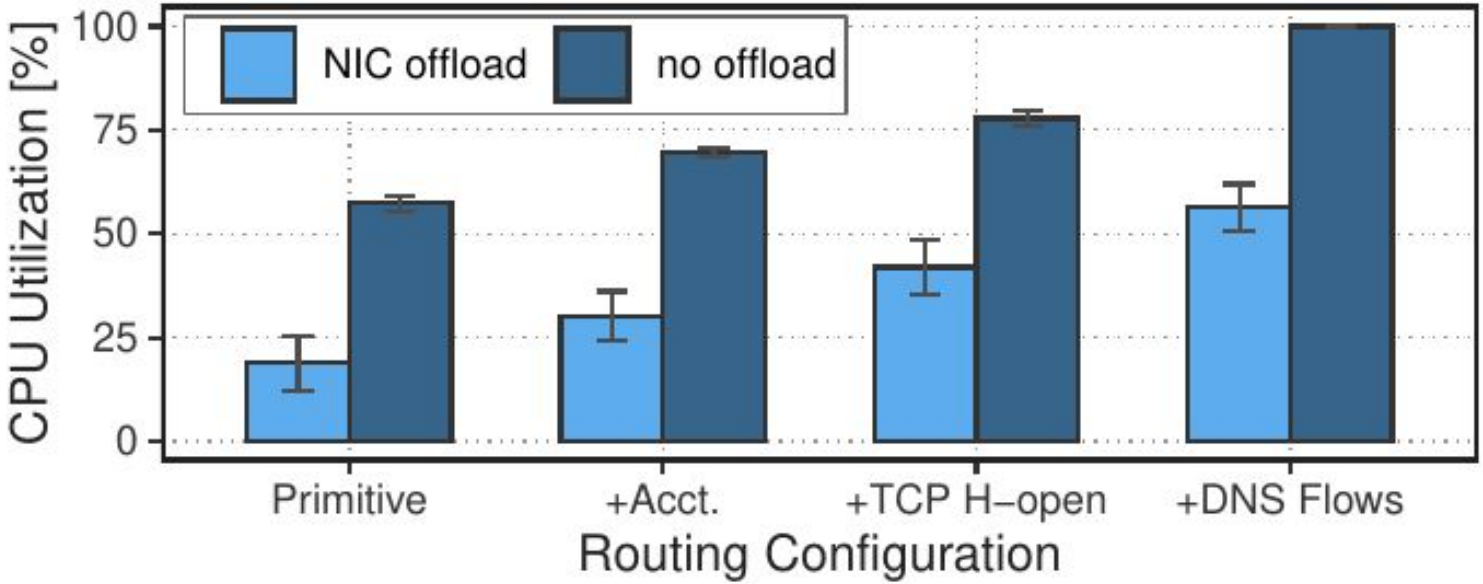

Evaluate
CPU efficiency
Scalability
In terms of throughput
As we add resources (CPUs)
HLL accuracy

Evaluation setup

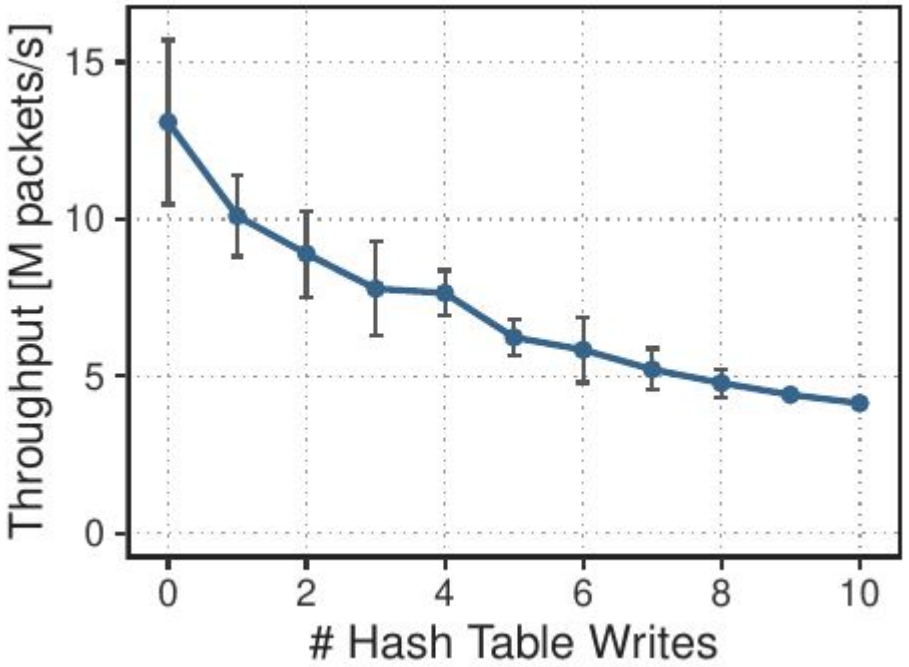


Efficiency in CPU utilization

Offered load 2 mpps

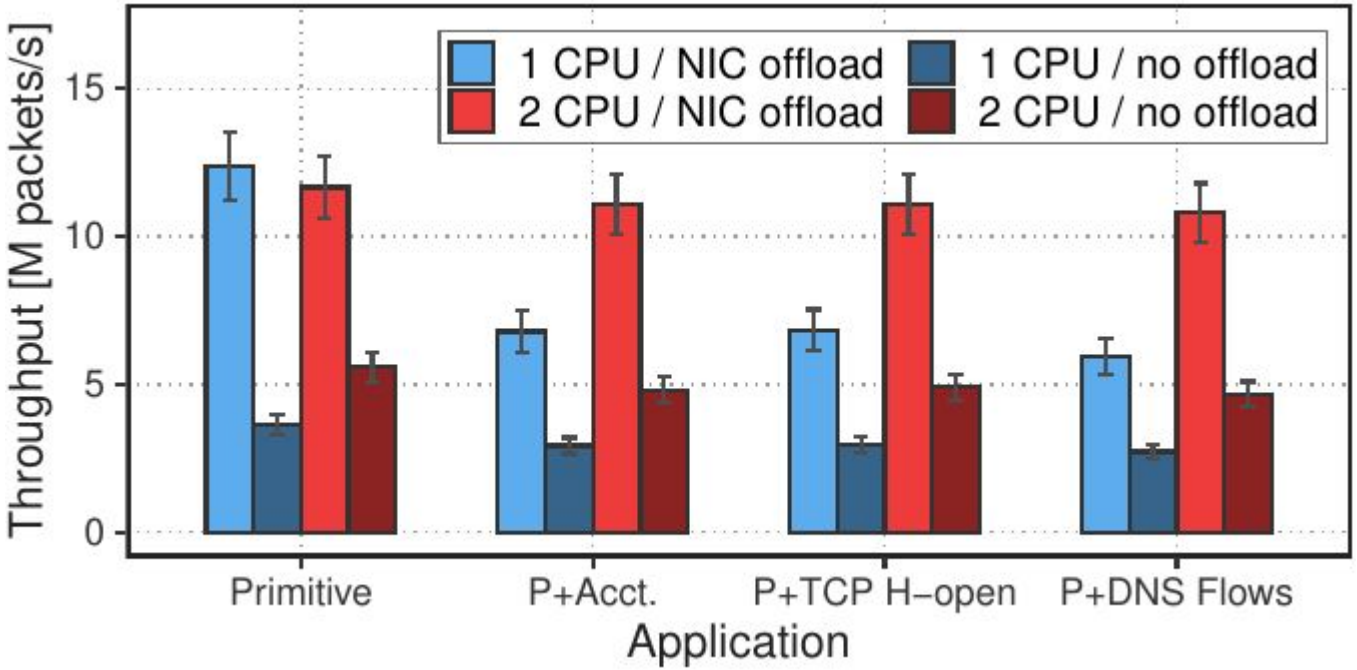


Scalability in terms of throughput



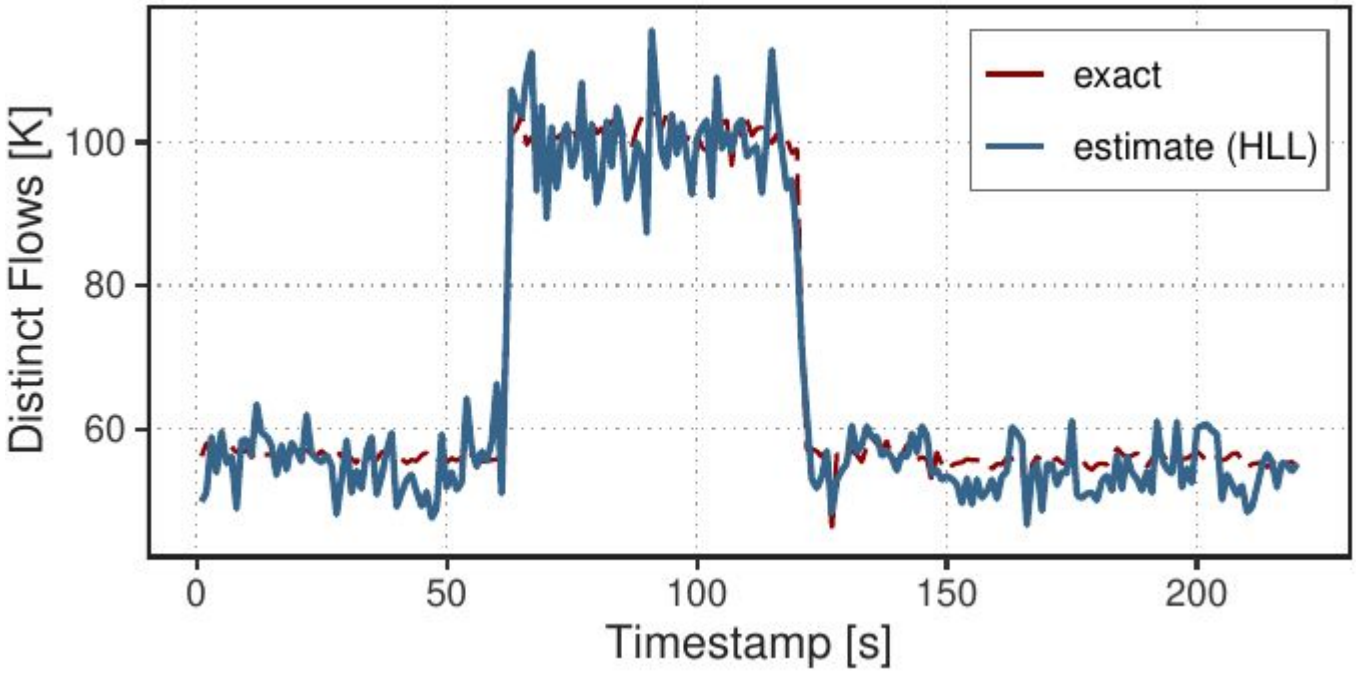
Scalability as we add resources

Offered load 12.5 mpps



High-level monitoring accuracy

Inject SYN flood attack for 60s



Conclusion

High-performance/high-coverage continuous monitoring can be simple, flexible, light and efficient

Thank you!