SwiSh: Distributed Shared State Abstractions for Programmable Switches

Lior Zeno, Dan R. K. Ports, Jacob Nelson, Daehyeok Kim, Shir Landau Feibish, Idit Keidar, Arik Rinberg, Alon Rashelbach, Igor De-Paula, Mark Silberstein

Microsoft

ACSL Accelerated Computing Systems Lab

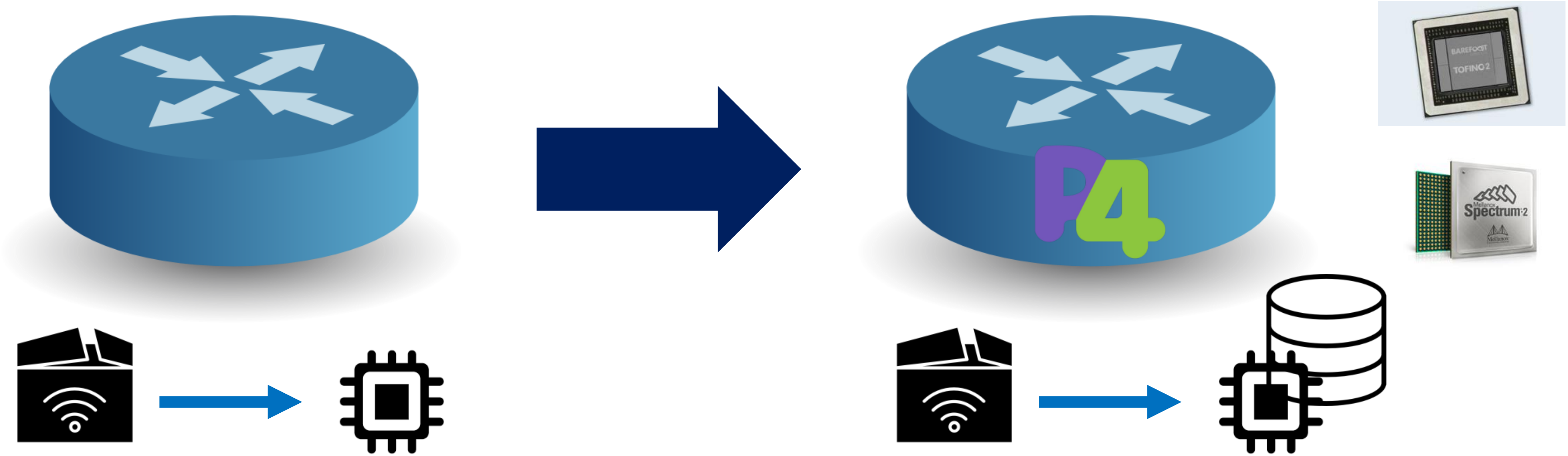
האוניברסיטה הפתוחה
THE OPEN UNIVERSITY OF ISRAEL
الجامعة المفتوحة

TECHNION Israel Institute of Technology

Stateful Packet Processing

Fixed-function Switch

Programmable Switch



Stateless processing

Stateful processing

Current Trend: In-Switch Acceleration

SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs [SIGCOMM 2017]

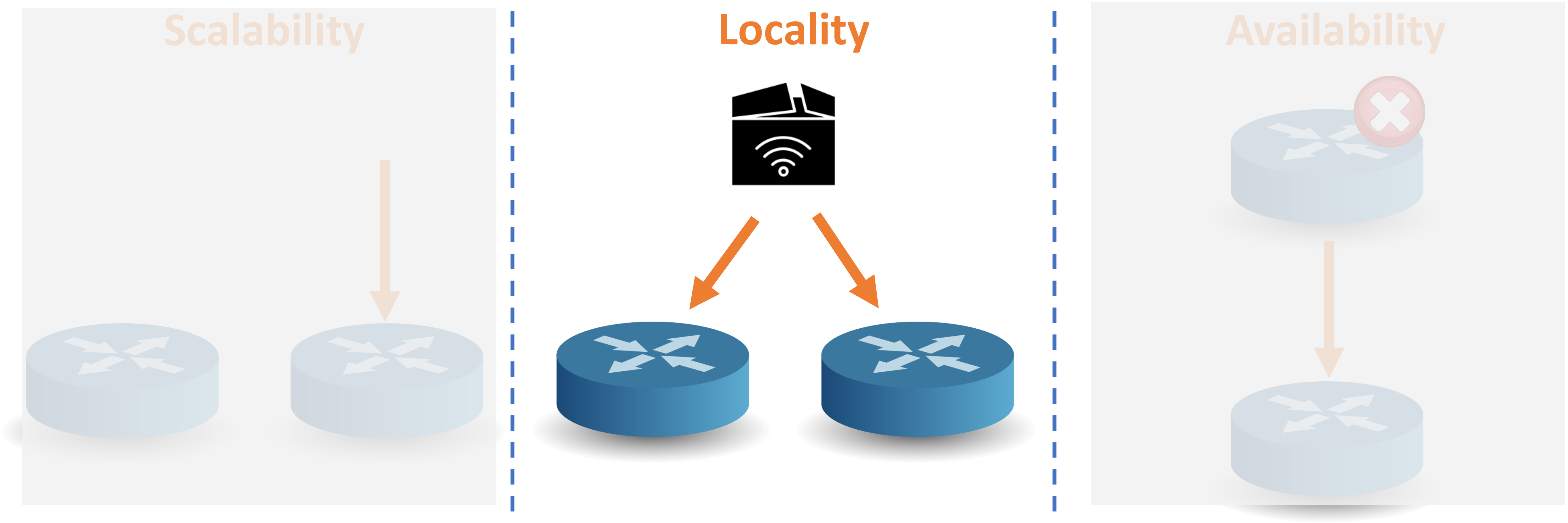
Offloading Real-time DDoS Attack Detection to Programmable Data Planes [IM 2019]

Heavy-Hitter Detection Entirely in the Data Plane [SOSR 2017]

Cheetah: Accelerating Database Queries with Switch Pruning [SIGMOD 2020]

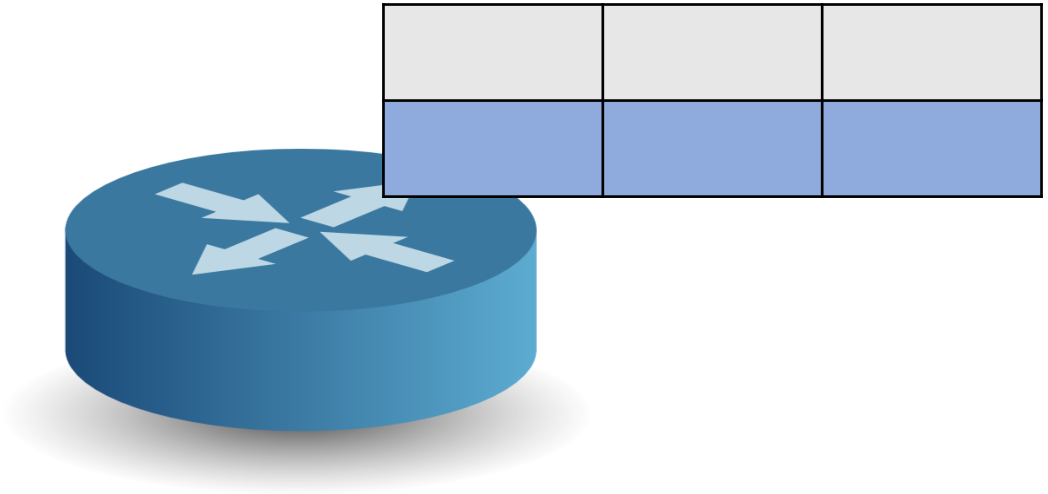
Designed for a single-switch

The Case for Many-Switch Designs

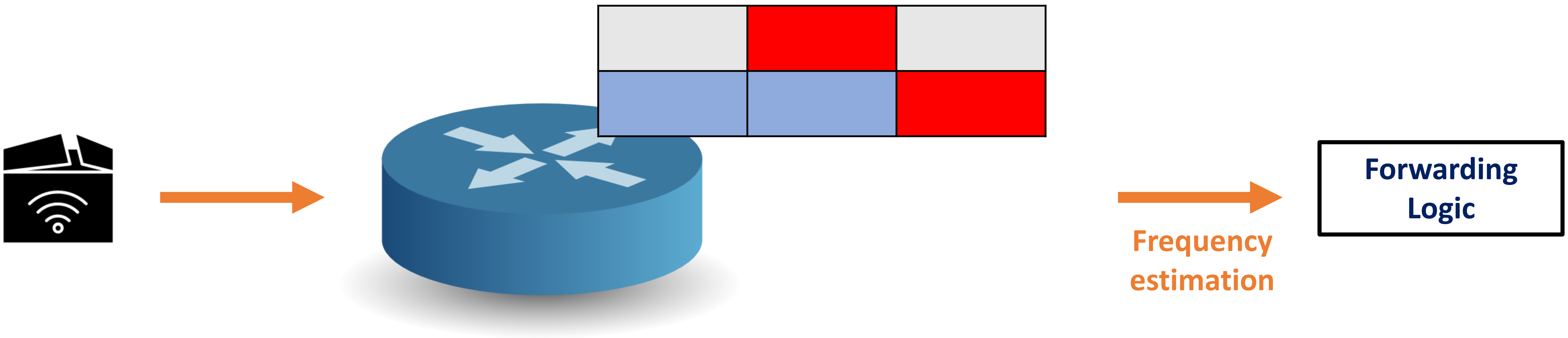


Not all information is available on all switches

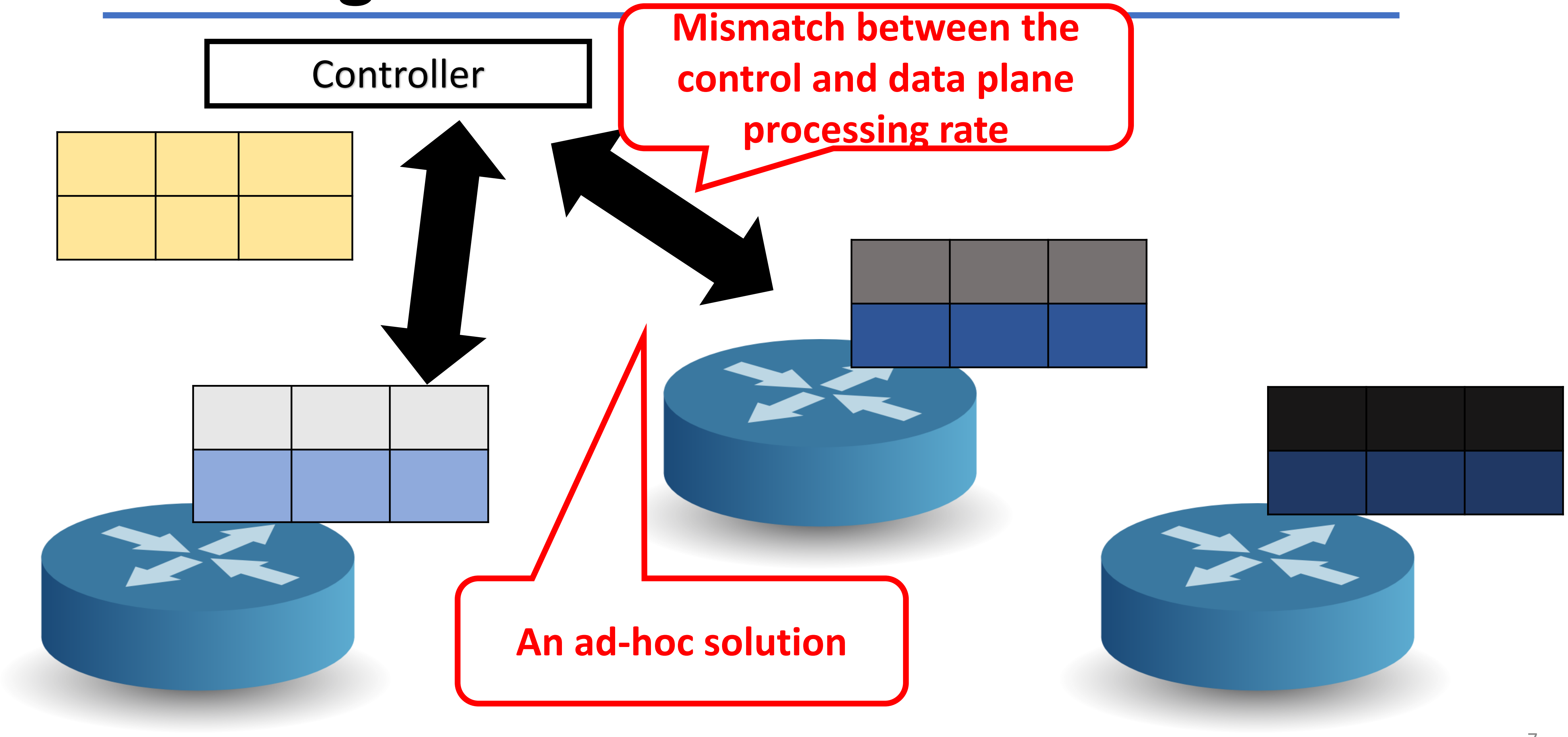
Example: Reactive Applications (DDoS detector)



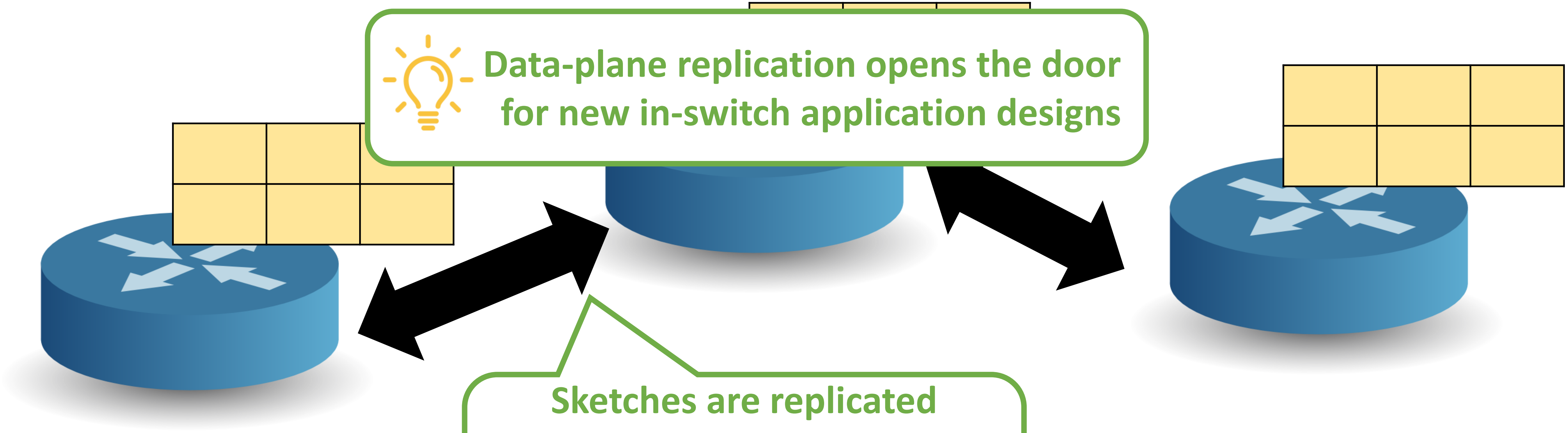
Example: Reactive Applications (DDoS detector)



Challenge: Network-Wide DDoS Detector



Our Work: Data Plane Replication



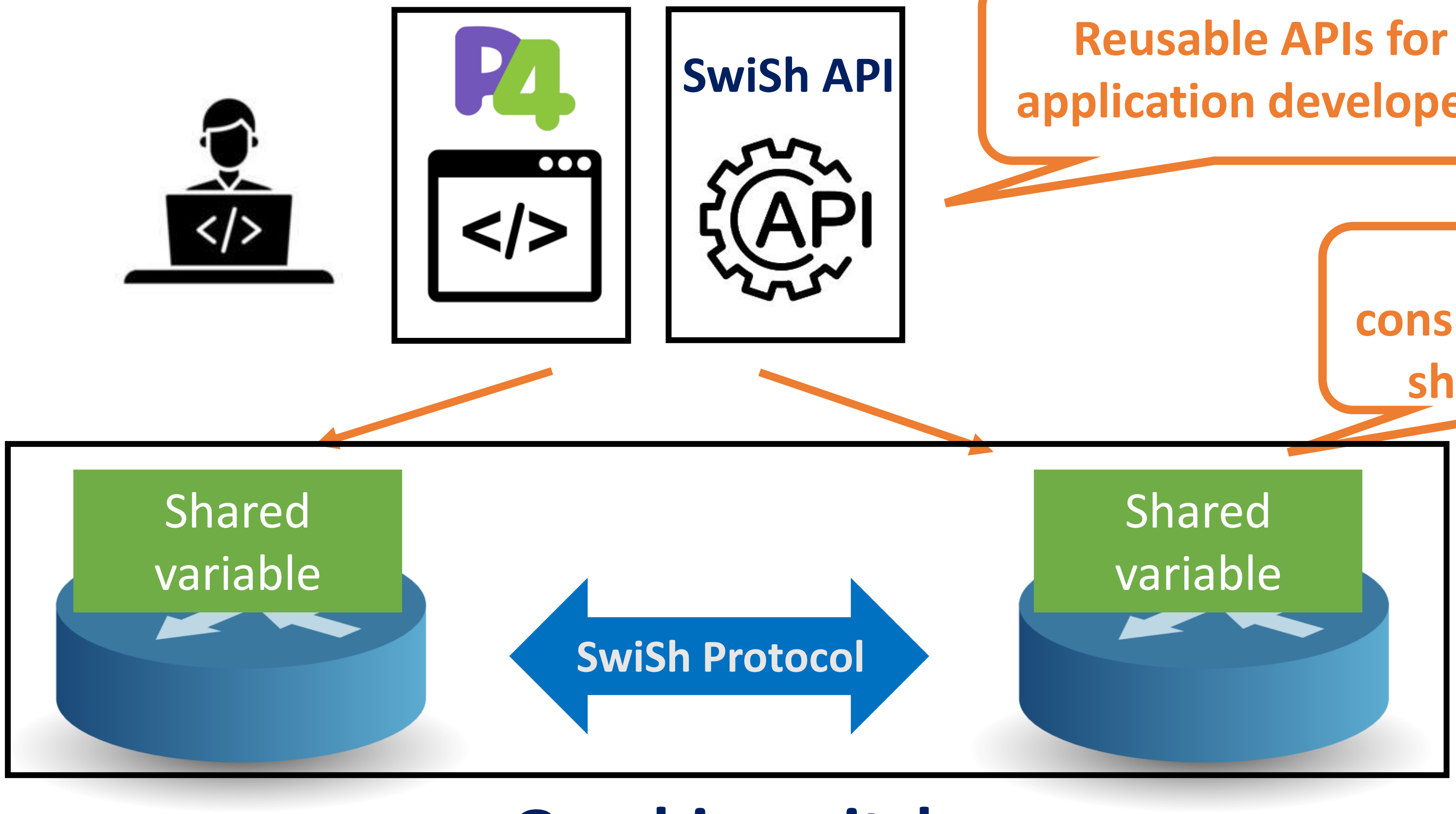
Data-plane replication opens the door for new in-switch application designs

Sketches are replicated entirely in the data-plane with provable consistency guarantees

Agenda

- The case for data-plane replication
- SwiSh design and challenges
- Experimental results

SwiSh Design



Reusable APIs for application developers

3 different consistency levels for shared variables

One big switch

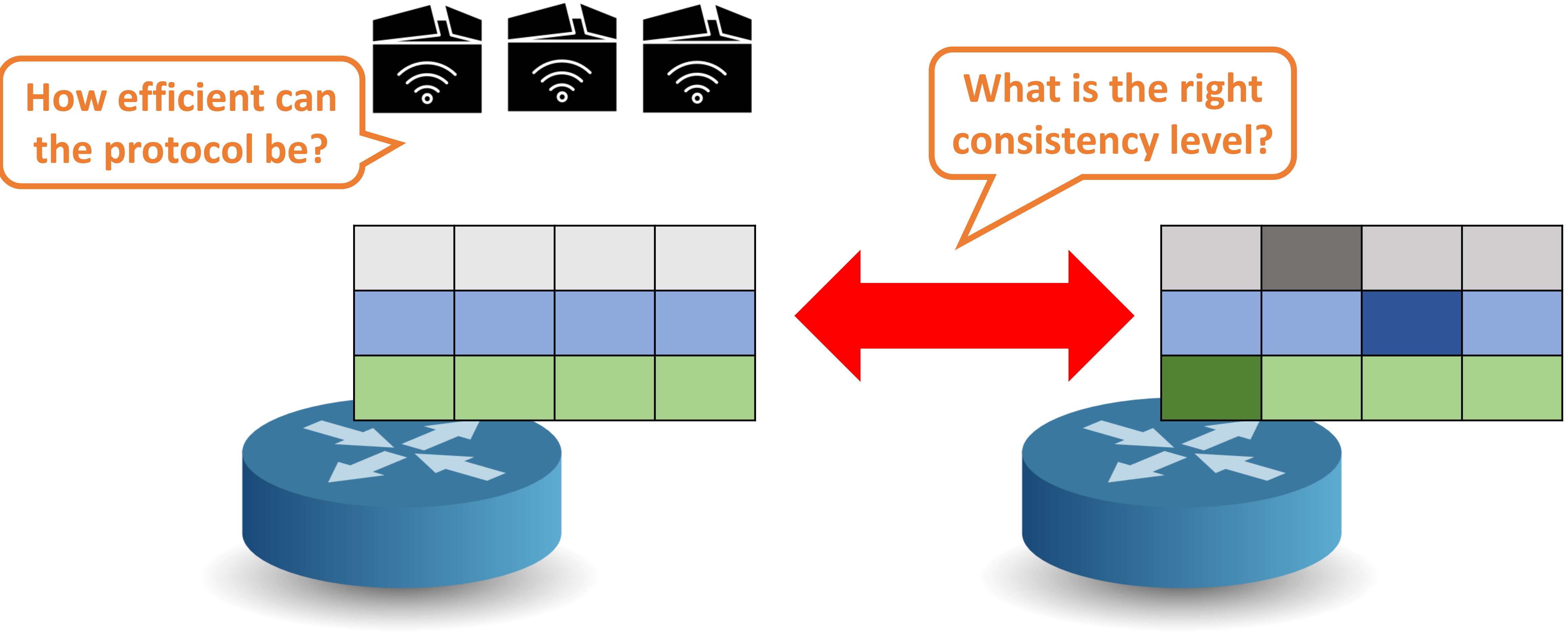
In-Switch Replication Protocols

- Strong Read-Optimized (SRO)
  - NAT
- Eventual Write-Optimized (EWO)
  - Rate limiter
- **Strong Delay-Writes (SDW)**
  - Sketch-based applications

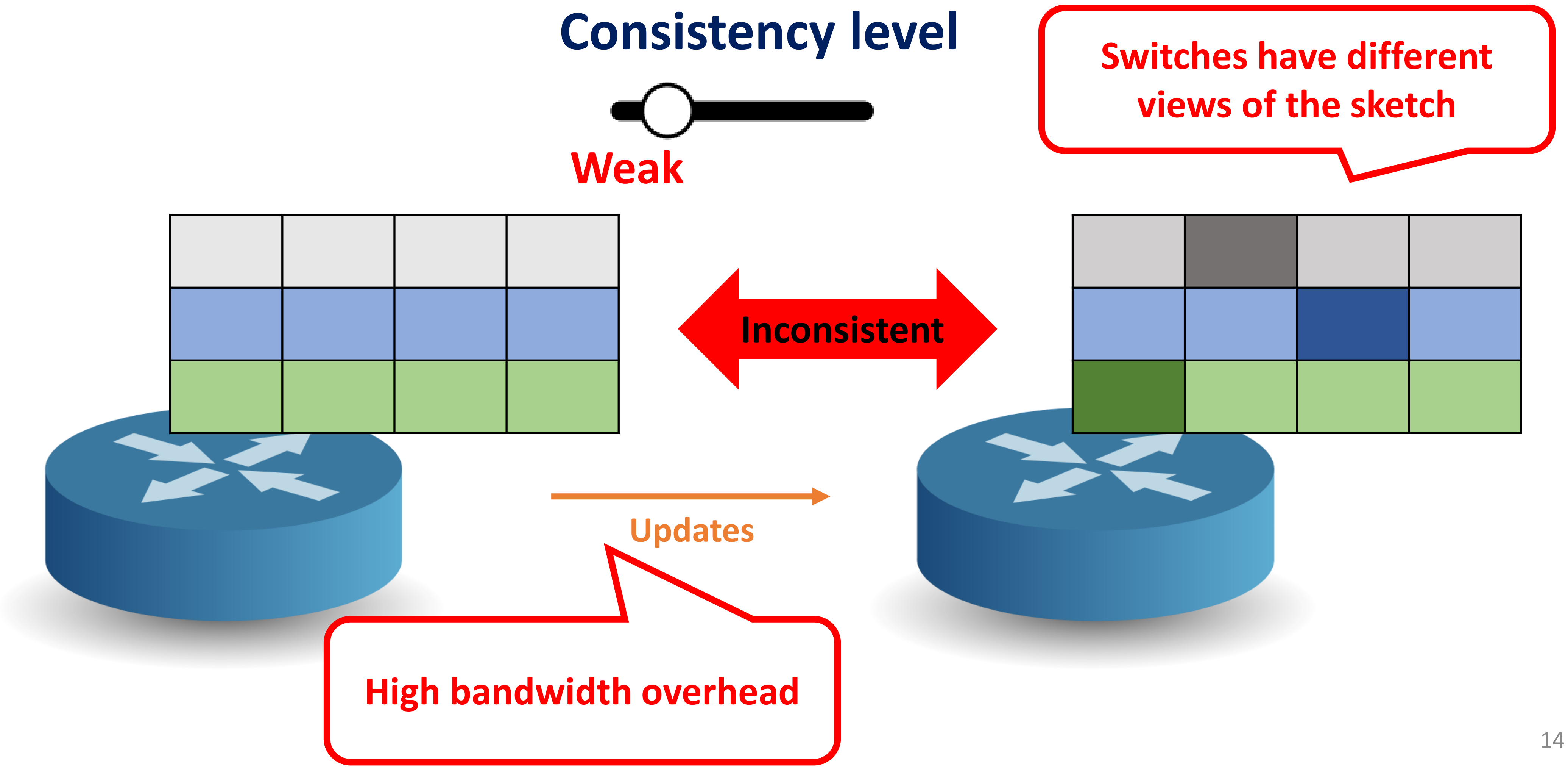
SDW Challenges

- **C1:** What is the most suitable consistency level for replicating sketches?
- **C2:** How to deal with packet drops?

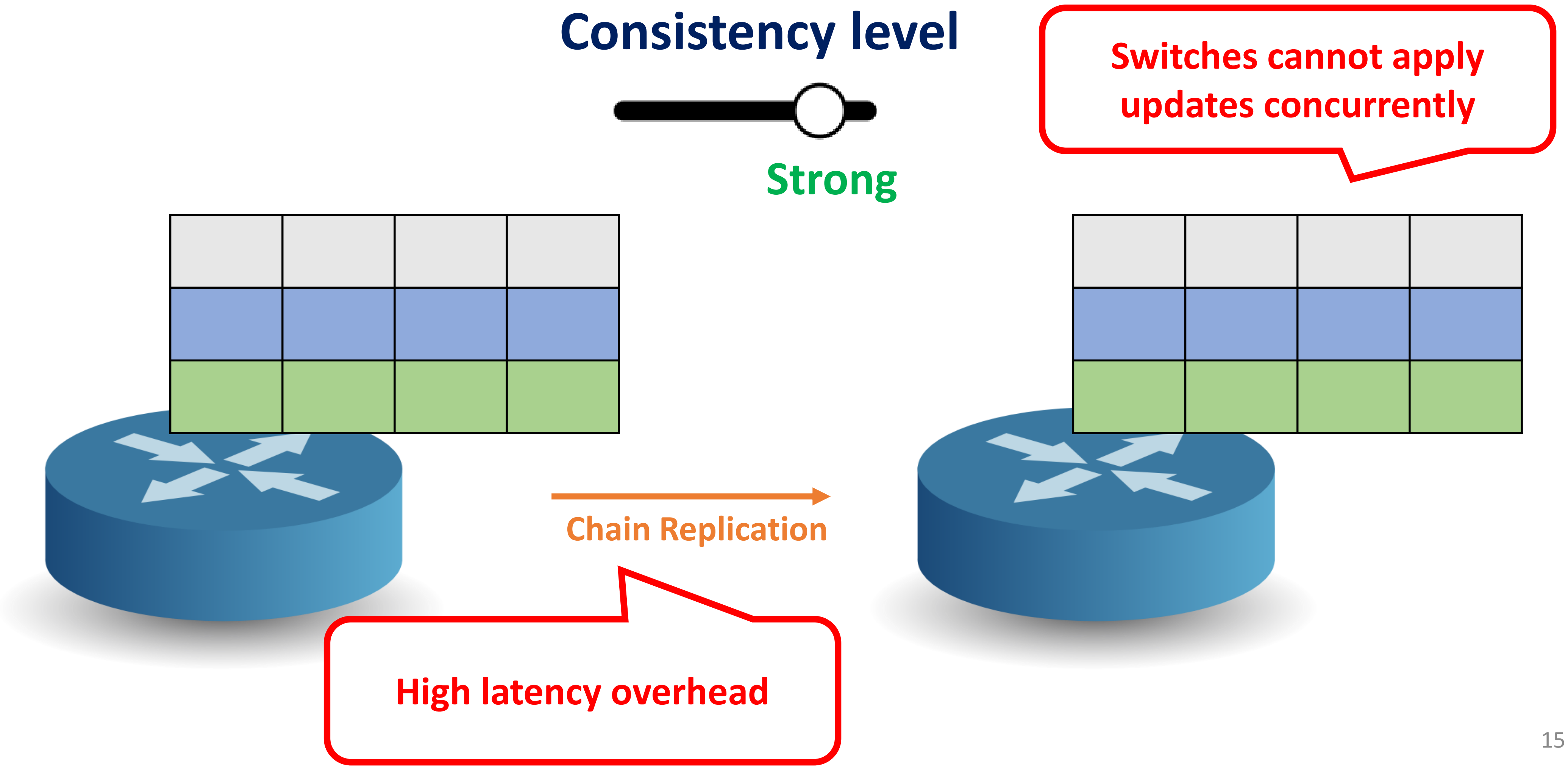
C1: Consistency vs. Performance



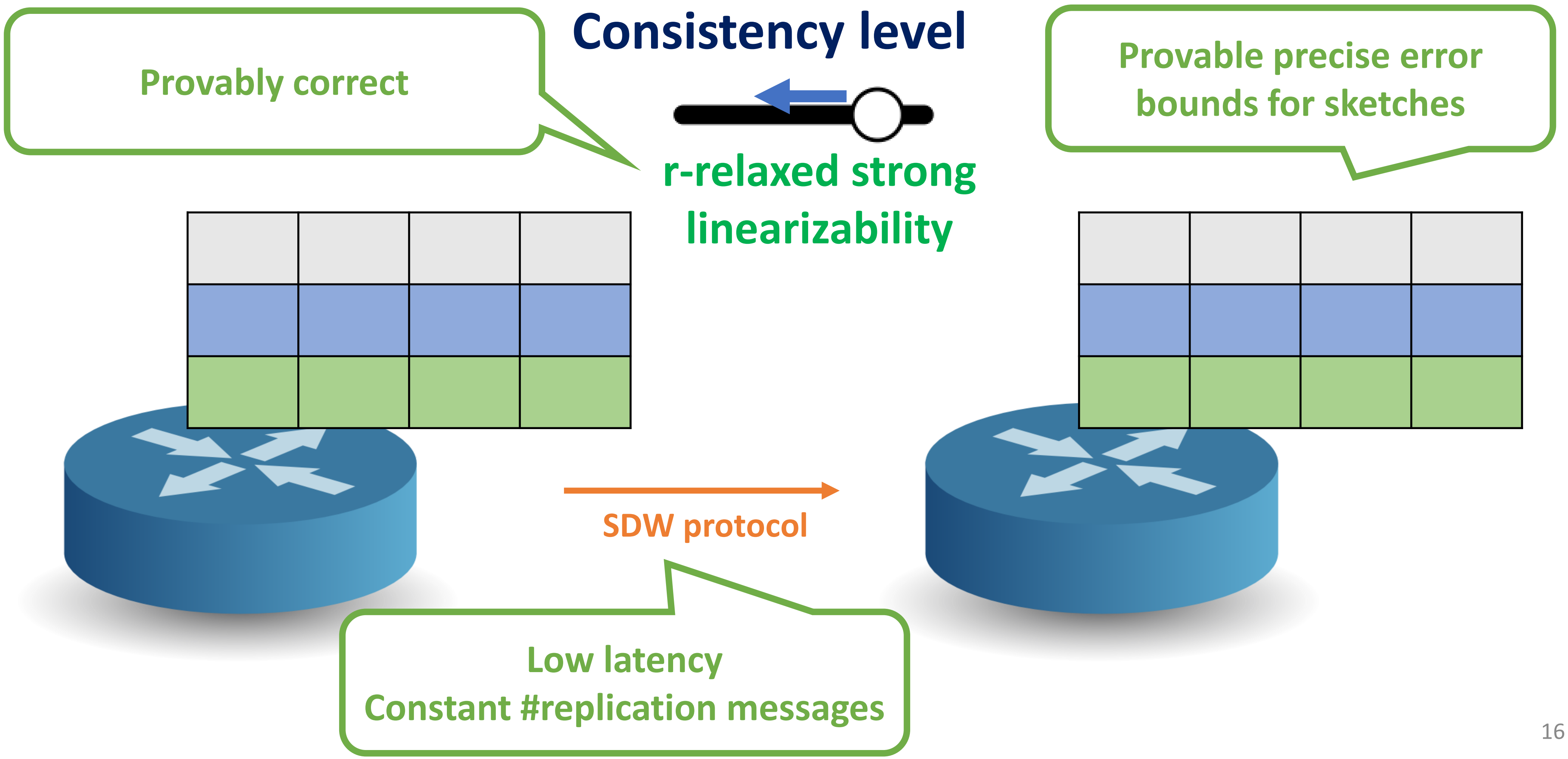
C1: Consistency vs. Performance



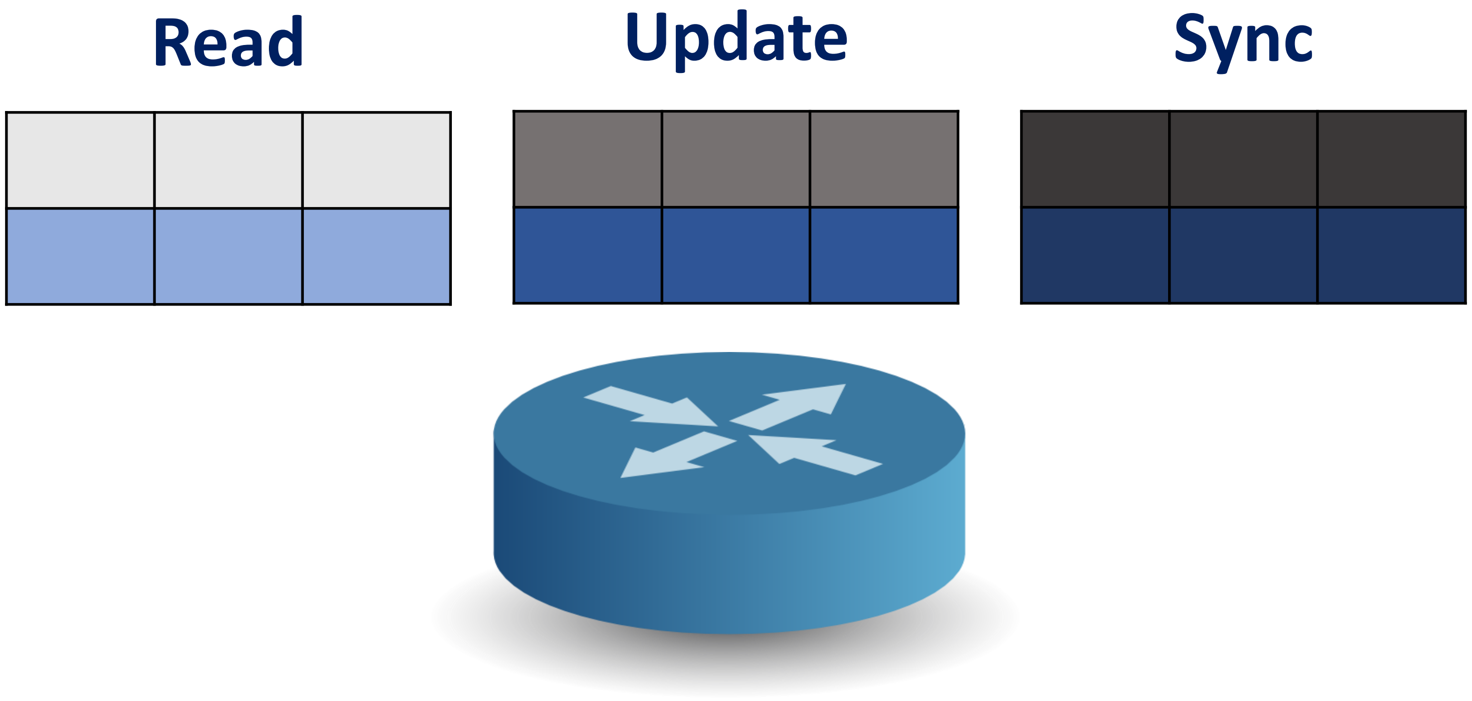
C1: Consistency vs. Performance



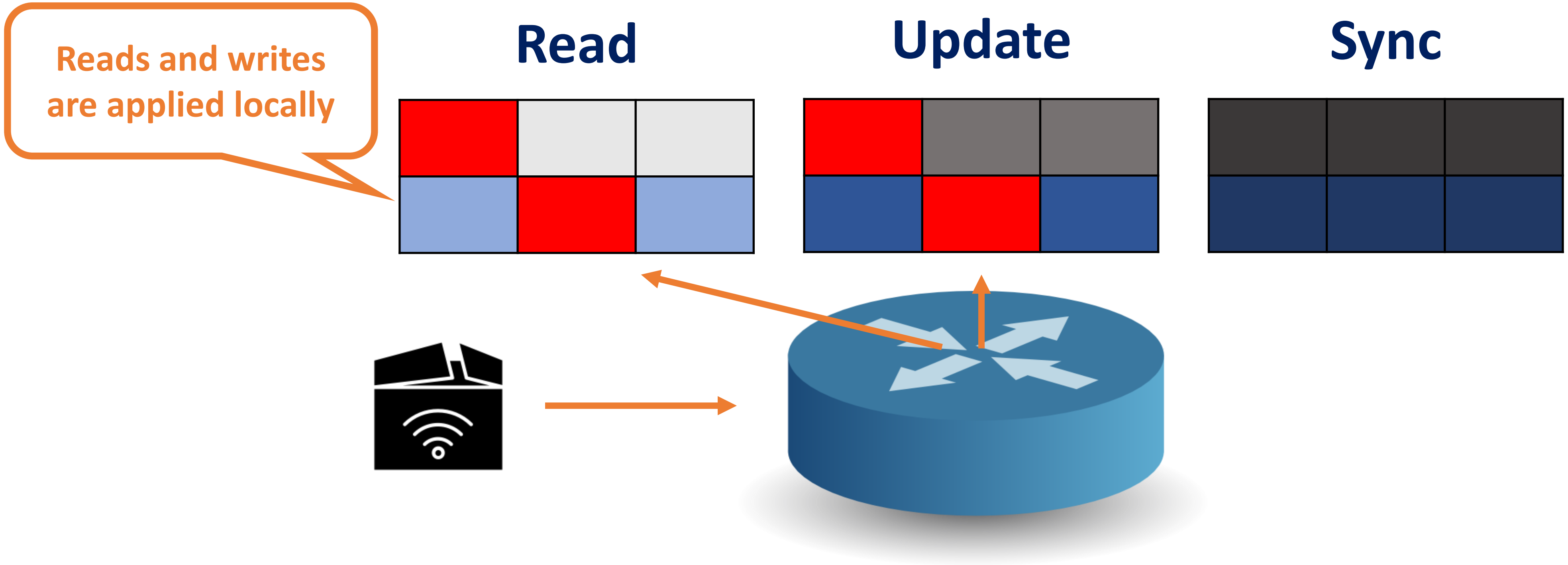
Solution: Strong Delayed-Writes (SDW)



SDW Protocol

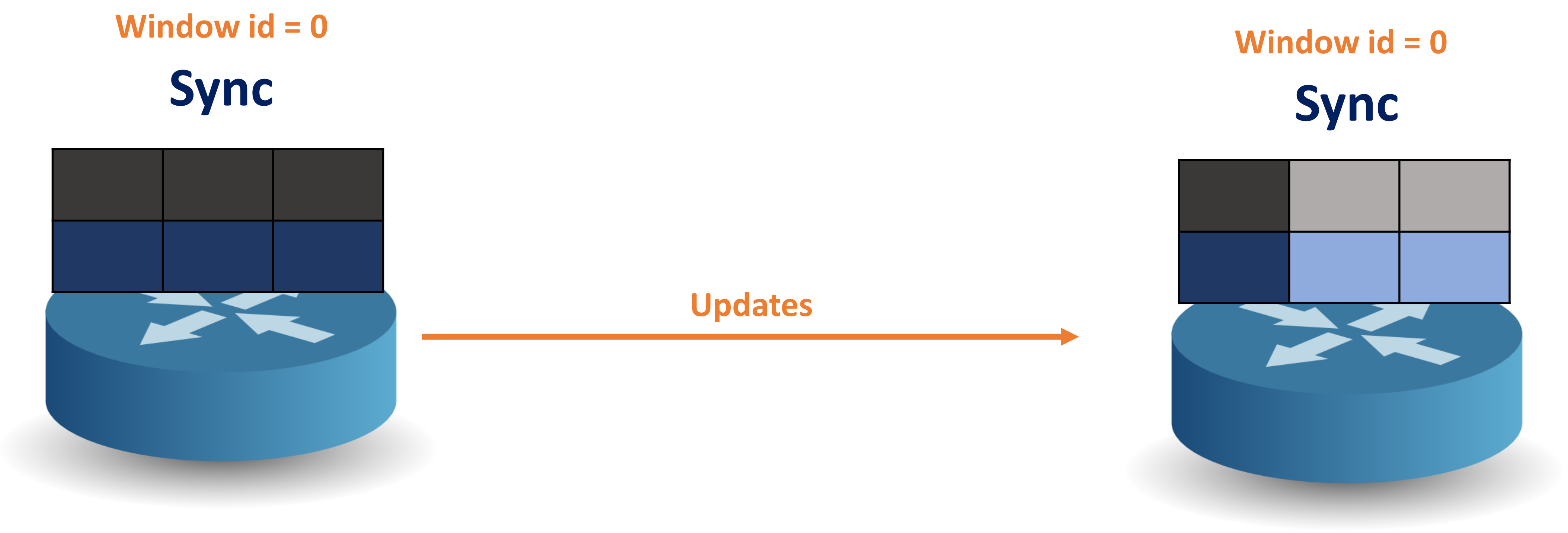


SDW Protocol

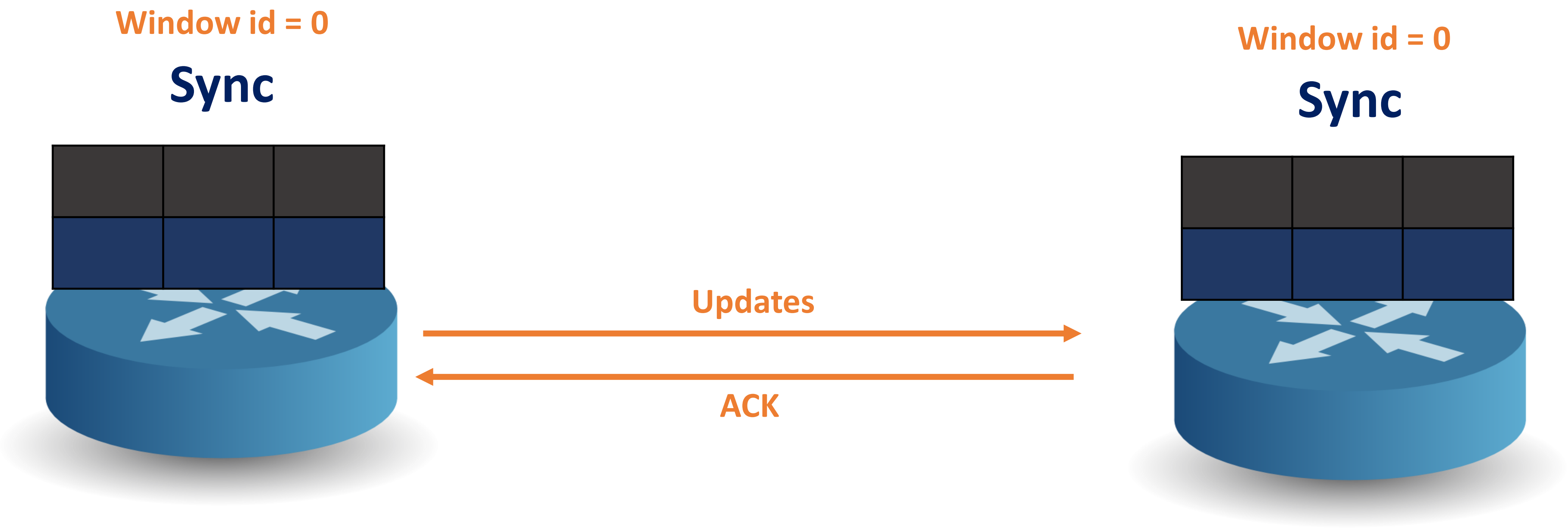


SDW Protocol

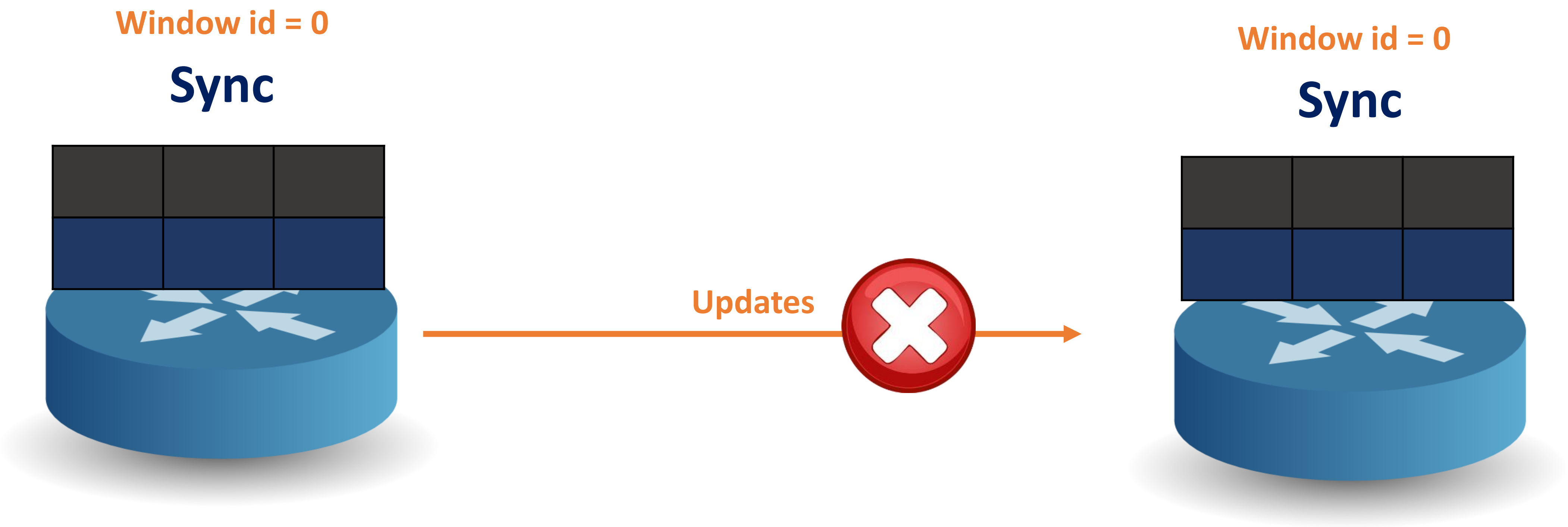
SDW Protocol



SDW Protocol

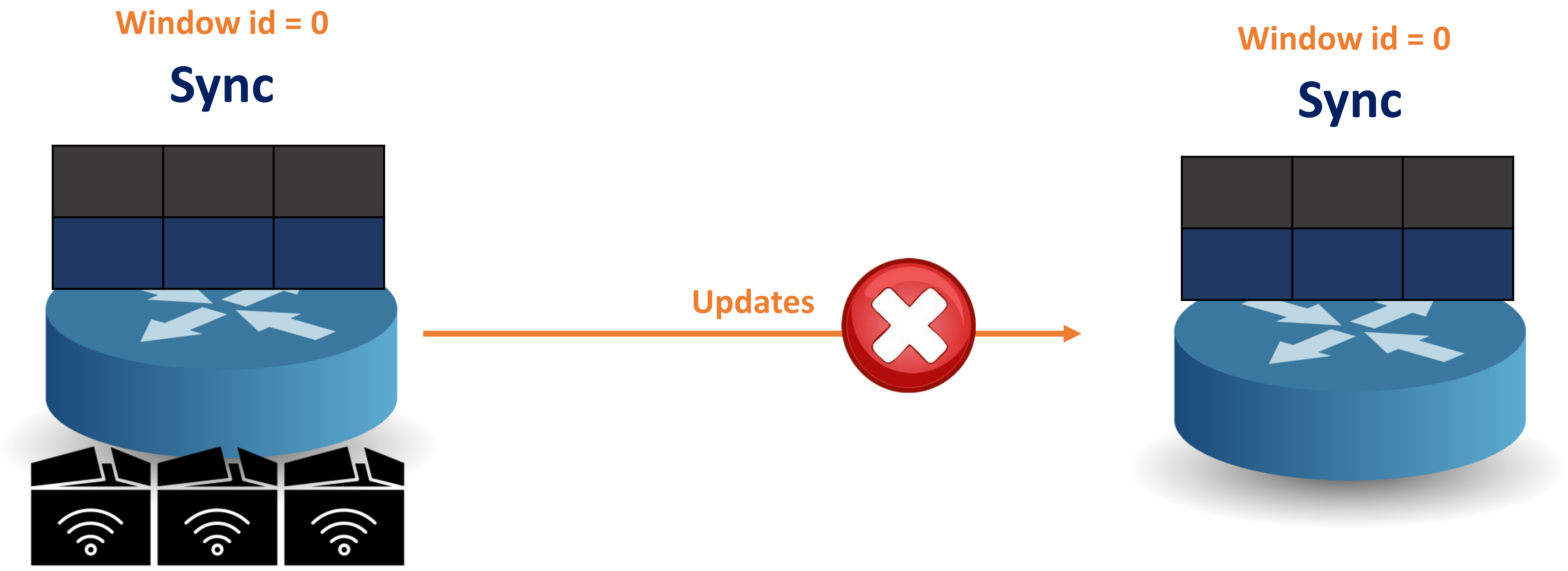


C2: Dealing with Packet Drops

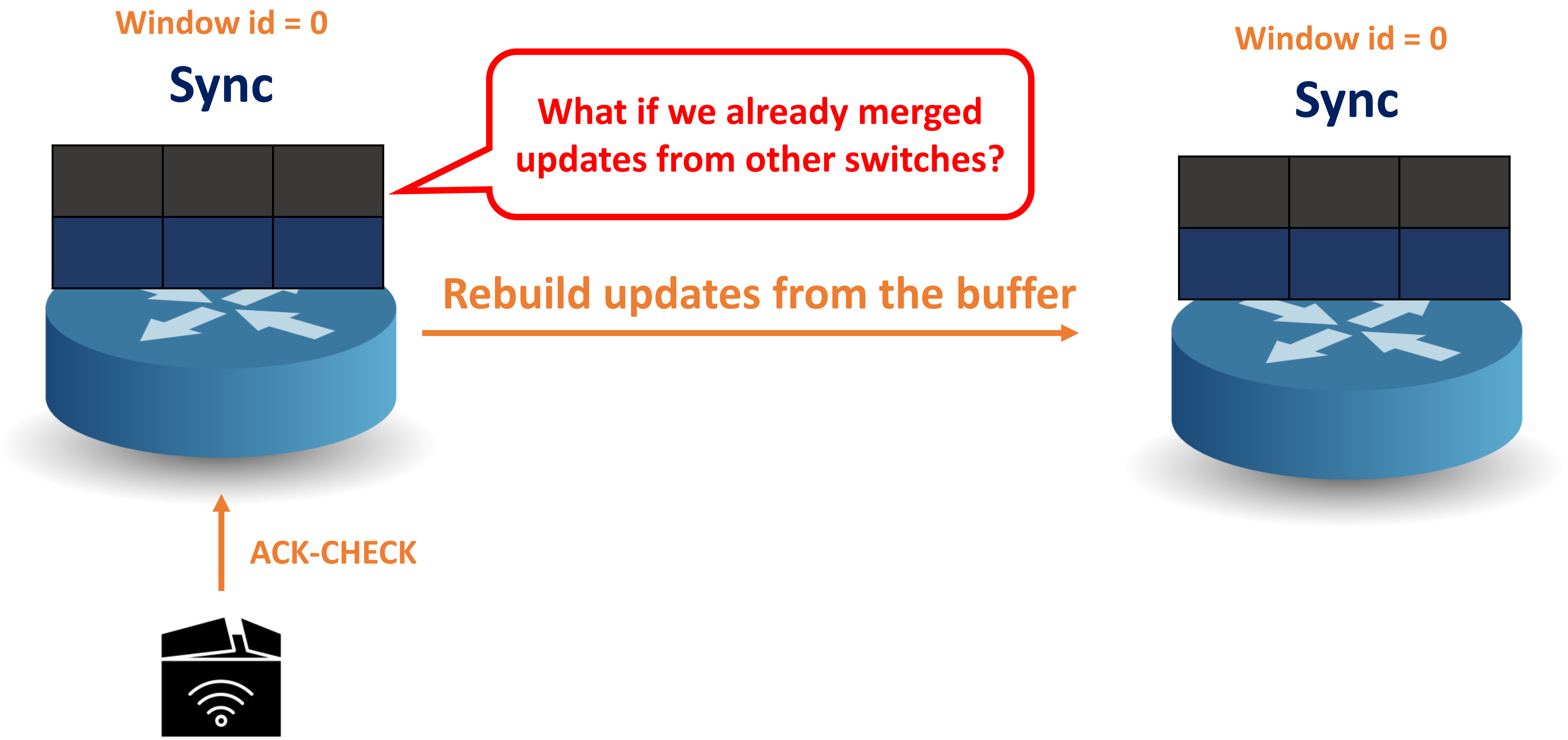


Common solution: implementing reliable delivery over an unreliable network

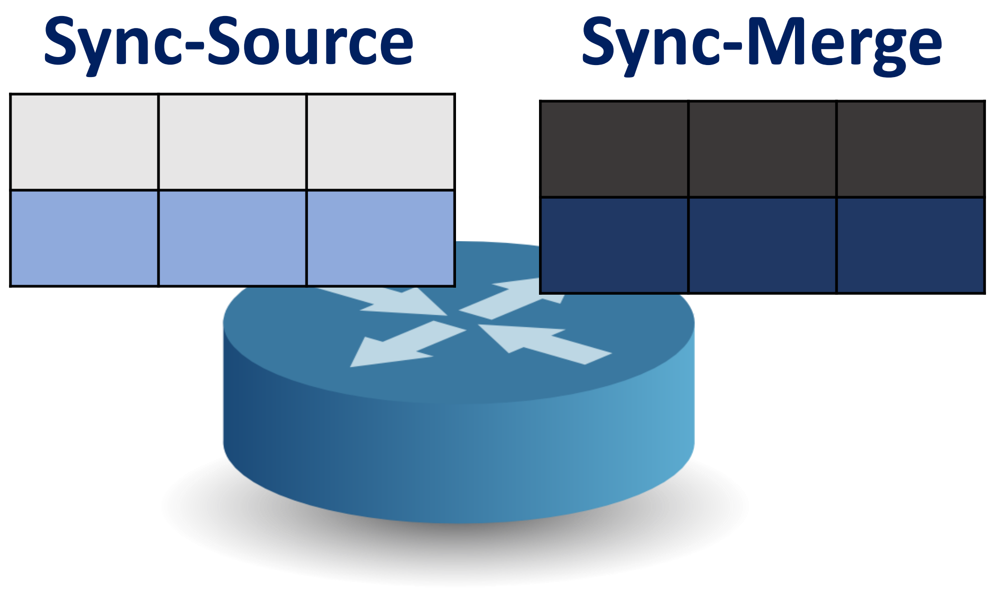
C2: Packet Buffering is Expensive



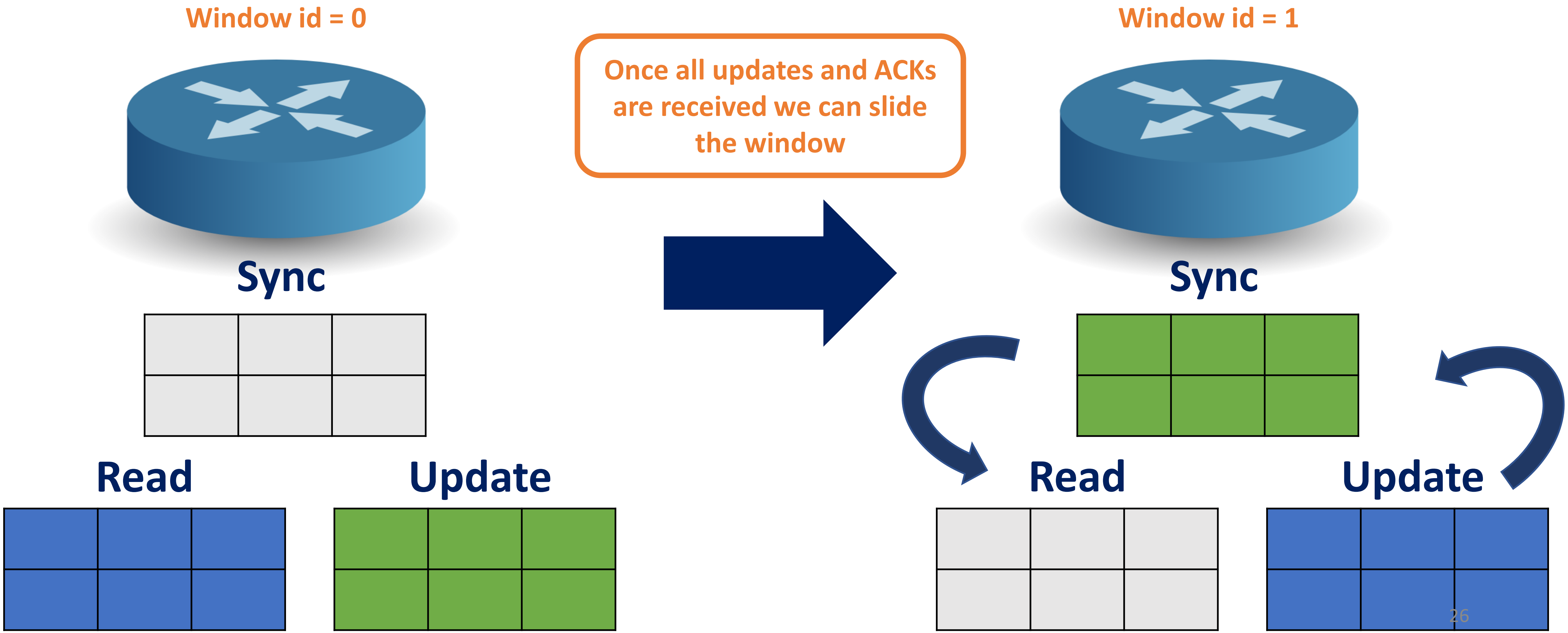
Solution: Reproducible Updates



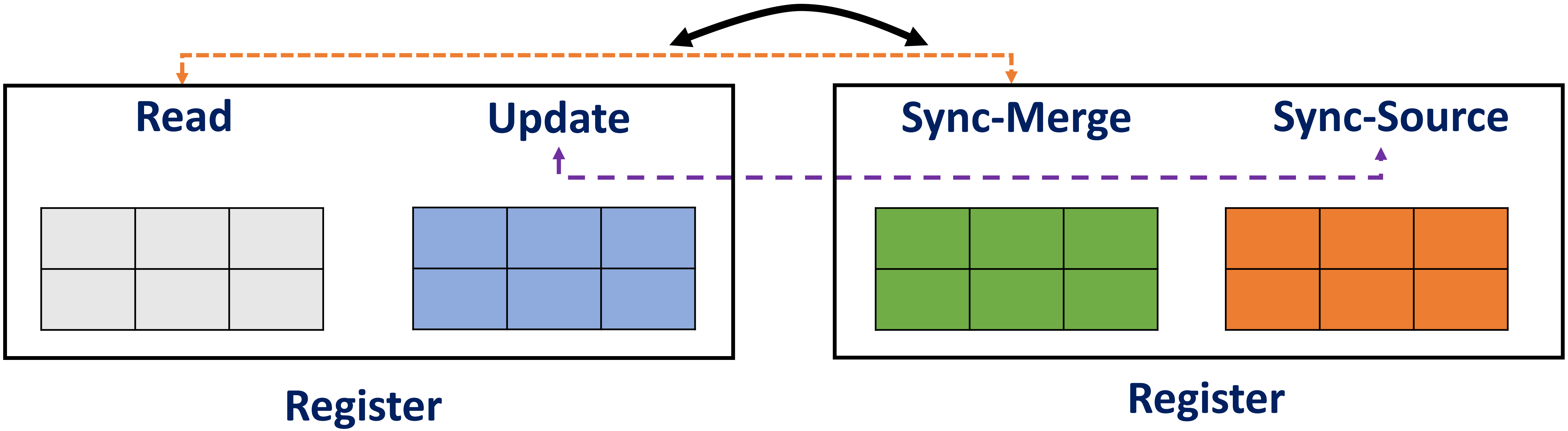
Solution: Reproducible Updates



SDW Protocol



Efficient Register Swapping



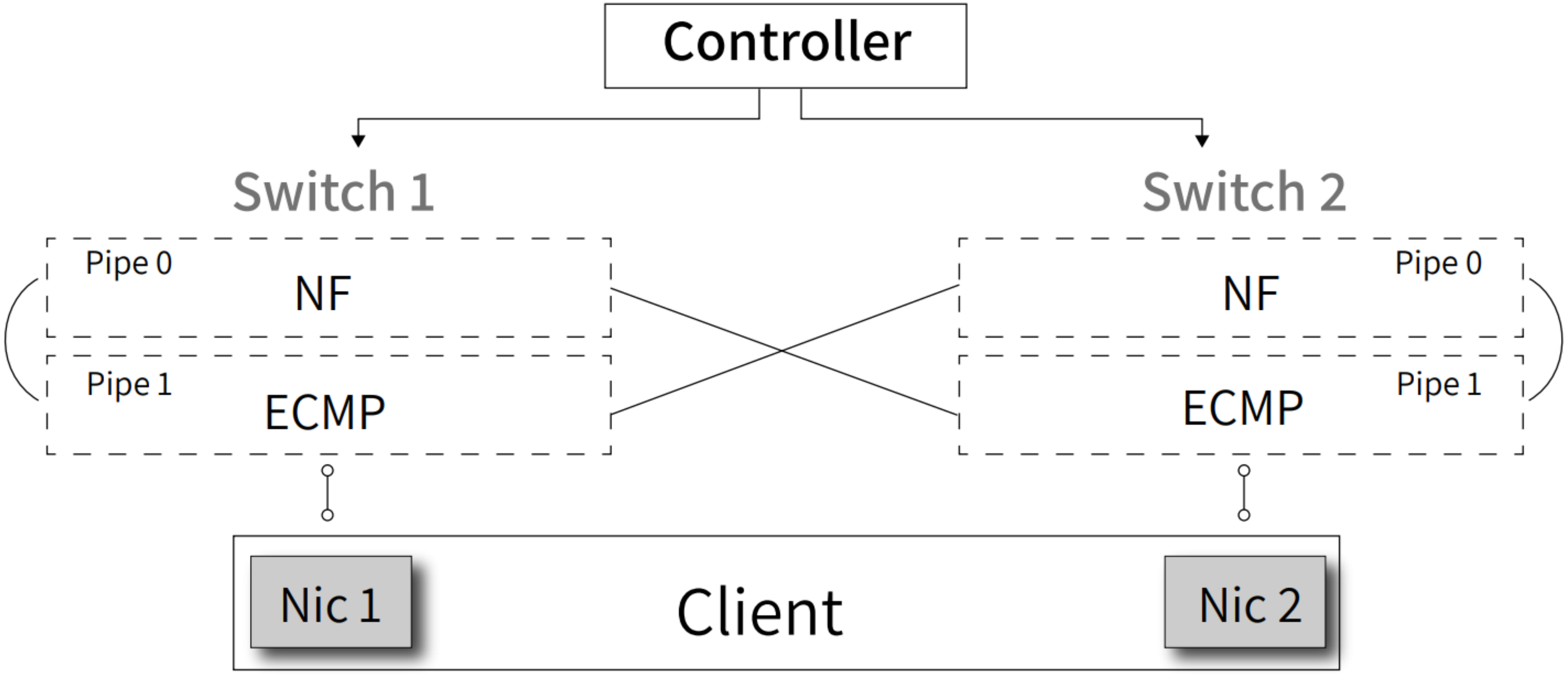
In the paper…

- Theoretical proof of SDW consistency guarantees
- Recovery protocols
- Asymmetric topologies
  - Ready phase
- SDW design
- Eventual Write-Optimized (EWO)
  - Eventual consistency (low read/write latency)
- Strong Read-Optimized (SRO)
  - Strong consistency

Evaluation

- Three real-world application:
  - NAT
  - Rate limiter
  - DDoS Detector
- Microbenchmarks and scalability analysis
- Recovery time

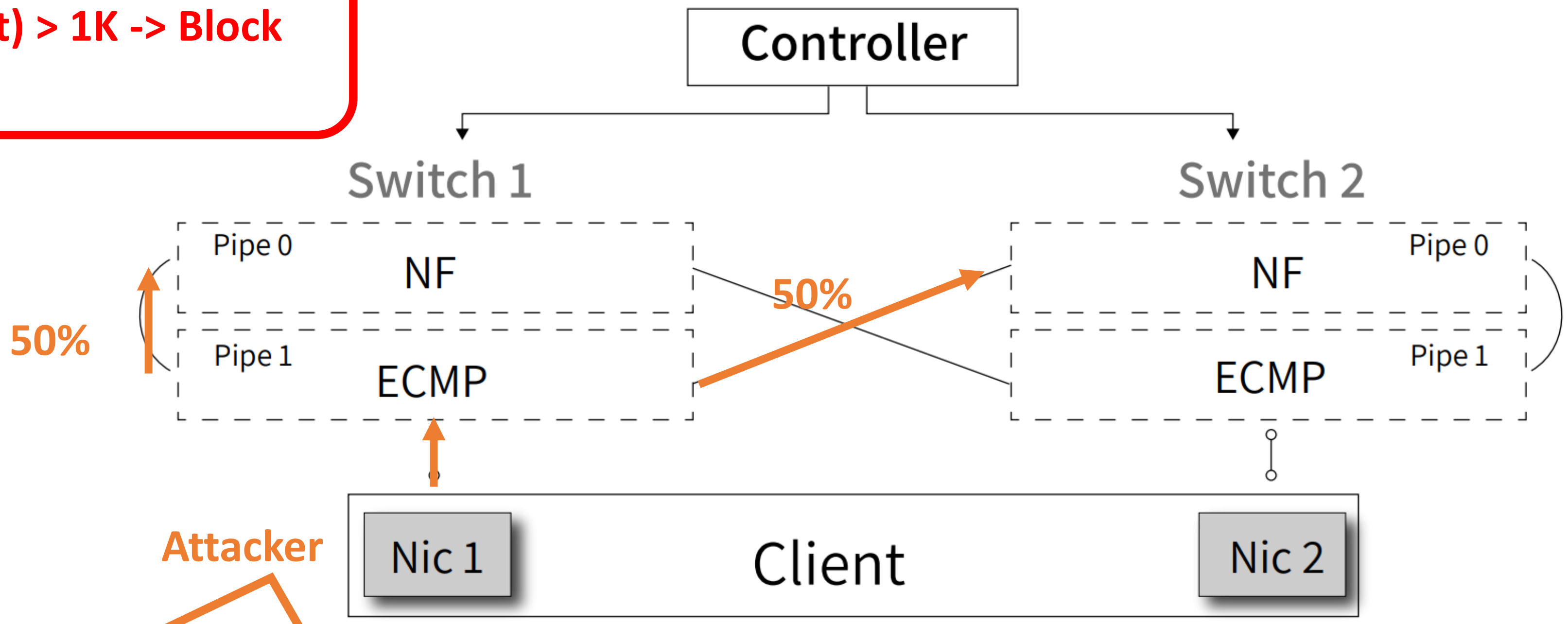
Evaluation



Controller
Switch 1
Switch 2
Pipe 0
NF
Pipe 1
ECMP
NF
Pipe 0
ECMP
Pipe 1
Nic 1
Client
Nic 2

Super-spreader Detector

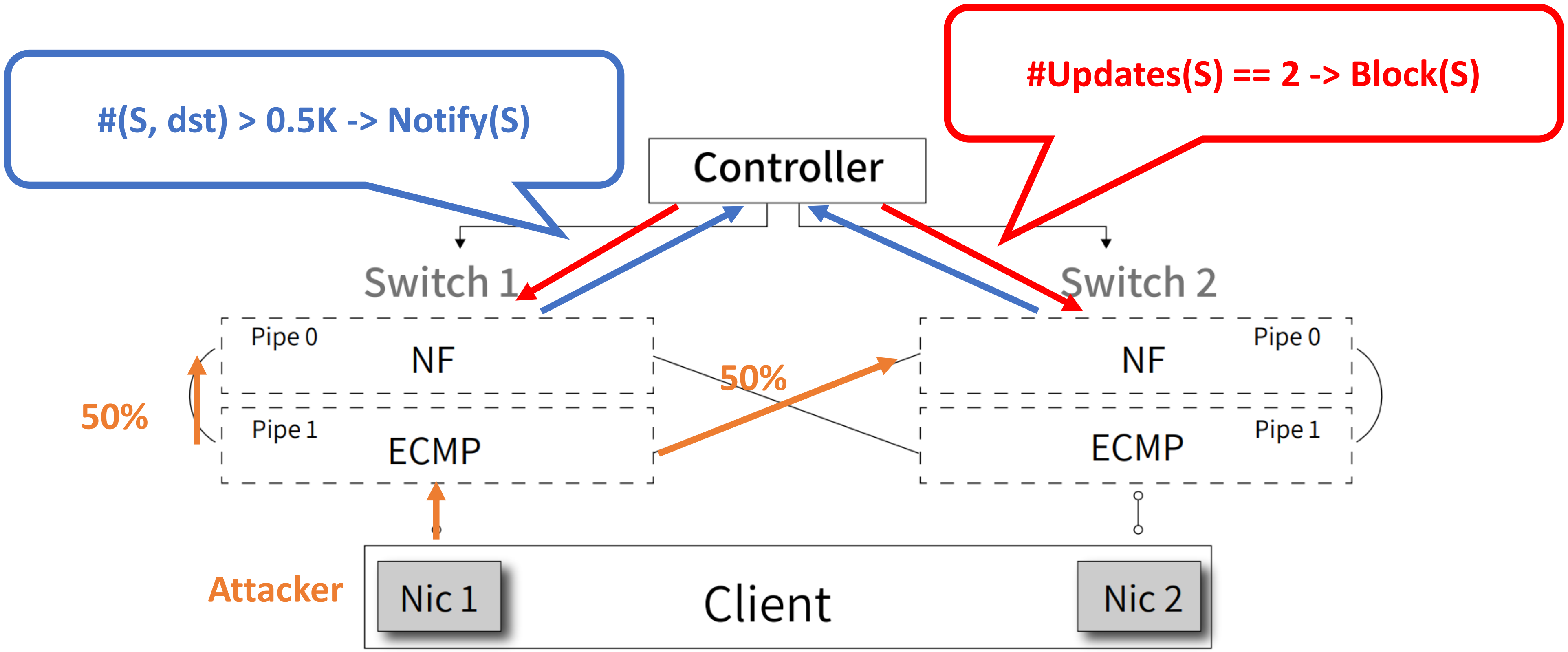
#(S, dst) > 1K -> Block



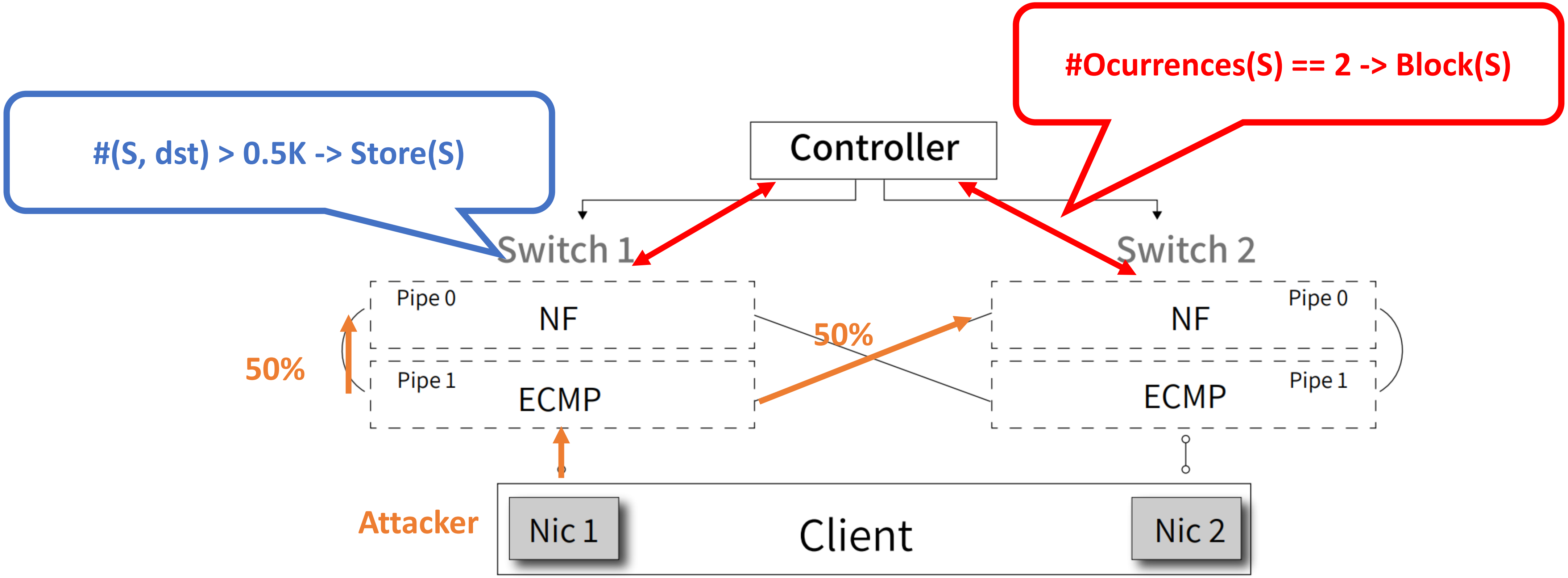
Sends 10K packets with the same source IP to different destinations

We measure how many packets are received

Push Design

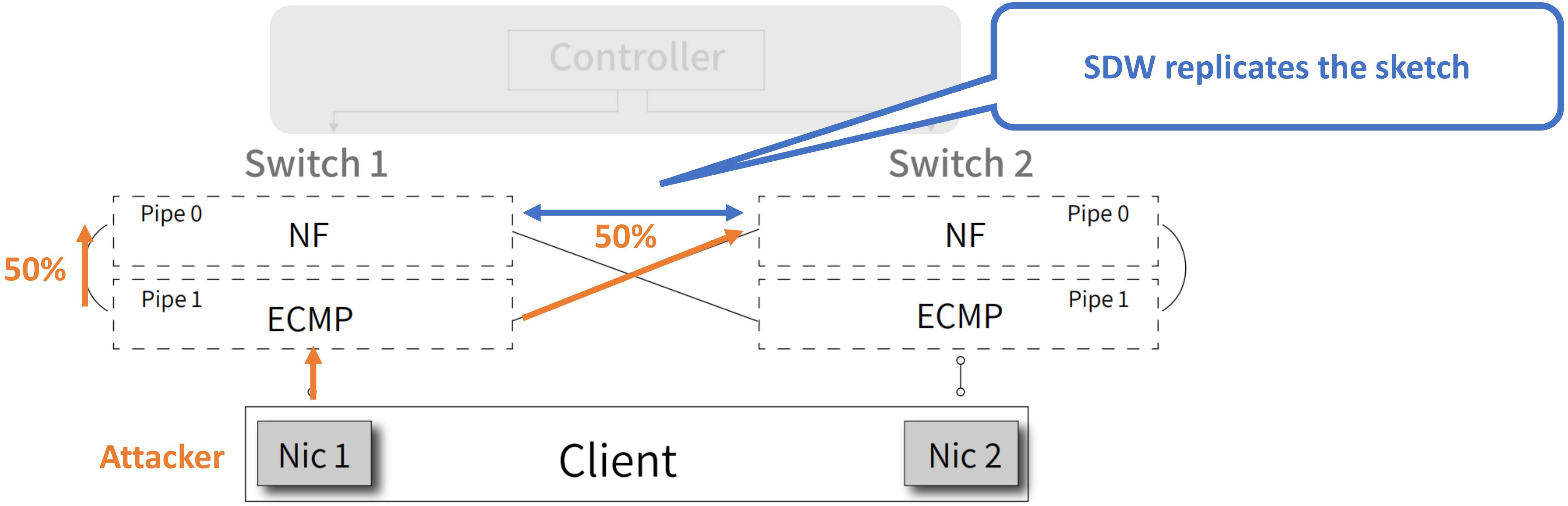

#(S, dst) > 0.5K -> Notify(S)
#Updates(S) == 2 -> Block(S)
Controller
Switch 1
Switch 2
Pipe 0
NF
Pipe 1
ECMP
Pipe 0
NF
Pipe 1
ECMP
50%
50%
Attacker
Nic 1
Client
Nic 2

Pull Design

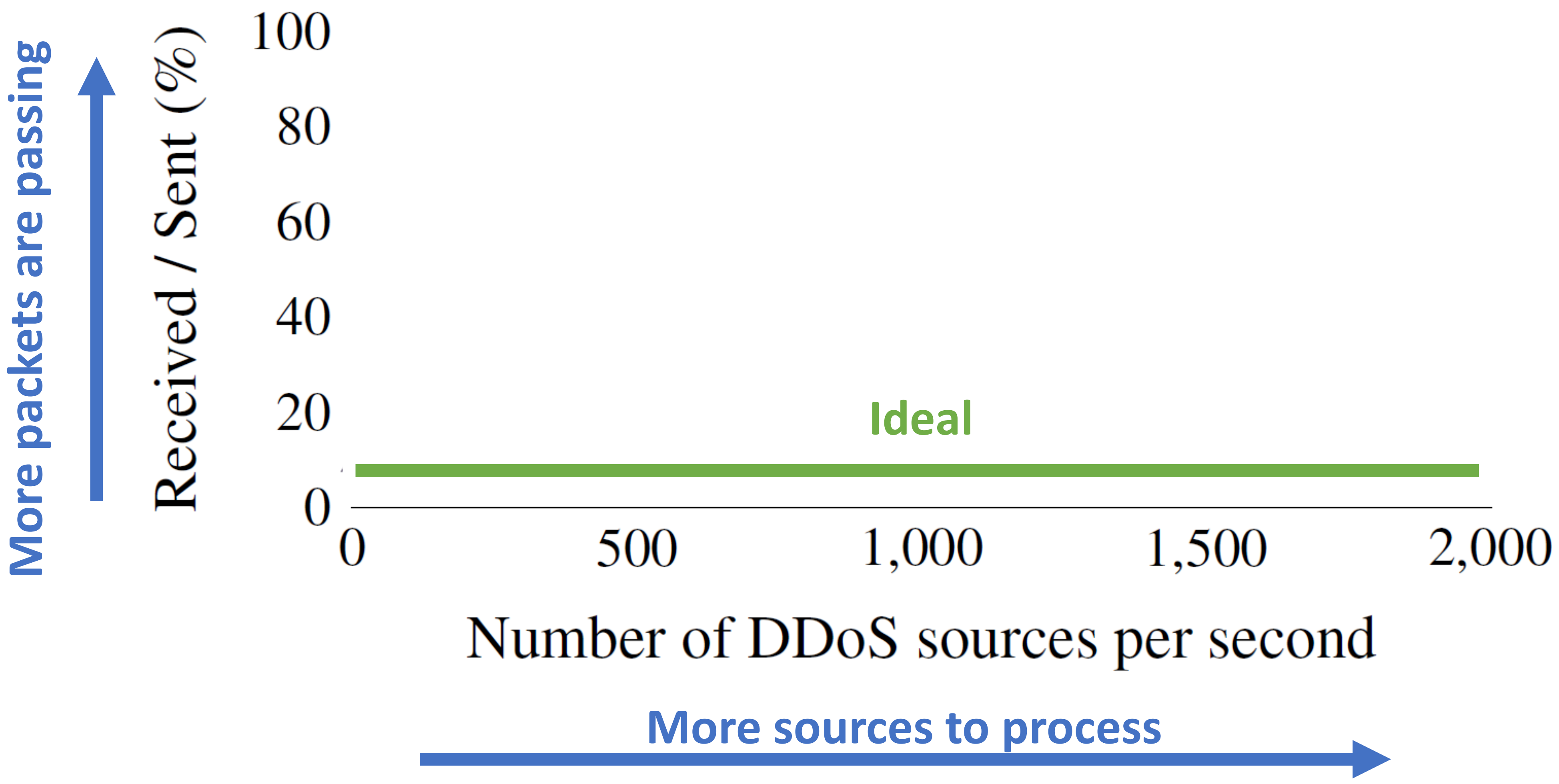


#(S, dst) > 0.5K -> Store(S)
#Ocurrences(S) == 2 -> Block(S)
Controller
Switch 1
Switch 2
Pipe 0
NF
Pipe 1
ECMP
50%
50%
Pipe 0
NF
Pipe 1
ECMP
Attacker
Nic 1
Client
Nic 2

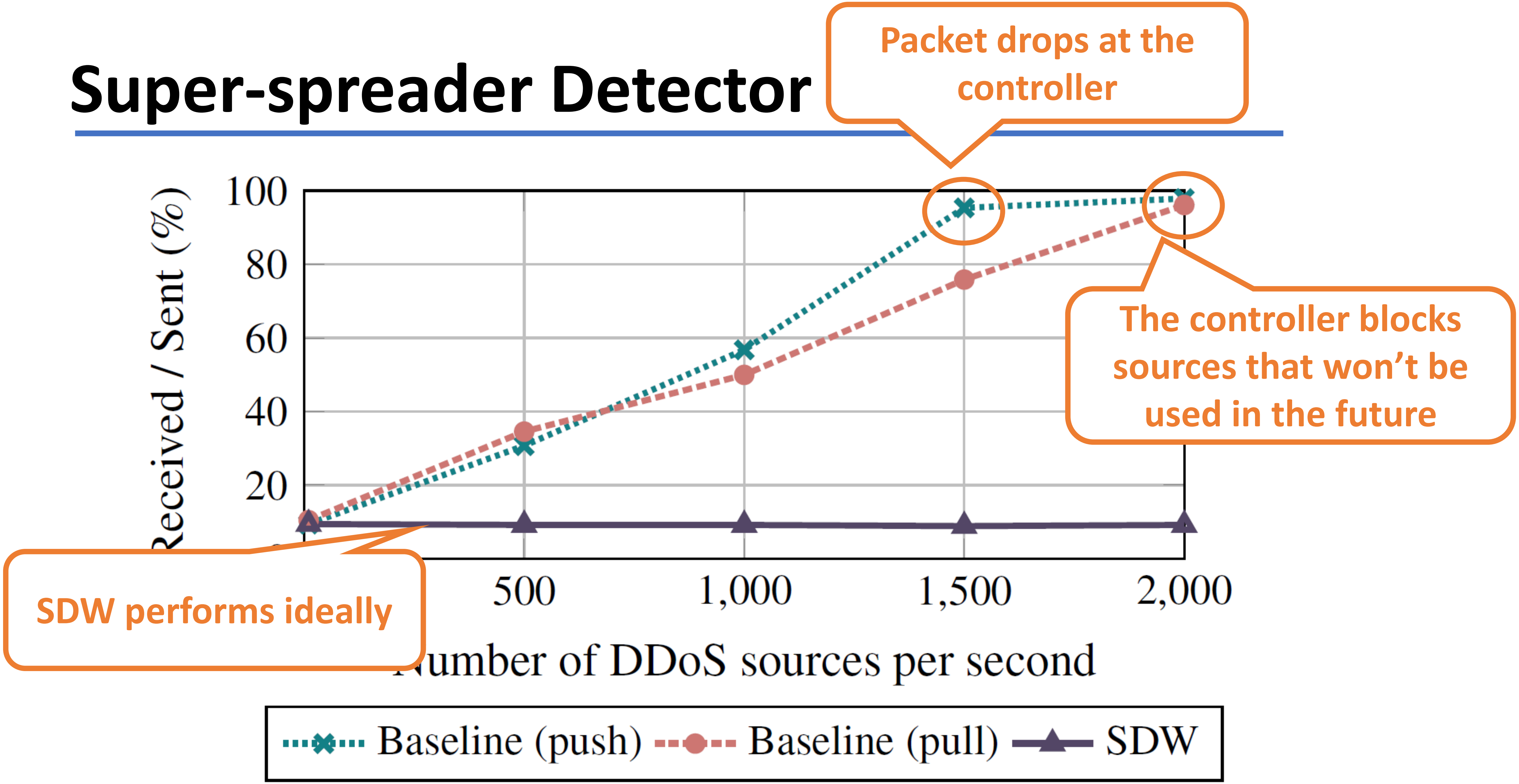
Data Plane-Only Design



Super-spreader Detector: Results



Super-spreader Detector



Conclusions

- Data plane replication is essential for reactive in-switch applications
- SwiSh provide reusable APIs for building distributed in-switch applications
- SwiSh provides a provably correct SDW protocol for sketch replication
- SwiSh is practical, performant and fault tolerant
- **Rethink distributed in-switch applications design**

Thank you!

Questions?

liorz@campus.technion.ac.il