P4 Expert Roundtable Series

April 28-29, 2020

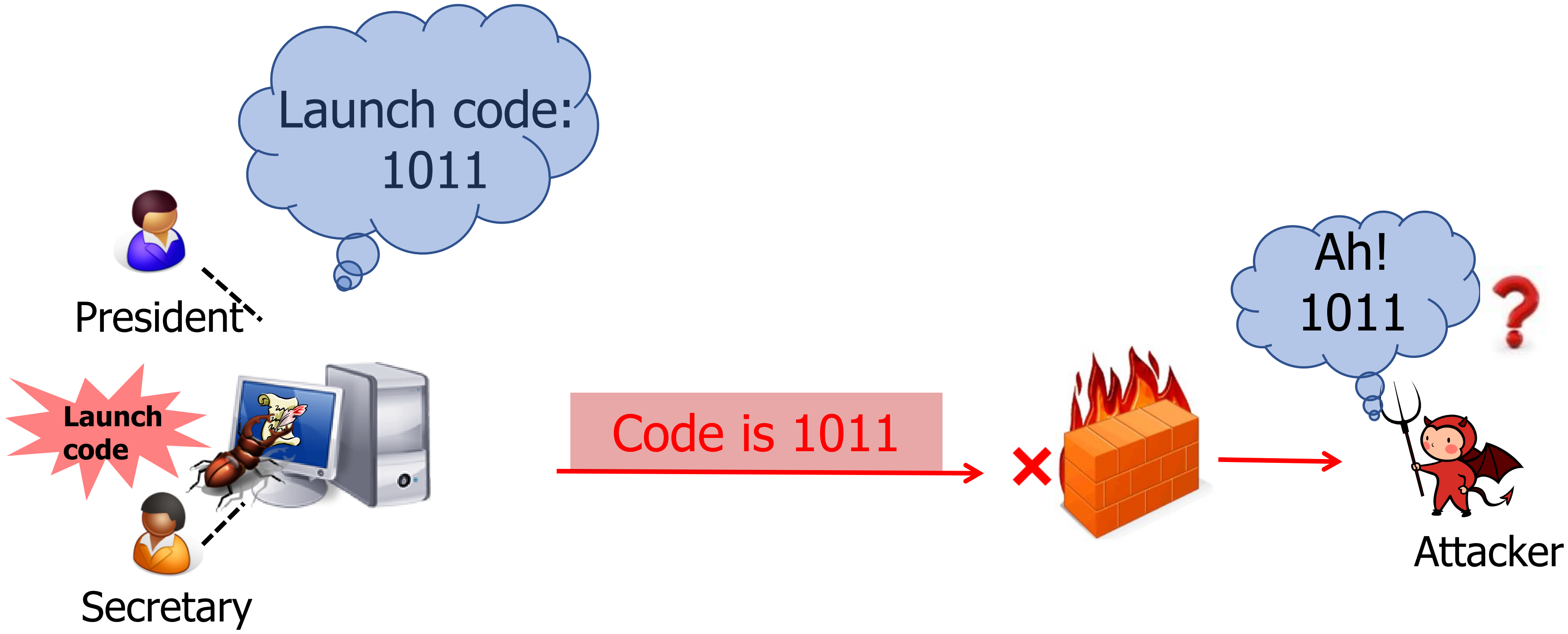
Hosted by: ONF

Sponsored By Google, intel, STORDIS

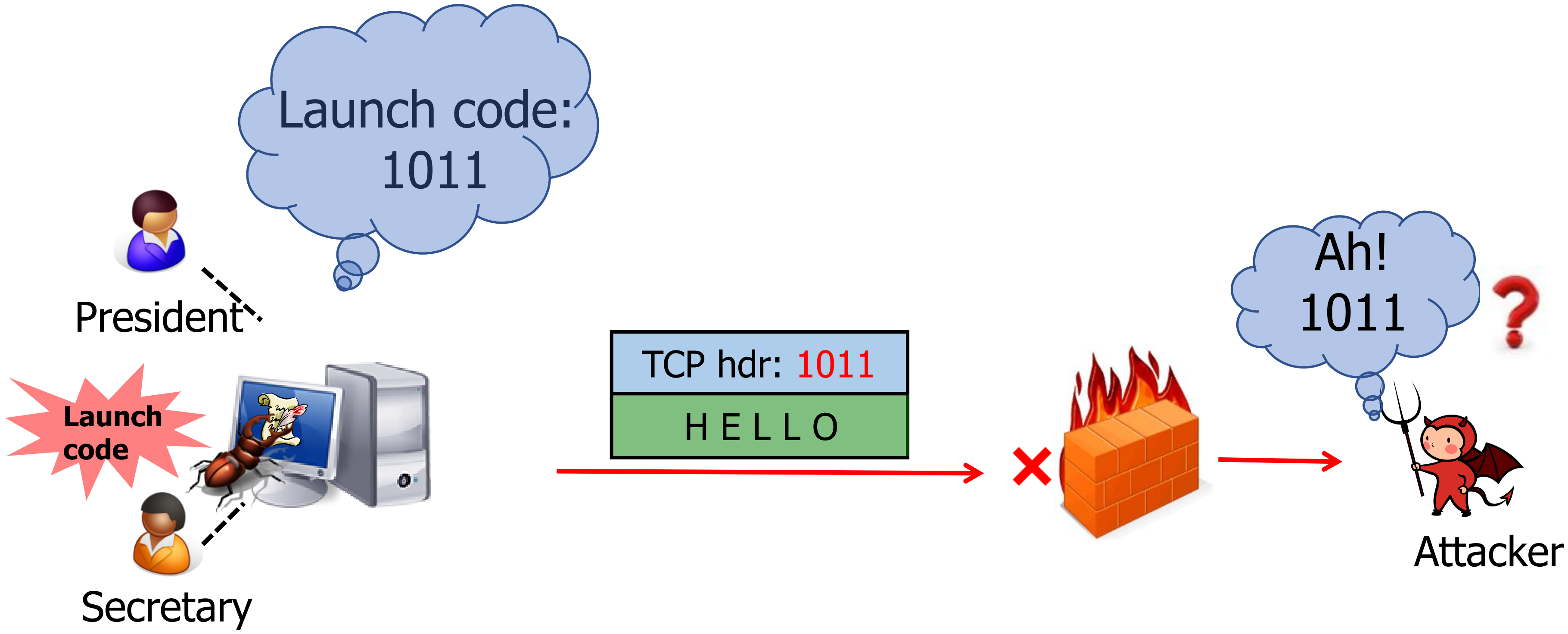
NetWarden: Mitigating Network Covert Channels While Preserving Performance

Jiarong Xing
Ph.D. student
Rice University

Motivation: Mitigating network covert channels

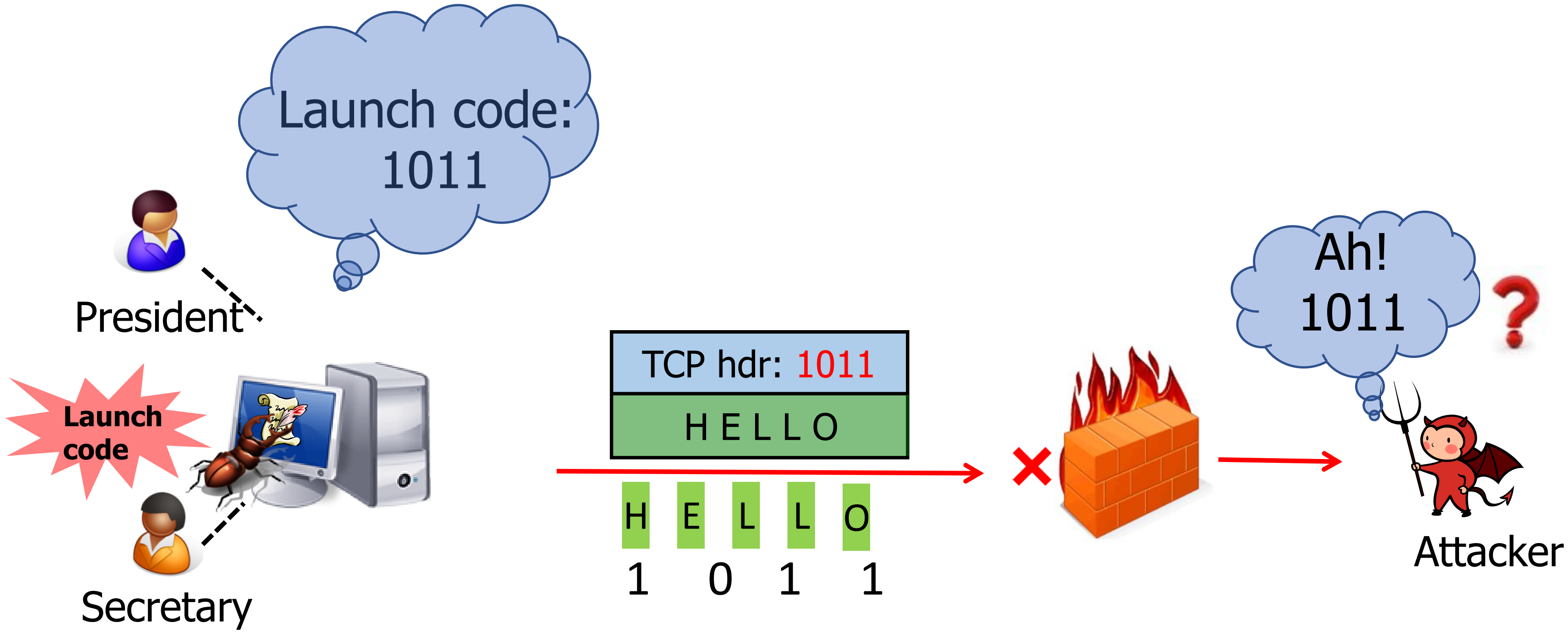


Motivation: Mitigating network covert channels



- Covert channels:
  - Storage channels: changing the packet **header fields.**

Motivation: Mitigating network covert channels



- Covert channels:
  - Storage channels: changing the packet **header fields.**
  - Timing channels: changing the **timing** of packets.

Covert channels are a longstanding problem

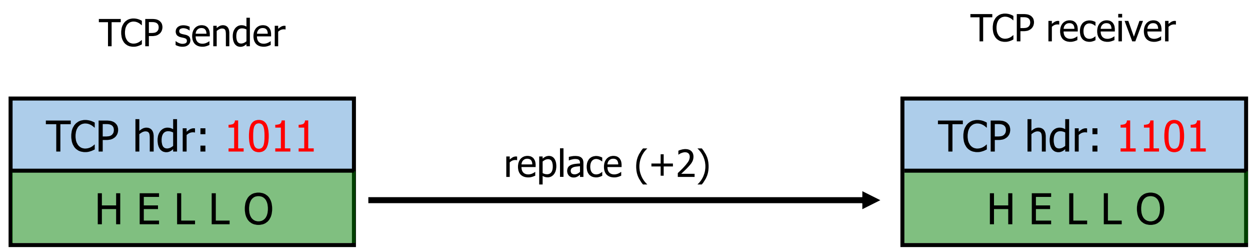
A note on the confinement problem

Author: Butler W. Lampson Authors Info & Affiliations

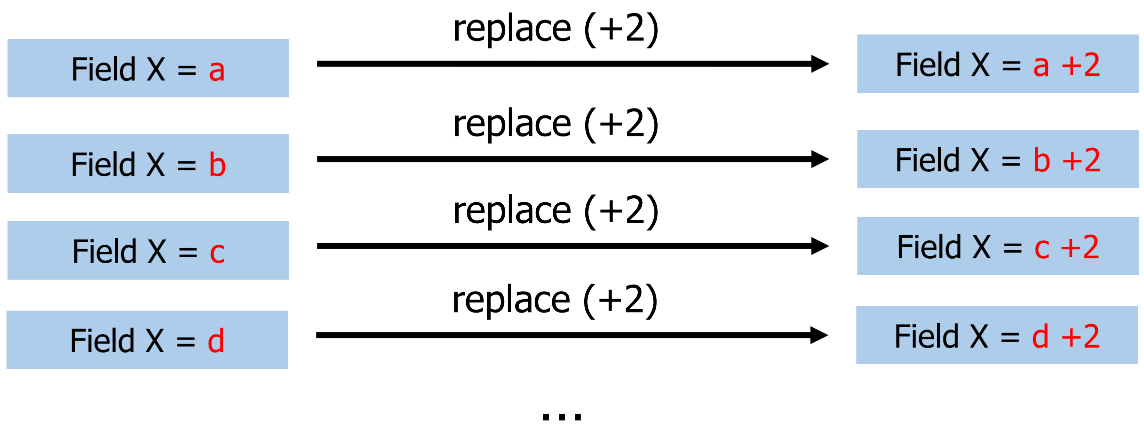
Publication: Communications of the ACM • October 1973 • https://doi.org/10.

- They can leak data over long distance effectively
  - Covert storage channels
    - TCP ISN (1997), TTL (2004), Partial ACK (2009)
  - Covert timing channels
    - IP-layer (2004), TCP-layer (2008), PHY-layer (2014)
- Major security standards require protection against them
  - E.g., Common Criteria

State of the art: Storage channel defense

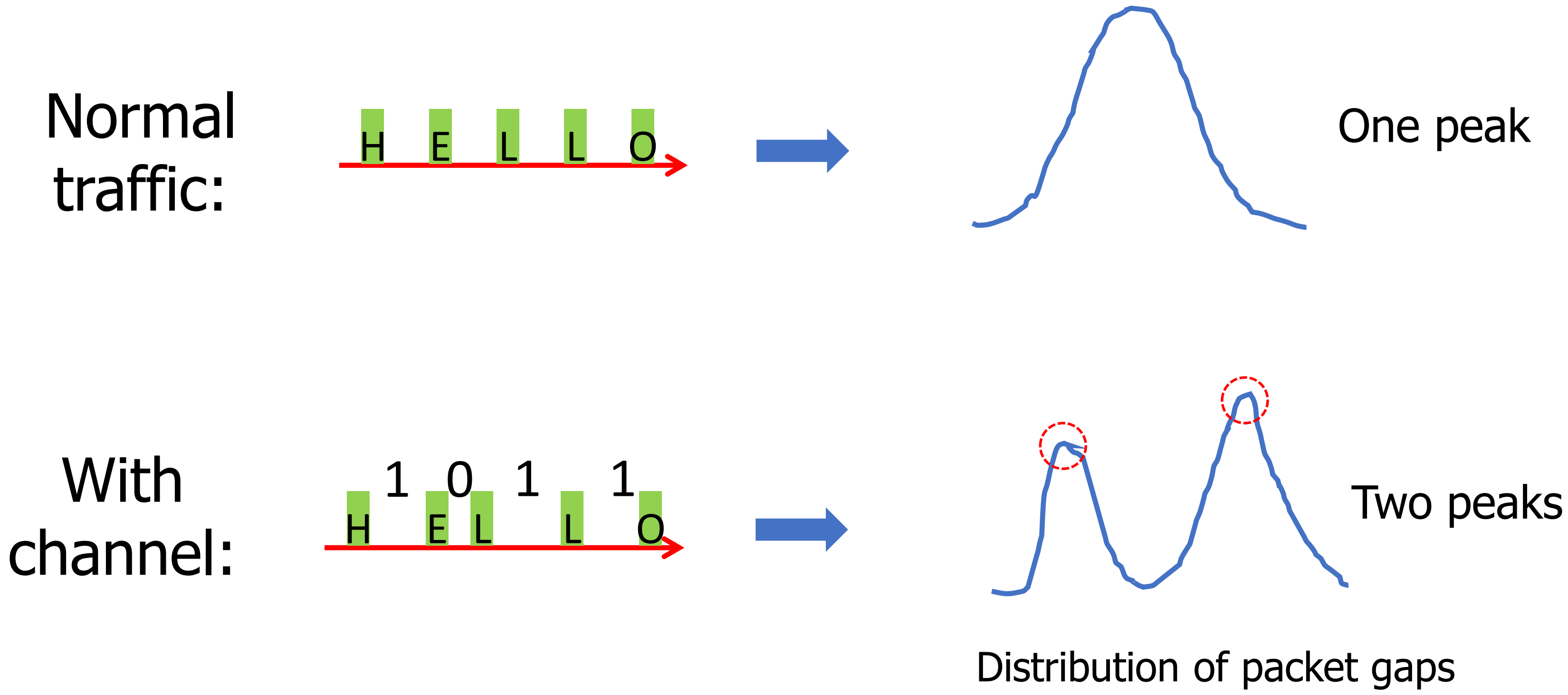


Repeat for **EVERY** packets for **Tbps** traffic



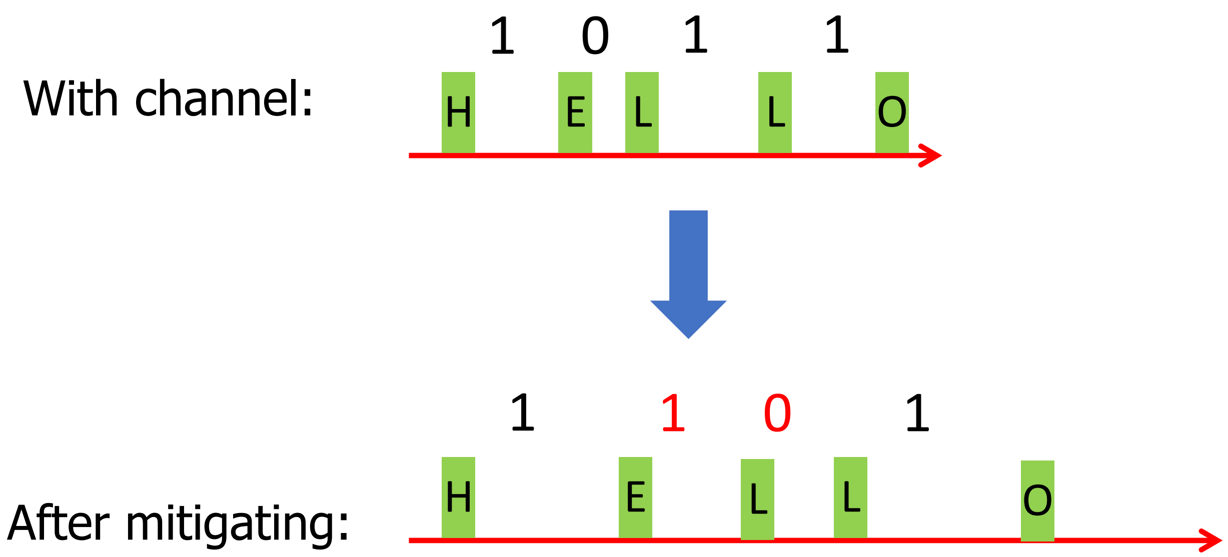
- State-of-the-art solutions are software-based
  - Detection: Per-packet header inspection
  - Mitigation: Per-packet header modification
- As a result, they are very inefficient!

State of the art: Timing channel detection



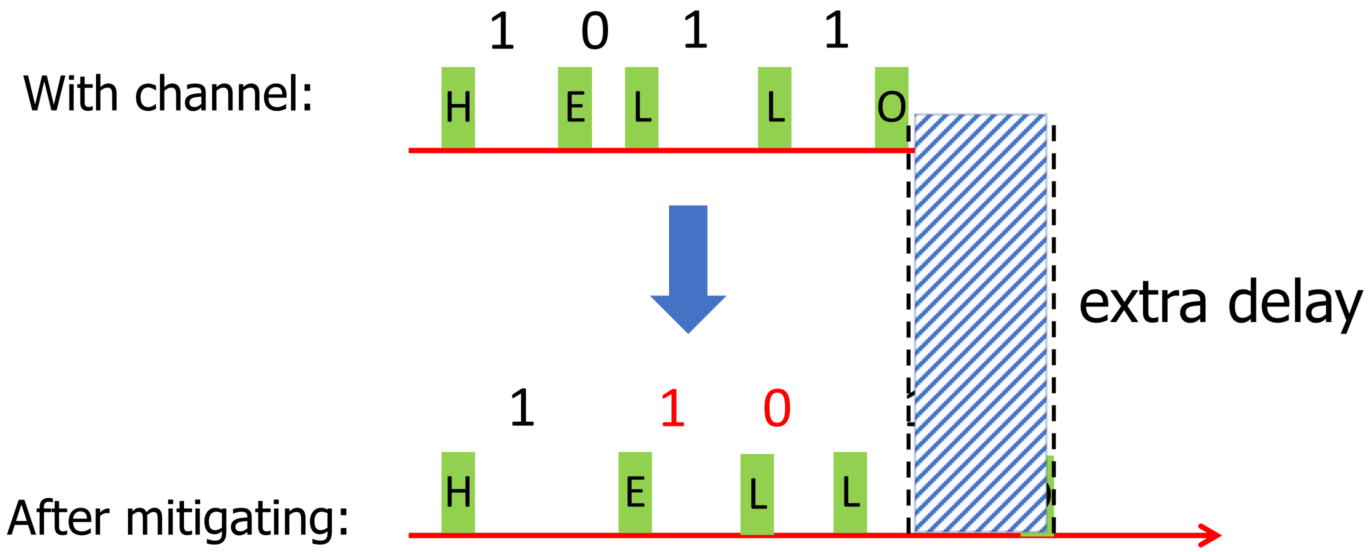
- Detection: Statistics-based tests over packet gaps
  - Looking for signs of statistical deviation
  - → Not always accurate

State of the art: Timing channel mitigation



- Mitigation: Add random delay to each packet
  - Destroy the original timing of the packets

State of the art: Timing channel mitigation



- Mitigation: Add random delay to each packet
  - Destroy the original timing of the packets
- It will increase the latency of TCP connections

Problem: Performance penalty

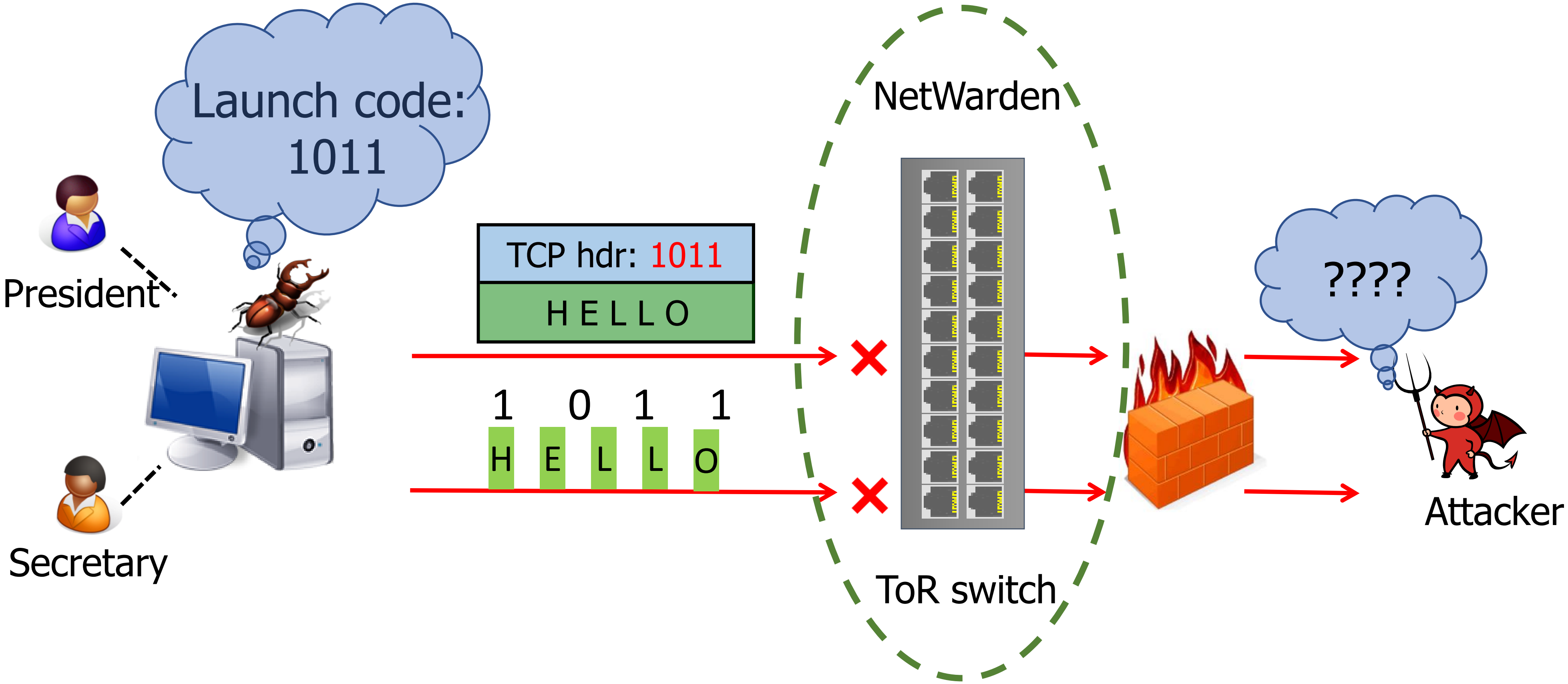
- Detection:
  - Per-packet inspection required
  - Software cannot keep up with Tbps traffic

- Mitigation:
  - Adding random delay to each packet → Increase latency
  - Collateral damage → Affects legitimate traffic (e.g., false positives)

Key question

Can we mitigate covert channels while preserving performance?

Approach: NetWarden



- **NetWarden**: A performance-preserving covert channel defense.

Key challenges and solutions

- **Key principle:** Hardware/software co-design

- **Challenge #1:** Efficient detection
  - **Solution:** Using **P4** switches

- **Challenge #2:** Performance-preserving mitigation
  - **Solution:** Performance “boosting”

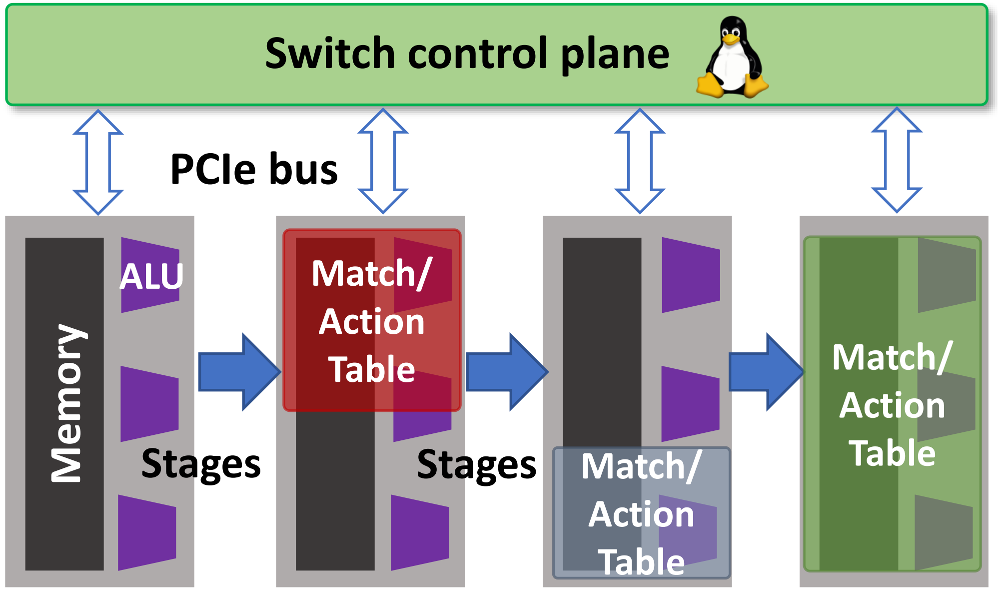
Outline

- Motivation: Mitigating network covert channels
- State of the art: Performance penalty
- Approach: NetWarden
- NetWarden design
  - Principles of hardware/software co-design
  - Challenge #1: Efficient detection
  - Challenge #2: Performance-preserving mitigation
- Evaluation
- Conclusion

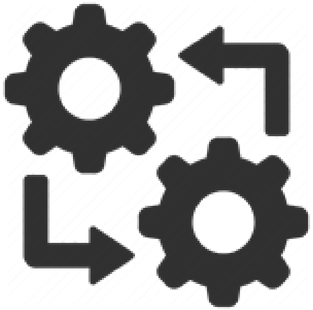
Key principle: Hardware/software co-design

- A **generic** principle that is applicable to many P4 applications.

P4 switch anatomy



- Data plane
  - Header modification, ns timestamp, per-flow state, line speed
  - Limited memory, simple math computation
- Control plane
  - Abundant memory, complex math computation
  - Software speed

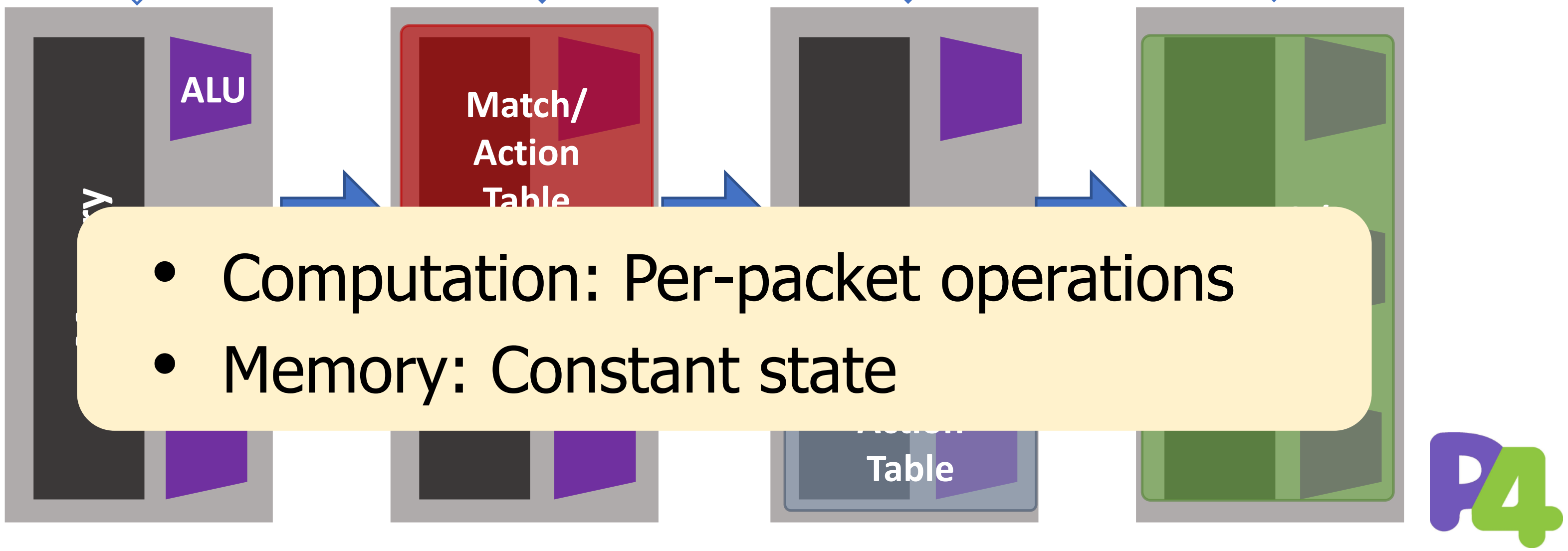


Complementary!

Key principle: Hardware/software co-design

- Computation: Batch operations
- Memory: Growing state

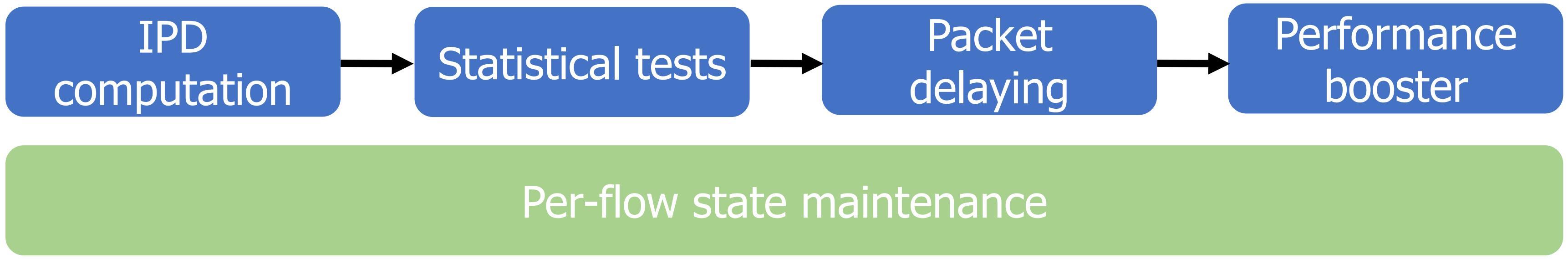
- Minimize crosstalk



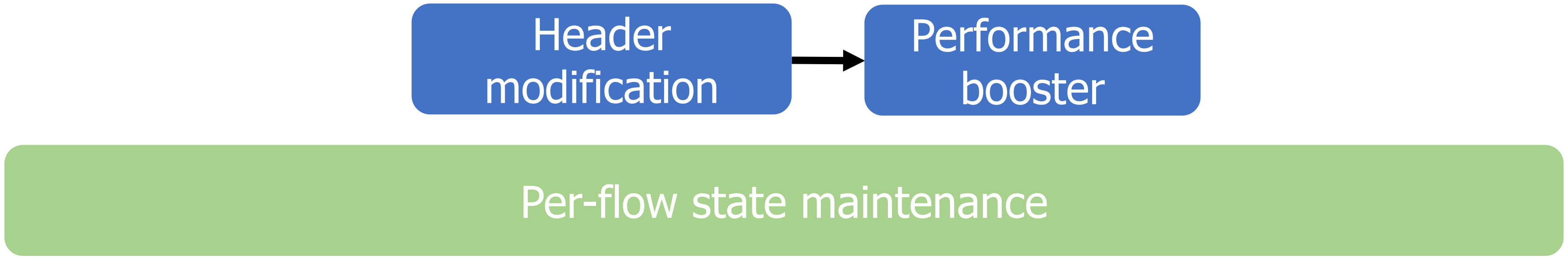
- Computation: Per-packet operations
- Memory: Constant state

Covert channel defense roadmap

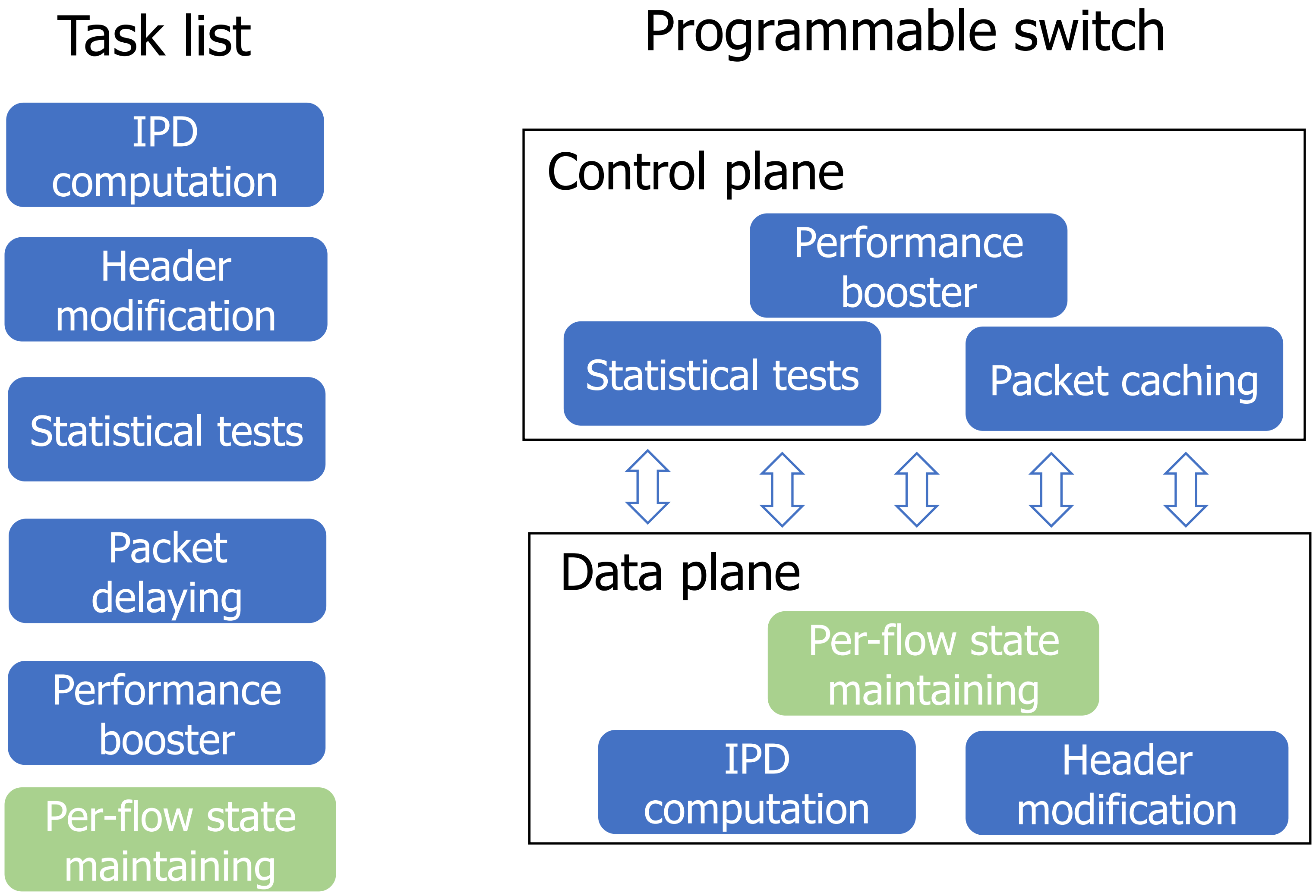
- Time channel defenses



- Storage channel defenses



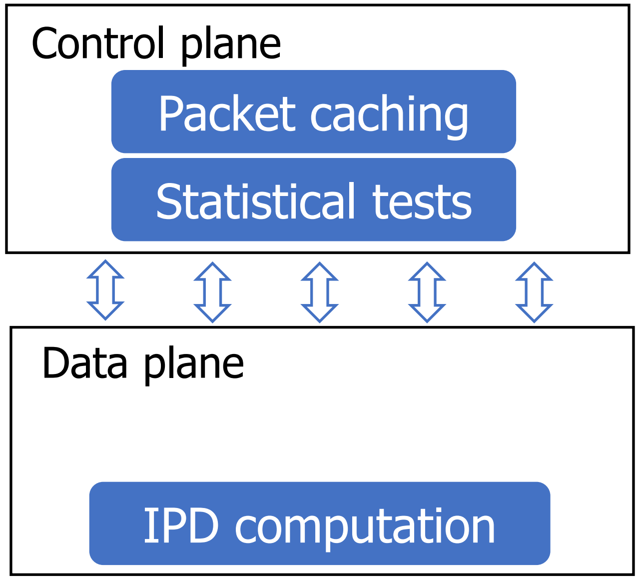
Applying the hardware/software co-design principle



Challenge #1: Efficient detection

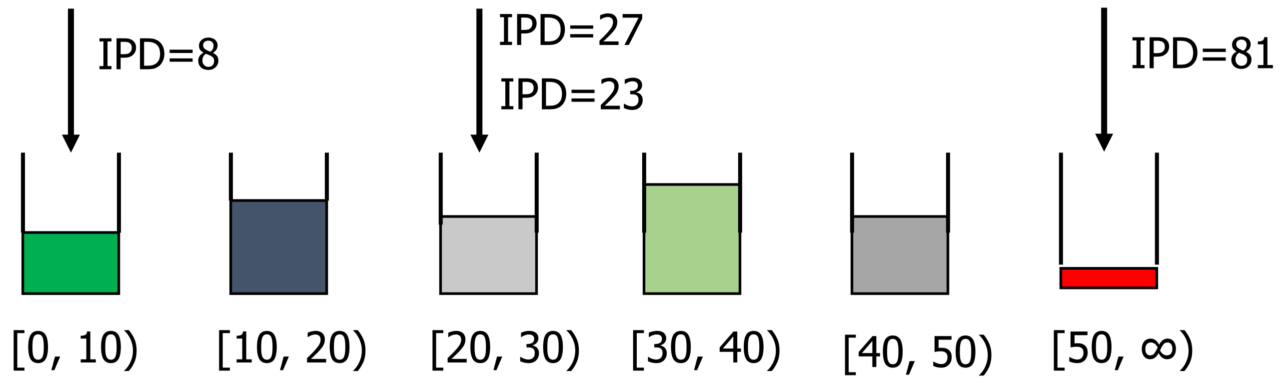
- **Solution:** Build efficient detections in **P4 switches** by applying the hardware/software co-design principle.

How to compute inter-packet delay (IPD)?



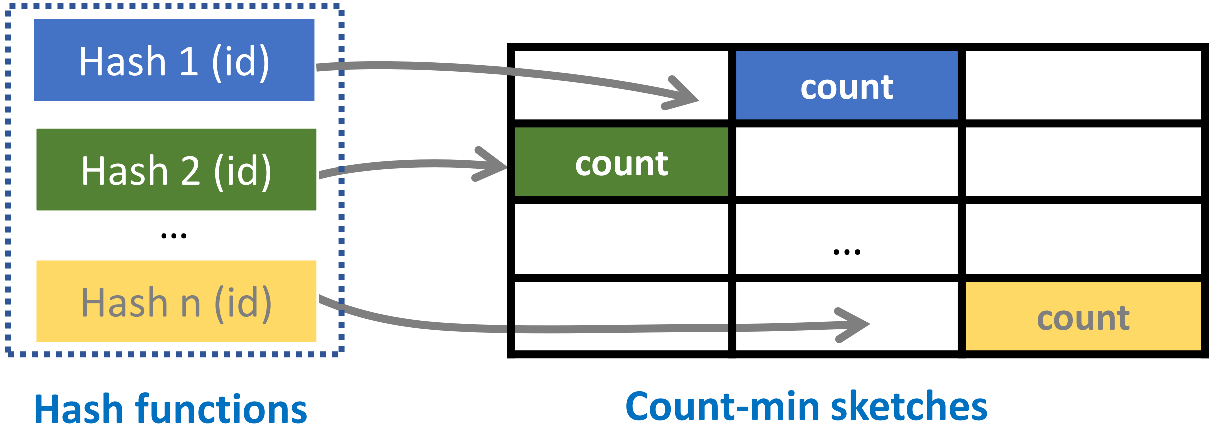
- Solution: Using ns-timescale P4 timestamps
  - IPD = Current timestamp – last timestamp
- Send all IPDs to the control plane directly?
  - **Crosstalk minimization principle!**

How to store IPDs in a memory-efficient manner?



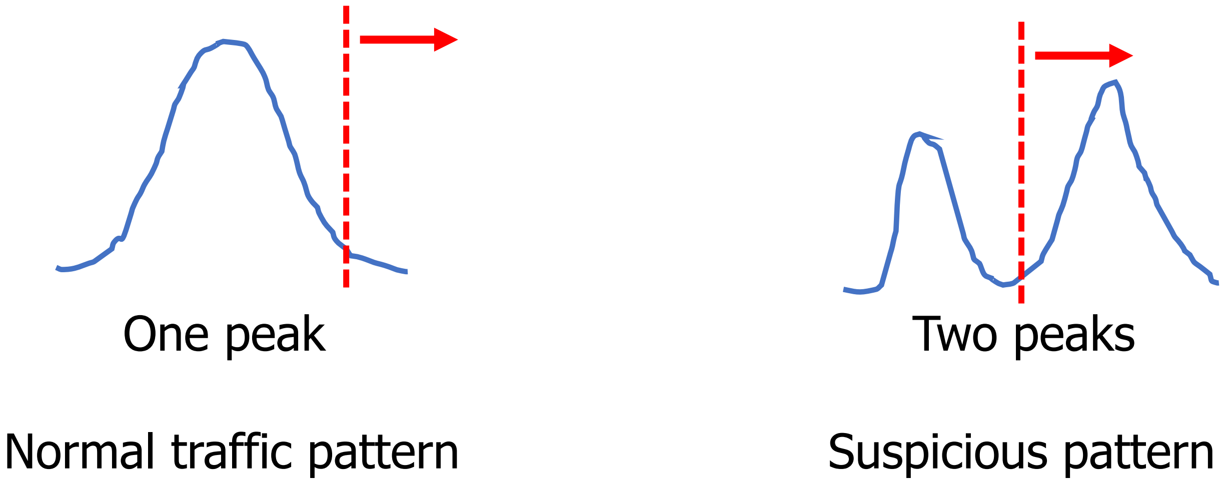
- Solution: IPD intervalization
  - Store IPD interval counters rather than exact IPDs

How to reduce per-flow memory consumption?



- Solution: IPD sketching
  - Trade off per-flow accuracy for space saving

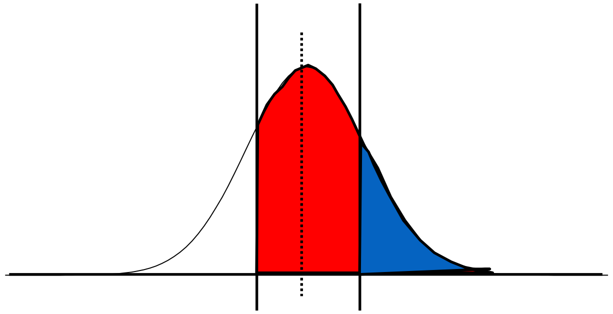
How to reduce IPDs sent to the control plane?

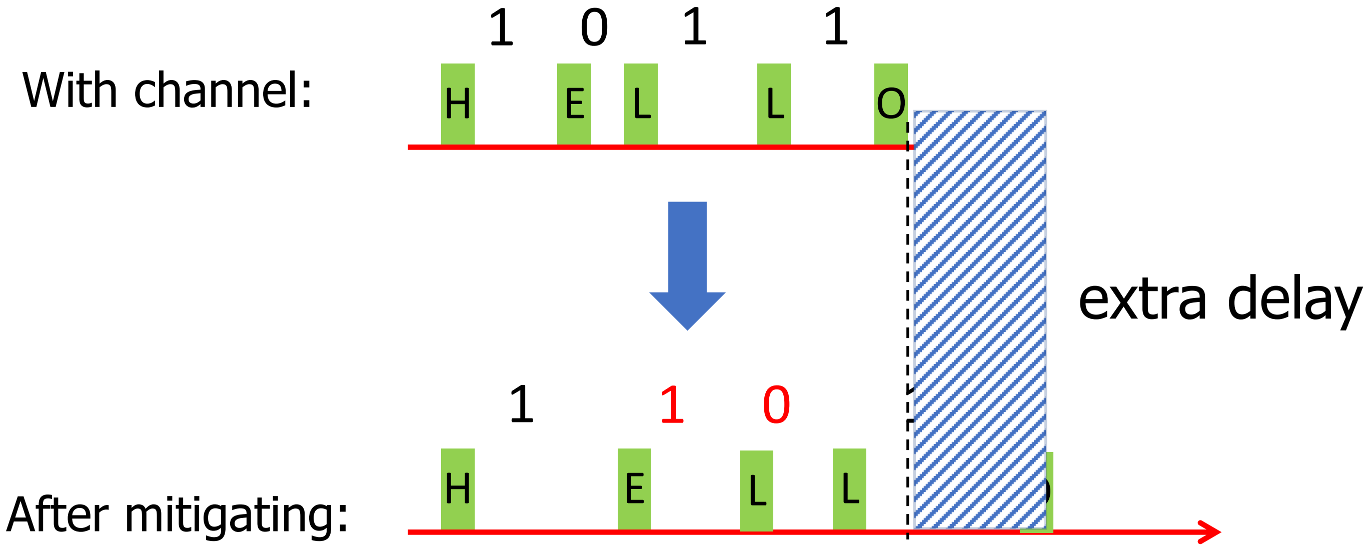


- Solution: IPD pre-check in the data plane
  - Do a quick check and only send suspicious flow IPDs to the control plane.

How to mitigate covert timing channels?

Statistical tests

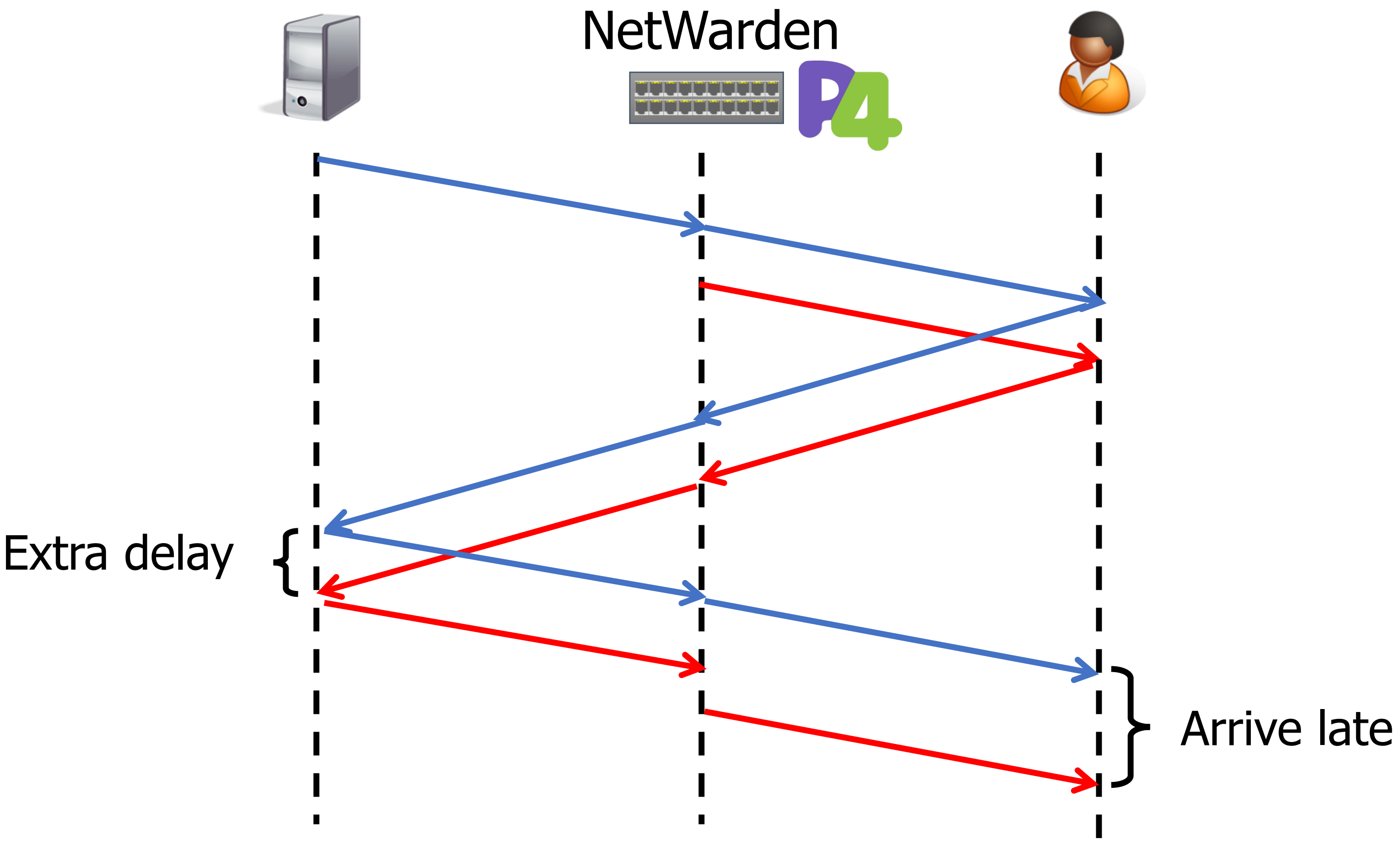




- Solution: Using the Control plane
  - Performs statistical tests
  - Adds random inter-packet delay by caching

- Note: This incurs extra delay.

Challenge #2: Performance-preserving mitigation



- Problem: Existing mitigations incur performance loss.

Challenge #2: Performance-preserving mitigation

- Solution: Temporarily boosting TCP performance to neutralize the performance penalty.

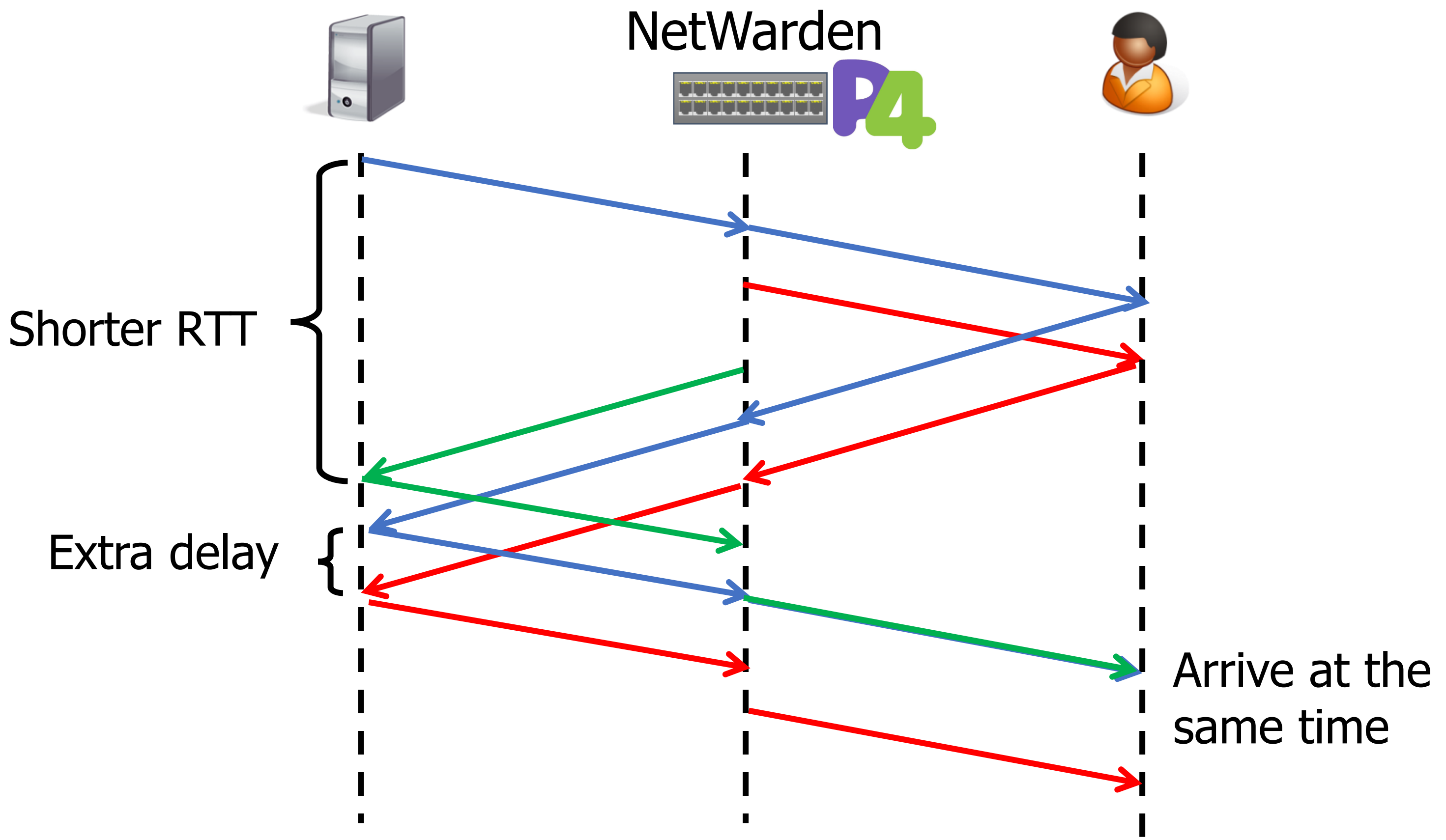
- Two boosters:
  - ACK booster: Generate ACK packets in advance.
  - Receive window booster: Enlarge receive window field temporarily.

Challenge #2: Performance-preserving mitigation

- Solution: Temporarily boosting TCP performance to neutralize the performance penalty.

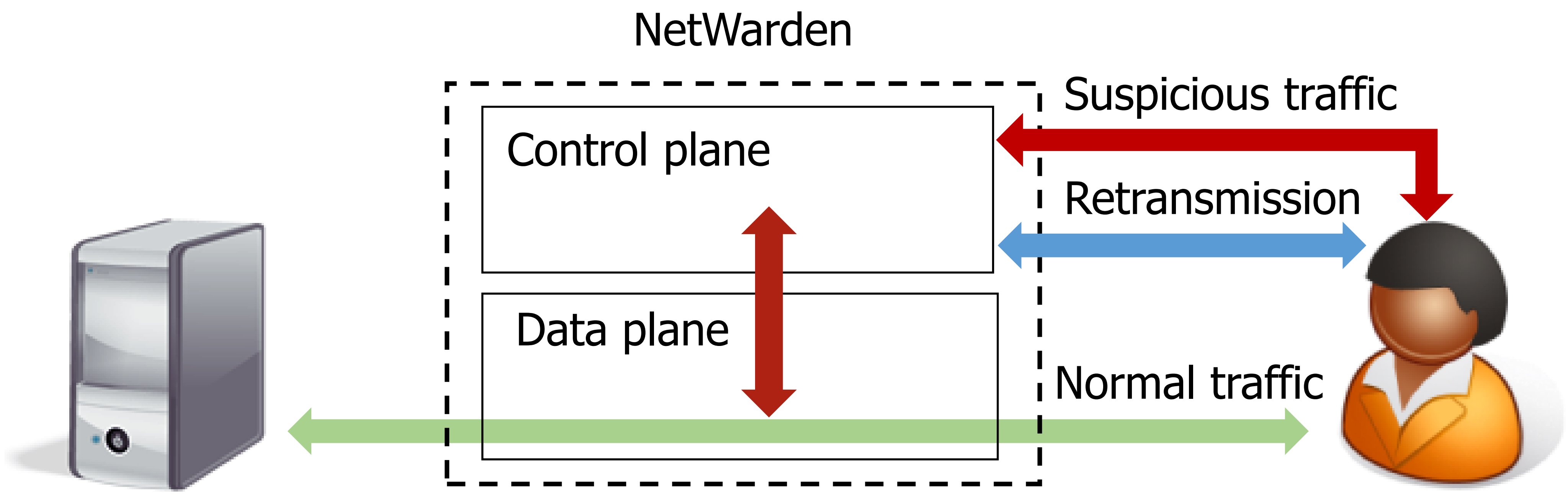
- Two boosters:
  - ACK booster: Generate ACK packets in advance.
  - Receive window booster: Enlarge receive window field temporarily.

Boosting performance: ACK booster



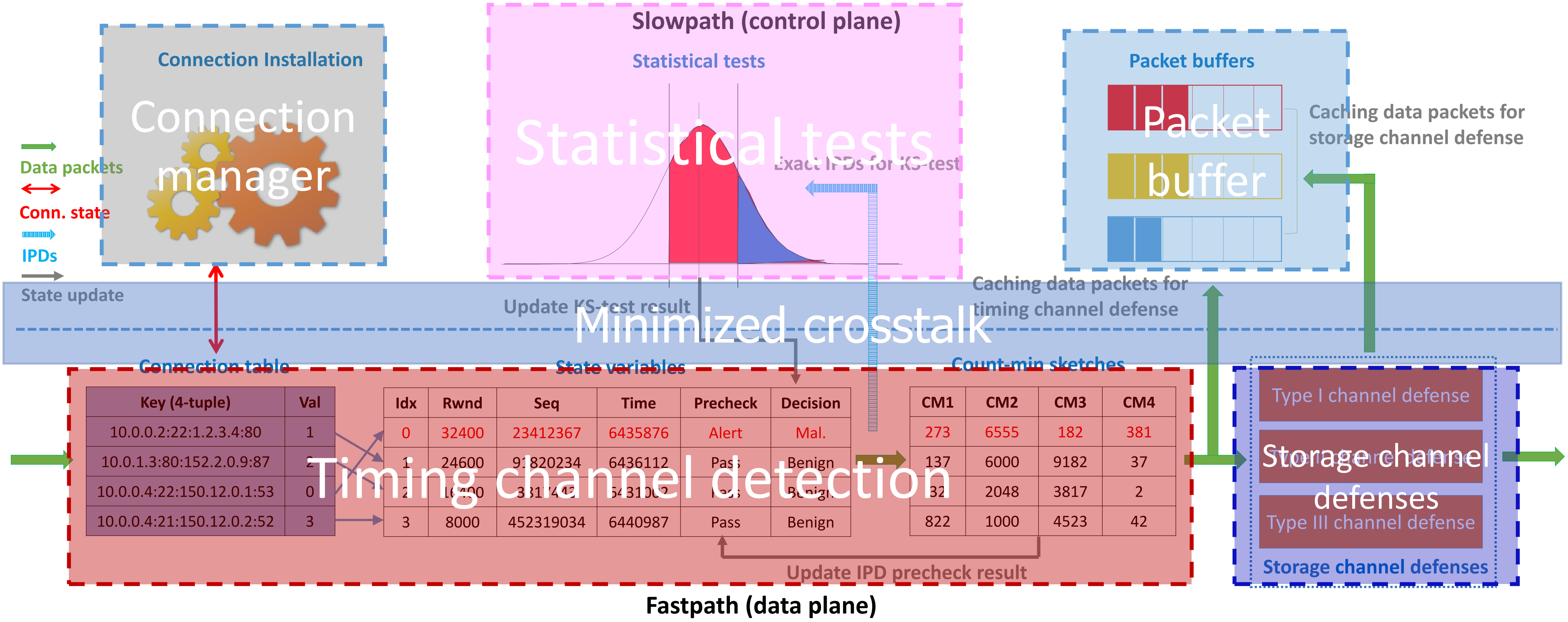
- Creates the **illusion** of a shorter latency as perceived by the sender.

Works as a TCP proxy



- NetWarden works as a **TCP proxy** for malicious traffic.

NetWarden panorama



Key (4-tuple)	Val
10.0.0.2:22:1.2.3.4:80	1
10.0.1.3:80:152.2.0.9:87	2
10.0.0.4:22:150.12.0.1:53	0
10.0.0.4:21:150.12.0.2:52	3

Idx	Rwnd	Seq	Time	Precheck	Decision
0	32400	23412367	6435876	Alert	Mal.
1	24600	91820234	6436112	Pass	Benign
2	[illegible]	[illegible]	[illegible]	Pass	Benign
3	8000	452319034	6440987	Pass	Benign

CM1	CM2	CM3	CM4
273	6555	182	381
137	6000	9182	37
32	2048	3817	2
822	1000	4523	42

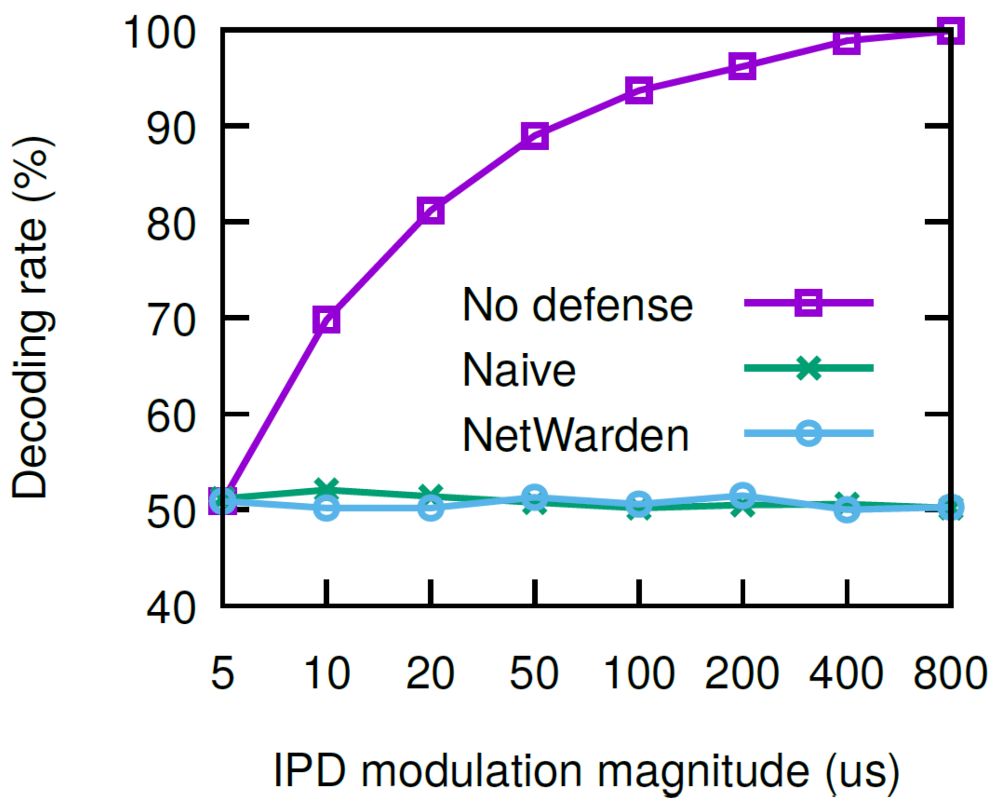
Outline

- ✓ Motivation: Mitigating network covert channels
- ✓ State of the art: Performance penalty
- ✓ Approach: NetWarden
- ✓ NetWarden design
  - Principles of hardware/software co-design
  - Challenge #1: Efficient detection
  - Challenge #2: Performance-preserving mitigation
- ➡ Evaluation
- Conclusion

Evaluation setup

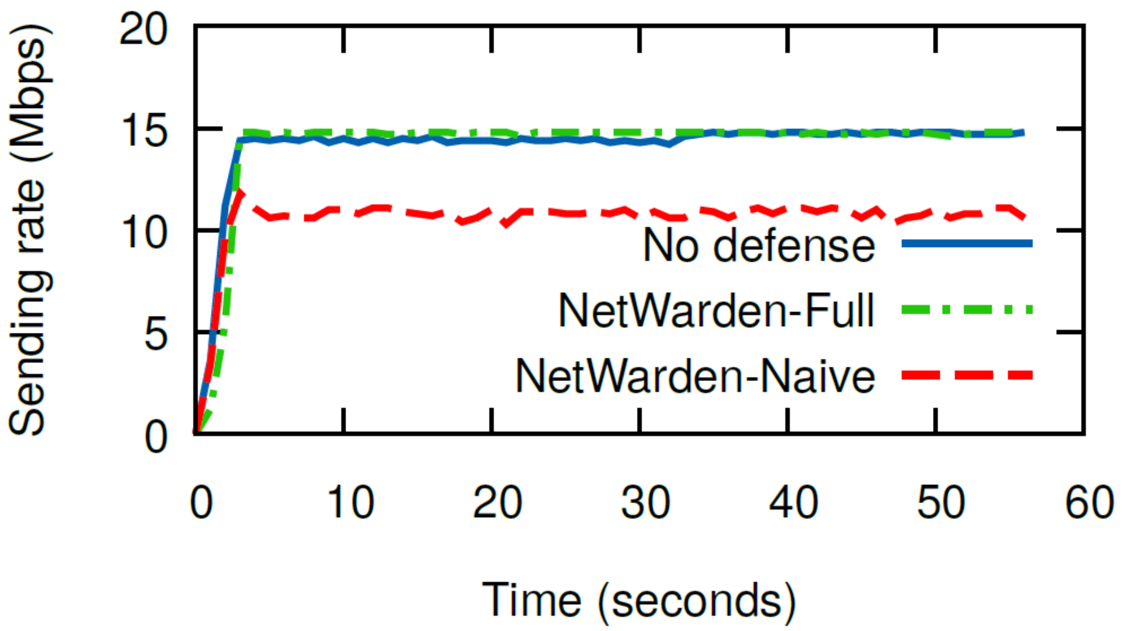
- NetWarden prototype:
  - Runs in Tofino Wedge 100BF-32X switch.
  - 2500 LoC of P4 + 3000 LoC of C+Python
- Threat model:
  - A compromised server + a trusted P4 switch running NetWarden
  - Leak a 2048-bit RSA key via covert channel.
- Workloads:
  - Web search (Alizadeh-SIGCOMM'15)
- Baseline:
  - Defenses without performance boosting

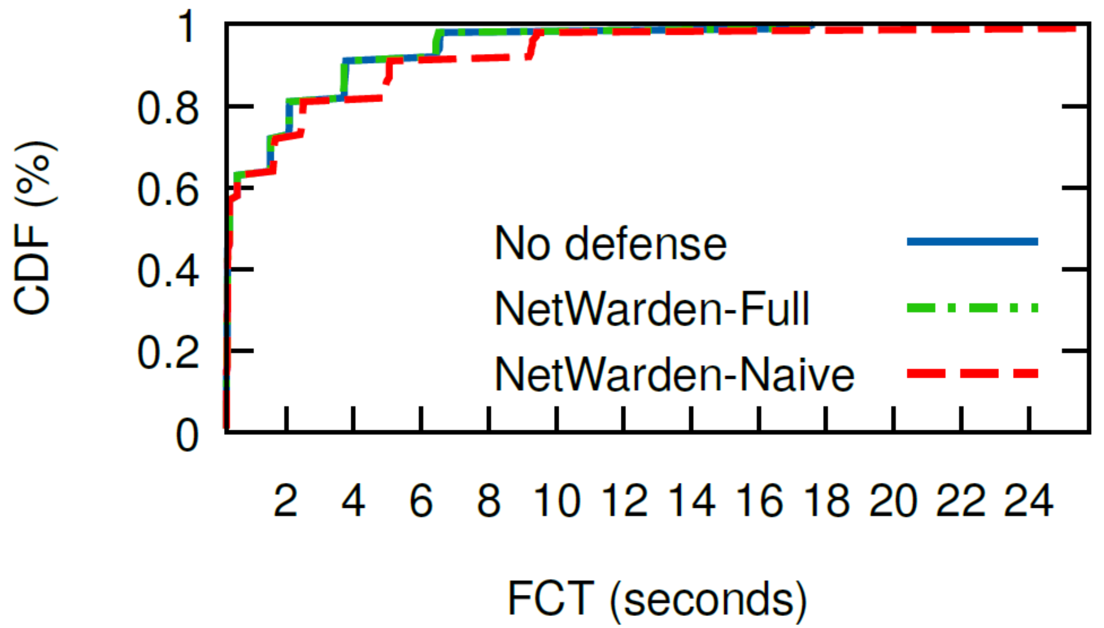
How effective is NetWarden in covert channels mitigation ?



- Naïve defense: renders decoding to a random guess.
- NetWarden: very close to a random guess.
- NetWarden can mitigate covert channels effectively.

How well does NetWarden preserve performance?





- Naïve defense incurs 25% performance penalty.
- NetWarden only has 1% performance deviation.
- NetWarden can mitigate covert channels with minimal performance loss.

See more evaluation results in our paper

- How effective is NetWarden in covert channels mitigation?
- How well does NetWarden preserve performance?
- How well does NetWarden work with different TCP variant?
- How scalable is NetWarden?
- How much overhead does NetWarden incur?
- How well does NetWarden work with complex applications?
- How robust is NetWarden in self attacks?

Outline

- ✓ - Motivation: Mitigating network covert channels
- ✓ - State of the art: Performance penalty
- ✓ - Approach: NetWarden
- ✓ - NetWarden design
  - Principles of hardware/software co-design
  - Challenge #1: Efficient detection
  - Challenge #2: Performance-preserving mitigation
- ✓ - Evaluation
- → - Conclusion

Conclusion

- Motivation: Mitigating network covert channels
- Key limitation of existing approaches:
  - **Performance penalty**
- Our approach: **NetWarden**
  - Principles of hardware/software co-design
  - Efficient detection and mitigation
  - Performance preservation
- Evaluation:
  - Mitigates covert channels with minimum performance loss!

Thank You!

Email: jxing@rice.edu
Personal web: https://jxing.me/
Project web: https://qiaokang.org/poise-web/