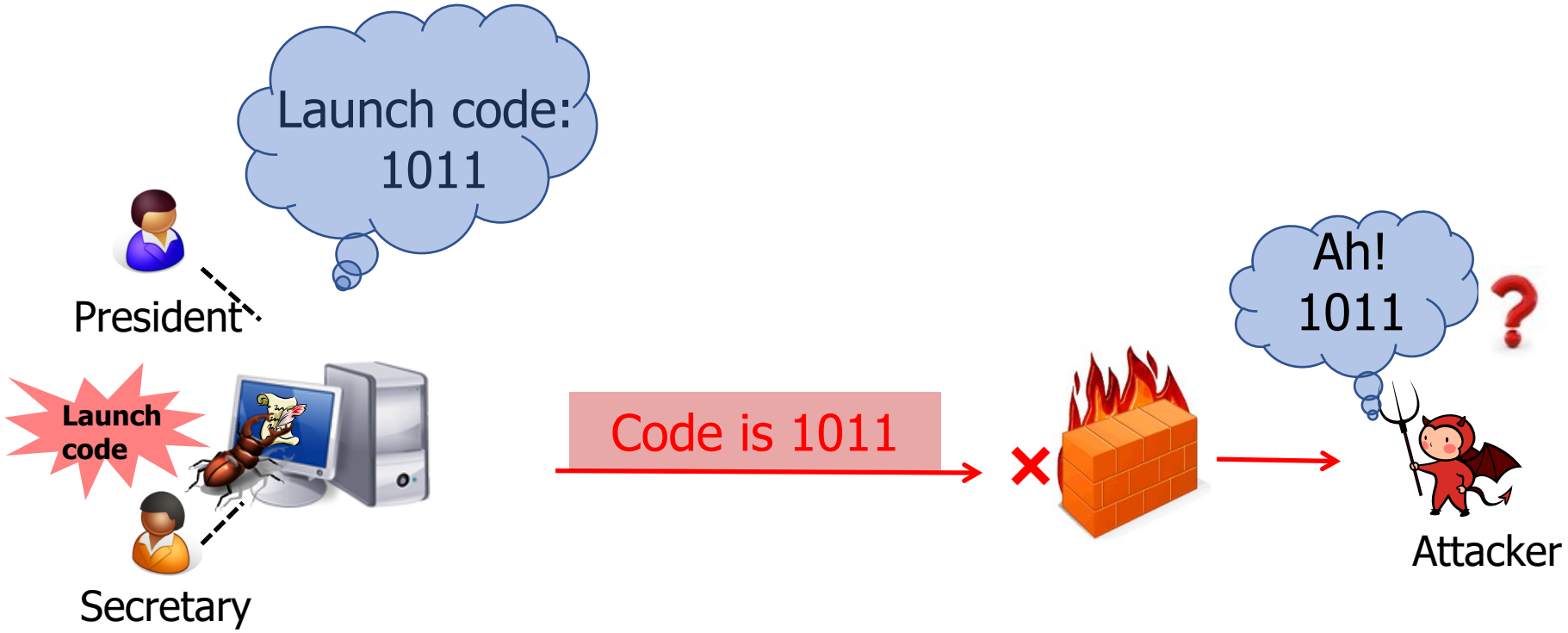
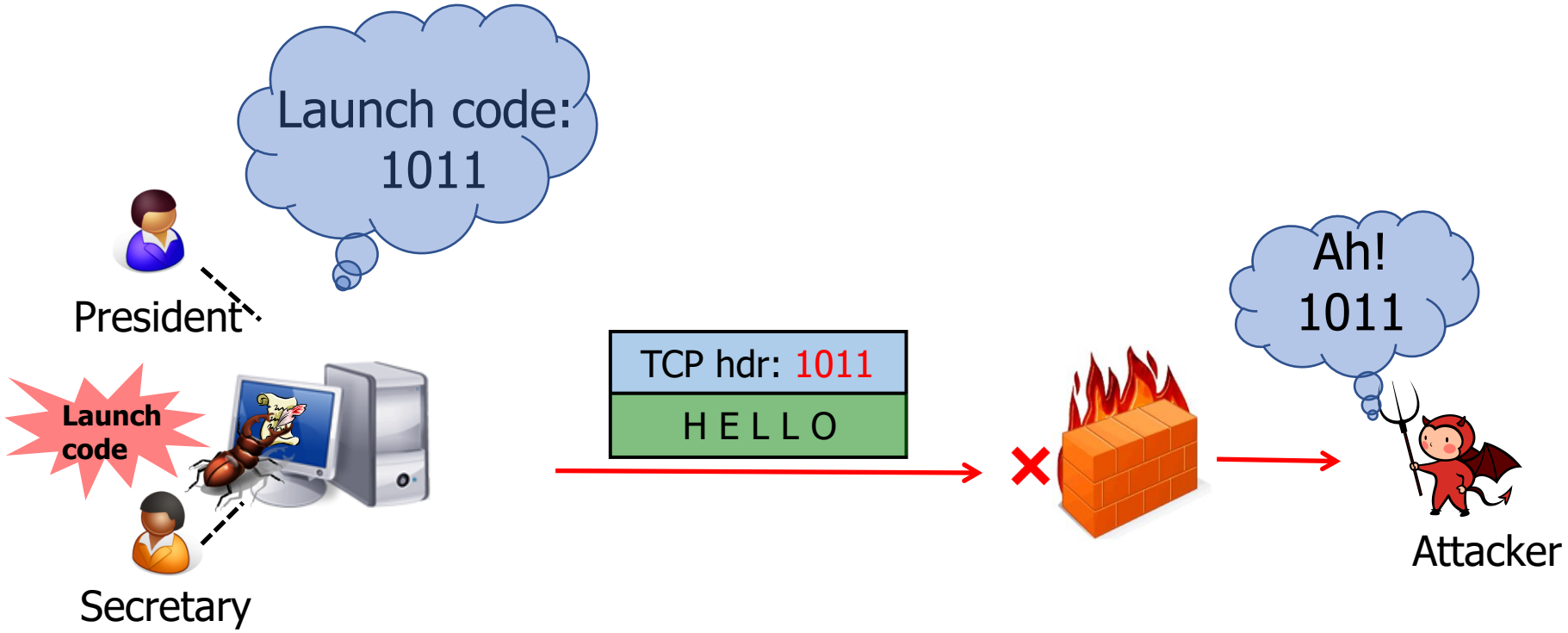
NetWarden: Mitigating Network Covert Channels while Preserving Performance

Jiarong Xing, Qiao Kang, Ang Chen
Rice University

Motivation: Mitigating network covert channels

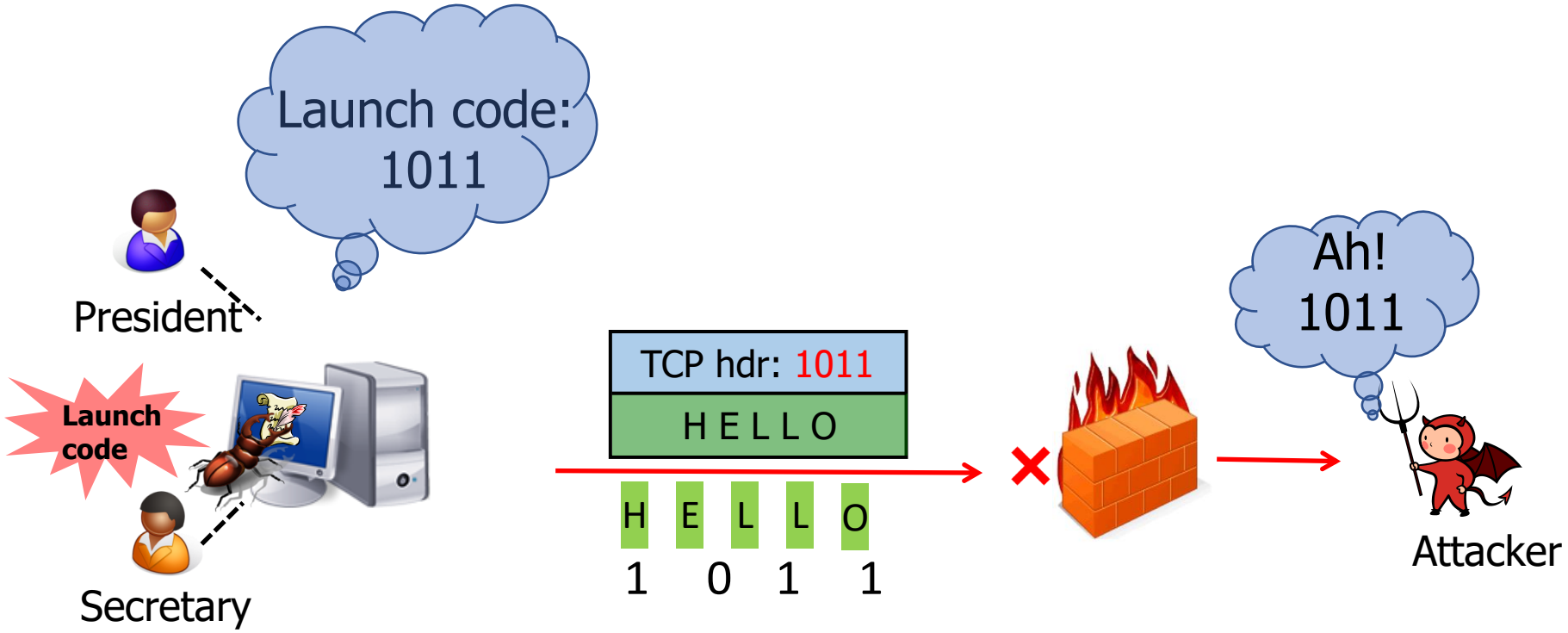


Motivation: Mitigating network covert channels



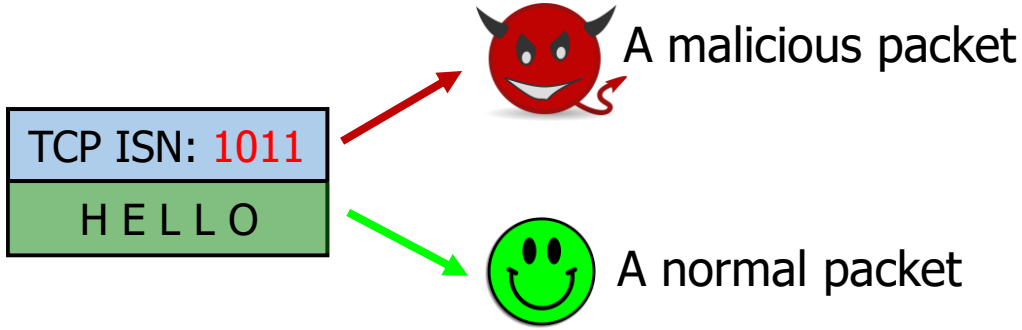
- Covert channels:
  - Storage channels: changing the packet **header fields**.

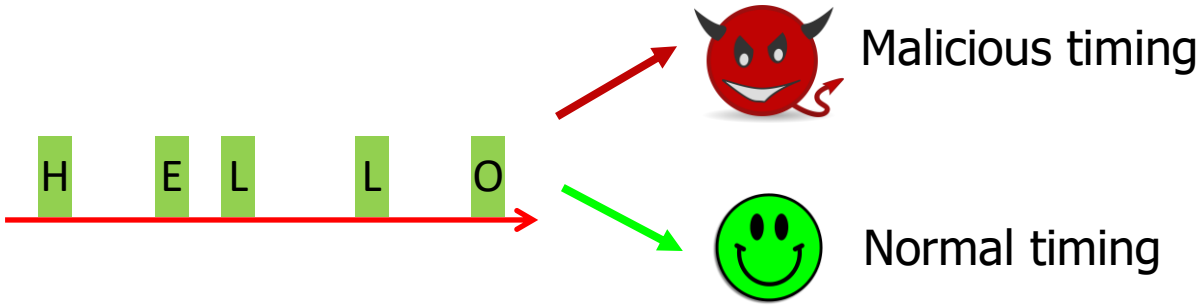
Motivation: Mitigating network covert channels



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

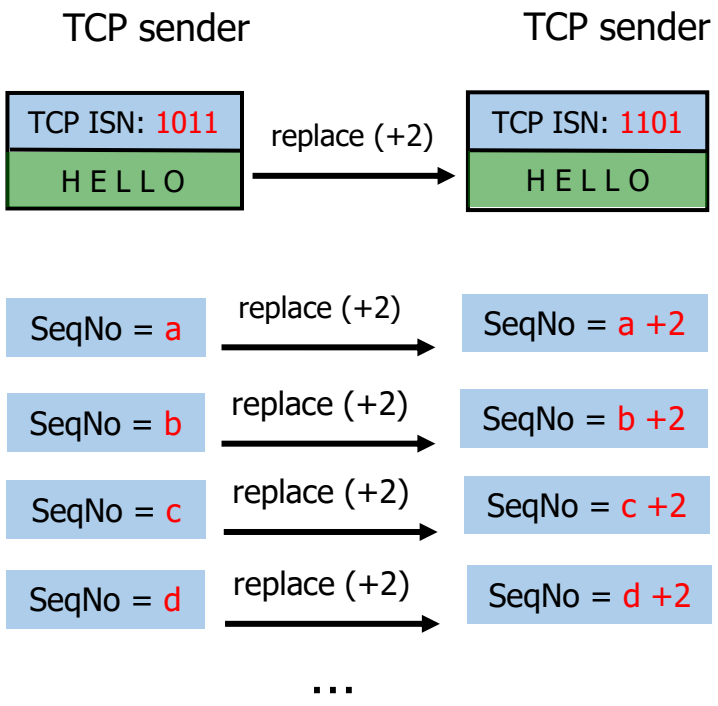
Covert channels are a difficult problem

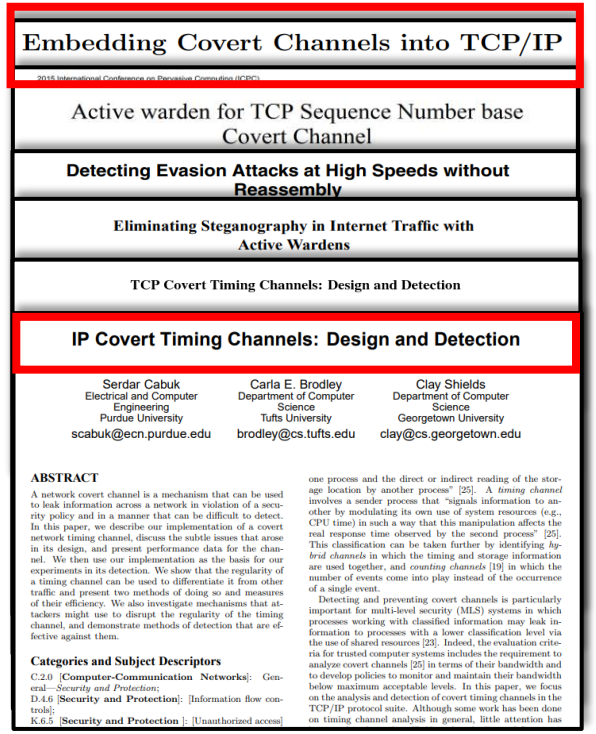




- Detection is difficult due to non-determinism
  - Protocols fields, e.g., TCP initial sequence numbers
  - Protocol events, e.g., inter-packet gaps
- Mitigation is difficult due to high traffic volume
  - Operate on every packet of Tbps traffic

State of the art





Embedding Covert Channels into TCP/IP

2015 International Conference on Pervasive Computing (ICPC)

Active warden for TCP Sequence Number base Covert Channel

Detecting Evasion Attacks at High Speeds without Reassembly

Eliminating Steganography in Internet Traffic with Active Wardens

TCP Covert Timing Channels: Design and Detection

IP Covert Timing Channels: Design and Detection

Serdar Cabuk, Electrical and Computer Engineering, Purdue University, scabuk@ecn.purdue.edu
Carla E. Brodley, Department of Computer Science, Tufts University, brodley@cs.tufts.edu
Clay Shields, Department of Computer Science, Georgetown University, clay@cs.georgetown.edu

ABSTRACT

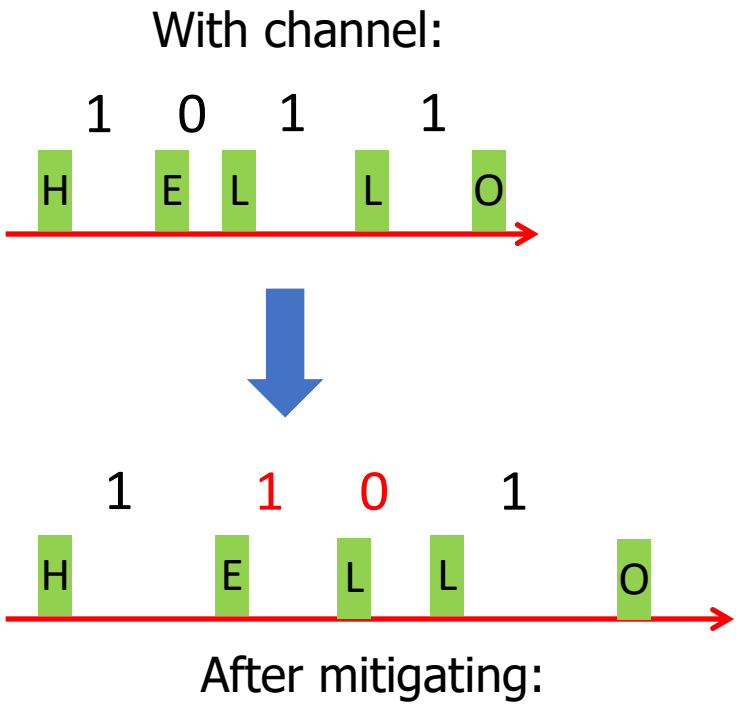
A network covert channel is a mechanism that can be used to leak information across a network in violation of a security policy and in a manner that can be difficult to detect. In this paper, we describe our implementation of a covert network timing channel, discuss the subtle issues that arose in its design, and present performance data for the channel. We then use our implementation as the basis for our experiments in its detection. We show that the regularity of a timing channel can be used to differentiate it from other traffic and present two methods of doing so and measures of their efficiency. We also investigate mechanisms that attackers might use to disrupt the regularity of the timing channel, and demonstrate methods of detection that are effective against them.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General—*Security and Protection*;
D.4.6 [Security and Protection]: [Information flow controls];
K.6.5 [Security and Protection]: [Unauthorized access]

one process and the direct or indirect reading of the storage location by another process" [25]. A *timing channel* involves a sender process that "signals information to another by modulating its own use of system resources (e.g., CPU time) in such a way that this manipulation affects the real response time observed by the second process" [25]. This classification can be taken further by identifying *hybrid channels* in which the timing and storage information are used together, and *counting channels* [19] in which the number of events come into play instead of the occurrence of a single event.

Detecting and preventing covert channels is particularly important for multi-level security (MLS) systems in which processes working with classified information may leak information to processes with a lower classification level via the use of shared resources [23]. Indeed, the evaluation criteria for trusted computer systems includes the requirement to analyze covert channels [25] in terms of their bandwidth and to develop policies to monitor and maintain their bandwidth below maximum acceptable levels. In this paper, we focus on the analysis and detection of covert timing channels in the TCP/IP protocol suite. Although some work has been done on timing channel analysis in general, little attention has



- Many defenses have been proposed:
  - ISN storage channel:
    - Add the same offset to TCP sequence numbers
  - IP timing channel:
    - Add random delays to destroy original timing

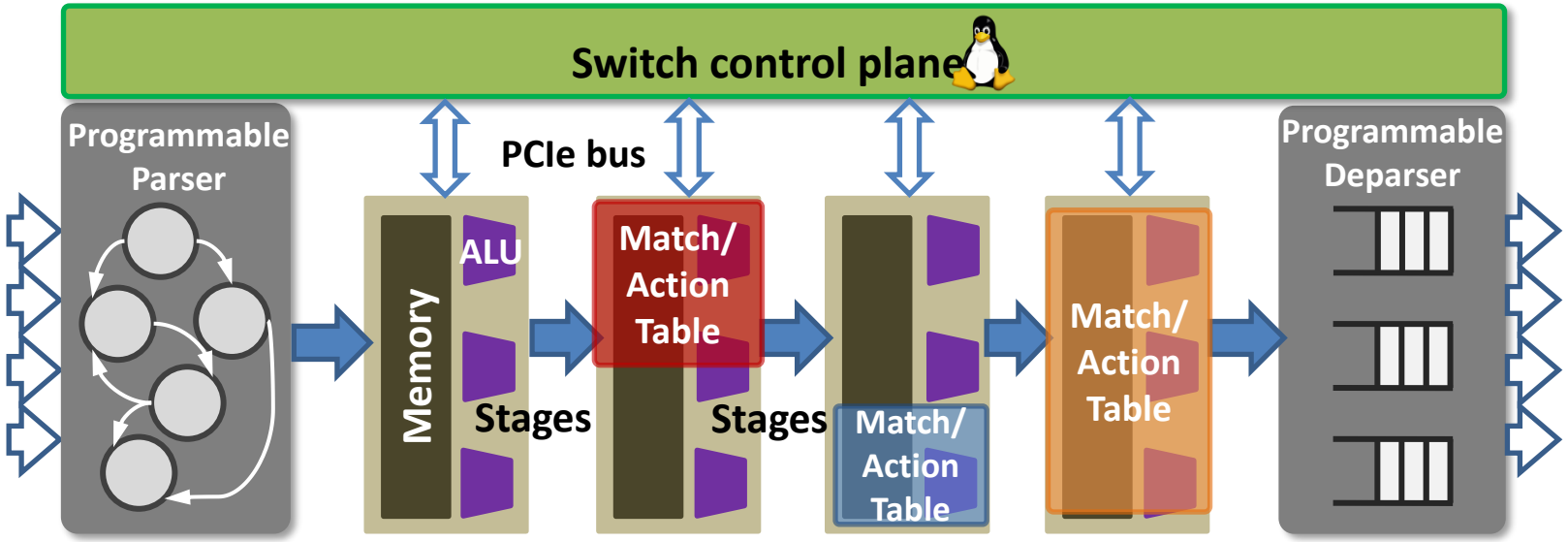
Common problem: Performance penalty

- Per-packet operation running in software is very inefficient
  - Switches have to handle Tbps traffic
- Incur extra delay to TCP connections
  - Many flows are delay sensitive
- Not practical for deployment!

Key question

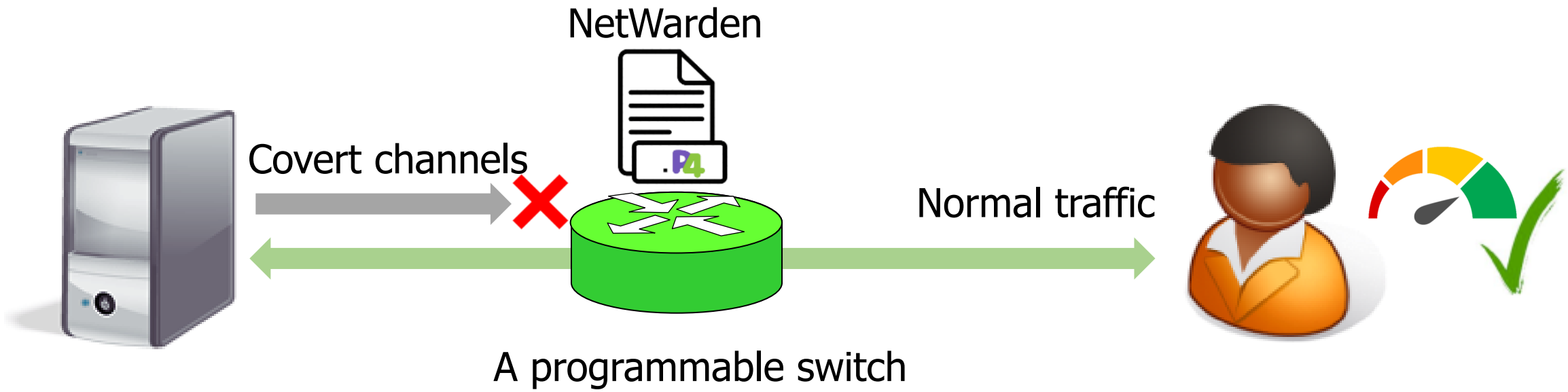
Can we mitigate covert channels while **preserving performance**?

New opportunity: Programmable switches



- Programmable switches
  - Can be programmed in high-level languages (e.g., P4)
  - Run at linespeed (Tbps)
  - Widely used in network community:
    - SilkRoad-Sigcomm'17, Speedlight-Sigcomm'18, Blink-NSDI'19, Contra-NSDI'20, etc.
  - An ideal candidate for covert channel defenses!

NetWarden: A programmable switch defense



- NetWarden mitigates covert channels while preserving performance.

Key features

Effective defenses

Defend against a wide range of covert channels

High throughput

~100Gbps per switch port

NetWarden

Performance preservation

< 1% performance deviation

Great transparency

Totally transparent to end hosts

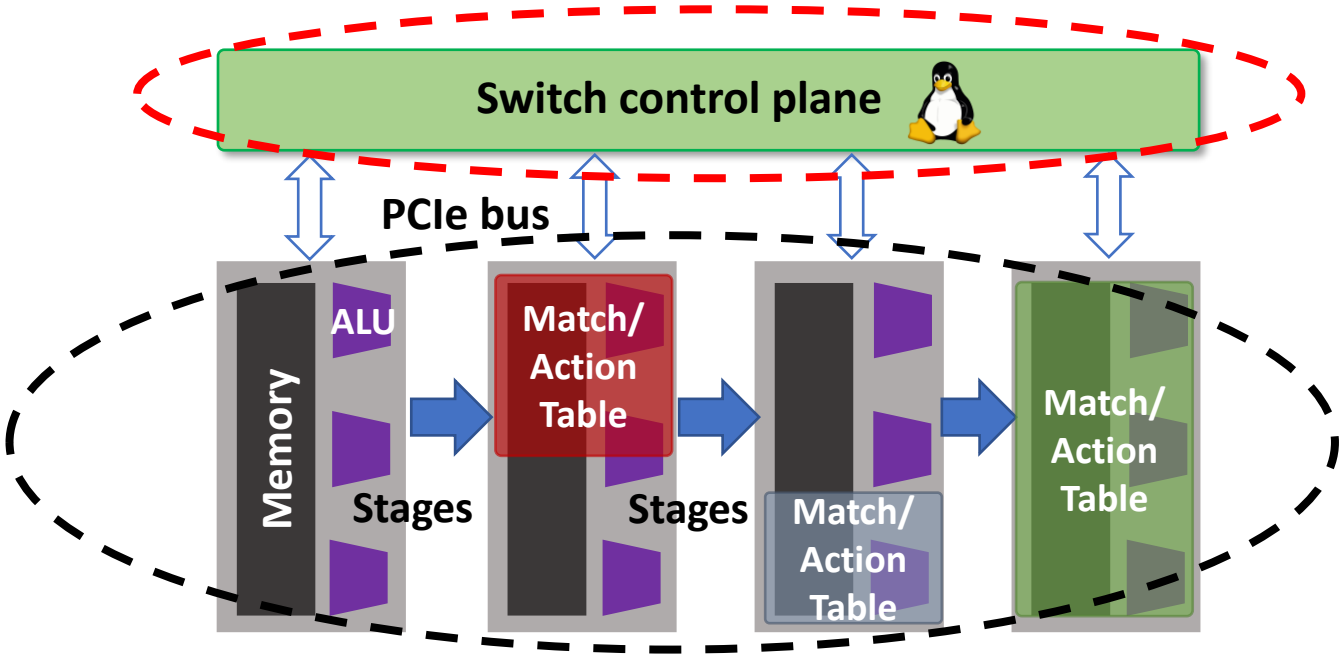
Outline

- ✓ Motivation: Mitigating network covert channels
- ✓ State of the art: Performance penalty
- ✓ Approach: NetWarden
- → NetWarden design
  - Challenge #1: Limited programming model
  - Challenge #2: Performance preservation
- Evaluation
- Conclusion

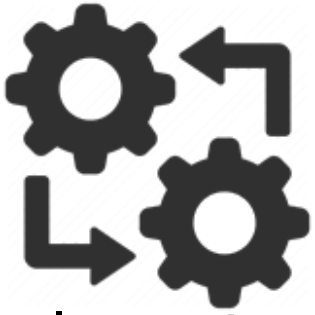
Challenge #1: Limited programming model

- **Solution:** hardware/software co-design principle.

Programmable switch anatomy



- Data plane
  - Header modification, ns timestamp, per-flow state, line speed
  - Limited memory, limited arithmetic computation
- Control plane
  - Abundant memory, complex arithmetic computation
  - Software speed, make the circle disappear



Complementary!

Applying the hardware/software co-design principle

Switch control plane

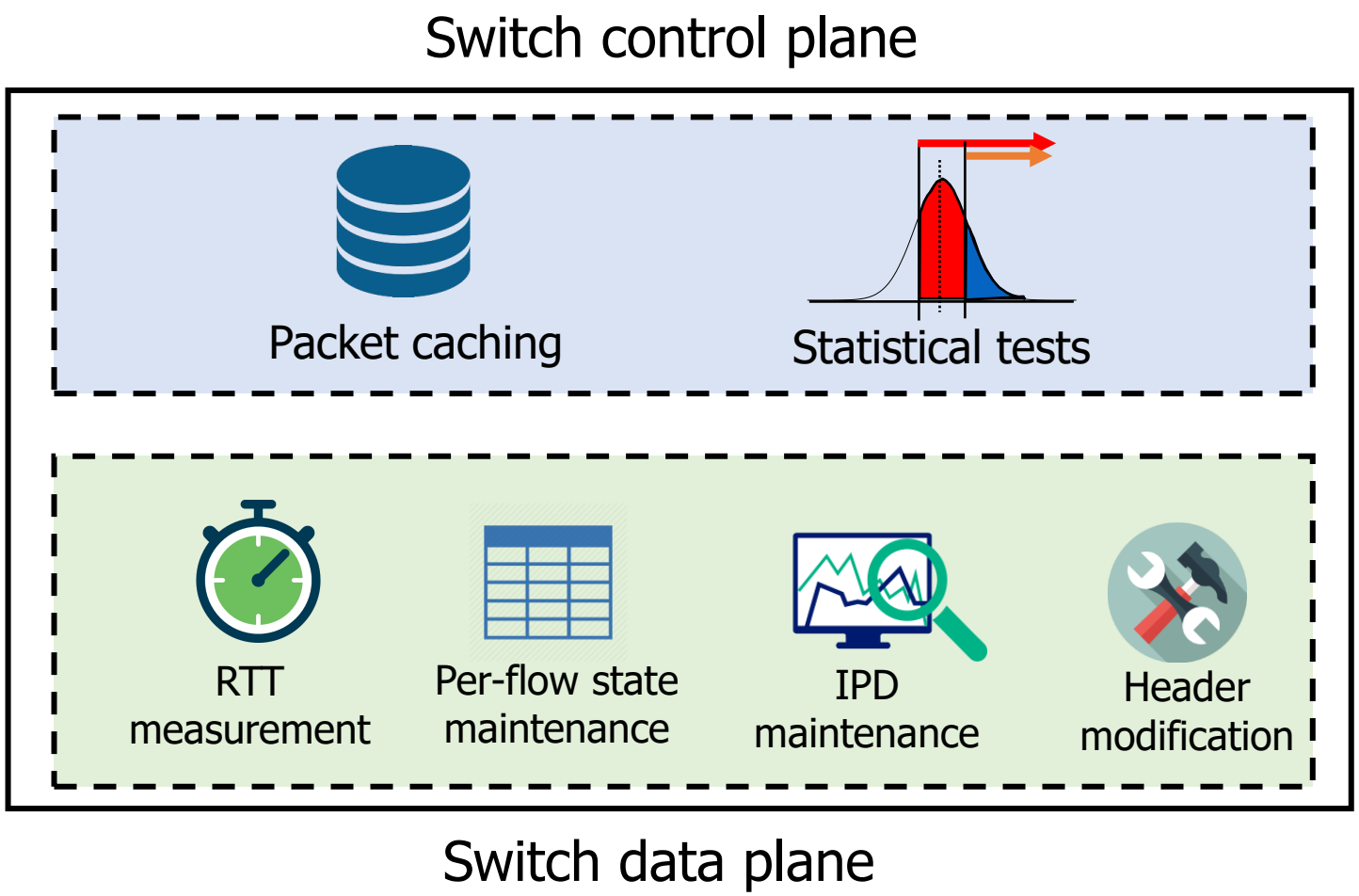
- Computation: Batch operations
- Memory: Growing state

- Minimize crosstalk

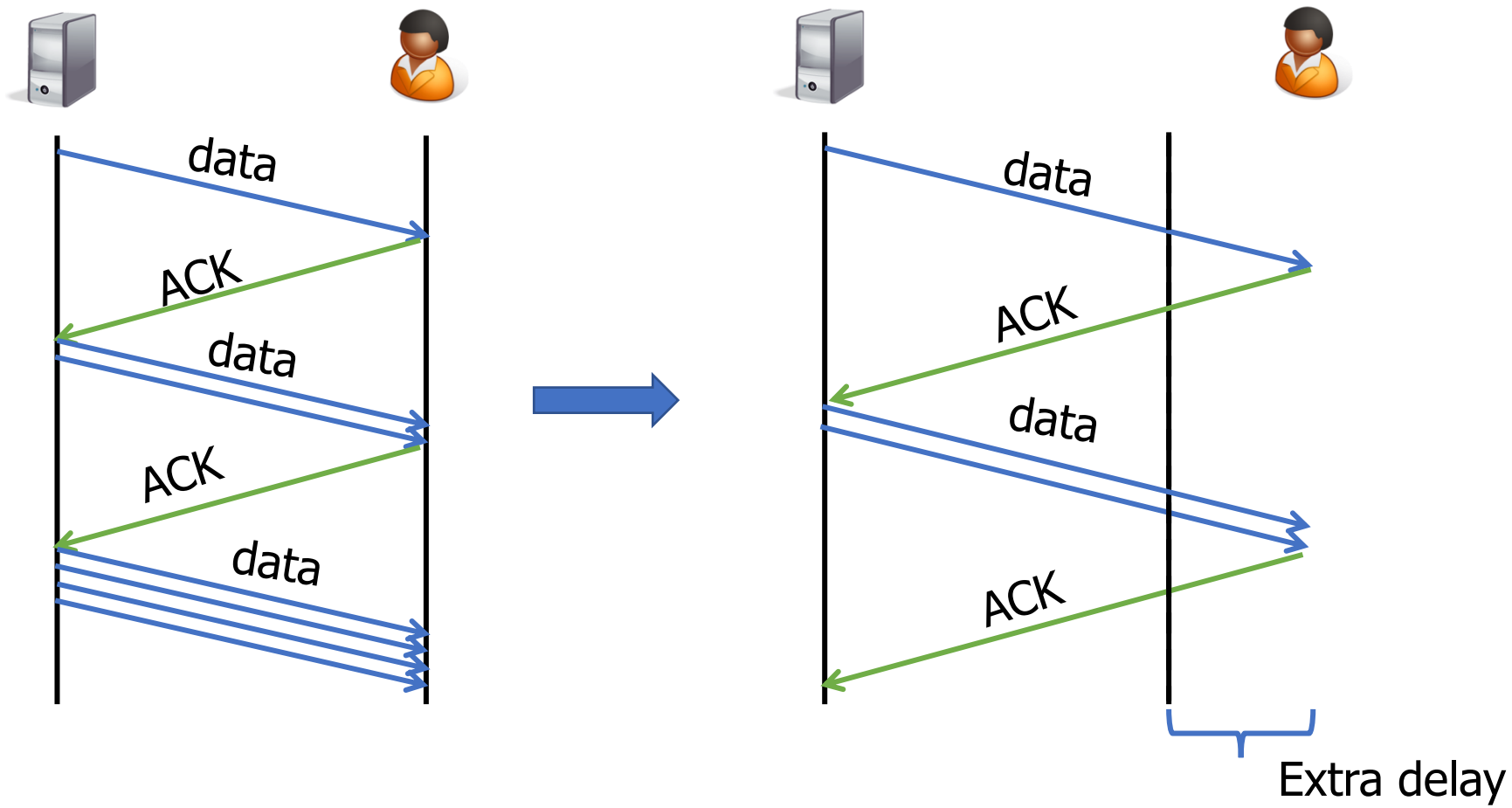
- Computation: Per-packet operations
- Memory: Constant state

Switch data plane

Applying the hardware/software co-design principle



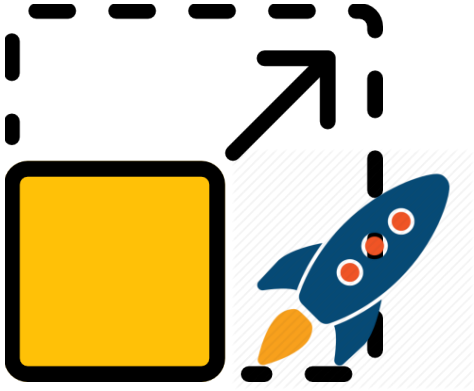
Challenge #2: Performance preservation



- Problem: Existing mitigations incur performance loss.
- Takeaway: Create the illusion of the same RTT for the sender.

Solution: Performance boosters

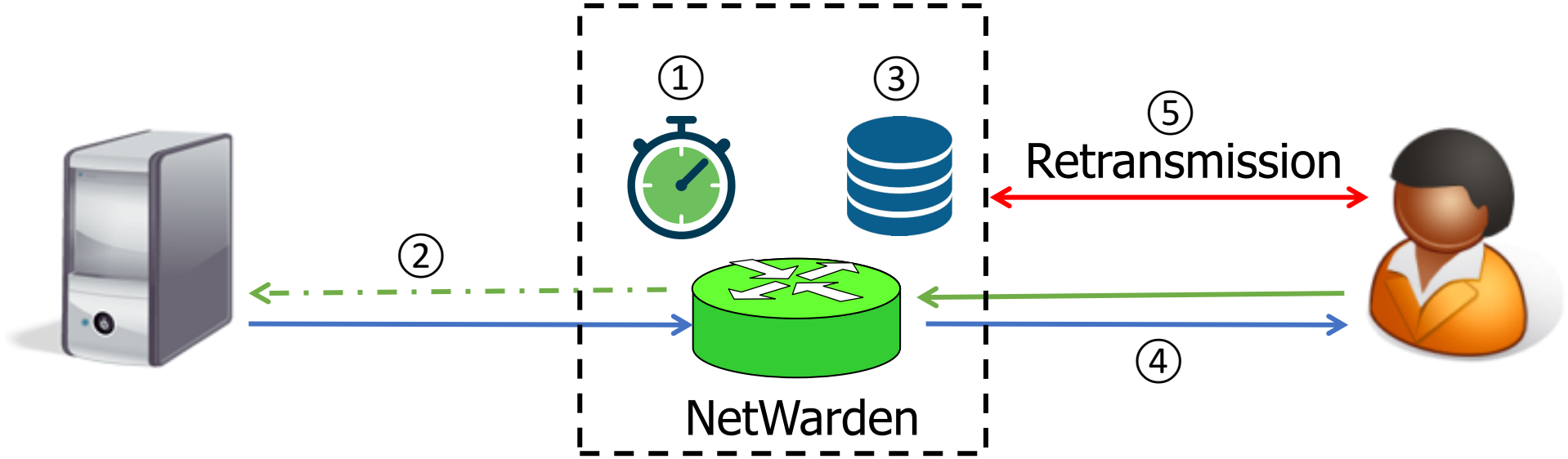
ACK booster



Receiving window booster

- ACK booster
  - Maintain the same RTT by generating ACK packets in advance.
- Receiving window booster
  - Send more data per RTT by enlarging the receiving window size.

Performance preservation: ACK booster



- Creates the **illusion** of the same RTT as perceived by the sender.

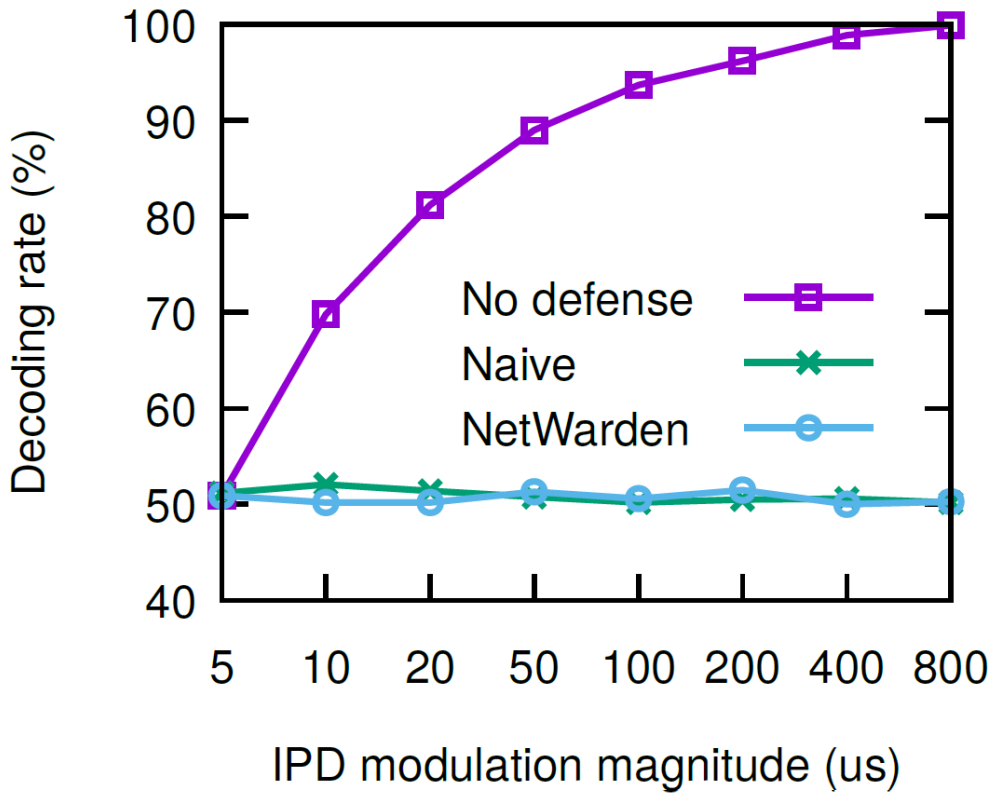
Outline

- ✓ Motivation: Mitigating network covert channels
- ✓ State of the art: Performance penalty
- ✓ Approach: NetWarden
- ✓ NetWarden design
  - Challenge #1: Limited programming model
  - Challenge #2: Performance preservation
- ➡ Evaluation
- Conclusion

Evaluation setup

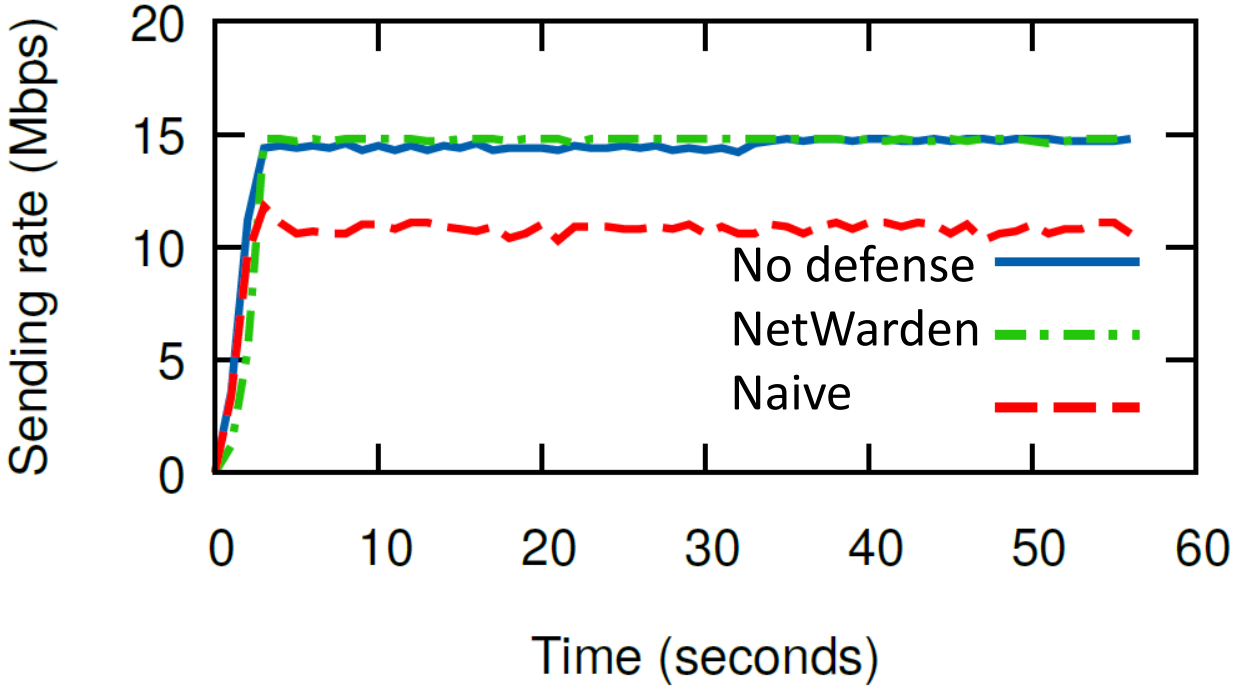
- NetWarden prototype:
  - Runs in Tofino Wedge 100BF-32X switch.
  - 2500 LoC of P4 + 3000 LoC of C+Python
- Threat model:
  - A compromised server + a trusted P4 switch running NetWarden
  - Leak a 2048-bit RSA key via covert channel.
- Real world applications:
  - Apache servers, FTP servers, Nodejs servers
- Baseline:
  - No defense: No covert channel defenses are deployed.
  - Naïve defense: Covert channel defenses without performance preservation

How effective is NetWarden in covert channels mitigation ?



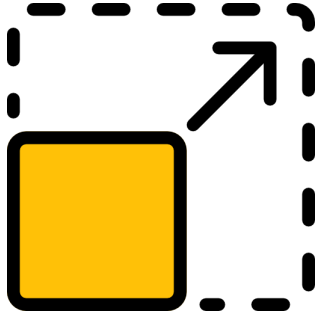
- Naïve defense: renders decoding to a random guess.
- NetWarden: very close to a random guess.
- NetWarden can mitigate covert channels **effectively**.

How well does NetWarden preserve performance?



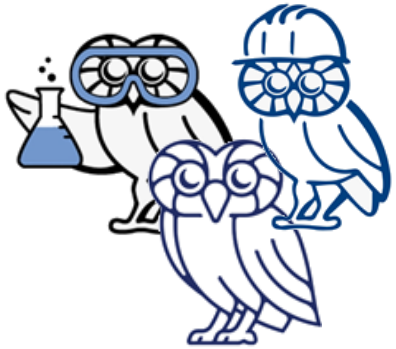
- Naïve defense incurs 25% performance penalty.
- NetWarden only has 1% performance deviation.
- NetWarden can mitigate covert channels **with minimal performance loss**.

See more evaluation results in our paper



System scalability

System overhead



Different TCP variants

Complex Applications

Self-defense techniques

Outline

- ✓ Motivation: Mitigating network covert channels
- ✓ State of the art: Performance penalty
- ✓ Approach: NetWarden
- ✓ NetWarden design
  - Challenge #1: Limited programming model
  - Challenge #2: Performance preservation
- ✓ 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 defenses
  - Performance preservation
- Evaluation:
  - Mitigates covert channels with **minimum performance loss**!
- Source code: https://github.com/jiarong0907/NetWarden
- Looking for internship in 2021 summer!

Questions?