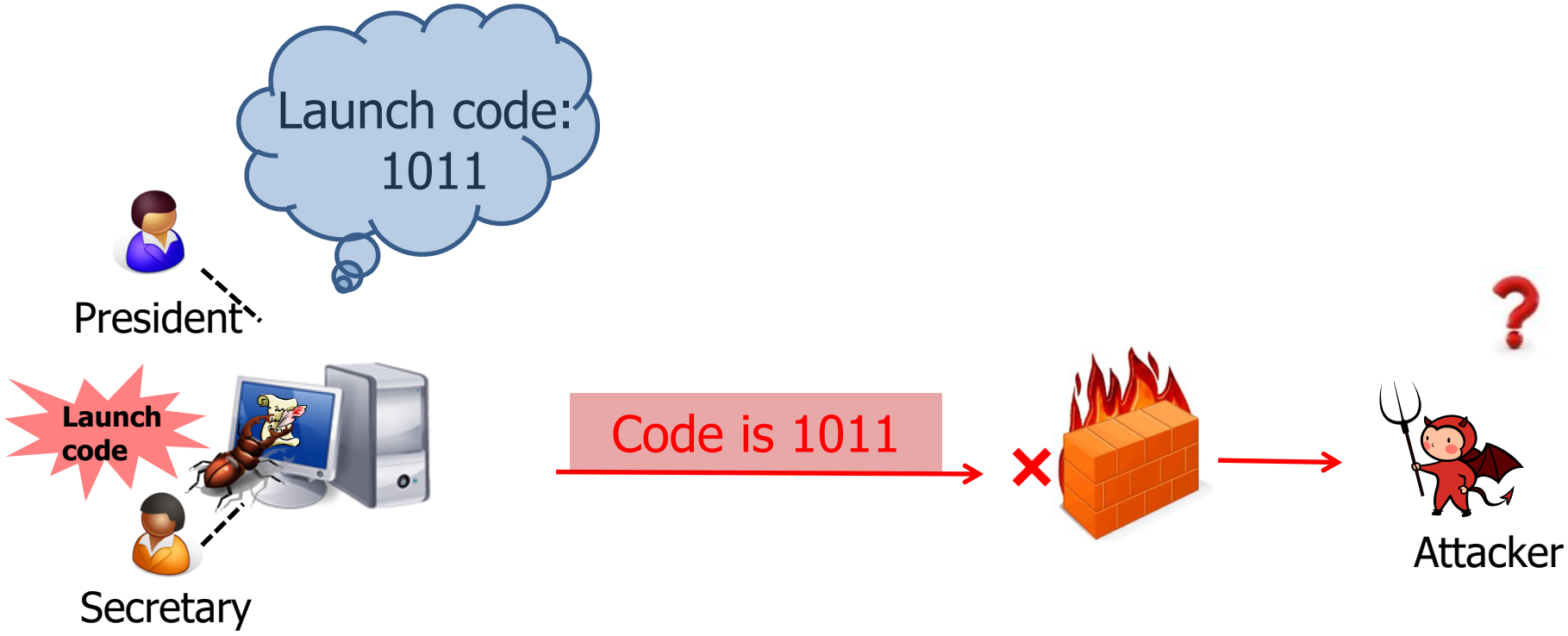
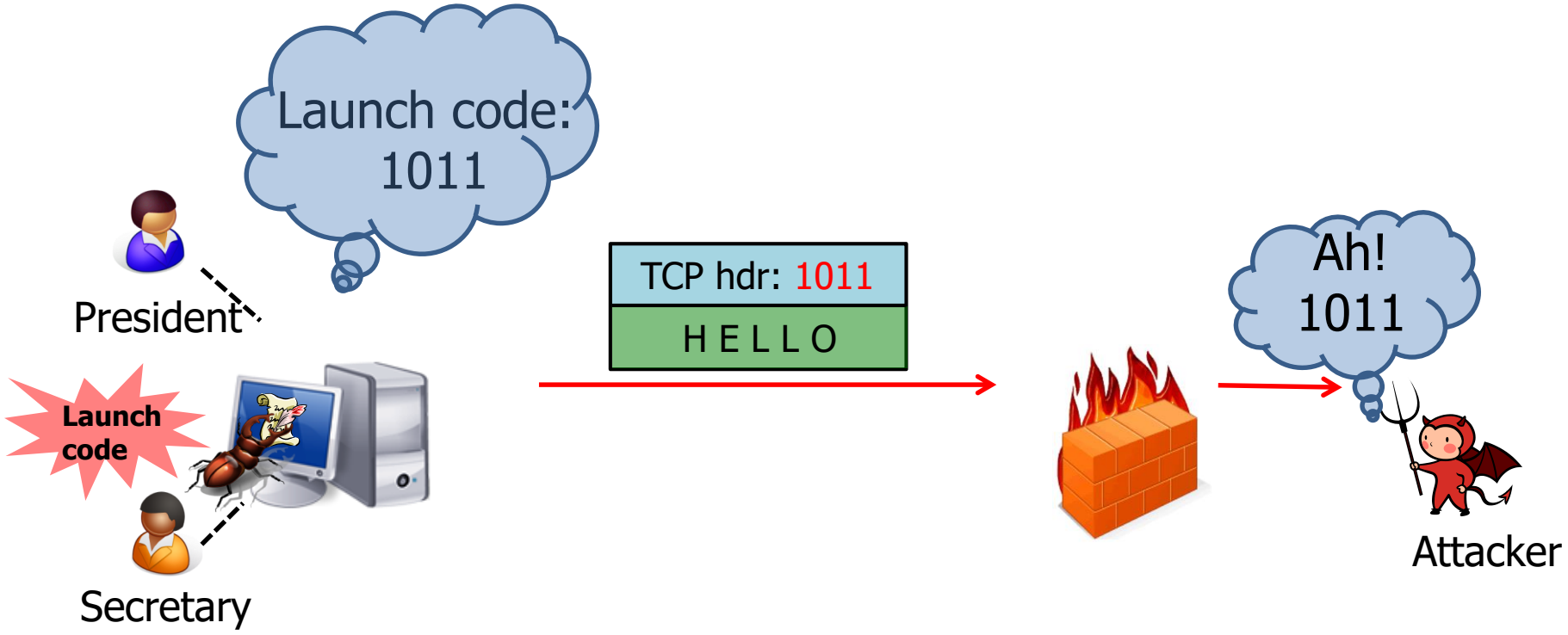
NetWarden: Mitigating Network Covert Channels without Performance Loss

Jiarong Xing, Adam Morrison, Ang Chen
Rice University

Motivation: Mitigating network covert channels

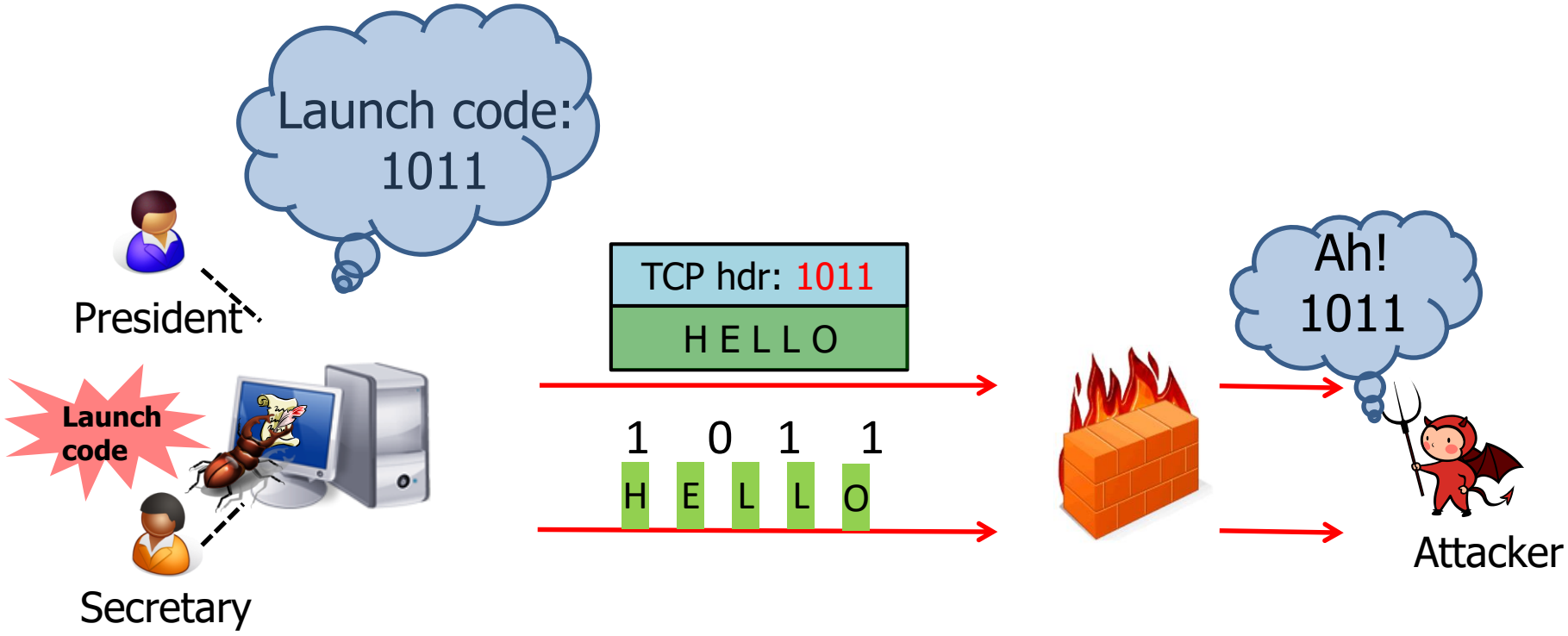


Motivation: Mitigating network covert channels



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

Motivation: Mitigating network covert channels

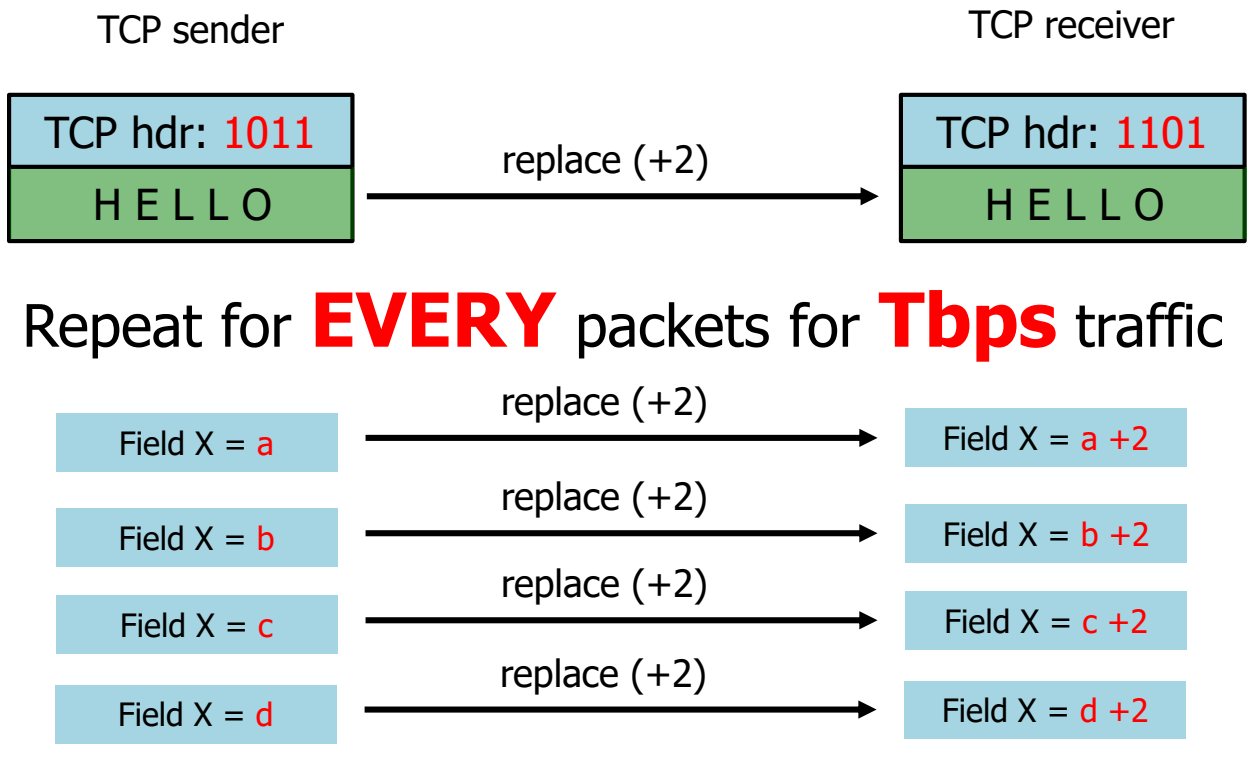


- Covert channels:
  - Storage channels: changing the packet **header fields**.
  - Timing channels: changing the **timing** of packets.

State of the art: Existing channels

- **Covert storage channels:**
  - TCP initial sequence number channel (Rowland, 1997)
  - IP TTL channel (Qu, 2004)
  - Partial ACK channel (Luo, 2009)
  - **…**
- **Covert timing channels:**
  - IP timing channel (Cabuk, 2004)
  - TCP timing channel (Luo, 2008)
  - Physical layer channel (Lee, 2014)
  - **…**
- Covert channels can leak data over long distance effectively
- "Common Criteria" require protection against both channel types!

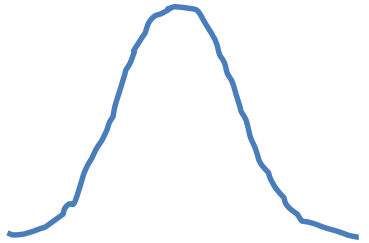
State of the art: Storage channel defense



- 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!

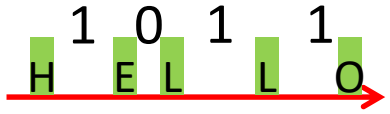
State of the art: Timing channel detection

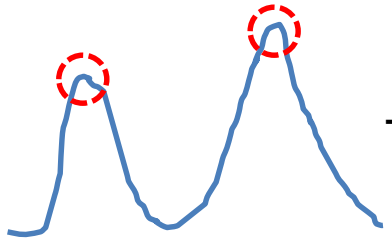
Normal traffic:



One peak

With channel:



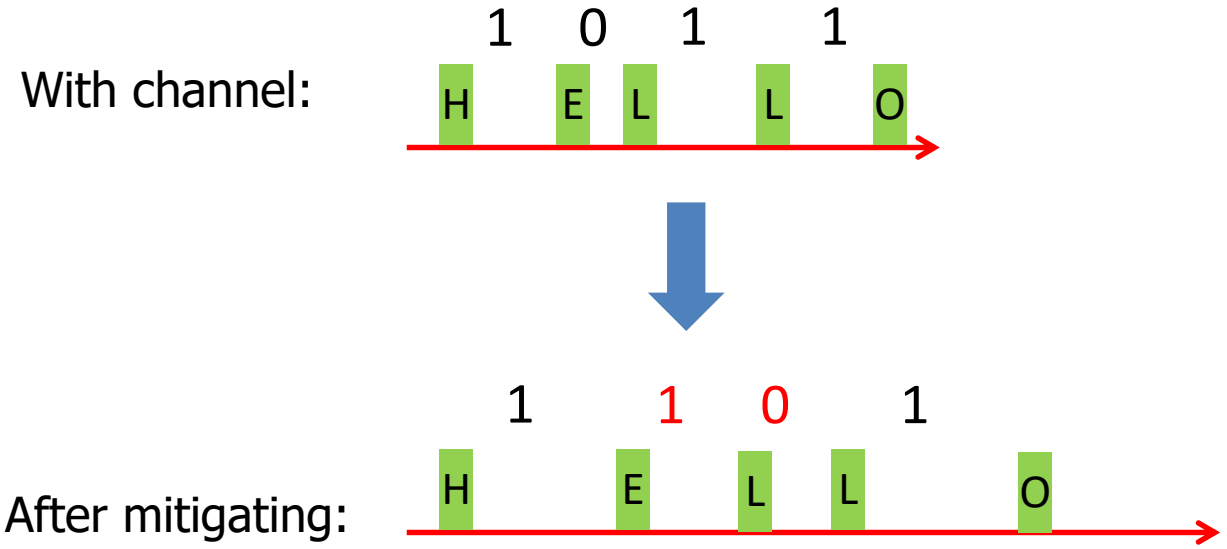


Two peaks

Distribution of packet gaps

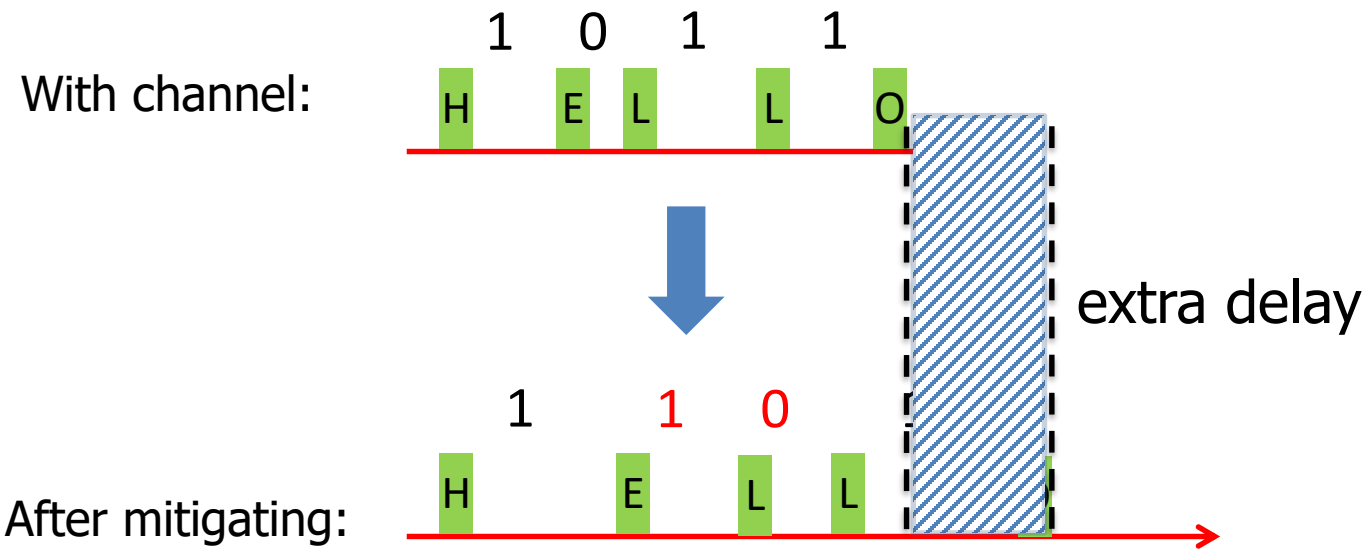
- Detection: Statistics-based tests over packet gaps
  - Looking for signs of statistical deviation
  - → Not always accurate

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

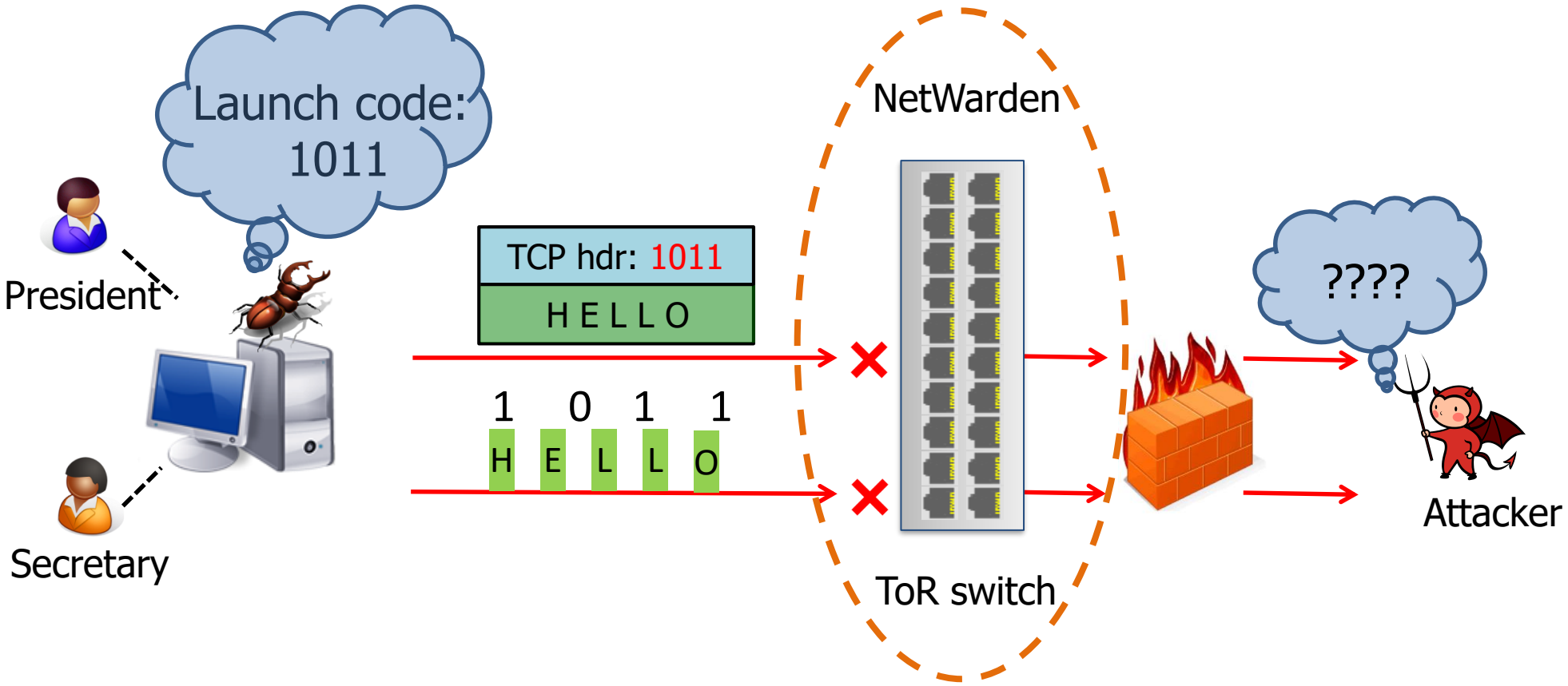
- Detection:
  - Per-packet inspection required
  - Software cannot keep up with Tbps traffic

- 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.

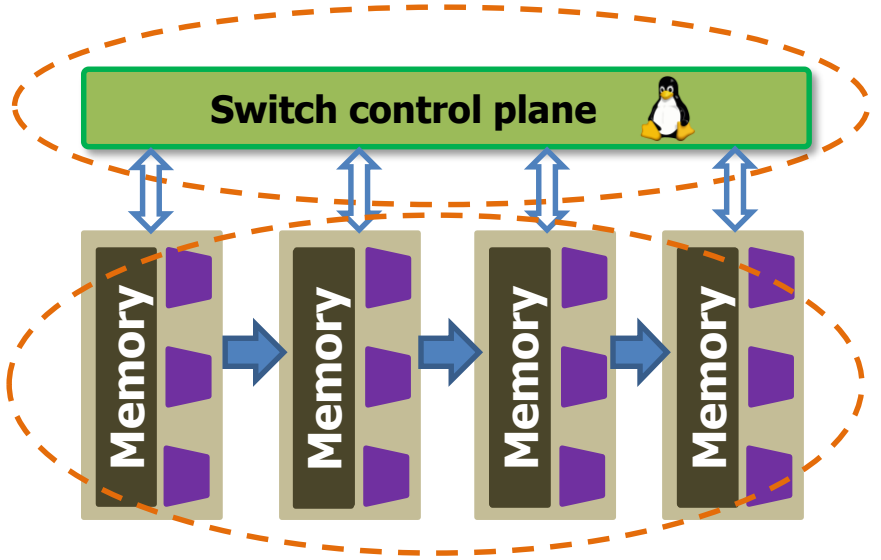
Key challenges and solutions

- **Challenge #1:** Efficient detection
  - **Solution:** Use programmable switches
- **Challenge #2:** Performance-preserving mitigation
  - **Solution:** Performance "boosting"
- **Challenge #3:** Hardware limitations
  - **Solution:** Fastpath/slowpath co-design

Outline

- ✓ Motivation: Mitigating network covert channels
- ✓ State of the art: Performance penalty
- ✓ Approach: NetWarden
- NetWarden design
  - Challenge #1: Efficient detection
  - Challenge #2: Performance-preserving mitigation
  - Challenge #3: Hardware limitations
- Initial validation
- Ongoing work
- Conclusion

Challenge #1: Efficient detection



- Problem: Software-based detection cannot handle Tbps traffic.
- Solution: Detecting covert channels on programmable switches.
- Programmable switches have two layers:
  - **Control plane:** General purpose CPU.
  - **Data plane:** Programmable ASIC.

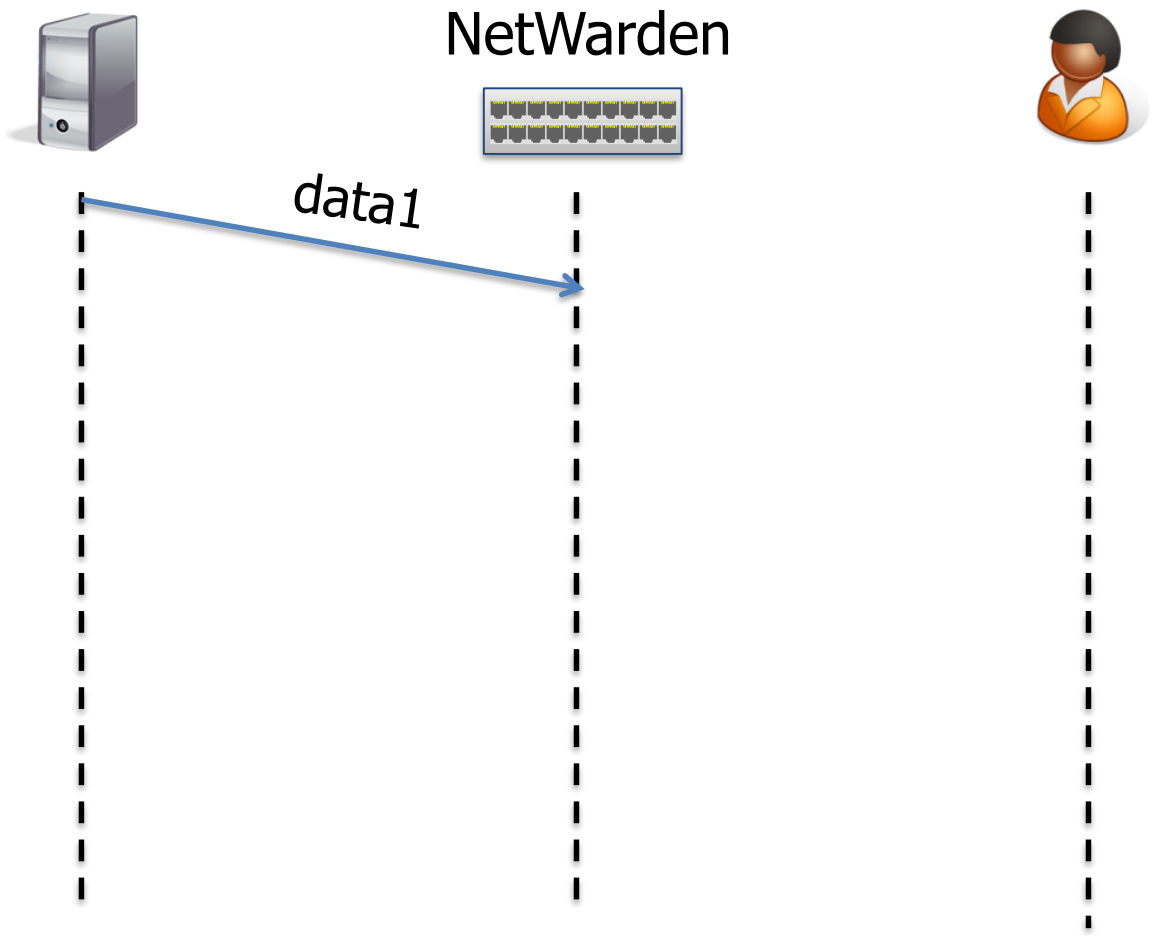
Challenge #2: Performance-preserving mitigation

- Problem: Existing mitigations incur performance loss.
- 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.

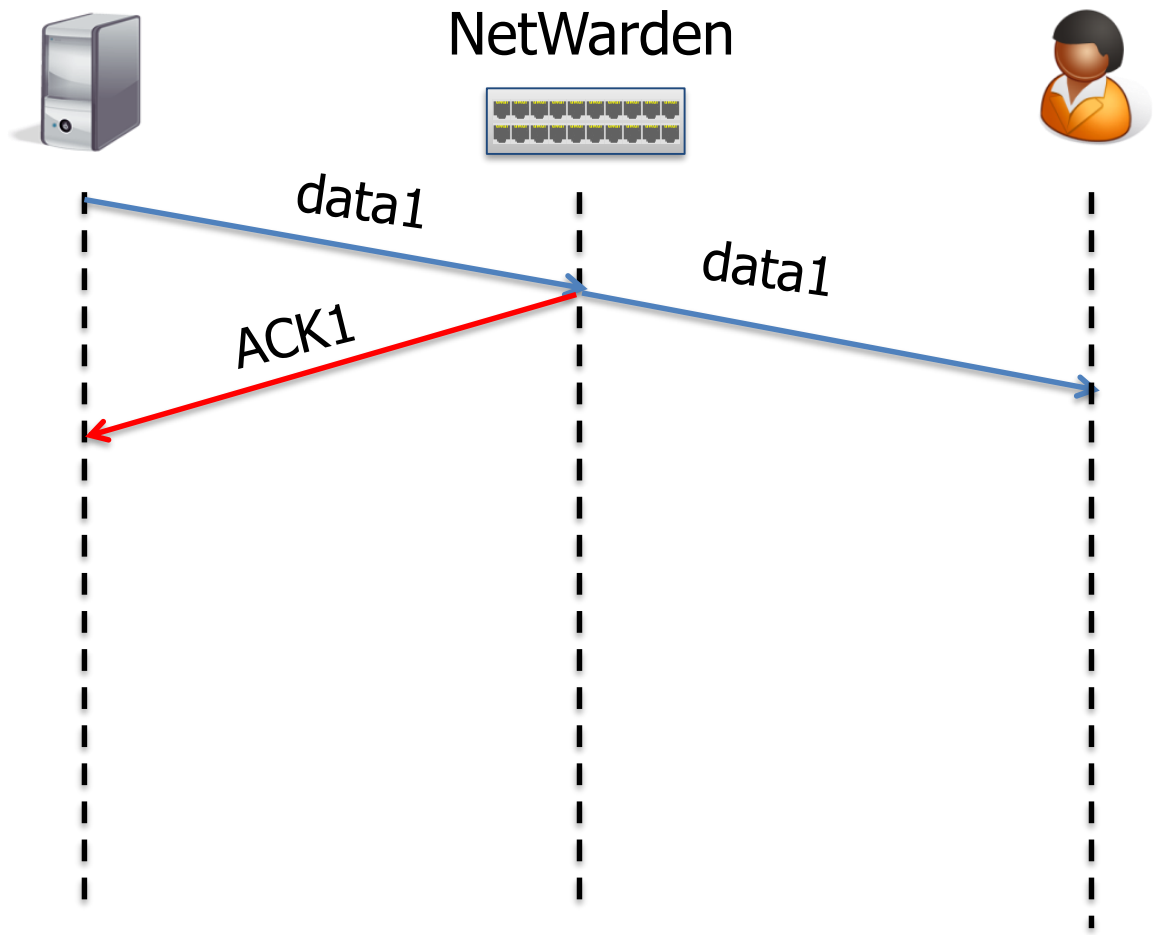
Challenge #2: Performance-preserving mitigation

- Problem: Existing mitigations incur performance loss.
- 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.

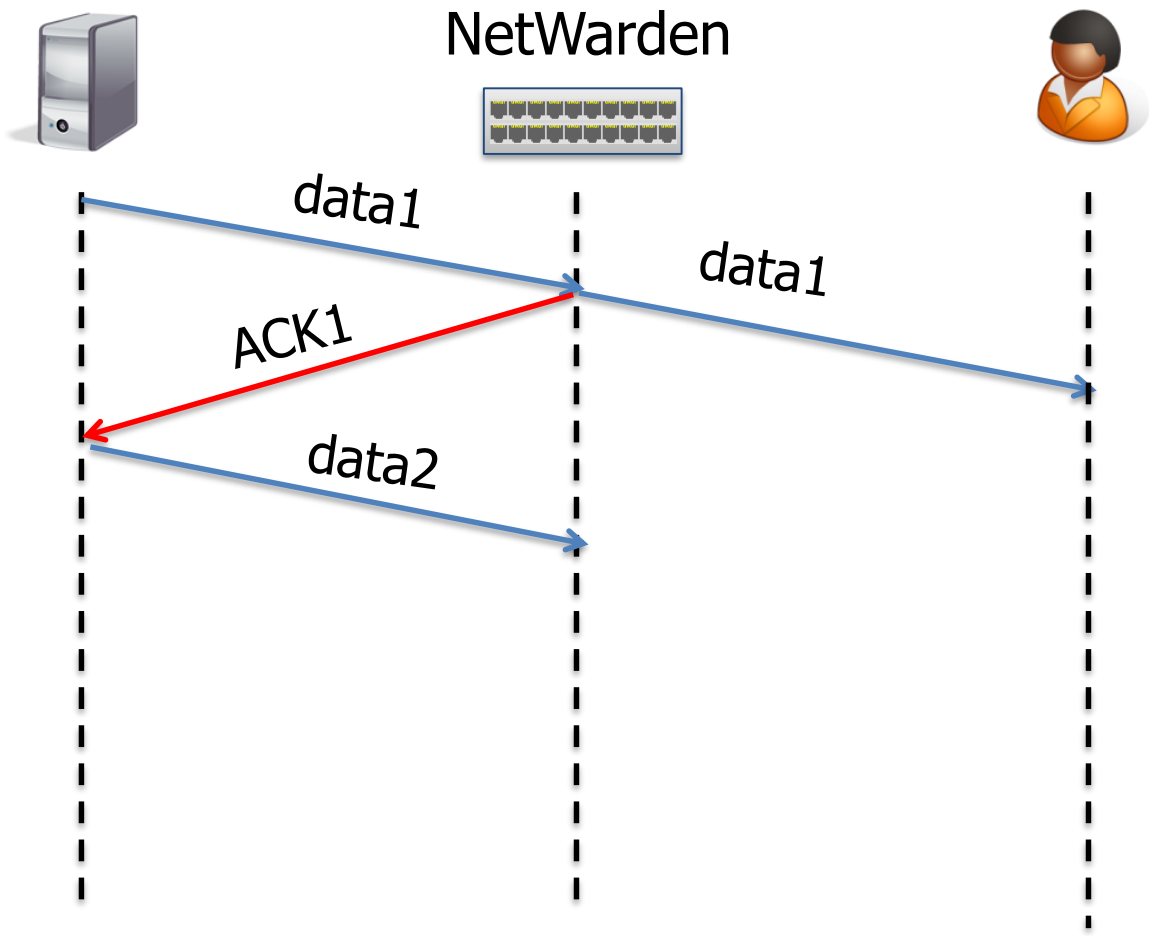
Boosting performance: ACK booster



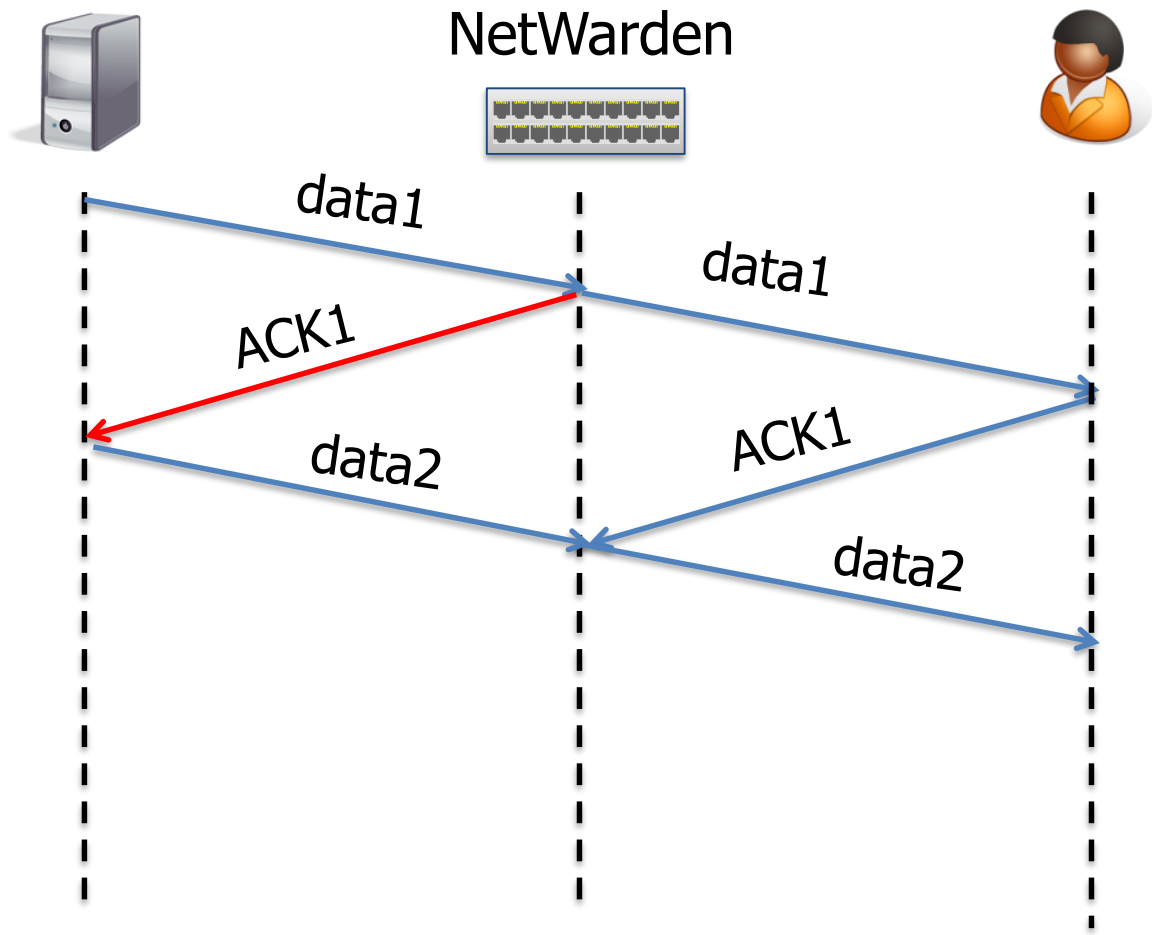
Boosting performance: ACK booster



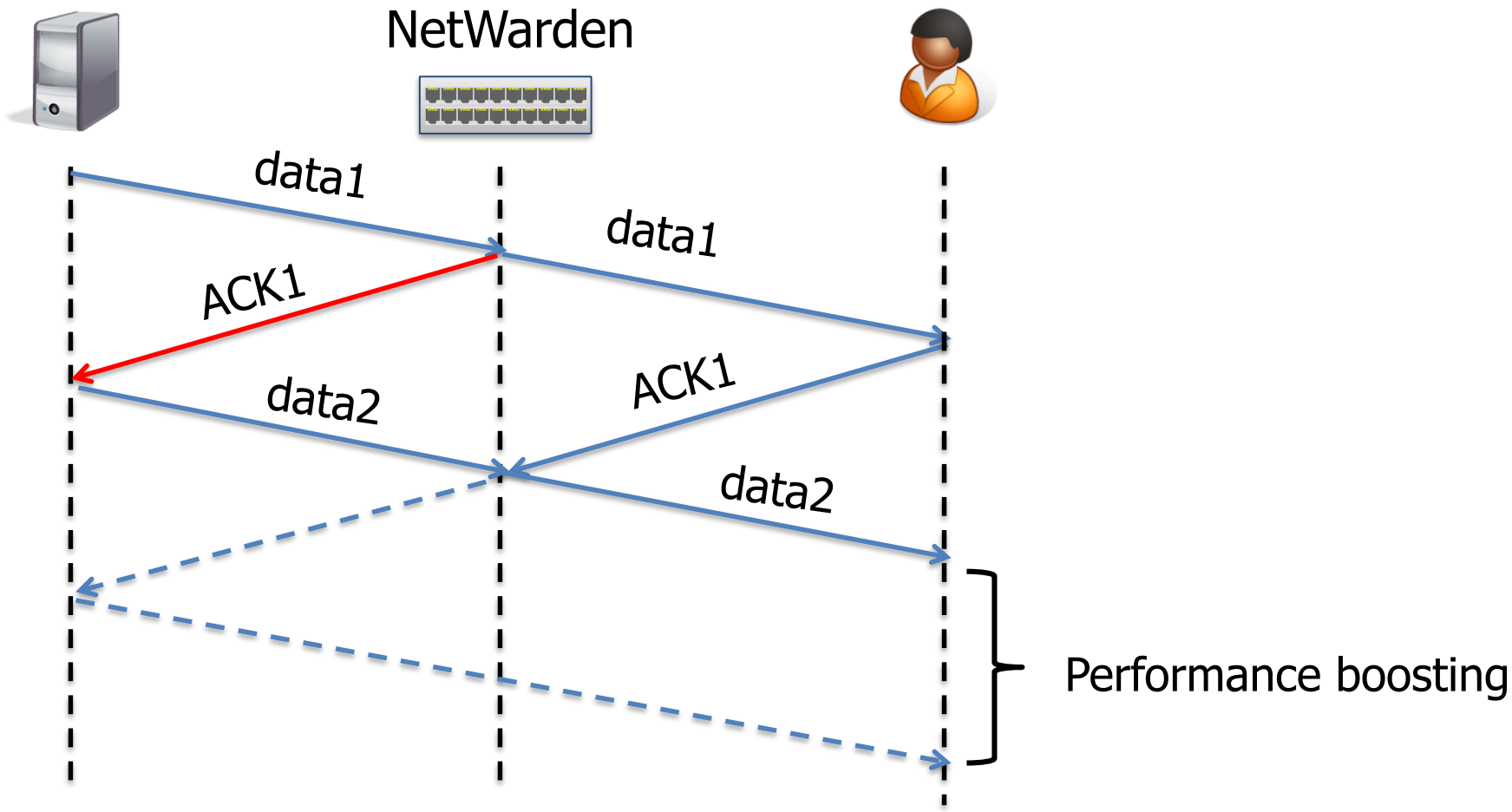
Boosting performance: ACK booster



Boosting performance: ACK booster

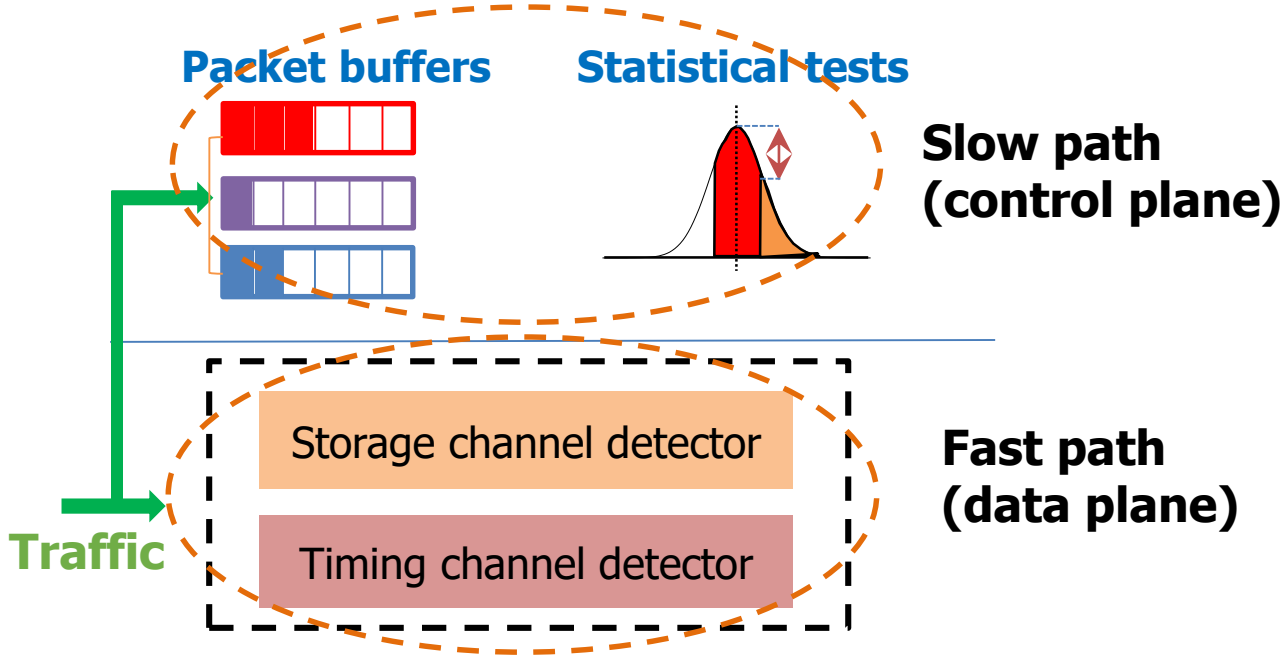


Boosting performance: ACK booster



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

Challenge #3: Hardware limitations



- Problem: The data plane has hardware restrictions
  - E.g., does not support packet caching
  - Or complex statistical tests
- Solution: Fastpath/slowpath co-design

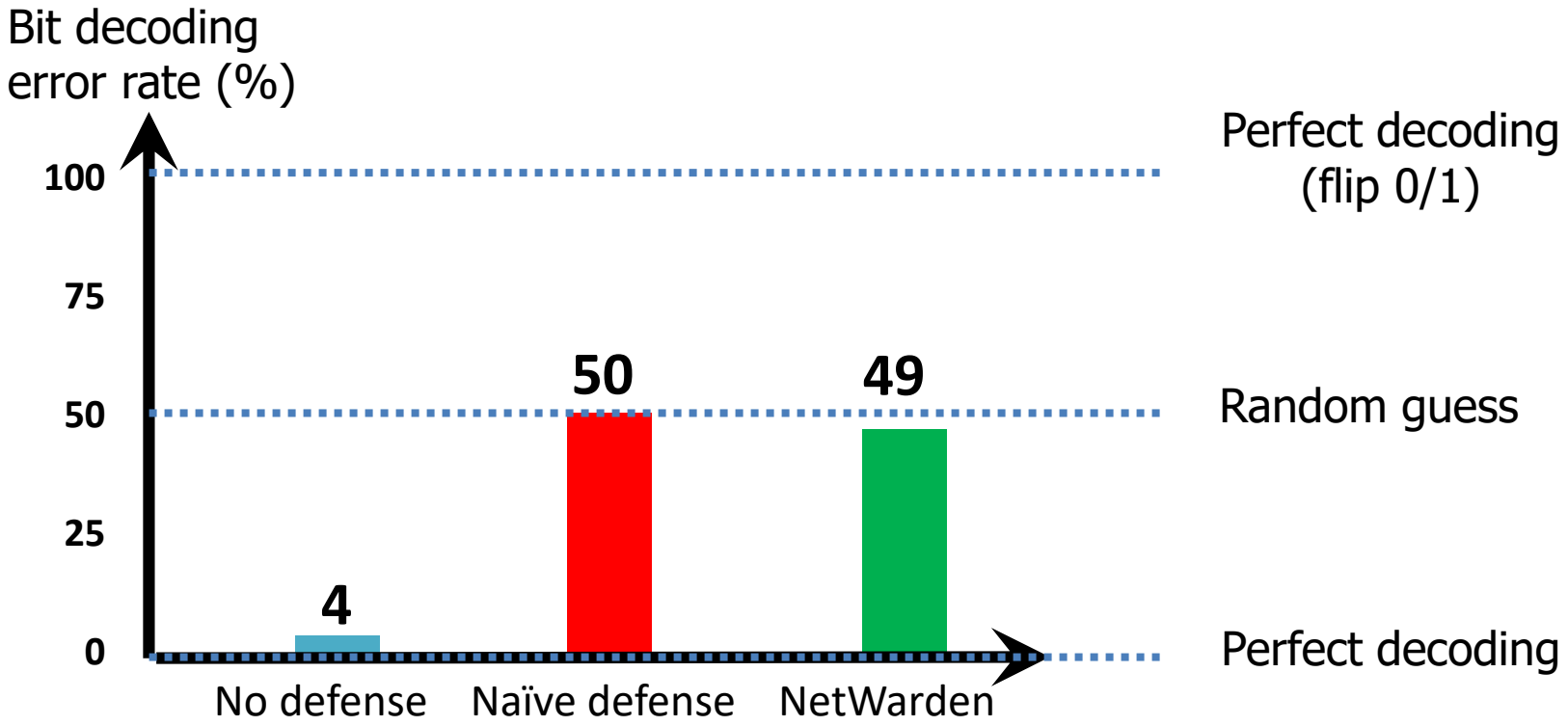
Outline

- ✓ - Motivation: Mitigating network covert channels
- ✓ - State of the art: Performance penalty
- ✓ - Approach: NetWarden
- ✓ - NetWarden design
  - ✓ - Challenge #1: Efficient detection
  - ✓ - Challenge #2: Performance-preserving mitigation
  - ✓ - Challenge #3: Hardware limitations
- - Initial validation
- - Ongoing work
- - Conclusion

Experimental setup

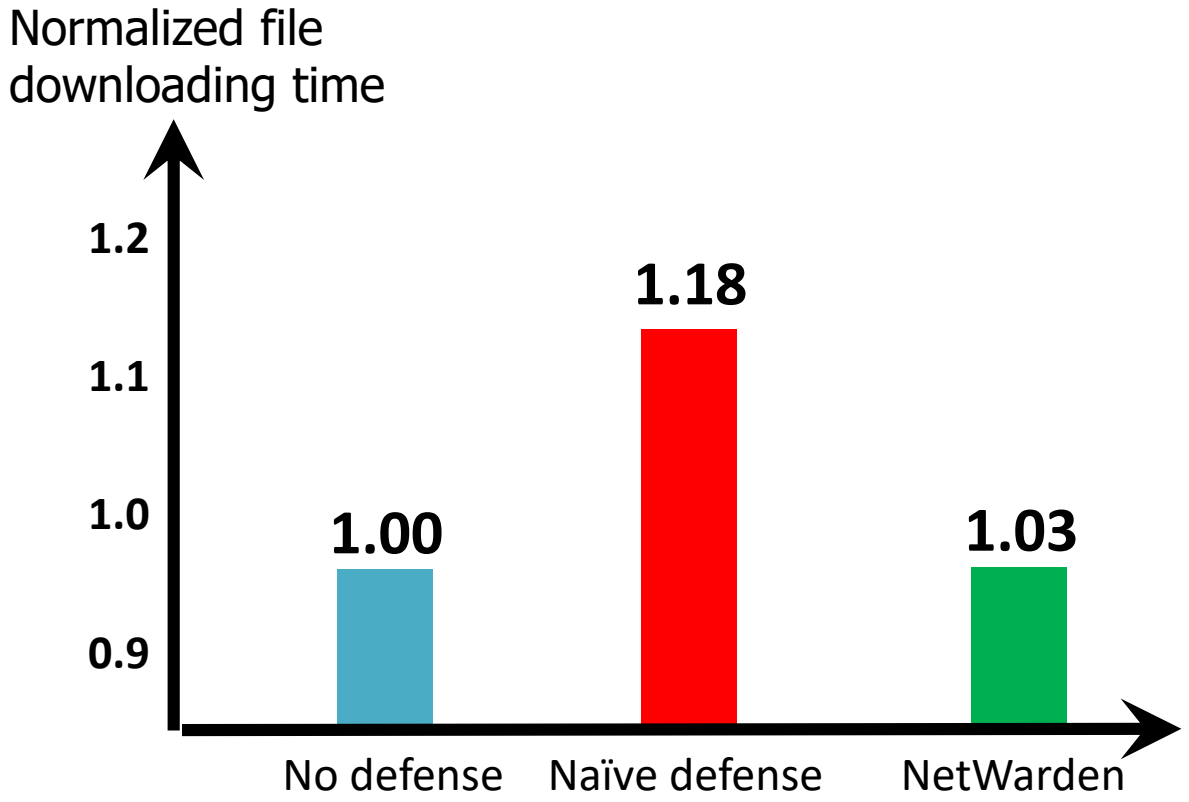
- Proof-of-concept prototype:
  - P4 for fastpath
  - Python for slowpath
  - Runs in software-based simulator.
- Two covert channels:
  - Channel #1: TCP storage channel
  - Channel #2: IP timing channel
- Scenario: Client downloads file from NetWarden-protected server
- Baseline: A naïve defense without performance boosting

Results: Effectiveness



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

Results: Performance



- Naïve defense incurs 18% extra downloading time.
- NetWarden only incurs 3% extra downloading time.
- NetWarden can mitigate covert channels with minimal performance loss.

Ongoing work

- How should we divide the labor between fastpath and slowpath?
  - "Optimal" division of labor
- How much performance boosting should NetWarden perform?
  - Too much → unfair to other connections
  - Too little → cannot neutralize performance loss
- How can we make NetWarden effective for all TCP variants?

Conclusion

- Motivation: Mitigating network covert channels
- Key limitation of existing approaches:
  - **Performance penalty**
- Our approach: NetWarden
  - Using programmable data planes
  - Performance boosting
  - Fastpath/slowpath design
- The goal of NetWarden:
  - **Mitigating covert channels without performance loss!**

Questions?