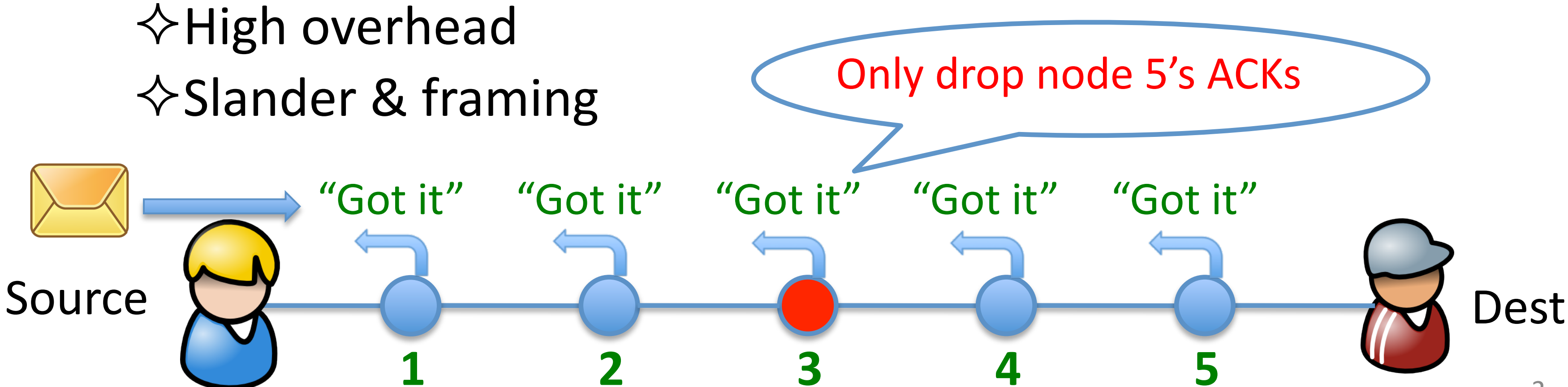
Carnegie Mellon CyLab
4616 HENRY STREET
PITTSBURGH, PA 15213
PH: (412) 268-7195
FX: (412) 268-7196
www.cylab.cmu.edu

ShortMAC:
Efficient Data-plane Fault Localization

Xin Zhang, Zongwei Zhou, Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim
Adrian Perrig and Patrick Tague

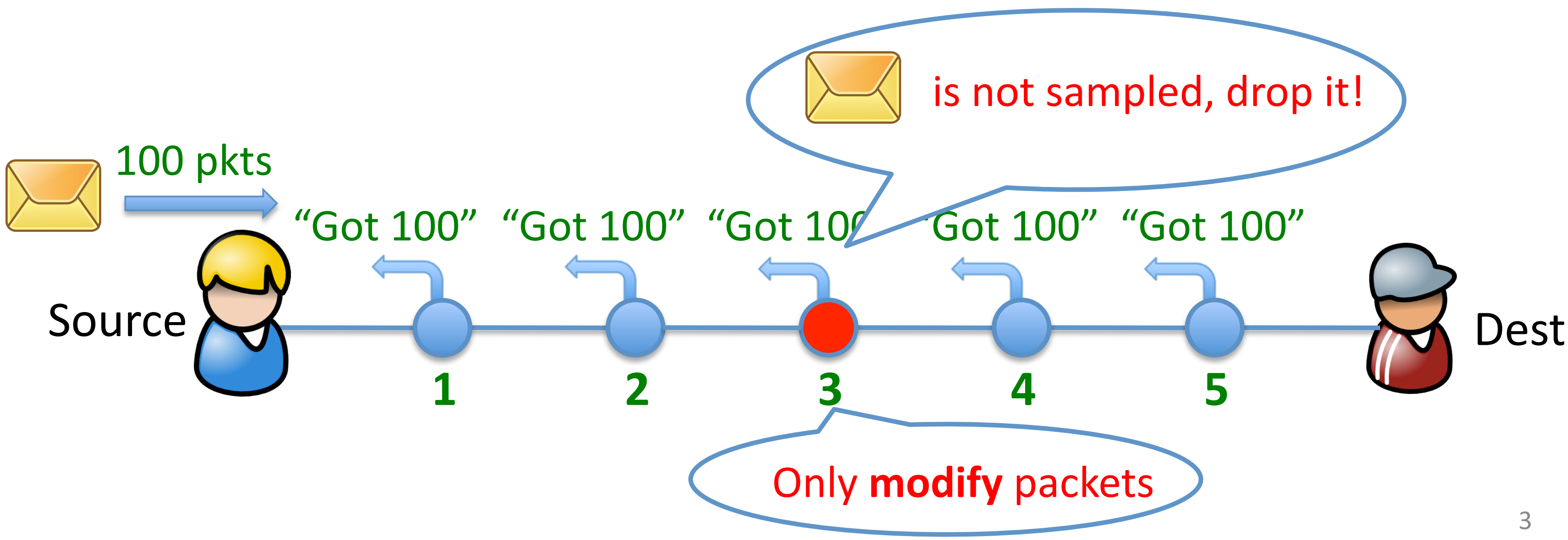
What is Fault Localization?

- Problem definition
  - Identify faulty links during packet forwarding
- Attacker Model
  - Drop, modify, misroute, or inject packets at data plane
- Challenges
  - Selective attack: break ping, traceroute, etc
  - High overhead
  - Slander & framing



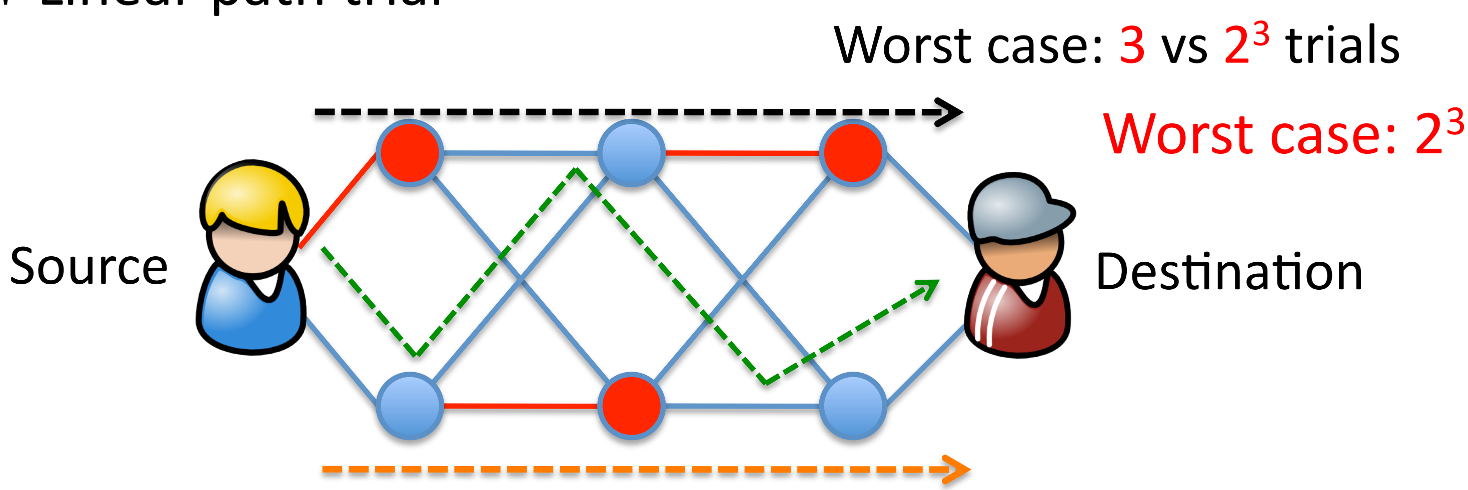
What is Fault Localization?

- Challenges (cont'd)
  - Attacks against sampling
  - Forgery attack: break Netflow, Bloom Filter, etc
  - Natural packet loss



Why is Fault Localization Important?

- The current Internet
  - Best effort, purely end-to-end
- Fault localization enables:
  - Data-plane accountability
  - Intelligent path selection
  - Linear path trial

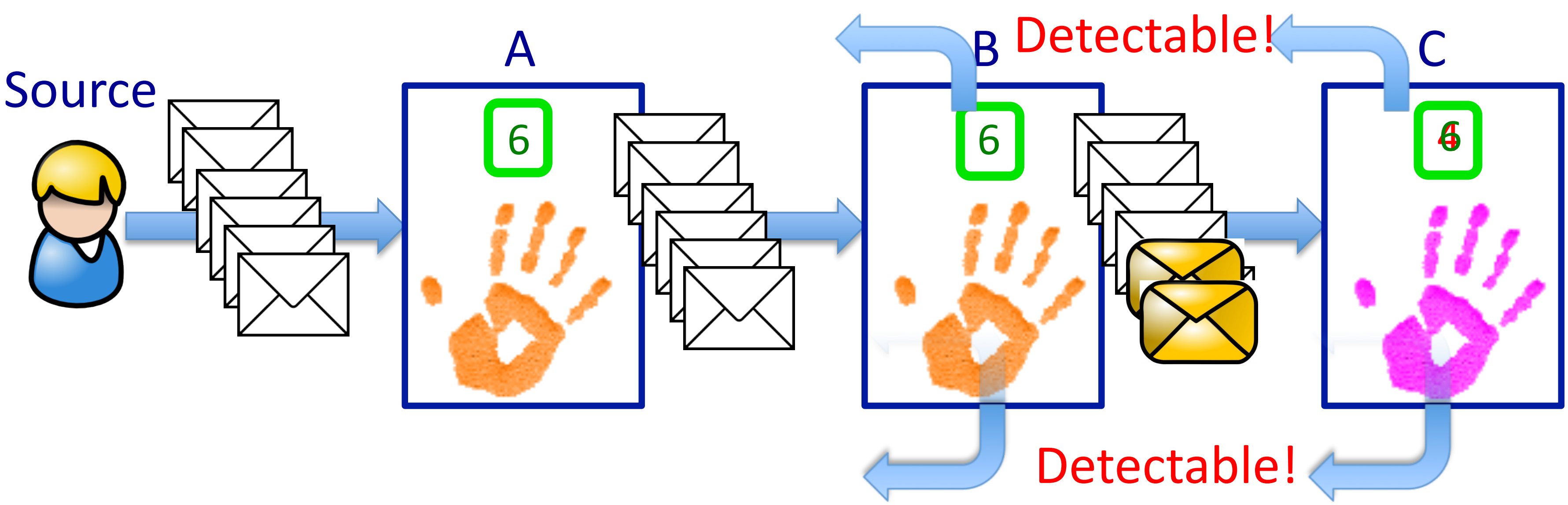


Design Goals

- Security
  - Against drop, modify, inject, and replay packets
  - Against multiple colluding nodes
- Efficiency
  - Low detection delay
  - Low storage, communication and computation overhead
- Provable guarantees
  - Upper bound of damage without being detected
  - Lower bound of forwarding correctness if no fault detected

ShortMAC Key Insight #1

- Fault Localization → Packet authentication
  - Fault Localization → monitor packet *count* and *content*
  - W/ pkt authen, content → count
  - Only counts → small state, low bandwidth cost



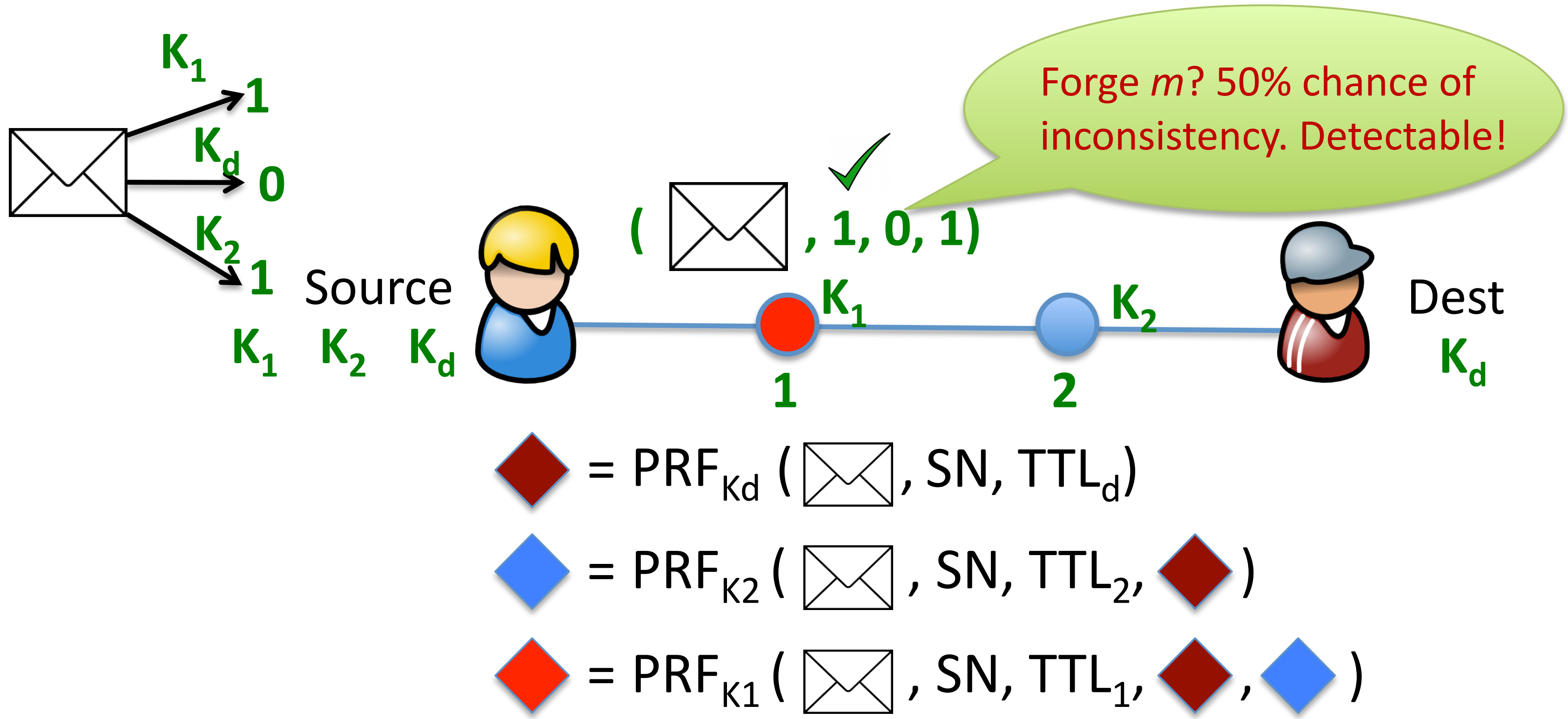
ShortMAC Key Insight #2

- *Limiting* attacks instead of perfect detection
  - Detect every misbehavior? Costly! Error-prone!
  - Absorb low-impact attack: tolerance threshold
  - Trap the attacker into a *dilemma*
  - Enable probabilistic algorithms with provable bounds

ShortMAC Key Ideas

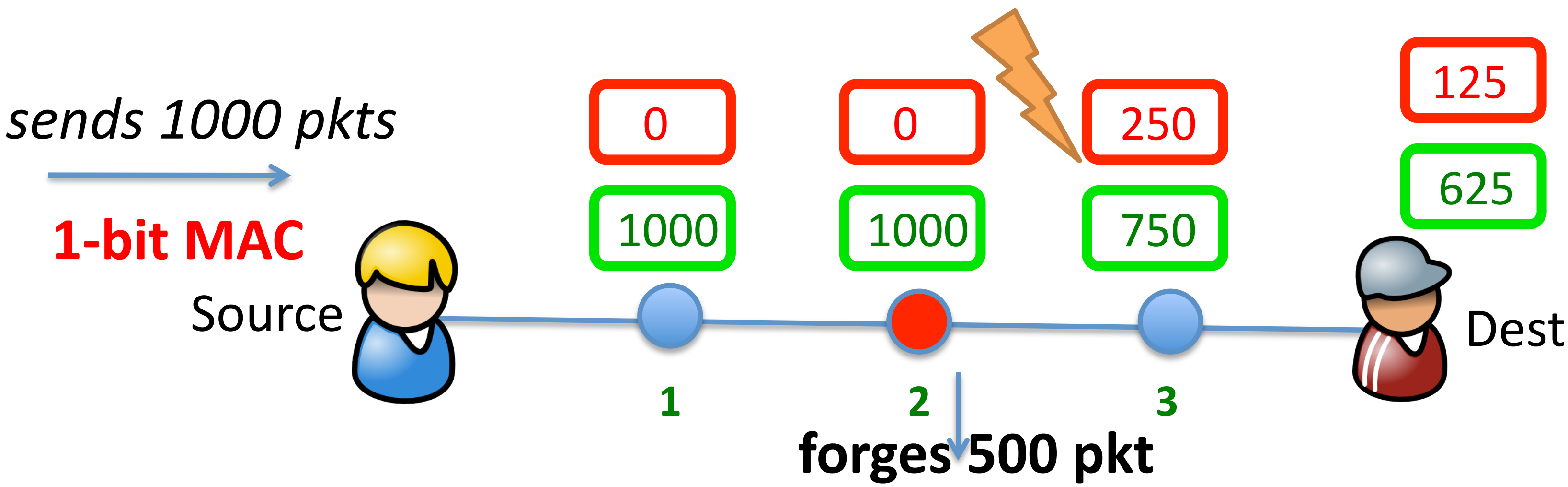
- The ShortMAC packet marking
  - Limiting instead of perfectly detecting fake packets
  - Source marks each packet with *k* bits (with keyed PRF)

***k*-bit MAC, e.g., *k* = 1**



ShortMAC Key Ideas

- High-level steps
  - Each node maintains two counters (*counter only*!)
  - *Secure* reporting
  - Threshold-based detection robust to *natural errors*



- More details: Onion ACK for reporting, threshold-based detection, etc

Theoretical Bounds

- The math

$$\alpha = 1 - (1 - T_{dr})^2 + \frac{\beta}{N(1-T_{dr})^d} \qquad \beta = \frac{T_{in}}{q} + \frac{\sqrt{\left(\ln\frac{2}{\delta}\right)^2 + 8qT_{in}\ln\frac{2}{\delta} + \ln\frac{2}{\delta}}}{4q^2}$$

$$\theta = (1 - T_{dr})^d - \frac{\beta}{N} \qquad N = \frac{\ln(\frac{2d}{\delta})}{2\left(T_{dr}-\rho\right)^2\left(1-T_{dr}\right)^d}$$

- The numbers

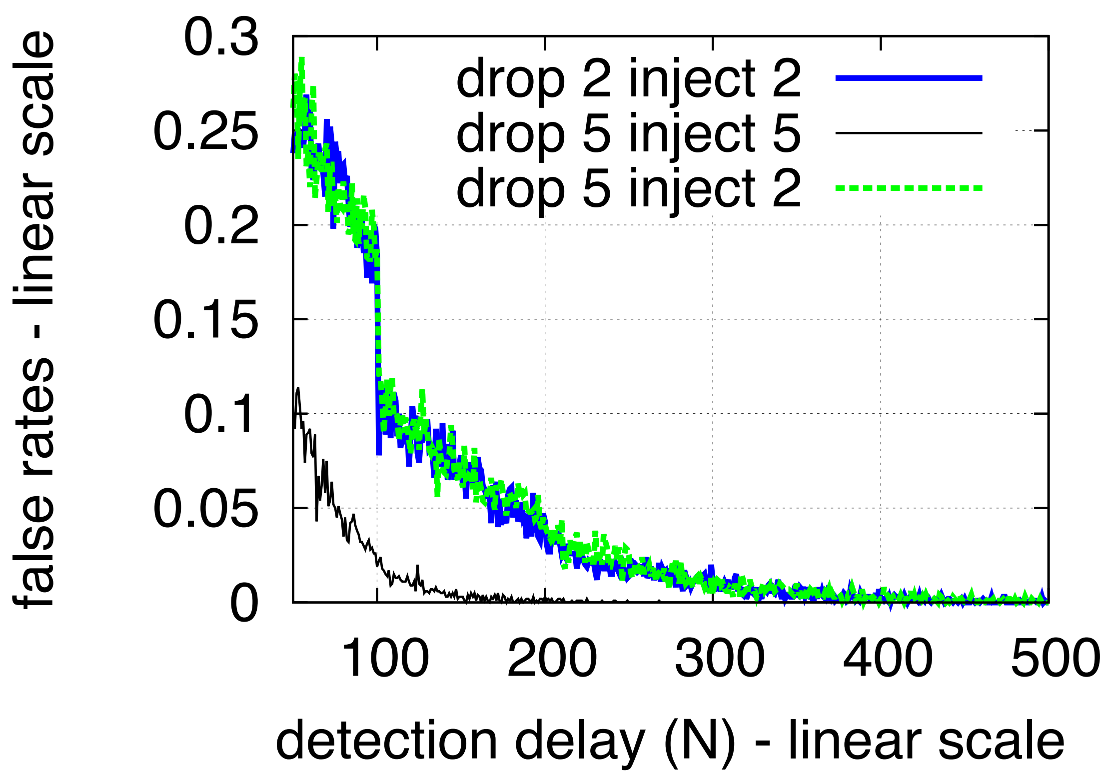
Protocol	ShortMAC	PAAI-1	SSS	Sketch
Delay (pkt)	3.8×10^4	7.1×10^5	1.6×10^8	$\approx10^6$
State (bytes)	21	2×10^5	4×10^3	≈500

Experimental Evaluation

- ❖ Average-case performance, proof of concept
- ❖ Simulation + Prototyping
  - ✧ Simulation: large-scale, security properties
  - ✧ Prototype: computational overhead
- ❖ SSF-net based simulation
  - ✧ Single 6-hop path
  - ✧ Malicious node in the middle
  - ✧ Independently dropping/injecting packets

Simulation Results

- False rates, detection delay, and comparison
  - 2-bit-MAC

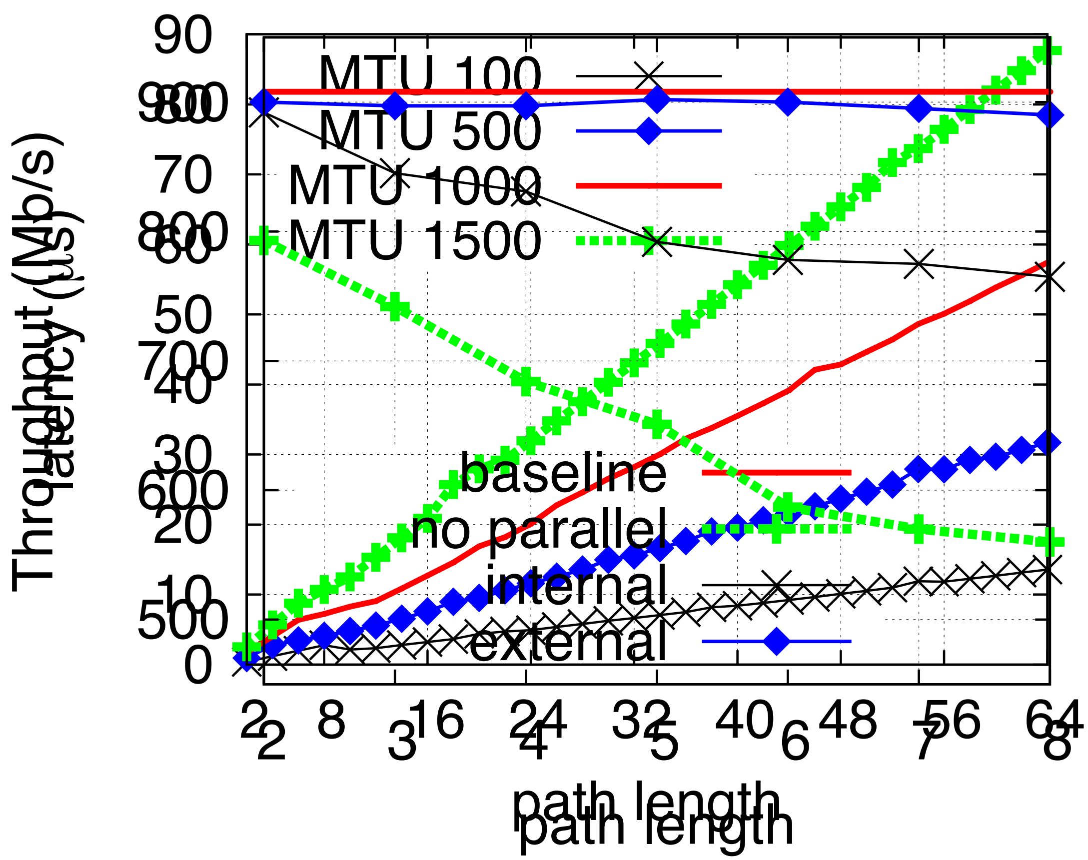


Prototyping Results

- Pure-software router prototype in Linux/Click
- Evaluation of fast path performance
  - Per-packet PRF computation
  - Different MACs with AES-ni
- Computational overhead
  - Throughput and latency
  - Linear path topology
  - Netperf benchmark

Prototyping Results

- Throughput and latency



Phew… the end

- *Limiting* instead of perfectly detecting
  - Enables efficient algorithms
- *Provable* security guarantee
  - Theoretical bounds, against strong adversaries
- *High* efficiency
  - Low detection delay, router state, comm. overhead
- *Probabilistic* packet authentication
  - Building block for other applications

Thank you!

Questions?

Xin Zhang <xzhang1@cmu.edu>