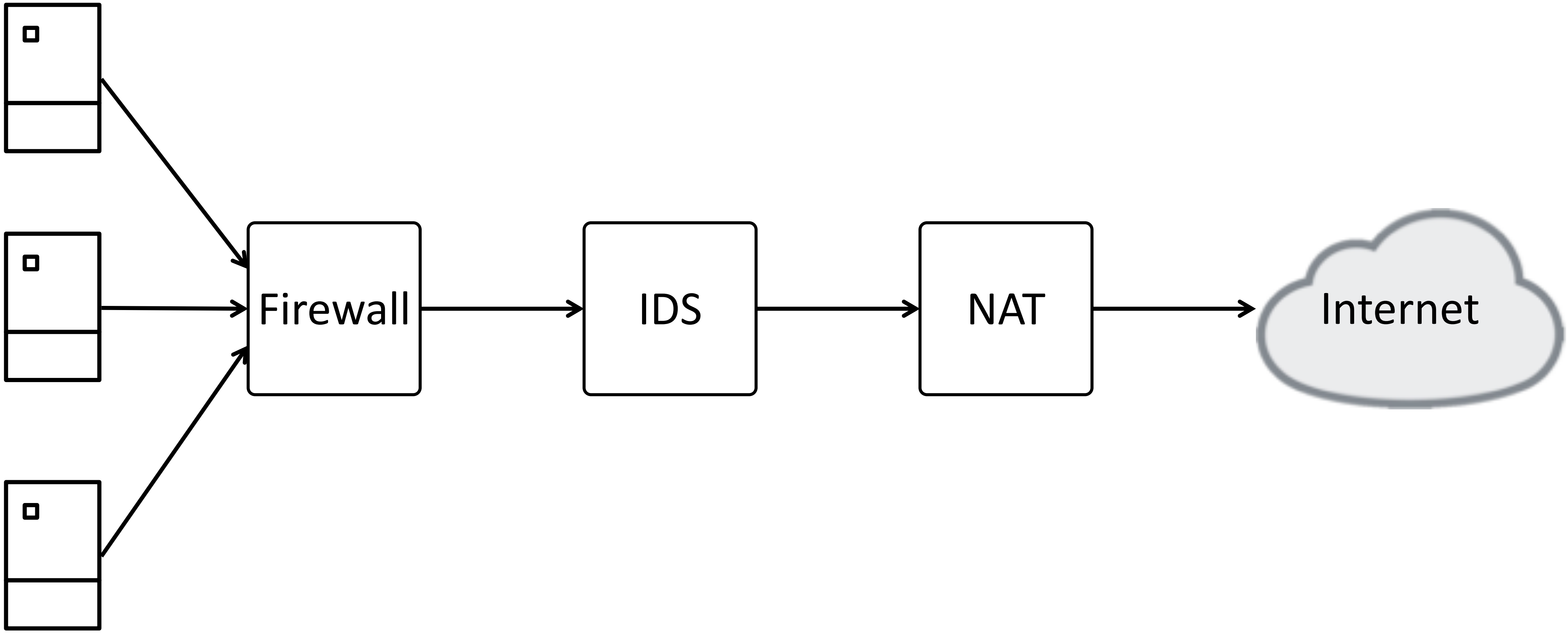
Fault Tolerant Service Function Chaining

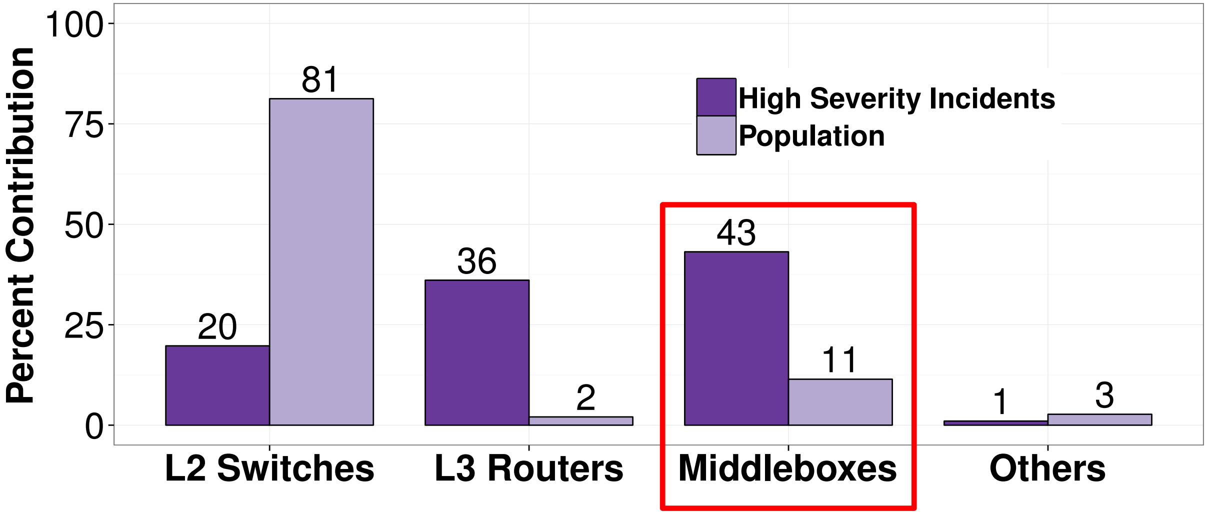
M. GHAZNAVI, E. JALALPOUR, B. WONG, R. BOUTABA, A. MASHTIZADEH

UNIVERSITY OF WATERLOO

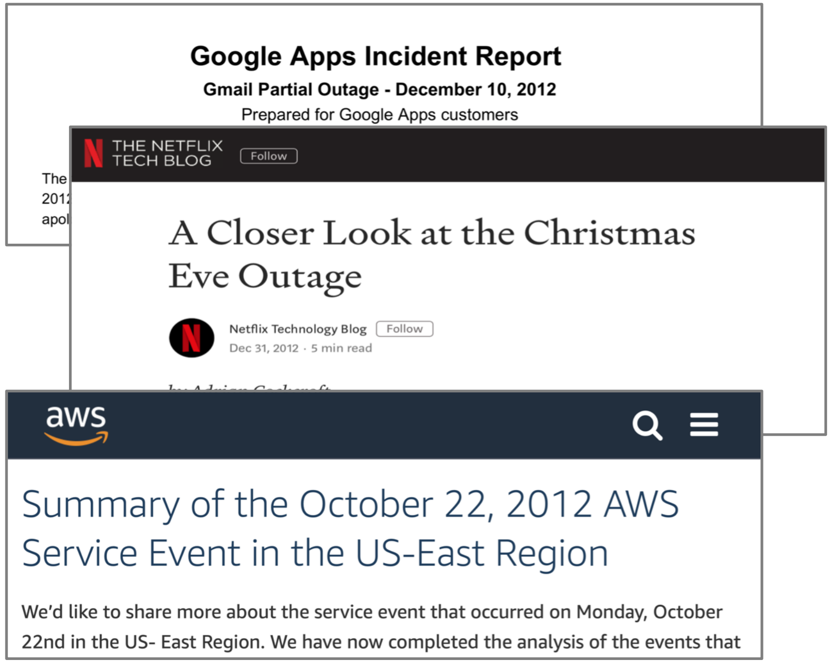
Middleboxes and Service Function Chains



Middlebox Failures



Demystifying the dark side of the middle:
A field study of middlebox failures in datacenters
IMC 2013



Google Apps Incident Report

Gmail Partial Outage - December 10, 2012

Prepared for Google Apps customers

The
2012
apol

A Closer Look at the Christmas Eve Outage

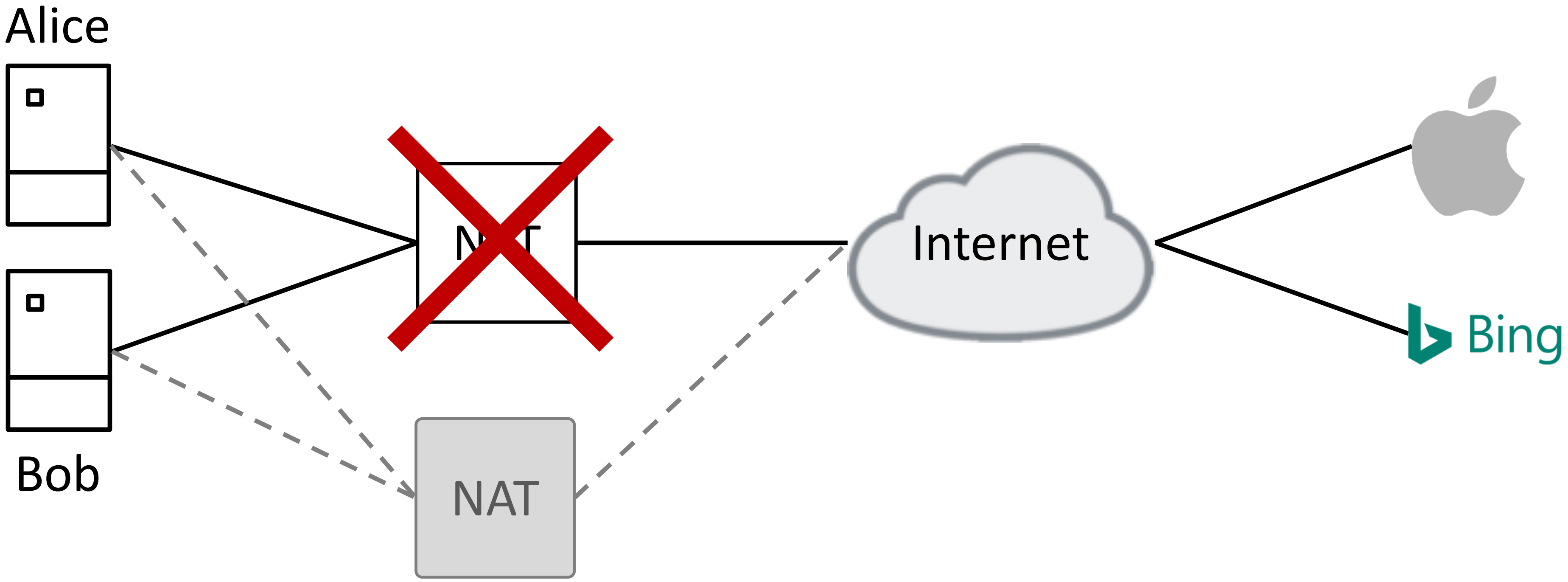
Dec 31, 2012 · 5 min read

by Adrian Cockcroft

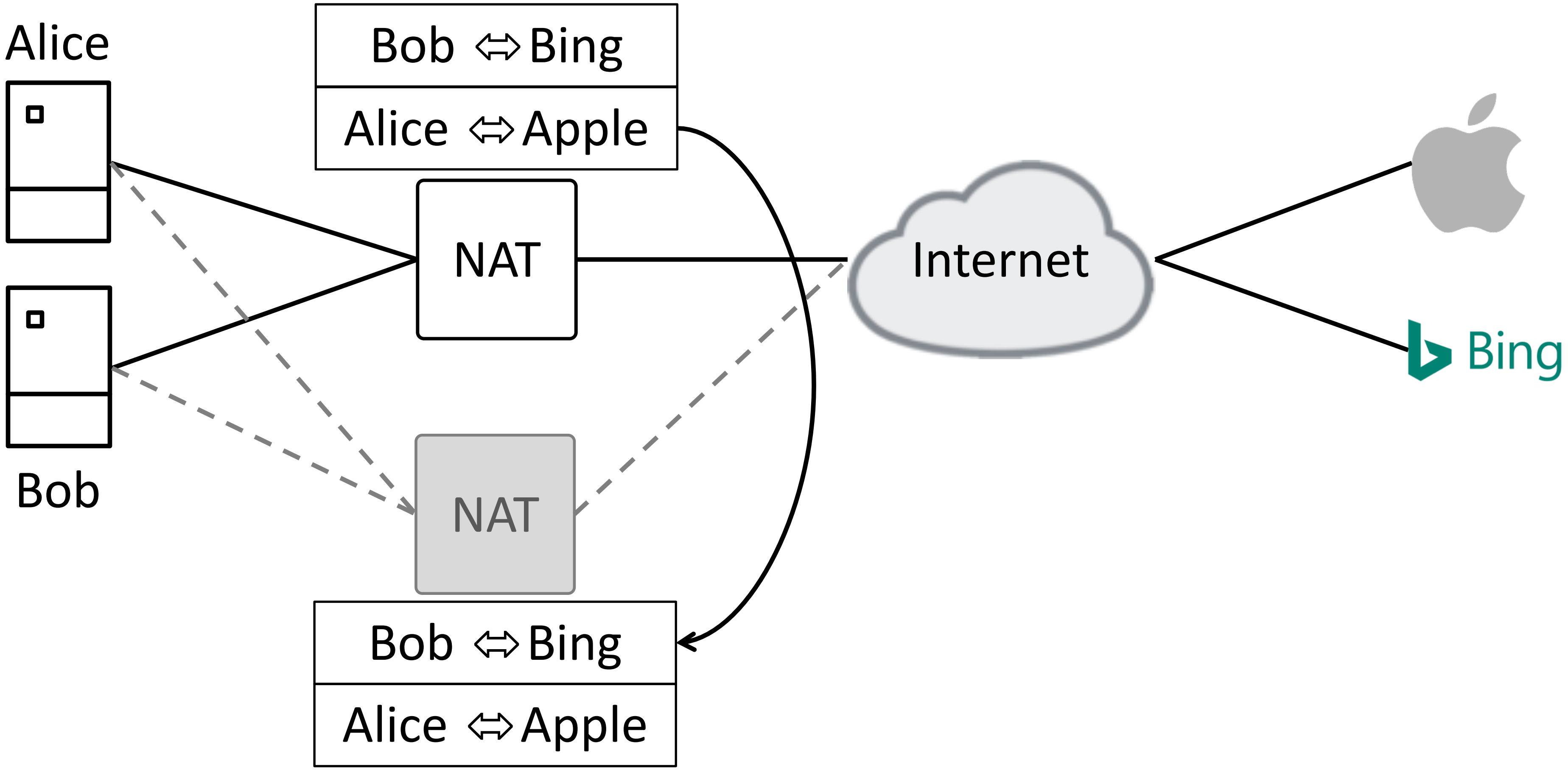
Summary of the October 22, 2012 AWS Service Event in the US-East Region

We'd like to share more about the service event that occurred on Monday, October 22nd in the US- East Region. We have now completed the analysis of the events that

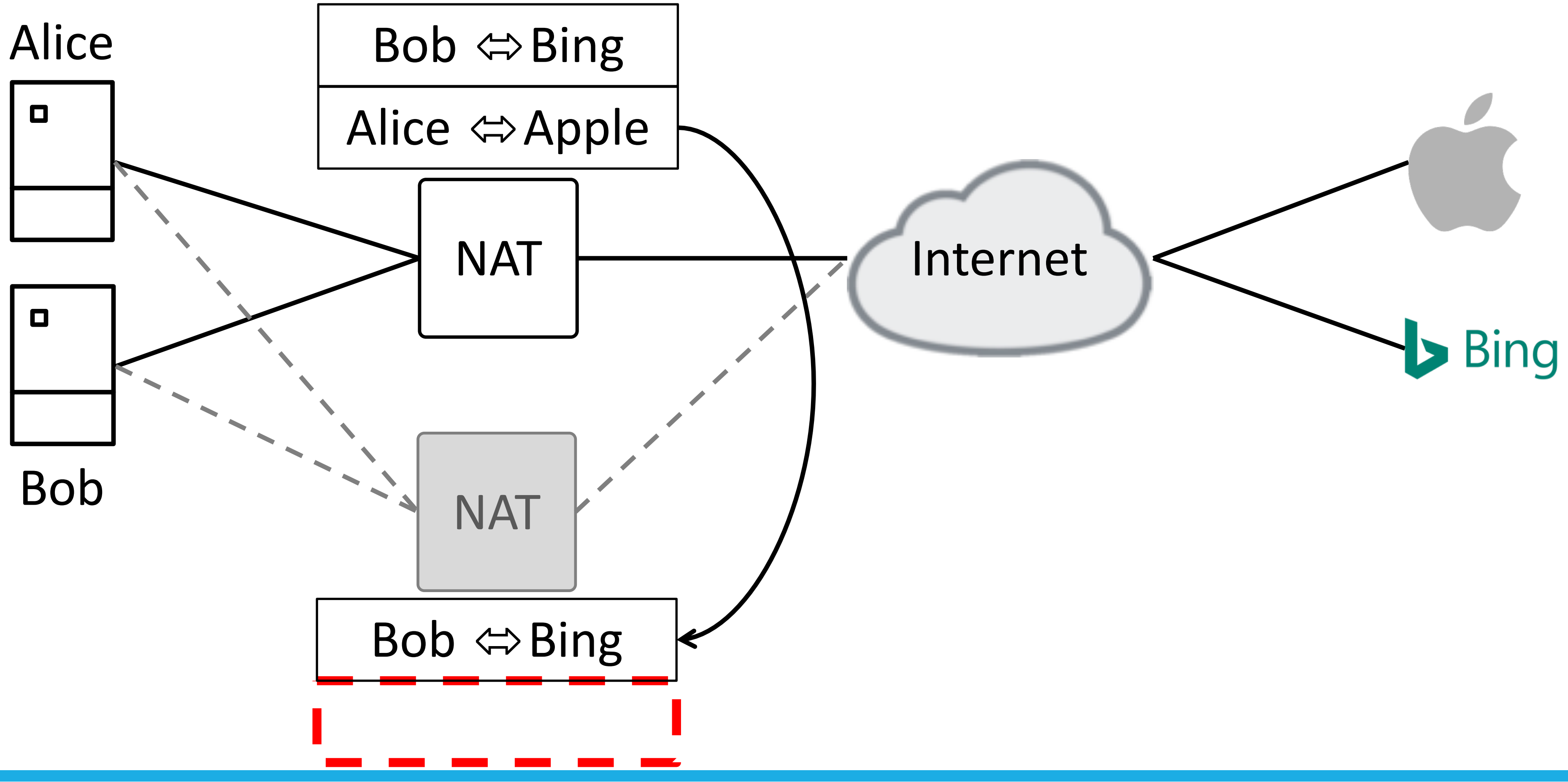
Middlebox Fault Tolerance



Consistent State Replication



Consistent State Replication



Previous Approaches

EXTERNALIZED STATE

StatelessNF, NSDI 2017

CHC, NSDI 2019

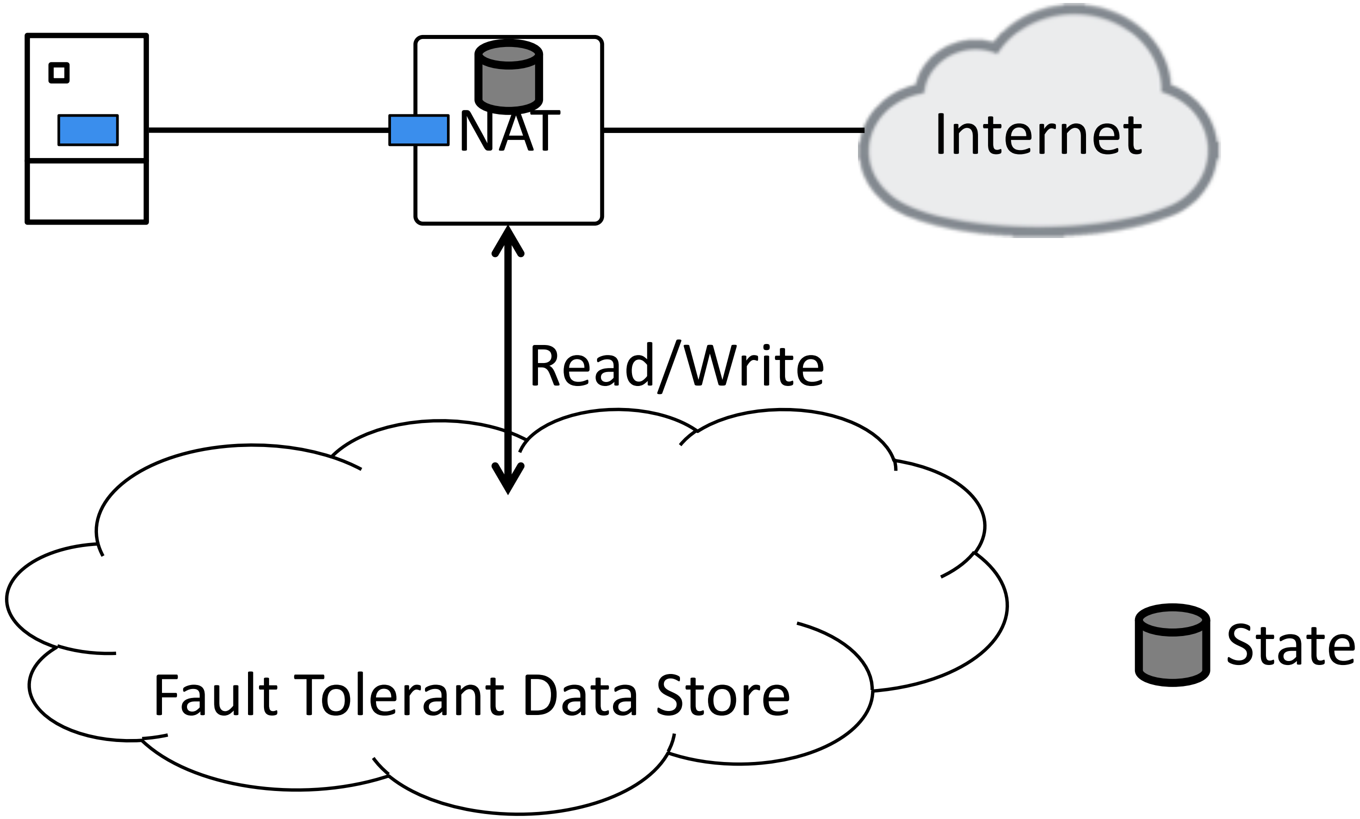
SNAPSHOT BASED

Pico Replication, SoCC 2013

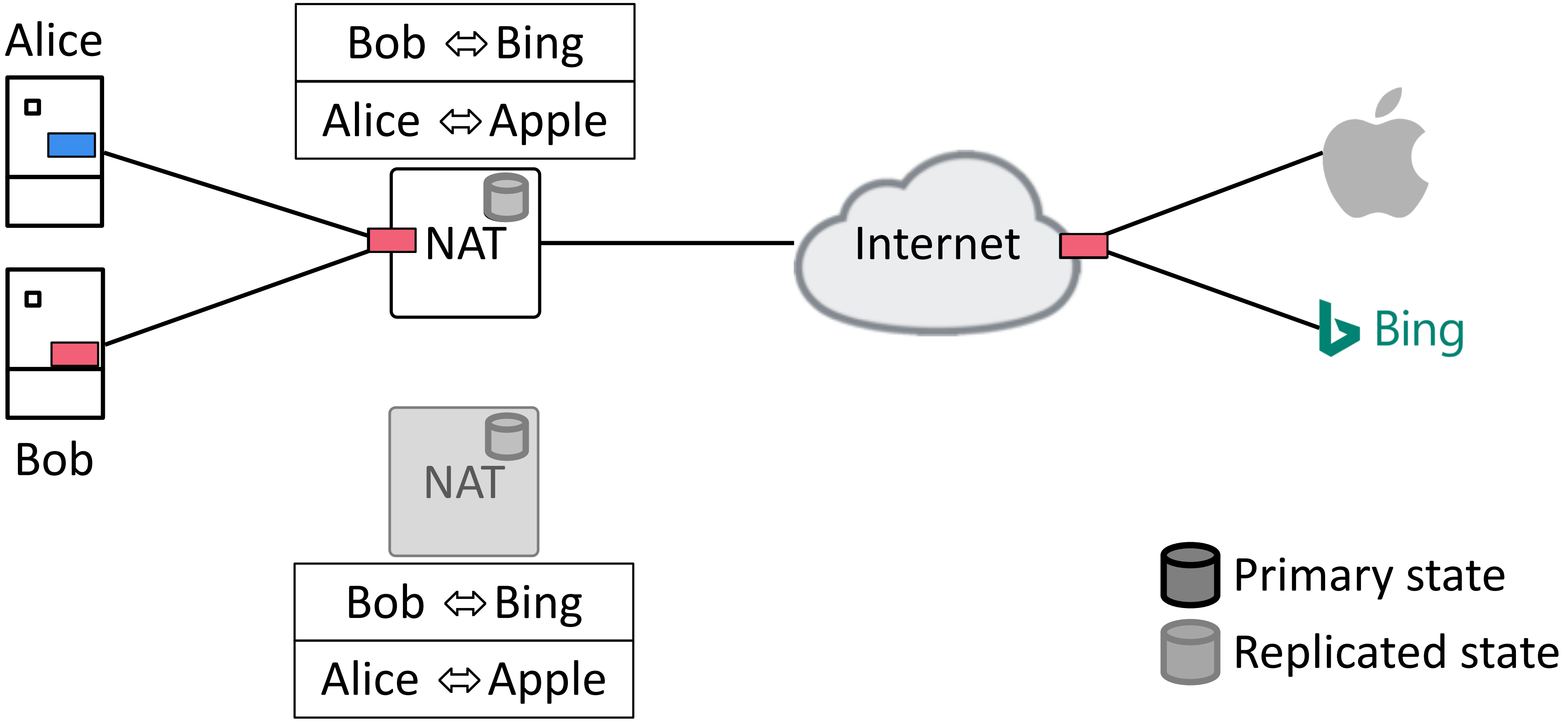
FTMB, SIGCOMM 2015

REINFORCE, CoNEXT 2018

Externalized State Approach



Snapshot Based Approaches



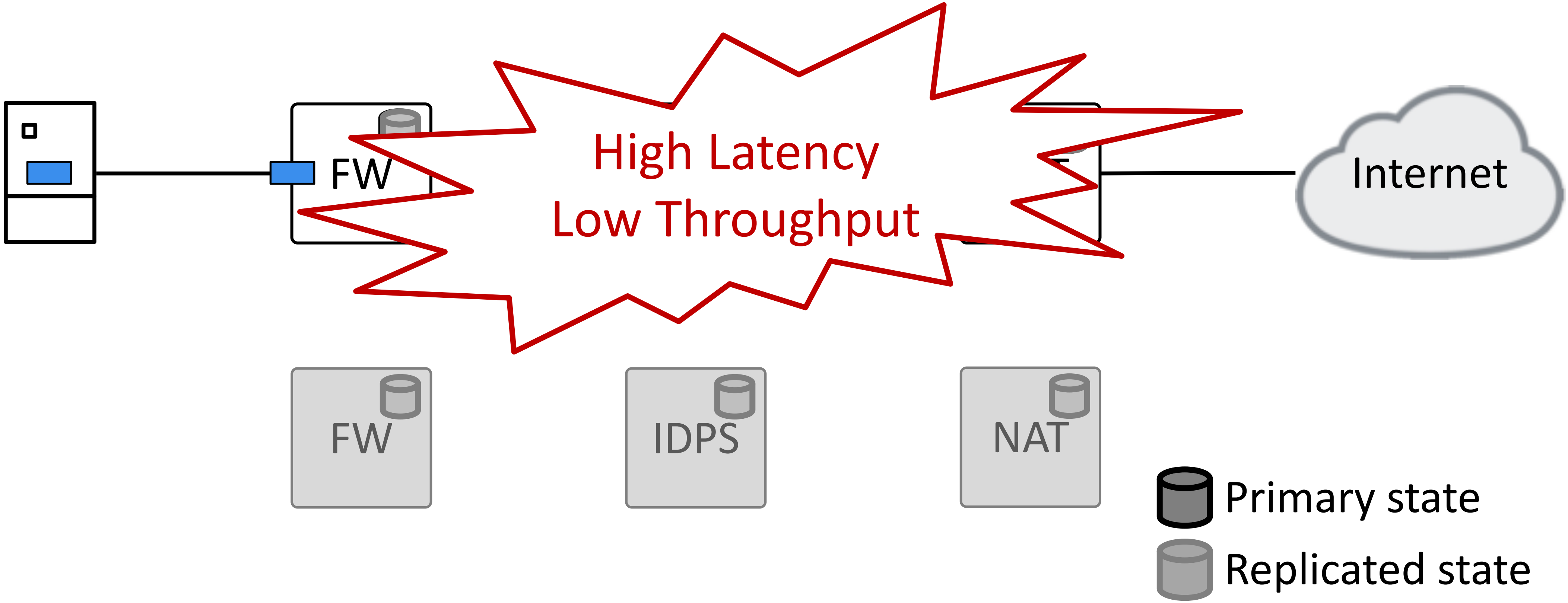
Bob ⇔ Bing

Alice ⇔ Apple

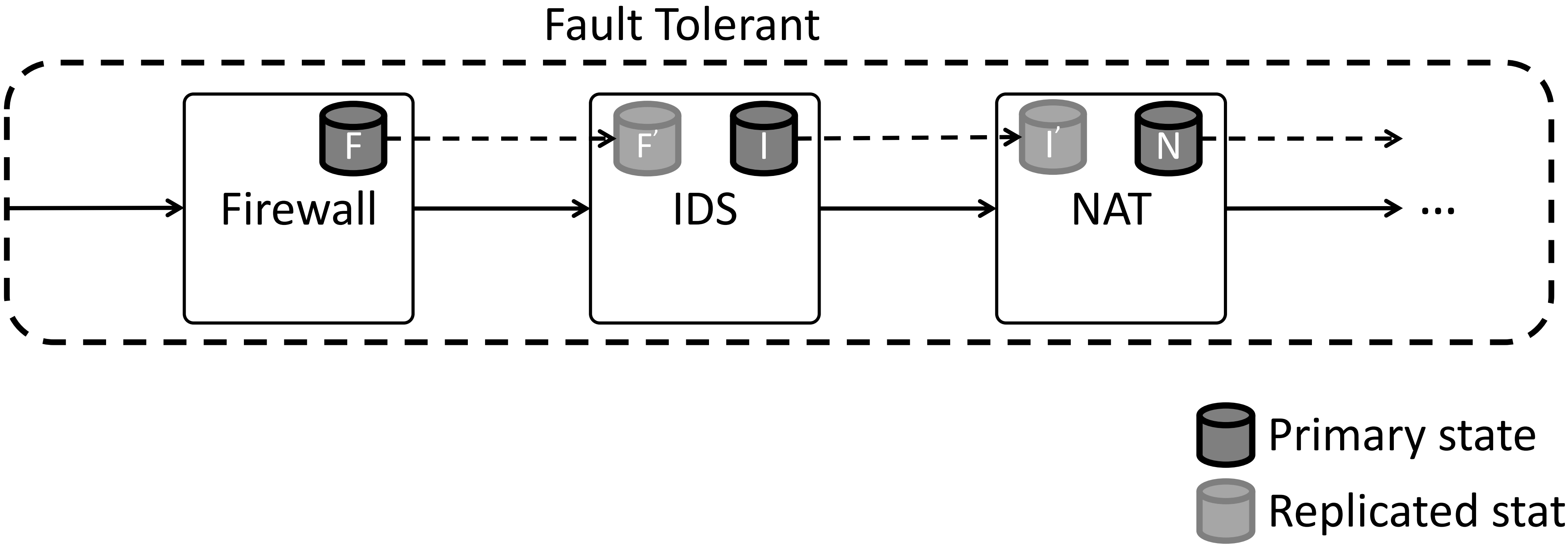
Bob ⇔ Bing

Alice ⇔ Apple

Snapshot Based Approaches for a Chain



Our Approach



Goals

Consistent state replication to tolerate f middlebox failures

Minimizing performance overhead during normal operation

Minimizing disruption during middlebox failures

Fault Tolerant Chaining (FTC)

In-chain replication

- Replicates a chain’s state instead of the state of individual middleboxes
- Each middlebox’s state replicated to subsequent f middlebox servers

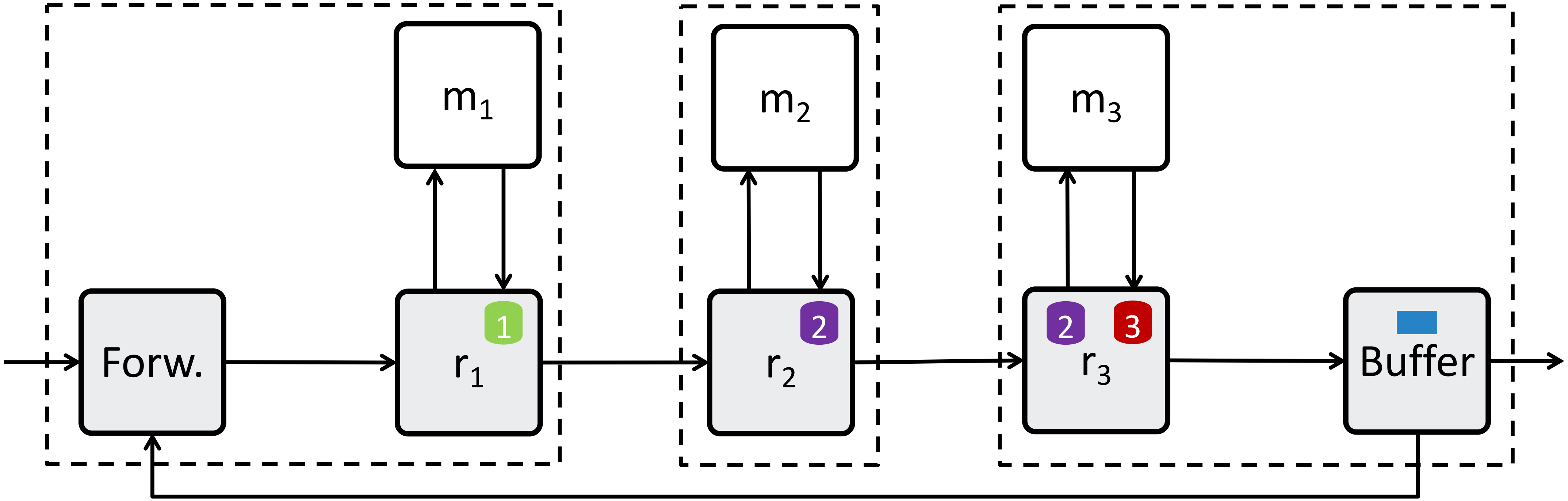
Transactional packet processing

- Simplifies the development of multi-threaded middleboxes
- Improves scalability and performance

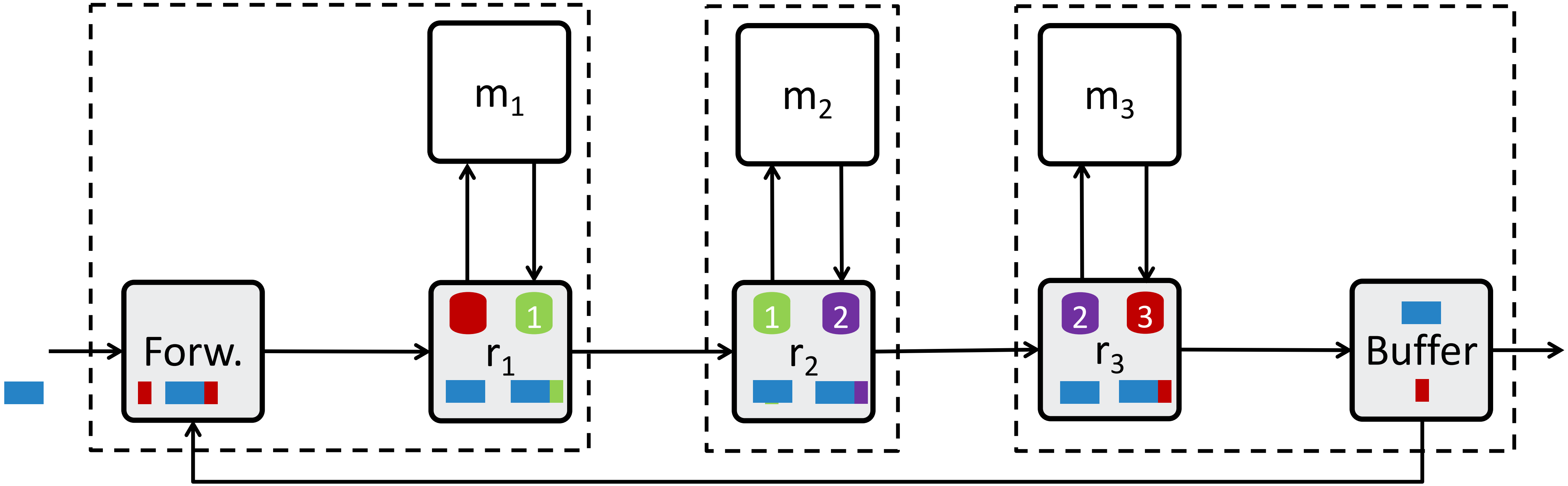
Data dependency vectors

- Enables concurrent state replication

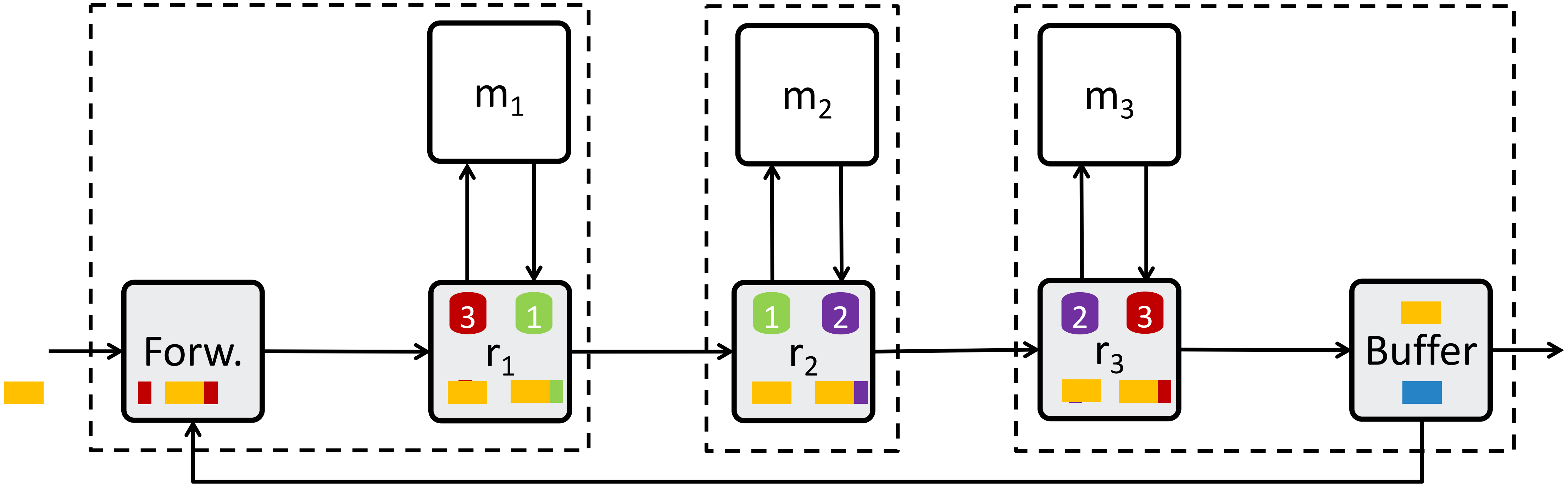
Normal Operation



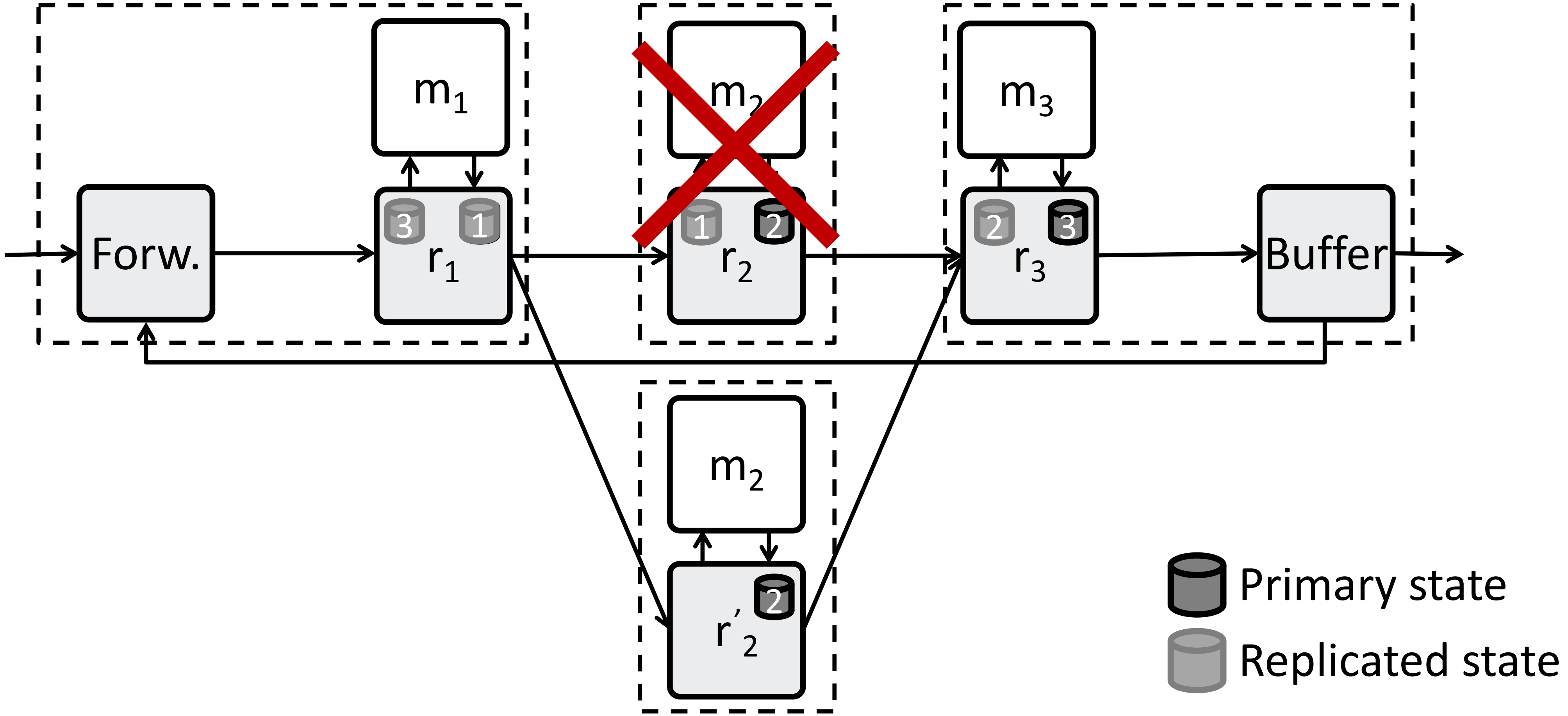
Normal Operation



Normal Operation



Failure Recovery



Transactional Packet Processing

Existing approaches

- Single thread or batched packet processing
- FTMB: multi threaded packet processing
  - Tracking state changes in granularity of each state variable read/write
  - Frequent periodic state snapshots

Our approach

- Packet transaction model for concurrent packet processing
- Using isolation property to tracking state changes in granularity of packet transactions

Data Dependency Vectors

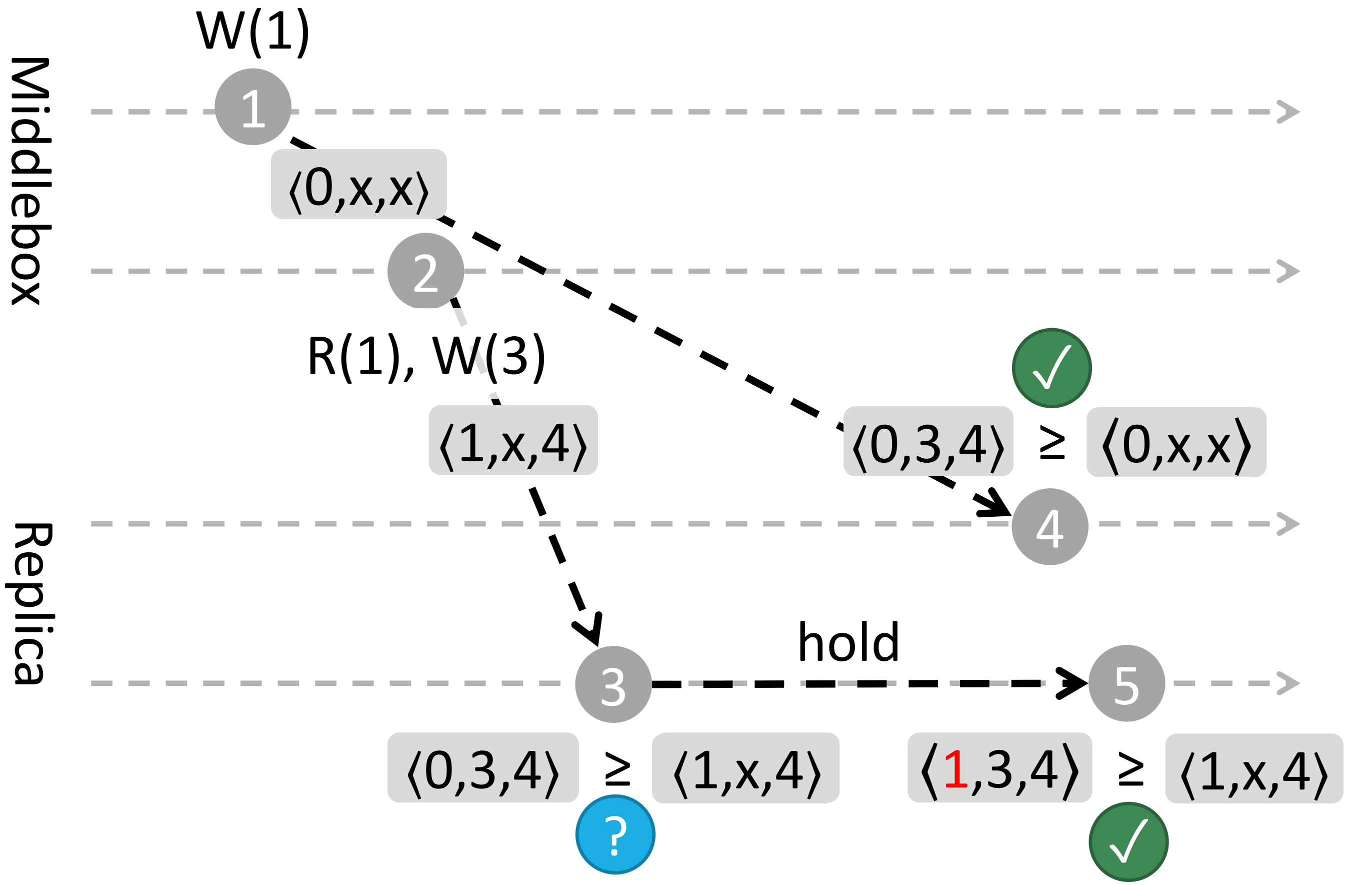
Tracking data changes instead of thread operations

Enabling different number of threads at the middlebox and replicas

- Fail over to smaller machine
- Scale up to a larger machine

Middlebox	Product	Throughput	CPU Core
IPSec	HP VSR1001	268 Mbps	1
	HP VSR1008	926 Mbps	8
WAN Optimizer	STEELHEAD CCX770M	10 Mbps	2
	STEELHEAD CCX1555M	50 Mbps	4
WAF	Barracuda Level 1	100 Mbps	1
	Barracuda Level 5	200 Mbps	2

Data Dependency Vectors Example



Middlebox's dependency vector:

⟨0,3,4⟩ – 1 → ⟨1,3,4⟩ – 2 → ⟨2,3,5 ⟩

⟨0,x,x⟩ ⟨1,x,4⟩

Replica's dependency vector:

⟨0,3,4⟩ – 4 → ⟨1,3,4⟩ – 5 → ⟨2,3,5 ⟩

Evaluation

METHOD

Comparing FTC with:

NF, **N**on-**F**ault tolerant system

- Ideal performance

FTMB (SIGCOMM 2015)

- State logging ~~+ Snapshots~~

FTMB + Snapshot (SIGCOMM 2015)

- State logging + Snapshots

ENVIRONMENTS

A cluster of 12 servers

- 40 Gbps network

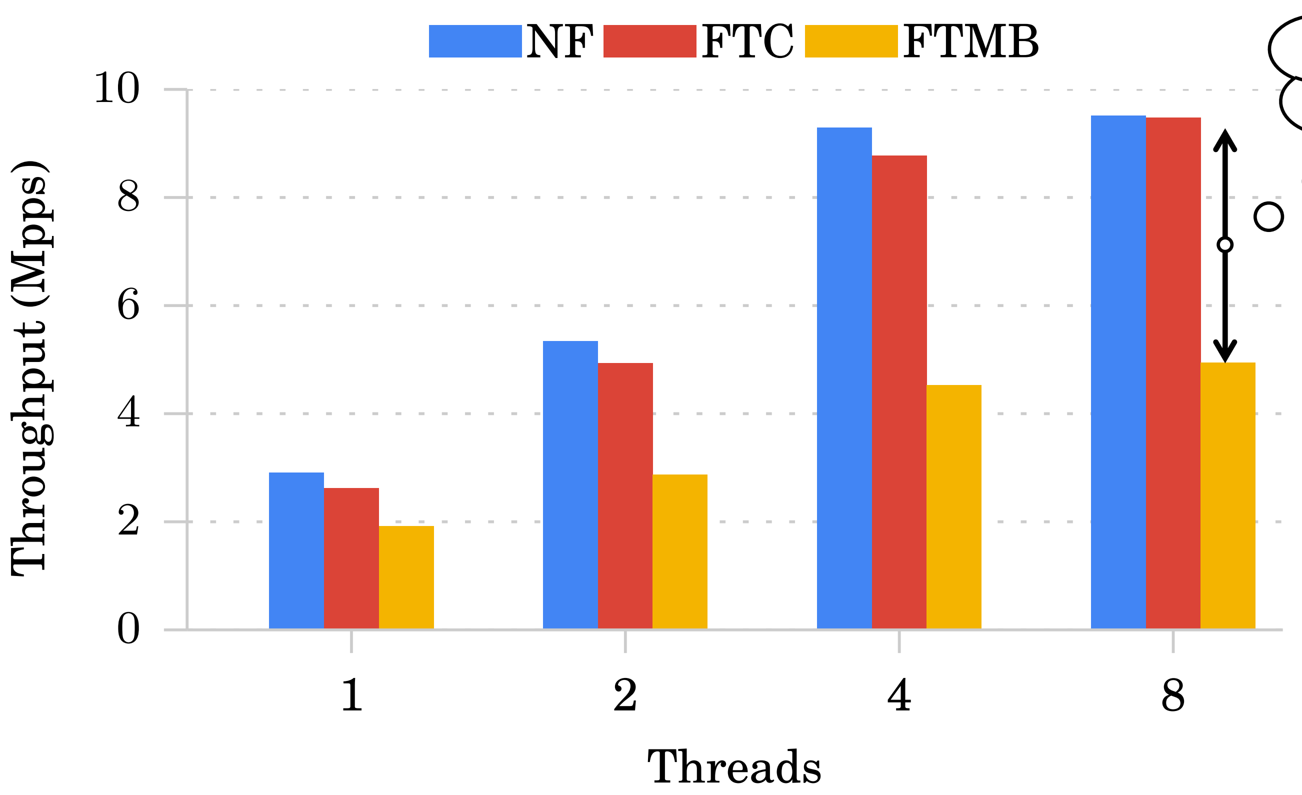
SAVI Cloud environment

- A virtual network of OVS switches

MoonGen and pktGen traffic generators

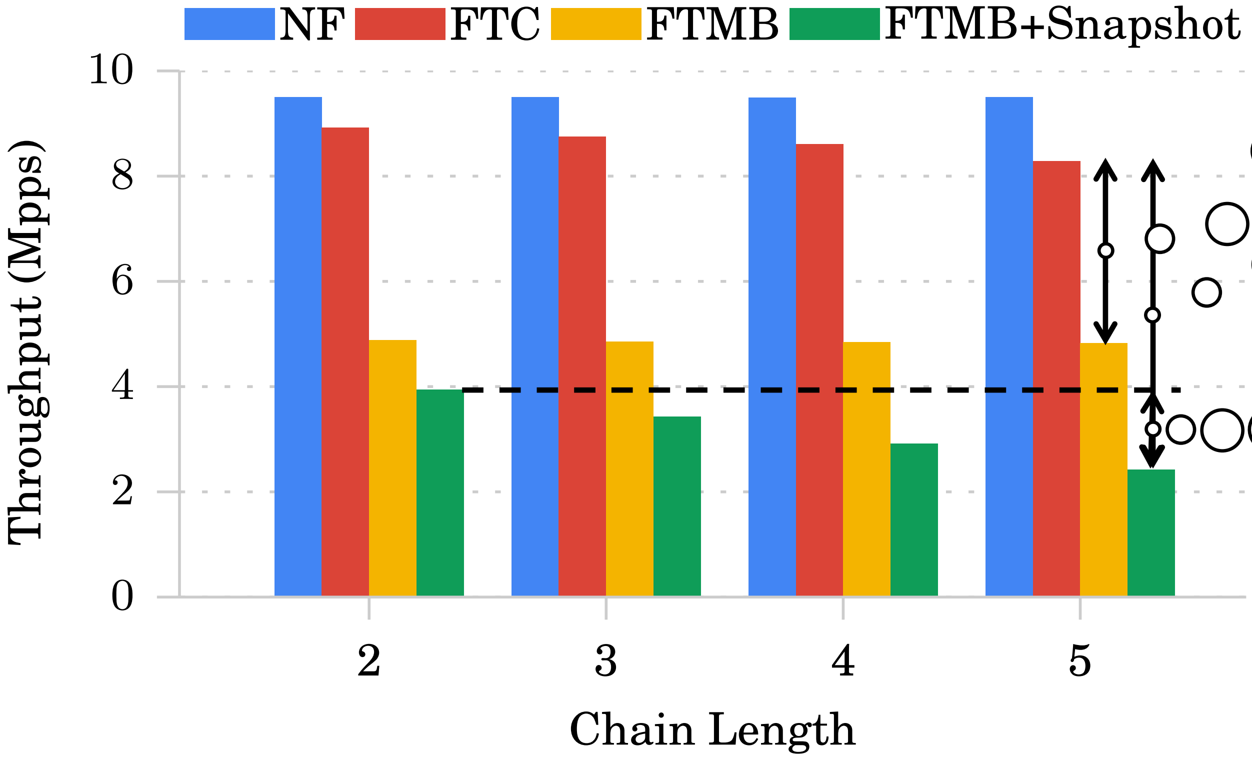
- UDP traffic
- Packet size: 256 B

Fault Tolerant NATs



2x higher throughput

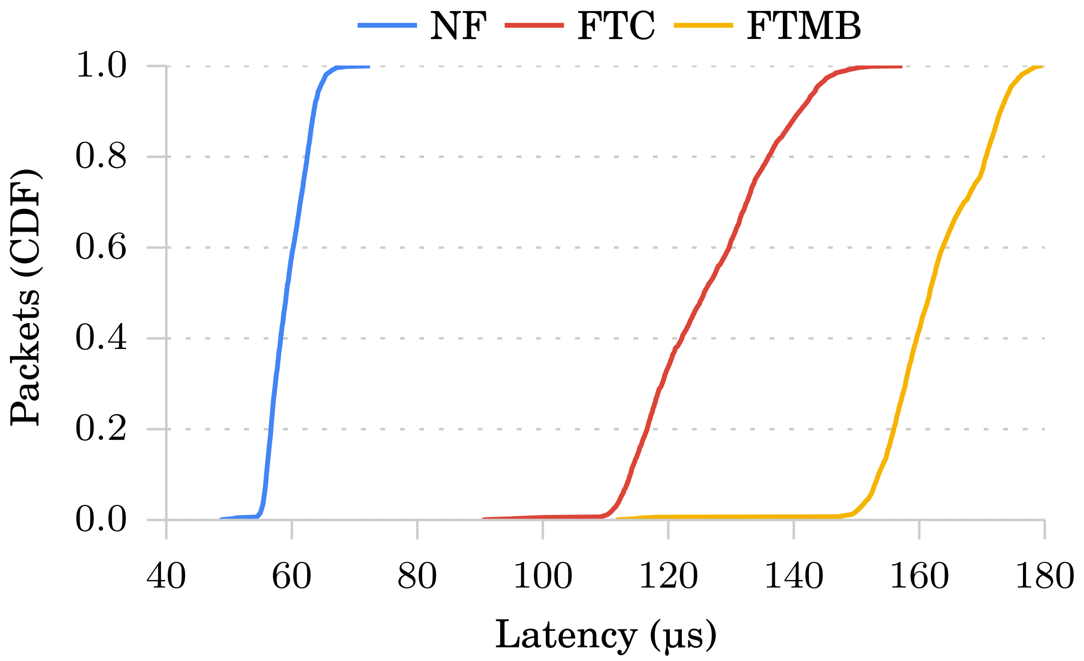
Fault Tolerant Chains – Throughput



1.8x higher throughput

39% drop due to snapshots

Fault Tolerant Chains – Latency



Conclusion

Keep operation of a chain of middleboxes online after f middleboxes fail

- In-chain replication
- Transactional packet processing
- Data dependency vectors