Poster #61: Composing SDN Controller Enhancements with Mozart

Zhenyu Zhou[1,*] and Theophilus A. Benson[2]

[1]Duke University, [2]Brown University, [*]Presenter

1 Background

Cloud providers employ Software Defined Networking to

- simplify network management
- configure networking infrastructure using higher level abstractions

Decoupling SDN

- SDNApps: Networking functionality
- SDNEnhancements: Optimizations

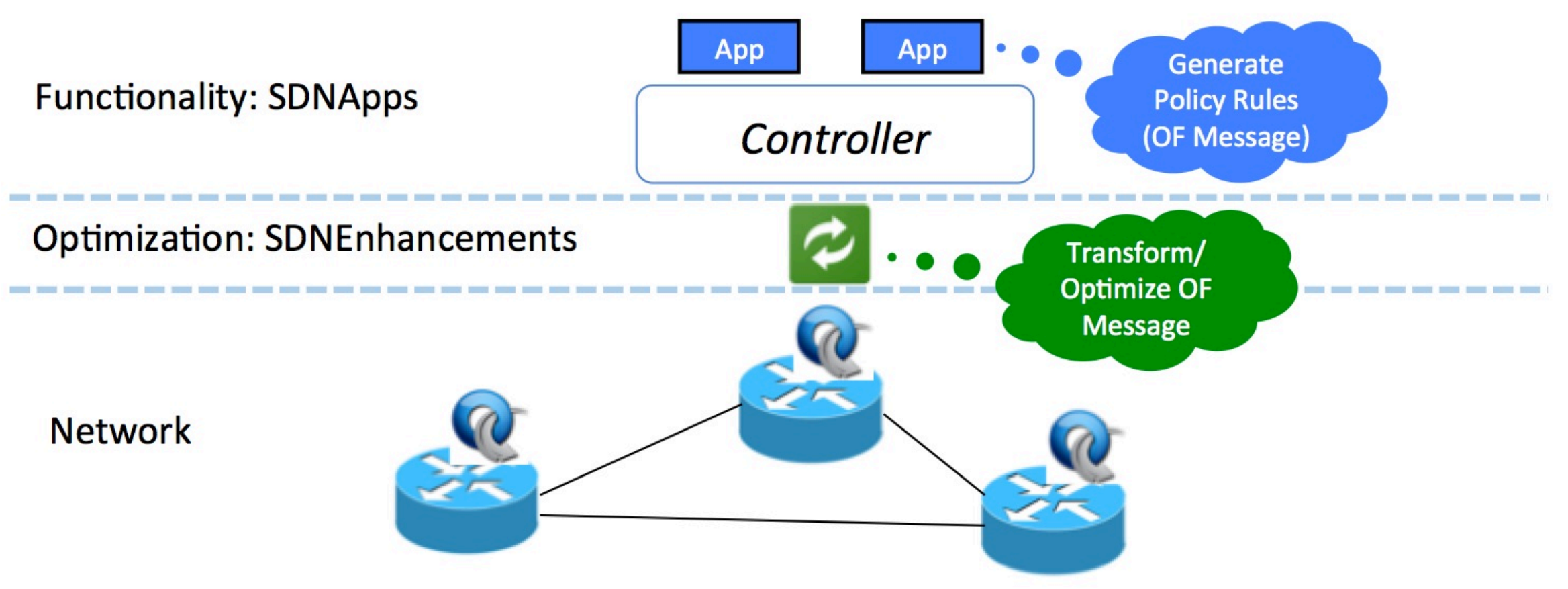

Functionality: SDNApps
App
App
Controller
Generate Policy Rules (OF Message)
Optimization: SDNEnhancements
Transform/ Optimize OF Message
Network

Figure 1: **Introducing SDNEnhancements.**

Hidden Danger

However, the SDNEnhancements creates a disconnect between the SDNApps' view of the network and the actual network state!

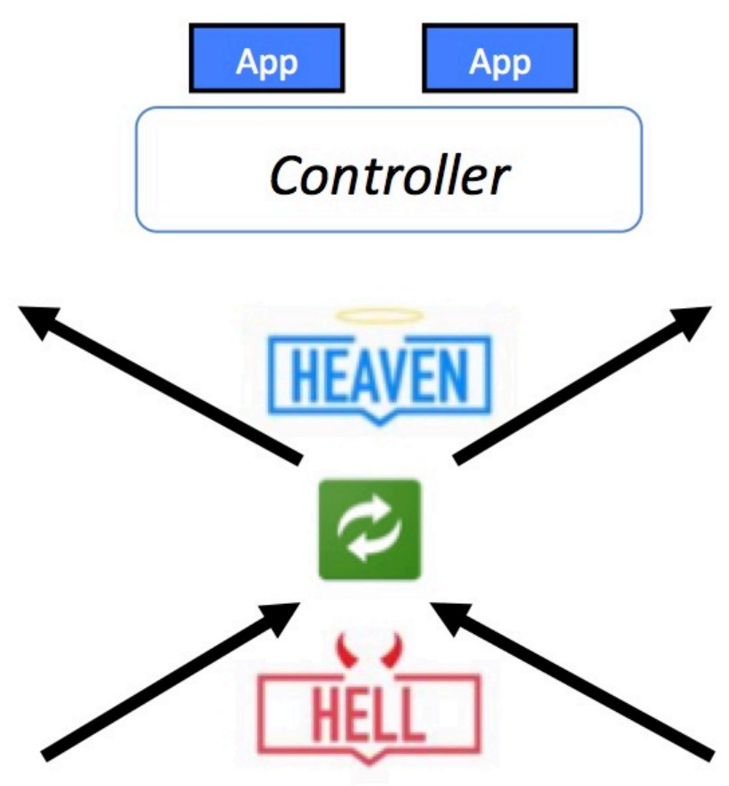


Figure 2: **SDNApps' View is Disconnected from the Actual Network State.**

2 Motivation

SDNApps have assumptions to the network

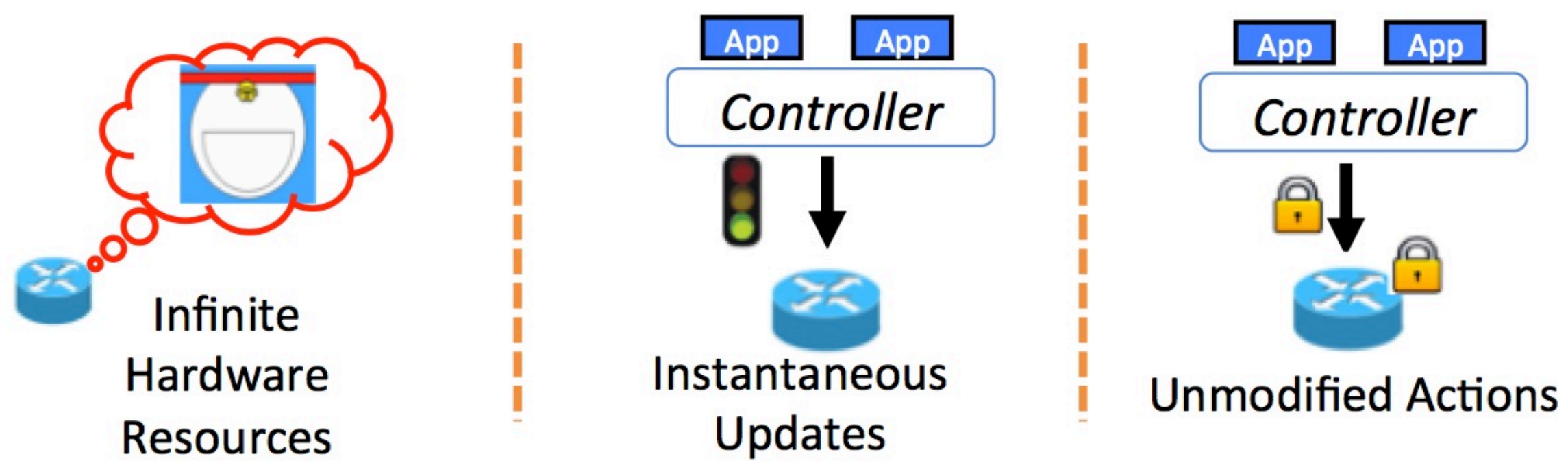


Figure 3: **SDNApps' Assumptions.**

Case Study: Hedera

- Improving data center performance by load balancing elephant flows

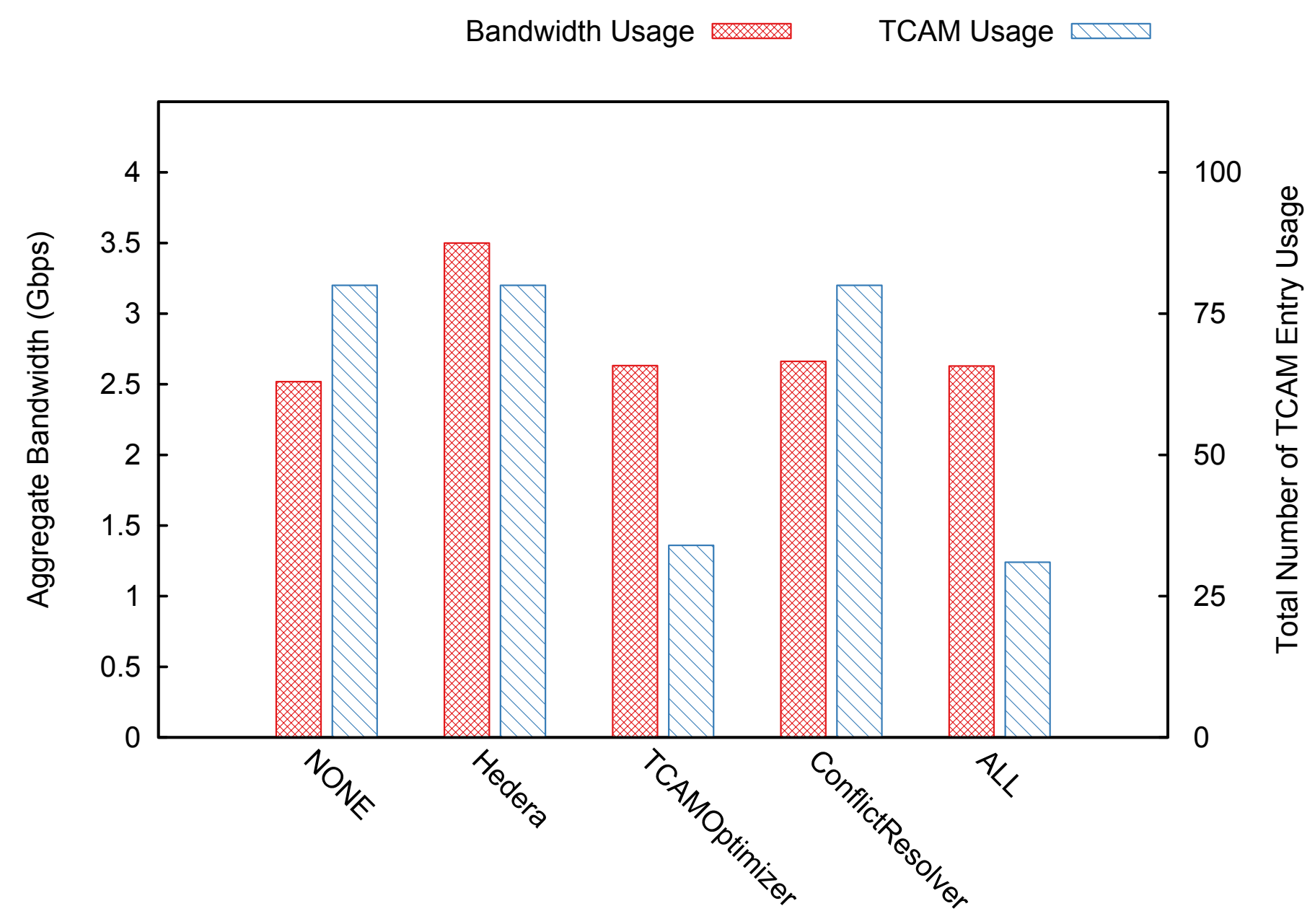


Figure 4: **Aggregate Bandwidth and TCAM Usage.**

Two fundamental questions remain not answered

Research Questions

What is the right interface for enabling principled interactions between SDNApps and SDNEnhancements?

What abstractions are required to systematically include SDNEnhancements into the SDN ecosystem?

3 Design

Main Idea

- Developers simply specify the class of transformations that are tolerable, or not.
- No requirements to understand all SDNEnhancements.

Analogy to Compiler Optimization:

Compilers for SDNs

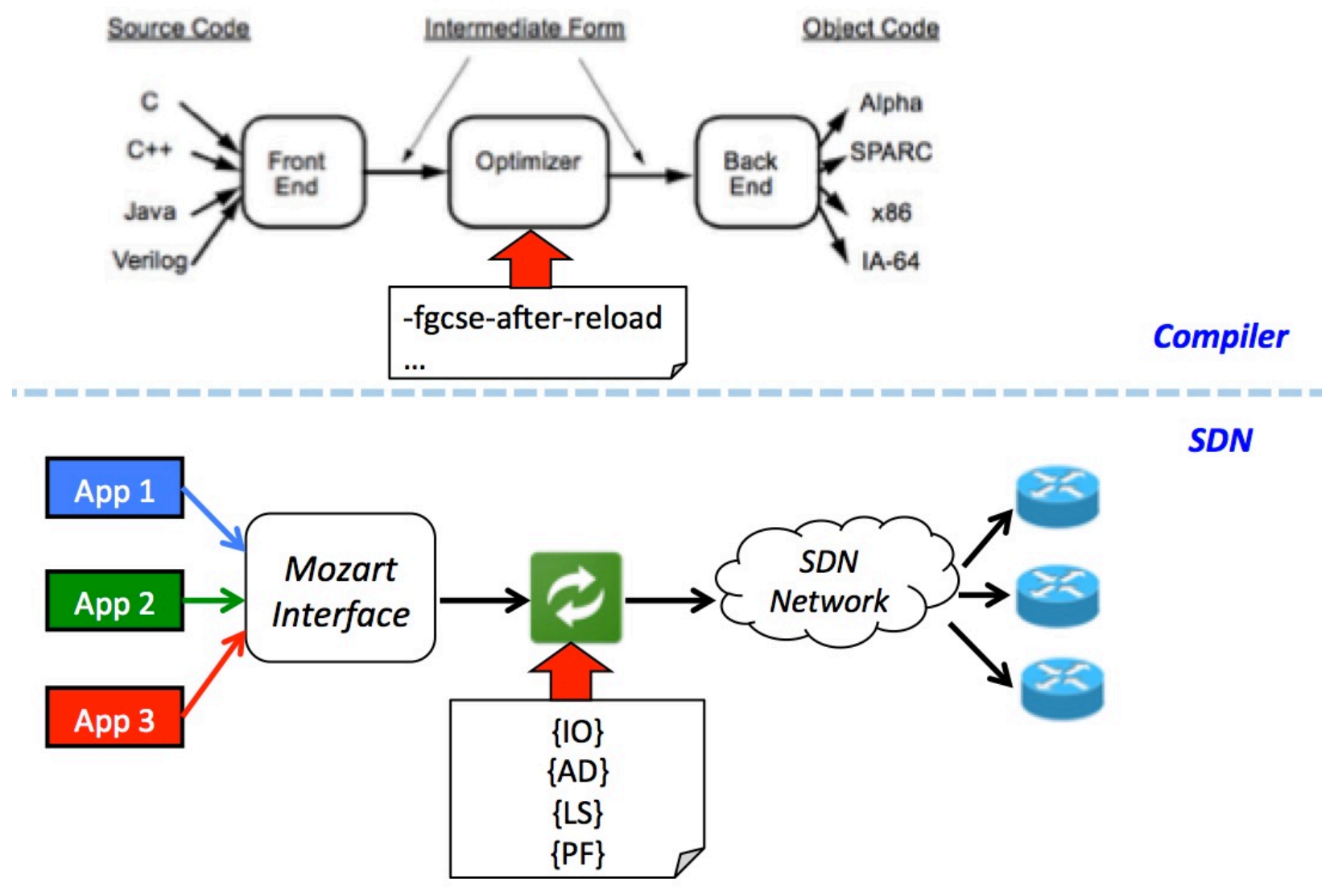


Figure 5: **Analogy to Compiler Optimization.**

(Ref: `https://www.cs.cmu.edu/afs/cs/academic/class/15745-s02/www/lectures/lect1.pdf`)

- *SDN assembly code*: low-level control messages
- *"Code block"*: policies among a certain group of hosts
- *Compilation*: SDNEnhancement function
- *SDN compiler flags*: *SDN-Flags*

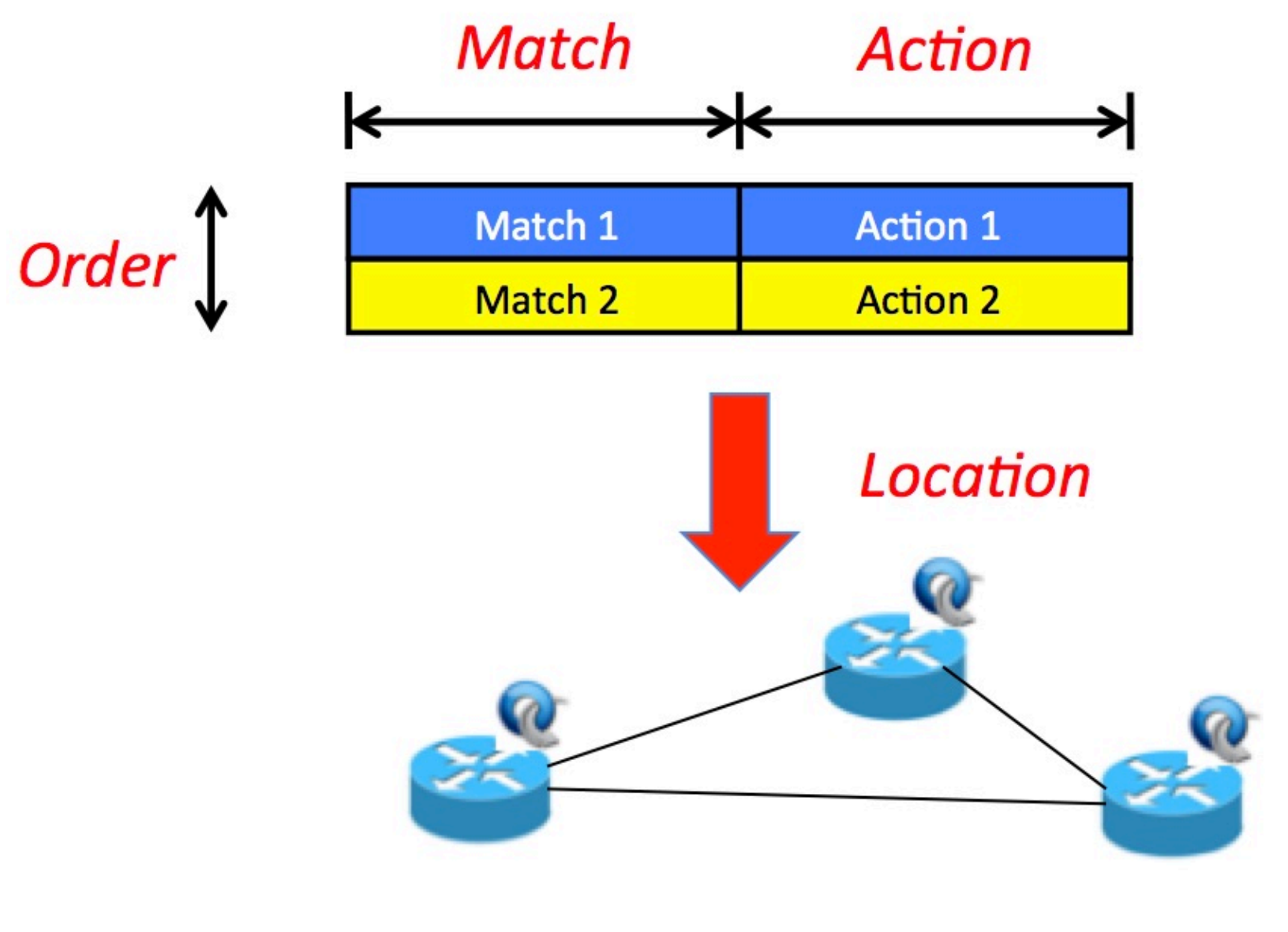


Figure 6: **Dissecting SDN-Flags.**

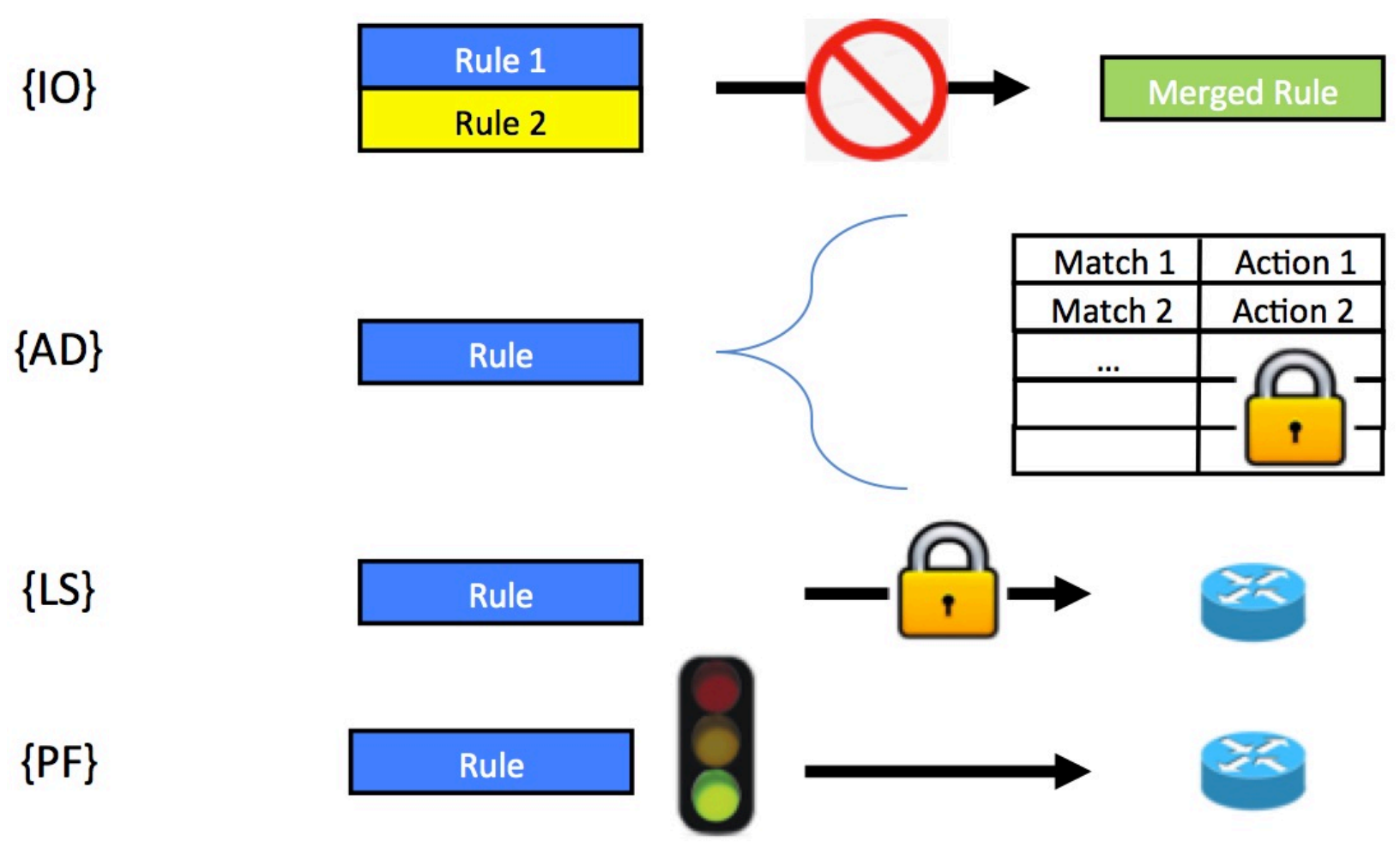


Figure 7: **SDN-Flags.**

The Mozart Orchestrator ensures that SDN-Flags are respected.

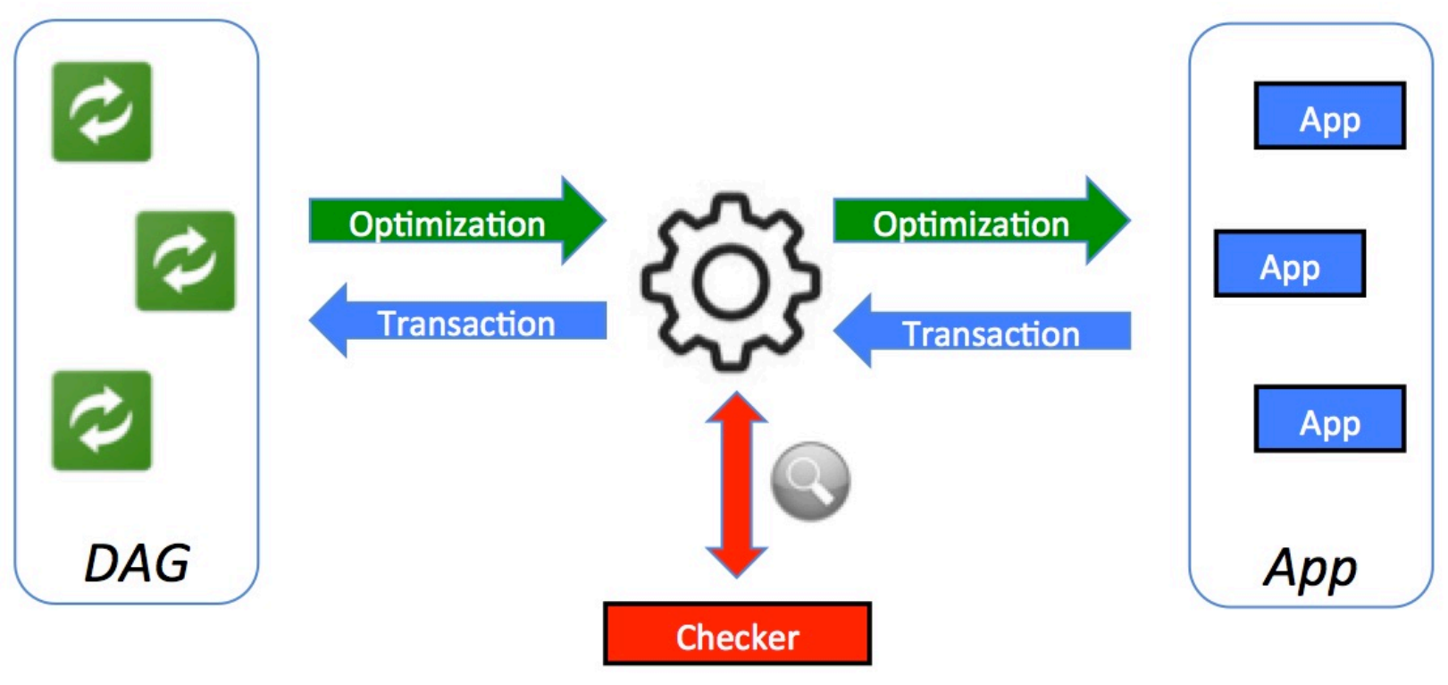


Figure 8: **The Mozart Orchestrator.**

4 Evaluation

Performance Improvement

- Proactive SDNApp (Hedera): Saves 24.8% reduction in aggregate bandwidth introduced by TCAMOptimizer and decreases TCAM usage saving from 57.5% to 18.2%.

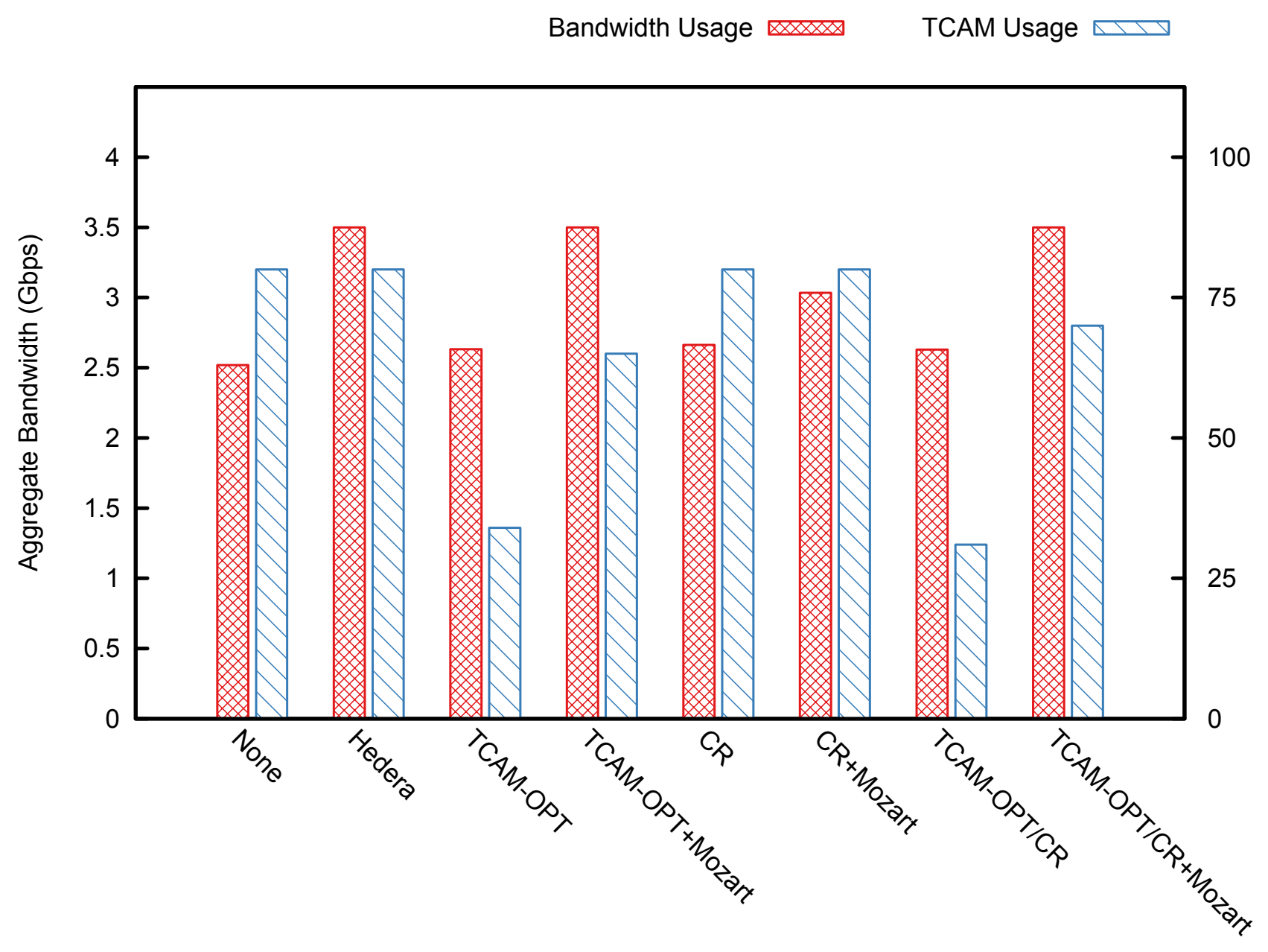


Figure 9: **Aggregate Bandwidth and TCAM Usage.**

- Reactive SDNApp (RtFlow): Flows get activated 7.8 times faster at initial ramp of phase and 44.8 times faster regarding to time to recovery.

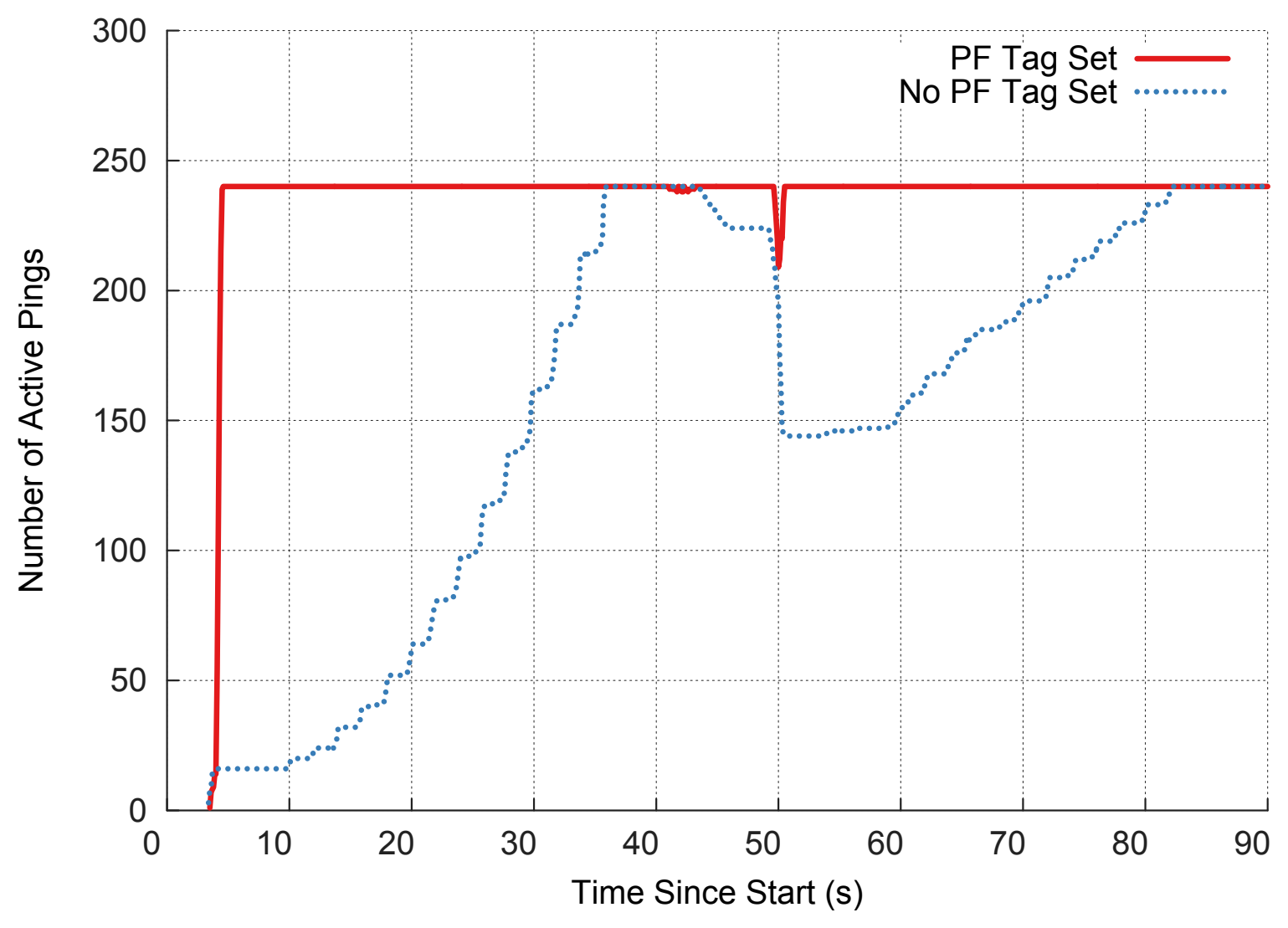
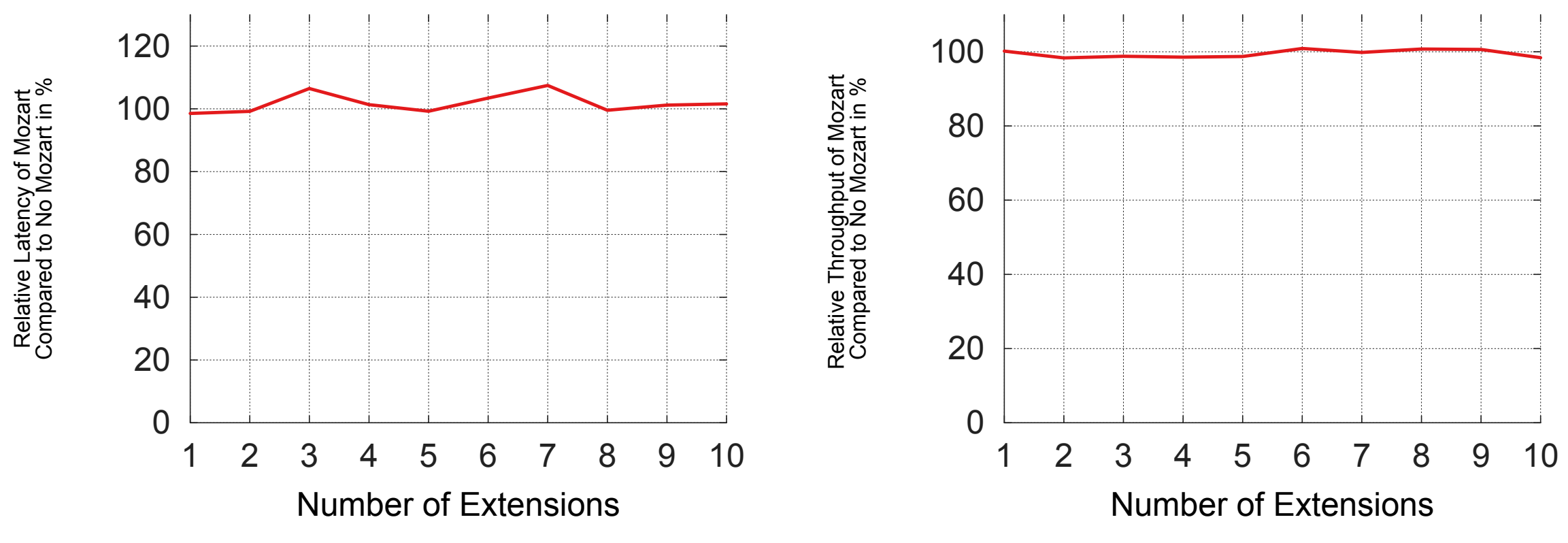


Figure 10: **Ping Latency in Link Failure Experiment.**

Mozart Overhead

- Sublinear.
- Only increases 1.58% to latency.



(a) (b)

Figure 11: **(a) Relative Latency of Mozart Compared to No Mozart in %. (b) Relative Throughput of Mozart Compared to No Mozart in %.**