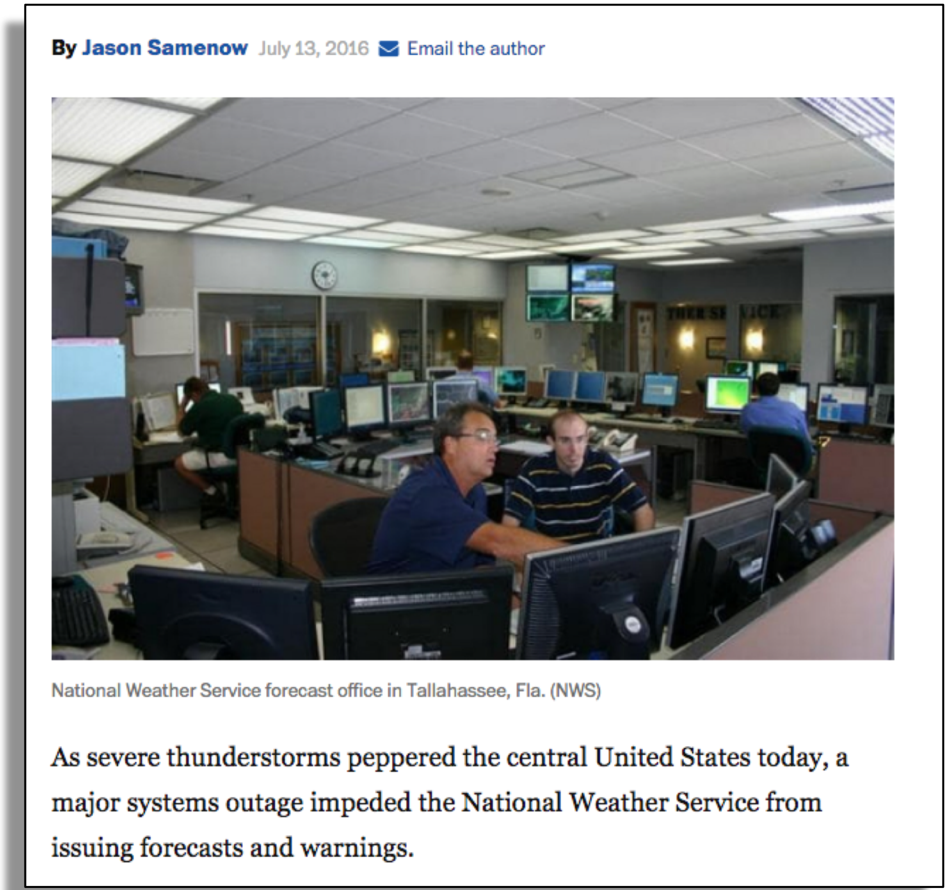
Network Verification

[A preview for SIGCOMM 2018]

David Walker

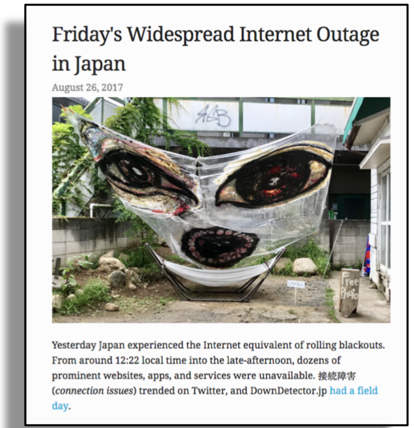
PRINCETON UNIVERSITY

Weather Service suffers 'major network issue,' warning system compromised


By Jason Samenow July 13, 2016 Email the author

National Weather Service forecast office in Tallahassee, Fla. (NWS)

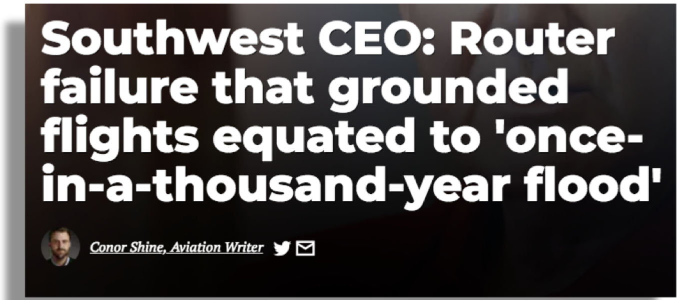
As severe thunderstorms peppered the central United States today, a major systems outage impeded the National Weather Service from issuing forecasts and warnings.


Friday's Widespread Internet Outage in Japan

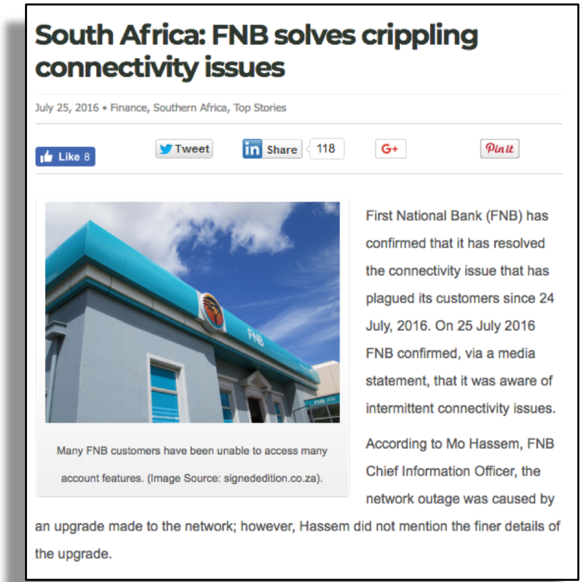
August 26, 2017

Yesterday Japan experienced the Internet equivalent of rolling blackouts. From around 12:22 local time into the late-afternoon, dozens of prominent websites, apps, and services were unavailable. 接続障害 (*connection issues*) trended on Twitter, and DownDetector.jp had a field day.

United says router issue responsible for grounding all flights


Southwest CEO: Router failure that grounded flights equated to 'once-in-a-thousand-year flood'

Conor Shine, Aviation Writer

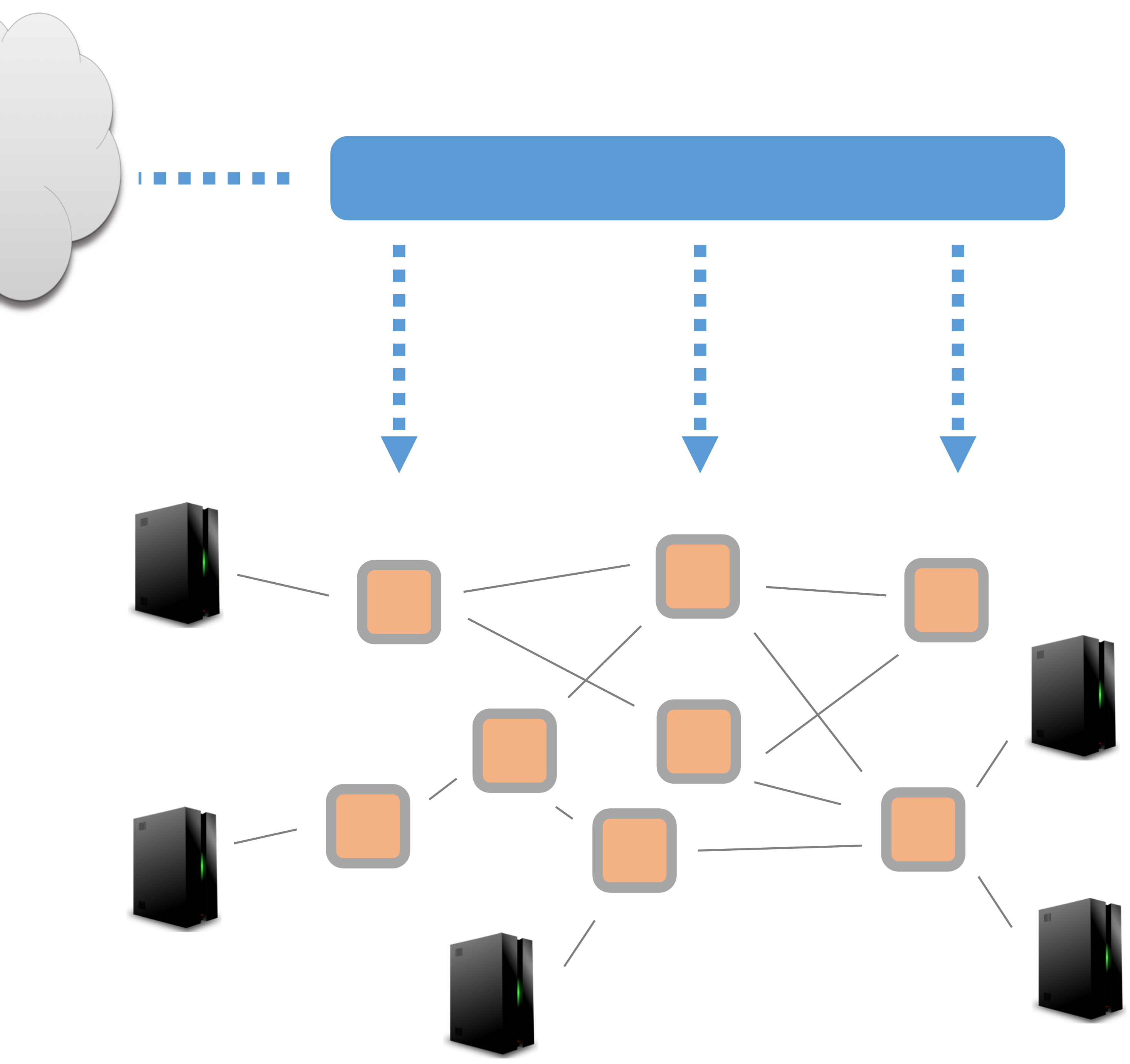

South Africa: FNB solves crippling connectivity issues

July 25, 2016 • Finance, Southern Africa, Top Stories

First National Bank (FNB) has confirmed that it has resolved the connectivity issue that has plagued its customers since 24 July, 2016. On 25 July 2016 FNB confirmed, via a media statement, that it was aware of intermittent connectivity issues.

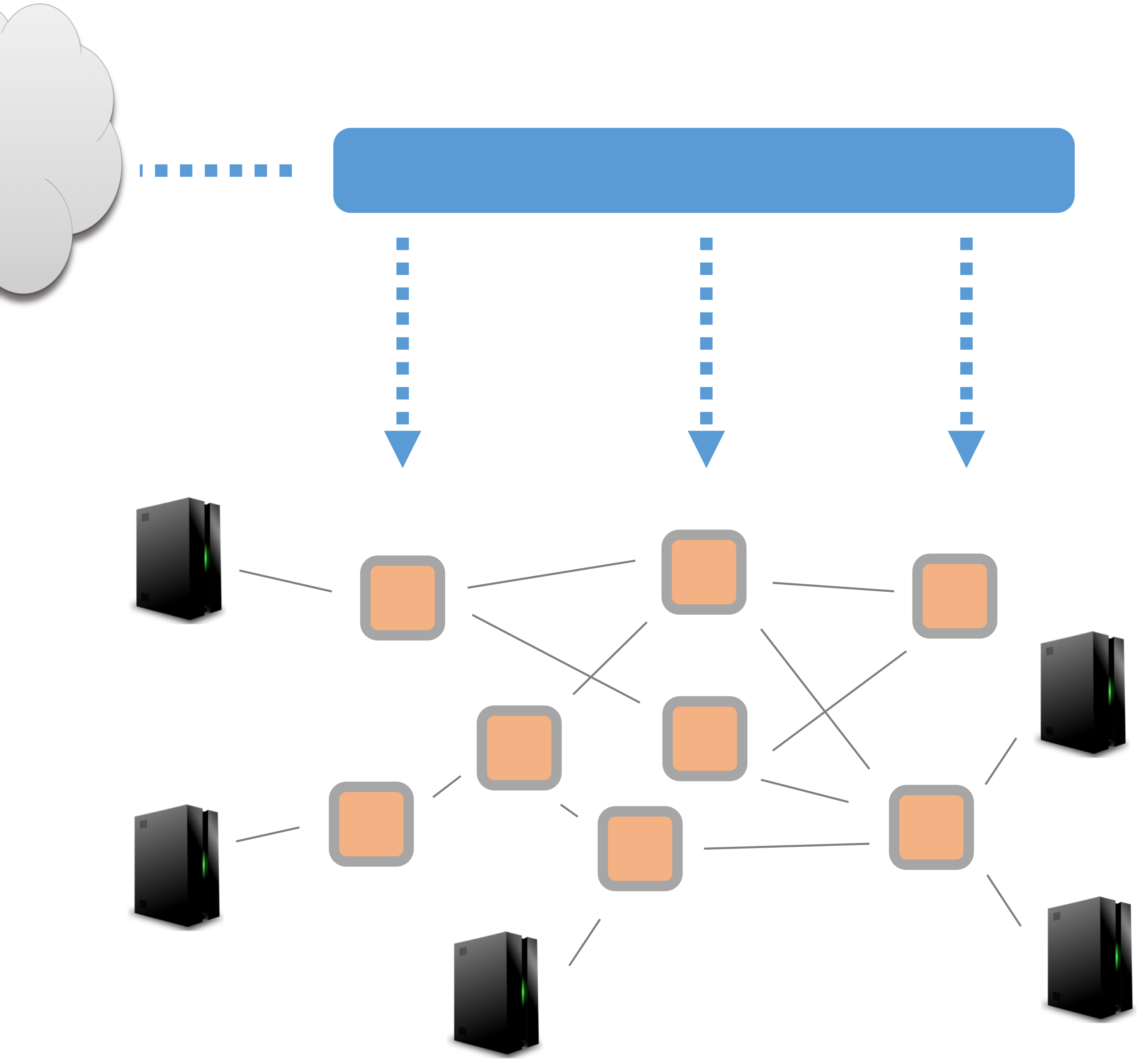
Many FNB customers have been unable to access many account features. (Image Source: signededition.co.za).

According to Mo Hassem, FNB Chief Information Officer, the network outage was caused by an upgrade made to the network; however, Hassem did not mention the finer details of the upgrade.



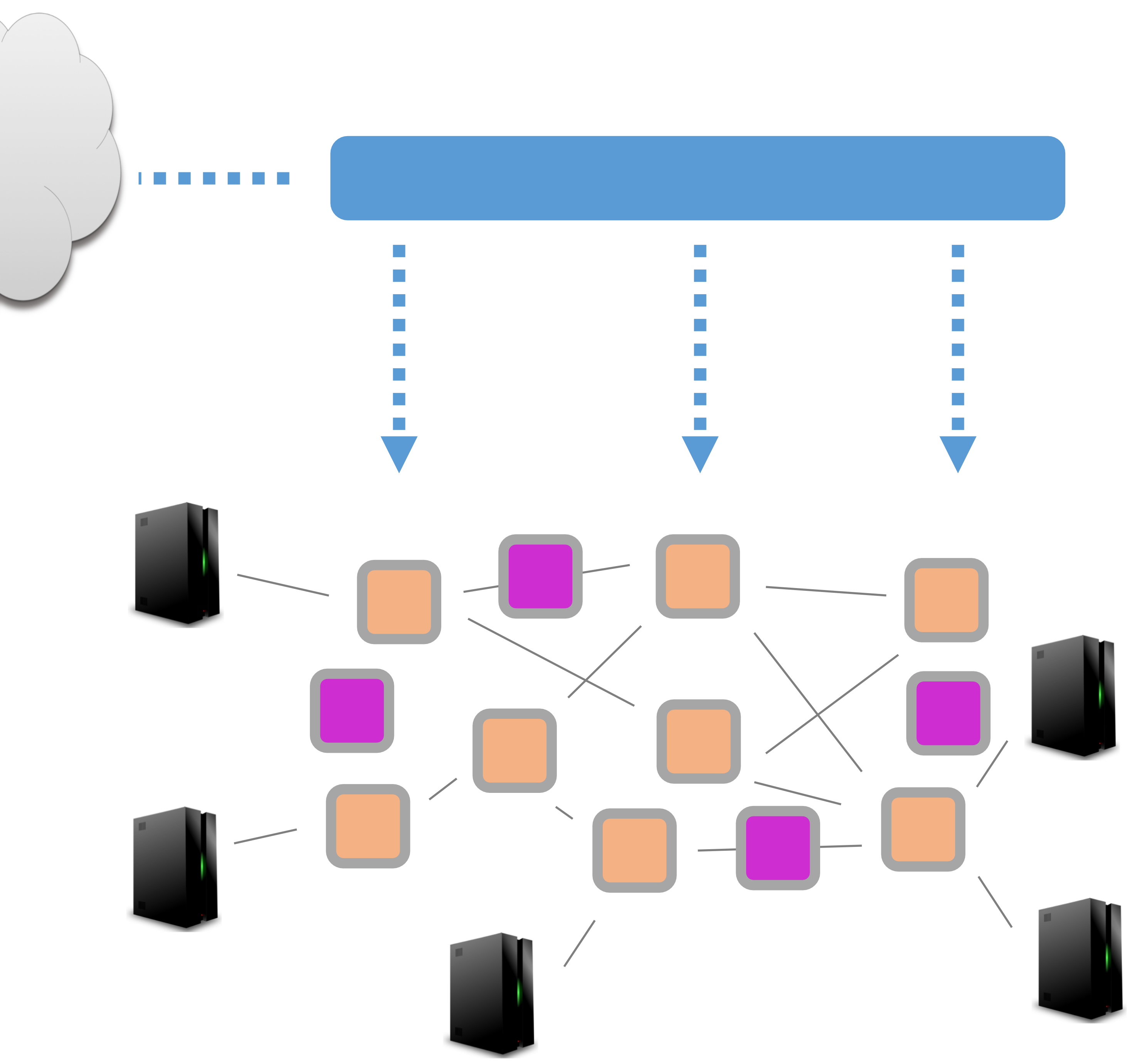
<u>Control Plane:</u>

- the algorithms that make routing decisions
- in traditional networks:
  - distributed protocols
  - defined by device configs
- in SDN networks:
  - centralized
  - defined by SDN programs



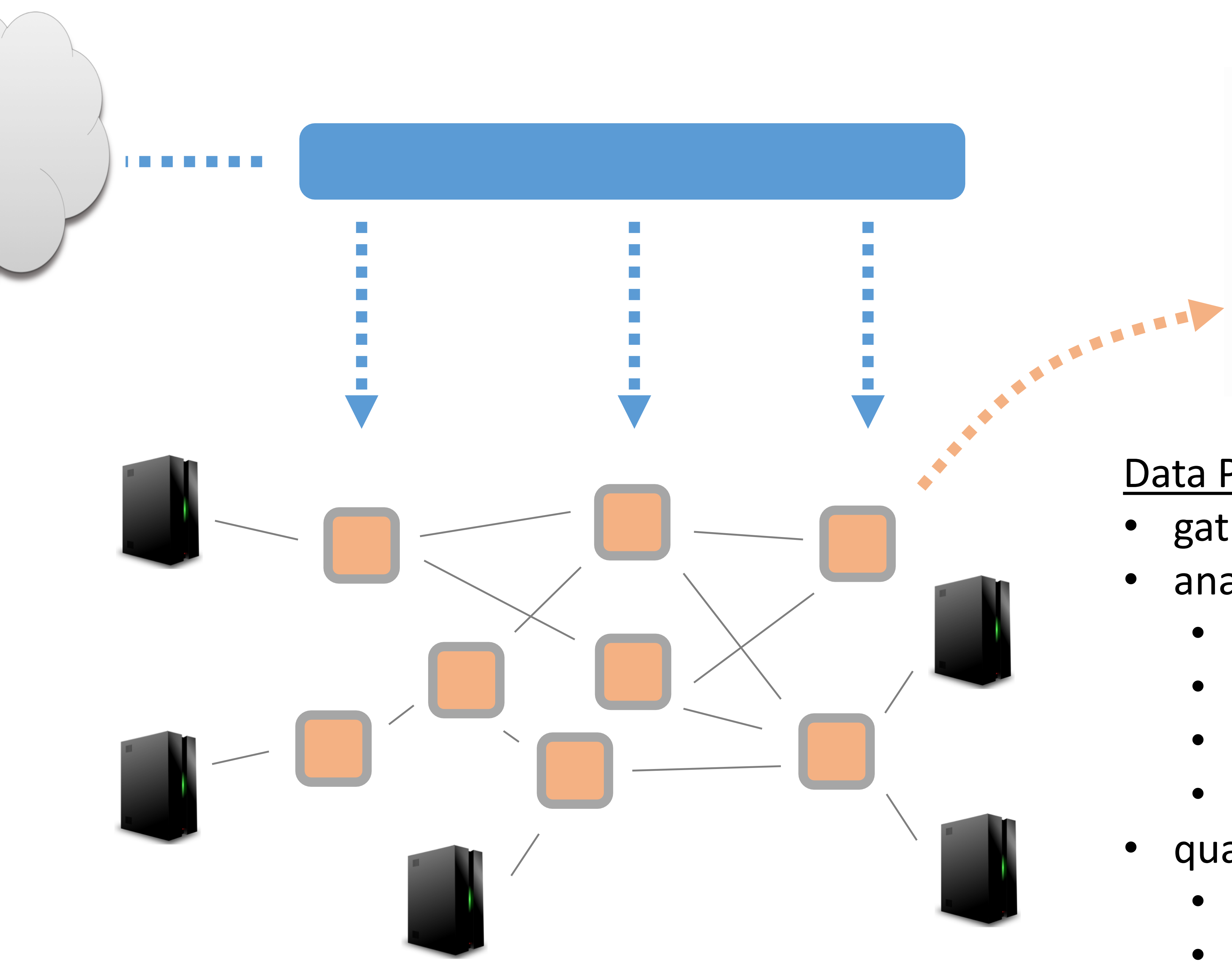
Data Plane

- implements routes
- defined by a series of match-action tables



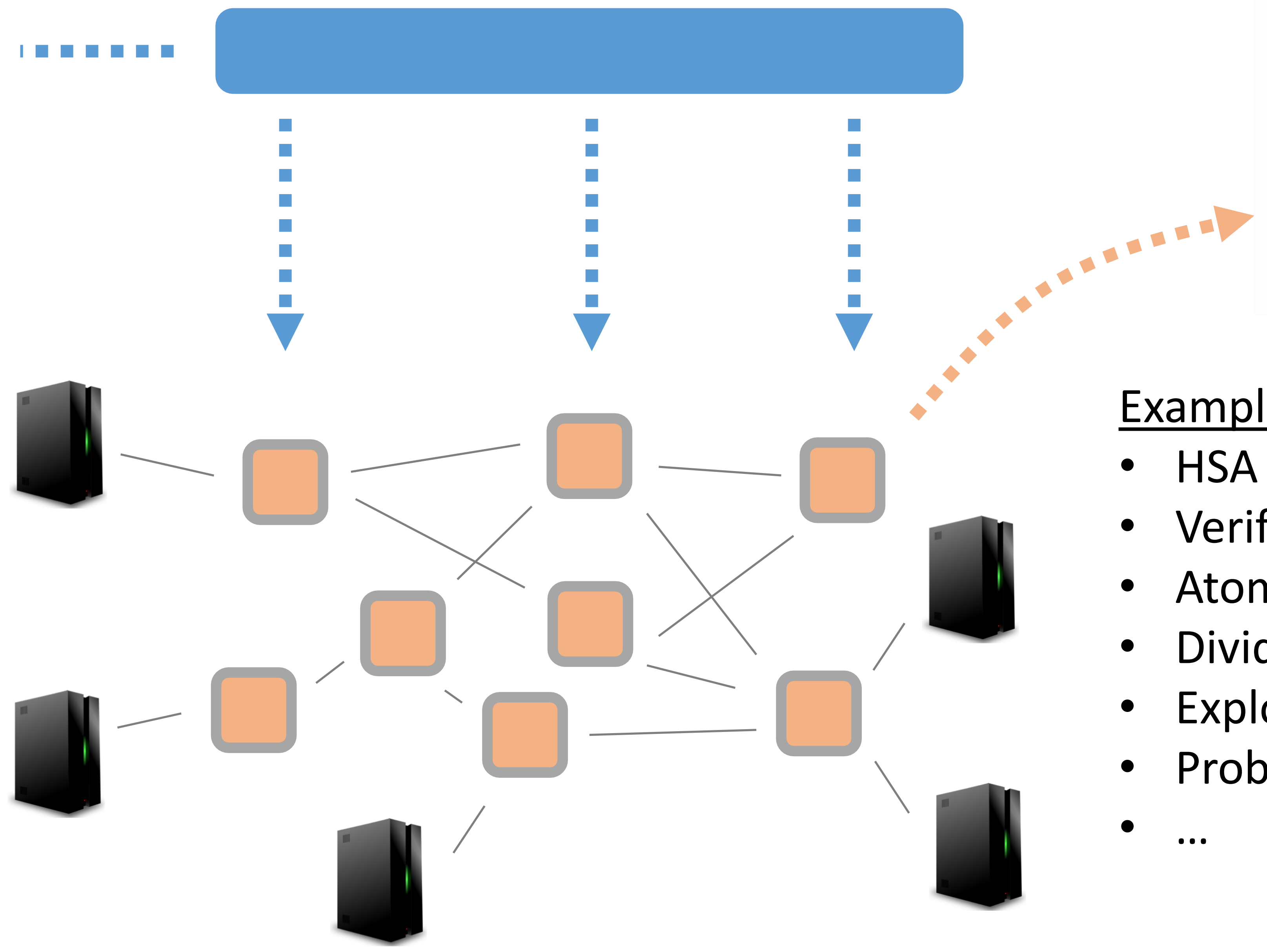
Middleboxes

- Many services:
    - Firewalls, intrusion detection
    - Load balancers, WAN optimizers
    - Application caches
    - …



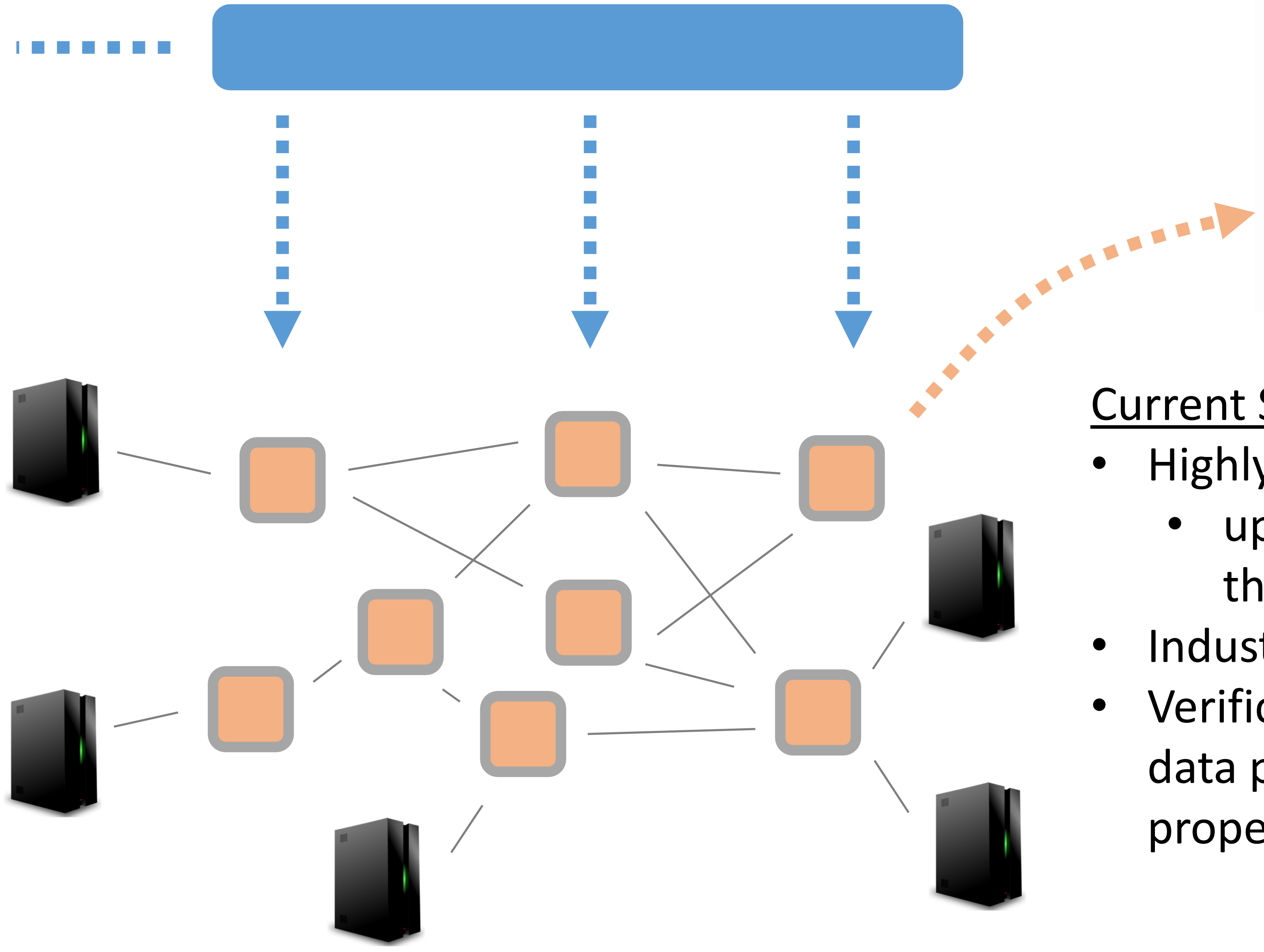
Data Plane Verification

- gather single DP snapshot
- analyze path properties
  - reachability
  - non-reachability
  - no black holes
  - no loops
- quantitative properties
  - congestion
  - failure probabilities



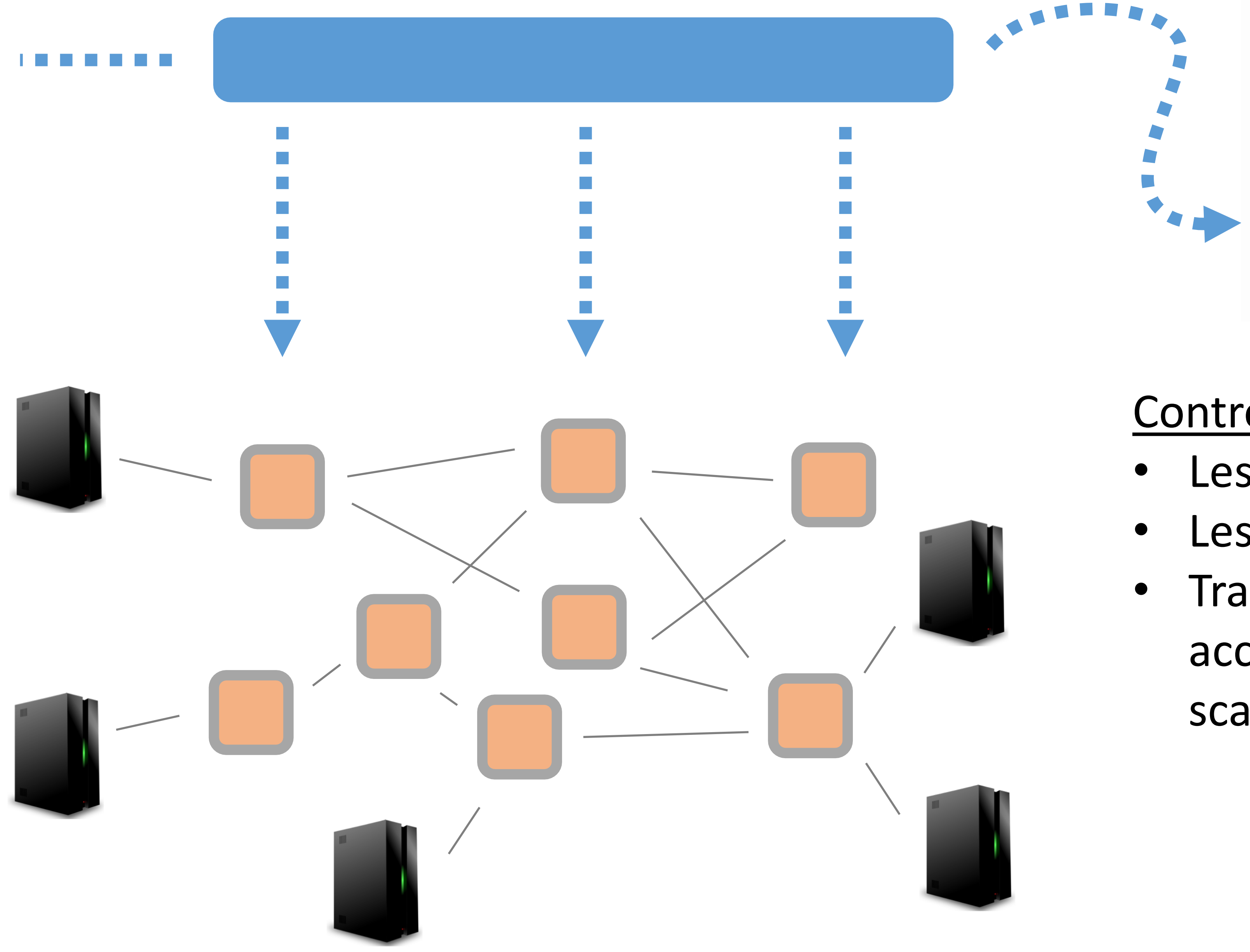
Example Systems

- HSA [NSDI ‘12]
- Veriflow [NSDI ‘13]
- Atomic predicates [ICNP ‘13]
- Divide-and-conquer [NSDI ‘14]
- Exploiting symmetry [POPL ’16]
- Probabilistic reasoning [POPL ‘17]
- …



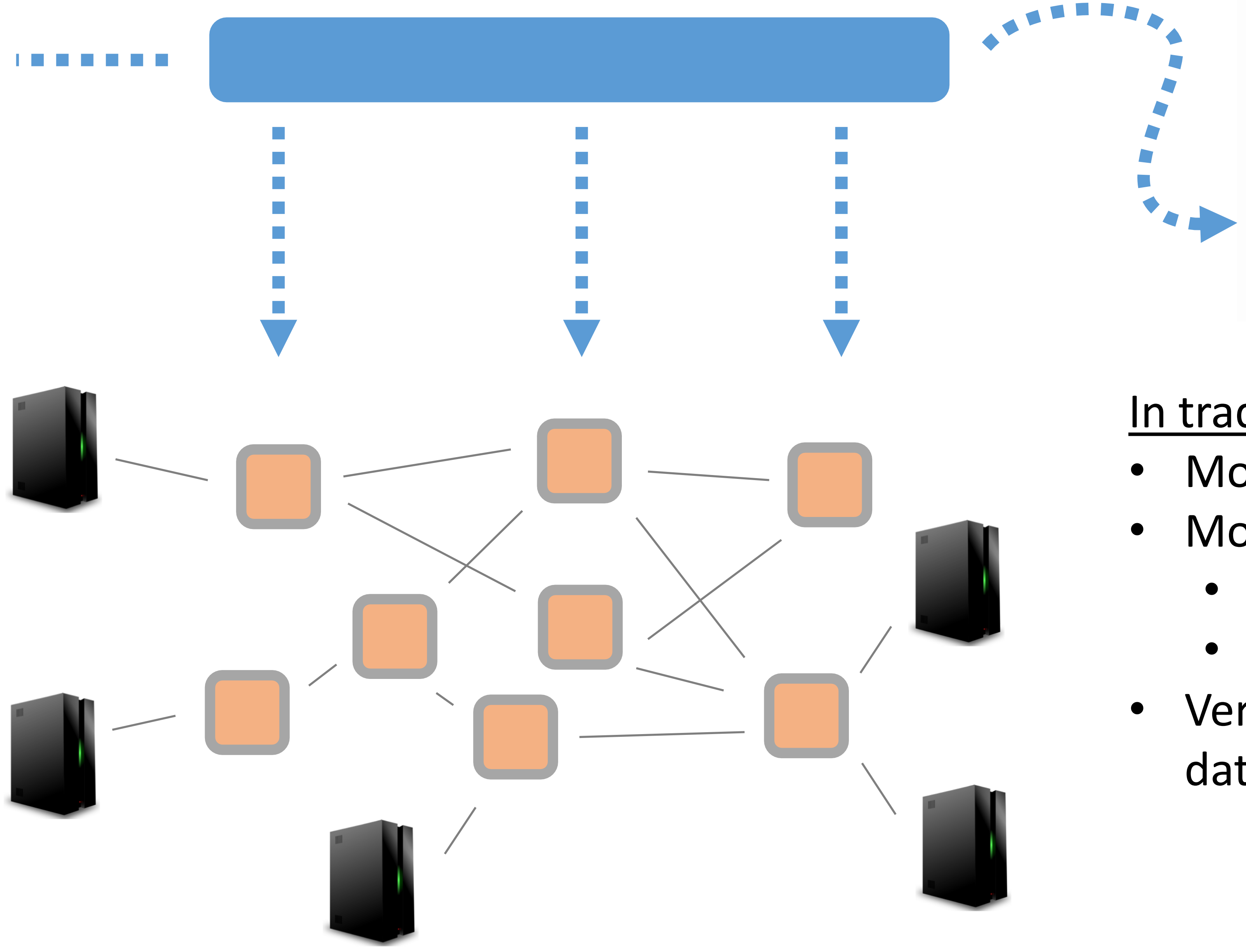
<u>Current Status</u>

- Highly scalable
  - up to ~10,000 switches in less than a minute [NSDI '14]
- Industry-ready; start-ups
- Verification of *static*, *stateless* data planes for *non-quantitative* properties is a solved problem



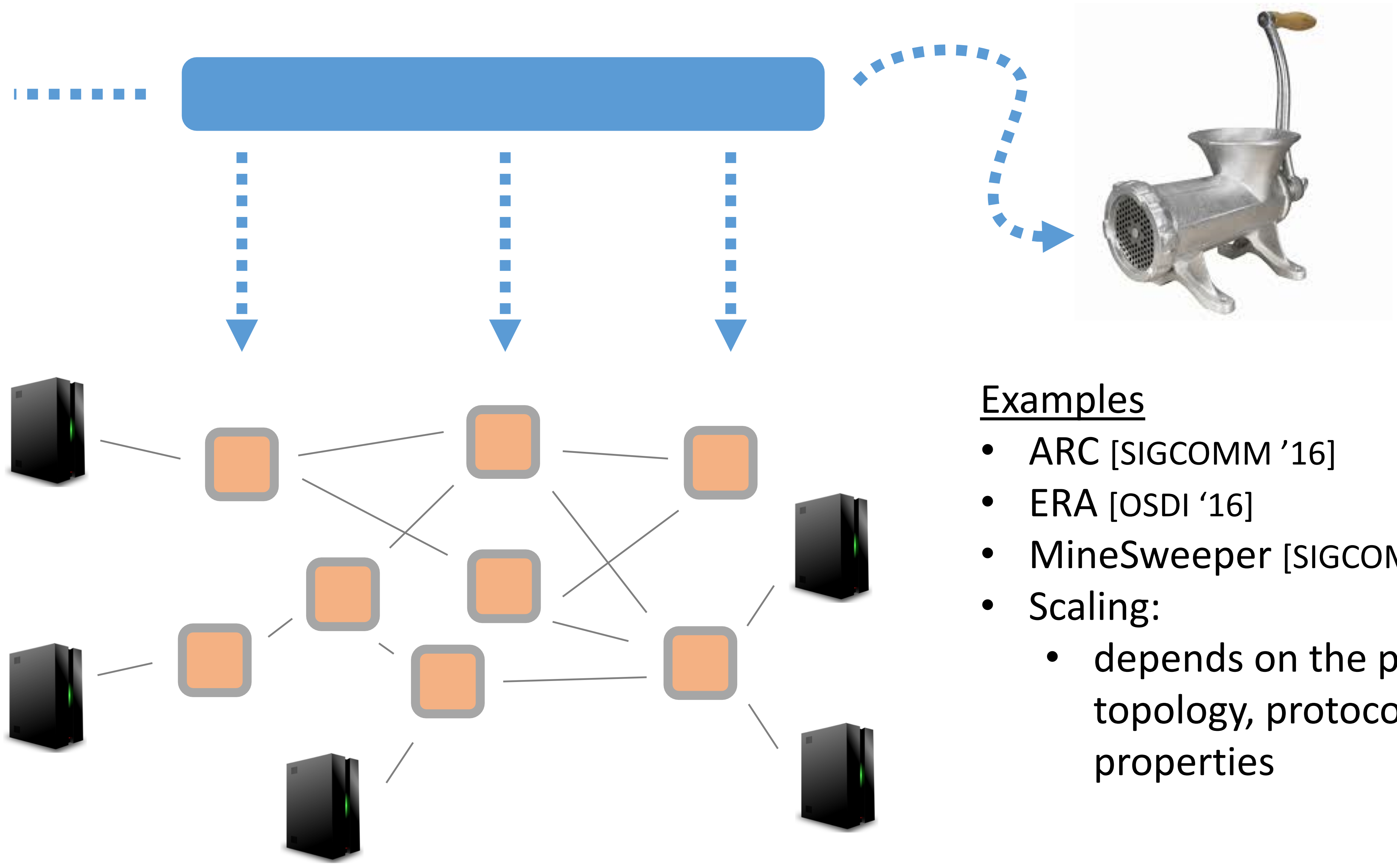
Control Plane Verification

- Less well understood
- Less scalable
- Trade-offs between model accuracy, properties and scalability



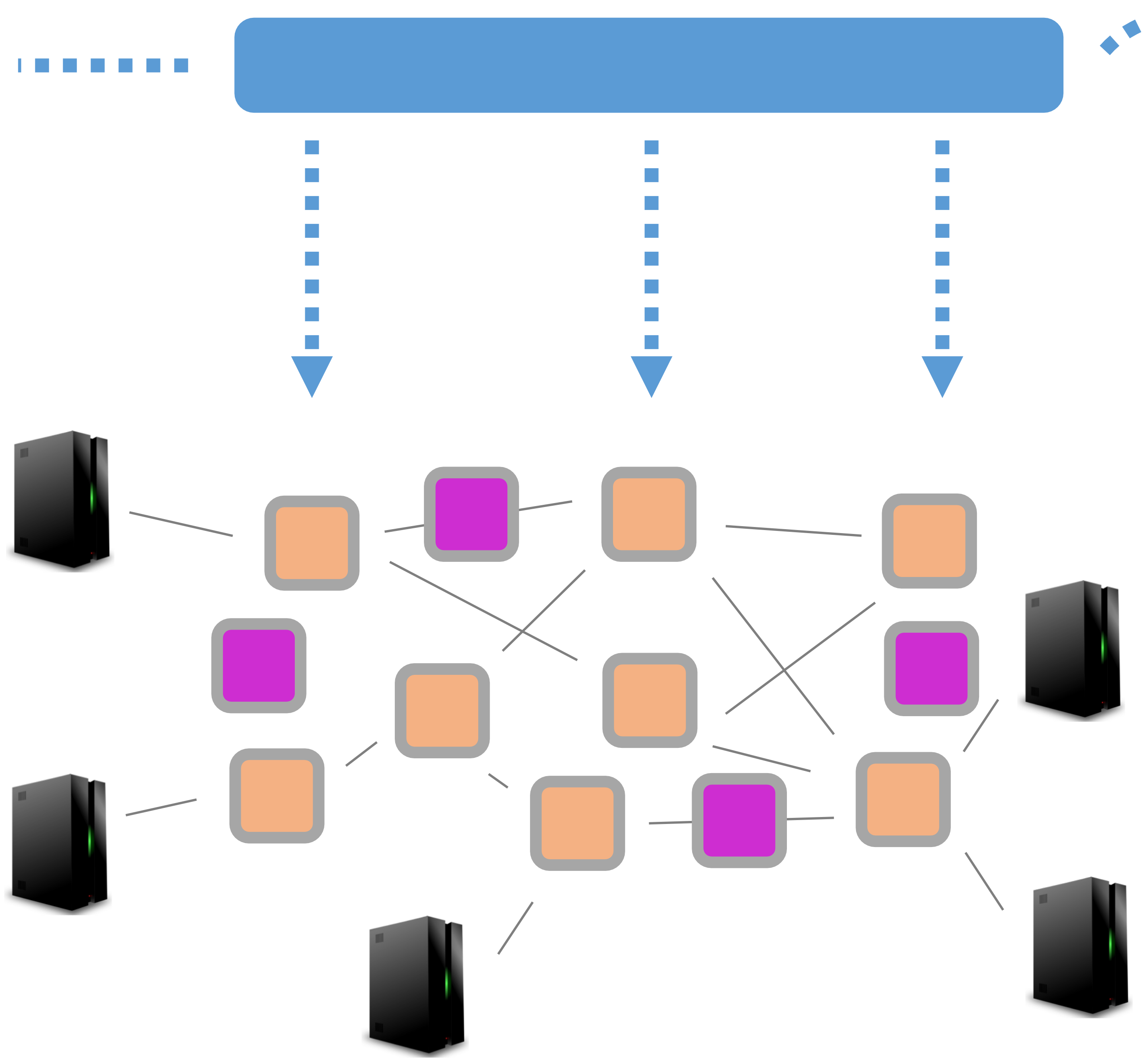
<u>In traditional networks</u>

- Model the configurations
- Model the environment
  - possible faults
  - possible external messages
- Verify path properties of *all* data planes produced



Examples

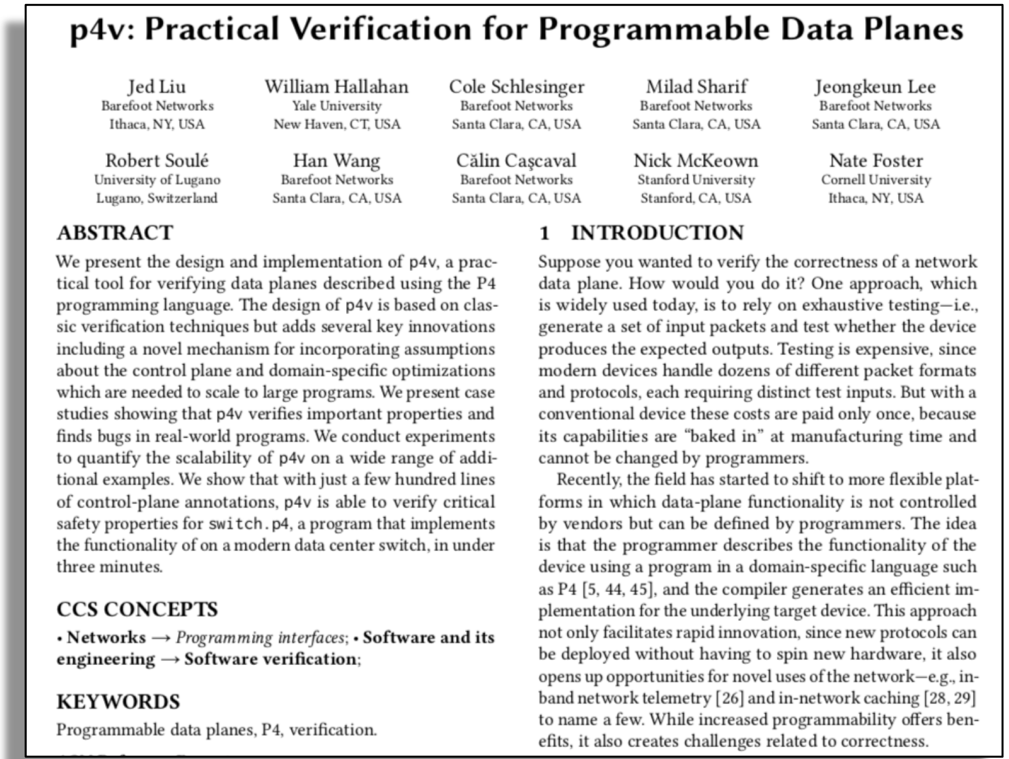
- ARC [SIGCOMM '16]
- ERA [OSDI '16]
- MineSweeper [SIGCOMM '17]
- Scaling:
  - depends on the policy, topology, protocols, properties



Middleboxes

- Middleboxes have more complex computing demands and are often stateful
- Much more research required but see, eg, [NSDI '17]

@ SIGCOMM 2018



p4v: Practical Verification for Programmable Data Planes

Jed Liu
Barefoot Networks
Ithaca, NY, USA

William Hallahan
Yale University
New Haven, CT, USA

Cole Schlesinger
Barefoot Networks
Santa Clara, CA, USA

Milad Sharif
Barefoot Networks
Santa Clara, CA, USA

Jeongkeun Lee
Barefoot Networks
Santa Clara, CA, USA

Robert Soulé
University of Lugano
Lugano, Switzerland

Han Wang
Barefoot Networks
Santa Clara, CA, USA

Călin Caşcaval
Barefoot Networks
Santa Clara, CA, USA

Nick McKeown
Stanford University
Stanford, CA, USA

Nate Foster
Cornell University
Ithaca, NY, USA

ABSTRACT

We present the design and implementation of p4v, a practical tool for verifying data planes described using the P4 programming language. The design of p4v is based on classic verification techniques but adds several key innovations including a novel mechanism for incorporating assumptions about the control plane and domain-specific optimizations which are needed to scale to large programs. We present case studies showing that p4v verifies important properties and finds bugs in real-world programs. We conduct experiments to quantify the scalability of p4v on a wide range of additional examples. We show that with just a few hundred lines of control-plane annotations, p4v is able to verify critical safety properties for `switch.p4`, a program that implements the functionality of on a modern data center switch, in under three minutes.

CCS CONCEPTS

• **Networks** → *Programming interfaces*; • **Software and its engineering** → **Software verification**;

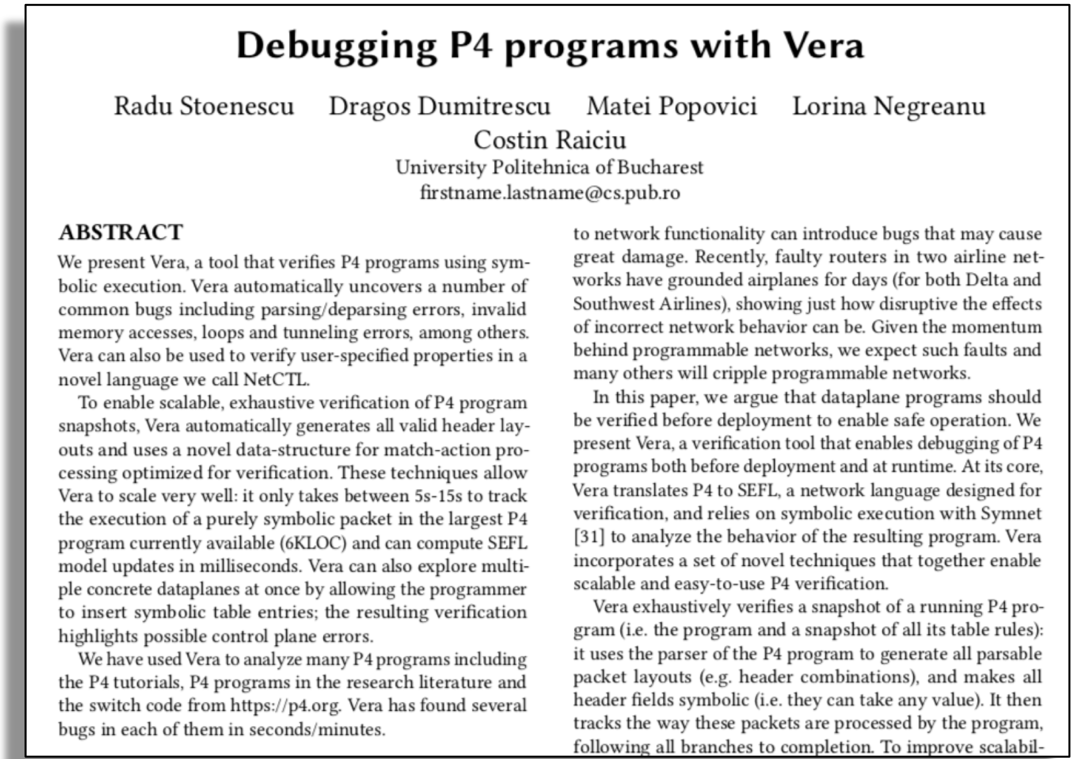
KEYWORDS

Programmable data planes, P4, verification.

1 INTRODUCTION

Suppose you wanted to verify the correctness of a network data plane. How would you do it? One approach, which is widely used today, is to rely on exhaustive testing—i.e., generate a set of input packets and test whether the device produces the expected outputs. Testing is expensive, since modern devices handle dozens of different packet formats and protocols, each requiring distinct test inputs. But with a conventional device these costs are paid only once, because its capabilities are "baked in" at manufacturing time and cannot be changed by programmers.

Recently, the field has started to shift to more flexible platforms in which data-plane functionality is not controlled by vendors but can be defined by programmers. The idea is that the programmer describes the functionality of the device using a program in a domain-specific language such as P4 [5, 44, 45], and the compiler generates an efficient implementation for the underlying target device. This approach not only facilitates rapid innovation, since new protocols can be deployed without having to spin new hardware, it also opens up opportunities for novel uses of the network—e.g., in-band network telemetry [26] and in-network caching [28, 29] to name a few. While increased programmability offers benefits, it also creates challenges related to correctness.



Debugging P4 programs with Vera

Radu Stoenescu Dragos Dumitrescu Matei Popovici Lorina Negreanu
Costin Raiciu
University Politehnica of Bucharest
firstname.lastname@cs.pub.ro

ABSTRACT

We present Vera, a tool that verifies P4 programs using symbolic execution. Vera automatically uncovers a number of common bugs including parsing/deparsing errors, invalid memory accesses, loops and tunneling errors, among others. Vera can also be used to verify user-specified properties in a novel language we call NetCTL.

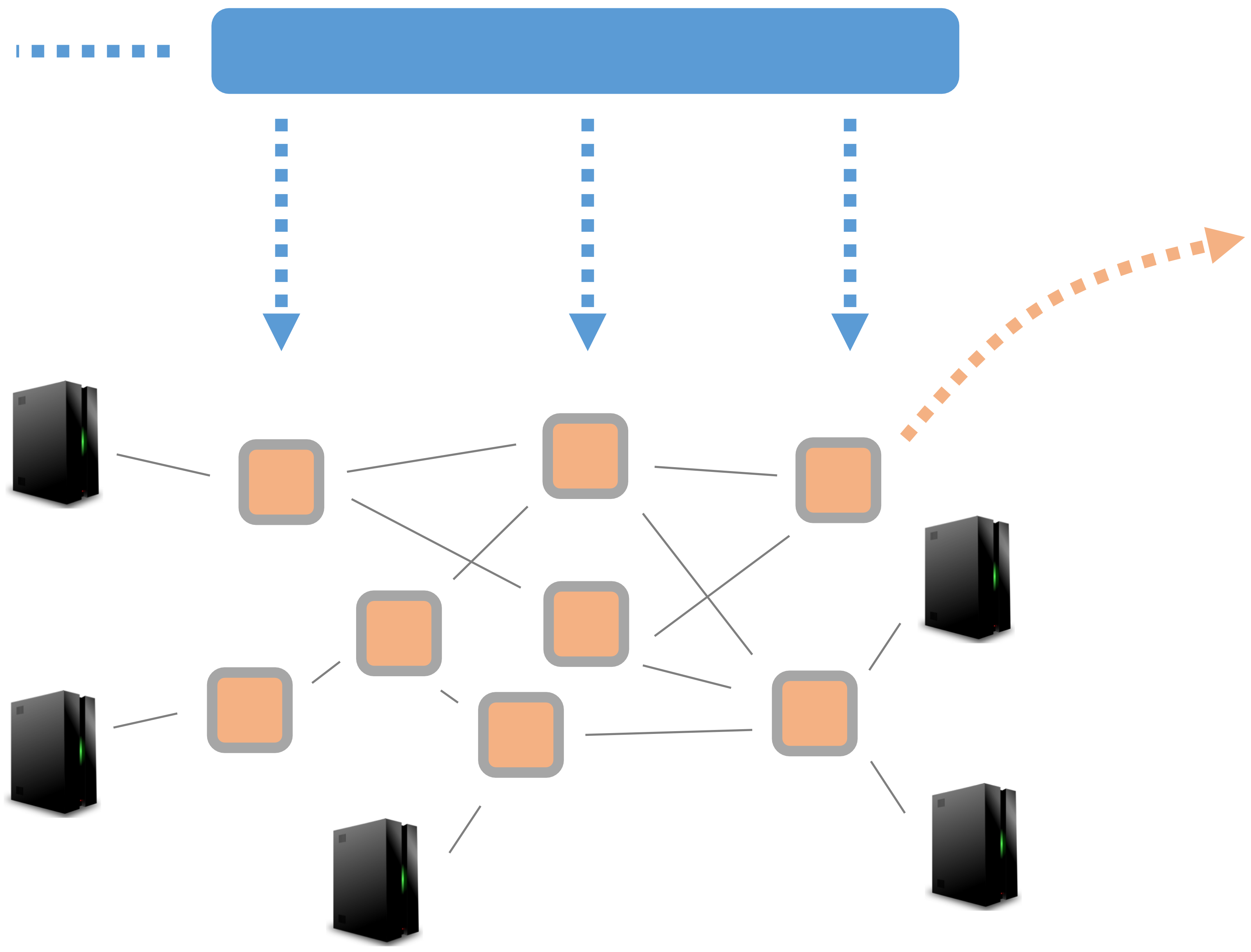
To enable scalable, exhaustive verification of P4 program snapshots, Vera automatically generates all valid header layouts and uses a novel data-structure for match-action processing optimized for verification. These techniques allow Vera to scale very well: it only takes between 5s-15s to track the execution of a purely symbolic packet in the largest P4 program currently available (6KLOC) and can compute SEFL model updates in milliseconds. Vera can also explore multiple concrete dataplanes at once by allowing the programmer to insert symbolic table entries; the resulting verification highlights possible control plane errors.

We have used Vera to analyze many P4 programs including the P4 tutorials, P4 programs in the research literature and the switch code from https://p4.org. Vera has found several bugs in each of them in seconds/minutes.

to network functionality can introduce bugs that may cause great damage. Recently, faulty routers in two airline networks have grounded airplanes for days (for both Delta and Southwest Airlines), showing just how disruptive the effects of incorrect network behavior can be. Given the momentum behind programmable networks, we expect such faults and many others will cripple programmable networks.

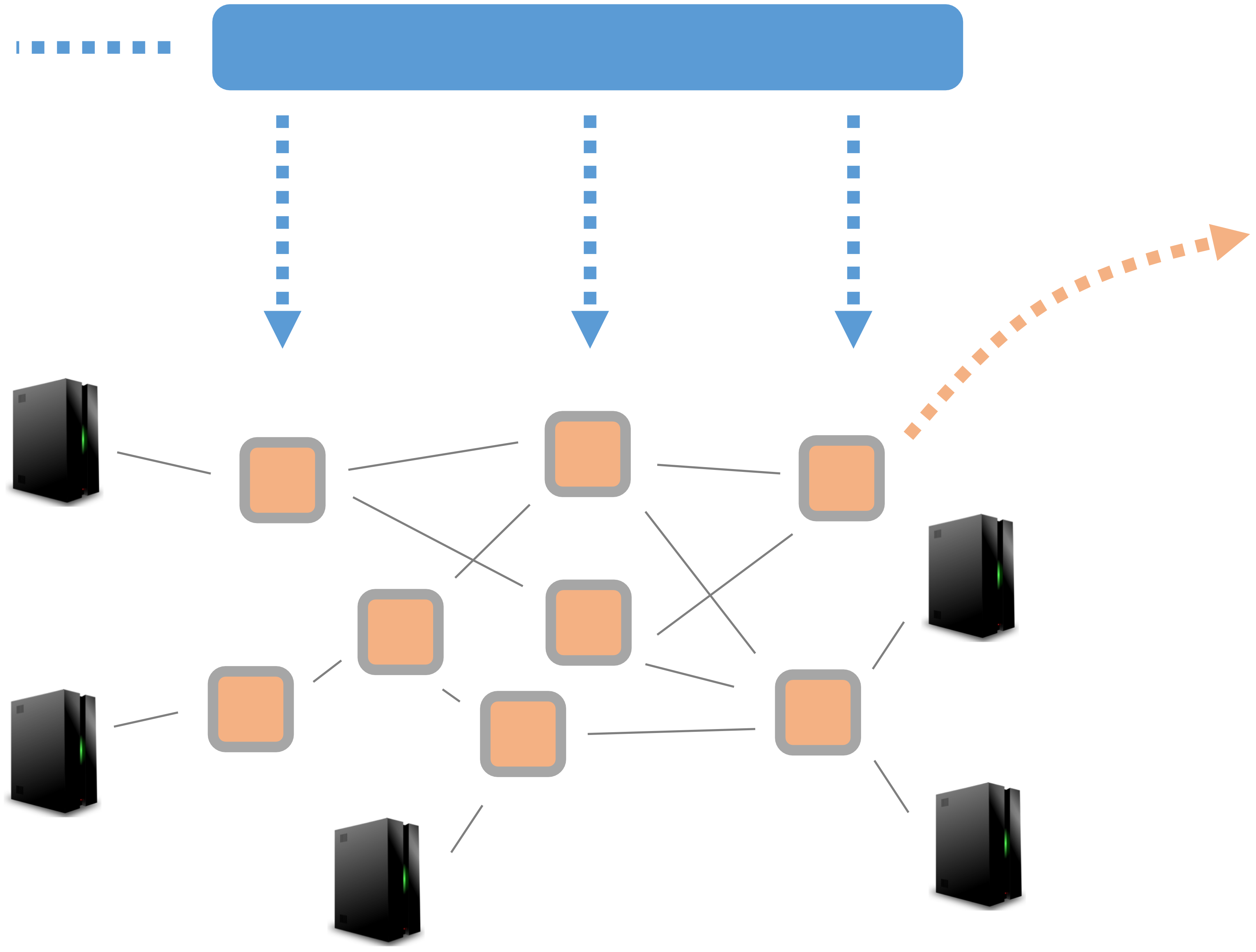
In this paper, we argue that dataplane programs should be verified before deployment to enable safe operation. We present Vera, a verification tool that enables debugging of P4 programs both before deployment and at runtime. At its core, Vera translates P4 to SEFL, a network language designed for verification, and relies on symbolic execution with Symnet [31] to analyze the behavior of the resulting program. Vera incorporates a set of novel techniques that together enable scalable and easy-to-use P4 verification.

Vera exhaustively verifies a snapshot of a running P4 program (i.e. the program and a snapshot of all its table rules): it uses the parser of the P4 program to generate all parsable packet layouts (e.g. header combinations), and makes all header fields symbolic (i.e. they can take any value). It then tracks the way these packets are processed by the program, following all branches to completion. To improve scalabil-



<u>Verifying P4</u>

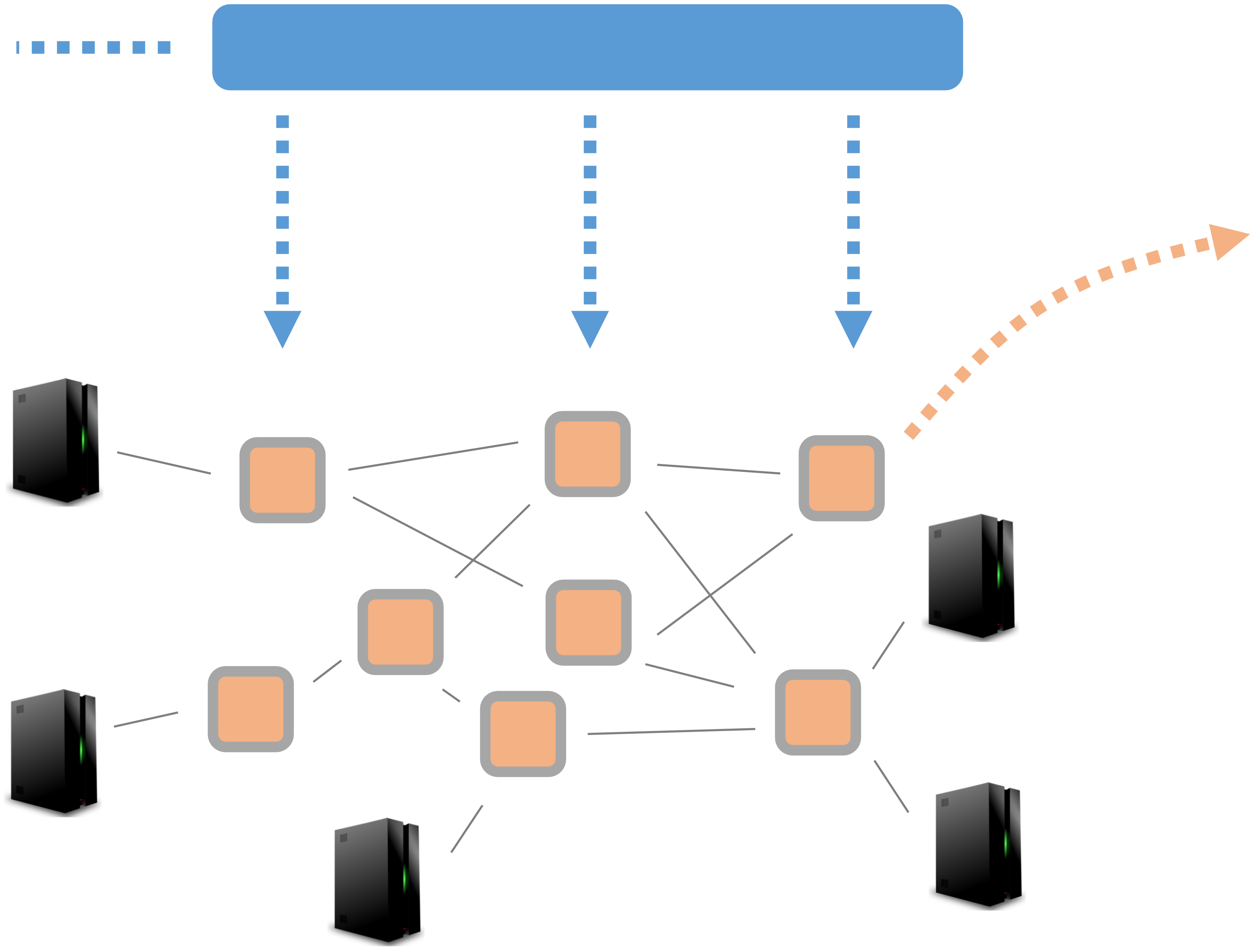
Is a P4 program a data plane program or a control plane program?



<u>Verifying P4</u>

It is a bit of both!

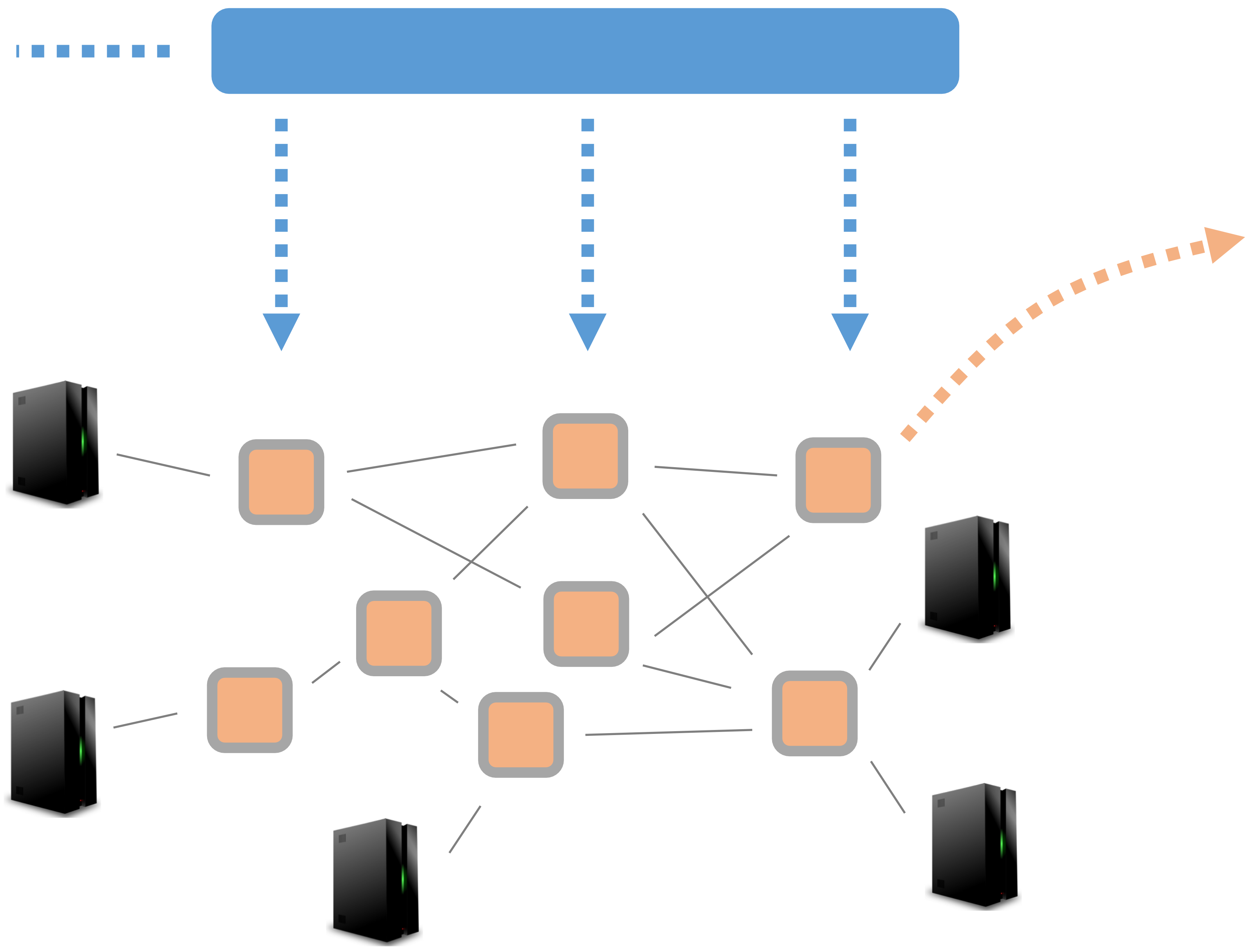
P4 programs connect a series of data plane match-action tables together. But those tables are populated by the control plane during execution.



Verifying P4

One key problem involves dealing with the unknown table contents

A second problem involves scaling analysis to complex P4 programs and networks.



Primary Techniques

P4v: Verification Condition Generation

Vera: Symbolic Execution

Learning More

Weakest Preconditions/Hoare Logic

- C. A. R. Hoare. ***An Axiomatic Basis for Computer Programming.*** CACM 12, 10. 1969.
- Edsger W. Dijkstra. ***Guarded Commands, Nondeterminacy, and Formal Derivation of Programs.*** CACM 18, 8. 1975.
- Cormac Flanagan and James B. Saxe. ***Avoiding Exponential Explosion: Generating Compact Verification Conditions.*** POPL 2001.
- Microsoft Research. Dafny tutorial. https://rise4fun.com/Dafny/tutorial

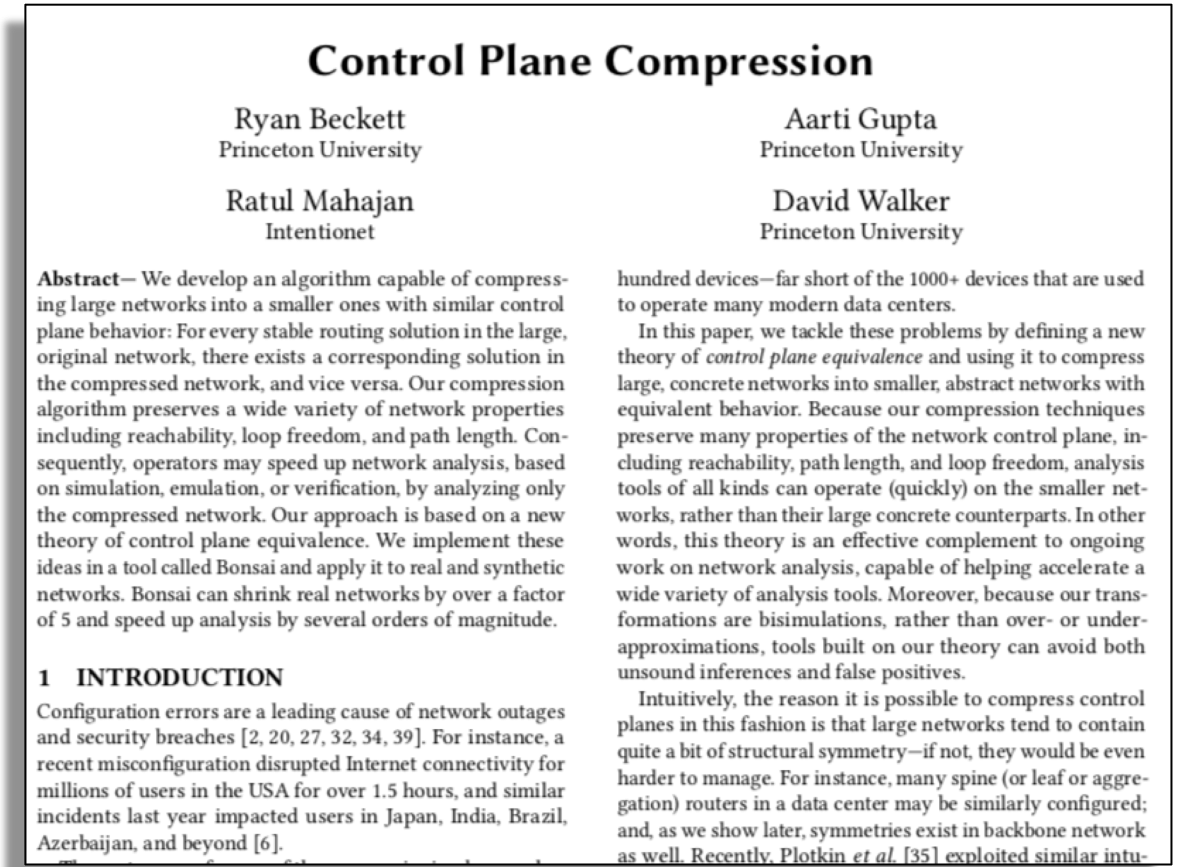
Symbolic Execution

- J. C. King. ***Symbolic execution and program testing.*** CACM, 19(7). 1976.
- Christian Cadar, Daniel Dunbar, Dawson Engler. ***KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems*** Programs. OSDI 2008. See also: http://klee.github.io/publications/

A Comparison:

- Ioannis T. Kassios, Peter Müller, and Malte Schwerhoff. ***Comparing Verification Condition Generation with Symbolic Execution: An Experience Report.*** VSTTE 2012.

@ SIGCOMM 2018



Control Plane Compression

Ryan Beckett
Princeton University

Aarti Gupta
Princeton University

Ratul Mahajan
Intentionet

David Walker
Princeton University

Abstract— We develop an algorithm capable of compressing large networks into a smaller ones with similar control plane behavior: For every stable routing solution in the large, original network, there exists a corresponding solution in the compressed network, and vice versa. Our compression algorithm preserves a wide variety of network properties including reachability, loop freedom, and path length. Consequently, operators may speed up network analysis, based on simulation, emulation, or verification, by analyzing only the compressed network. Our approach is based on a new theory of control plane equivalence. We implement these ideas in a tool called Bonsai and apply it to real and synthetic networks. Bonsai can shrink real networks by over a factor of 5 and speed up analysis by several orders of magnitude.

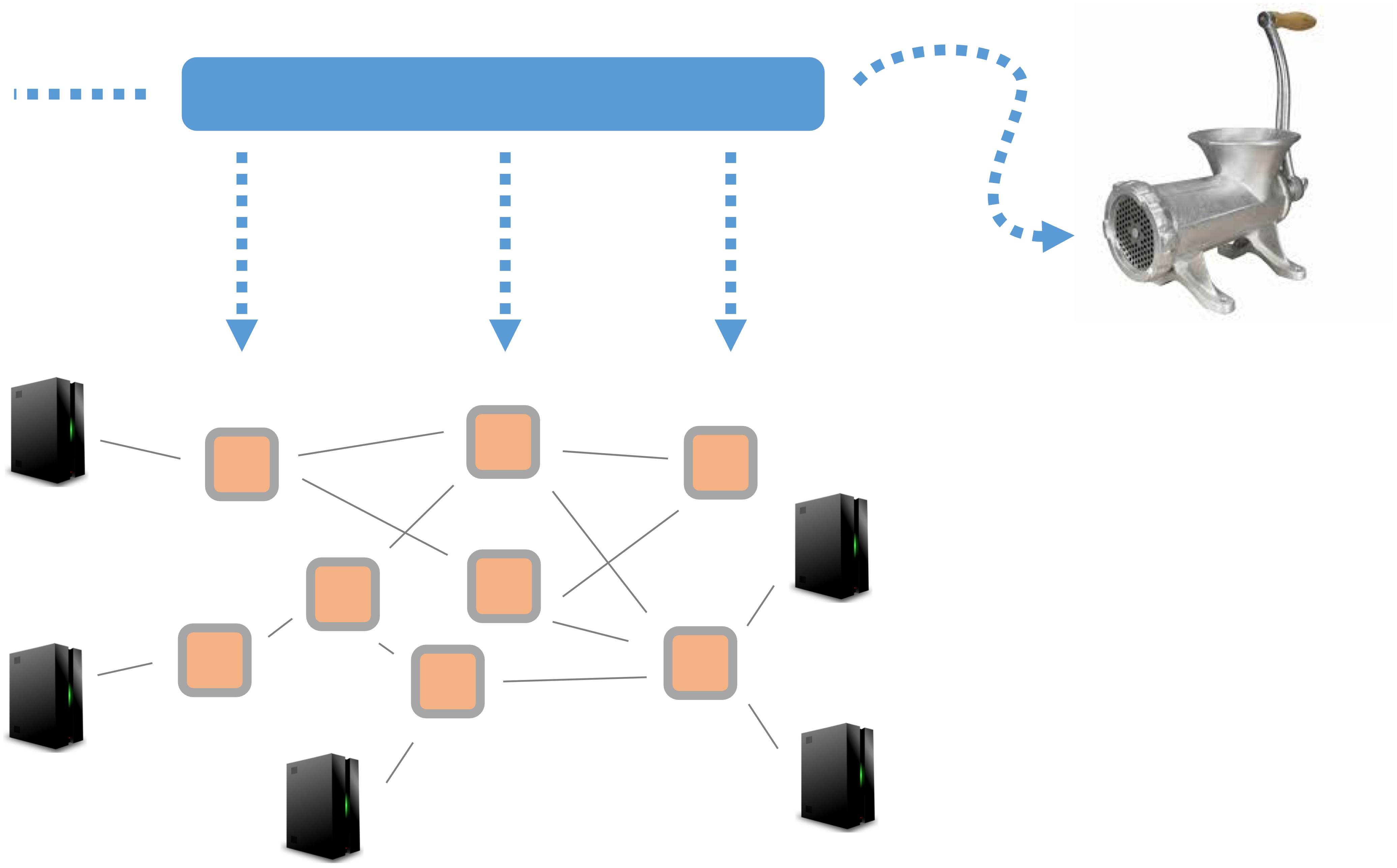
1 INTRODUCTION

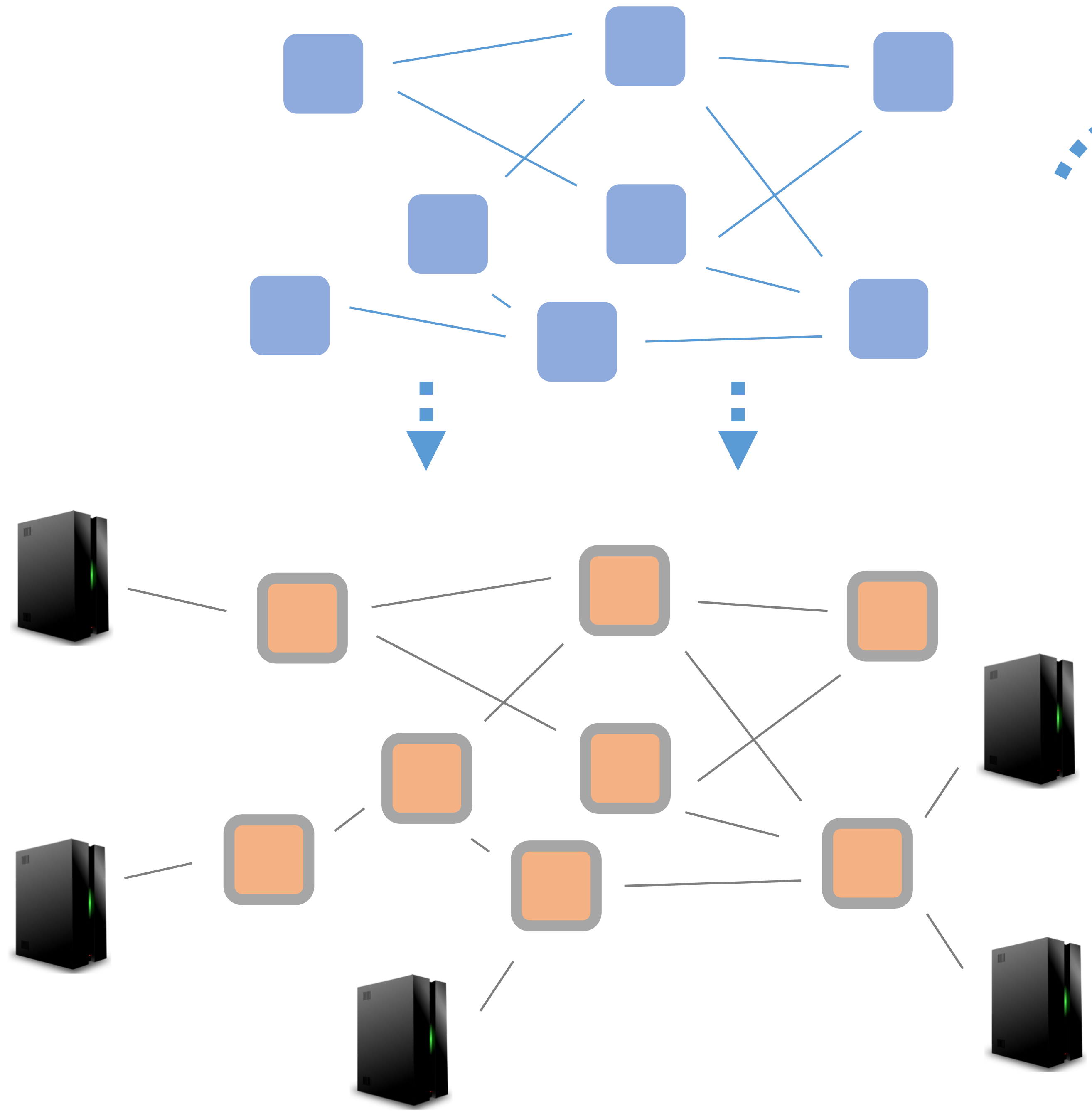
Configuration errors are a leading cause of network outages and security breaches [2, 20, 27, 32, 34, 39]. For instance, a recent misconfiguration disrupted Internet connectivity for millions of users in the USA for over 1.5 hours, and similar incidents last year impacted users in Japan, India, Brazil, Azerbaijan, and beyond [6].

hundred devices—far short of the 1000+ devices that are used to operate many modern data centers.

In this paper, we tackle these problems by defining a new theory of *control plane equivalence* and using it to compress large, concrete networks into smaller, abstract networks with equivalent behavior. Because our compression techniques preserve many properties of the network control plane, including reachability, path length, and loop freedom, analysis tools of all kinds can operate (quickly) on the smaller networks, rather than their large concrete counterparts. In other words, this theory is an effective complement to ongoing work on network analysis, capable of helping accelerate a wide variety of analysis tools. Moreover, because our transformations are bisimulations, rather than over- or under-approximations, tools built on our theory can avoid both unsound inferences and false positives.

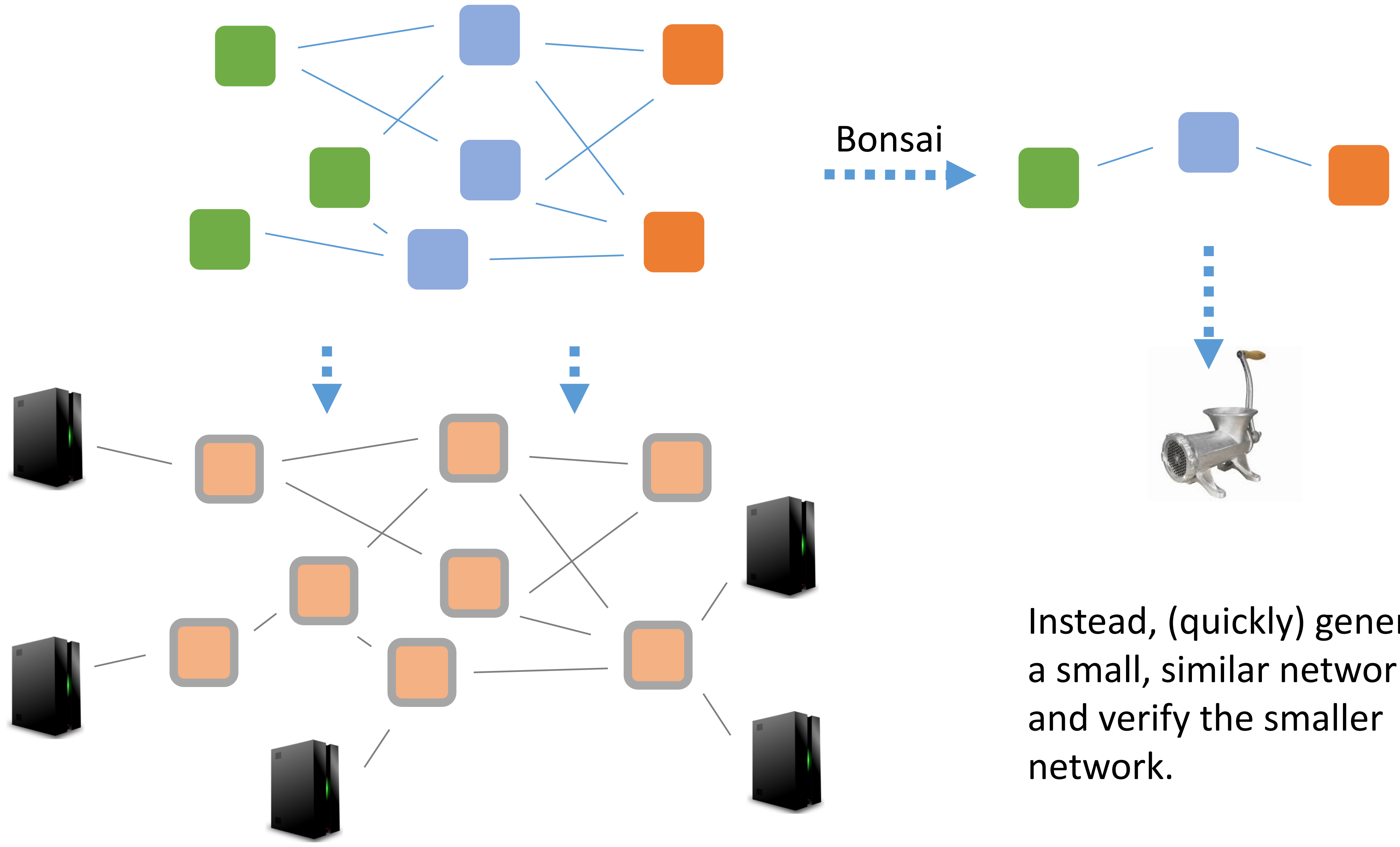
Intuitively, the reason it is possible to compress control planes in this fashion is that large networks tend to contain quite a bit of structural symmetry—if not, they would be even harder to manage. For instance, many spine (or leaf or aggregation) routers in a data center may be similarly configured; and, as we show later, symmetries exist in backbone network as well. Recently, Plotkin *et al.* [35] exploited similar intu-





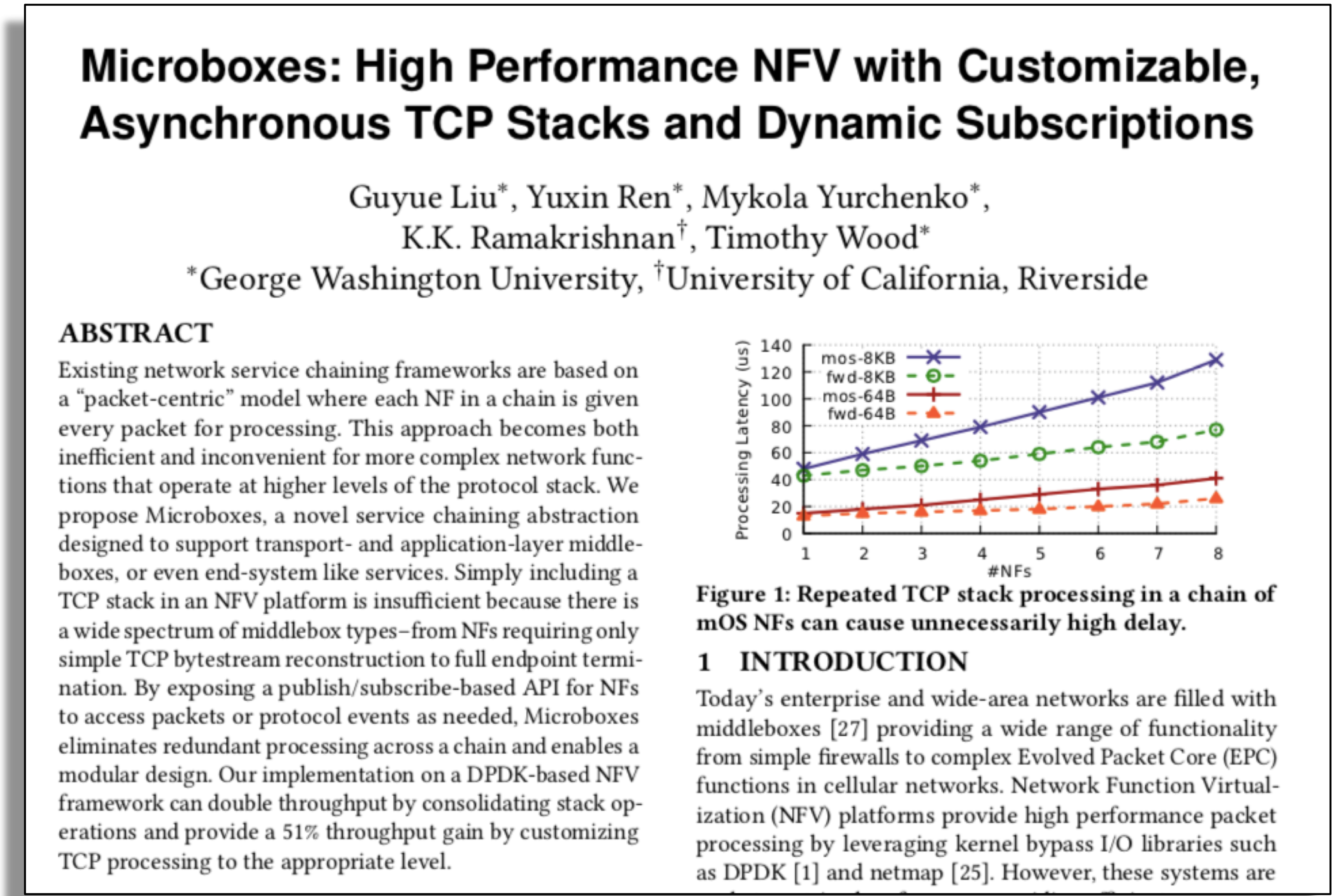
Verifying big, complicated control planes is costly.

Most CP verification techniques scale non-linearly, some exponentially in the worst case



Instead, (quickly) generate a small, similar network and verify the smaller network.

@ SIGCOMM 2018



Microboxes: High Performance NFV with Customizable, Asynchronous TCP Stacks and Dynamic Subscriptions

Guyue Liu*, Yuxin Ren*, Mykola Yurchenko*,
K.K. Ramakrishnan†, Timothy Wood*
*George Washington University, †University of California, Riverside

ABSTRACT

Existing network service chaining frameworks are based on a "packet-centric" model where each NF in a chain is given every packet for processing. This approach becomes both inefficient and inconvenient for more complex network functions that operate at higher levels of the protocol stack. We propose Microboxes, a novel service chaining abstraction designed to support transport- and application-layer middleboxes, or even end-system like services. Simply including a TCP stack in an NFV platform is insufficient because there is a wide spectrum of middlebox types–from NFs requiring only simple TCP bytestream reconstruction to full endpoint termination. By exposing a publish/subscribe-based API for NFs to access packets or protocol events as needed, Microboxes eliminates redundant processing across a chain and enables a modular design. Our implementation on a DPDK-based NFV framework can double throughput by consolidating stack operations and provide a 51% throughput gain by customizing TCP processing to the appropriate level.

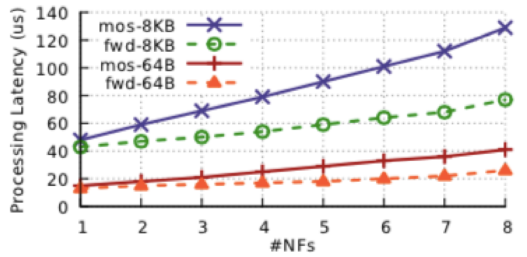
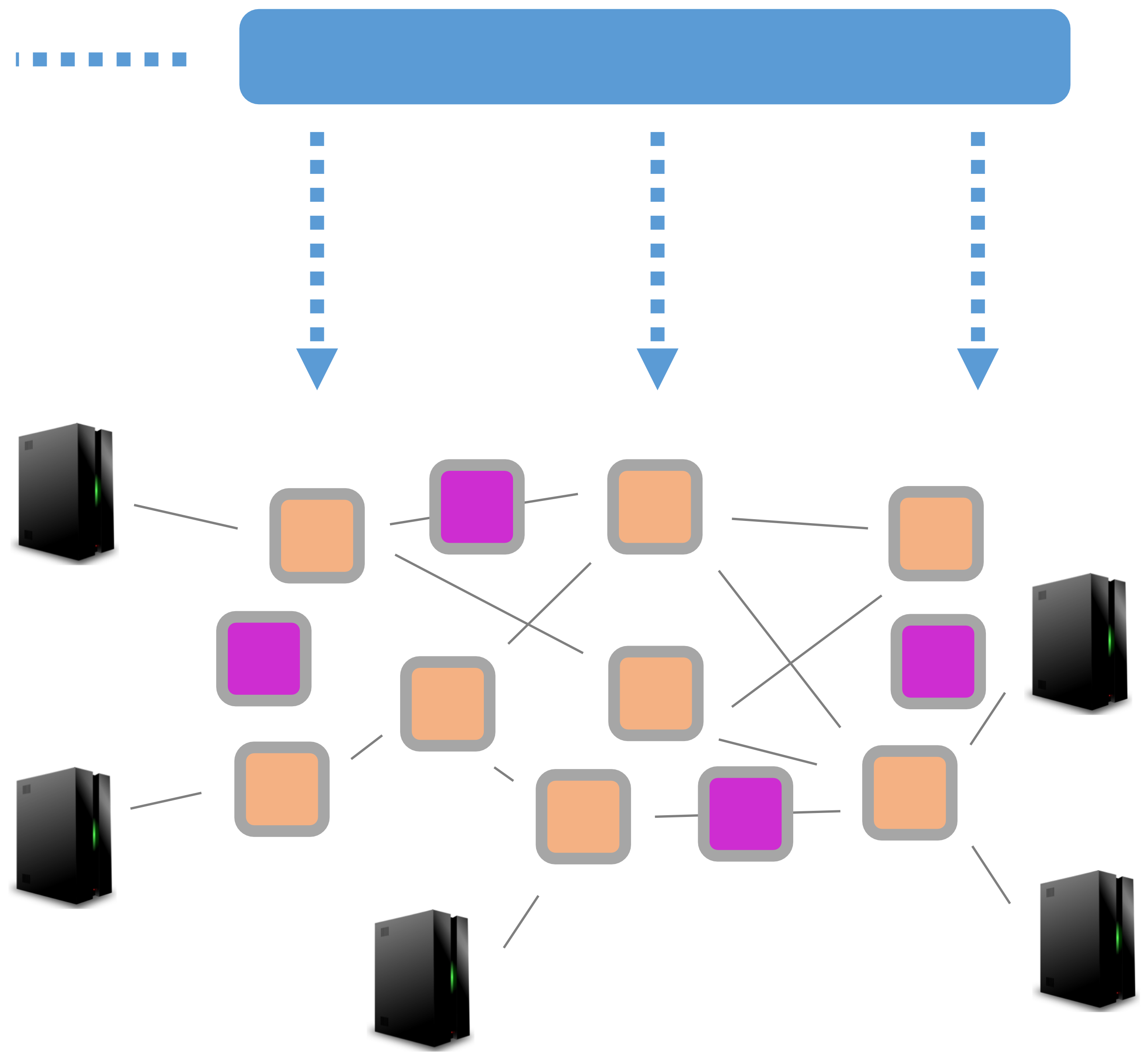


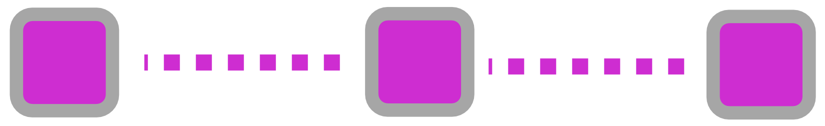
Figure 1: Repeated TCP stack processing in a chain of mOS NFs can cause unnecessarily high delay.

1 INTRODUCTION

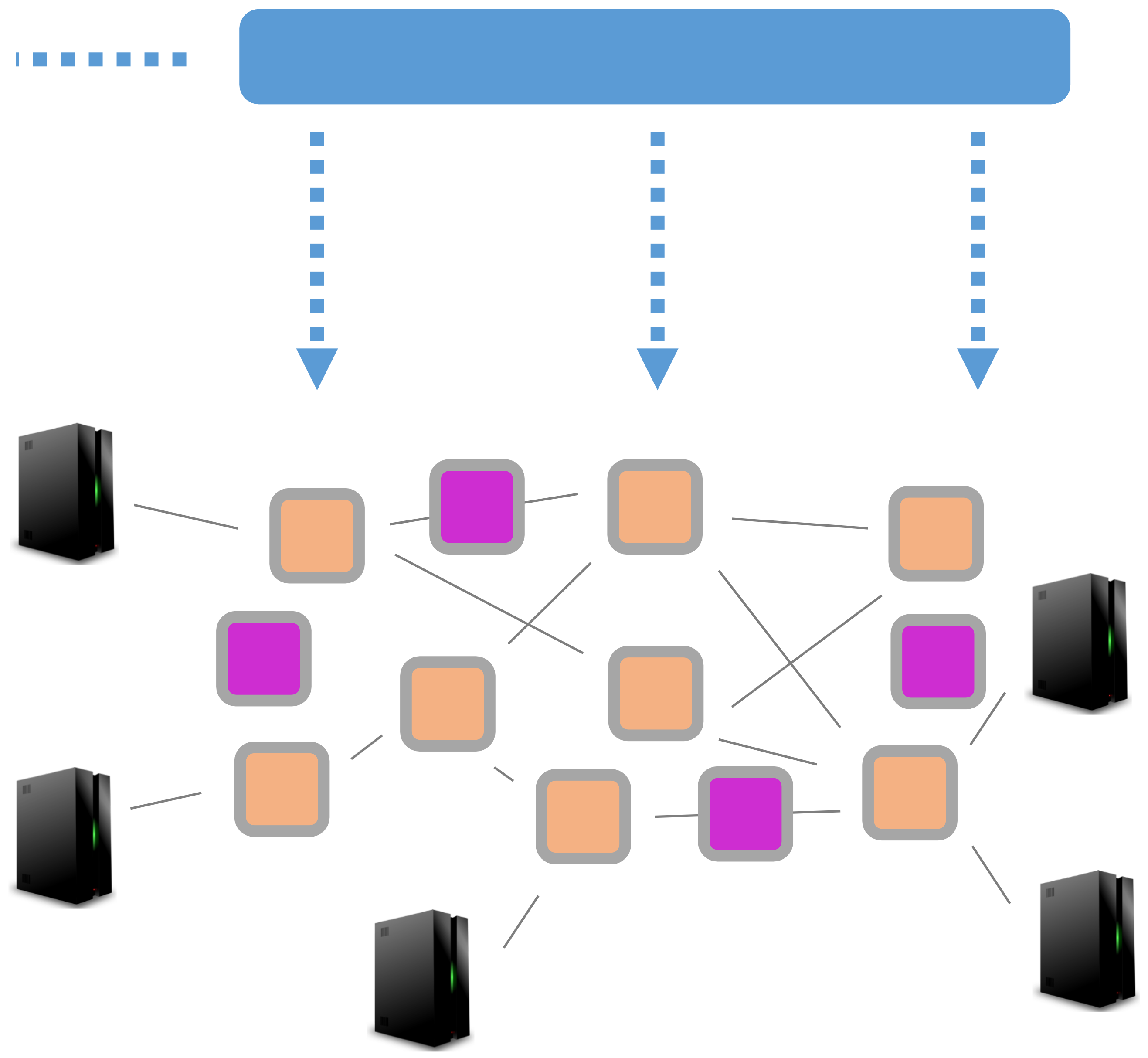
Today's enterprise and wide-area networks are filled with middleboxes [27] providing a wide range of functionality from simple firewalls to complex Evolved Packet Core (EPC) functions in cellular networks. Network Function Virtualization (NFV) platforms provide high performance packet processing by leveraging kernel bypass I/O libraries such as DPDK [1] and netmap [25]. However, these systems are



Observation:



It is costly to repeat TCP processing at each step in the chain.



<u>Solution:</u>

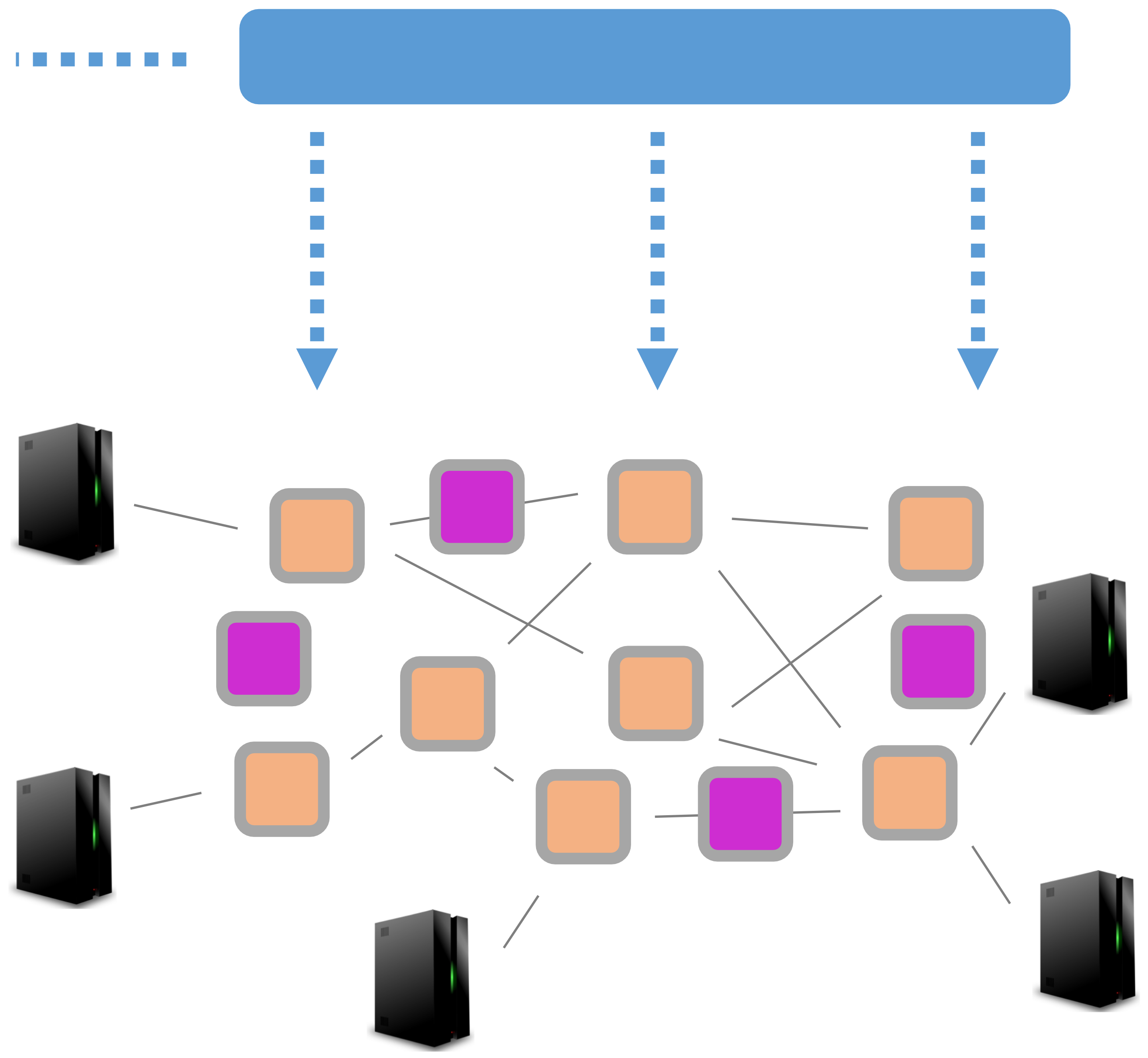
Microboxes: A new event-based architecture for middlebox construction.

<u>Primary contribution:</u>

Improved performance

<u>Challenges:</u>

Under what conditions is microbox parallelization safe?

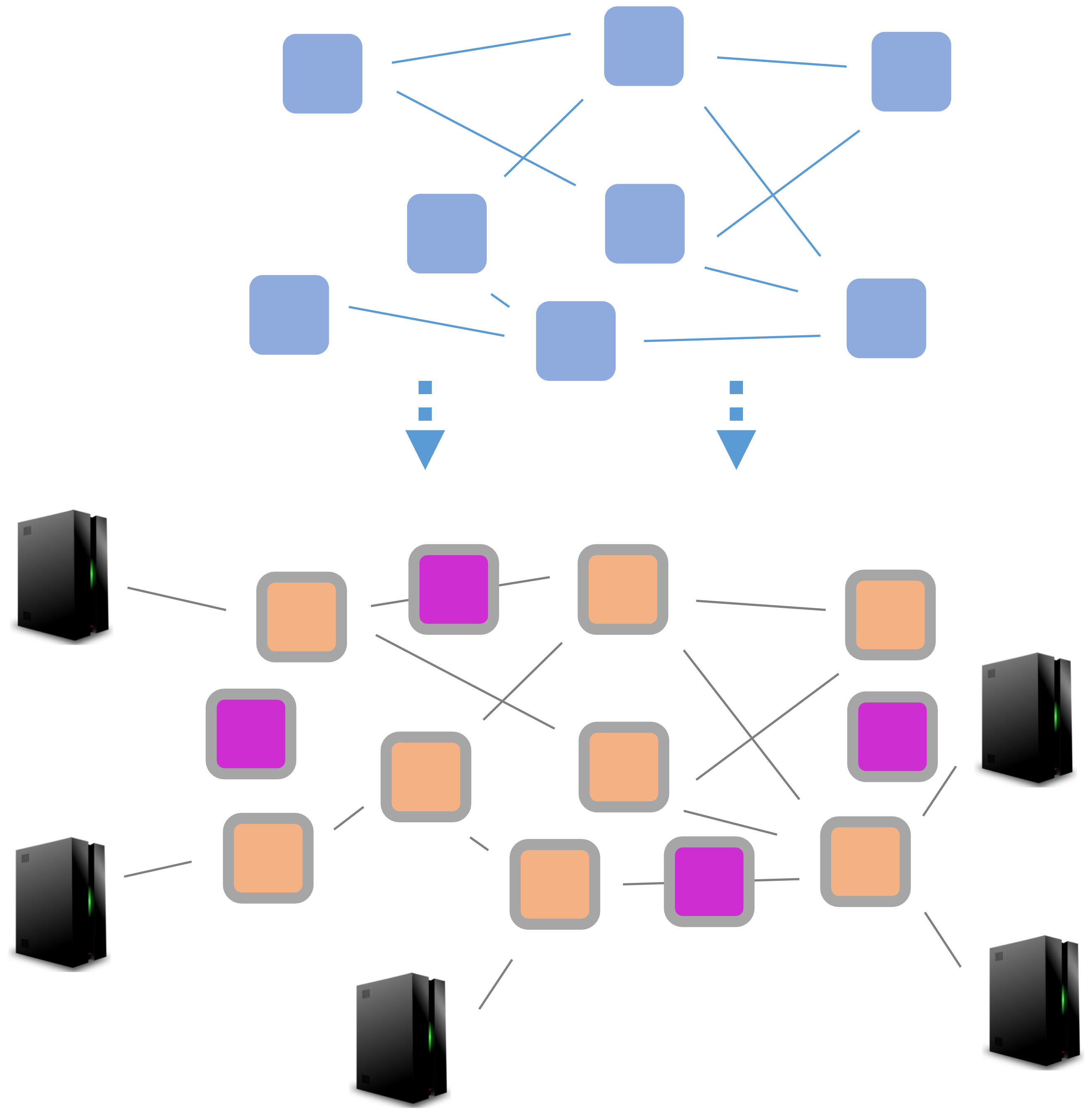


An Opportunity:

The microbox architecture
- simplifies middlebox programming
- helps identify common safety properties

Verification of microboxes may be more tractable than verification of general-purpose code.

General lesson: *Good programming environments provide constraints that simplify reasoning about programs.*



Have fun at the SIGCOMM Verification Session [2:10PM]