lecture 15:
virtualization with OpenFlow

5590: software defined networking

anduo wang, Temple University

TTLMAN 401B, R 17:30-20:00

OpenFlow revisit

OpenFlow aims …

infrastructure ossification

OpenFlow — a pragmatic comprise

- running experiments on heterogenous hardware
  - in a unified way, at line-rate, high port density
  - without vendors exposing their working internals

short-term question

- can we run experiments in campus network

alternatives

name-brand vendors expose open, programmable, virtualized platform

- switches, routers to deploy new protocols

open software platform

- performance
- a PC *NOT* support
  - # of ports needed for college wiring closet
  - packet processing: closet switch 100Gbits/s v.s. PC 1Gbit/s

OpenFlow aims …

short-term question

- *can we run experiments in campus network?*
  - commercial solution: closed, inflexible
  - research solution: insufficient performance, expensive

OF goals

- amenable to high-performance, low-cost implementation
- support a board range of research
- isolate experimental traffic from production traffic
- vendor's need for closed platform

using OpenFlow

experiments in production network

more examples

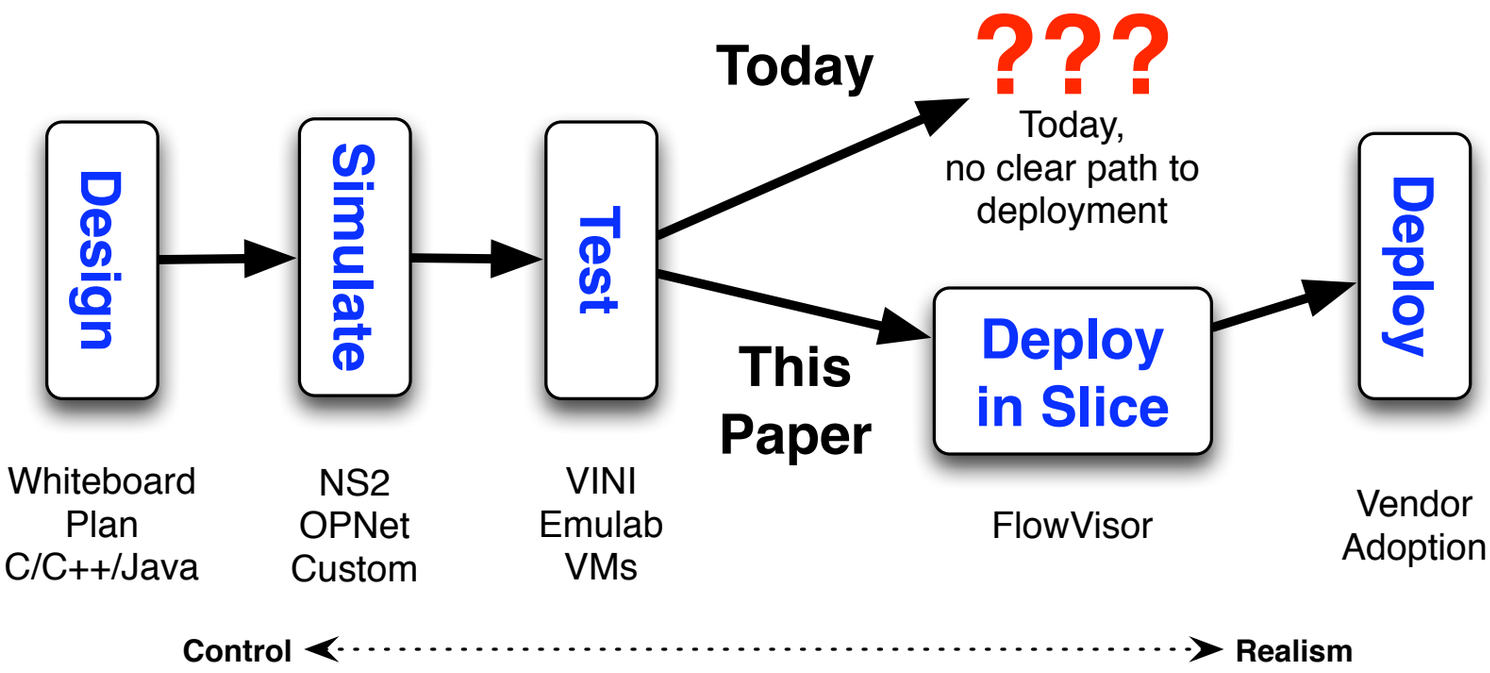
- ACL
- VLANs
- VOIP (mobile wireless clients): seamless handoff
- non-IP
- processing packets (NetPGA)

from OpenFlow to FlowVisor

feature	system	techniques
run experiments on heterogeneous devices in a uniform way	OpenFlow	abstraction, central controller
transparently run multiple experiments in isolated slices	FlowVisor	control message inspecting, rewriting

FlowVisor

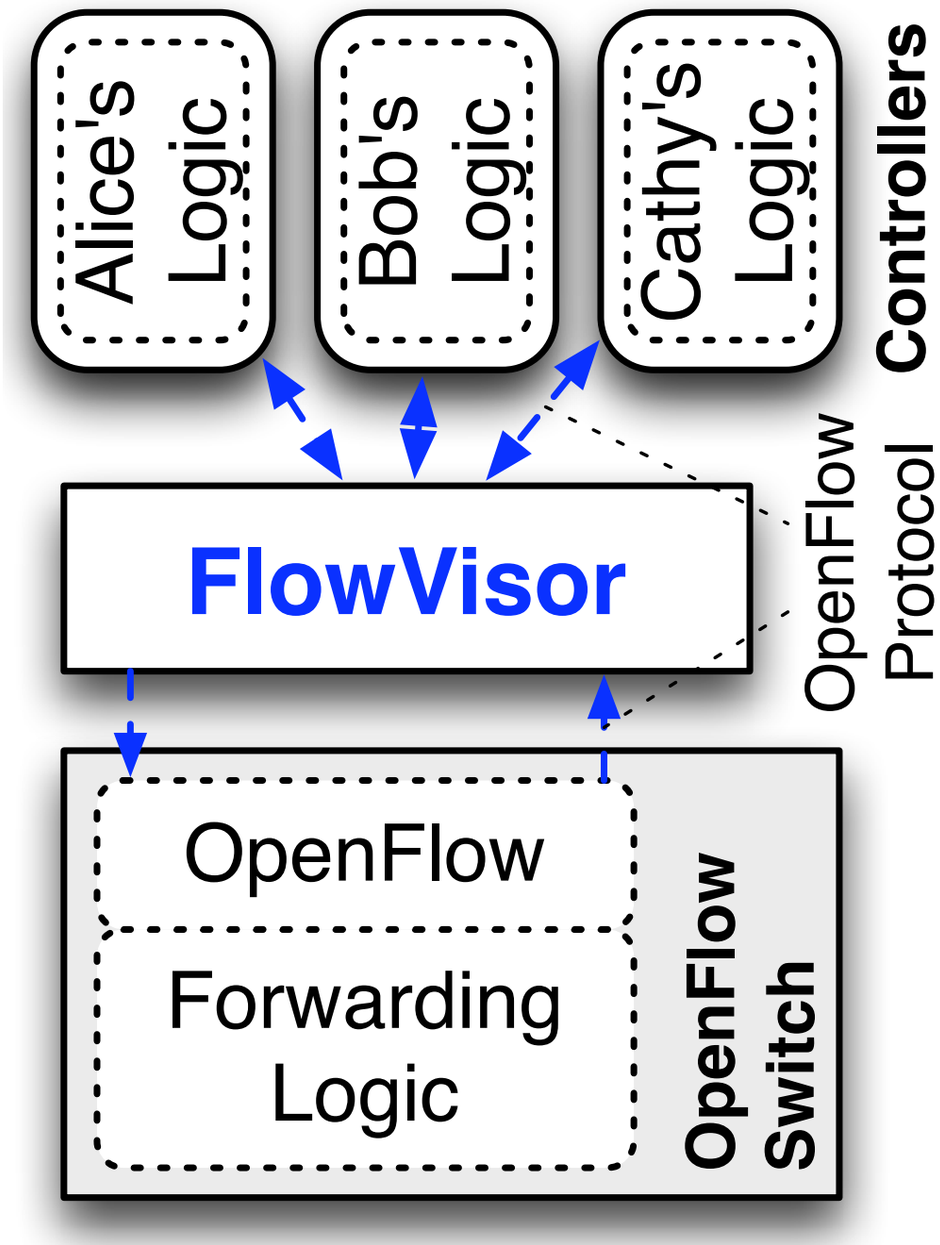
from OpenFlow to FlowVisor



idea

- unmodified hardware supports basic OpenFlow primitives
- a worldwide testbed into
  - extend all the way to the end user
  - carry real user traffic

FlowVisor overview

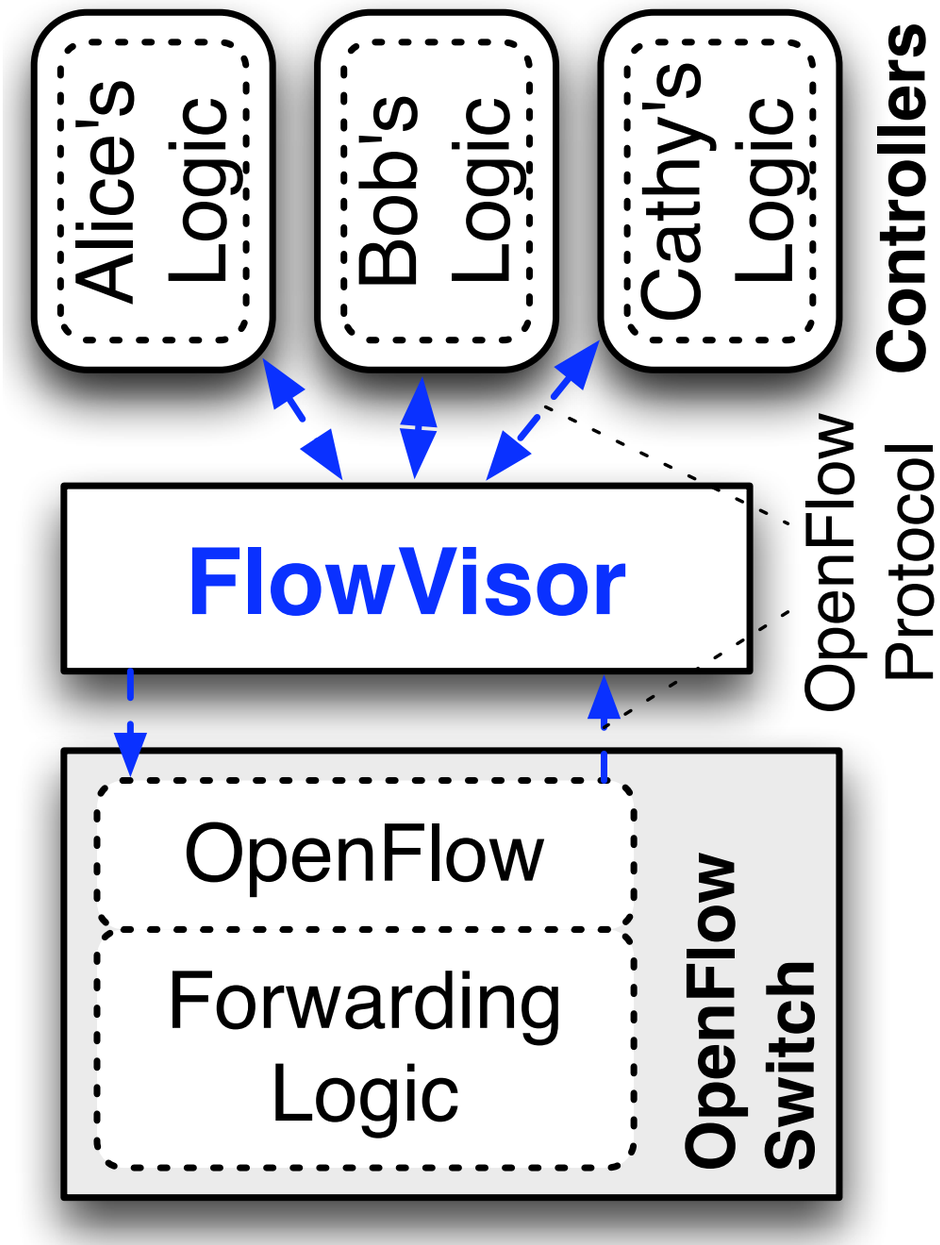


Sliced OpenFlow Switch Architecture

idea

- slicing network hardware
- a layer/proxy between data/ control planes

FlowVisor overview

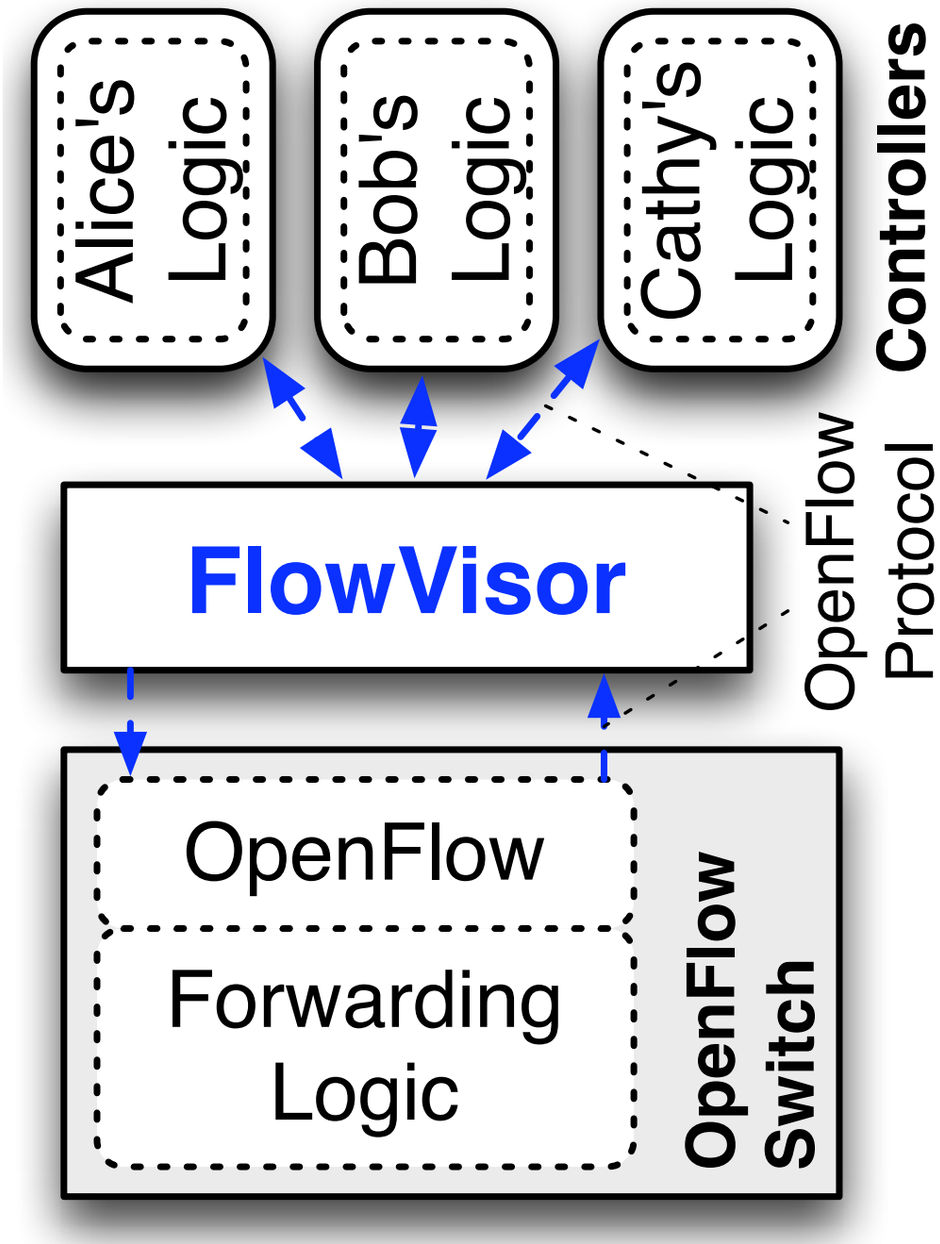


Sliced OpenFlow Switch Architecture

goals

- speed
  - packet processing and forwarding
- scale
- technology transfer
  - takes effort to transfer to specialized hardware

FlowVisor overview



Sliced OpenFlow Switch Architecture

modern switches today

- implements flow tables — TCAMs
- OpenFlow compliant via firmware upgrade

FlowVisor overview

assumes data/control plane separation

slice policy

- resource (topology, bandwidth, CPU)
- traffic (flowspace)

|

FlowVisor

slices communication between data/control planes

↓

multiple isolated control planes

- each belongs to a slice

slice policy

network resources

- topology
- bandwidth, forwarding tables
  - prevent one slice from starving another
- device CPU
  - switches can stop forwarding slow-path packets, processing updates …

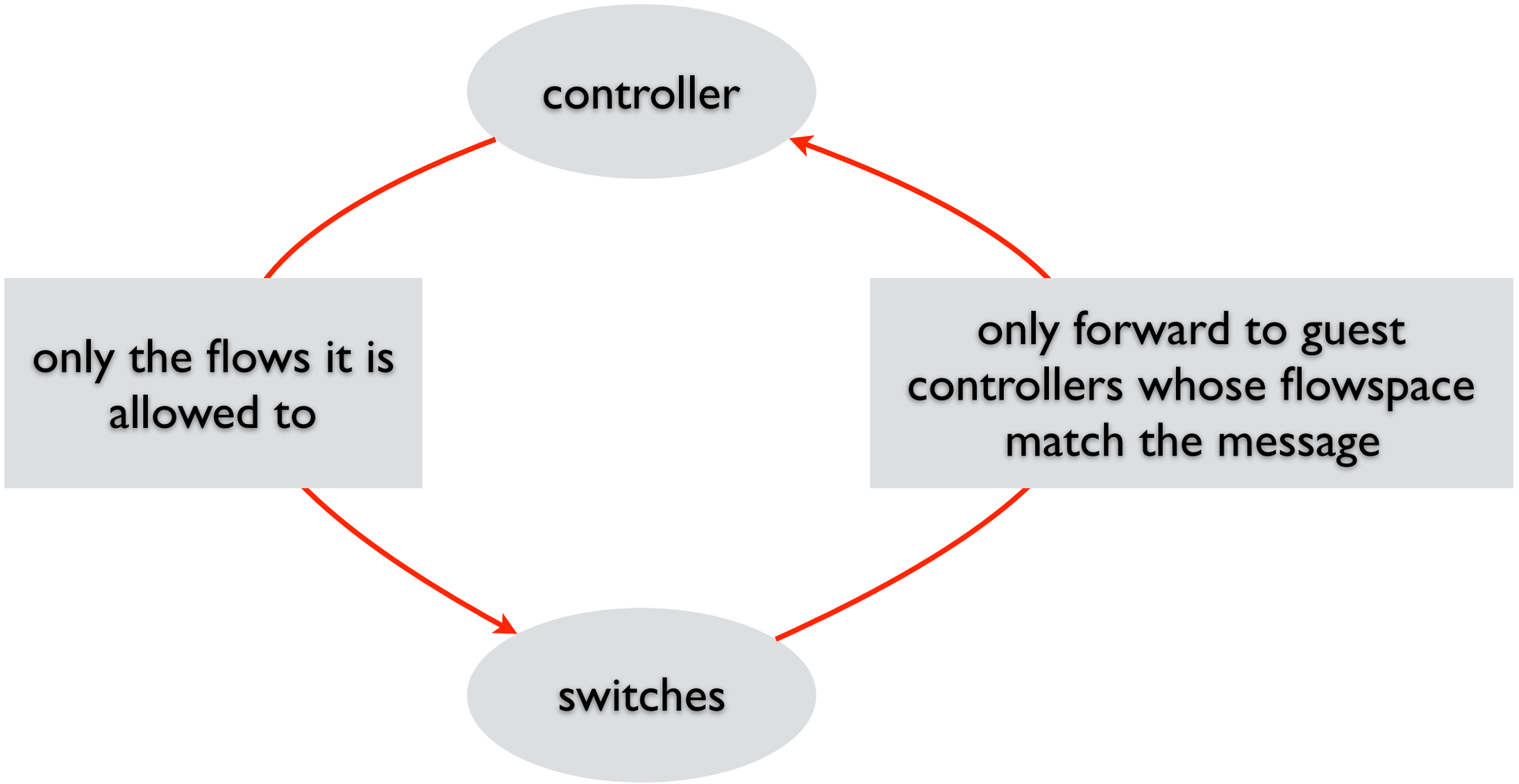
flowspace

- packet match pattern; action
- e.g.,

```
allow: tcp-port: 80 and ip = user_ip
```

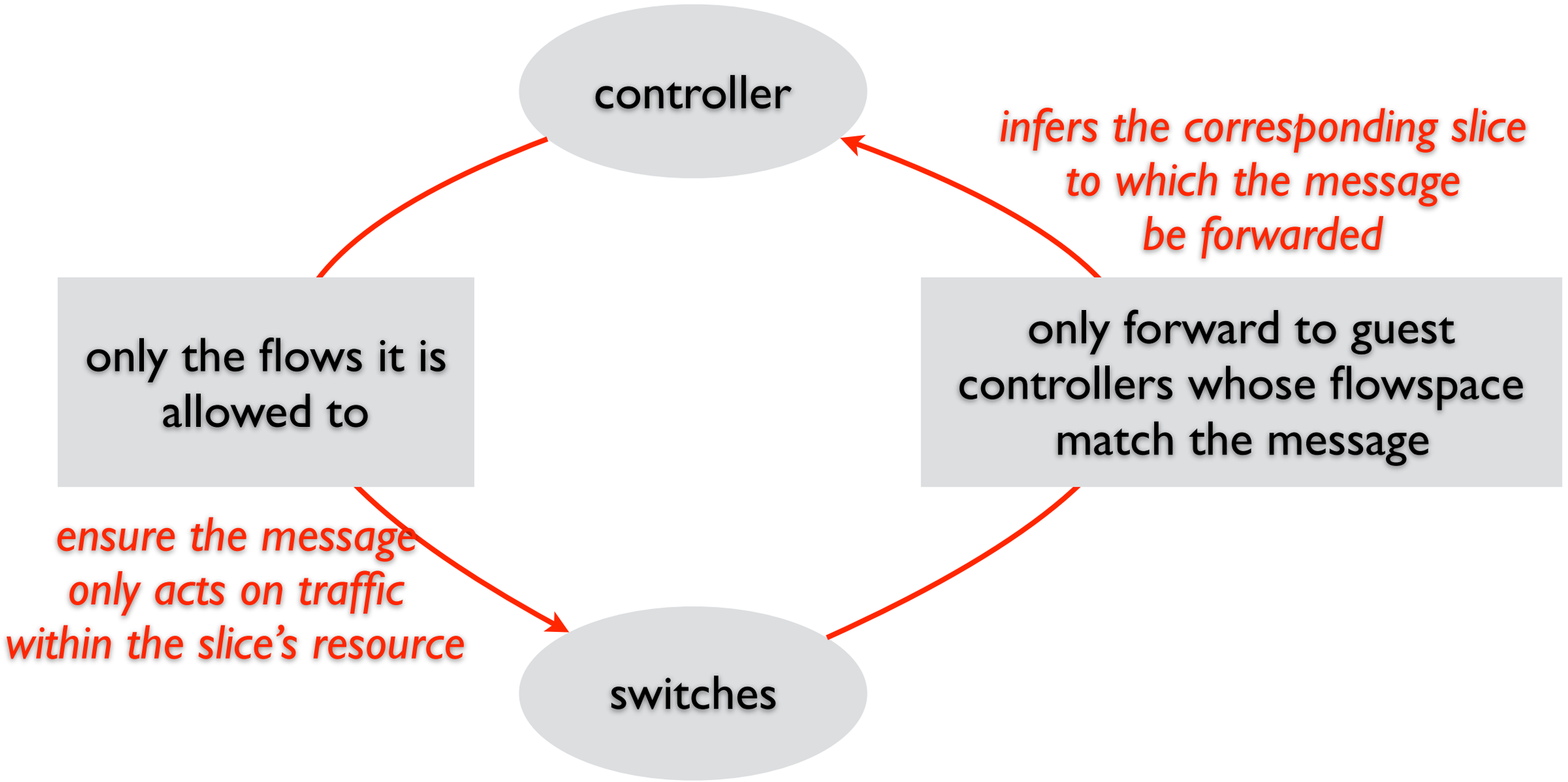
enforce slice policy

inspect, rewrite, and police OF messages as they pass



enforce slice policy

inspect, rewrite, and police OF messages as they pass



enforce policy — topology, flowspace

ensure the control only acts on the slice's

- topology
- flowspace

rewrite

- intersect flow definition with slice's flowspace, topology

enforce policy — forwarding rule

bookkeeping

- ensure *NOT* exceeding a preset limit

enforce policy — bandwidth isolation

```
send out port X
```

rate limit

```
send out queue Y on port X
```

enforce policy — CPU isolation

workaround with OF abstraction

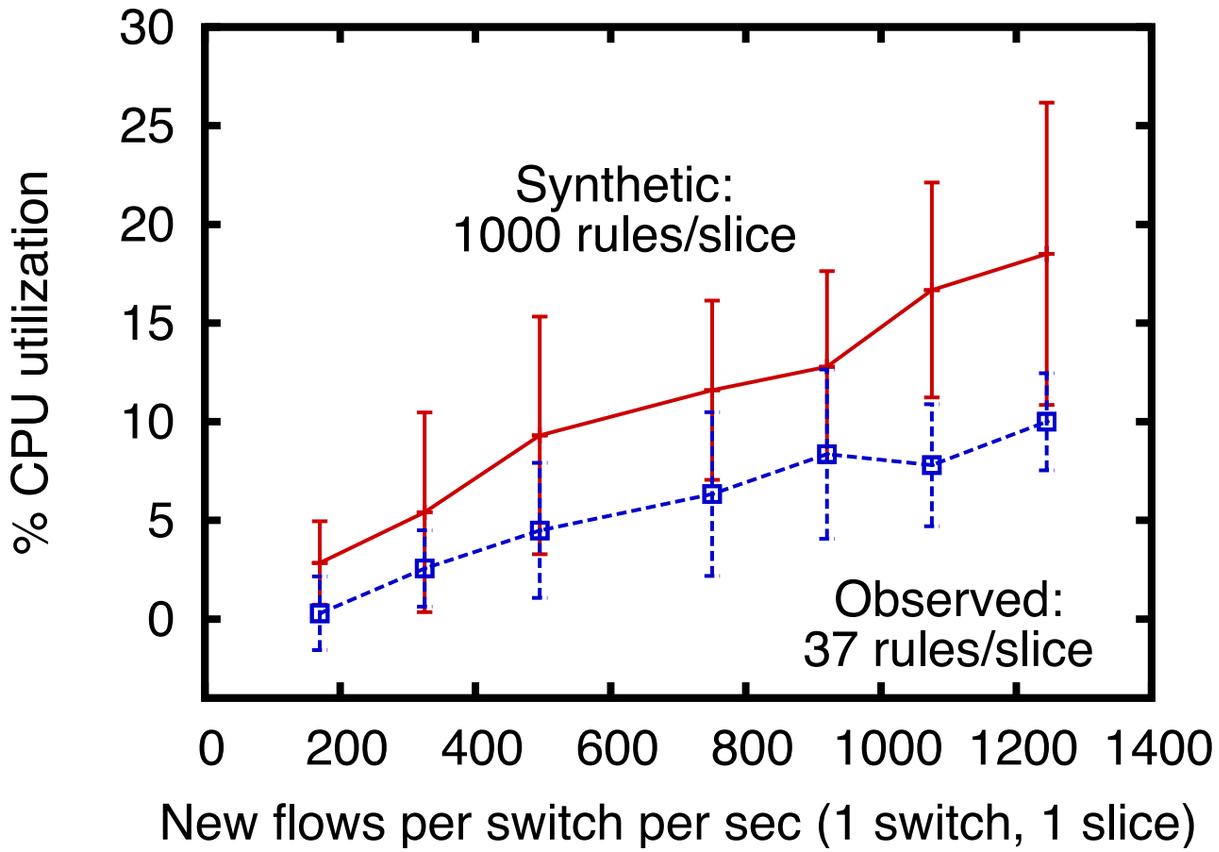
- new flow
  - rate limit arrival rate
- controller request
  - throttle message rate
- slow-path forwarding
  - rewrite to one-time packet forwarding events (then rate limit)
- internal bookkeeping
  - reserve sufficient CPU

scaling workload

measure CPU utilization

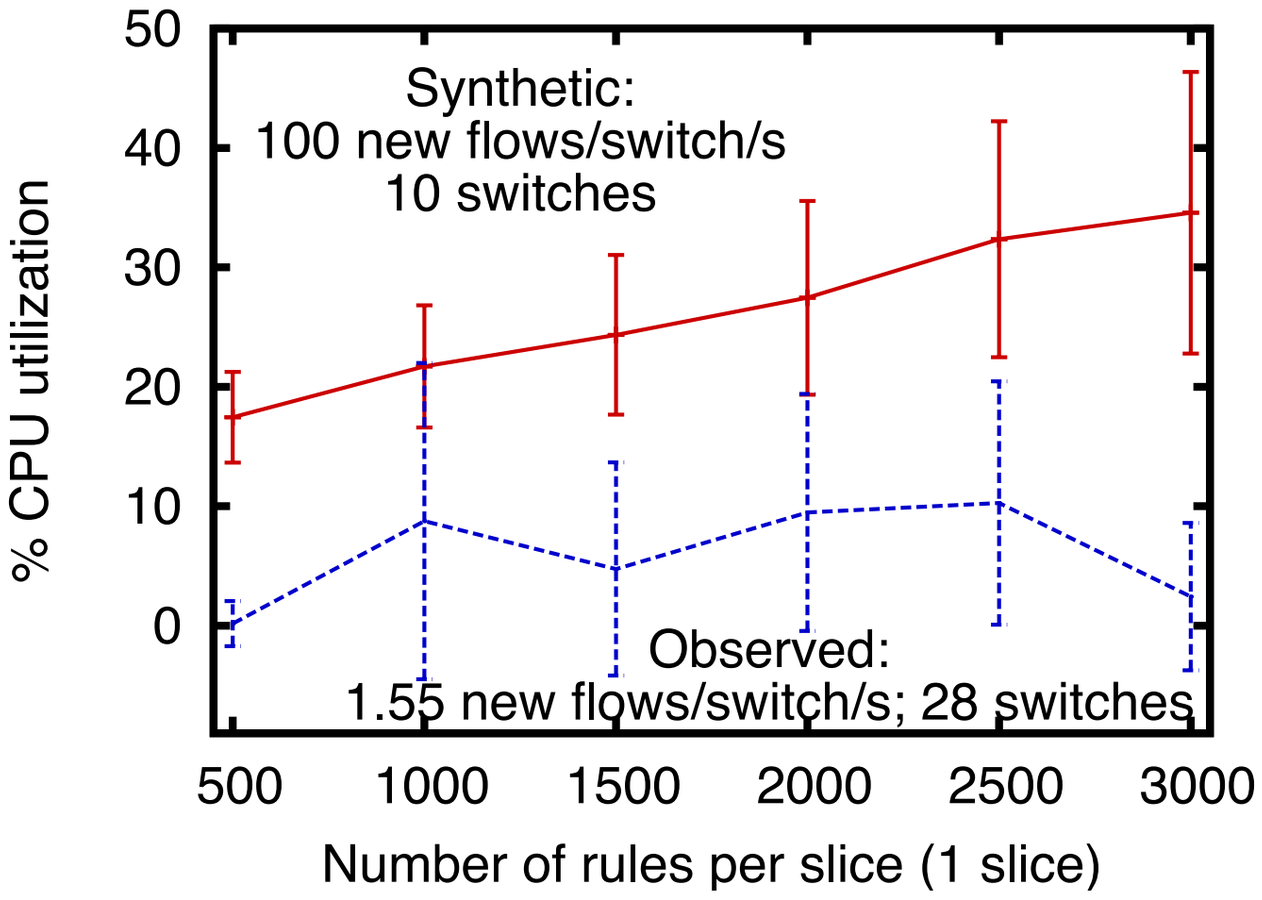
- new flow rate
- number of rules / slice
- number of slice

scaling workload



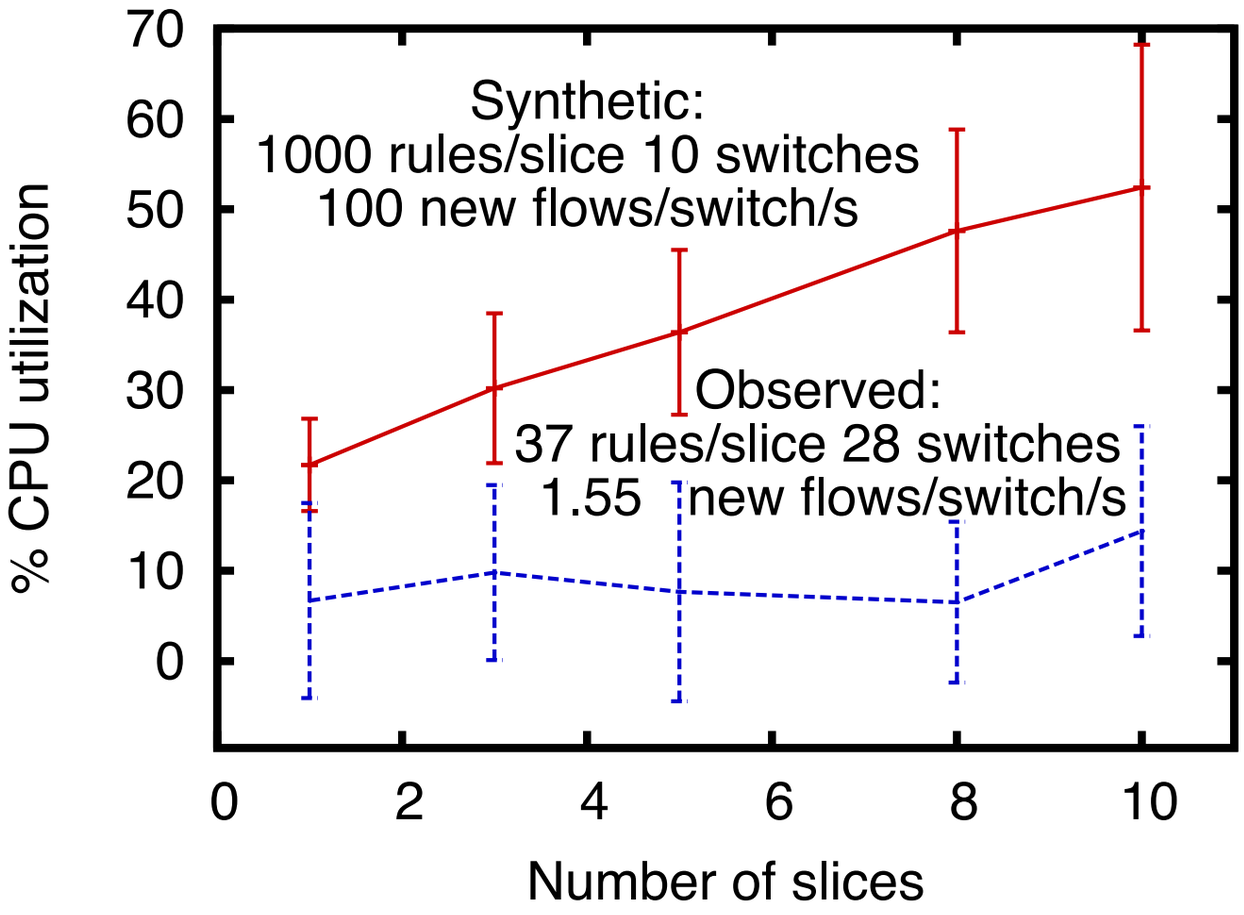
- liner growth with new flow rate
- question
  - 1k new flow/second — peak rate of published real-world network [IMC'05]

scaling workload



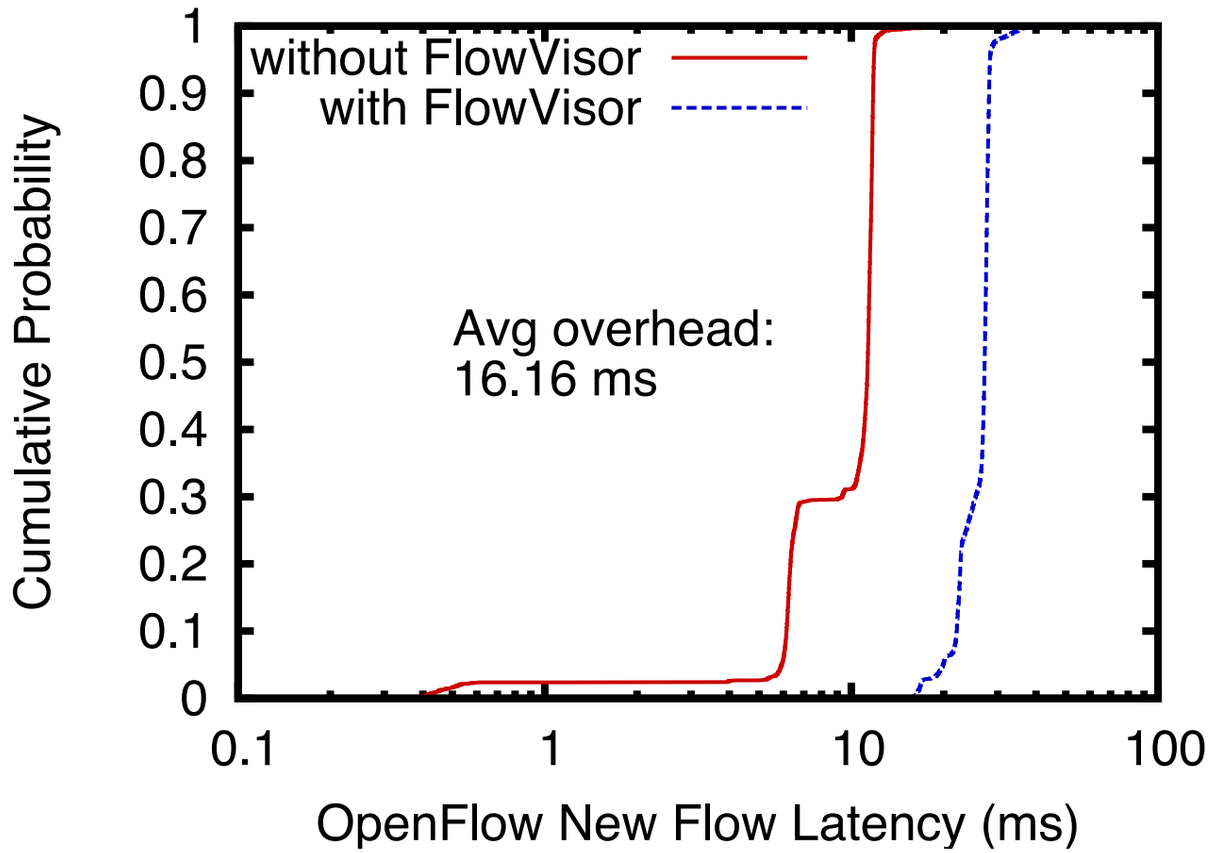
- linear growth with number of rule per-slide
- fluctuation with observed workload

scaling workload



- linear growth with number slides
- fluctuation with observed workload

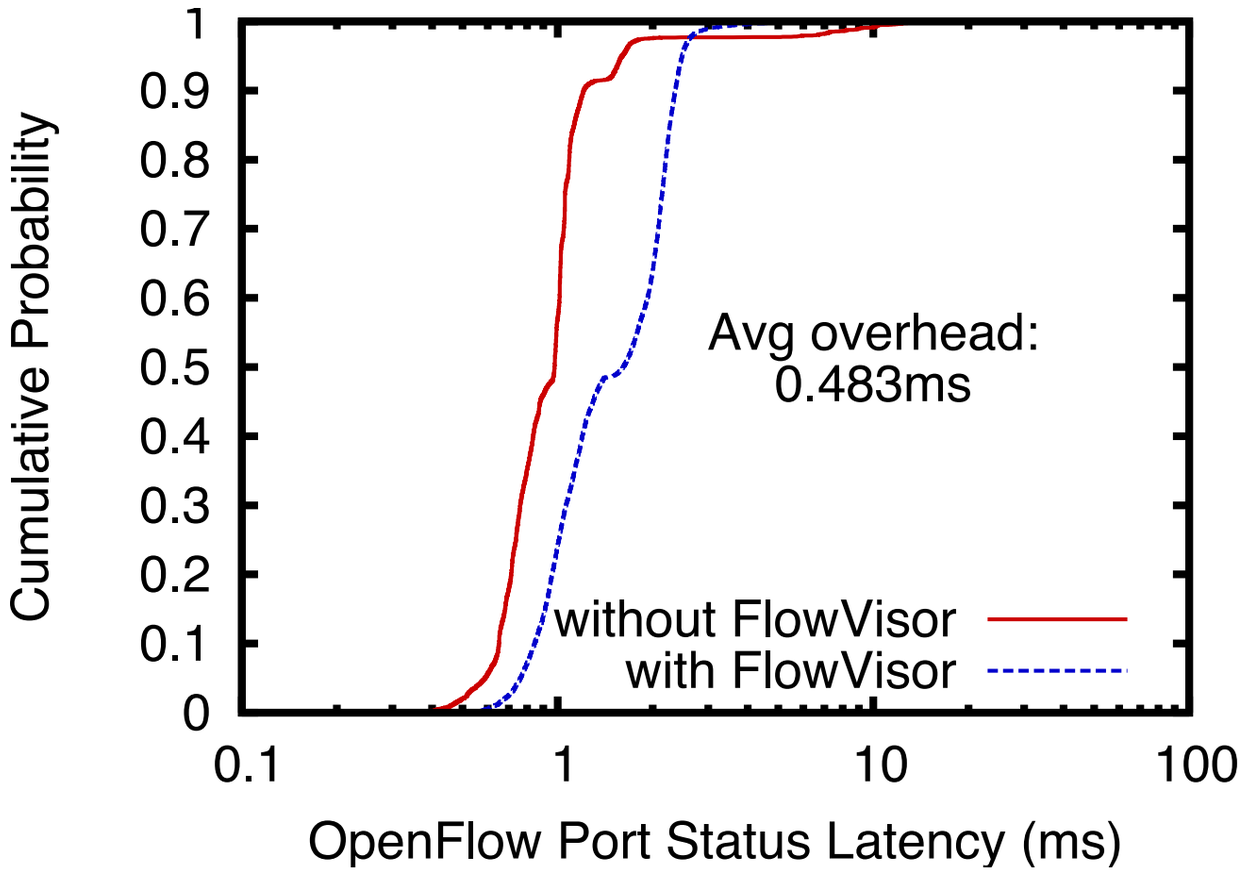
cross-layer overhead



measure response time for slice controller requests w/wo FlowVisor

- switch to controller
- new flow request
- wo: 12ms
- w: 16ms

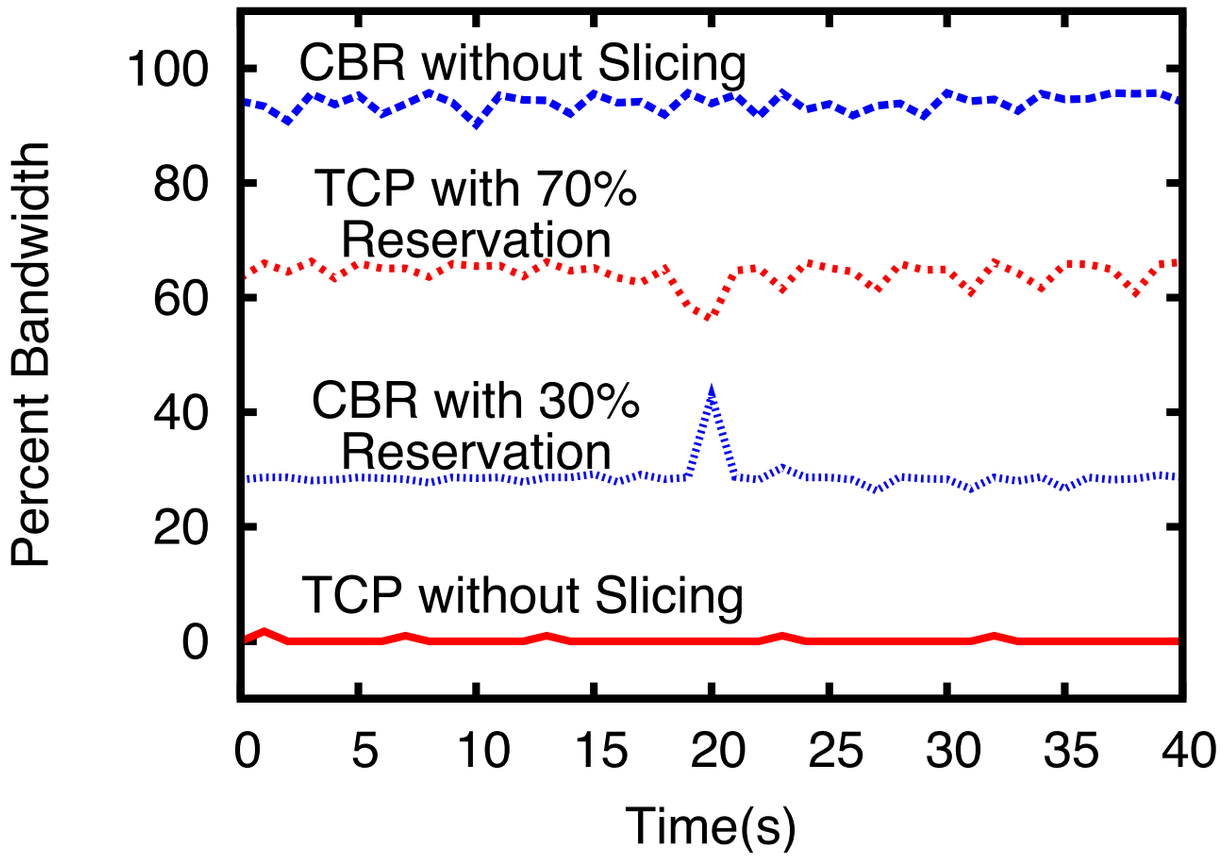
cross-layer overhead



measure response time for slice controller requests w/wo FlowVisor

- controller to switch
  - port status request
  - average: ,483ms
  - better optimization (not inherent)

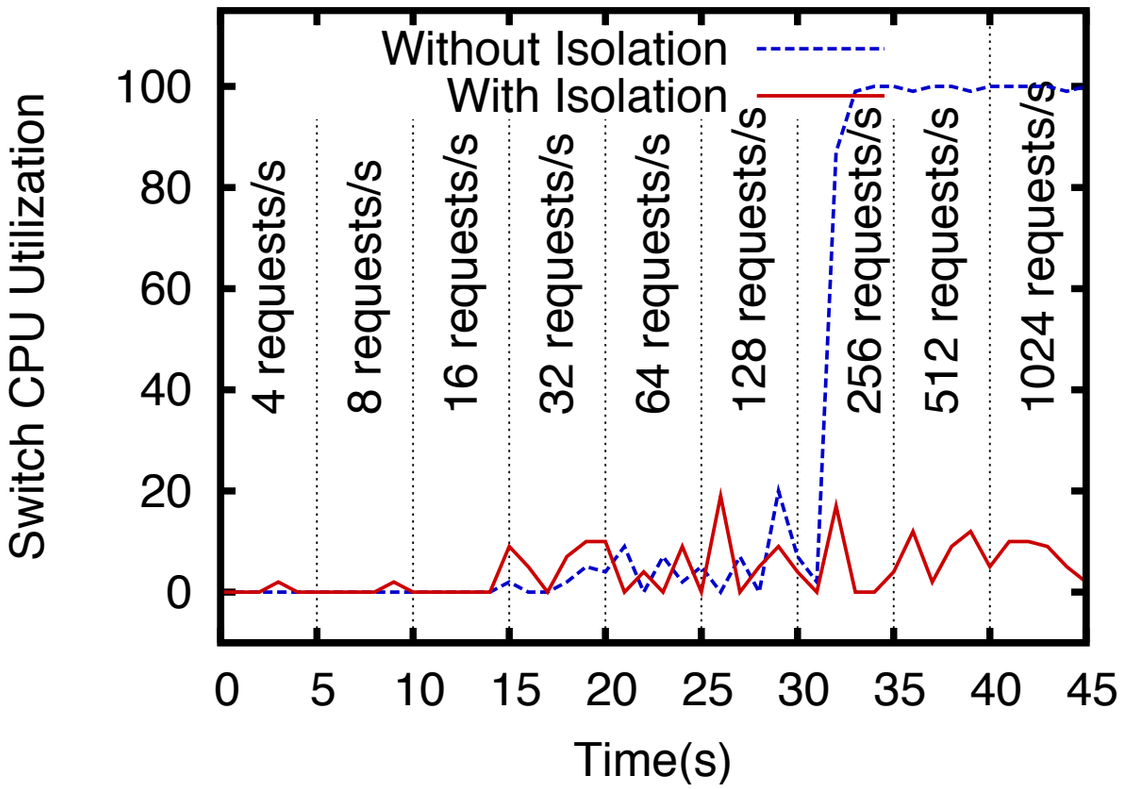
isolation validation



bandwidth

- slice 1: TCP-friendly traffic
- slice 2: TCP unfriendly (CBR)

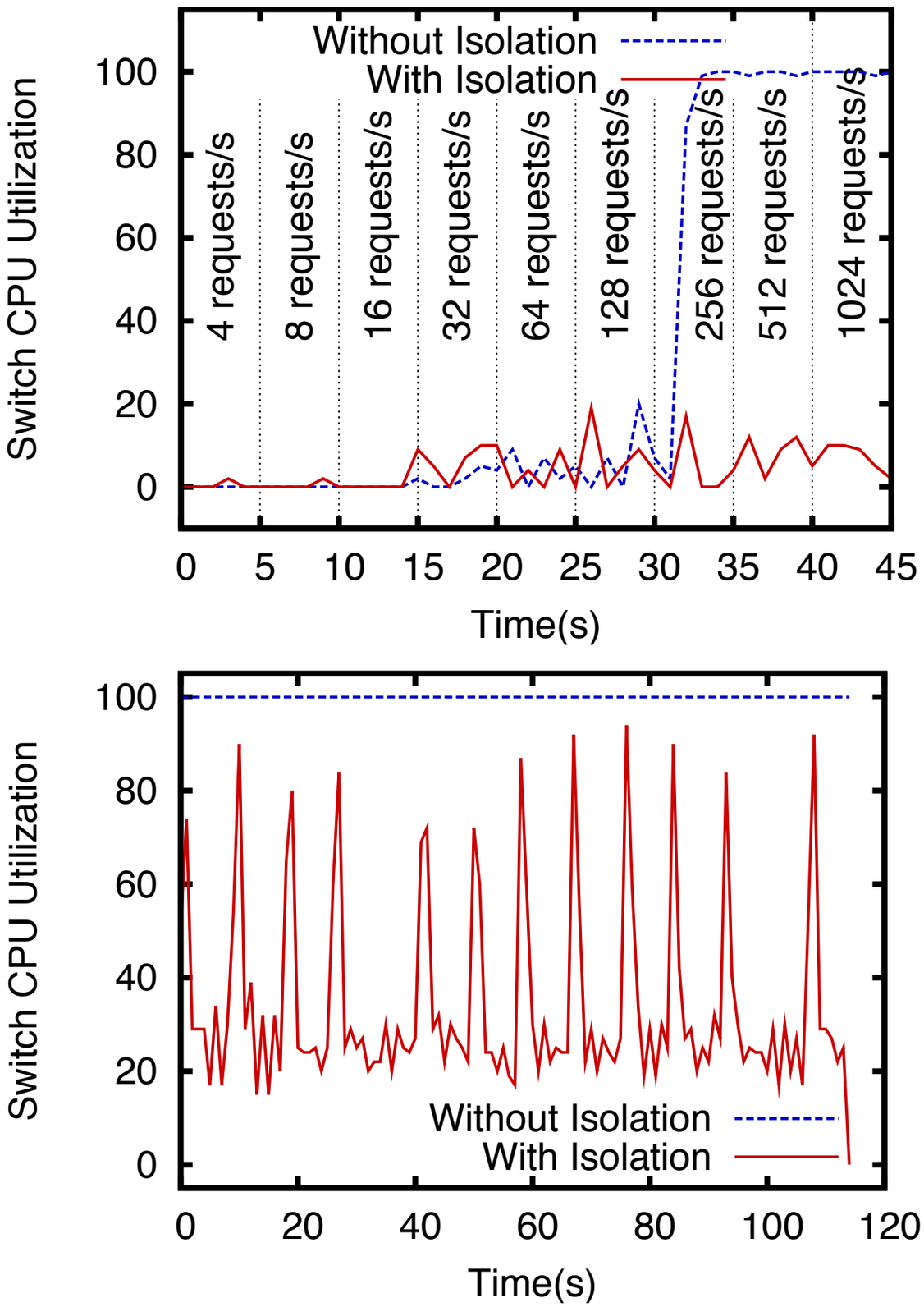
isolation validation



switch CPU

- malicious port status request message
  - switch CPU can handle < 256 requests

isolation validation



switch CPU

- malicious port status request message
  - switch CPU can handle < 256 requests
- new flow message
  - bursts: OF rate limit by null forwarding rules
  - fix: future OF expose better hardware CPU limiting feature

discussion

FlowVisor assumes control/data plane separation by OpenFlow

- OF makes very few of the hardware capability available
  - packet scheduling, MPLS
- OF exposes basic set of primitive "plumbing" sufficient for a wide range of experiments

discussion

FlowVisor assumes control/data plane separation by OpenFlow

- OF makes very few of the hardware capability available
  - packet scheduling, MPLS
- OF exposes basic set of primitive "plumbing" sufficient for a wide range of experiments

sidestep difficulties

- transfer to specialized hardware
- all the way down to the end users
- real user traffic

re-cap

OpenFlow

- "just enough" abstraction of the forwarding element

FlowVisor

- transparent isolation