enPiT Cloud

Cloud Q9

Experience from Measuring Per-Packet Cost of Software Defined Networking

Marat Zhanikeev
maratishe@gmail.com

2013/06/17

enPiT Cloud
大阪大学、東京大学、
東京工業大学、神戸大学、
九州工業大学

enPiT Security
東北大学、北陸先端科学技術大学院大学、
奈良先端科学技術大学院大学、
慶應義塾大学、情報セキュリティ大学院大学

enPiT Emb
名古屋大学、九州大学

enPiT BizApp
筑波大学、公立はこだて未来大学，
産業技術大学院大学

SDN Implementation Today

- OpenVSwitch is by far the most popular software
  - replaces Linux bridge
  - software-controlled
- on control side, OpenFlow seems to be the popular default in Japan
  - there are other APIs, some also relatively popular in abroad
  - some just use web APIs without calling it a protocol

The Problem is...

... that there are very few studies which measure performance of SDN solutions

The Target

That Linux Bridge Replacement, ...

... what is its performance compared to the native Linux network stack?

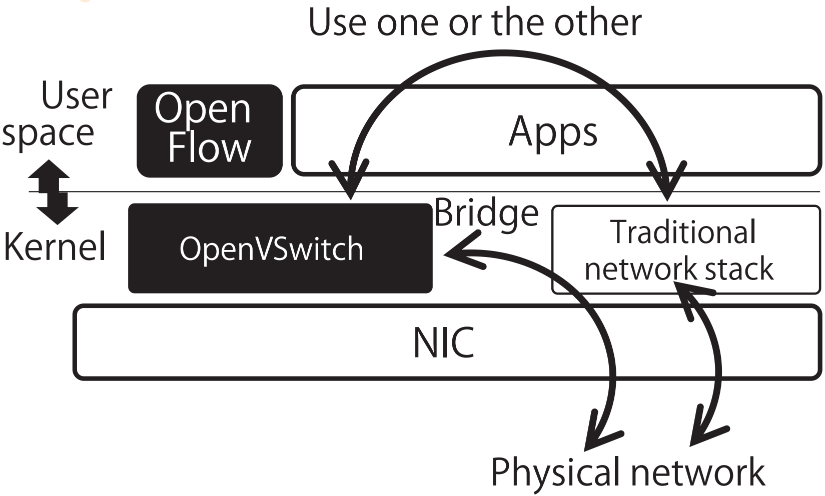
- focus on OpenVSwitch
  - with and without OpenFlow controller
- measure per-packet overhead (cost)
- 1Gbps for now, will move on to 10Gbps in the next study

Performance Metrics...

one-way delay is difficult, so, throughput for now

Bridge: One or the Other

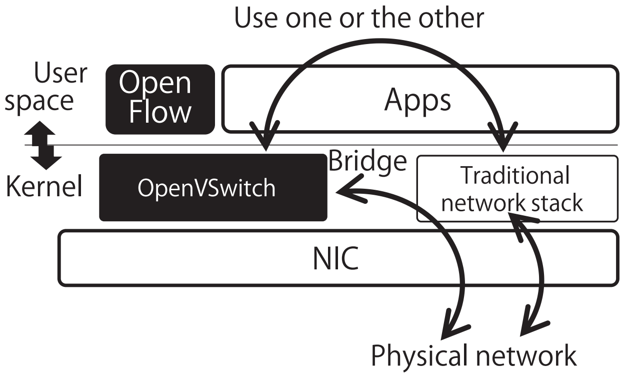
- bad news: today it is difficult to switch back and forth between virtual and physical networking on the fly



Actual Components

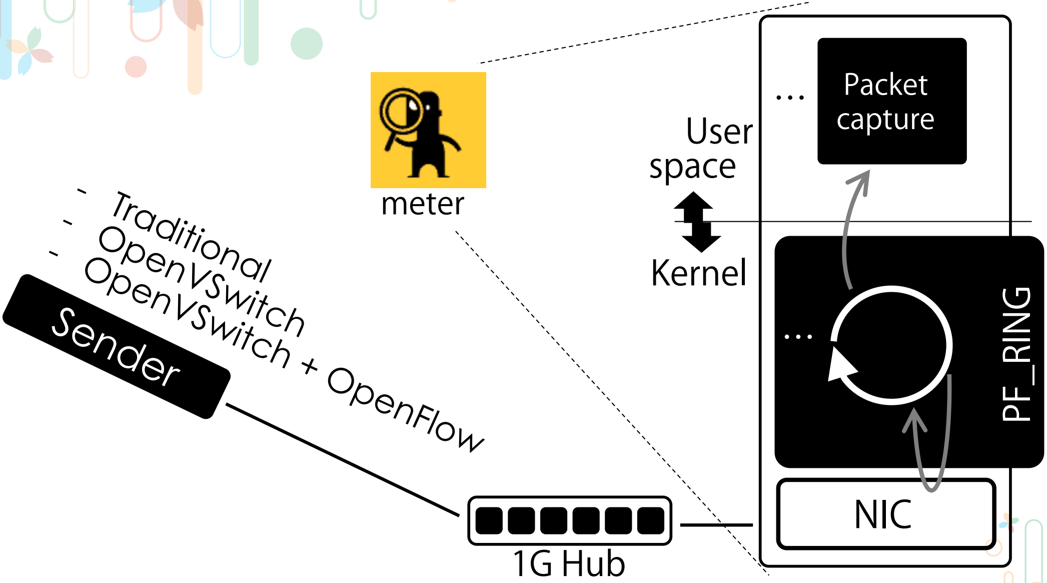
- Xen Cloud Platform (XCP) v.1.6
  - released May 2013, merged with OpenVStack of late 2012
- Floodlight as OpenFlow controller
  - running on XCP machine for quick round trip
  - in learning switch mode
- Meter / Packet Capture using PF_RING
  - fastest capture engine today
  - precise + low overhead capture

Sender Models



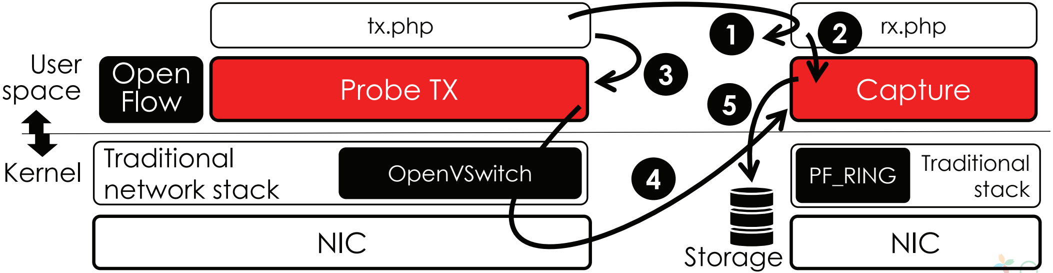
1. OpenVSwitch and OpenFlow models
   - XCP1.6
   - OpenVSwitch is active by default (cannot be disabled)
   - OpenFlow running on the same machine, controller is registered with XCP
2. Traditional Model
   - Fedora 18 machine, out of the box

Experiment Design



(one) Measurement Run

- (1) wget rx.php with random setups, (2) rx.php runs capture process (C/C++), (3) tx.php runs the PROBE (C/C++), (4) probe sends a back-to-back stream of packets towards Meter, (5) CAPTURE stores statistics about captured packets to a file



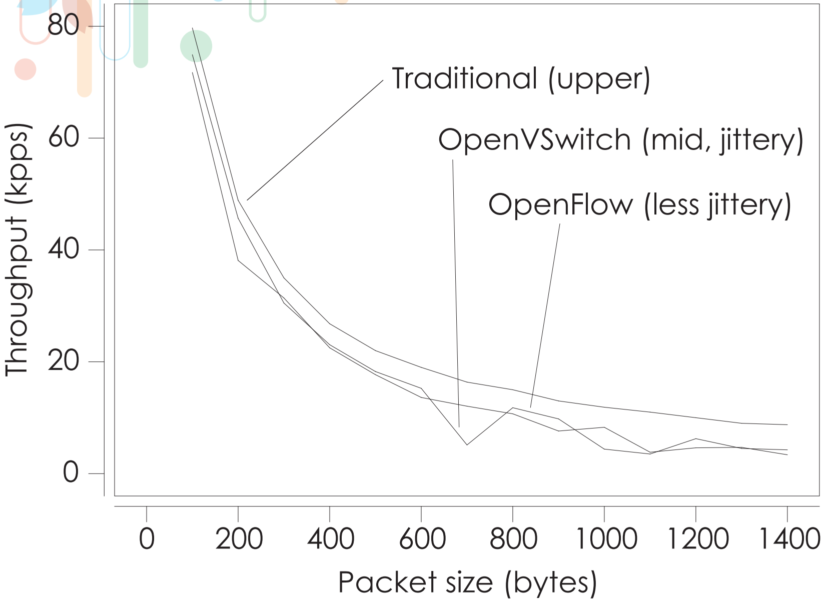
Avoiding Bottlenecks (Meter)

- cannot write to file for each packet -- bottleneck
- split into 100-packet batches
- record gaps in packet IDs
  - each time sequence number is skipped due to packet loss

Other/Smaller Tricks

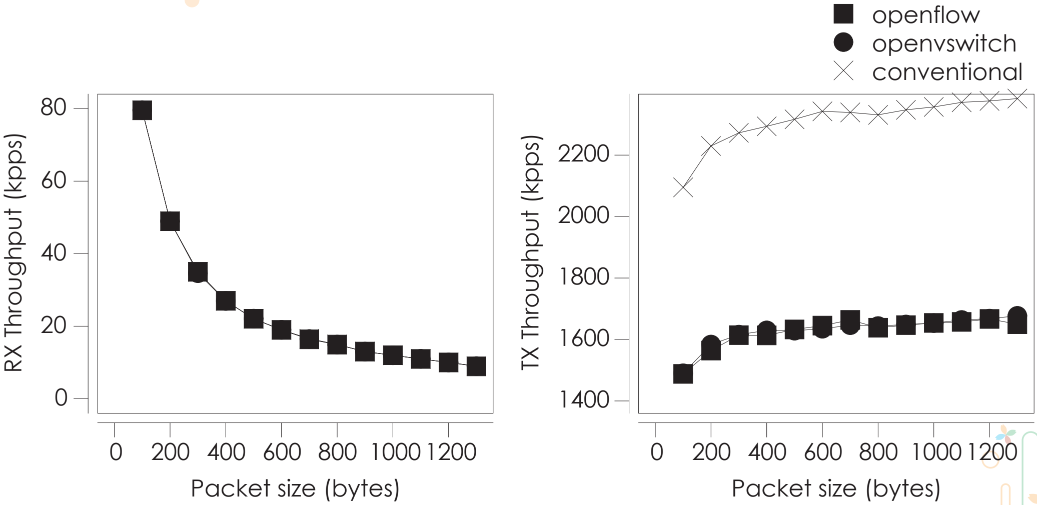
- each run uses new port
  - looks like new/fresh flow to OpenVSwitch
- same machine as Sender to exclude hardware differences
- because packet IDs are monitored, can measure packet loss
- triggered by web API (wget), so, no need to sync machines
  - time is relative

Results (1)


Traditional (upper)
OpenVSwitch (mid, jittery)
OpenFlow (less jittery)
Throughput (kpps)
0
20
40
60
80
Packet size (bytes)
0
200
400
600
800
1000
1200
1400

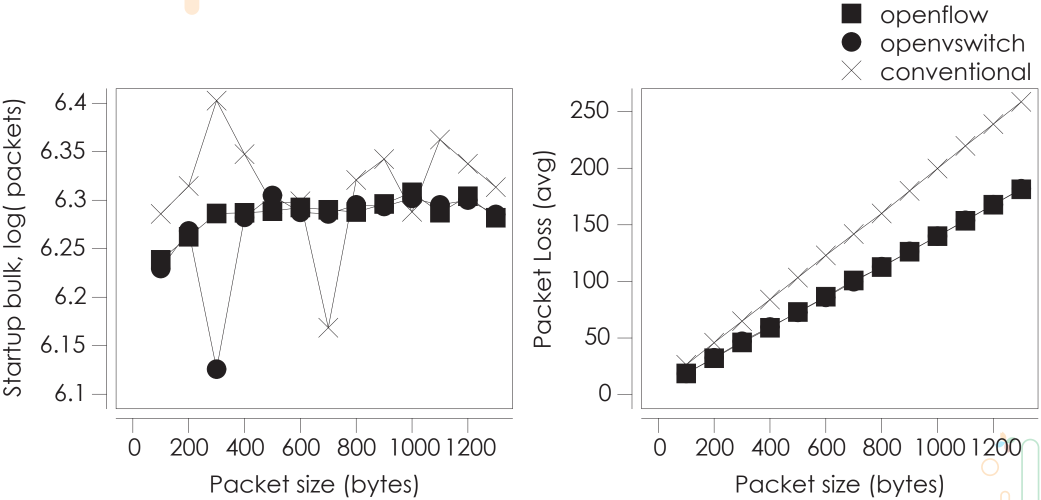
Results (2) outliers removed

- remove bottom/top 25% of data (extremes)
- TX throughput above capacity (loss)



Results (3) outliers removed

- startup bulk: how many packets can be pushed until capture kicks in



Where to Go From Here

- 10Gbps
  - Sender can clearly send above 1Gbps
- scenarios when OpenFlow is remove and more active
  - measure overhead from initial setup
- add VLANs, both local (VM--VM) and local--remote

That's all, thank you ...