Devolve-Redeem

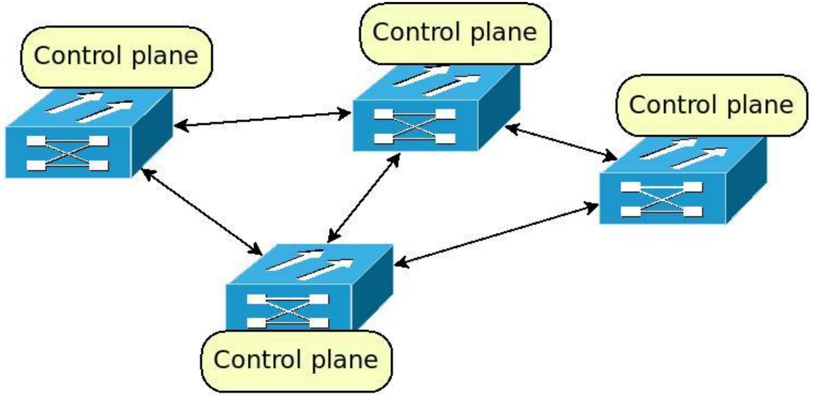
Hierarchical SDN controllers with adaptive offloading

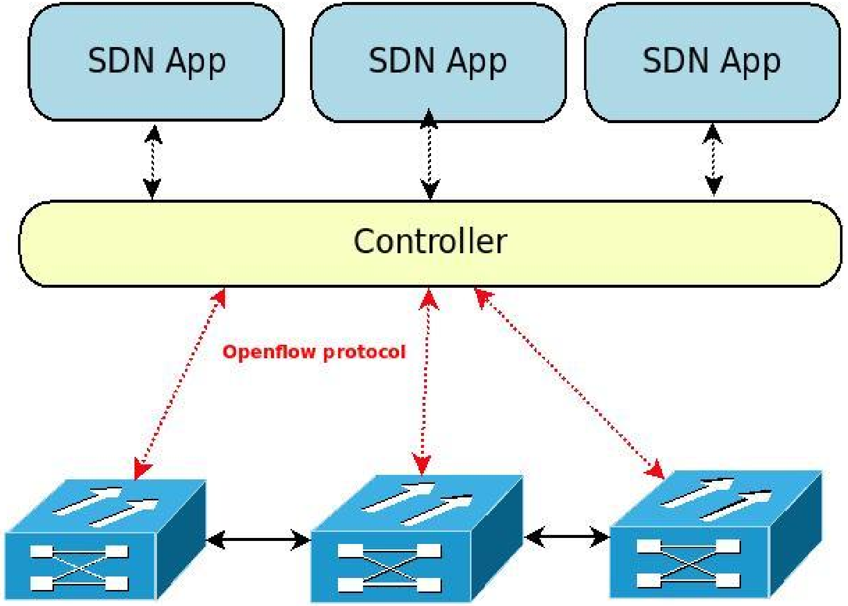
Rinku Shah
Mythili Vutukuru
Purushottam Kulkarni

IIT Bombay, India

3rd August, APNet 2017

Traditional network vs Software-defined network





- ❑ Simplified network mgmt
- ❑ Ease of control-plane programming

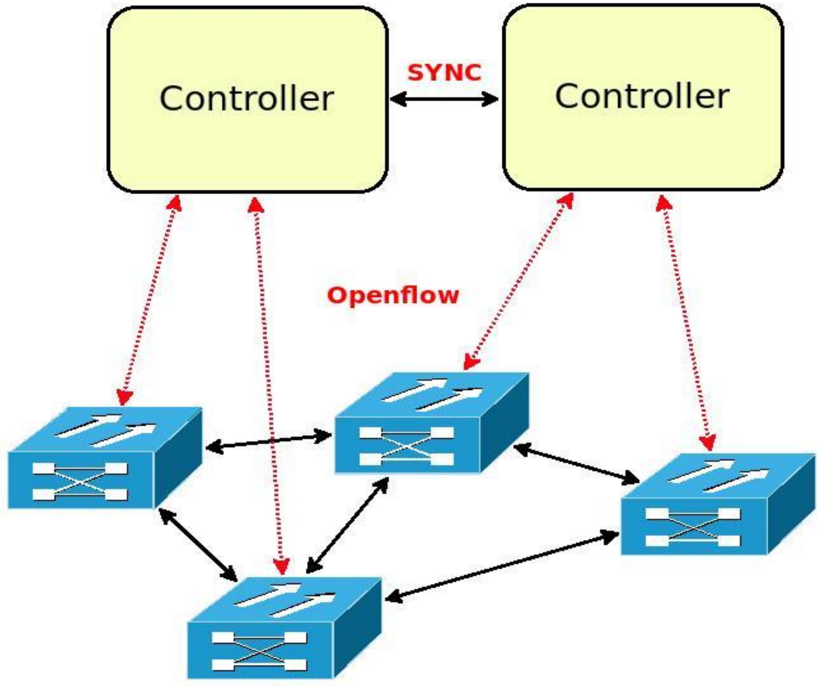
How far can SDN Controllers scale?

Openflow controllers support around **10K to 30K flows/sec** *

SDN networks have flow arrival rate of **100K to 1M flows/sec****

* Marcial P Fernandez and others. Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive. Advanced information and applications, IEEE 2013.
** Kandula and others.. The nature of data center traffic: measurements & analysis. In Proceedings of IMC 2009

Controller Scaling technique - HORIZONTAL



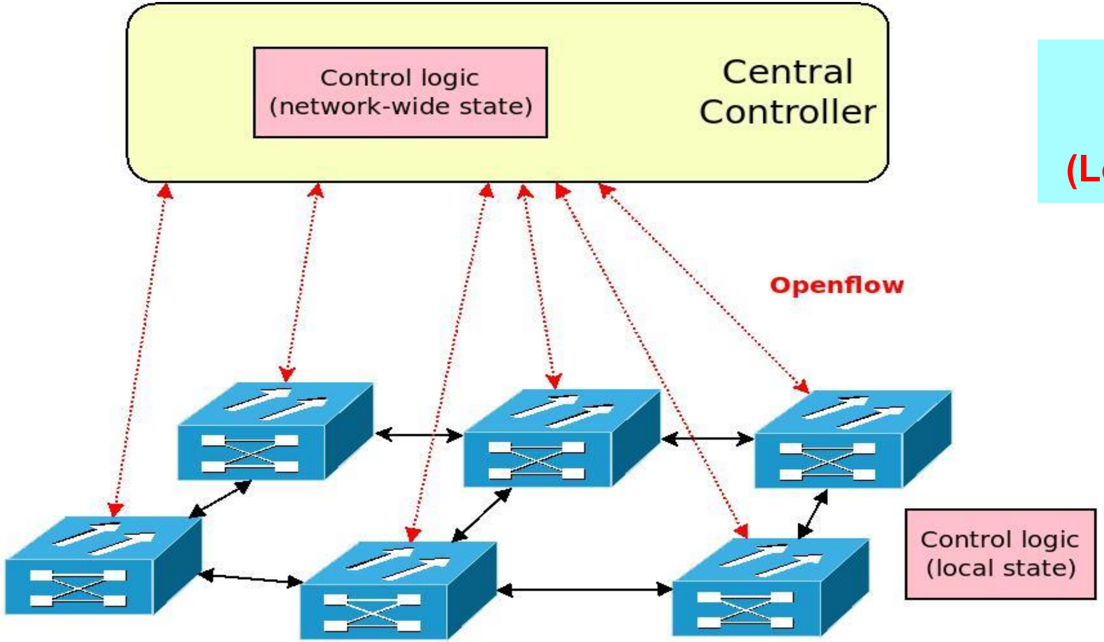
Subset of switches assigned to each controller

Need for synchronization between controllers

Onix*
Hyperflow**

* Teemu Koponen and others. Onix: A Distributed Control Platform for Large-scale Production Networks. In Proc of the Conference on OSDI, 2010.
** Amin Tootoonchian and Yashar Ganjali. HyperFlow: A Distributed Control Plane for OpenFlow. In Proc of the Internet Network Management Conference on Research on Enterprise Networking, 2010.

Controller Scaling technique - VERTICAL



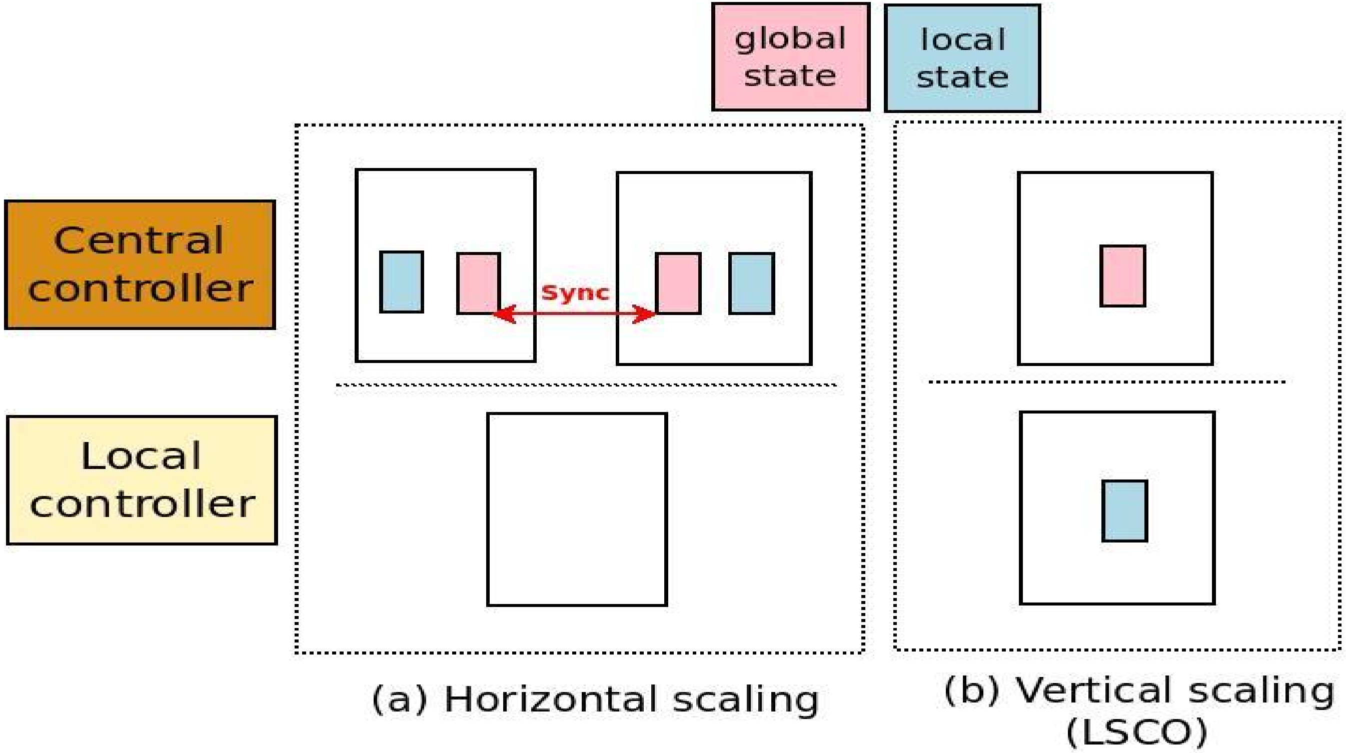
We call this technique,
LSCO
(Local state based compute offload)

Devoflow*
Kandoo**
FOCUS***

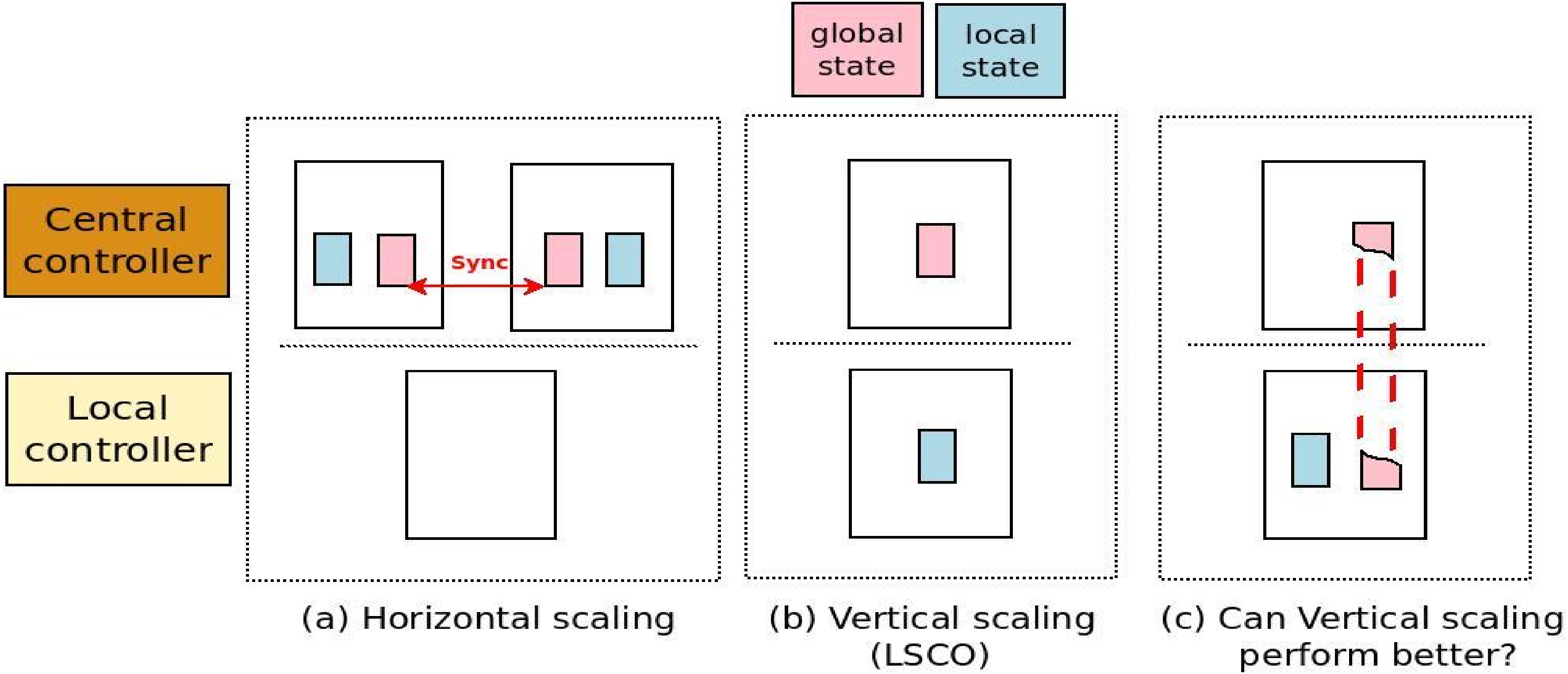
LOCAL state egs
Flow stats
Switch mappings

* Andrew R. Curtis and others. DevoFlow: Scaling Flow Management for High-performance Networks. In Proc of the SIGCOMM, 2011.
** Soheil Hassas Yeganeh and Yashar Ganjali. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. In Proc of the Workshop on HoTSDN, 2012.
*** Ji Yang and others. FOCUS: Function Offloading from a Controller to Utilize Switch Power. In Proc of IEEE Conference on NFV-SDN, 2016.

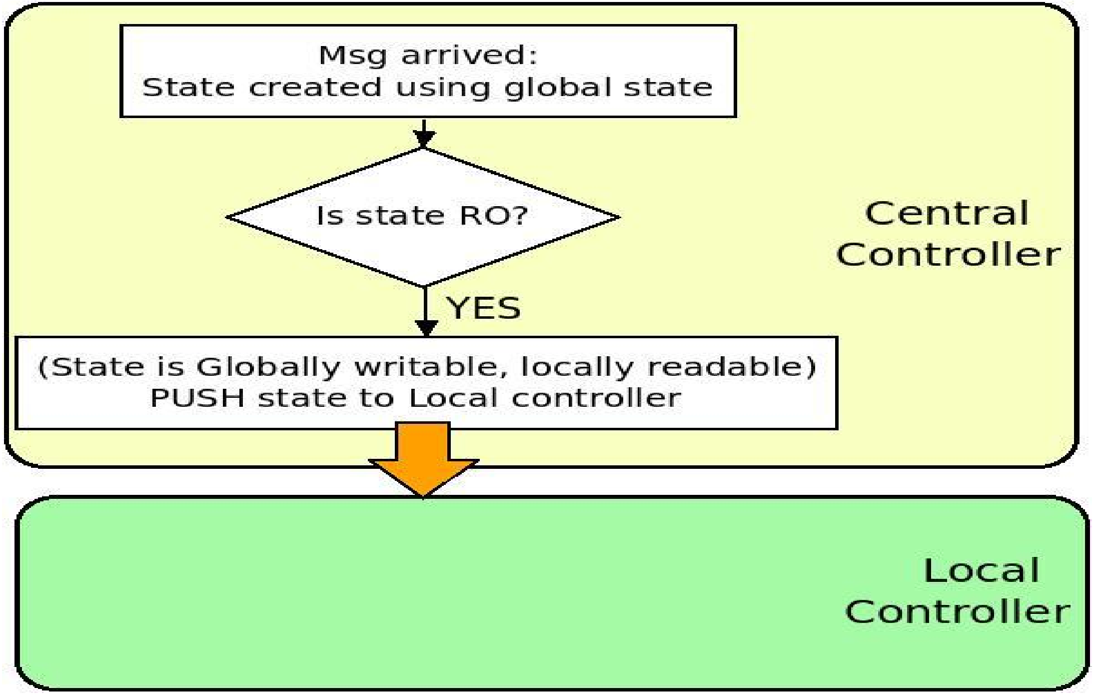
Controller Scaling techniques: Abstract view



Can Vertical Scaling perform better?



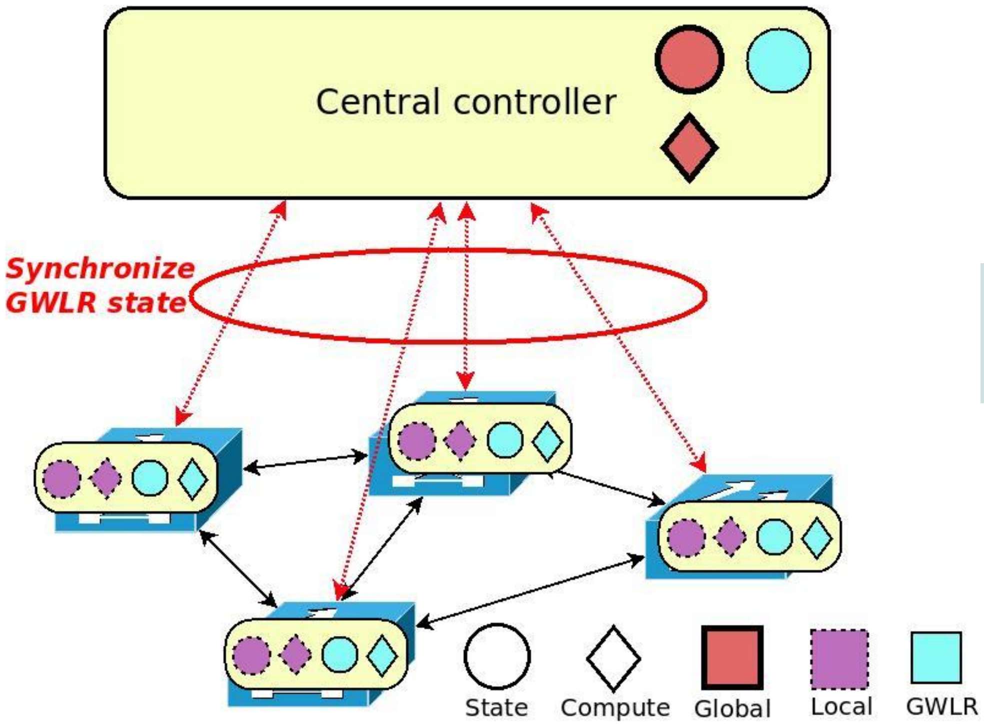
Key insight - GWLR state(Globally writable, but locally readable)



GWLR state examples-

1. **Tunnel Id (LTE EPC)**
2. **MPLS label**
3. **Session state**
4. **Network Policy state**

GSCO (GWLR state based compute offload)

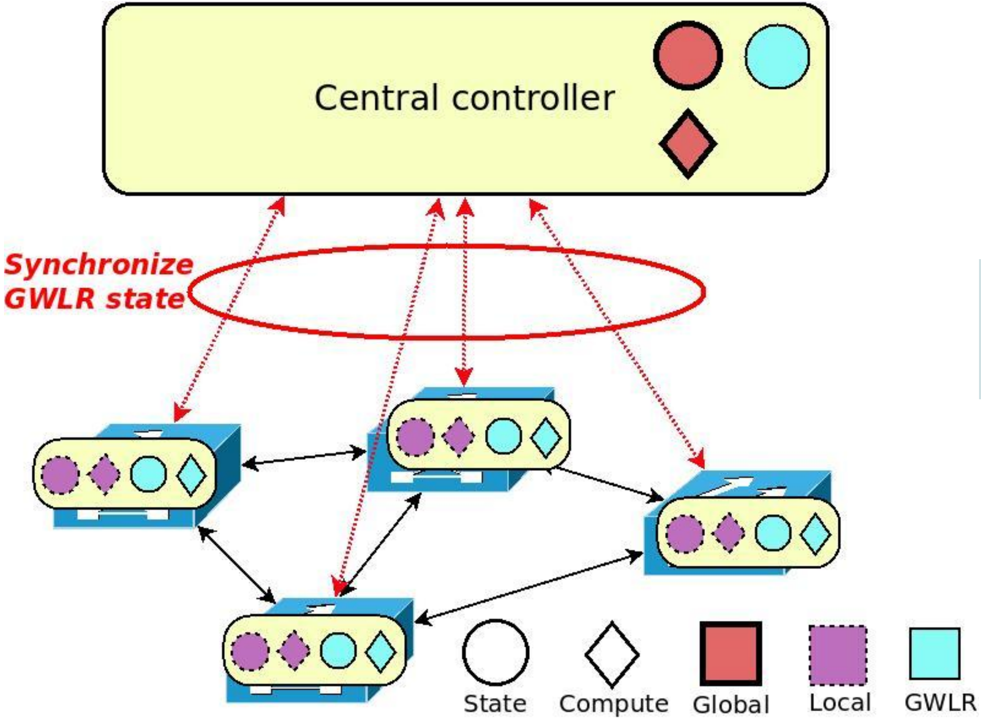


Offload computations based on GWLR state

Should we offload all GWLR state ?

Synchronization cost may be high

GSCO (GWLR state based compute offload)



Offload computations based on GWLR state

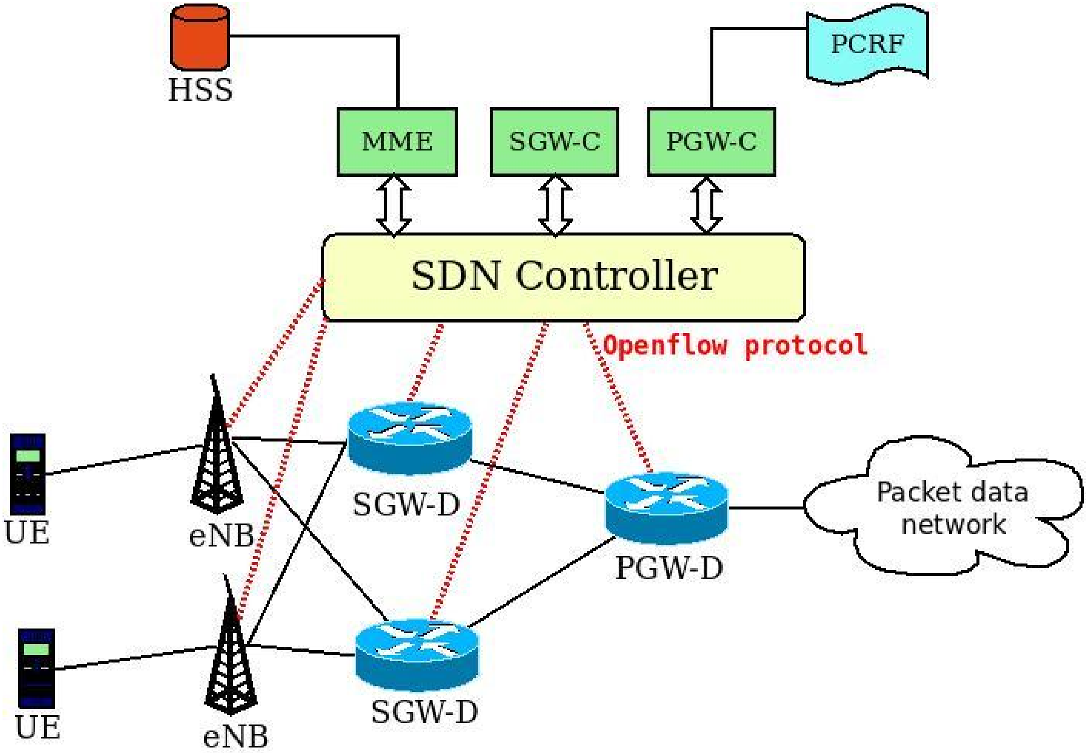
Should we offload all GWLR state ?
Synchronization cost may be high

Centralized or LSCO or GSCO?

Key Contributions

1. **GWLR state** based offload technique
    a. **GSCO** (GWLR state based computation offload)
2. Application code is **agnostic** to scalability design
3. Framework that aids **Adaptive Offload**
    a. Designed Cost metric
    b. Implemented OVS feature for Compute Placement

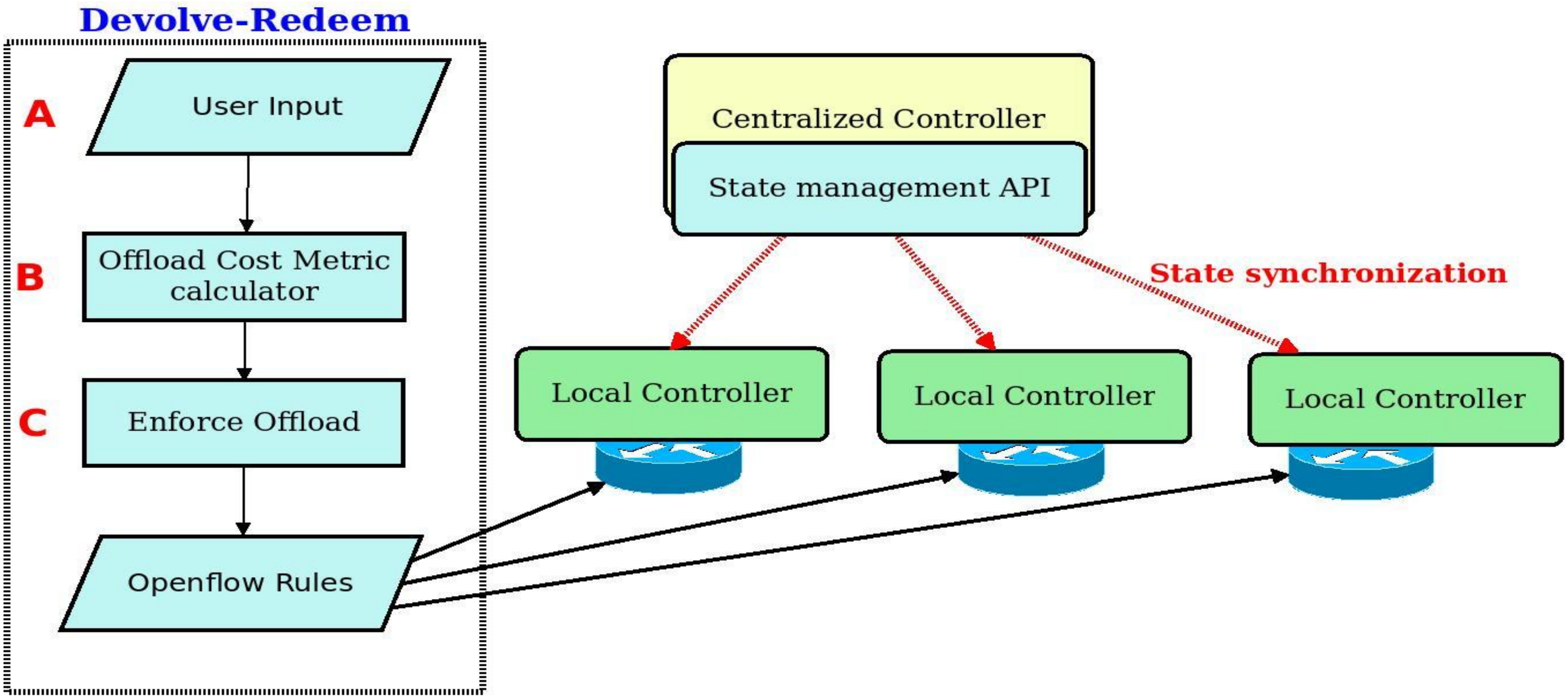
Use-case: SDN based LTE-EPC application



LTE-EPC procedures considered-

1. **Attach Request**
2. **Service Request**

Devolve-Redeem Design



A. User Input for LTE-EPC

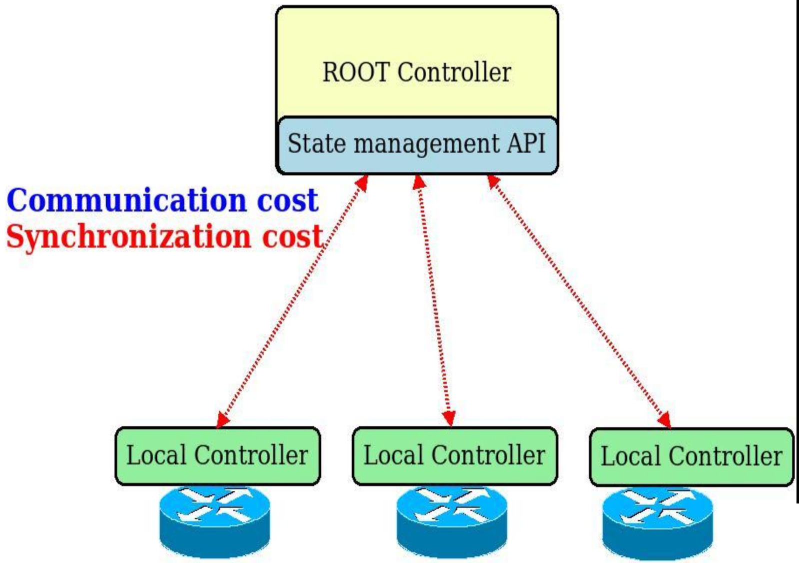
Msg-id, <N^{LR}, N^{LW}, N^{XR},N^{XW}, N^{GR}, N^{GW}, N^{RL}>

N^{L} : # of **Local** states accessed
N^{X} : # of **GWLR** states accessed
N^{G} : # of **Global** states accessed
N^{RL} : # of Openflow **Rules**

Example LTE-EPC Messages	**N^{LR}**	**N^{LW}**	**N^{XR}**	**N^{XW}**	**N^{GR}**	**N^{GW}**	**N^{RL}**
Auth_Step_1	0	0	0	0	1	2	0
Send_UE_TEID	0	0	2	1	0	0	2
UE Context Release	2	0	0	1	0	0	3
Context Setup Response	0	0	1	1	0	0	2

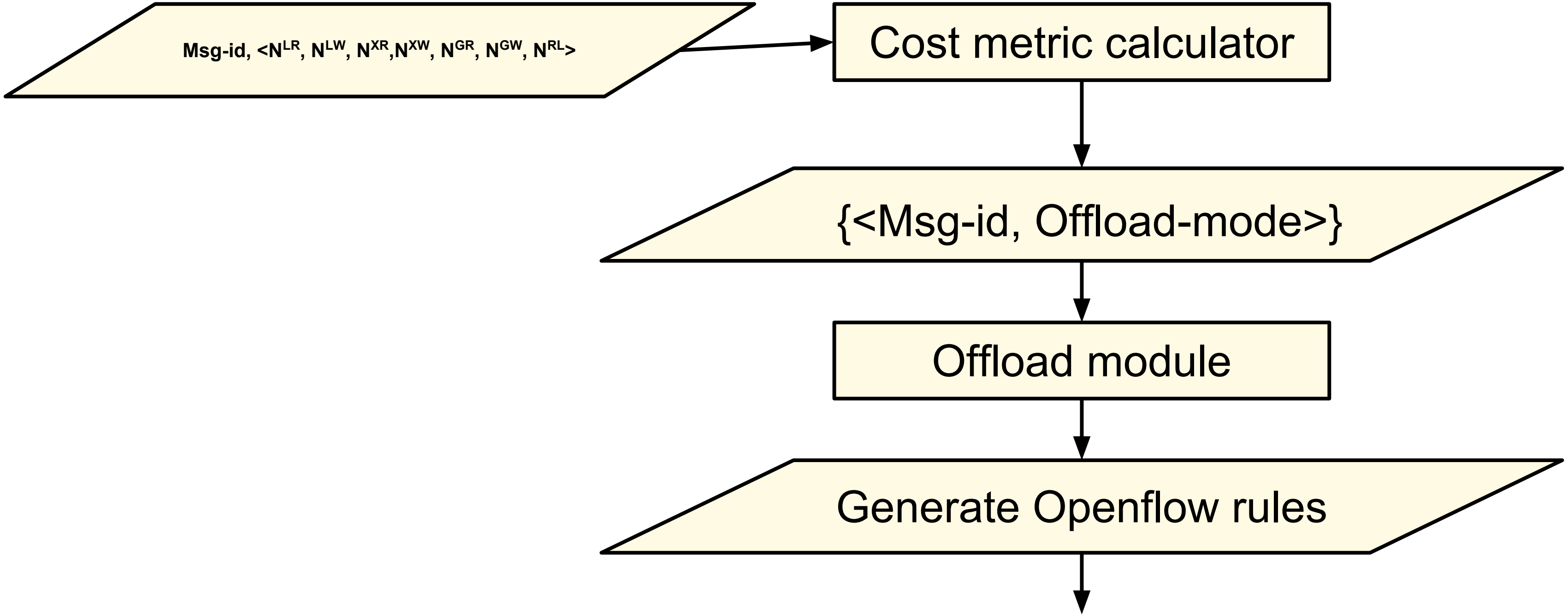
B. Offload Cost-metric

Cost_mode = State-access cost + Communication cost + Synchronization cost



Offload Mode	Communication Cost	Synchronization Cost
Centralized	RTT to ROOT	0
LSCO	RTT to LOCAL/ROOT	0
GSCO	RTT to LOCAL/ROOT	Depends on current traffic mix

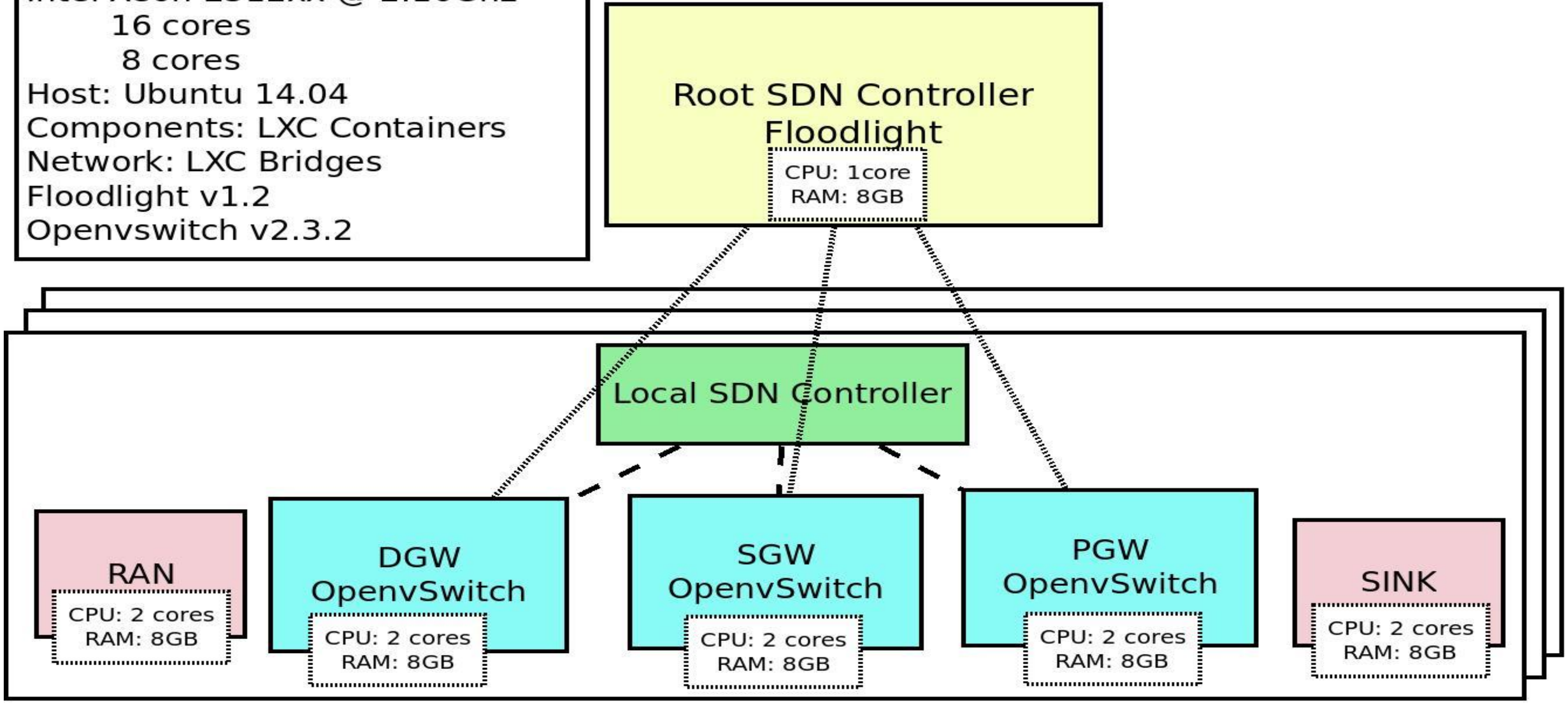
C. Enforce Offload module



This flow should be followed for each message

Experimental Setup

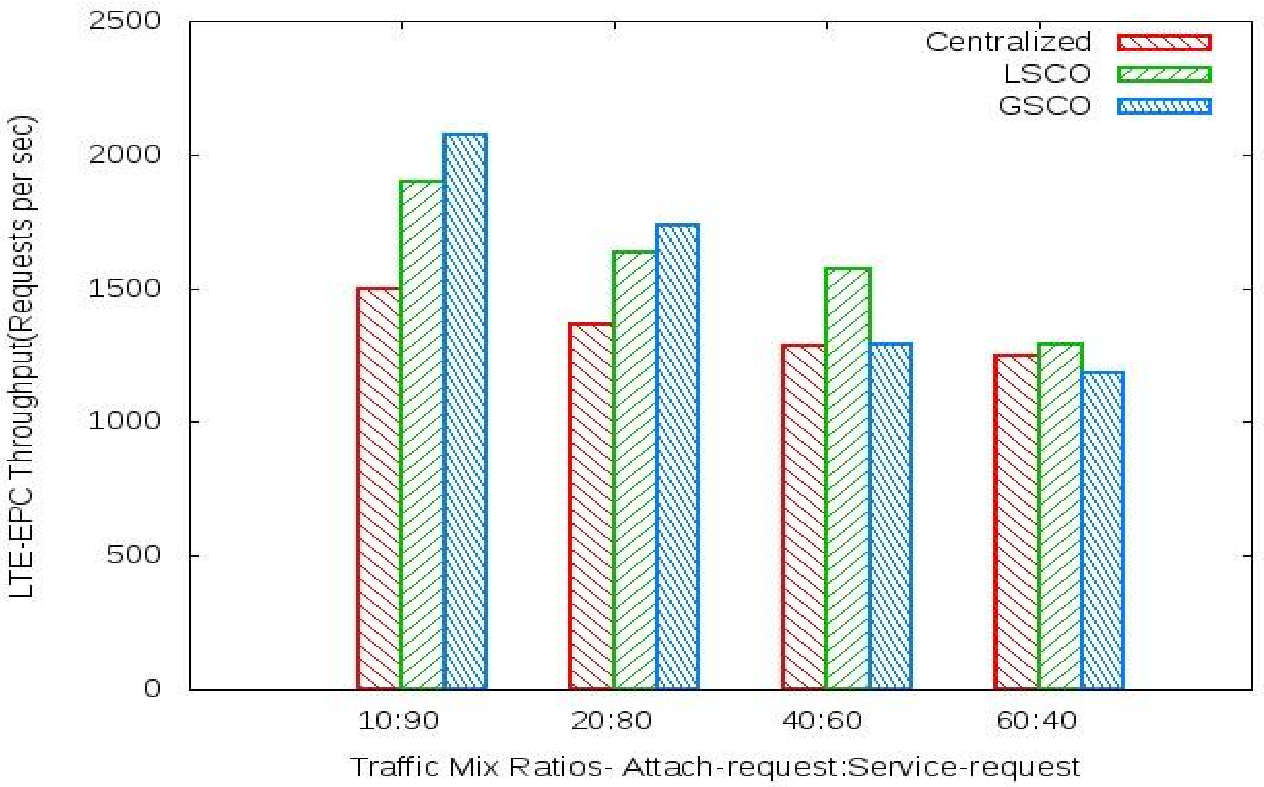
Intel Xeon E312xx @ 2.16Ghz
16 cores
8 cores
Host: Ubuntu 14.04
Components: LXC Containers
Network: LXC Bridges
Floodlight v1.2
Openvswitch v2.3.2



Questions to be answered?

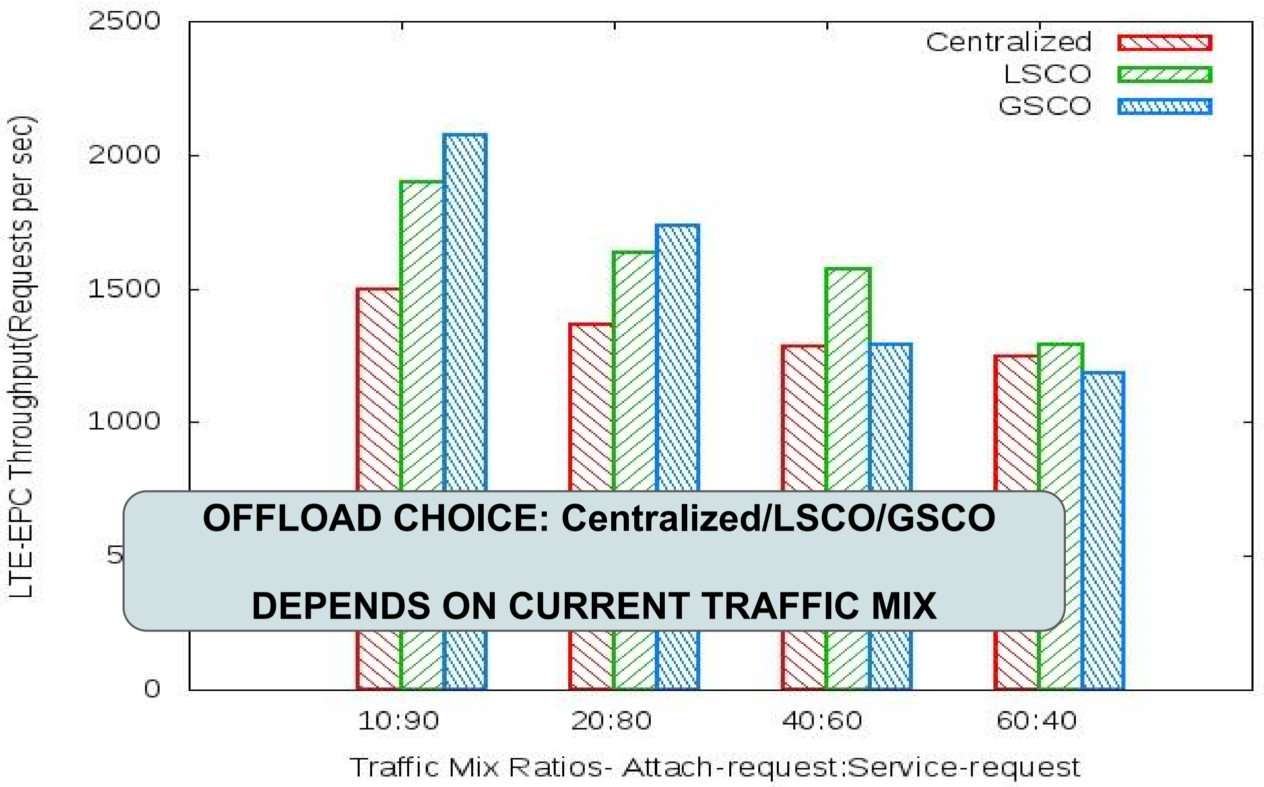
1. **What is the best offload scheme for a given traffic mix?**
2. What is the impact of the offload choice on-
   a. Request Completion Time (Latency)
   b. Root Controller Traffic
   c. Root Synchronization Cost

Evaluation - Offload A: All GWLR state



ATTACH <= 20% GSCO = 1.4X Centralized
20% < ATTACH <= 60% LSCO = 1.27X Centralized
ATTACH > 90%

Evaluation - Offload A: All GWLR state

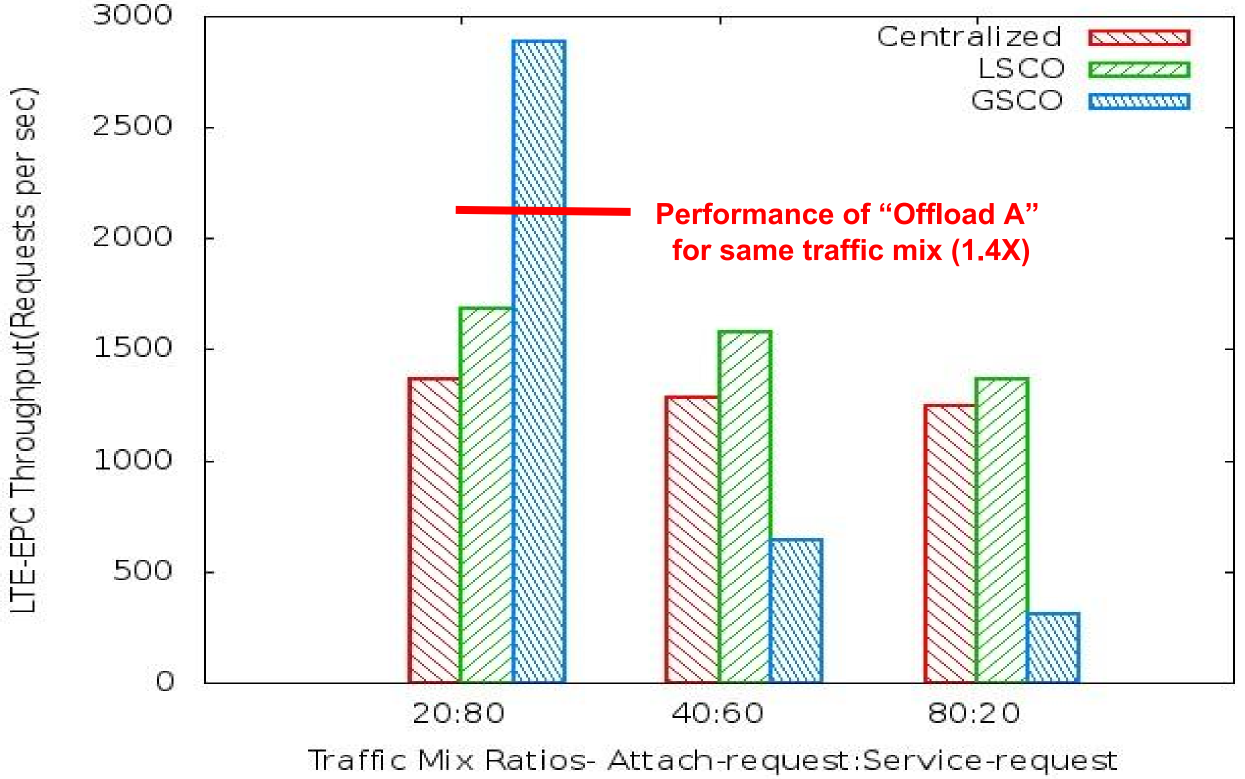


OFFLOAD CHOICE: Centralized/LSCO/GSCO

DEPENDS ON CURRENT TRAFFIC MIX

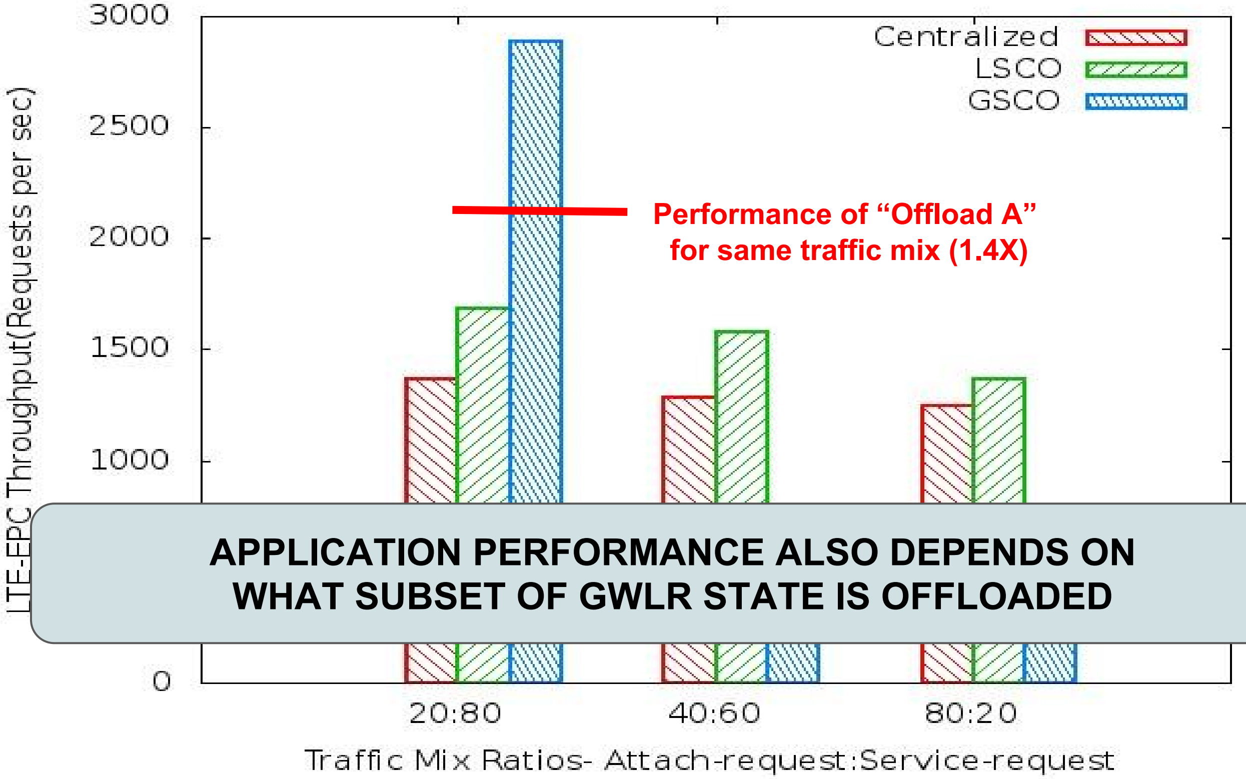
ATTACH <= 20% **GSCO = 1.4X Centralized**
20% < ATTACH <= 60% **LSCO = 1.27X Centralized**
ATTACH > 90%

Evaluation - Offload B: Subset of GWLR state



ATTACH <= 20% **GSCO = 2.11X Centralized**
20% < ATTACH <= 60% **LSCO = 1.23X Centralized**
ATTACH > 90%

Evaluation - Offload B: Subset of GWLR state



APPLICATION PERFORMANCE ALSO DEPENDS ON WHAT SUBSET OF GWLR STATE IS OFFLOADED

ATTACH <= 20% GSCO = 2.11X Centralized
20% < ATTACH <= 60% LSCO = 1.23X Centralized
ATTACH > 90%

Ongoing work

- Evolving the cost metric using **dynamic parameters**

  Goals:

  - Improve accuracy
  - Reduce parameter capture & monitoring overheads

- Implement the **Online Adaptive Offload** framework

Conclusion

- **Application performance** depends on:
  - Controller Scalability design chosen
  - Subset of GWLR state offloaded
- There is need for an **Online Adaptive Offload**
- LSCO/GSCO **reduces traffic to the ROOT controller**, enabling controller scale