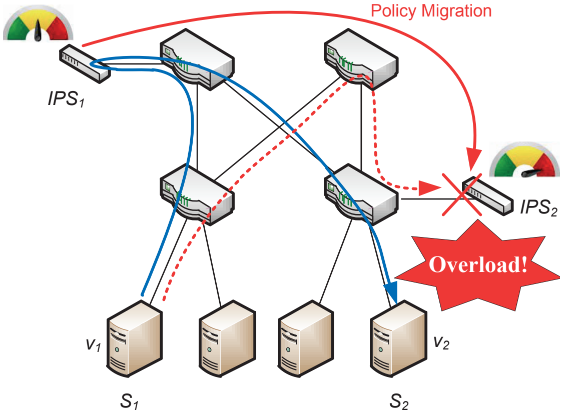
Experimental Evaluation of SDN-Controlled, Joint Consolidation of Policies and Virtual Machines

Wajdi Hajji∗ **Posco Tso**∗ Lin Cui† Dimitrios Pezaros‡

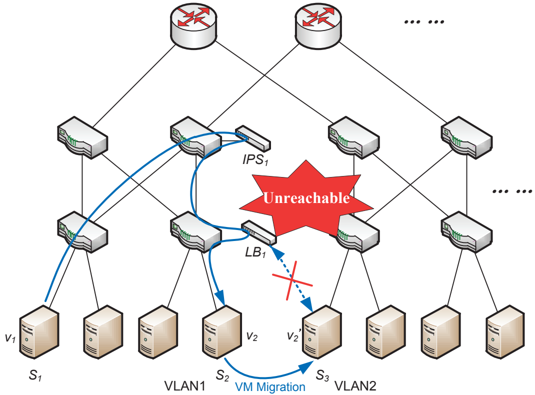
∗Loughborough University, UK
†Jinan University, China
‡Glasgow University, UK

Loughborough University

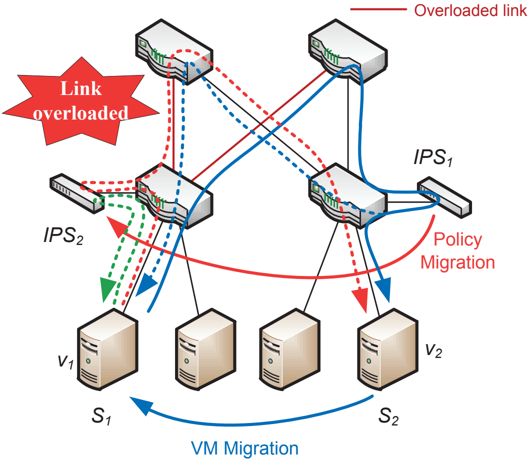
Middlebox challenges



Middlebox challenges



Middlebox challenges



Middlebox challenges

- We have proposed Sync: *Synergistic Policy and Virtual Machine Consolidation in Cloud Data Centers*[1].
- Plain lanuage: Sync migrates virtual machines and network policies at the same time.

[1] L. Cui, R. Cziva, F. P. Tso and D. P. Pezaros, "Synergistic policy and virtual machine consolidation in cloud data centers," IEEE INFOCOM 2016, San Francisco, CA, 2016, pp. 1-9. doi: 10.1109/INFOCOM.2016.7524354

Middlebox challenges

- We have proposed Sync: *Synergistic Policy and Virtual Machine Consolidation in Cloud Data Centers*[1].
- Plain lanuage: Sync migrates virtual machines and network policies at the same time.

Does it scale in a real data centre environment?

[1] L. Cui, R. Cziva, F. P. Tso and D. P. Pezaros, "Synergistic policy and virtual machine consolidation in cloud data centers," IEEE INFOCOM 2016, San Francisco, CA, 2016, pp. 1-9. doi: 10.1109/INFOCOM.2016.7524354

How does *Sync* work? – Sync algorithms

Get Communicating VM Groups

The algorithm partitions all VMs into isolated groups in which VMs do not communicate with a VM outside their group. These VM groups will be the input of other algorithms.

Policy Migration

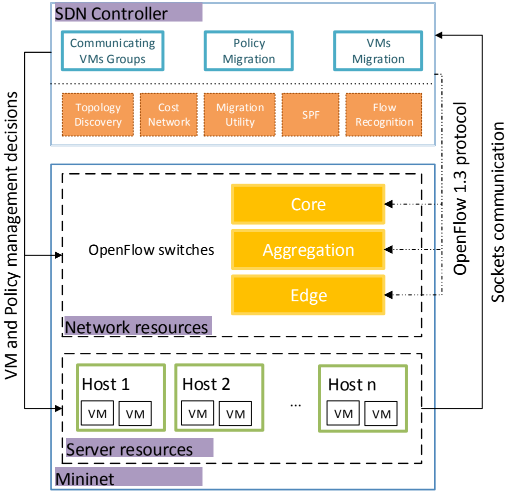
This algorithm focuses on migrating the policies, in other words defining again the MBs; replace them with the same type of MBs as the deployed ones.

VM Migration

The VM migration algorithm, for a given VM group, initialises and obtains the preference list (where no policy violation or overused server capacity) of all servers.

System architecture

The topology and the controller communicate through OpenFlow to add rules to switches and via out-of-band control channel.



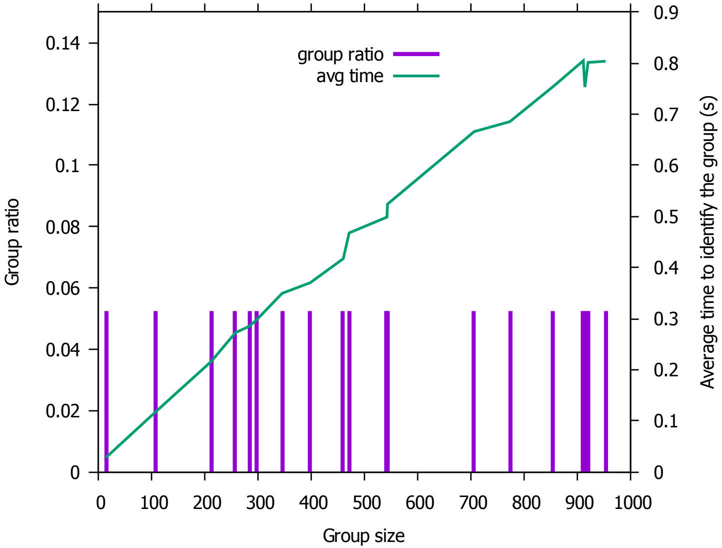
Source code

Source code available on GitHub
https://github.com/wajdihajji

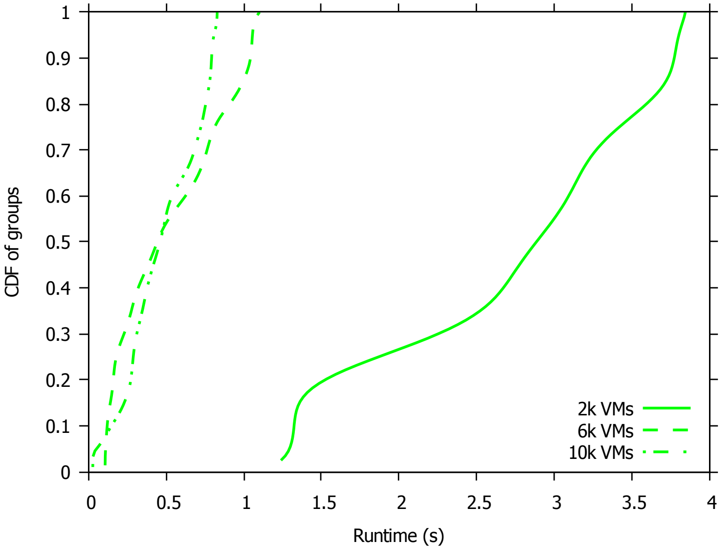
Experiment Environment

- We have run our experiments on two identical servers, each has 8 Cores/1.2Ghz CPU and 8GB Memory. Both servers have Ubuntu 14.04 is running atop.
- In server A, we have installed Mininet version 2.3.0d1, OpenFlow 1.35 and Python 2.7.6.
- In server B, we have installed Ryu controller 4.10.
- Two servers are connected through a 1Gbps switch.

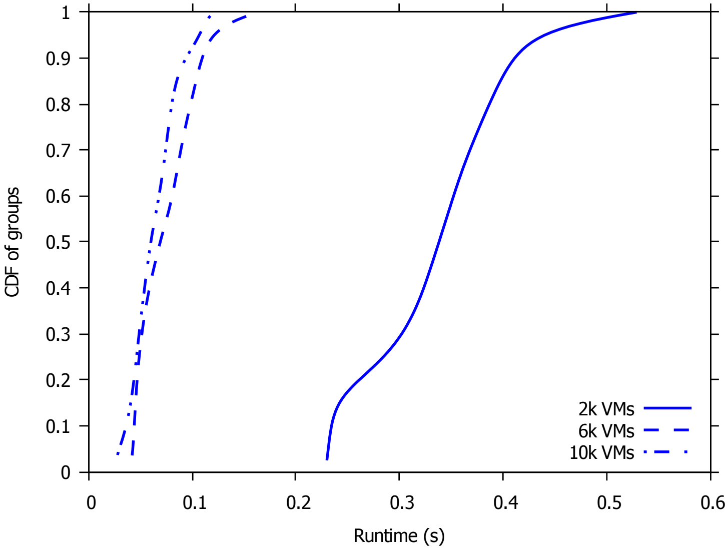
Group Formation



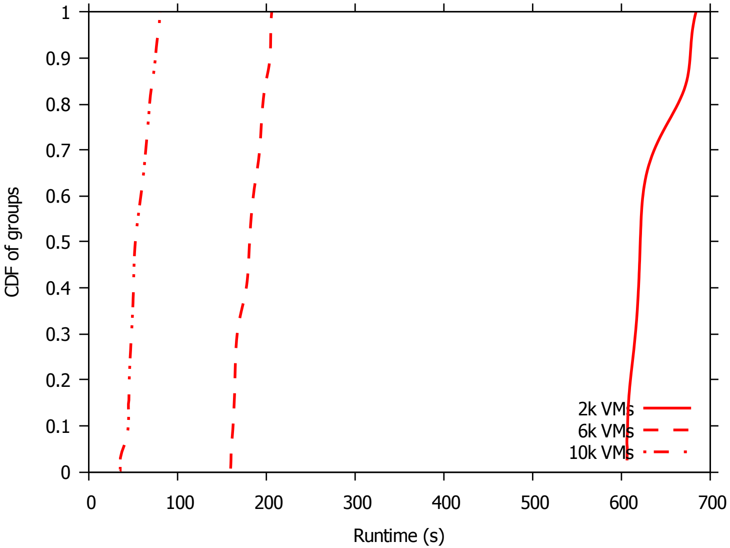
Sync runtime with growing number of VMs – VM groups



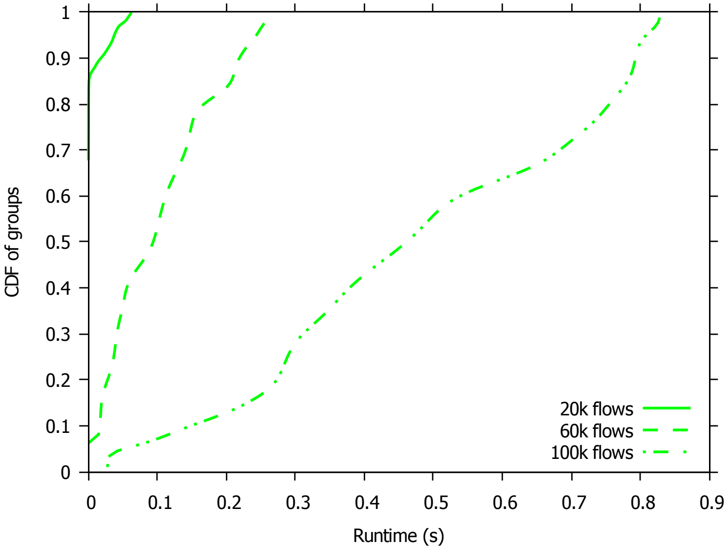
Sync runtime with growing number of VMs – Policy Migration



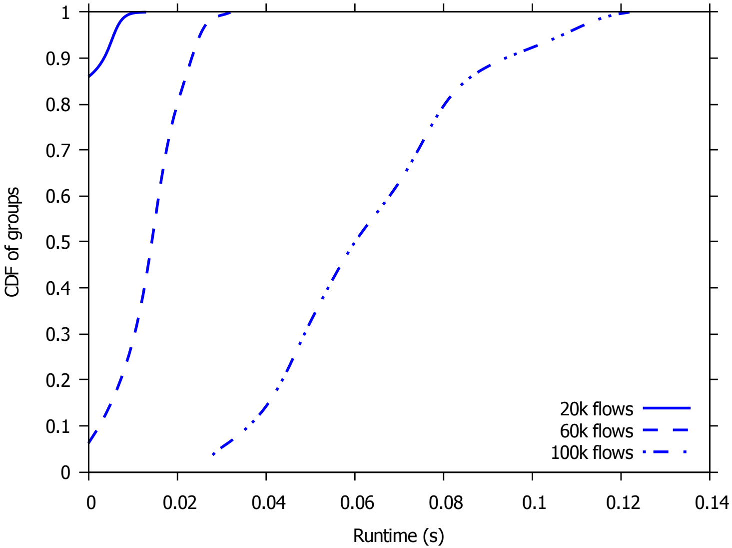
Sync runtime with growing number of VMs – VM Migration



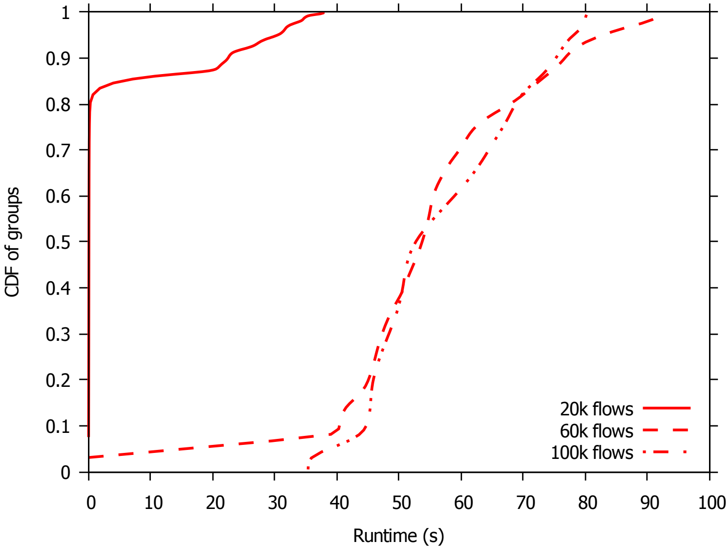
Sync runtime with growing number of flows – VM groups



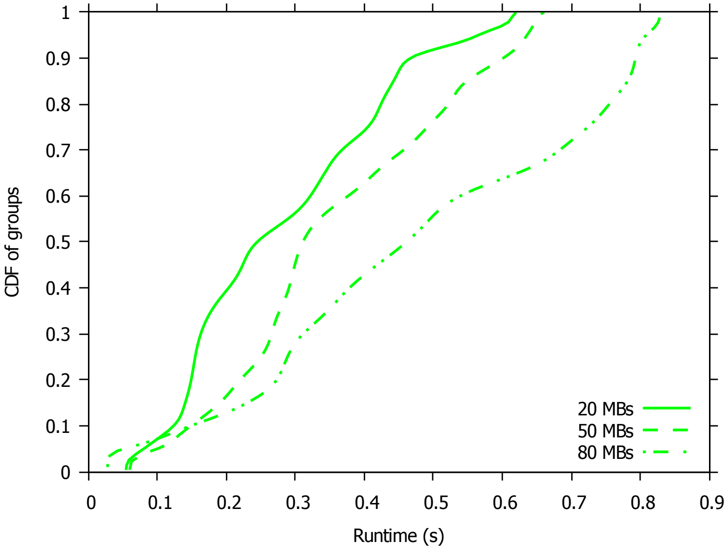
Sync runtime with growing number of flows – Policy Migration



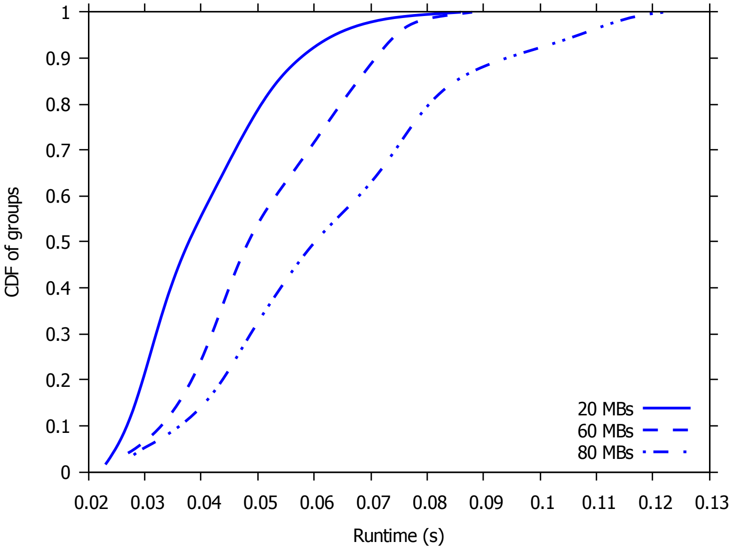
Sync runtime with growing number of flows – VM Migration



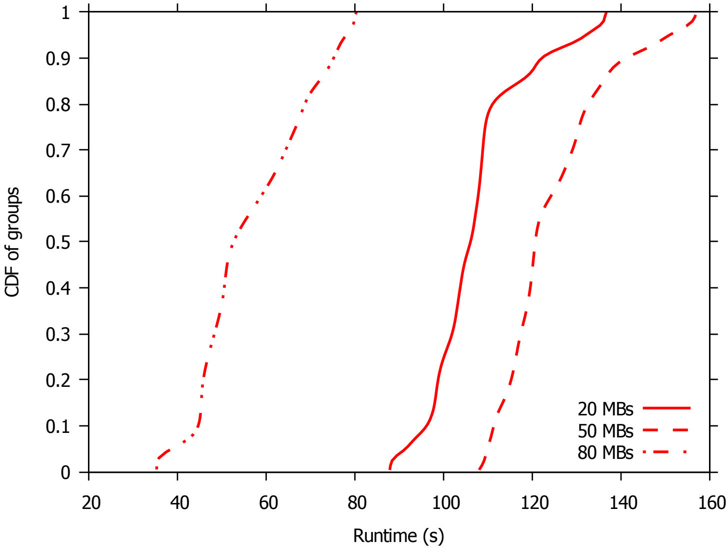
Sync runtime with growing number of MBs – VM groups



Sync runtime with growing number of MBs – Policy Migration



Sync runtime with growing number of MBs – VM Migration



Conclusions

- The number of VMs has a measurable effect on *Get communicating VM Groups* and *Policy migration* on one hand, and *VM migration* on the other hand.
- The three factors have a different impact on the *Sync* algorithms, flows impacts more *Get Communicating VMs* and *Policy Migration* algorithms, while the number of VMs can significantly alter the time needed by *VM migration* algorithm.
- The number of MBs has a known effect on *Get communicating VM Groups* and *Policy migration*, whereas, in *VM migration*, its impact becomes unpredictable because VM migration decision depends more on policy violation prevention strategy.
- Because of its fractional use of CPU resources, *Sync* is very resource efficient and has room to scale to much bigger topologies.

Thank you! Questions?