Inter-Flow Consistency: Novel SDN Update Abstraction for Supporting Inter-Flow Constraints

Weijie Liu* (wliu43@illinois.edu),
Rakesh B. Bobba†, Sibin Mohan*, Roy H. Campbell*

* University of Illinois at Urbana-Champaign

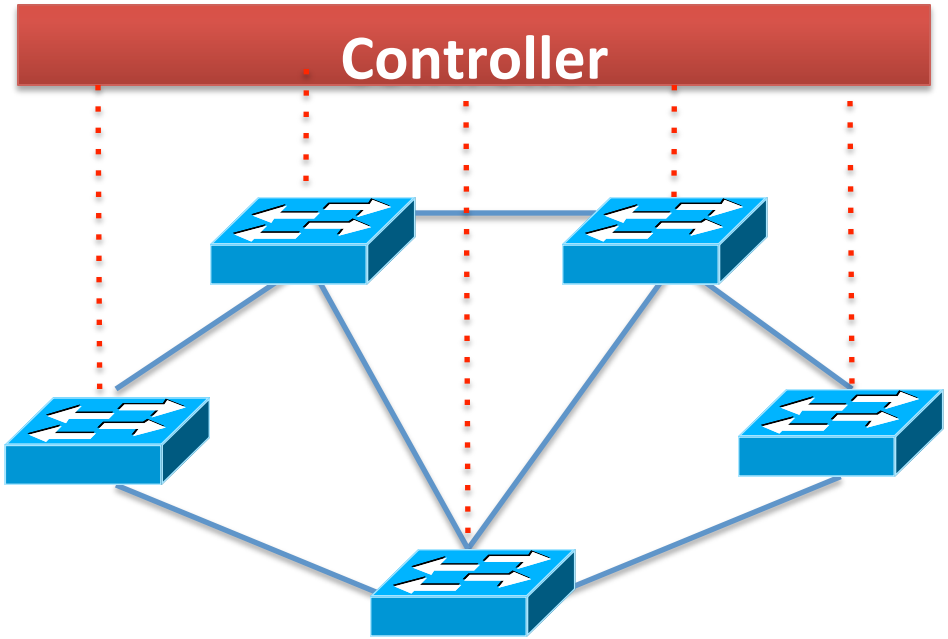
† Oregon State University

Outline

- **SDN & Inter-flow Consistency**
- **Our Approach**
- **Experiments**
- **Conclusion**

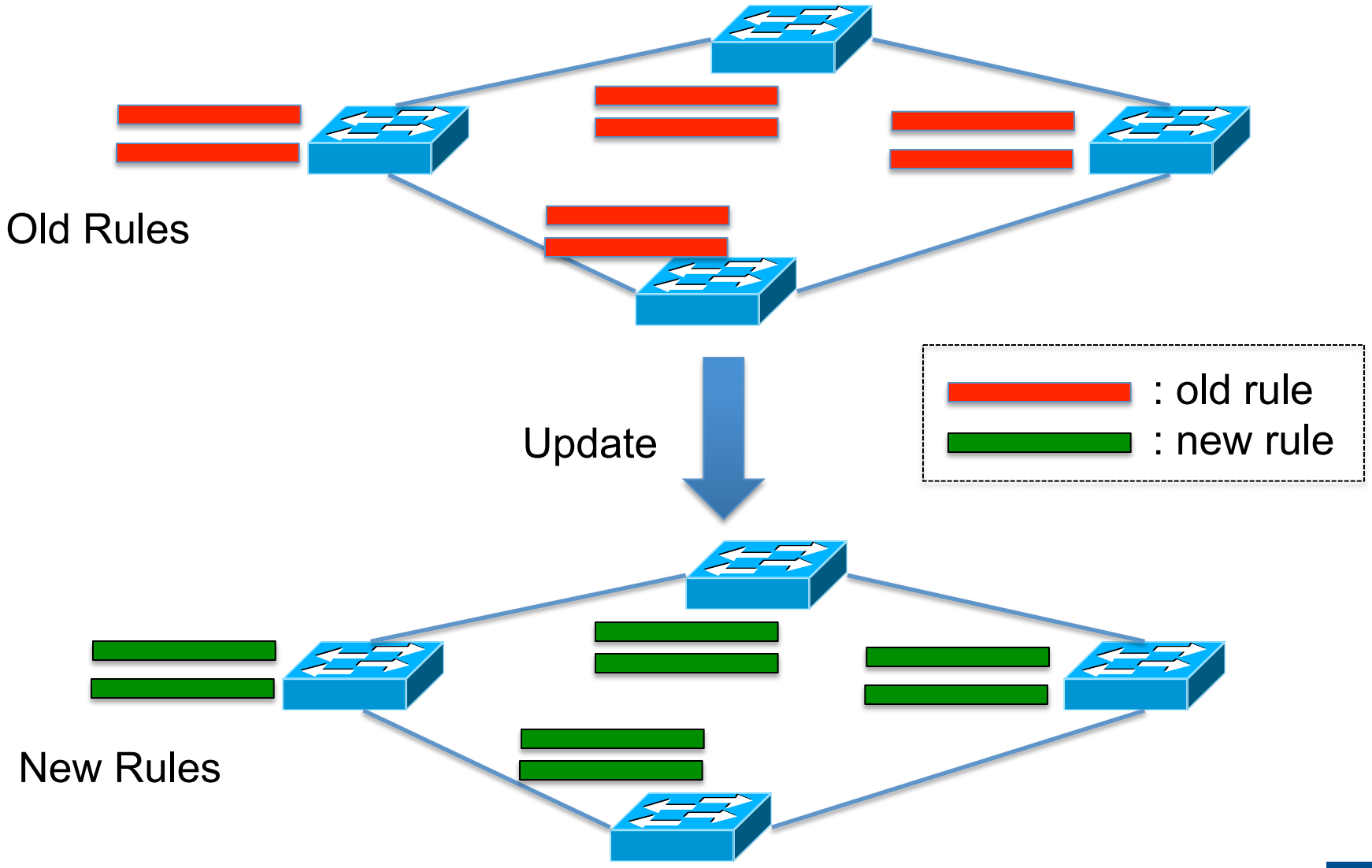
Software Defined Networks (SDN)

- **Decouple** Control Plane and Data Plane
- Controller installs **forwarding rules** in switches



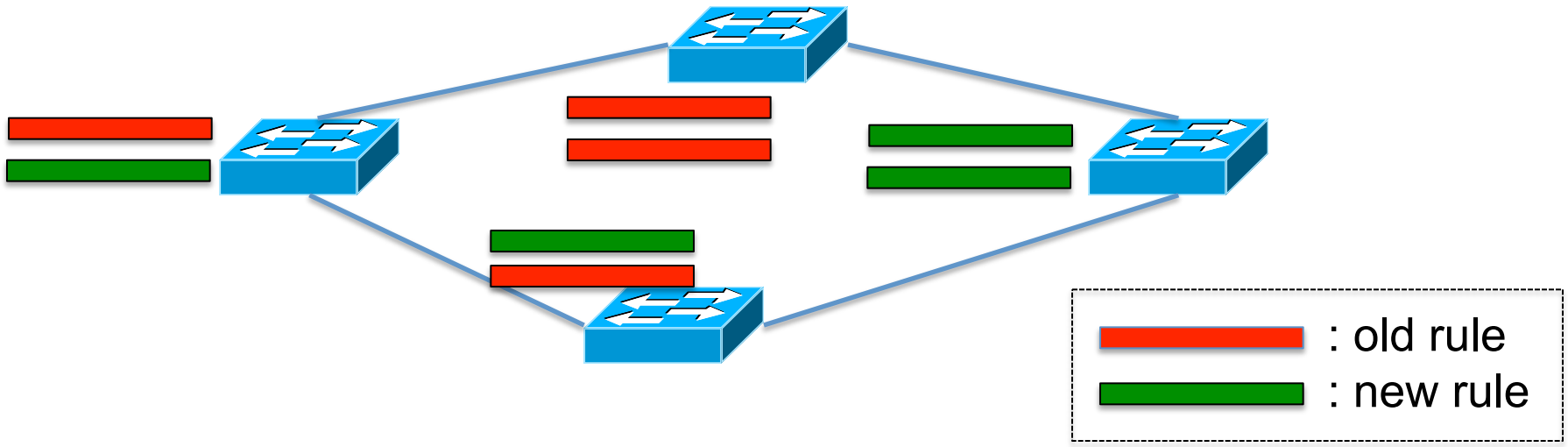
Software Defined Network

Update Forwarding Rules



Update Inconsistency

- **Fail** to update all the devices **at the same time**
- Packets processed by both **old** & **new** rules
- **Problems:**
  - loops, black hole, congestion …



Update Inconsistency

- **Fail** to update all the devices **at the same time**
- Packets processed by both **old** & **new** rules
- **Problems:**
  - loops, black hole, congestion …
- **Existing Solutions***:
  - **Per-packet Consistency:** processed by either old or new
  - **Per-flow Consistency:** processed by either old or new

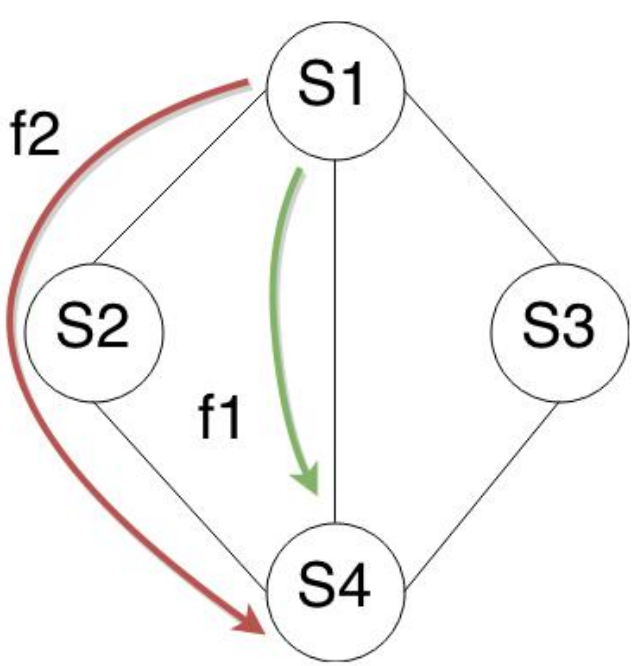
* Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. ACM SIGCOMM 2012.

Inter-flow Constraints

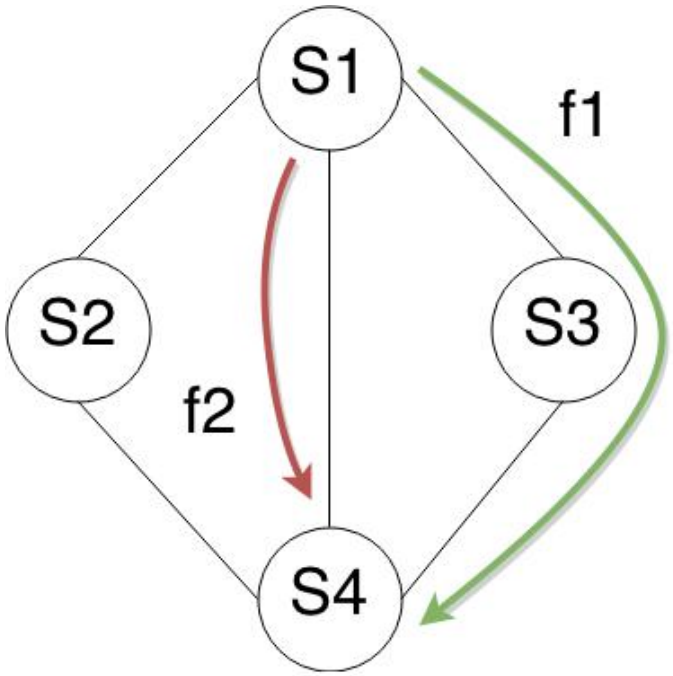
- Enforcing **constraints** across **different flows**
  - for Security or Reliability
  - Ex 1. Power Grid: isolation of critical control flows from engineering flows
  - Ex 2. Network Operator: isolation between data flows of diff companies
  - Ex 3. Data Center: related data flows need to be updated at the same time

Question: Will these **constraints** be respected during SDN updates?

Example I



(a) Original Configuration

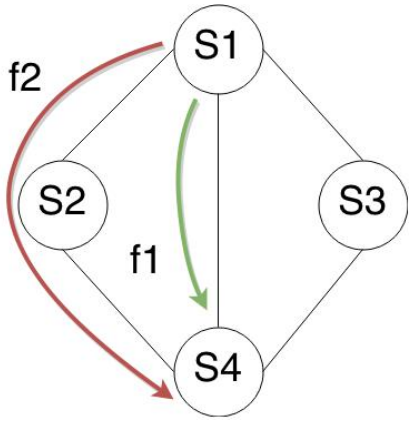


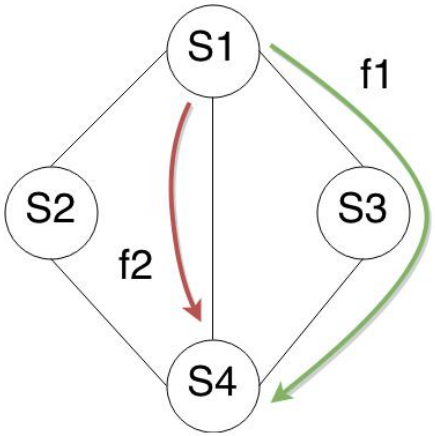
(b) Target Configuration

- Security Policy: f1 & f2 should **NOT** pass through the same link

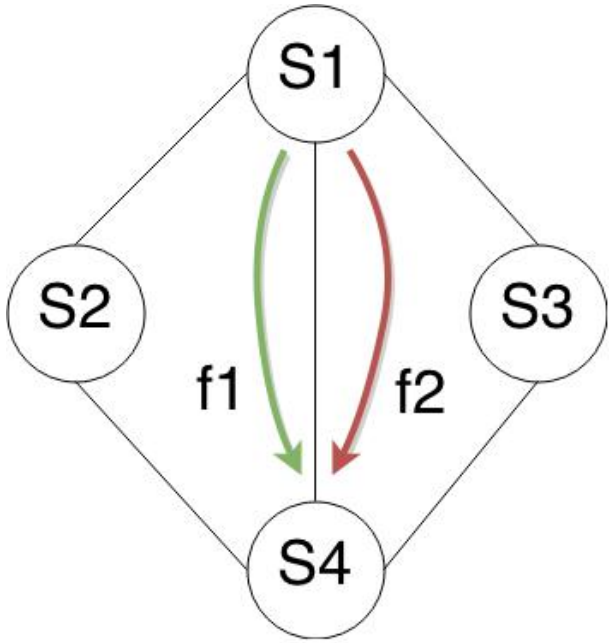
Example I

- What if f2 gets updated before f1?





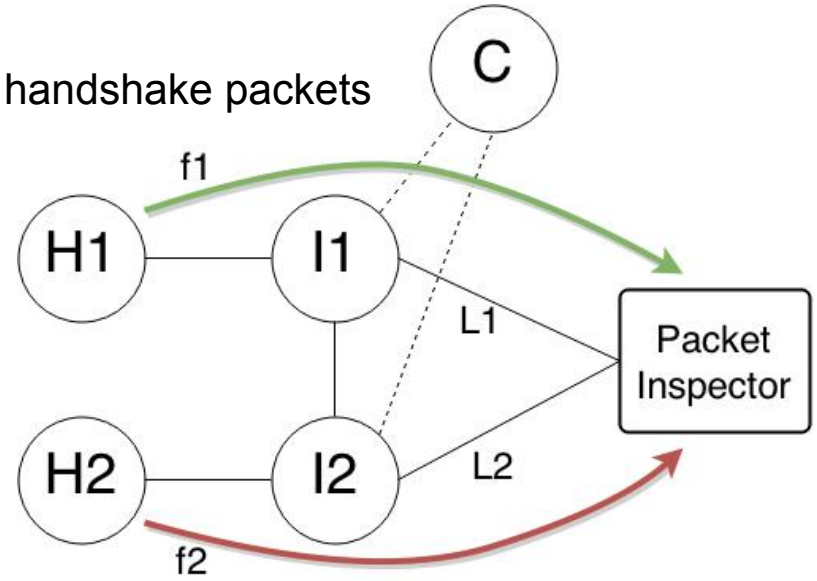
(a) Original Configuration (b) Target Configuration



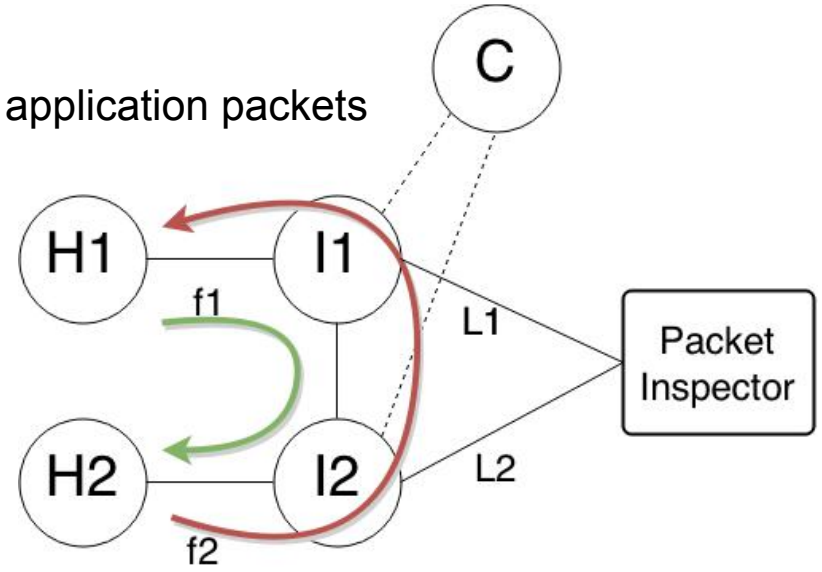
(c) Transitional Configuration

Example II*

- H1 & H2: first inspected, then talk with each



(a) Original Configuration

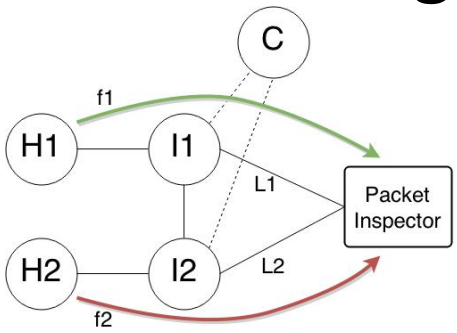


(b) Target Configuration

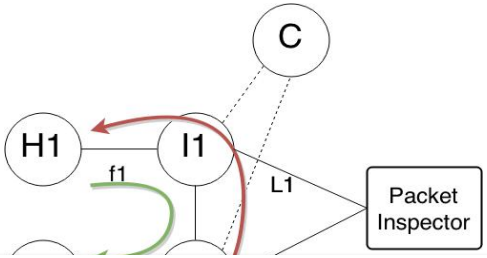
*Soudeh Ghorbani and Brighten Godfrey. Towards correct network virtualization. HotSDN, 2014.

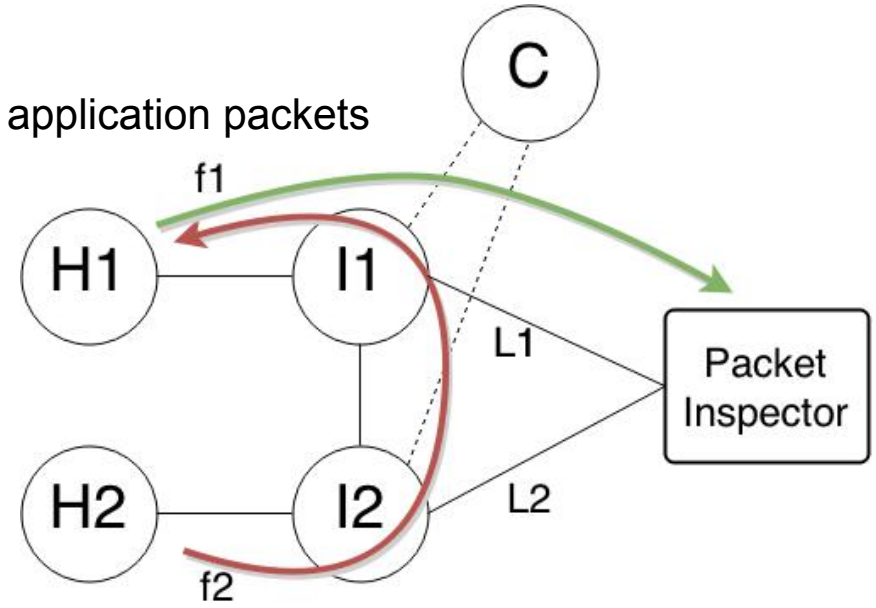
Example II*

- What if f2 gets updated before f1?



(a) Original Configuration





f1 & f2 should be updated at the same time; not guaranteed by existing update mechanisms

Observation:

Inter-flow constraints may be violated during SDN updates.

Problem:

Can we **schedule** update operations to **guarantee inter-flow** constraints during updates?

Theoretical Abstraction

- We propose: **Interflow Consistency**

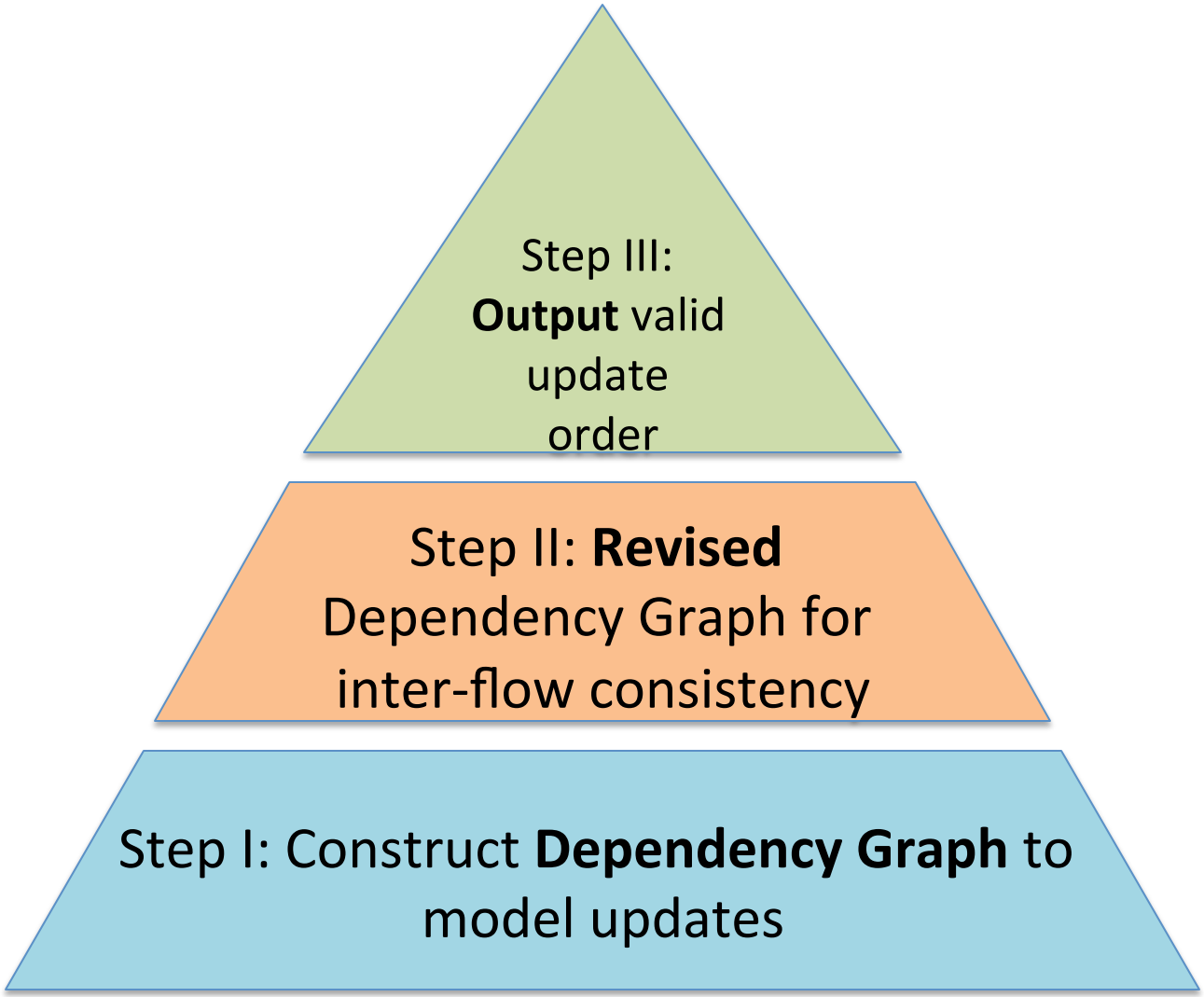
Spatial Isolation:

Packets from different flows cannot pass through the same link or device

Version Isolation:

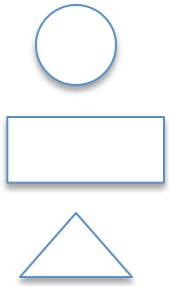
Packets from different related flows cannot be processed by two different versions of flow rules

Our Approach: 3 steps



Step I: Construct Dependency Graph*

- **Dependency Graph** (DG)
  - 3 types of node:



Operation Node (add/delete/modify a rule)

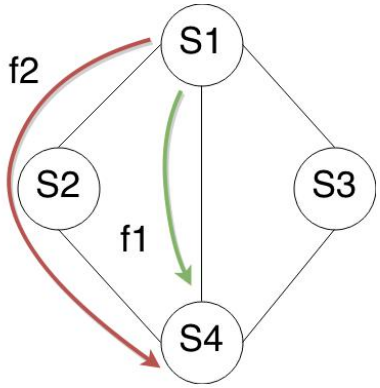
Path Node (links passed by a flow)

Resource Node (link bandwidth)

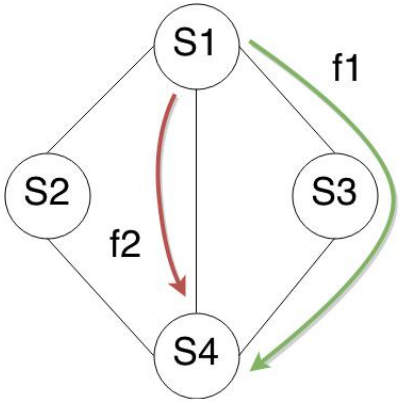
  - Direction of edge between 2 nodes:
    - Resource Consumption
    - Operation Dependency

*Xin Jin, et al. Dynamic scheduling of network updates. *SIGCOMM*, 2014.

Step I: Example I



(a) Original Configuration



(b) Target Configuration

We need 4 operations nodes:

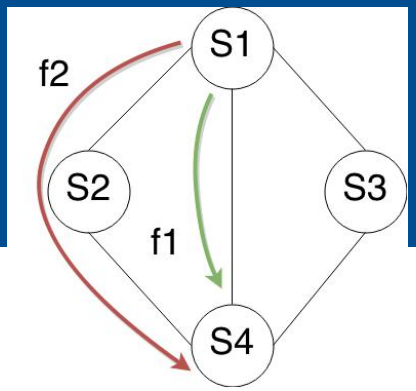
ID	Entity	Update Operation
a	S_3	Add: forward f_1 to S_4
b	S_1	Modify: forward f_1 to S_3
c	S_1	Modify: forward f_2 to S_4
d	S_2	Delete rules of f_2

Also, 4 path nodes:

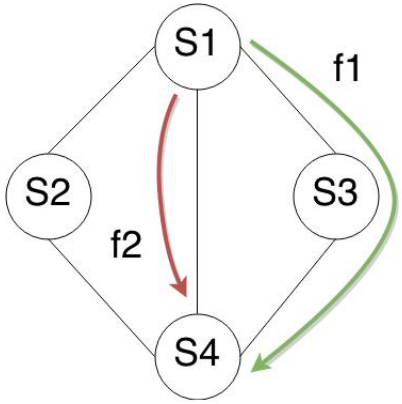
p1: f1's old path;
p2: f2's old path;
p3: f1's new path;
p4: f2's new path.

5 resource nodes for each link

Step I: construct DG

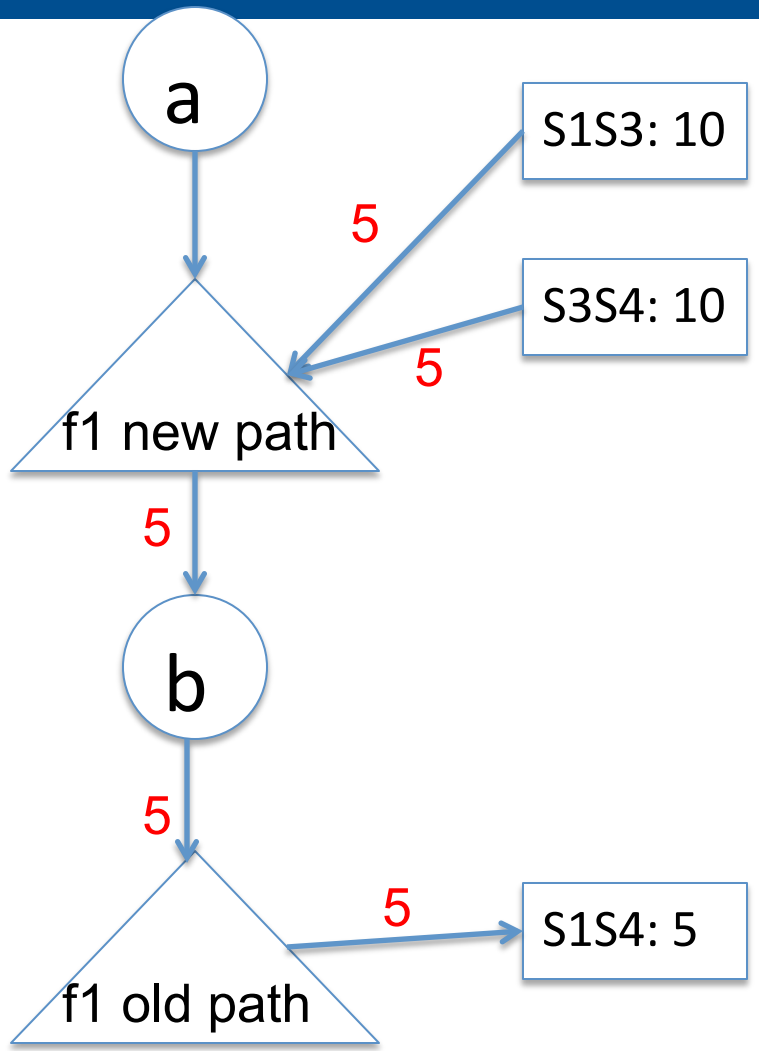


(a) Original Configuration



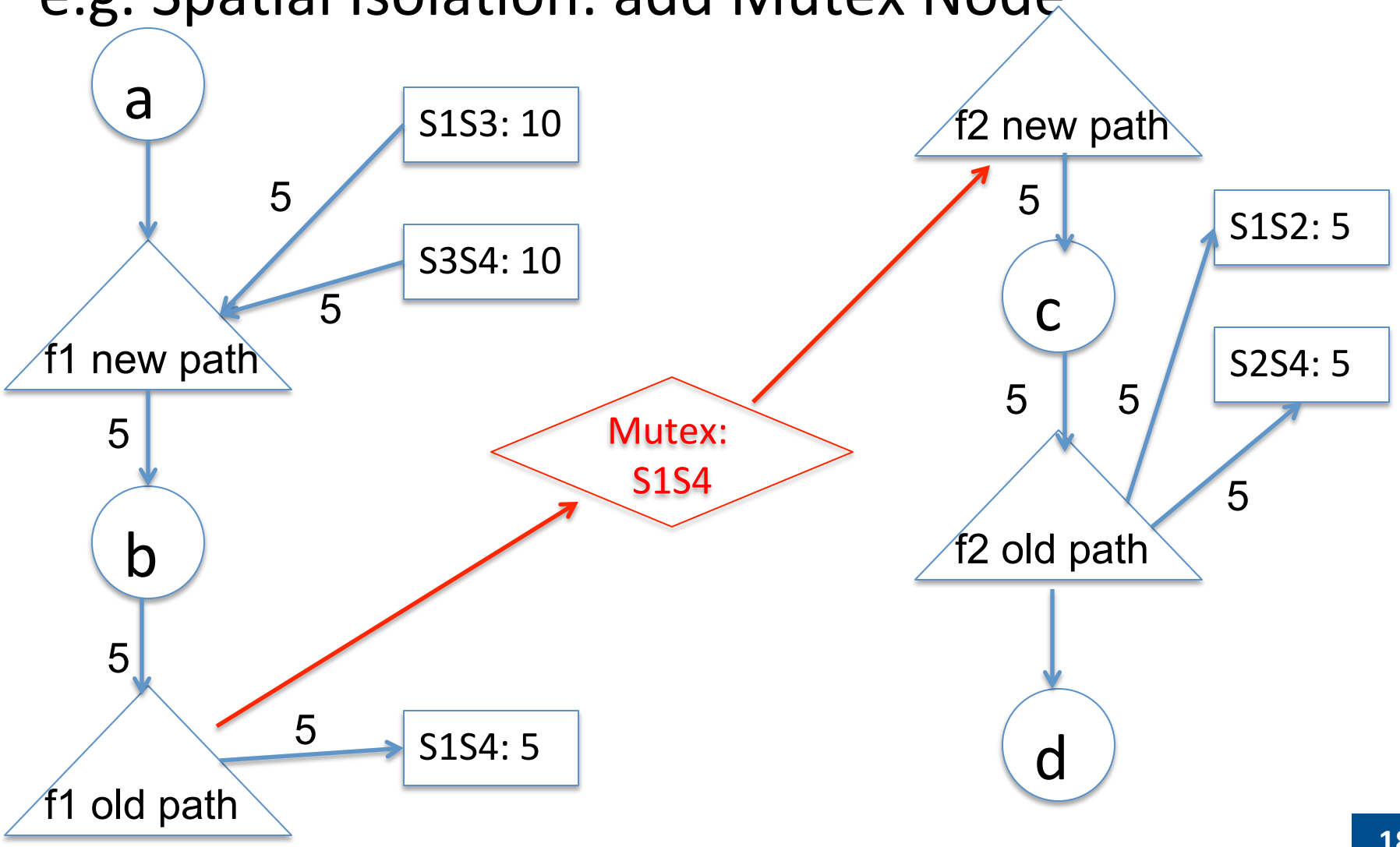
(b) Target Configuration

ID	Entity	Update Operation
a	S_3	Add: forward f_1 to S_4
b	S_1	Modify: forward f_1 to S_3
c	S_1	Modify: forward f_2 to S_4
d	S_2	Delete rules of f_2



Step II: **Revised** DG for inter-flow

- e.g. Spatial Isolation: add Mutex Node



Step III: Output Operation Sequence

do:

    for each Operation Node, *O*:

        if O has no operation ancestors & has sufficient resources:

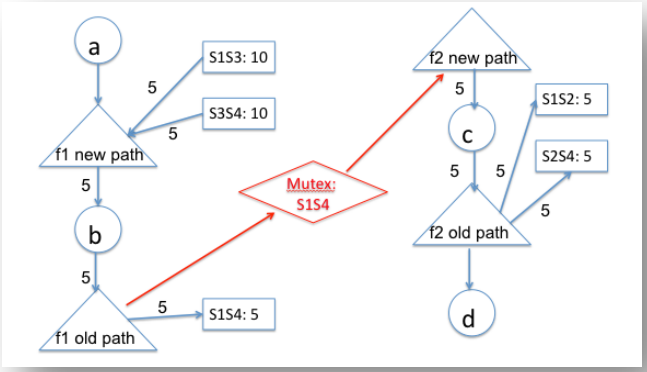
            Schedule O;

            Delete O;

        end if

    end for

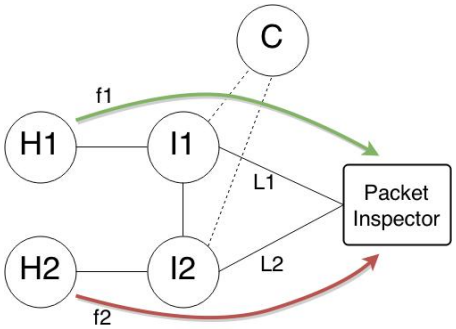
until there is no *O;*



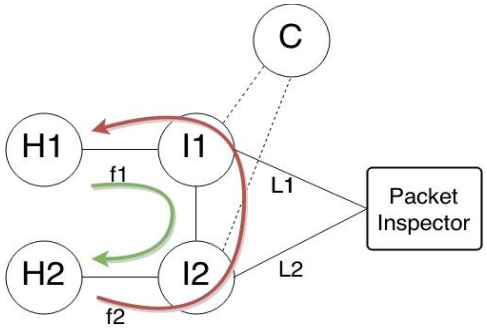
Finally, we can get: a→b→c→d

Version Isolation

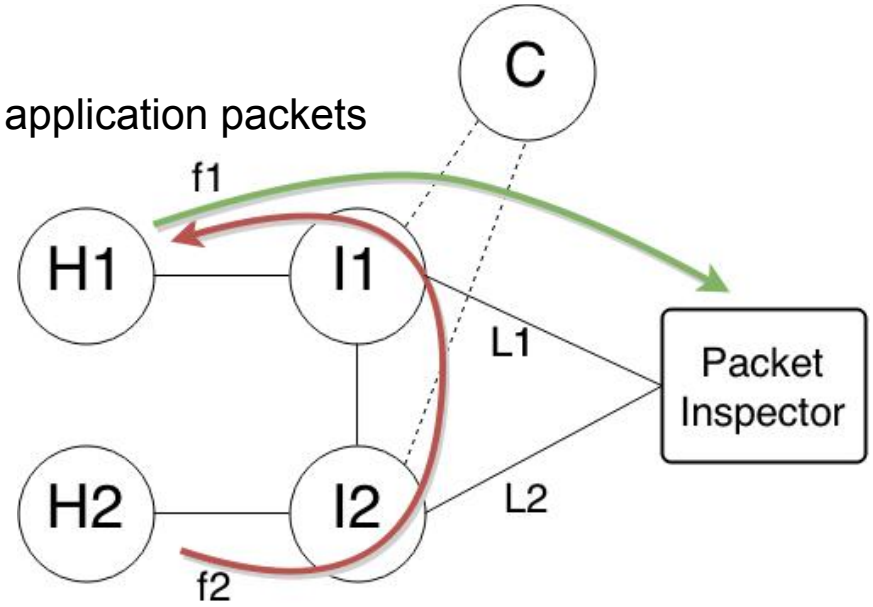
Packets from different related flows cannot be processed by two different versions of flow rules



(a) Original Configuration



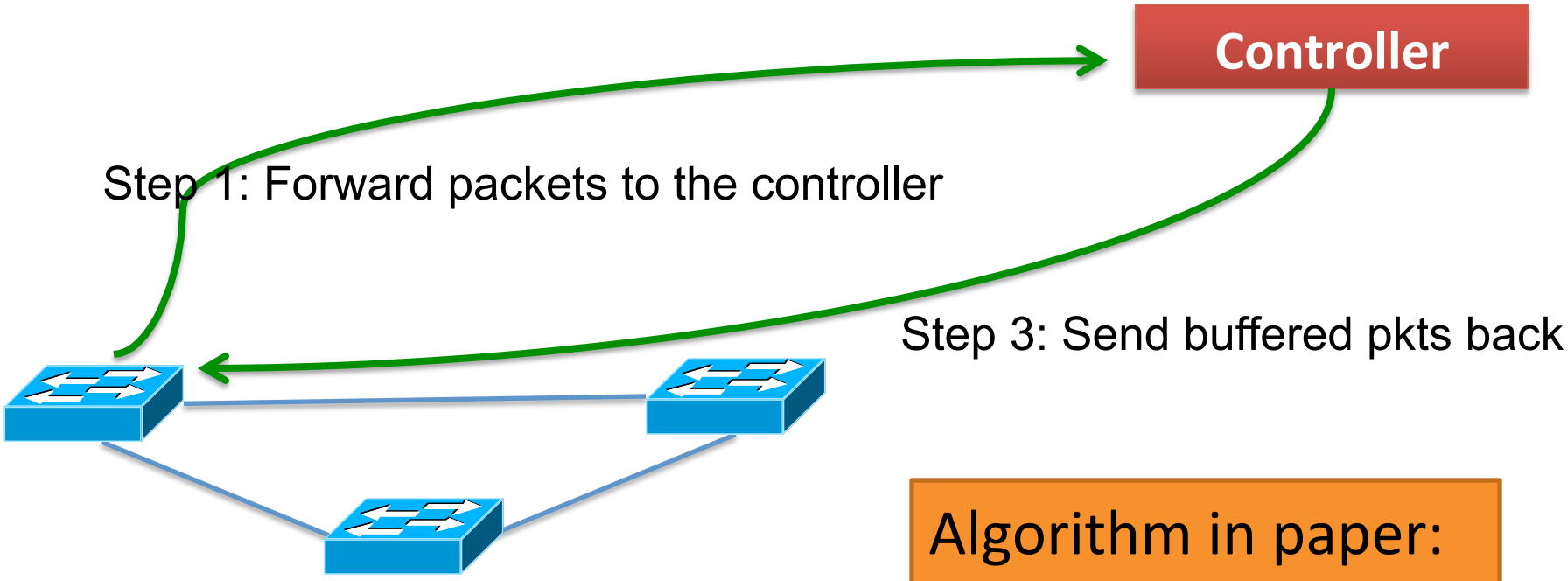
(b) Target Configuration



(c) Transitional Configuration

Solution for Version Isolation

- forward related packets to controller before updates



Algorithm in paper: implemented in DG

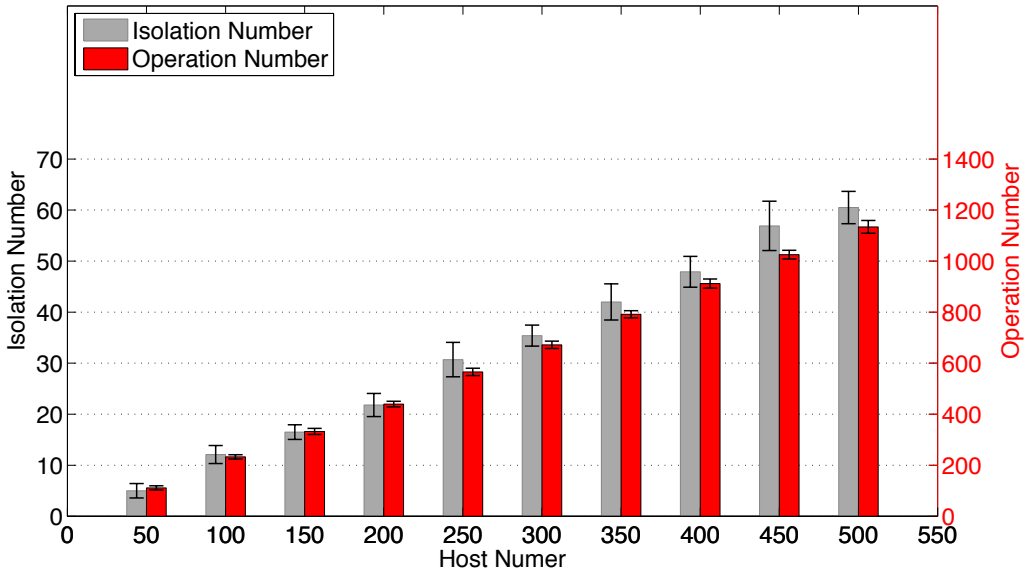
*Rick McGeer. A safe, efficient update protocol for OpenFlow networks. *HotSDN*, 2012.

Experiments

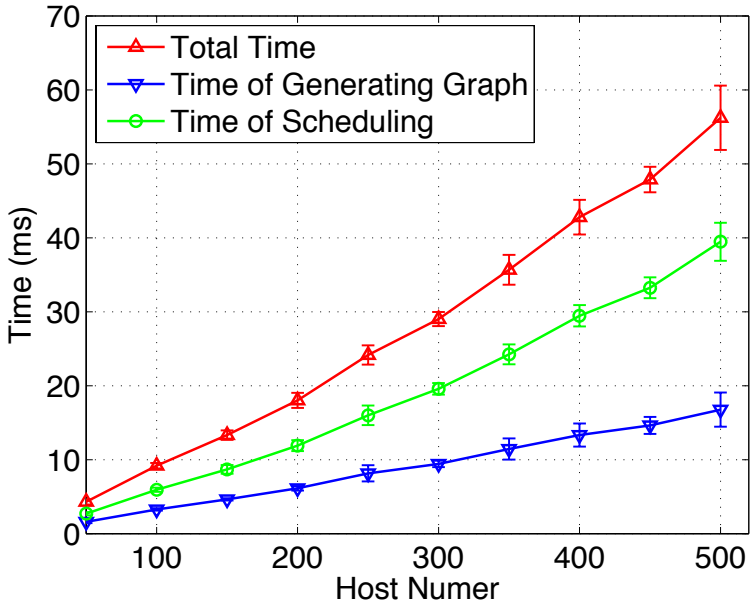
- **A prototype system**
  - Spatial Isolation
  - Version Isolation being implemented

- **Experiment settings**:
  - Network Application: shortest-path routing
  - Control Plane: **Ryu**
  - Data Plane: **Mininet**, a 3-layer tree topology
  - Hardware: Intel i5-2400 3.1 GHz CPU & 16 GB memory

Experimental Results



(a) Number of Isolation Constraints & Update Operations on different Host Numbers



(b) Algorithm Running Time on different Host Numbers

Initial experiments show very good performance

Future Work

- Implementation of version isolation
  - optimized algorithm
- Evaluation
  - in real networks
  - in a large-scale simulation
- Further discussion: inter-flow consistency
  - relationship of two isolations
  - drawbacks

Conclusion

- Inter-flow consistency abstraction:
  - Spatial Isolation
  - Version Isolation
- An approach using dependency graph
- A prototype system
  - a preliminary performance evaluation

Questions?

- feel free to contact: wliu43@illinois.edu

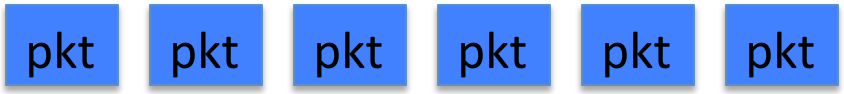
Thank you!

- Thanks to Prof. Carl Gunter for slide template!

Two Consistency Abstractions

- Per-packet Consistency:
  - Each packet will be processed by the old configuration or the new, but NOT the mixture of the two.

- Per-flow Consistency:
  - Each flow will be processed by the old configuration or the new, but NOT the mixture of the two.



Spatial Isolation

- certain flows are not allowed to share a link or a switch before, during and after an update for security and/or reliability reasons.

- E.g. critical flows should be isolated from engineering flows

Version Isolation

- packets from different related flows cannot be processed by two different *versions of flow rules* during its passage through the network.

- E.g. A flow's rules updated from R_{A1} to R_{A2}; another flow's updated from R_{B1} to R_{B2}; the network can have $R_{A1}R_{B1}$ or $R_{A2}R_{B2}$, but not $R_{A1}R_{B2}$ or $R_{A2}R_{B1}$

Enforcing Spatial Isolation

- Randomly generate flows between 2 hosts in the tree-like network (old & new)

- Brute Force Search:
  - for any flow A and another flow B: if they are spatially isolated both in old and new configuration, but not during the transitions (*i.e.,* A's old path overlaps with B's new path) then assign a spatial isolation constraint to A and B.

Controller-buffer Method for Version Isolation[4]

- Basic idea: use controller as a transitional point
  (1) Forward packets to the controller
  (2) Then update rules in switches
  (3) Re-inject buffered packets from controller to data plane

For N flows of Version Isolation, we can first forward N-1 flow to controller, then update rules. After all updates completed, controller sends buffered packets back to network

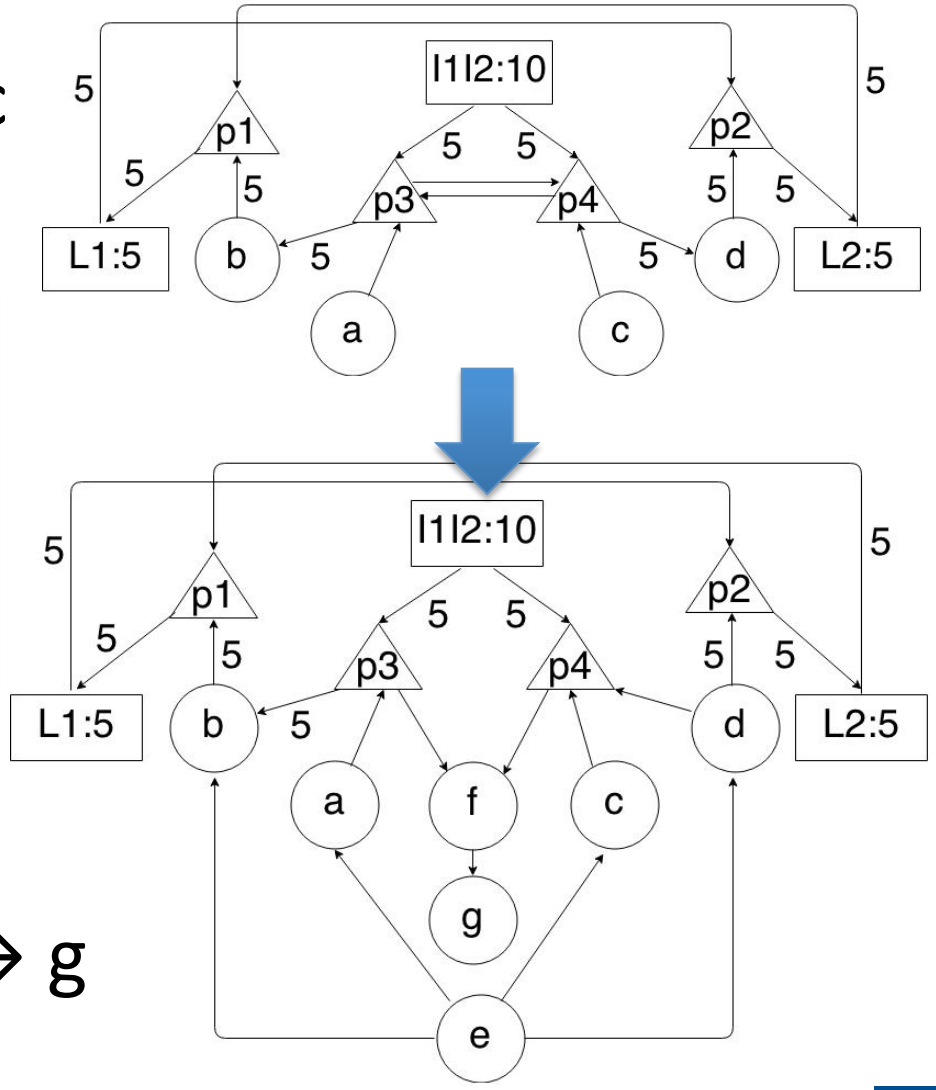
DG Solution for Version Isolation

- After constructing basic DG, add operations to represent:
  - (1) forward certain flows to controller
  - (2) controller sends buffered flows back to network
- Then perform the topological sorting of operations

DG Solution for Version Isolation

- After constructing basic DG, add operations:

e: forward packets to controller;
f: delete rules of "e";
g: controller send packets back to network.

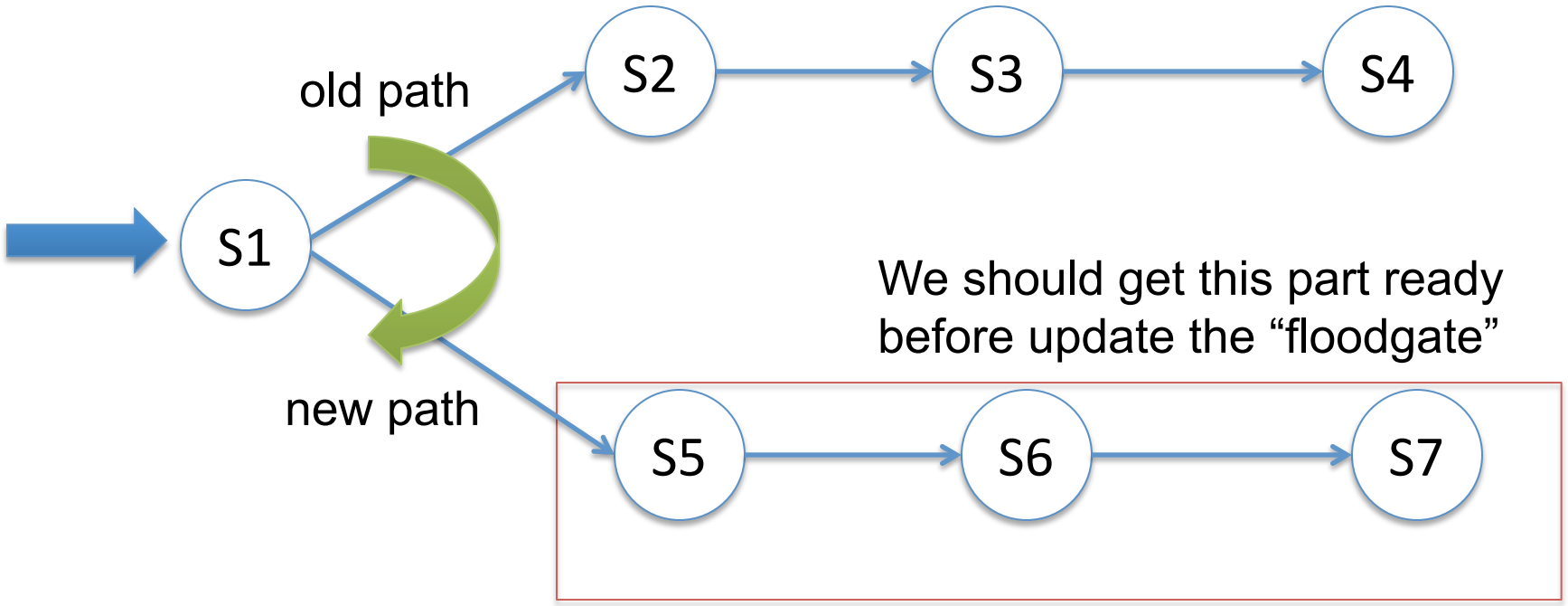


- We can get:

e → a → b → c → d → f → g

Update Order Consideration

- S1:
  - Floodgate node, change the path of flow



Selected References

- [1] Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 323– 334. ACM, 2012.

- [2] Soudeh Ghorbani and Brighten Godfrey. Towards correct network virtualization. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2014.

- [3] Xin Jin, Hongqiang Harry Liu, Rohan Gandhi, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Jennifer Rexford, and Roger Wattenhofer. Dynamic scheduling of network updates. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 539–550. ACM, 2014.

- [4] Rick McGeer. A safe, efficient update protocol for OpenFlow networks. In *Proceedings of HotSDN*, 2012.