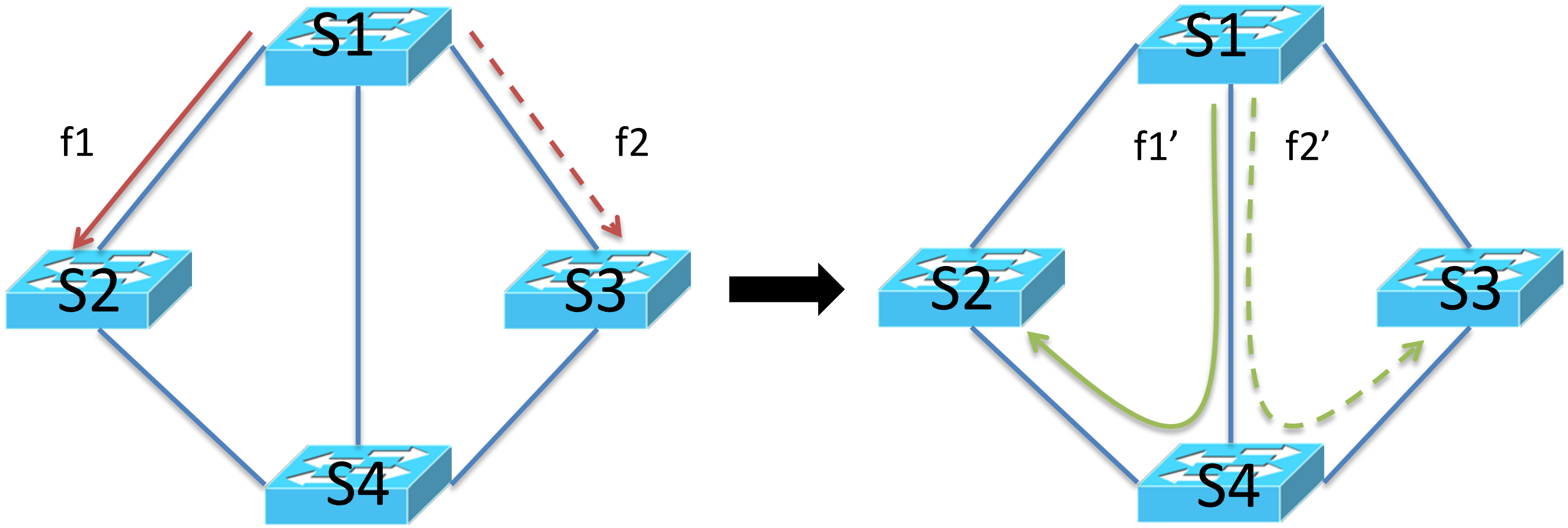
Achieving Efficient and Fast Update for Multiple Flows in SDN

Yujie Liu, Yong Li, Yue Wang,
Athanasios V. Vasilakos, Jian Yuan

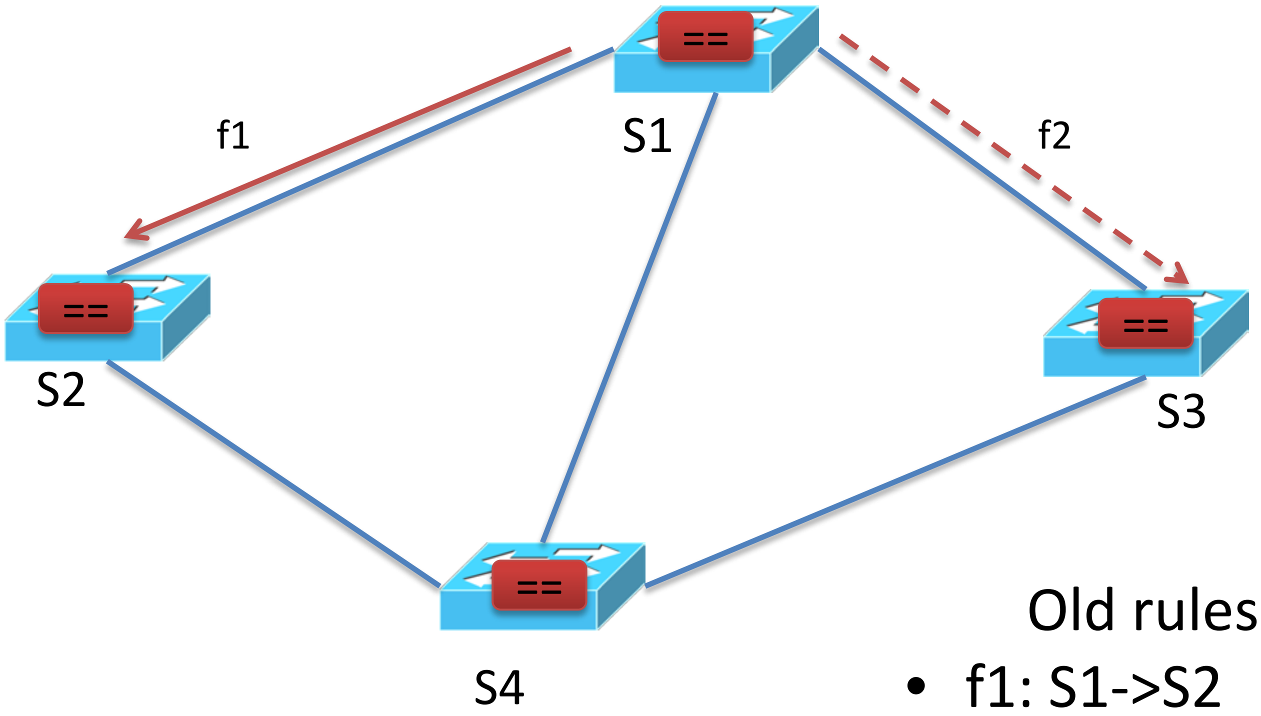
Tsinghua University
University of Western Macedonia

Flow update in SDN



- Network maintenance
- Traffic engineering
- Dealing with failure

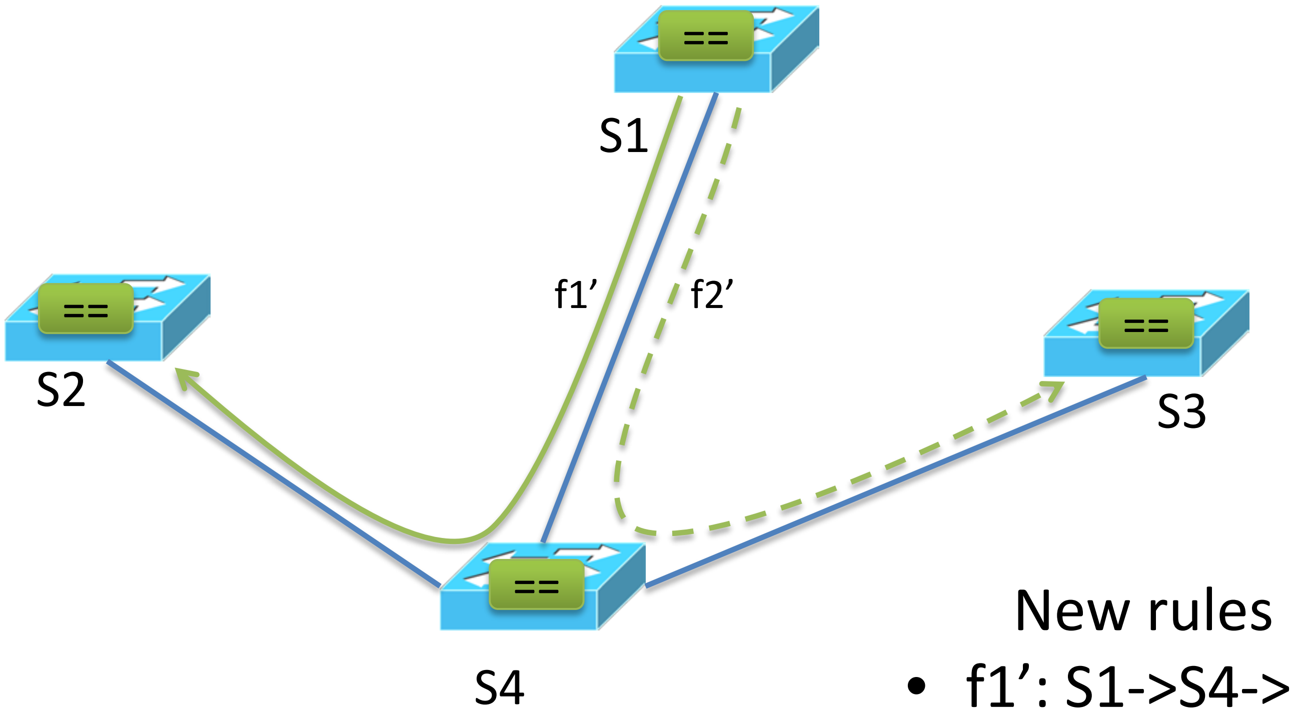
Flow update in SDN



Old rules

- f1: S1->S2
- f2: S1->S3

Flow update in SDN



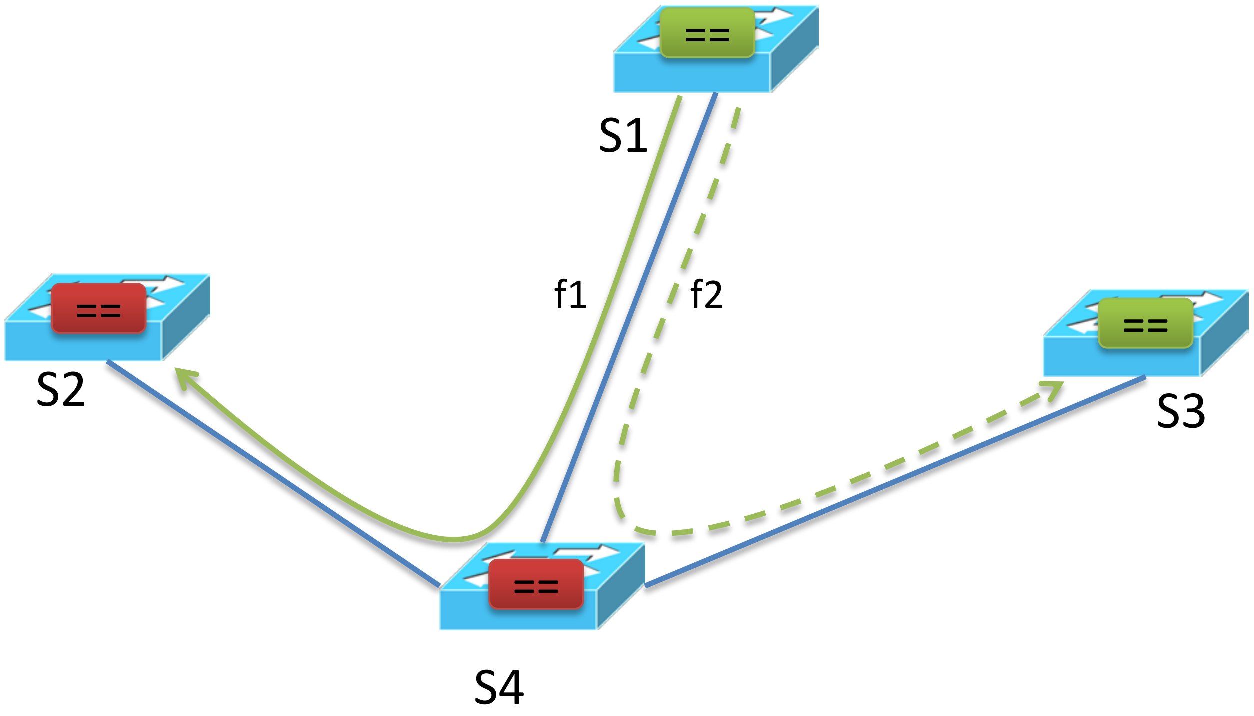
New rules

- f1’: S1->S4->S2
- f2’: S1->S4->S3

Multi-flow update is difficult

- Achieve a flow update which is
  - Consistent
  - Congestion-free
  - Efficient
  - Successful

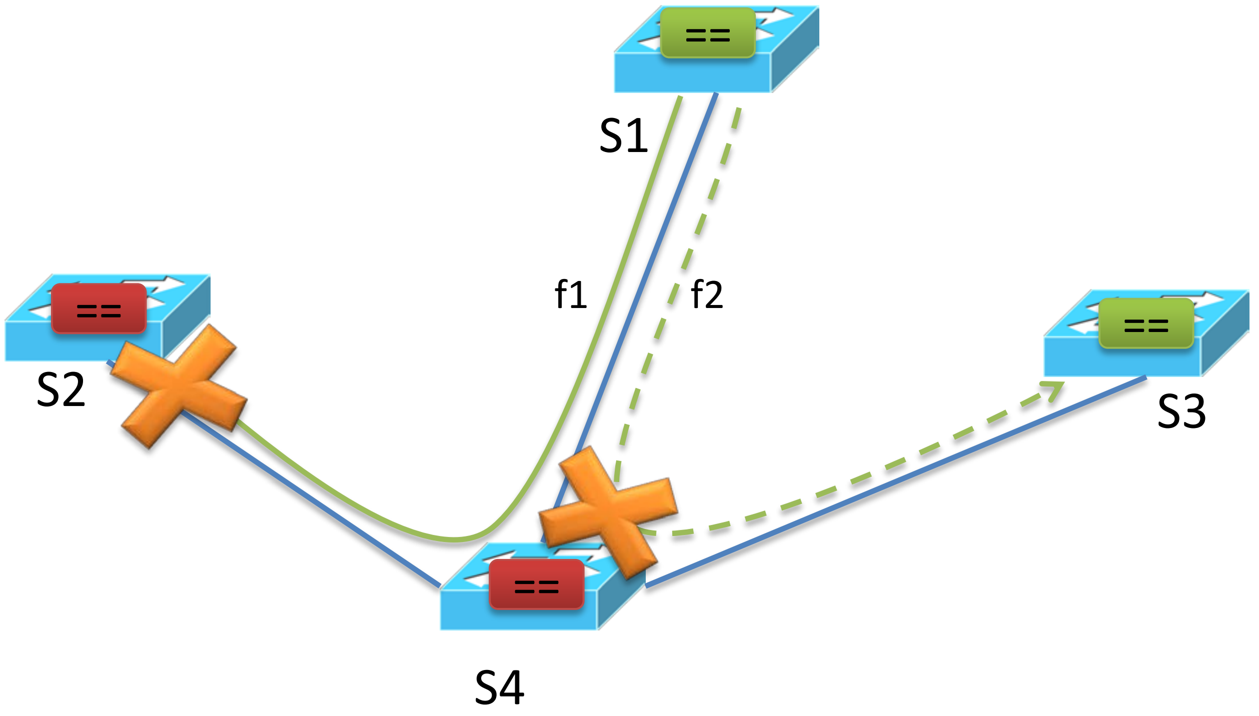
Inconsistent update



S1 and S3 have installed new rules

S2 and S4 have not installed new rules

Inconsistent update

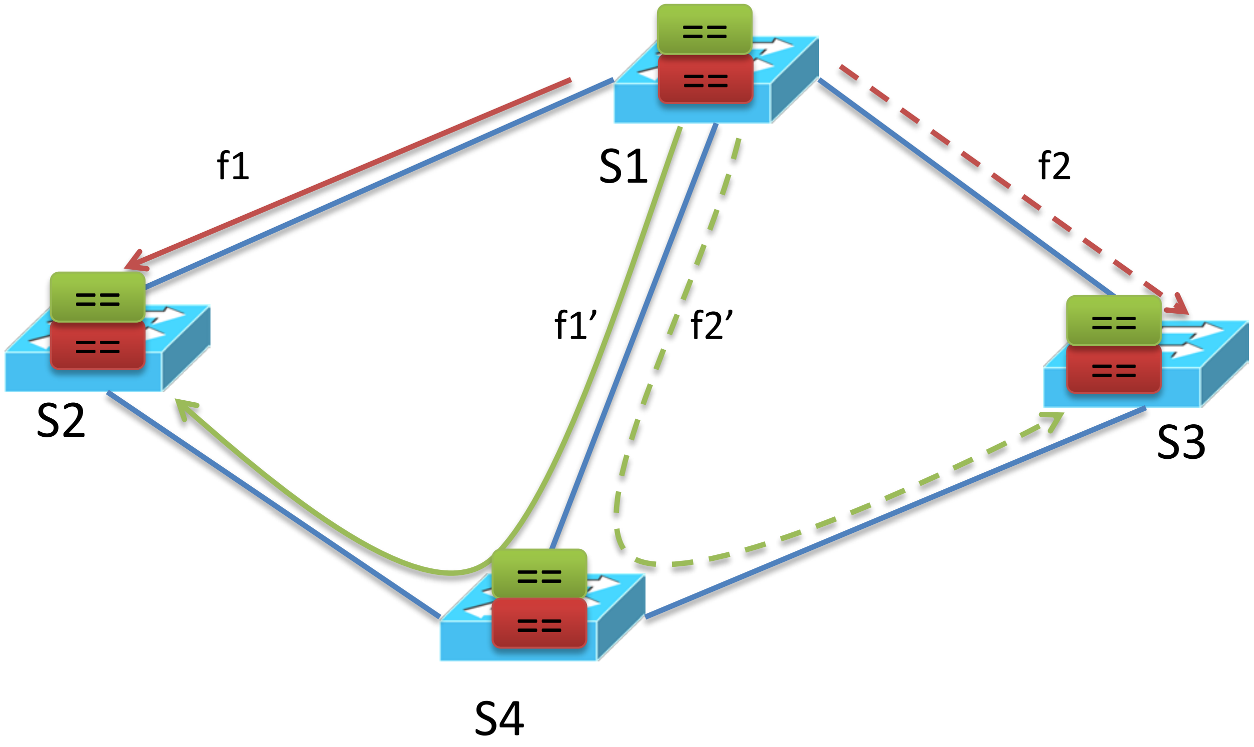


Packet forwarding error

Consistent update

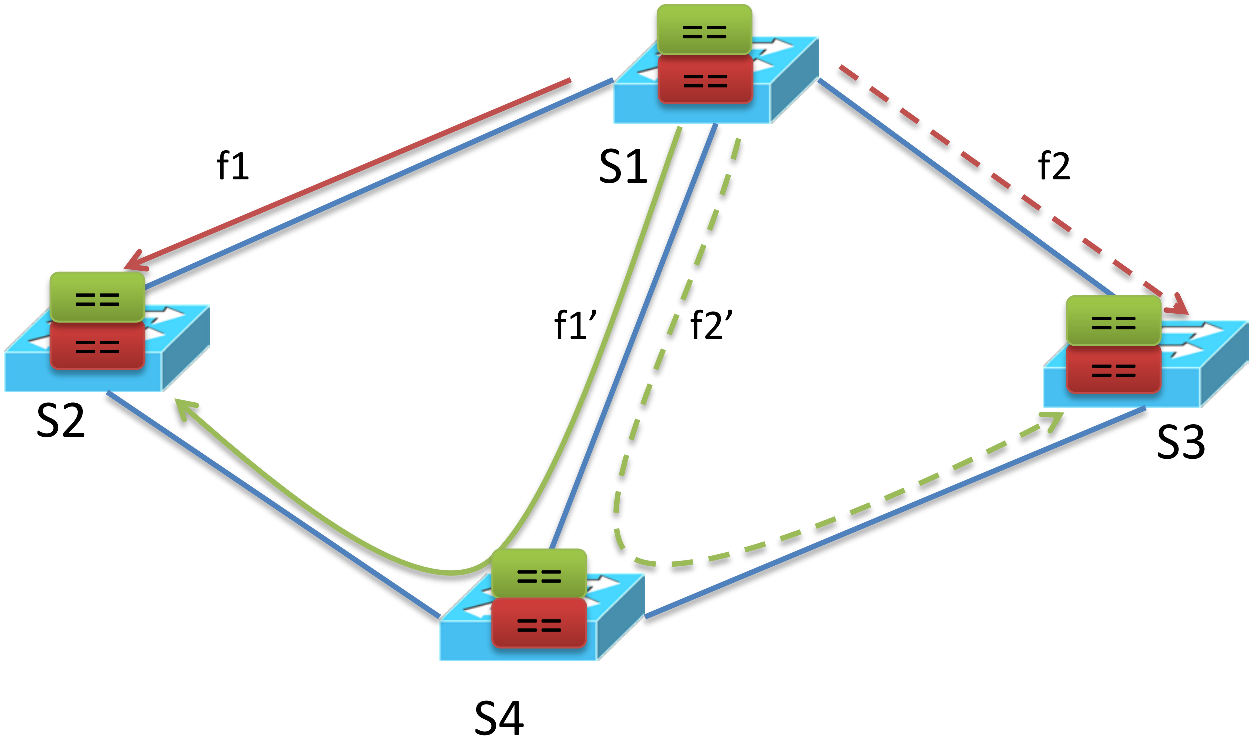
- Per-packet consistent update [Reitblatt’12]
  - A packet is processed by either the new or the old rules, not a mixture of the two.

Solution: 2-phase update [Reitblatt'12]



Keep both the old and the new rules during the update.

Solution: 2-phase update [Reitblatt'12]

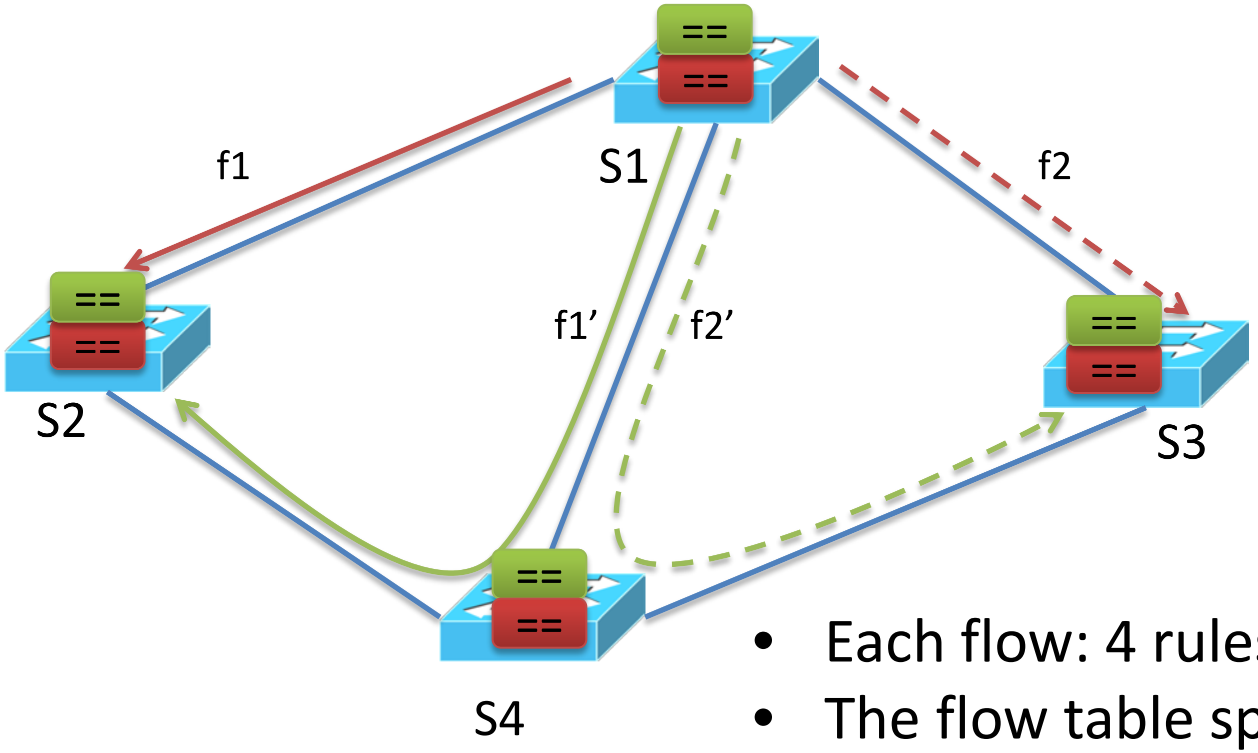


Double flow table space required

The flow table space may not be enough

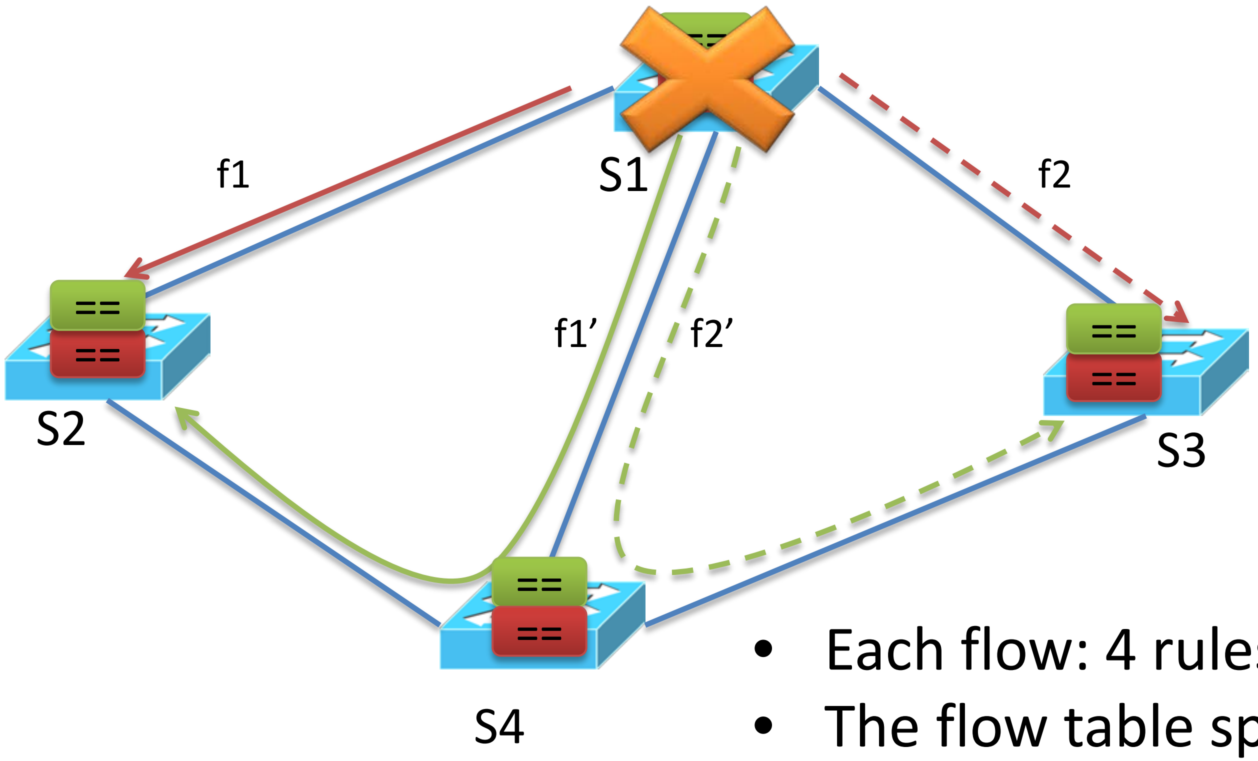
- TCAM is expensive and power hungry. [Hong '13]
  - Today's OpenFlow switch: **1-4K** rules
  - Next generation: **16K** rules
- An example
  - 50 sites
  - 15-shortest path routing
  - **20K** rules are required

One-step update



- Each flow: 4 rules
- The flow table space: 15 rules
- 16 rules are needed at most

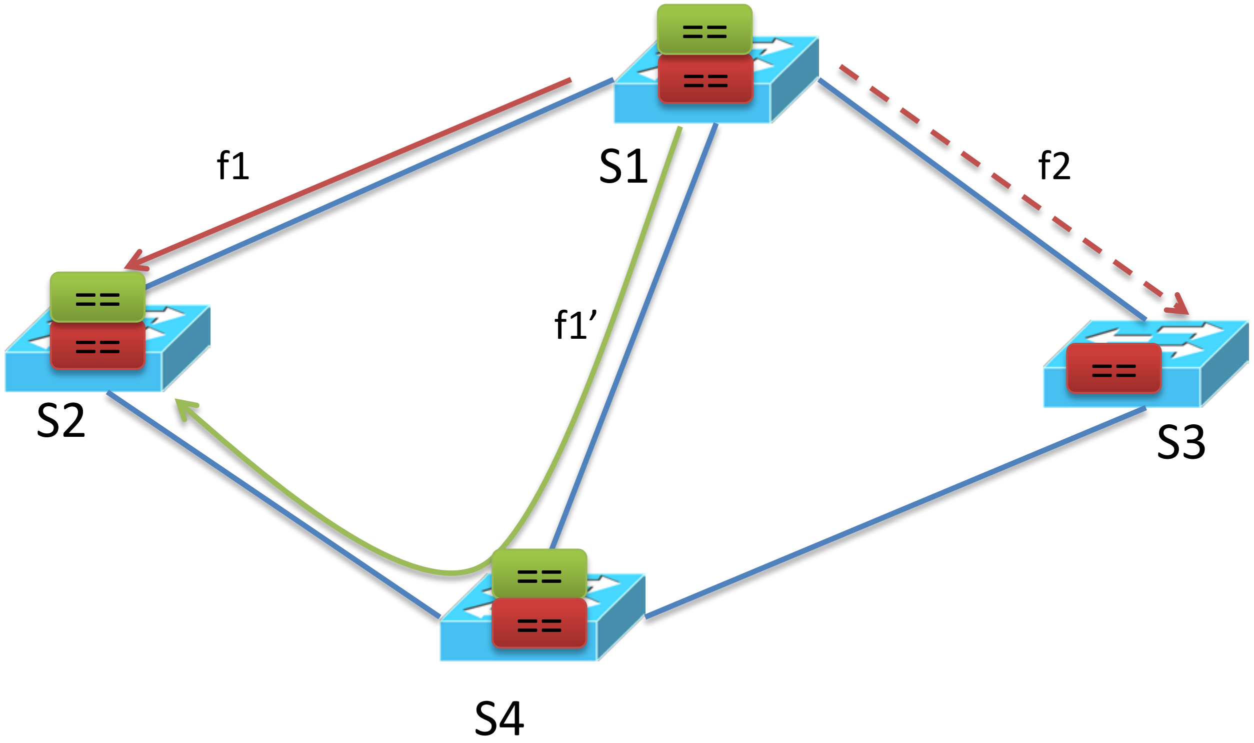
One-step update



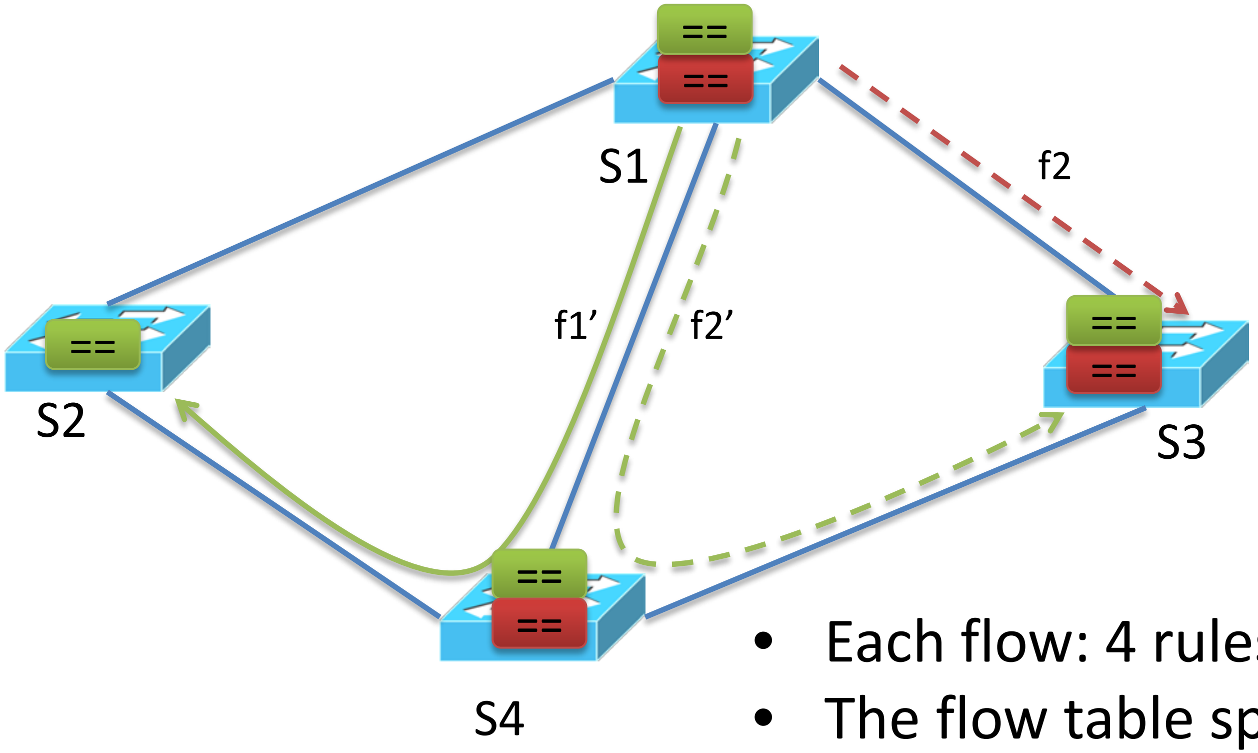
- Each flow: 4 rules
- The flow table space: 15 rules
- 16 rules are needed at most

The method doesn't work!

Separate the update into steps



Separate the update into steps



- Each flow: 4 rules
- The flow table space: 15 rules
- 12 rules are needed at most

The method works!

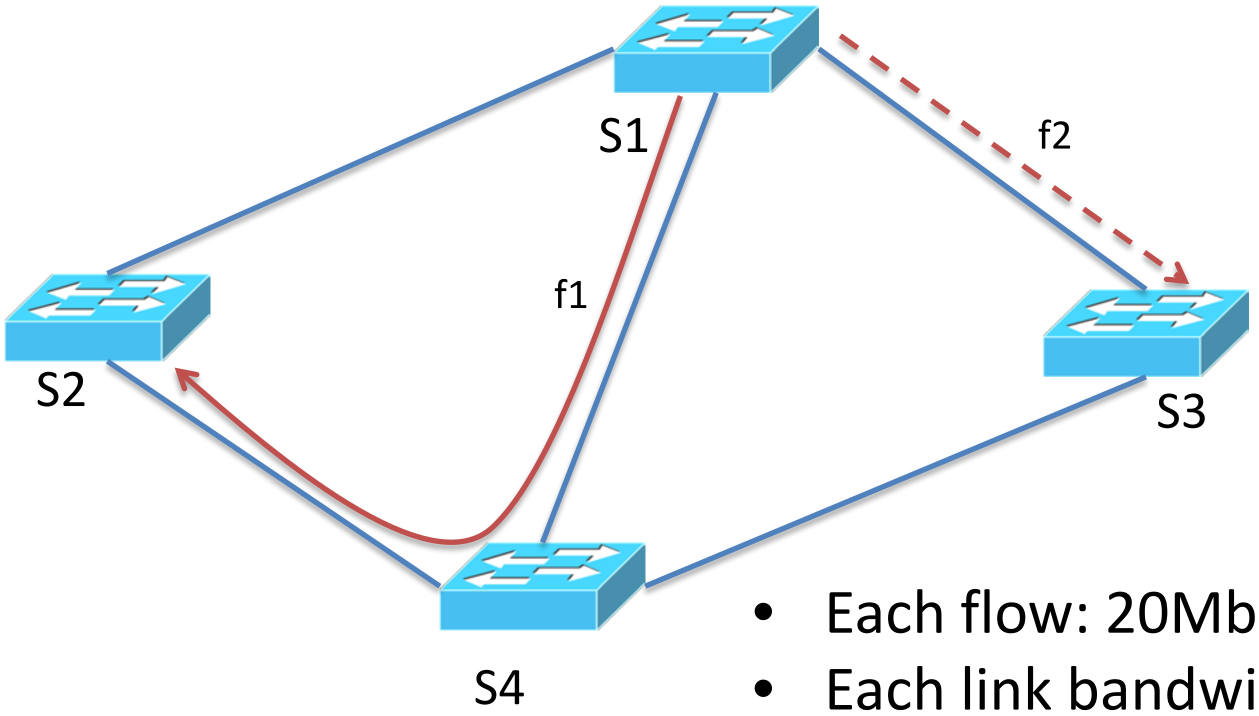
Separate the update into steps

- Tradeoff between flow table space & update time
  - To reduce flow table space overhead, more steps are required.
  - To complete the update in the shortest time, we should update as many flows as possible in one step.

How to complete the update using the least number of steps?

Congestion-free update

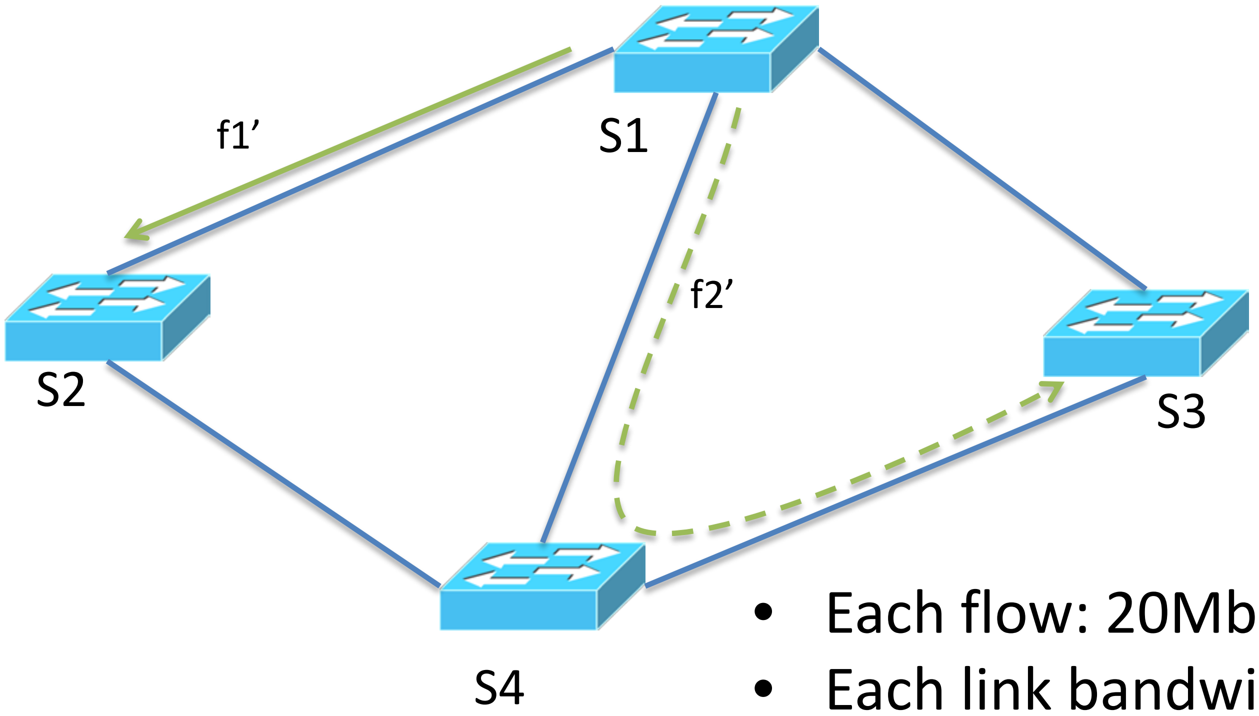
Initial network state



- Each flow: 20Mbps
- Each link bandwidth: 30Mbps

Congestion-free update

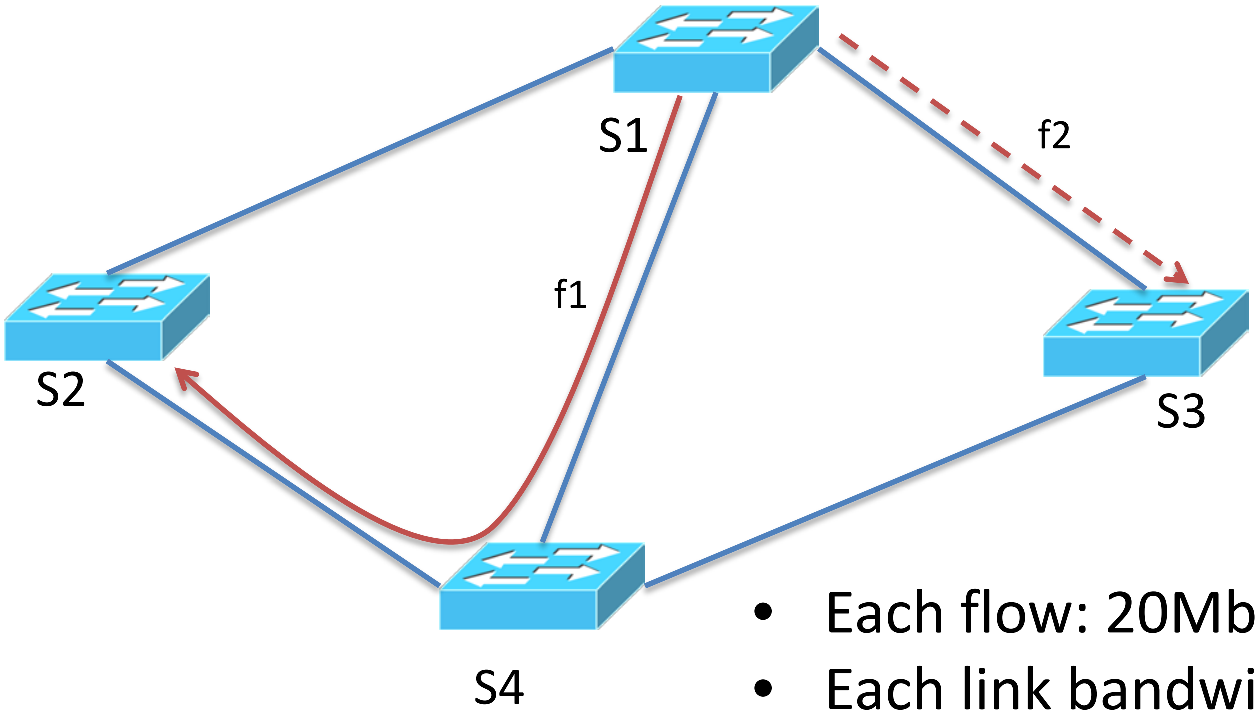
Final network state



- Each flow: 20Mbps
- Each link bandwidth: 30Mbps

Flow update may lead to congestion

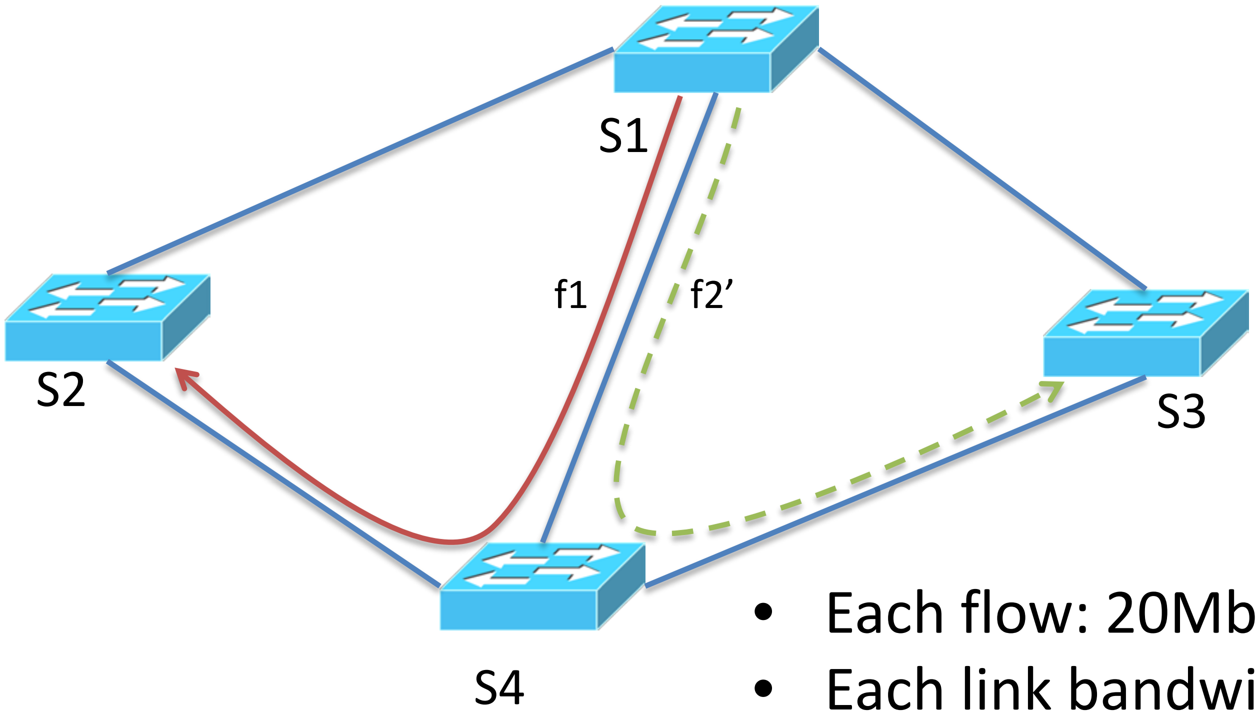
Update sequence: f2->f1



- Each flow: 20Mbps
- Each link bandwidth: 30Mbps

Flow update may lead to congestion

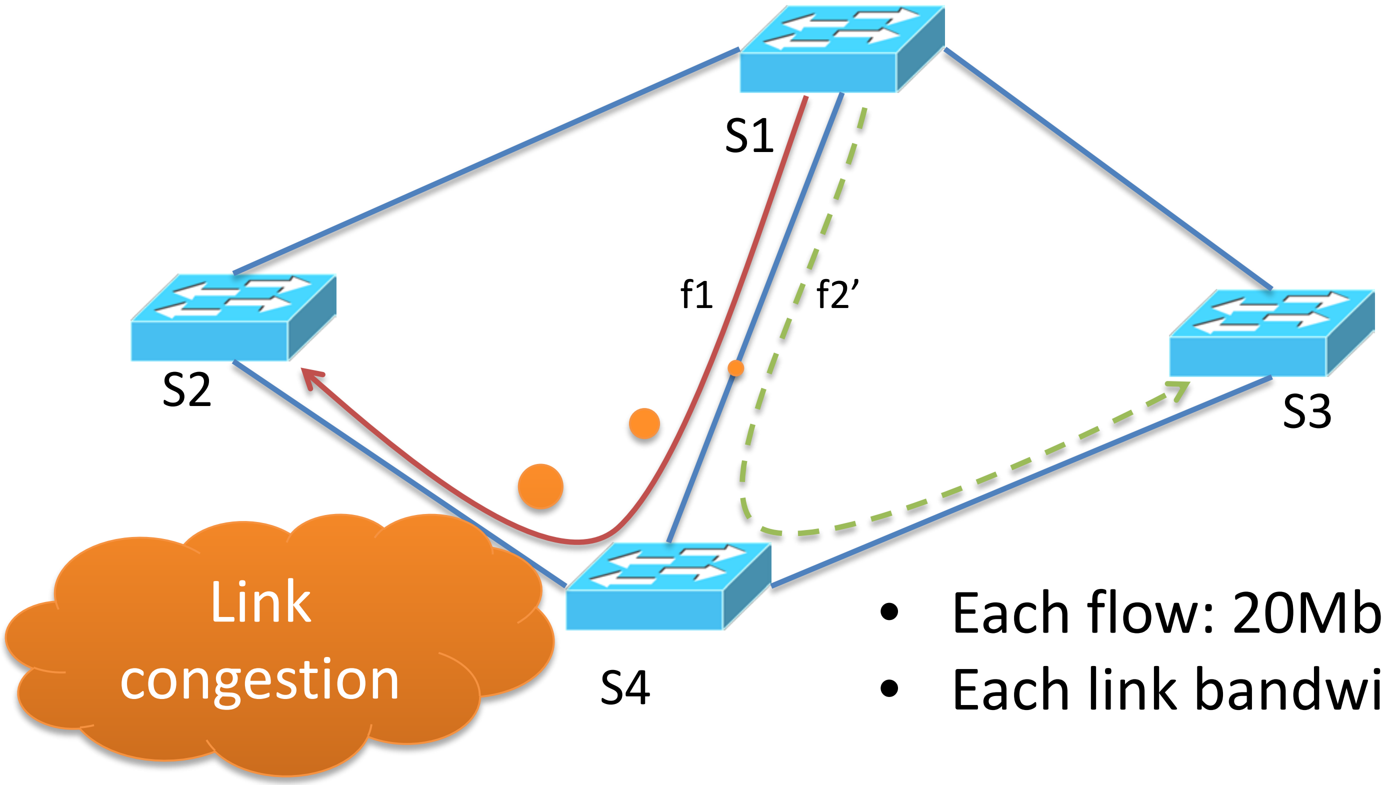
Update sequence: f2->f1



- Each flow: 20Mbps
- Each link bandwidth: 30Mbps

Flow update may lead to congestion

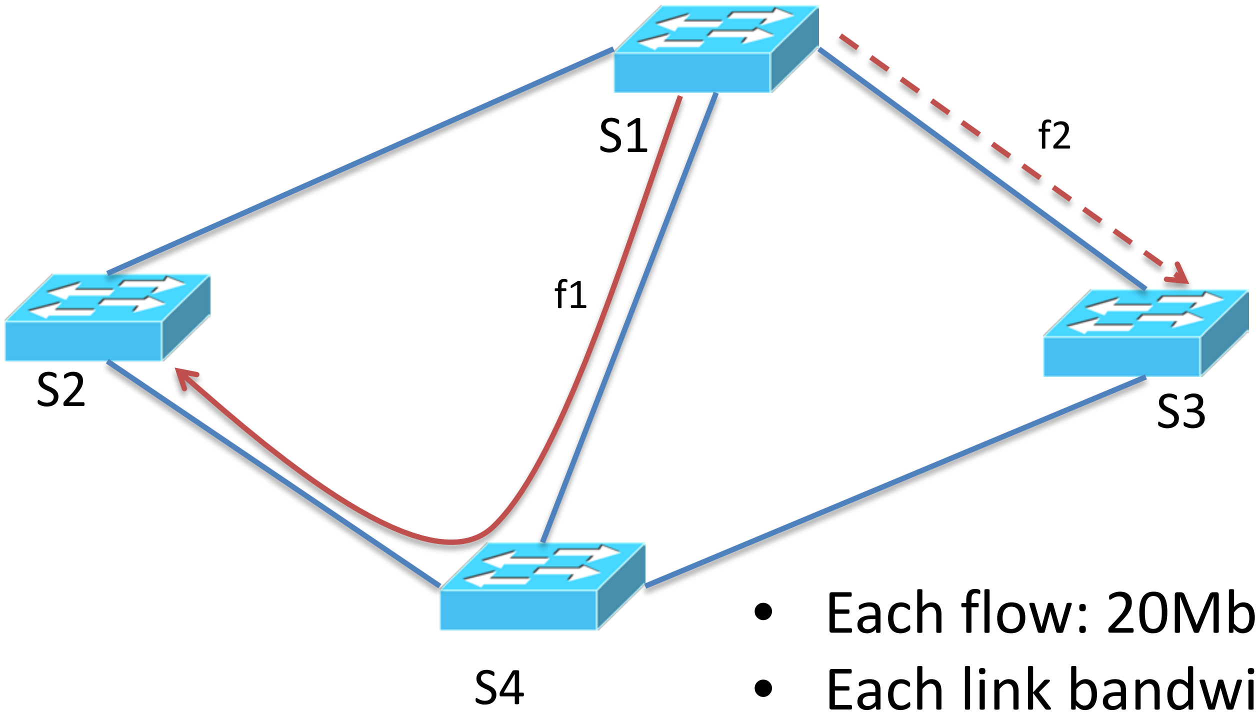
Update sequence: f2->f1



- Each flow: 20Mbps
- Each link bandwidth: 30Mbps

Congestion-free update

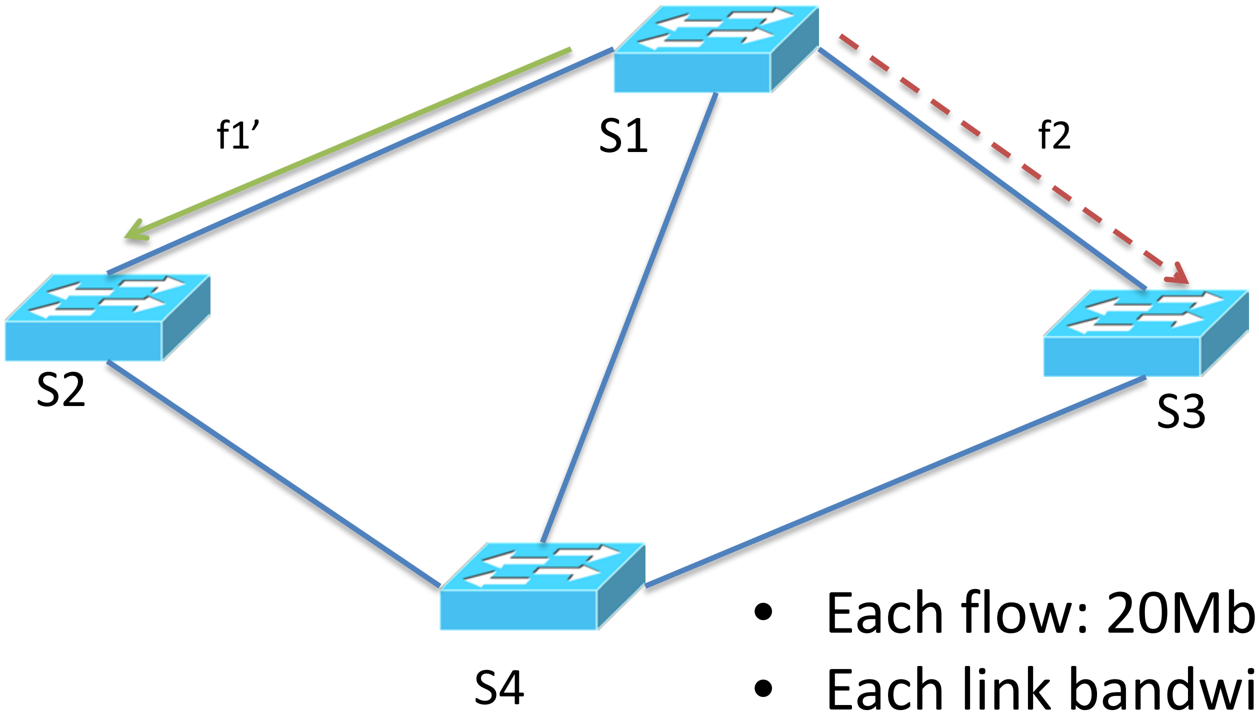
Update sequence: f1->f2



- Each flow: 20Mbps
- Each link bandwidth: 30Mbps

Congestion-free update

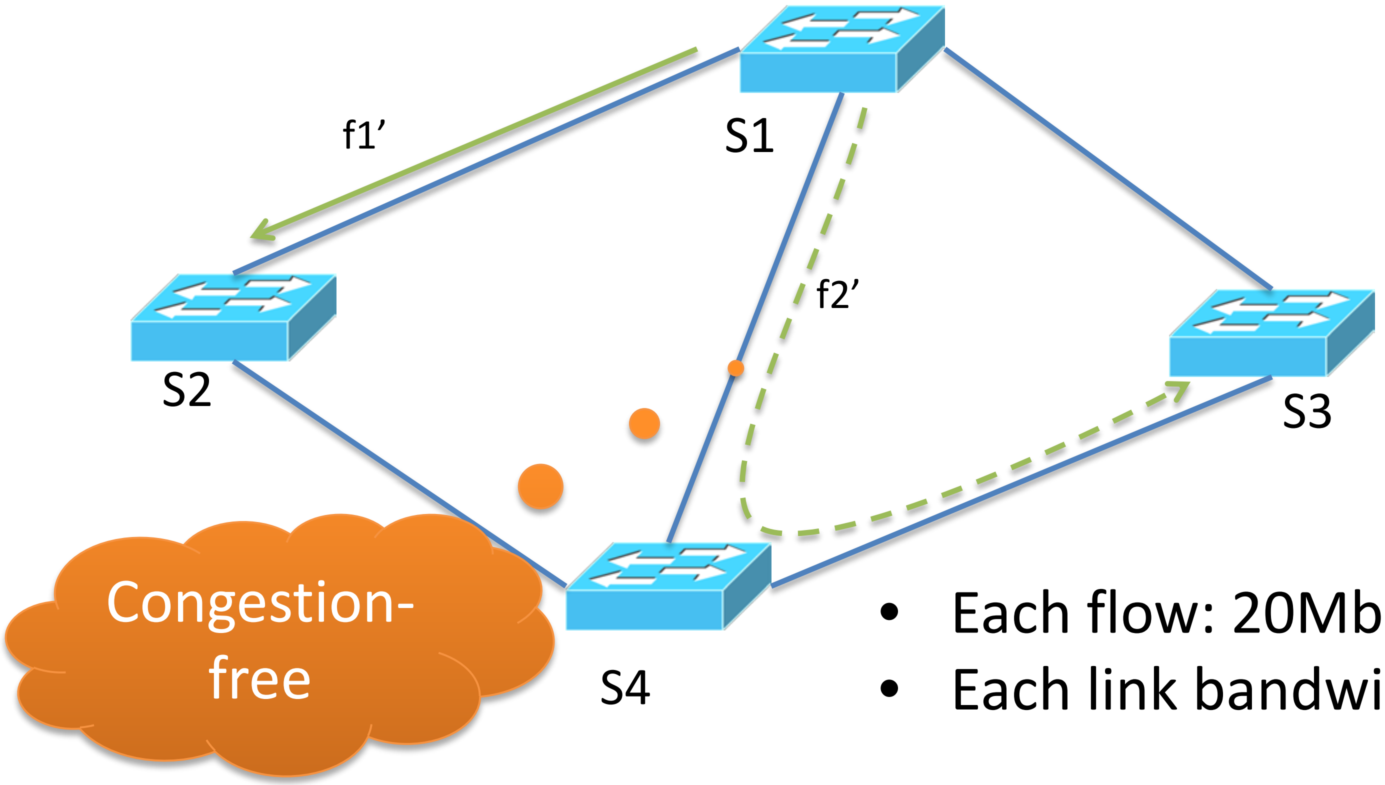
Update sequence: f1->f2



- Each flow: 20Mbps
- Each link bandwidth: 30Mbps

Congestion-free update

Update sequence: f1->f2



- Each flow: 20Mbps
- Each link bandwidth: 30Mbps

Problem statement

How to schedule the update of multiple flows?

- Considering link bandwidth
- Under the constraint of flow table space
- Using the least steps

Scheduling the multi-flow update

- Given the initial and final network states, compute which flows to update in each step.
- Formulate it as a Mixed Integer Problem
  - **Minimize** the number of steps
  - Constraints
    - $link\ utilization \ \le link\ bandwidth,$
      $\forall\ link, \forall\ time$
    - $flow\ table\ usage \ \le flow\ table\ space,$
      $\forall\ switch, \forall\ time$

Problem formulation

$$\min \quad \sum_{k=1}^{K} I_k$$

$$s.t. \begin{cases} f_i(e) \leq \theta c(e), \forall i \in \{1, \cdots, K\}, \forall e \in E; \\ n_i(u) \leq q(u), \forall i \in \{1, \cdots, K\}, \forall u \in U; \\ 0 \leq x_{ij} \leq 1, \forall i, j \in \{1, \cdots, K\}; \\ \sum_{i=1}^{K} x_{ij} = 1, \forall j \in \{1, \cdots, K\}; \\ I_k \geq \sum_{j=1}^{K} \frac{1}{K} x_{kj}, \forall k \in \{1, \cdots, K\}; \\ I_k \geq I_{k+1}, \forall k \in \{1, \cdots, K-1\}. \end{cases}$$

I_k	whether the update is still in progress in step k
x_{ij}	the part of flow j updated in step i
$c(e)$	link capacity
q(u)	flow table space
θ	max allowed link utilization

Our heuristic solution

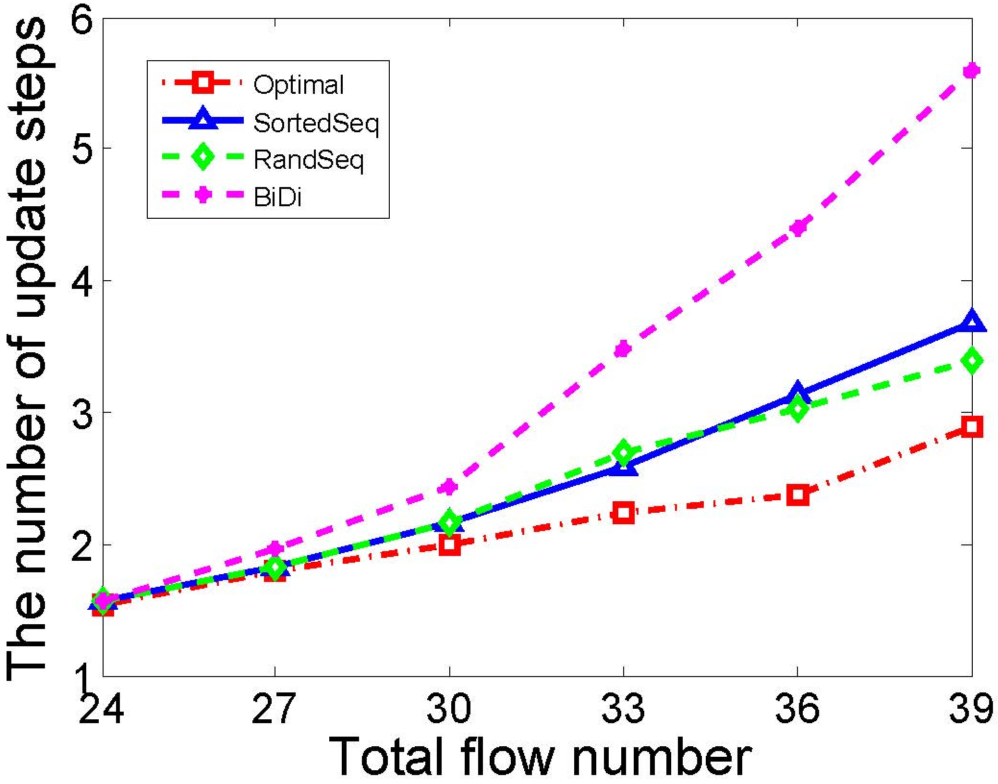
- SortedSeq (Update the flows in a sorted sequence.)
  - Update the **key flows** first
    - The flows that utilize more flow table space
    - Or the available flow table resource is scarce on the new path
  - Update as many flows as possible in each step
    - As long as the link and switch constraints are not violated

Evaluation

- Google’s inter-datacenter WAN
- Routing policy
  - Shortest path
  - The source and destination of each flow are selected randomly.
  - The traffic rate follows uniform distribution.
- Comparison algorithms
  - Optimal algorithm
  - RandSeq, BiDi

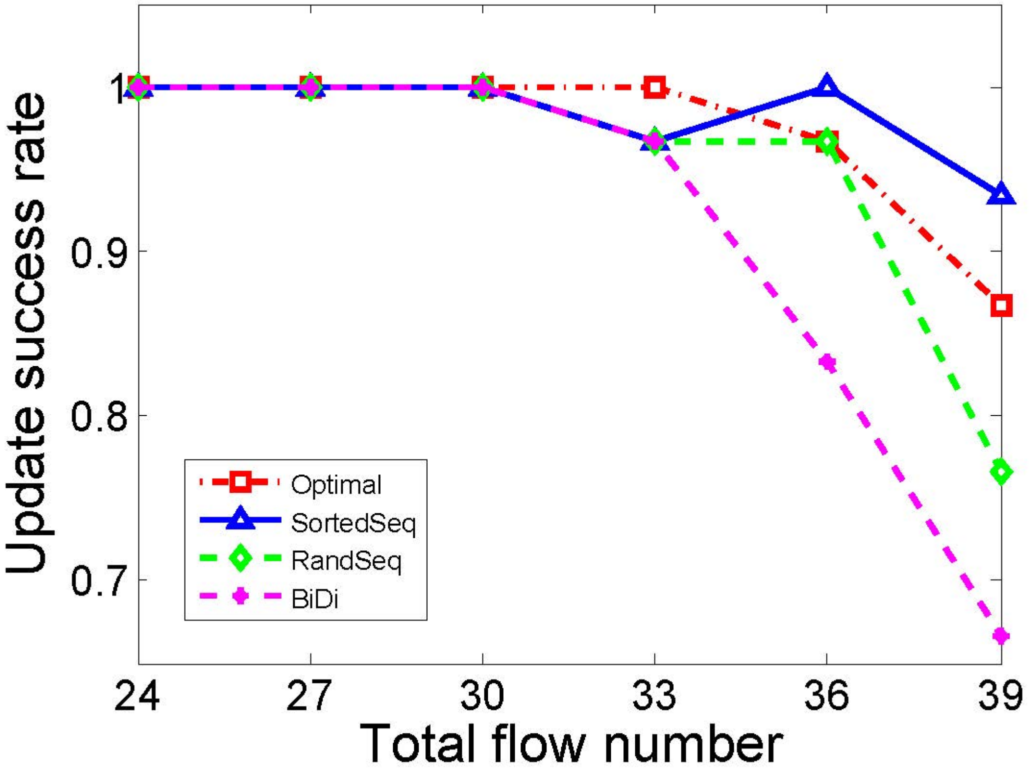
SortedSeq has near-optimal performance

- Impact of flow number



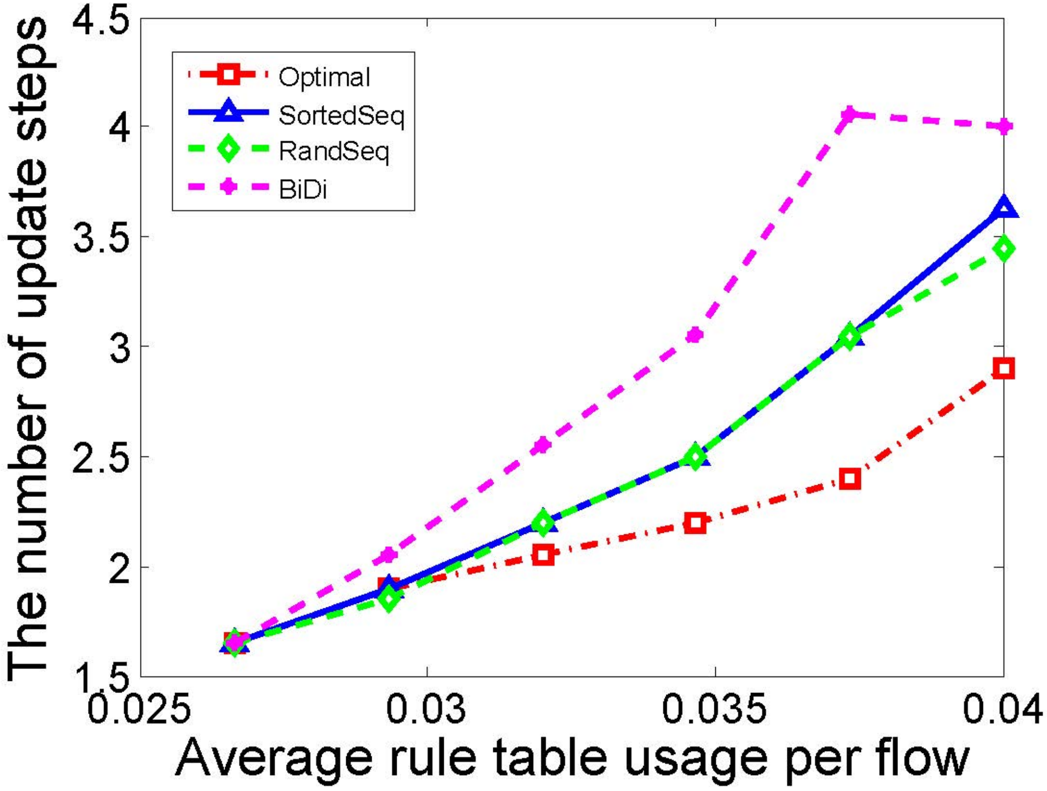
SortedSeq achieves high success rate

- Impact of flow number



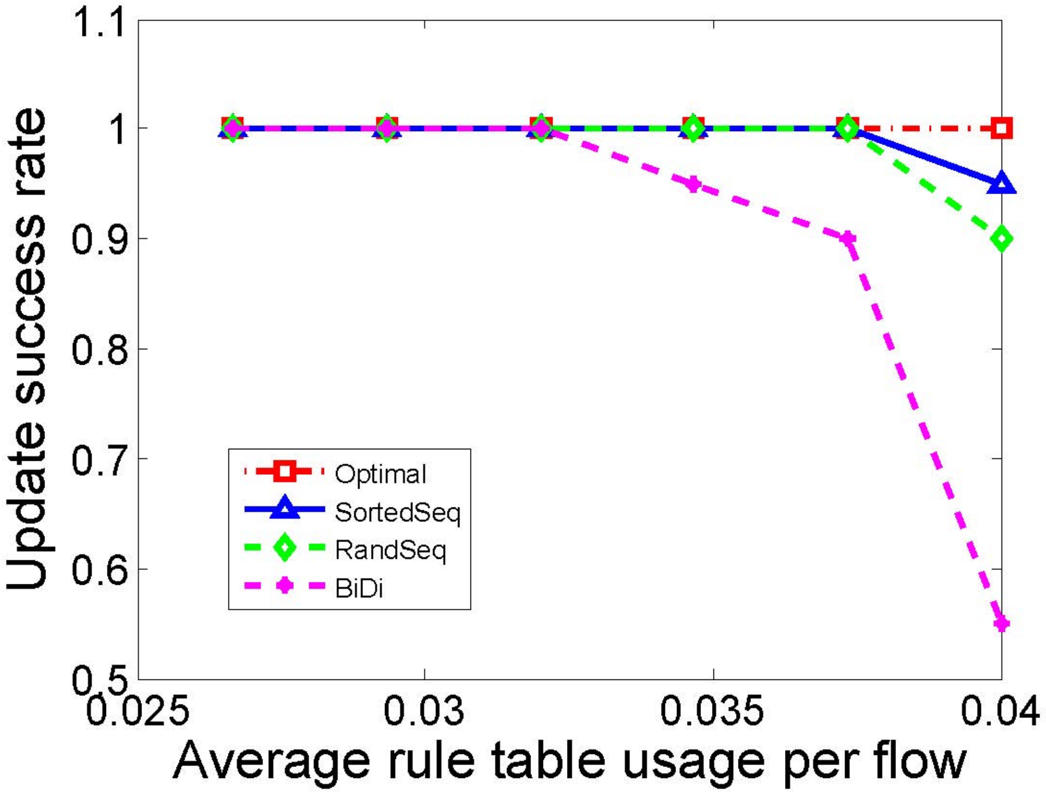
SortedSeq has near-optimal performance

- Impact of flow table space



SortedSeq achieves high success rate

- Impact of flow table space



Summary

- Link bandwidth and flow table space constraints should be considered in multi-flow update.
- Our algorithm
  - Finds the solution with near-optimal steps
  - Completes the update efficiently and successfully

Summary

- Link bandwidth and flow table space constraints should be considered in multi-flow update.
- Our algorithm
  - Finds the solution with near-optimal steps
  - Completes the update efficiently and successfully

Future work

- Carry out experiments on practical platforms.
- Apply it into the multi-path scenario
- Analyze the tradeoff between link utilization and flow table overhead.

Thanks!

Yujie Liu

yj-liu11@mails.tsinghua.edu.cn