Flow Migrations in Software Defined Networks: Consistency, Feasibility, and Optimality

Yang Chen, Huanyang Zheng, and Jie Wu

Center for Networked Computing

Temple University, USA

1. Introduction

Flow migration in SDN

- Upon traffic changes

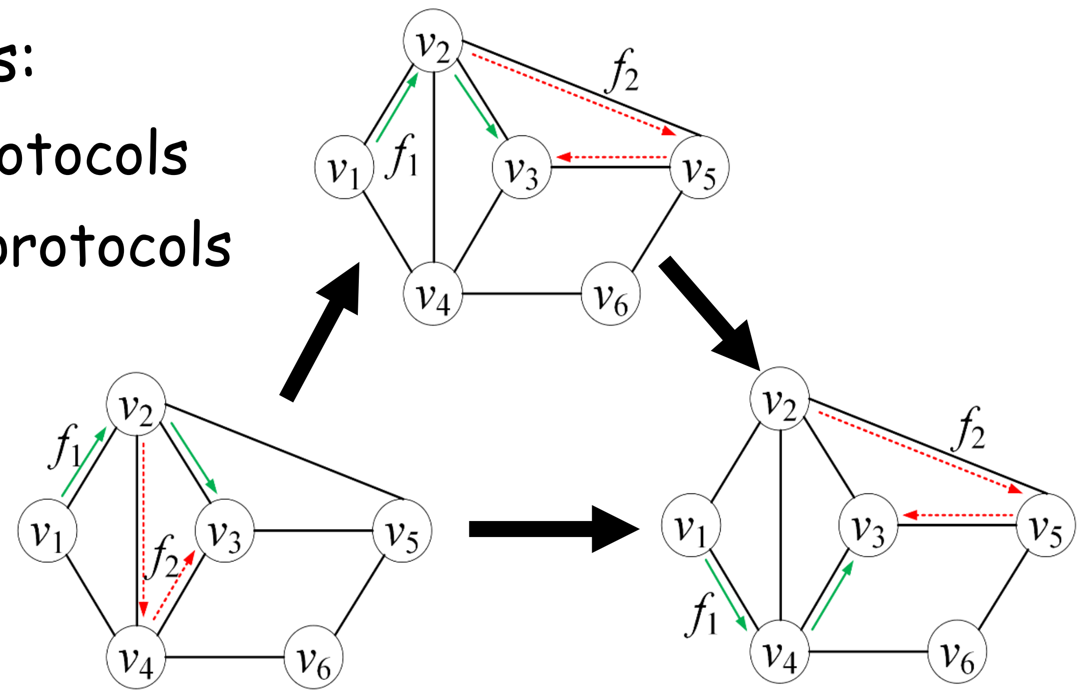
Challenges:

- Asynchronous rule updates -> congestion -> deadlocks

Basic update methods:

- Ordering migrate protocols
- Two-phase migrate protocols

Flow	Tag	Action
f_2	old	forward to v_4
f_2	new	forward to v_5
…	…	…



Initial State

Final State

Proposed Method

- Ordering migrate protocols
  - Pros: no additional overhead
  - Cons: do not always exist
- Two-phase migrate protocols
  - Pros: simple and fast
  - Cons: overhead from packet tagging and extra rules, imperfect synchronization

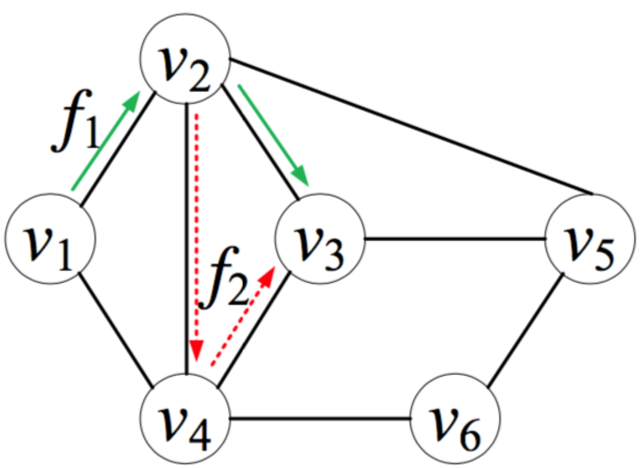
In this paper we use spare paths for flow migration.

A Motivating Example

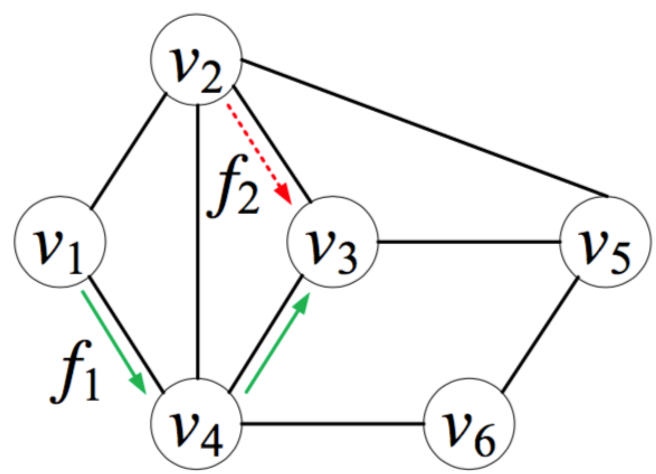
Ordering protocols do not always work

- overlap between flows' initial and final paths

Initial State



Final State

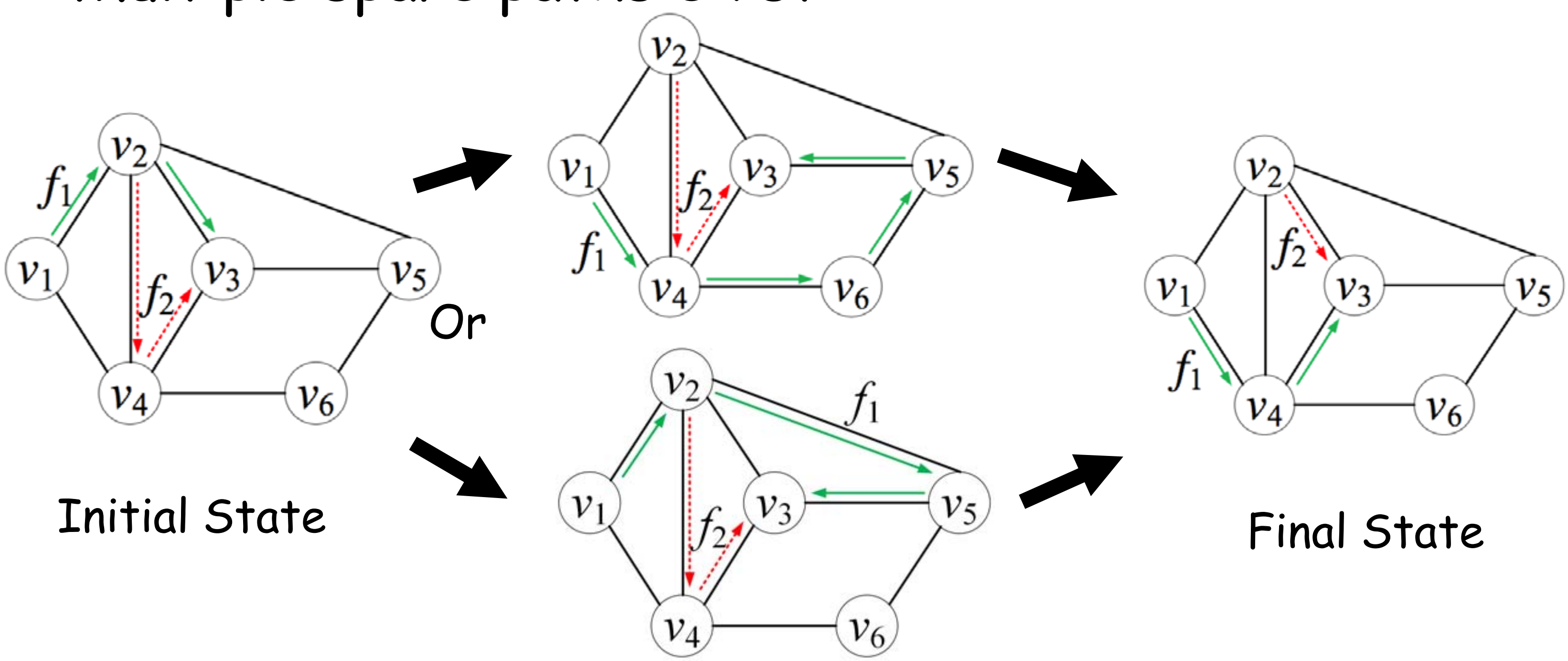


Unit flow capacity/flow demand
Flows are un-splittable

The initial path of f_1 overlaps the final path of f_2, and vice versa

Example (Cont'd)

- Spare path for flow f
  - Enough bandwidth to hold f
  - Same source and destination for f
  - Does not overlap with initial or final paths of f
- Multiple spare paths exist



2. Model

- Model

  A network with capacitated links and a set of flows with demands

- Objective

  Migrate flows from given initial paths to final paths

- Consistency

  Migration constraint: no congestion or packet loss

- Feasibility

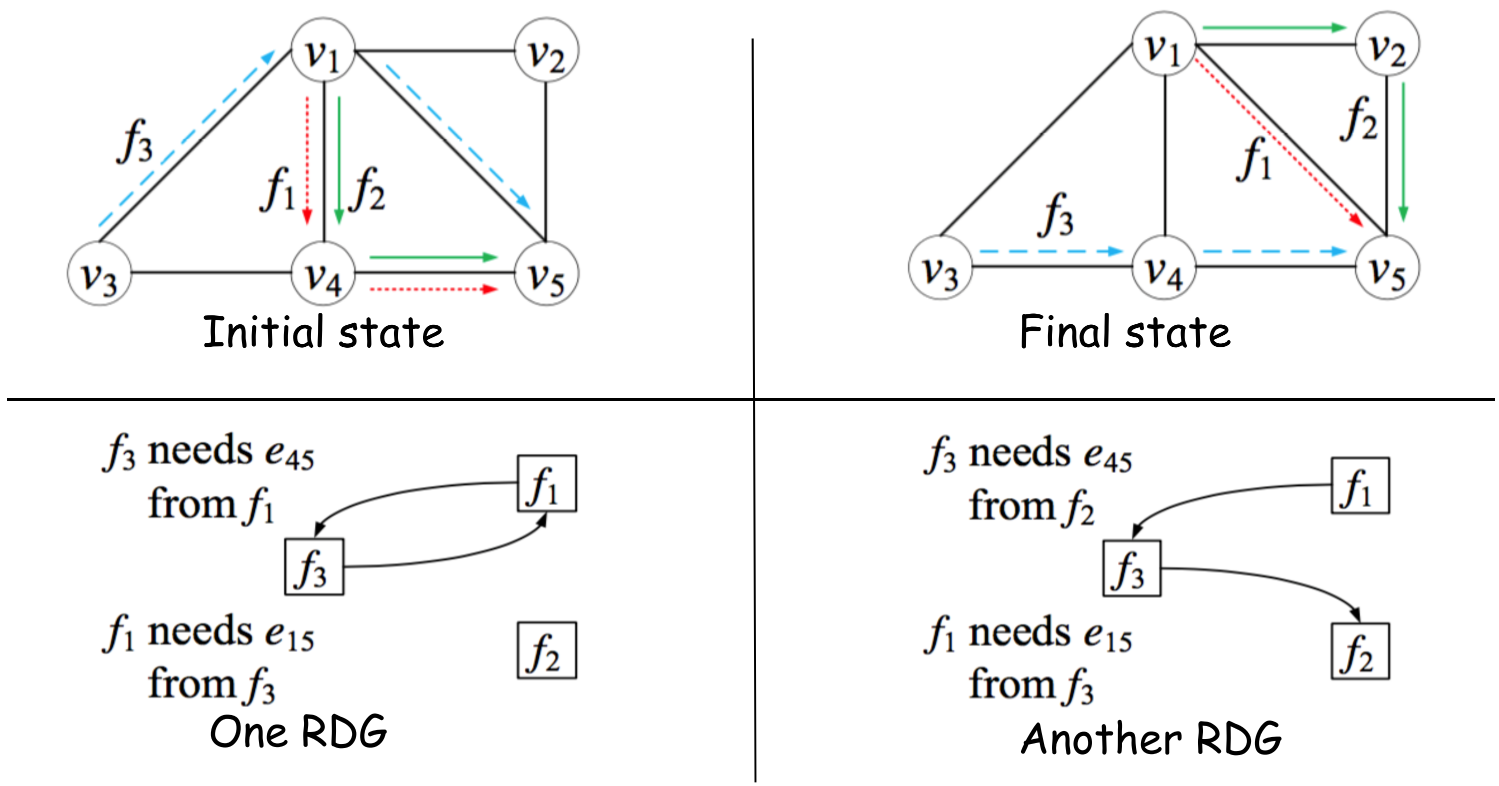
  Existence of consistent flow migration

- Optimality

  Use the fewest spare links

Concepts

- Resource Dependency Graph (RDG): RDGs are not unique
- Deadlock: A cycle exists in all RDGs
- Spare path collection: A set of spare paths resolve a deadlock



Unit flow demand. Link's capacity: e_{14}=e_{45}=2; others= 1

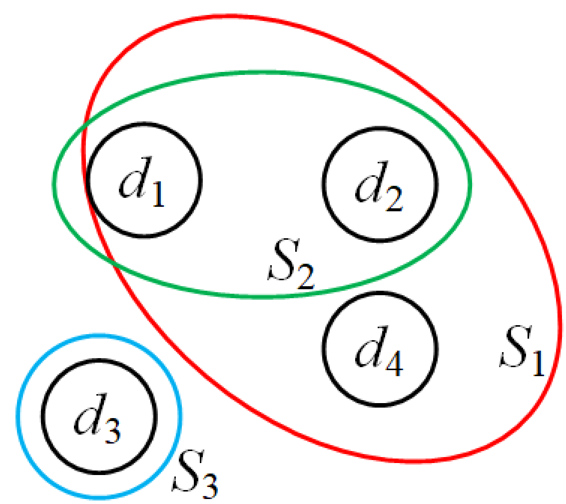
3. Feasibility and Optimality

Feasibility: Each deadlock has a spare path collection

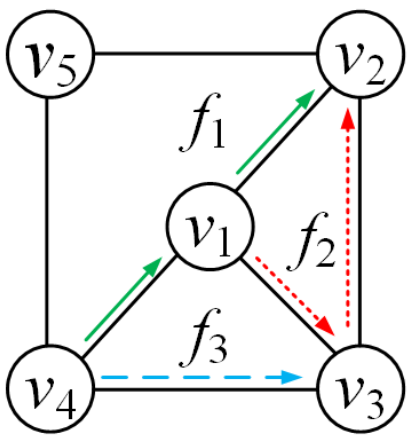
Theorem: If multiple consistent flow migrations exist, it is NP- hard to find the optimal one which occupies the spare links.

Proof ideas:

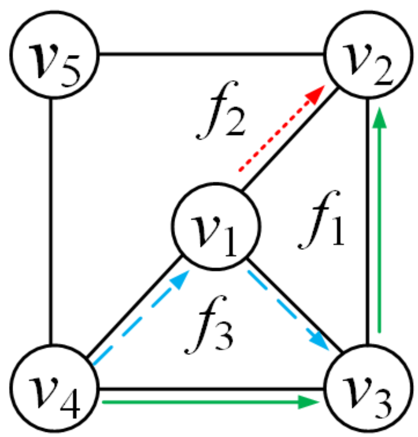
1. Reduction from set cover problem
2. Deadlocks as elements: L= $\{d_1, d_2, \ldots, d_{|L|}\}$
3. Spare path collections as sets: $\{S_1, S_2, \ldots\}$;
   $S_1=\{d_1, d_2, d_4\}$, $S_2=\{d_1, d_2\}$, $S_3=\{d_3\}$



Intertwined Deadlocks: d_1 and d_2



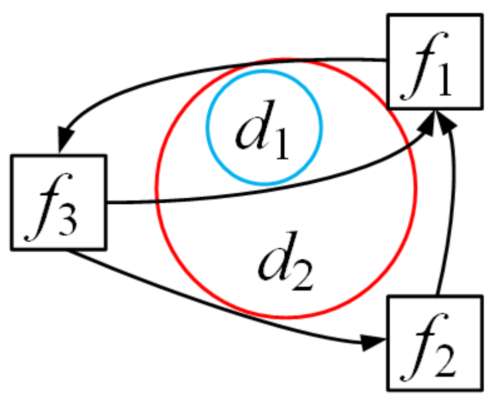
Initial State



Final State

- A spare path collection resolves two deadlocks
  - Move f_1 to a spare path $\{e_{45}, e_{52}\}$
  - Move f_3 and then f_2
  - Move f_1 to its final state

Link's capacity: 1
Flow demand: 1



RDG with two deadlocks

Network Update through Spare Links (NUSL)

Iteratively choose the spare path collection with the max marginal benefit-to-cost ratio

1. Benefit: the number of broken deadlocks
2. Cost: the marginal gain of spare resources

until all deadlocks are resolved

Managing the complexity

- H: the maximum number of hops in a spare path
- C: the maximum number of flow replacements (by its spare path) in resolving a deadlock

  (i.e., cardinality of a spare path collection)

Complexity Analysis

- The algorithm achieves an approximation ratio of $O(H \cdot C \cdot \ln |L|)$

Proof Ideas:

- Use the classic set cover approximation algorithm for reference

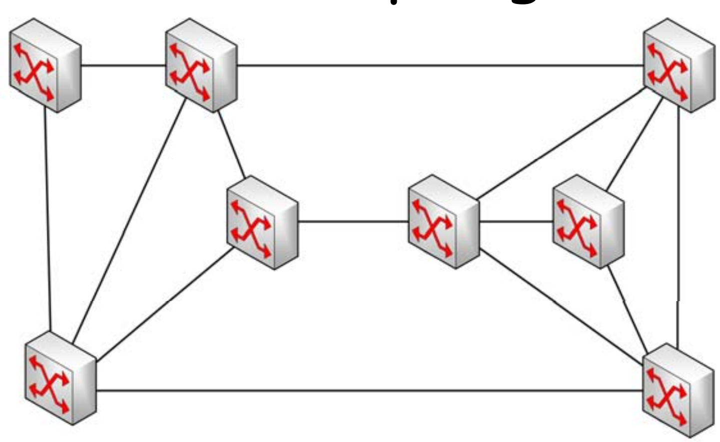
- Worst case time complexity: $O(|L| \cdot \sum_{i=1}^{C} \binom{|F|}{i} \cdot (|F| \cdot |E|^{H})^{i})$

Proof Ideas

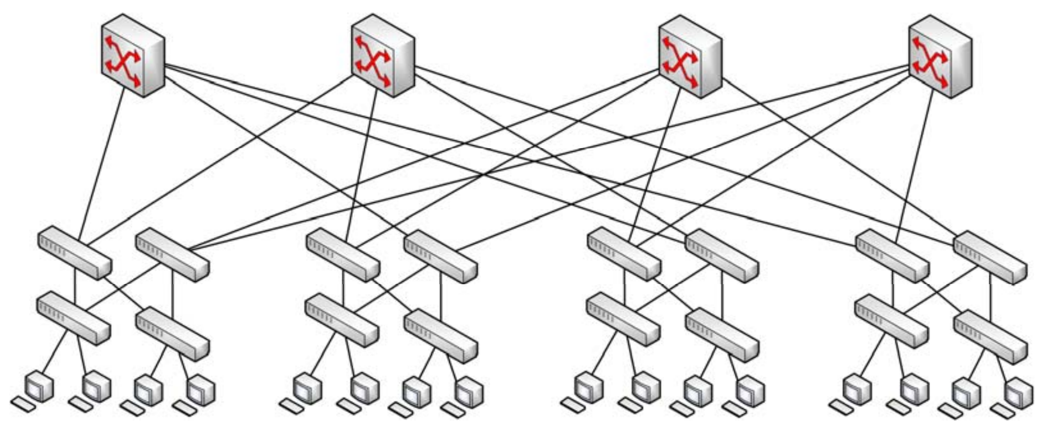
- Use a spare path collection of i paths: $\binom{|F|}{i} \cdot (|F| \cdot |E|^{H})^{i}$

4. Simulation

- Two comparison algorithms:
  1. One-shot: cuts off all the current flows and allows new ones in after the network is vacant (Baseline)
  2. Dionysus: migrates flows in a topological order and opportunistically rate limits flows as zero to resolve deadlocks (SIGCOMM 14)
- Network topologies



WAN network



Fat-tree network

Settings and Measurements

- Settings
  1. WAN topology (link capacity: 1 Gbps)

Traffic load	0.2	0.4	0.6	0.8
Flow number	729	1538	2387	3120

  2. Fat-tree topology (link capacity: 1 Gbps)

Traffic load	0.2	0.4	0.6	0.8
Flow number	2168	4532	6352	8423

- Measurements
  1. The number of rate-limiting flows

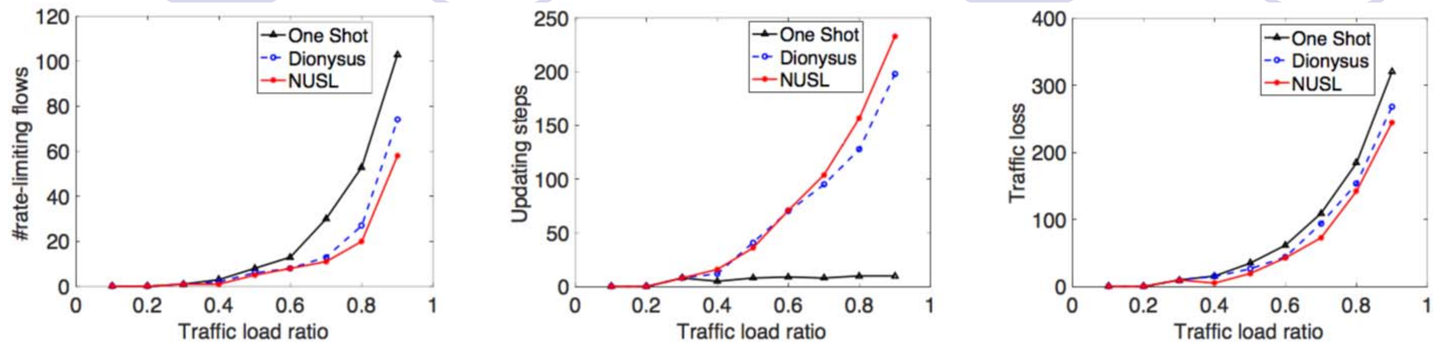
     when a consistent migration plan does not exist
  2. Update steps

     time from the first migration until all flows are migrated
  3. Traffic loss

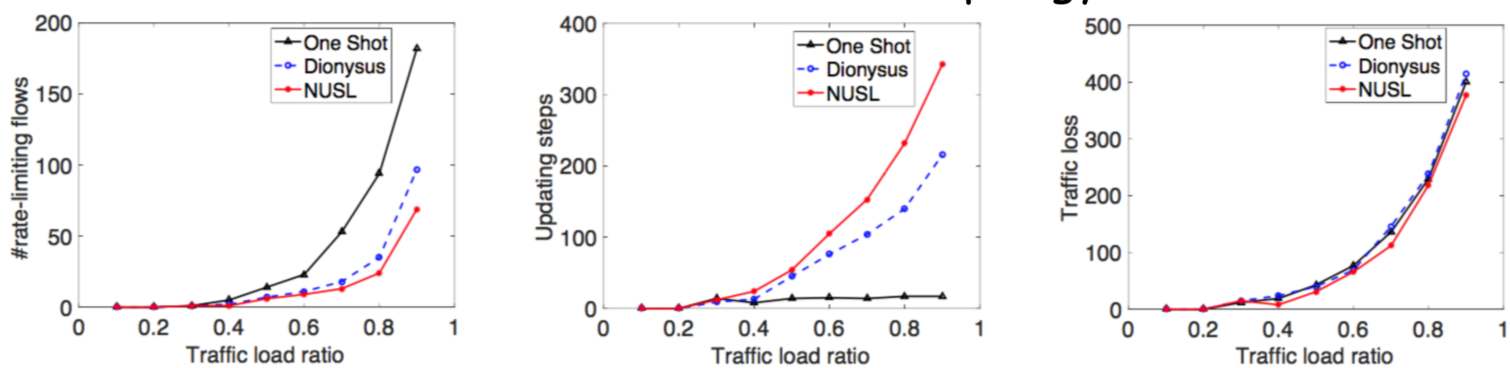
     the total number of lost packets

Simulation Results

Performance in the WAN topology



Performance in the fat-tree topology

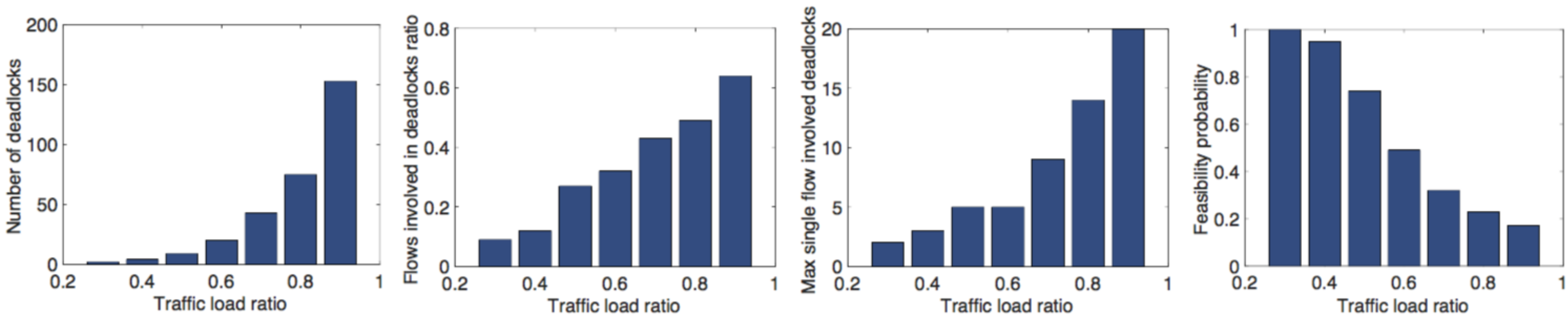


(a) Rate limiting flows. (b) Updating time. (c) Traffic loss.

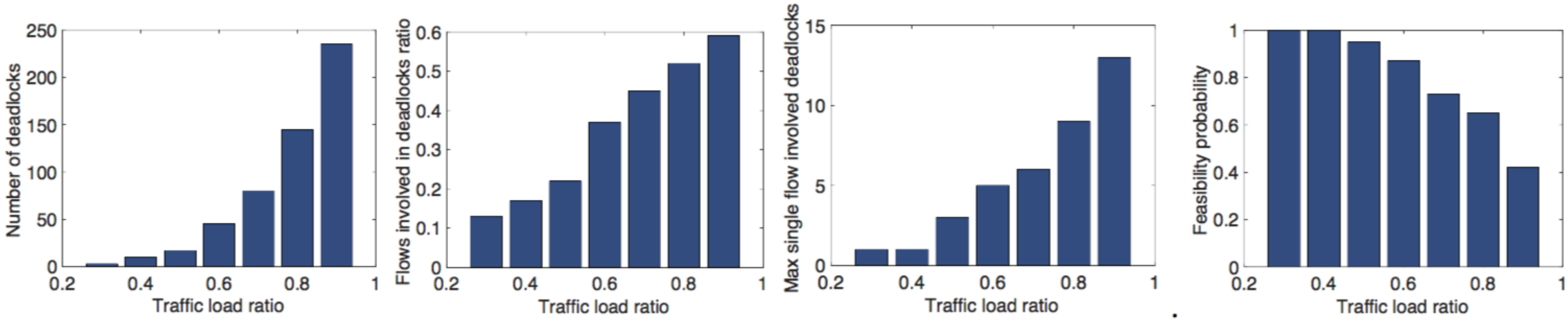
- NUSL rate-limiting: 51% of One Shot, 78% of Dionysus on average (80% ratio)
- NUSLtakes about 19% (WAN) and 33% (fat-tree) more steps than Dionysus
  NUSL always has the least traffic loss

Simulation Results (cont'd)

Performance in the WAN topology



Performance in the fat-tree topology



- Heavier traffic load causes more deadlock
- Fat-tree topology is more likely to find a feasible solution

5. Conclusion

- Migrate flows using spare paths
  - Deadlock resolution
  - Spare path feasibility determination
- NP-hardness
  - Deadlock resolution: using the fewest spare links
- Approximation
  - Set cover: deadlocks are covered by spare path collections
- Future works
  1. Finer granularity: link-based migration solutions