Dynamic Scheduling of Network Updates

Xin Jin

Hongqiang Harry Liu, Rohan Gandhi, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Jennifer Rexford, Roger Wattenhofer

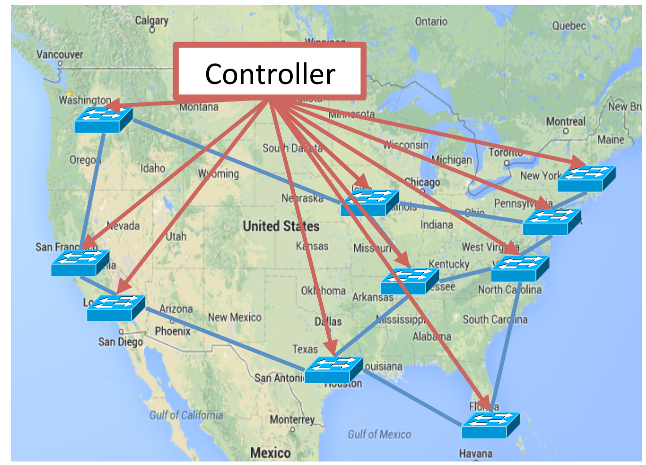
Microsoft Research

PRINCETON UNIVERSITY

ETH Zürich

SDN: Paradigm Shift in Networking

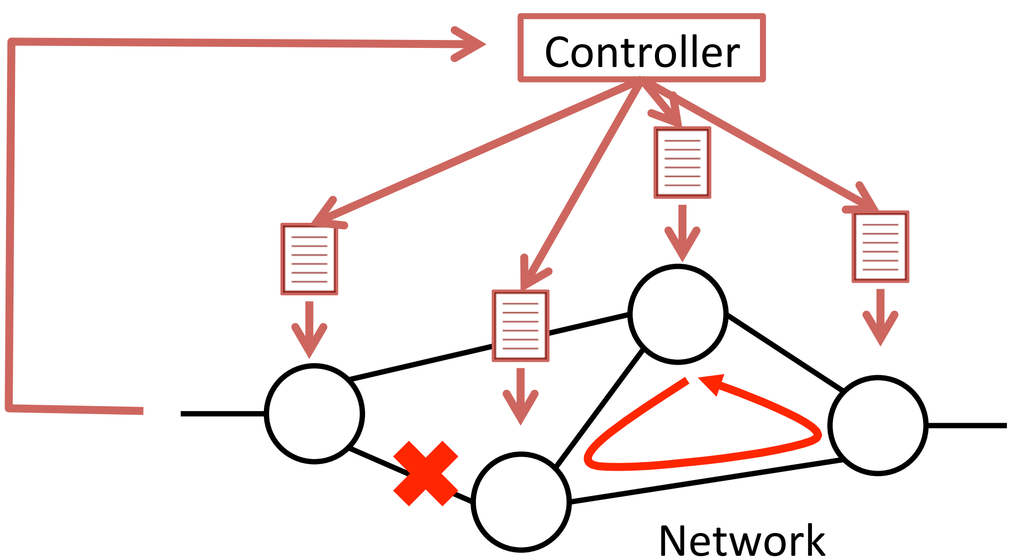
- Direct, centralized updates of forwarding rules in switches



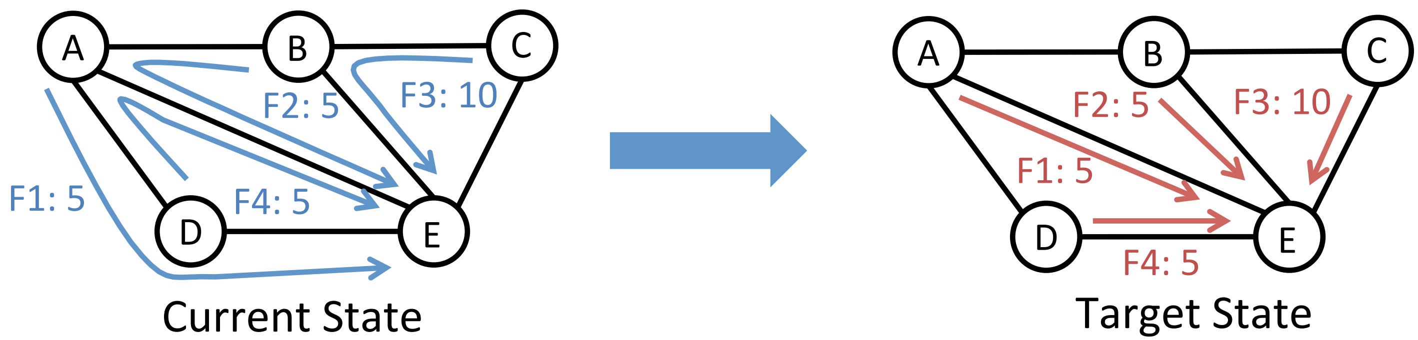
- Many benefits
  – Traffic engineering [B4, SWAN]
  – Flow scheduling [Hedera, DevoFlow]
  – Access control [Ethane, vCRIB]
  – Device power management [ElasticTree]

Network Update is Challenging

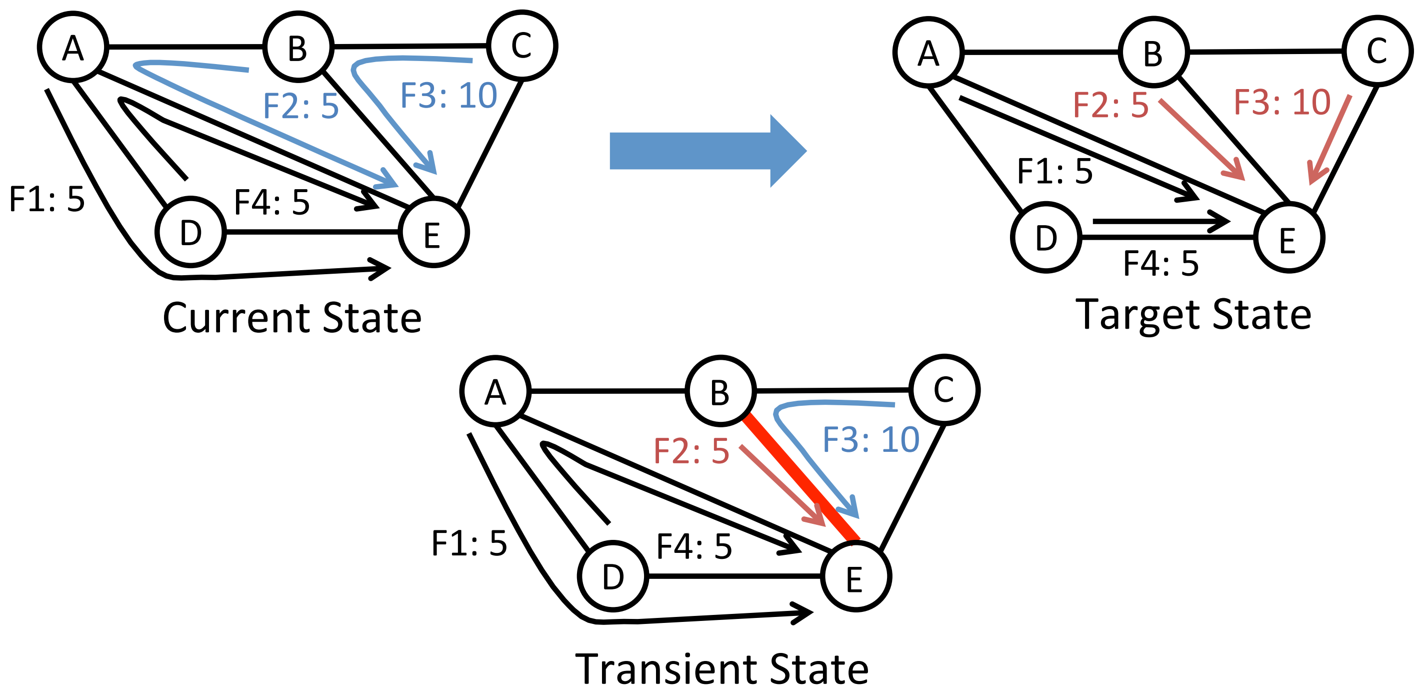
- Requirement 1: fast
  - The agility of control loop
- Requirement 2: consistent
  - No congestion, no blackhole, no loop, etc.



What is Consistent Network Update



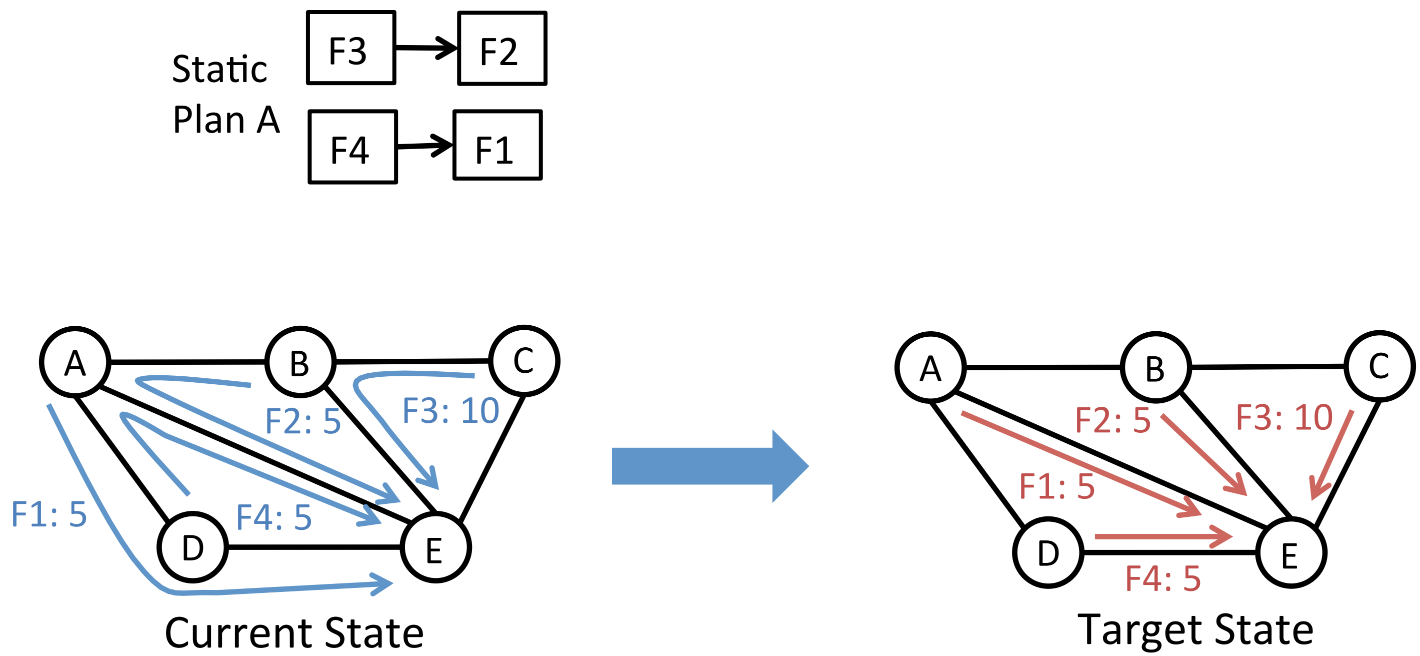
What is Consistent Network Update



- Asynchronous updates can cause congestion
- Need to carefully order update operations

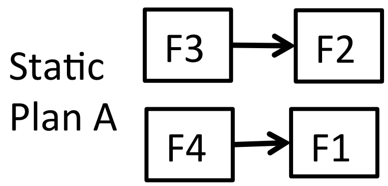
Existing Solutions are Slow

- Existing solutions are static [ConsistentUpdate'12, SWAN'13, zUpdate'13]
  – Pre-compute an order for update operations



Existing Solutions are Slow

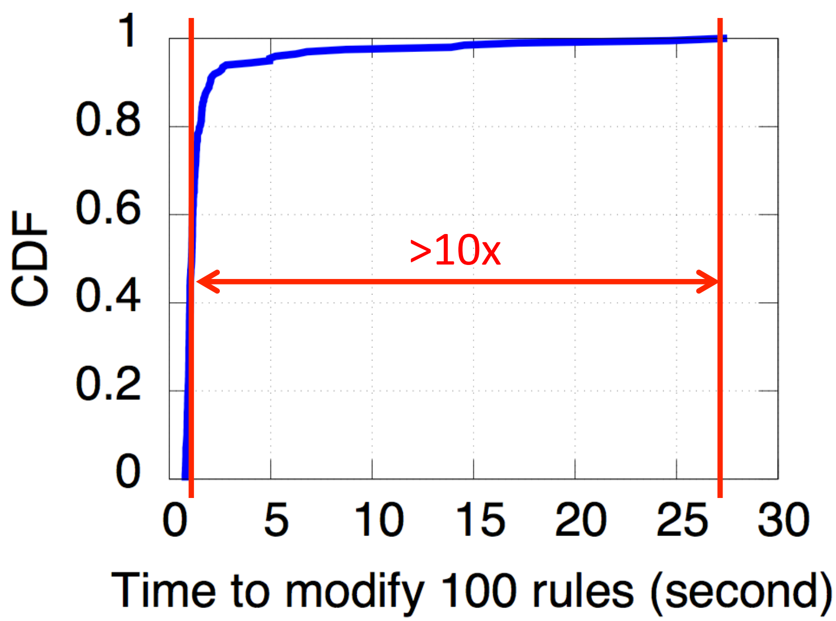
- Existing solutions are static [ConsistentUpdate'12, SWAN'13, zUpdate'13]
  - Pre-compute an order for update operations



- Downside: Do not adapt to runtime conditions
  - Slow in face of highly variable operation completion time

Operation Completion Times are Highly Variable

- Measurement on commodity switches

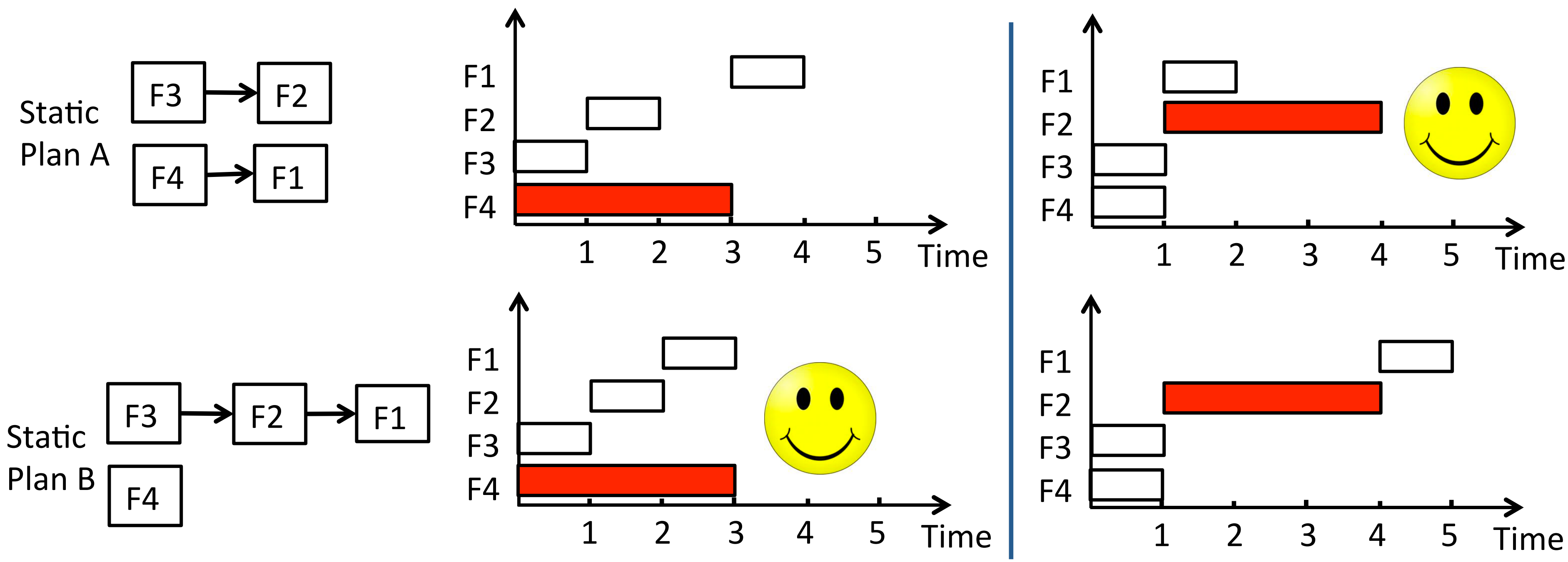


Operation Completion Times are Highly Variable

- Measurement on commodity switches

- Contributing factors
  - Control-plane load
  - Number of rules
  - Priority of rules
  - Type of operations (insert vs. modify)

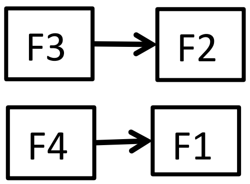
Static Schedules can be Slow



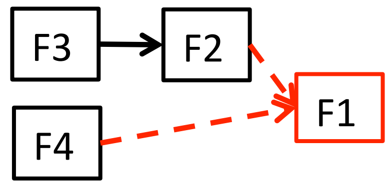
No static schedule is a clear winner under all conditions!

Dynamic Schedules are Adaptive and Fast

Static
Plan A

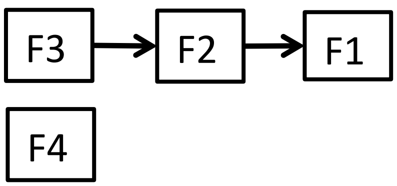


Dynamic
Plan



Adapts to actual conditions!

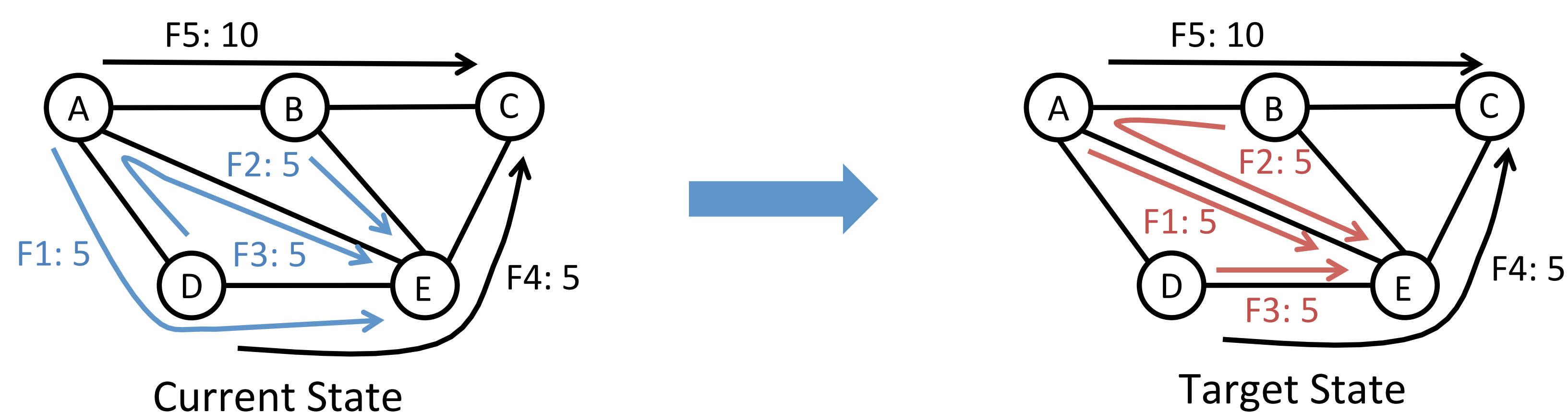
Static
Plan B



No static schedule is a clear winner under all conditions!

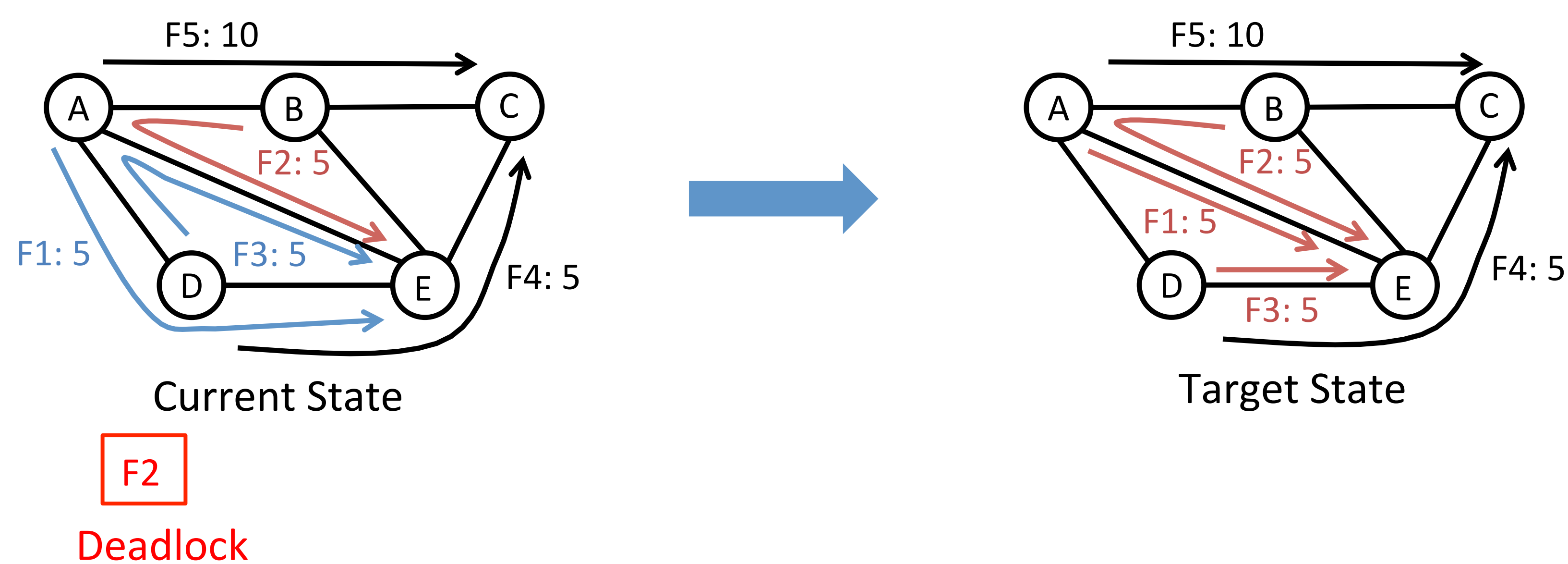
Challenges of Dynamic Update Scheduling

- Exponential number of orderings
- Cannot completely avoid planning



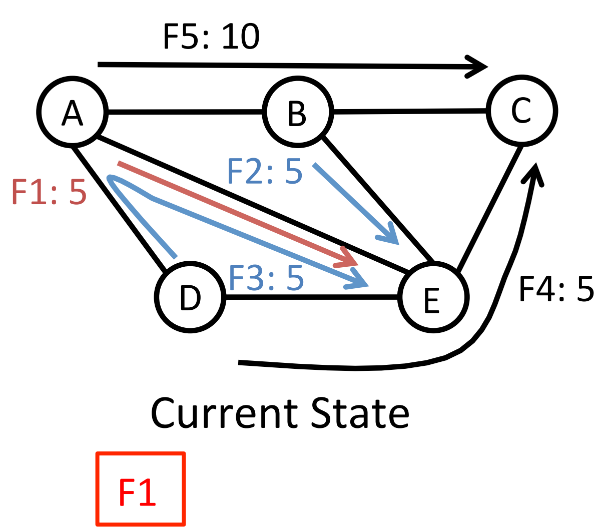
Challenges of Dynamic Update Scheduling

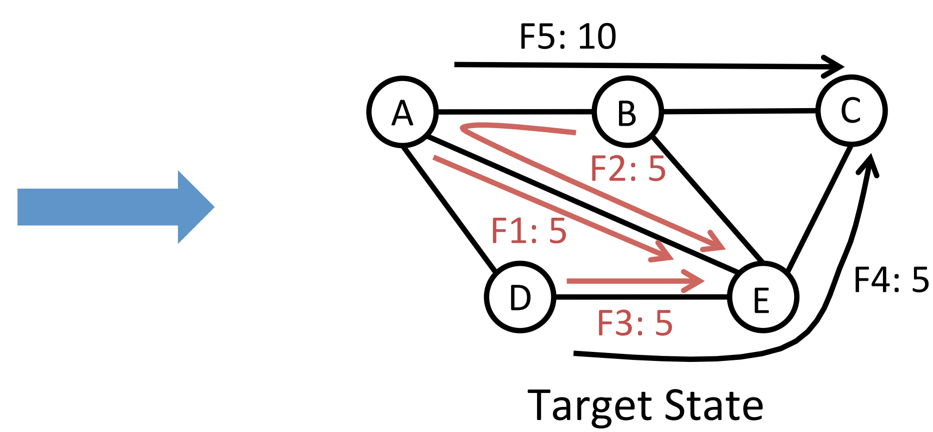
- Exponential number of orderings
- Cannot completely avoid planning



Challenges of Dynamic Update Scheduling

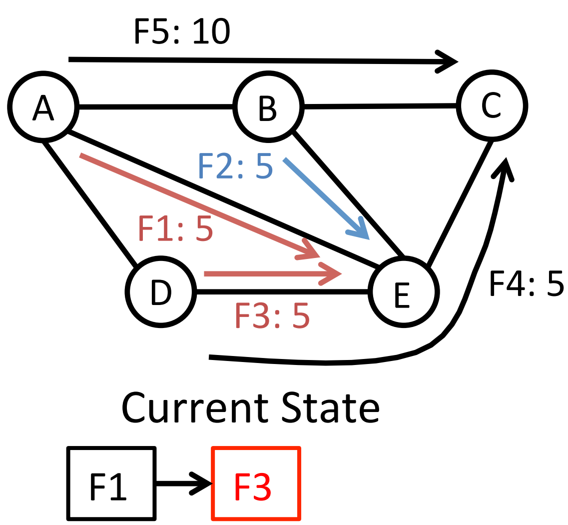
- Exponential number of orderings
- Cannot completely avoid planning

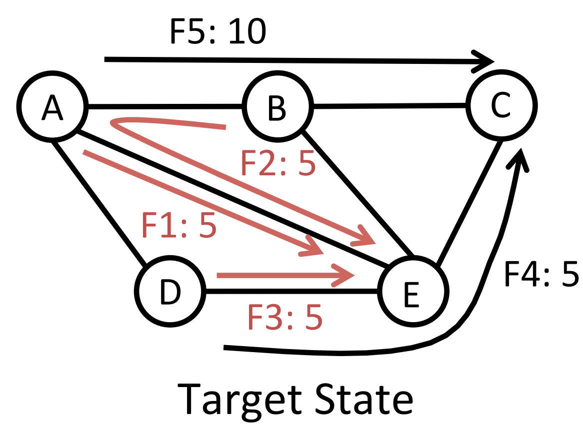




Challenges of Dynamic Update Scheduling

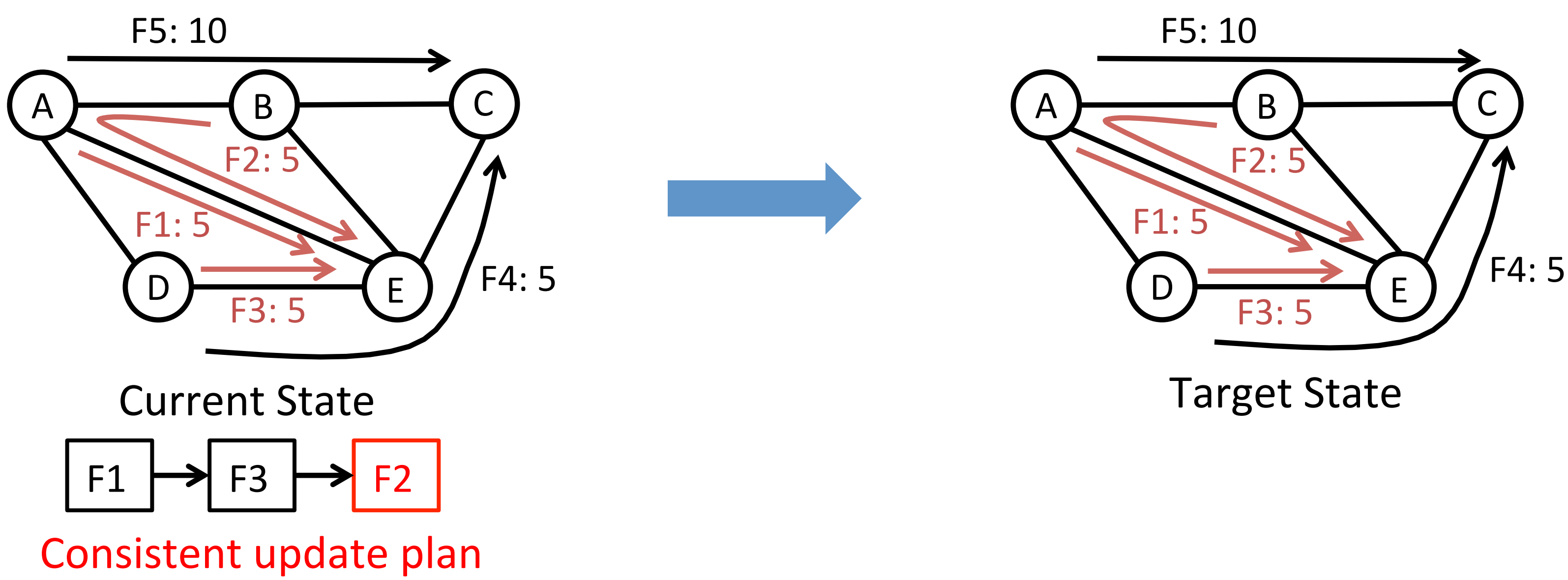
- Exponential number of orderings
- Cannot completely avoid planning



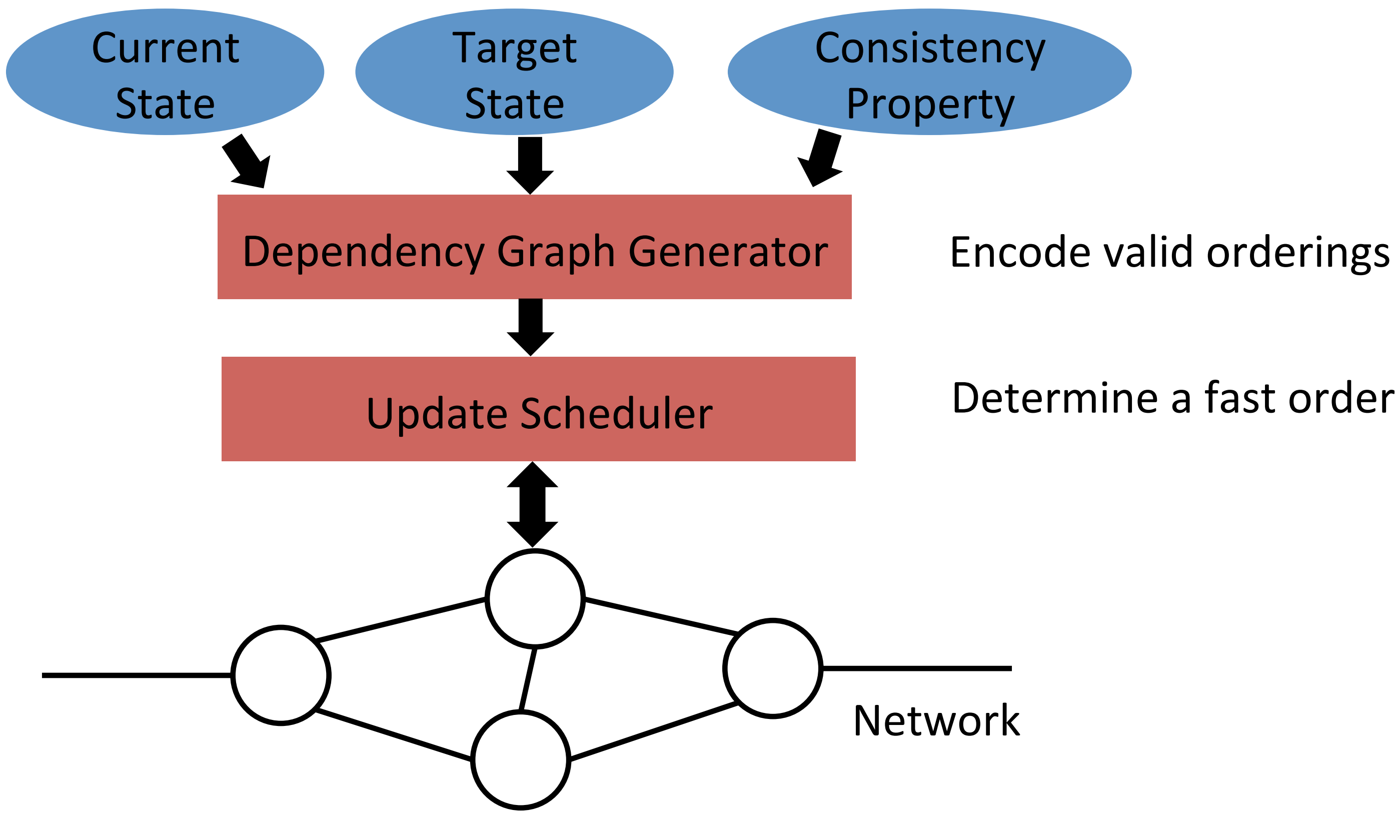


Challenges of Dynamic Update Scheduling

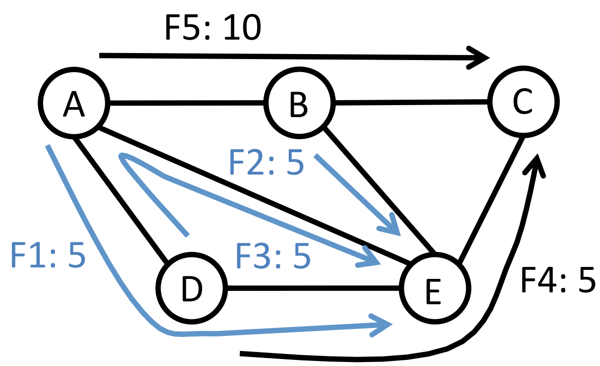
- Exponential number of orderings
- Cannot completely avoid planning



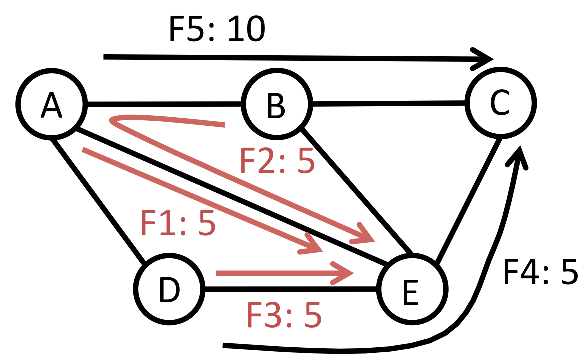
Dionysus Pipeline



Dependency Graph Generation



Current State

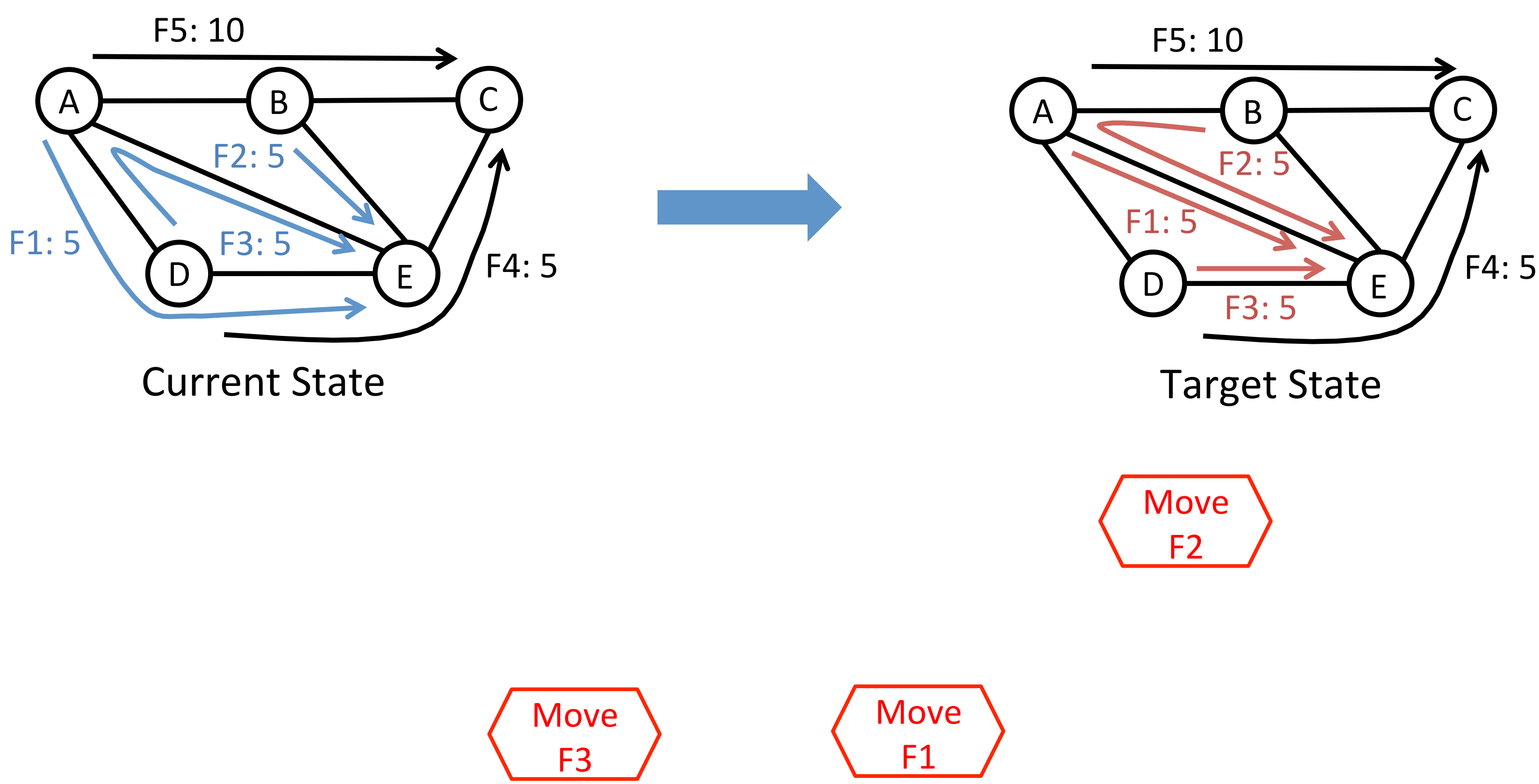


Target State

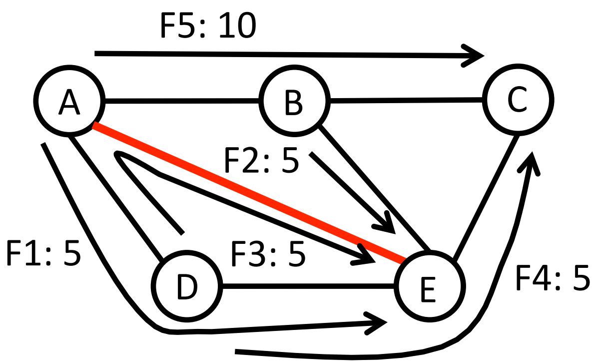
Dependency Graph Elements

Node	Operation node
	Resource node (link capacity, switch table size)
	Path node
Edge	Dependencies between nodes

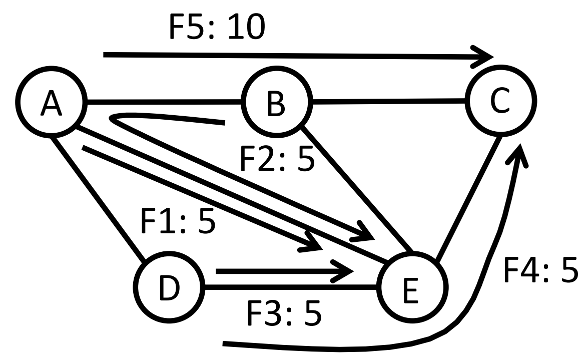
Dependency Graph Generation



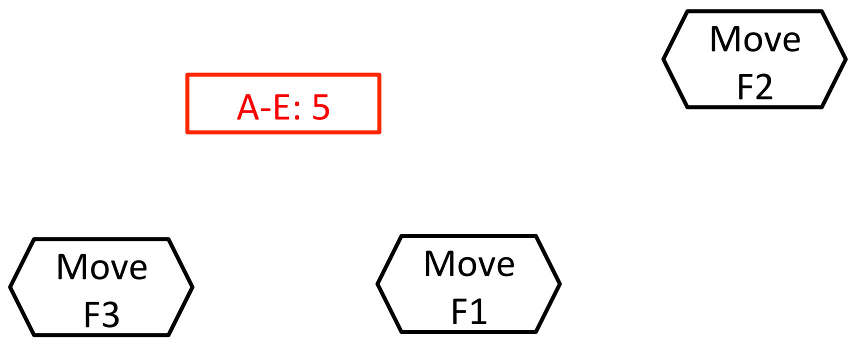
Dependency Graph Generation



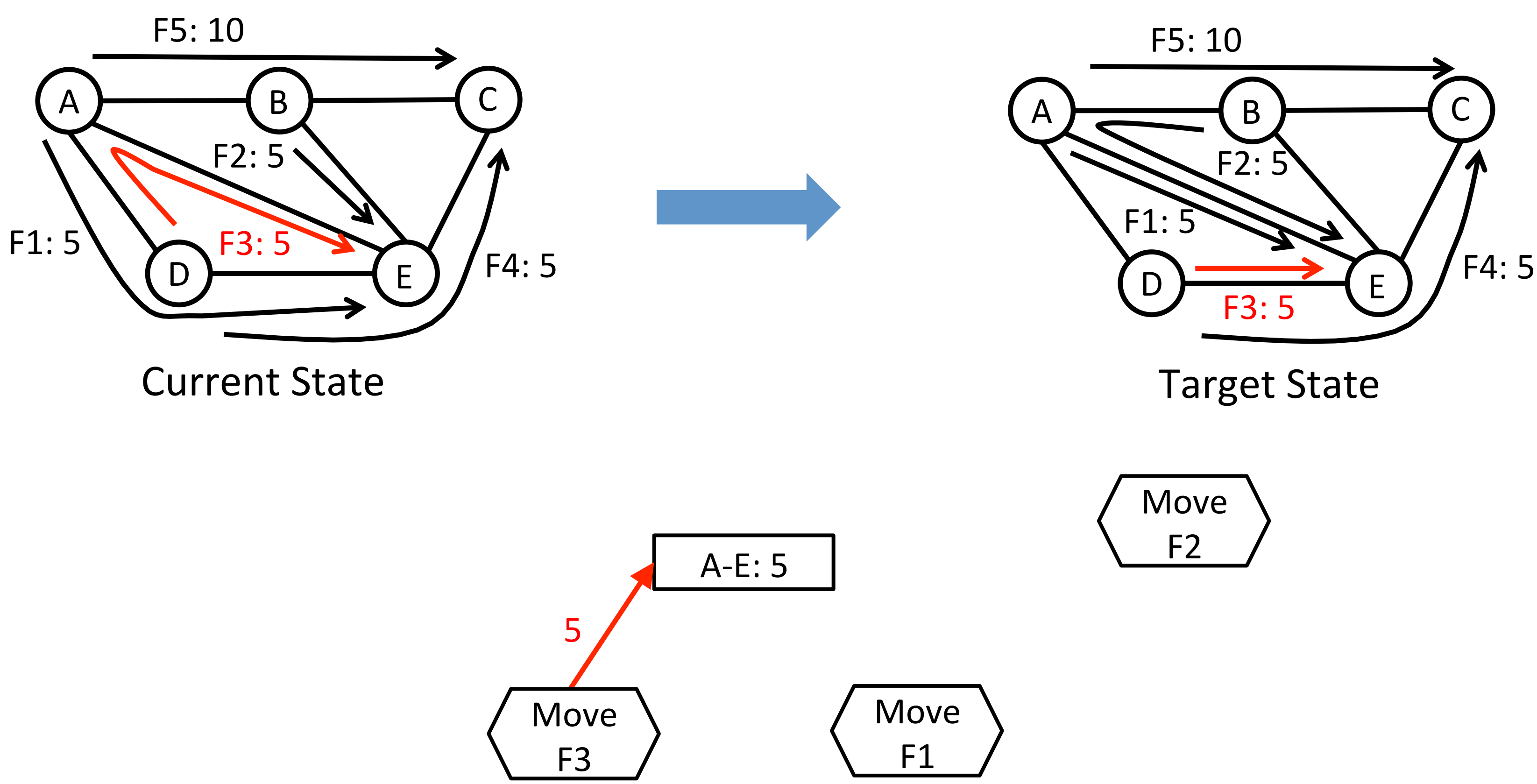
Current State



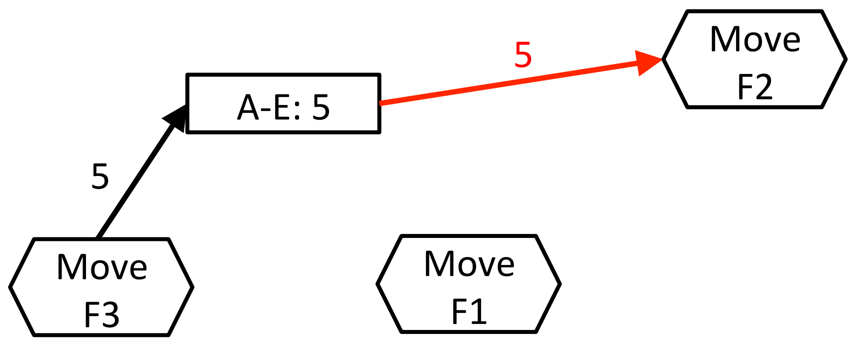
Target State



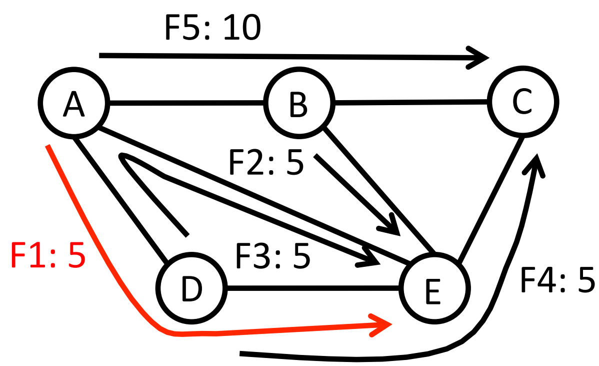
Dependency Graph Generation



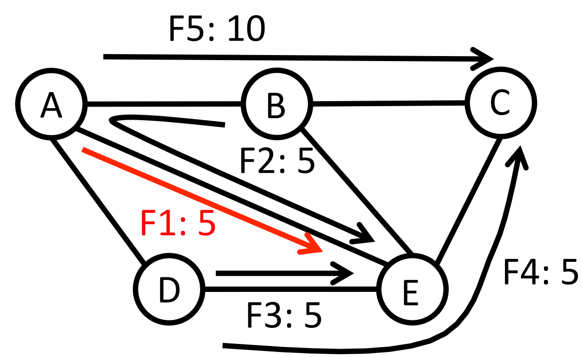
Dependency Graph Generation



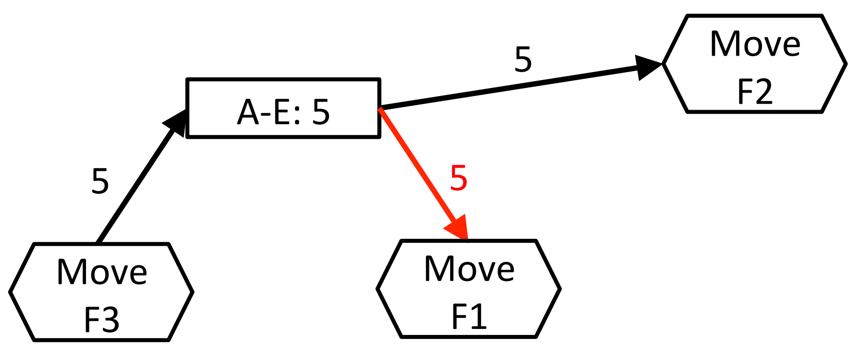
Dependency Graph Generation



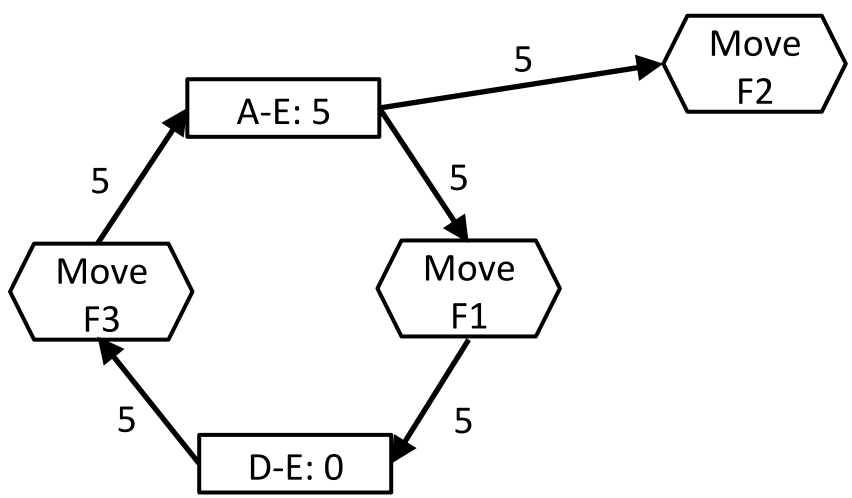
Current State



Target State



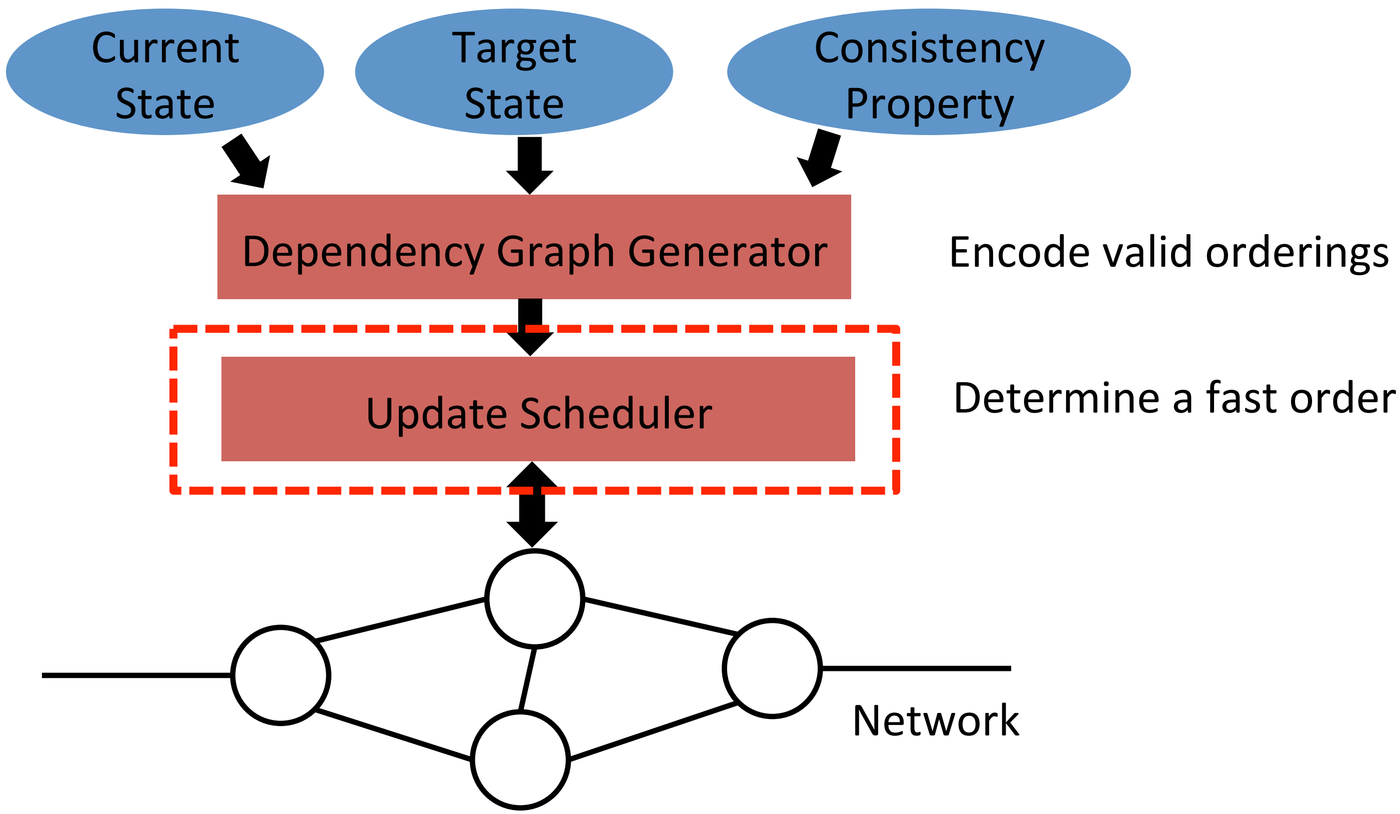
Dependency Graph Generation



Dependency Graph Generation

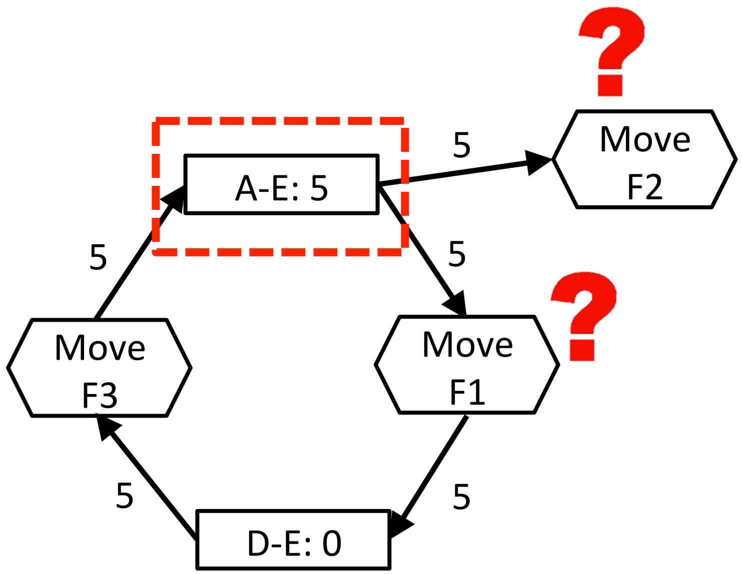
- Supported scenarios
  - Tunnel-based forwarding: WANs
  - WCMP forwarding: data center networks
- Supported consistency properties
  - Loop freedom
  - Blackhole freedom
  - Packet coherence
  - Congestion freedom
- Check paper for details

Dionysus Pipeline



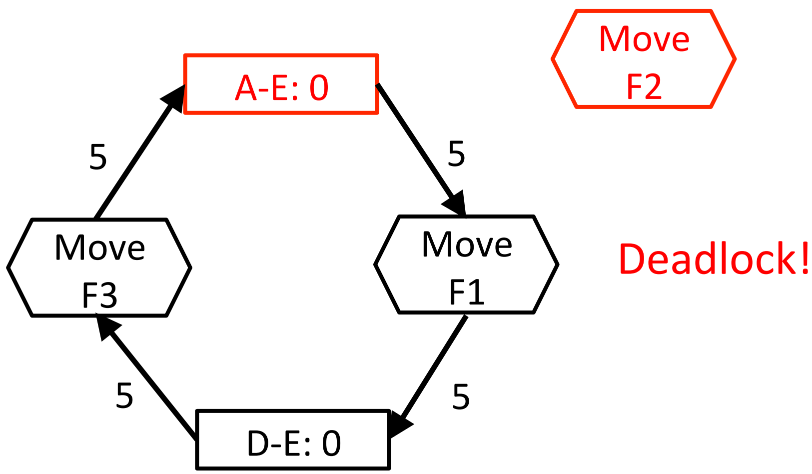
Dionysus Scheduling

- Scheduling as a resource allocation problem



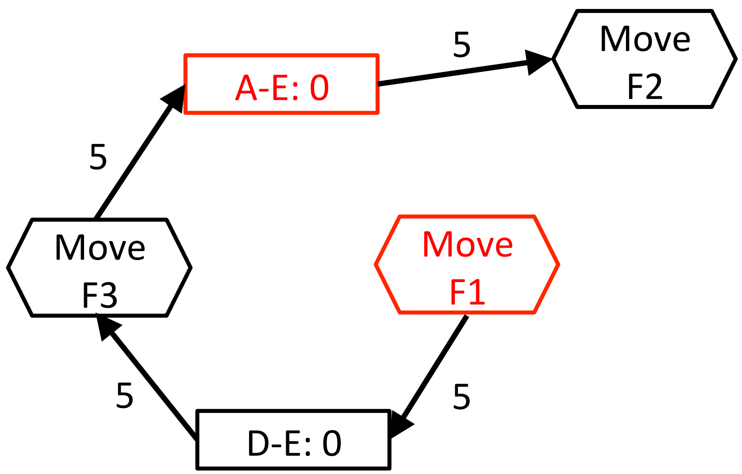
Dionysus Scheduling

- Scheduling as a resource allocation problem



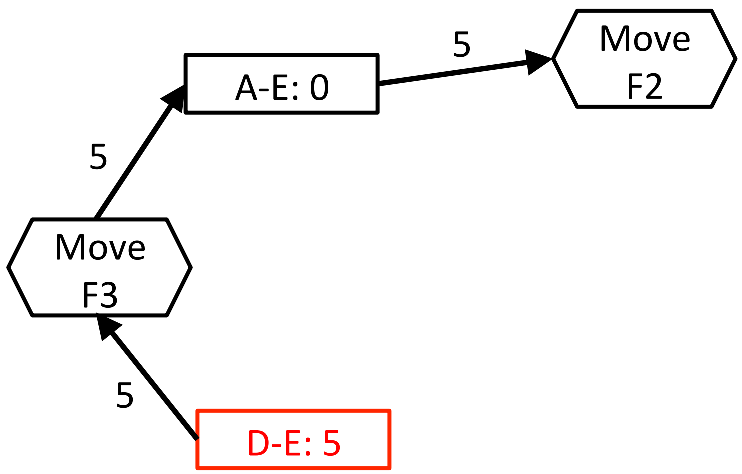
Dionysus Scheduling

- Scheduling as a resource allocation problem



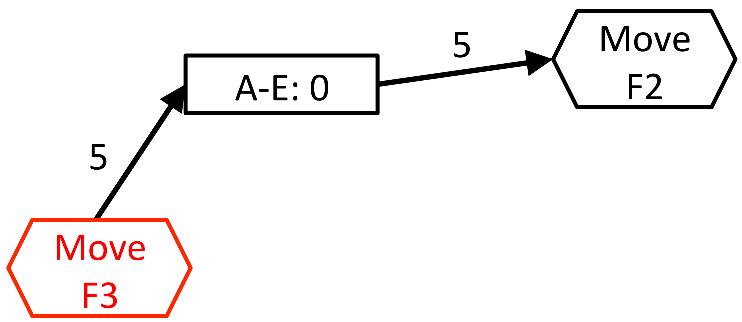
Dionysus Scheduling

- Scheduling as a resource allocation problem



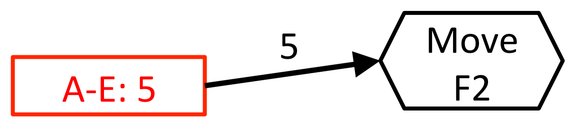
Dionysus Scheduling

- Scheduling as a resource allocation problem



Dionysus Scheduling

- Scheduling as a resource allocation problem



Dionysus Scheduling

- Scheduling as a resource allocation problem

Move
F2

Done!

Dionysus Scheduling

- Scheduling as a resource allocation problem

- NP-complete problems under link capacity and switch table size constraints

- Approach
  - DAG: always feasible, critical-path scheduling
  - General case: covert to a virtual DAG
  - Rate limit flows to resolve deadlocks

Critical-Path Scheduling

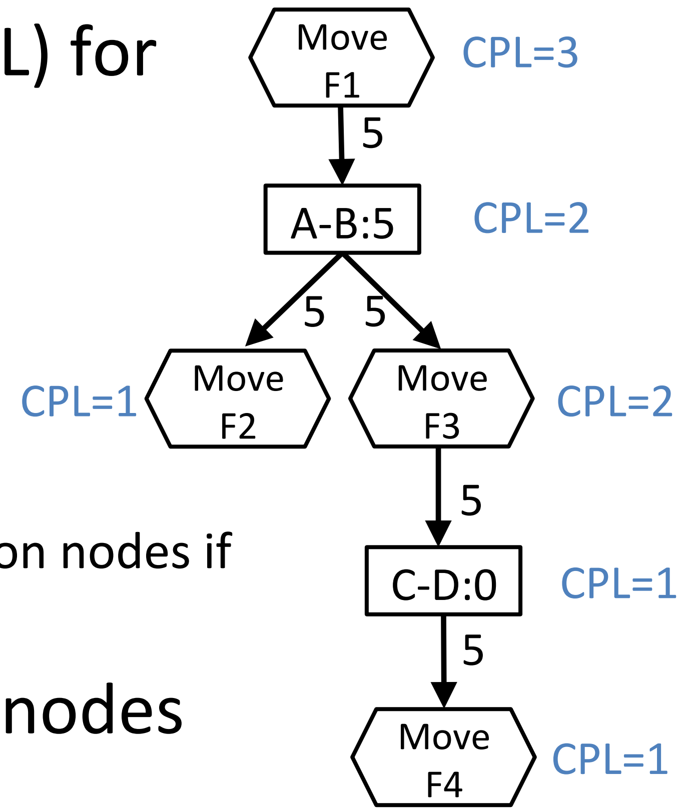
- Calculate critical-path length (CPL) for each node

$$CPL_i = w_i + \max_{j \in children(i)} CPL_j$$

$$w_i = \begin{cases} 1, if\ i\ is\ operation\ node \\ 0, otherwise \end{cases}$$

  – Extension: assign larger weight to operation nodes if we know in advance the switch is slow

- Resource allocated to operation nodes with larger CPLs



Critical-Path Scheduling

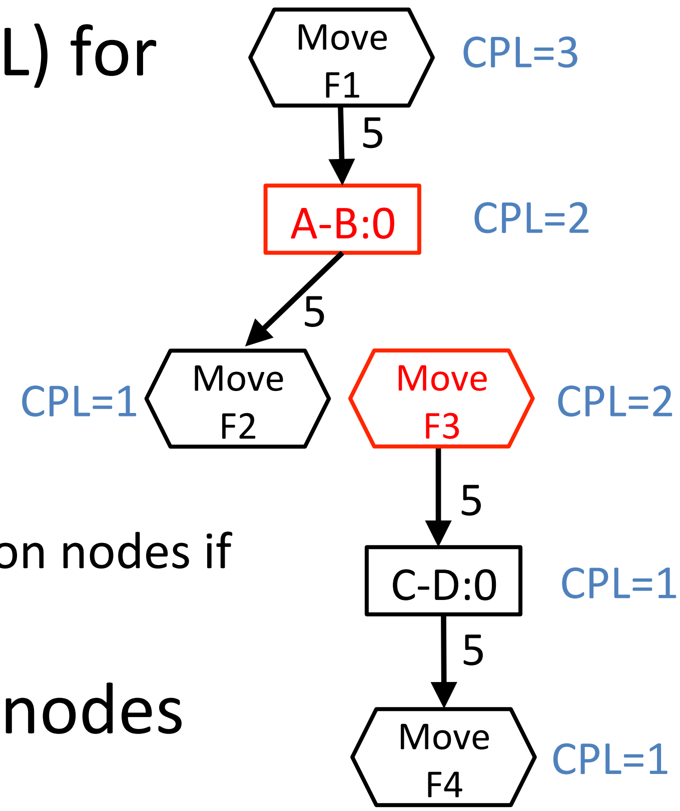
- Calculate critical-path length (CPL) for each node

$$CPL_i = w_i + \max_{j \in children(i)} CPL_j$$

$$w_i = \begin{cases} 1, if\ i\ is\ operation\ node \\ 0, otherwise \end{cases}$$

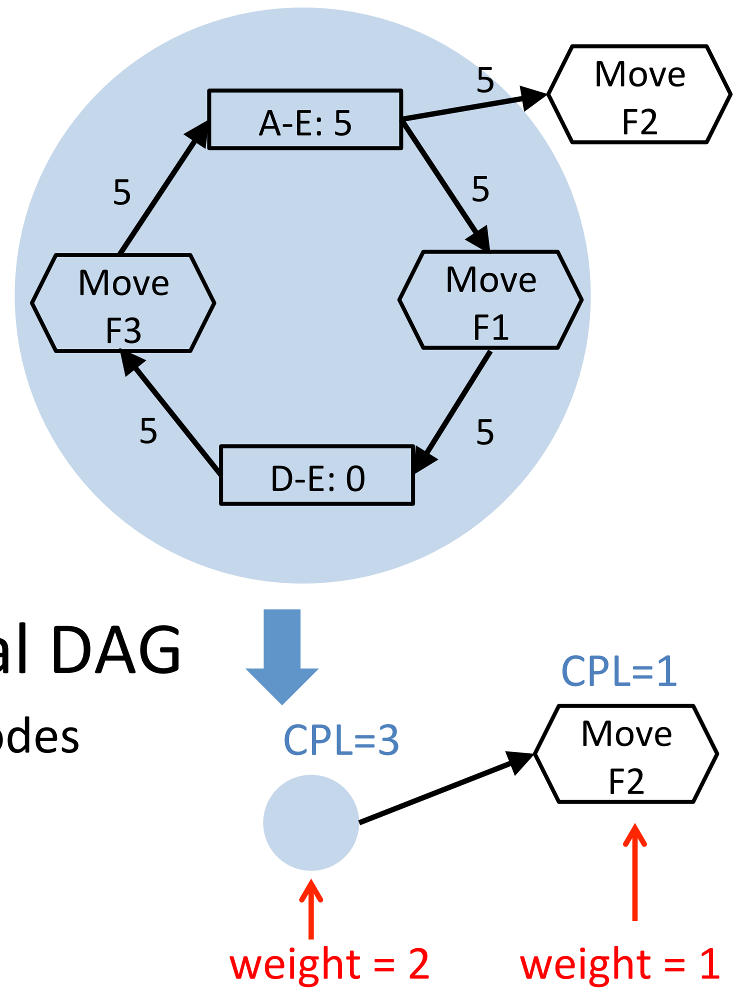
  - Extension: assign larger weight to operation nodes if we know in advance the switch is slow

- Resource allocated to operation nodes with larger CPLs



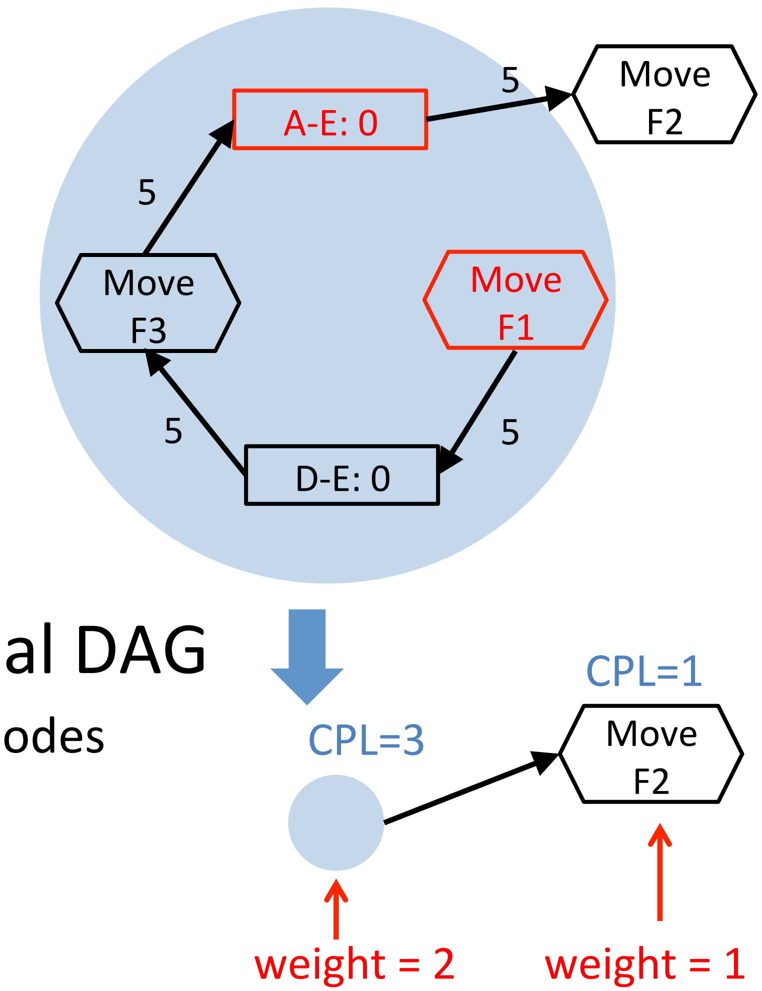
Handling Cycles

- Convert to virtual DAG
  - Consider each strongly connected component (SCC) as a virtual node
- Critical-path scheduling on virtual DAG
  - Weight w_i of SCC: number of operation nodes

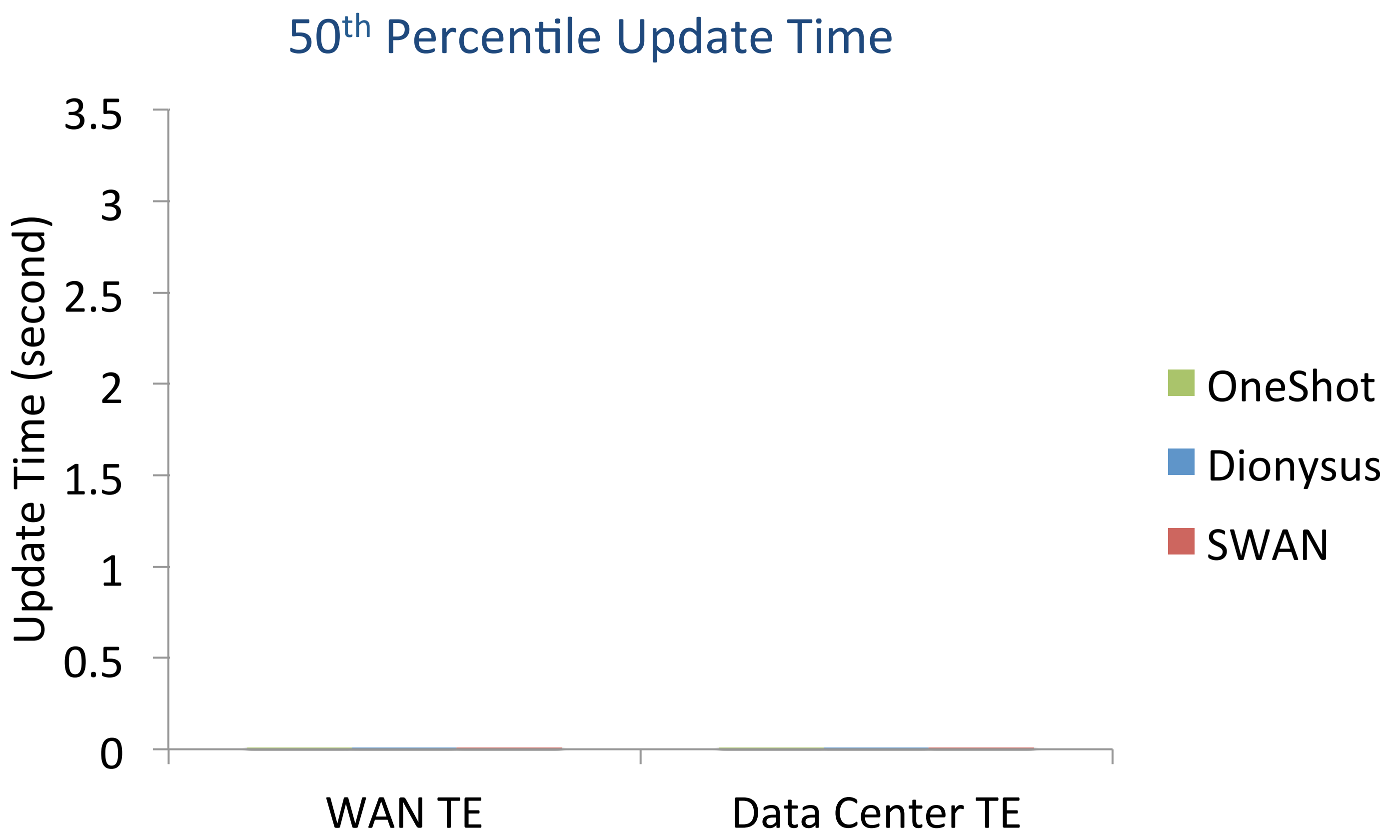


Handling Cycles

- Convert to virtual DAG
  - Consider each strongly connected component (SCC) as a virtual node
- Critical-path scheduling on virtual DAG
  - Weight w_i of SCC: number of operation nodes



Evaluation: Traffic Engineering

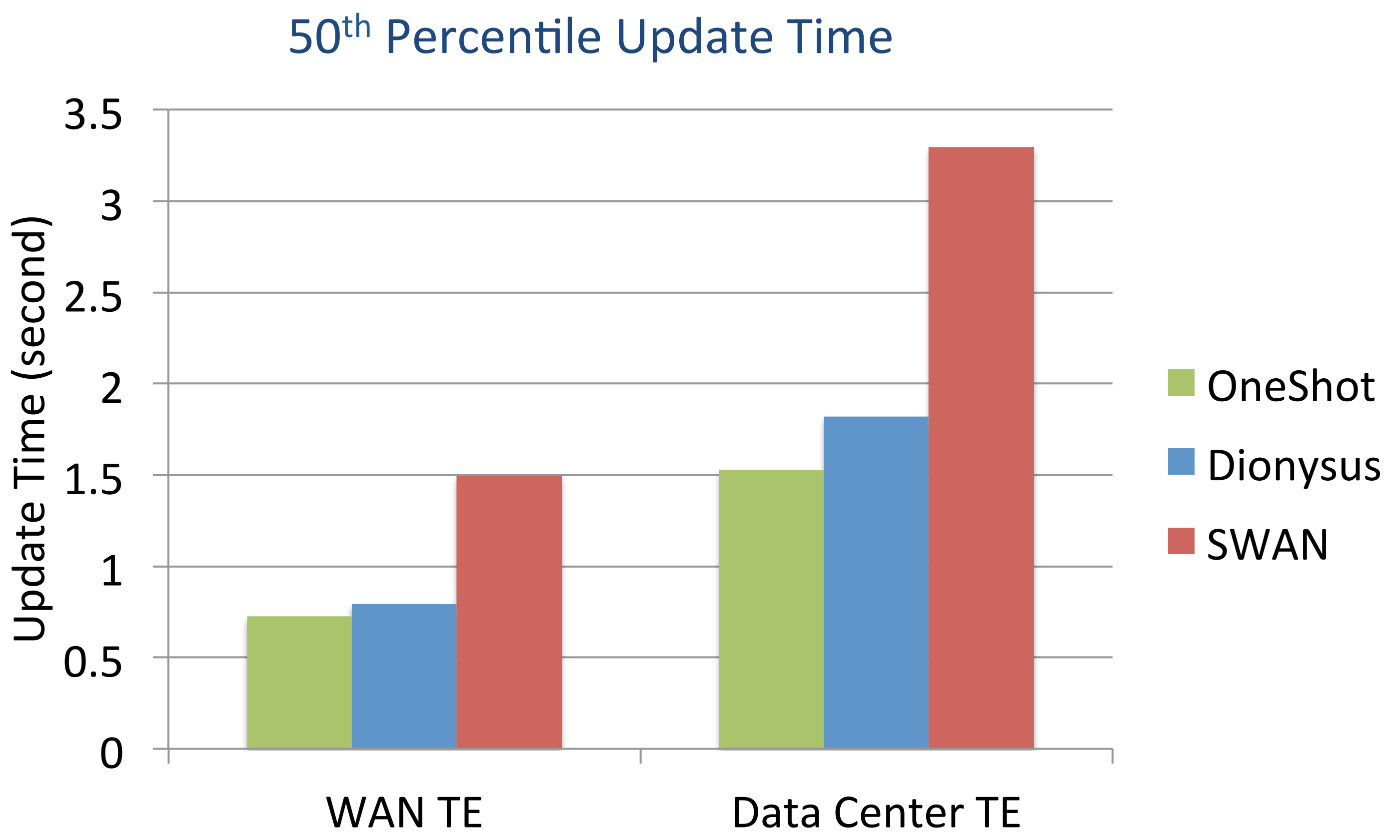


Not congestion-free

Congestion-free

Congestion-free

Evaluation: Traffic Engineering



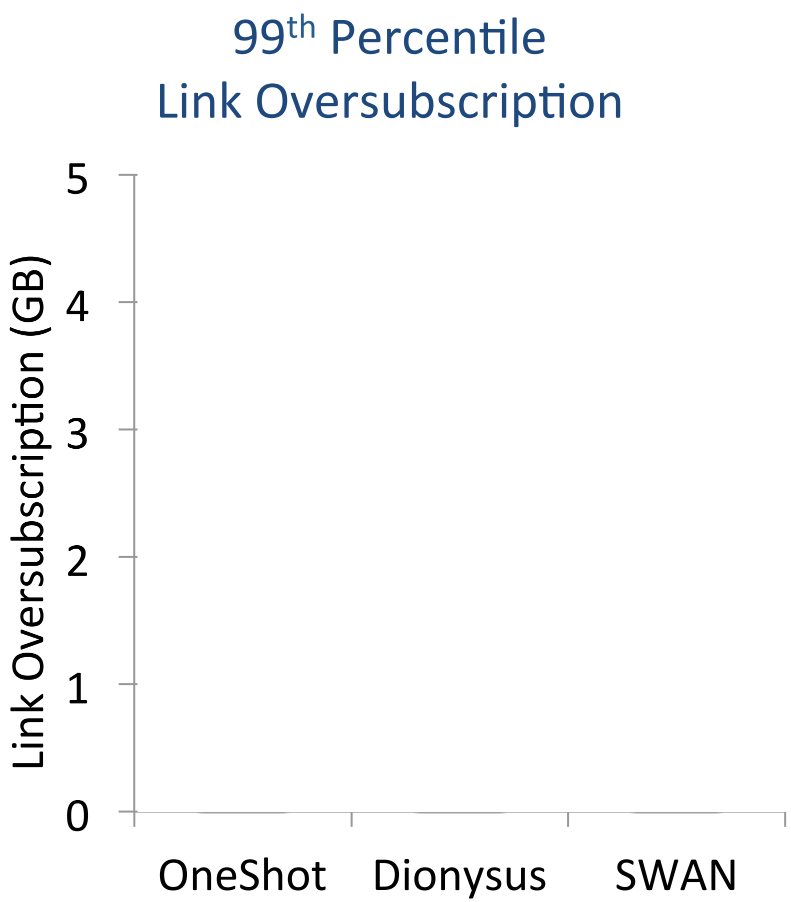
Not congestion-free

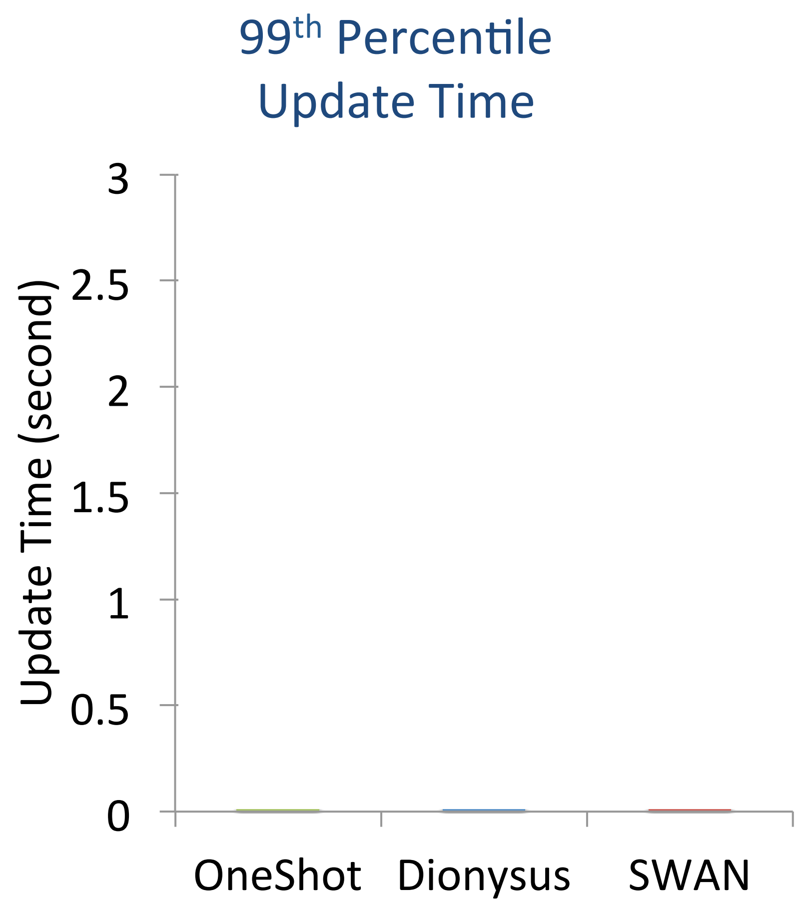
Congestion-free

Congestion-free

Improve 50th percentile update speed by 80%
compared to static scheduling (SWAN), close to OneShot

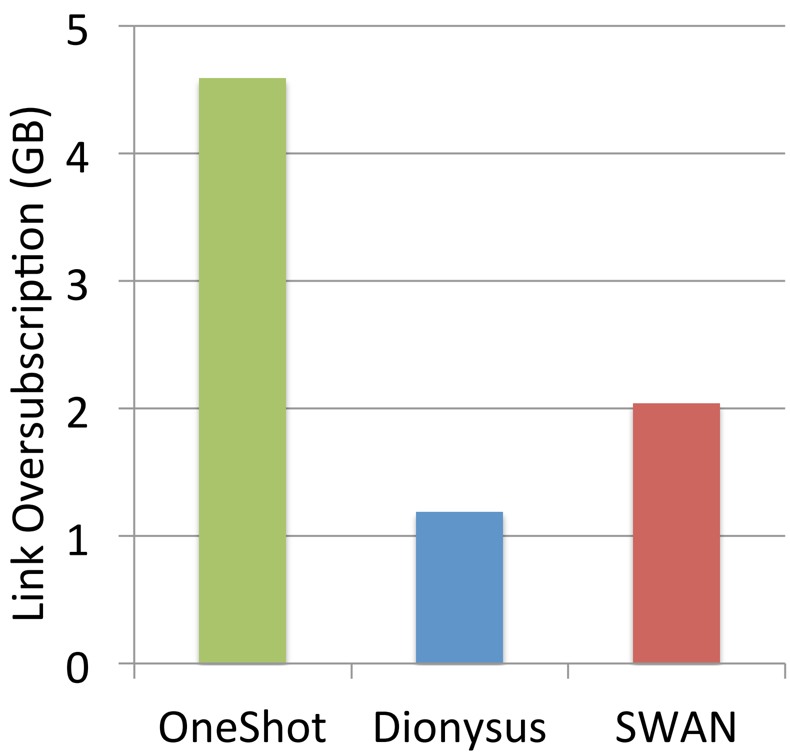
Evaluation: Failure Recovery



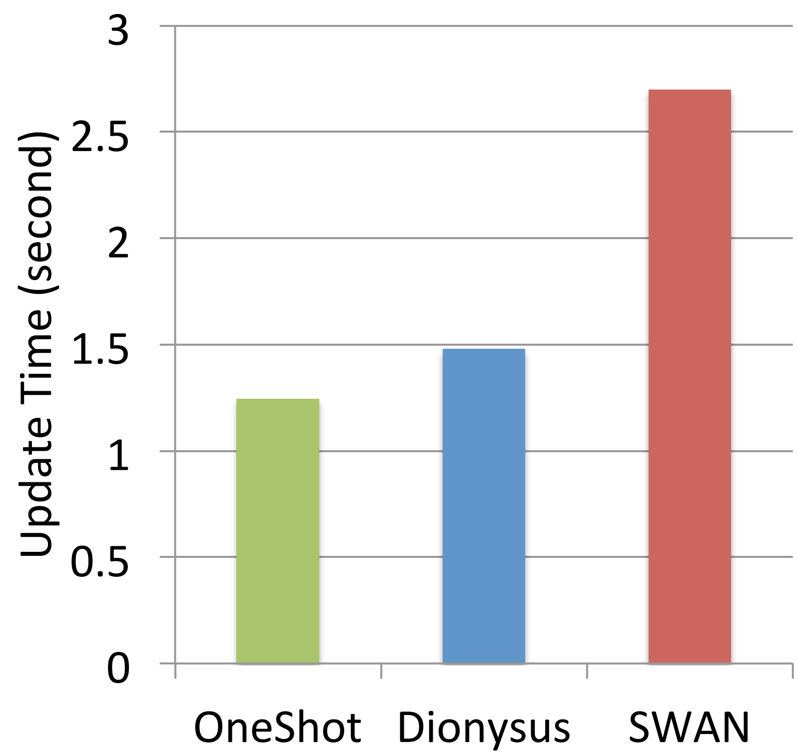


Evaluation: Failure Recovery

99th Percentile
Link Oversubscription



99th Percentile
Update Time



Reduce 99th percentile link oversubscription by 40% compared to static scheduling (SWAN)

Improve 99th percentile update speed by 80% compared to static scheduling (SWAN)

Conclusion

- Dionysus provides fast, consistent network updates through dynamic scheduling
  - Dependency graph: compactly encode orderings
  - Scheduling: dynamically schedule operations

Dionysus enables more agile SDN control loops

Thanks!