Dynamic Scheduling of Network Updates

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

Current State

Target State
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

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

Current State

Target State

Deadlock
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

Current State

Consistent update plan

Target State
Dionysus Pipeline

Current State → Dependency Graph Generator → Update Scheduler

Target State → Dependency Graph Generator

Consistency Property

Encode valid orderings
Determine a fast order

Network
Dependency Graph Generation

Current State

Target State

Dependency Graph Elements

- Operation node
- Resource node (link capacity, switch table size)
- Path node
- Dependencies between nodes
Dependency Graph Generation

Current State

Target State

Move F2

Move F3

Move F1
Dependency Graph Generation

Current State

Target State

A-E: 5

Move F2

Move F3

Move F1

A-E: 5

Move F2
Dependency Graph Generation

Current State

Target State

Move F2

Move F3

Move F1

A-E: 5
Dependency Graph Generation

Current State

Target State

Move F3

Move F1

Move F2

A-E: 5

F1: 5

F2: 5

F3: 5

F4: 5

F5: 10

F1: 5

F2: 5

F3: 5

F4: 5

F5: 10
Dependency Graph Generation

Current State

Target State

Move F3

Move F1

Move F2

A-E: 5

F1: 5

F2: 5

F3: 5

F4: 5

F5: 10

Move F3

Move F1

Move F2

A-E: 5

F1: 5

F2: 5

F3: 5

F4: 5

F5: 10
Dependency Graph Generation

Current State

Target State

Move F3

Move F1

Move F2

A-E: 5

F1: 5

F2: 5

F3: 5

F4: 5

D-E: 0

F5: 10

F5: 10

F1: 5

F2: 5

F3: 5

F4: 5

A

B

C

D

E

F
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

Current State

Target State

Consistency Property

Encode valid orderings

Determine a fast order

Dependency Graph Generator

Update Scheduler

Network
Dionysus Scheduling

- Scheduling as a resource allocation problem

![Diagram of scheduling process]
Dionysus Scheduling

- Scheduling as a resource allocation problem

```
A-E: 0
5
Move F3
5
D-E: 0
5
Move F1
5
Move F2

Deadlock!
```
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

\[ CPL_i = w_i + \max_{j \in \text{children}(i)} CPL_j \]

\[ w_i = \begin{cases} 1, & \text{if } i \text{ is operation node} \\ 0, & \text{otherwise} \end{cases} \]

– 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

\[ CPL_i = w_i + \max_{j \in \text{children}(i)} CPL_j \]

\[ w_i = \begin{cases} 1, & \text{if } i \text{ is operation node} \\ 0, & \text{otherwise} \end{cases} \]

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

- Resource allocated to operation nodes with larger CPLs

```
Move F1  CPL=3
\downarrow 5
Move F2  CPL=1
\downarrow 5
Move F3  CPL=2
\downarrow 5
Move F4  CPL=1
```

```
A-B:0  CPL=2
\downarrow 5
C-D:0  CPL=1
\downarrow 5
```
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

50th Percentile Update Time

- OneShot: Not congestion-free
- Dionysus: Congestion-free
- SWAN: Congestion-free
Evaluation: Traffic Engineering

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

99th Percentile Link Oversubscription

99th Percentile Update Time
Evaluation: Failure Recovery

99th Percentile Link Oversubscription

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

99th Percentile Update Time

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!