Towards Decentralized Fast Consistent Updates

MARCO CHIESA

UNIVERSITÉ CATHOLIQUE
DE LOUVAIN

JOINT WORK WITH:

THANH DANG NGUYEN (UC LOUVAIN),
MARCO CANINI (UC LOUVAIN)

Updating the network configuration

A fundamental network operations performed whenever:

- network policy changes
- network devices fail
- traffic load changes
- security attacks

Updating the network configuration

A fundamental network operations performed whenever:

- network policy changes —— planned updates
- network devices fail
- traffic load changes
- security attacks

Updating the network configuration

A fundamental network operations performed whenever:

- network policy changes

planned updates

- network devices fail
- traffic load changes
- network attacks

unplanned updates

Unplanned network update scenario

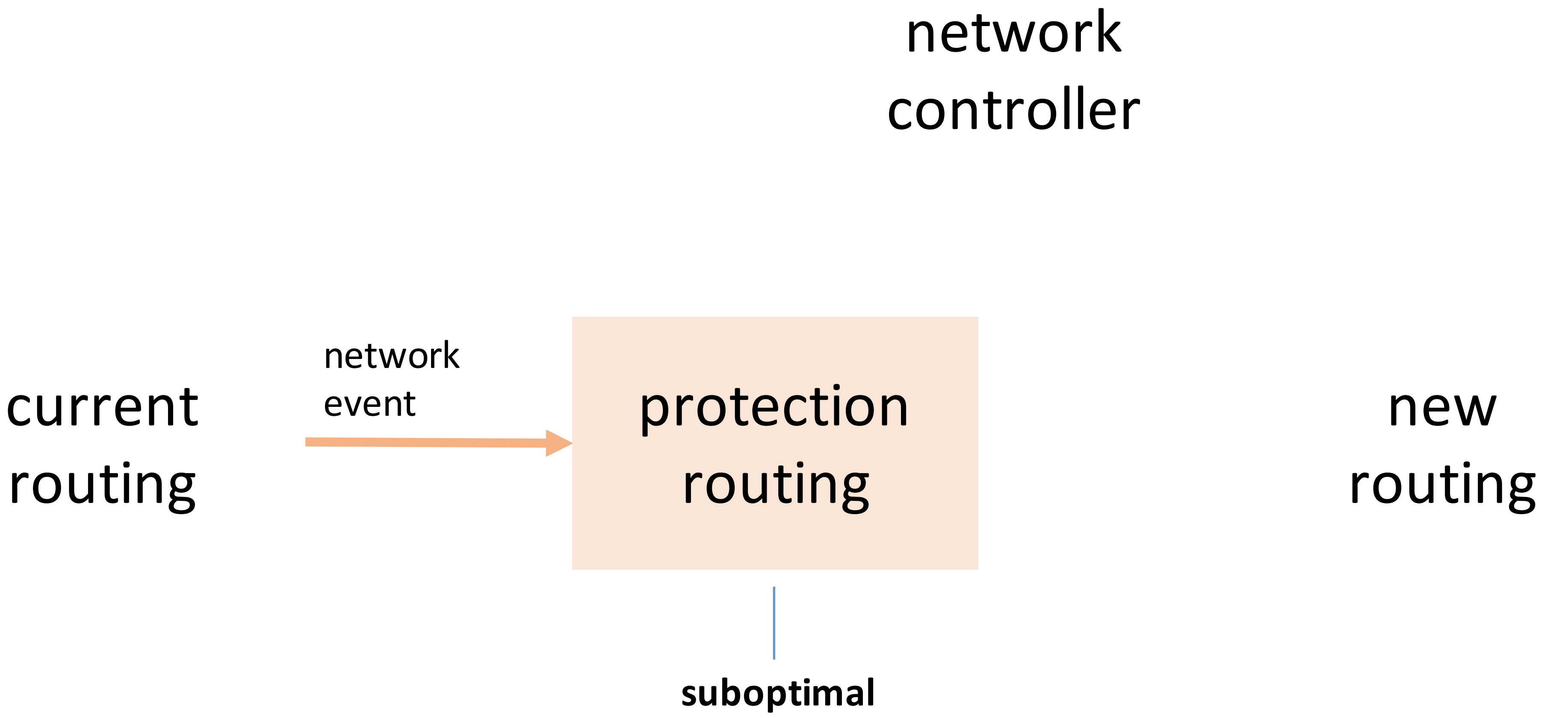
network controller

current routing

protection routing

new routing

Unplanned network update scenario



network controller

current routing

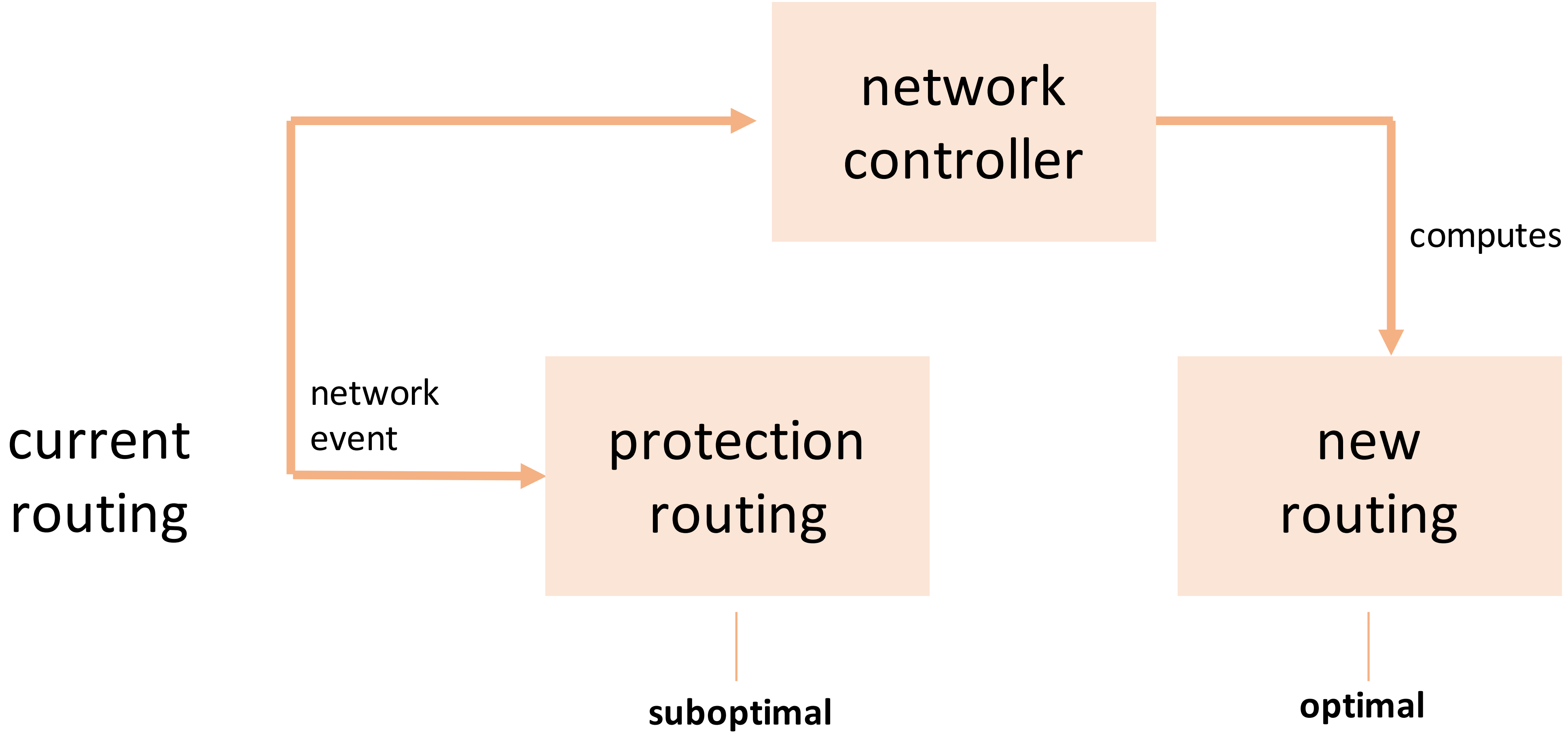
network event

protection routing

suboptimal

new routing

Unplanned network update scenario



current routing

network event

network controller

computes

protection routing

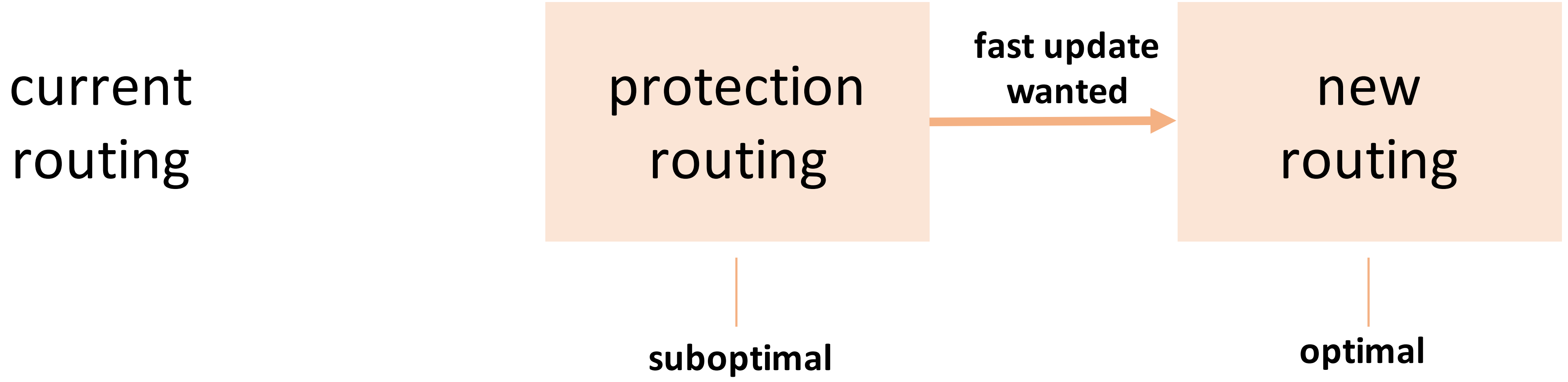
new routing

suboptimal

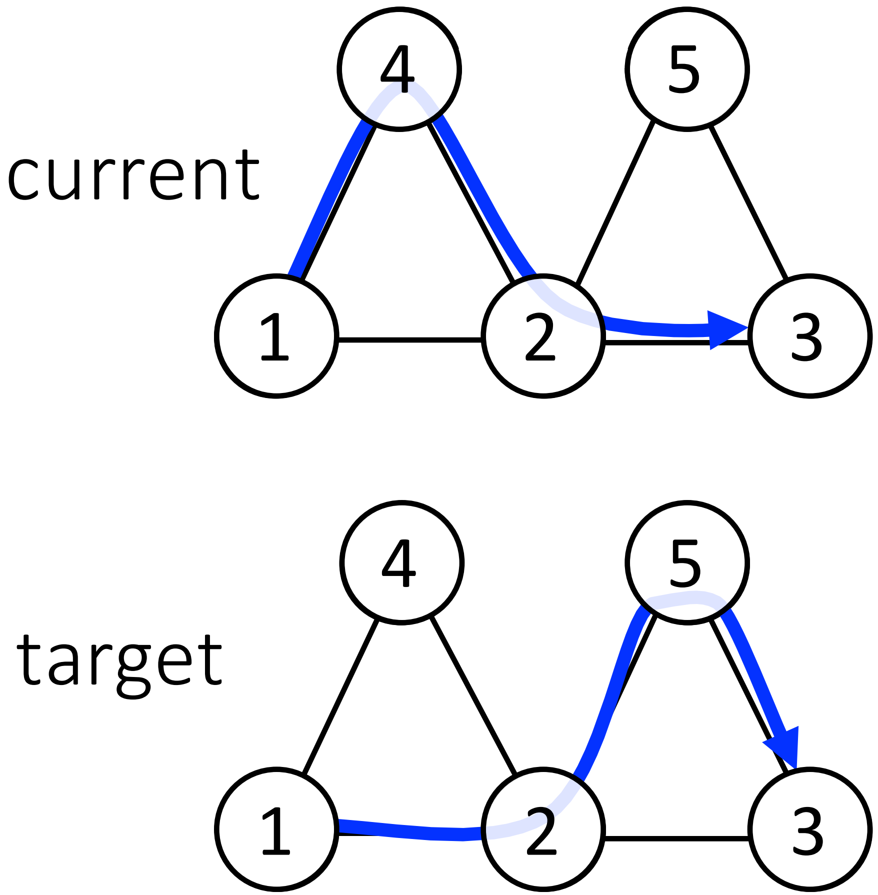
optimal

Desiderata: fast network update to the new configuration

network controller



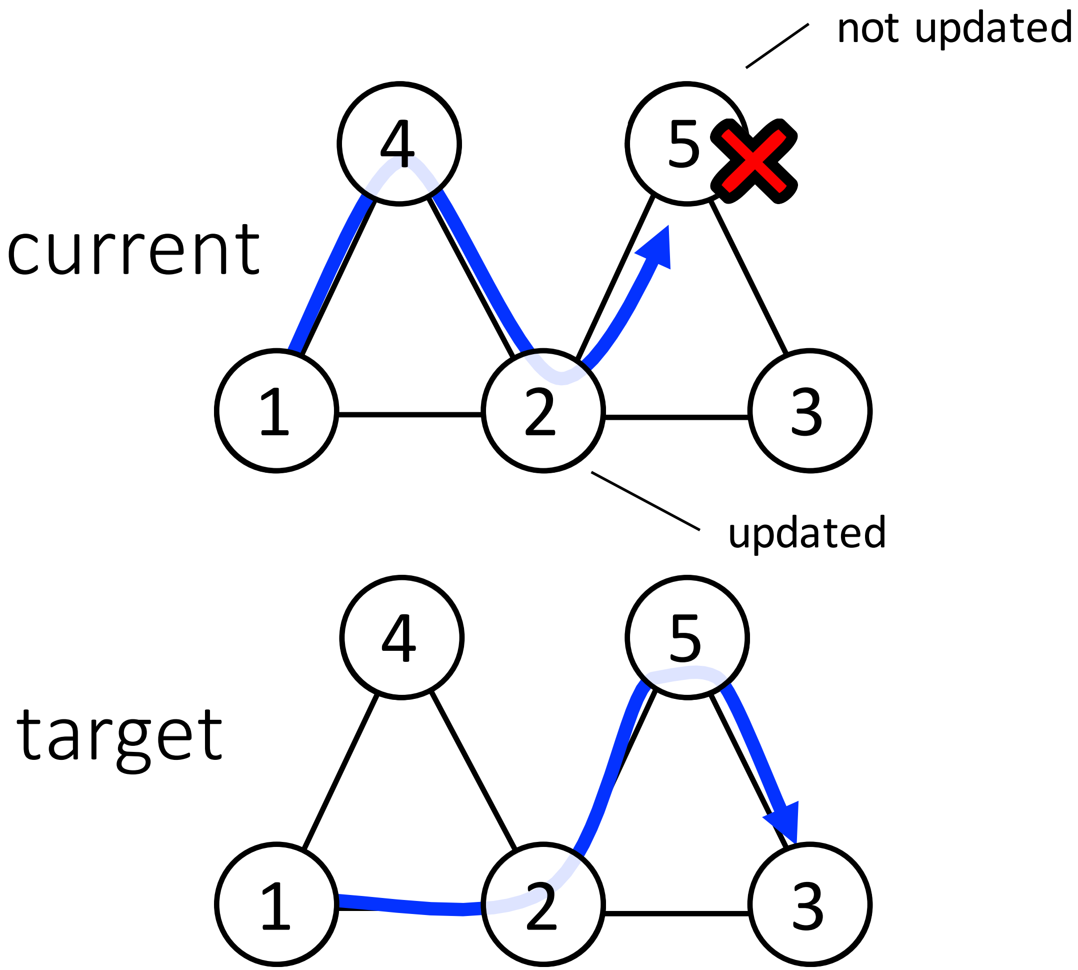
One-shot updates are **dangerous**



The network is a distributed system

Asynchronous update messages

One-shot updates are **dangerous**

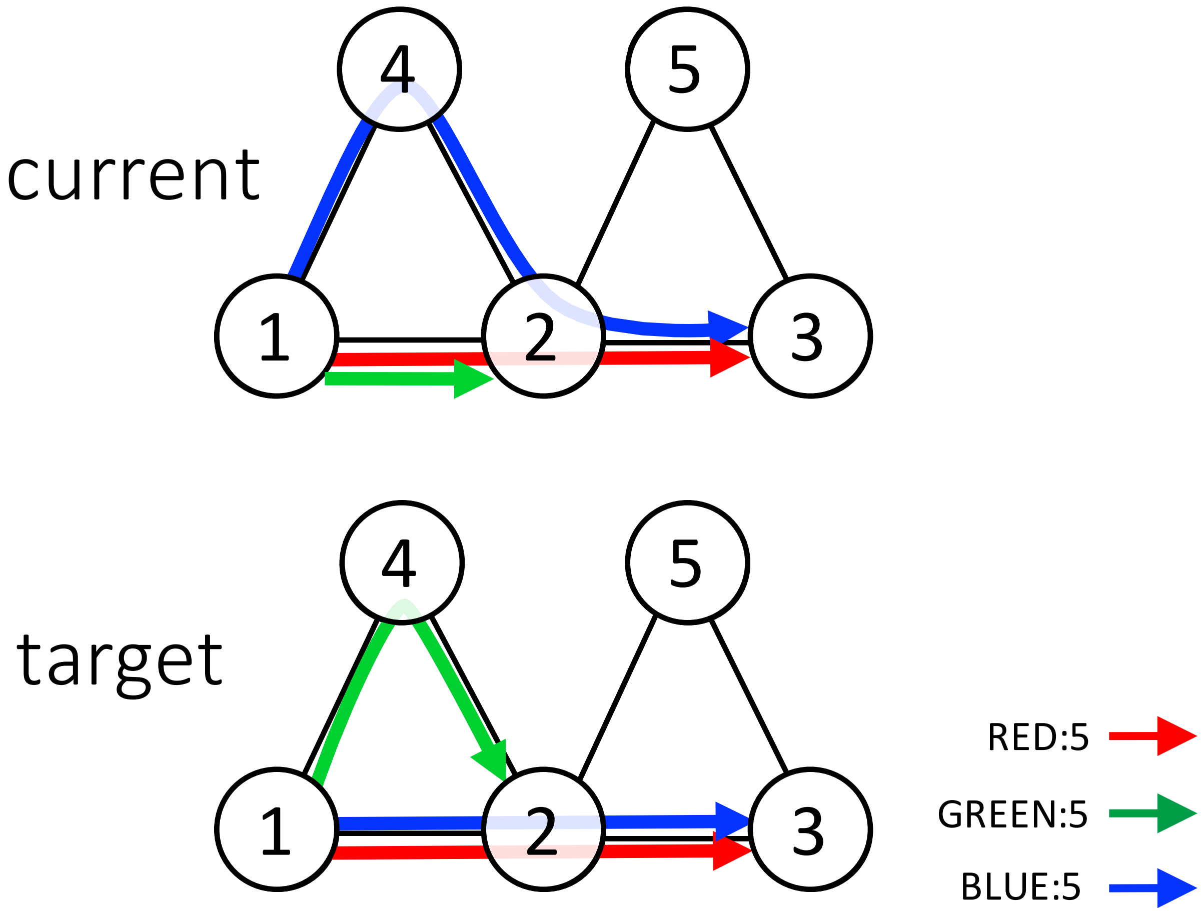


The network is a distributed system

Asynchronous update messages

If Switch 2 updates BLUE before Switch 5 installs the forwarding rule for BLUE, traffic is blackholed!!

Flow dependencies must be met to avoid link congestion



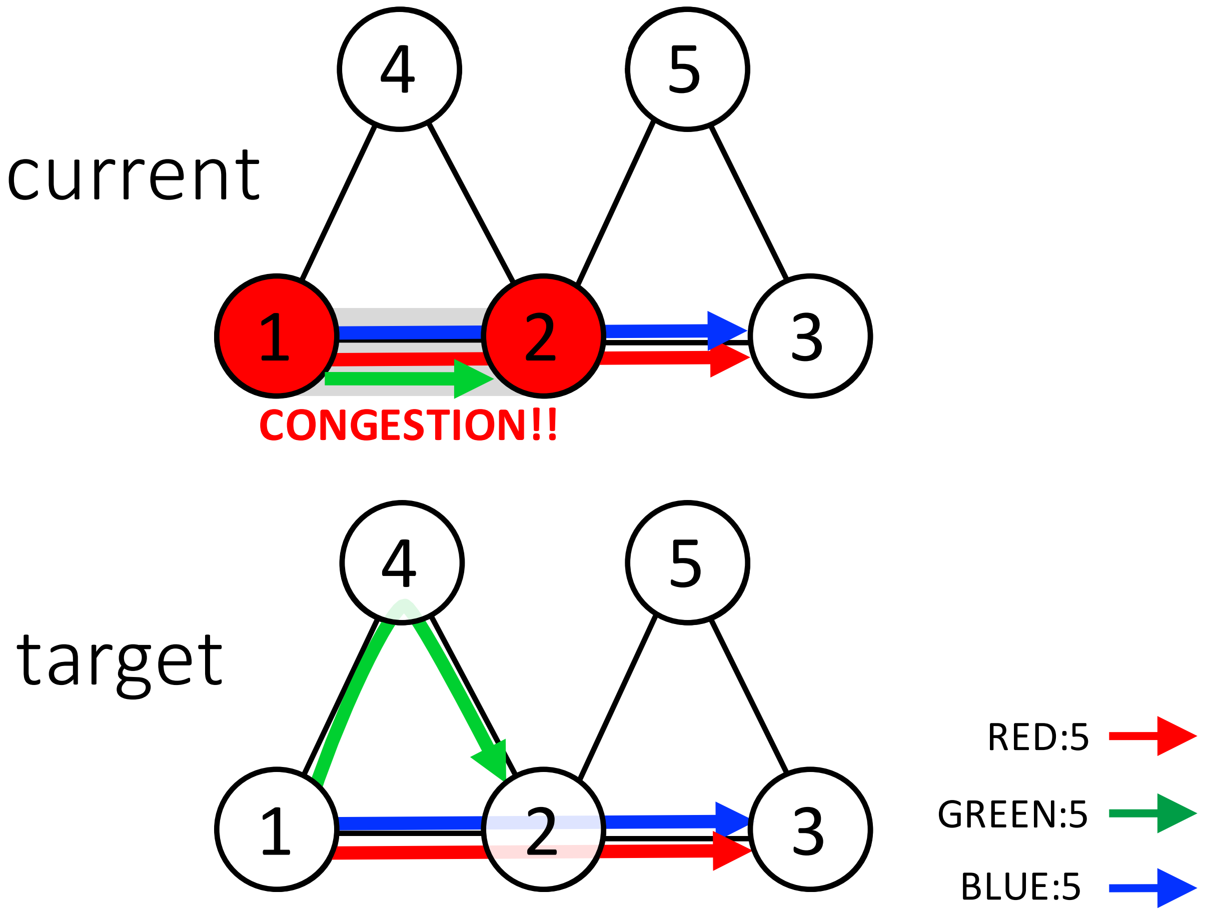
A simple network:

- Each link capacity is 10
- Each flow requires 5 units

Goal: update from **current** to **target**

- GREEN and BLUE are updated

Flow dependencies must be met to avoid link congestion



A simple network:

- Each link capacity is 10
- Each flow requires 5 units

Goal: update from **current** to **target**

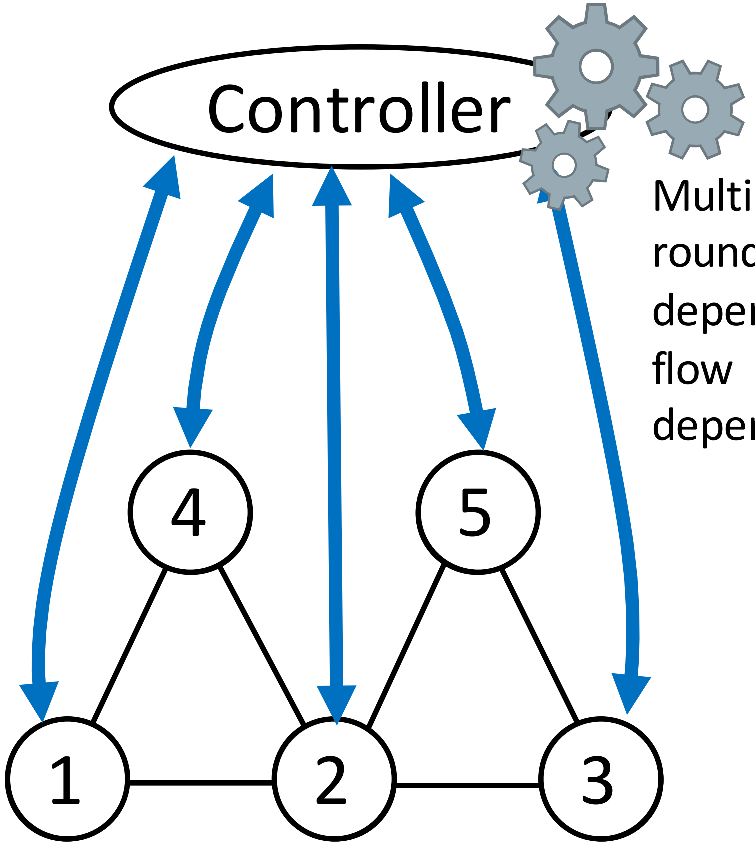
- GREEN and BLUE are updated

If Switch 1 updates BLUE before GREEN, link 1-2 is congested!!

Most relevant related work

Dyonisus: centralized synchronization scheduling computation

Synchronization between switches and controller slows down the update



Multiple rounds depending on flow dependencies

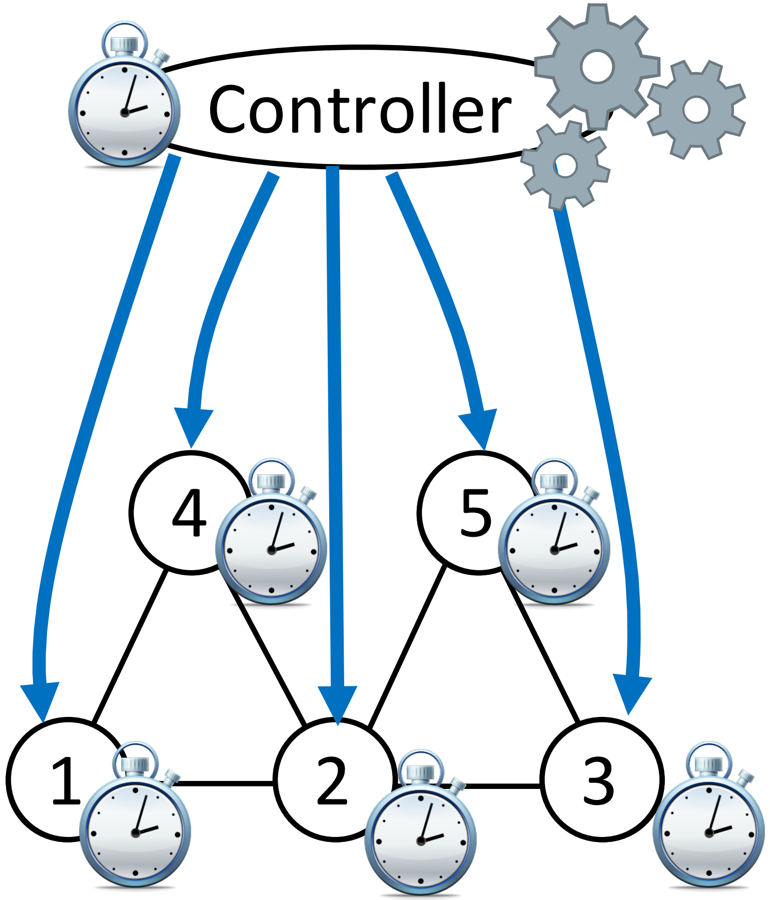
Most relevant related work

Dyonisus: centralized synchronization scheduling computation

Synchronization between switches and controller slows down the update

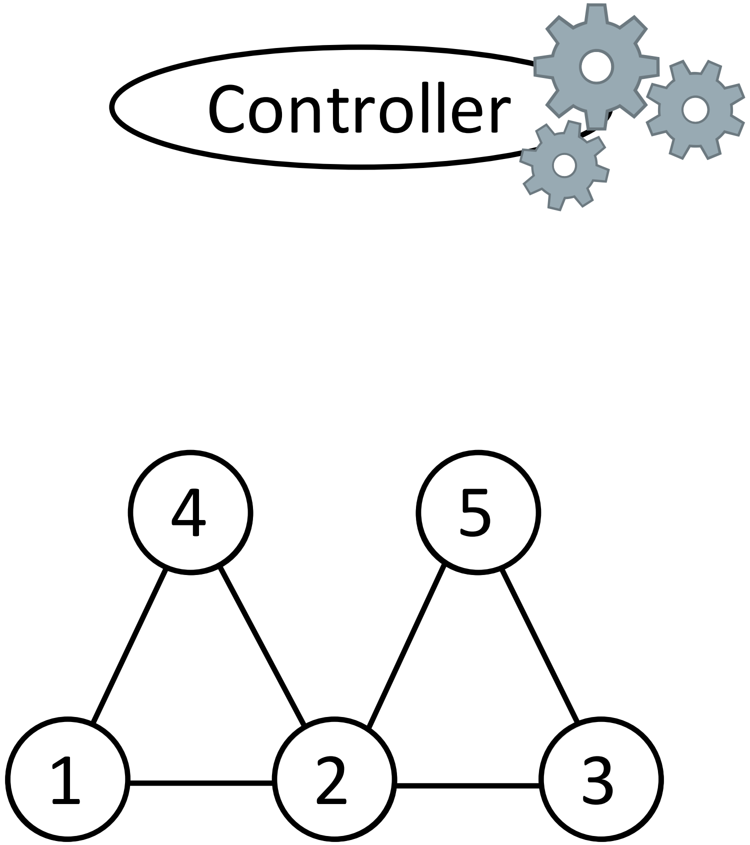
TIME4: one-shot update by means of clock synchronization

Inaccuracy in clock synchronization leads to anomalies



Ez-segway

Decentralized fast network update

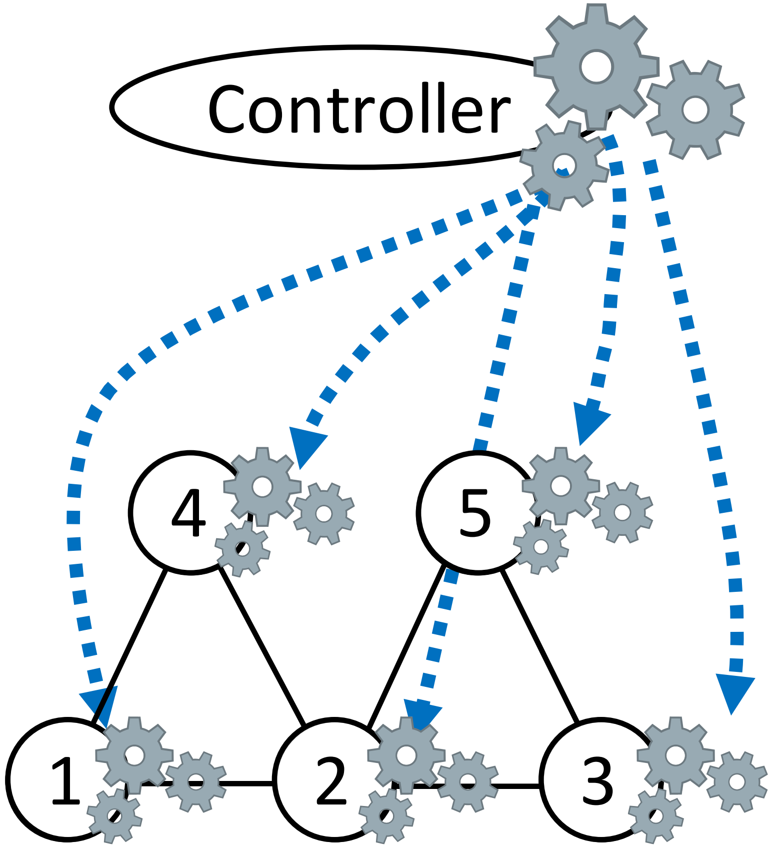


Ez-segway

Decentralized fast network update

<u>Key idea</u>

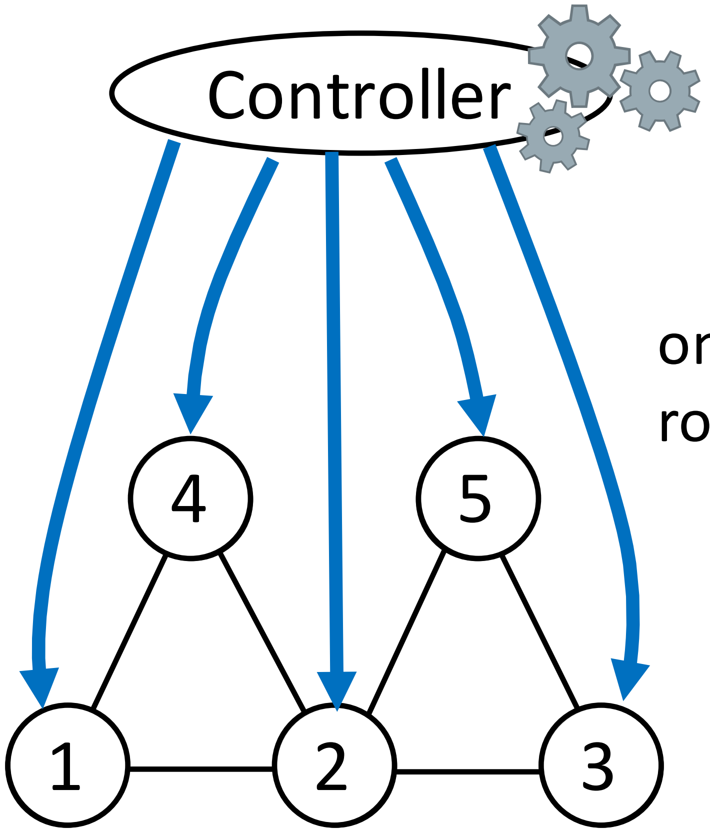
Move simple, yet powerful, logic from the controller to the switches



Ez-segway
Separation of concerns

Central controller role:

- detects flow dependencies
- computes flows *partial* ordering
- sends ordering to the switches



only one round

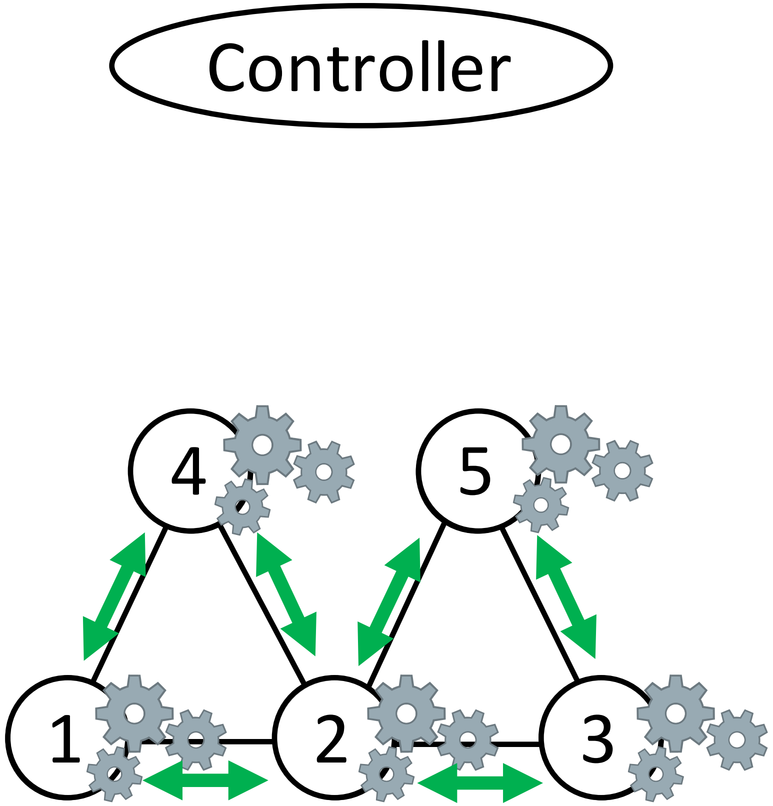
Ez-segway
Separation of concerns

Central controller role:

- detects flow dependencies
- computes flows *partial* ordering
- sends ordering to the switches

Switches role:

- combine local and global (pre-computed) information to perform the update
- coordinate with neighbors



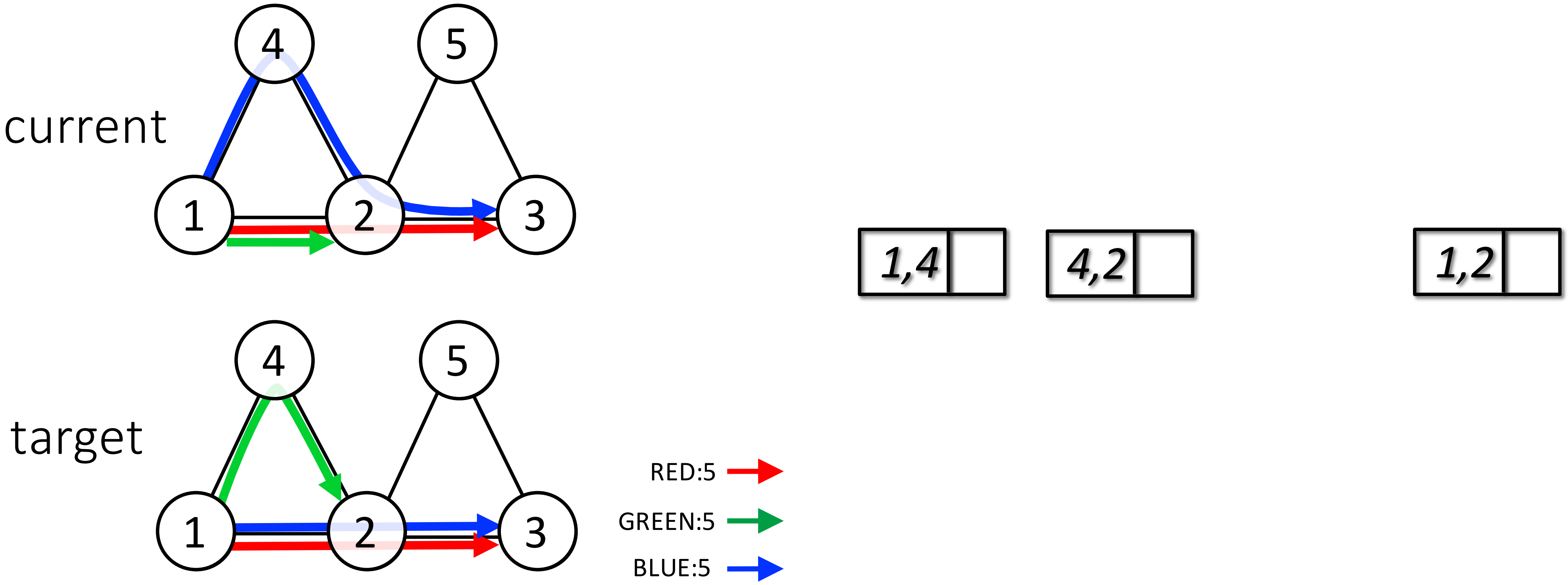
Ez-segway
the controller perspective

Central controller role:

1. detects flow dependencies
   - constructs a dependency graph

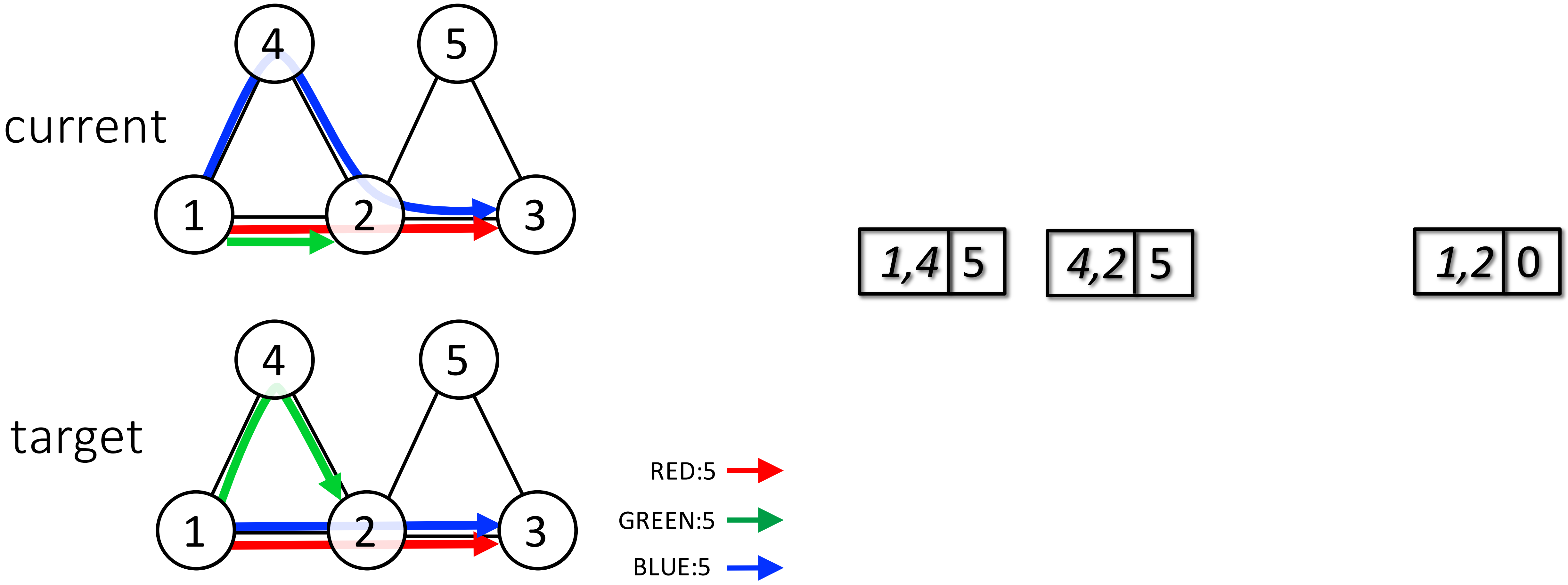
The dependency graph

Central controller role: 1. Detects flow dependencies



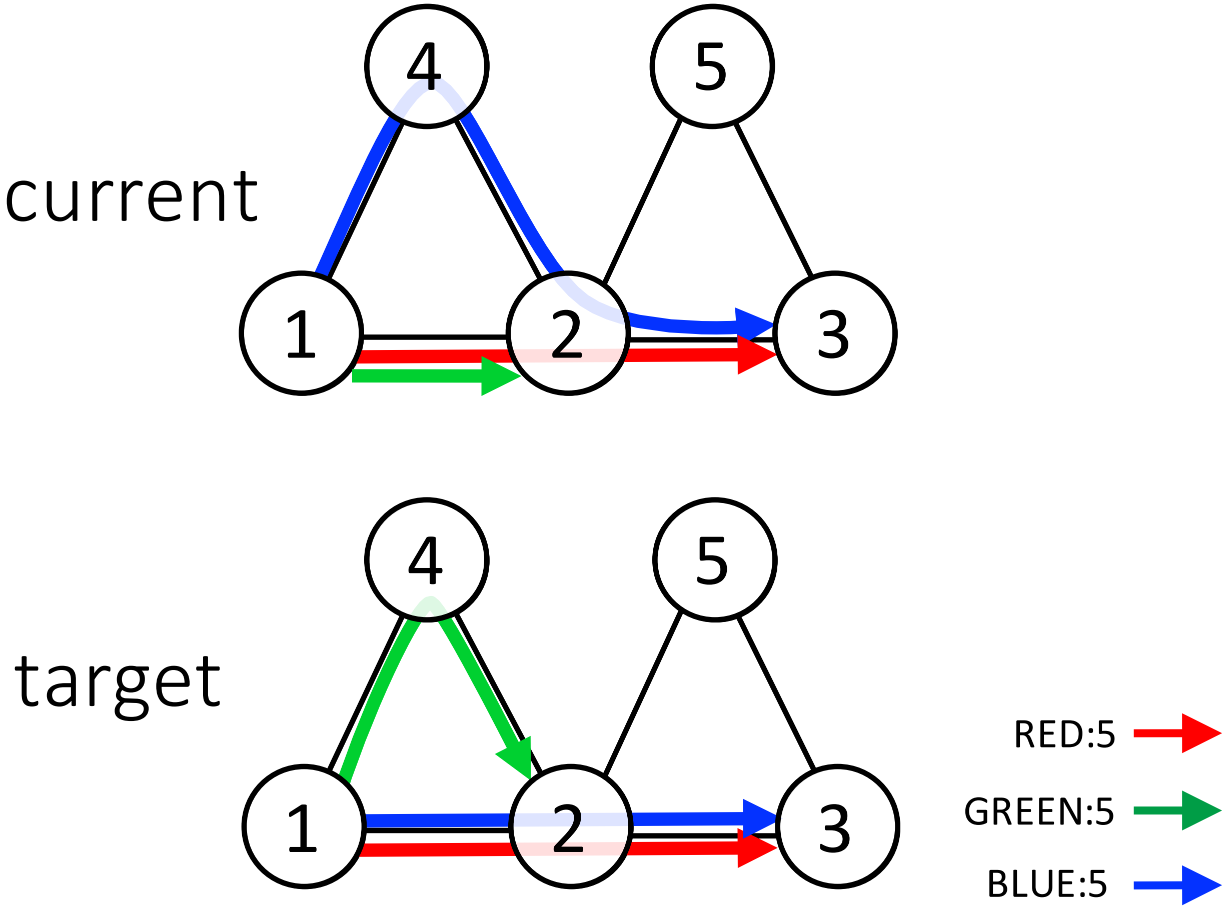
The dependency graph

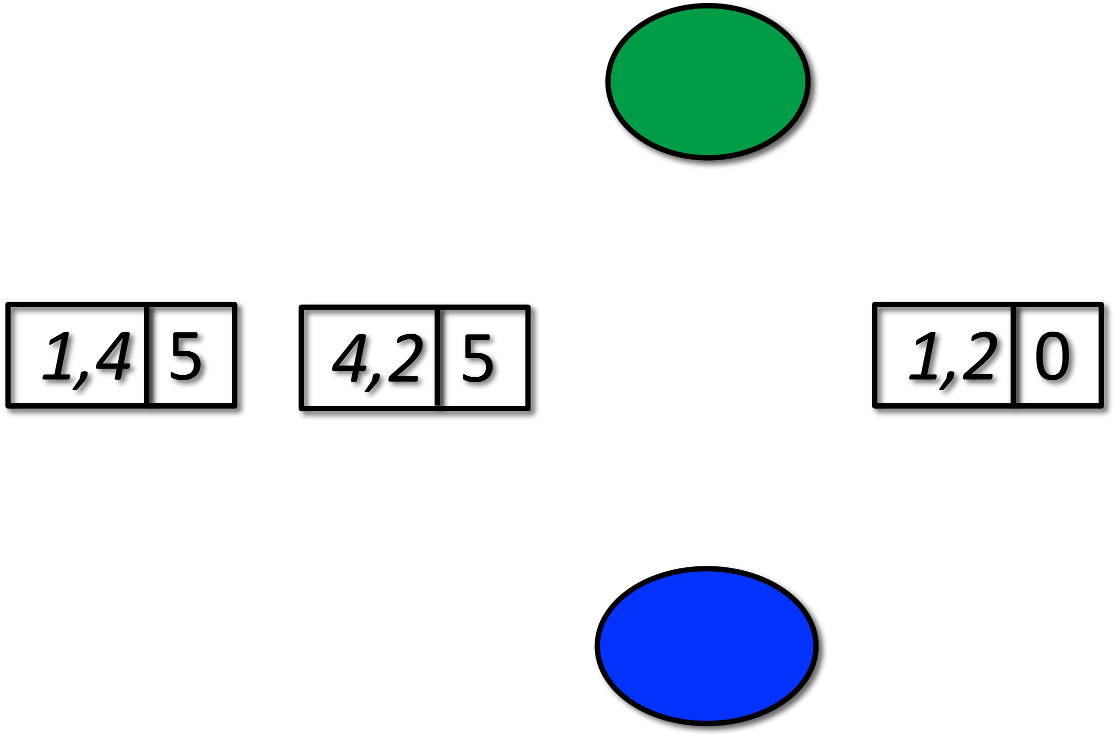
Central controller role: 1. Detects flow dependencies



The dependency graph

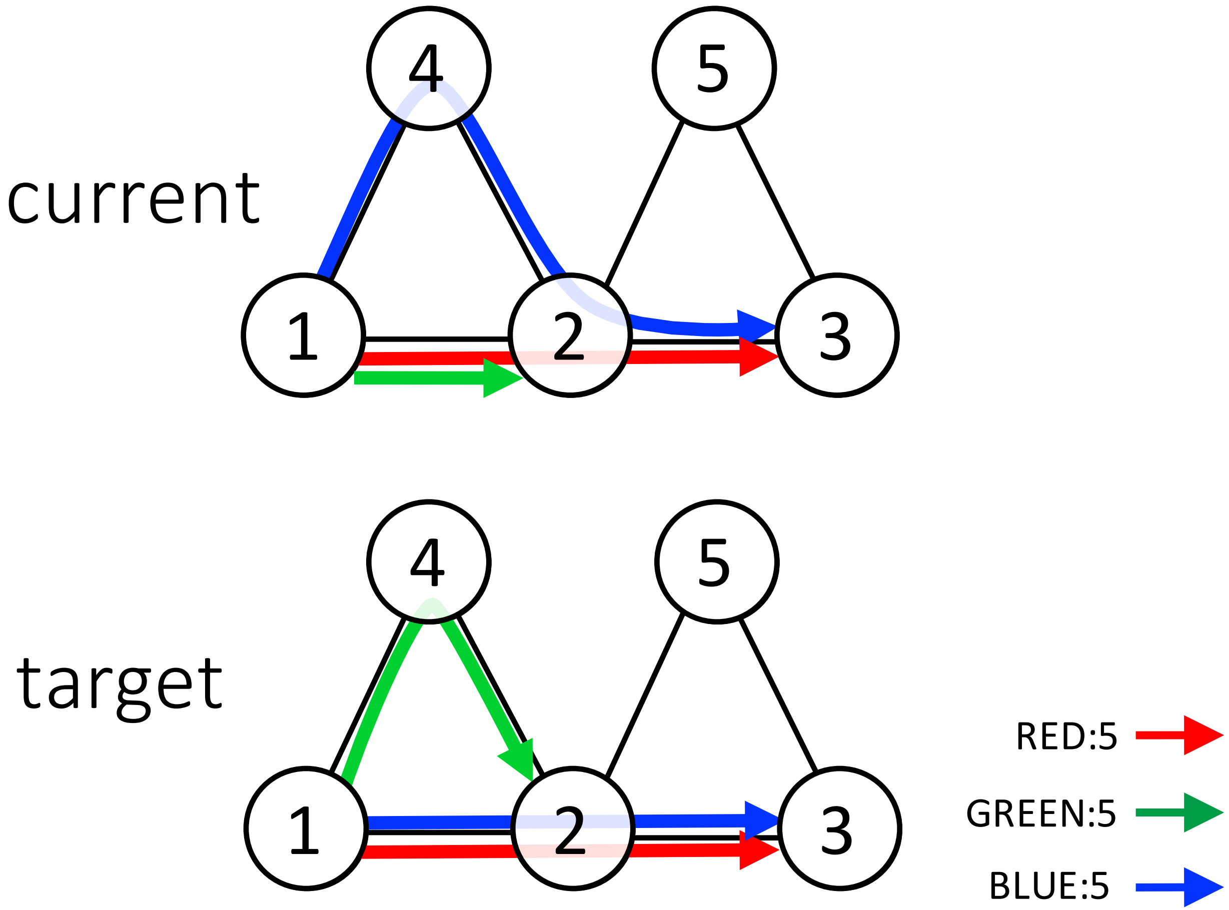
Central controller role: 1. Detects flow dependencies

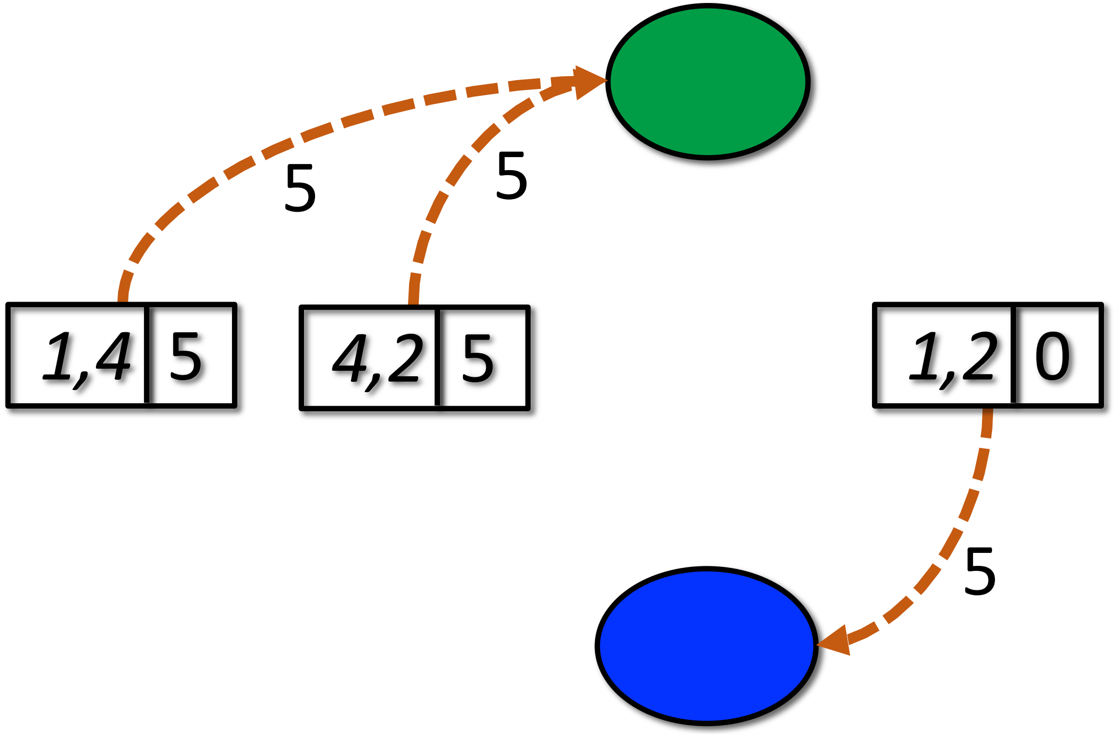




The dependency graph

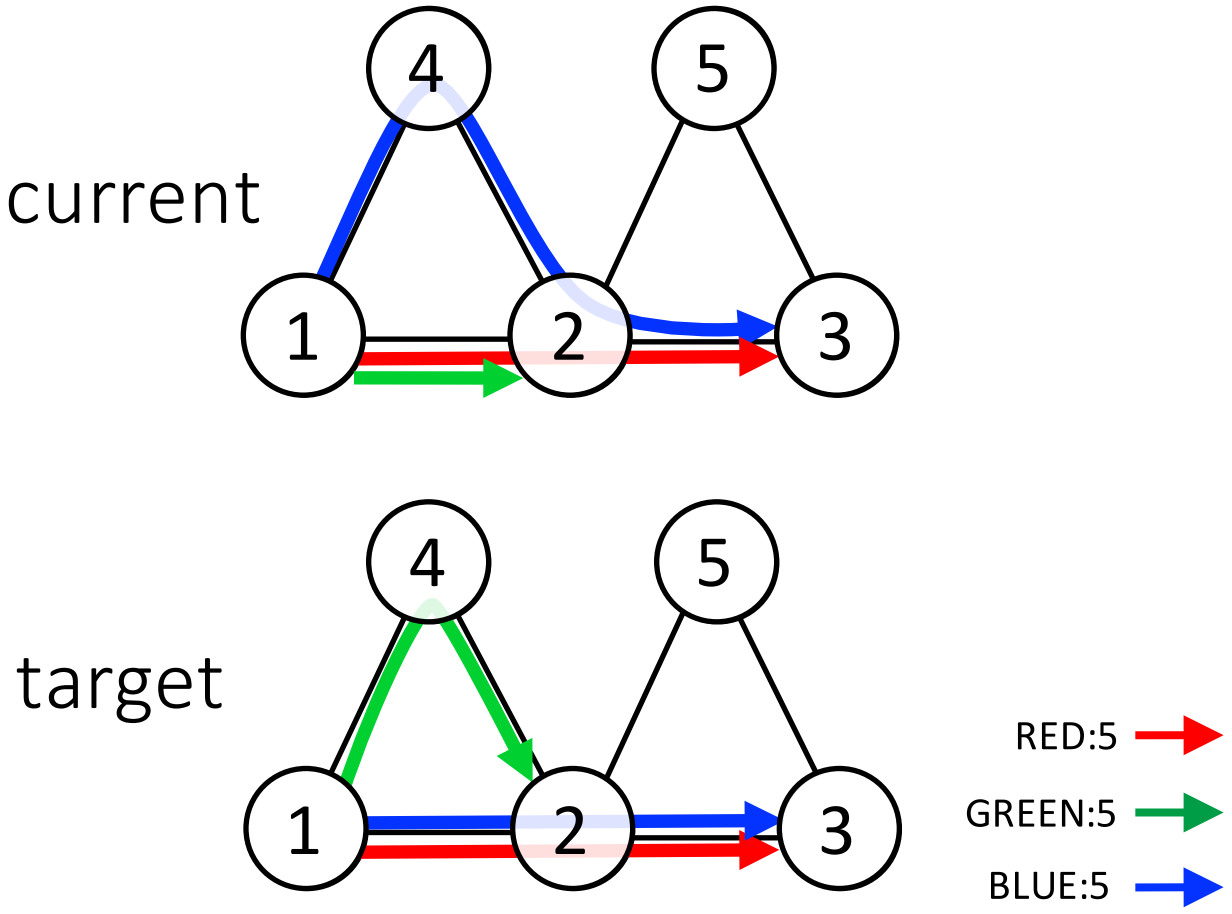
Central controller role: 1. Detects flow dependencies

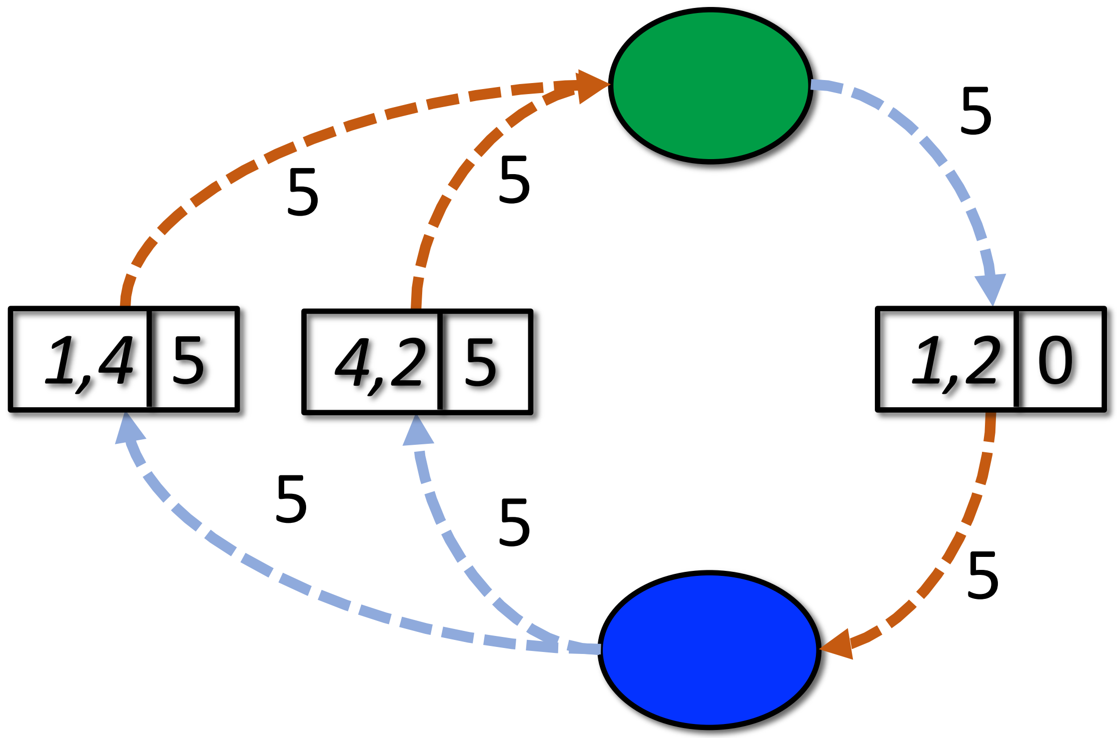




The dependency graph

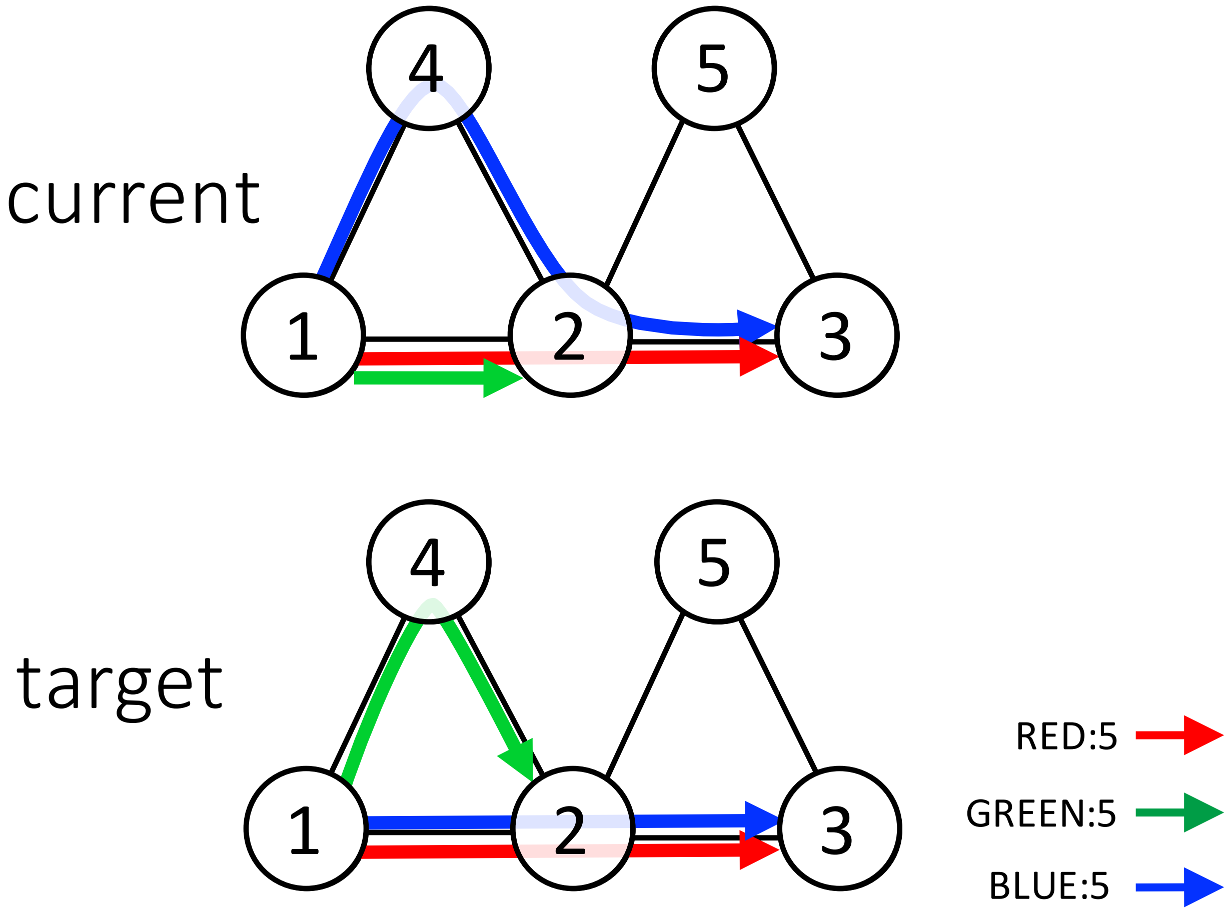
Central controller role: 1. Detects flow dependencies

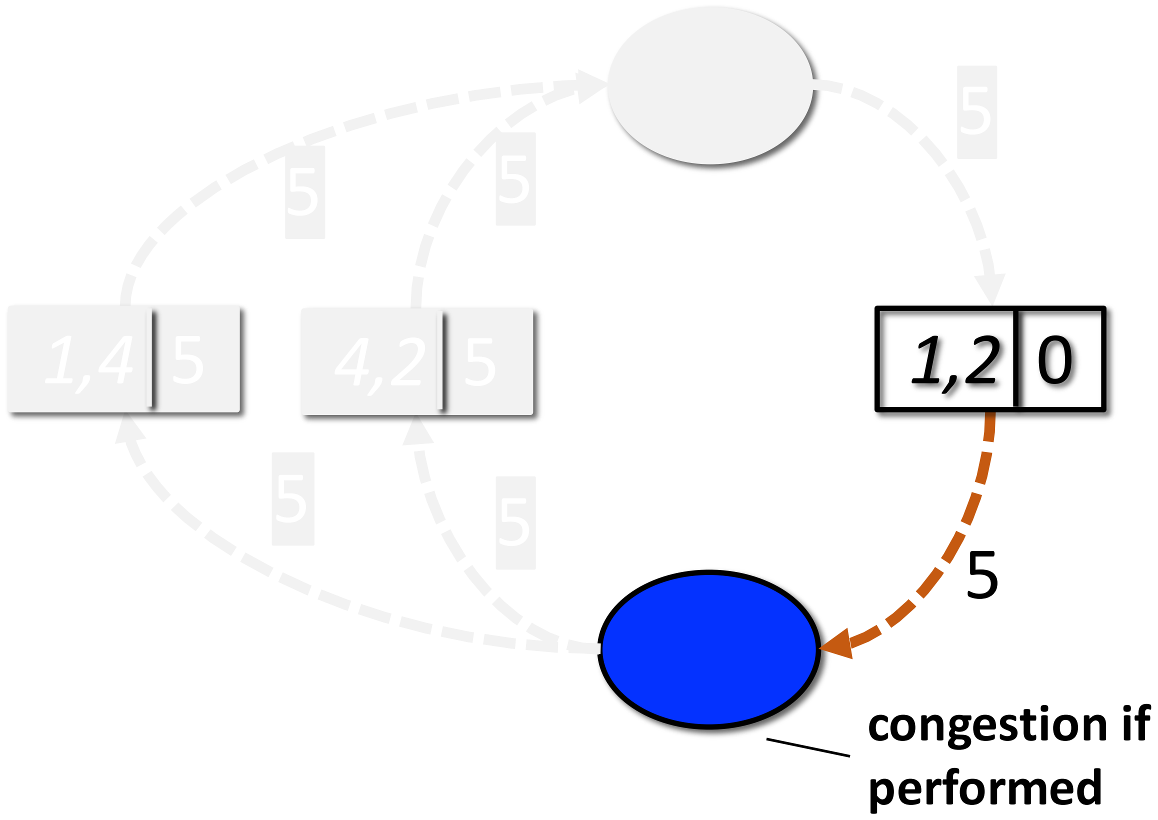




The dependency graph

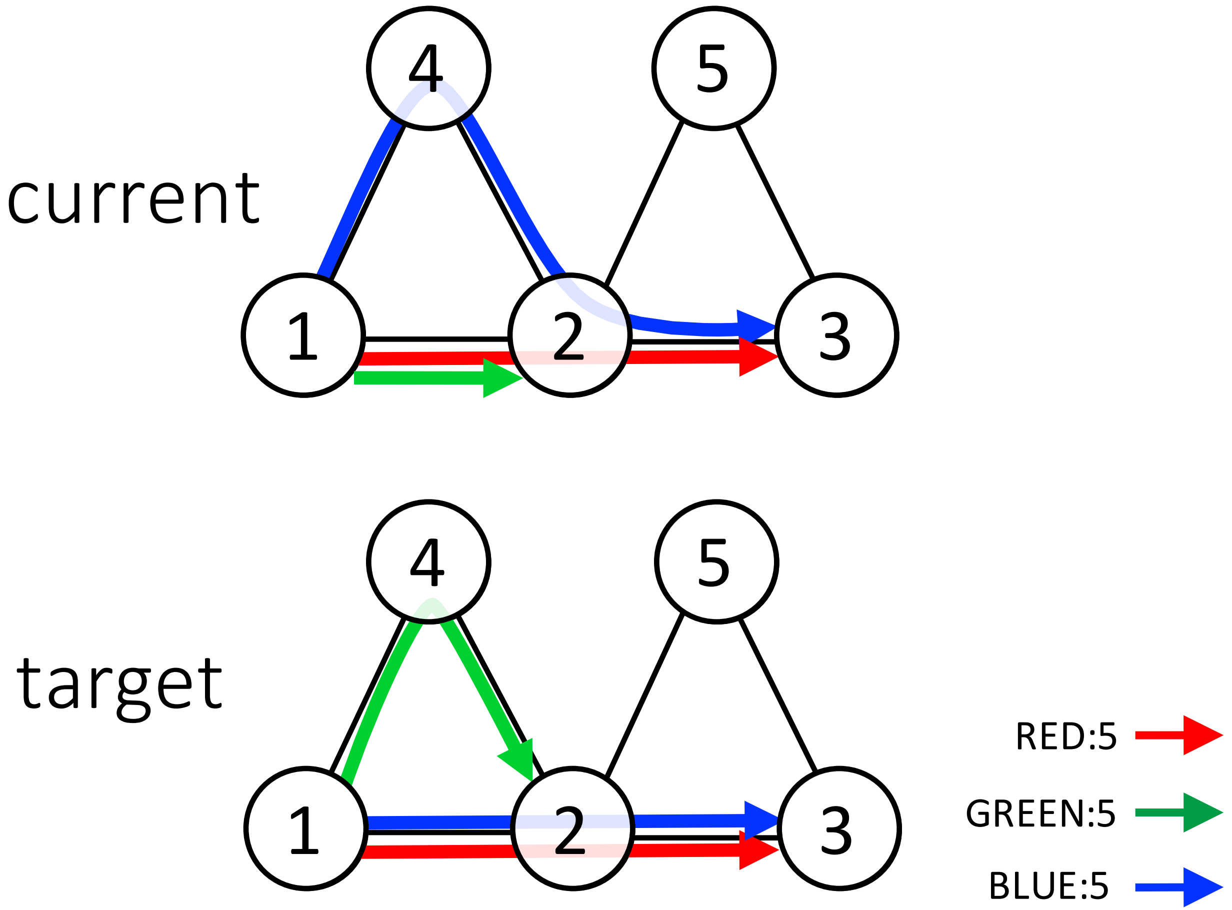
Central controller role: 1. Detects flow dependencies

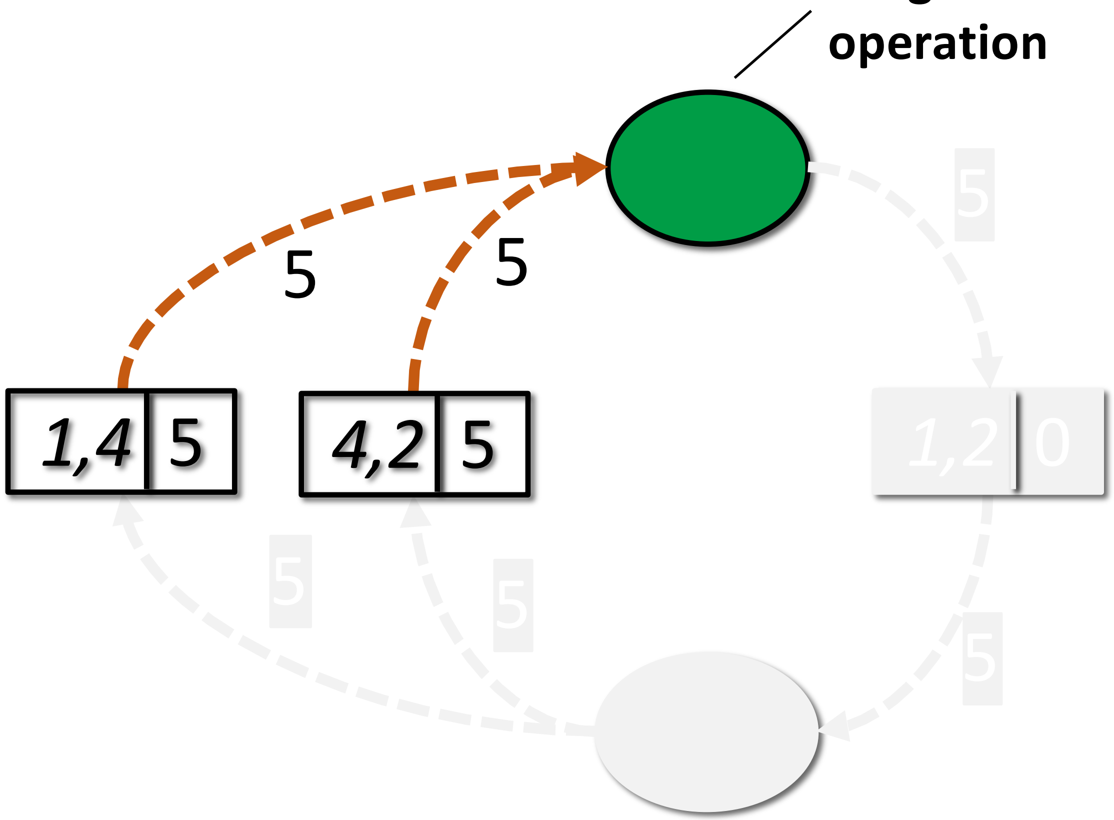




The dependency graph

Central controller role: 1. Detects flow dependencies





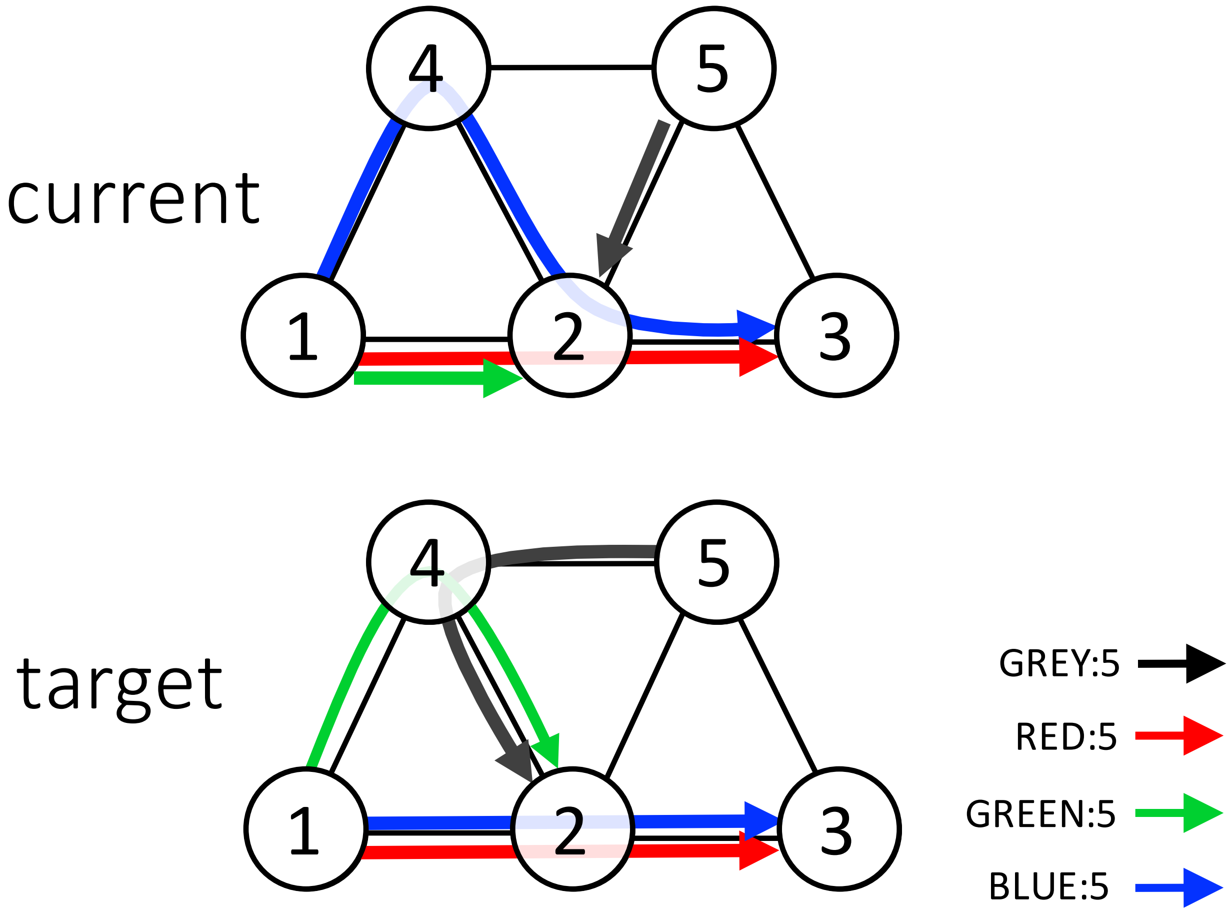
Ez-segway
the controller perspective

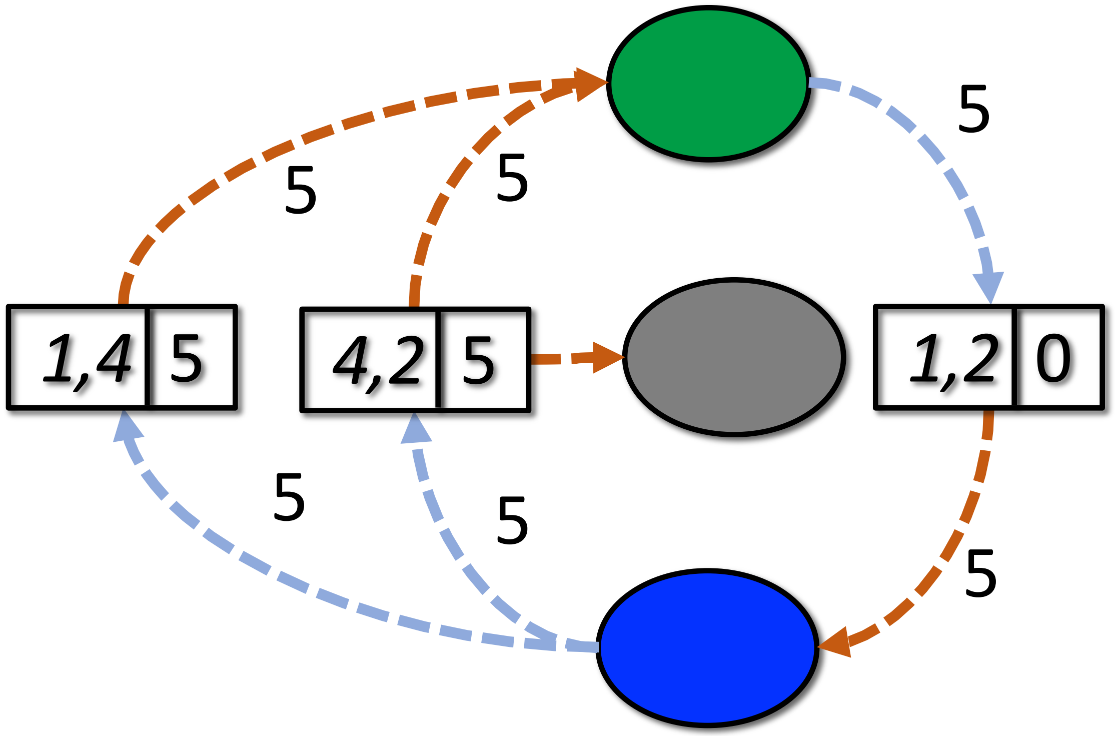
Central controller role:

1. detects flow dependencies
    - constructs a dependency graph
2. computes flows *partial* ordering
    - assign priorities to flows

Assignign flow priorities

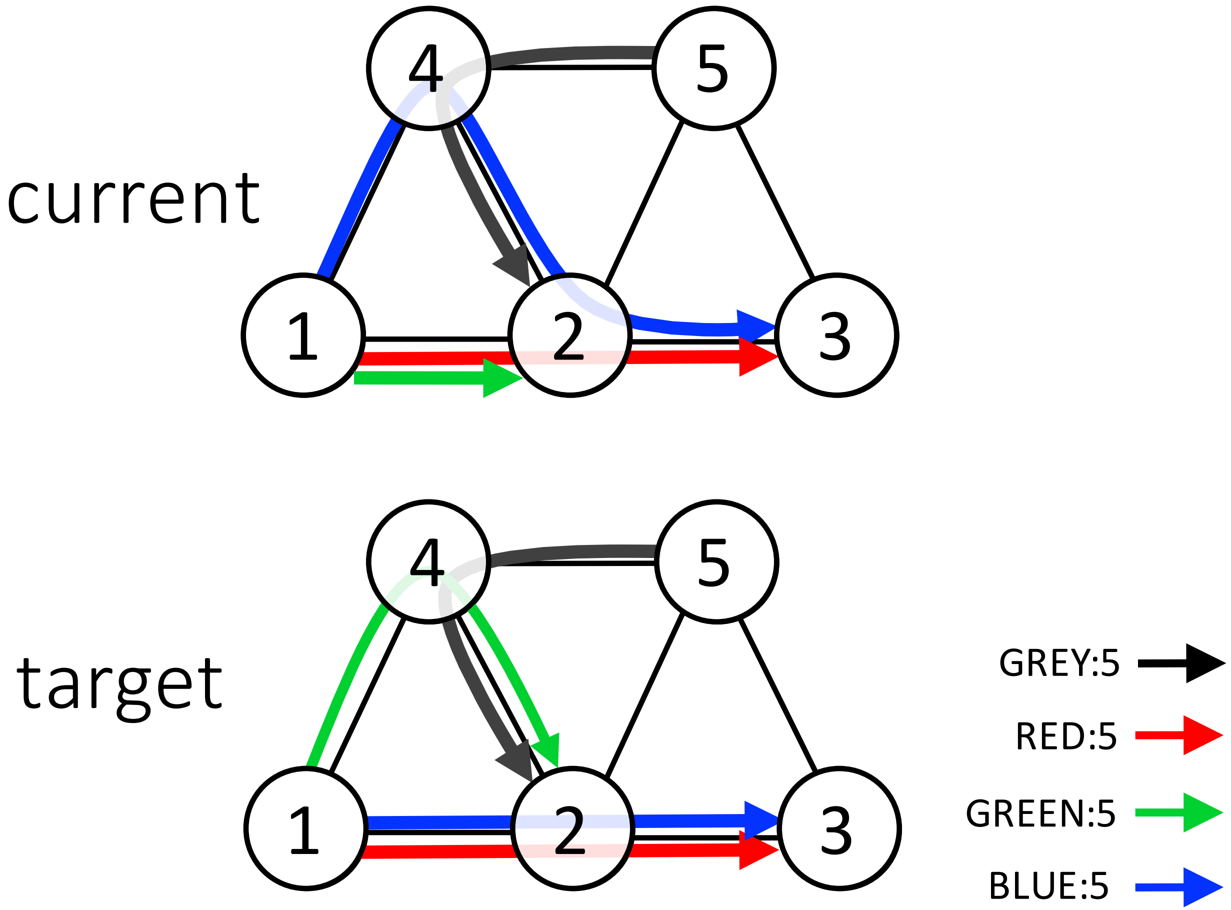
Central controller role: 2. computes flow partial ordering

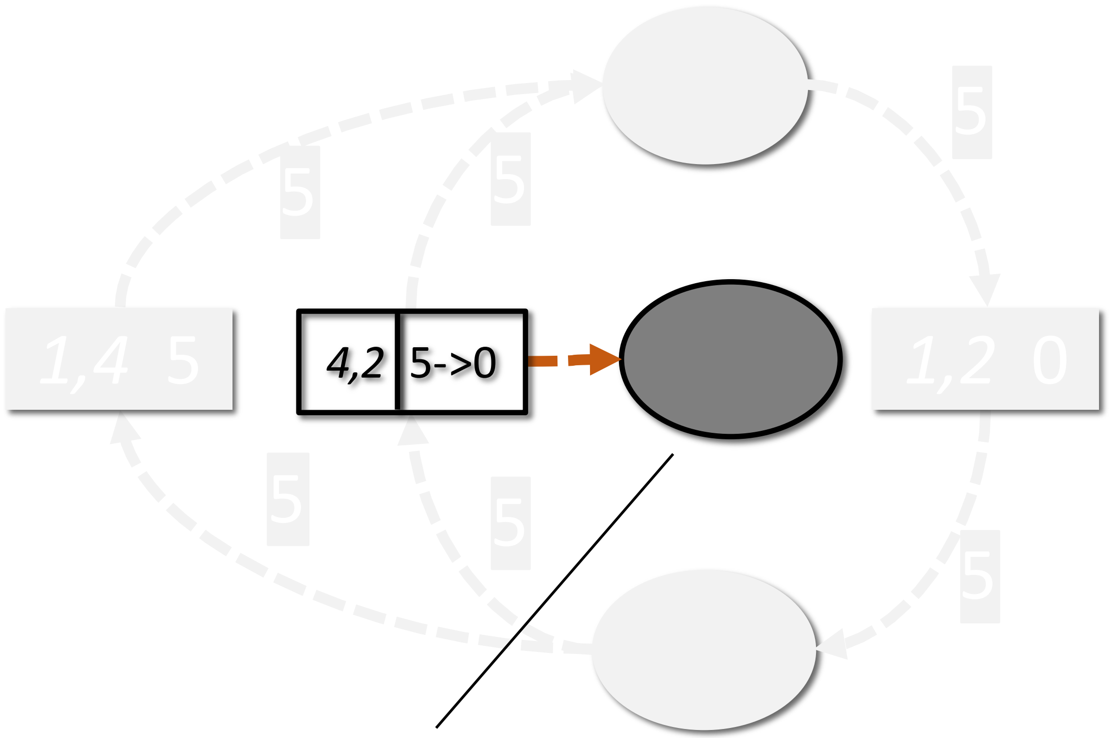




Assignign flow priorities

Central controller role: 2. computes flow partial ordering

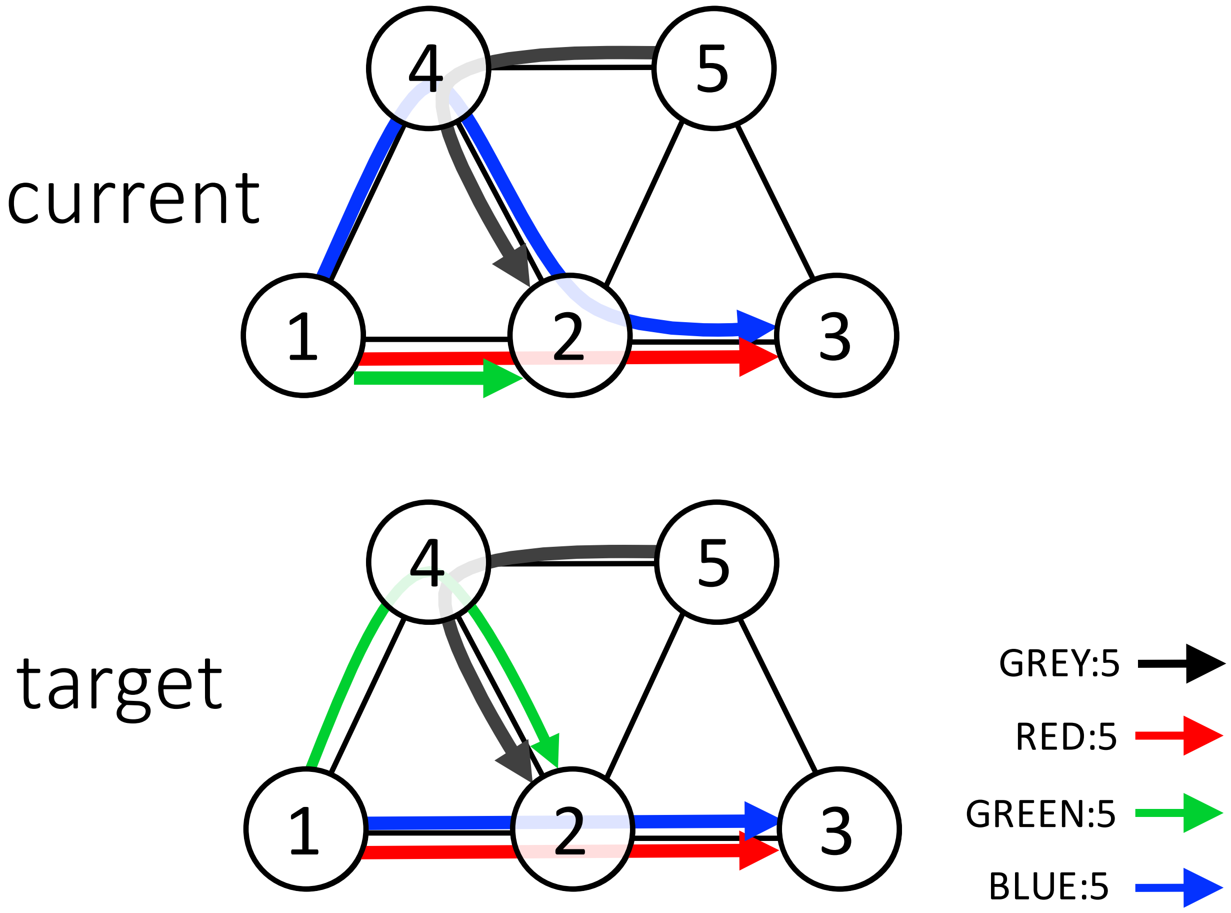


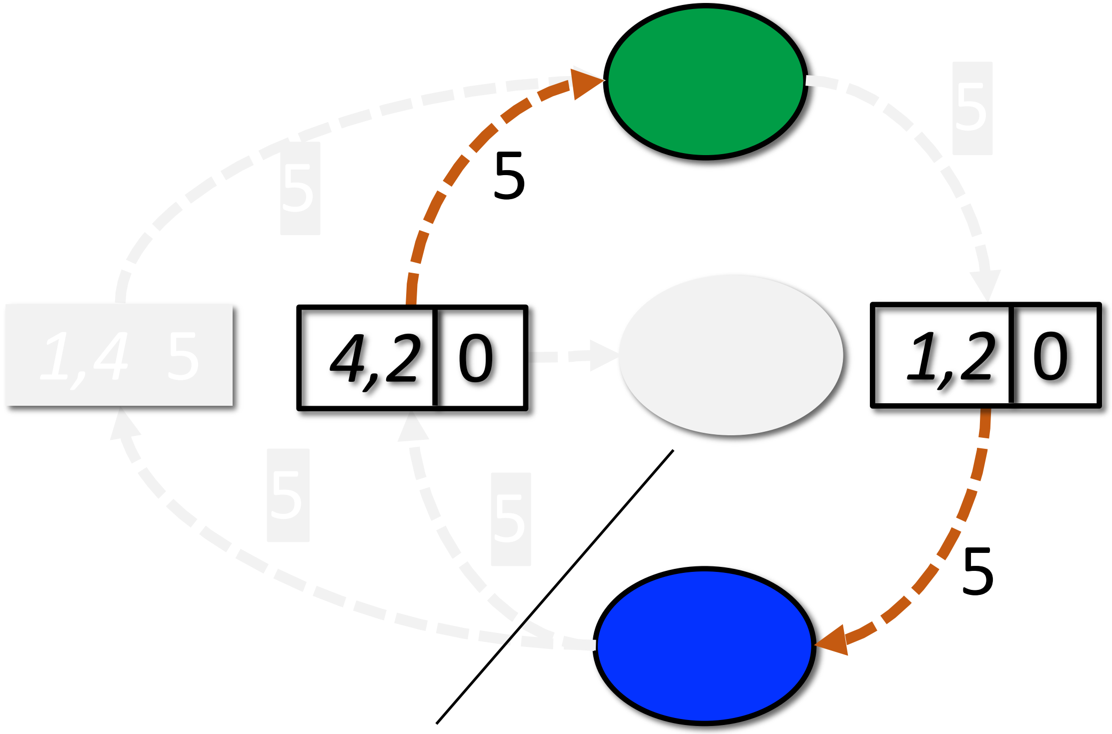


Update GREY first…

Assignign priorities

Central controller role: 2. computes flow partial ordering

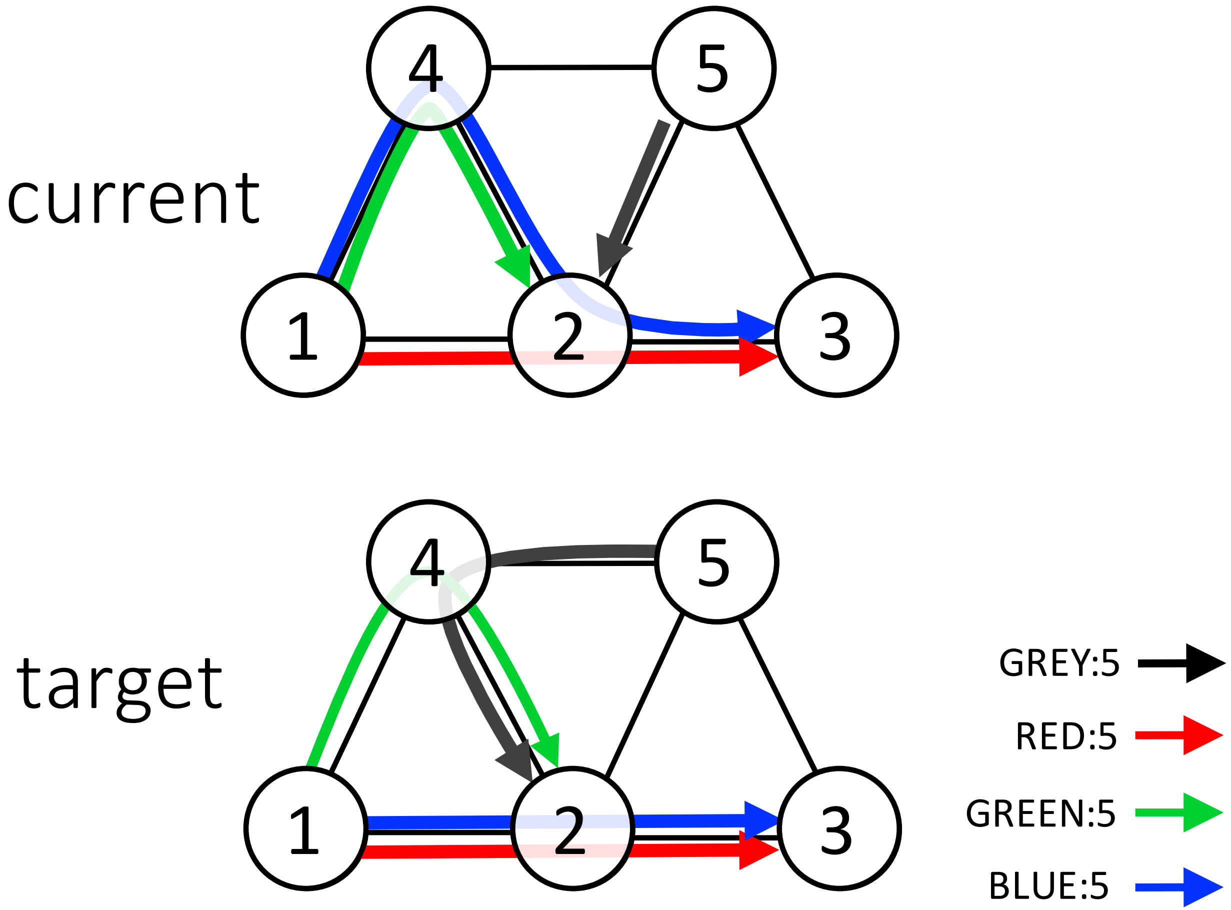


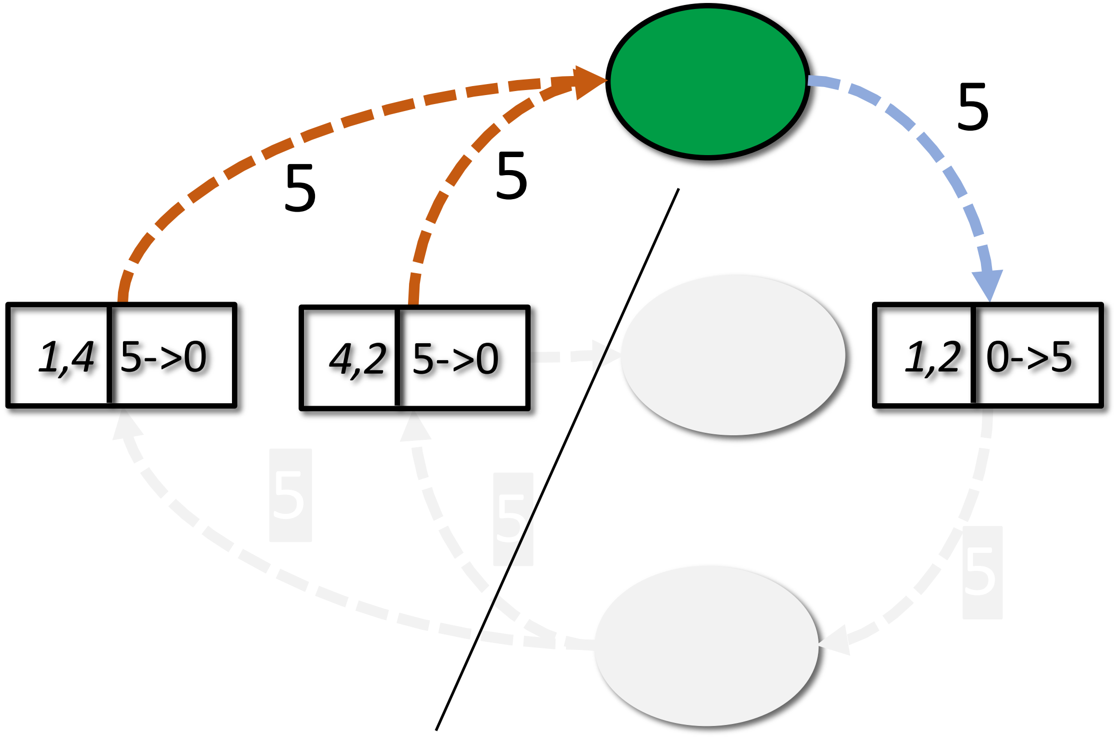


Update GREY first…
…leads to link congestion

Assignign priorities

Central controller role: 2. computes flow partial ordering

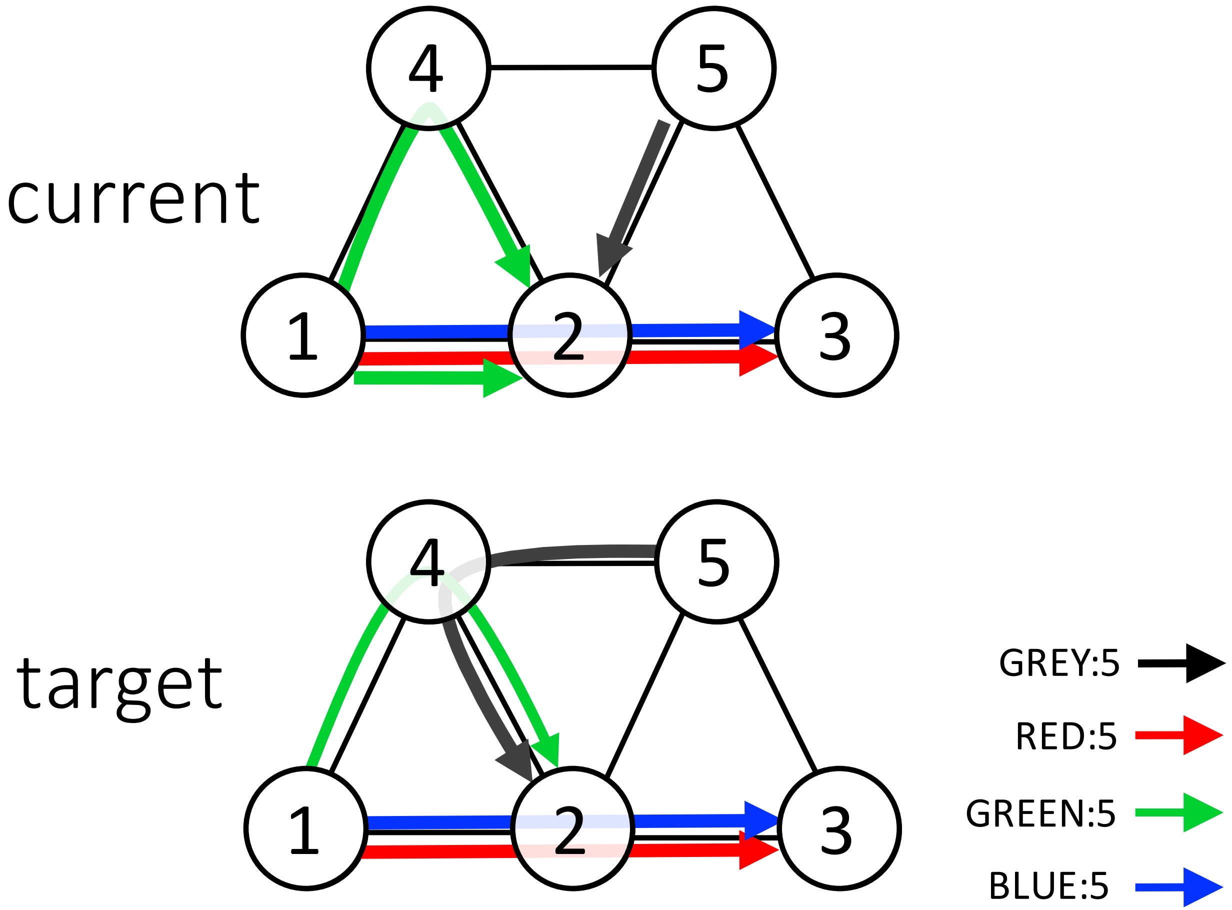


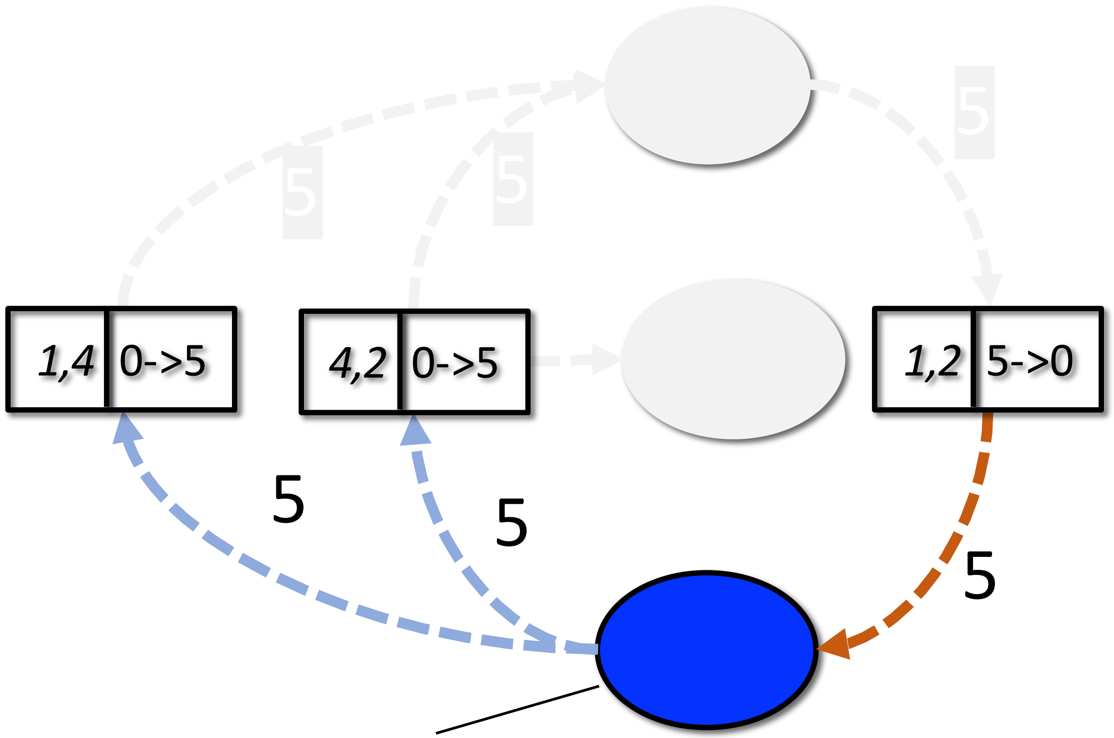


Update GREEN first…

Assignign priorities

Central controller role: 2. computes flow partial ordering



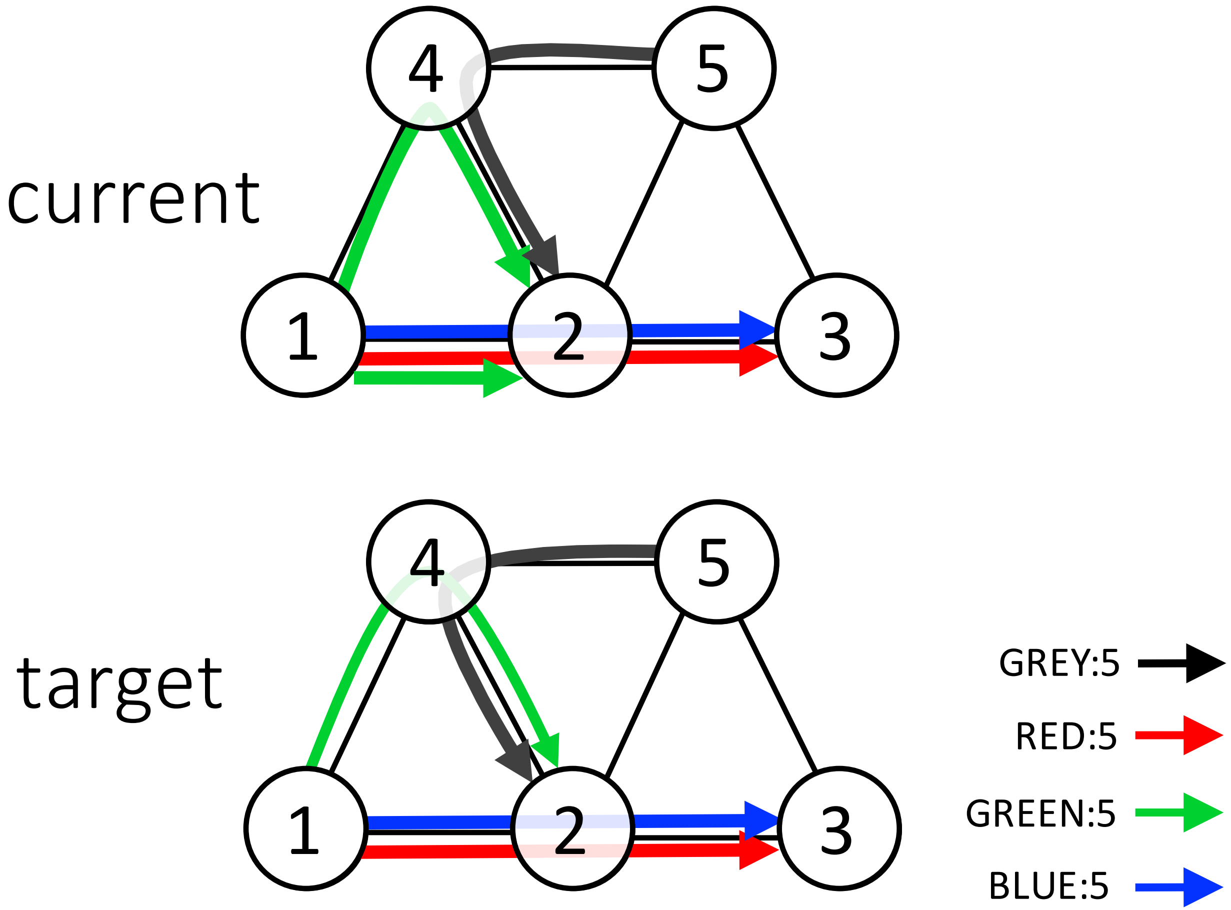


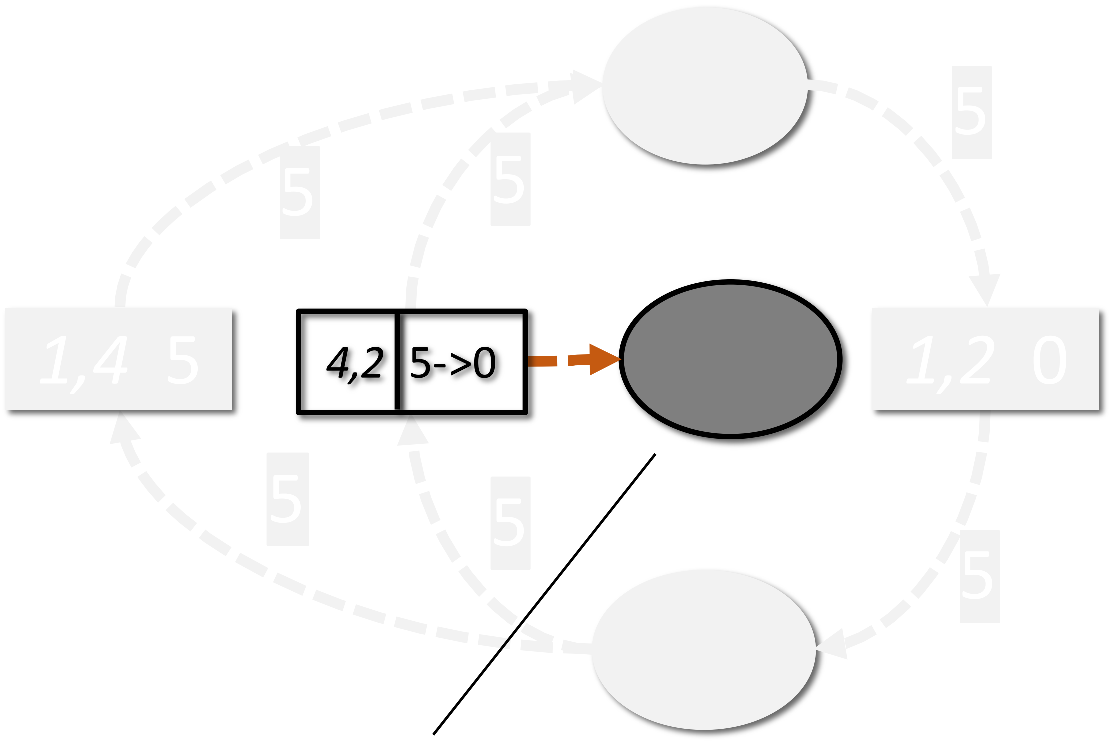
Update GREEN first…

…BLUE second…

Assignign priorities

Central controller role: 2. computes flow partial ordering

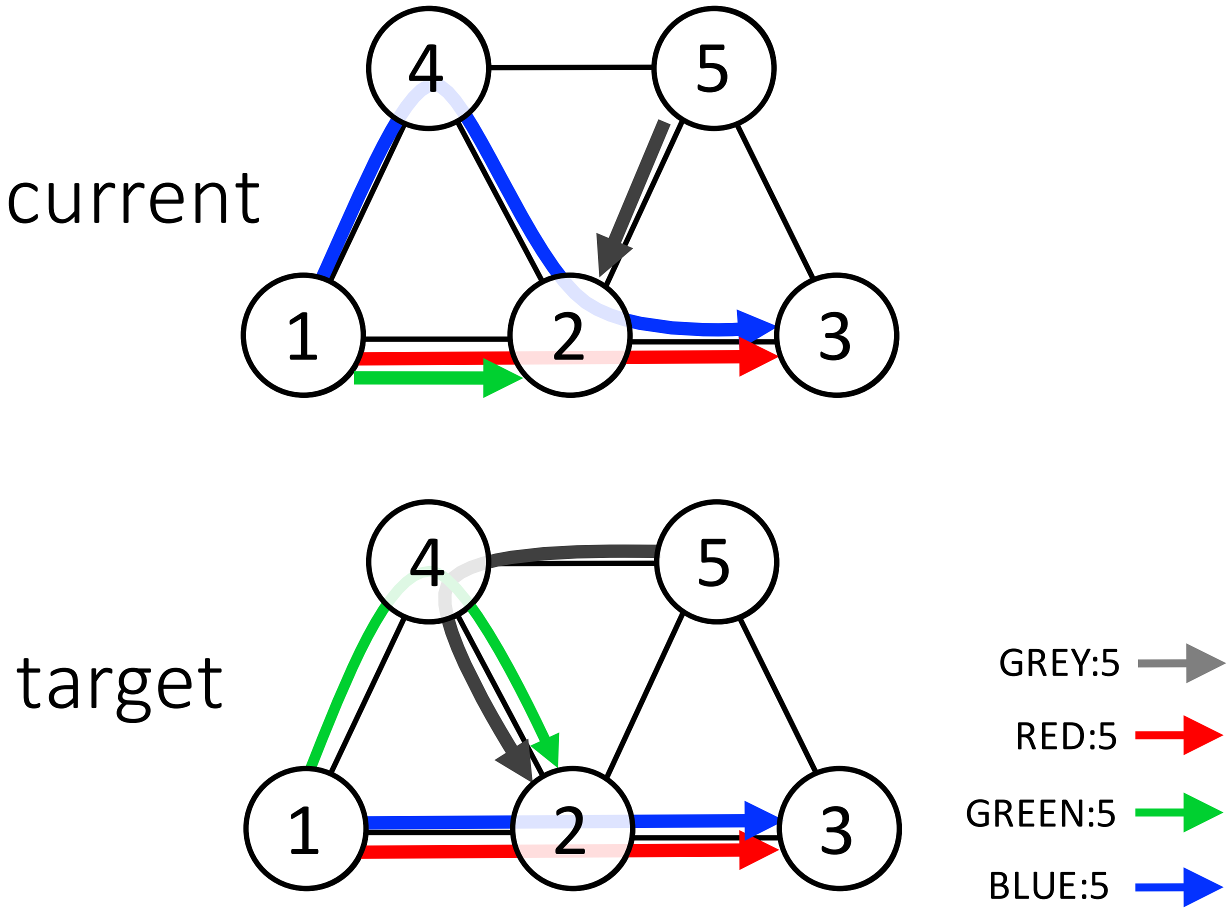


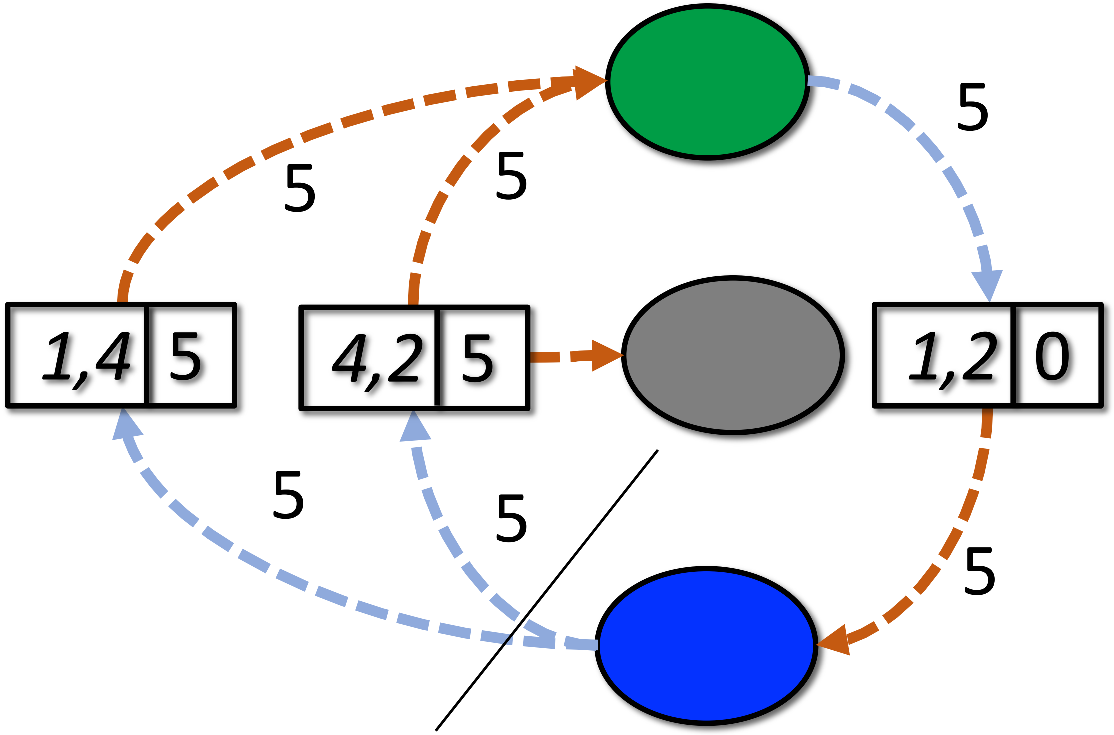


Update GREEN first…
…BLUE second… then GREY

Assignign flow priorities

Central controller role: 2. computes flow partial ordering





GREY should be given lower priority

Ez-segway: the controller perspective

Central controller role:

1. detects flow dependencies
   - constructs a dependency graph
2. computes flows *partial* ordering
   - assign priorities to flows
3. sends scheduling to the switches
   - only once

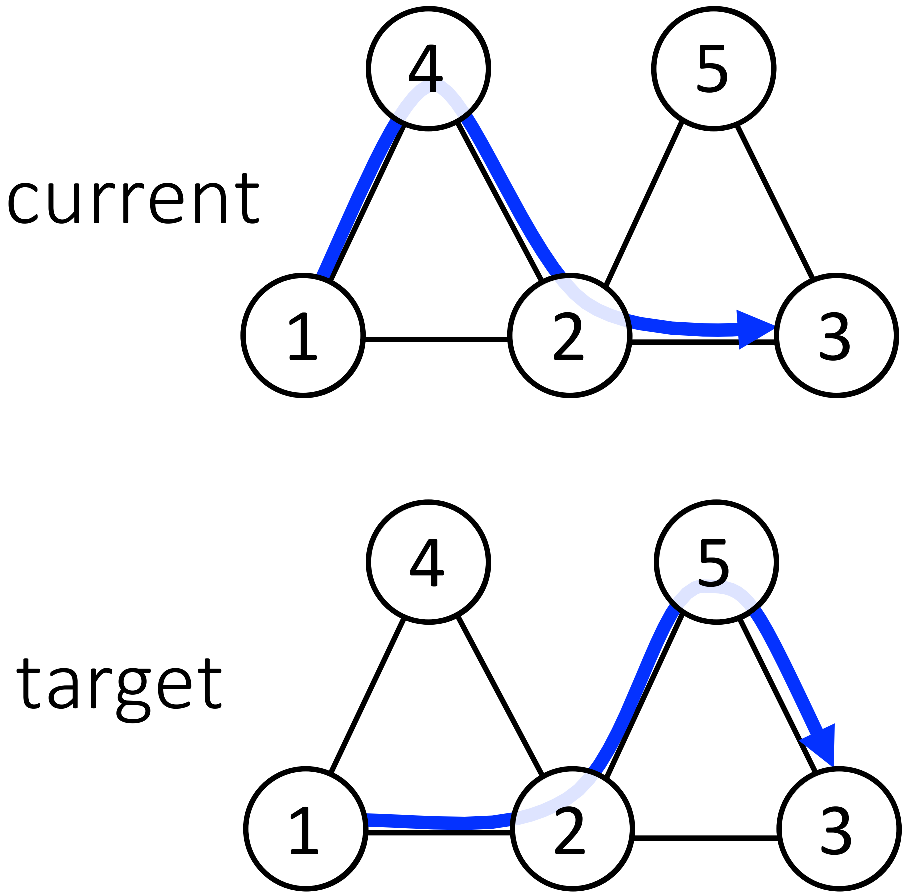
Ez-segway:
the switches perspective

Switches role:

- coordinate with neighbors

In-band message passing

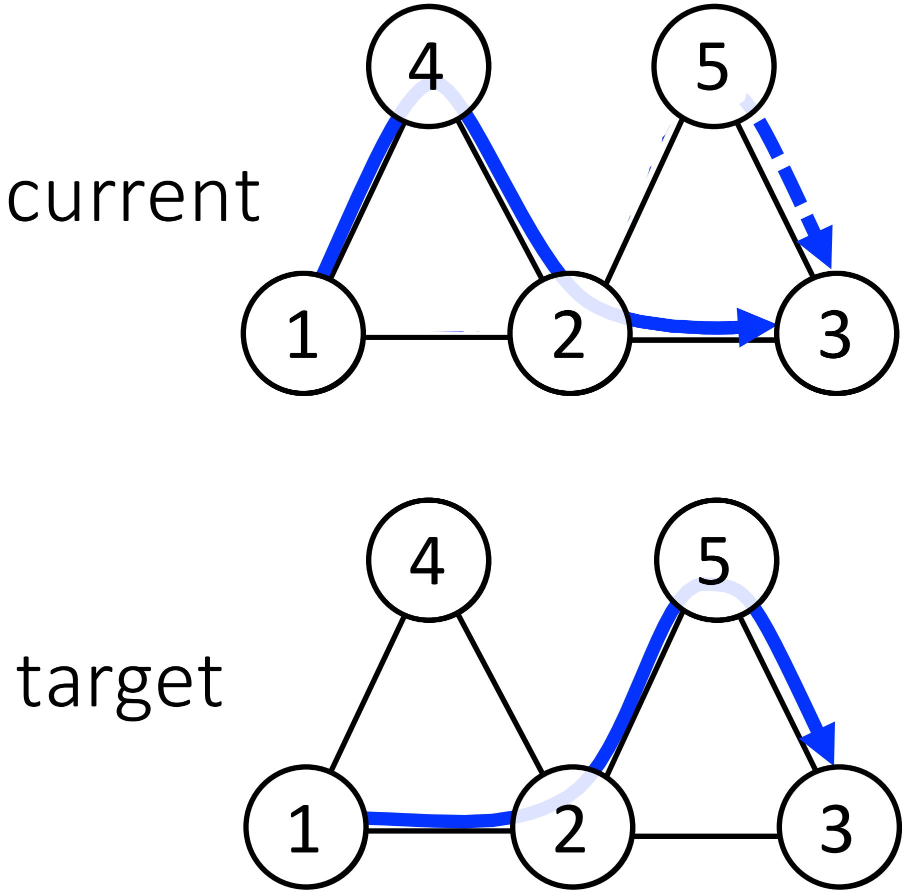
Switch role: **coordinate with neighbors**



New path installed upwards

In-band message passing

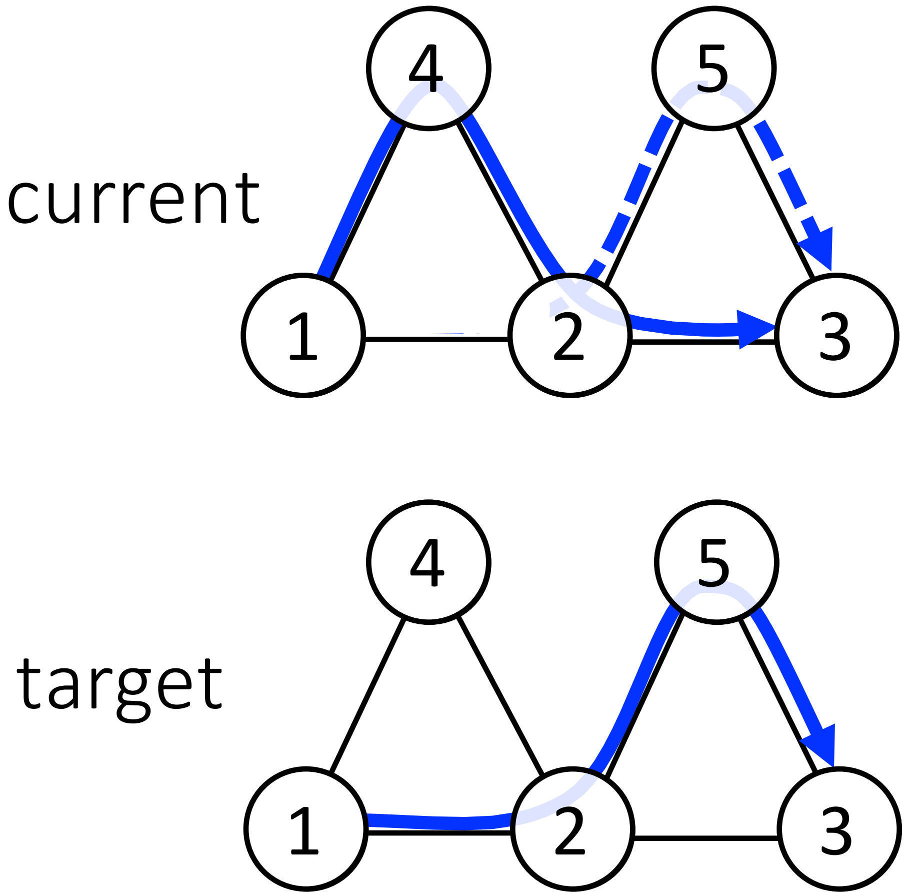
Switch role: coordinate with neighbors



New path installed upwards

In-band message passing

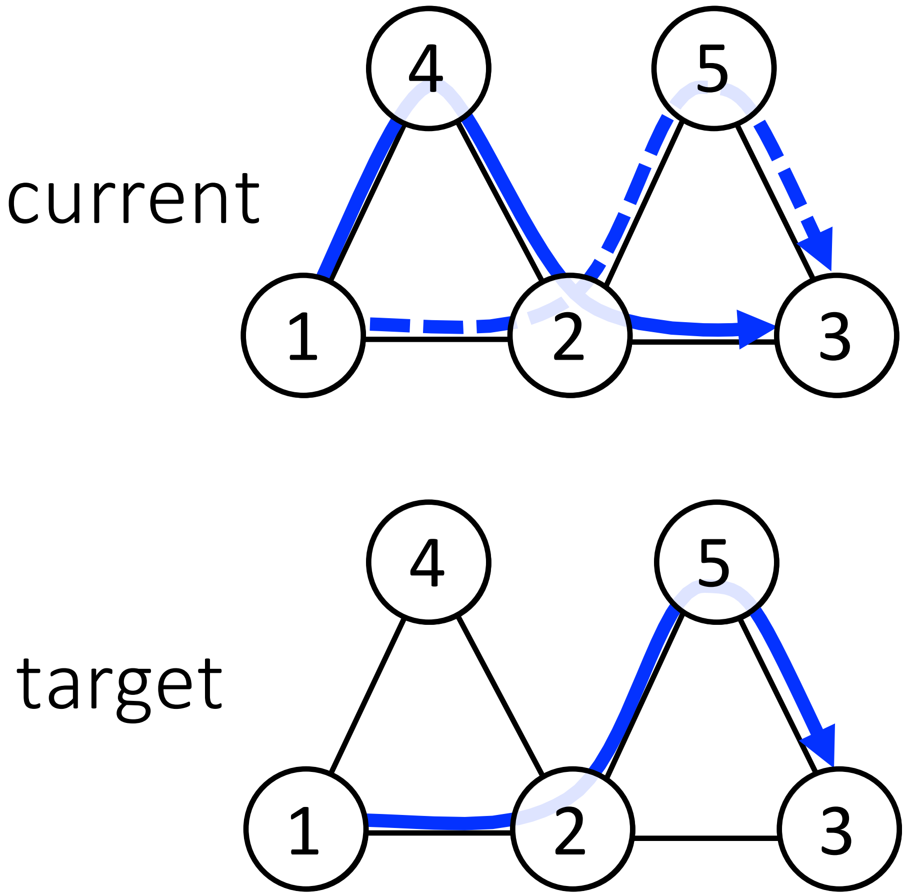
Switch role: coordinate with neighbors



New path installed upwards

In-band message passing

Switch role: coordinate with neighbors

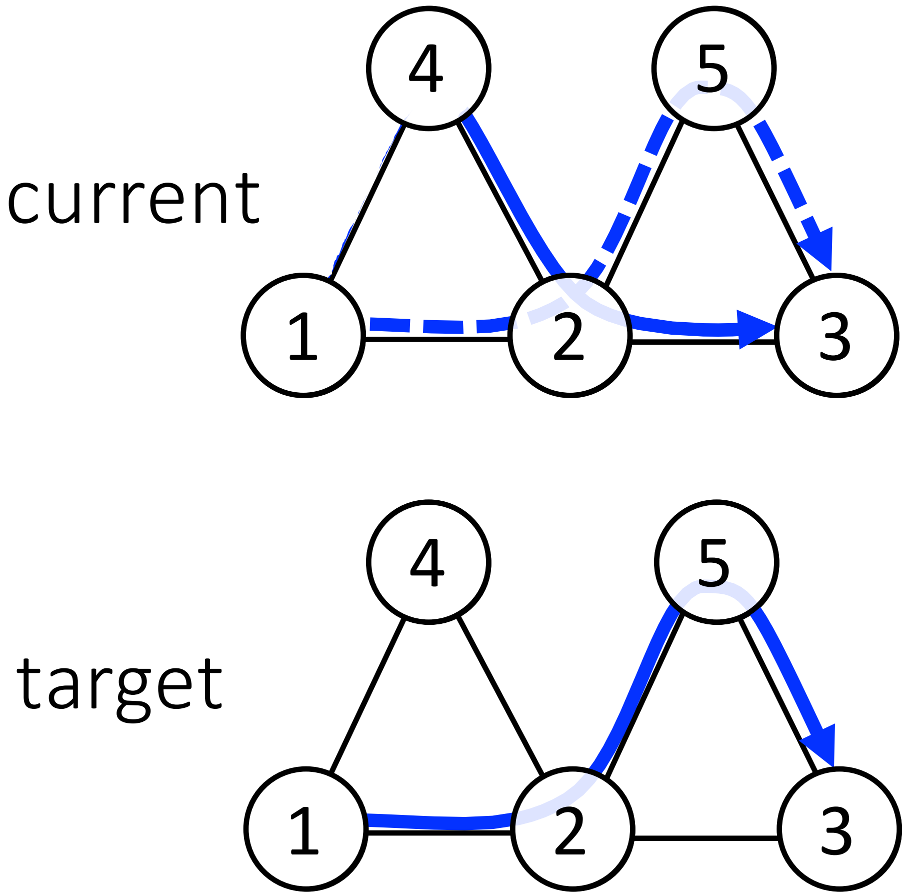


New path installed upwards

Packets are routed on the new path

In-band message passing

Switch role: coordinate with neighbors



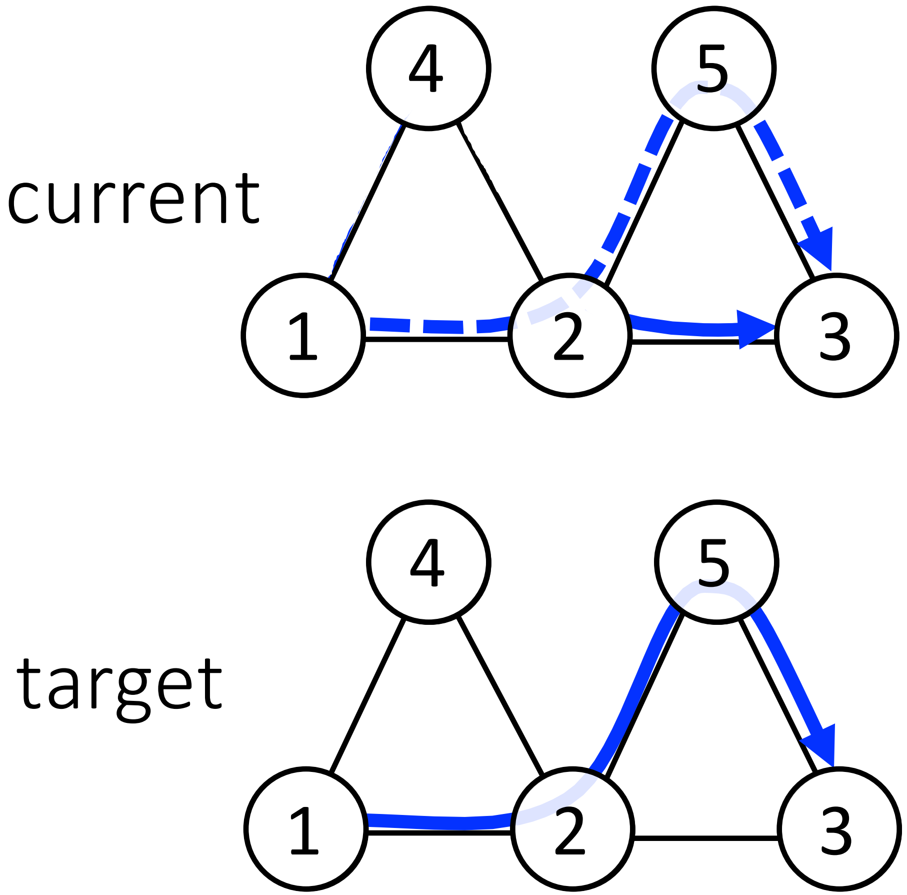
New path installed upwards

Packets are routed on the new path

Old path removed downwards

In-band message passing

Switch role: coordinate with neighbors



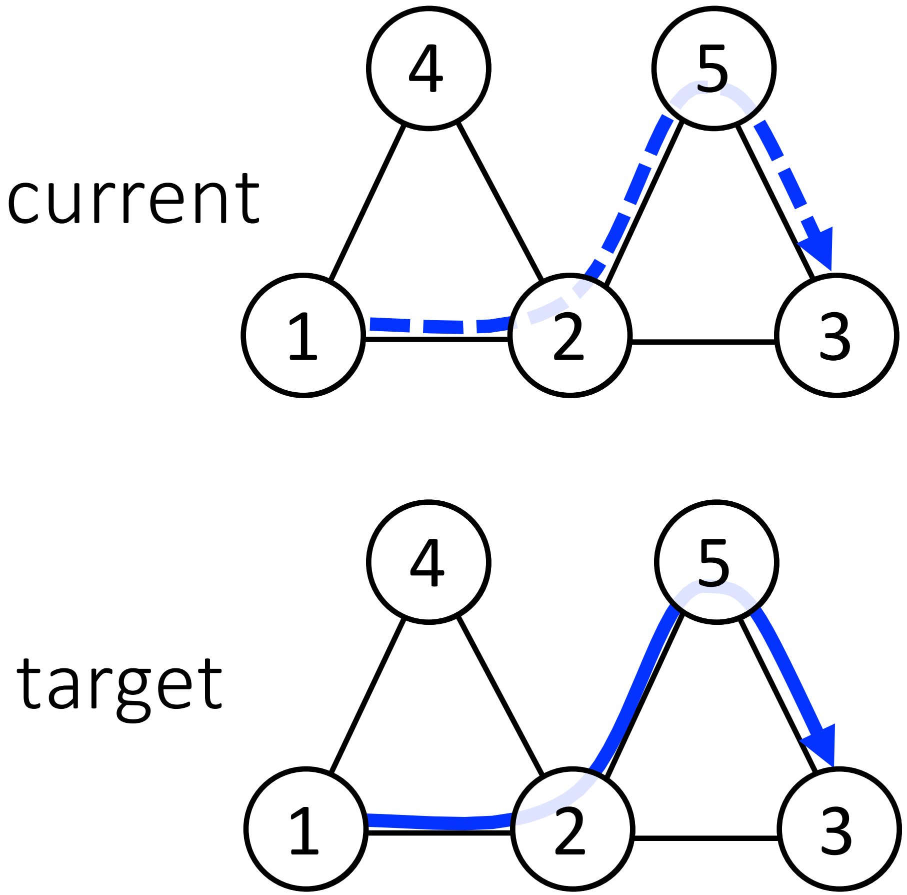
New path installed upwards

Packets are routed on the new path

Old path removed downwards

In-band message passing

Switch role: coordinate with neighbors



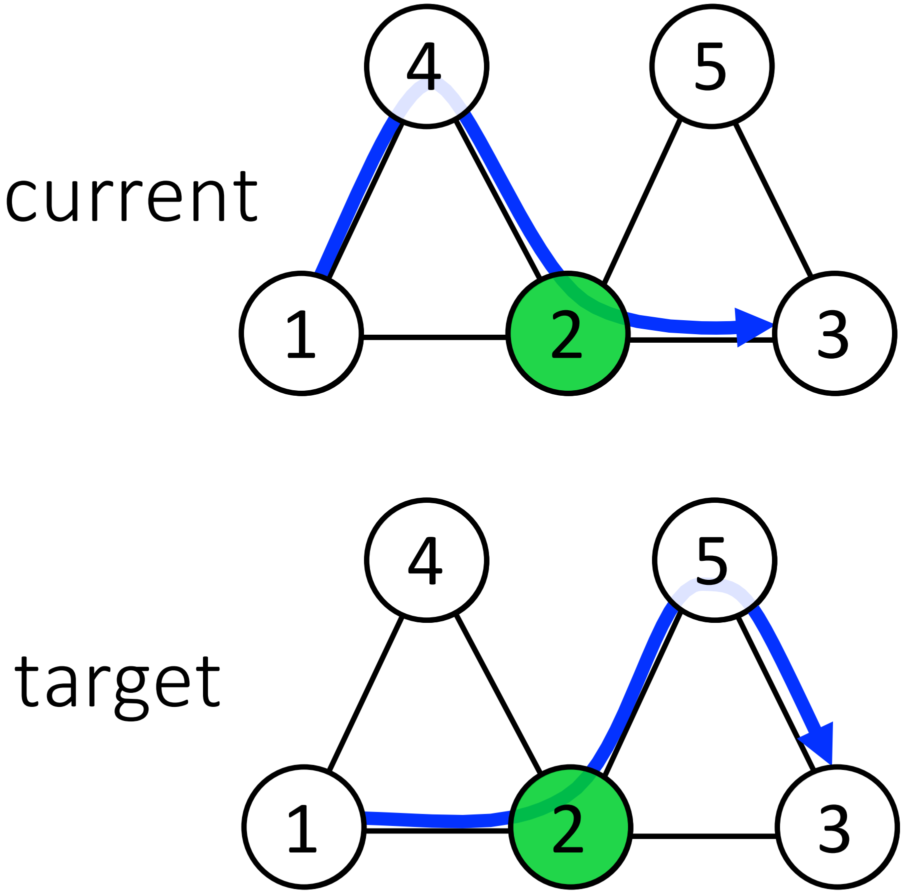
New path installed upwards

Packets are routed on the new path

Old path removed downwards

Update speed up via segmentation

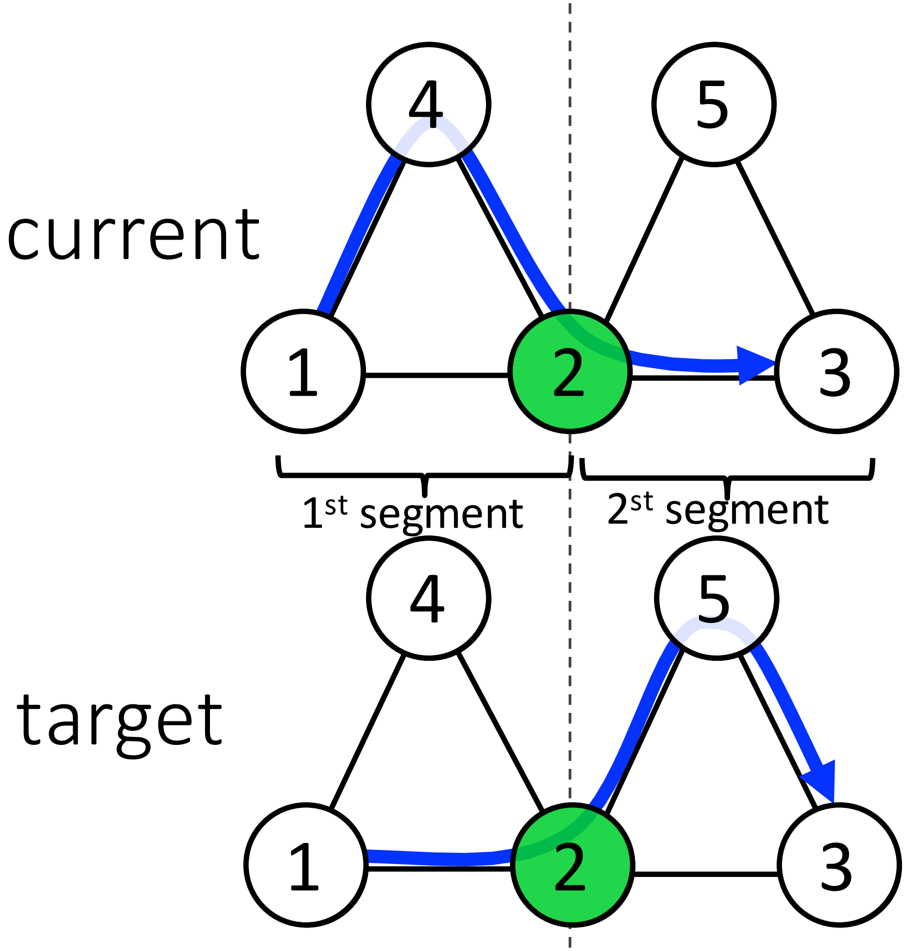
Switch role: coordinate with neighbors



Switch 2 is traversed in both paths

Update speed up via segmentation

Switch role: coordinate with neighbors

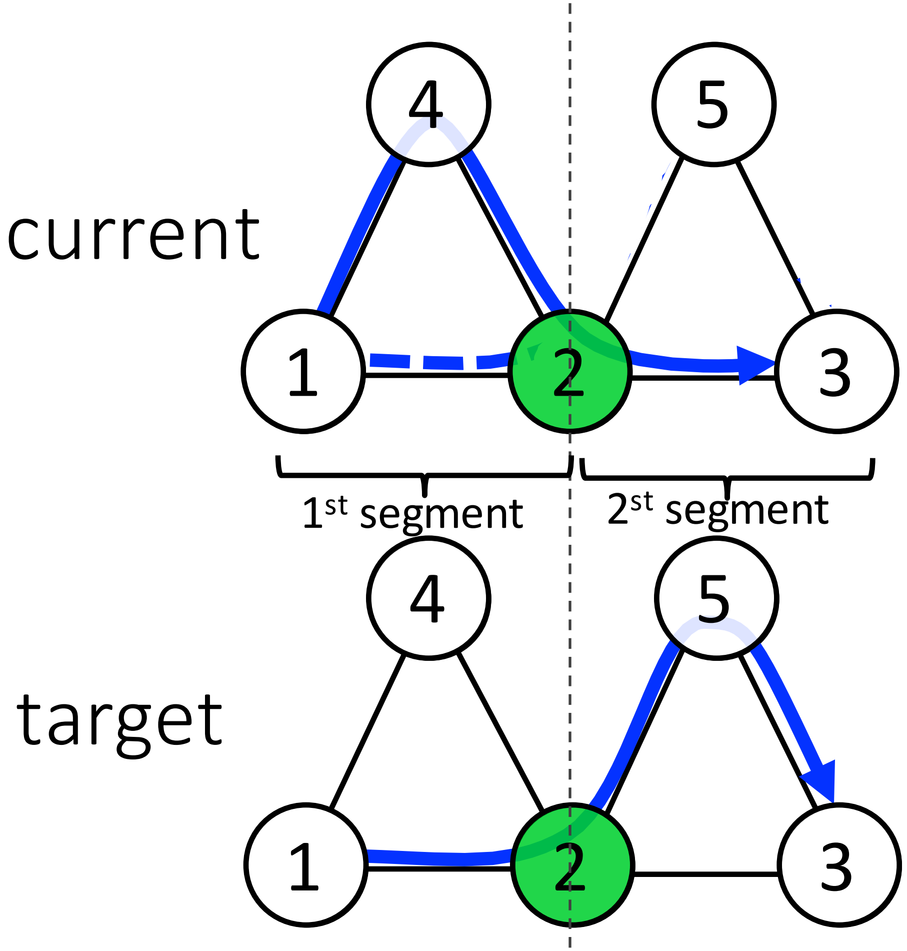


Switch 2 is traversed in both paths

Parallelizable segment updates

Update speed up via segmentation

Switch role: coordinate with neighbors

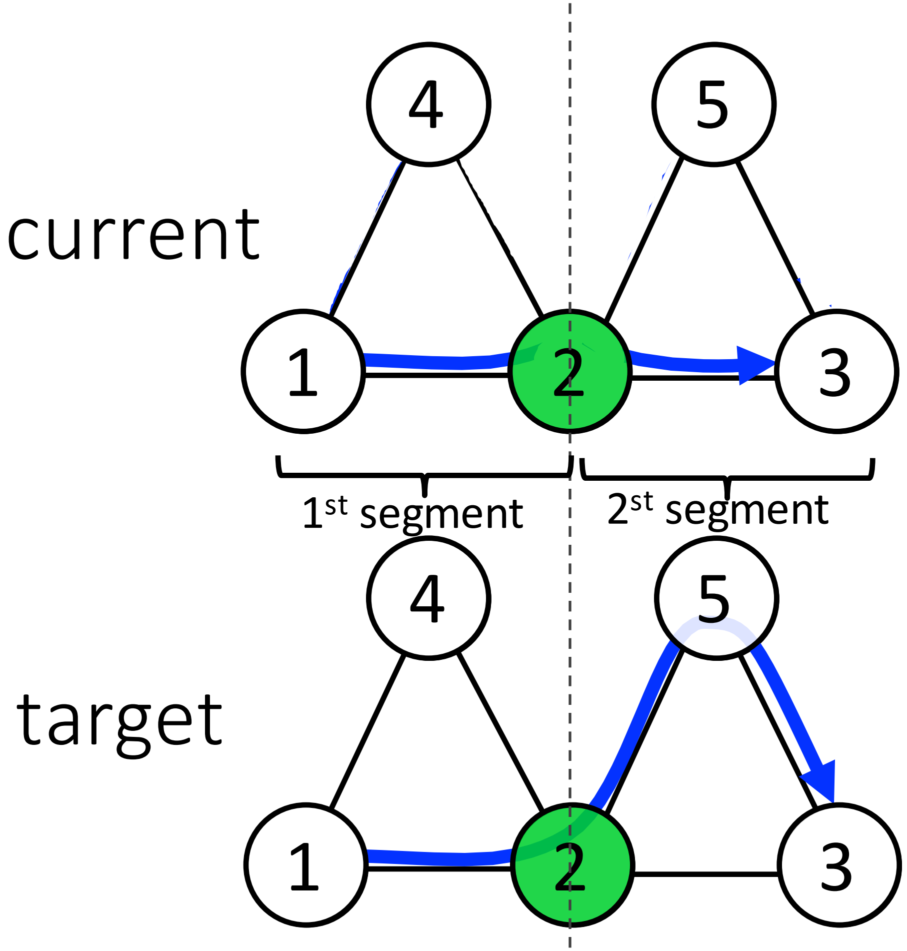


Switch 2 is traversed in both paths

Parallelizable segment updates

Update speed up via segmentation

Switch role: coordinate with neighbors

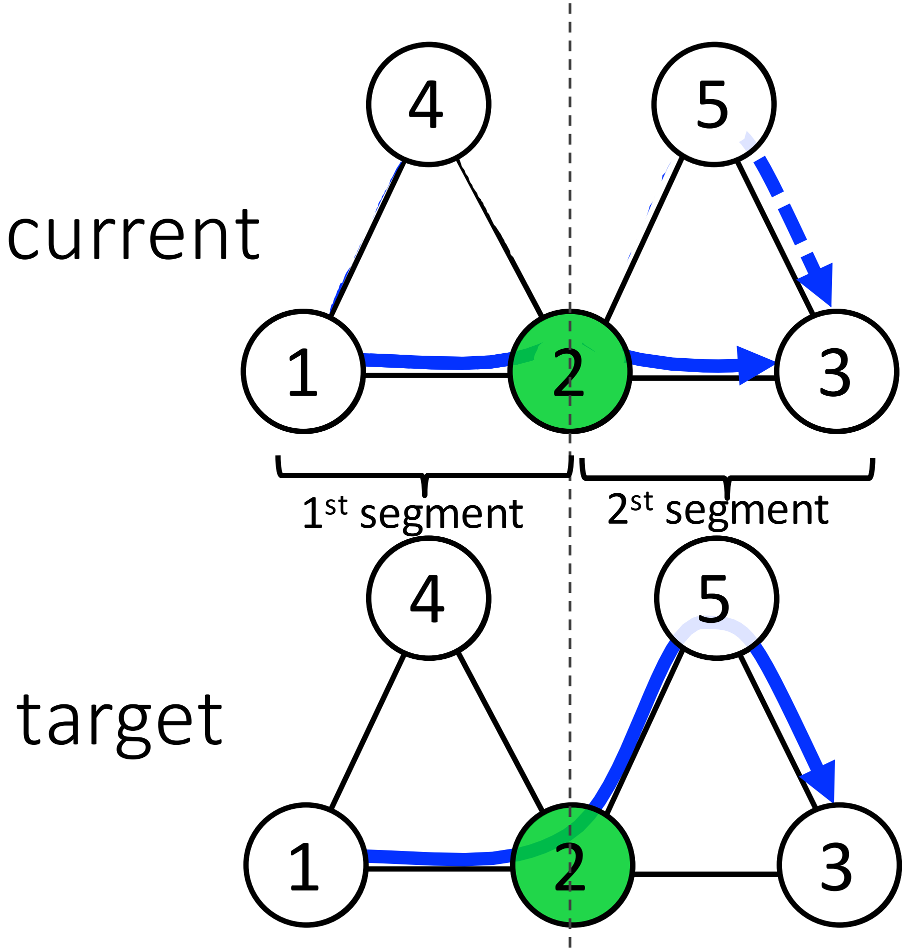


Switch 2 is traversed in both paths

Parallelizable segment updates

Update speed up via segmentation

Switch role: coordinate with neighbors

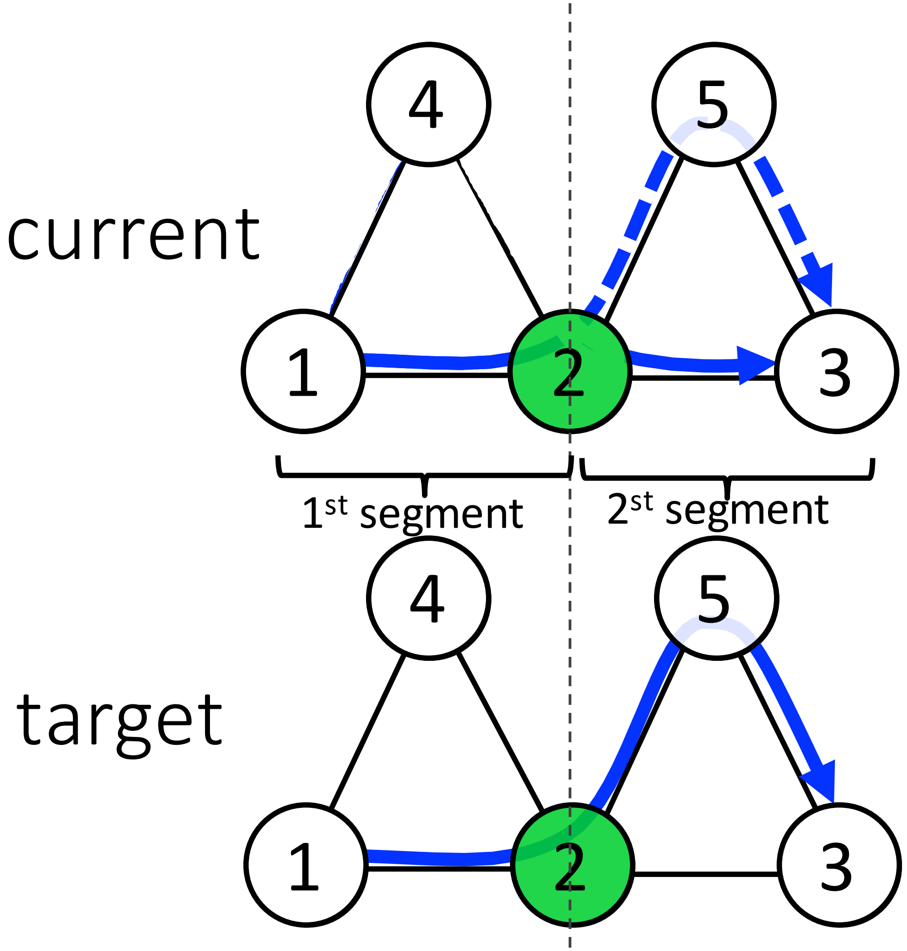


Switch 2 is traversed in both paths

Parallelizable segment updates

Update speed up via segmentation

Switch role: coordinate with neighbors

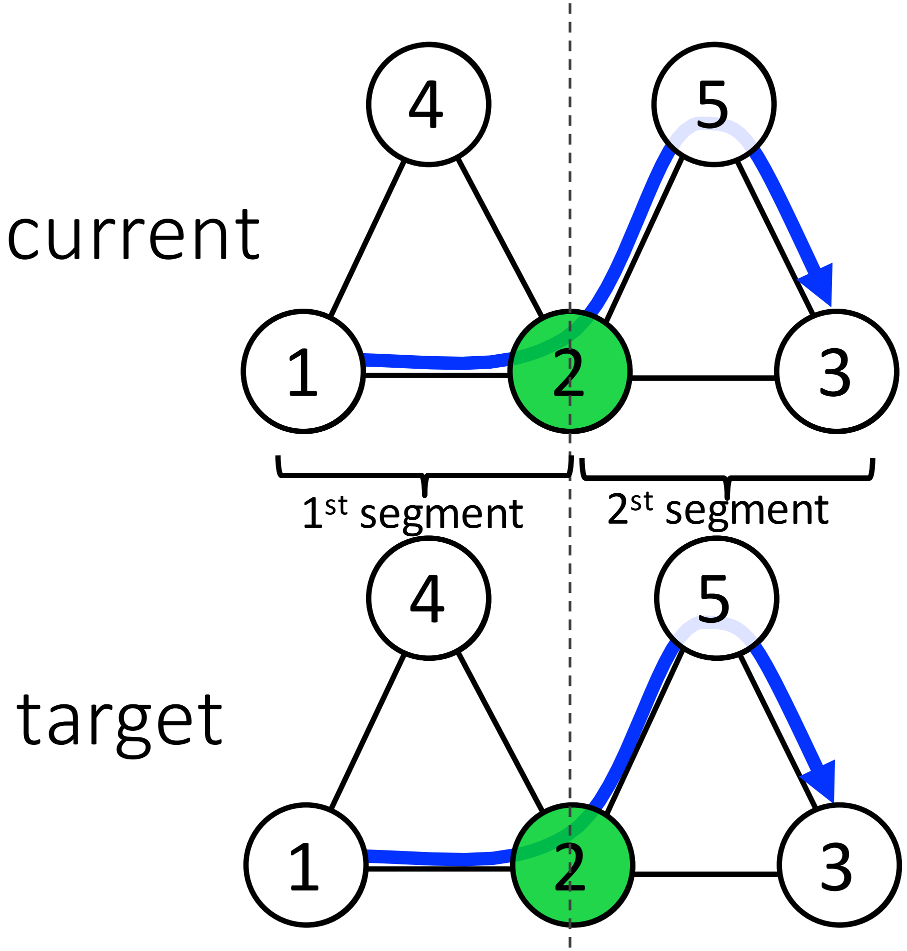


Switch 2 is traversed in both paths

Parallelizable segment updates

Update speed up via segmentation

Switch role: coordinate with neighbors



Switch 2 is traversed in both paths

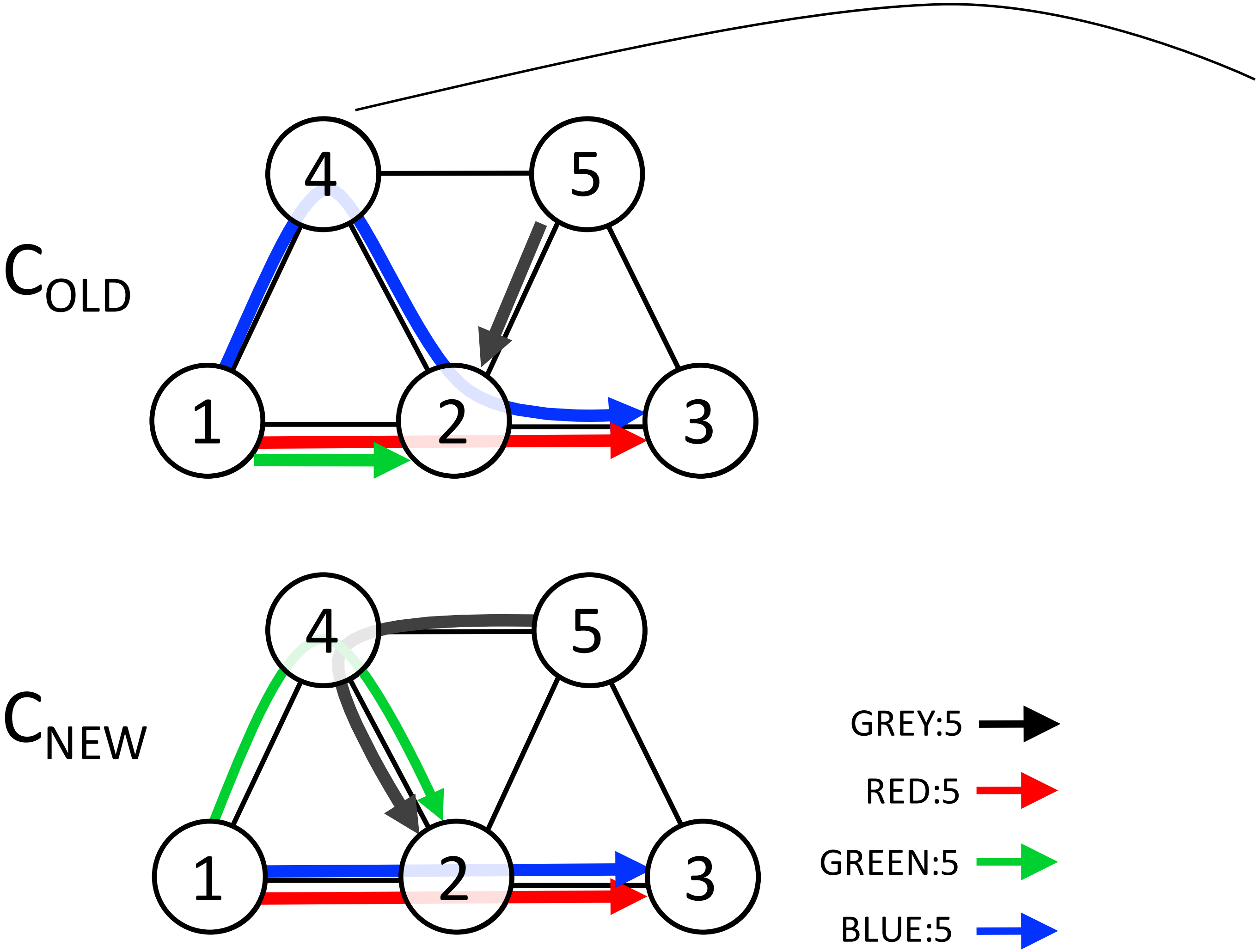
Parallelizable segment updates

Ez-segway: the switches perspective

Switches role:

- coordinate with neighbors
- combine local and global (pre-computed) information to perform the update
  - perform an update operations only if
    - i) there is enough spare capacity
    - ii) the update operations will not prevent any higher-priority update that is still not executable

Enforcing priorities



Switch 4 can move both GREEN and GREY.

It first moves GREEN since it has higher priority than GREY

Large-scale simulations

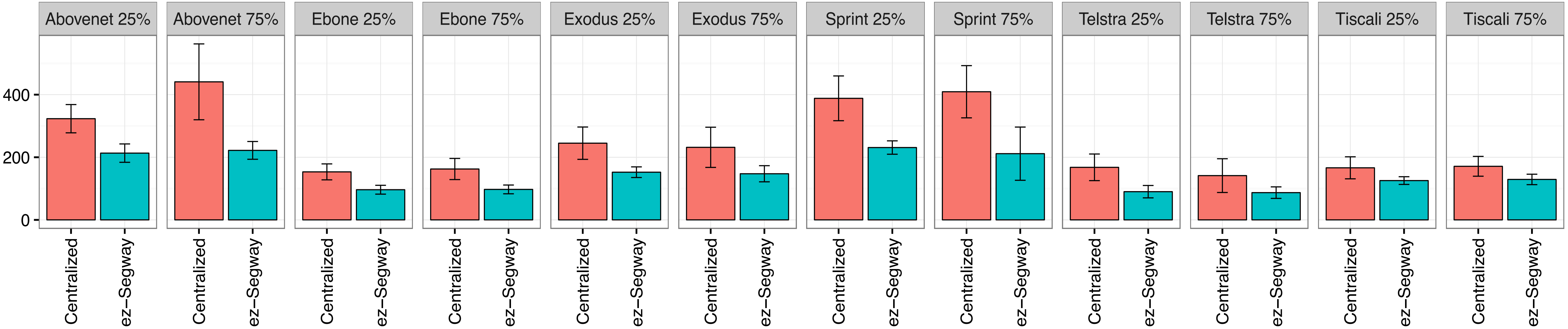
ez-segway: switches coordinate the update
centralized: controller coordinates the update

Measure: total update time

Setting:

- 6 real topologies from RocketFuel
- link capacities: 1…100 Gbps
- controller placed at centroid
- gravity traffic model
- shortest-path-via-random-node
- updates triggered by link failures
- 10 executions per topology

Update time comparison [ms]



Completion time reduced by 15%-50%

Summary

ezSegway design

- <u>**Control plane**</u> computes flows partial ordering
- <u>**Data plane**</u> coordinates the update

Better performance: Speeding up the update (up to 2x)

Ongoing work:

- Mininet evaluation
- Feasibility check on Centec switch
- Formalization