LOUP: The Principles and Practice of Intra-Domain Route Dissemination

Nikola Gvozdiev, Brad Karp, Mark Handley

The Rising Tide of Reachability Expectations

Internet users expect any-to-any reachability:

- Reliable transport masks losses caused by congestion
- Routing system adapts after topology changes

Loss under congestion and unreachability during routing convergence interrupt end-to-end connectivity

“Legacy” applications (e.g., file transfer, email) handle interruptions in connectivity well

Increasingly, applications are intolerant of brief interruptions in reachability:

VoIP, interactive gaming, high-frequency trading, …

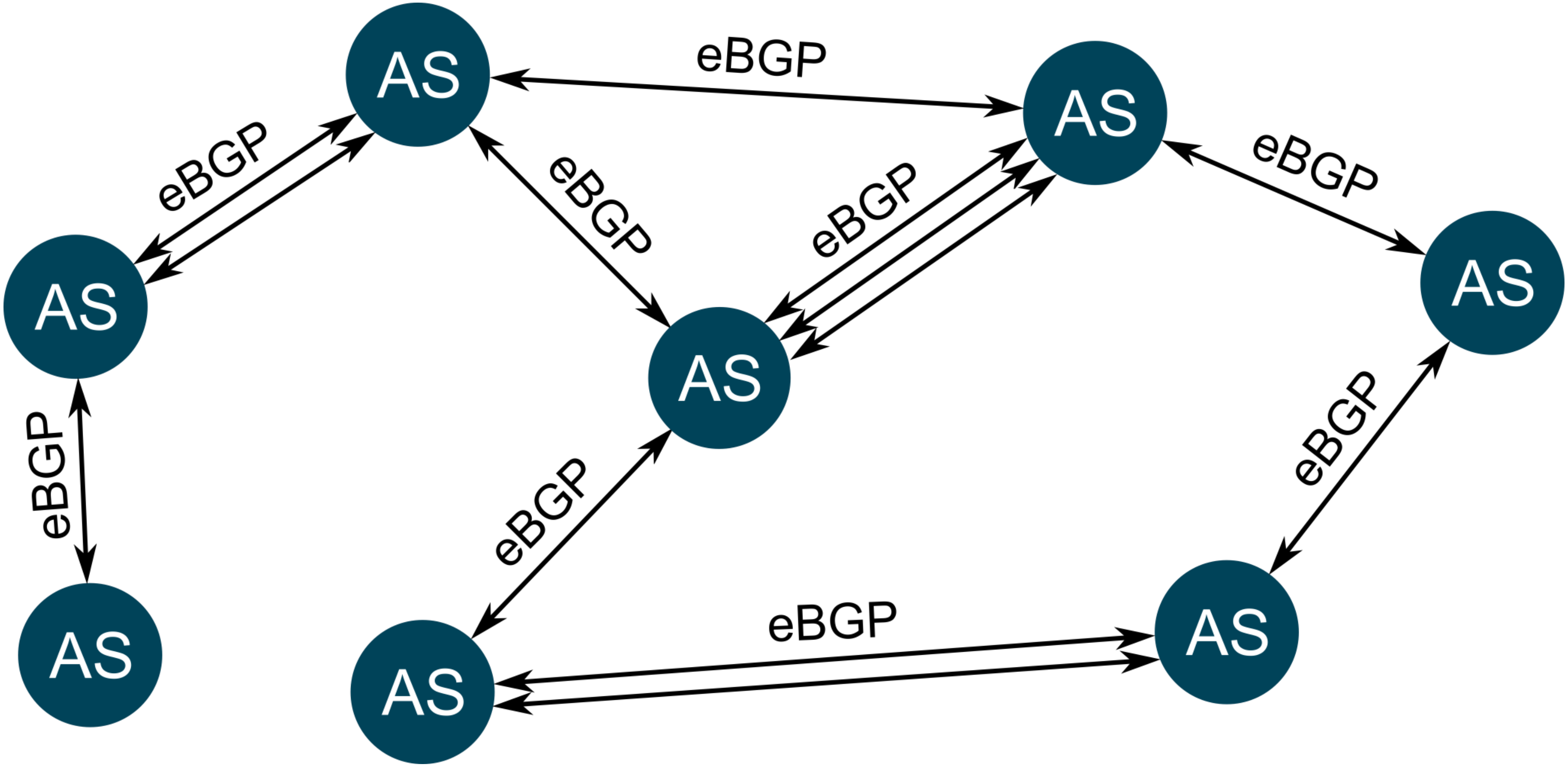
Routing a Major Source of Transient Unreachability

“VoIP usability is hindered as much by BGP's slow convergence as network congestion” [Kushman et al. 2007]

“Routing failures contribute to end-to-end packet loss significantly ... common iBGP configuration and MRAI timer values play a major role in causing packet loss during routing events.” [Wang et al. 2006]

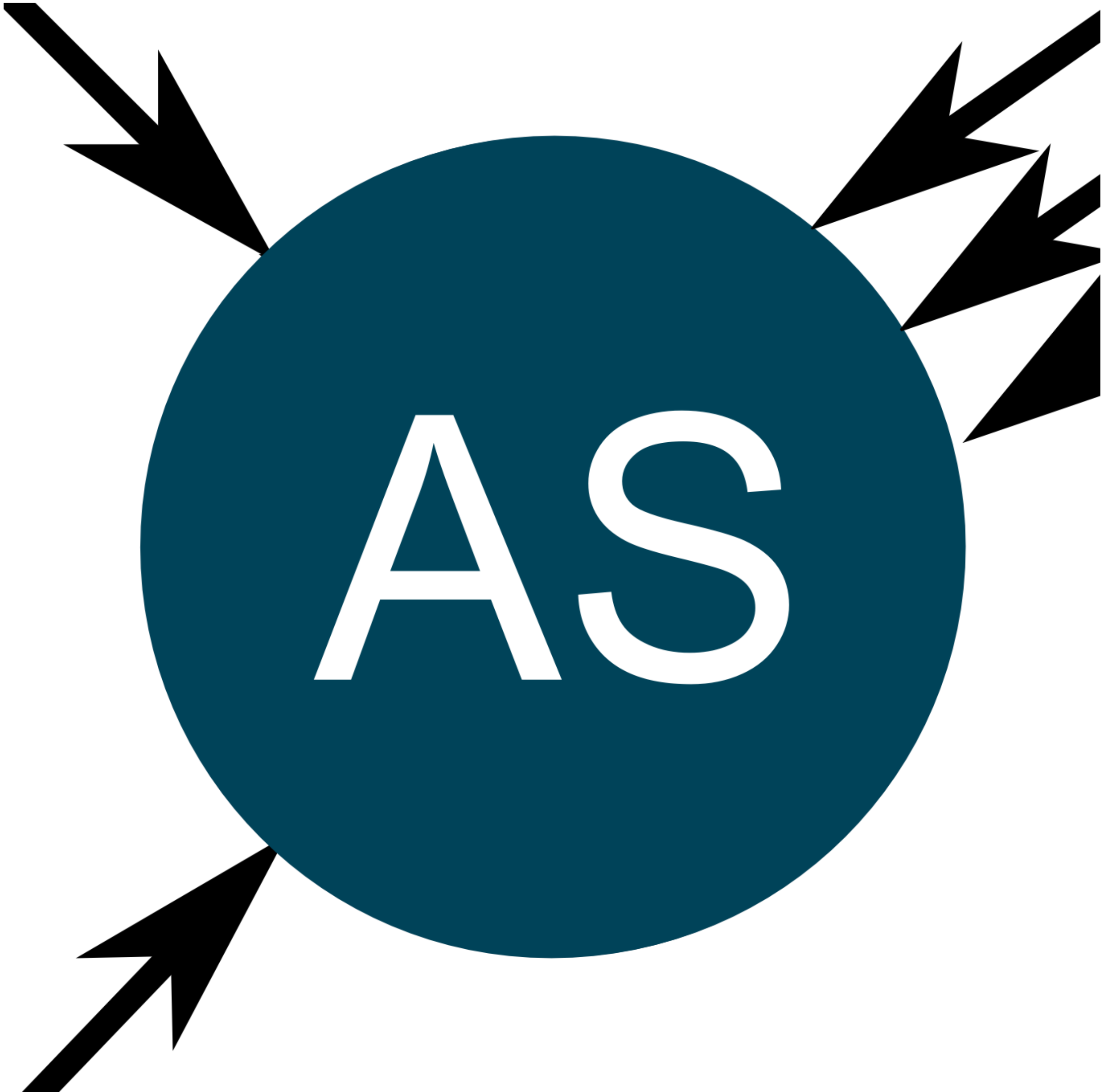
How can we make the routing system better support interruption-intolerant applications?

The big picture

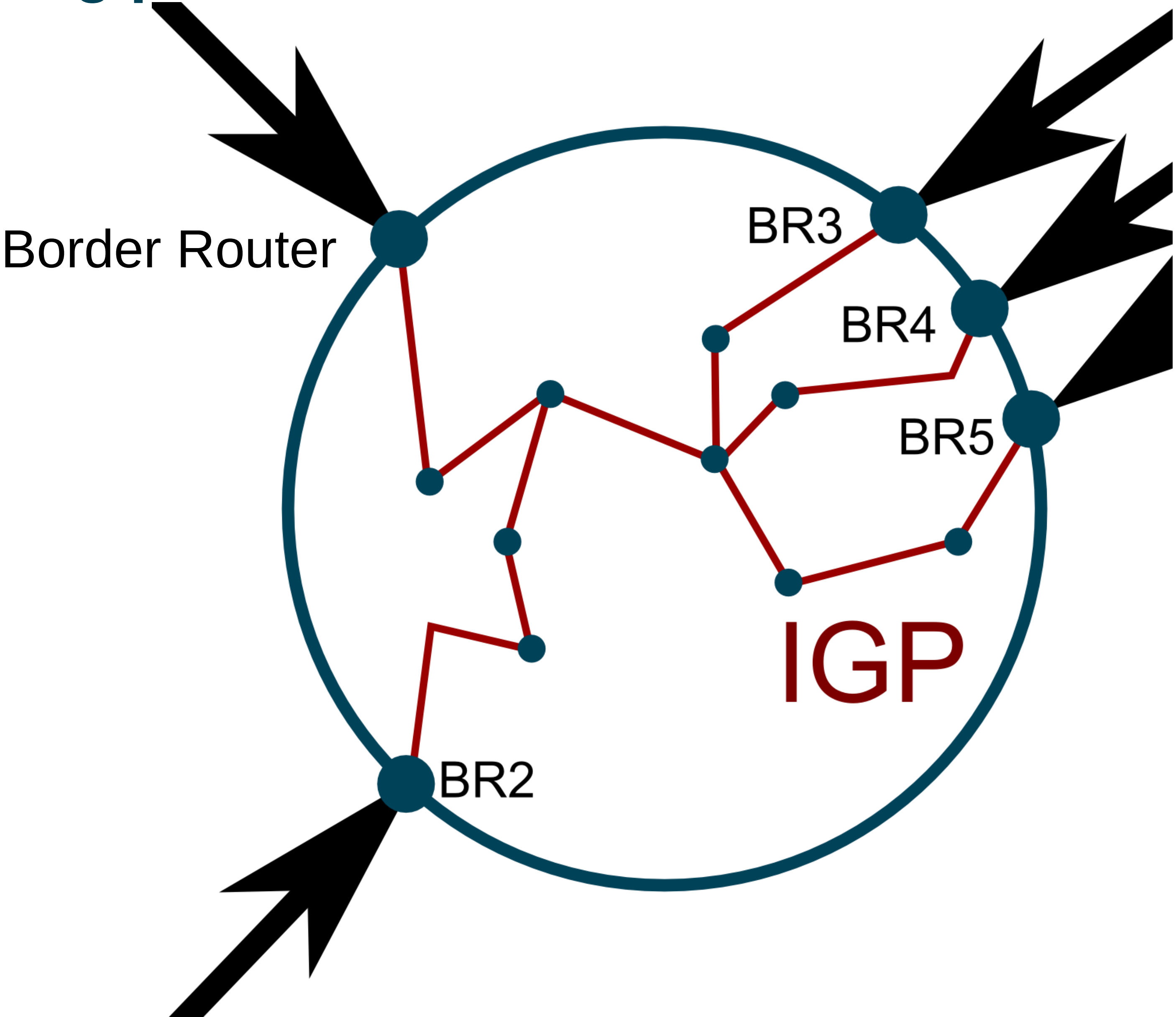


eBGP – external BGP

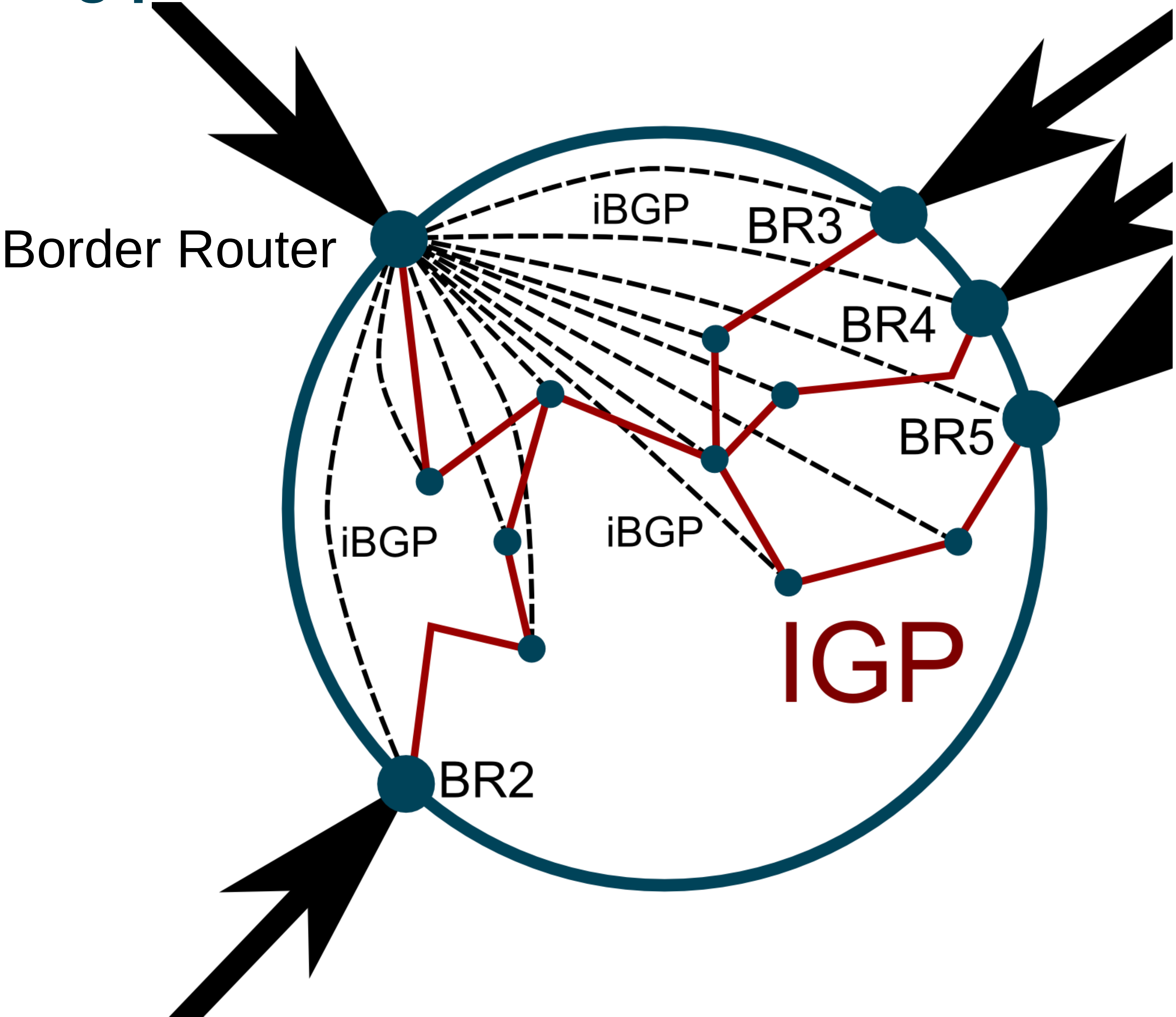
The big picture



The big picture



The big picture



The missing iBGP piece

Previous work has looked into

- the interior gateway protocol

  [Francois 2007], [Shaikh 2006], [Wu 2005], [Garcia-Luna-Aceves 1993]
- eBGP reliability, scalability and configuration

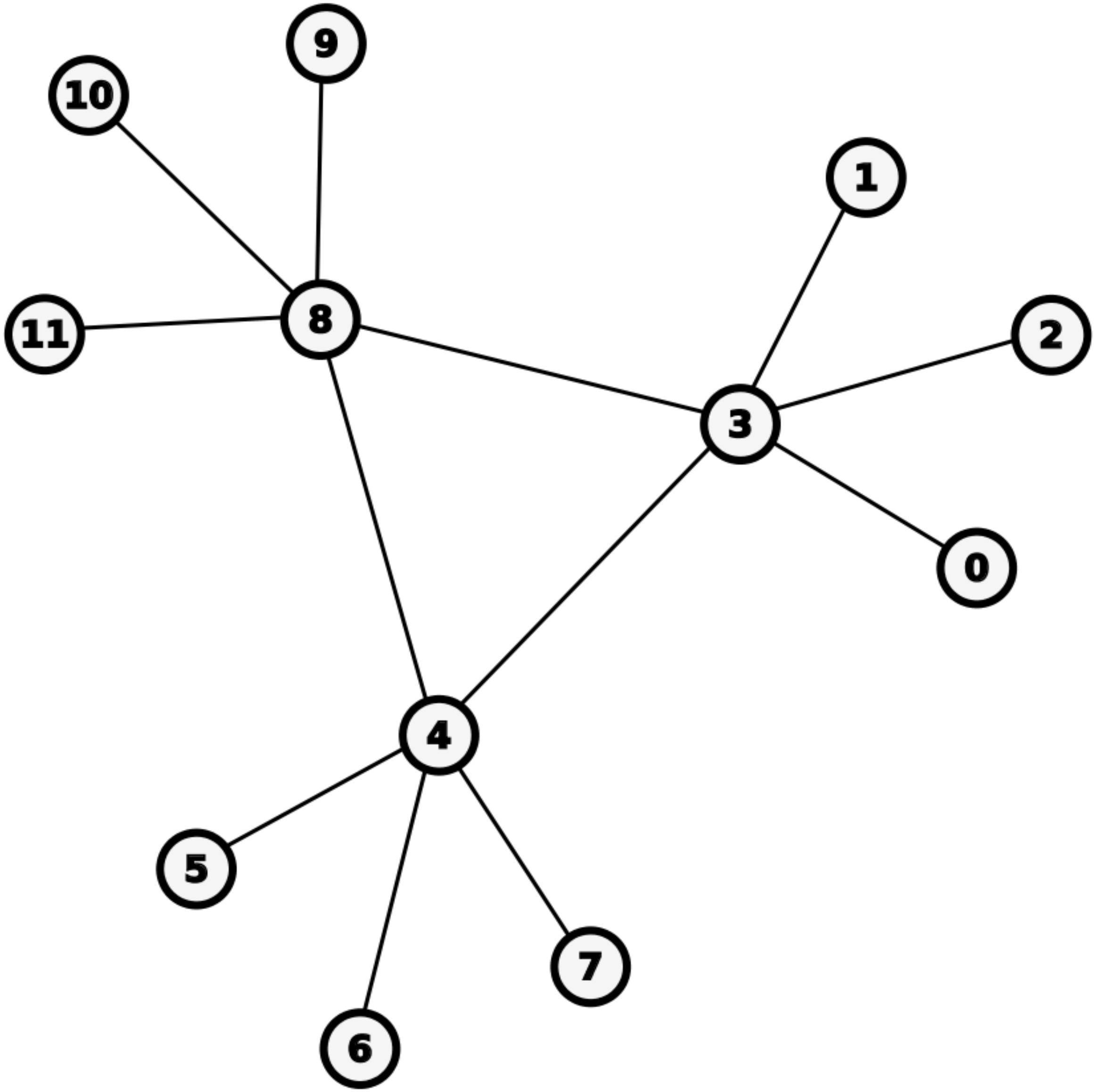
  [Bonaventure 2007], [Chandrashekar 2005], [Wu 2005], [Feamster 2004]
- reachability during eBGP convergence

  [Van Beijnum 2009], [John 2008], [Pei 2004], [Barr 2003]
- iBGP reliability, scalability and configuration

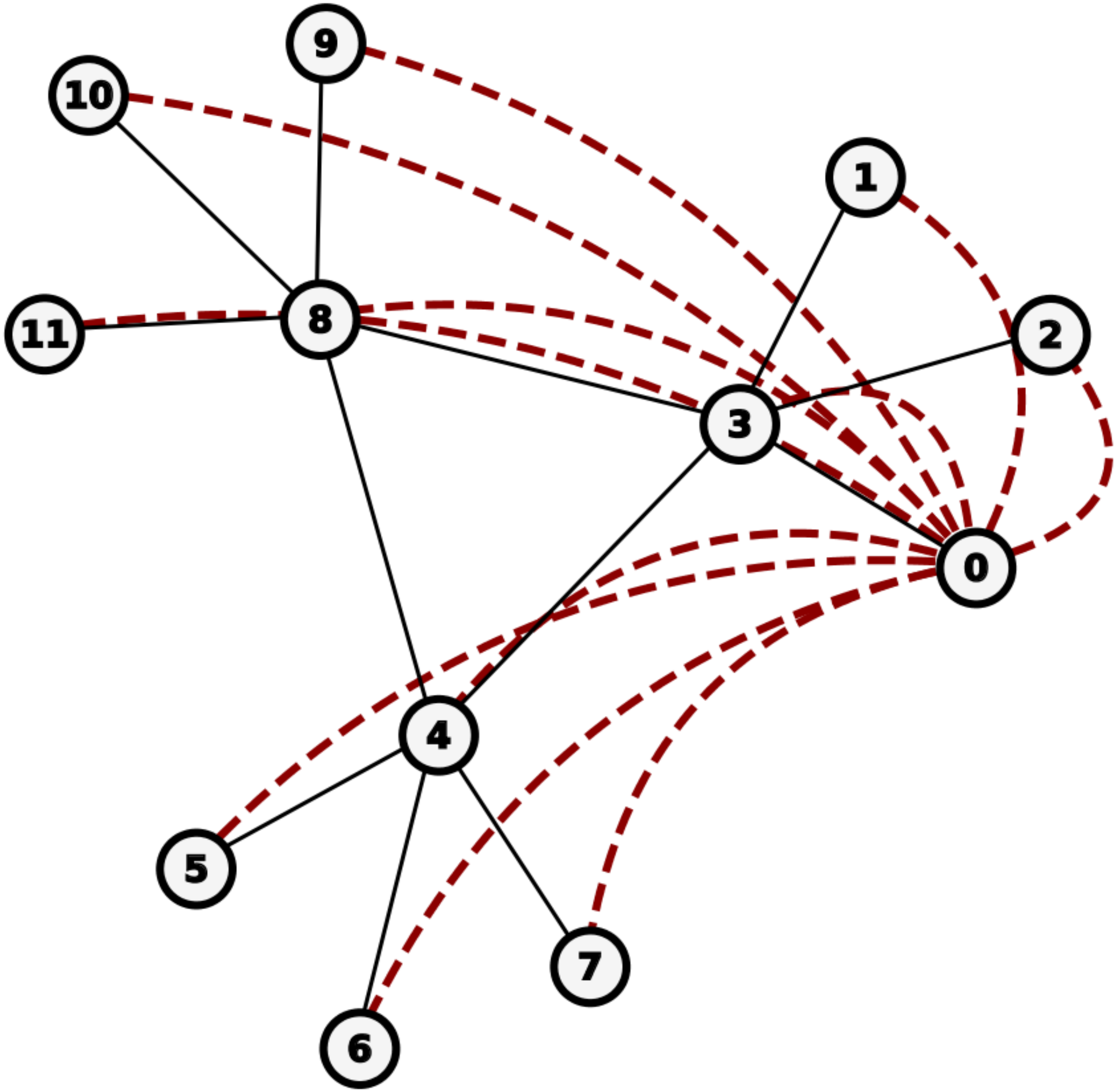
  [Caesar 2005], [Feamster 2005], [Bonaventure 2004], [Griffin 2002], [Gao 2001]
- reachability during iBGP convergence ?

Fundamental behavior of intra-AS route propagation unexamined

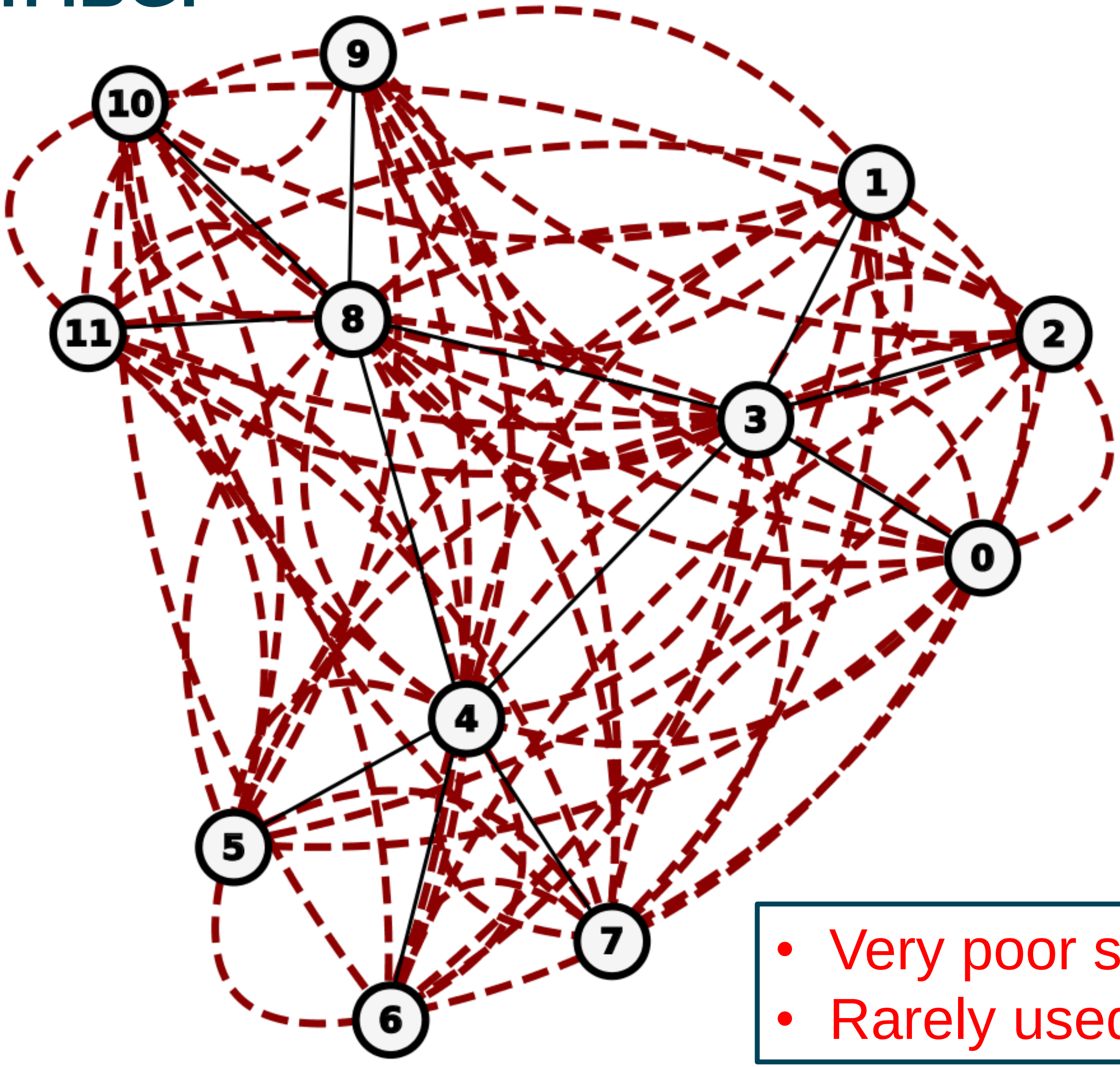
Full-mesh iBGP



Full-mesh iBGP

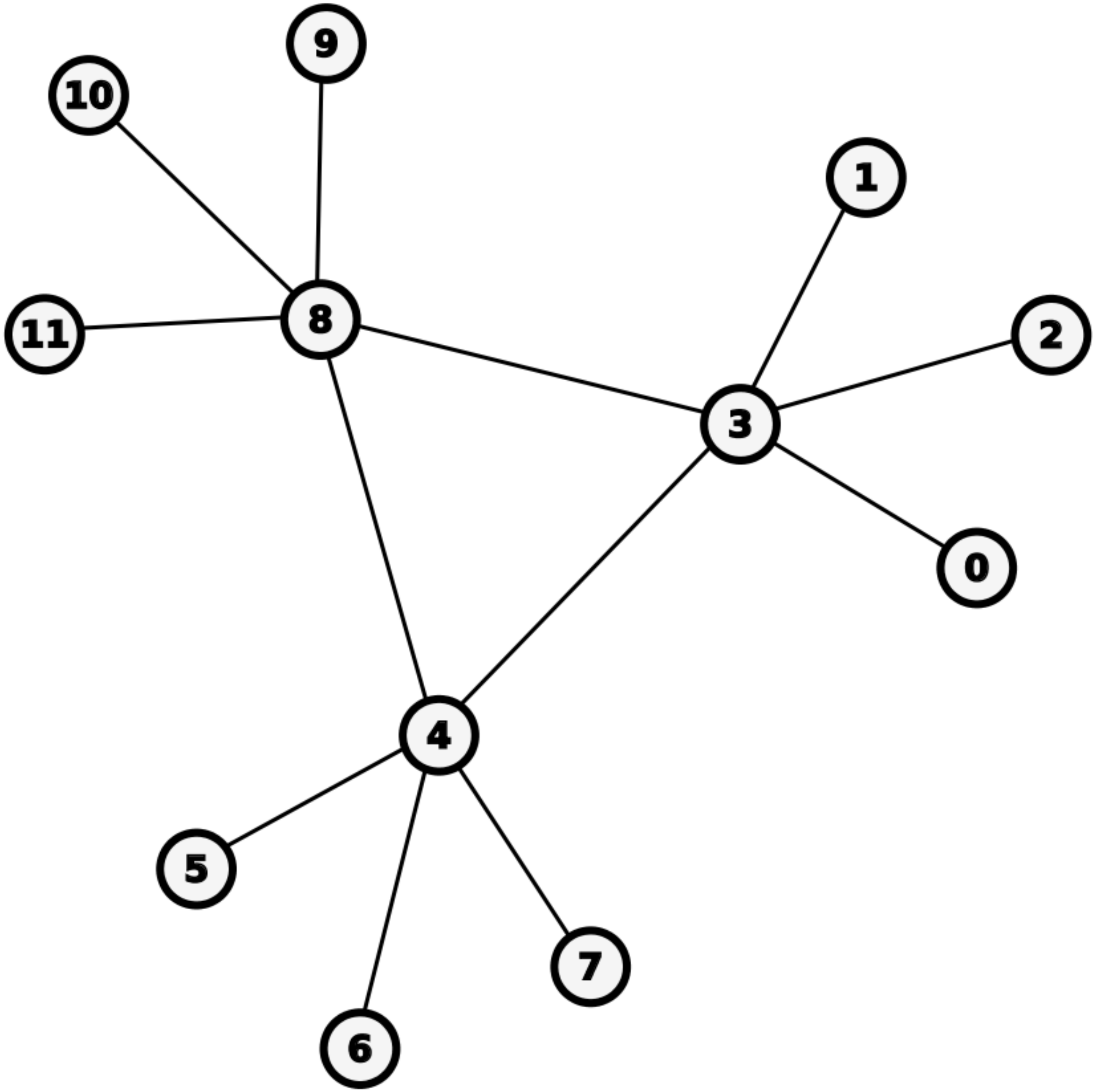


Full-mesh iBGP

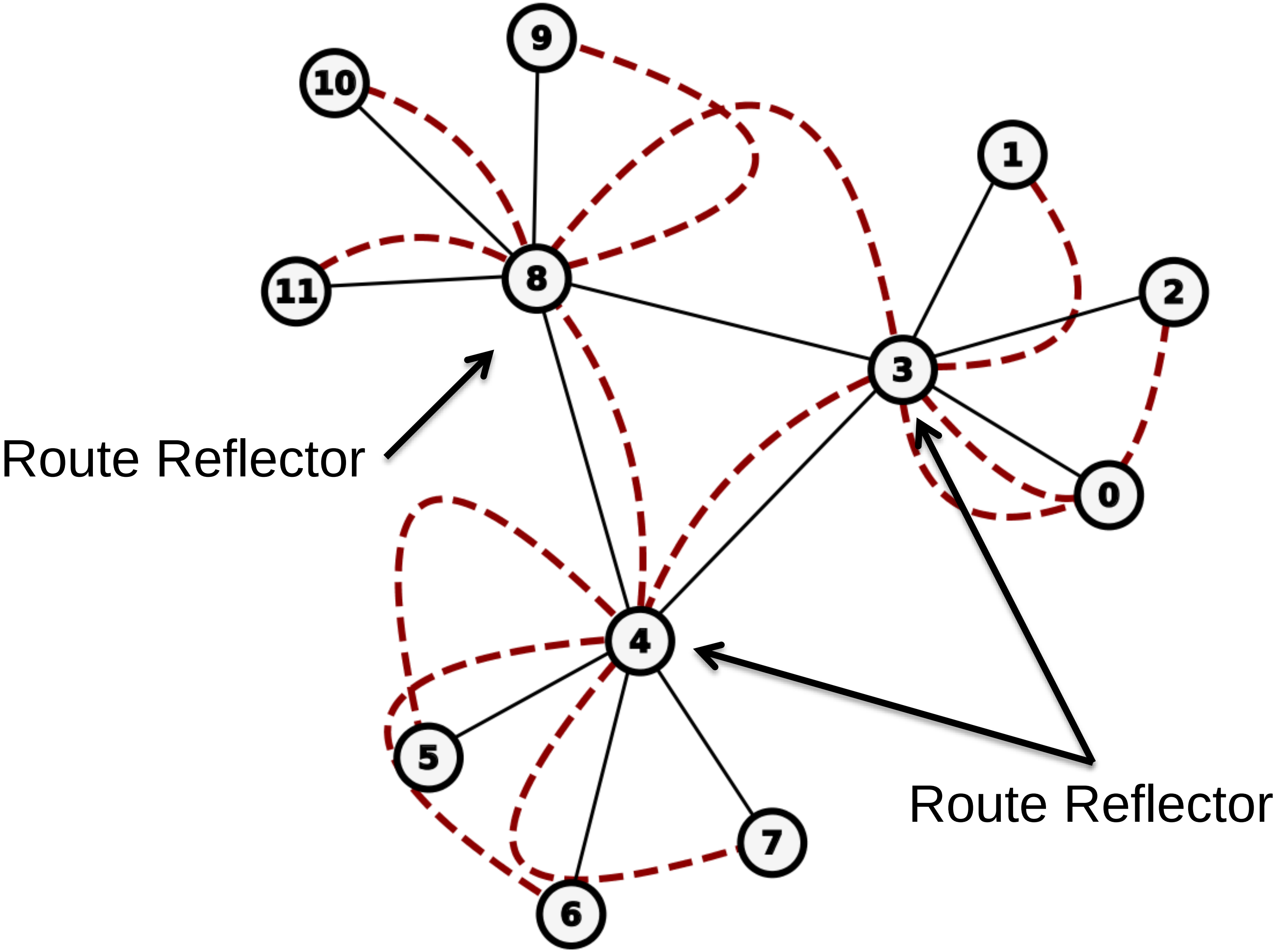


- Very poor scalability
- Rarely used

Route reflectors



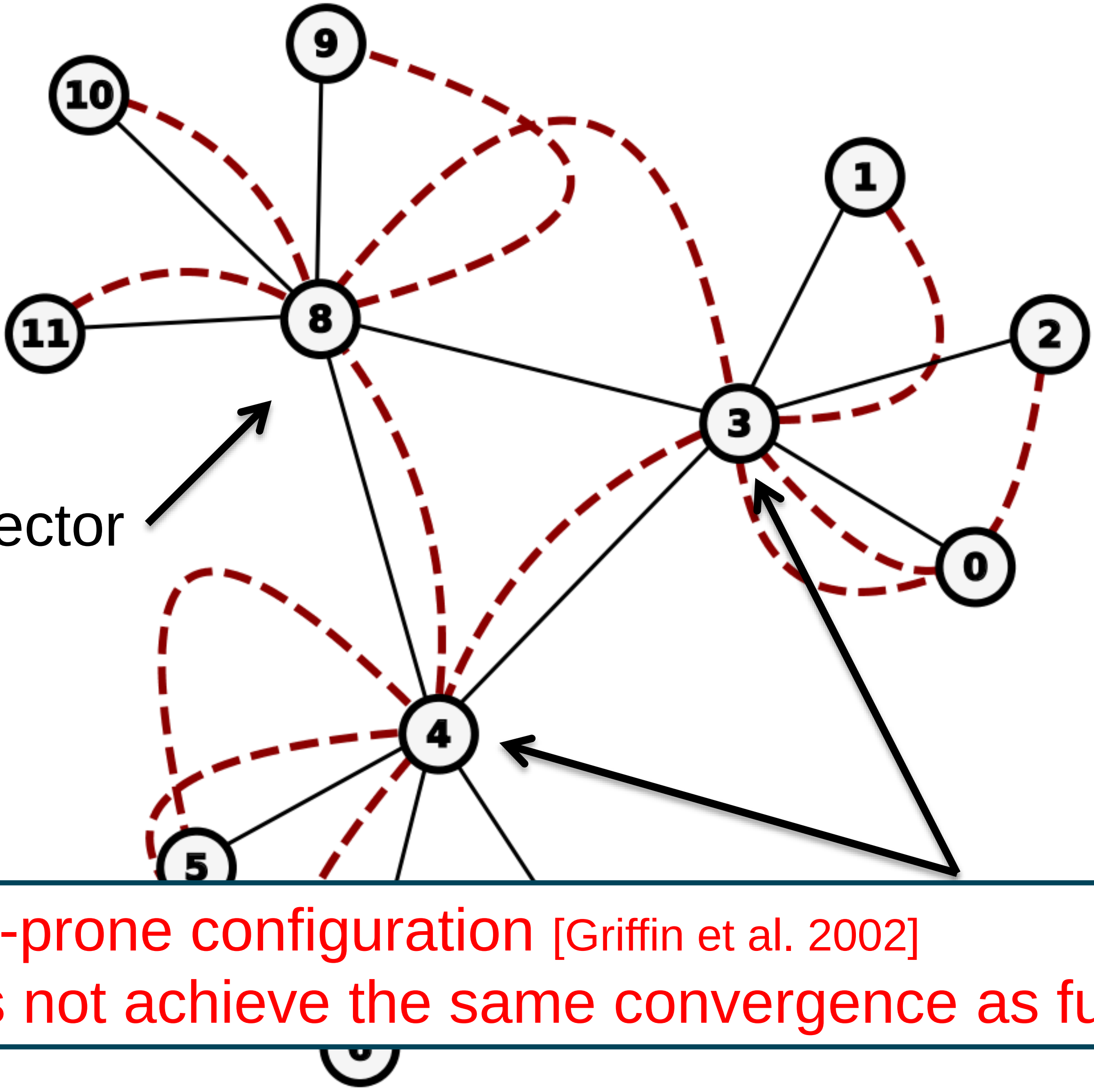
Route reflectors



Route Reflector

Route Reflector

Route reflectors

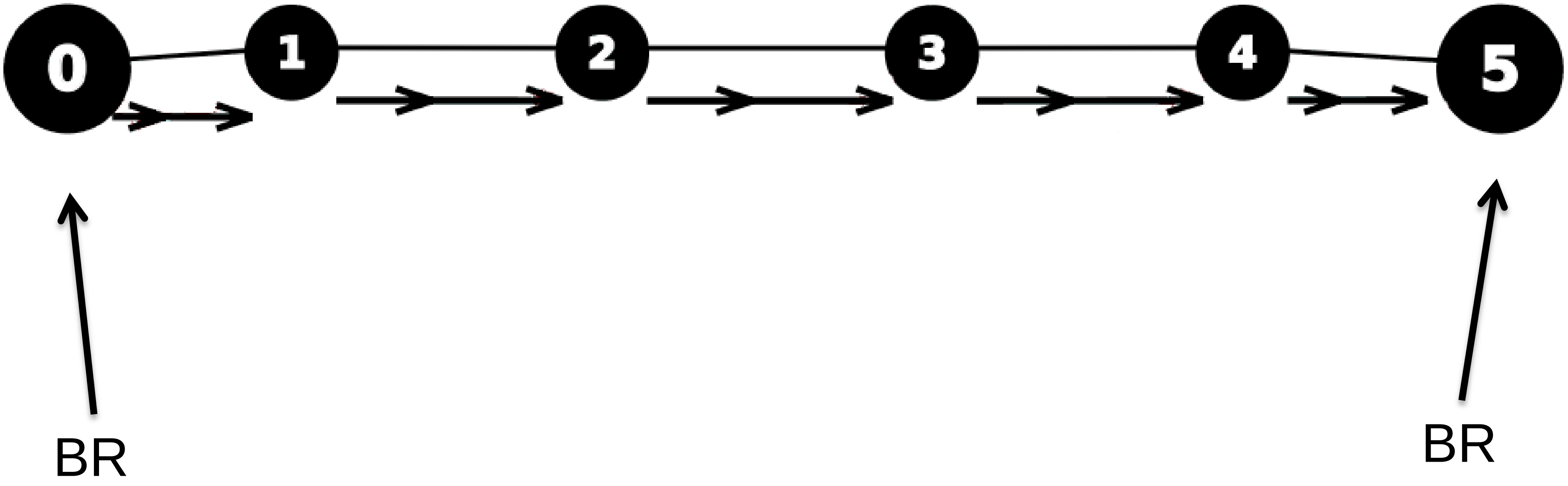


Route Reflector

- Error-prone configuration [Griffin et al. 2002]
- Does not achieve the same convergence as full-mesh

How does iBGP go wrong?

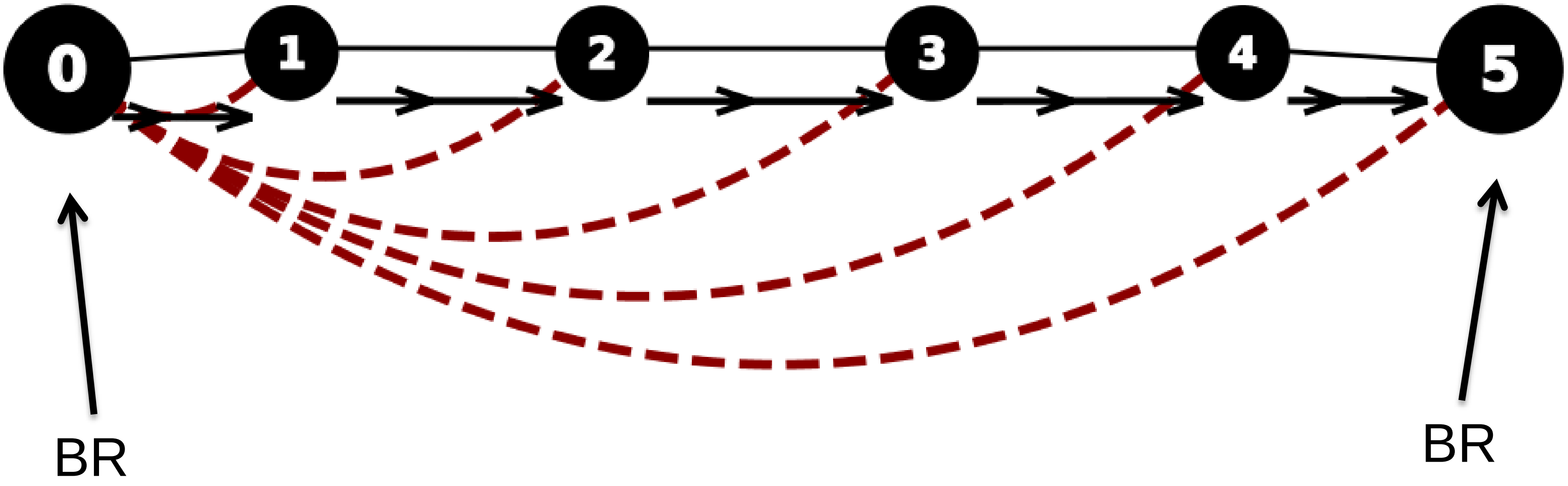
iBGP update causes transient loops



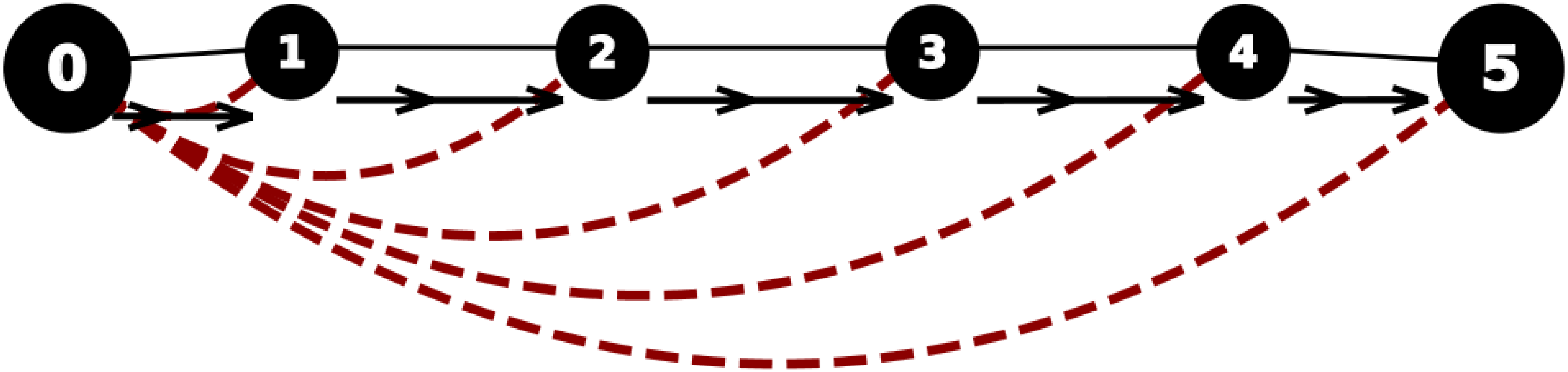
BR

BR

iBGP update causes transient loops



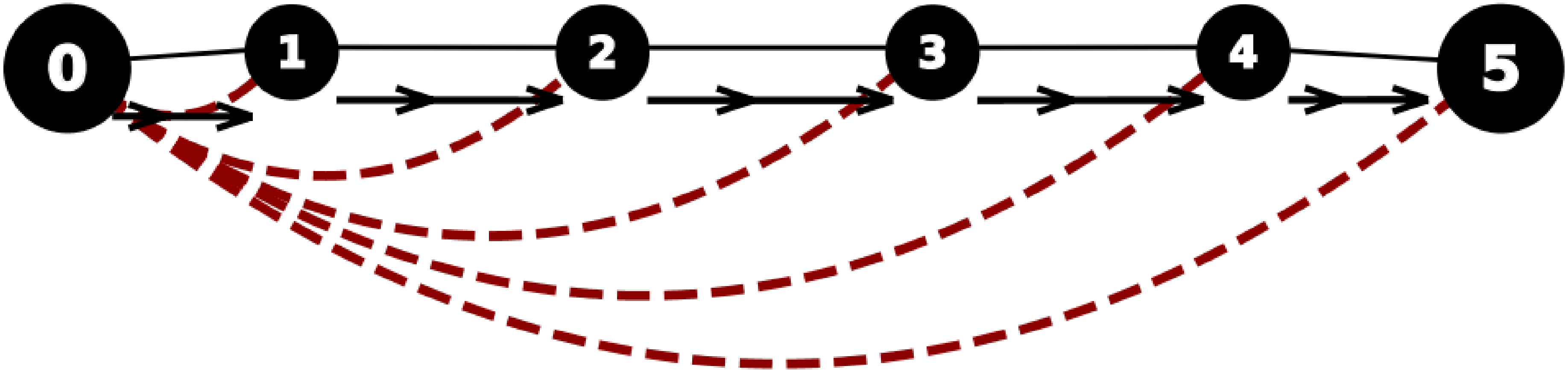
iBGP update causes transient loops



Router 0 receives a better alternative and switches

Router 0 sends messages to update all other routers

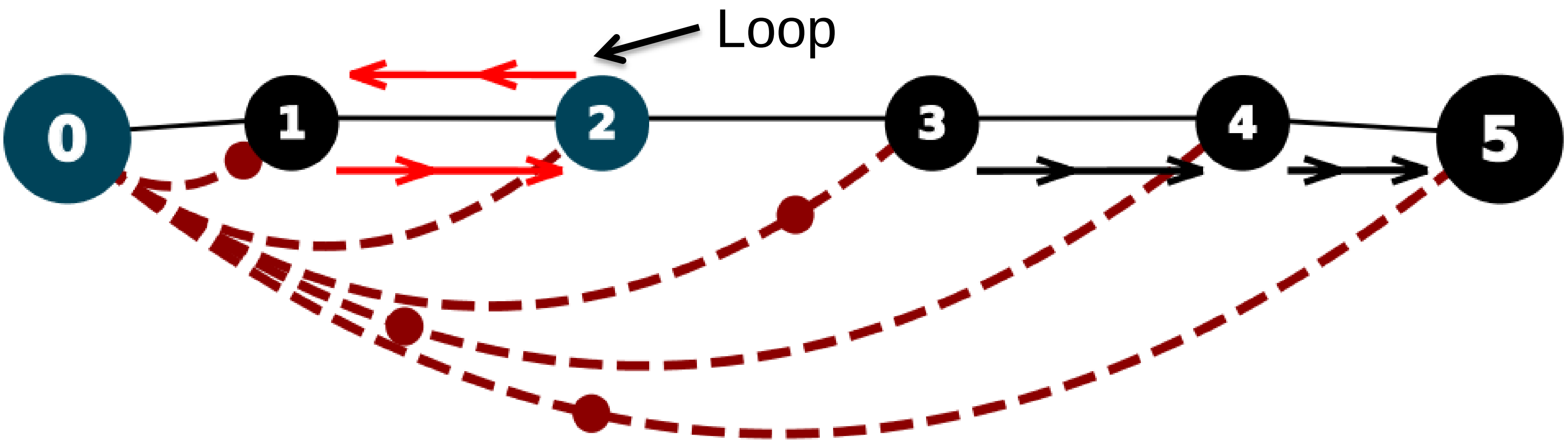
iBGP update causes transient loops



Router 0 receives a better alternative and switches

Router 0 sends messages to update all other routers

iBGP update causes transient loops



Router 1 is slow to process the message or 0-1 is congested

Loops due to lack of ordering ***between*** sessions

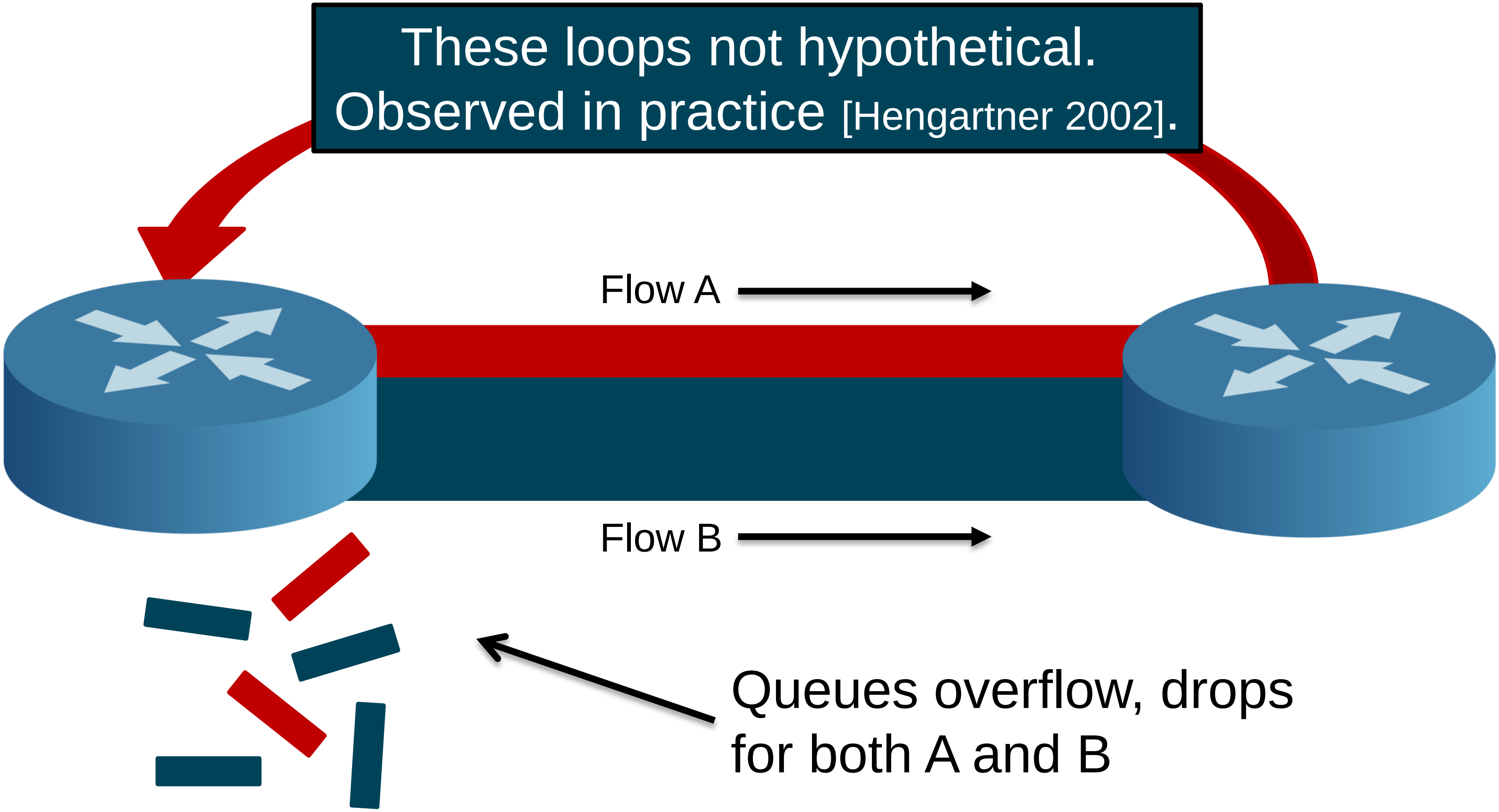
Why bother?

These loops surely are very transient. After all links in the core are very fast and control traffic is prioritized.

No. The control plane is a lot slower than the forwarding plane. BGP processing delays can be 100s of ms [Feldmann et al. 2004].

The collateral damage of routing loops

When a loop occurs if the link is busy all flows will experience loss.



These loops not hypothetical.
Observed in practice [Hengartner 2002].

Flow A

Flow B

Queues overflow, drops
for both A and B

SOUP and LOUP

In this talk we propose 2 iBGP replacements:

- SOUP (Simple Ordered Update Protocol)
- LOUP (Link-Ordered Update Protocol)

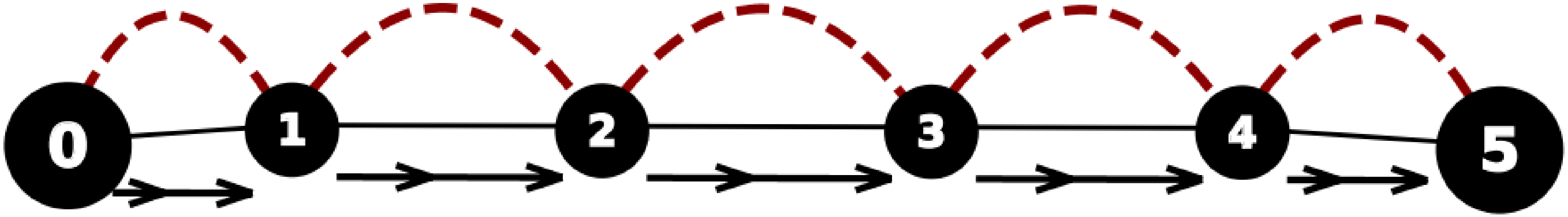
SOUP - ***provably loop-free***, but converges slowly in some cases

LOUP - ***converges faster***, but can loop in rare occasions

Ordering the application of updates avoids loops

We want to enforce ordering to the application of updates

Ordering the application of updates avoids loops



We want to enforce ordering to the application of updates

- Single-hop sessions between neighbors
- Only forward an update that you have processed
- Flood updates to propagate a “wavefront”

SOUP ingredients

Wavefront propagation

- Basic ordering of updates

Reverse Forwarding Tree (RFT) and Forward Activation (FA)

- New / improving routes

Reverse Activation (RA)

- Worsening routes / withdrawals

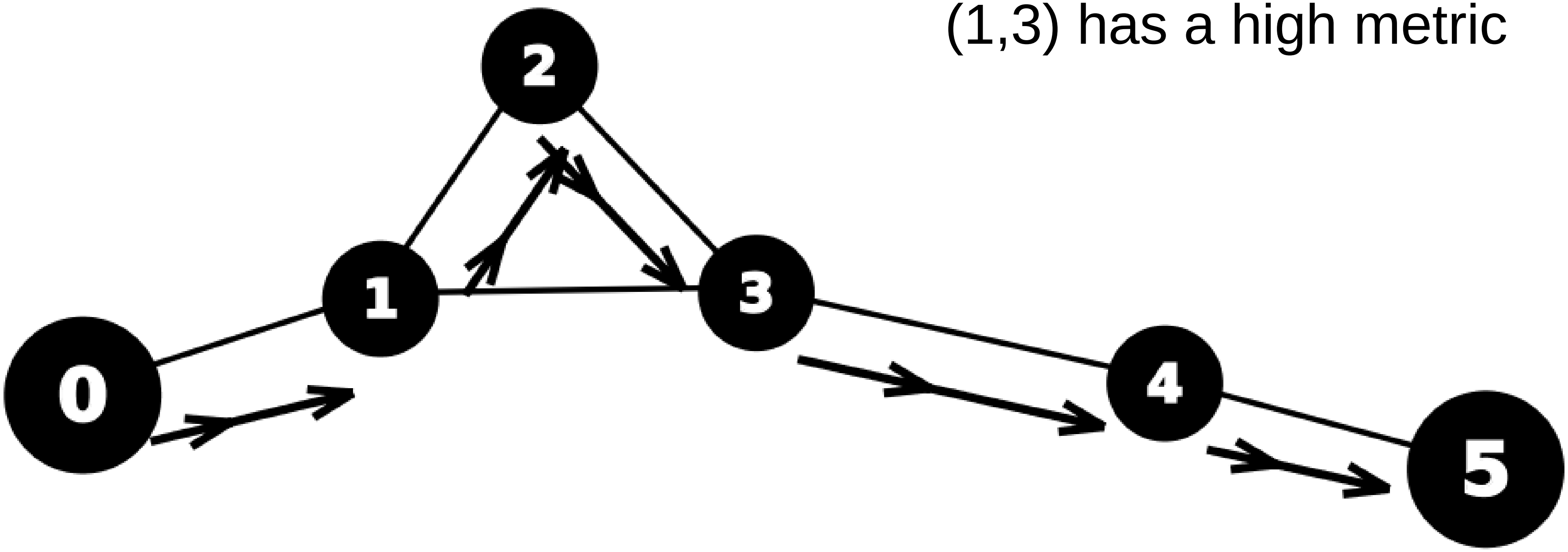
RA -> FA switch

- Multiple alternatives propagating simultaneously
- Complete loop freedom

What about more complex topologies?

Flooding a wavefront is insufficient

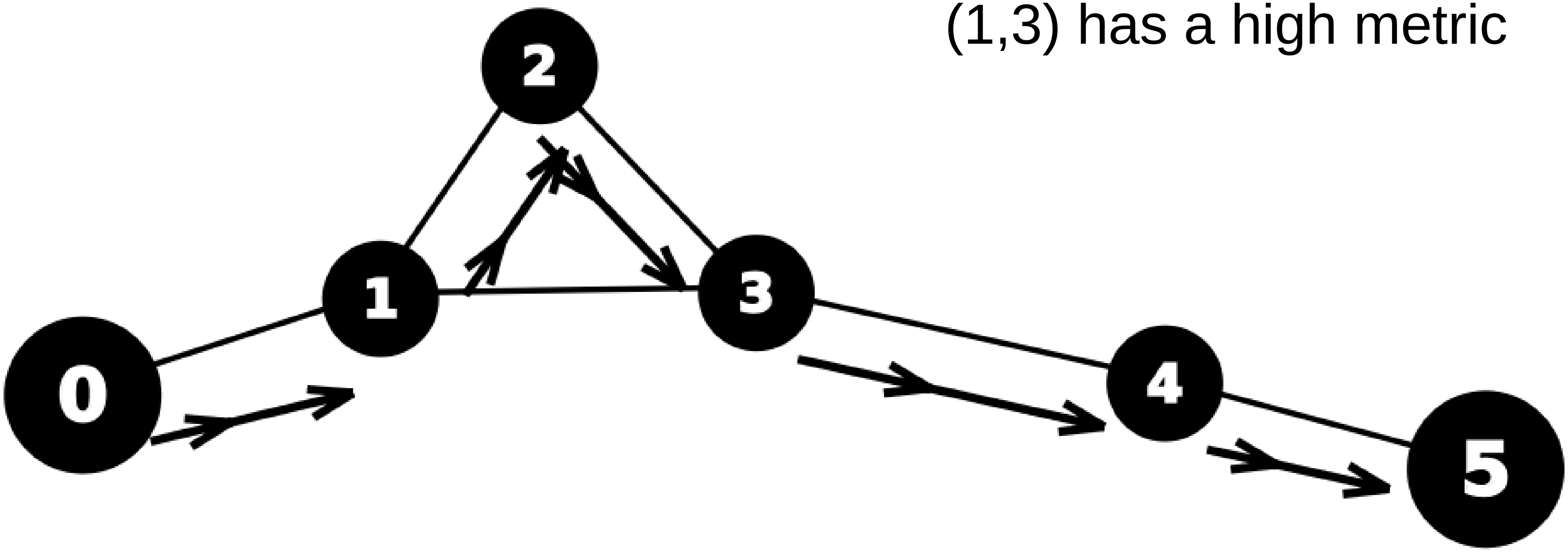
(1,3) has a high metric



Same setup, one more link

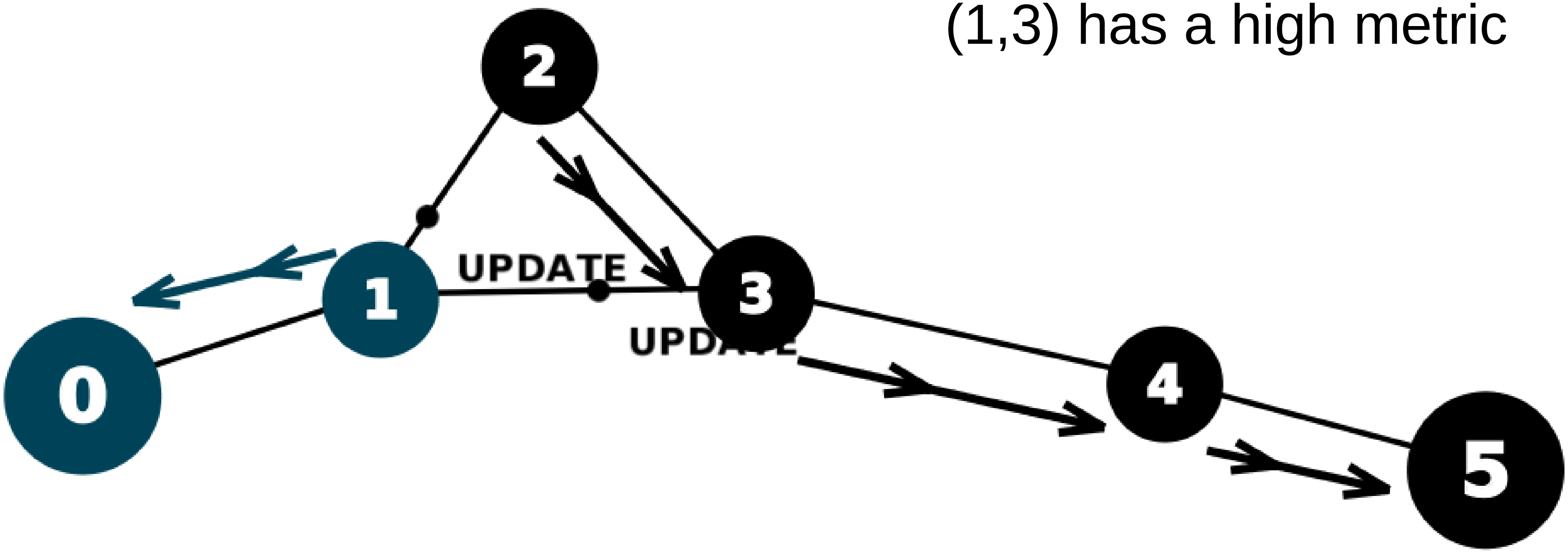
Flooding a wavefront is insufficient

(1,3) has a high metric



Router 0’s update is forwarded by 1 to both 2 and 3

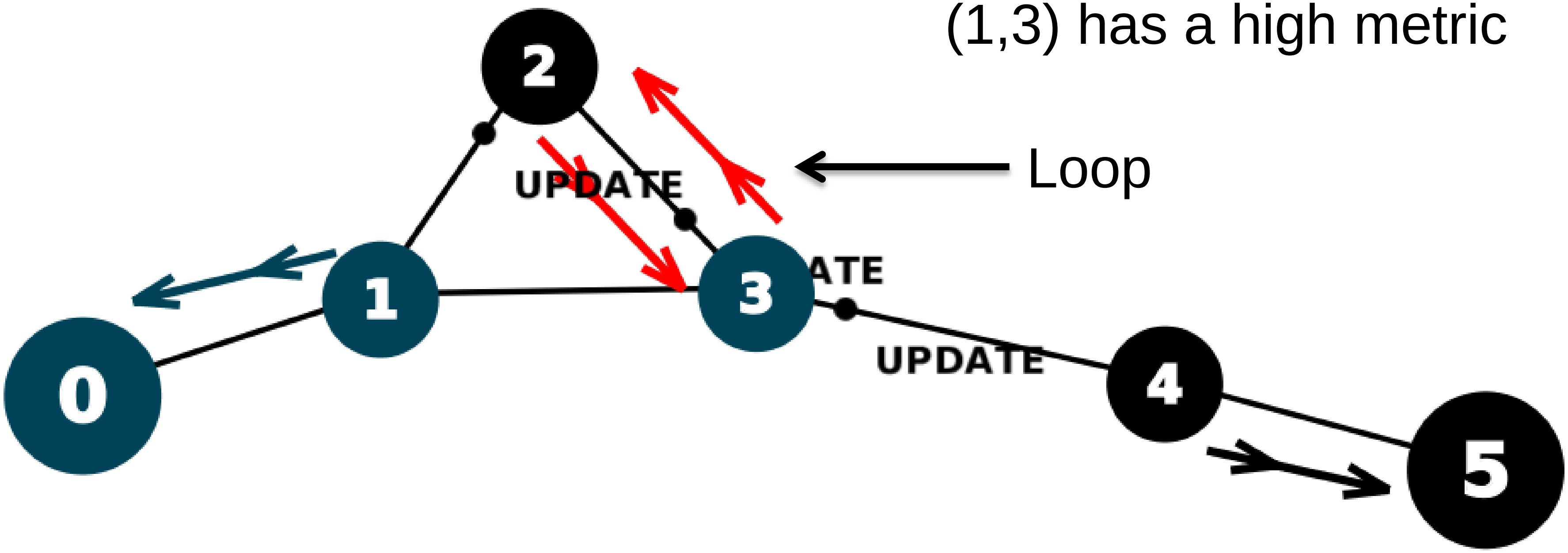
Flooding a wavefront is insufficient



(1,3) has a high metric

Router 0’s update is forwarded by 1 to both 2 and 3

Flooding a wavefront is insufficient



(1,3) has a high metric

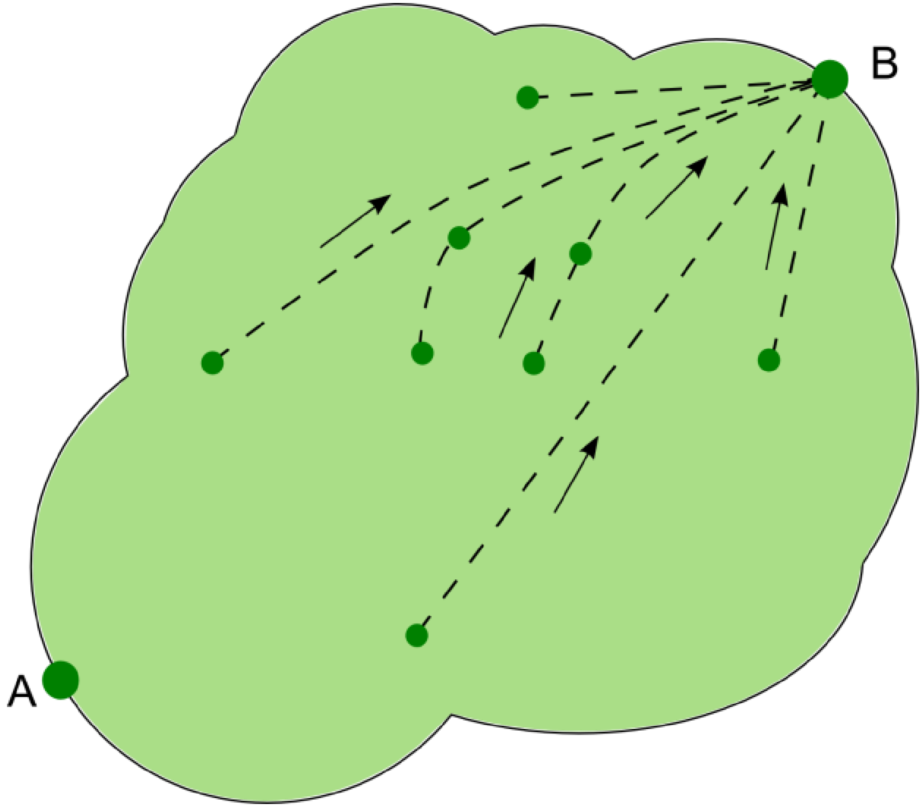
Loop

Loop due to 0’s update reaching 3 before 2

Even though (1,3) is not on anyone’s forwarding path

Why did it loop?

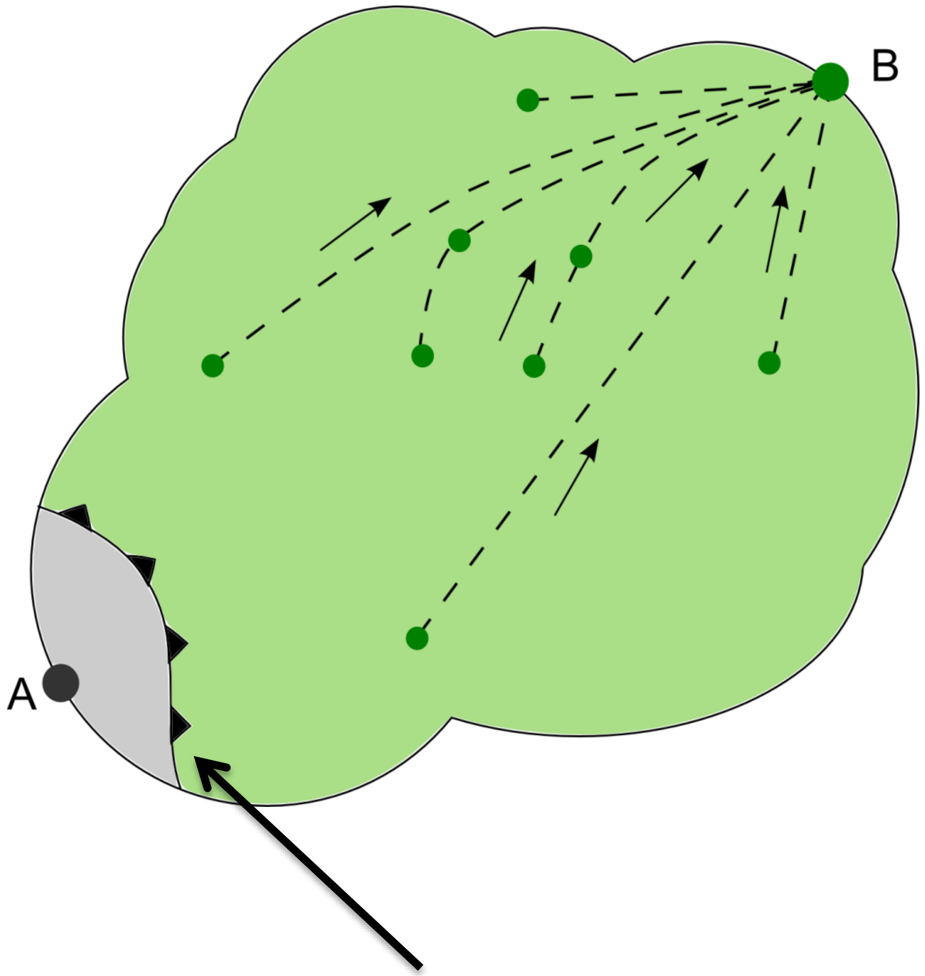
- Only one prefix
- Initially only one route via B



Why did it loop?

- Only one prefix
- Initially only one route via B

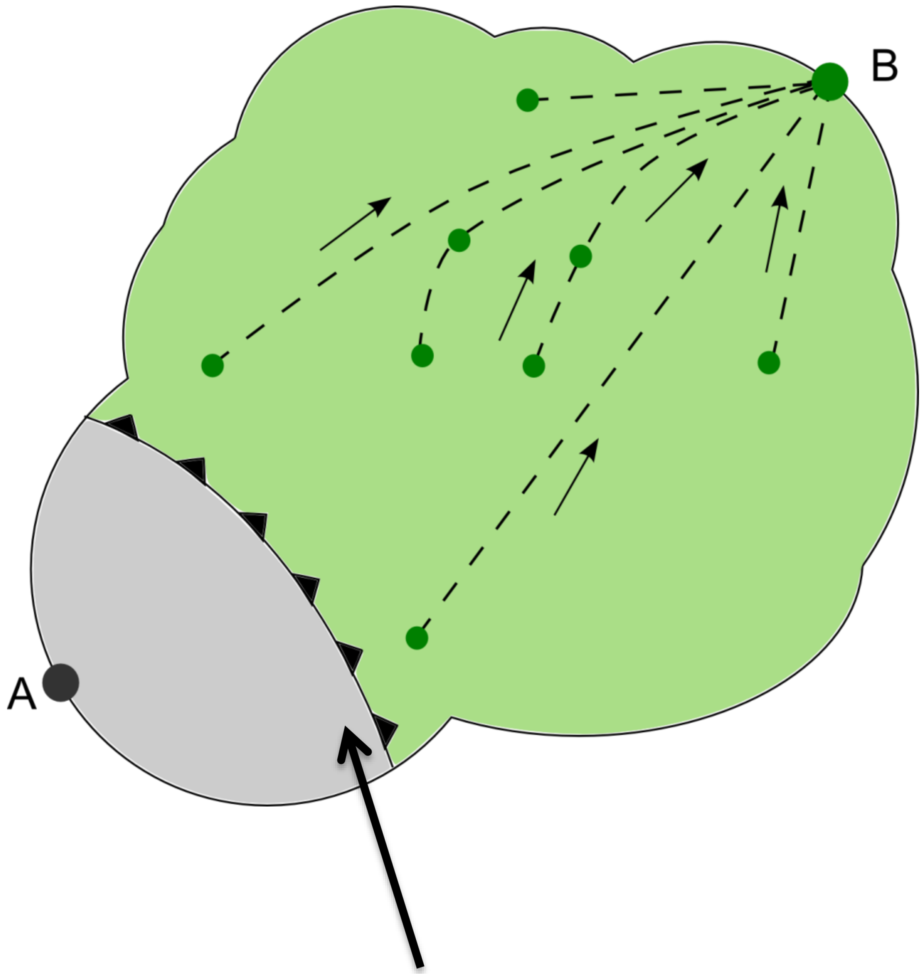
- New better alternative at A
- Propagates as a wavefront



Update from A

Why did it loop?

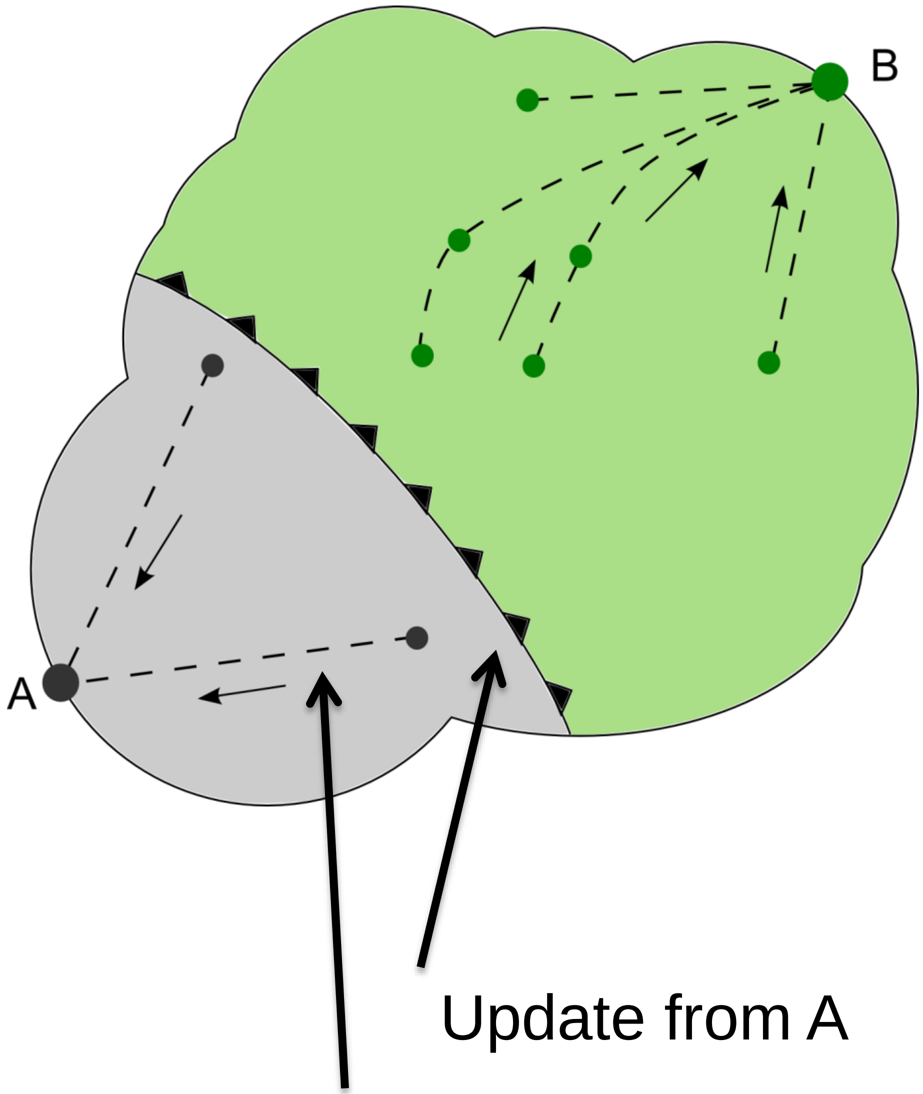
- Only one prefix
- Initially only one route via B
- New better alternative at A
- Propagates as a wavefront



Update from A

Why did it loop?

- Only one prefix
- Initially only one route via B
- New better alternative at A
- Propagates as a wavefront
- Routers switch to A

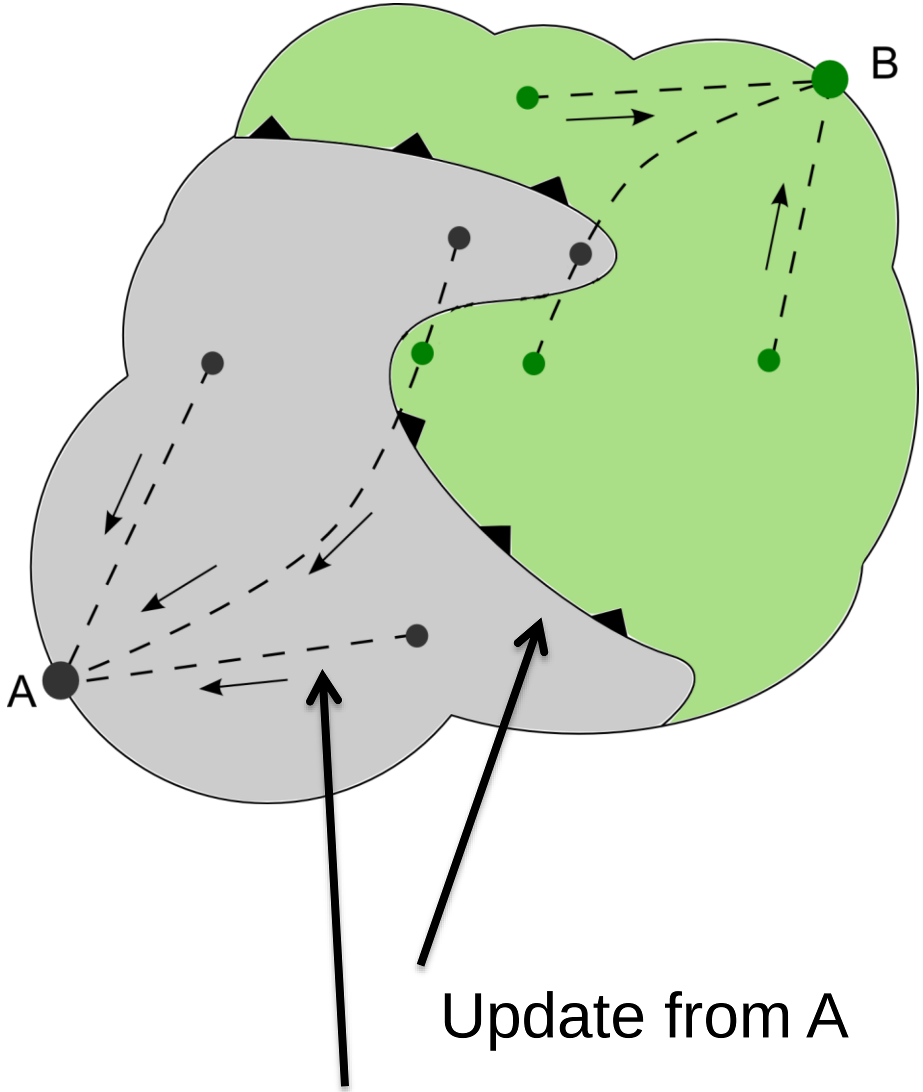


Update from A

Forwarding paths to A

Why did it loop?

- Only one prefix
- Initially only one route via B
- New better alternative at A
- Propagates as a wavefront
- Routers switch to A



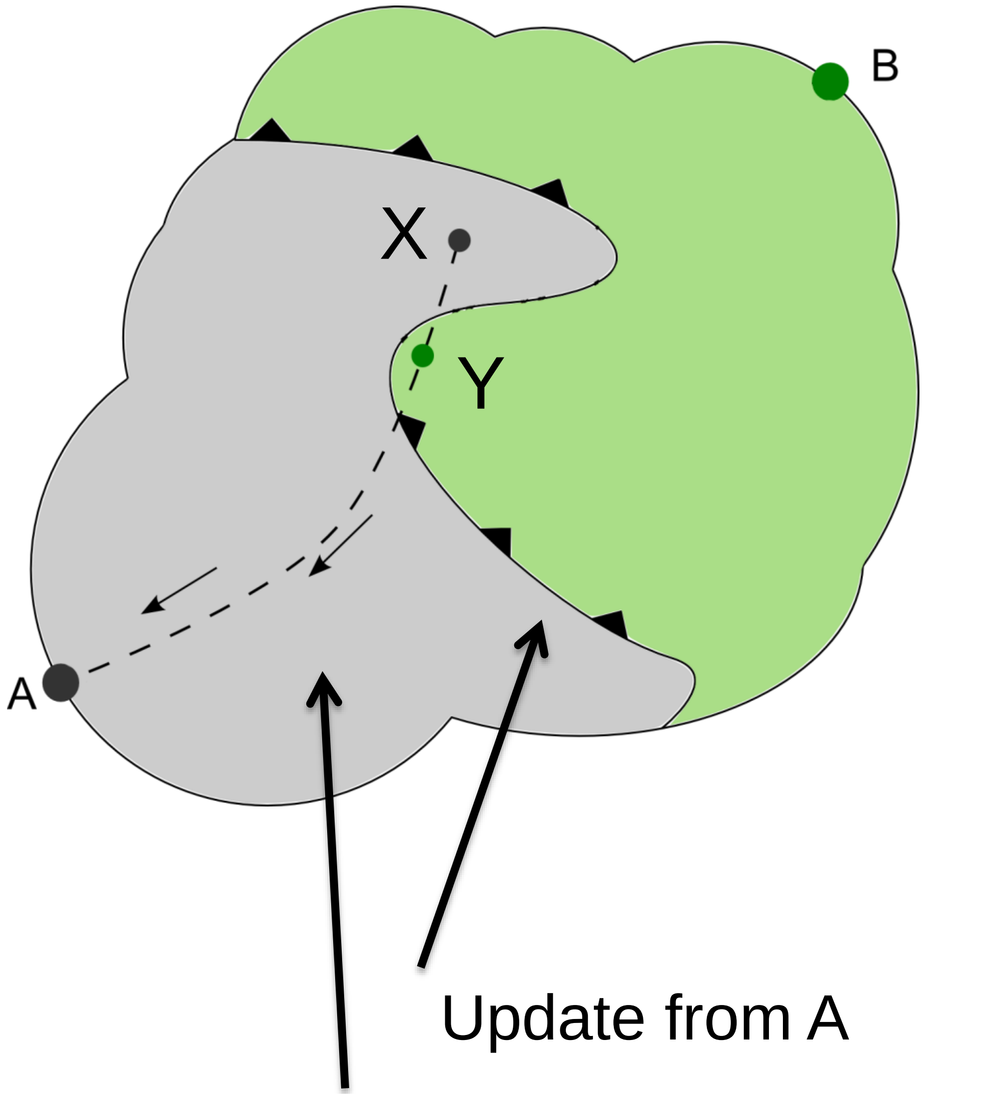
Update from A

Forwarding paths to A

Why did it loop?

- Only one prefix
- Initially only one route via B
- New better alternative at A
- Propagates as a wavefront
- Routers switch to A

X cannot reach A.
Y will forward back to X



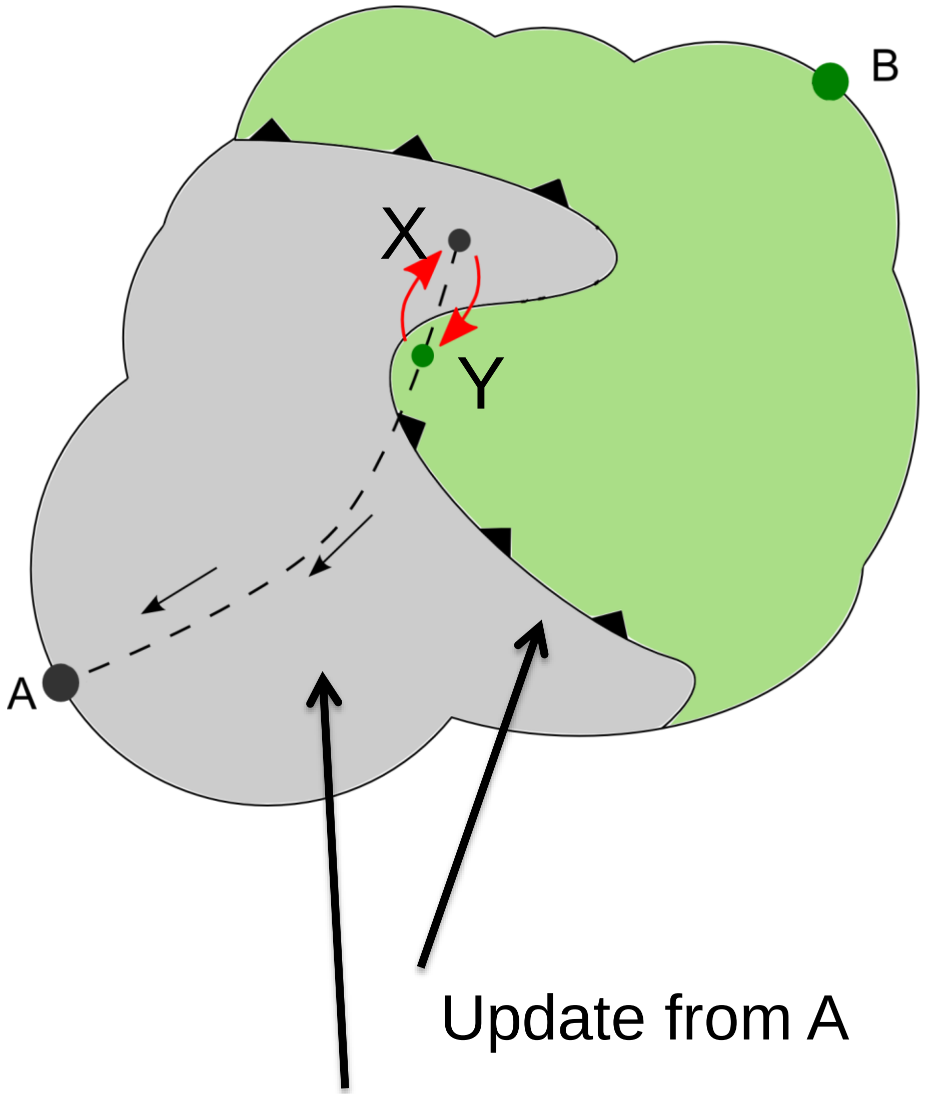
Update from A

Forwarding paths to A

Why did it loop?

- Only one prefix
- Initially only one route via B
- New better alternative at A
- Propagates as a wavefront
- Routers switch to A

X cannot reach A.
Y will forward back to X

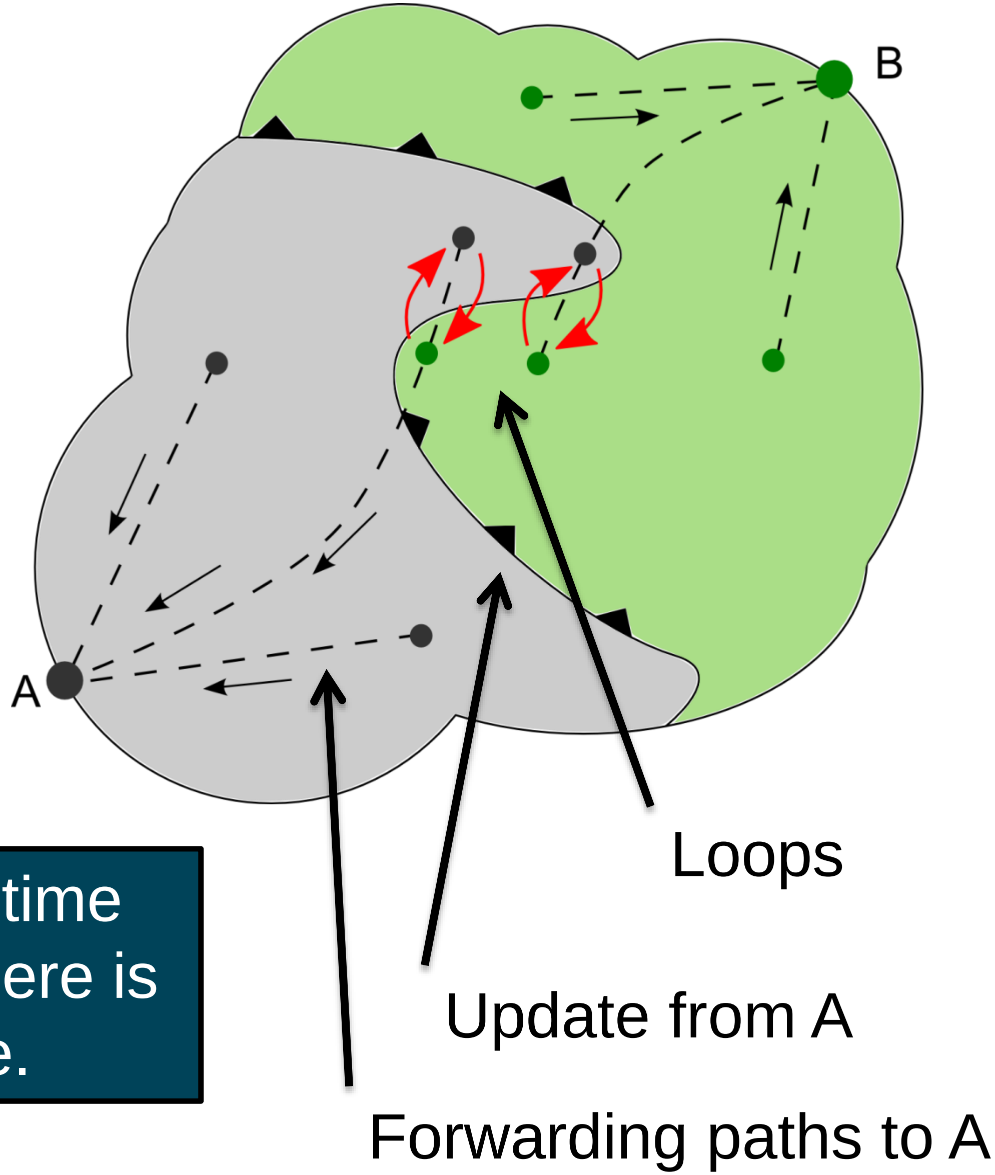


Update from A

Forwarding paths to A

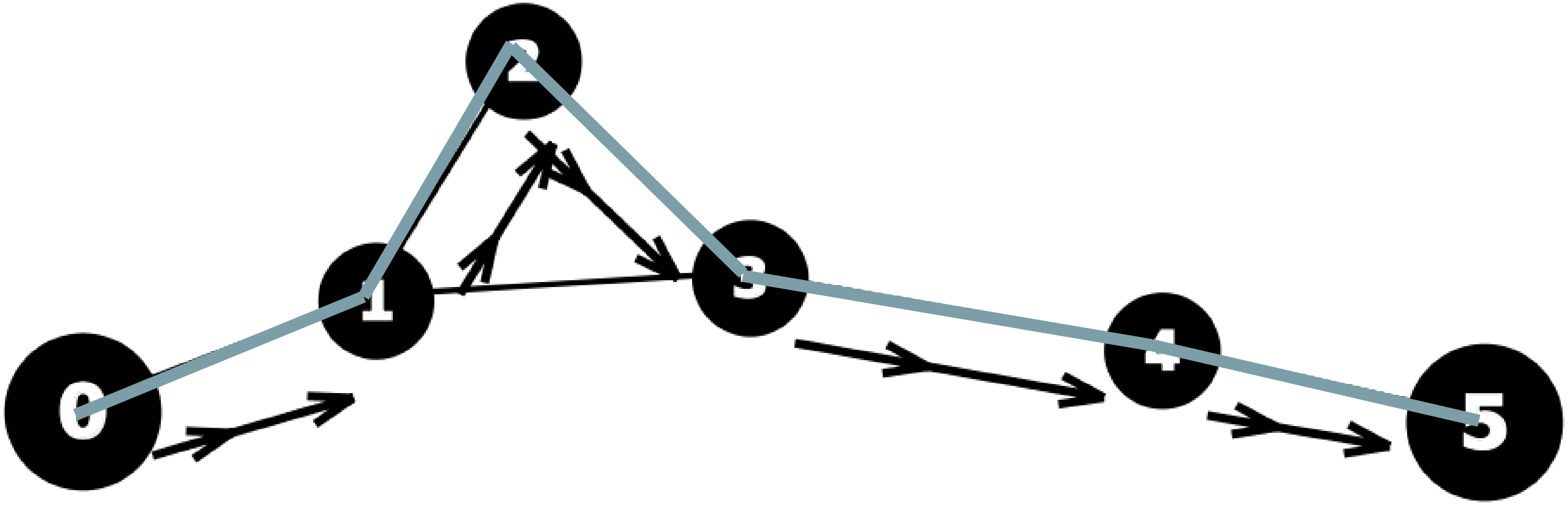
Why did it loop?

- Only one prefix
- Initially only one route via B
- New better alternative at A
- Propagates as a wavefront
- Routers switch to A
- Flooding not ordered
- Loops can form



Need to ensure that at any time along any forwarding path there is only one switch of route.

Reverse Forwarding Tree propagation avoids loops



To avoid loops, propagate over the concatenation of the forwarding paths to the BR.

SOUP ingredients

Wavefront propagation

- Basic ordering of updates

Reverse Forwarding Tree (RFT) and Forward Activation (FA)

- New / improving routes

Reverse Activation (RA)

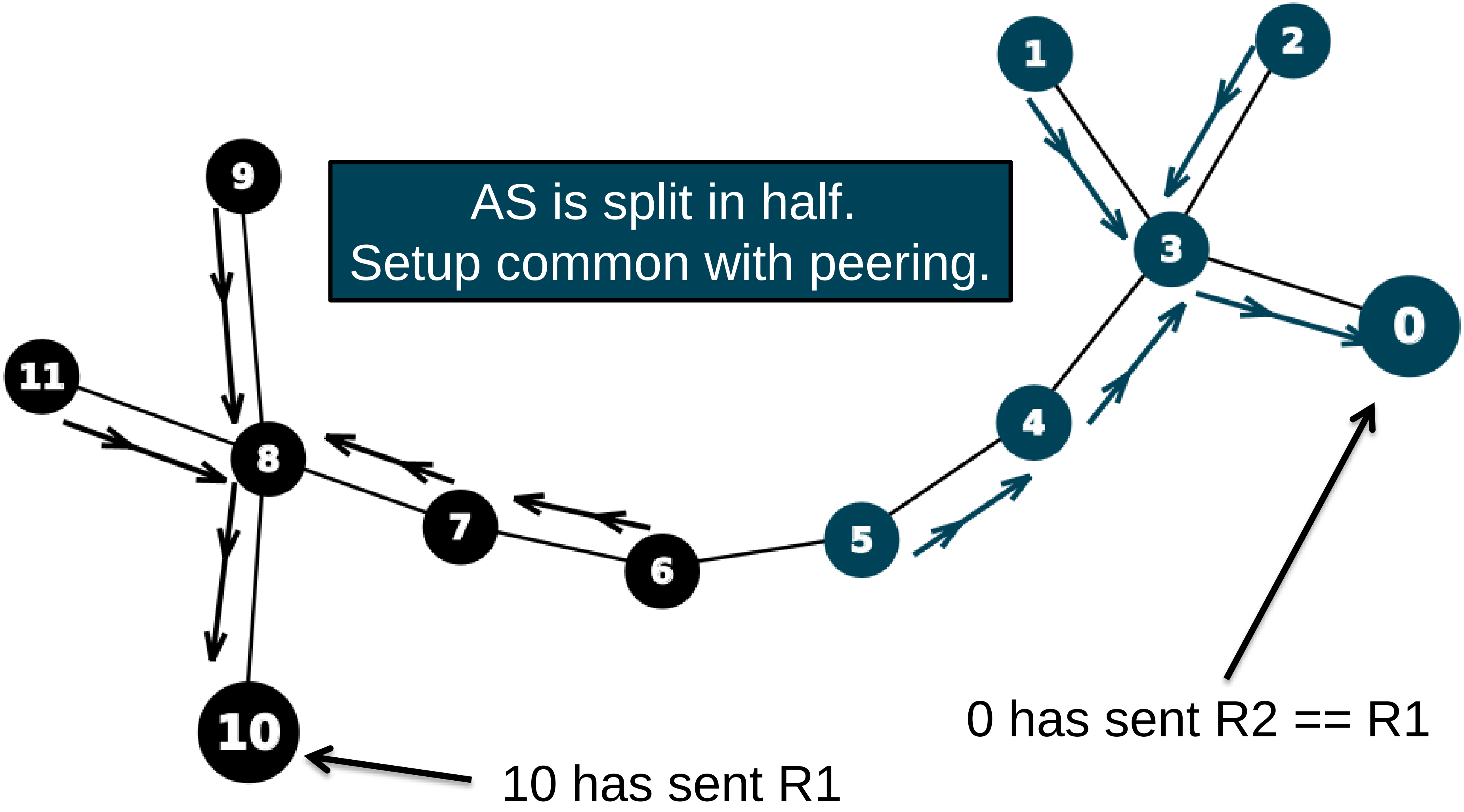
- Worsening routes / withdrawals

RA -> FA switch

- Multiple alternatives propagating simultaneously
- Complete loop freedom

What about withdrawals and routes worsening?

Immediate application of withdrawals causes loops

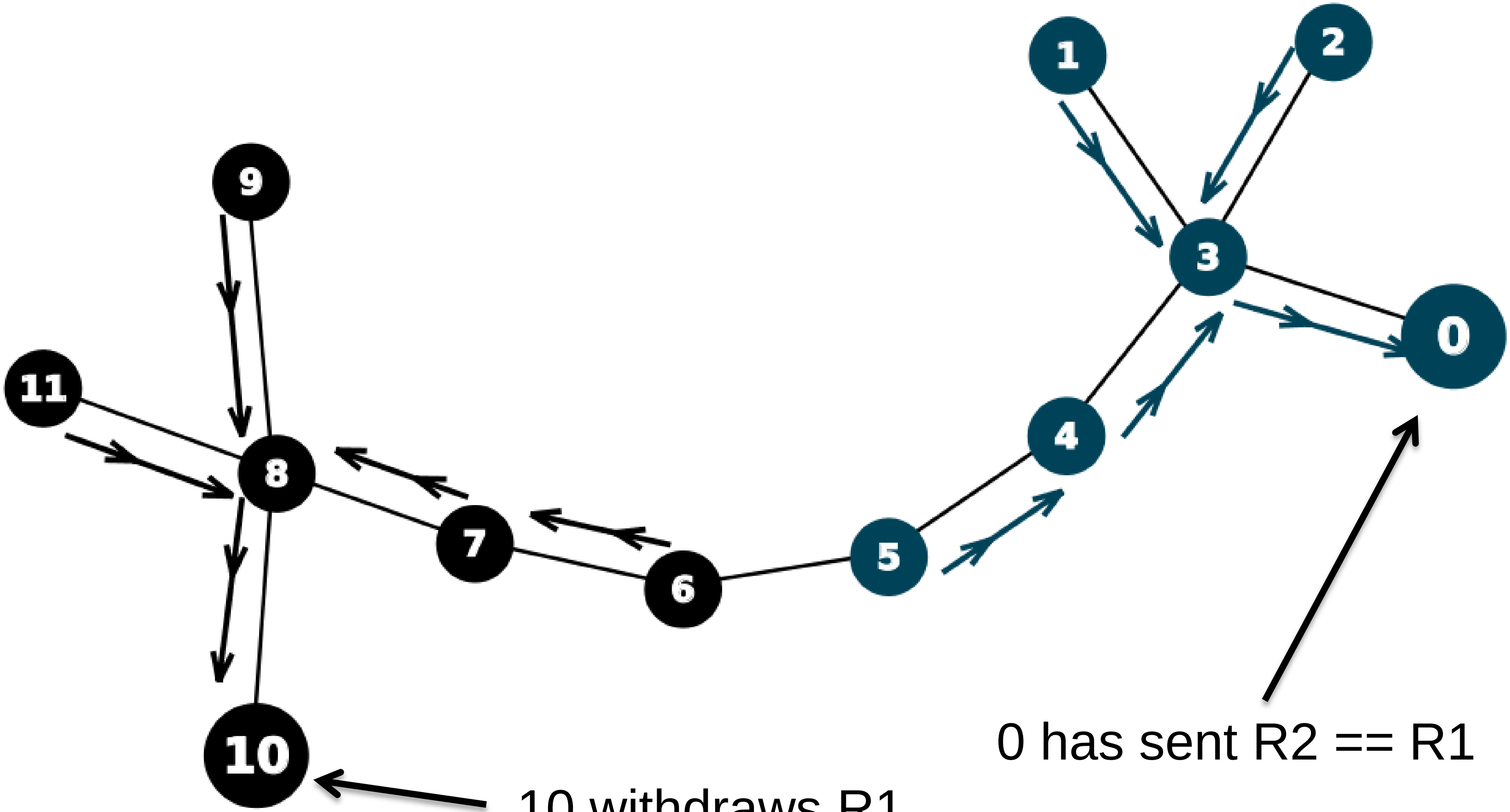


AS is split in half.
Setup common with peering.

0 has sent R2 == R1

10 has sent R1

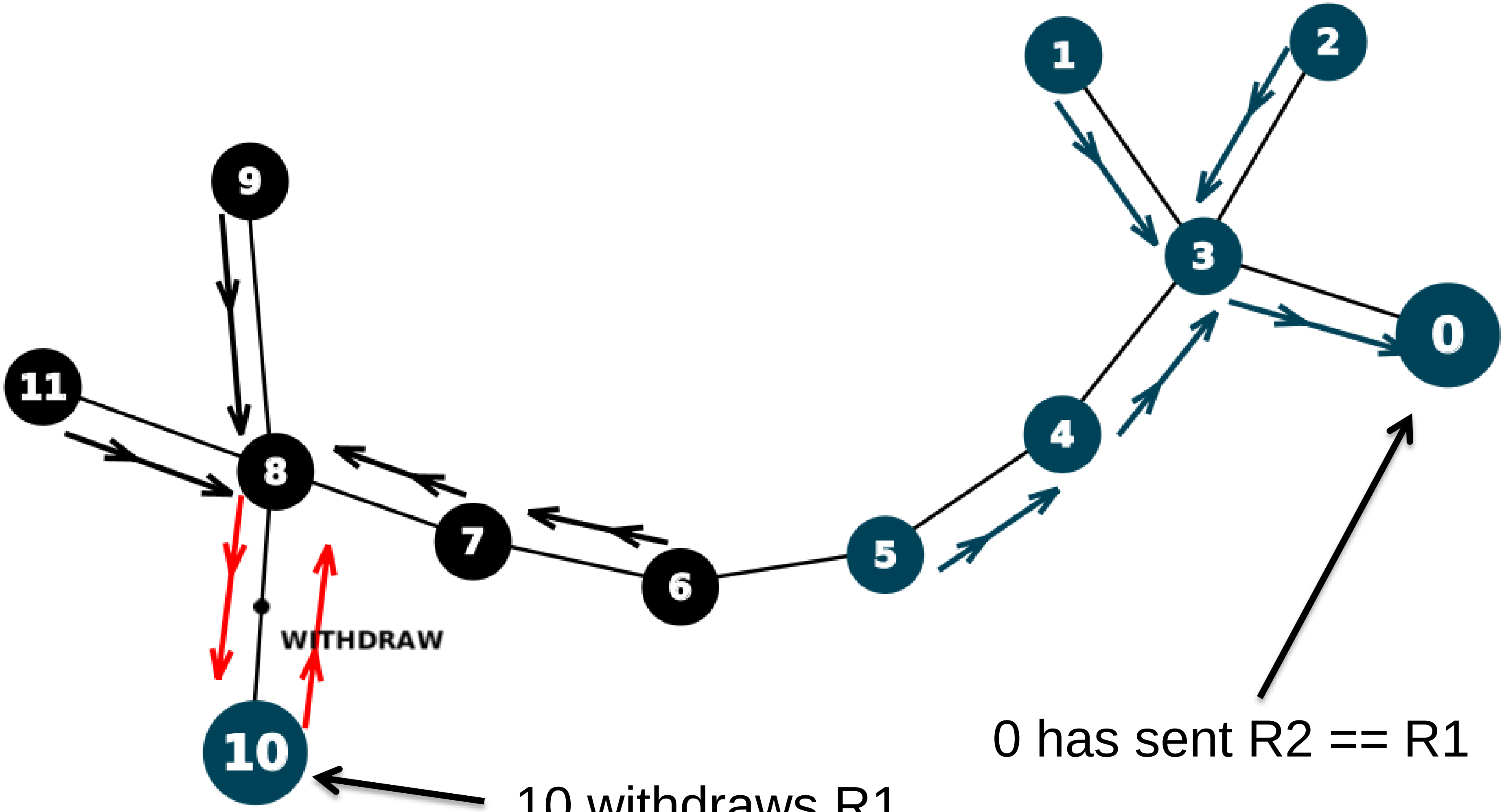
Immediate application of withdrawals causes loops



0 has sent R2 == R1

10 withdraws R1

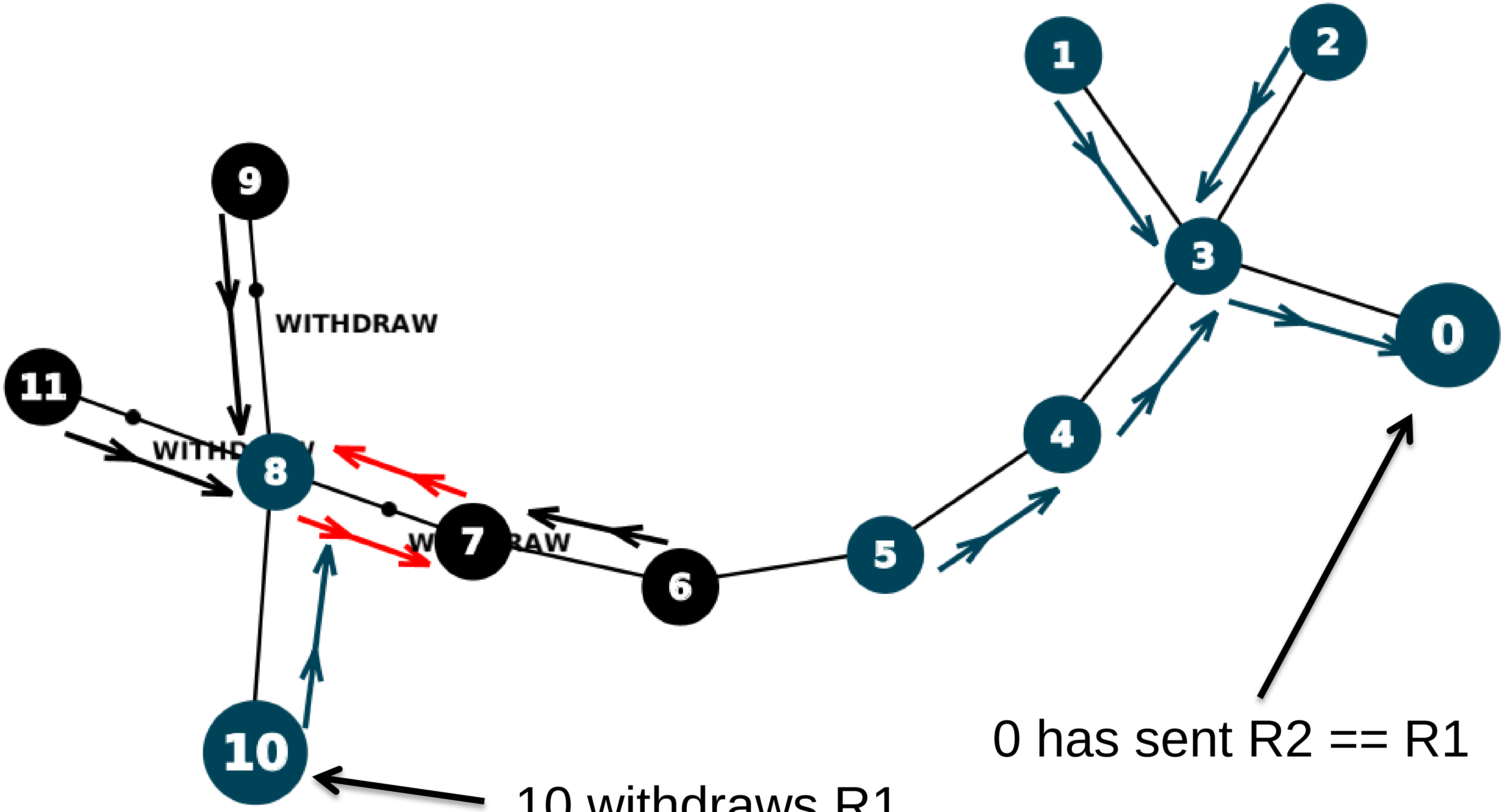
Immediate application of withdrawals causes loops



0 has sent R2 == R1

10 withdraws R1

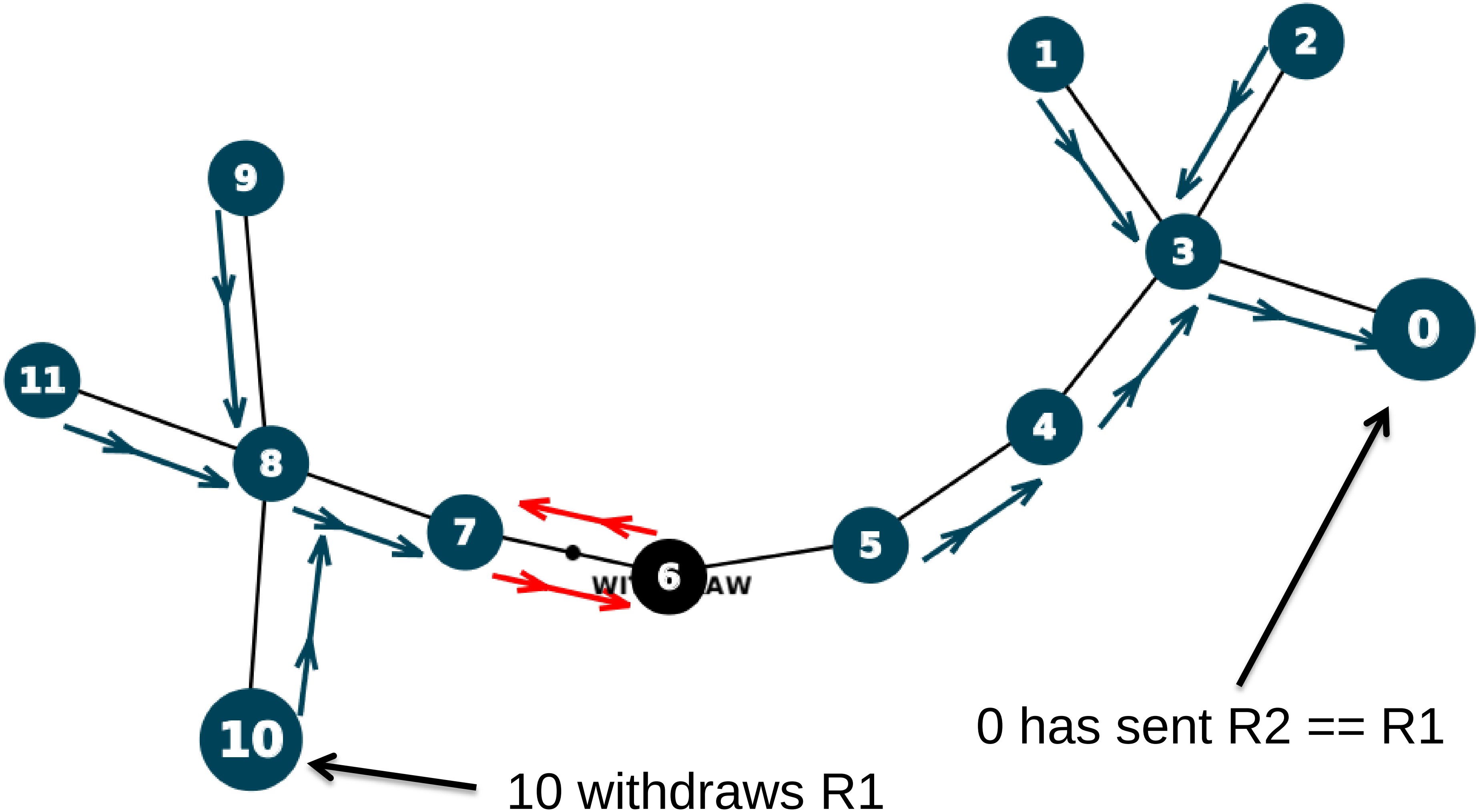
Immediate application of withdrawals causes loops



0 has sent R2 == R1

10 withdraws R1

Immediate application of withdrawals causes loops

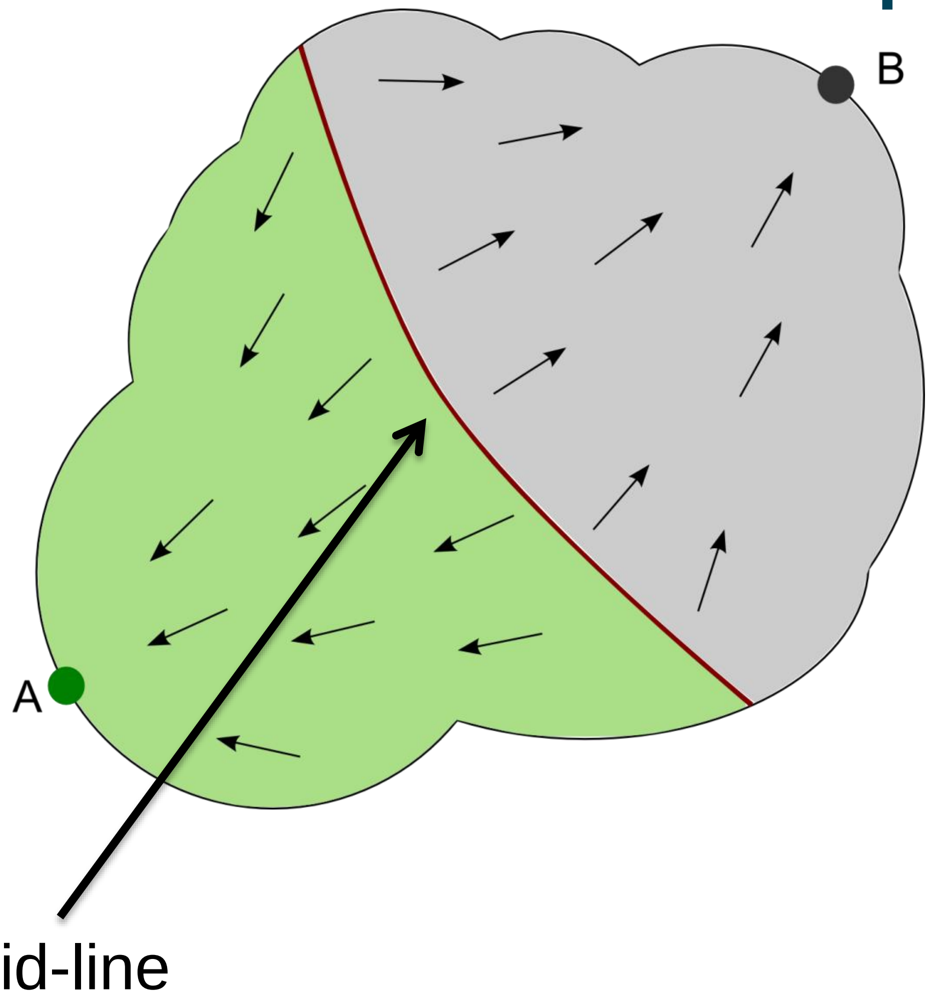


0 has sent R2 == R1

10 withdraws R1

Immediate application of withdrawals causes loops

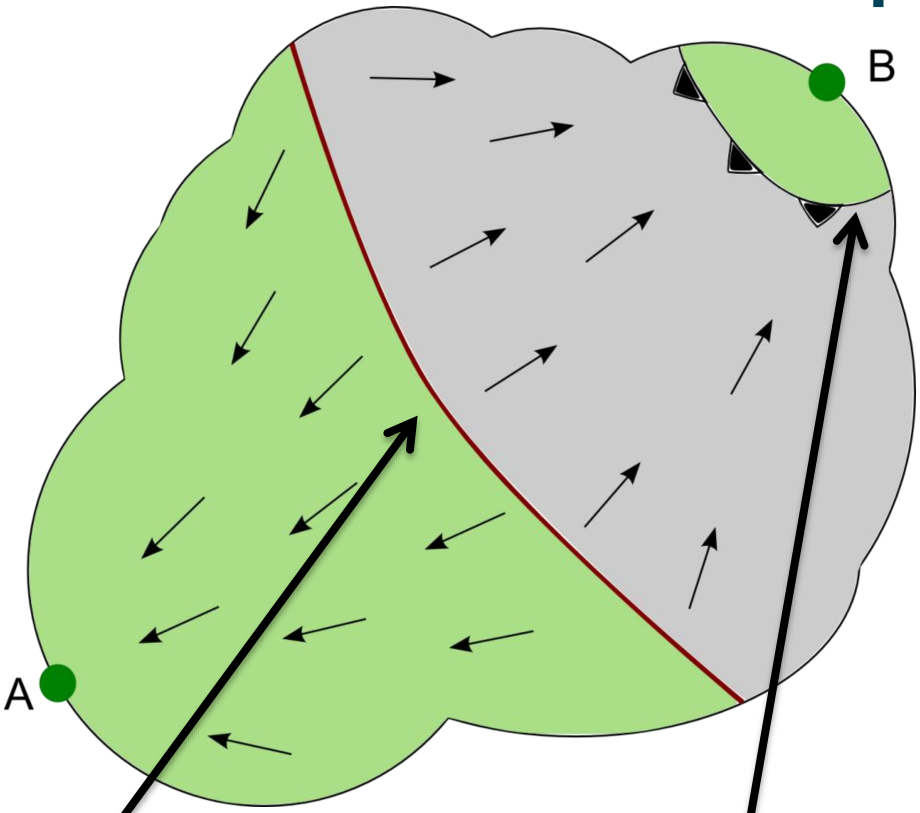
- More than one “best” route
- BGP splits the AS in two



Immediate application of withdrawals causes loops

- More than one “best” route
- BGP splits the AS in two

- B withdraws its route
- Withdrawal as a wavefront

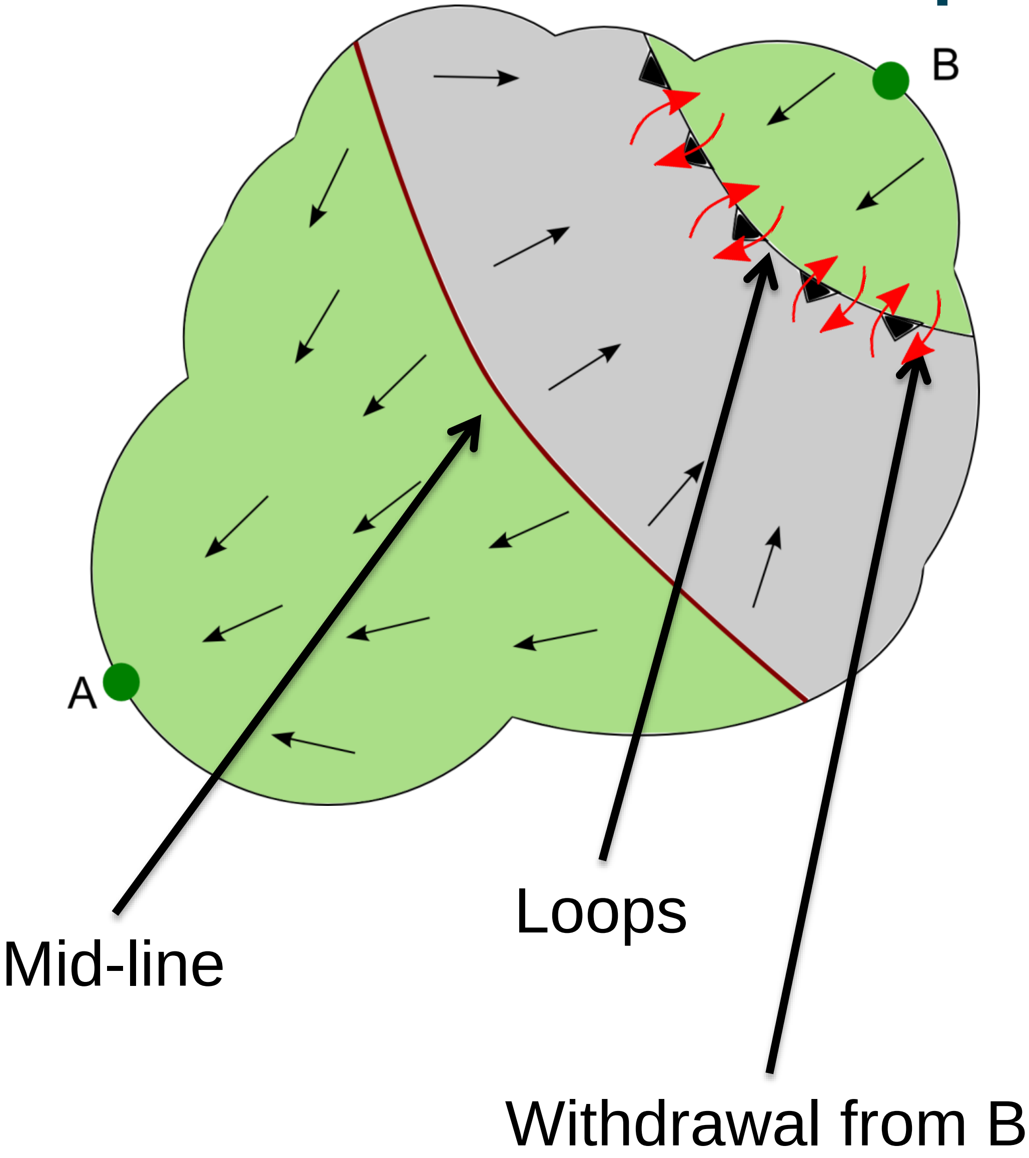


Mid-line

Withdrawal from B

Immediate application of withdrawals causes loops

- More than one “best” route
- BGP splits the AS in two
- B withdraws its route
- Withdrawal as a wavefront
- A wave of transient loops



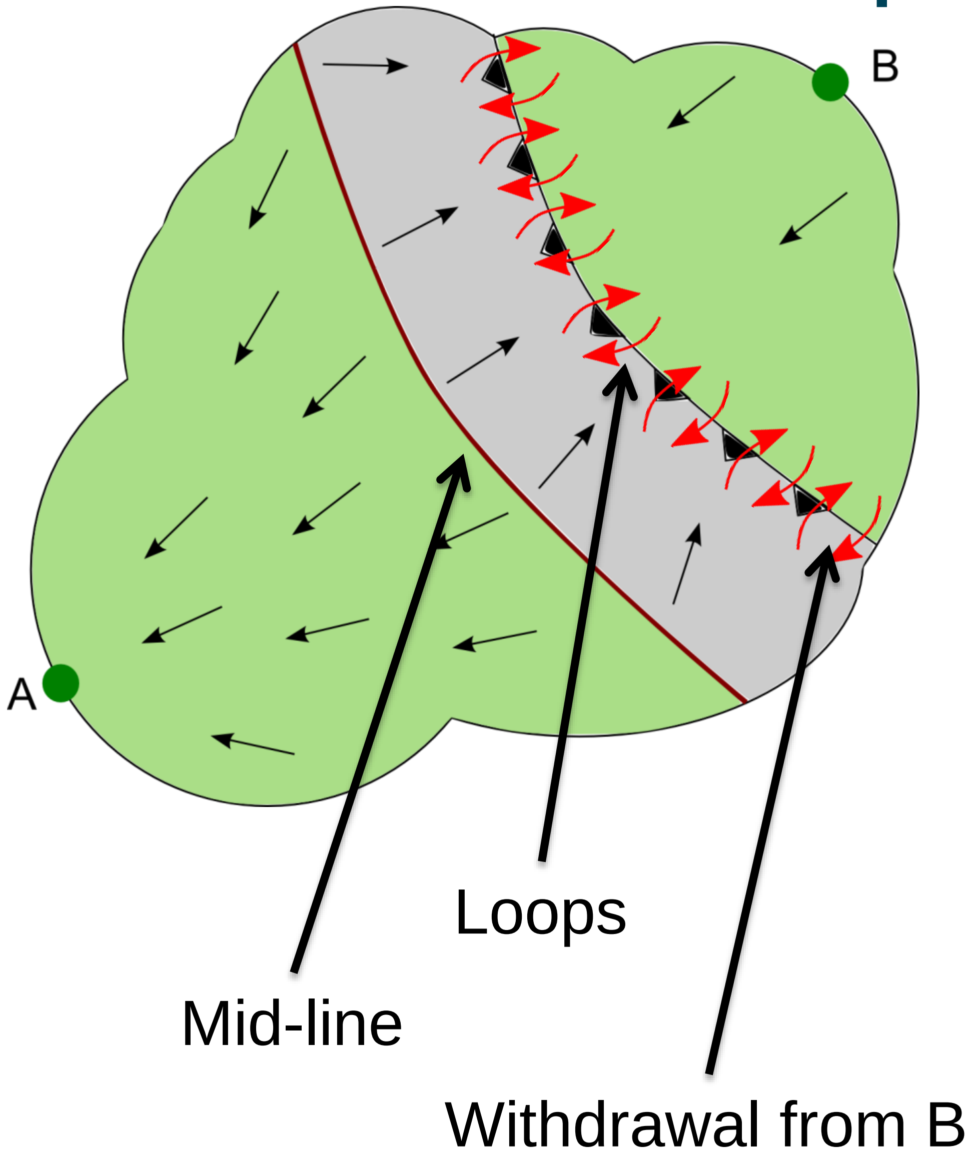
Immediate application of withdrawals causes loops

- More than one “best” route
- BGP splits the AS in two

- B withdraws its route
- Withdrawal as a wavefront

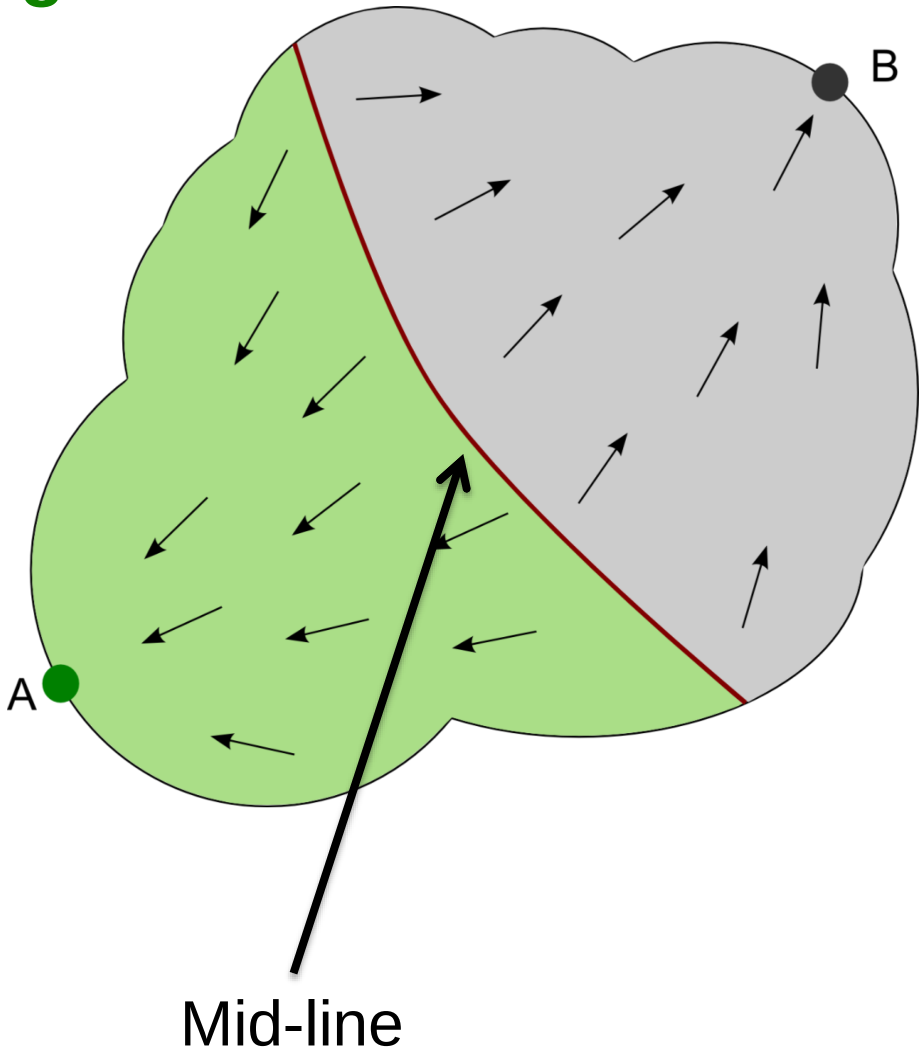
- A wave of transient loops
- Until the mid-line

How can we fix it?



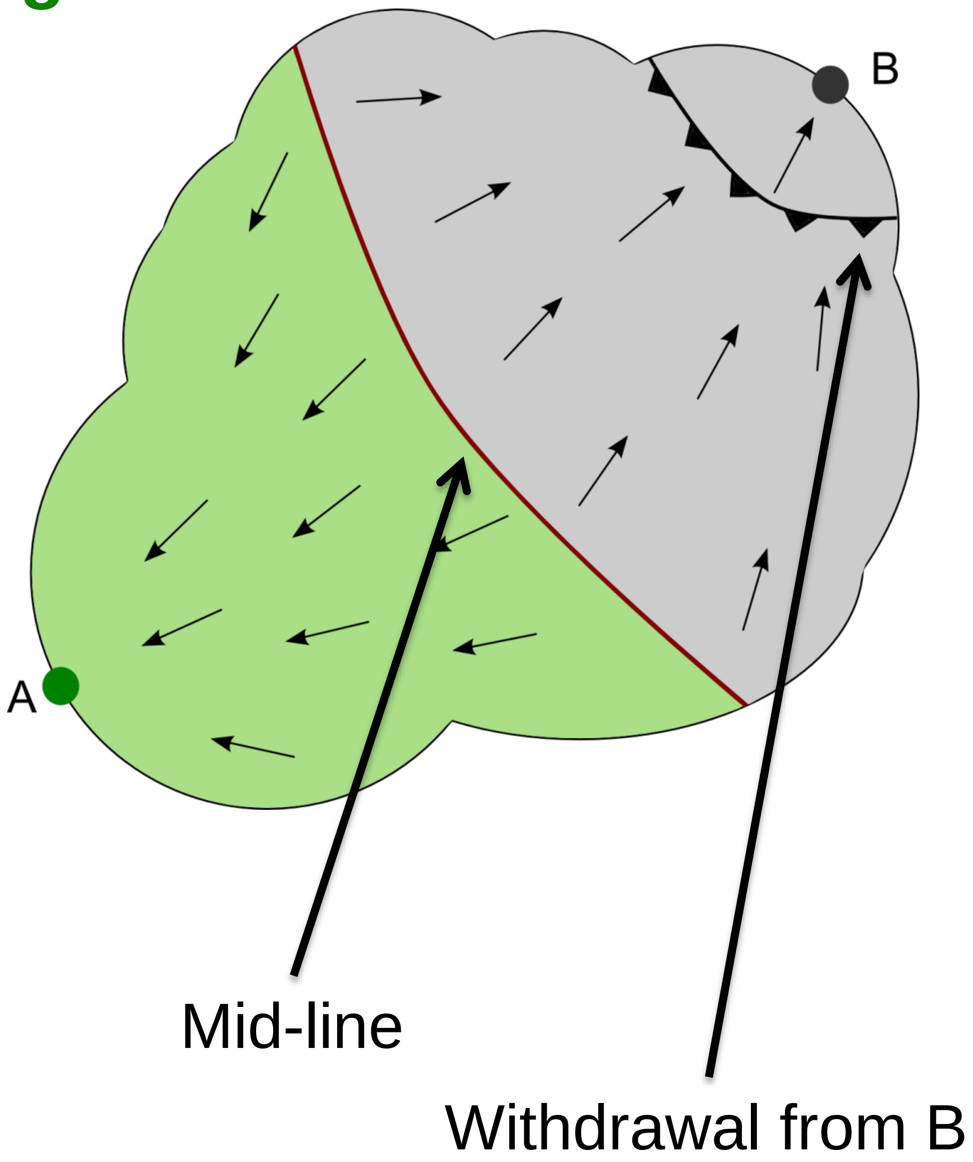
Withdrawal order done right

- Initially do not apply the withdrawal



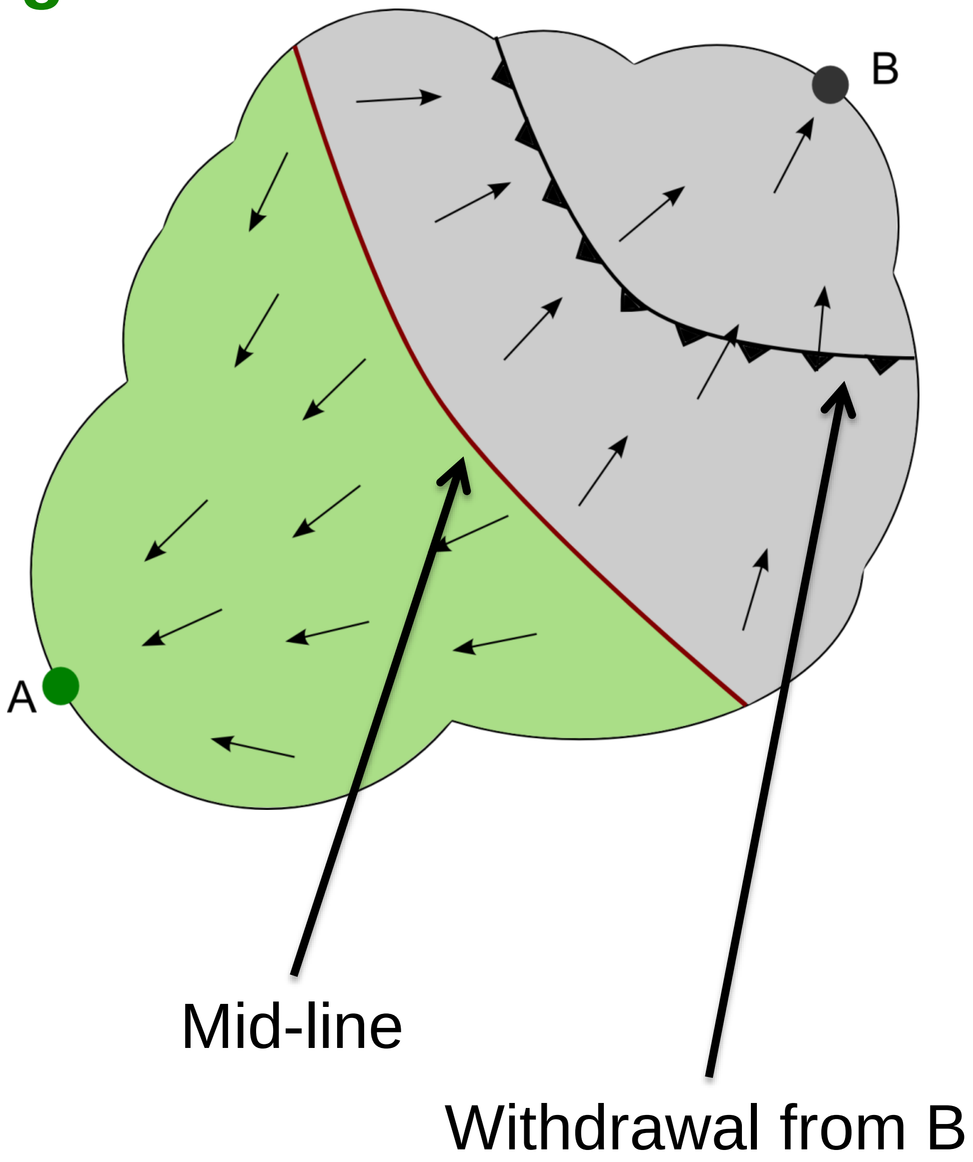
Withdrawal order done right

- Initially do not apply the withdrawal



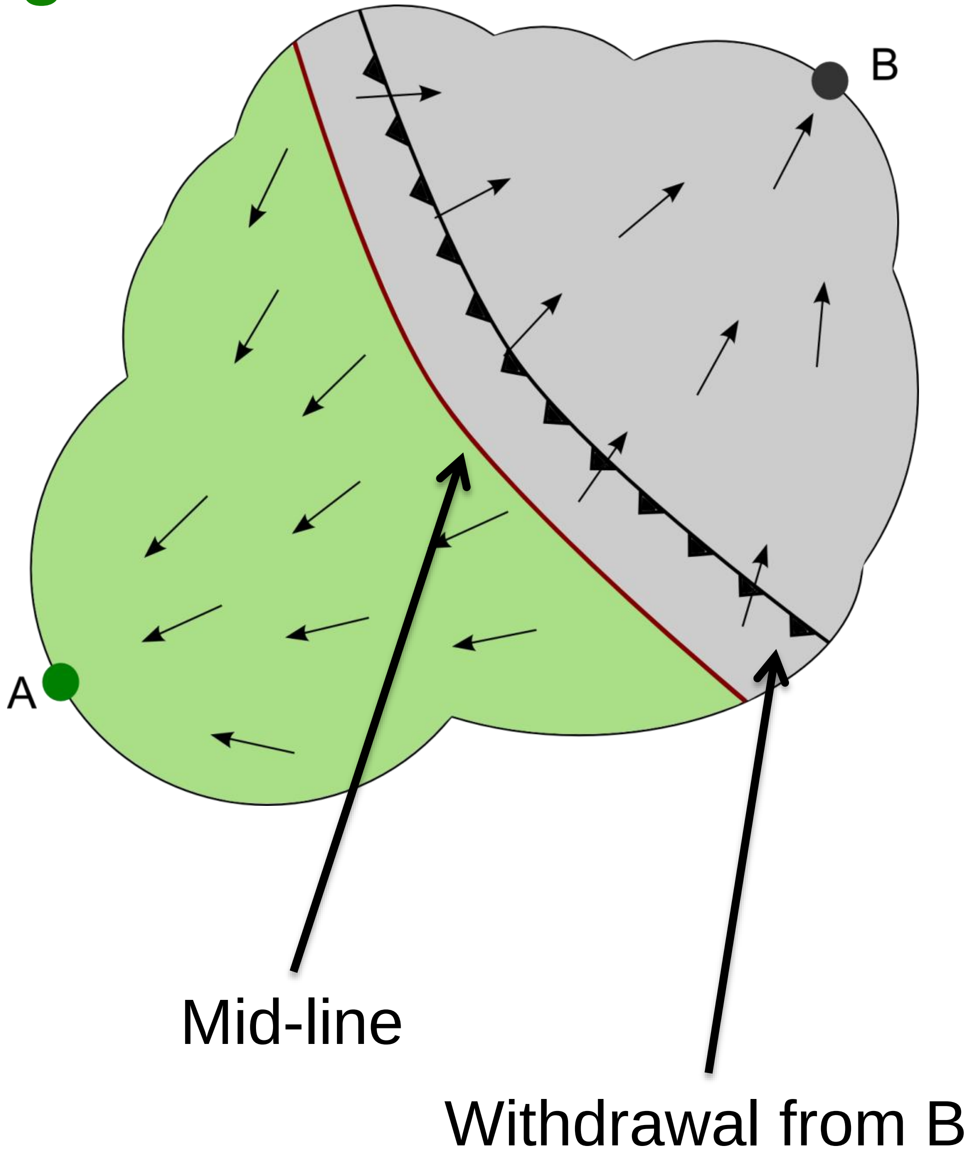
Withdrawal order **done right**

- Initially do not apply the withdrawal



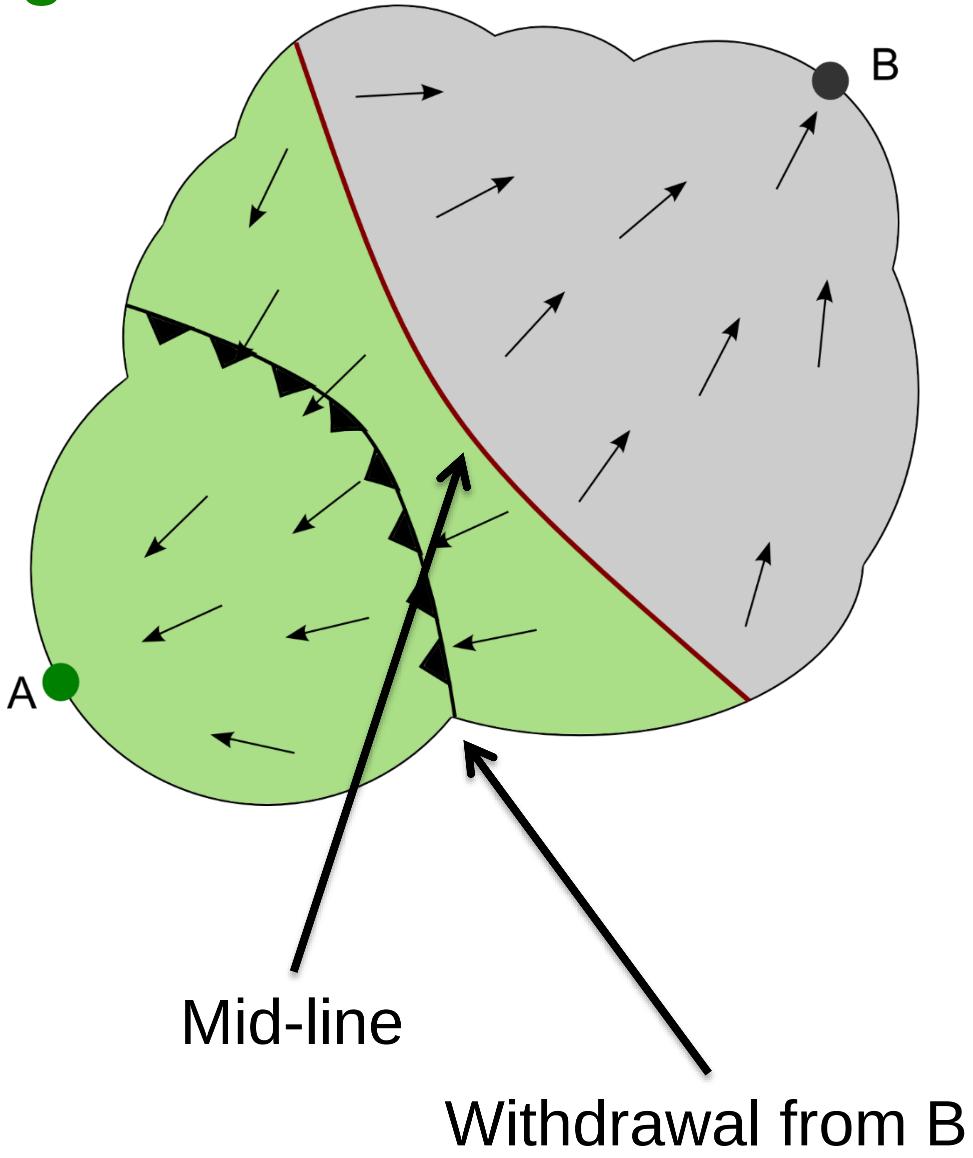
Withdrawal order **done right**

- Initially do not apply the withdrawal



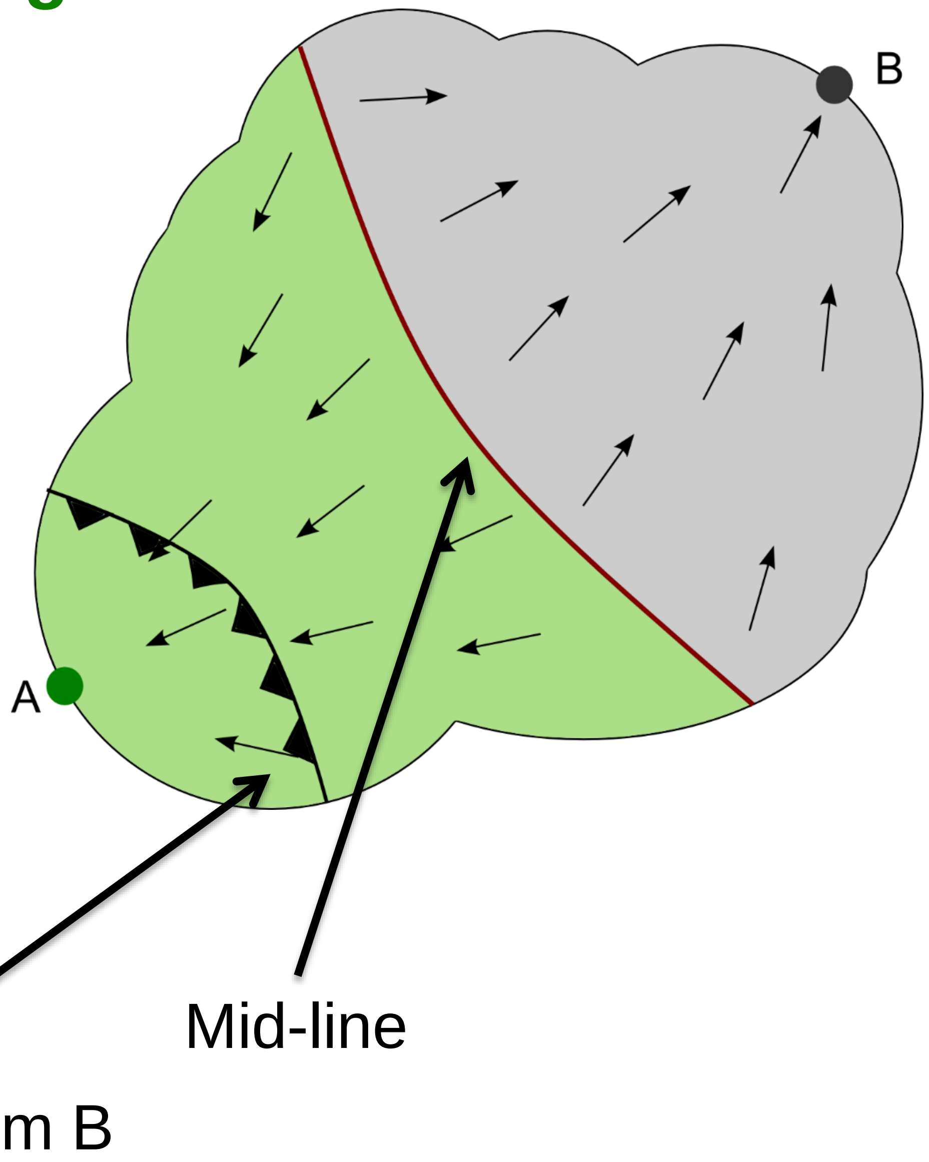
Withdrawal order done right

- Initially do not apply the withdrawal



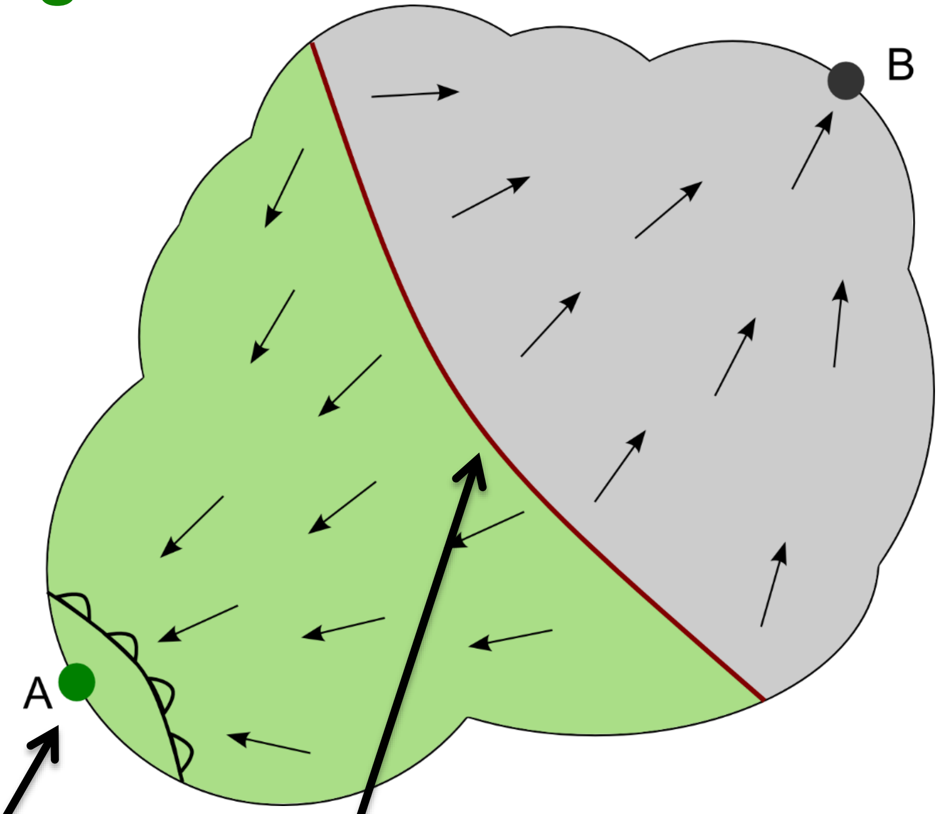
Withdrawal order done right

- Initially do not apply the withdrawal



Withdrawal order **done right**

- Initially do not apply the withdrawal
- Apply over the reverse of update propagation path

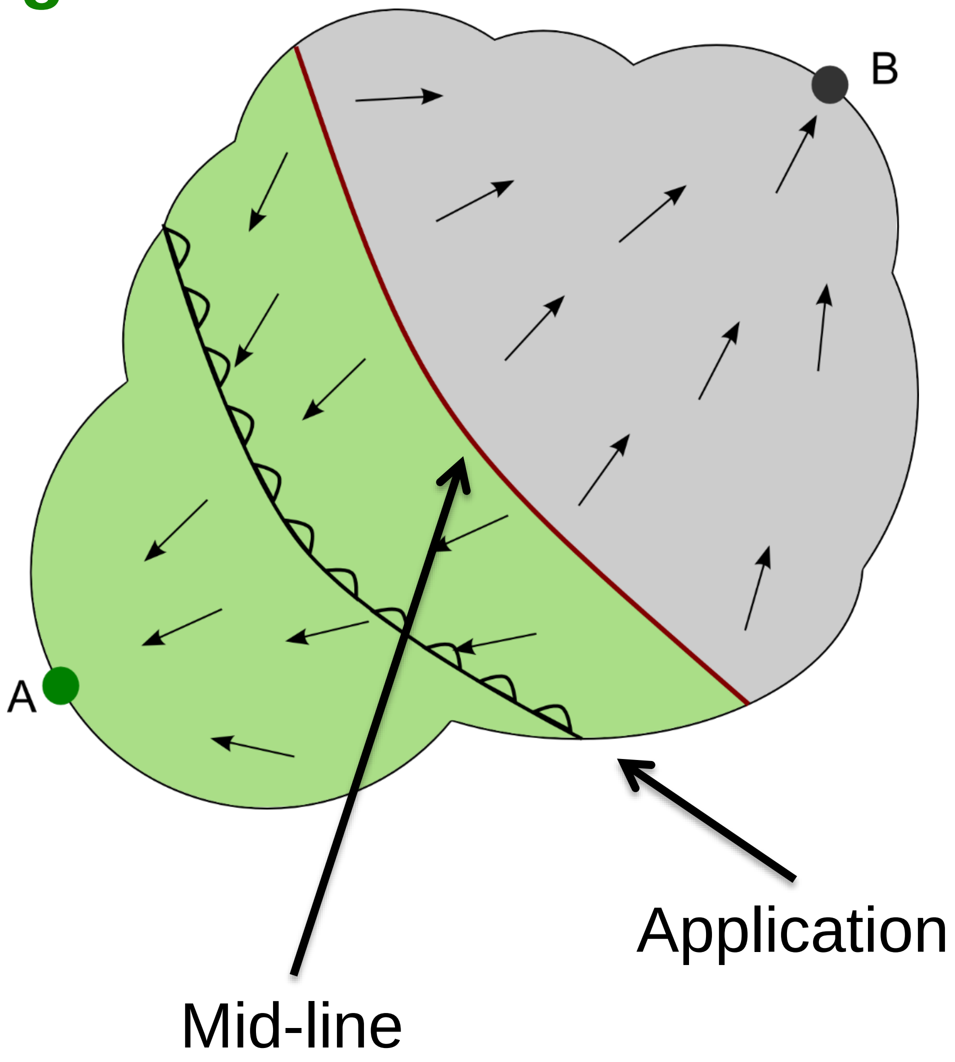


Mid-line

Application of the withdrawal sent by A

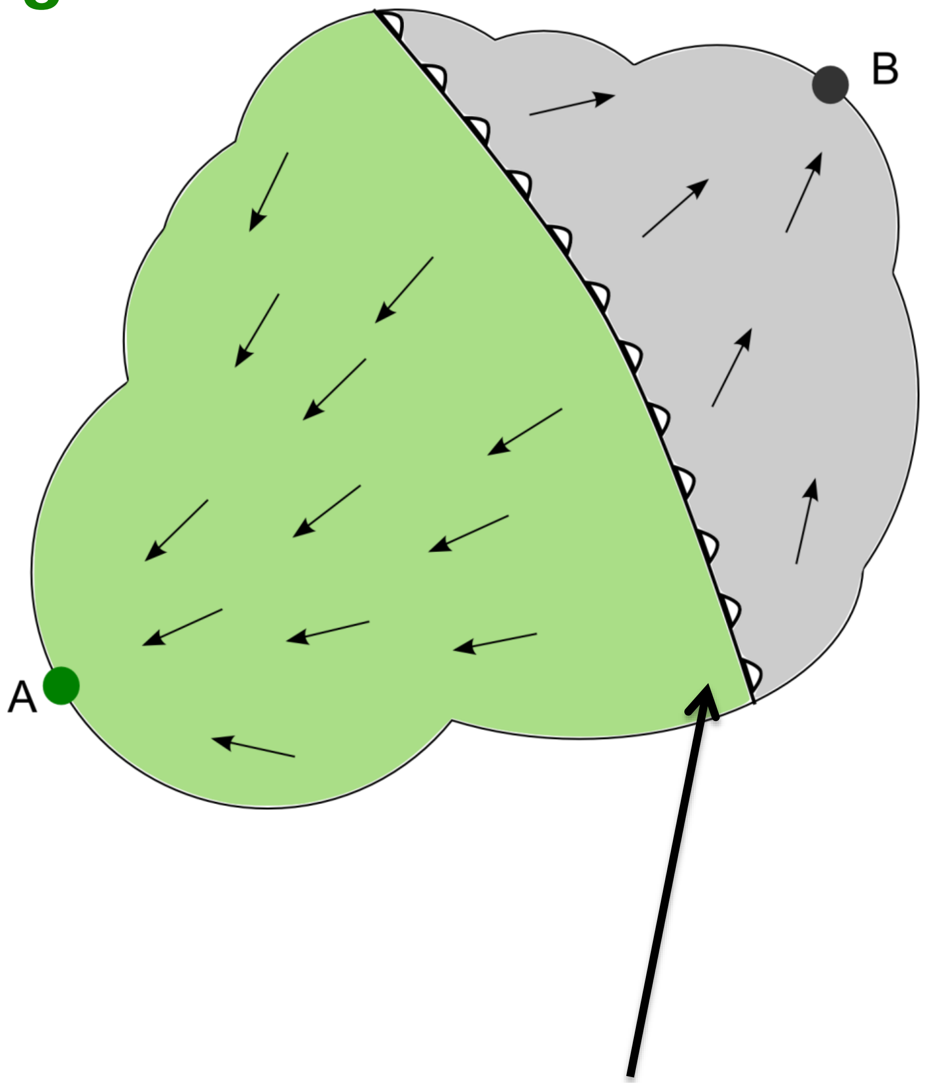
Withdrawal order **done right**

- Initially do not apply the withdrawal
- Apply over the reverse of update propagation path



Withdrawal order done right

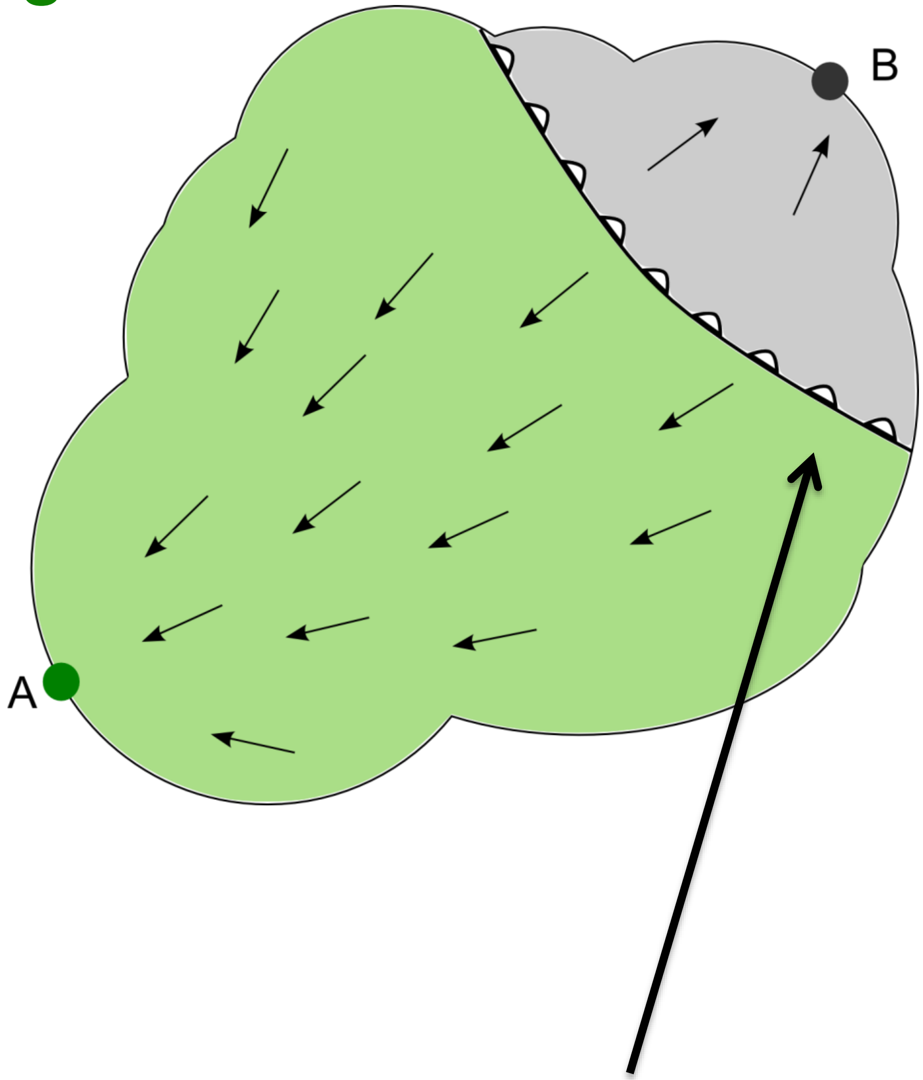
- Initially do not apply the withdrawal
- Apply over the reverse of update activation path



Application of withdrawal

Withdrawal order done right

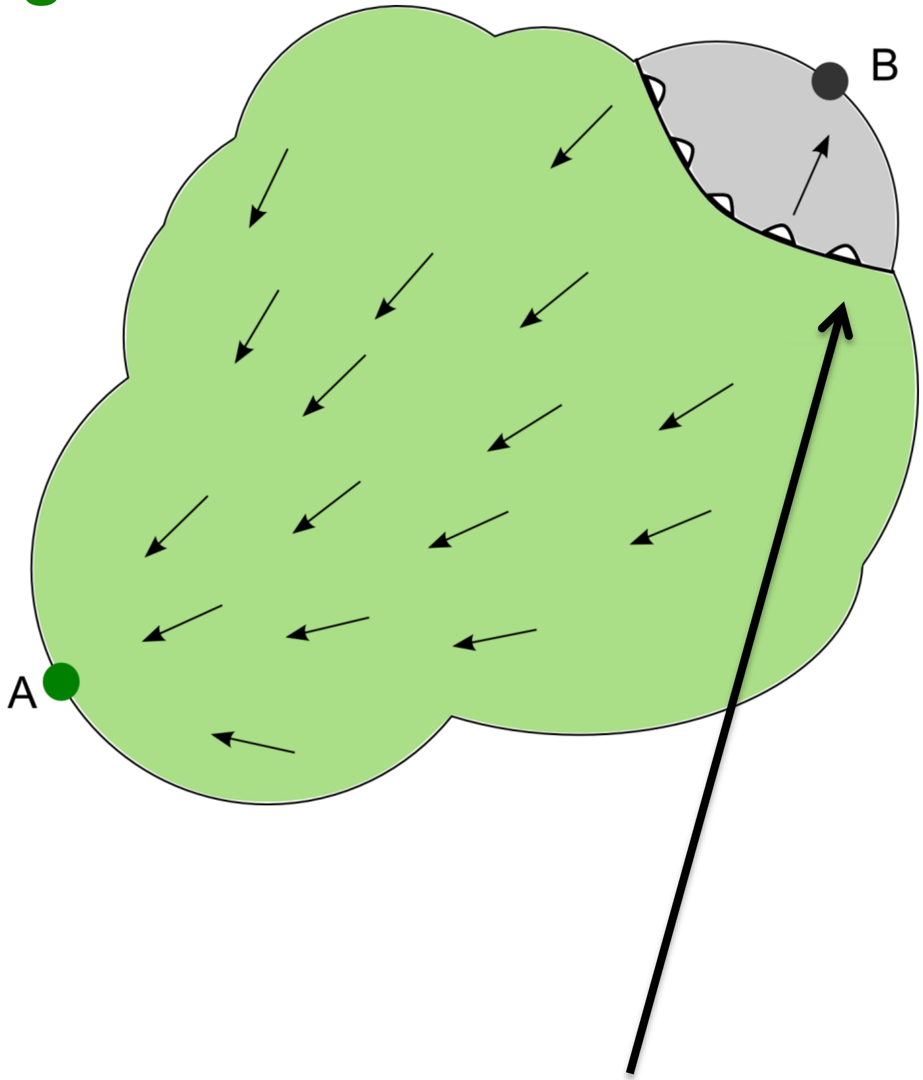
- Initially do not apply the withdrawal
- Apply over the reverse of update activation path



Application of withdrawal

Withdrawal order done right

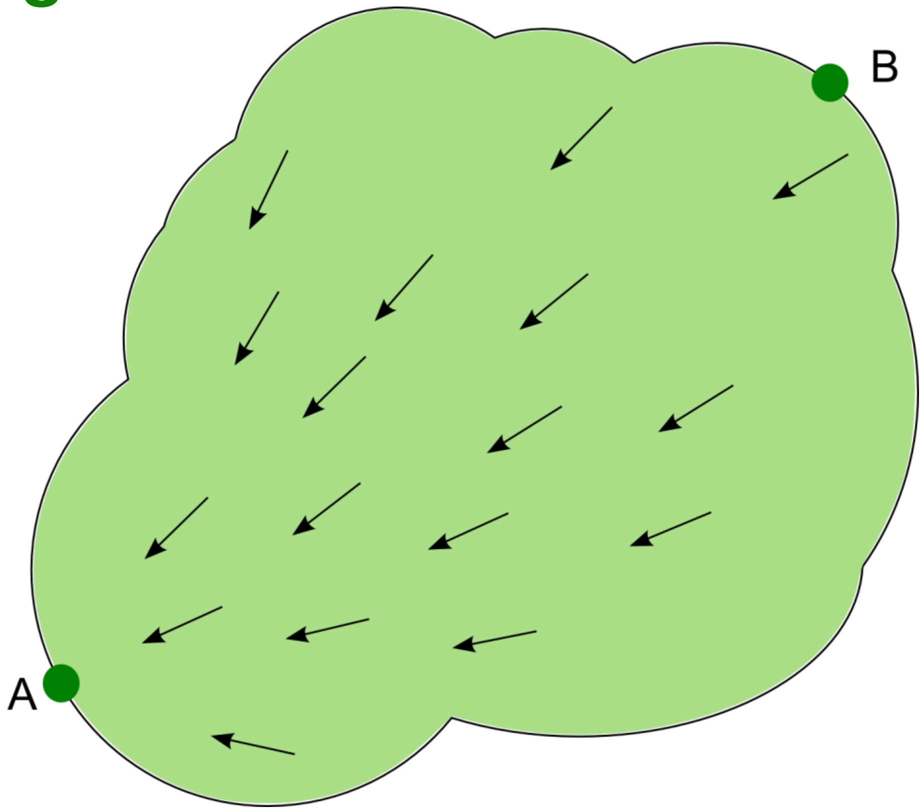
- Initially do not apply the withdrawal
- Apply over the reverse of update activation path



Application of withdrawal

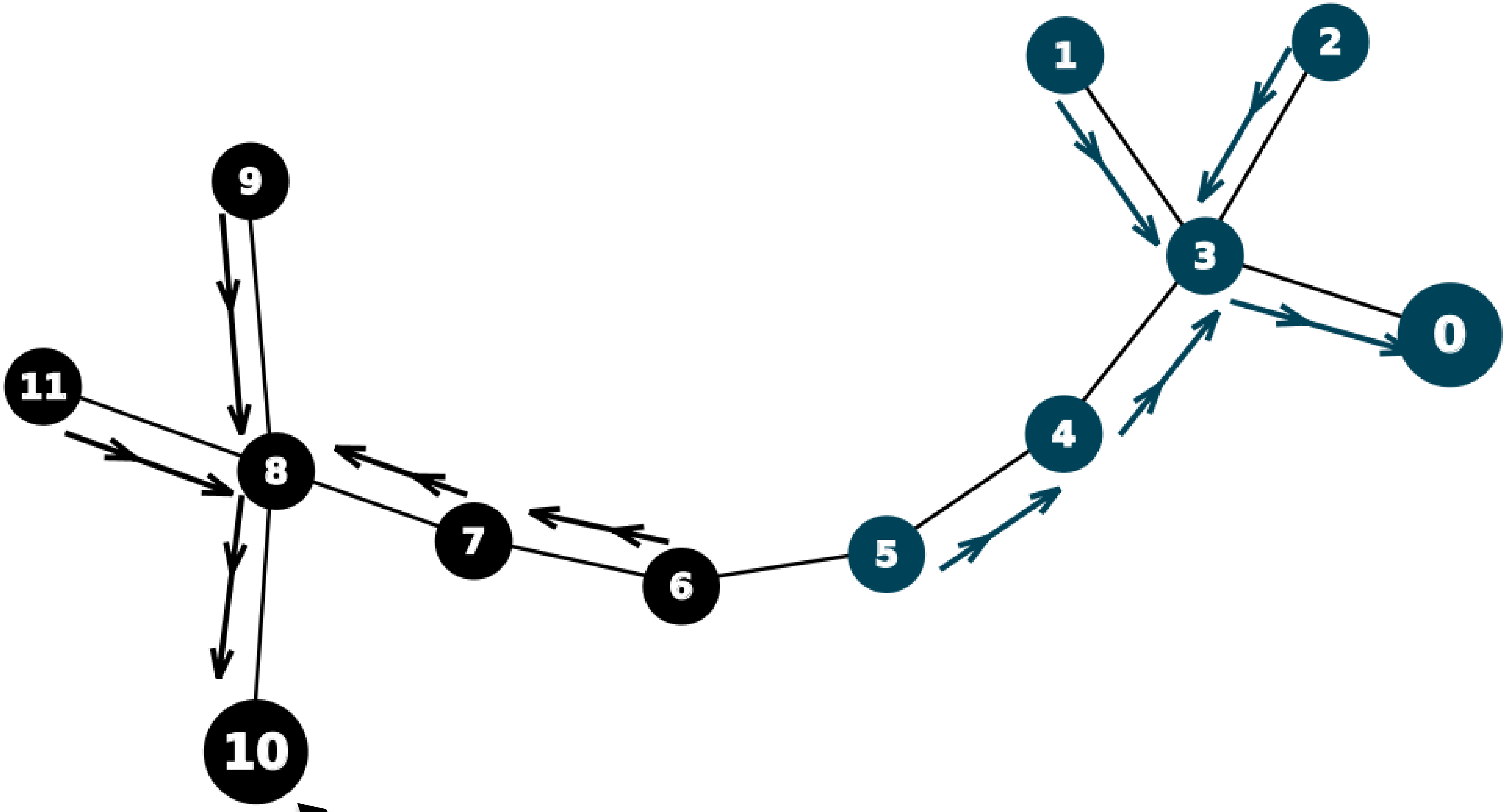
Withdrawal order **done right**

- Initially do not apply the withdrawal
- Apply over the reverse of update activation path



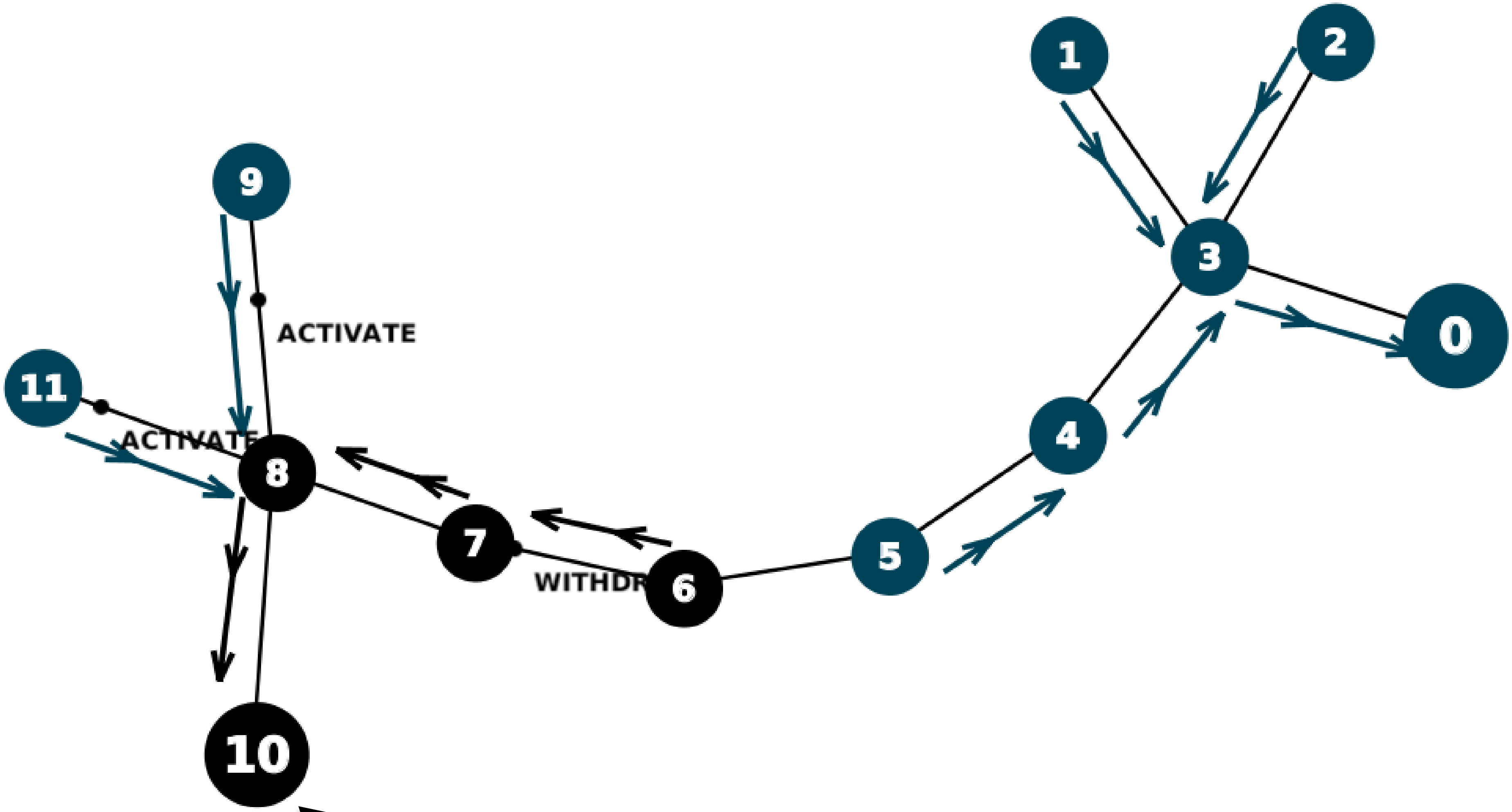
SOUP uses **reverse activation** to explicitly apply worsening routes

Reverse activation example



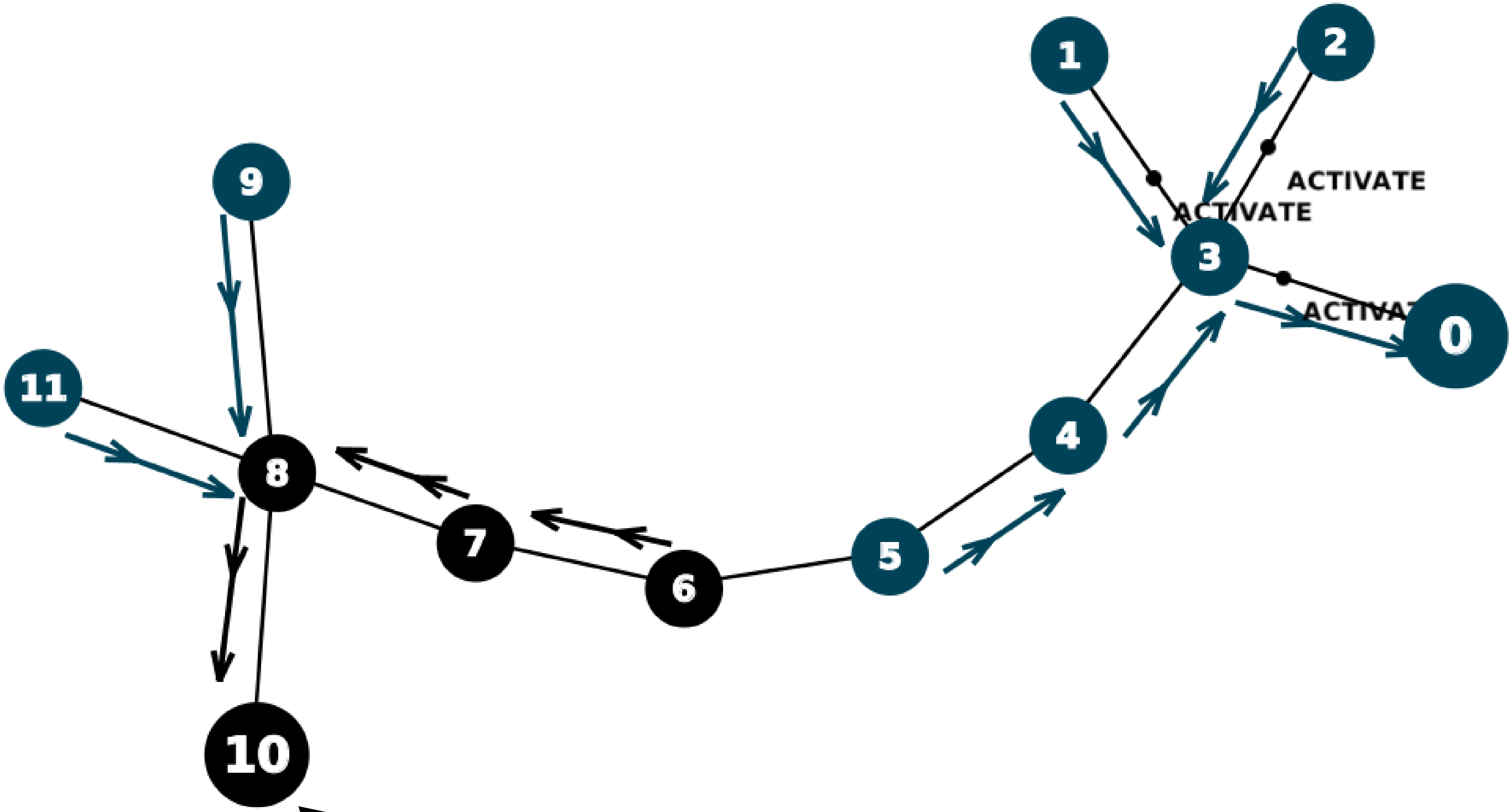
10 withdraws R1

Reverse activation example



10 withdraws R1

Reverse activation example



10 withdraws R1

SOUP ingredients

Wavefront propagation

- Basic ordering of updates

Reverse Forwarding Tree (RFT) and Forward Activation (FA)

- New / improving routes

Reverse Activation (RA)

- Worsening routes / withdrawals

RA -> FA switch

- Multiple alternatives propagating simultaneously
- Complete loop freedom

SOUP ingredients

Wavefront propagation

- Basic ordering of updates

Reverse Forwarding Tree (RFT) and Forward Activation (FA)

- New / improving routes

Reverse Activation (RA)

- Worsening routes / withdrawals

RA -> FA switch (in paper)

- Multiple alternatives propagating simultaneously
- Complete loop freedom

Last ingredient in paper

SOUP ingredients

Wavefront propagation

- Basic ordering of updates

Reverse Forwarding Tree (RFT) and Forward Activation (FA)

SOUP is provably loop-free at all instants if the internal topology is stable. Proof in paper.

Reverse Activation (RA)

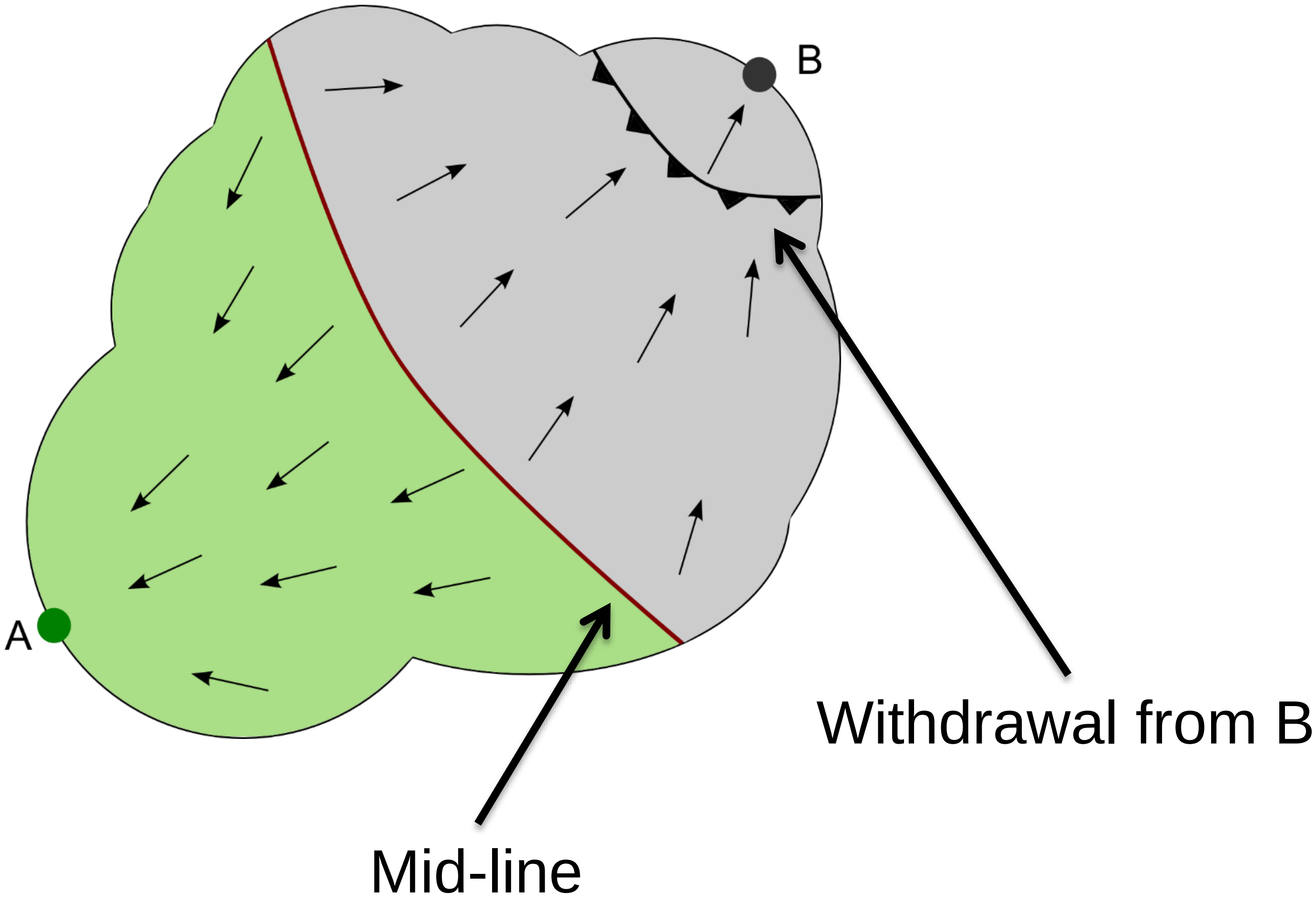
- Worsening routes / withdrawals

RA -> FA switch (in paper)

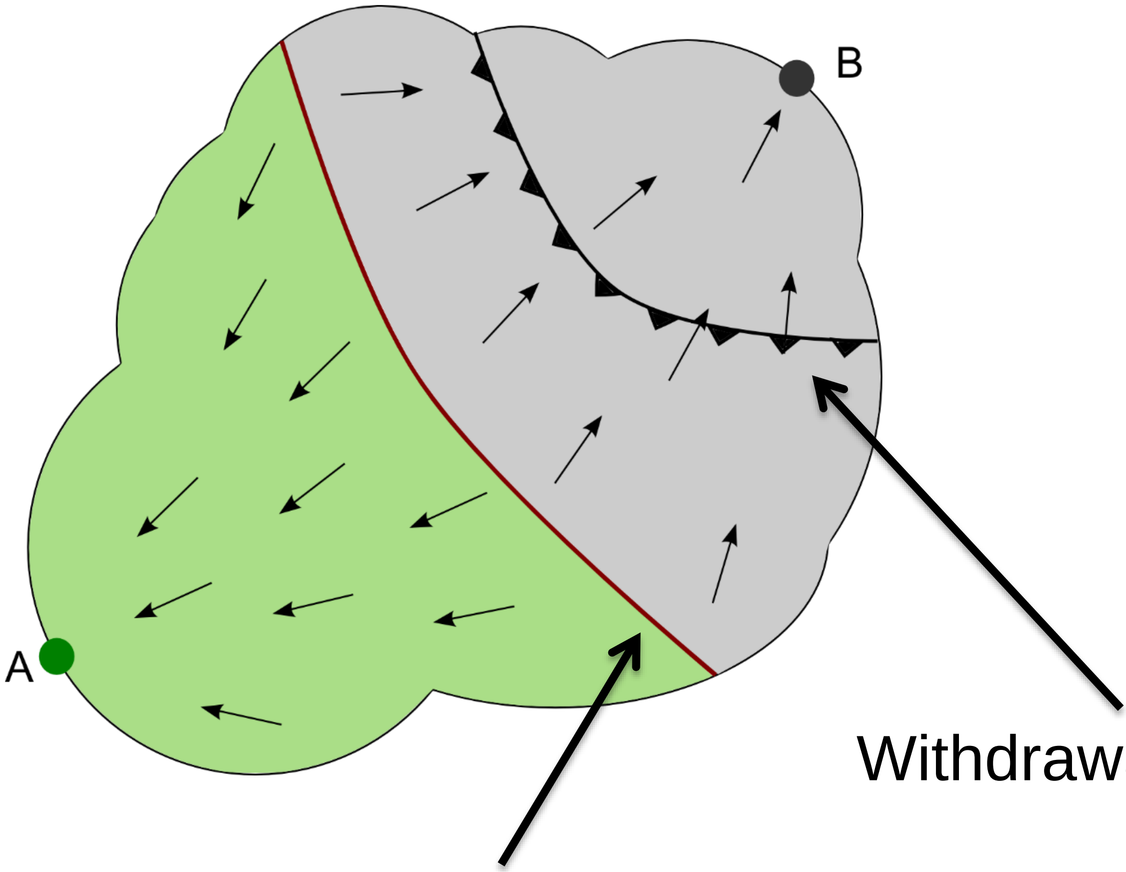
- Multiple alternatives propagating simultaneously
- Complete loop freedom

Last ingredient in paper

A fly in my SOUP



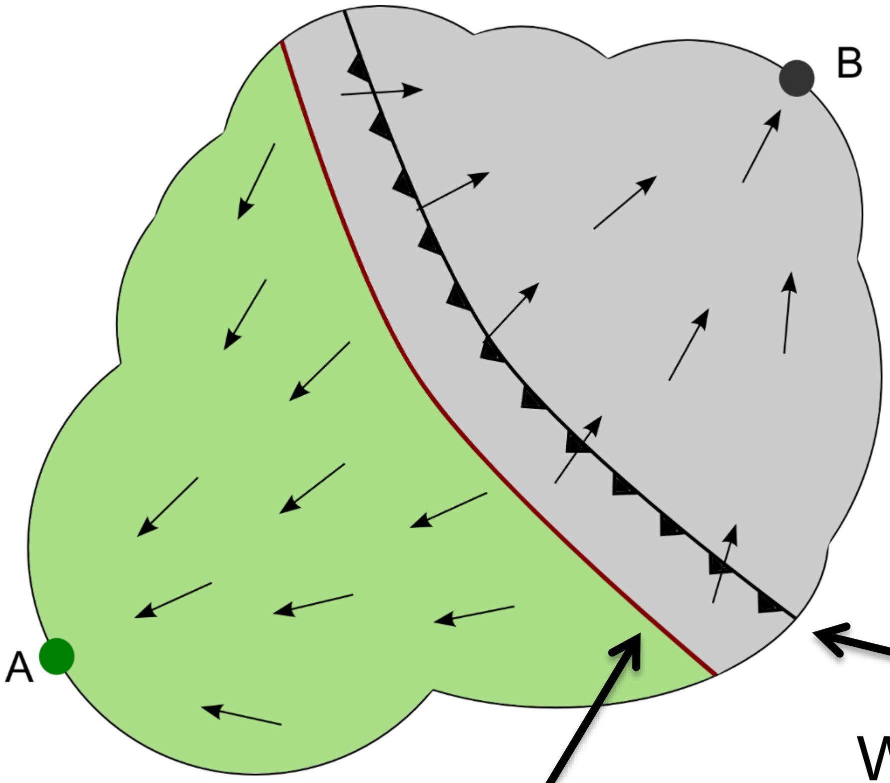
A fly in my SOUP



Withdrawal from B

Mid-line

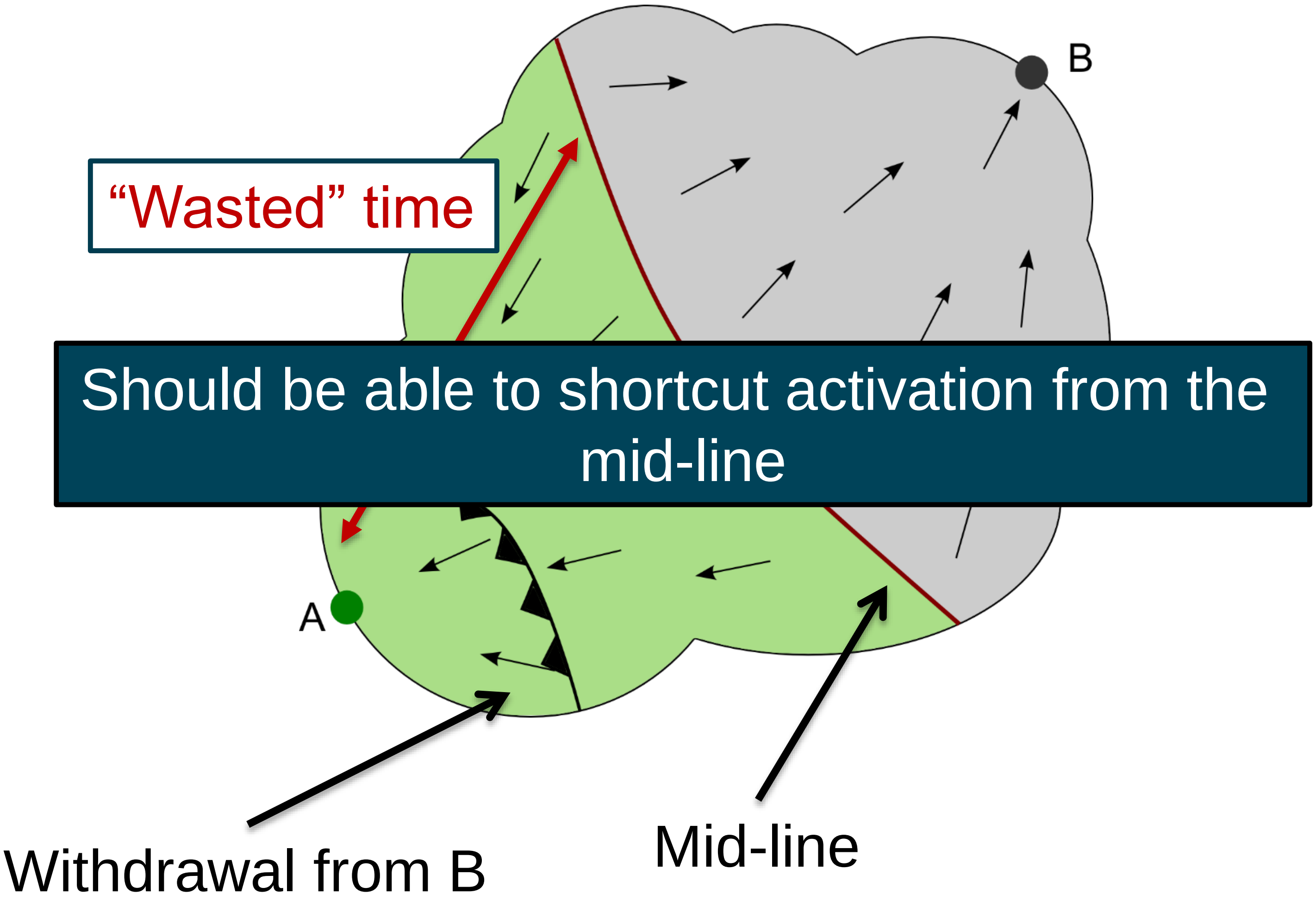
A fly in my SOUP



Withdrawal from B

Mid-line

A fly in my SOUP



Should be able to shortcut activation from the mid-line

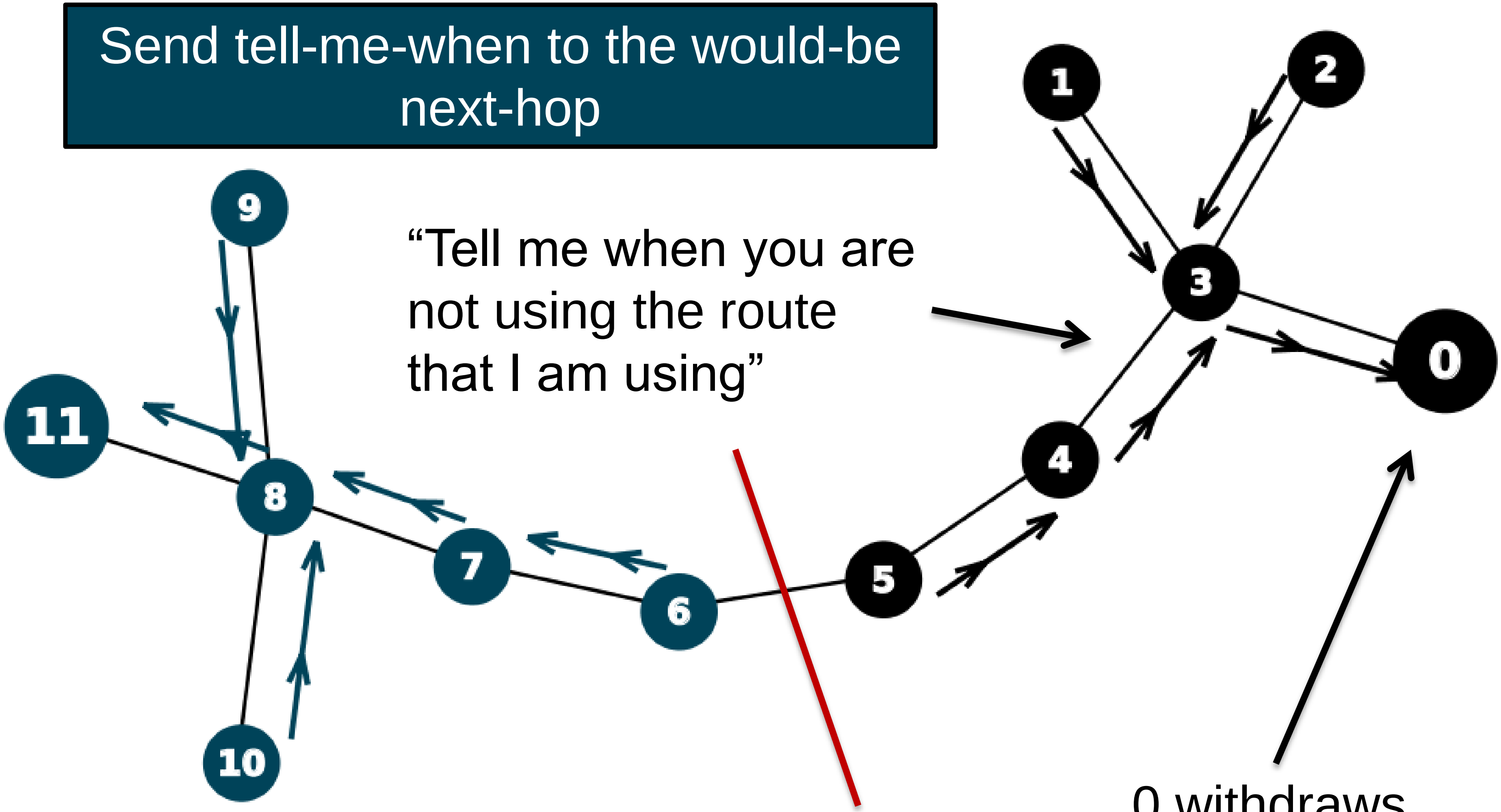
LOUP to the rescue

A superset of SOUP. Inherits all mechanisms previously discussed.

Adds tell-me-when messages to shortcut activation

LOUP to the rescue

Send tell-me-when to the would-be next-hop

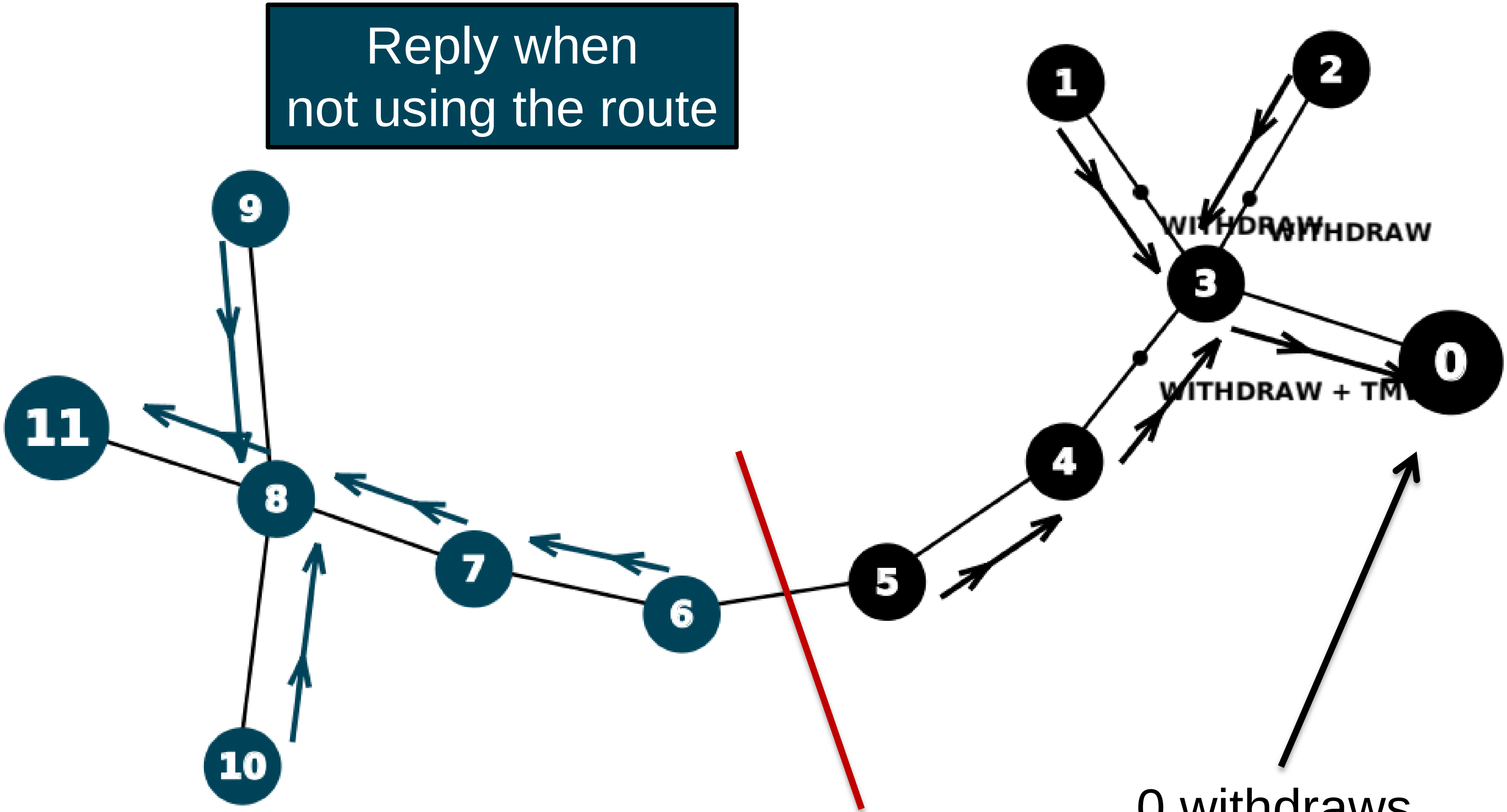


“Tell me when you are not using the route that I am using”

0 withdraws

LOUP to the rescue

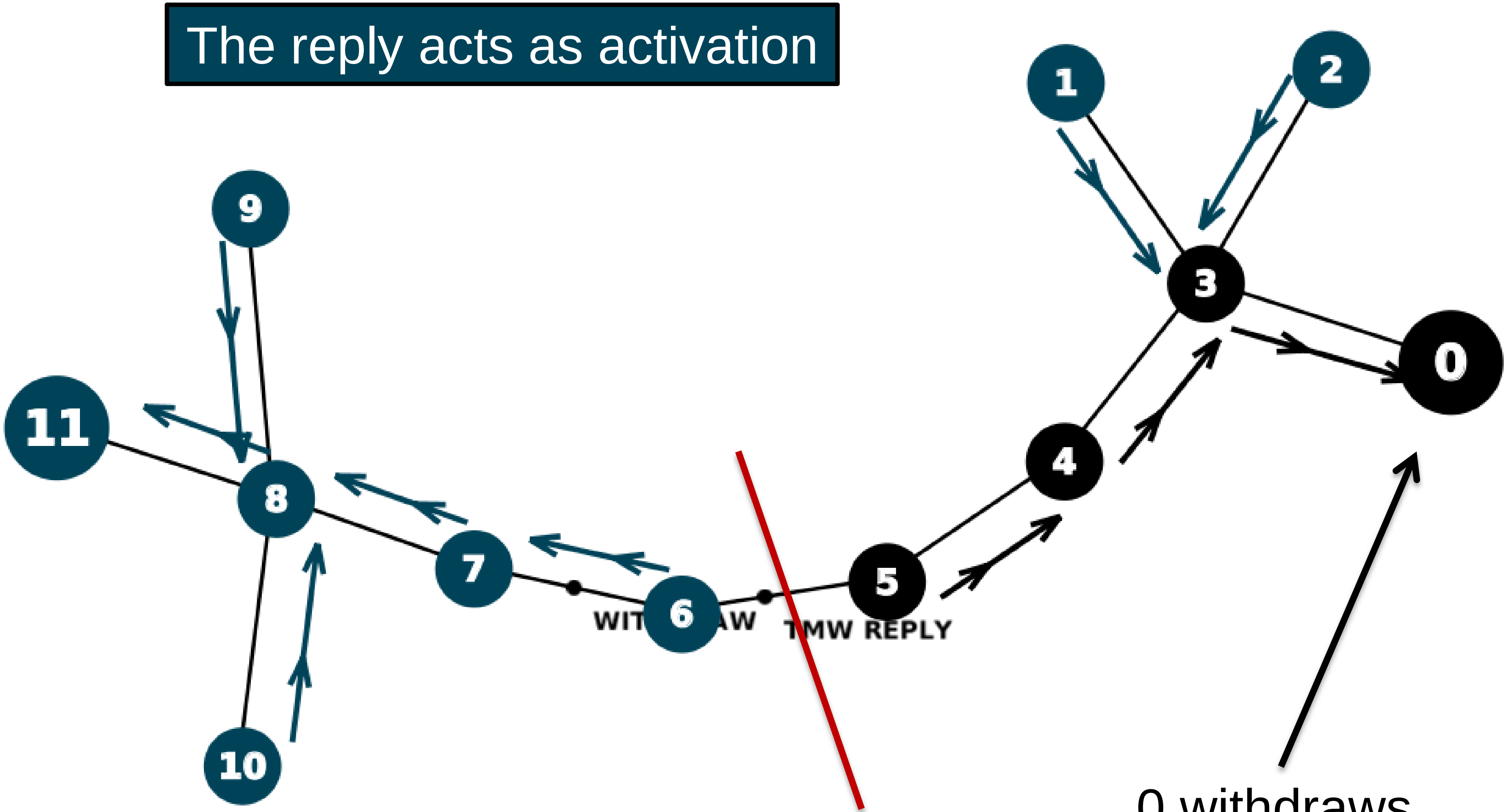
Reply when
not using the route



0 withdraws

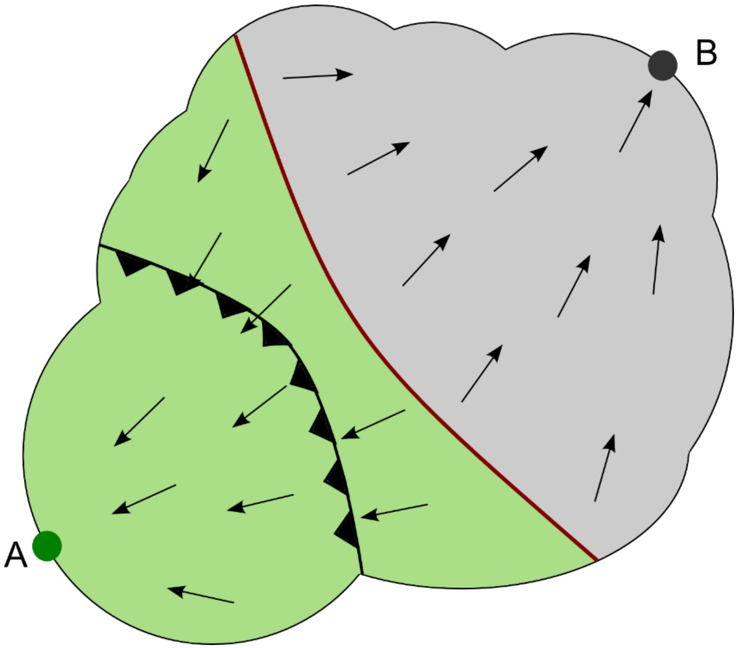
LOUP to the rescue

The reply acts as activation



0 withdraws

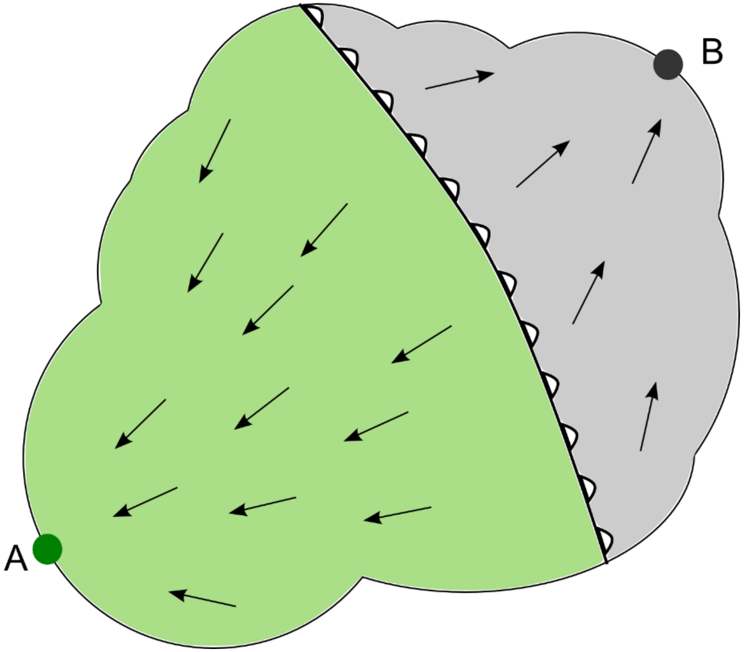
SOUP vs LOUP



SOUP

Need to propagate activation all the way to the other end of the network

Provably does not loop



LOUP

Can shortcut activation using explicit tell-me-when messages

Can loop in the presence of unusually high churn

Evaluation

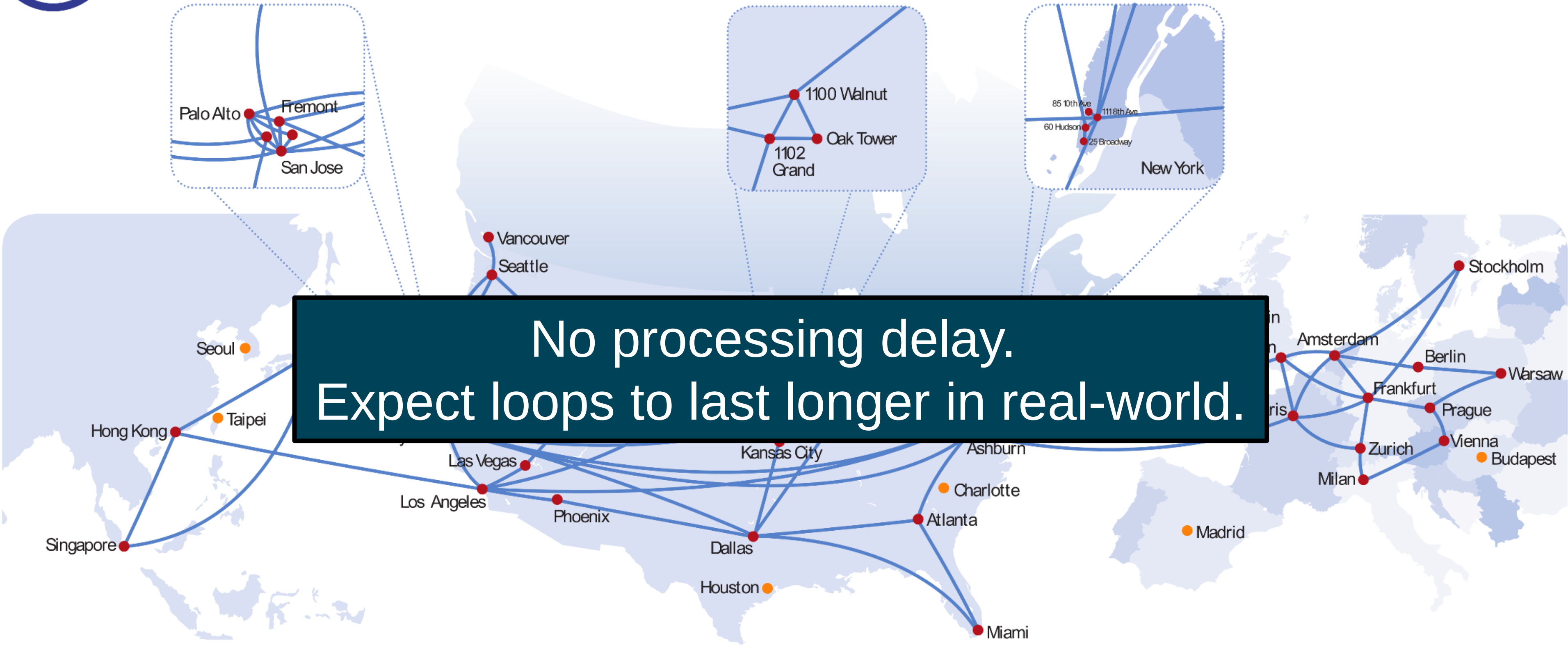
Evaluation

- Loop freedom on update
- Delay on withdrawal
- Why not replace iBGP with DUAL[Aceves 1993]?
- Loop freedom on withdrawal
- Delay on update
- Load on the network
- FIB churn introduced
- Stability in the presence of IGP events
- Evaluation of real-world prototype

More evaluation in paper

Evaluation setup

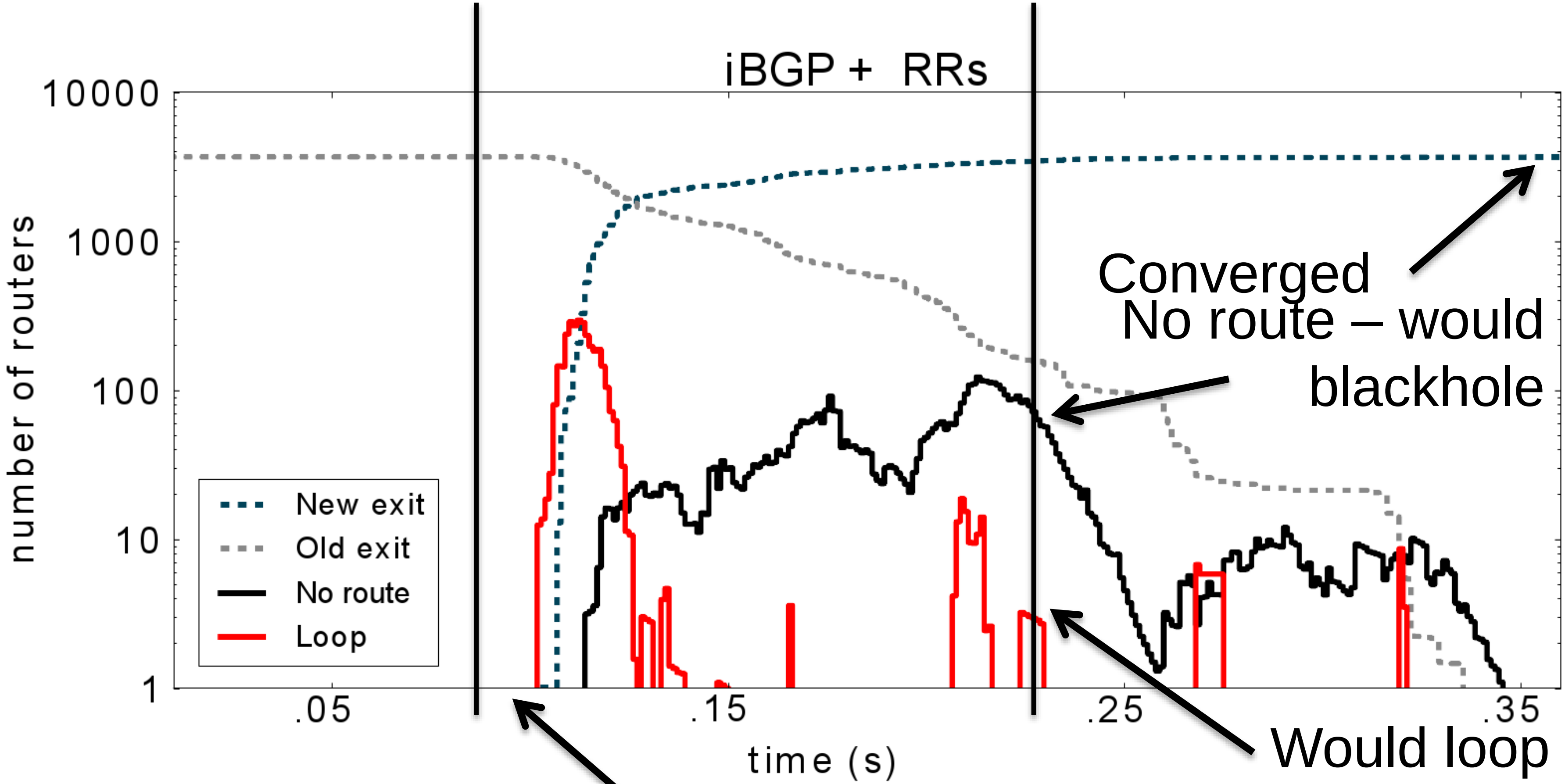
HURRICANE ELECTRIC
INTERNET SERVICES



No processing delay.
Expect loops to last longer in real-world.

- Simulation results based on publicly available HE topology
- Connectivity in POPs inferred from iBGP session data
- Model delay as speed-of-light + [0-10]ms

iBGP+RRs causes loops on update



Converged

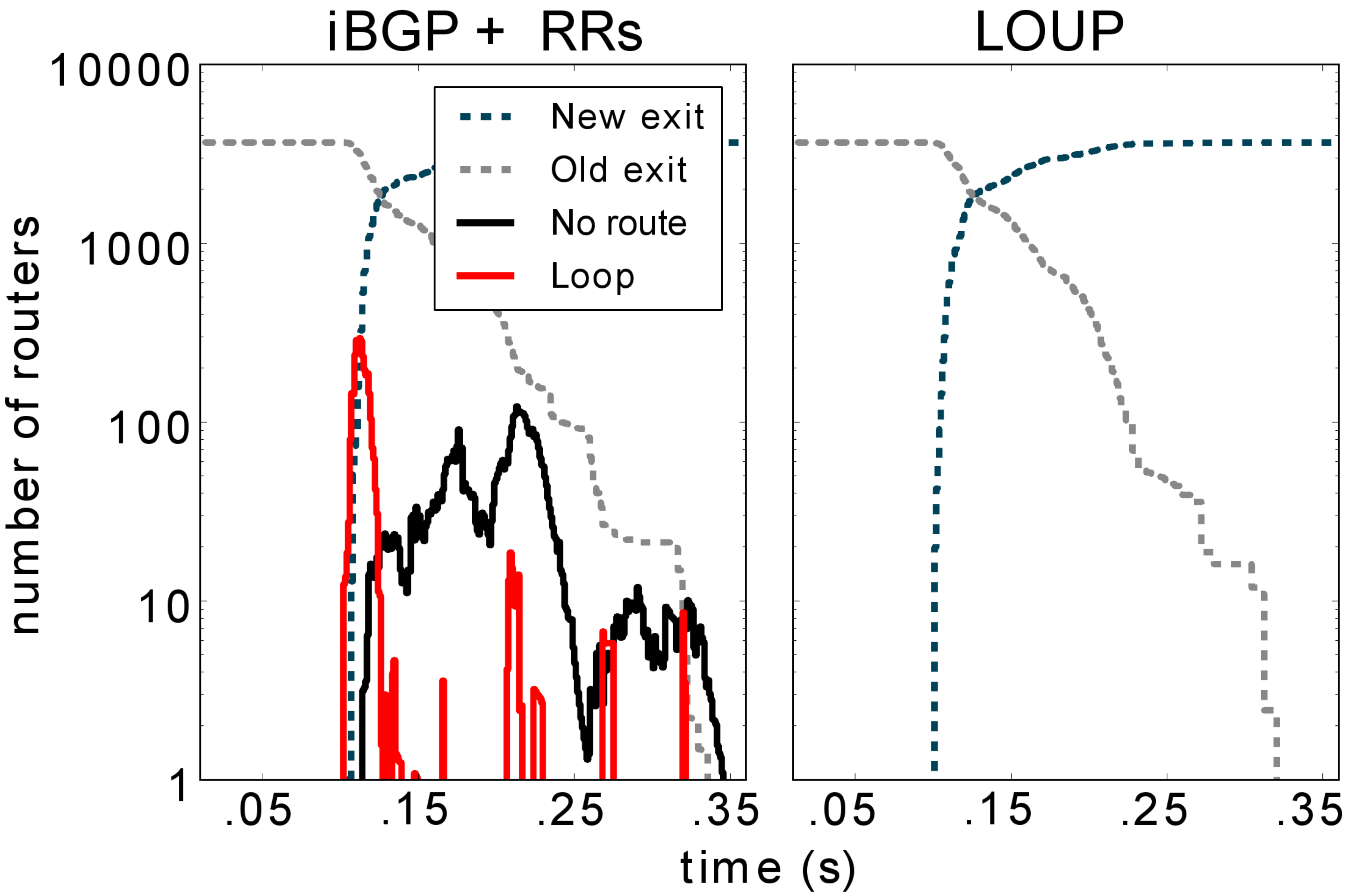
No route – would blackhole

Would loop

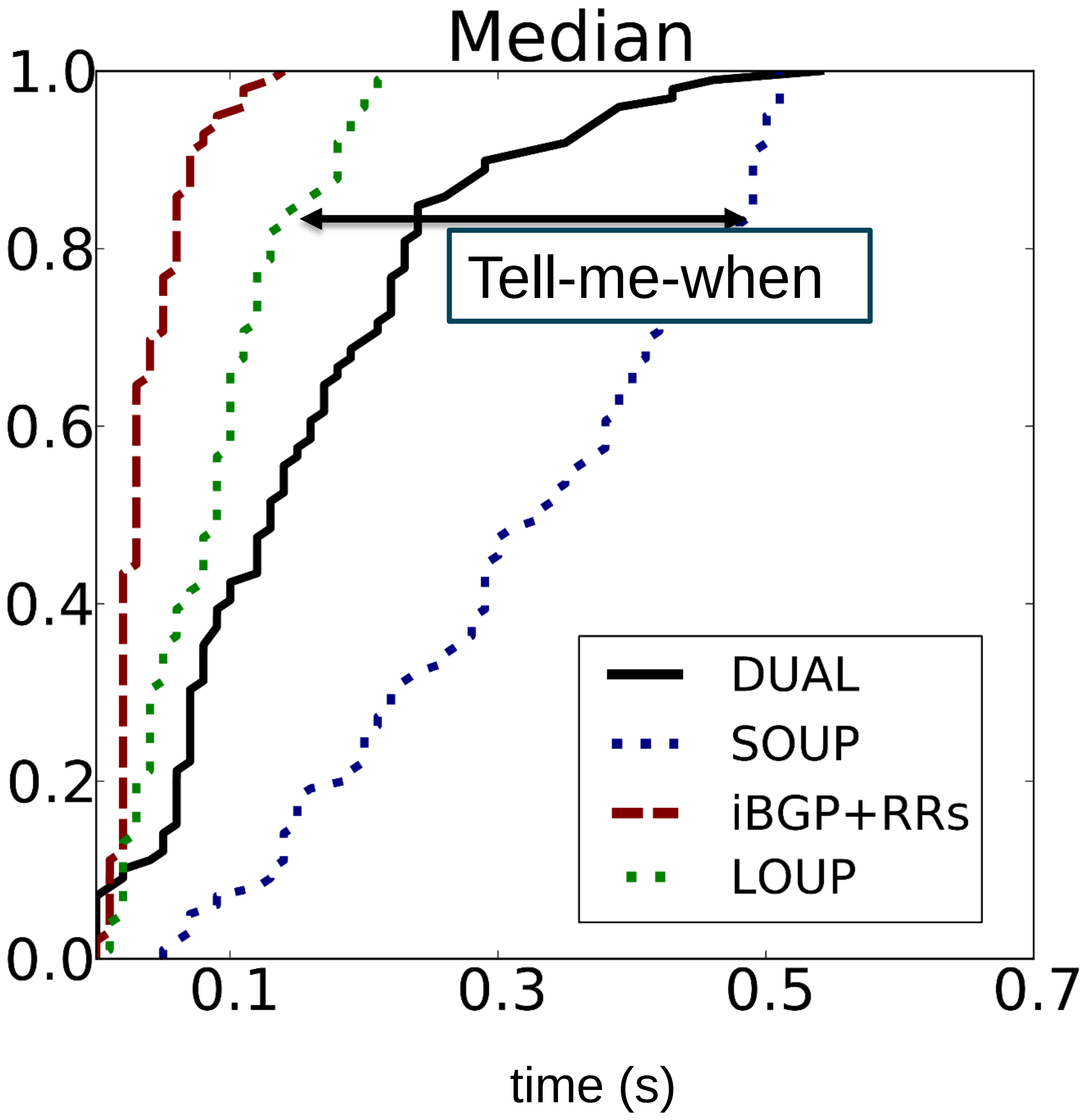
New better route arrives

- One route, a better one is received at 0.1s

LOUP causes no loops



Delay on withdrawal



Tell-me-when

Conclusion

- iBGP’s transient loops disrupt end-to-end-reachability
- Careful ordering and application of routing changes prevents loops
- Simple Ordered Update Protocol (SOUP):
  - Fully distributed
  - Provably prevents all transient loops when the underlying topology is stable
  - Lightweight (vs. Consensus Routing, DUAL)
  - Configuration free (vs. route reflectors)
- Fast convergence with Link-Ordered Update Protocol (LOUP)

MPLS does not get you off the hook

Because …

- Even a BGP-free core still uses BGP to distribute routes
- Route reflectors are still present if a lot of customer routes
- Some of the ordering problems shown still exist
- LOUP can also do VPNs

What about DUAL?

SOUP is different because

- It does not flood and does not require activations from all neighbors
- It does not need a complicated state machine to handle multiple simultaneous route events
- It is not maintaining the IGP – it runs on top of it and when an IGP event occurs it does not need to activate external prefixes

RFT maintenance

If the underlying topology changes the RFT must follow

- LOUP actively maintains the RFT using periodic messages
- All messages stored in log-like data structures
- If the IGP is stable (99+% of the time), LOUP enough
    - For complete protection during IGP changes use EIGRP