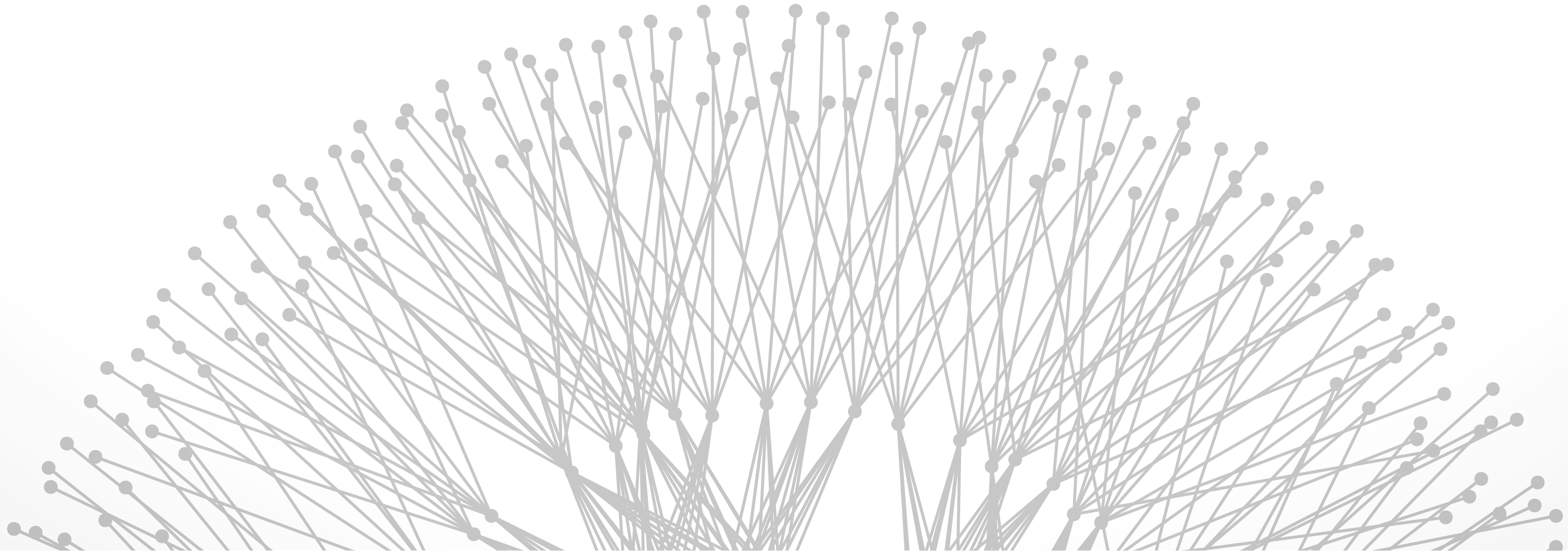
Network Reliability

Brighten Godfrey
CS 538 March 7, 2018



Network reliability in context

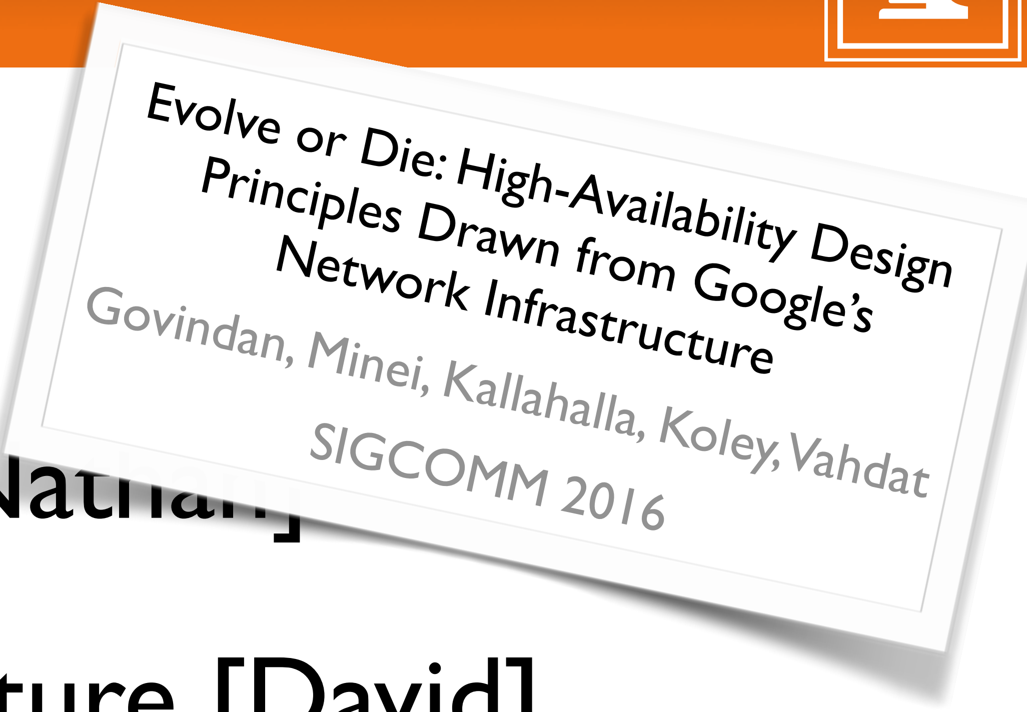
Every component can fail

- Data plane (e.g. physical equipment failures)
- Control plane (e.g. software bug)
- Management plane (e.g. human error updating configs)
  - *"The presence of persistent loops of durations on the order of hours is quite surprising, and suggests a lack of good tools for diagnosing network problems."* – Paxson

Diverse ways problems can manifest

- downtime
- congestion, quality of service degradation
- "grey" failures
- "This app is slow; what's wrong?"

"Evolve or Die" discussion


Evolve or Die: High-Availability Design Principles Drawn from Google's Network Infrastructure

Govindan, Minei, Kallahalla, Koley, Vahdat

SIGCOMM 2016

What is the management plane? [Nathan]

Unified system vs. custom architecture [David]

Software bugs will increase. Simulate? [Ashwini]

- See: CrystalNet (SOSP'17)

What's the frequency of failures? [Liia]

Surprising that failures evenly distributed [Shivam]

"Evolve or Die": My thoughts

103 types of failures: Hard to focus!

- Though, knowing the frequency might help…

High rate of change (58 MOps/wk was typical)

Sequence of multiple software bugs often the culprit

- e.g. failover timer too slow => split brain in OFCs => OFC software bug causing inconsistent state
- Not as unlikely as you might think!

Need for more than monitoring

- root cause analysis
- automated response

Highly Available Routing

Control & data speeds don't match

Reliability problems in Internet routing

- Basic issue: controlling a distributed system => inconsistent state across routers => loops, black holes
- Also in link state, distance vector

Problem: control plane is slow...

- Control plane routing does eventually converge!
- But may take 100s of milliseconds (milliseconds possible after careful tuning of protocol timers & algorithms*)

...and data plane is fast

- Sending 50 byte packet at 40 Gbps = 10 nanoseconds

* "Toward Millisecond IGP Convergence", Cengiz Alaettinoglu, Van Jacobson, Haobo Yu, in NANOG 20, October 2000

Reliability in the data plane

Fast path (data plane) needs failure reaction!

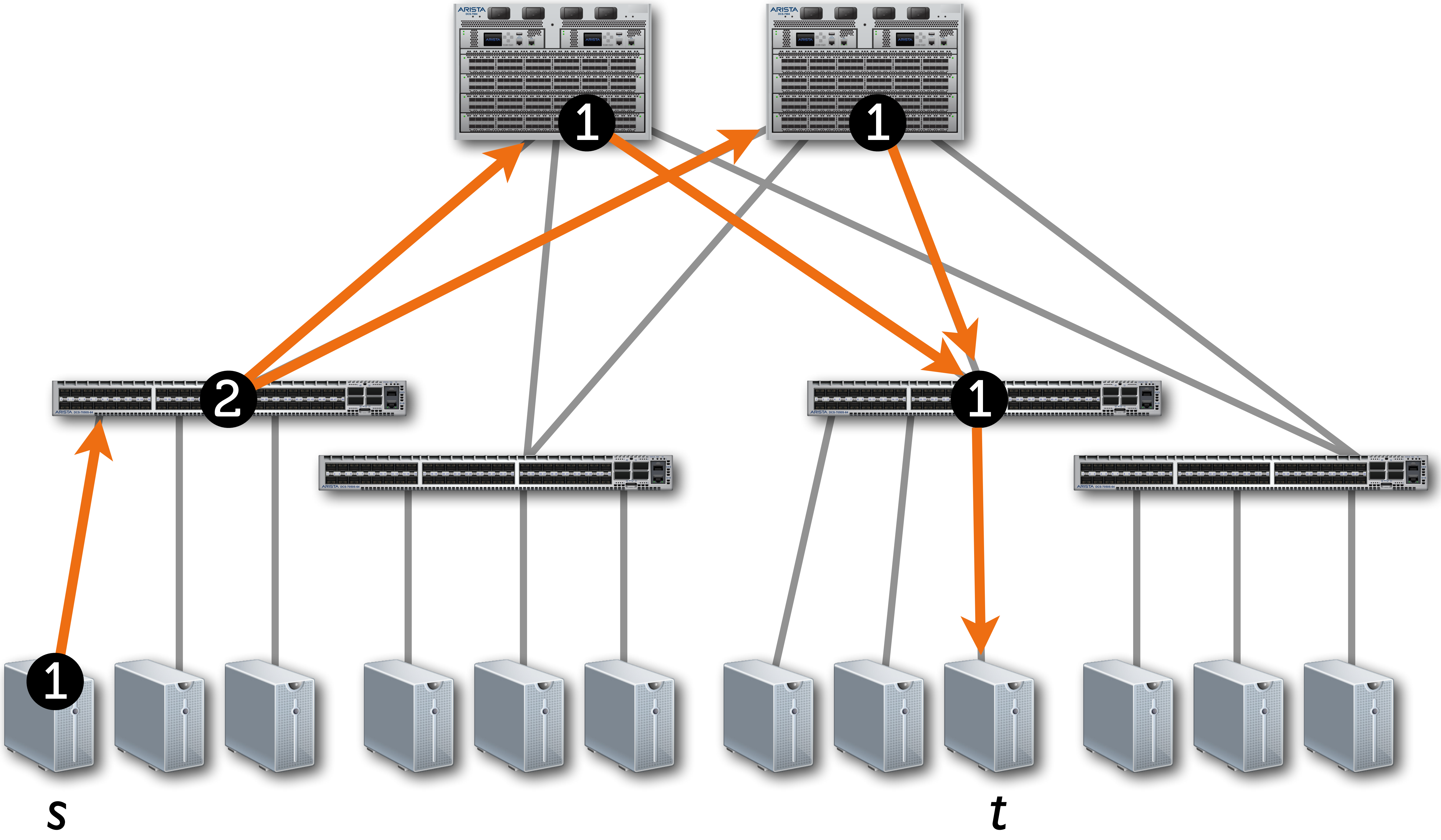
Rest of this lecture: building a solution

Techniques in practice

Equal Cost Multipath (ECMP)

- Control plane produces not one next-hop, but many
- Next hops must be closer to destination (so no loops)
- Data plane sends packet to any next-hop that's working

How many next-hops in ECMP?



Techniques in practice

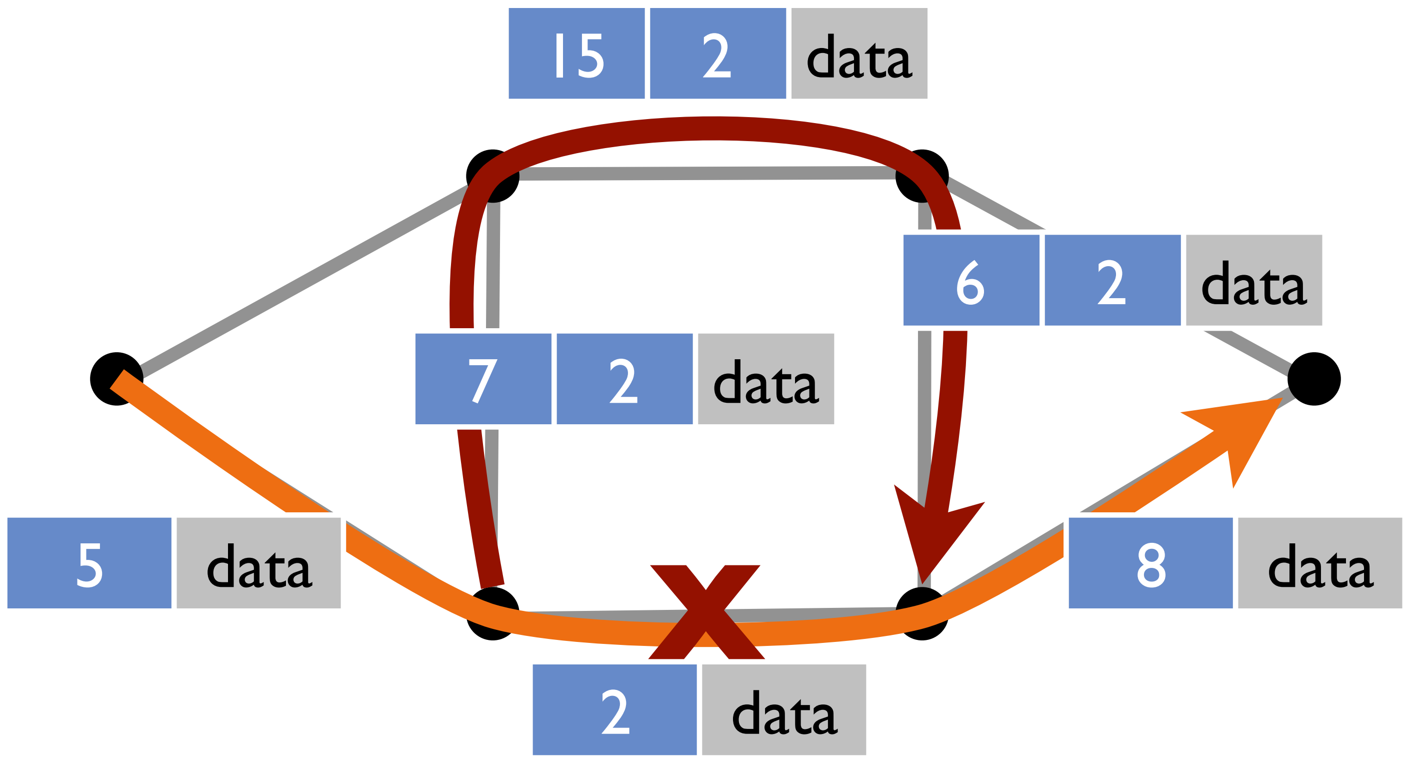
Equal Cost Multipath (ECMP)

- Control plane produces not one next-hop, but many
- Next hops must be closer to destination (so no loops)
- Data plane sends packet to any next-hop that's working
- Defeated by even a single link failure in some cases

MPLS Fast Re-Route link protection

- Explicit backup path for each failure case (link or node failure)

MPLS FRR Link Protection



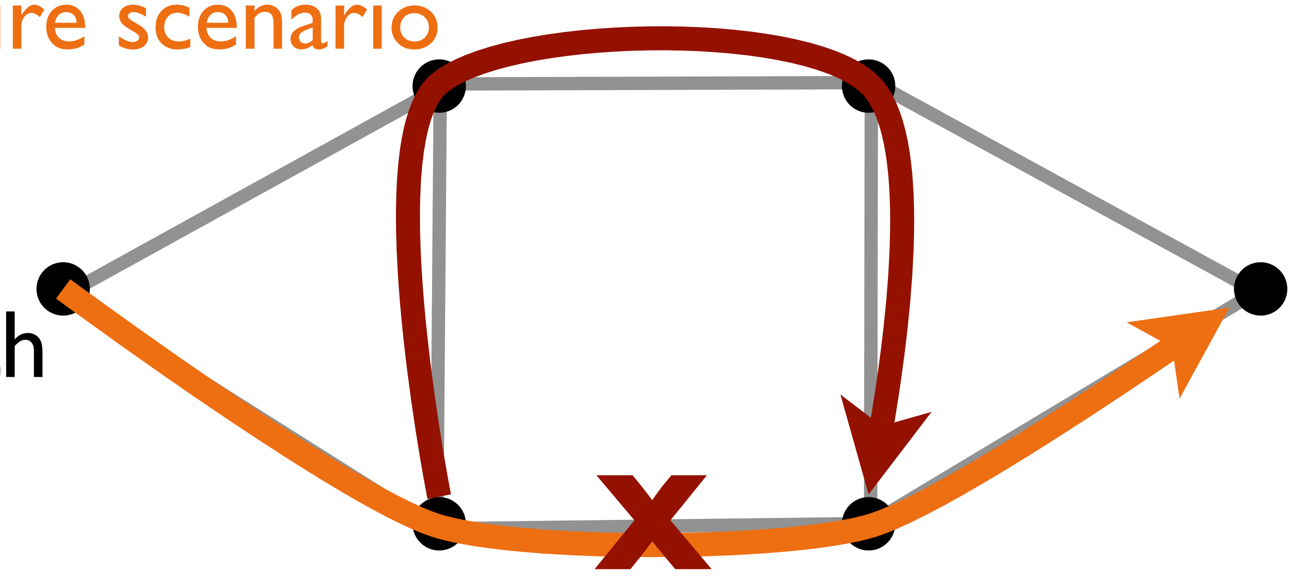
Techniques in practice

Equal Cost Multipath (ECMP)

- Control plane produces not one next-hop, but many
- Next hops must be closer to destination (so no loops)
- Data plane sends packet to any next-hop that's working
- Defeated by even a single link failure

MPLS Fast Re-Route link protection

- Explicit backup path for each link
- Protects against single failure scenario (shared risk link group)
- Uses more FIB entries
- Not shortest alternate path



DOOMED!! .... ?

Holy Grail: "Ideal connectivity"

- Data plane always correctly forwards packets towards destination, even with arbitrary link failures

Is it possible?

- Yes!
- BGP, OSPF, RIP, ISIS, ..., all have loops & black holes during convergence, ultimately causing packet loss
- But that is not fundamentally necessary!

5 minutes in small group:
Devise a correct solution

Ideal connectivity

5 minutes in small group:
Devise a correct solution

1. Every packet is eventually forwarded to destination correctly
    - Assume: arbitrary failures, but a path exists
    - Assume: no congestion or physical layer problems
2. Simple technique implementable in data plane
    - Feel free to play with packet header formats, protocols, etc.

Achieving ideal connectivity

Ideal connectivity: correct, but...

The random walk

- If failure encountered, set a "random walk" bit in packet
- Whenever packet has random walk bit, send to random neighbor
- Slightly silly solution

Failure-carrying packets (FCP)

Achieving convergence-free routing using failure-carrying packets
Lakshminarayanan, Caesar, Rangan, Anderson, Shenker, Stoica
SIGCOMM 2007

Approach

- Link state routing + link failure info carried inside packet
- Router recomputes shortest paths on the fly given new information inside packet

Key points

- Separate two functions: long-term topology distribution, handling transient changes
- Trick: carry topology updates in packet
- Demonstrates feasibility of ideal connectivity

Failure-carrying packets (FCP)

Achieving convergence-free routing using failure-carrying packets
Lakshminarayanan, Caesar, Rangan, Anderson, Shenker, Stoica
SIGCOMM 2007

Approach

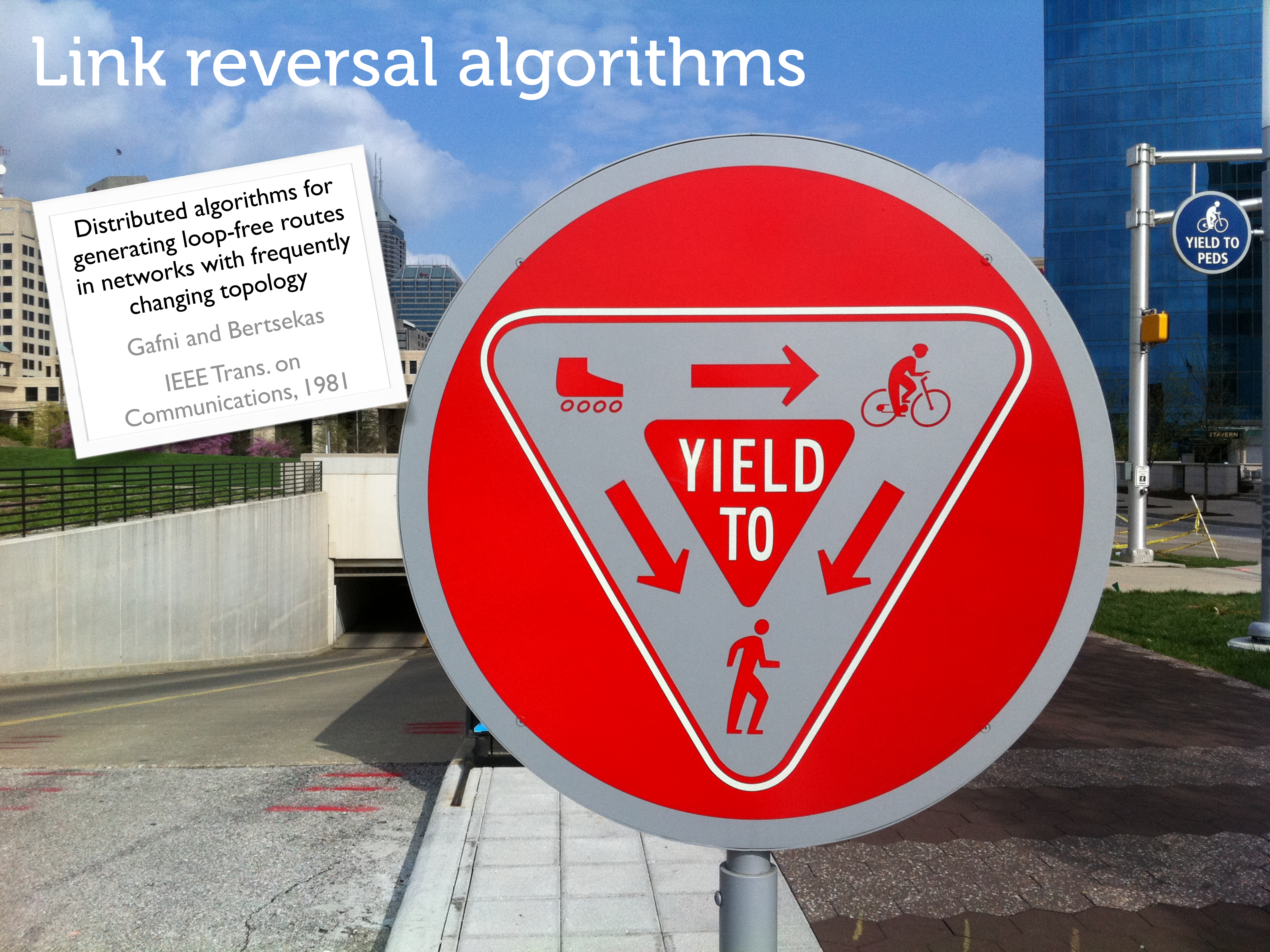
- Link state routing + link failure info carried inside packet
- Router recomputes shortest paths on the fly given new information inside packet

Difficult for data plane. Can we do better?

Key points

- Separate two functions: long-term topology distribution, handling transient changes
- Trick: carry topology updates in packet
- Demonstrates feasibility of ideal connectivity

Link reversal algorithms

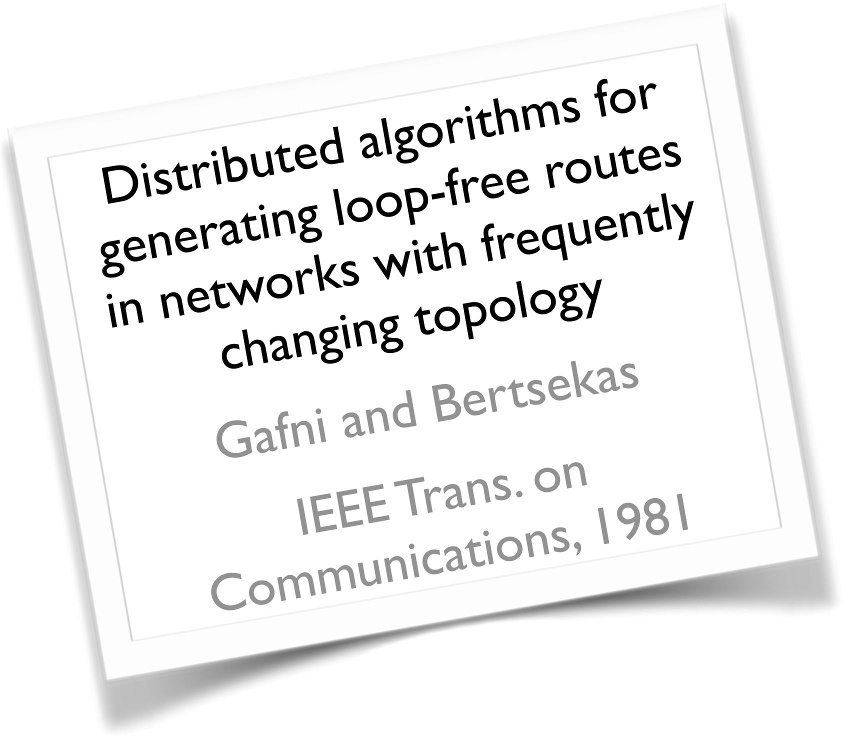


Distributed algorithms for generating loop-free routes in networks with frequently changing topology

Gafni and Bertsekas

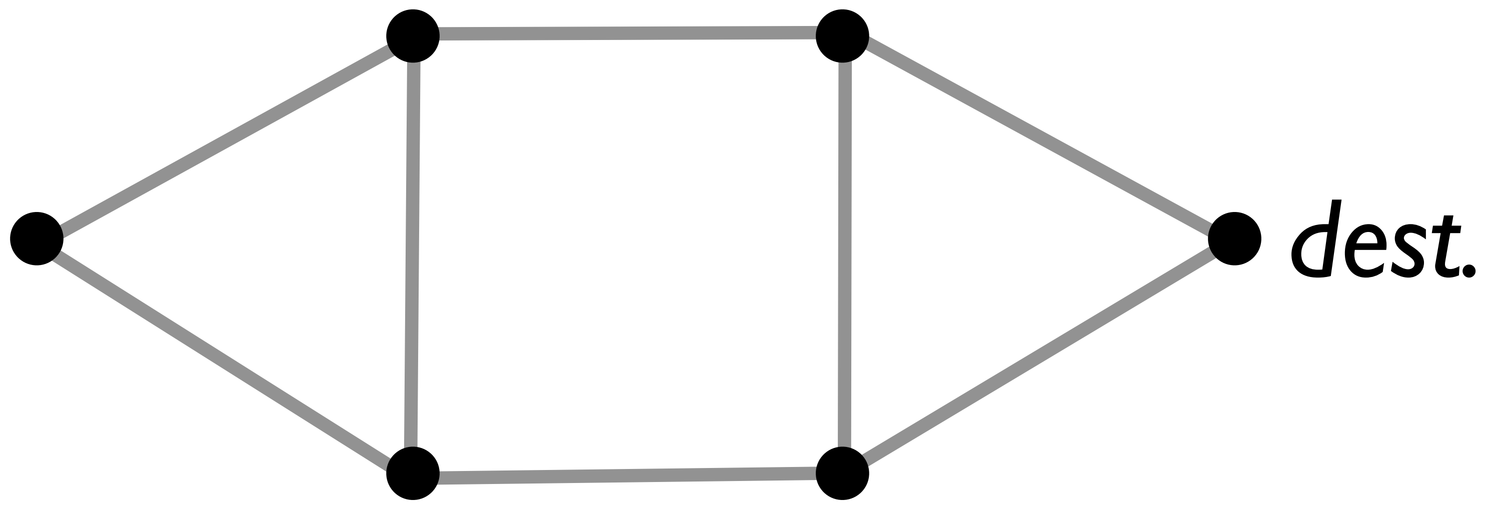
IEEE Trans. on Communications, 1981

Link reversal algorithms


Distributed algorithms for generating loop-free routes in networks with frequently changing topology

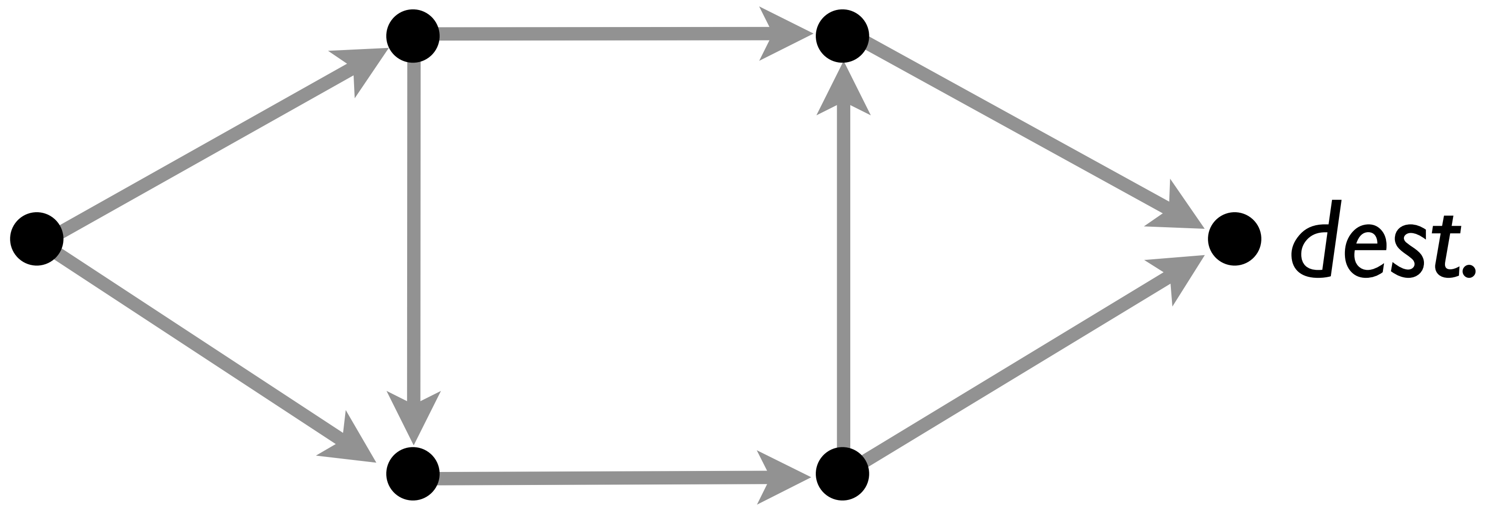
Gafni and Bertsekas

IEEE Trans. on Communications, 1981



Link reversal algorithms

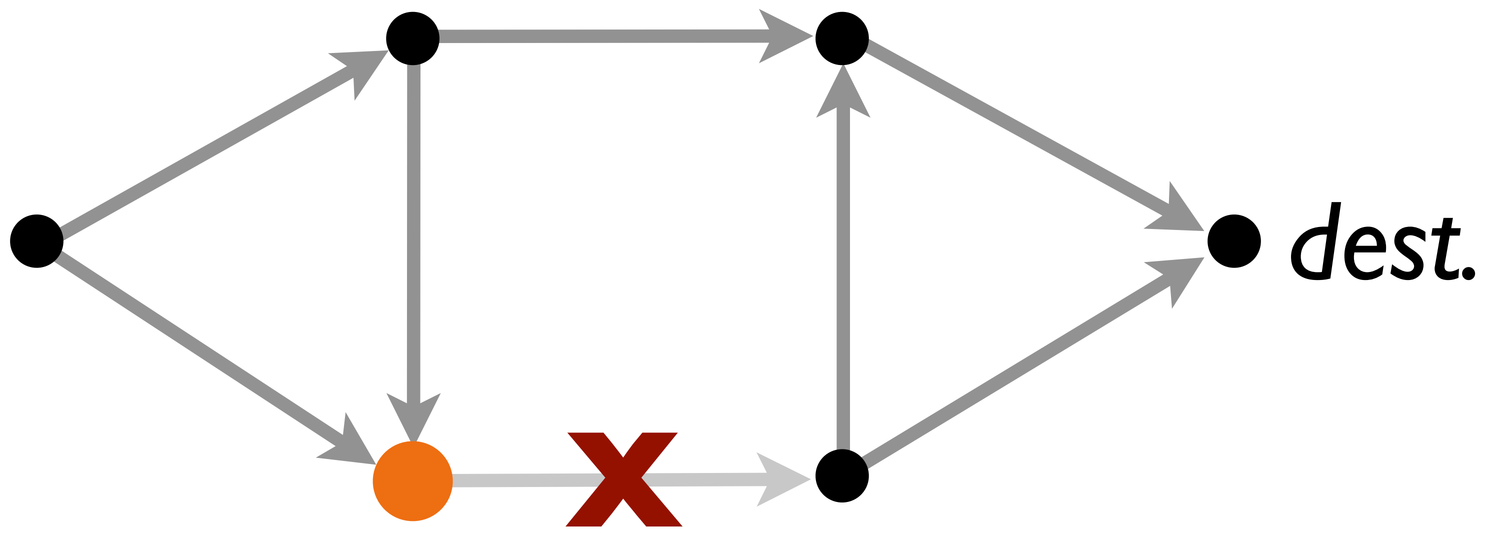
For each destination: begin with a directed acyclic graph (DAG) where destination is the sole sink



Link reversal algorithms

At each node:

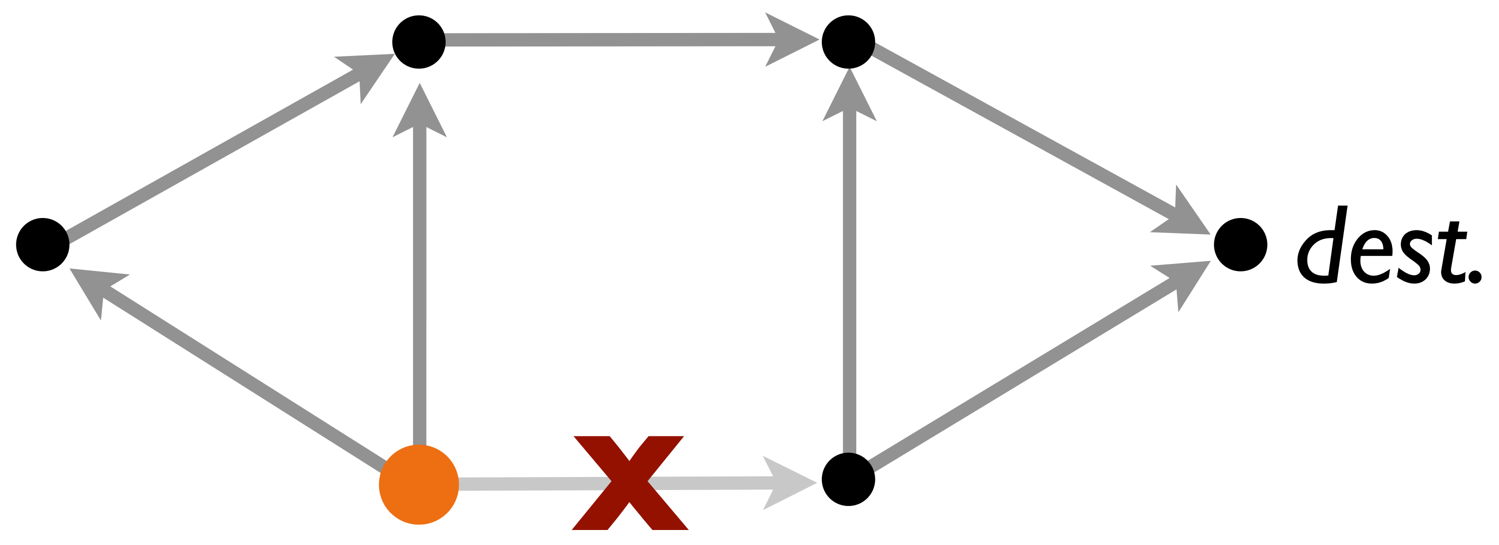
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

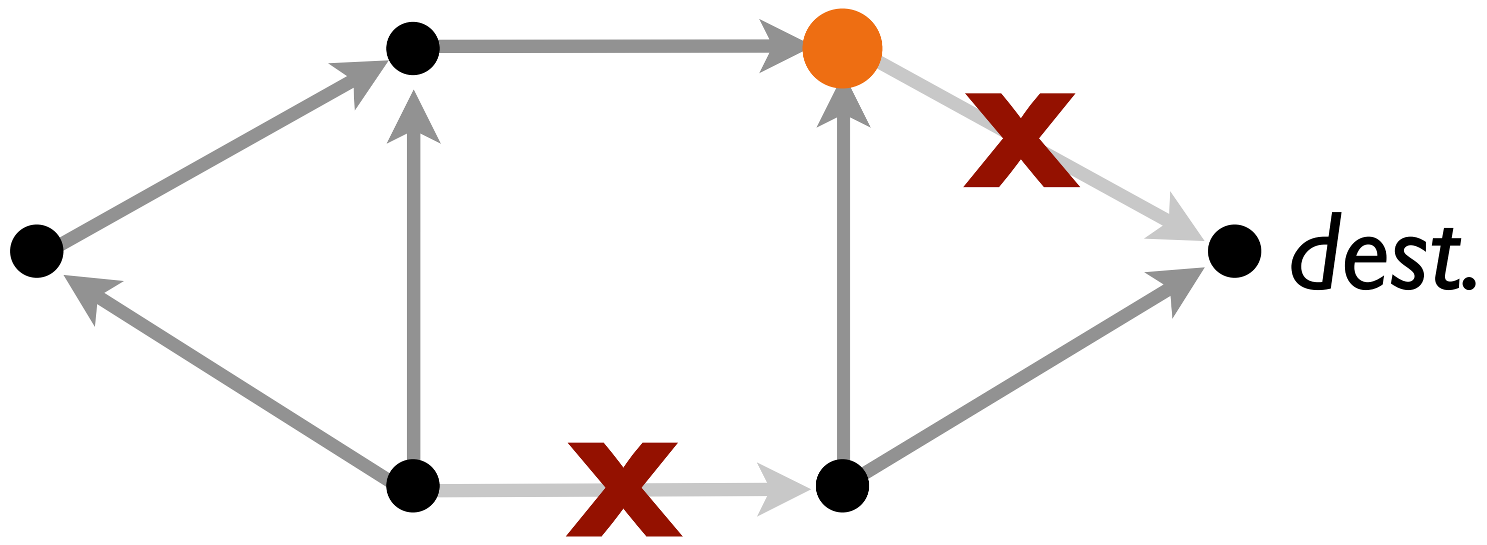
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

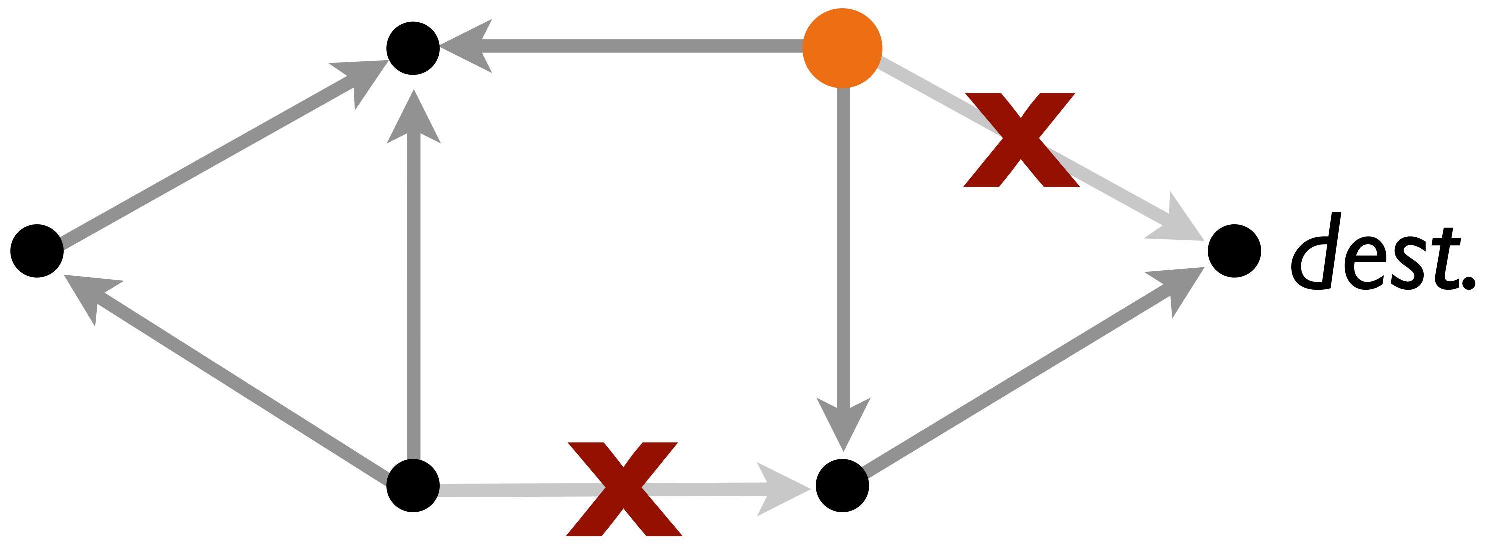
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

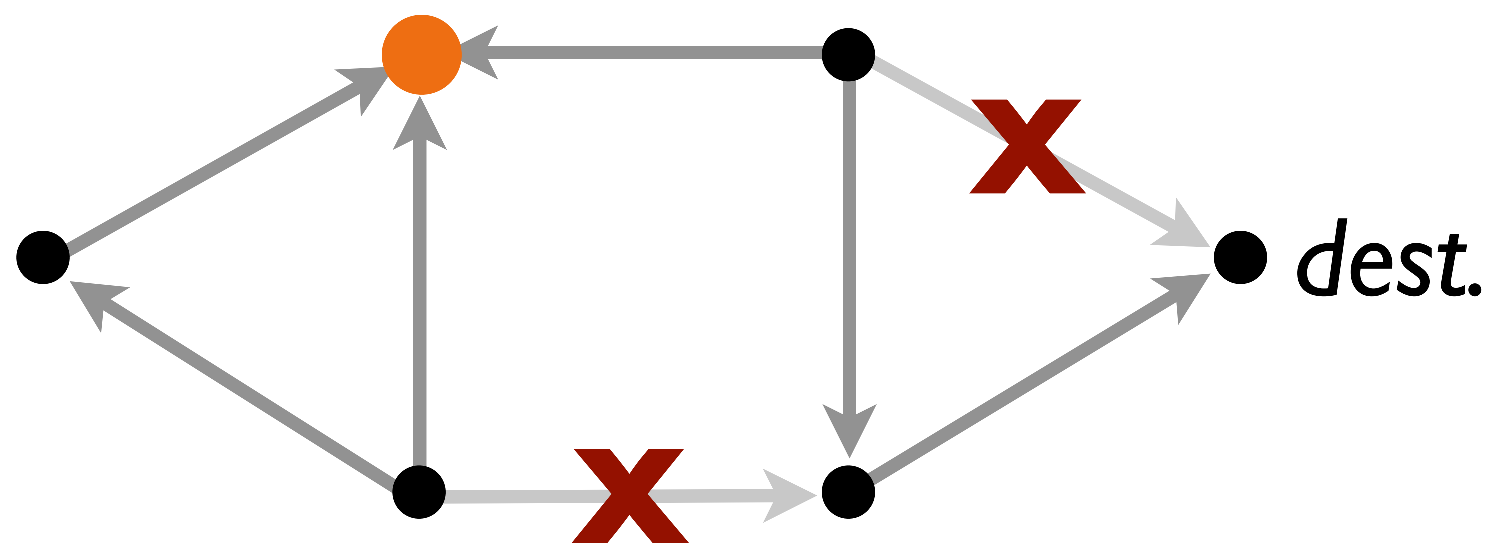
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

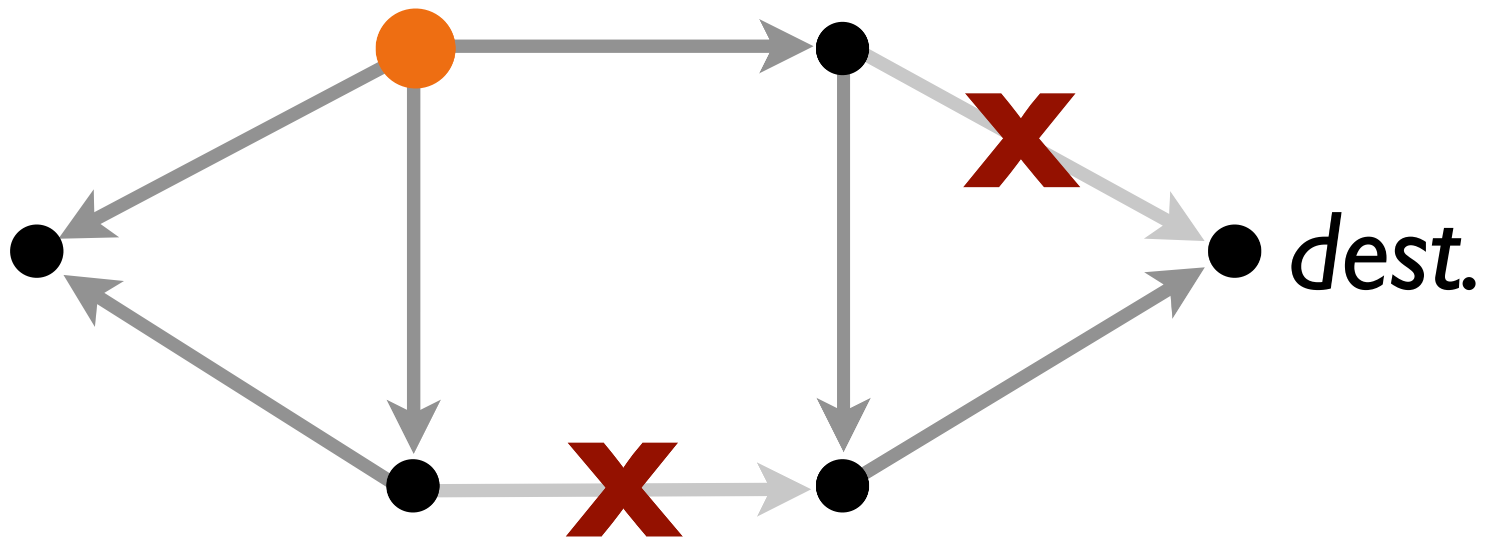
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

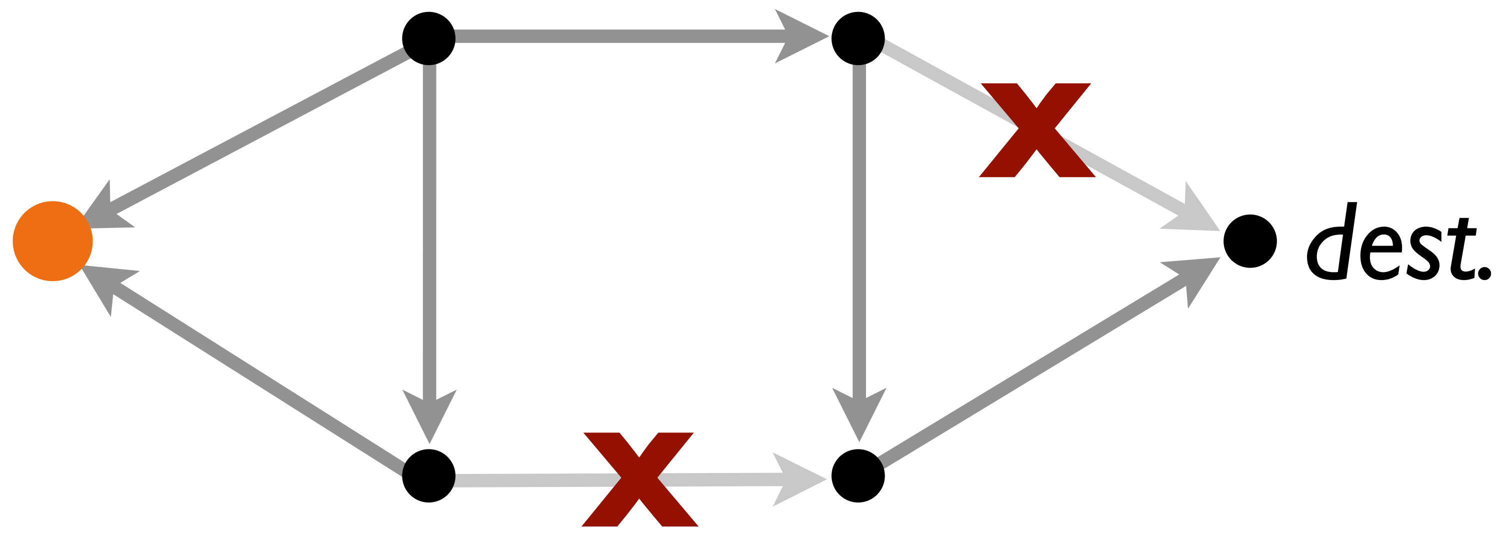
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

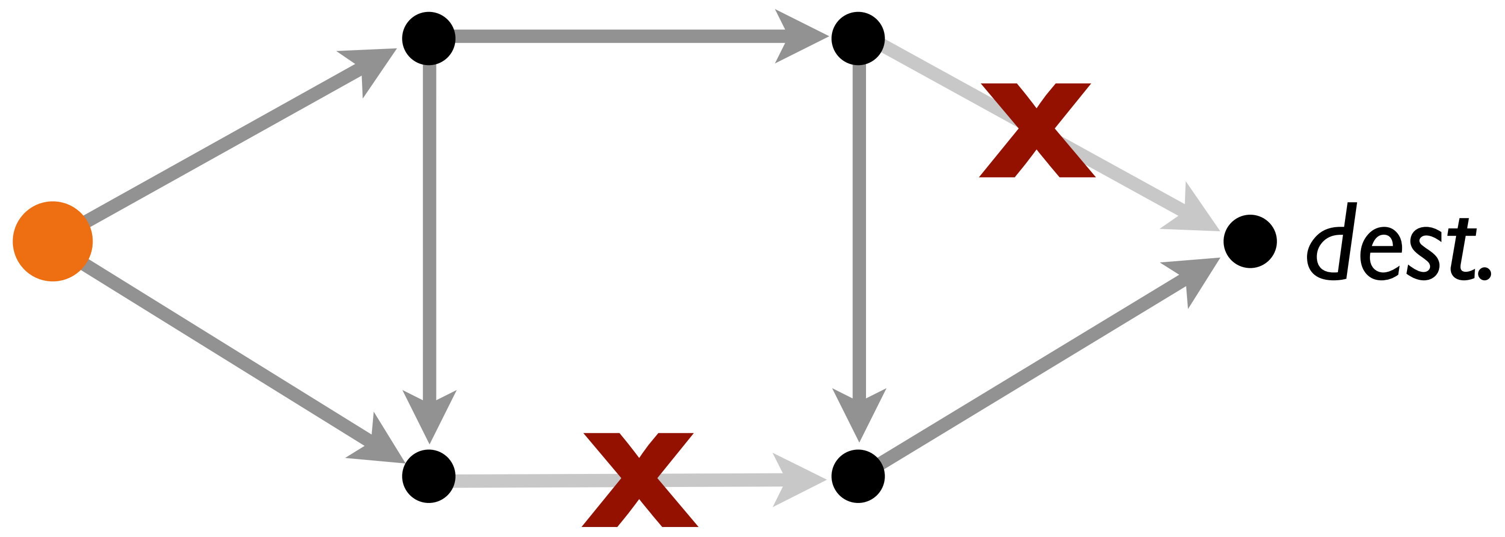
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

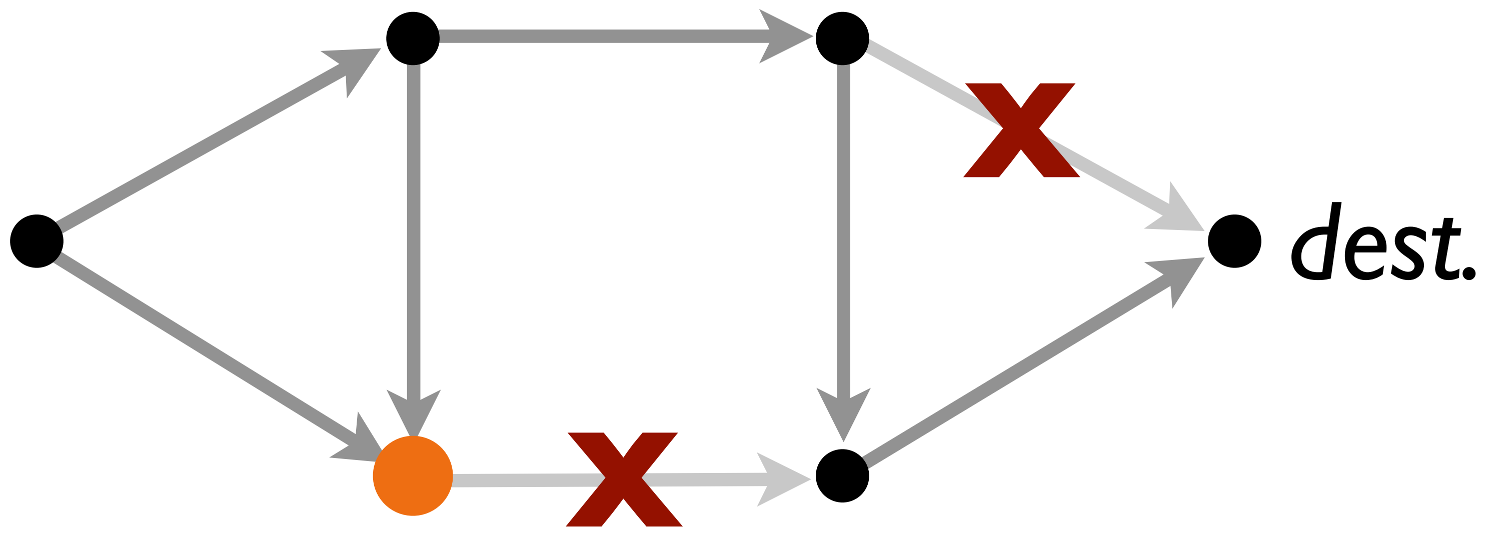
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

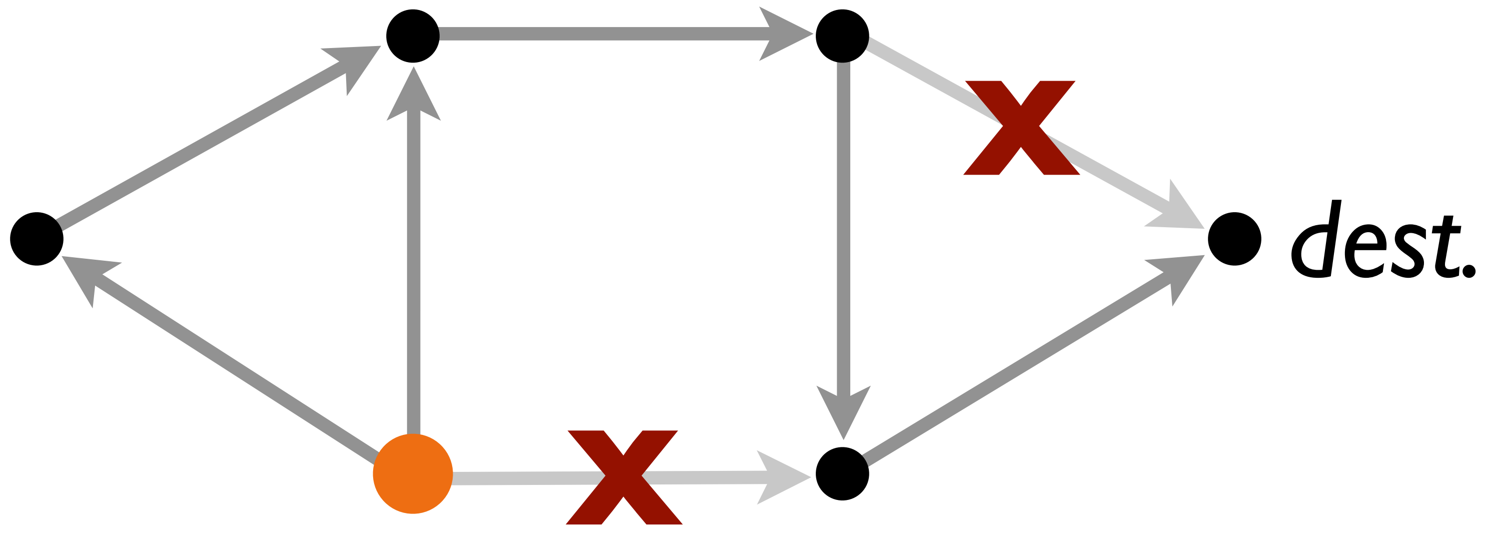
- If ever all links point inward,
  - Reverse all links



Link reversal algorithms

At each node:

- If ever all links point inward,
  - Reverse all links



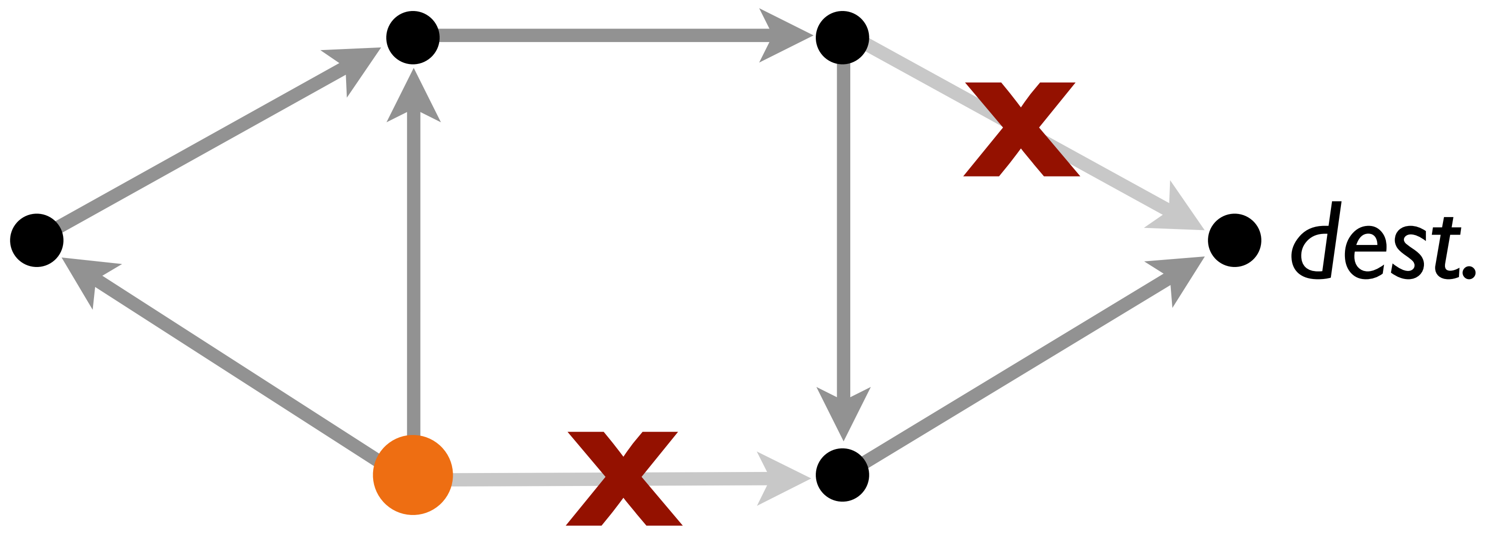
Whew! Done!

In the end, only one link flipped!

Guaranteed to converge

Proof

- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all

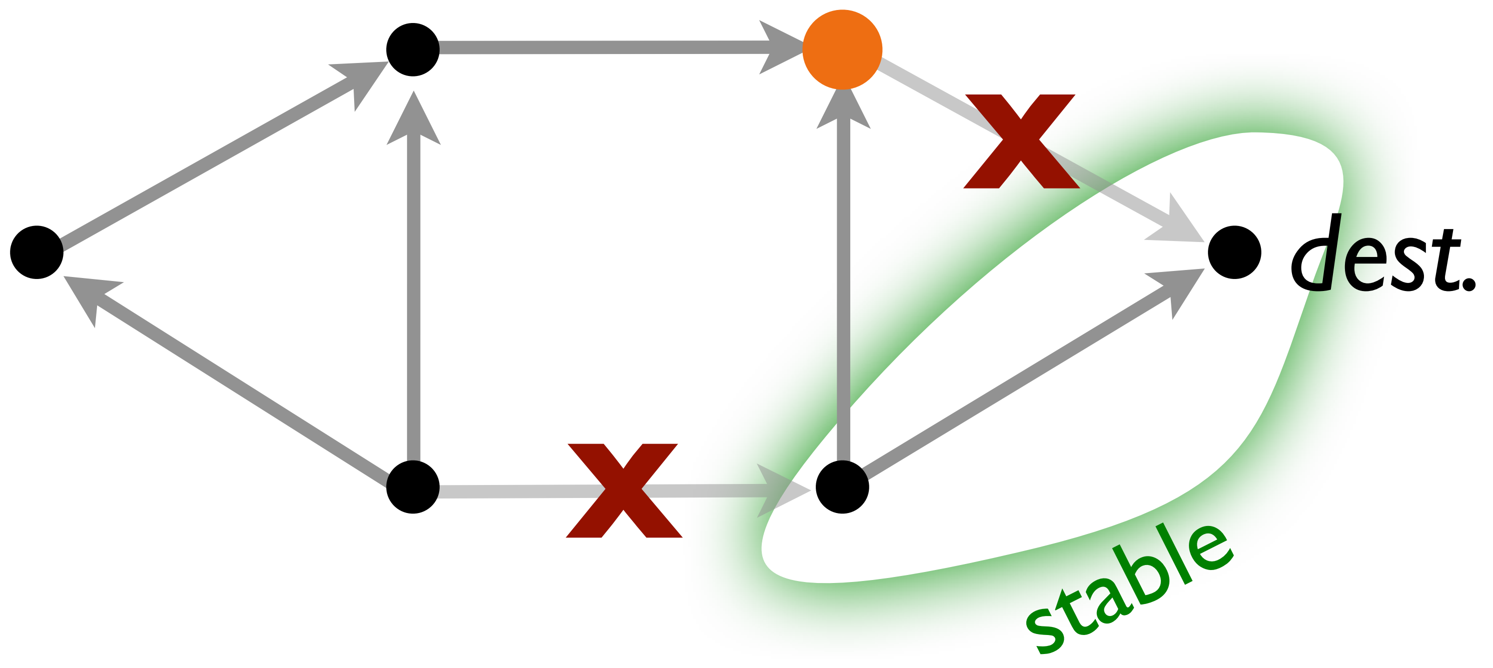


Let’s return to the beginning before convergence...

Guaranteed to converge

Proof

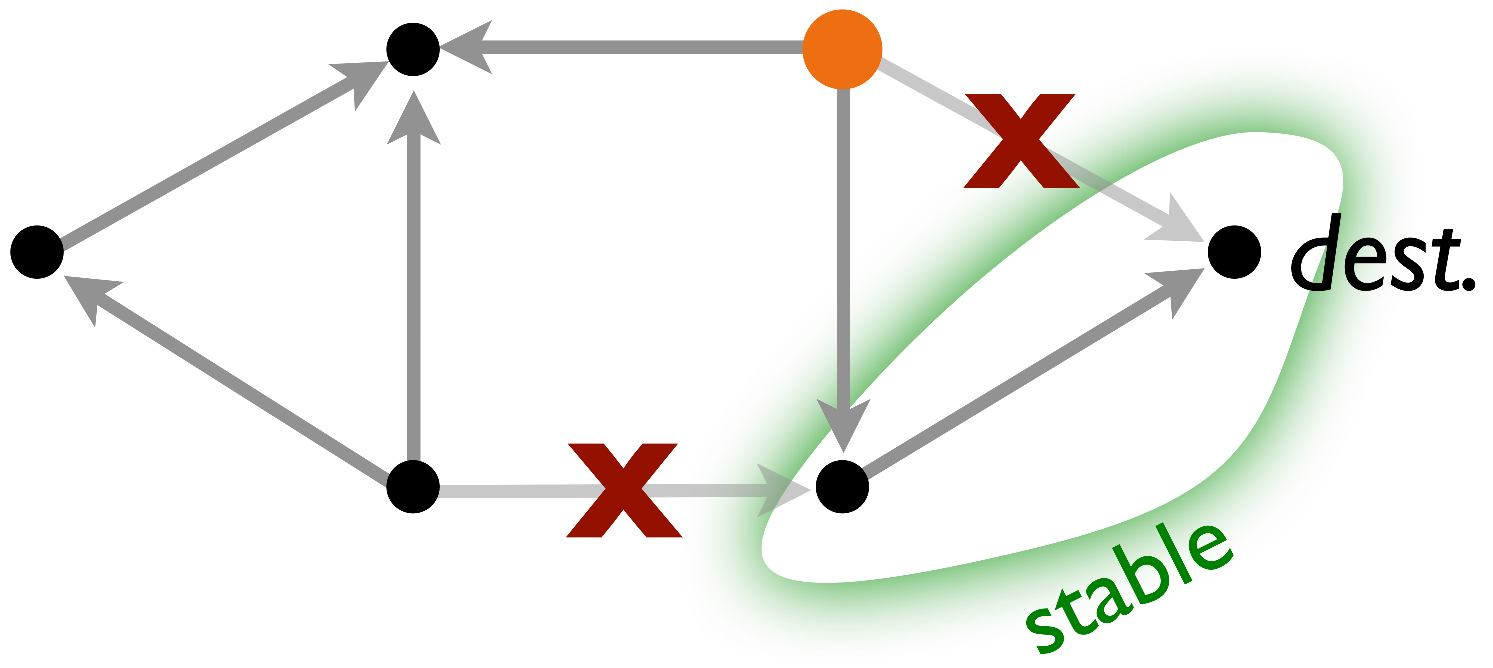
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

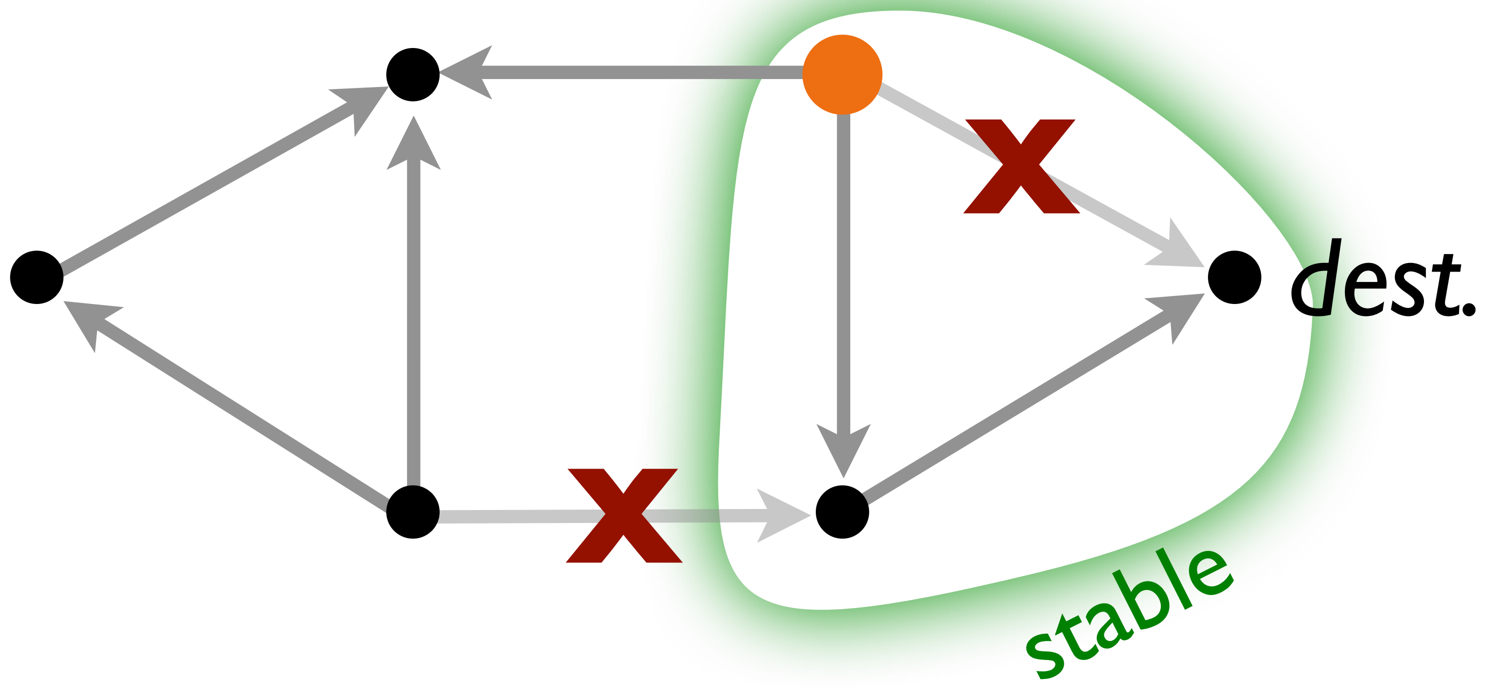
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

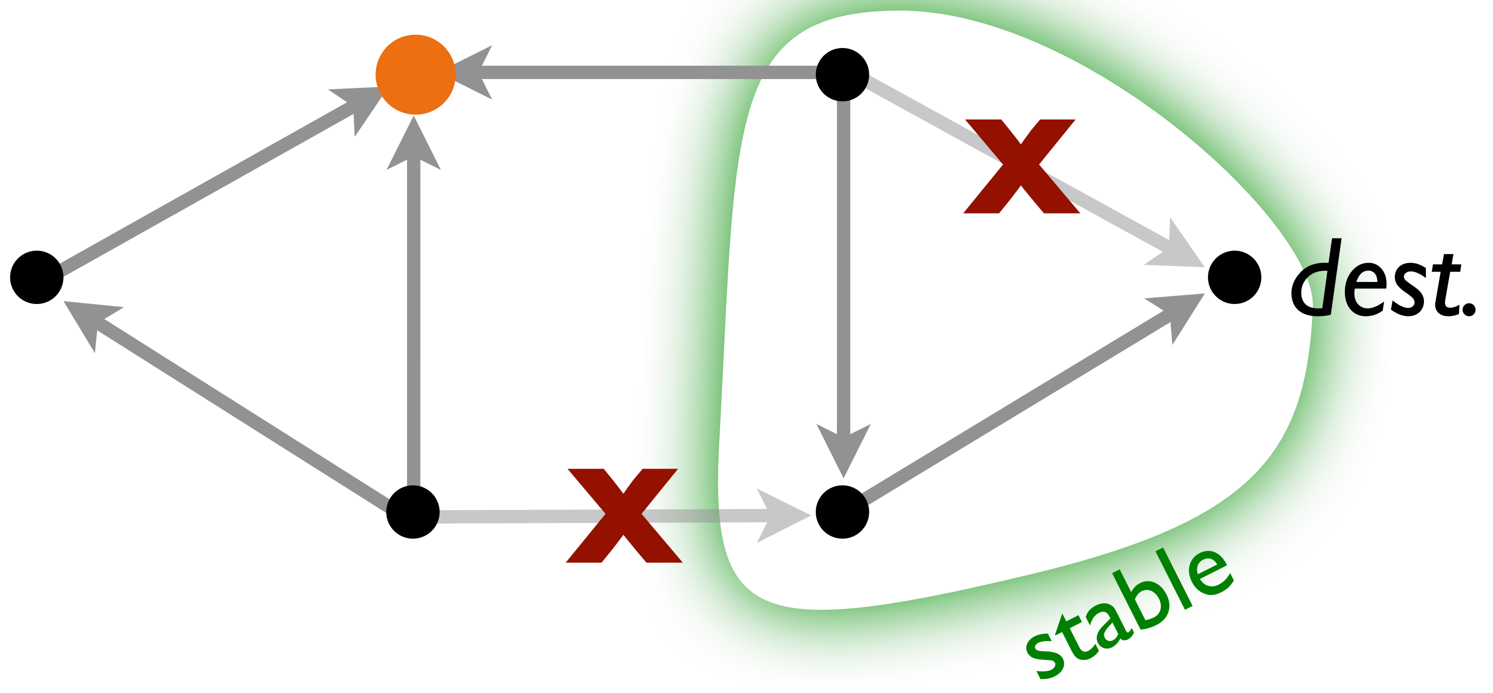
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

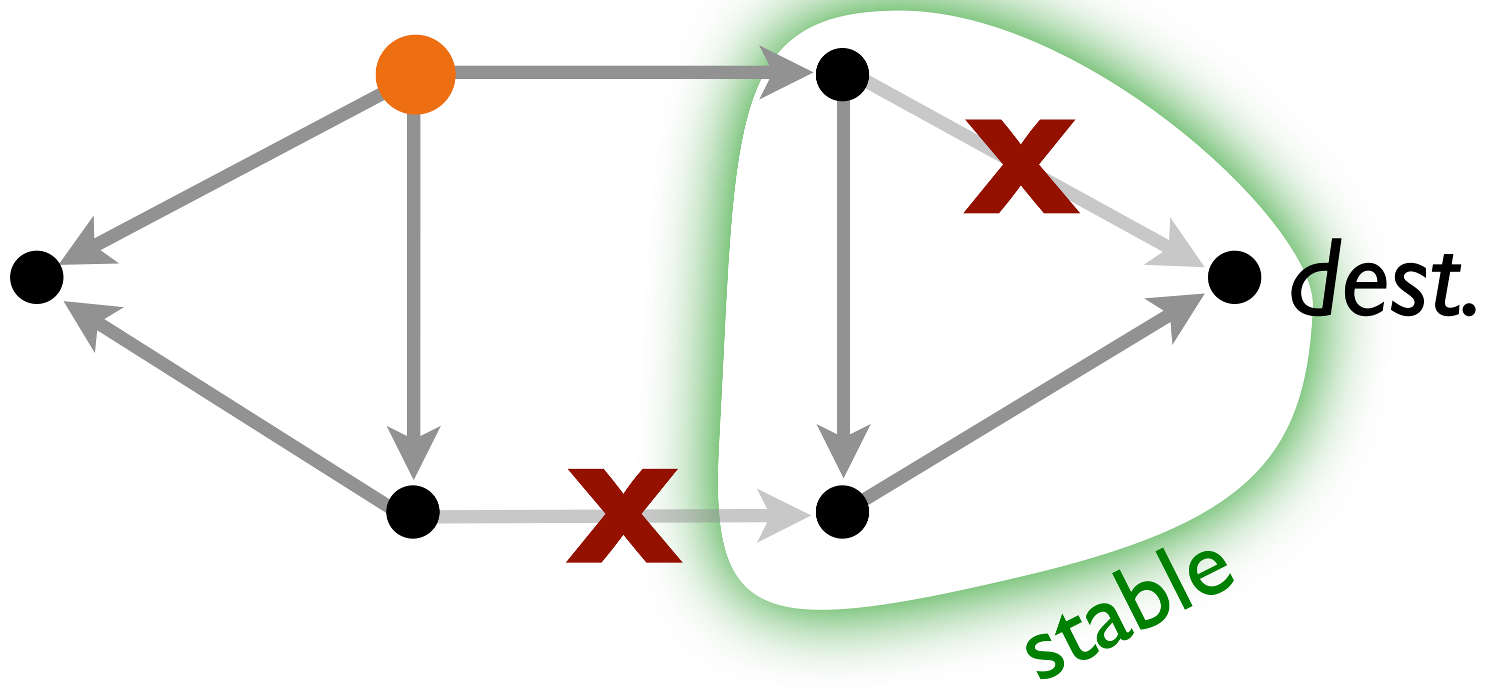
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

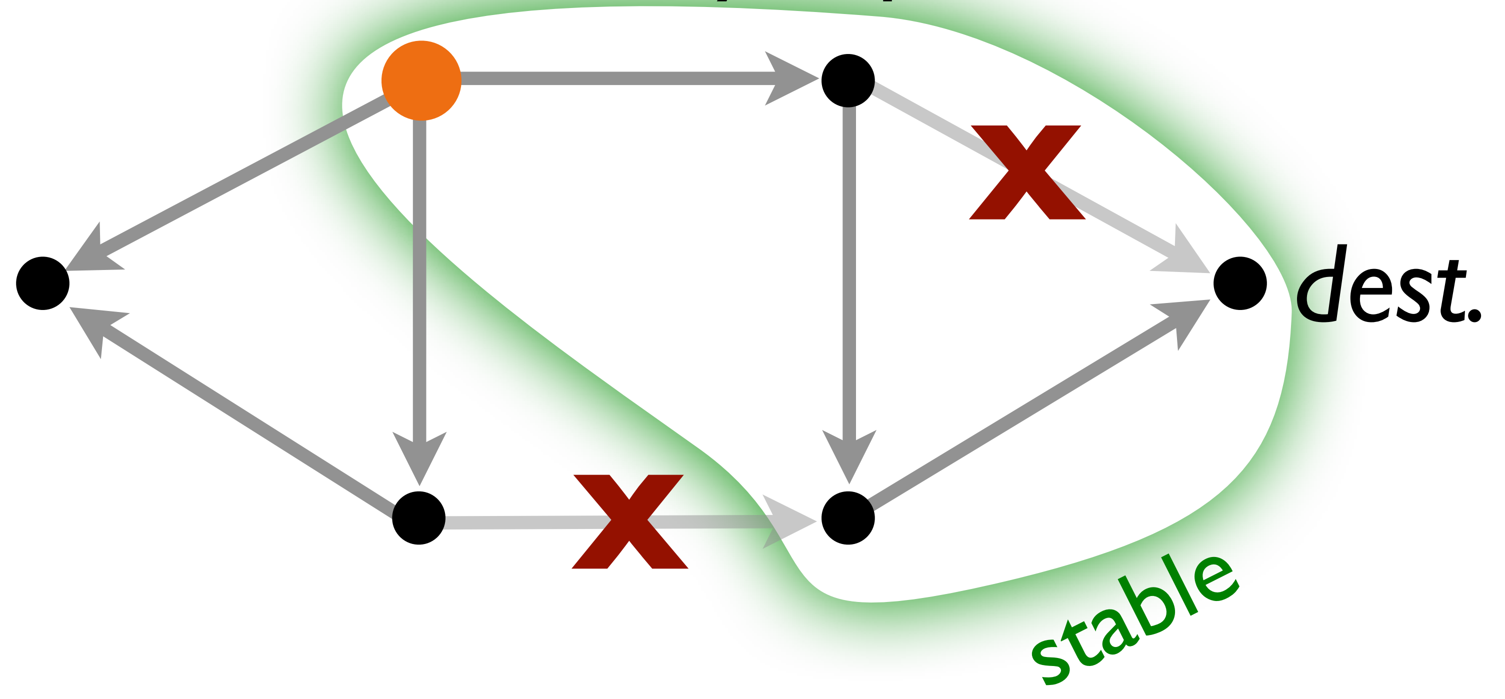
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

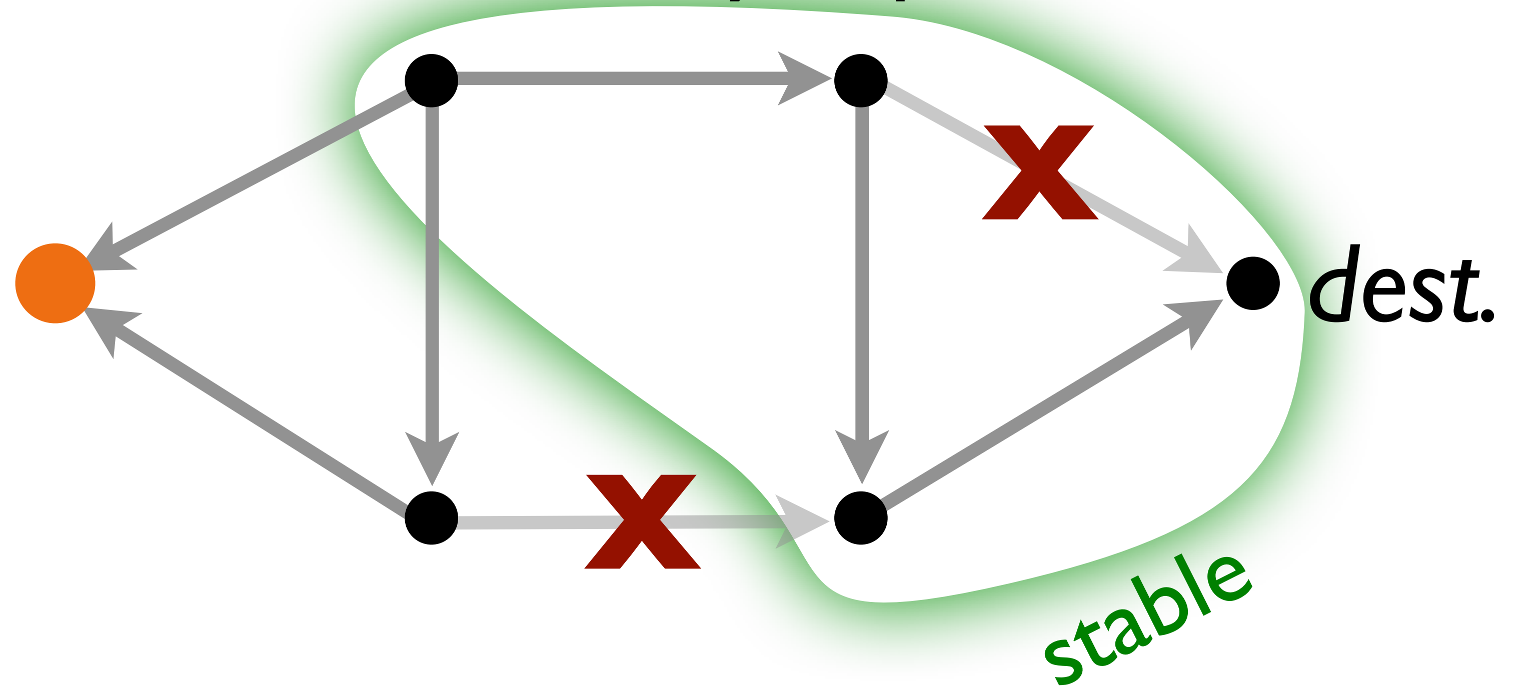
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

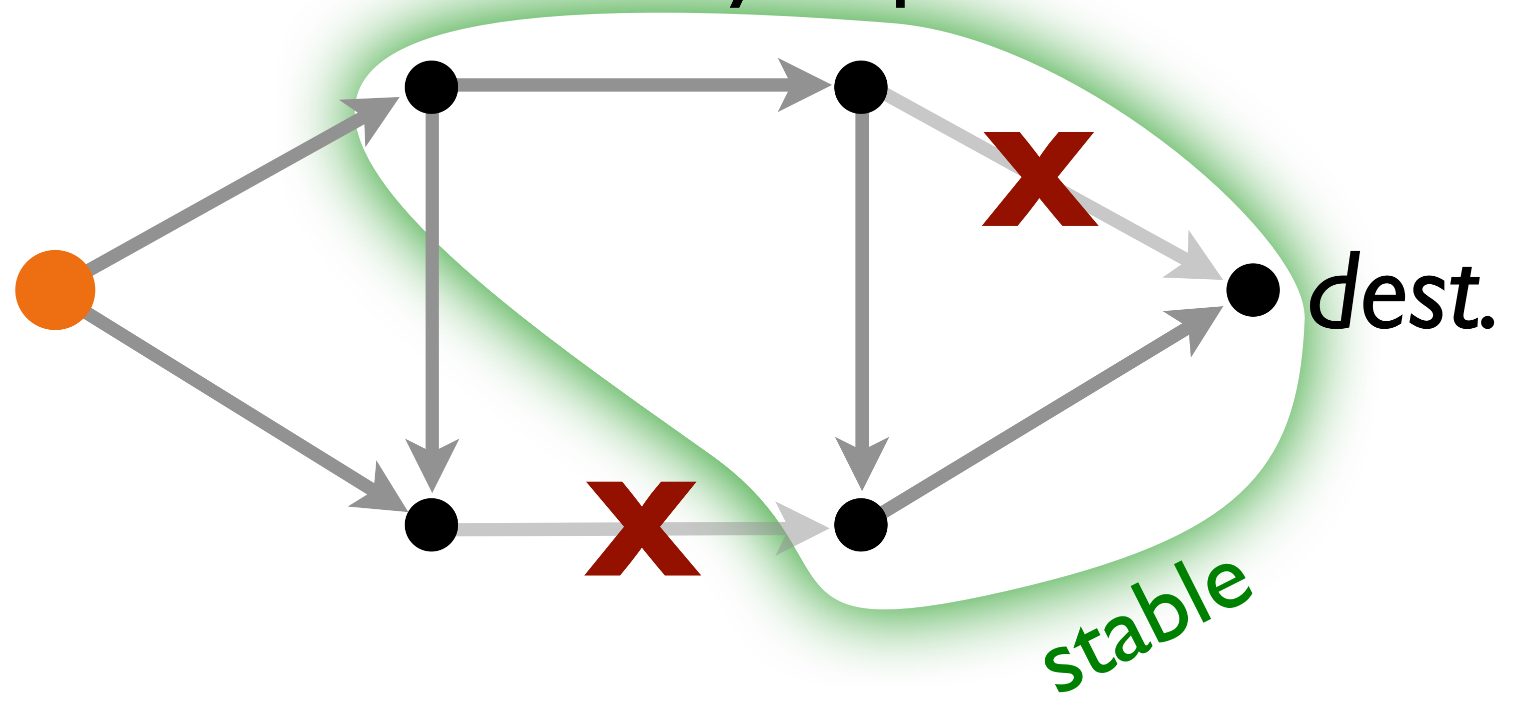
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

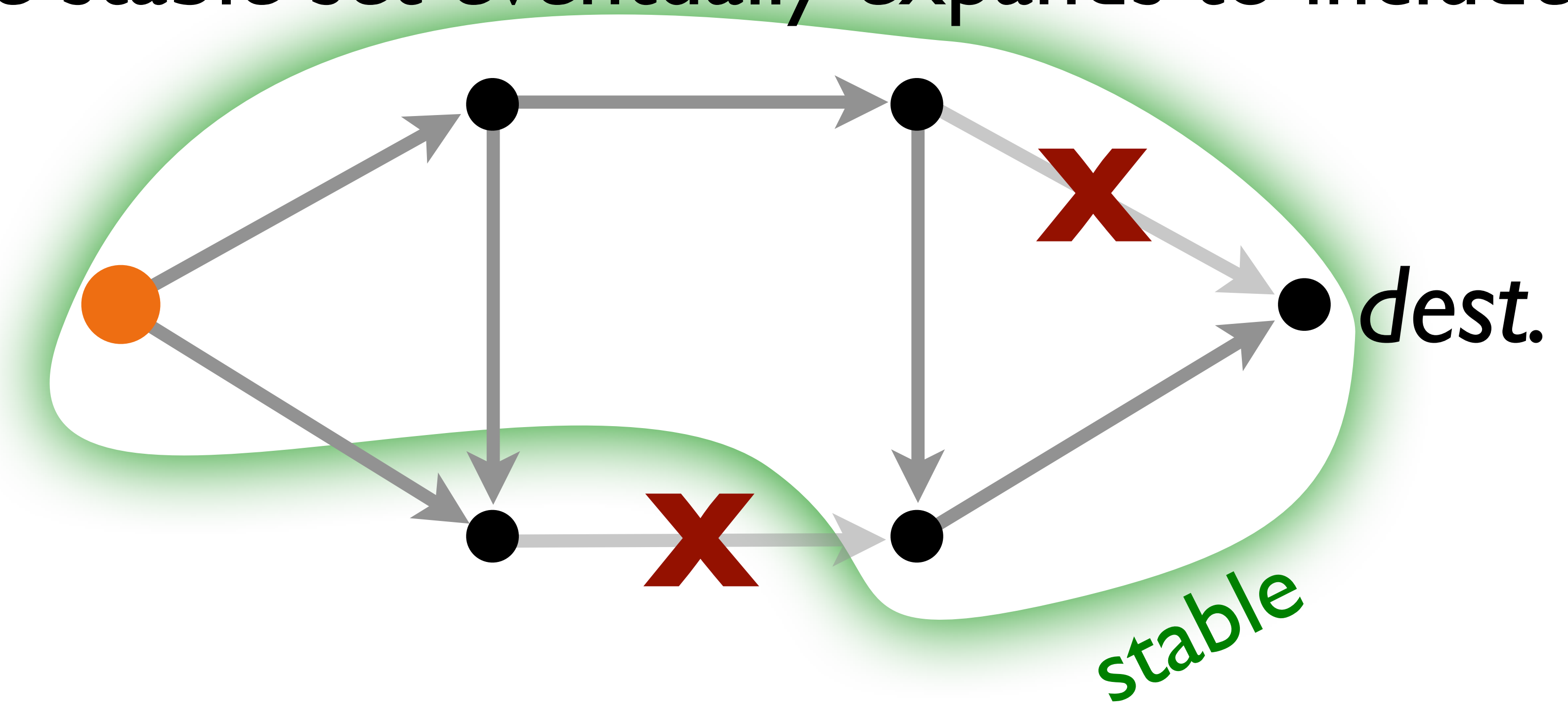
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

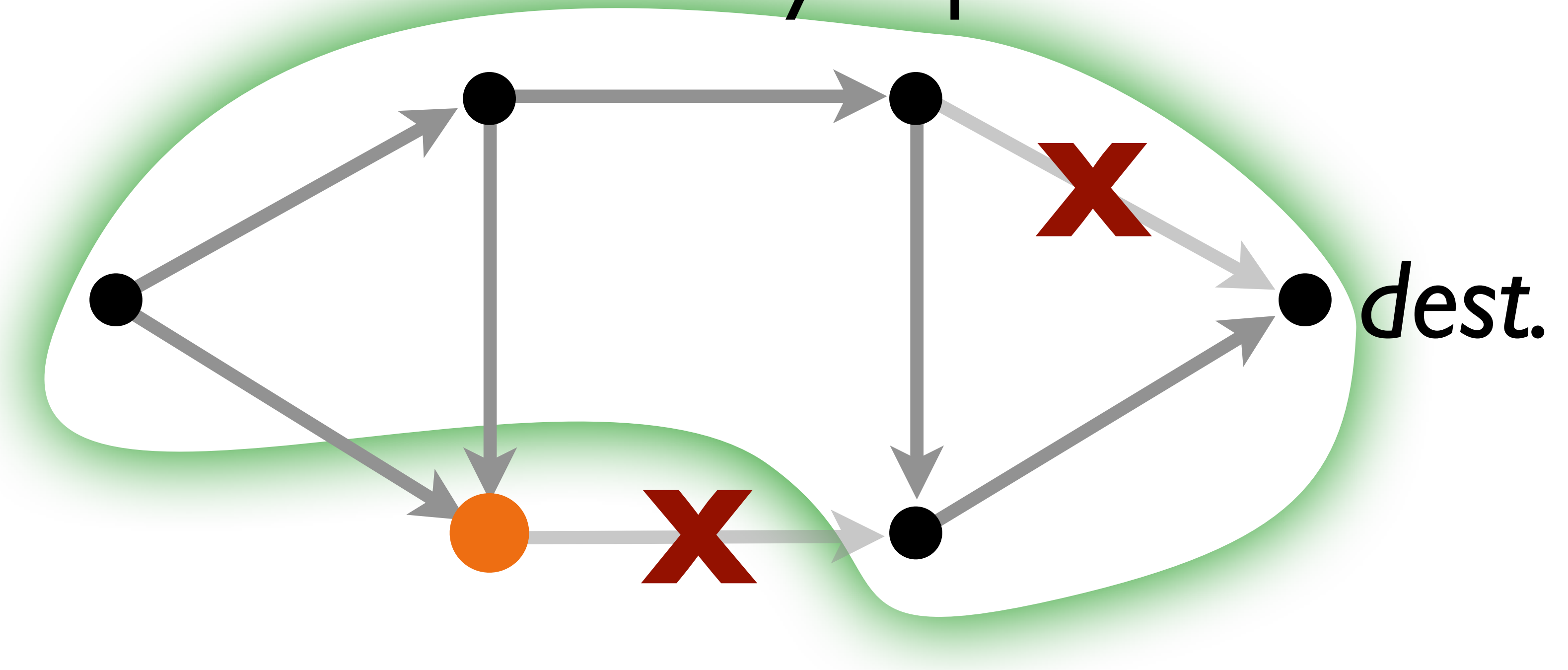
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

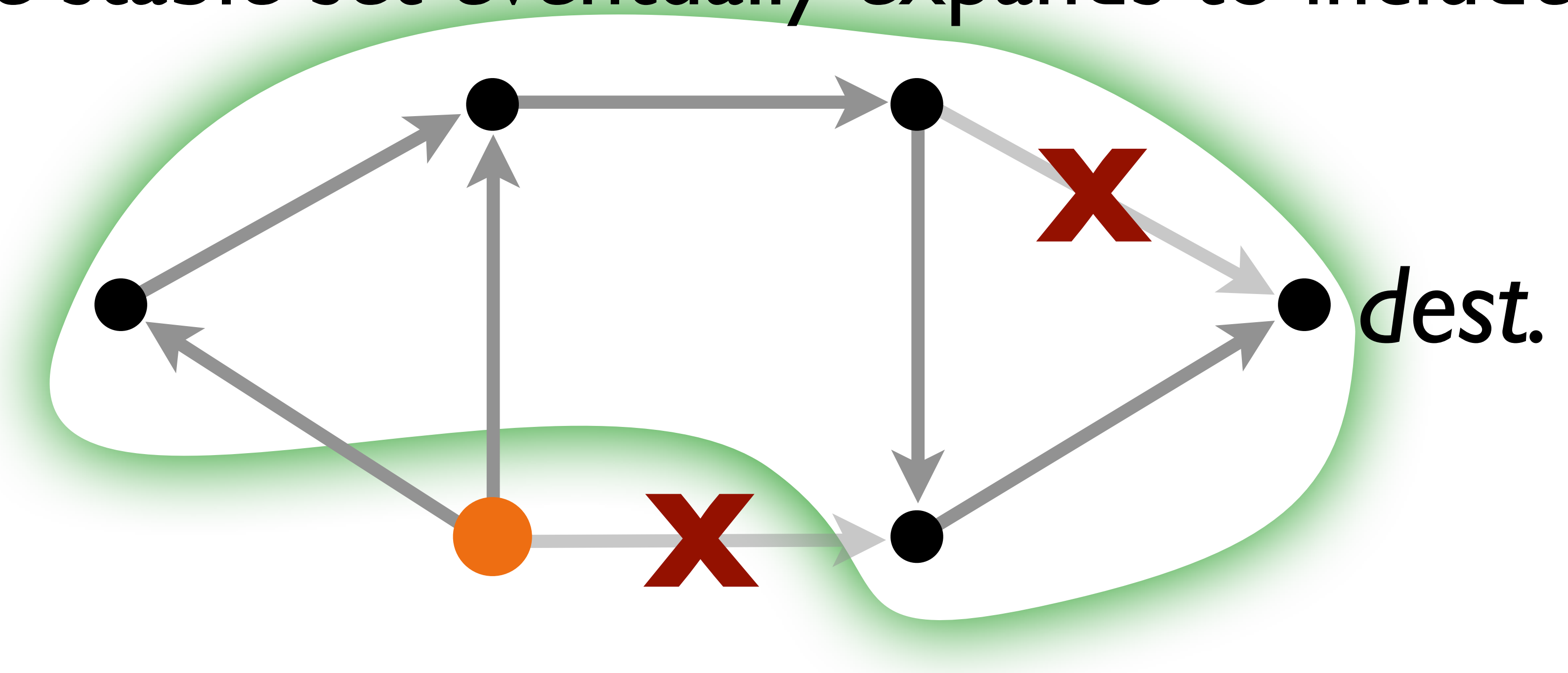
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

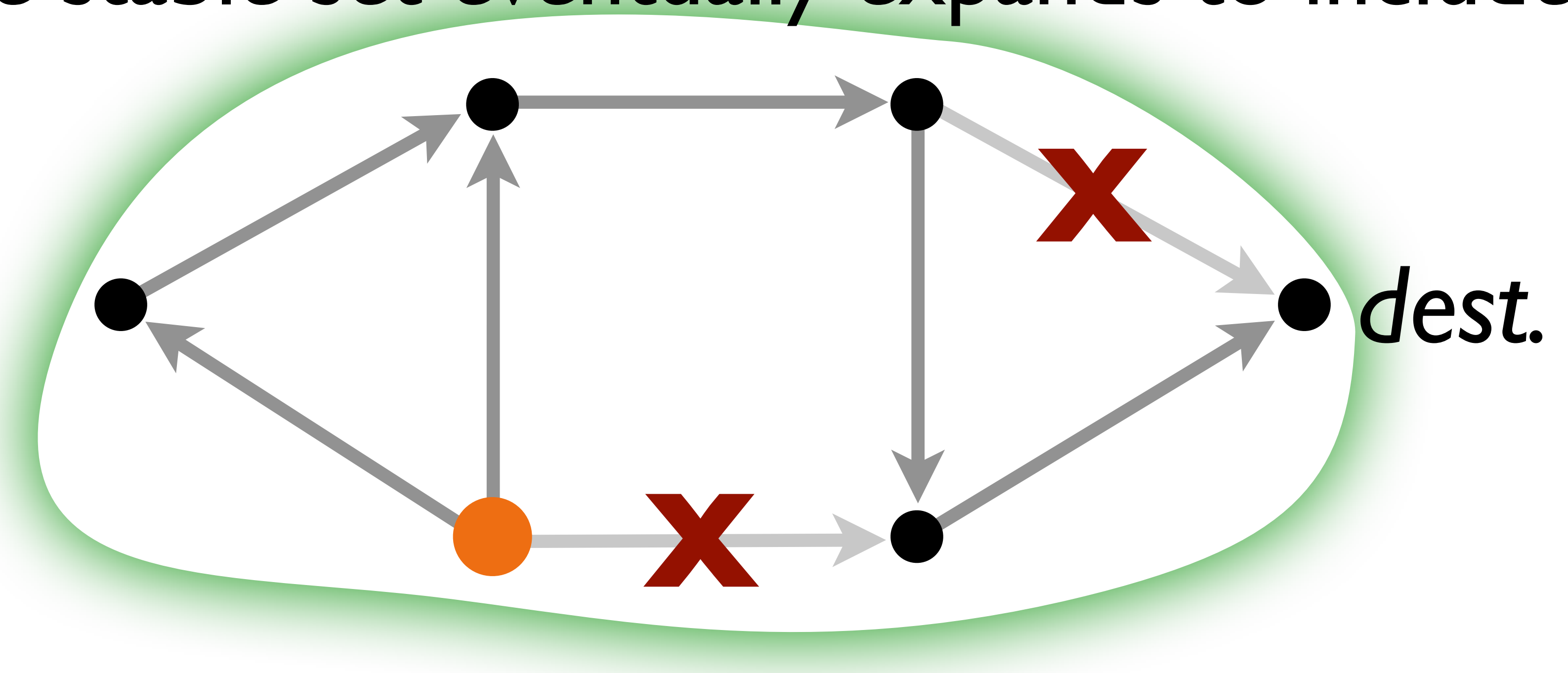
- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Guaranteed to converge

Proof

- Define stable node: no more reversals
- Destination is always stable
- If node *x* reverses adjacent to stable node *y*, then *x* also becomes stable
- Thus the stable set eventually expands to include all



Whew! Done!

Done!! … please?

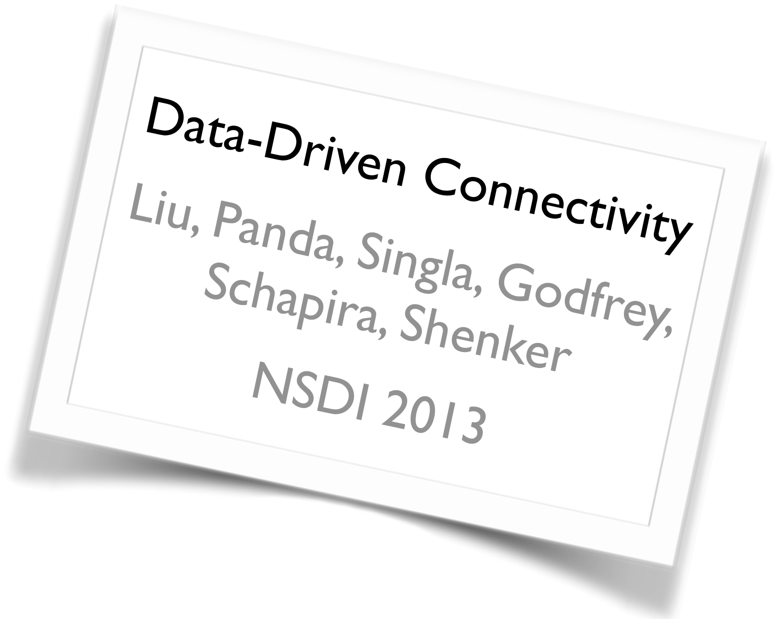
No: Protocol not yet suitable for the data plane

To reverse links, we assumed:

- router can create new control messages (“reverse this link”) in the fast path
- these control messages arrive instantly and reliably
- router has perfect information about distributed state!
  - link reversal state depends on if the other end has reversed it

Back where we started?

DDC: LR in the data plane


Data-Driven Connectivity

Liu, Panda, Singla, Godfrey, Schapira, Shenker

NSDI 2013

Key architectural point

- Make connectivity the job of the data plane
- Optimality (e.g. shortest paths) is still the job of the control plane

Problem

- LR requires sending control messages & distributed agreement on link direction – too slow for the data plane

DDC: LR in the data plane

Key algorithmic idea

- Allow stale info about link directions
- Each node can unilaterally reverse; notify neighbors later!
- Notification piggybacked on data packets using one-bit version number

Properties

- Strangely, this works...
- All events triggered by pkt arrival; no extra pkts created
- Simple bit manipulation operations

Stretch: DDC vs. MPLS

No guarantees on stretch, but empirically it's good

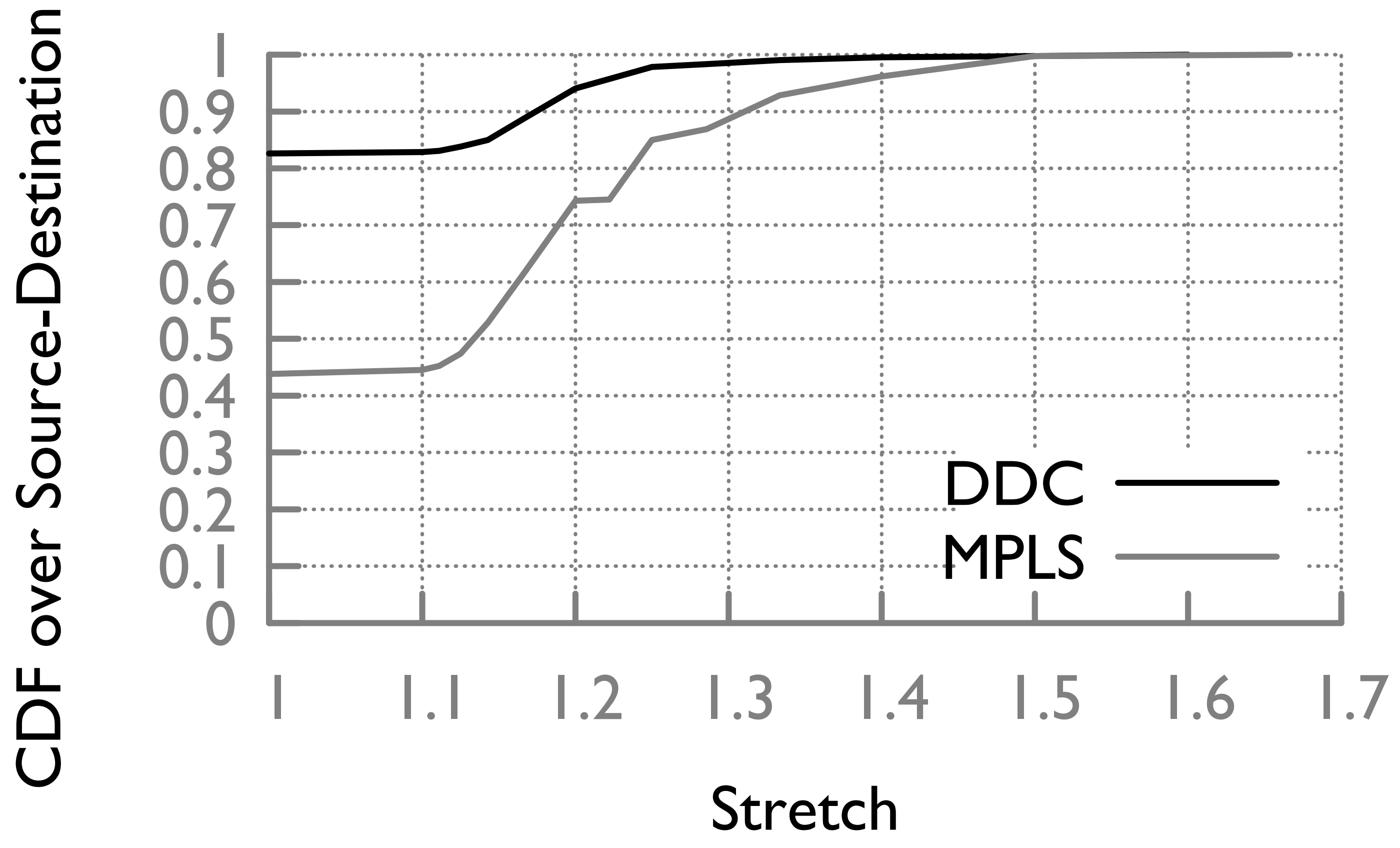


Figure 3: CDF of steady state stretch for MPLS FRR and DDC in AS2914.

Take-aways

Networks are messy and opaque

- Empirically, unreliability is common
- End-to-end measurements provide a view "inside the black box"

Highly reliable routing is possible

- requires failure response in the data plane
- single-failure protection practical, with backup paths
- surprisingly, ideal connectivity is achievable

But even that is just one of many components!

Next week: Project presentations

Two key goals

- Benchmark: Demonstrate concrete progress
- Feedback & discussion with your peers

Content

- What problem are you solving?
- Why has past work not addressed the problem?
- What is your approach for solving it?
- What are your preliminary results & progress?

Logistics

- 10 minutes total: 6:40 min presentation + 4 min discuss
- PechaKucha format: 20 slides x 20 seconds, auto-advance