MAKING NETWORKS ROBUST TO COMPONENT FAILURE

A Dissertation Outline presented by Daniel Gyllstrom
University of Massachusetts Amherst USA
Advisor: Jim Kurose

3/5/13

Talk Outline

Talk Outline

- thesis introduction

Talk Outline

- thesis introduction
- describe 3 technical chapters

Talk Outline

- thesis introduction
- describe 3 technical chapters
  - smart grid sensor failure

Talk Outline

- thesis introduction
- describe 3 technical chapters
  - smart grid sensor failure
  - failed communication links in a smart grid

Talk Outline

- thesis introduction
- describe 3 technical chapters
  - smart grid sensor failure
  - failed communication links in a smart grid
  - malicious nodes injecting false state

Talk Outline

- thesis introduction
- describe 3 technical chapters
  - smart grid sensor failure
    - failed communication links in a smart grid
    - malicious nodes injecting false state
- outline for future work and conclusions

Talk Outline

- thesis introduction
- describe 3 technical chapters
  - smart grid sensor failure
  - failed communication links in a smart grid
  - malicious nodes injecting false state
- outline for future work and co

Gyllstrom, Rosensweig and Kurose, *e-Energy 2012*

Gyllstrom, Vasudevan and Kurose, IFIP Networking 2010

Talk Outline

- thesis introduction
- describe 3 technical chapters
  - smart grid sensor failure
  - failed communication links in a smart grid
  - malicious nodes injecting false state
- outline for future work and co

Gyllstrom, Rosensweig and Kurose, *e-Energy 2012*

new

Gyllstrom, Vasudevan and Kurose, IFIP Networking 2010

Thesis Problem Statement

How can networks -- Internet, wireless networks, & networked cyber-physical systems -- be made more robust to component failure?

3 Component Failure Problems

3 Component Failure Problems

1. failure of critical power grid sensors

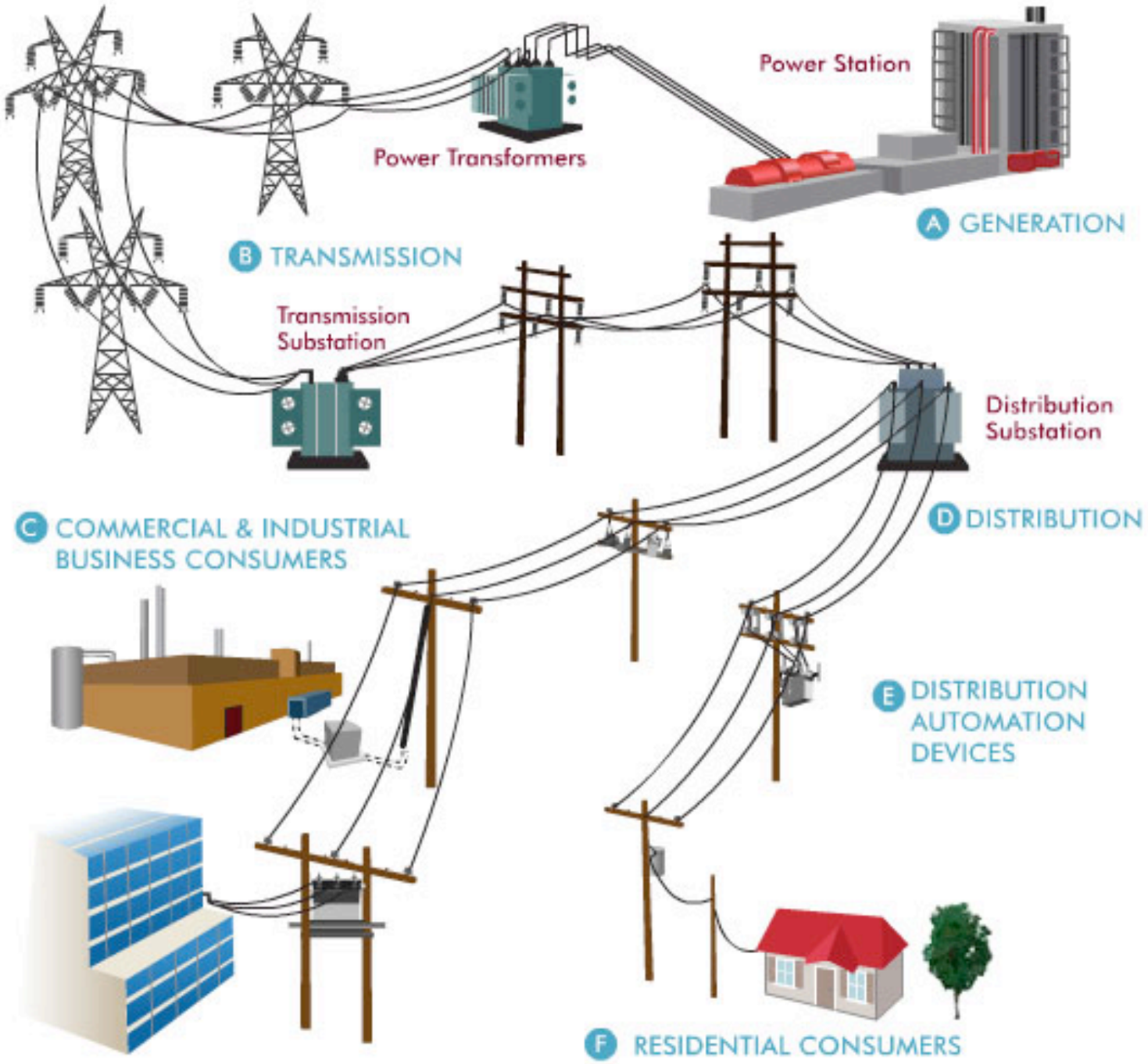
3 Component Failure Problems

1. failure of critical power grid sensors
2. link failures in a power grid communication network

3 Component Failure Problems

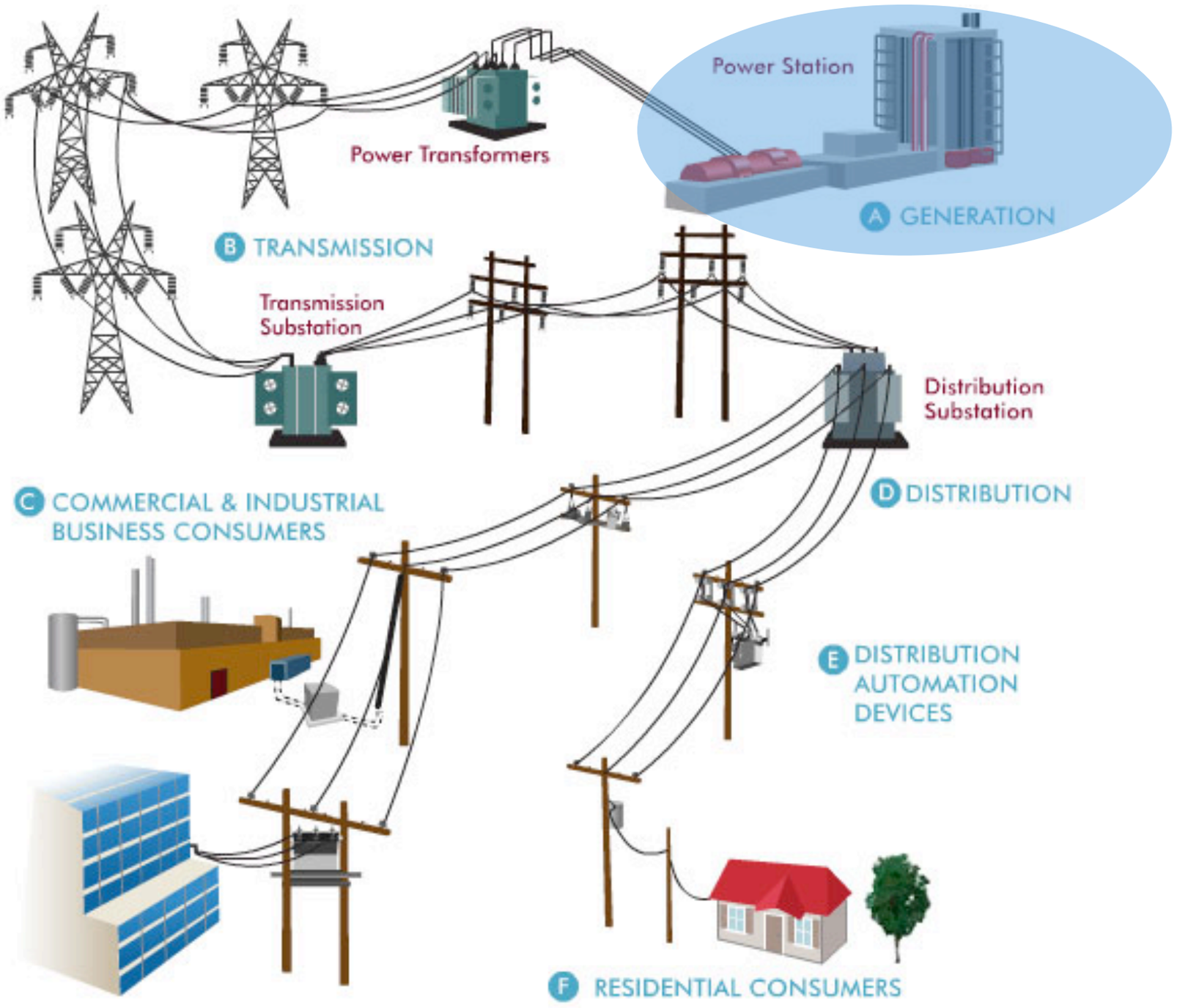
1. failure of critical power grid sensors
2. link failures in a power grid communication network
3. router failure in traditional communication networks

Power and Smart Grid Definition



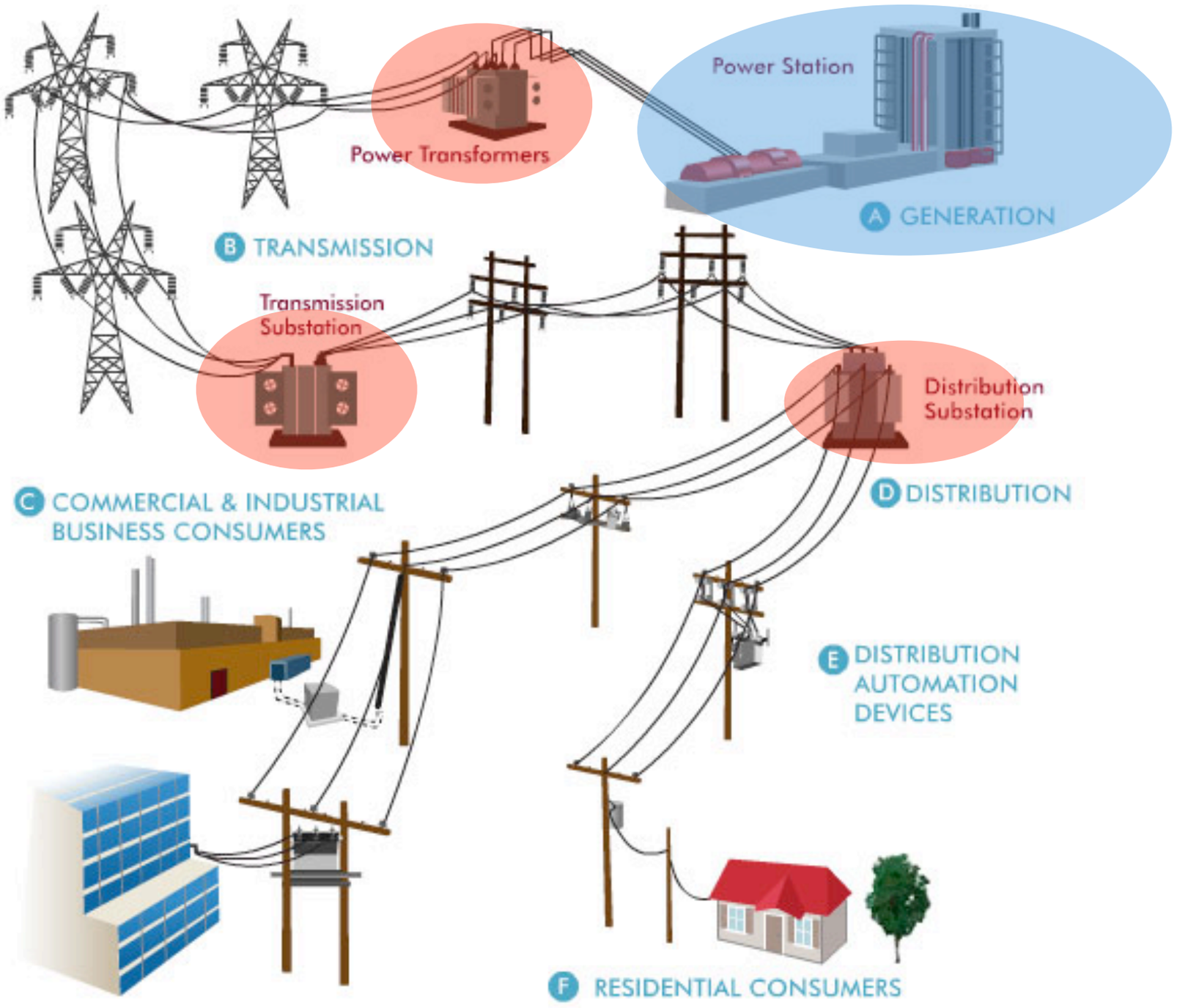
- power grid consists of buses (nodes) + transmission lines connecting buses

Power and Smart Grid Definition



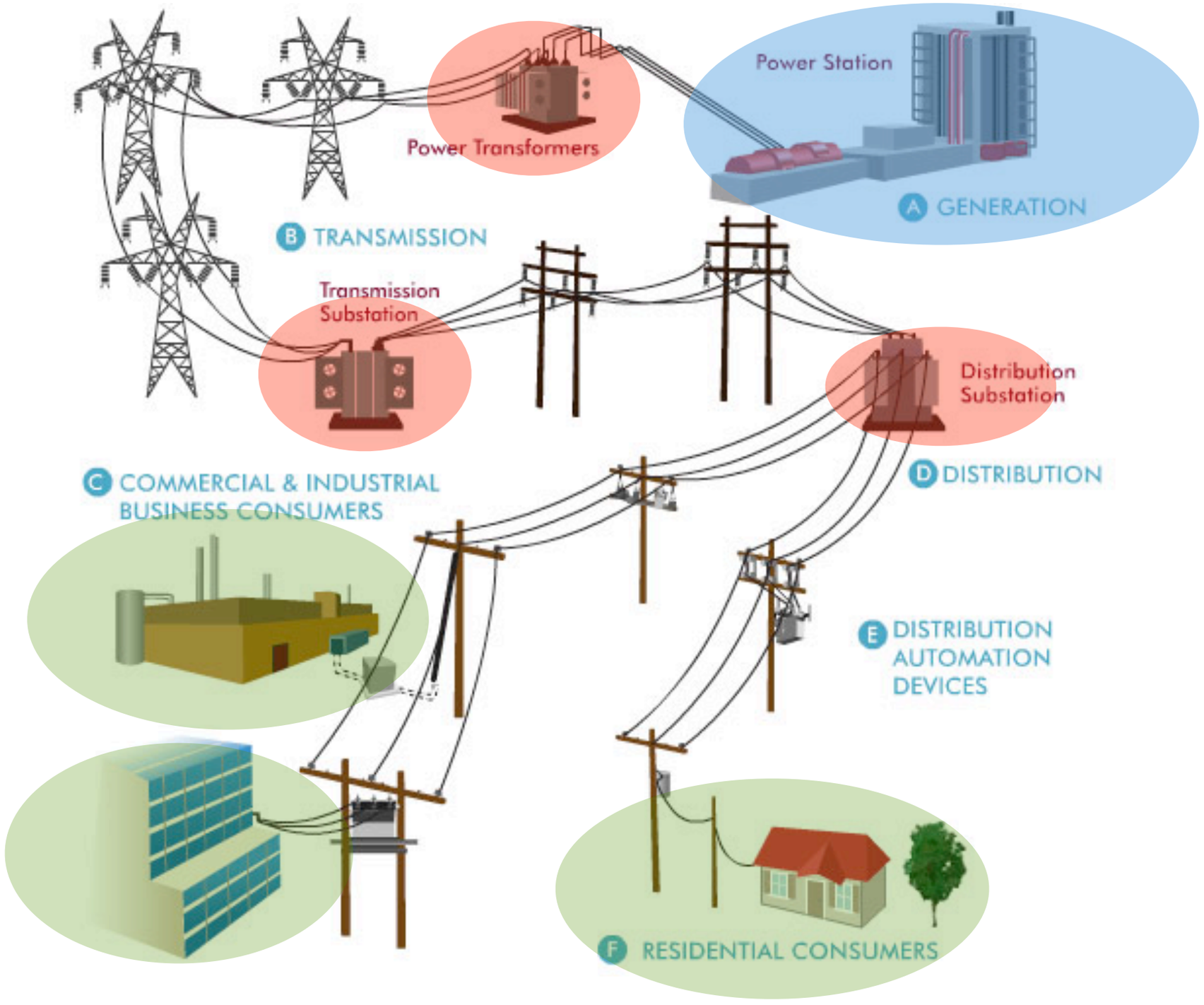
- power grid consists of buses (nodes) + transmission lines connecting buses

Power and Smart Grid Definition



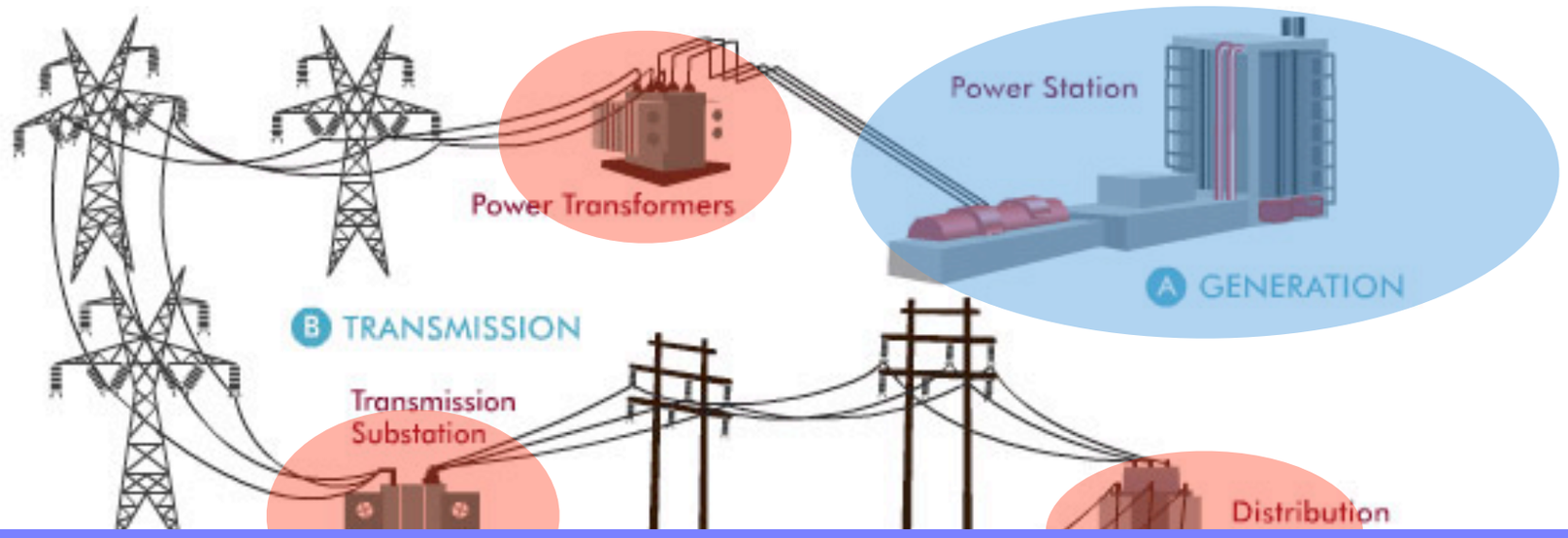
- power grid consists of buses (nodes) + transmission lines connecting buses

Power and Smart Grid Definition



- power grid consists of buses (nodes) + transmission lines connecting buses

Power and Smart Grid Definition



smart grid ≈ power grid + (sensors, computers, wide-area communication to operate & manage the grid)

- power grid consists of buses (nodes) + transmission lines connecting buses

PMU: Smart Grid Sensor

- Phasor Measurement Unit (PMU)
- high frequency voltage and current measurements
  - measures the “pulse” of the power grid

Ch 1+2: Smart Grid Failures

- Ch 1: PMU sensor failure
- Ch 2: link failures in communication network used to disseminate PMU measurements

Ch 1+2: Smart Grid Failures

- Ch 1: PMU sensor failure
- Ch 2: link failures in communication network used to disseminate PMU measurements

these failures can cause critical errors in smart grid applications used to operate the grid

theguardian

India blackouts leave 700 million without power

Power cuts plunge 20 of India's 28 states into darkness as energy suppliers fail to meet growing demand

Helen Pidd in Delhi
The Guardian, Tuesday 31 July 2012 10.48 EDT

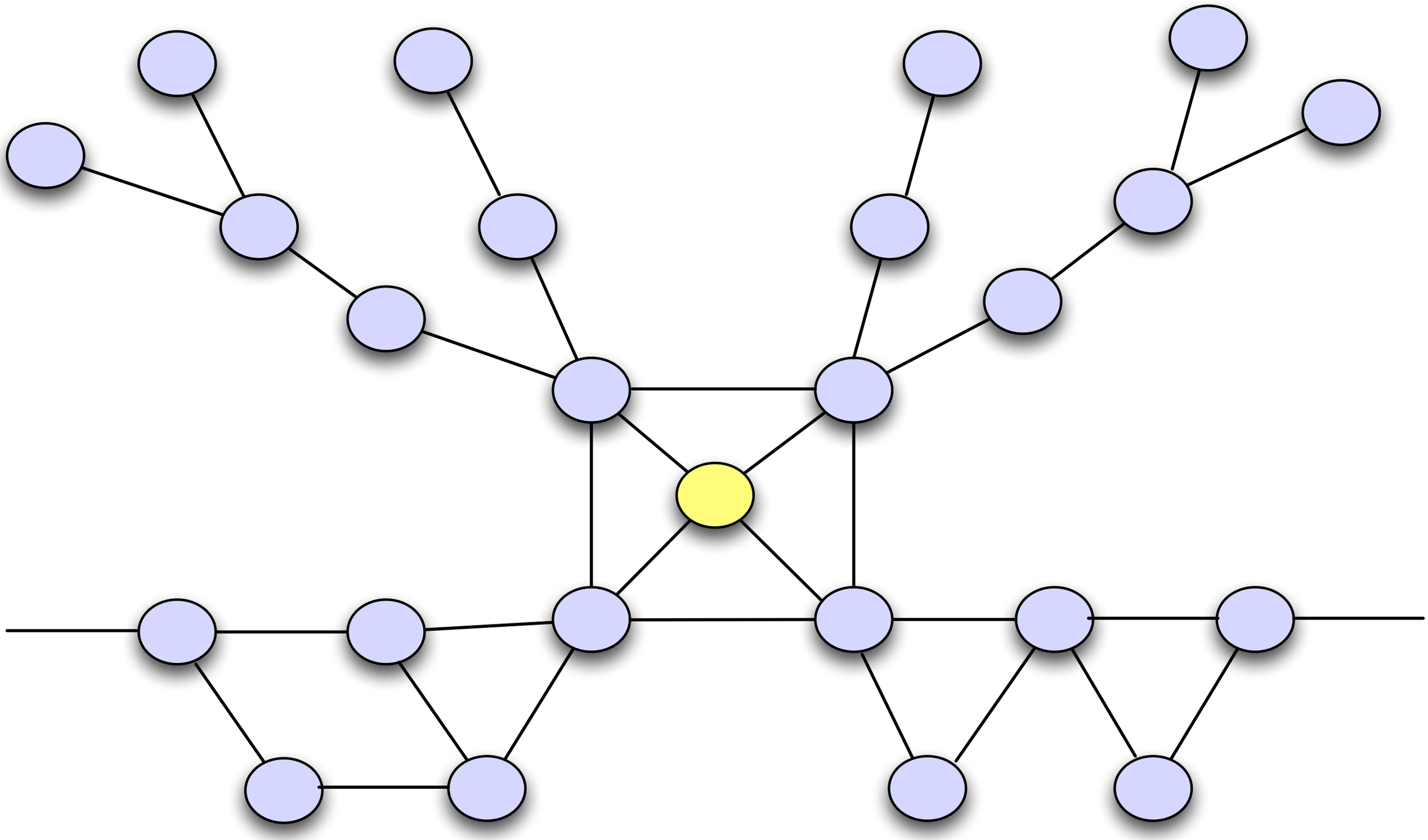
More than 700 million people in India have been left without power in the world's worst blackout of recent times, leading to fears that protests and even riots could follow if the country's electricity supply continues to fail to meet growing demand.

Twenty of India's 28 states were hit by power cuts, along with the capital, New Delhi, when three of the country's five electricity grids failed at lunchtime.

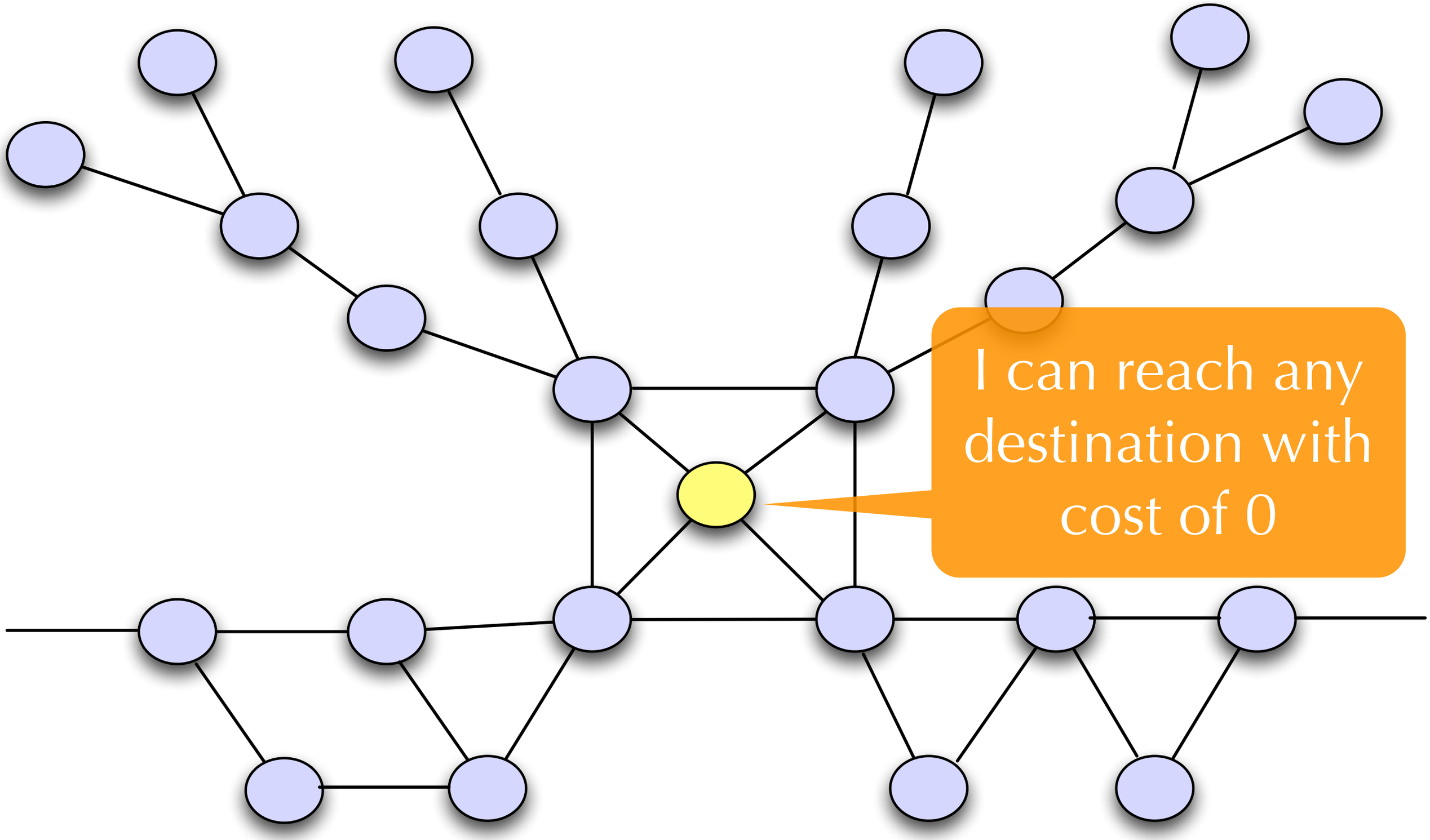
As engineers struggled for hours to fix the problem, hundreds of trains failed, leaving passengers stranded along thousands of miles of track from Kashmir in the north to Nagaland on the eastern border with Burma.

Traffic lights went out, causing jams in New Delhi, Kolkata and other cities. Surgical operations were cancelled across the country, with nurses at one hospital just outside Delhi having to operate life-saving equipment manually when back-up generators failed.

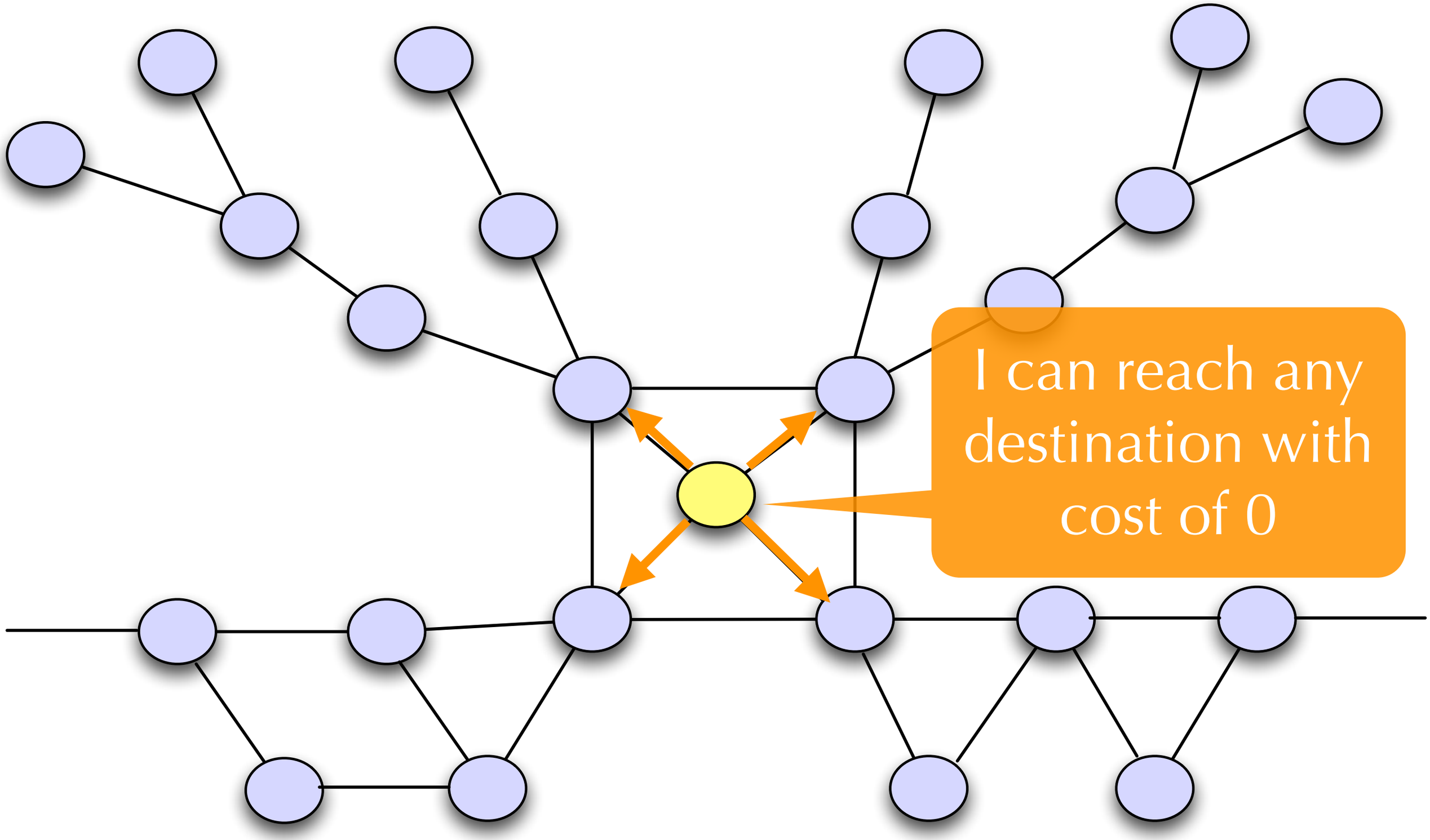
Ch 3: Network Router Failure



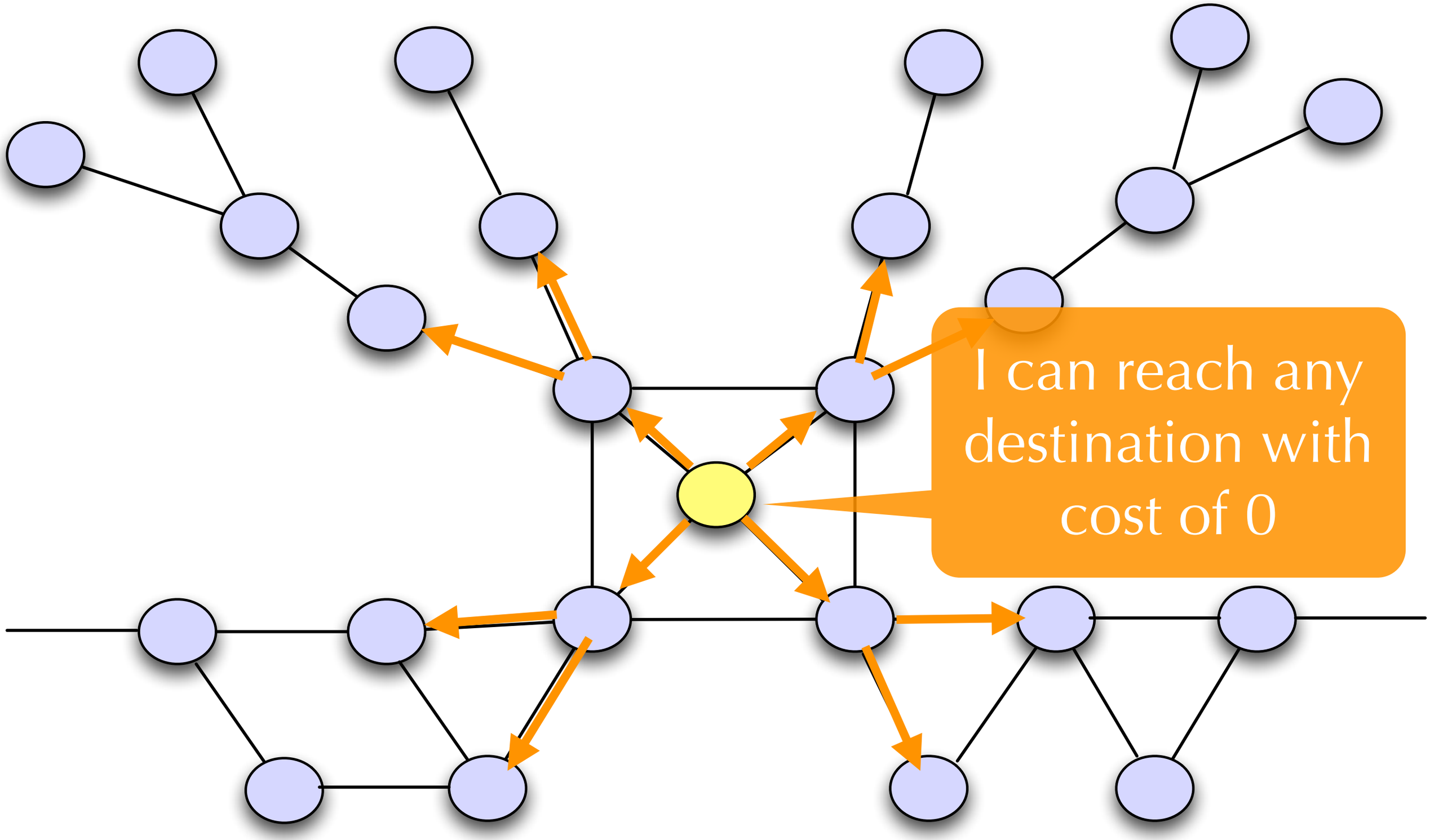
Ch 3: Network Router Failure



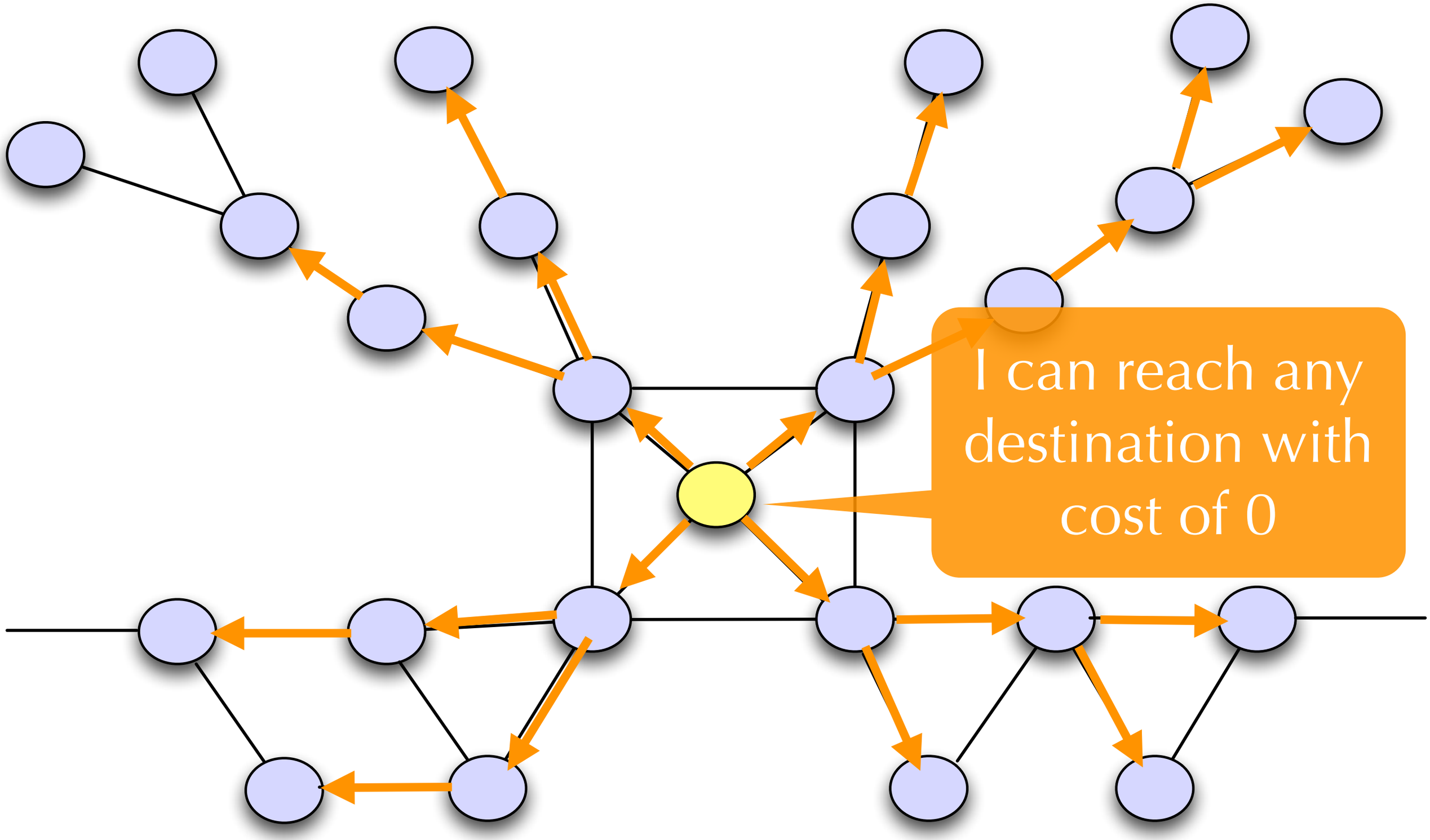
Ch 3: Network Router Failure



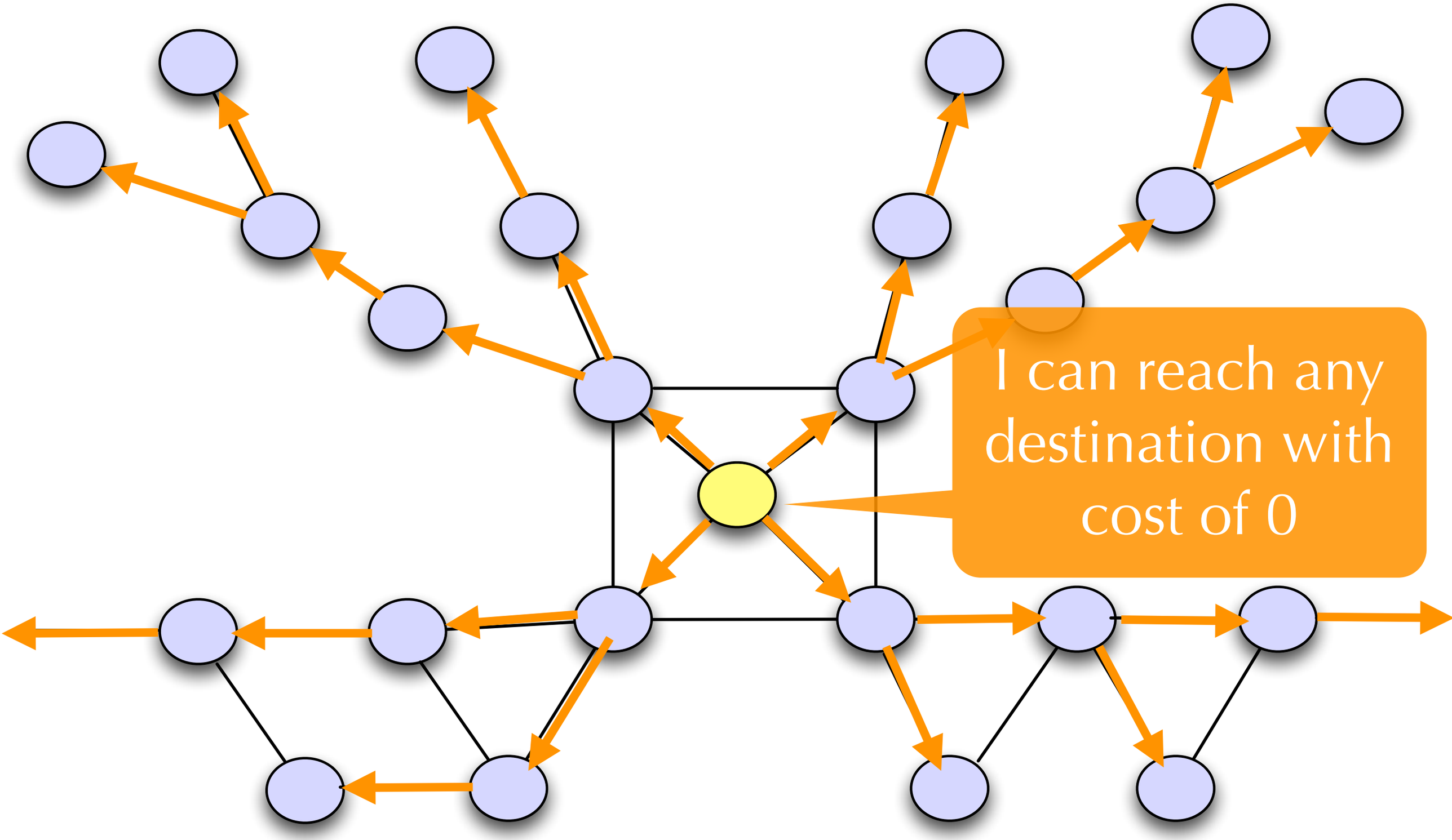
Ch 3: Network Router Failure



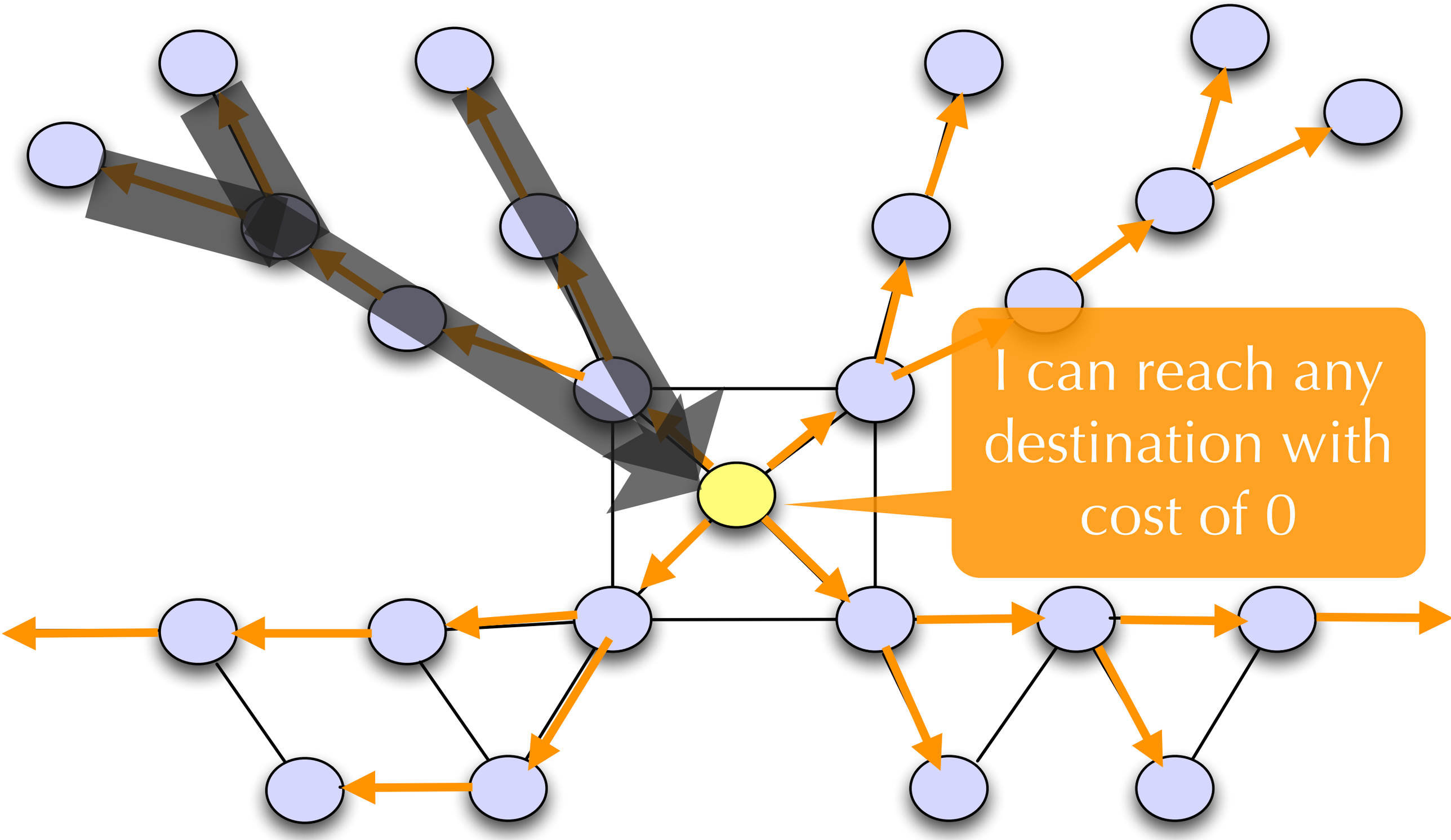
Ch 3: Network Router Failure



Ch 3: Network Router Failure



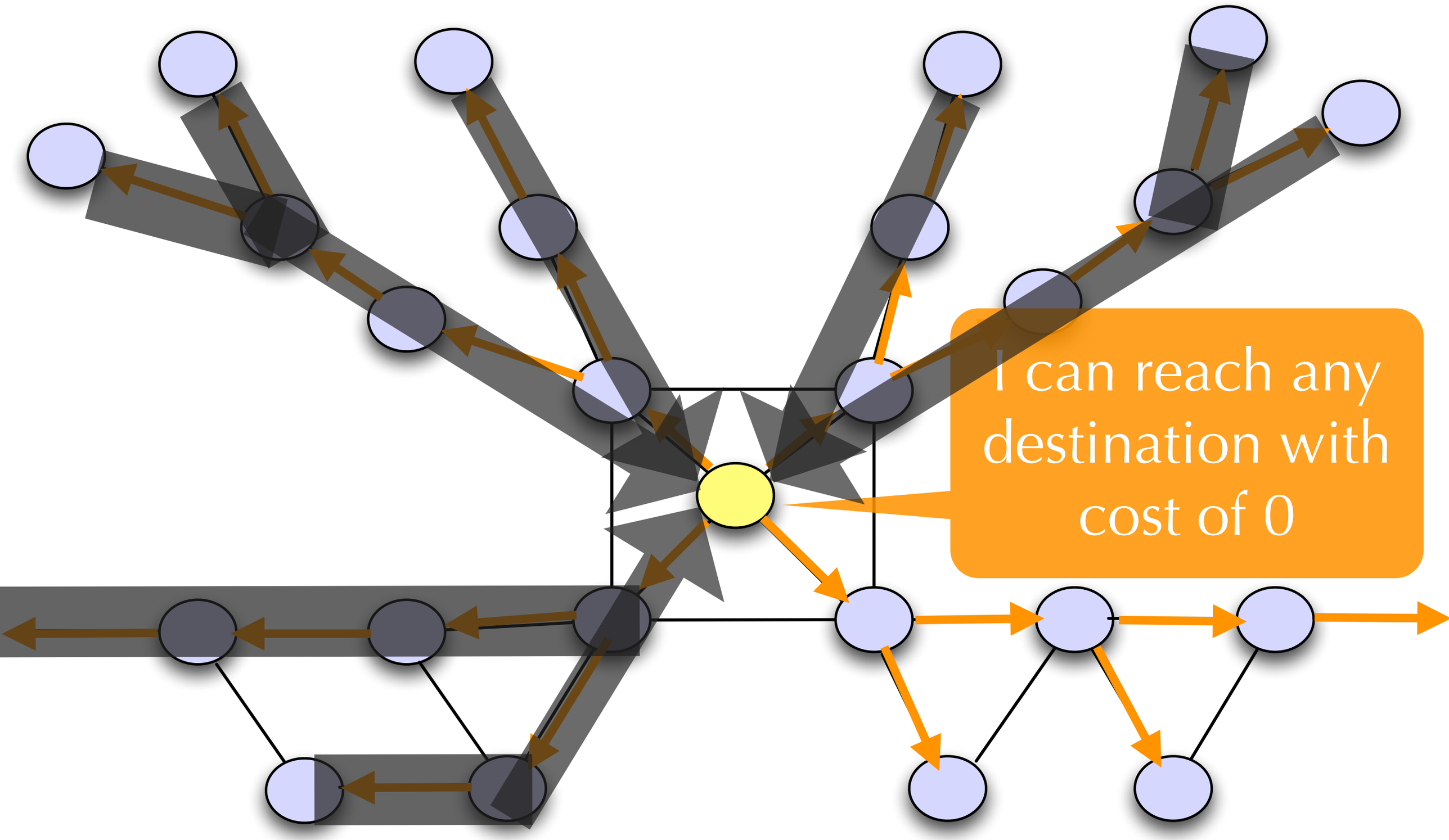
Ch 3: Network Router Failure



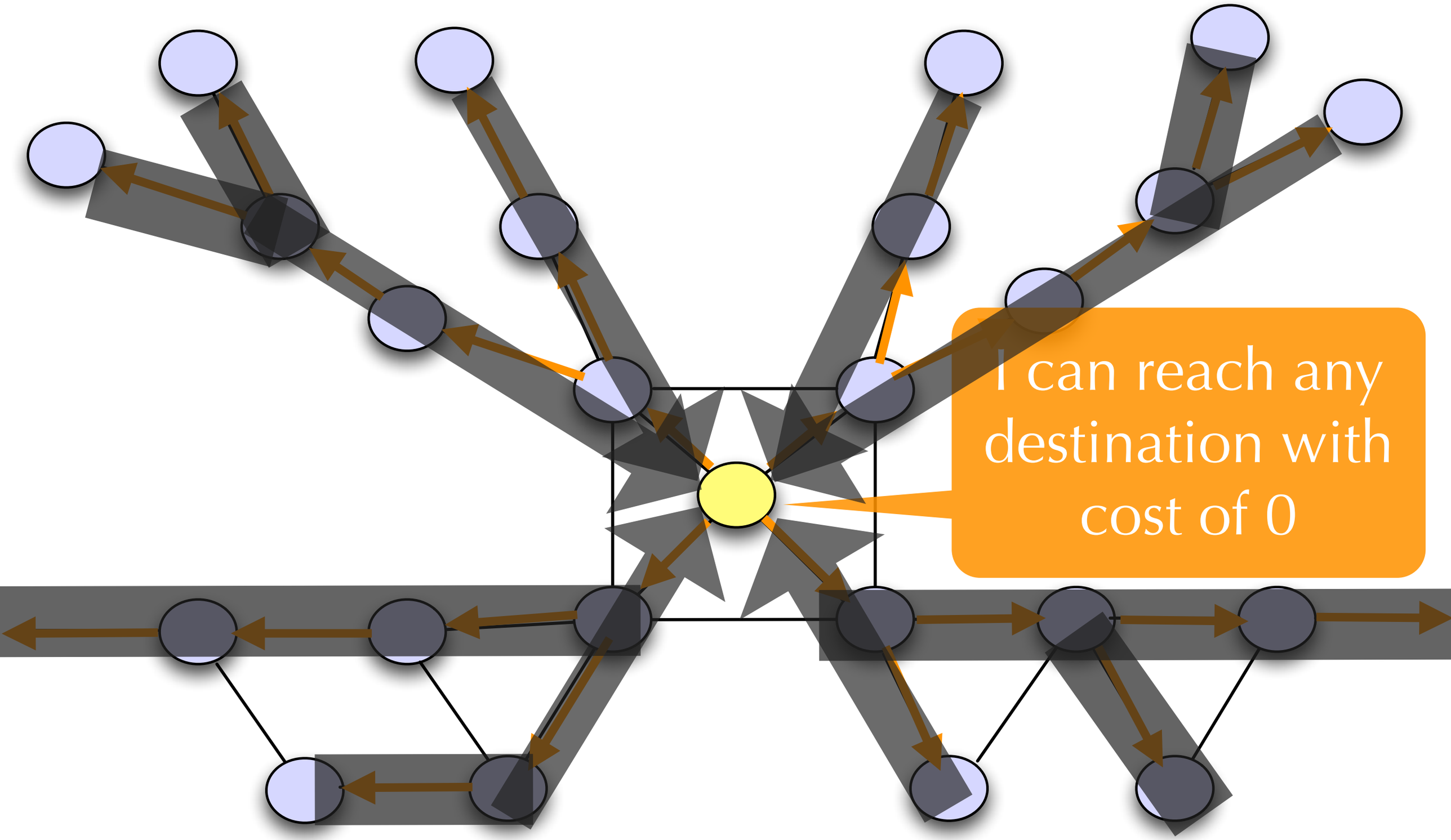
Ch 3: Network Router Failure

I can reach any destination with cost of 0

Ch 3: Network Router Failure



Ch 3: Network Router Failure



April 25, 1997 7:00 PM PDT

Router glitch cuts Net access

By CNET News.com Staff
Staff Writer, CNET News

Related Stories

Net blackout hits some regions

April 25, 1997

Software blamed for AOL blackout

February 5, 1997

WorldNet service restored

November 9, 1996

Web gets an Olympian workout

July 13, 1996

What started out as a router glitch at a small Internet service provider in Virginia today triggered a major outage in Internet access across the country, lasting more than two hours in some places.

The problem started this morning at 8:30 a.m. PT when MAI Network Services, an ISP headquartered in a McLean, Virginia, unwittingly passed some bad router information from one of its customers onto Sprint, one of the largest Internet backbone operators in North America. Because Sprint's backbone is used by so many other smaller ISPs, the router problem was echoed, causing temporary network outages across the country and, perhaps, internationally.

The outage underscored the fragility of the infrastructure that underlies the global network and how easily a problem with one small ISP can be amplified throughout the Internet. Even so, the Net displayed a remarkable resilience that seems to disprove its doomsayers, who have predicted that the network is on the verge of collapse.

"This particular thing was a confluence of two or three things happening--human error, bug, and some policy problems--that all came together on the same day," said Jack Rickard, publisher of *BoardWatch* magazine.

"There are probably a hundred guys in back rooms keeping this stuff together, just barely," Ricard said of the Internet.

Automated Recovery Is Needed

- automated recovery needed to reduce
  - short-term disruption
  - increase long-term network survivability

Automated Recovery Is Needed

- automated recovery needed to reduce
  - short-term disruption
  - increase long-term network survivability

this thesis designs algorithms to make networks robust to these component failures

Unifying The 3 Subproblems

- each problem considers network robustness in the face of component failure
- our solutions
  - preplanned recovery for smart grid apps where reliability is key
  - on-demand recovery for distributed network algorithms

Contributions (1/3)

1. define 4 new problems for smart grid sensor placement that aim to max measurement coverage + provide measurement error detection

- prove each is NP-Complete
- develop approximation algs + evaluate w/ simulations

Contributions (2/3)

2. define problem of recovery from link failure in smart grid communication network
   - outline OpenFlow-based link failure detection algorithm
   - outline 3 algs to compute backup multicast trees

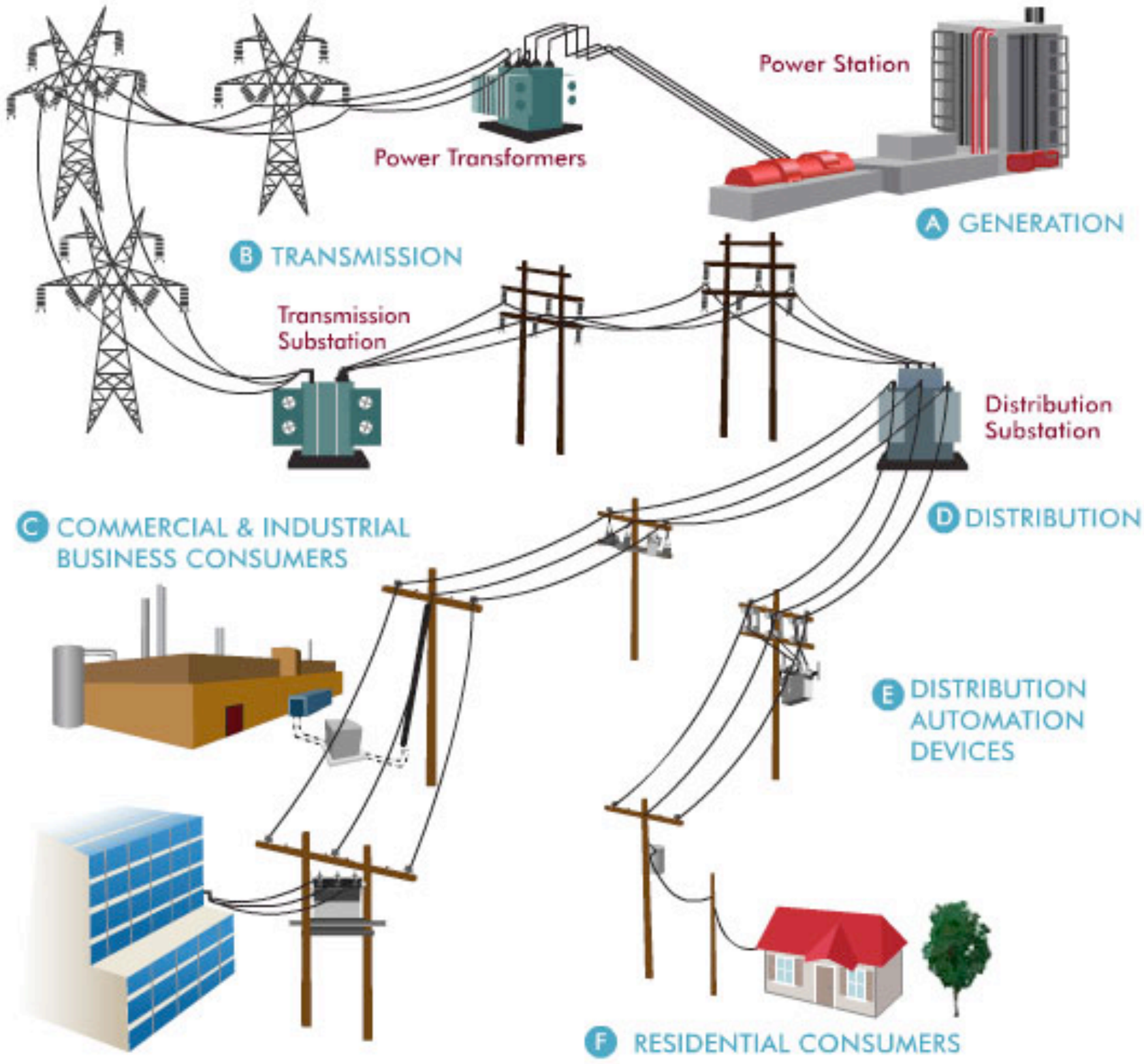
Contributions (3/3)

3. design + develop 3 new algorithms for recovery from injection of false state in distributed sys.
   - evaluate w/ simulations + derive complexity bounds

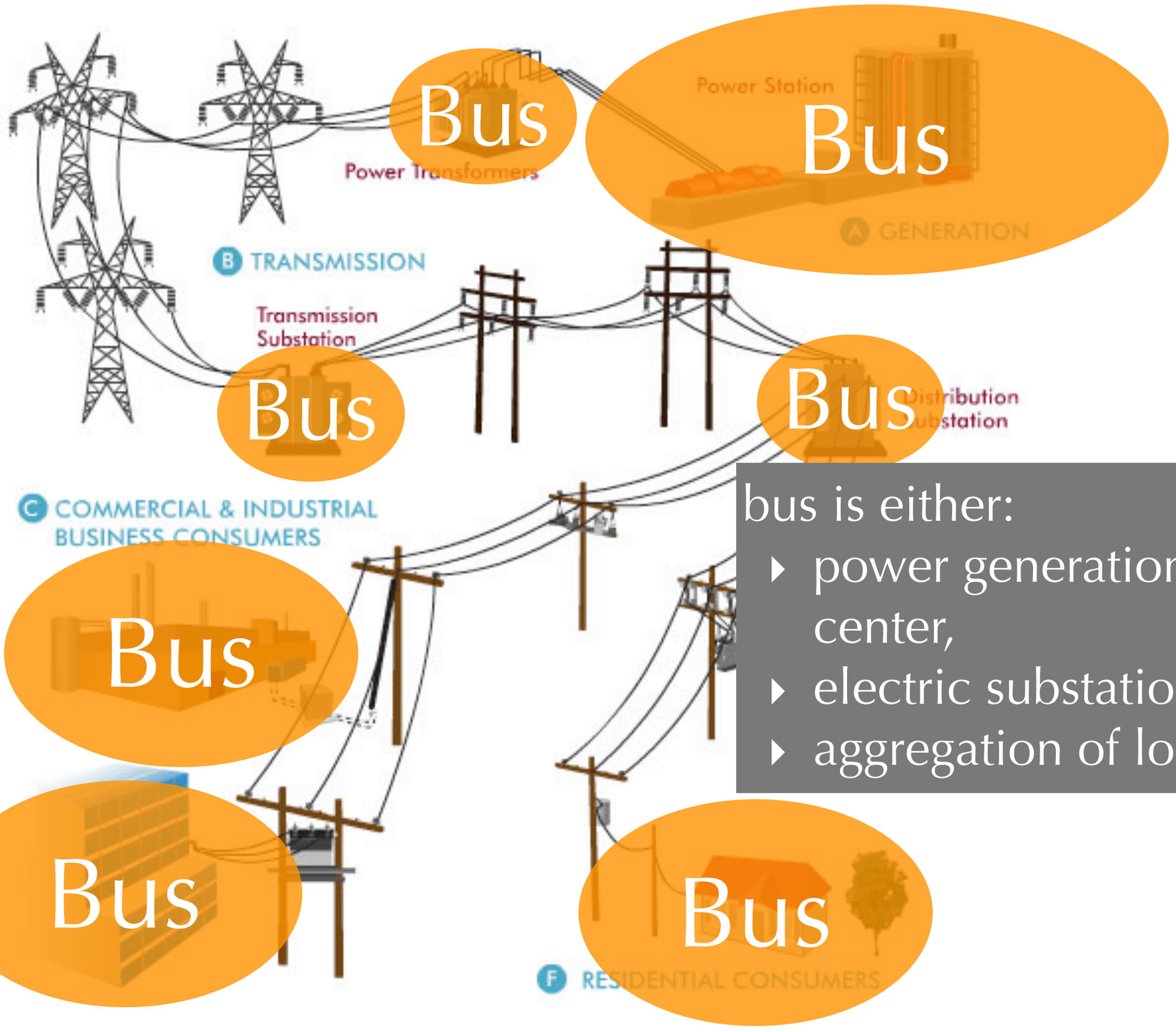
Talk Outline

- thesis introduction
- placement of smart grid sensors to enable measurement error detection
- recovery from failed communication links in a smart grid
- recovery from malicious nodes injecting false routing state
- outline for future work and conclusions

Refresher: Power Grid Definition



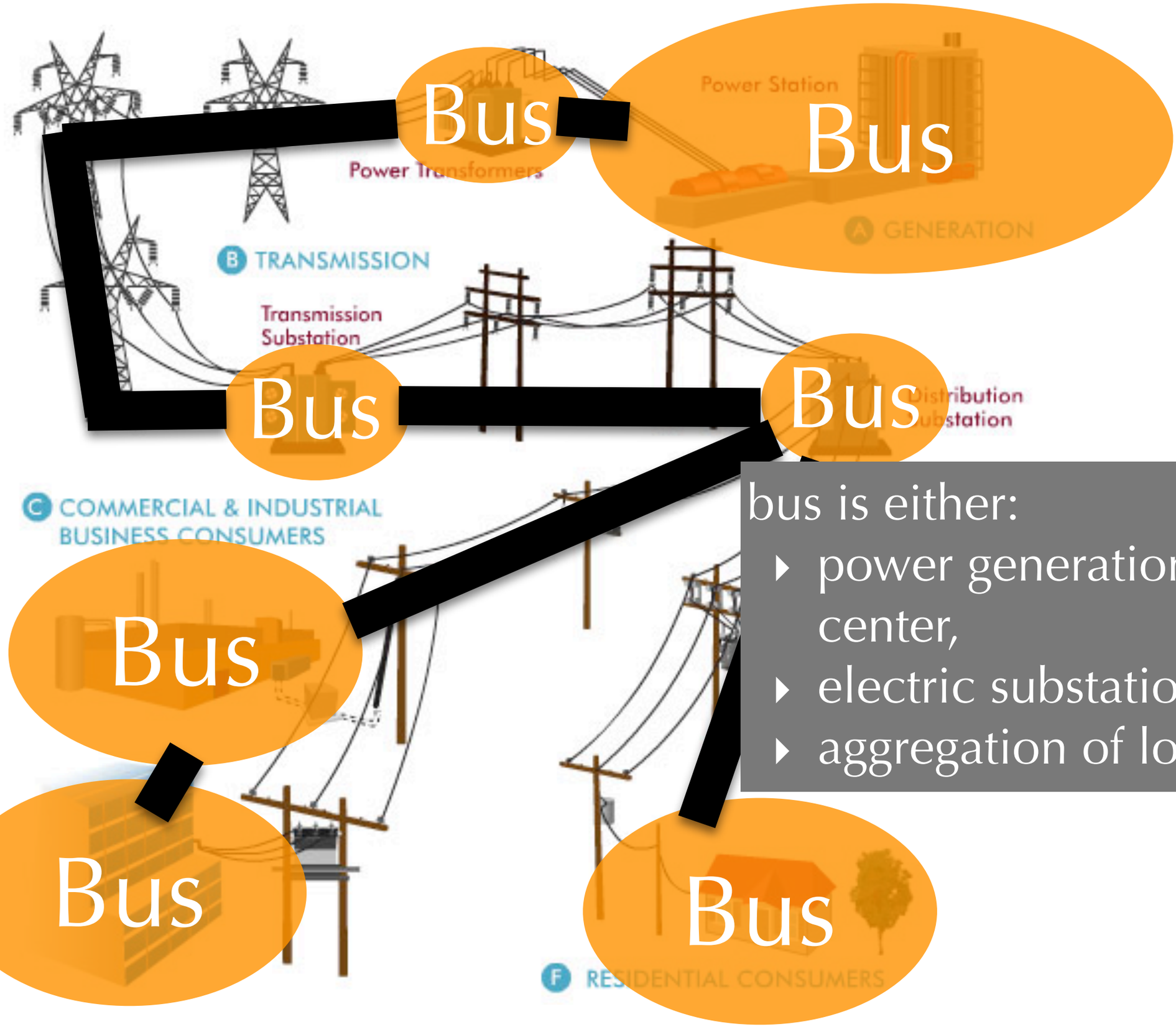
Refresher: Power Grid Definition



bus is either:

- power generation center,
- electric substation, or
- aggregation of loads

Refresher: Power Grid Definition



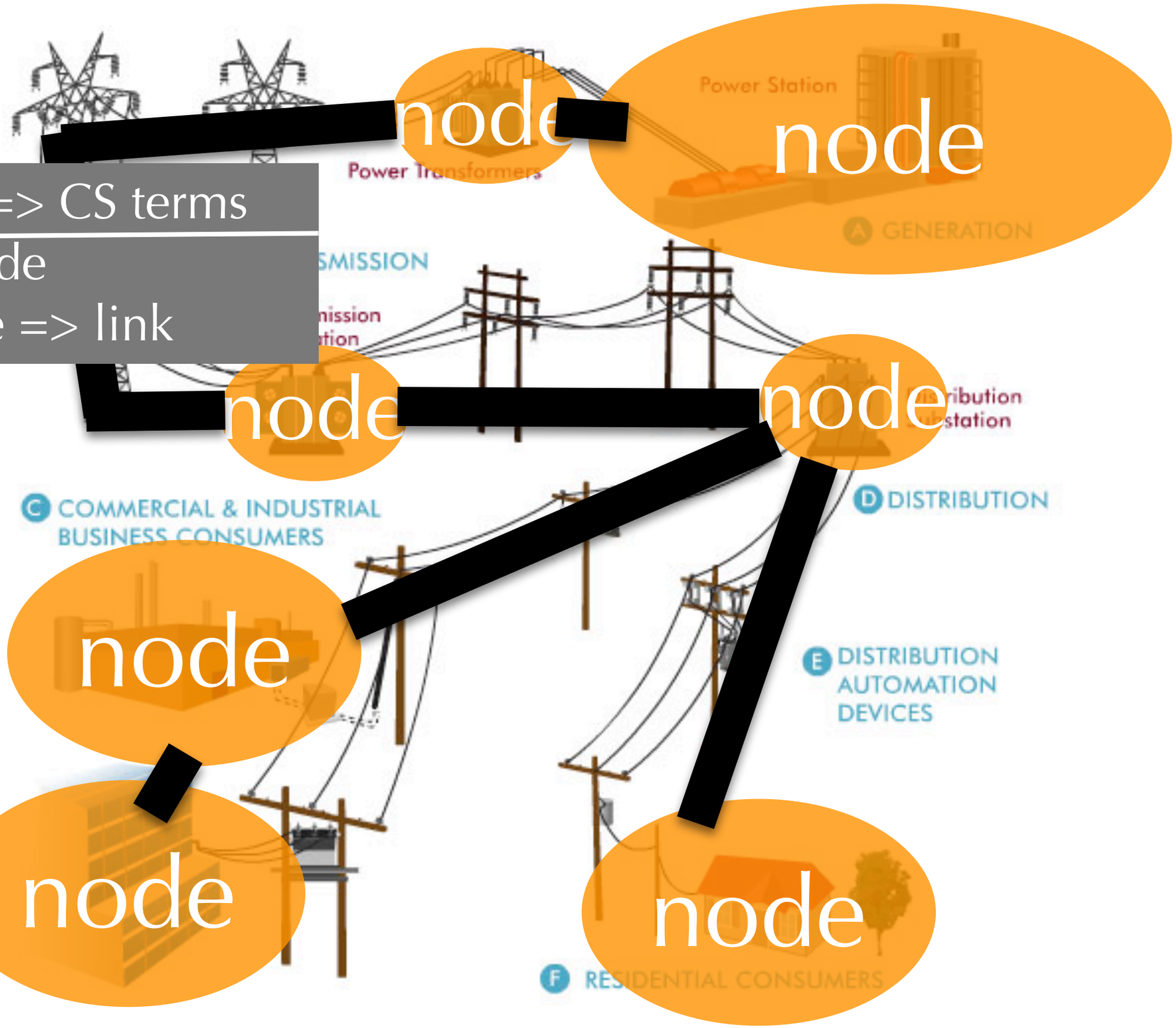
bus is either:

- power generation center,
- electric substation, or
- aggregation of loads

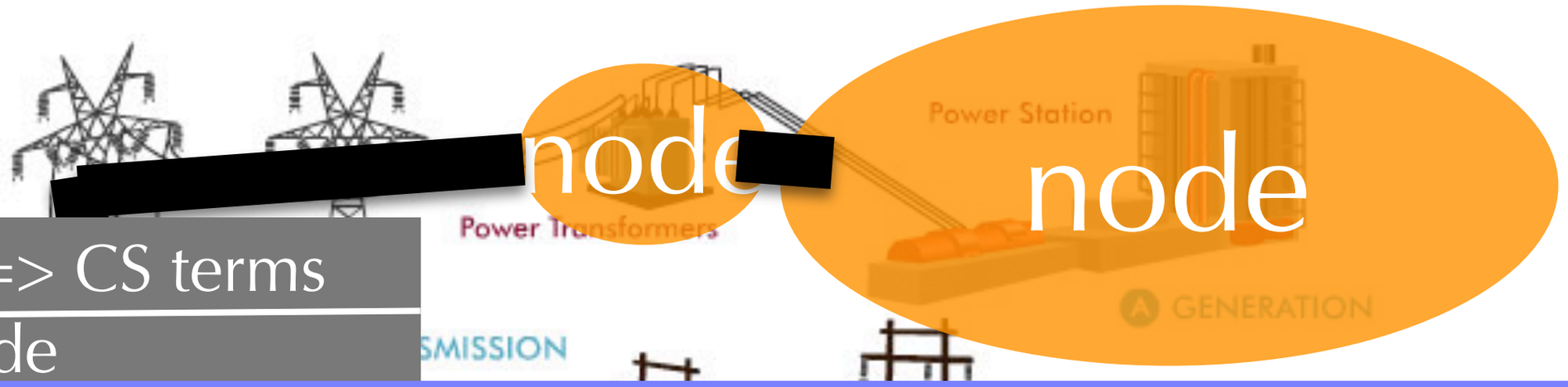
Refresher: Power Grid Definition

power terms => CS terms

- bus => node
- power line => link



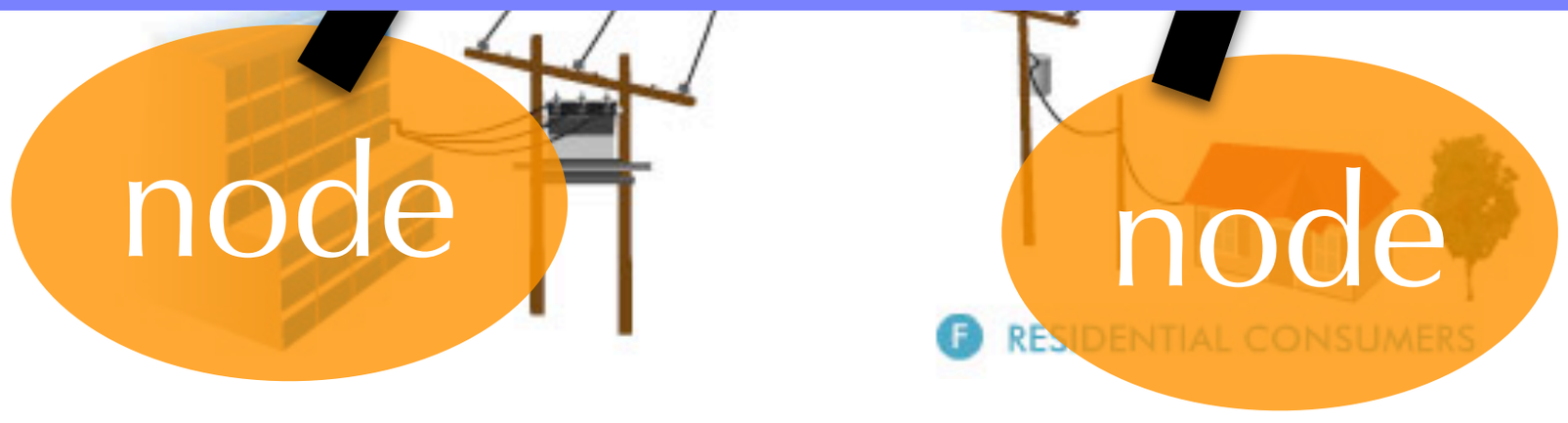
Refresher: Power Grid Definition



power terms => CS terms

- bus => node

major plan for smart grid is to deploy PMU sensors (at nodes) + wide-area communication to better mange and operate the grid



More on PMU Sensors

- deployed at nodes (buses)
- measure voltage and current of bus (node) and connected transmission lines (links)
  - high sampling rate (60 samples/sec)
  - measurements synchronized with GPS clock

More on PMU Sensors

- deployed at nodes (buses)
- measure voltage and current of bus (node) and connected transmission lines (links)
  - high sampling rate (60 samples/sec)
  - measurements synchronized with GPS clock

PMUs: instantaneous snapshot of grid of health

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

- node is *observable* if voltage can be directly measured or computed

D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

- node is *observable* if voltage can be directly measured or computed
- rule 1: if a PMU is placed at v then v and its neighbors are observed

D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

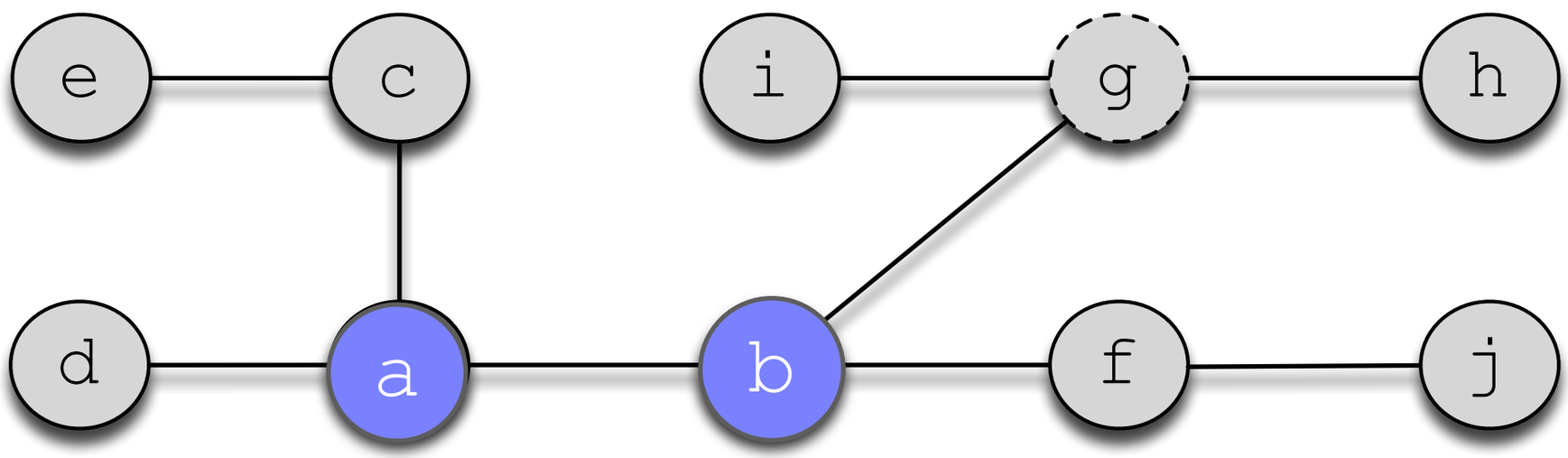
- node is *observable* if voltage can be directly measured or computed
- rule 1: if a PMU is placed at v then v and its neighbors are observed
- rule 2: if v is observed + all of v's neighbors but one are observed, then all v's neighbors are observed

D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

- node is *observable* if voltage can be directly measured or computed
- rule 1: if a PMU is placed at v then v and its neighbors are observed
- rule 2: if v is observed + all of v's neighbors but one are observed, then all v's neighbors are observed

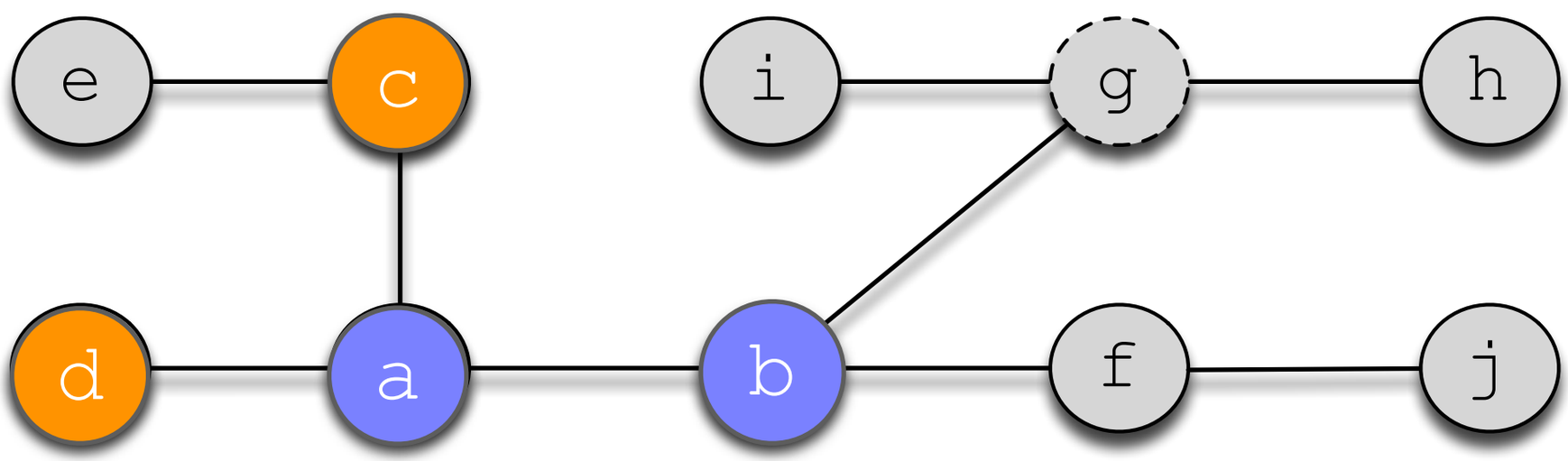


D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

- node is *observable* if voltage can be directly measured or computed
- rule 1: if a PMU is placed at v then v and its neighbors are observed
- rule 2: if v is observed + all of v's neighbors but one are observed, then all v's neighbors are observed

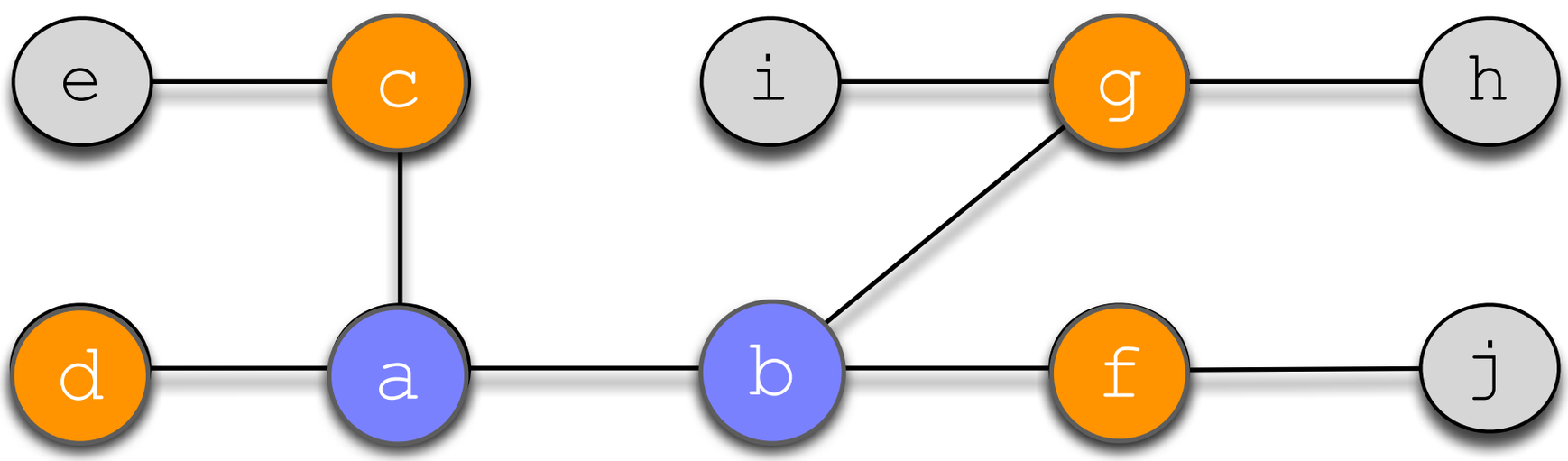


D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

- node is *observable* if voltage can be directly measured or computed
- rule 1: if a PMU is placed at v then v and its neighbors are observed
- rule 2: if v is observed + all of v's neighbors but one are observed, then all v's neighbors are observed

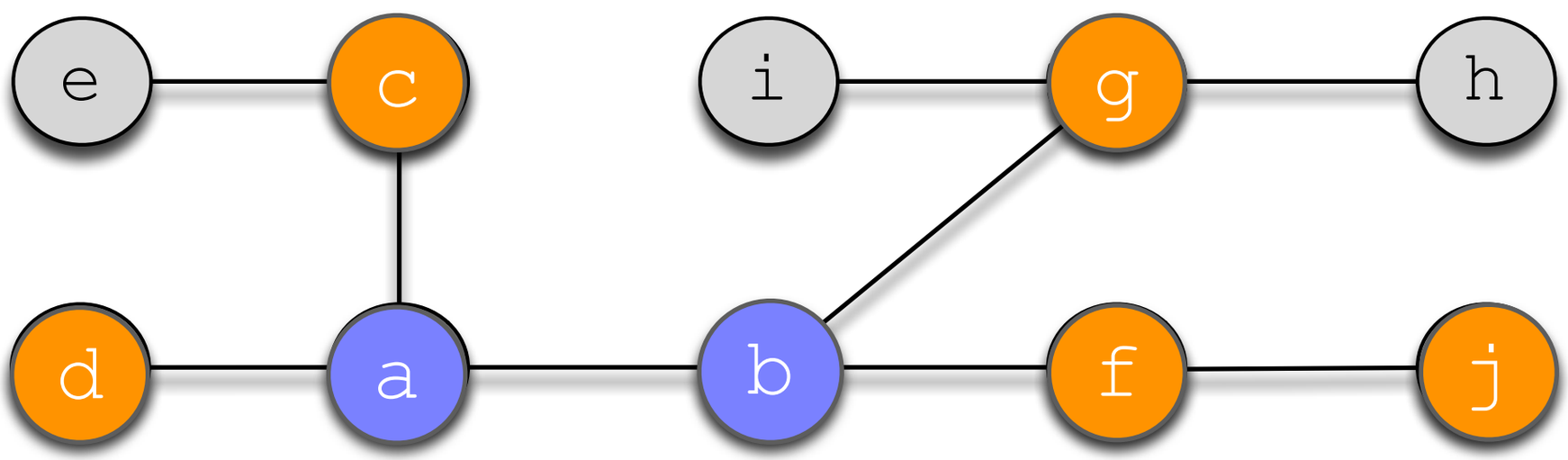


D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

- node is *observable* if voltage can be directly measured or computed
- rule 1: if a PMU is placed at v then v and its neighbors are observed
- rule 2: if v is observed + all of v's neighbors but one are observed, then all v's neighbors are observed

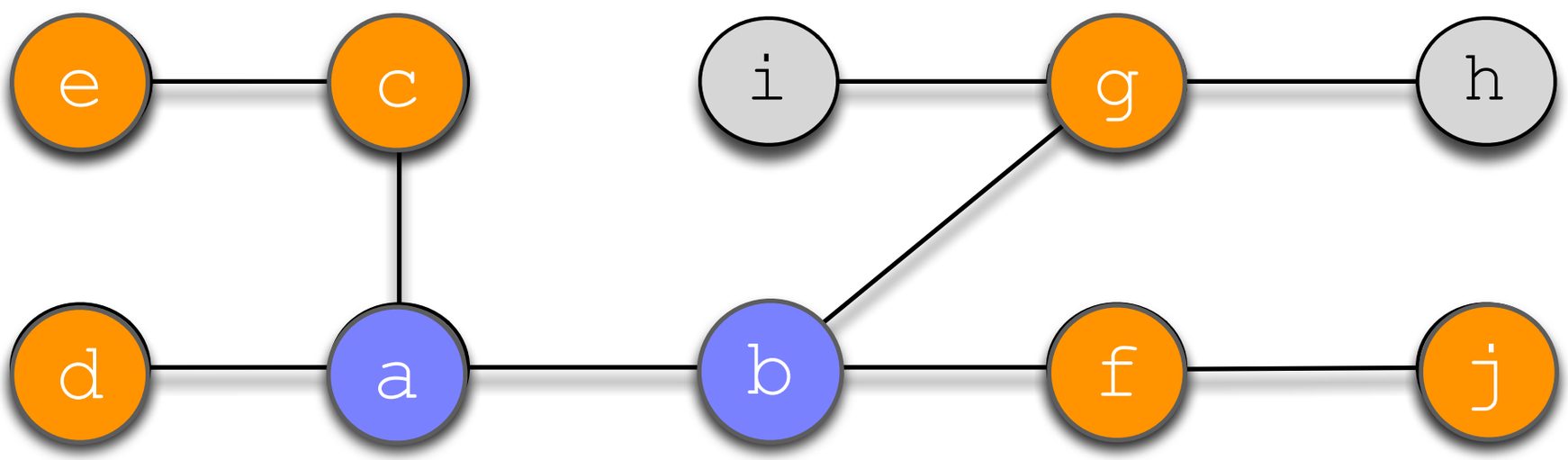


D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

Grid Measurement + Monitoring

where to place PMUs to maximize observability?

- node is *observable* if voltage can be directly measured or computed
- rule 1: if a PMU is placed at v then v and its neighbors are observed
- rule 2: if v is observed + all of v's neighbors but one are observed, then all v's neighbors are observed



D. Brueni and L. Heath. The PMU Placement Problem. SIAM Journal on Discrete Math, 2005

MAX-OBSERVE Problem

- Input: $G=(V,E)$ and k PMUs
- Output: placement of k PMUs to maximize number of observed nodes

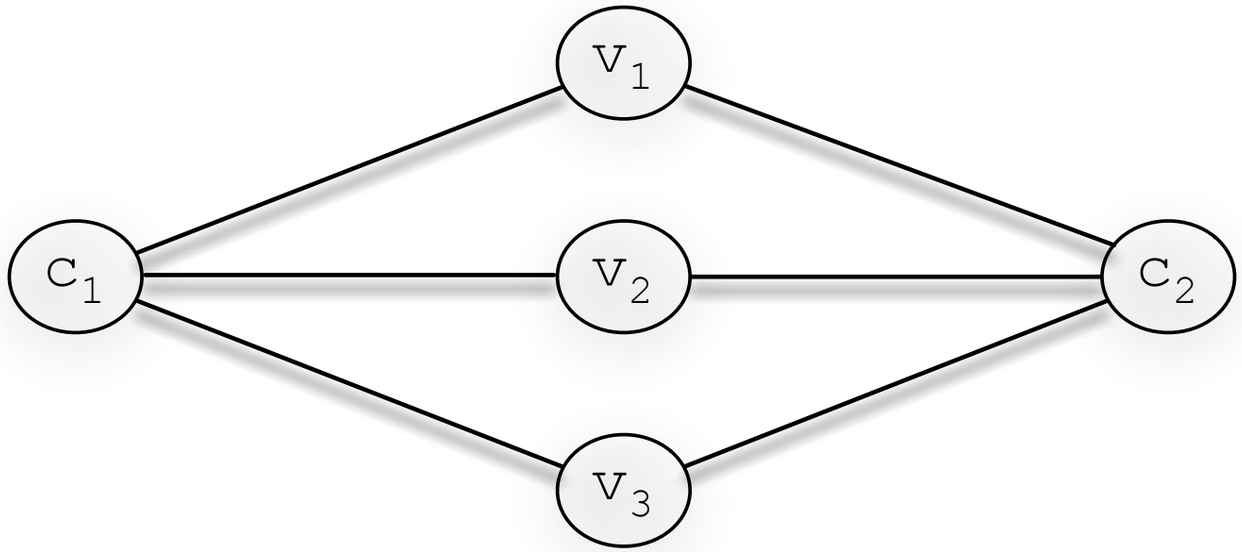
MAX-OBSERVE Problem

- Input: $G=(V,E)$ and k PMUs
- Output: placement of k PMUs to maximize number of observed nodes

MAX-OBSERVE is NP-Complete, reduce from planar 3SAT

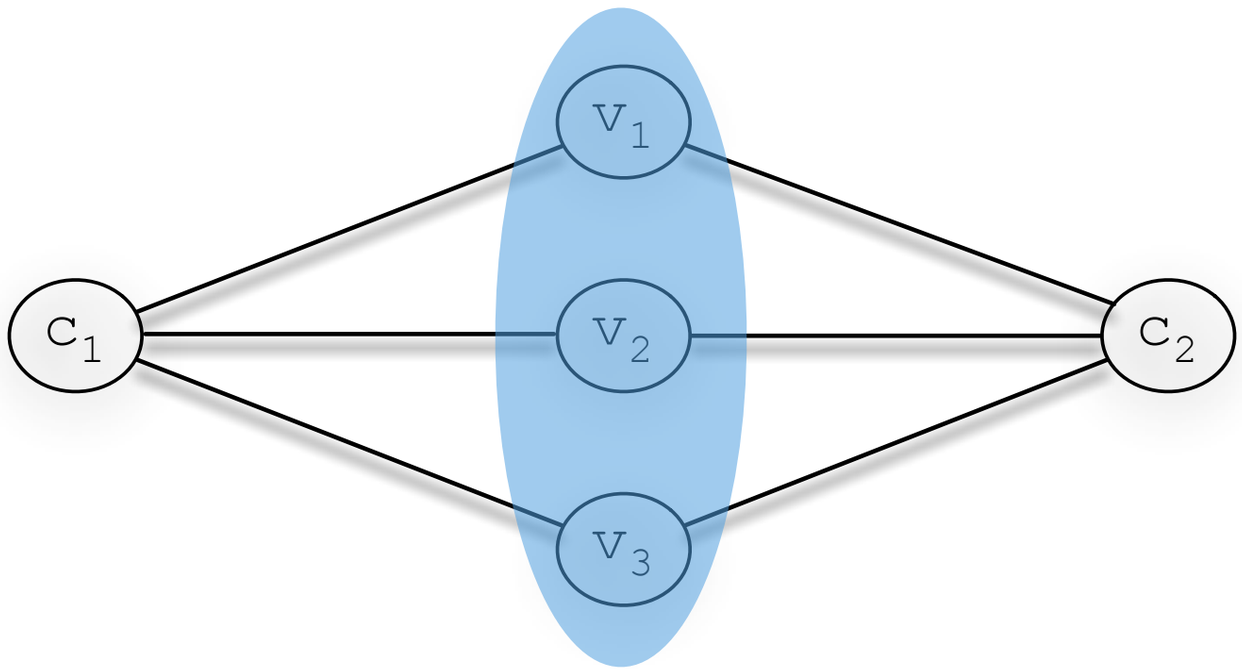
Proof: MaxObserve is NPC

$$\phi = (\overline{v_1} \vee v_2 \vee v_3) \wedge (v_1 \vee \overline{v_2} \vee \overline{v_3})$$



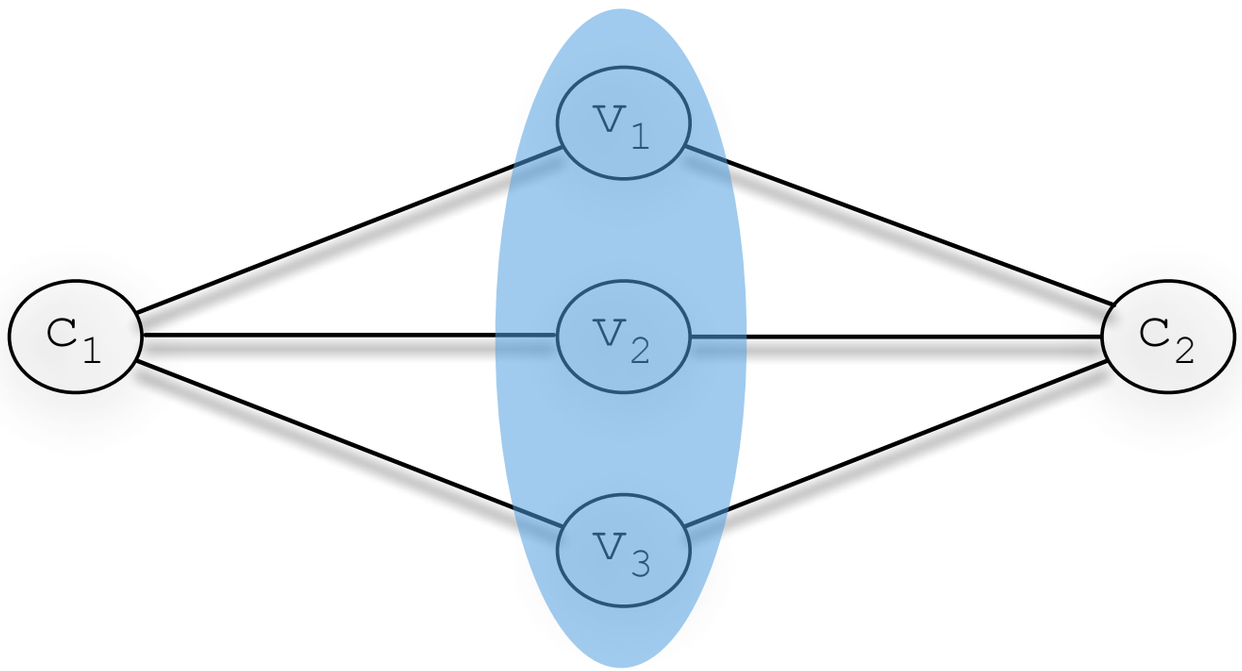
Proof: MaxObserve is NPC

$$\phi = (\overline{v_1} \vee v_2 \vee v_3) \wedge (v_1 \vee \overline{v_2} \vee \overline{v_3})$$

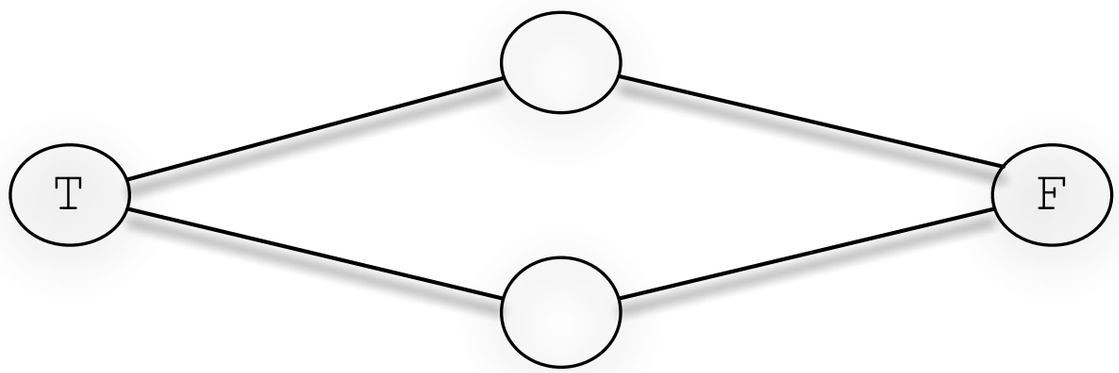


Proof: MaxObserve is NPC

$$\phi = (\overline{v_1} \vee v_2 \vee v_3) \wedge (v_1 \vee \overline{v_2} \vee \overline{v_3})$$



replace each variable with set of nodes below (gadget)



MAX-OBSERVE Related Work

- we adapt proof technique from Brueni and Heath [13]
- previous work makes unrealistic assumptions Brueni and Heath [13], Aazami and Stilp, [4]
  - assume all nodes are zero-injection
    - in practice: ~5% of nodes are zero-injection

PMU Placement w/ Cross-Validation

where to place PMUs to maximize observability with requirement that all PMUs are cross-validated?

Vanfretti et al. A Phasor-data-based State Estimator Incorporating Phase Bias Correction. IEEE Transactions on Power Systems, 2010

PMU Placement w/ Cross-Validation

where to place PMUs to maximize observability with requirement that all PMUs are cross-validated?

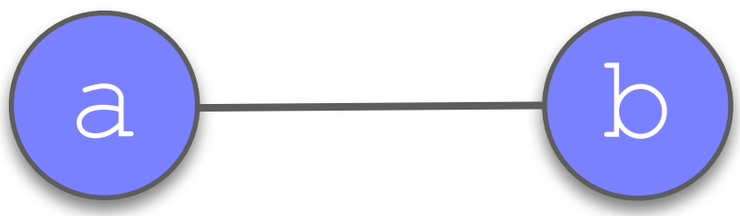
- cross-validation provides measurement error detection

Vanfretti et al. A Phasor-data-based State Estimator Incorporating Phase Bias Correction. IEEE Transactions on Power Systems, 2010

PMU Placement w/ Cross-Validation

where to place PMUs to maximize observability with requirement that all PMUs are cross-validated?

- cross-validation provides measurement error detection
- cross-validation rule 1: if PMUs are placed at adjacent nodes, they cross-validate each other

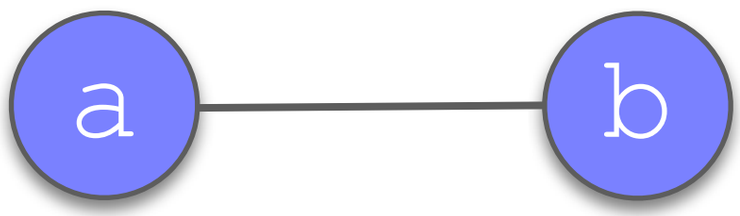


Vanfretti et al. A Phasor-data-based State Estimator Incorporating Phase Bias Correction. IEEE Transactions on Power Systems, 2010

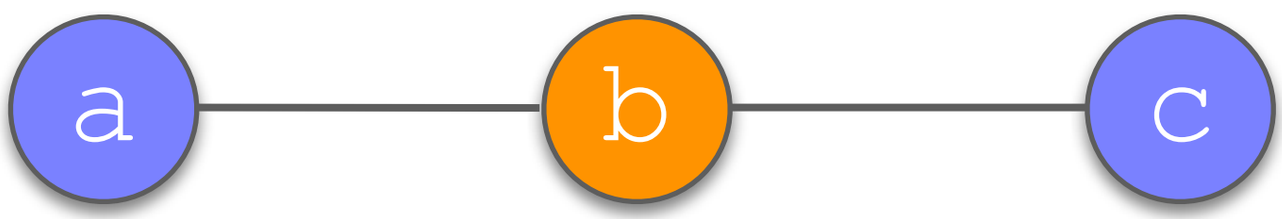
PMU Placement w/ Cross-Validation

where to place PMUs to maximize observability with requirement that all PMUs are cross-validated?

- cross-validation provides measurement error detection
- cross-validation rule 1: if PMUs are placed at adjacent nodes, they cross-validate each other



- cross-validation rule 2: if two PMUs share a common neighbor, they cross-validate each other



Vanfretti et al. A Phasor-data-based State Estimator Incorporating Phase Bias Correction. IEEE Transactions on Power Systems, 2010

MAX-OBSERVE-XV Problem

- Input: $G=(V,E)$ and k PMUs
- Output: placement of k PMUs to maximize number of observed nodes, requiring that all PMUs are cross-validated

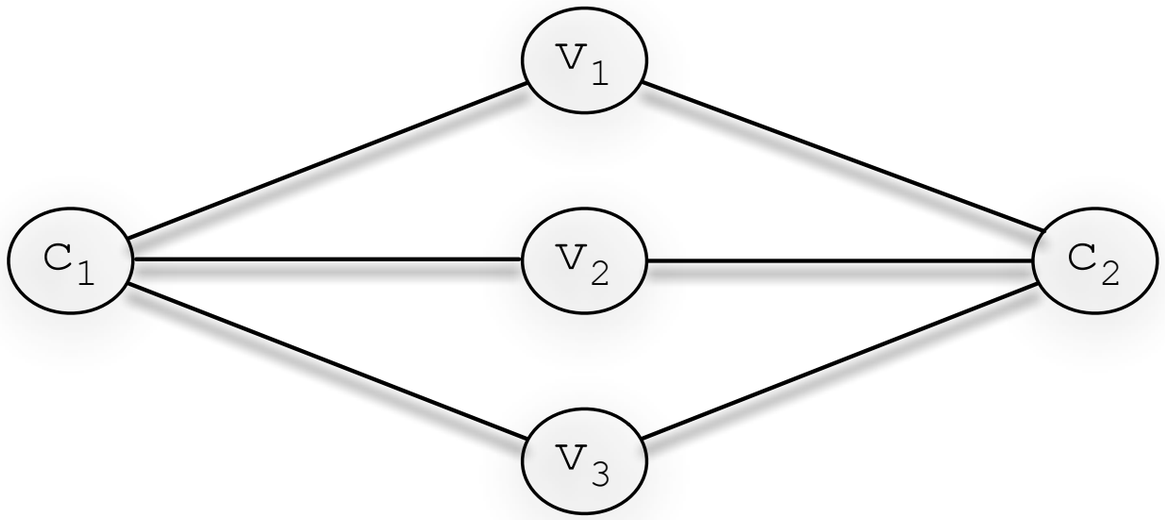
MAX-OBSERVE-XV Problem

- Input: $G=(V,E)$ and k PMUs
- Output: placement of k PMUs to maximize number of observed nodes, requiring that all PMUs are cross-validated

MAX-OBSERVE-XV is NP-Complete, reduce from planar 3SAT

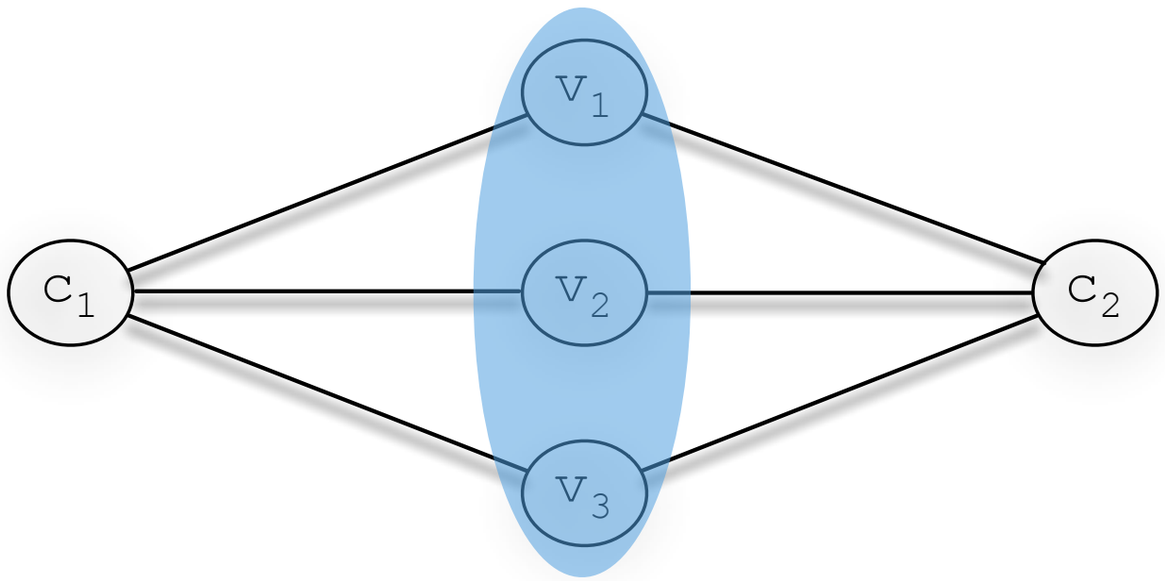
Proof: MaxObserve-XV is NPC

$$\phi = (\overline{v_1} \vee v_2 \vee v_3) \wedge (v_1 \vee \overline{v_2} \vee \overline{v_3})$$



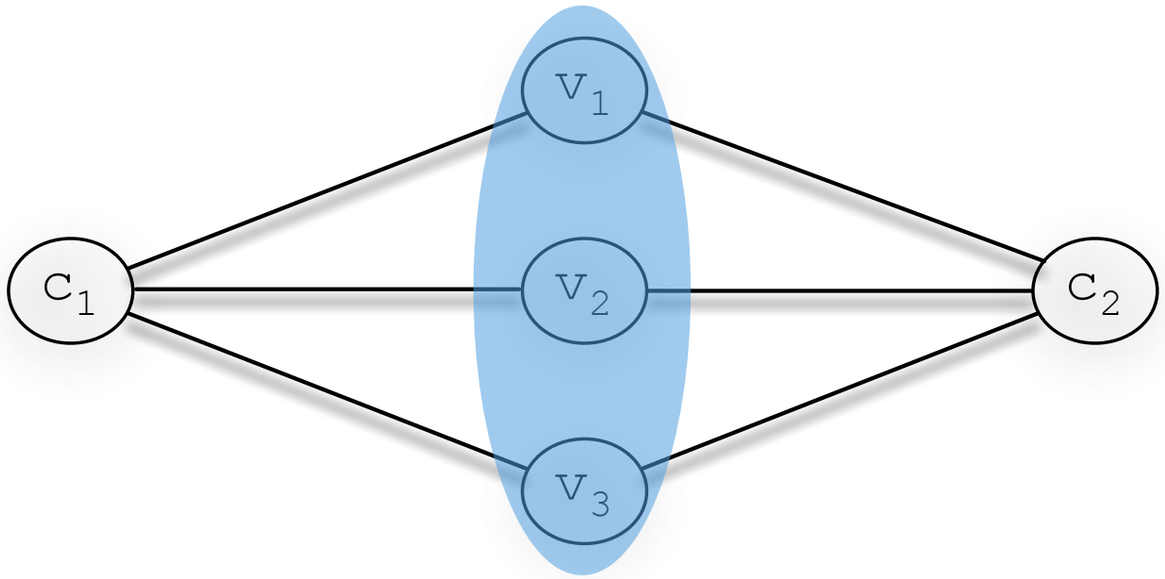
Proof: MaxObserve-XV is NPC

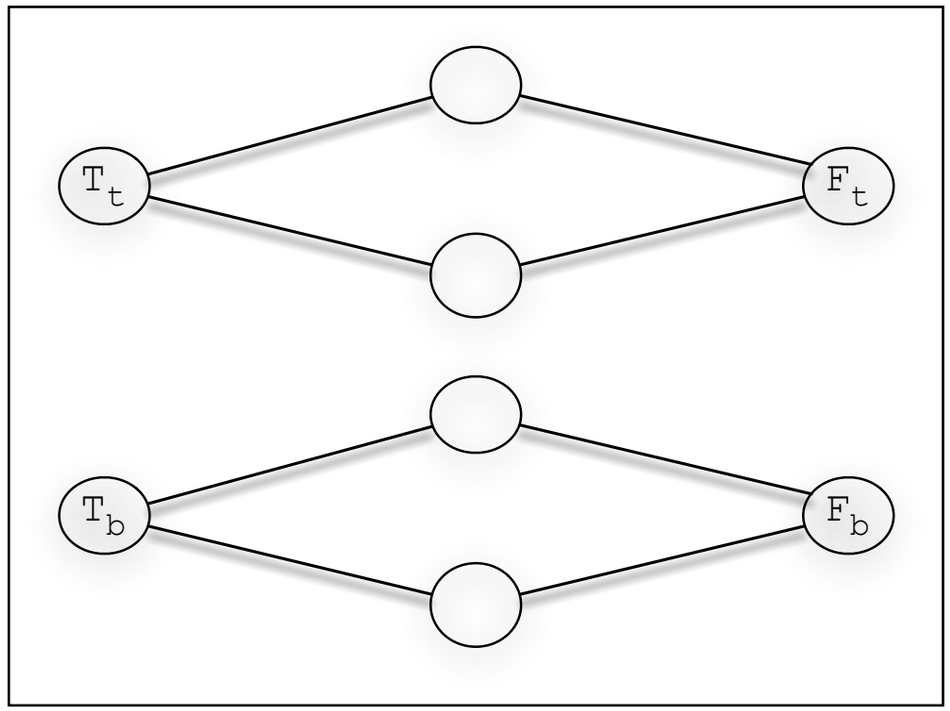
$$\phi = (\overline{v_1} \vee v_2 \vee v_3) \wedge (v_1 \vee \overline{v_2} \vee \overline{v_3})$$



Proof: MaxObserve-XV is NPC

$$\phi = (\overline{v_1} \vee v_2 \vee v_3) \wedge (v_1 \vee \overline{v_2} \vee \overline{v_3})$$





replace each variable with set of nodes (gadget)

MAX-OBSERVE-XV Related Work

- MAX-OBSERVE-XV is completely new problem
  - related problem: Chen and Abur [46] add PMUs so PMU loss can be tolerated

Greedy Approximations

Greedy Approximations

GREEDY:

- iteratively add PMU to node that results in the observation of max # of nodes

Greedy Approximations

GREEDY:

- iteratively add PMU to node that results in the observation of max # of nodes

GREEDY-XV:

- iteratively place PMU pairs at nodes `{u,v}` such that `u` and `v` are cross-validated and results in the observation of max # of nodes

Simulations

- study 2 questions:

Simulations

- study 2 questions:
  - how do greedy approximations compare with brute-force-optimal results?

Simulations

- study 2 questions:
  - how do greedy approximations compare with brute-force-optimal results?
  - what is the (observability) overhead of cross-validation?

Simulation Setup

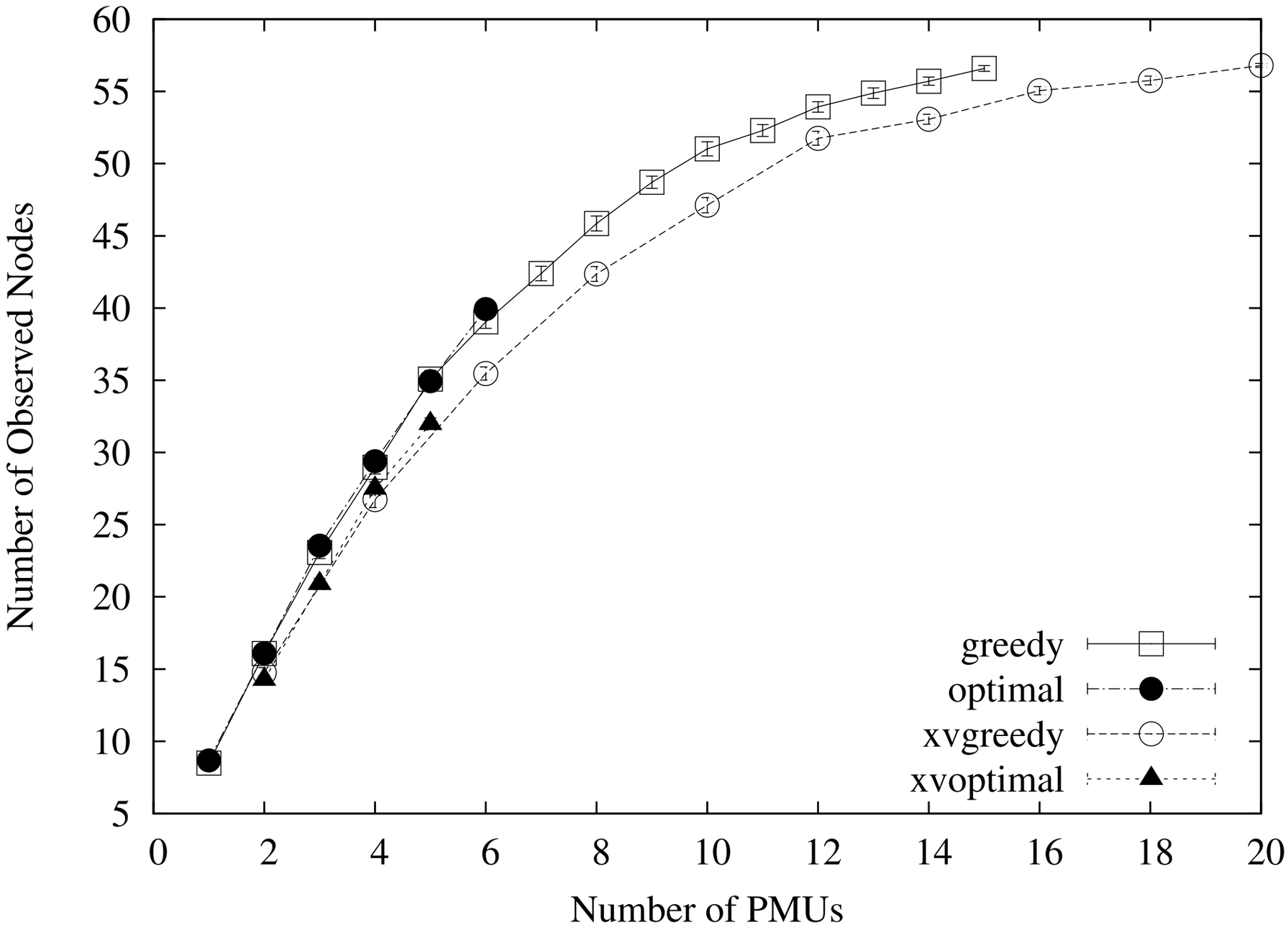
Simulation Setup

- generate grid networks with same degree distribution as IEEE bus systems
  - IEEE bus networks: standardized set of graphs used in power systems literature
  - show results for synthetic topologies based on IEEE Bus 57

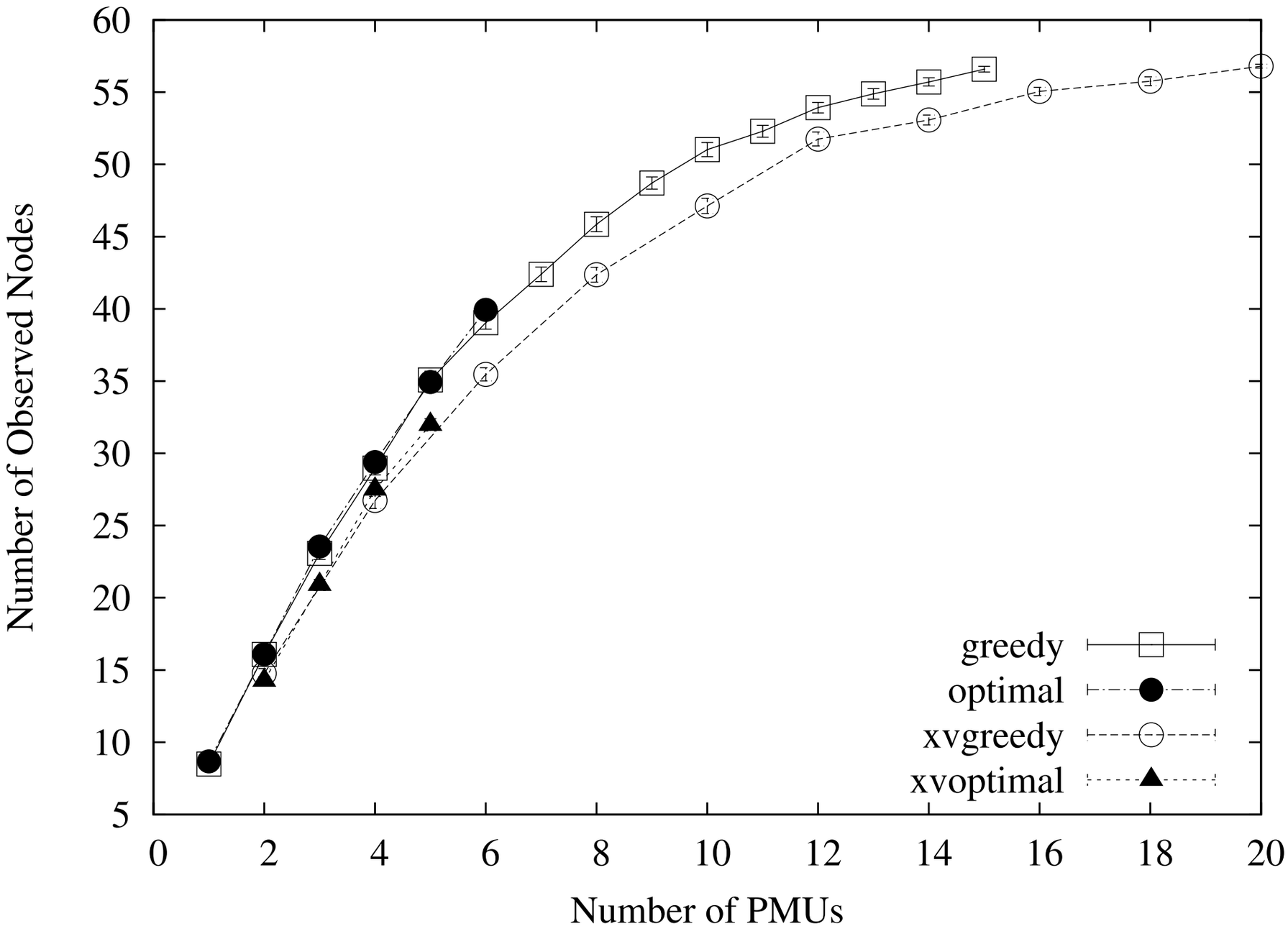
Simulation Setup

- generate grid networks with same degree distribution as IEEE bus systems
  - IEEE bus networks: standardized set of graphs used in power systems literature
  - show results for synthetic topologies based on IEEE Bus 57
- compare with brute-force optimal solution by enumeration with small # of PMUs

Simulation Results

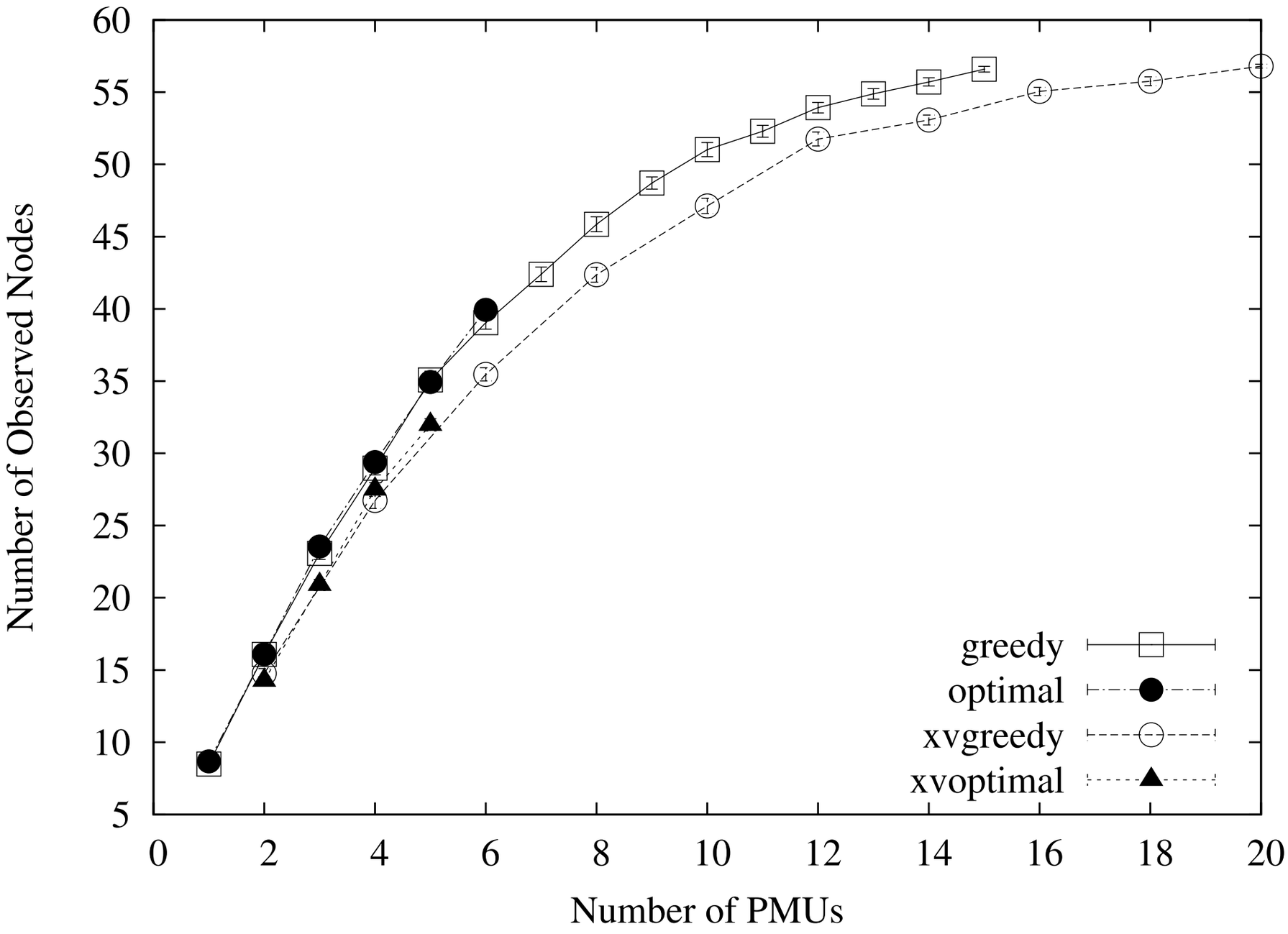


Simulation Results



result 1: greedy solutions within 97% of optimal

Simulation Results



result 1: greedy solutions within 97% of optimal

result 2: cross-validation decreases # observed by ~ 5%

Talk Outline

- thesis introduction
- placement of smart grid sensors to enable measurement error detection
- recovery from failed communication links in a smart grid
- recovery from malicious nodes injecting false routing state
- outline for future work and conclusions

Power Grid Operation

- currently use outdated SCADA system (1960s)
  - course-grained measurements
  - data only locally distributed

Power Grid Operation

- currently use outdated SCADA system (1960s)
  - course-grained measurements
  - data only locally distributed

current power grid operations inefficient b/c large safety buffers required due to lack of measurements + poor data dissemination

Smart Grid Operation

- better sensing
  - more modern sensors: PMUs, IEDs, ...
  - fine-grained measurement
- disseminate data widely
  - utility companies, balancing authorities

Smart Grid Operation

- better sensing
  - more modern sensors: PMUs, IEDs, ...
  - fine-grained measurement
- disseminate data widely
  - utility companies, balancing authorities

better sensing + wide-area data dissemination will enable more efficient ways to operate and manage the grid

Smart Grid Data Dissemination

- richer communication paradigms
  - QoS guarantees (bandwidth, delay, rate)
  - multicast delivery (1 to many)
    - high reliability protocols
      - self-healing multicast

Smart Grid Data Dissemination

- richer communication paradigms
  - QoS guarantees (bandwidth, delay, rate)
  - multicast delivery (1 to many)
  - high reliability protocols
    - self-healing multicast

our focus

Smart Grid Data Dissemination

- richer communication paradigms
  - QoS guarantees (bandwidth, delay, rate)
  - multicast delivery (1 to many)
  - high reliability protocols
    - self-healing multicast

our focus

focus on recovery from comm. link failures in smart grid
=> precompute backup MTs offline in case link fails

Smart Grid App QoS Requirements

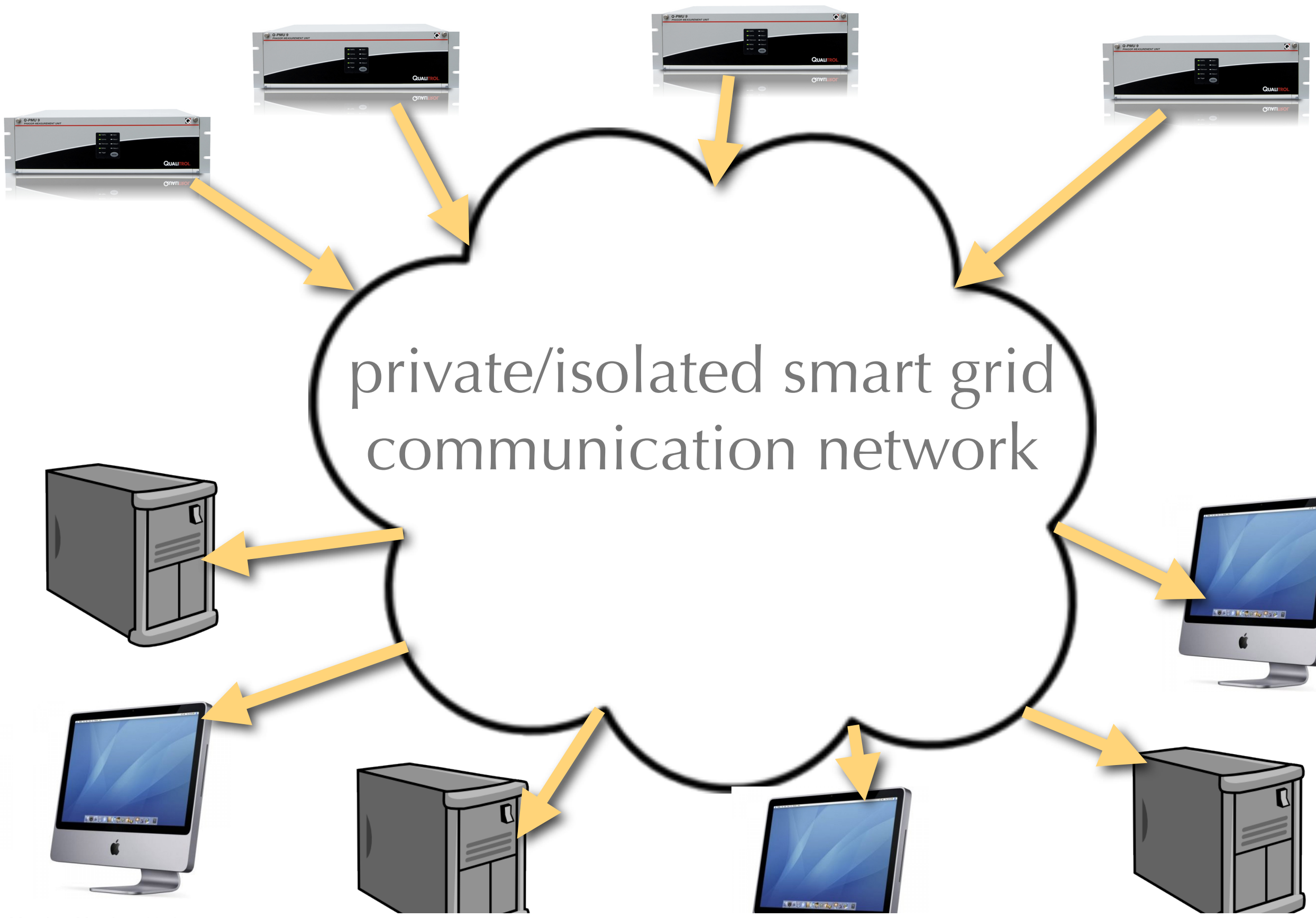
- critical power grid applications have stringent and challenging QoS requirements [7]
  - low latency
    - 8-16ms per packet delay for real-time control applications [7]
  - low tolerance for packet loss

Smart Grid App QoS Requirements

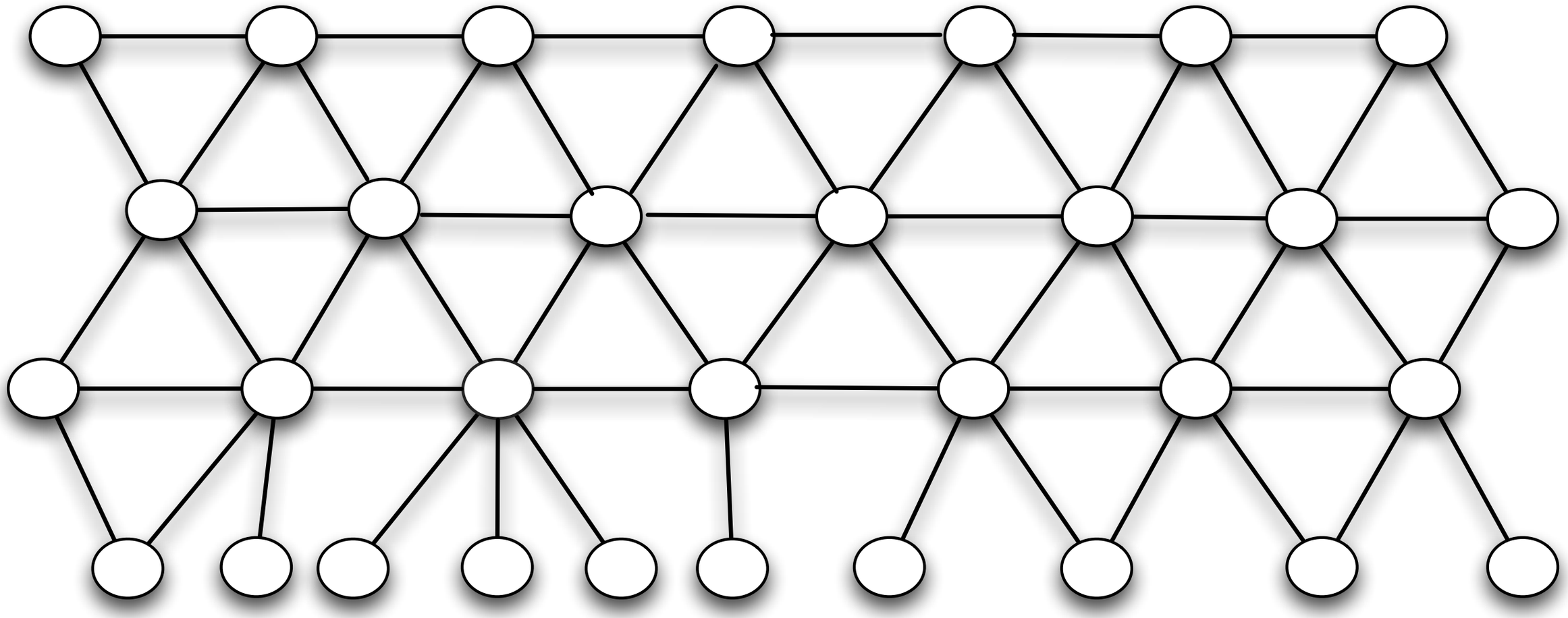
- critical power grid applications have stringent and challenging QoS requirements [7]
  - low latency
    - 8-16ms per packet delay for real-time control applications [7]
  - low tolerance for packet loss

hard E2E delivery guarantees are needed!

Smart Grid Communication Network

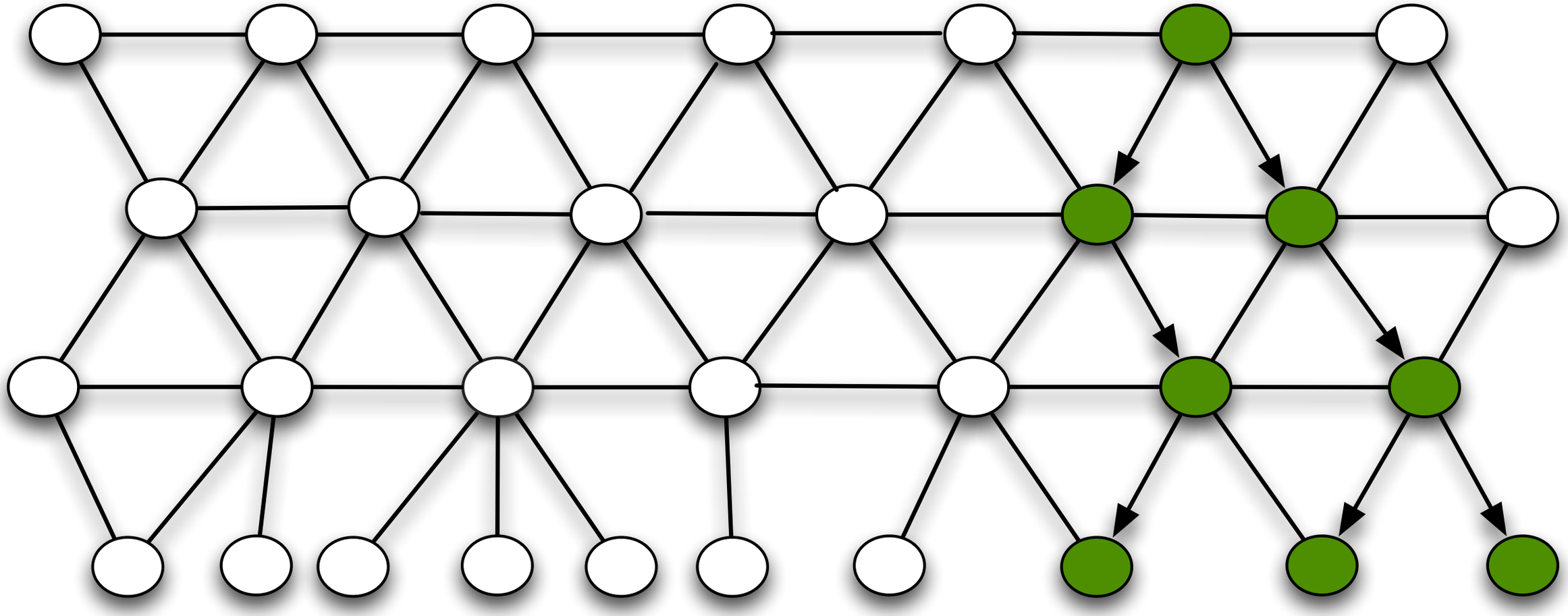


Problem Description



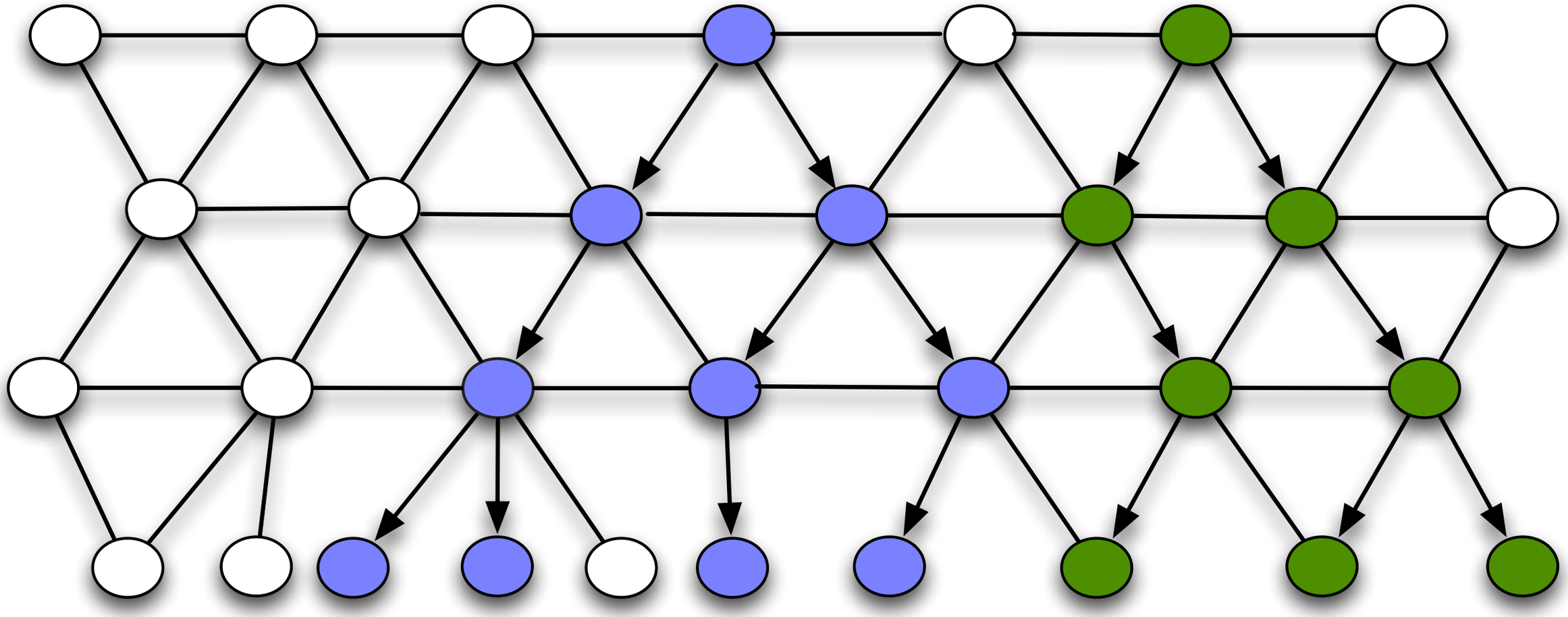
- multiple source-based multicast trees (MT)

Problem Description



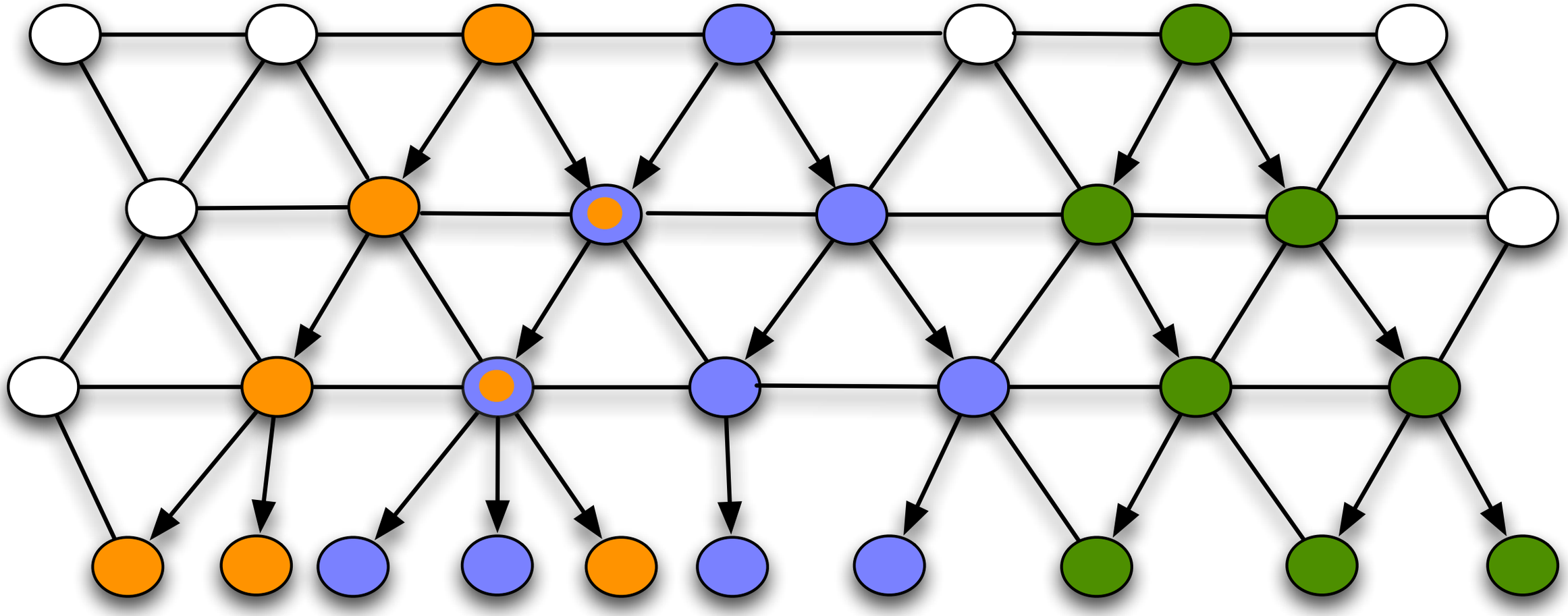
- multiple source-based multicast trees (MT)

Problem Description



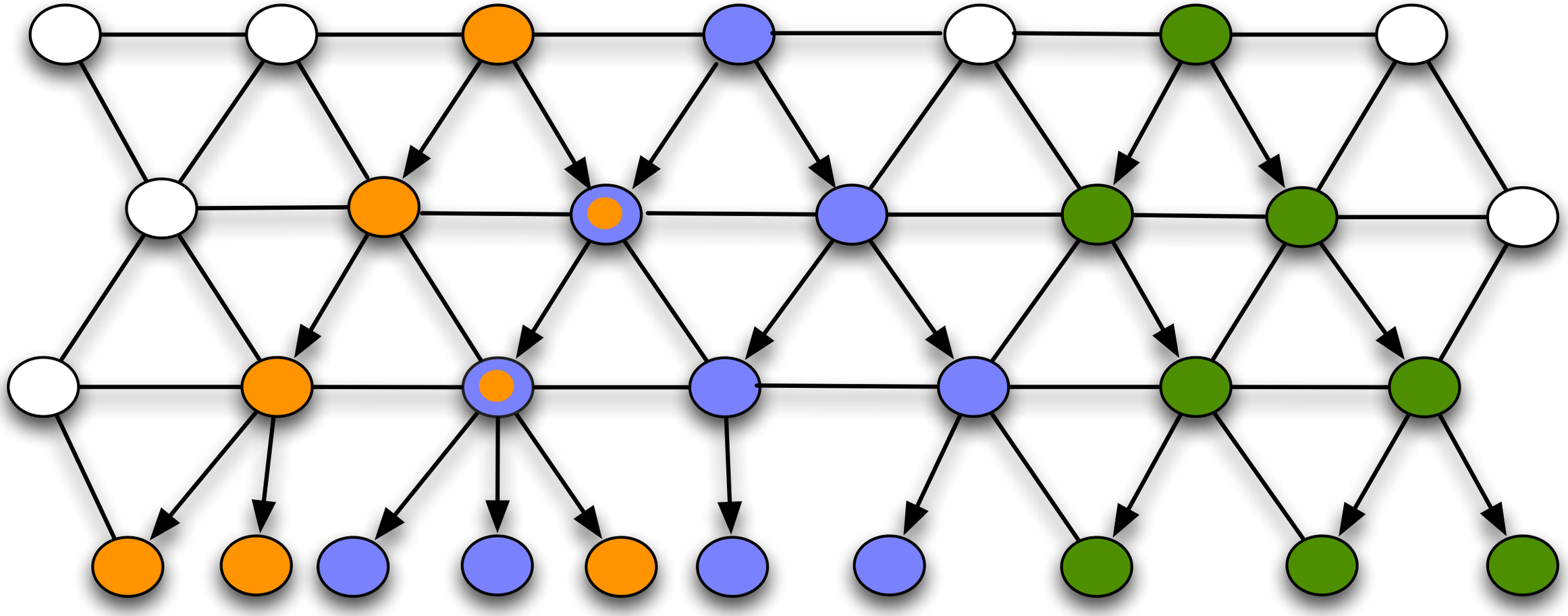
- multiple source-based multicast trees (MT)

Problem Description



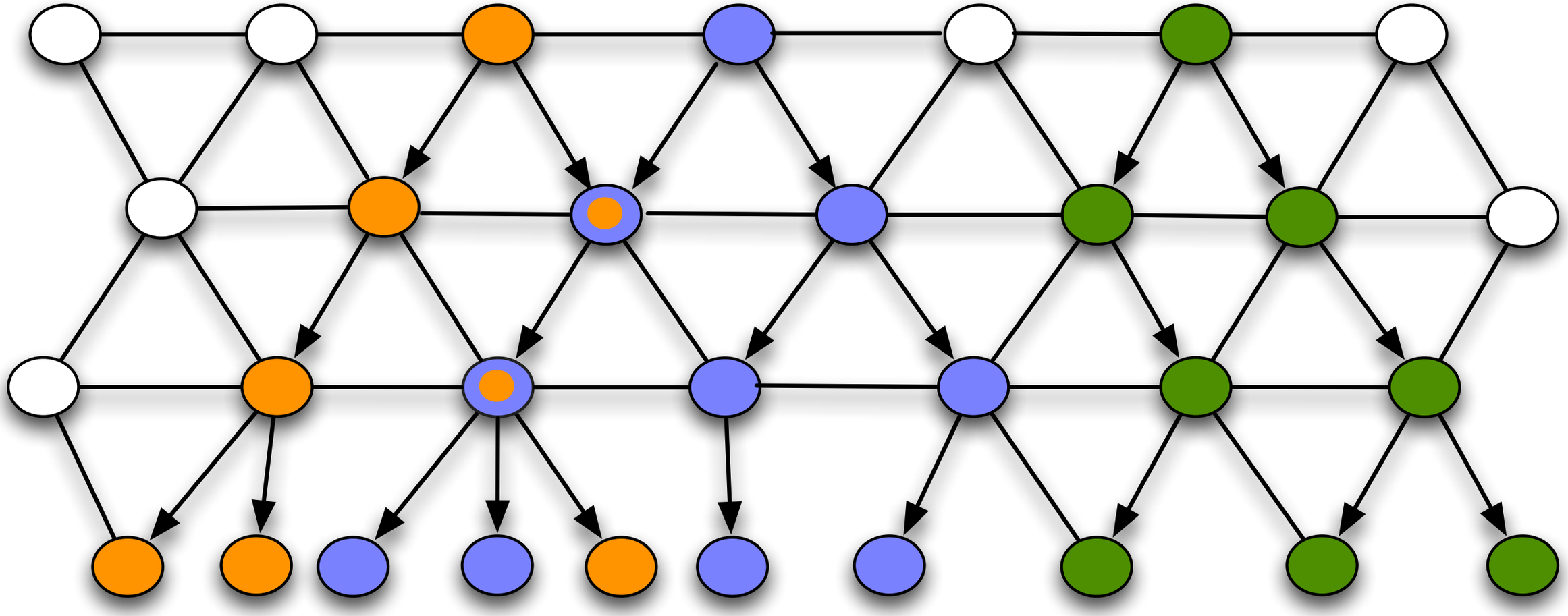
- multiple source-based multicast trees (MT)

Problem Description



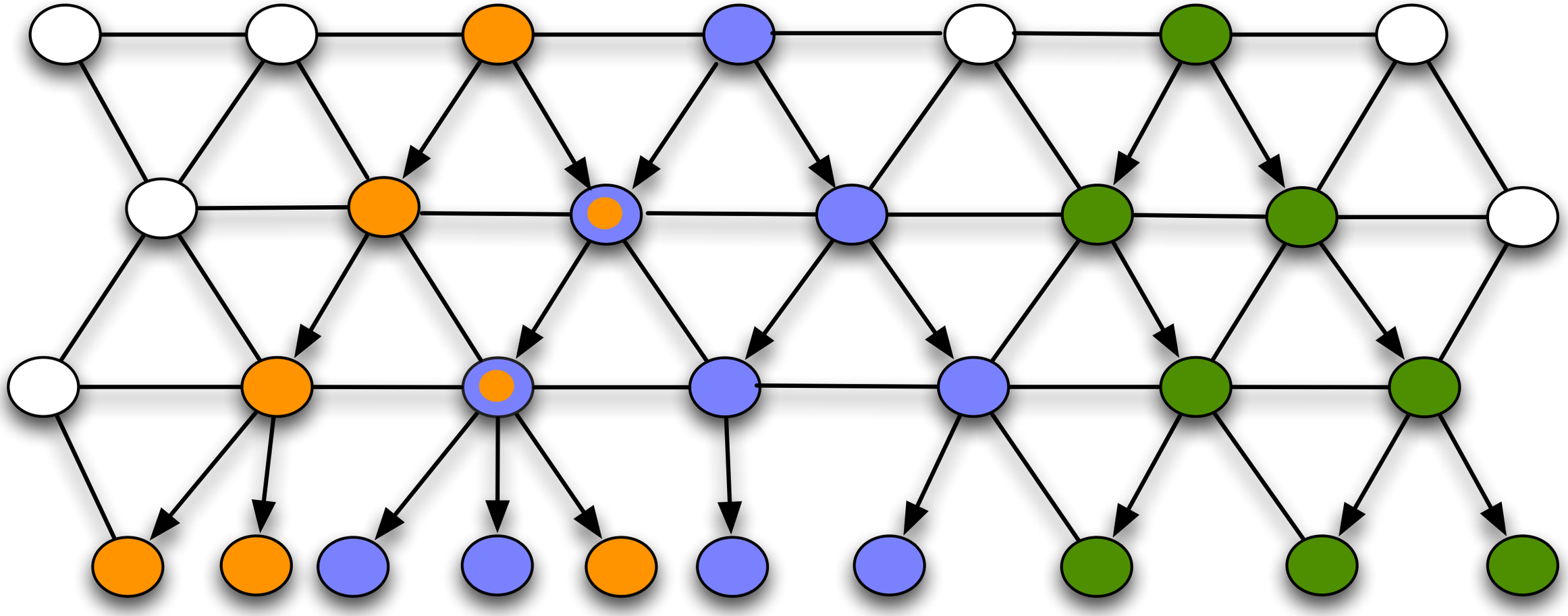
- multiple source-based multicast trees (MT)
  - rate based traffic from sender (e.g., a PMU)

Problem Description



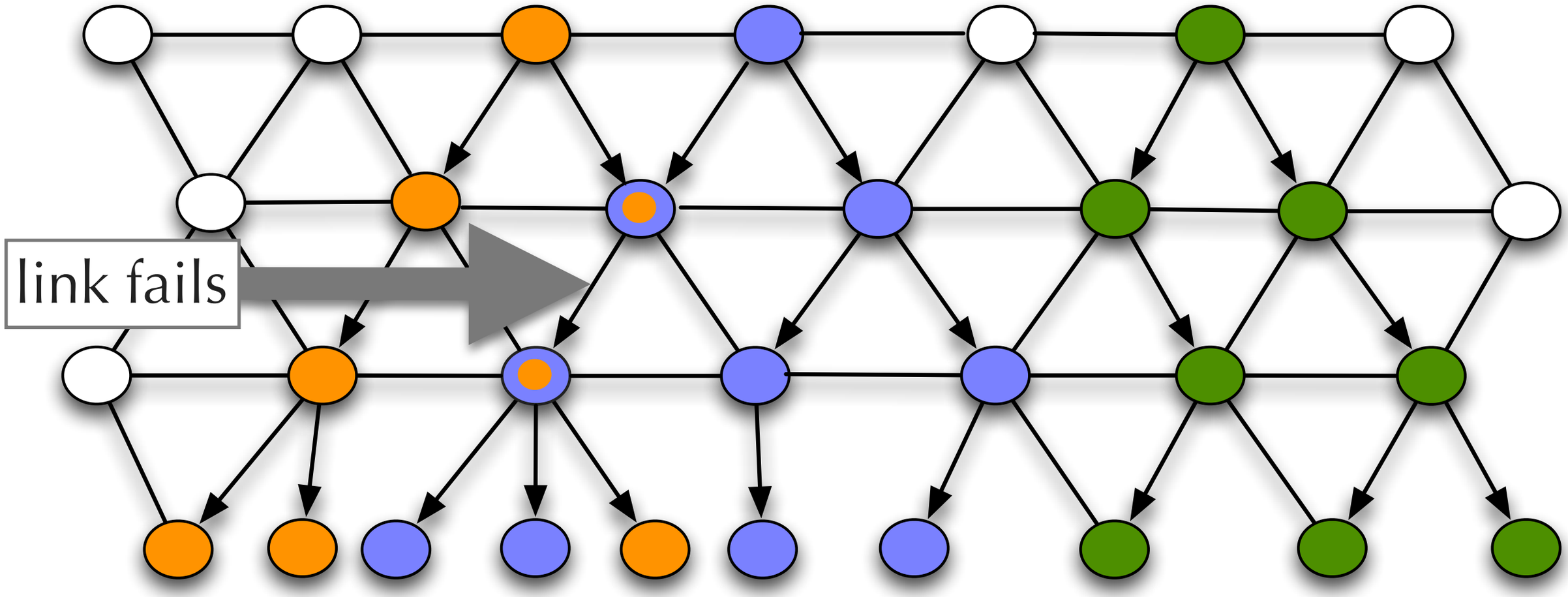
- multiple source-based multicast trees (MT)
  - rate based traffic from sender (e.g., a PMU)
  - each (source,sink) pair has E2E per packet delay req.

Problem Description



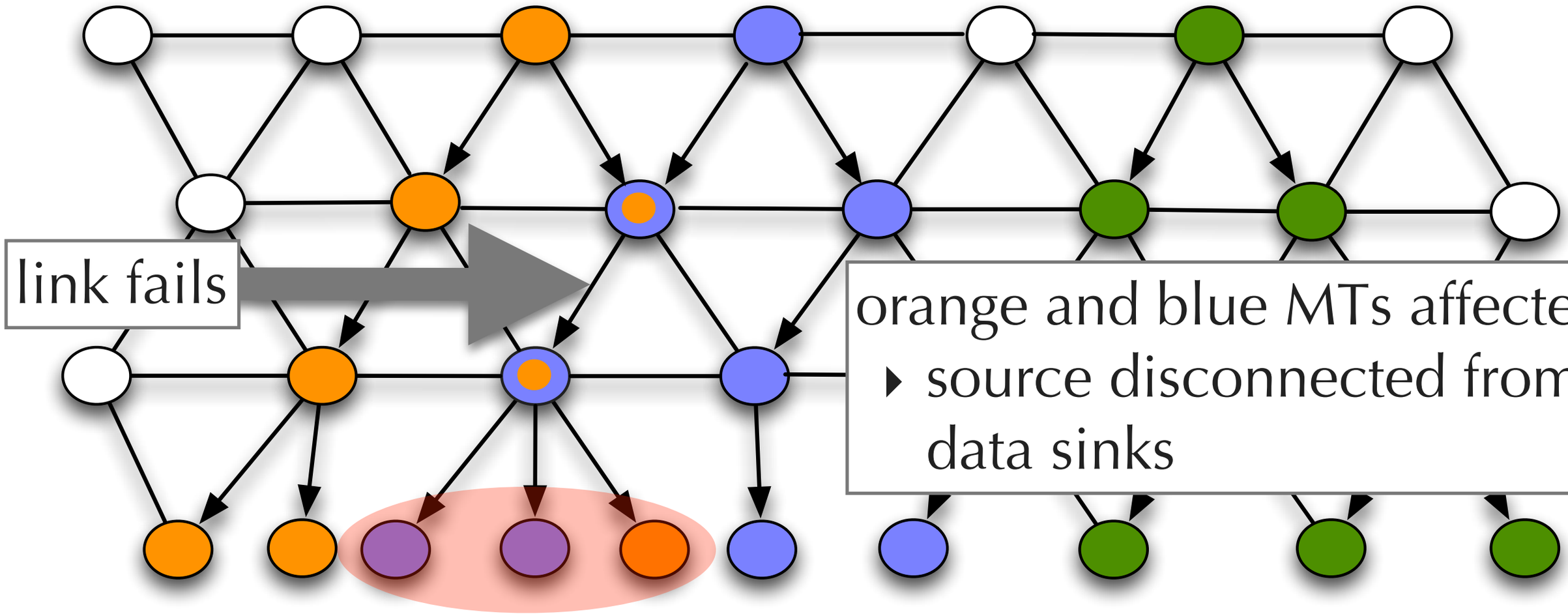
- multiple source-based multicast trees (MT)
  - rate based traffic from sender (e.g., a PMU)
  - each (source,sink) pair has E2E per packet delay req.

Problem Description



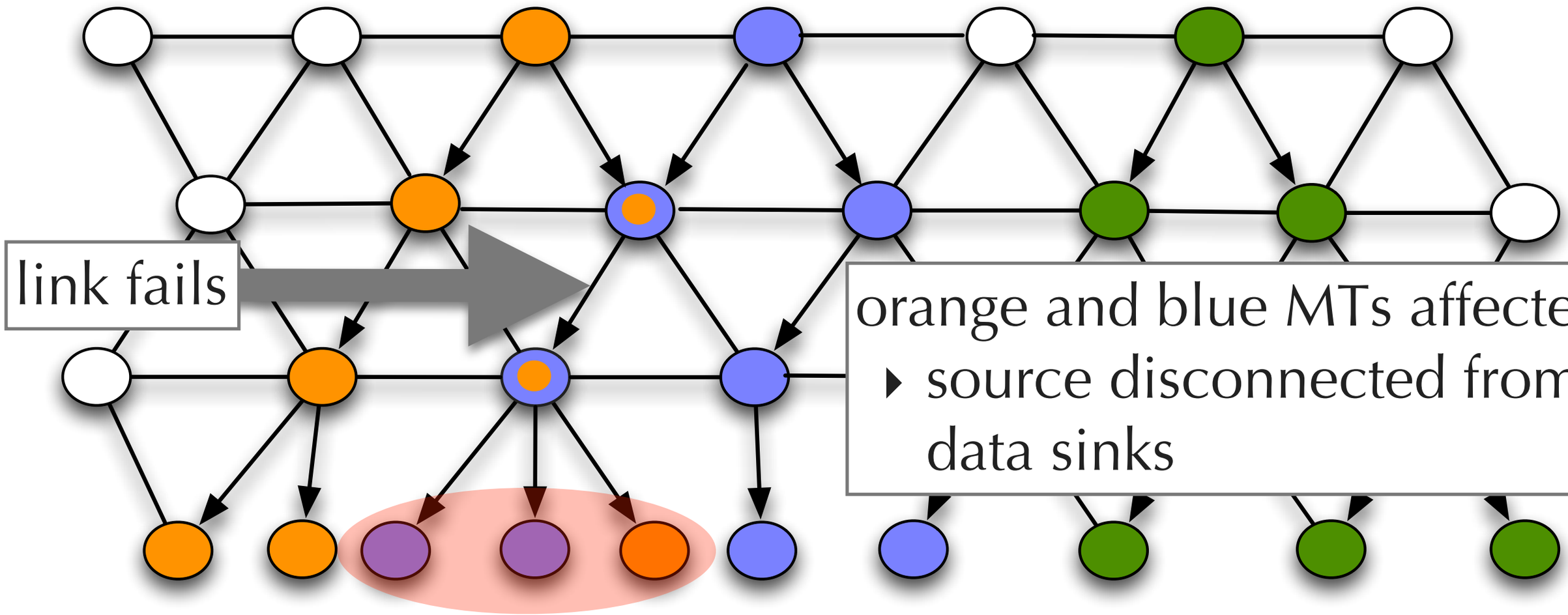
- multiple source-based multicast trees (MT)
  - rate based traffic from sender (e.g., a PMU)
  - each (source,sink) pair has E2E per packet delay req.

Problem Description



- multiple source-based multicast trees (MT)
  - rate based traffic from sender (e.g., a PMU)
  - each (source,sink) pair has E2E per packet delay req.

Problem Description



how to detect MT link failure + recover such that E2E packet loss and delay is minimized?

- each (source,sink) pair has E2E per packet delay req.

2-Step Solution

- two steps to provide link failure robustness

1. fast detection of link failures
   - detect link failure inside network
2. install precomputed backup multicast trees

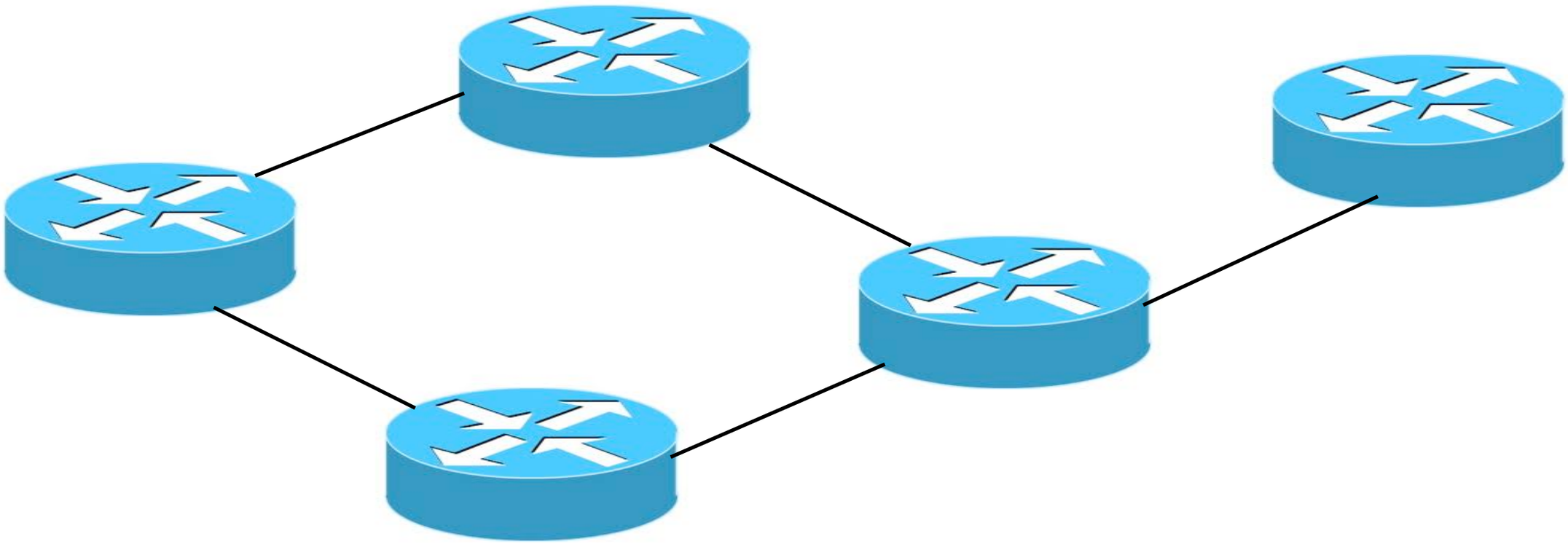
2-Step Solution

- two steps to provide link failure robustness

1. fast detection of link failures
   - detect link failure inside network
2. install precomputed backup multicast trees

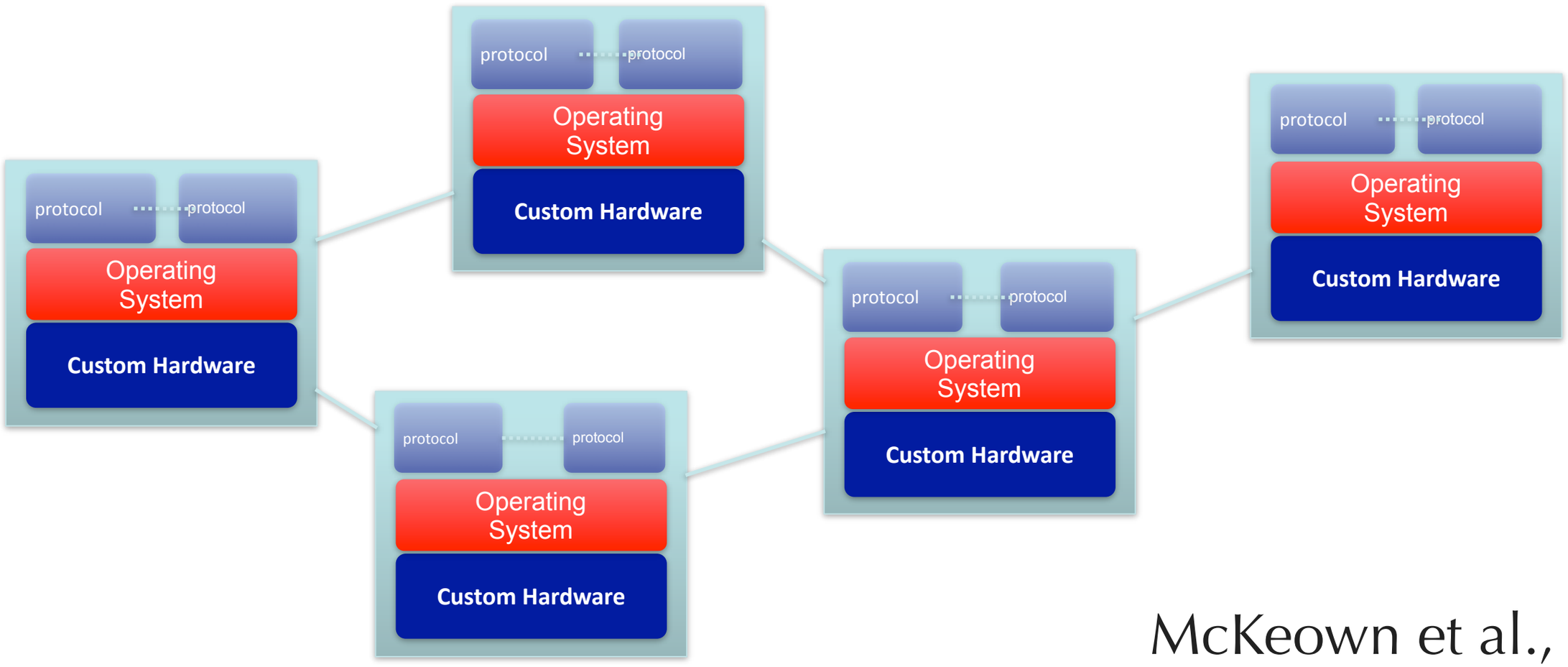
requires changes to existing routers ... how?

OpenFlow: Open Network Control Plane



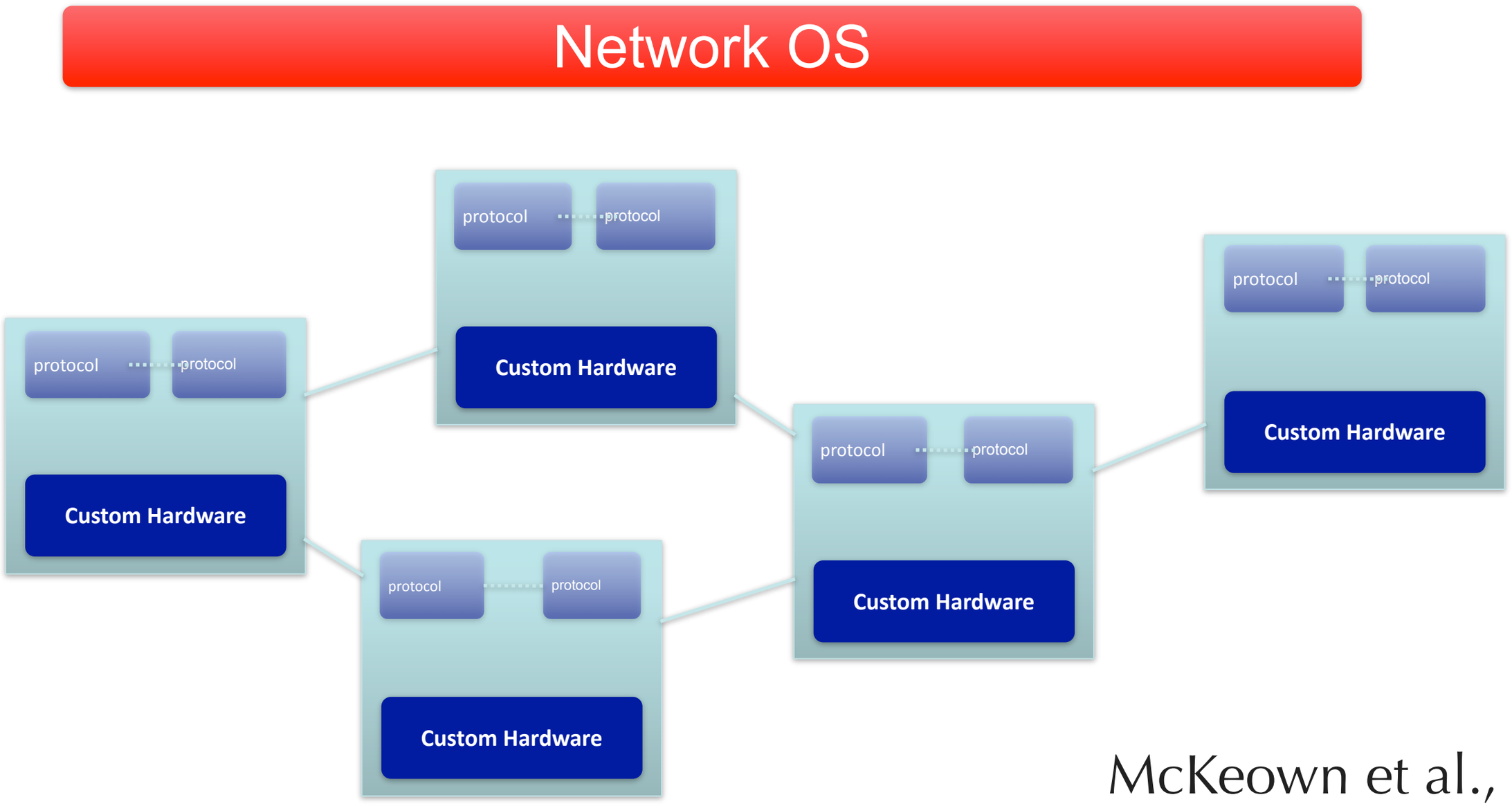
McKeown et al., [36]

OpenFlow: Open Network Control Plane



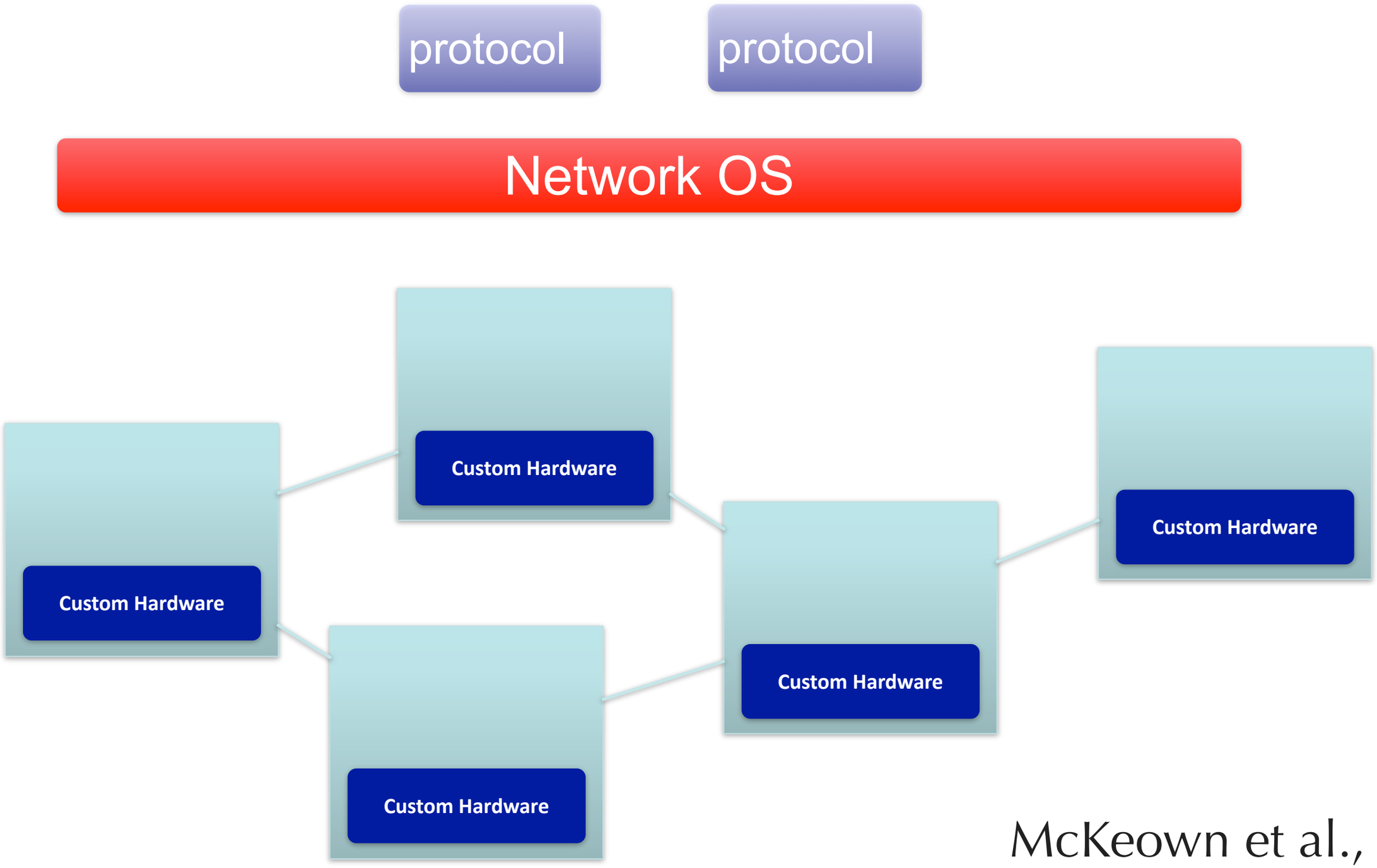
McKeown et al., [36]

OpenFlow: Open Network Control Plane



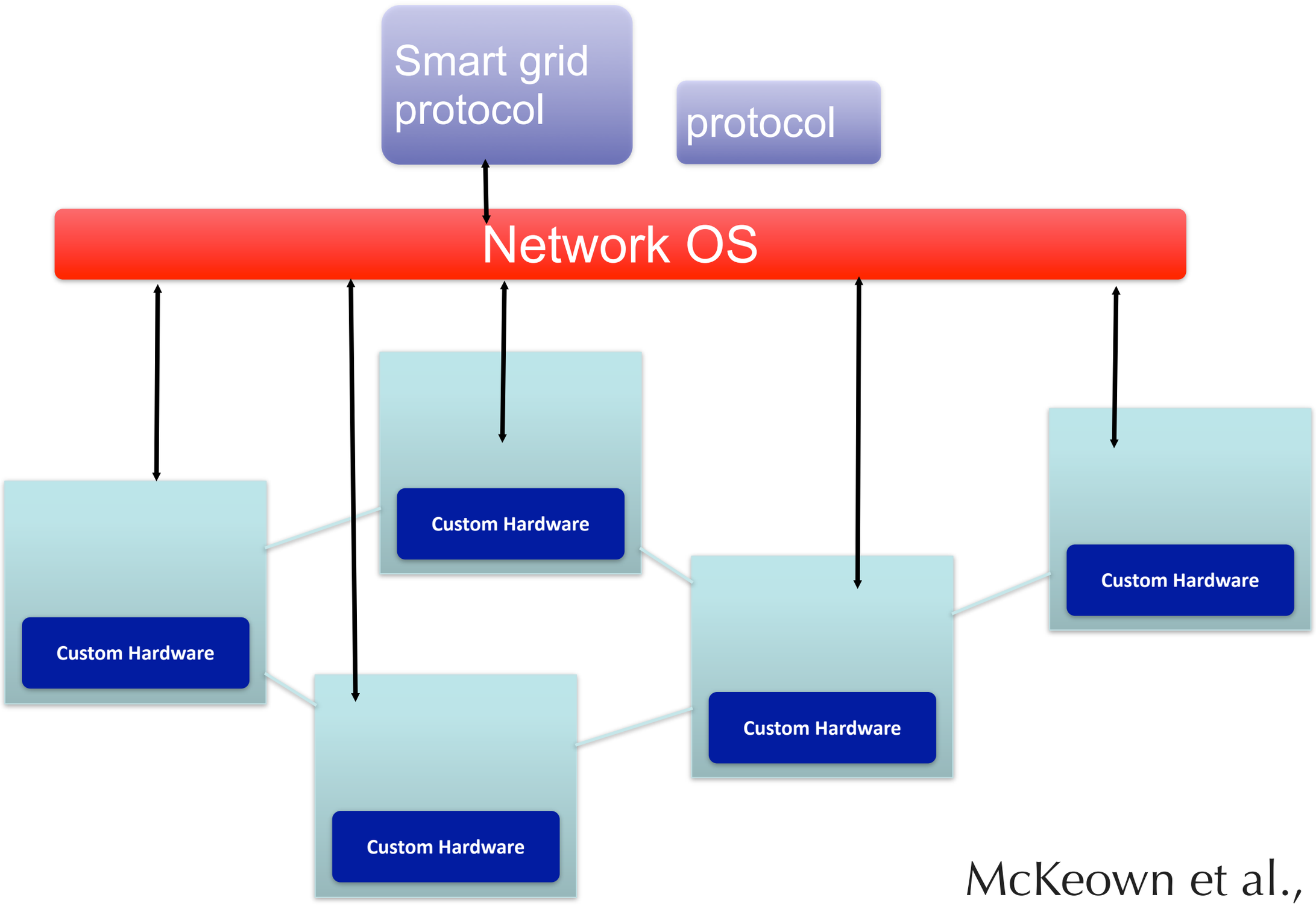
McKeown et al., [36]

OpenFlow: Open Network Control Plane



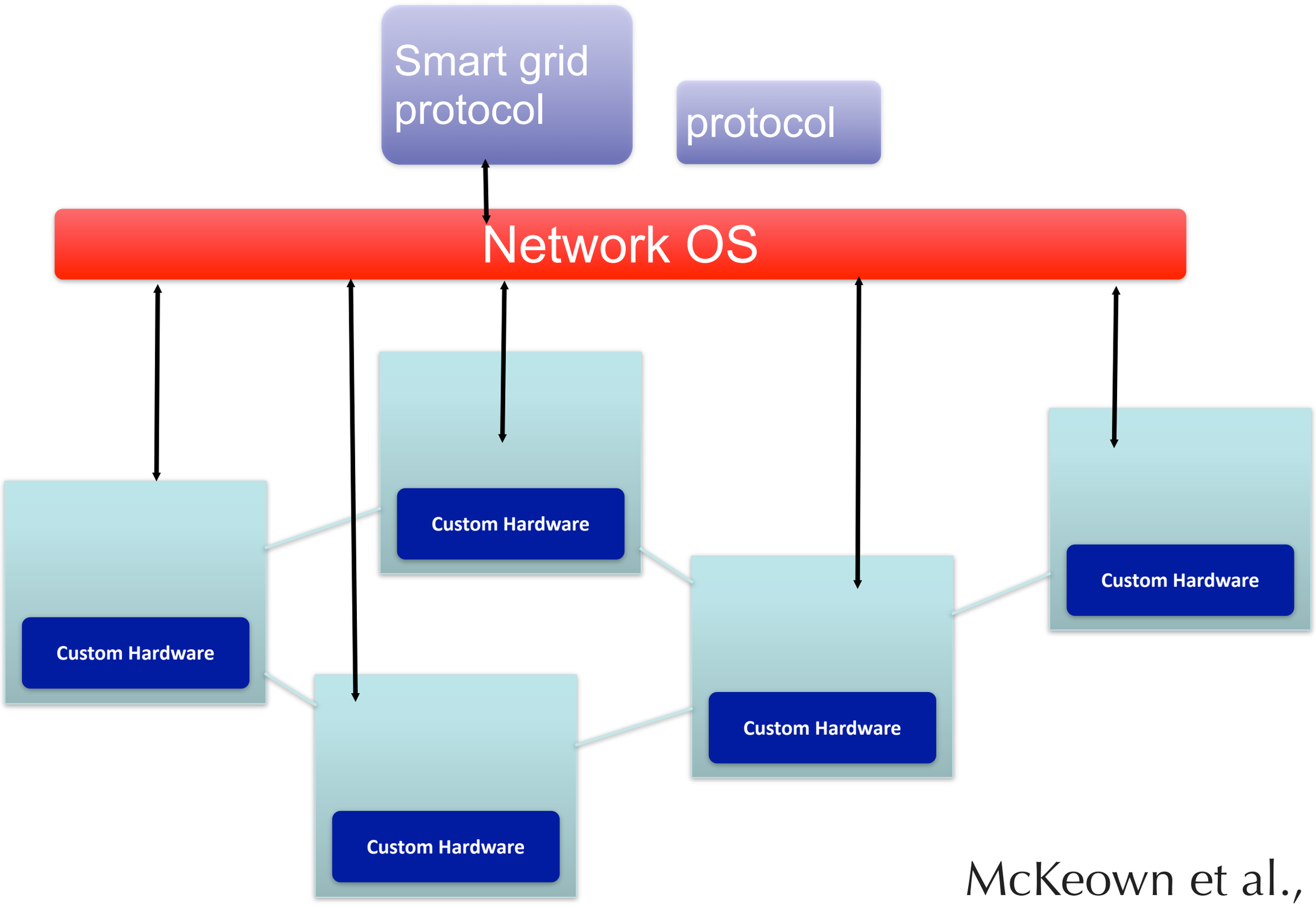
McKeown et al., [36]

OpenFlow: Open Smart Grid Control Plane



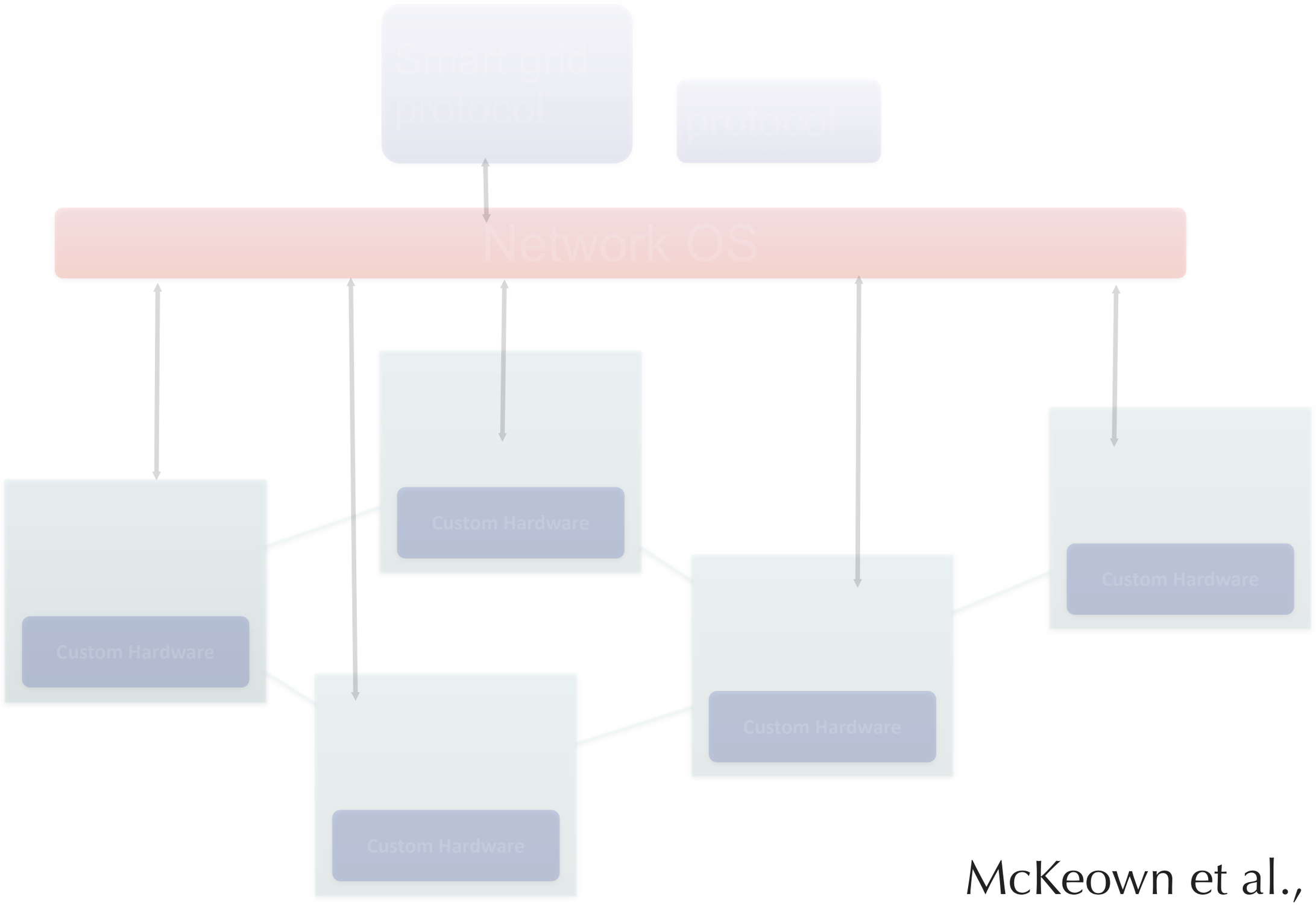
McKeown et al., [36]

OpenFlow: Open Smart Grid Control Plane



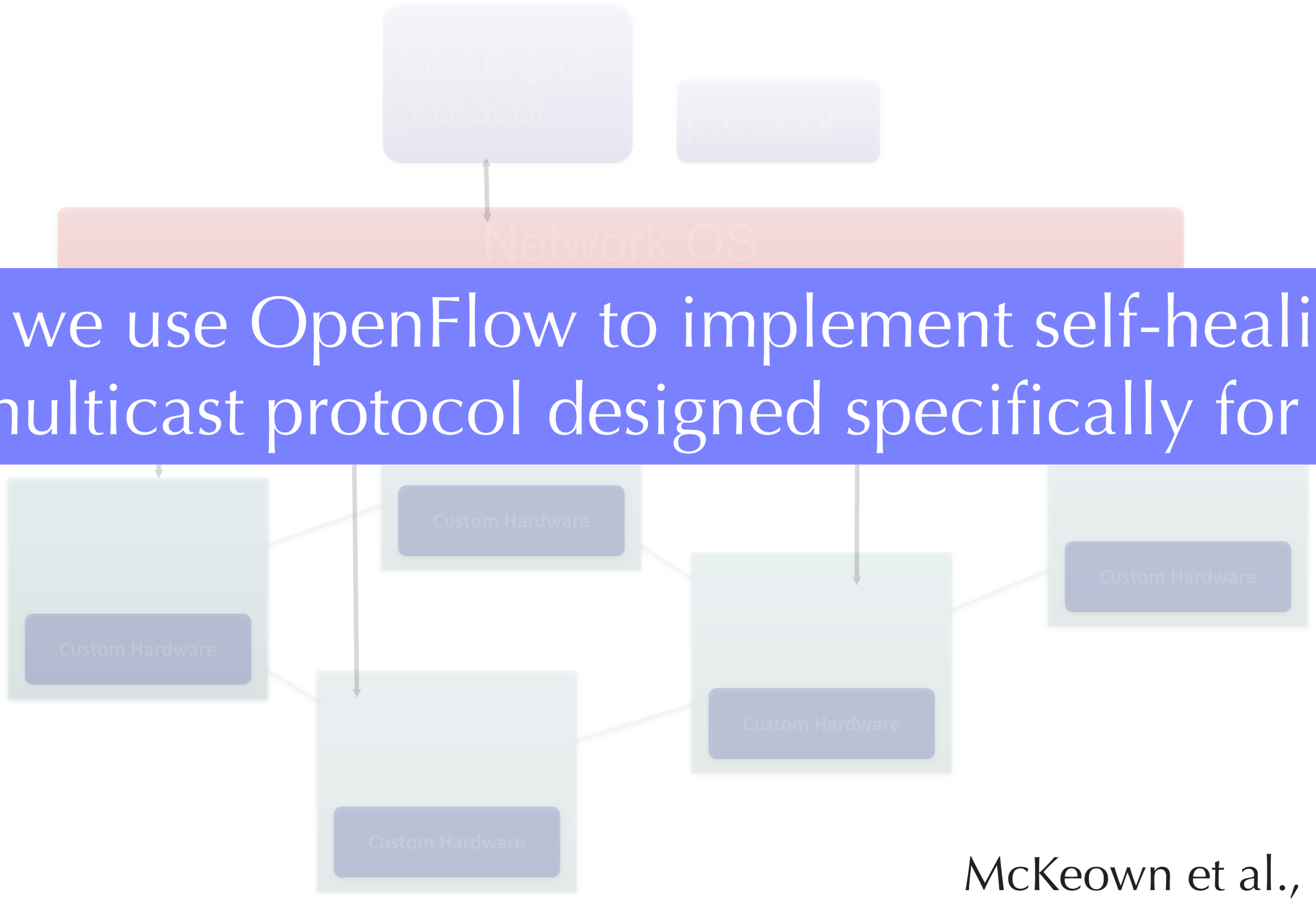
McKeown et al., [36]

OpenFlow: Open Smart Grid Control Plane



McKeown et al., [36]

OpenFlow: Open Smart Grid Control Plane



we use OpenFlow to implement self-healing multicast protocol designed specifically for grid

McKeown et al., [36]

Why Develop a New Solution?

Why Develop a New Solution?

- public Internet:
  - best-effort model unsuitable to meeting hard E2E delay requirements Bakken et al. [7]
  - general purpose, not optimized for power grid
    - small scale, traffic is known

Why Develop a New Solution?

- public Internet:
  - best-effort model unsuitable to meeting hard E2E delay requirements Bakken et al. [7]
  - general purpose, not optimized for power grid
    - small scale, traffic is known
- Gridstat:
  - pub-sub system proposed by Bakken et al. [7]
  - does not address link failures

Why Develop a New Solution?

- MPLS fast reroute ([18,23,37,42,47]) similar but
  - typically consider unicast flows
  - optimization criteria specified over single MT
    - we consider criteria across multiple MTs
    - Li et al. [Infocomm 2006] is an exception
      - ★ minimize total backup path bandwidth reservation across *all* backup paths (across all MTs)

Link Failure Detection

are packets being lost at a given network link?

Link Failure Detection

are packets being lost at a given network link?

hypothesis: detecting packet loss inside network faster than E2E detection

Link Failure Detection

are packets being lost at a given network link?

hypothesis: detecting packet loss inside network faster than E2E detection

- leverage OpenFlow native packet counters + ability to tag packets

1. mark and count packets at upstream switch
2. count the marked packets at the downstream switch

Link Failure Detection

are packets being lost at a given network link?

hypothesis: detecting packet loss inside network faster than E2E detection

- leverage OpenFlow native packet counters + ability to tag packets

  1. mark and count packets at upstream switch
  2. count the marked packets at the downstream switch

- ensure that upstream and downstream switches consider the same set of packets

Precomputing Backup MTs

- propose 3 Algorithms
  - MIN-FLOWS, MIN-SINKS, MIN-CONTROL
  - same input and output (objective function differs)

Precomputing Backup MTs

- propose 3 Algorithms
  - MIN-FLOWS, MIN-SINKS, MIN-CONTROL
  - same input and output (objective function differs)

Input:
- undirected graph, set of all multicast trees, set of all multicast flows, a link (`l`)

Precomputing Backup MTs

- propose 3 Algorithms
  - MIN-FLOWS, MIN-SINKS, MIN-CONTROL
  - same input and output (objective function differs)

Input:

- undirected graph, set of all multicast trees, set of all multicast flows, a link (`l`)

Output:

- a backup MT for each MT using `l` (call this set T_l) s.t. if all MTs in T_l were installed:
  - all E2E per-packet delay requirements are met
  - objective function optimized

Optimize for Robustness

MIN-FLOW & MIN-SINK compute backup MTs that minimize worst case impact of next link failure

Optimize for Robustness

MIN-FLOW & MIN-SINK compute backup MTs that minimize worst case impact of next link failure

+ more robust network

Optimize for Robustness

MIN-FLOW & MIN-SINK compute backup MTs that minimize worst case impact of next link failure

\+ more robust network

– MT may have longer paths

MIN-FLOWS Backup Multicast Trees

- objective function
  - set of backup MTs (T_1) should be computed s.t. *across all network links and all network flows*, the maximum # of flows traversing a single link is minimized

MIN-FLOWS Backup Multicast Trees

global optimization criteria

- objective function
  - set of backup MTs (T_1) should be computed s.t. *across all network links and all network flows*, the maximum # of flows traversing a single link is minimized

MIN-FLOWS Backup Multicast Trees

- objective function
  - set of backup MTs (T_1) should be computed s.t. *across all network links and all network flows*, the maximum # of flows traversing a single link is minimized

global optimization criteria

for next link failure, protects against worst case

MIN-SINKS Backup Multicast Trees

- objective function
  - set of backup MTs (T_1) should be computed s.t. *across all network links and all sink nodes*, the maximum # of downstream sink nodes corresponding to a single link is minimized

MIN-SINKS Backup Multicast Trees

global optimization criteria

- objective function
  - set of backup MTs (T_1) should be computed s.t. *across all network links and all sink nodes*, the maximum # of downstream sink nodes corresponding to a single link is minimized

MIN-SINKS Backup Multicast Trees

global optimization criteria

- objective function
  - set of backup MTs (T_1) should be computed s.t. *across all network links and all sink nodes*, the maximum # of downstream sink nodes corresponding to a single link is minimized

for next link failure, protects against worst case

MIN-CONTROL Backup MTs

MIN-CONTROL Backup MTs

- intuition
  - quick installation of MTs by ensuring only a small # of switches need to be signaled

MIN-CONTROL Backup MTs

- intuition
  - quick installation of MTs by ensuring only a small # of switches need to be signaled
- objective function
  - the set of backup MTs (T_l) should be computed s.t. the minimum # of routers need to be signaled to change state

Research Questions

Q1: How much faster is detecting link failures inside the network than E2E detection?

Research Questions

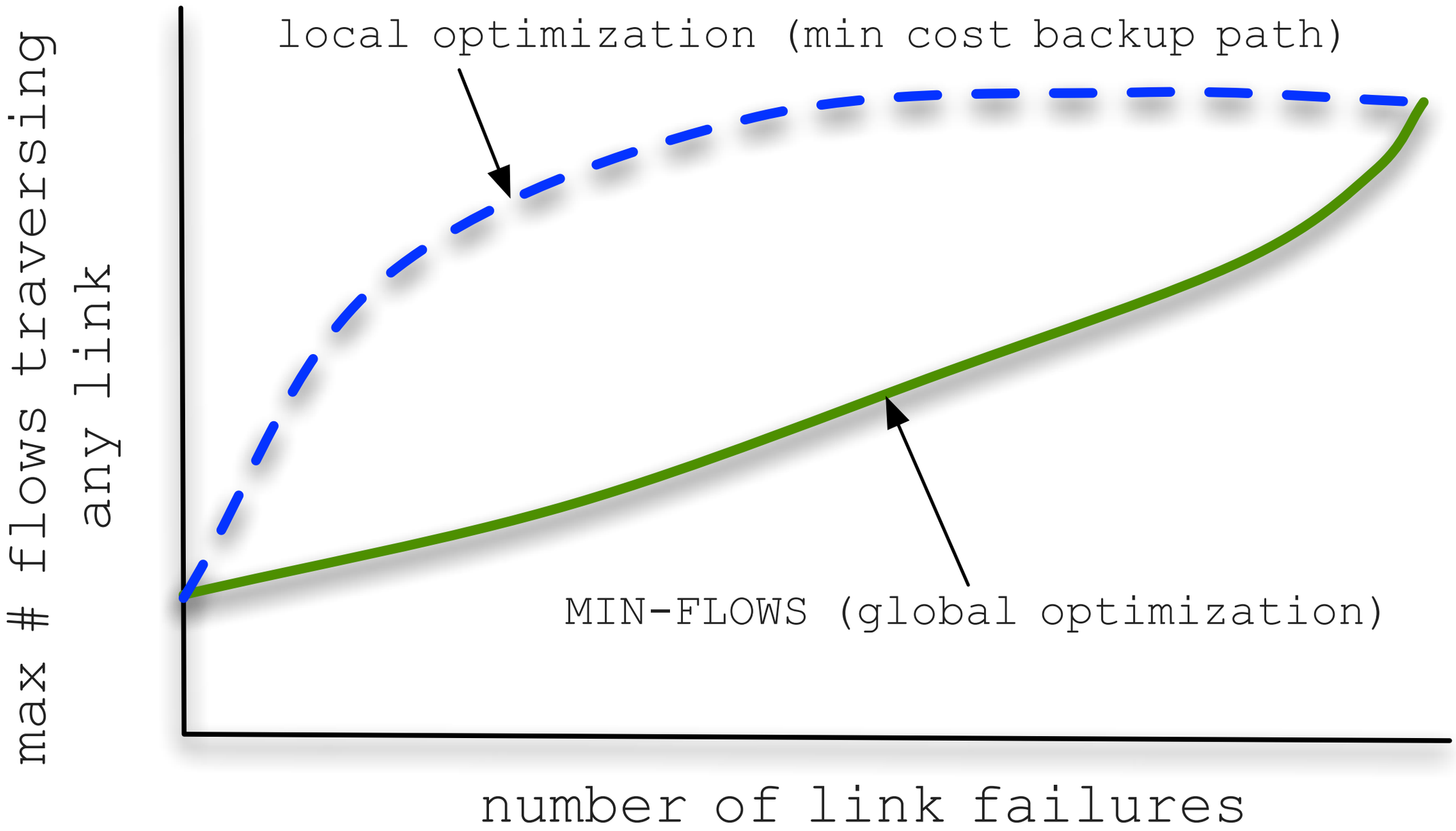
Q2: How to design and implement efficient algorithms for MIN-FLOWS, MIN-SINKS, + MIN-CONTROL?

Research Questions

Q3: How much more does global optimization increase robustness than doing localized opt.?

Research Questions

Q3: How much more does global optimization increase robustness than doing localized opt.?



Research Questions

Q4: how to efficiently install pre-computed backup MTs after link failure?

Research Questions

Q4: how to efficiently install pre-computed backup MTs after link failure?

OPTION 1

install entire backup tree after failure

Research Questions

Q4: how to efficiently install pre-computed backup MTs after link failure?

OPTION 1

install entire backup tree after failure

+ no extra fwd state

-- slow?

Research Questions

Q4: how to efficiently install pre-computed backup MTs after link failure?

OPTION 1

install entire backup tree after failure

+ no extra fwd state
-- slow?

OPTION 2

pre-install backup trees and, at the root node, switch to use backup tree after link failure (Kotani et al. [33])

Research Questions

Q4: how to efficiently install pre-computed backup MTs after link failure?

OPTION 1

install entire backup tree after failure

+ no extra fwd state
-- slow?

OPTION 2

pre-install backup trees and, at the root node, switch to use backup tree after link failure (Kotani et al. [33])

+ fast
-- much extra fwd state

Research Questions

Q4: how to efficiently install pre-computed backup MTs after link failure?

OPTION 1

install entire backup tree after failure

+ no extra fwd state
-- slow?

OPTION 2

pre-install backup trees and, at the root node, switch to use backup tree after link failure (Kotani et al. [33])

+ fast
-- much extra fwd state

OPTION 3 (Hybrid)

reuse flow entries from primary tree & only install non-overlapping sections of backup MT

Research Questions

Q4: how to efficiently install pre-computed backup MTs after link failure?

OPTION 1

install entire backup tree after failure

+ no extra fwd state
-- slow?

OPTION 2

pre-install backup trees and, at the root node, switch to use backup tree after link failure (Kotani et al. [33])

+ fast
-- much extra fwd state

OPTION 3 (Hybrid)

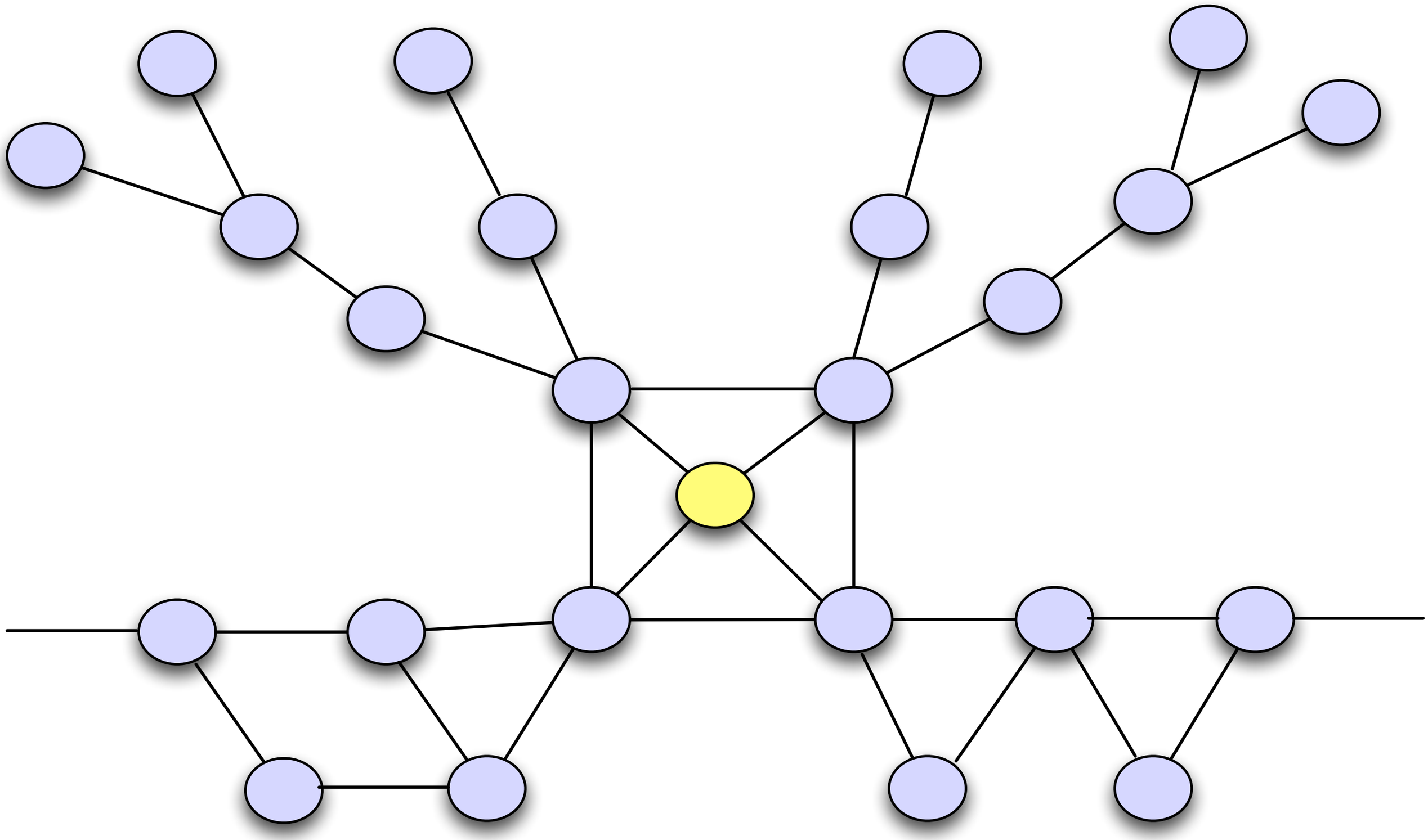
reuse flow entries from primary tree & only install non-overlapping sections of backup MT

+ less fwd state than Option 2
+ fast?

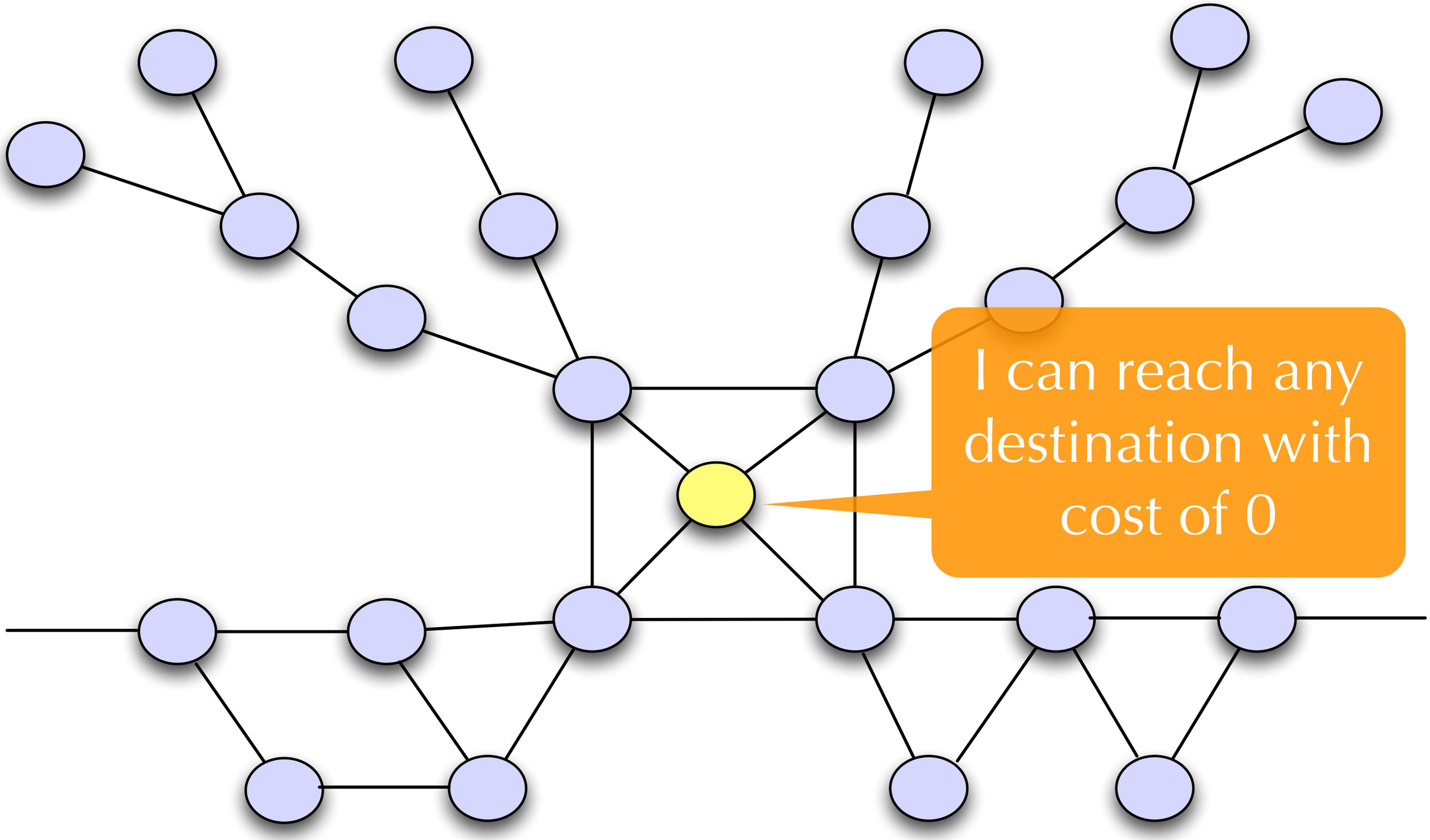
Talk Outline

- thesis introduction
- placement of smart grid sensors to enable measurement error detection
- recovery from failed communication links in a smart grid
- recovery from malicious nodes injecting false routing state
- outline for future work and conclusions

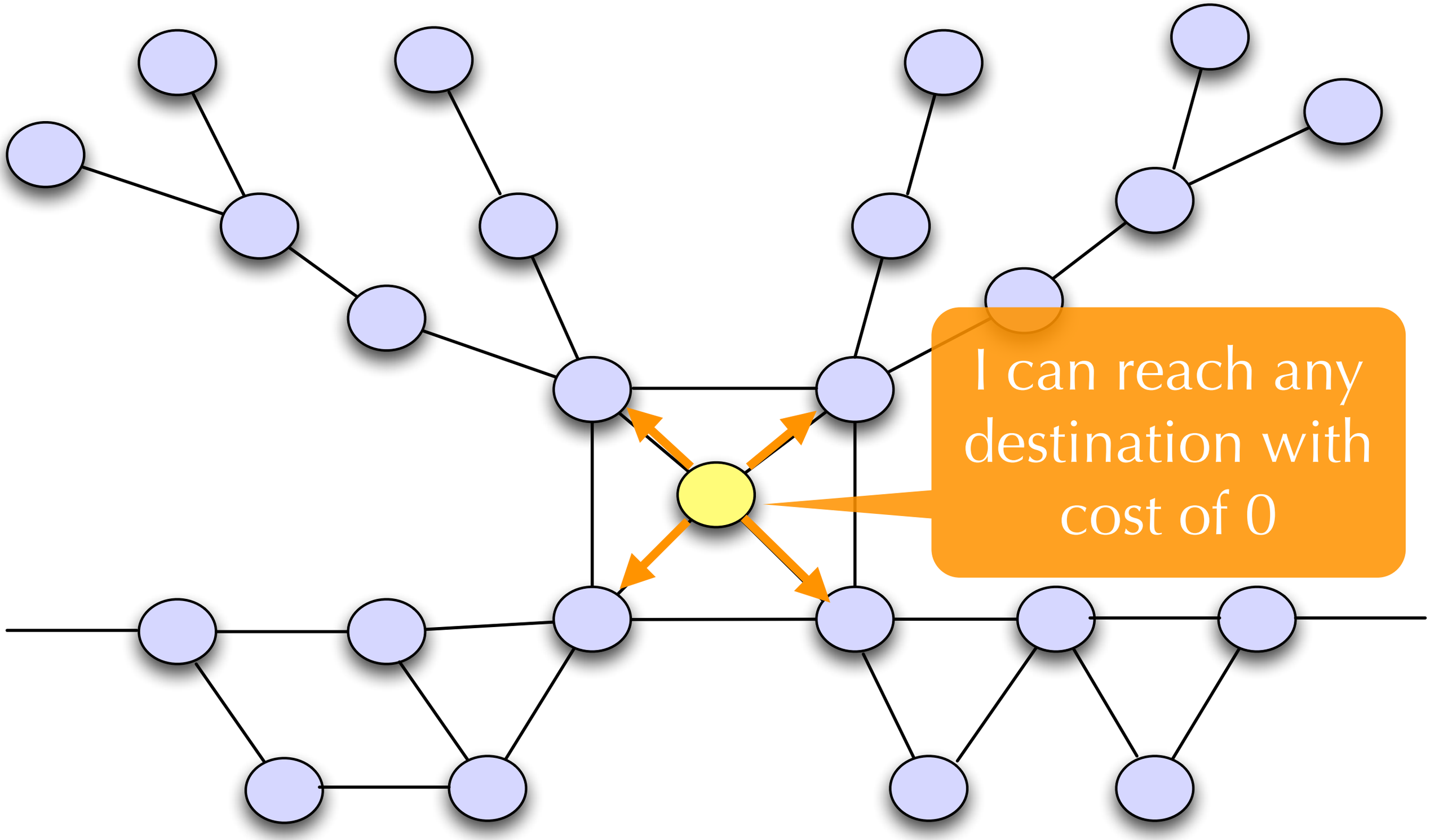
Ch 3: Network Router Failure



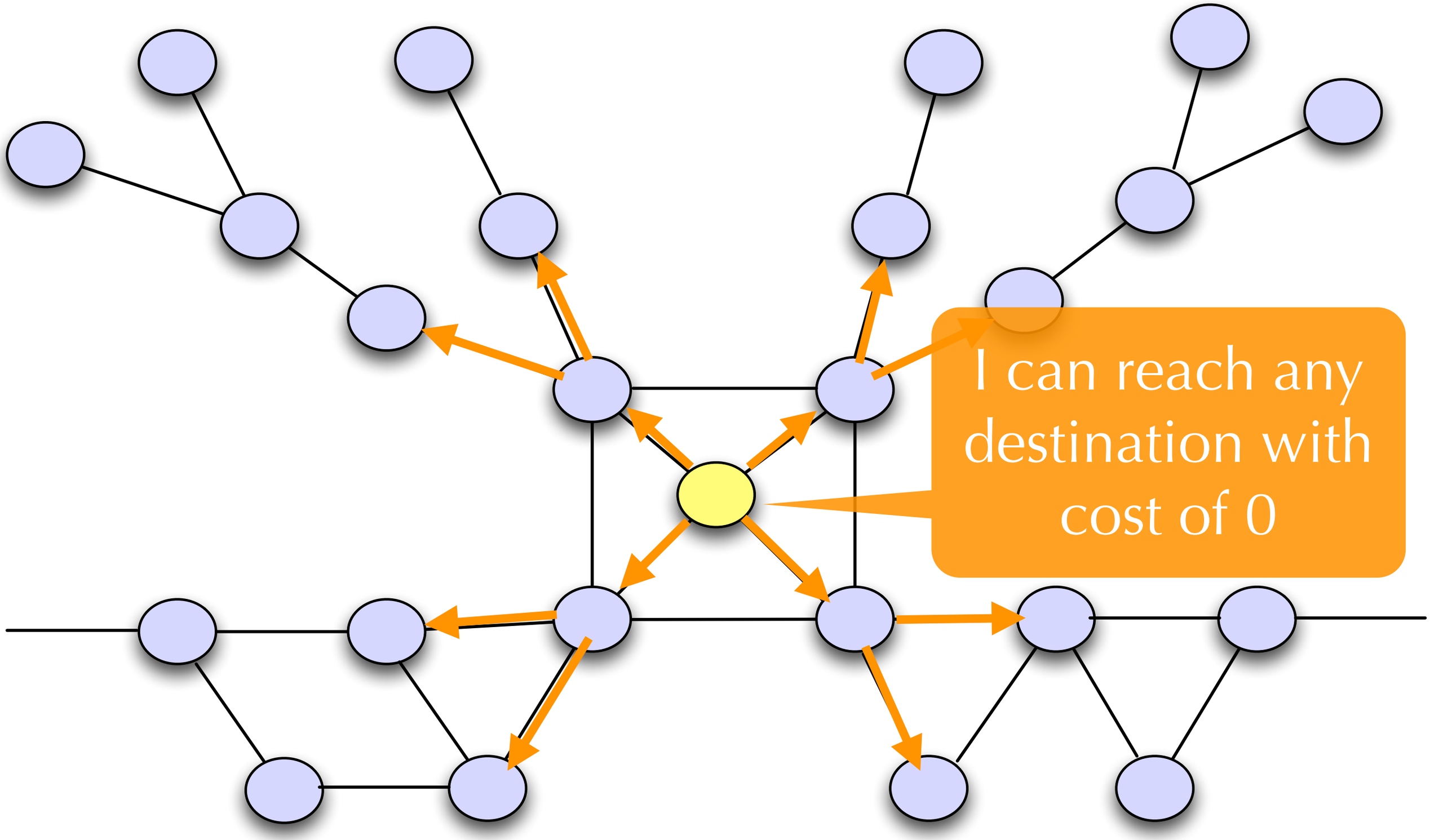
Ch 3: Network Router Failure



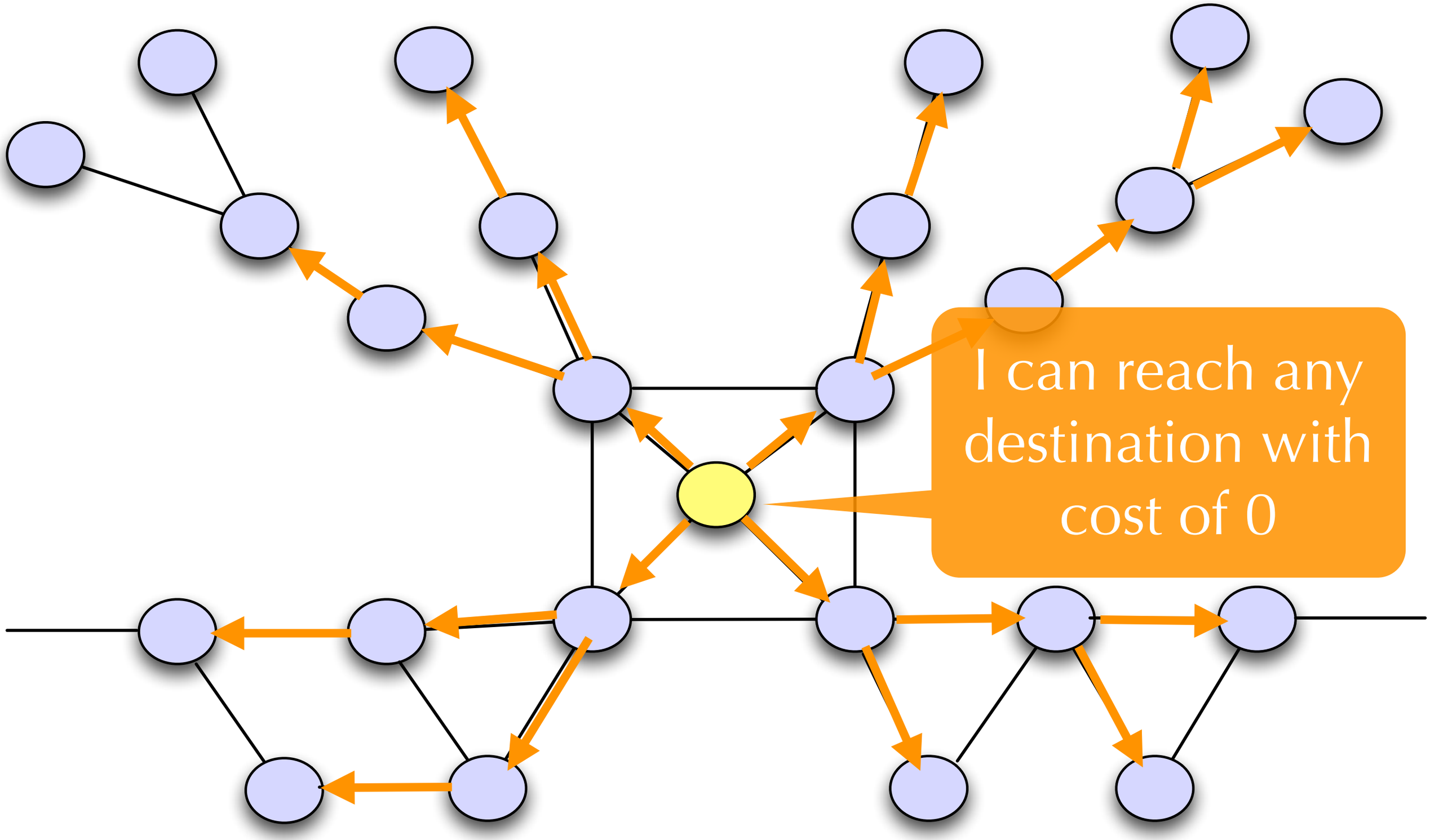
Ch 3: Network Router Failure



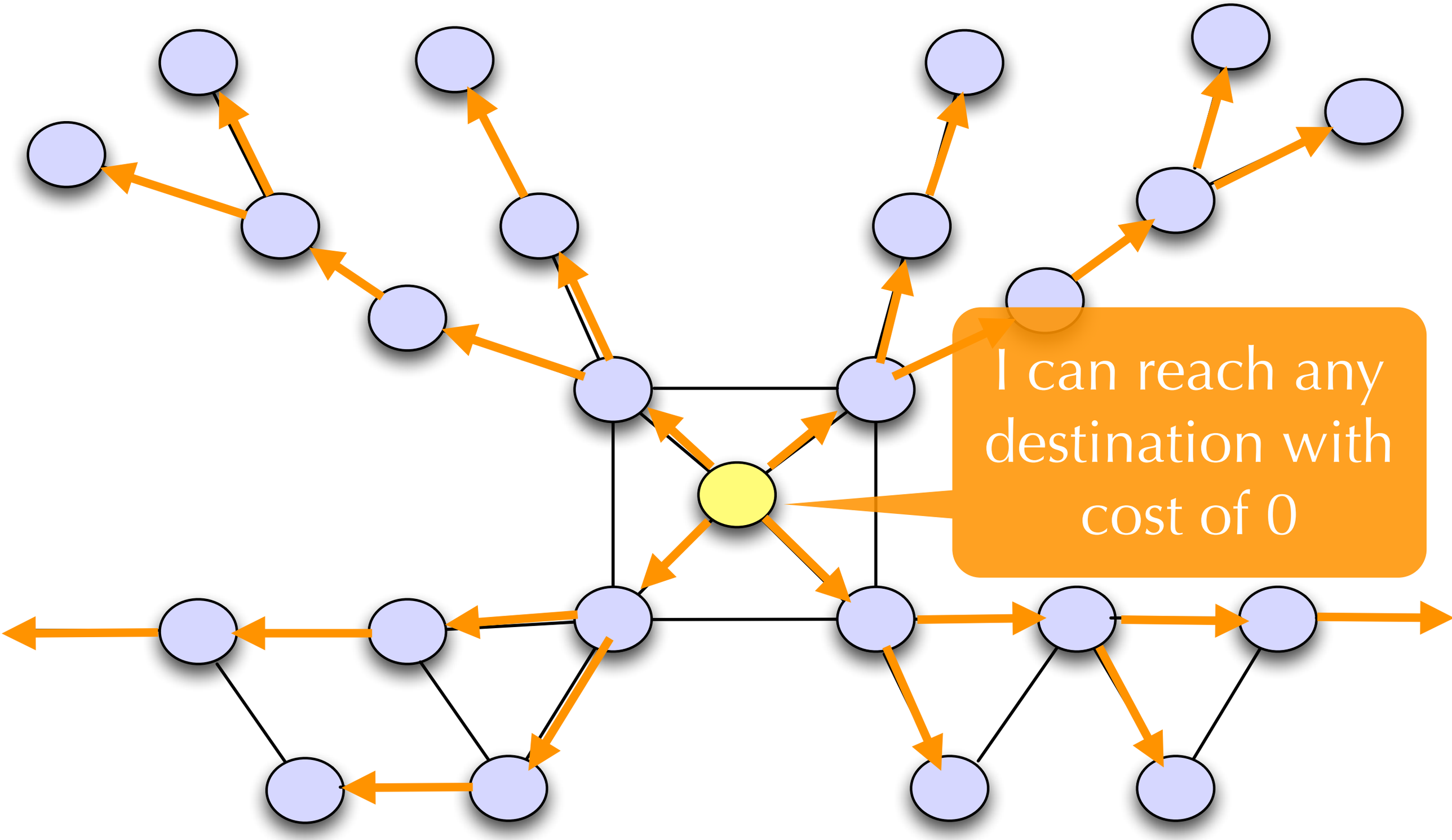
Ch 3: Network Router Failure



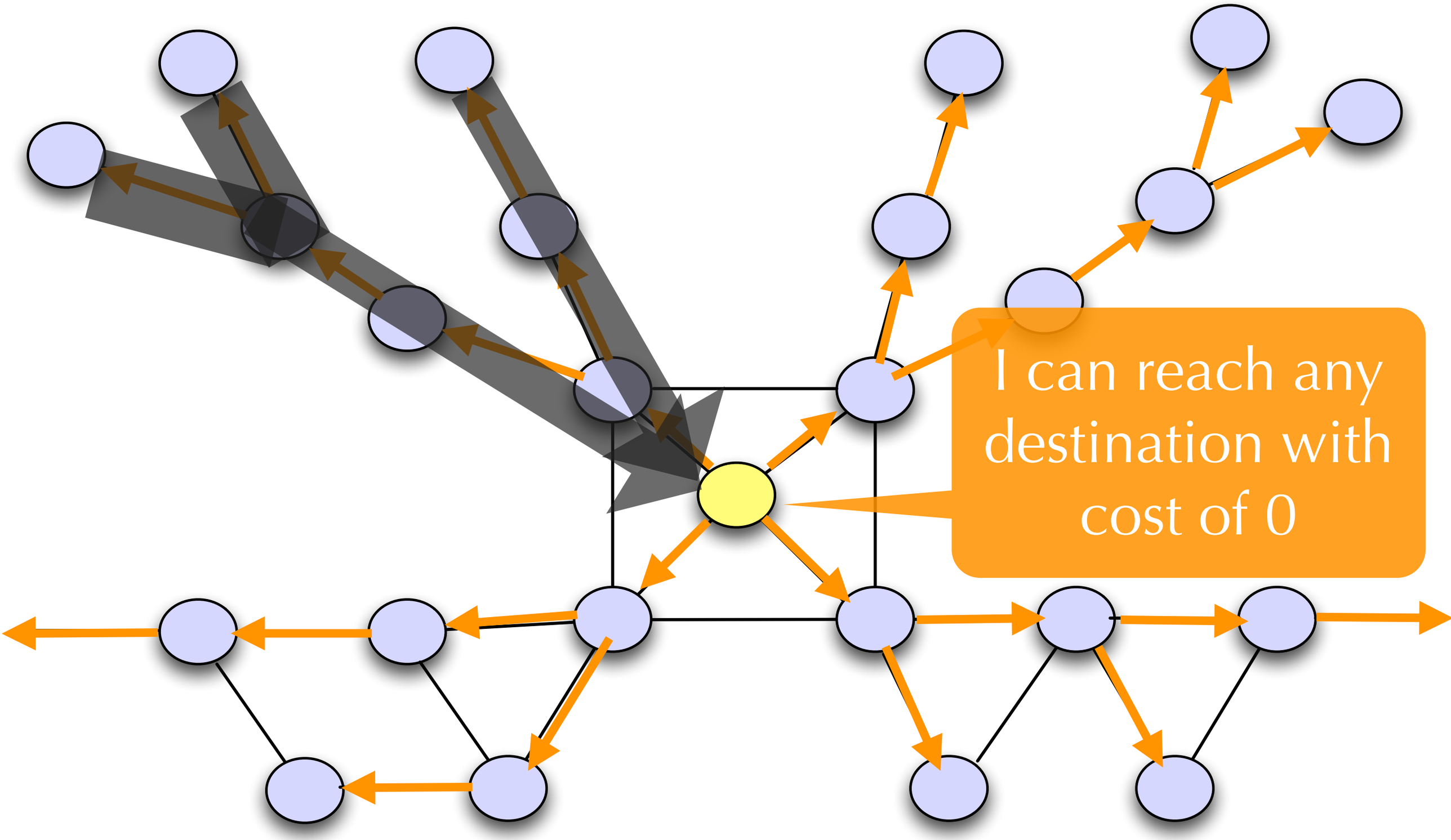
Ch 3: Network Router Failure



Ch 3: Network Router Failure



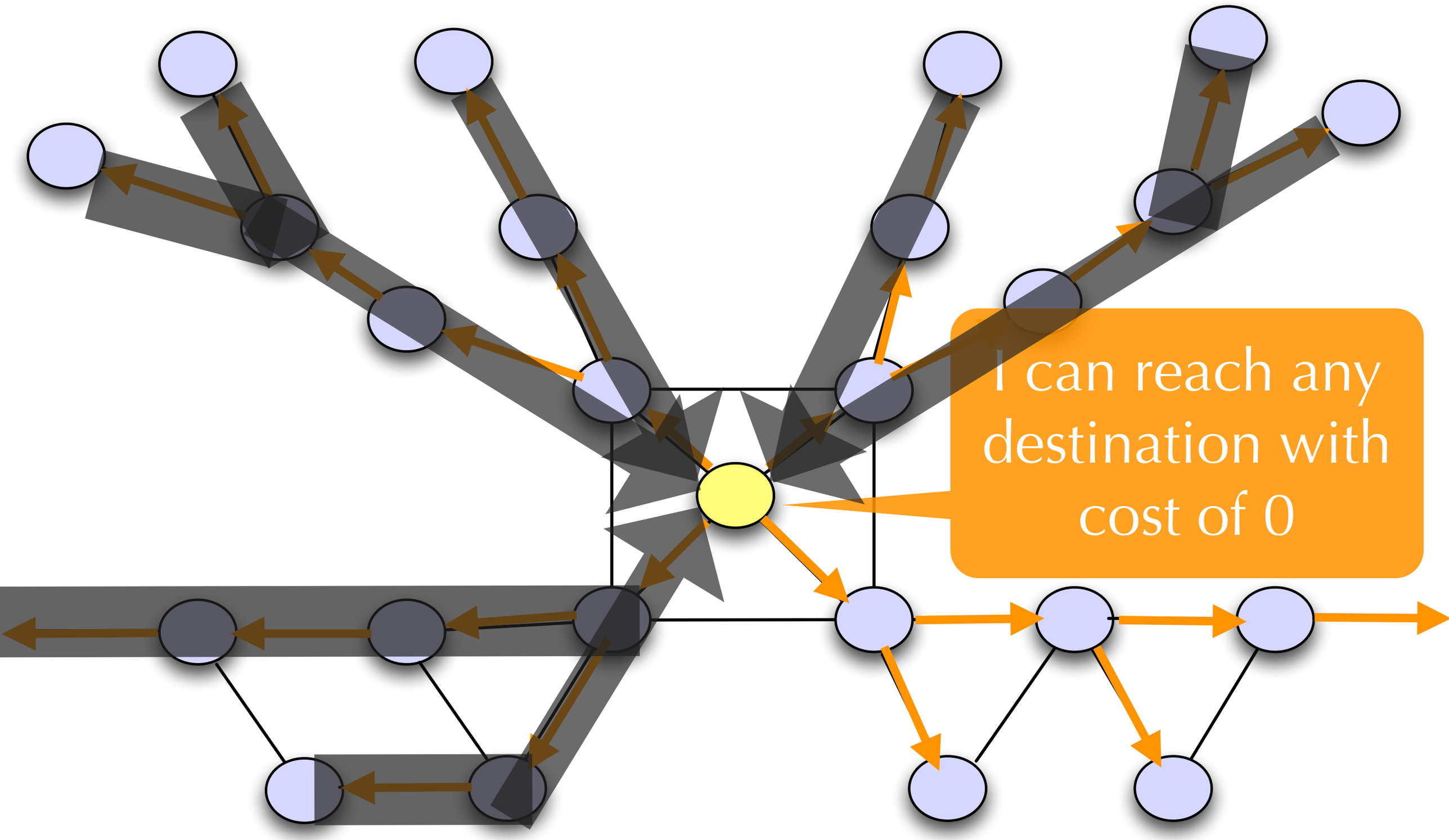
Ch 3: Network Router Failure



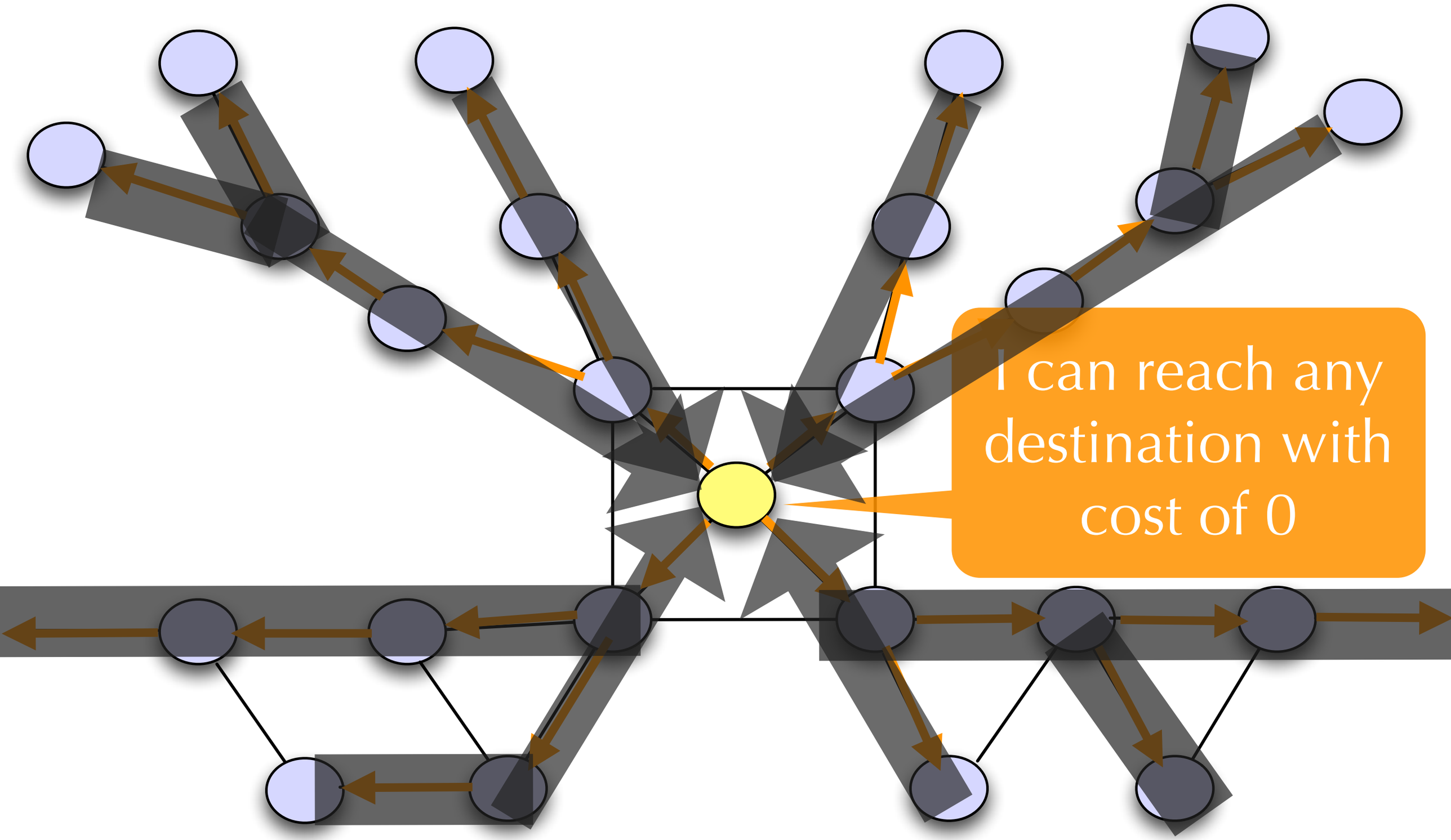
Ch 3: Network Router Failure

I can reach any destination with cost of 0

Ch 3: Network Router Failure



Ch 3: Network Router Failure



Problem Description

Problem Description

1. node have correct least cost paths

Problem Description

1. node have correct least cost paths
2. nodes compromised, sharing false routing state

Problem Description

1. node have correct least cost paths
2. nodes compromised, sharing false routing state
3. outside algorithm identifies compromised node

Problem Description

1. node have correct least cost paths
2. nodes compromised, sharing false routing state
3. outside algorithm identifies compromised node
4. recover

Problem Description

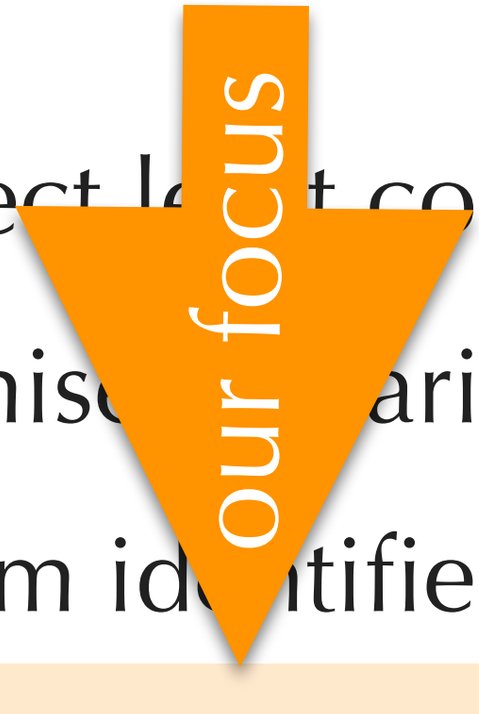
1. node have correct least cost paths
2. nodes compromised, sharing false routing state
3. outside algorithm identifies compromised node
4. recover
   a. remove compromised nodes from graph

Problem Description

1. node have correct least cost paths
2. nodes compromised, sharing false routing state
3. outside algorithm identifies compromised node
4. recover
   a. remove compromised nodes from graph
   b. compute least cost paths that route around compromised nodes

Problem Description

1. node have correct least cost paths
2. nodes compromise, sharing false routing state
3. outside algorithm identifies compromised node

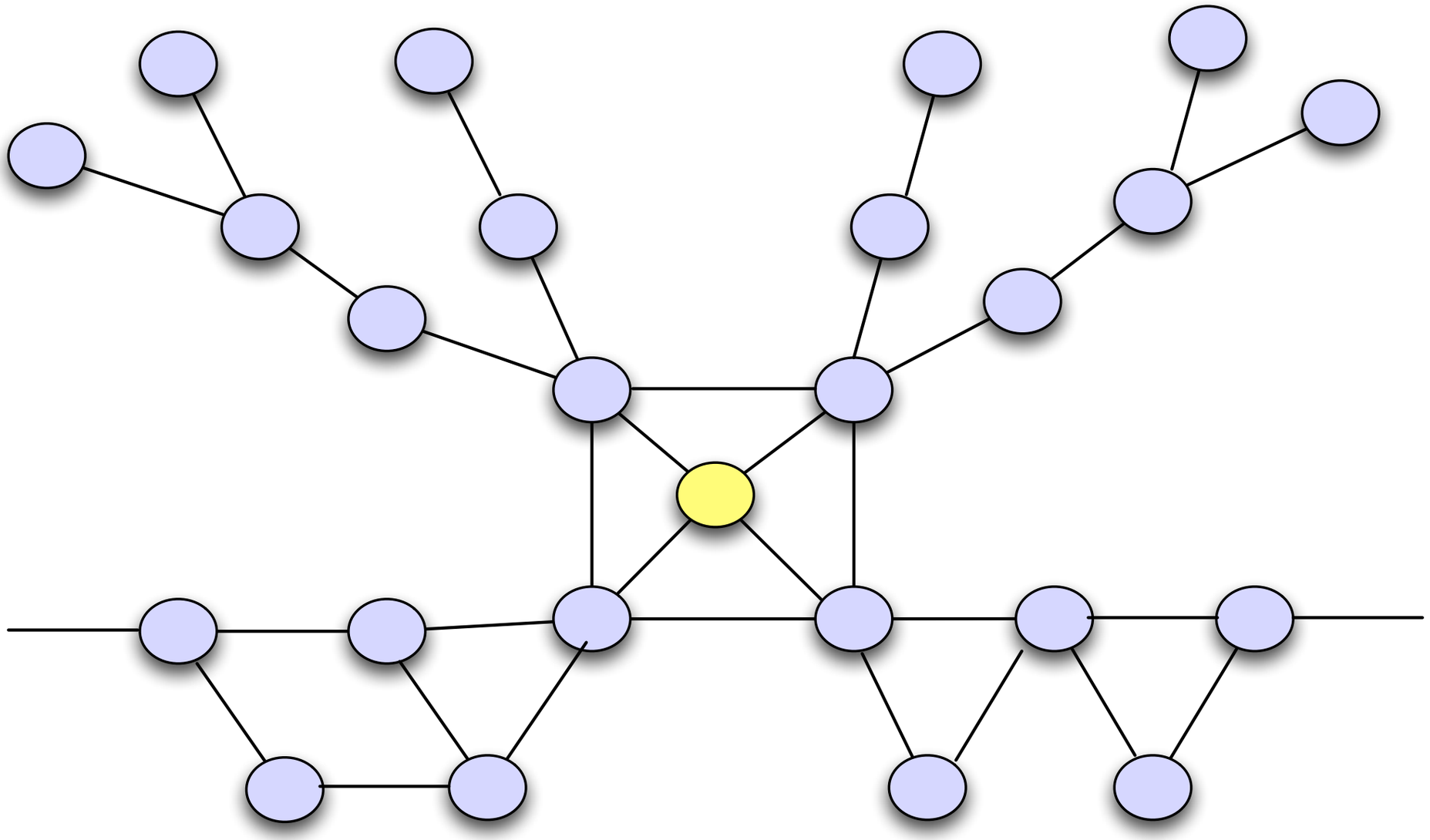


4. recover
   a. remove compromised nodes from graph
   b. compute least cost paths that route around compromised nodes

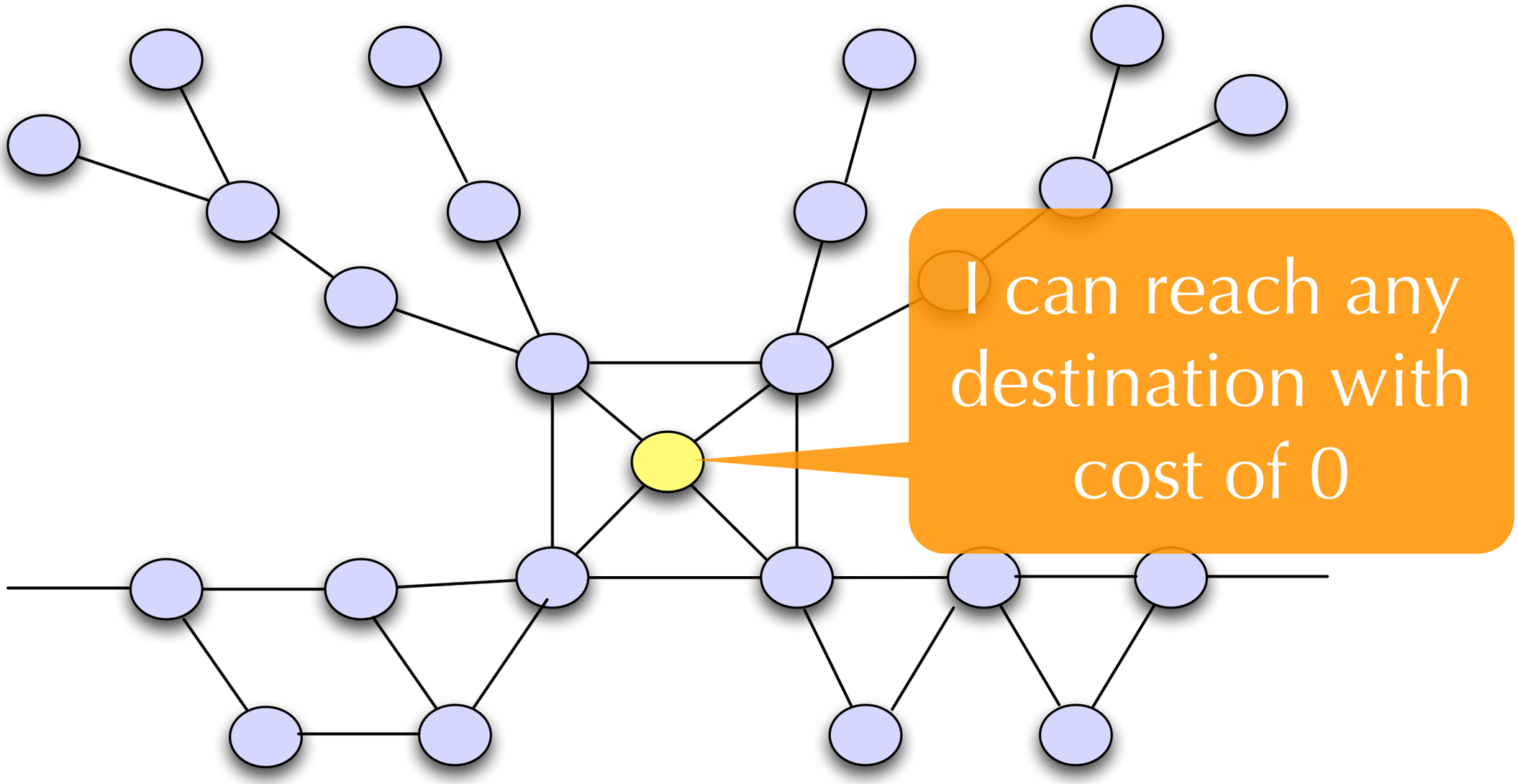
3 Recovery Algorithms

1. 2nd BEST algorithm
2. PURGE algorithm
3. CPR algorithm

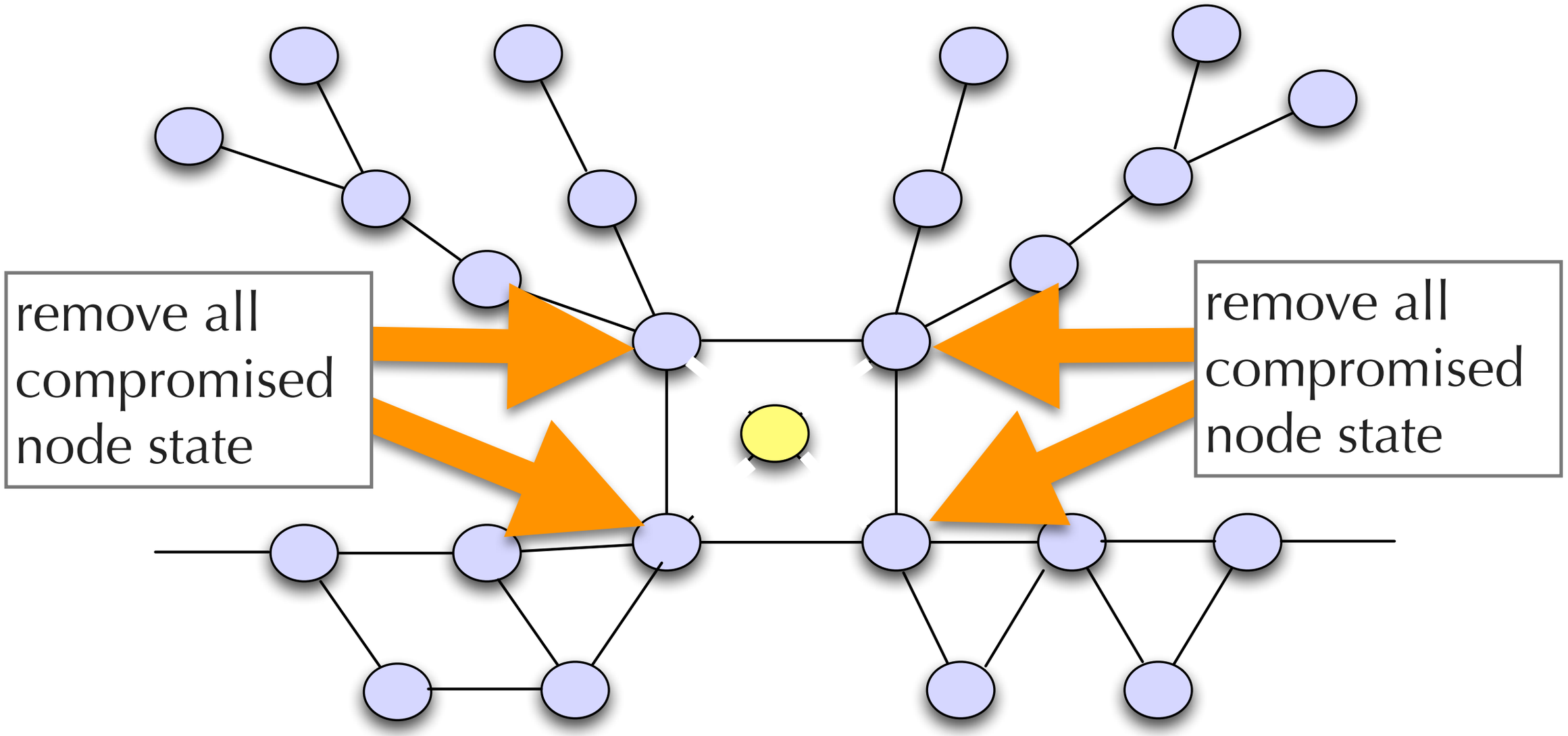
2nd BEST Recovery Algorithm



2nd BEST Recovery Algorithm

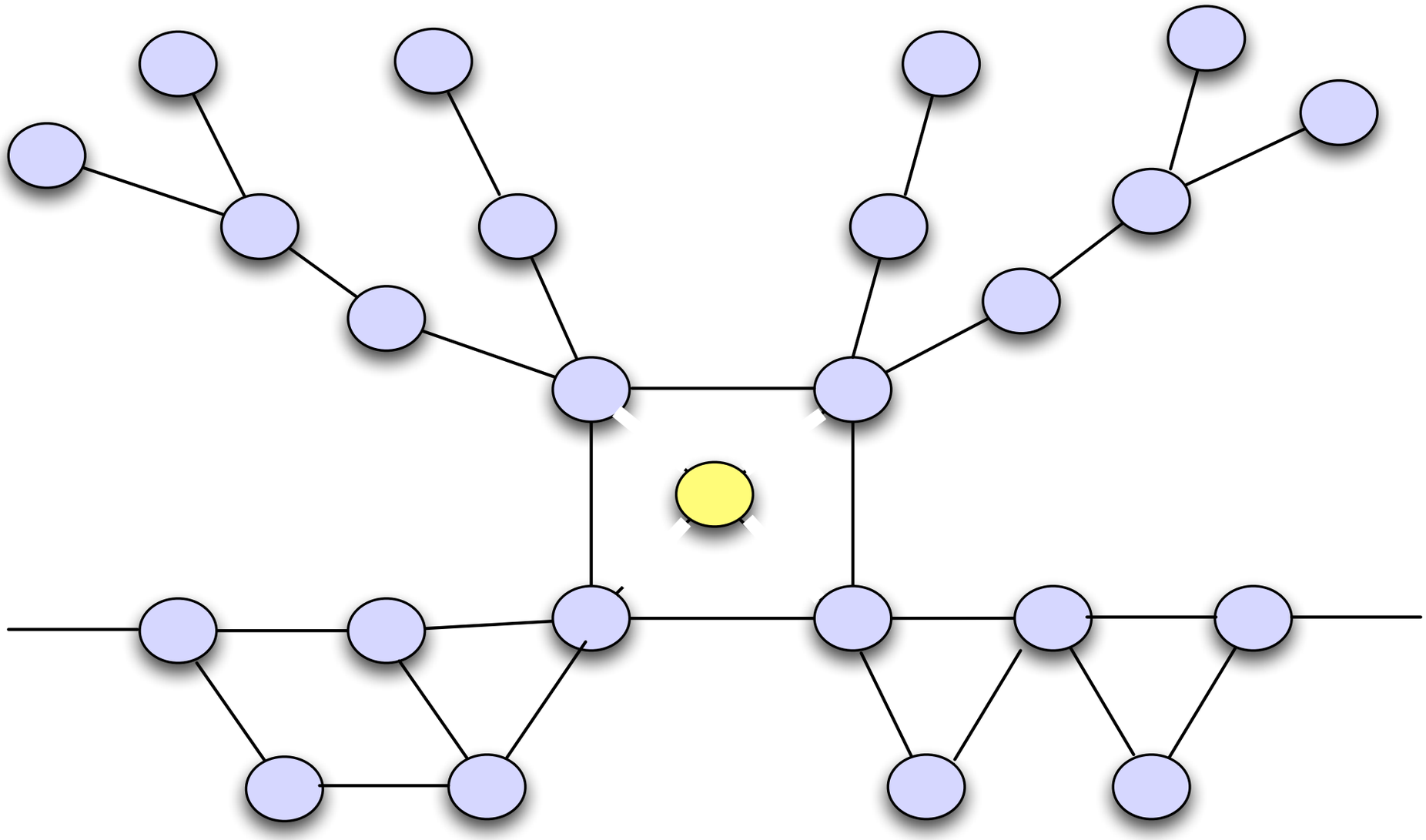


2nd BEST Recovery Algorithm



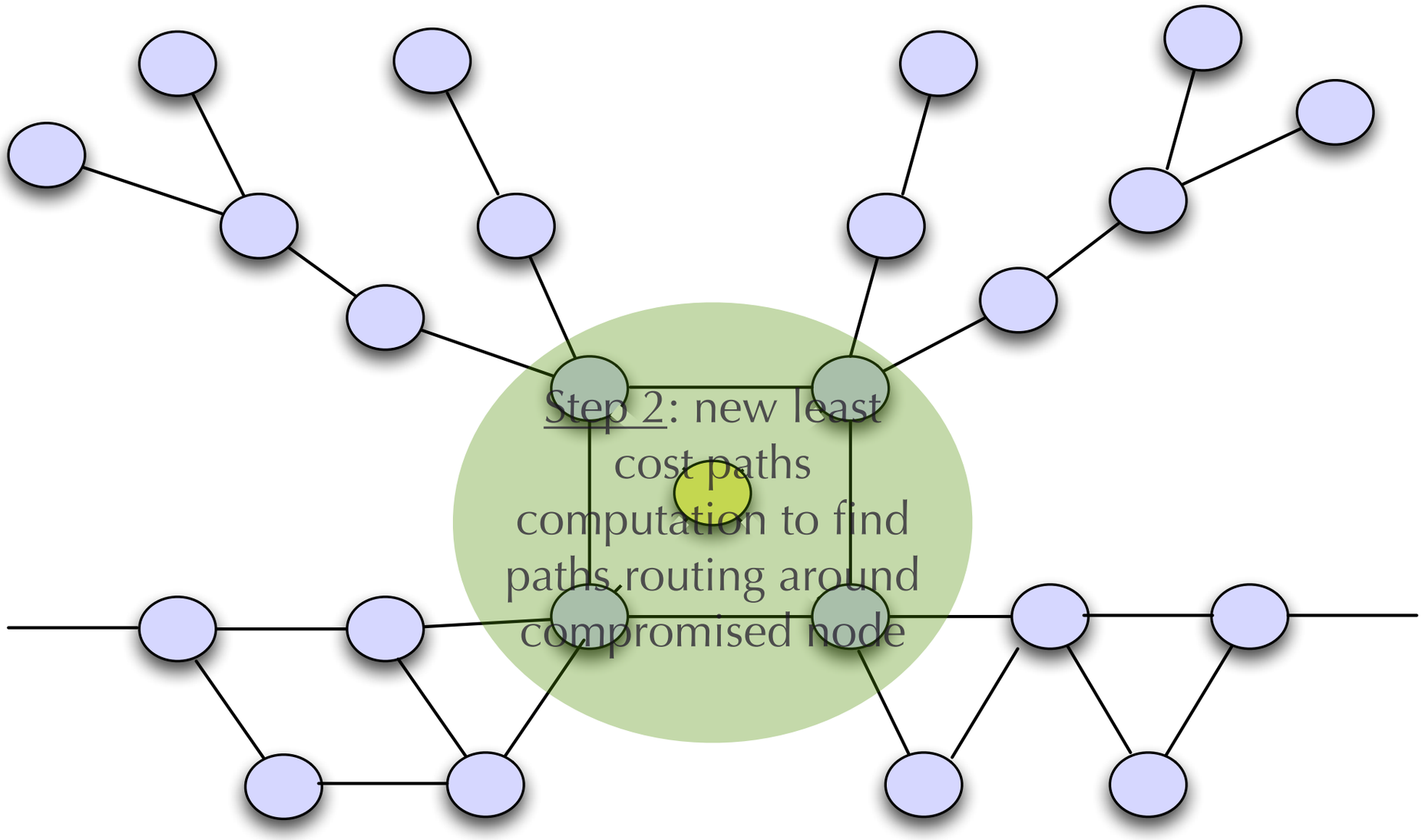
1. neighbors of compromised node locally remove false routing state

2nd BEST Recovery Algorithm



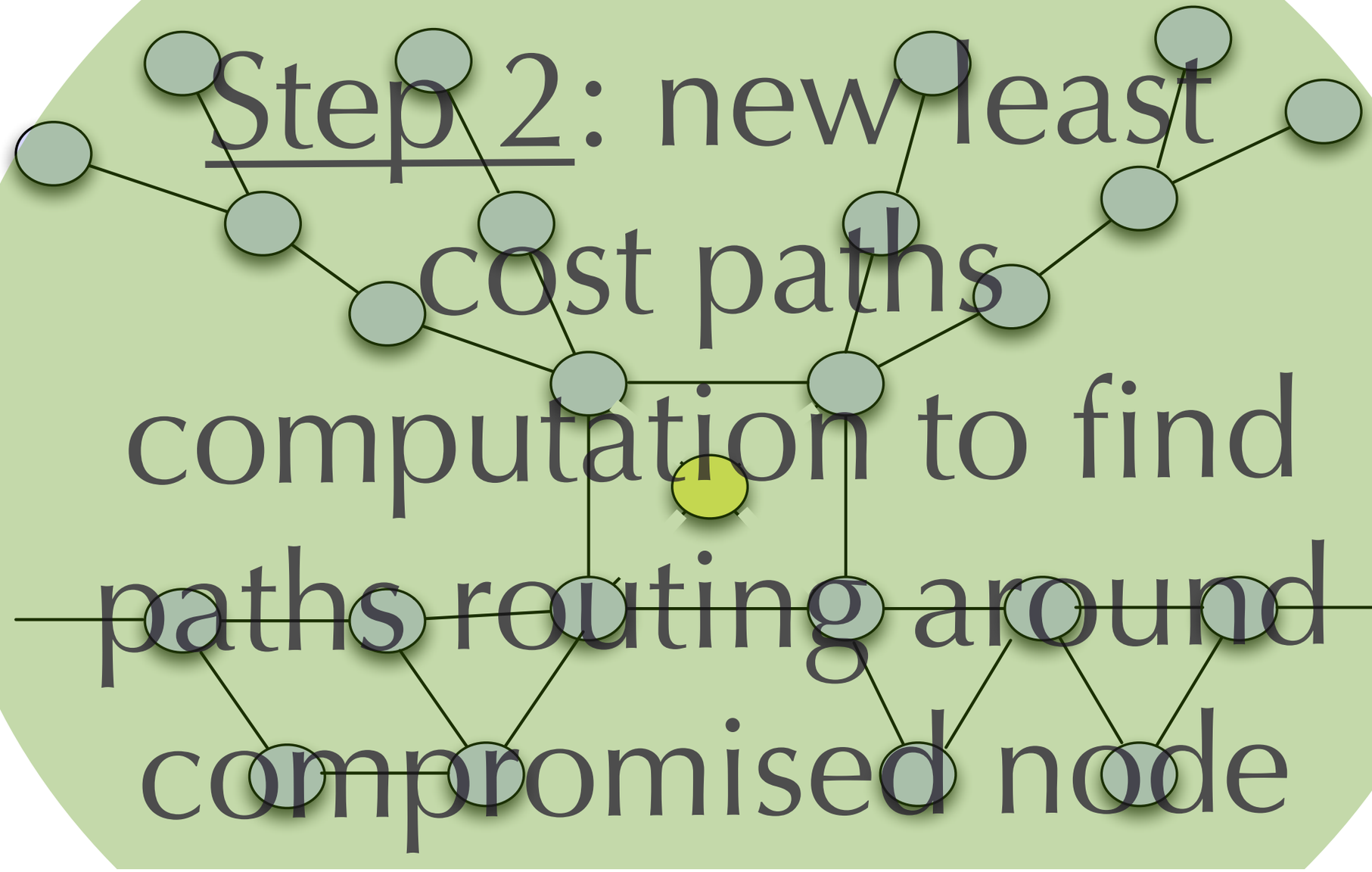
1. neighbors of compromised node locally remove false routing state
2. start distance vector computation to compute new least cost paths routing around compromised node

2nd BEST Recovery Algorithm



1. neighbors of compromised node locally remove false routing state
2. start distance vector computation to compute new least cost paths routing around compromised node

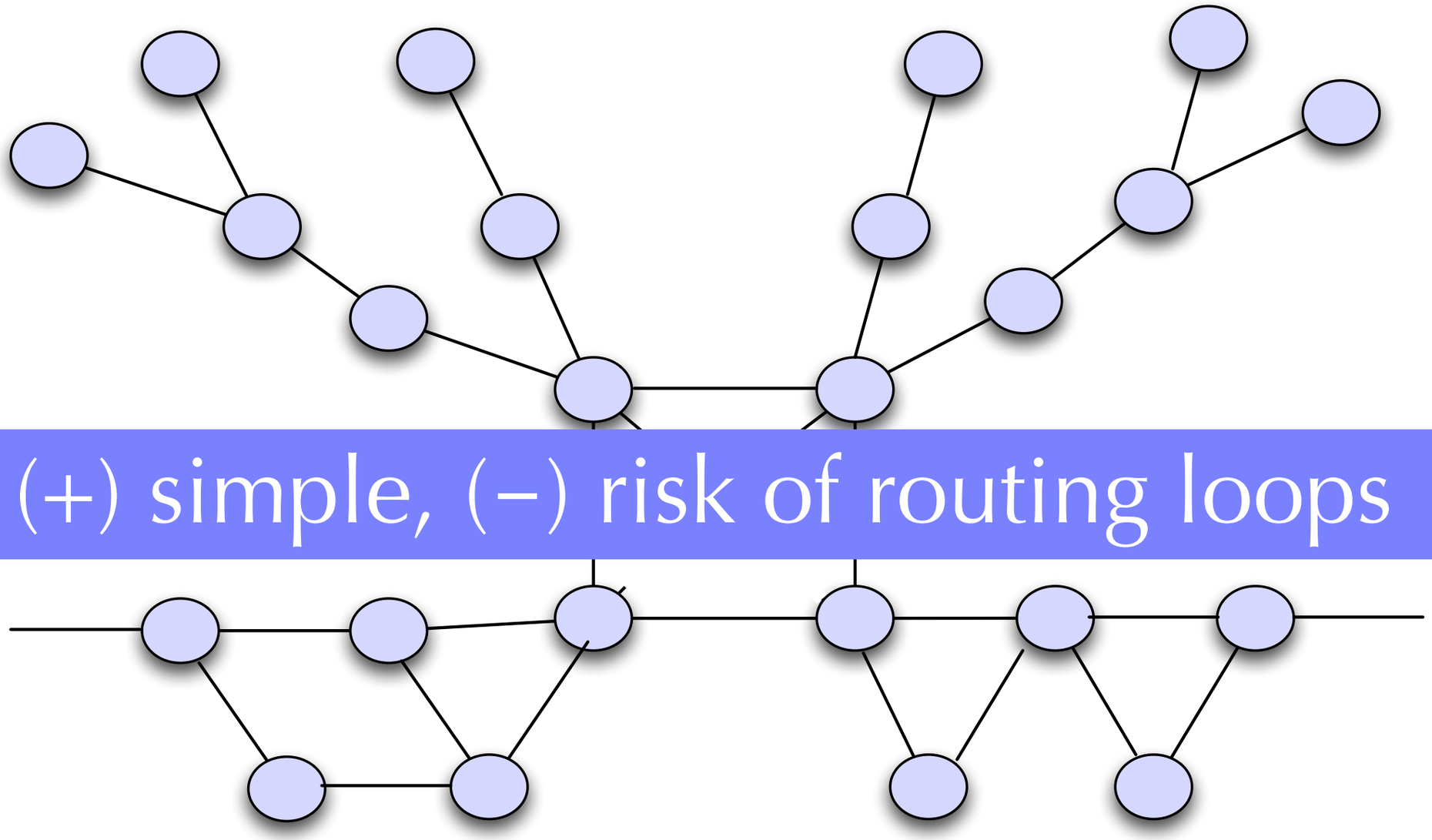
2nd BEST Recovery Algorithm



Step 2: new least cost paths computation to find paths routing around compromised node

1. neighbors of compromised node locally remove false routing state
2. start distance vector computation to compute new least cost paths routing around compromised node

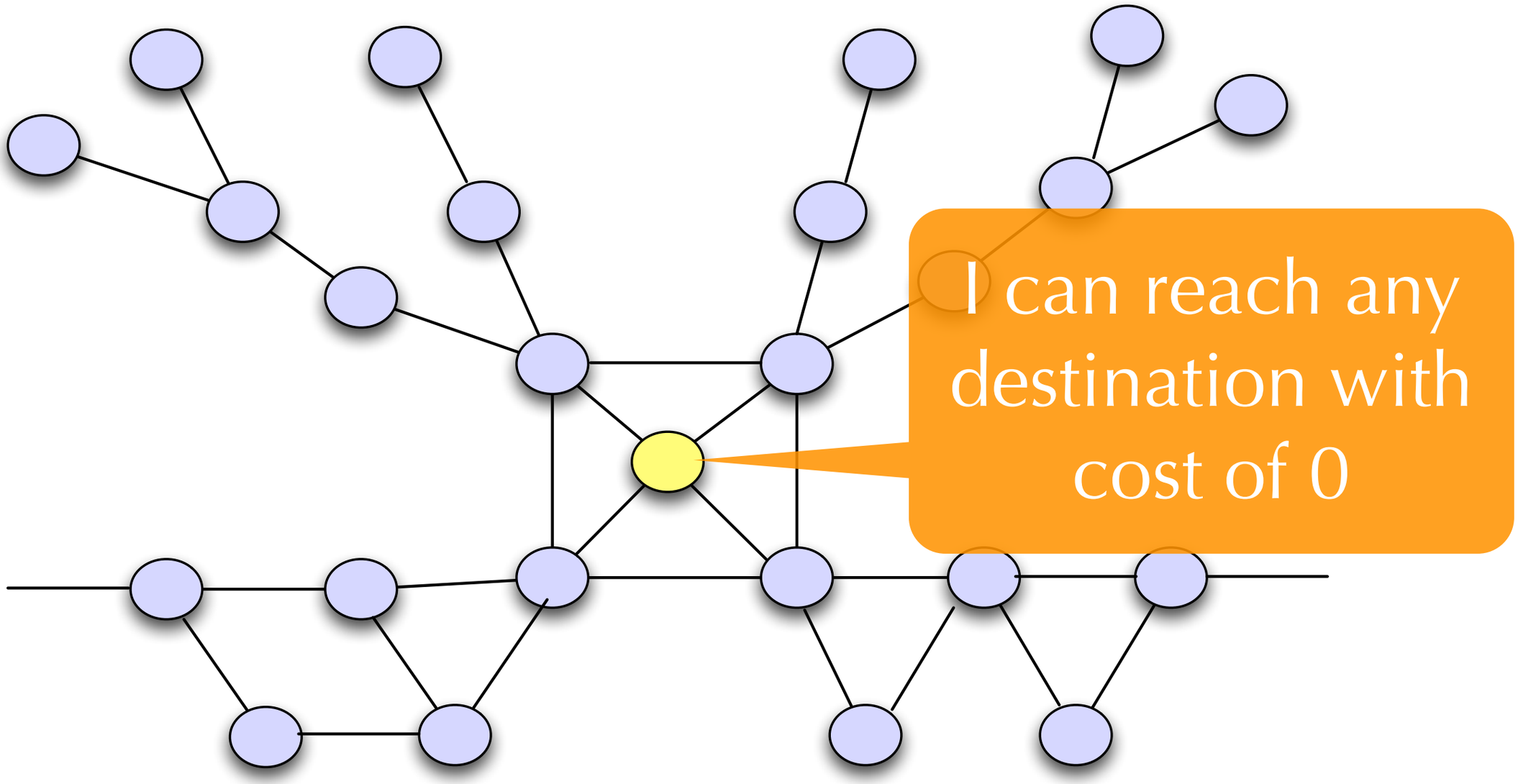
2nd BEST Recovery Algorithm



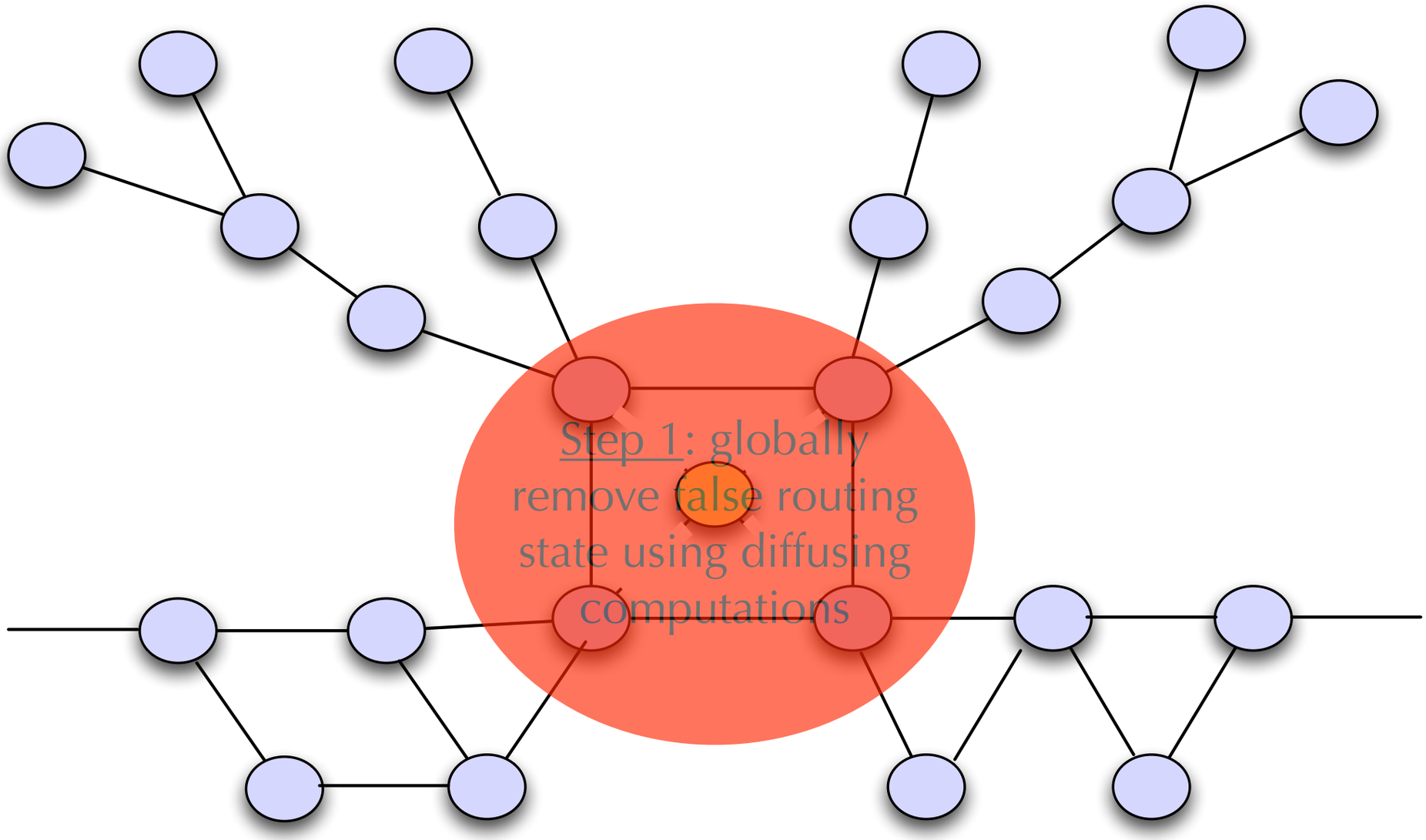
(+) simple, (–) risk of routing loops

1. neighbors of compromised node locally remove false routing state
2. start distance vector computation to compute new least cost paths routing around compromised node

PURGE Recovery Algorithm



PURGE Recovery Algorithm

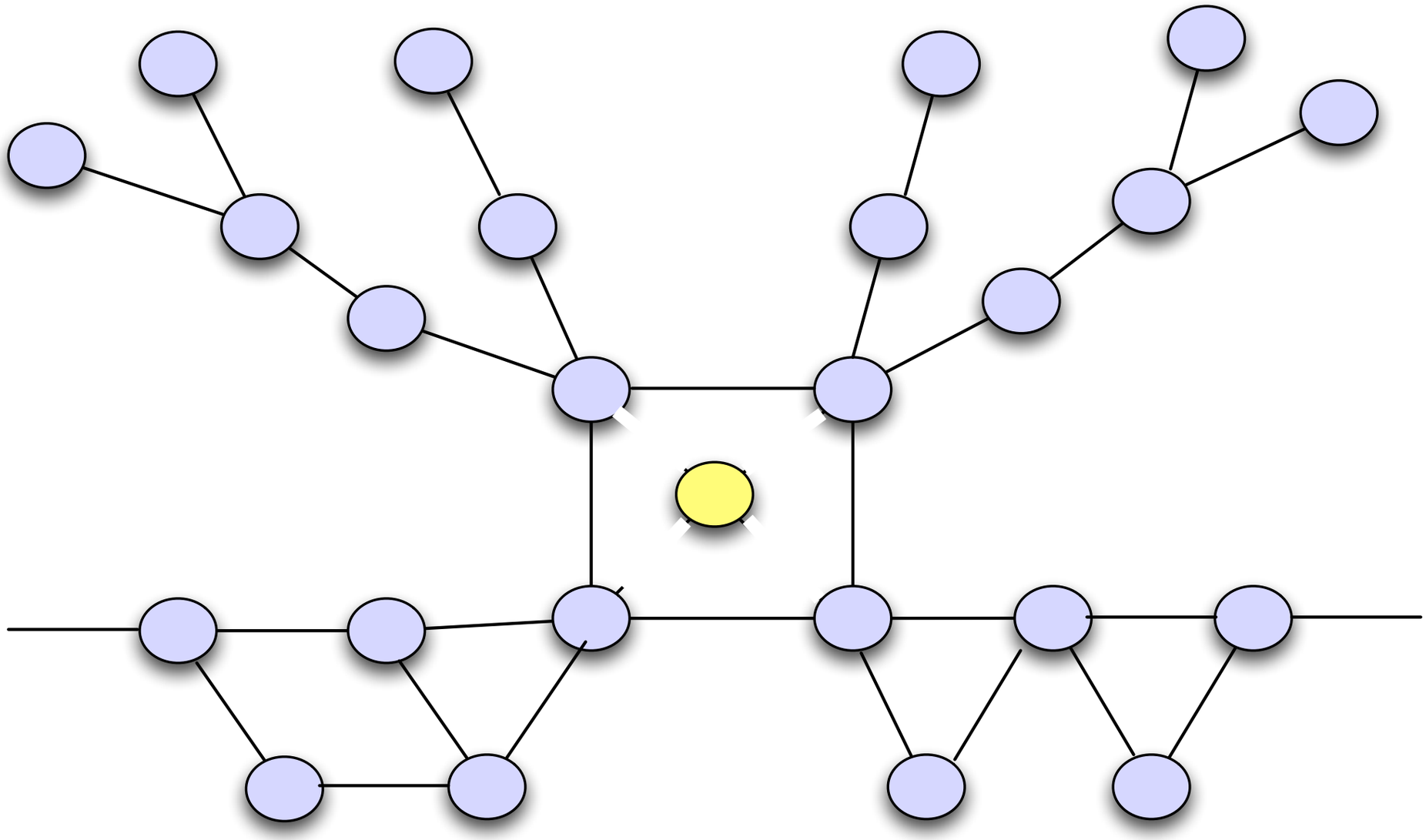


1. globally remove false routing state using diffusing computations

PURGE Recovery Algorithm

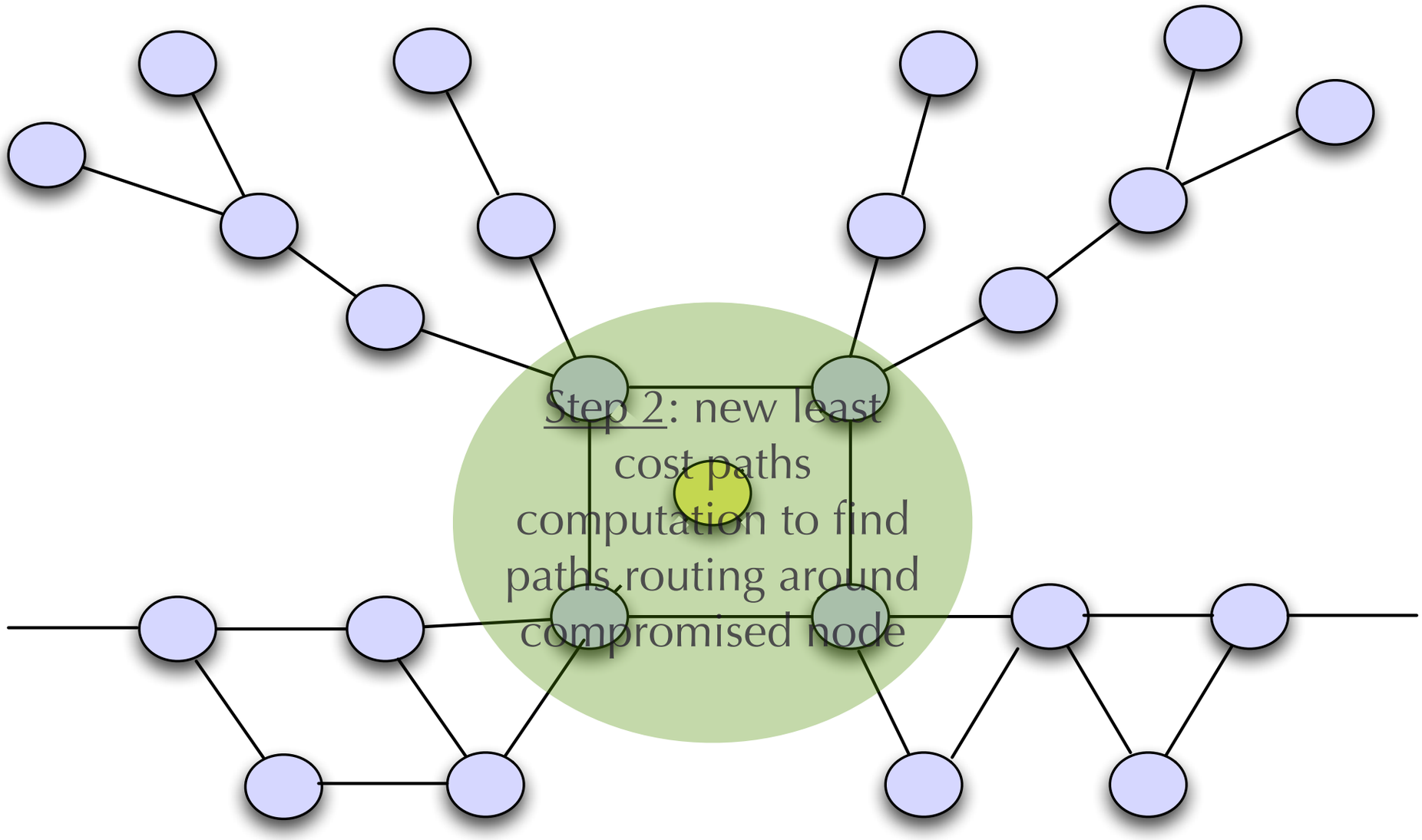
1. globally remove false routing state using diffusing computations

PURGE Recovery Algorithm



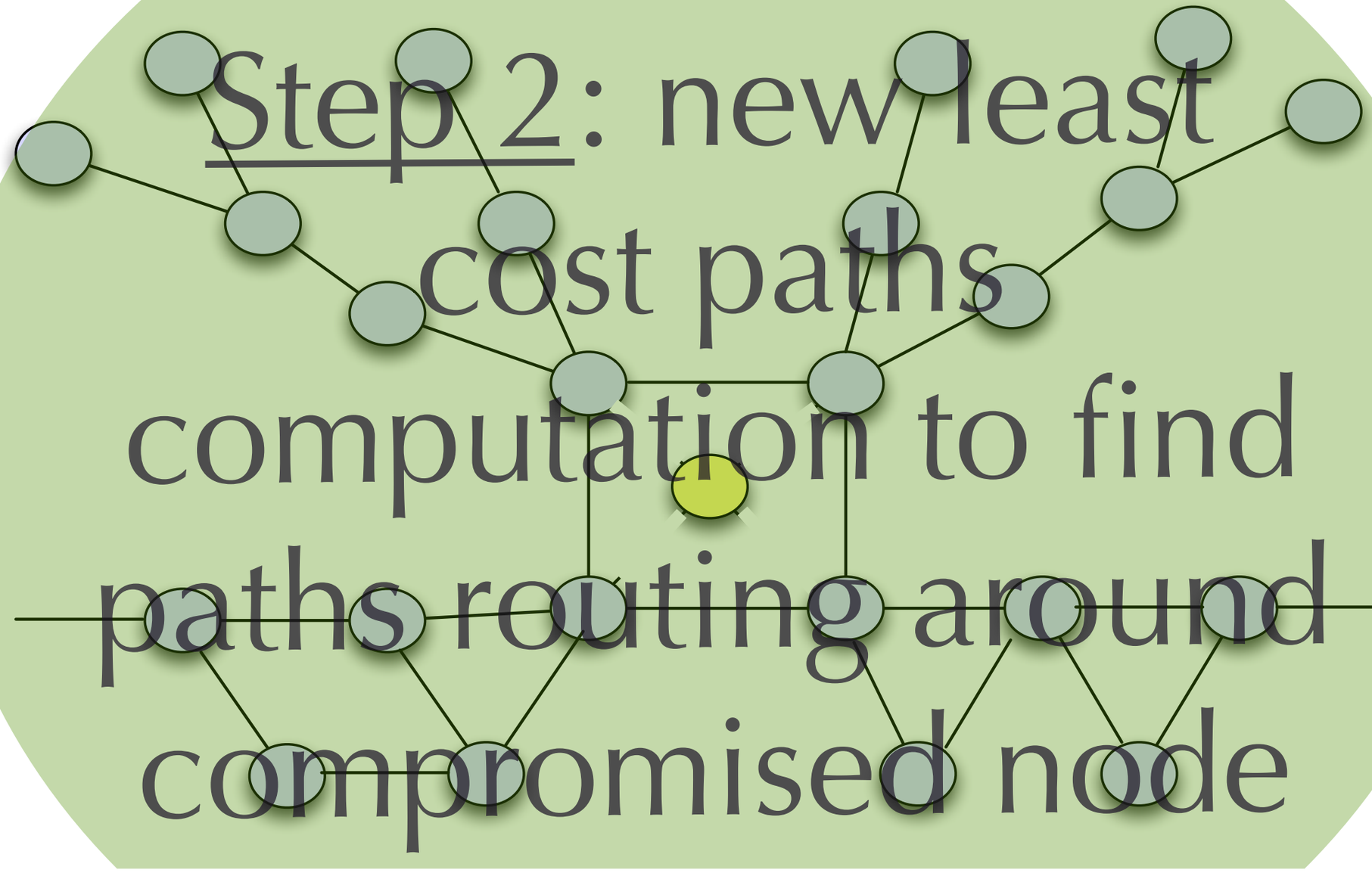
1. globally remove false routing state using diffusing computations
2. start distance vector computation to compute new least cost paths routing around compromised node

PURGE Recovery Algorithm



1. globally remove false routing state using diffusing computations
2. start distance vector computation to compute new least cost paths routing around compromised node

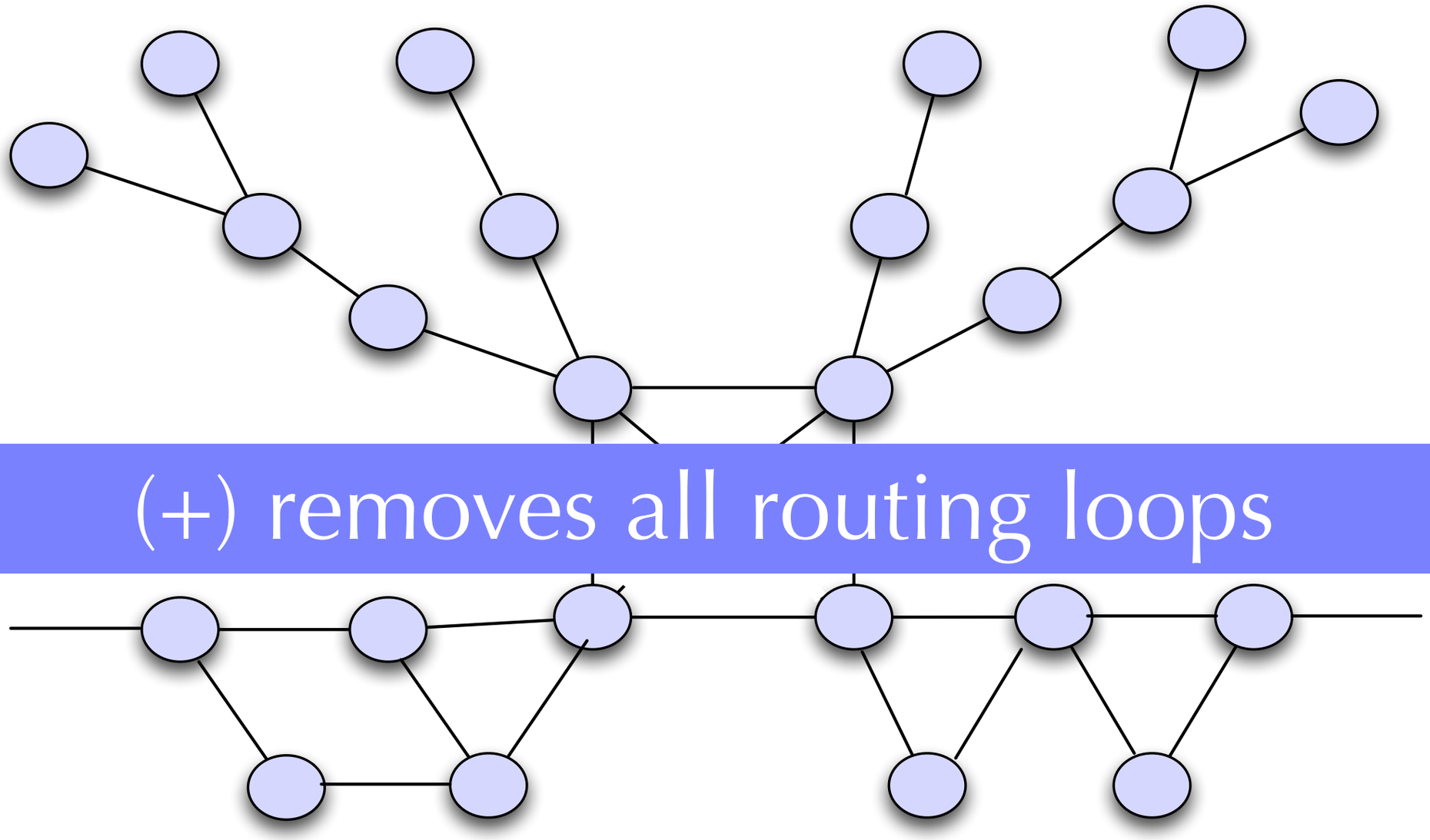
PURGE Recovery Algorithm



Step 2: new least cost paths computation to find paths routing around compromised node

1. globally remove false routing state using diffusing computations
2. start distance vector computation to compute new least cost paths routing around compromised node

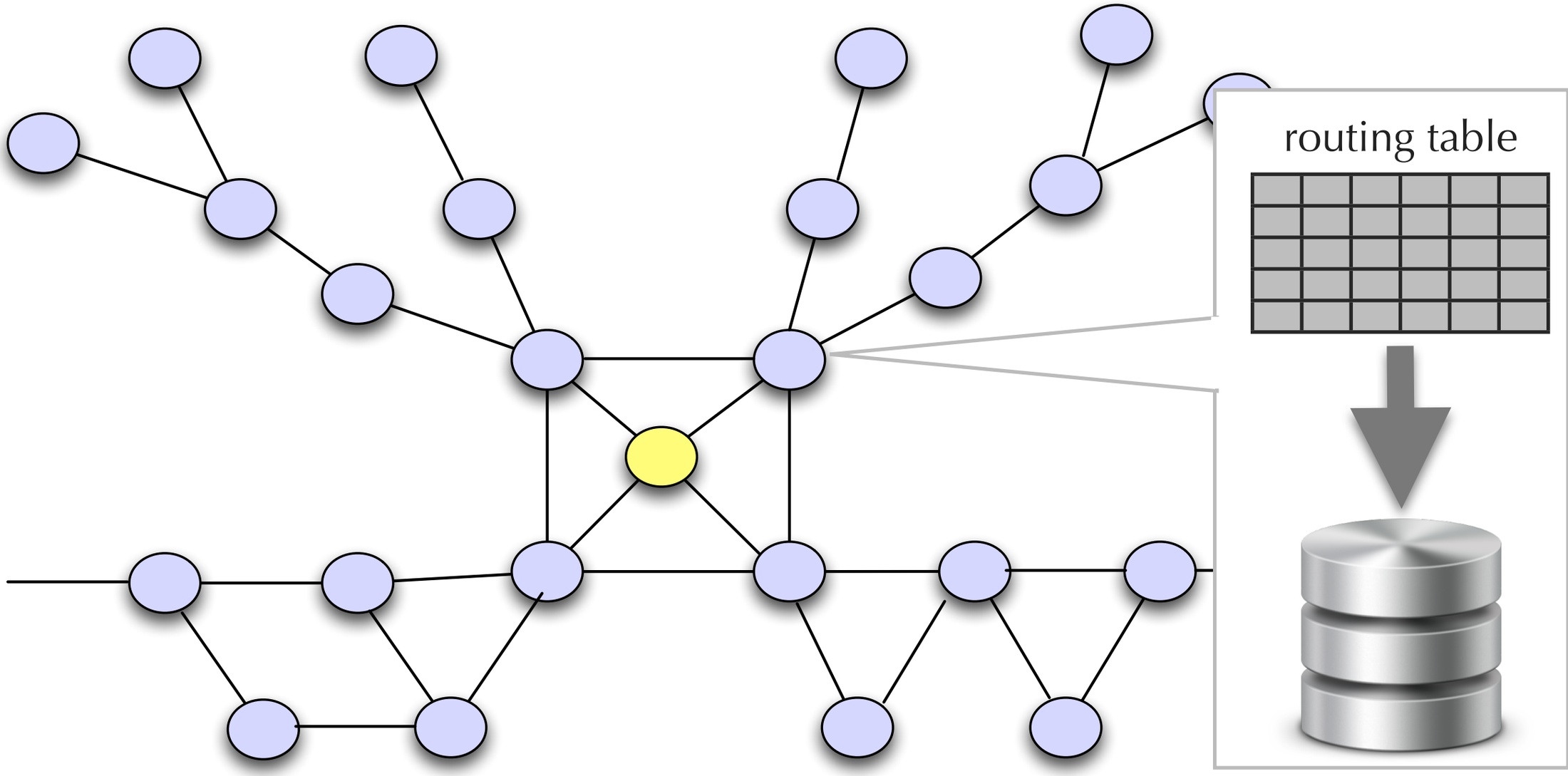
PURGE Recovery Algorithm



(+) removes all routing loops

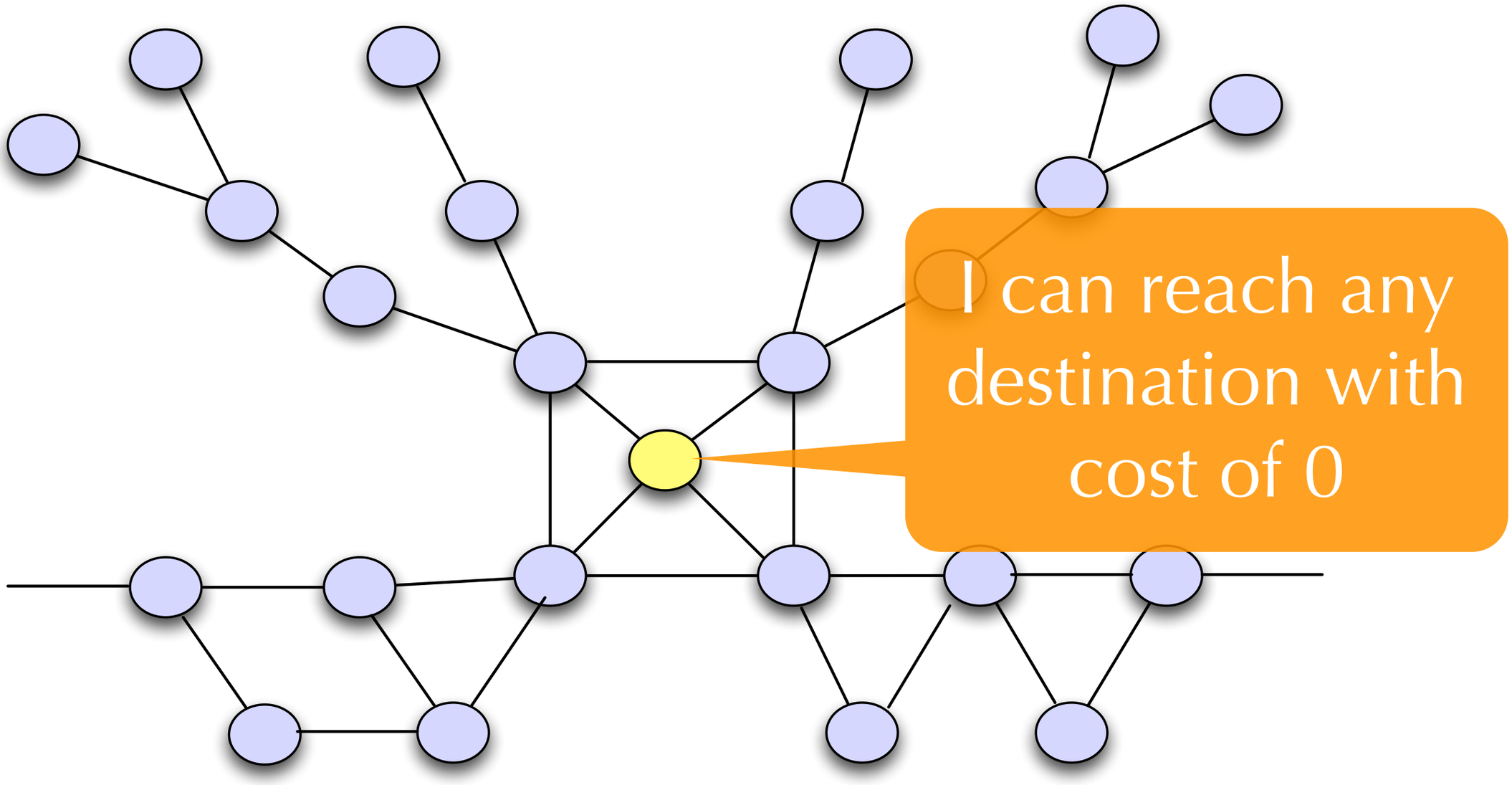
1. globally remove false routing state using diffusing computations
2. start distance vector computation to compute new least cost paths routing around compromised node

CPR Recovery Algorithm



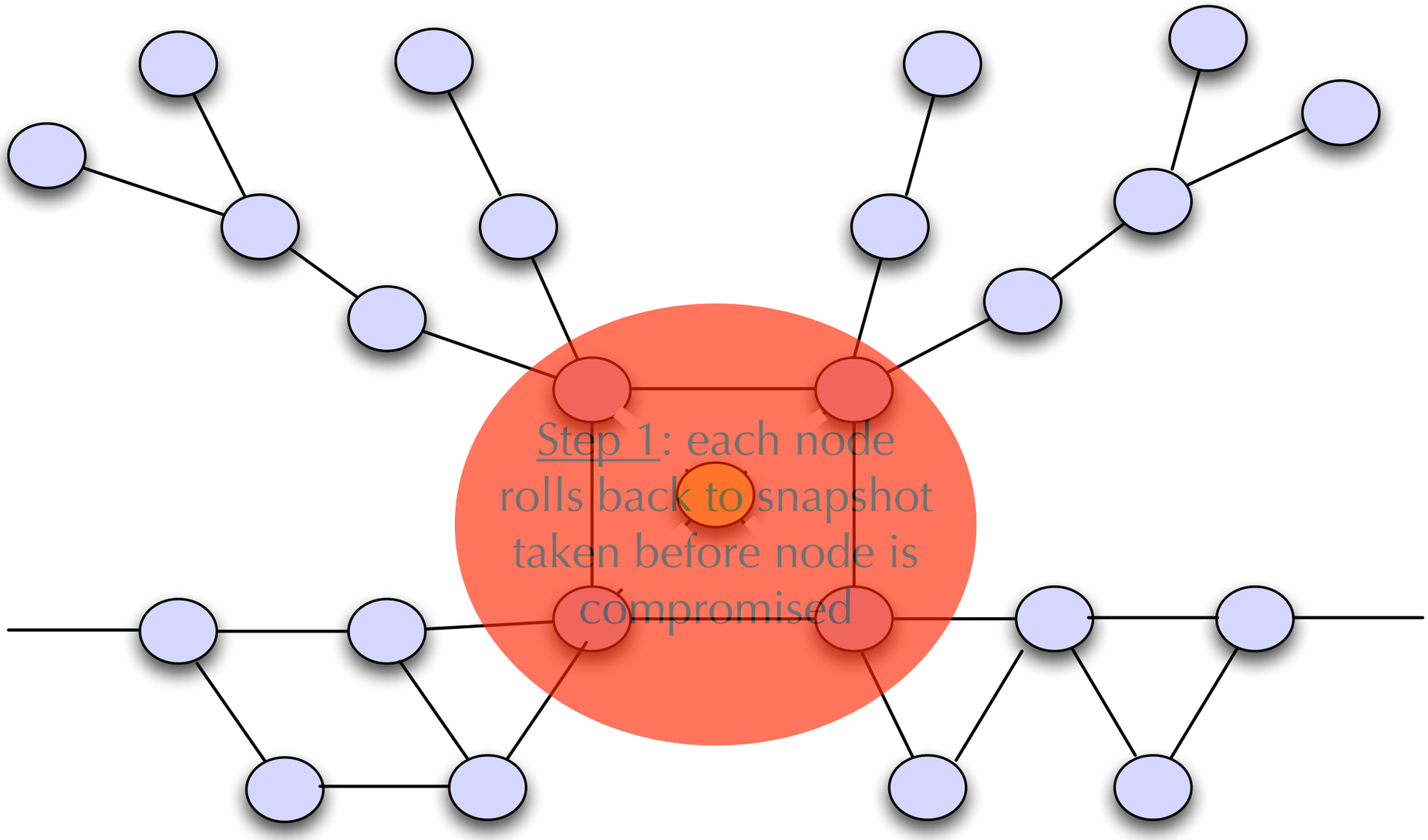
- each node takes snapshot of its routing table

CPR Recovery Algorithm



- each node takes snapshot of its routing table

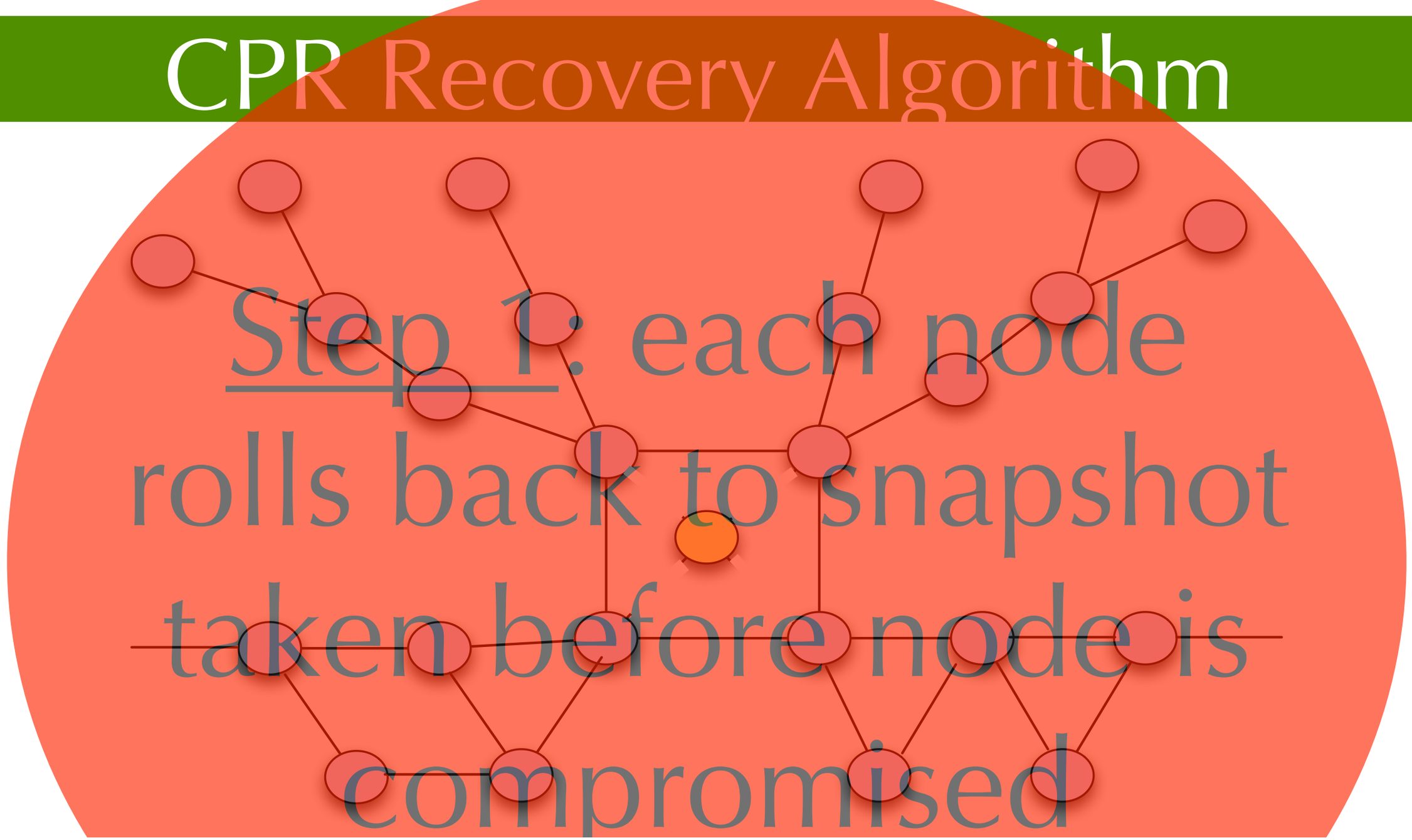
CPR Recovery Algorithm



- each node takes snapshot of its routing table

1. roll back to snapshot taken before node compromised

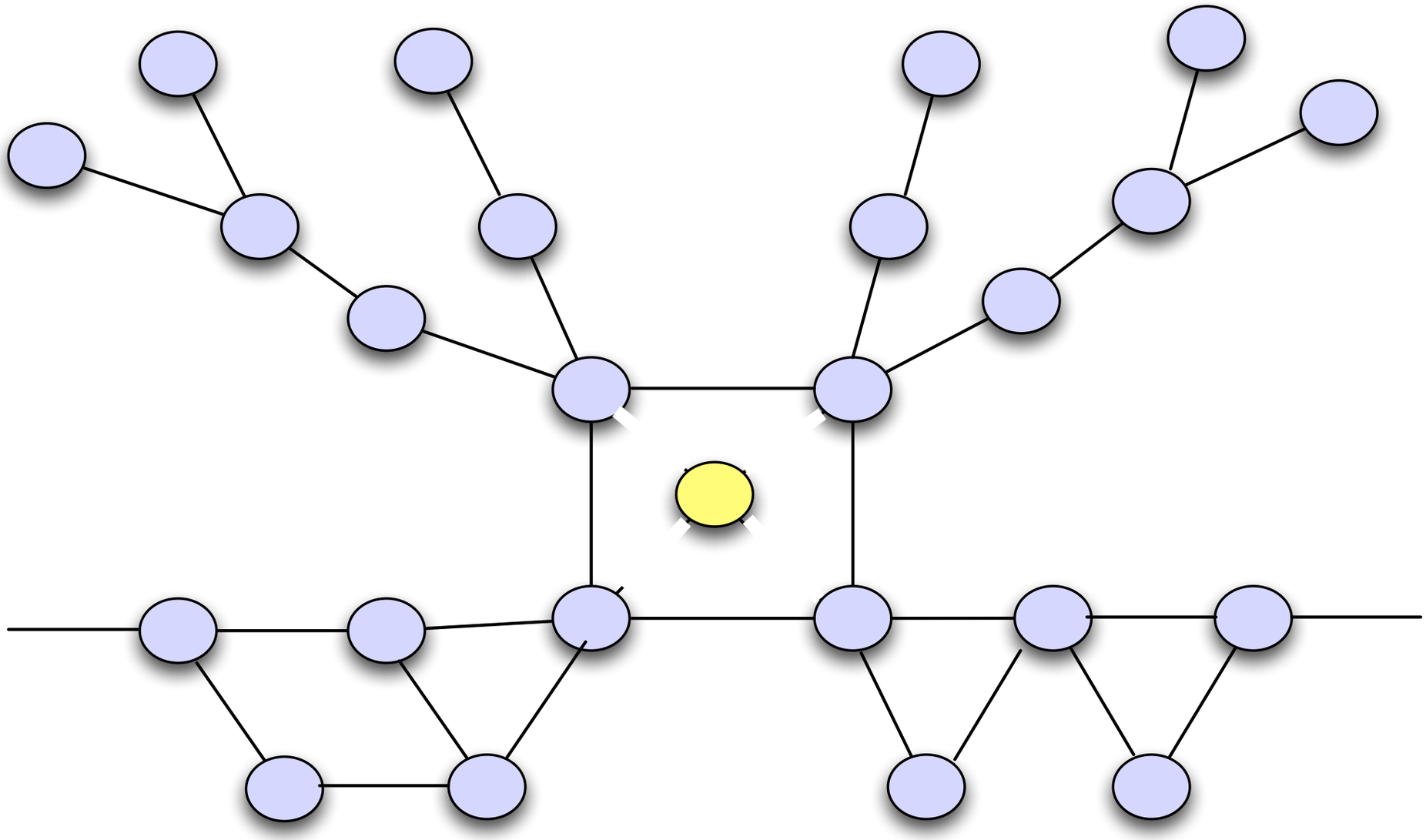
CPR Recovery Algorithm



- each node takes snapshot of its routing table

1. roll back to snapshot taken before node compromised

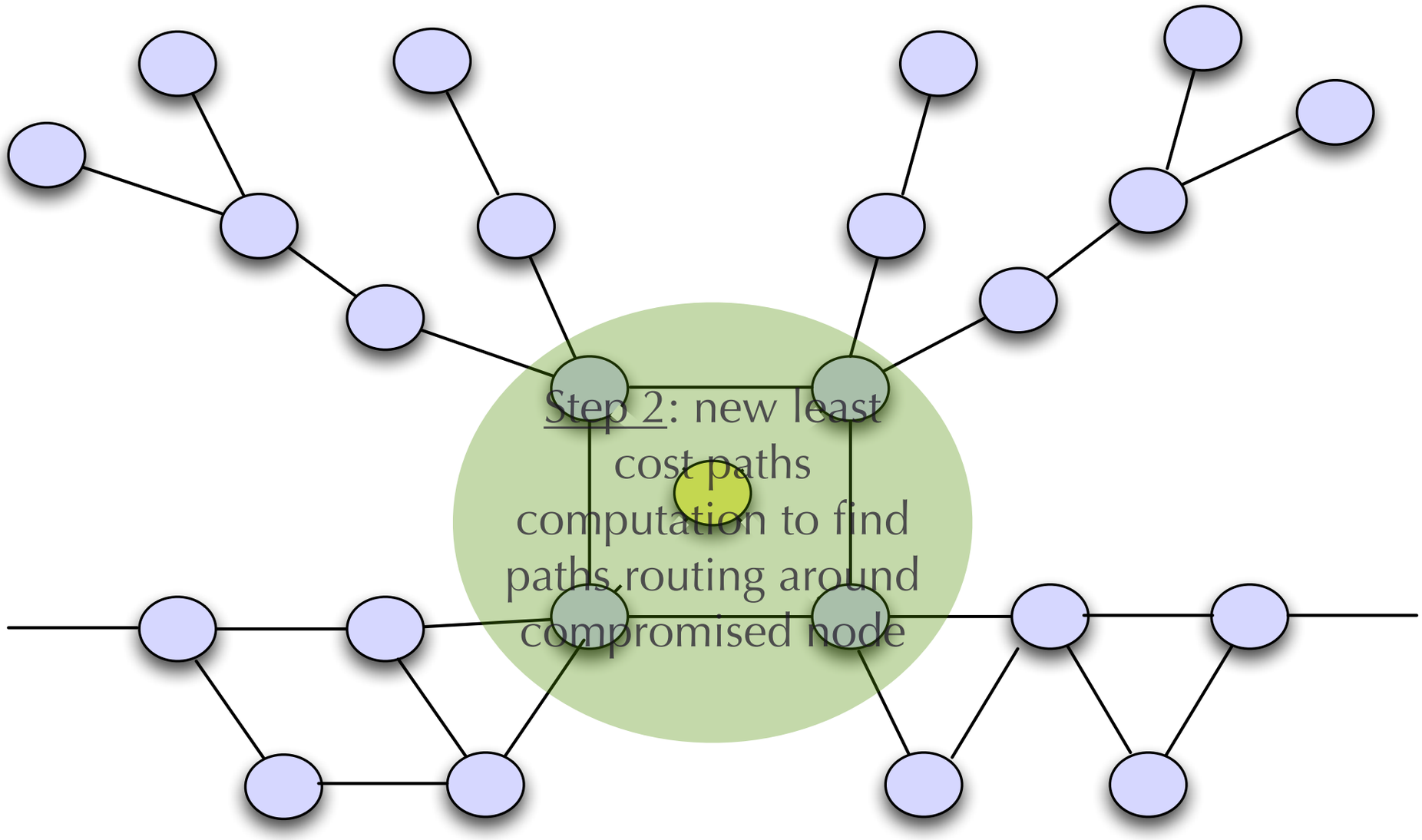
CPR Recovery Algorithm



- each node takes snapshot of its routing table

1. roll back to snapshot taken before node compromised
2. run DV to compute new least cost paths routing around compromised node + update any stale state

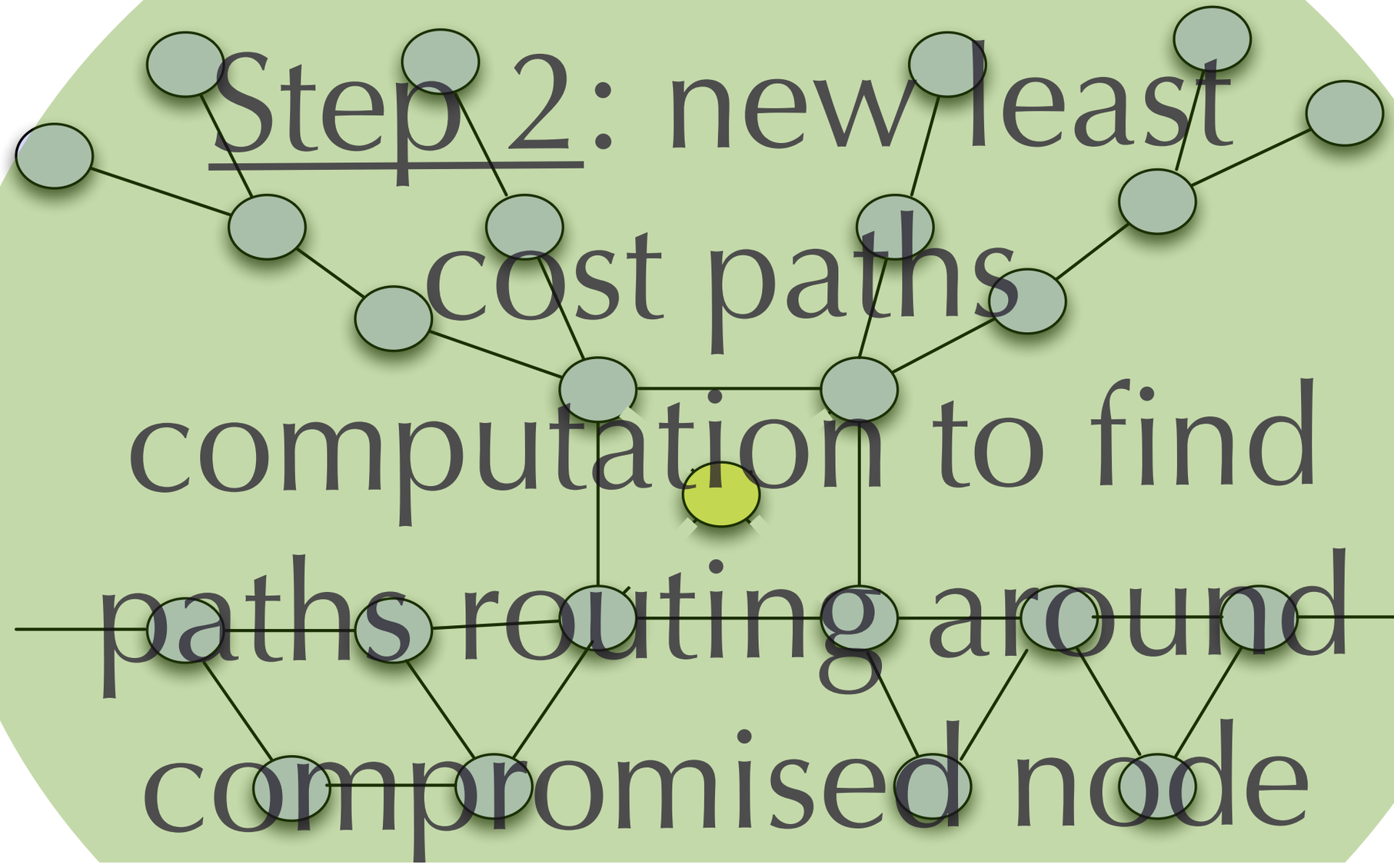
CPR Recovery Algorithm



- each node takes snapshot of its routing table

1. roll back to snapshot taken before node compromised
2. run DV to compute new least cost paths routing around compromised node + update any stale state

CPR Recovery Algorithm

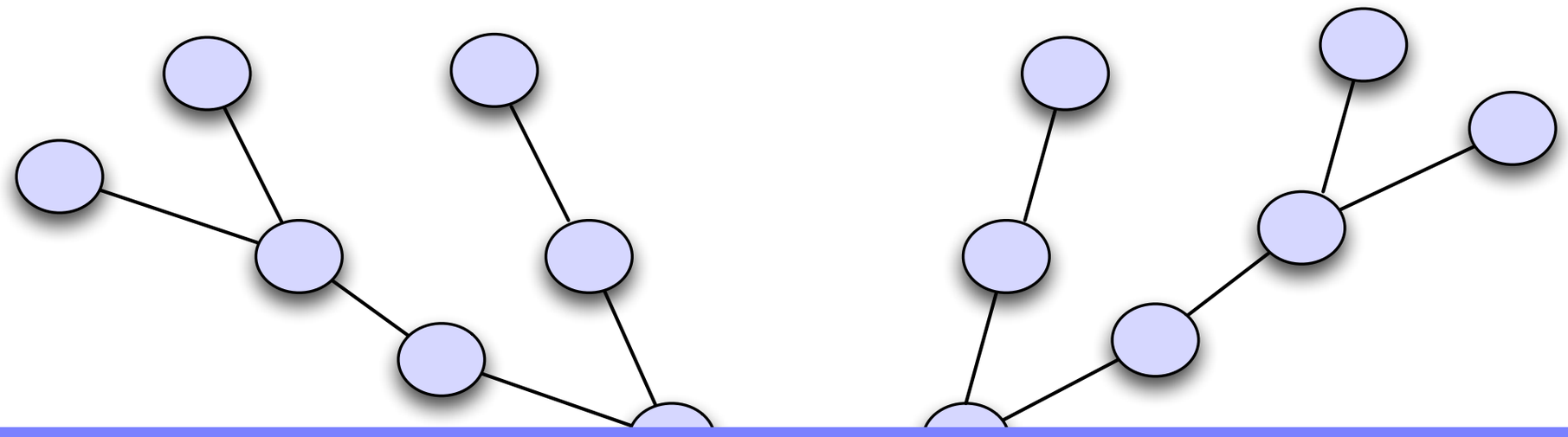


Step 2: new least cost paths computation to find paths routing around compromised node

- each node takes snapshot of its routing table

1. roll back to snapshot taken before node compromised
2. run DV to compute new least cost paths routing around compromised node + update any stale state

CPR Recovery Algorithm



(+) remove false state w/ single diffusing comp.,
(–) requires synchronized clocks,
(–) storage overhead

- each node takes snapshot of its routing table

1. roll back to snapshot taken before node compromised
2. run DV to compute new least cost paths routing around compromised node + update any stale state

Related Work

- PURGE is similar to Garcia-Luna-Aceves' DUAL algorithm for loop-free routing [26]
  - both use diffusing computations
- CPR borrows ideas from
  - database crash recovery Mohan et al. [39]
  - recovery from malicious but committed database transactions Ammann et al. [6], Liu et al. [35]

Evaluation

- what is the relative performance of each algorithm?
  - simulation study
    - use synchronous communication model
    - observe same trends w/ asynchronous model
  - complexity bounds in report

Simulation Scenario

Simulation Scenario

1. all nodes correctly compute least cost paths

Simulation Scenario

1. all nodes correctly compute least cost paths
2. a node is compromised and claims cost of “1” to every node

Simulation Scenario

1. all nodes correctly compute least cost paths
2. a node is compromised and claims cost of “1” to every node
3. nodes are notified of compromised node

Simulation Scenario

1. all nodes correctly compute least cost paths
2. a node is compromised and claims cost of “1” to every node
3. nodes are notified of compromised node
4. run recovery algorithm (messages counted here)

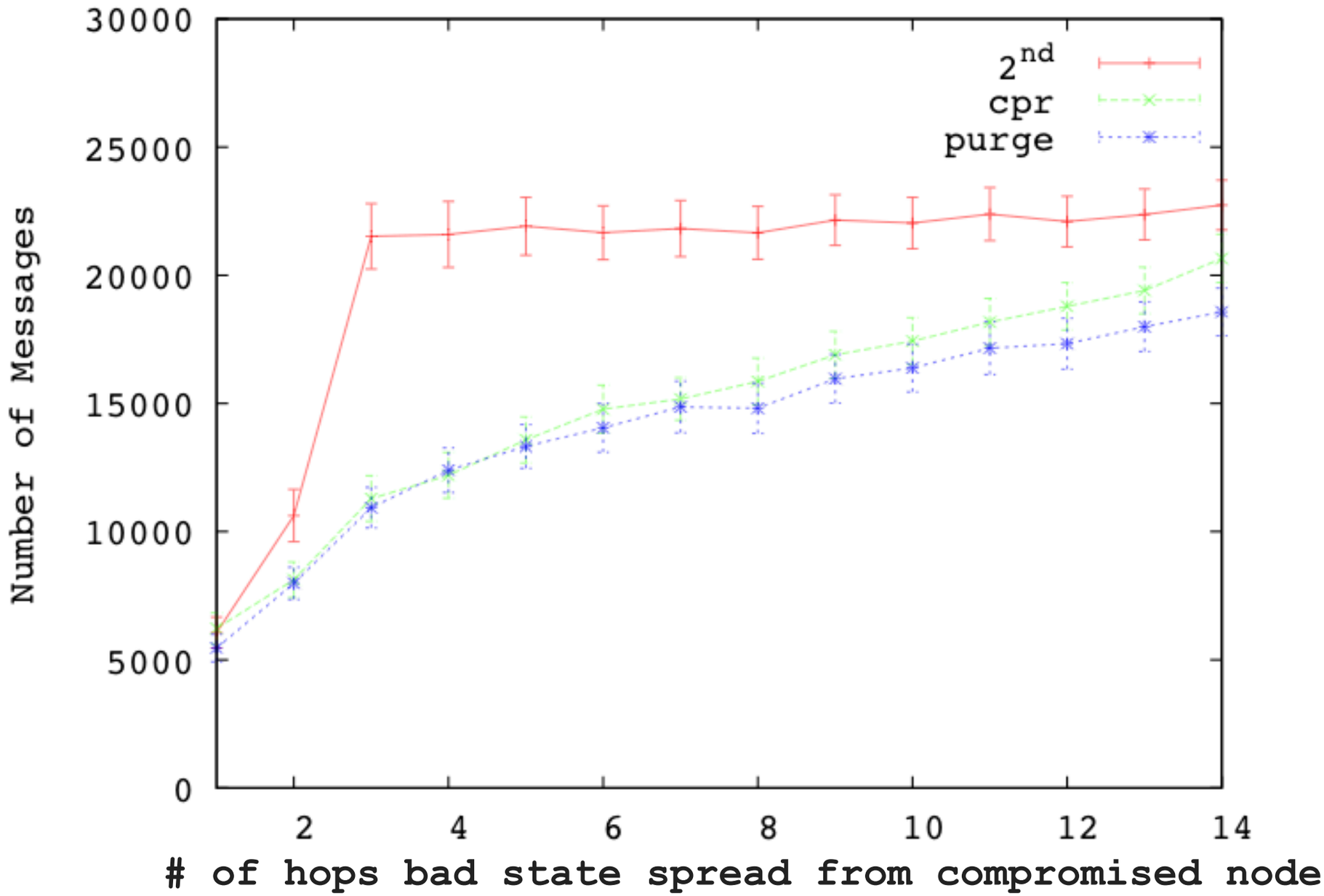
Simulation Scenario

1. all nodes correctly compute least cost paths
2. a node is compromised and claims cost of "1" to every node
3. nodes are notified of compromised node

4. run recovery algorithm (messages counted here)

- show results for Erdos-Renyi graphs
  - `n=100`, link weights between `[1,100]`, link weights can change after node compromised
  - same trends using Internet-like graphs (GT-ITM + Rocketfuel)

Message Overhead



- 2ND BEST: many routing loops
- CPR: stale state after rollback + assumes synch clocks
- PURGE: no routing loops + no stale state during recovery

Talk Outline

- thesis introduction
- placement of smart grid sensors to enable measurement error detection
- recovery from failed communication links in a smart grid
- recovery from malicious nodes injecting false routing state
- outline for future work and conclusions

Thesis Summary

- consider failure of network components
  - router spreading false routing state
  - smart grid sensor measurement error
  - link failures in smart grid communication network
- proposed algorithms for automated recovery

Contributions (1/3)

1. define 4 new problems for smart grid sensor placement that aim to max coverage + provide measurement error detection

- prove each is NP-Complete
- develop approximation algs + evaluate w/ simulations

Contributions (1/3)

1. define 4 new problems for smart grid sensor placement that aim to max coverage + provide measurement error detection

- prove each is NP-Complete
- develop approximation algs + evaluate w/ simulations

Gyllstrom, Rosensweig and Kurose, *e-Energy 2012*

Contributions (2/3)

2. define problem of recovery from link failure in smart grid communication network
   - outline OpenFlow-based link failure detection algorithm
   - outline 3 algs to compute backup multicast trees

Contributions (3/3)

3. design + develop 3 new algorithms for recovery from injection of false state in distributed sys.
   - evaluate w/ simulations + derive complexity bounds

Contributions (3/3)

3. design + develop 3 new algorithms for recovery from injection of false state in distributed sys.

- evaluate w/ simulations + derive complexity bounds

Gyllstrom, Vasudevan and Kurose, IFIP Networking 2010

Ch 2: Future Work Milestones

Ch 2: Future Work Milestones

- link failure detection

Ch 2: Future Work Milestones

- link failure detection
  - implement using Frenetic OF controller

Ch 2: Future Work Milestones

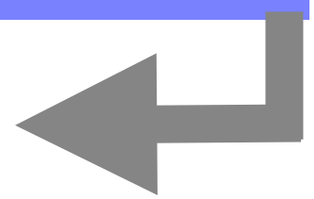
- link failure detection
  - implement using Frenetic OF controller
  - test + profile using Mininet

Ch 2: Future Work Milestones

- link failure detection

5 weeks - March

  - implement using Frenetic OF controller



  - test + profile using Mininet

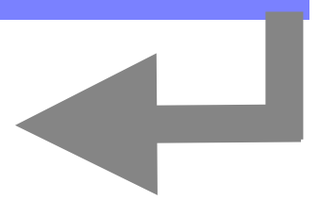
4 weeks - April

Ch 2: Future Work Milestones

- link failure detection

5 weeks - March

  - implement using Frenetic OF controller



  - test + profile using Mininet

4 weeks - April

- backup tree computation algorithms

Ch 2: Future Work Milestones

- link failure detection
  - implement using Frenetic OF controller
  - test + profile using Mininet
- backup tree computation algorithms
  - specify each algorithm

5 weeks - March

4 weeks - April

Ch 2: Future Work Milestones

- link failure detection

  5 weeks - March

  - implement using Frenetic OF controller

  - test + profile using Mininet

  4 weeks - April

- backup tree computation algorithms
  - specify each algorithm
  - implement subset (?) of the 3 algorithms using POX Openflow controller

Ch 2: Future Work Milestones

- link failure detection

  5 weeks - March

  - implement using Frenetic OF controller

  - test + profile using Mininet

  4 weeks - April

- backup tree computation algorithms
  - specify each algorithm
  - implement subset (?) of the 3 algorithms using POX Openflow controller
  - complexity analysis

Ch 2: Future Work Milestones

- link failure detection

  5 weeks - March

  - implement using Frenetic OF controller

  - test + profile using Mininet

  4 weeks - April

- backup tree computation algorithms
  - specify each algorithm
  - implement subset (?) of the 3 algorithms using POX Openflow controller
  - complexity analysis
  - simulation-based evaluation using Mininet

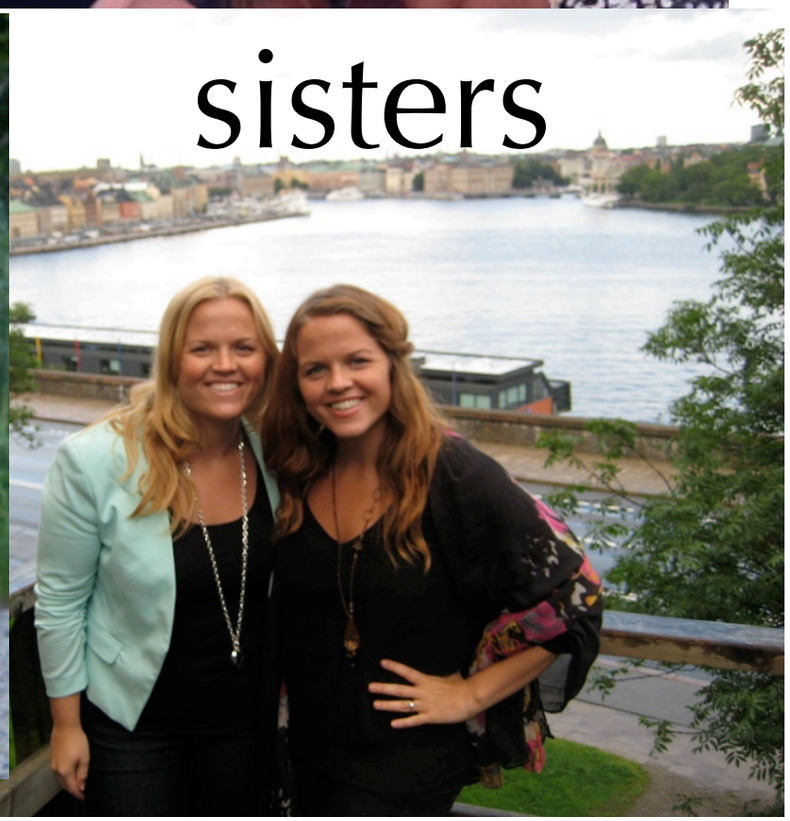
Ch 2: Future Work Milestones

- link failure detection — 5 weeks - March
  - implement using Frenetic OF controller
  - test + profile using Mininet — 4 weeks - April
- backup tree computation algorithms
  - specify each algorithm — 4 weeks - May
  - implement subset (?) of the 3 algorithms using POX Openflow controller — 4 weeks - June
  - complexity analysis — 8 weeks - May
  - simulation-based evaluation using Mininet

End of Technical Content

Thank You Family

Thank You Family



Quick Story

"Computer networking is boring, I'll never work in that field"

-- Dan Gyllstrom, circa 2002

Quick Story

"I'm not sure if this PhD program is for me, I have no advisor and no research prospects. I'll sit-in on this graduate networking course"

-- Dan Gyllstrom, early Fall 2007

Quick Story

"This is the best class I have ever taken, I need to work with Jim Kurose "

-- Dan Gyllstrom, mid-Fall 2007

Thank You Jim

Picture courtesy of Pablo Serrano and the 1980s

Thank You Committee

Thank You Colleagues + Friends

- NETWORKS Lab
- GEF: Marwan Mattar, Karl Gyllstrom
- Donald Knuth for the fruitful discussions

Finally, thank you all for coming.

Questions?

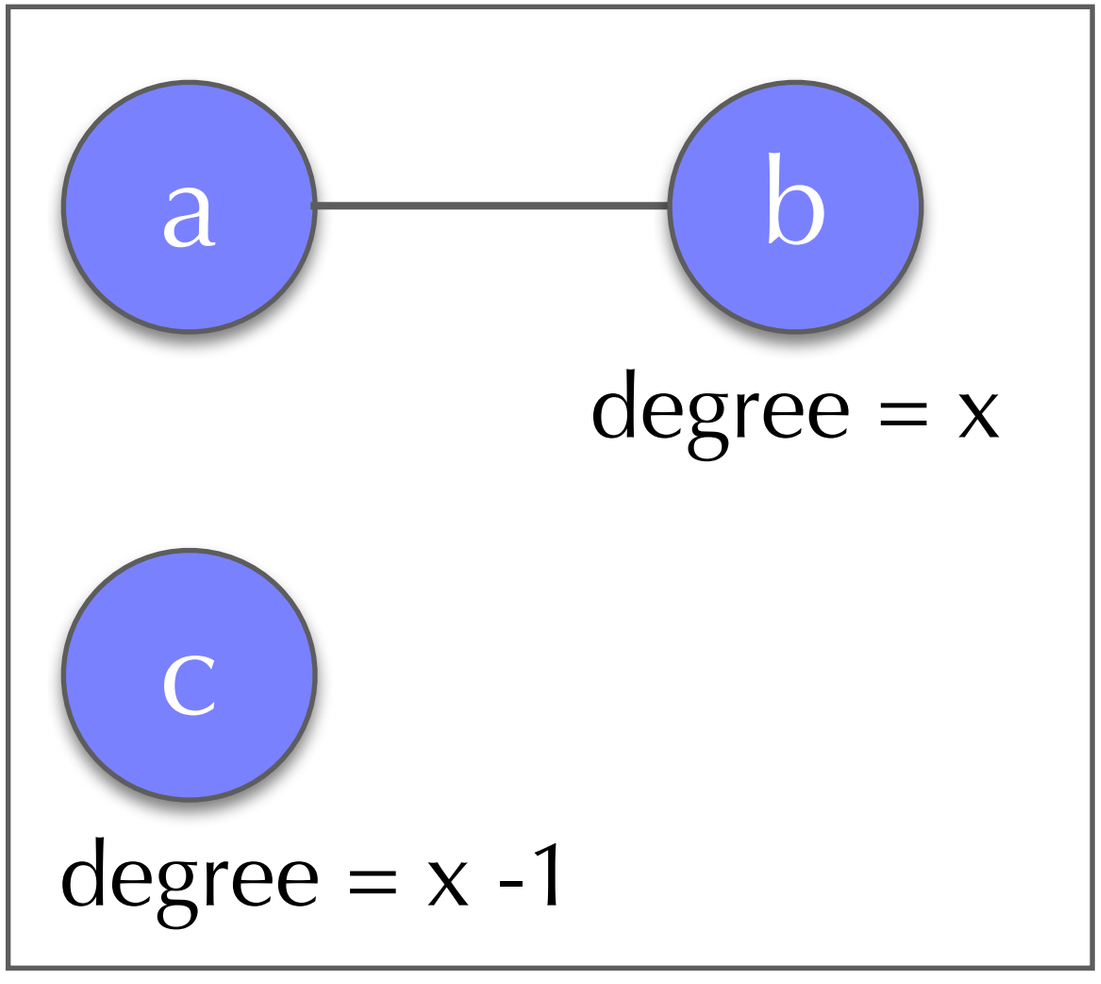
Comments?

Backup Slides

Ch 1: Synthetic Graph Generation

1. start with IEEE graph
2. swap edges until new graph shares no edges with original

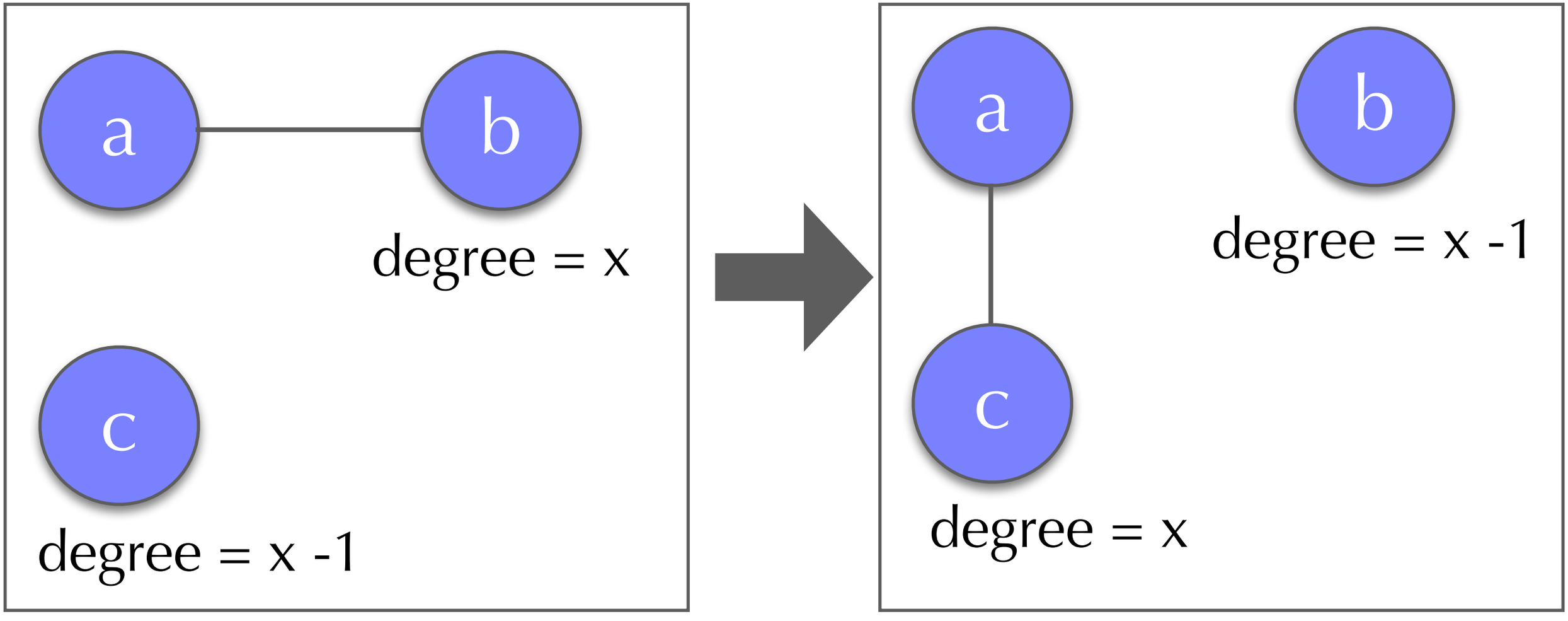
Output = same # number of nodes but new connectivity



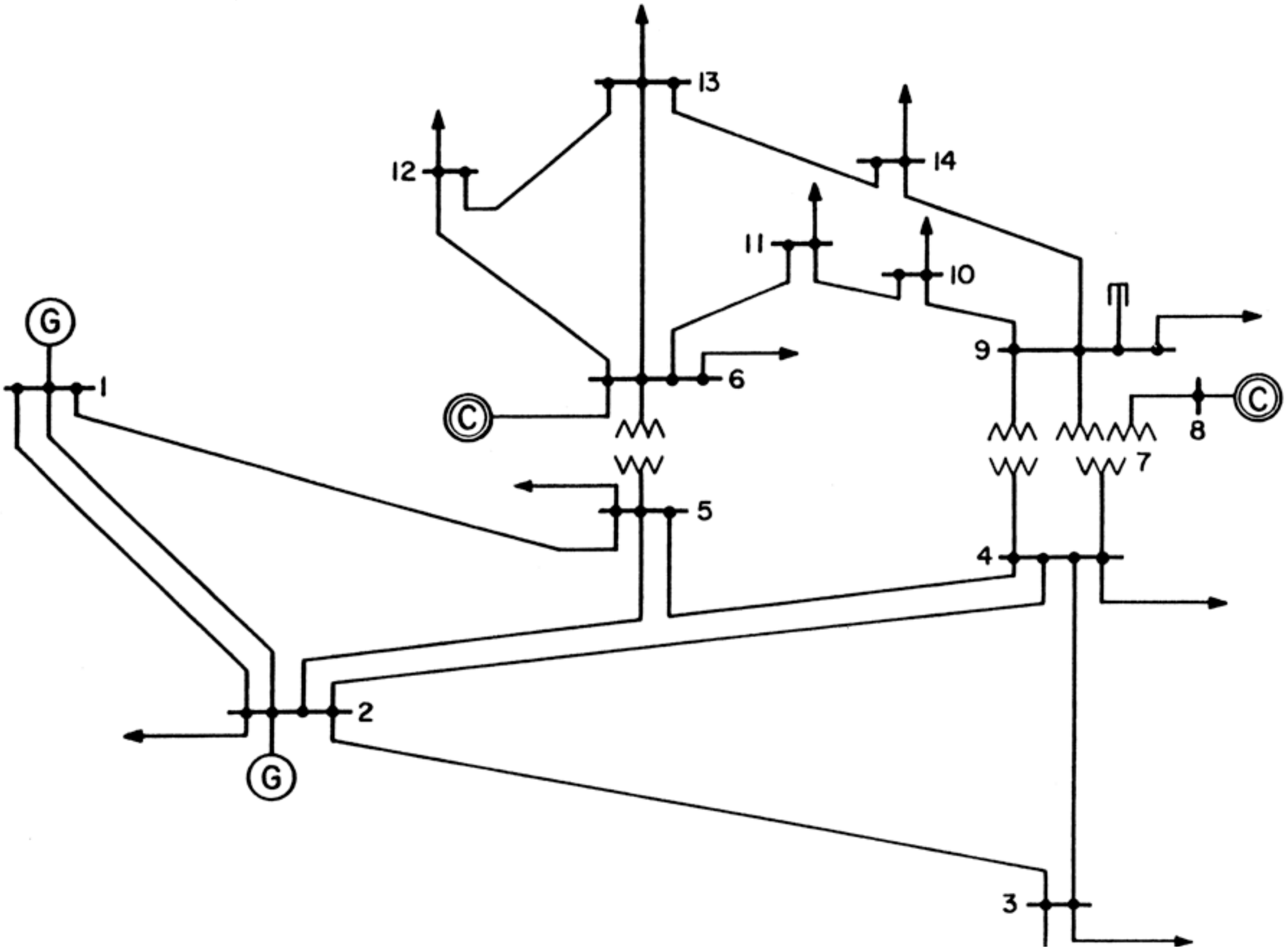
Ch 1: Synthetic Graph Generation

1. start with IEEE graph
2. swap edges until new graph shares no edges with original

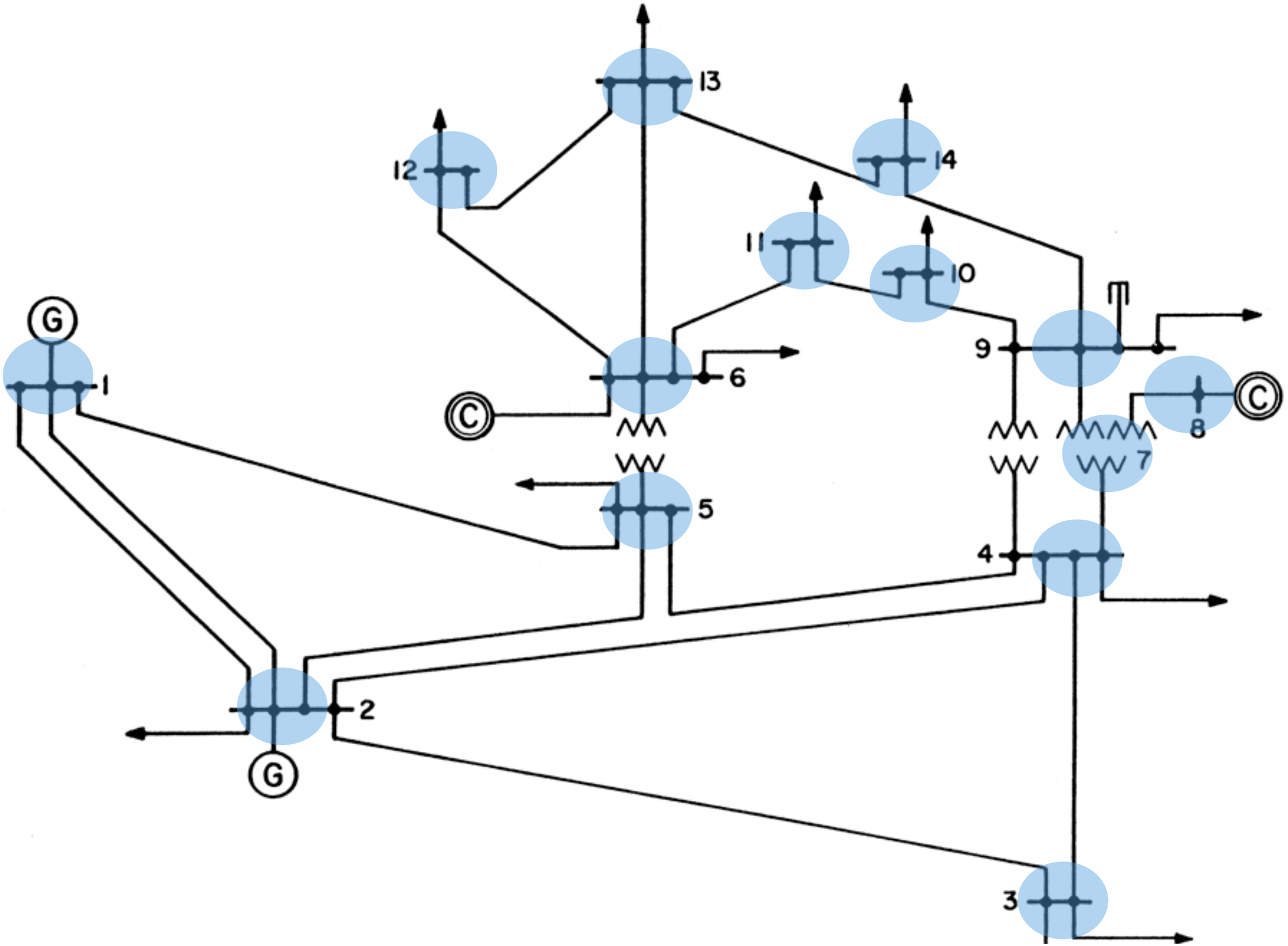
Output = same # number of nodes but new connectivity



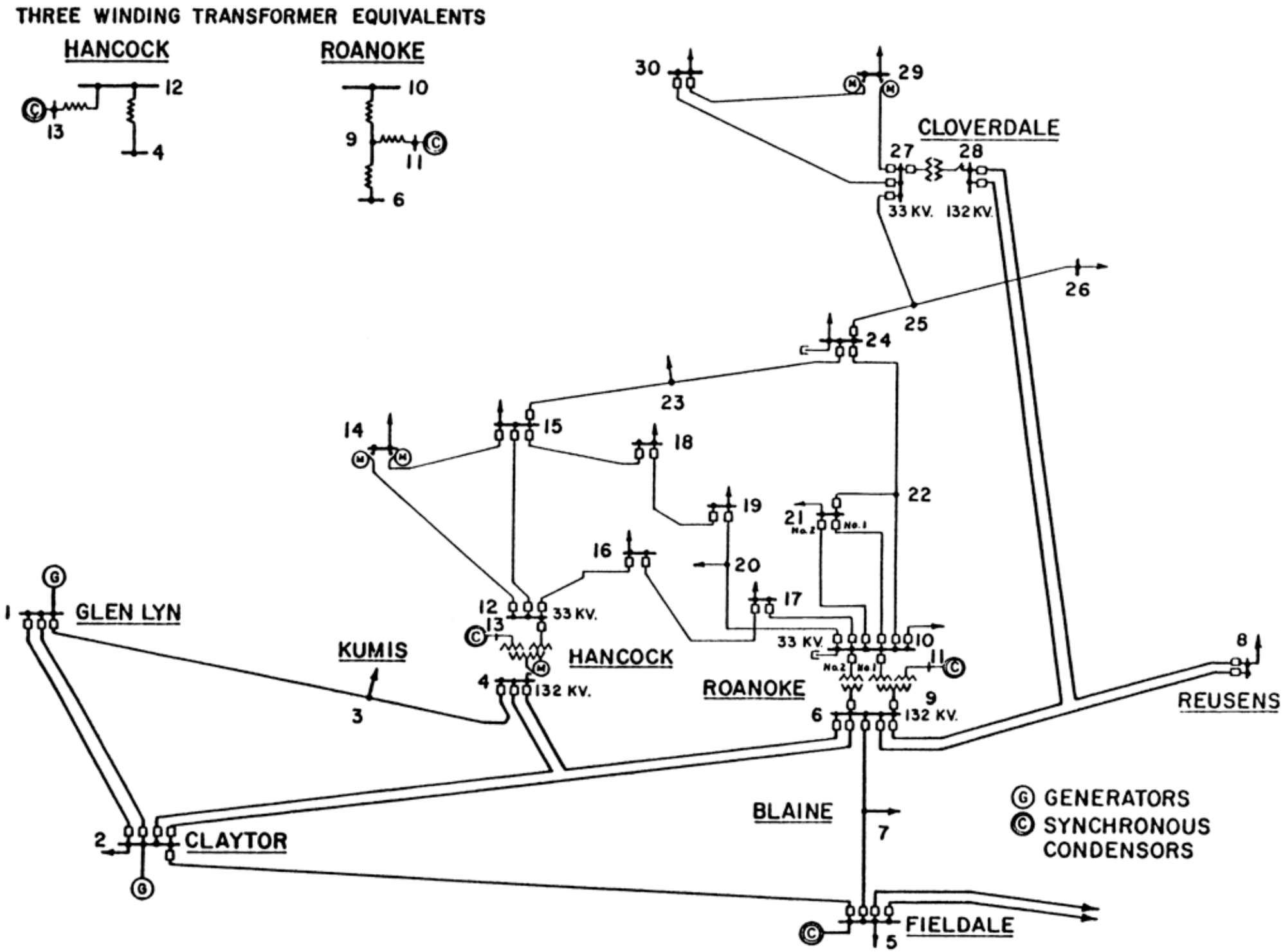
Ch 1: IEEE Bus System 14



Ch 1: IEEE Bus System 14



Ch 1: IEEE Bus System 30



Ch 1: Power Grid Terminology

- Node: system bus
  - injection node: power generation center or aggregation of loads
  - zero-injection node: substation where electrical lines meet (maintains conservation of flow)
- Observed: if voltage phasor of node can be directly measured or calculated

Ch 1: Zero-Injection Nodes

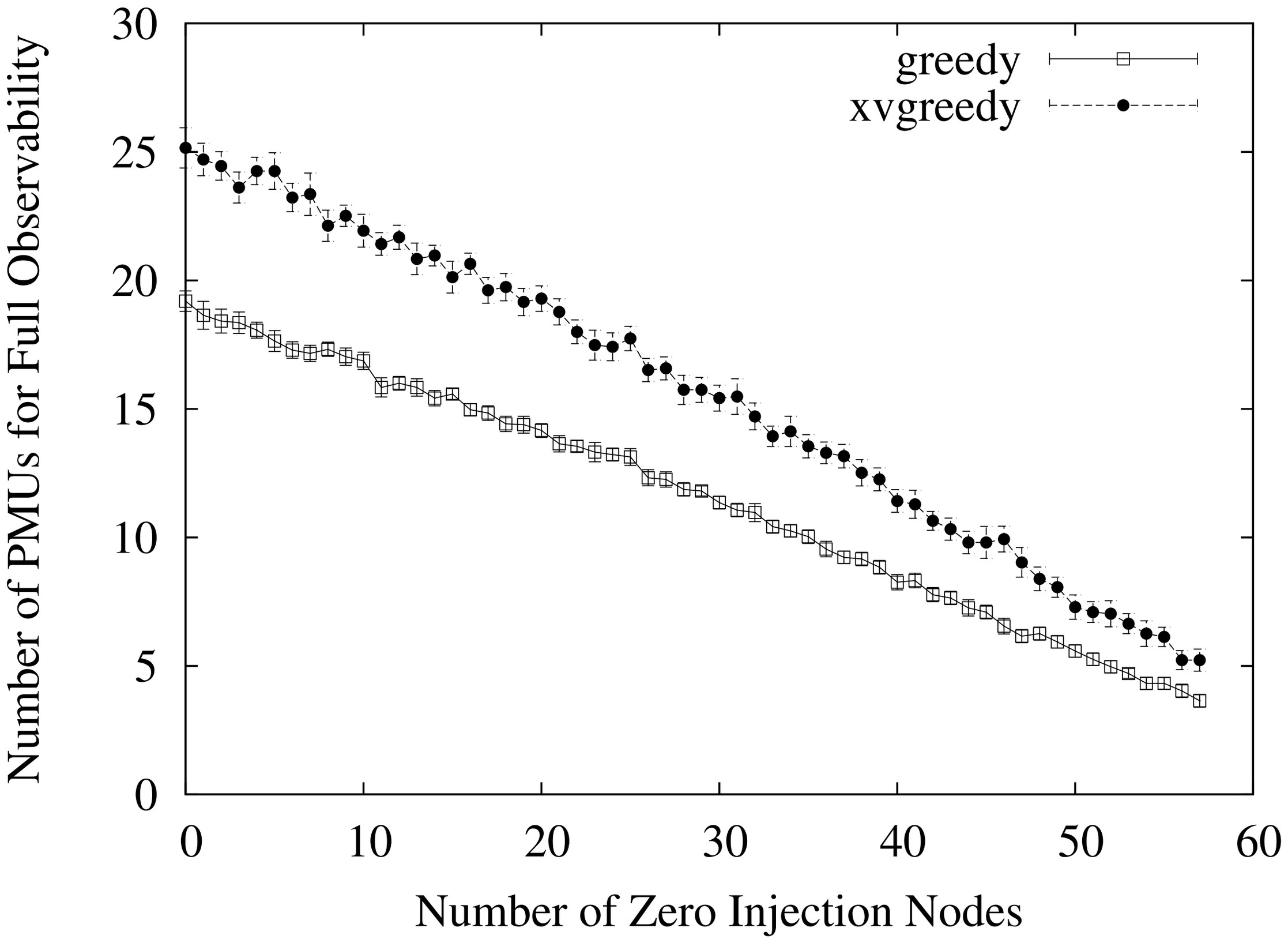
- Zero-Injection Node
  - substation where electrical lines meet
  - maintains conservation of flow
  - node must be zero-injection to apply Observability Rule 2

Ch 1: Zero-Injection Nodes

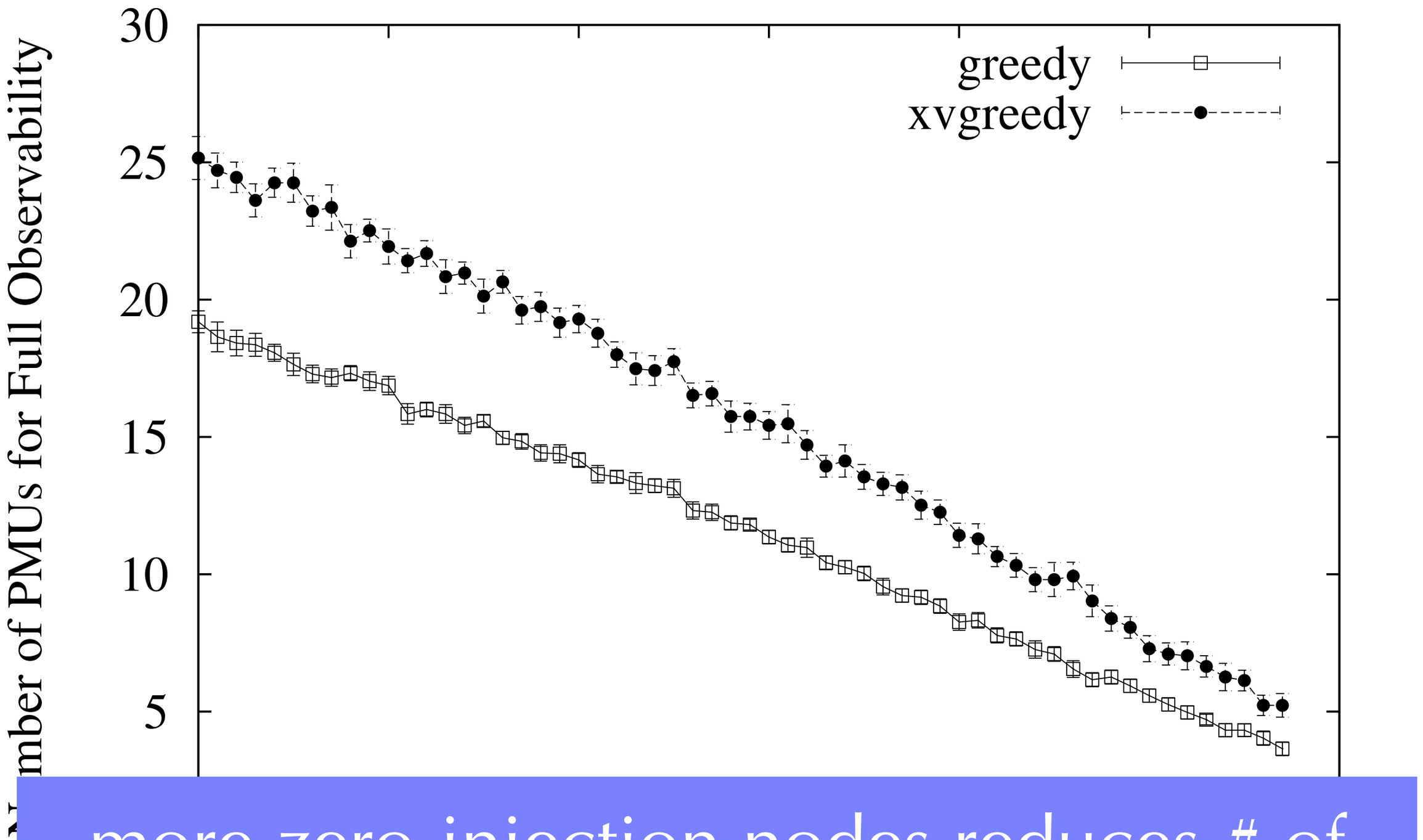
- Zero-Injection Node
  - substation where electrical lines meet
  - maintains conservation of flow
  - node must be zero-injection to apply Observability Rule 2

How does # of zero-injection nodes effect observability?

Ch 1: Vary # Zero-Injection Nodes



Ch 1: Vary # Zero-Injection Nodes



more zero-injection nodes reduces # of PMUs needed to observe all graph nodes

Ch 1: Why is XV Limited to 2 Hops?

- Computing voltage phasor of non-PMU nodes
  - equations have variables to account for measurement error
- More than 2 Hops
  - have more unknowns that equations

=> no error detection

Ch 2: More Related Work

- OpenFlow-based solution by Kotani et al. [33]
  - pre-install backup MT
  - packet headers carry `tree ID` to indicate which tree to use
    - after link failure, root writes `tree ID` of backup tree in each packet header
  - (+) promising approach to fast tree switching
  - (–) basic MT computation (Dijkstra's algorithm)

Ch 2: Packet Loss or Delay?

- goal: minimize packet loss and delay related to link failure
  - consider any packet not meeting its E2E delay requirement as lost
    - critical power grid applications: no tolerance for loss
  - link failure detection
    - OpenFlow no native timer support
    - use loss as indicator of delay

Ch 2: More on Backup MTs

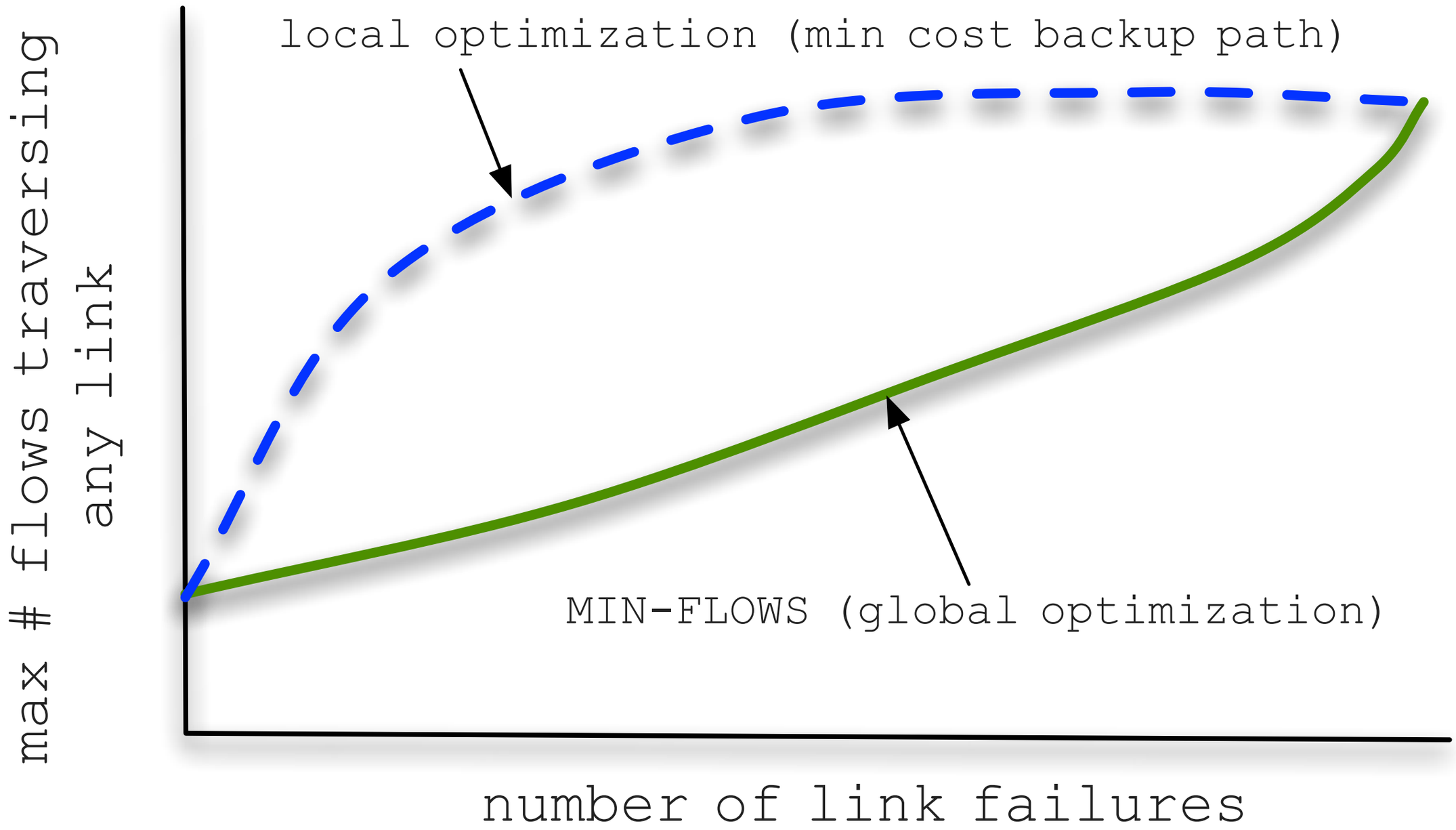
- how many?
  - for each MT: compute backup MT for each link
- when are backup MTs computed?
  - initialization
  - recompute *all* backup MTs after each link failure
    - state of network changes after each failure

Ch 2: Research Questions

Q3: How much more does global optimization increase robustness than doing localized opt.?

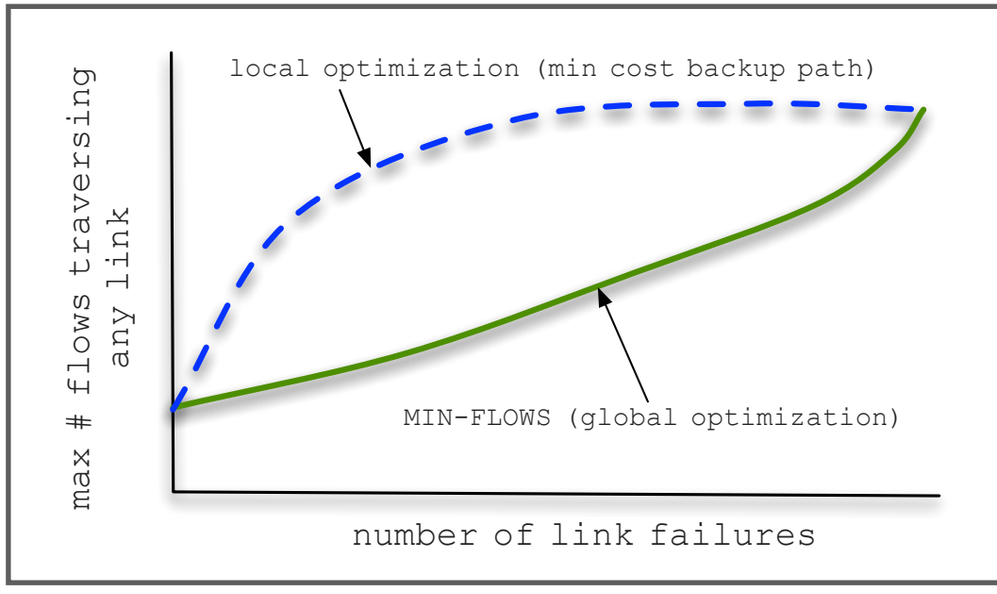
Ch 2: Research Questions

Q3: How much more does global optimization increase robustness than doing localized opt.?



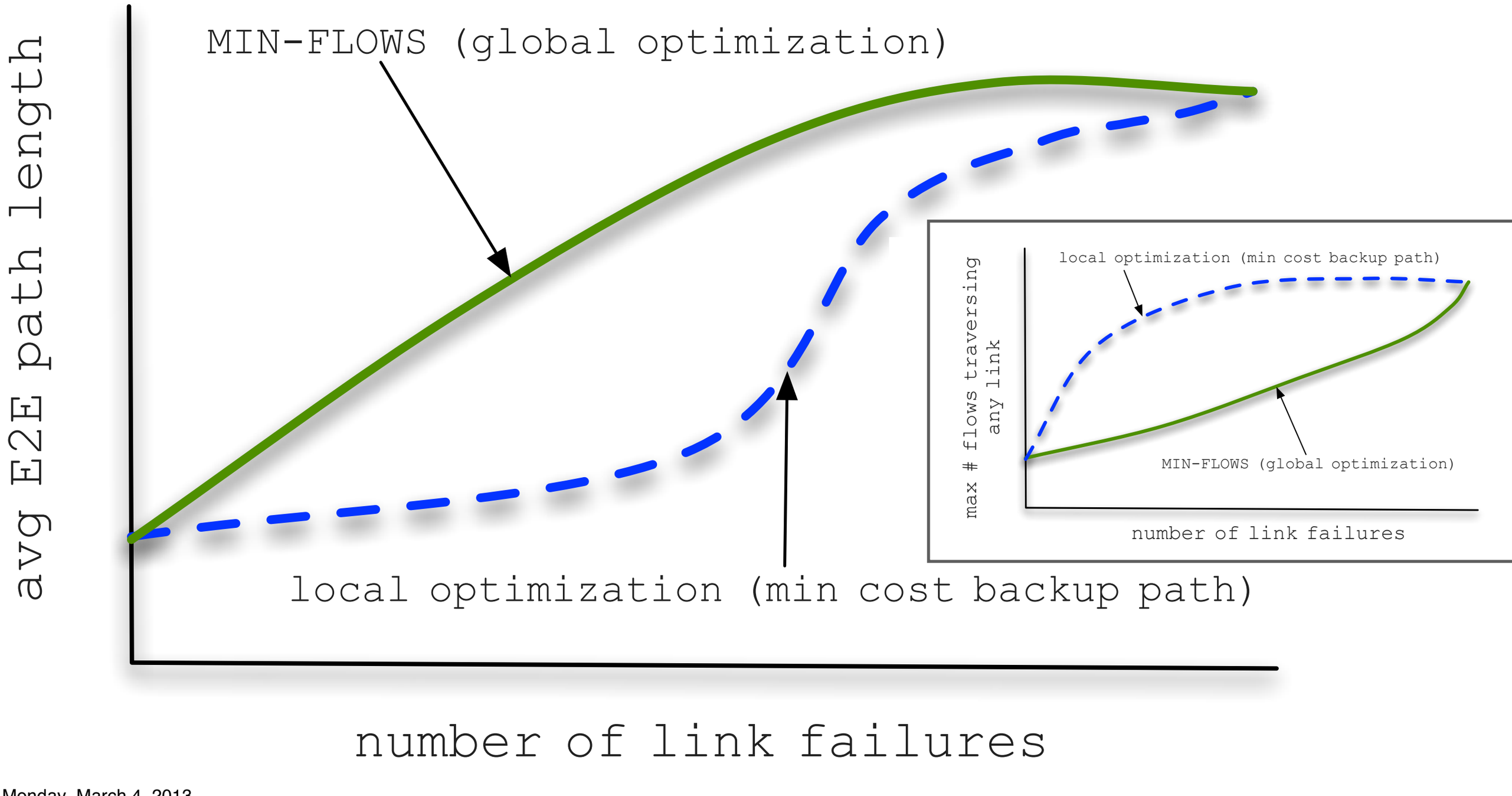
Ch 2: Research Questions

Q3: How much more does global optimization increase robustness than doing localized opt.?



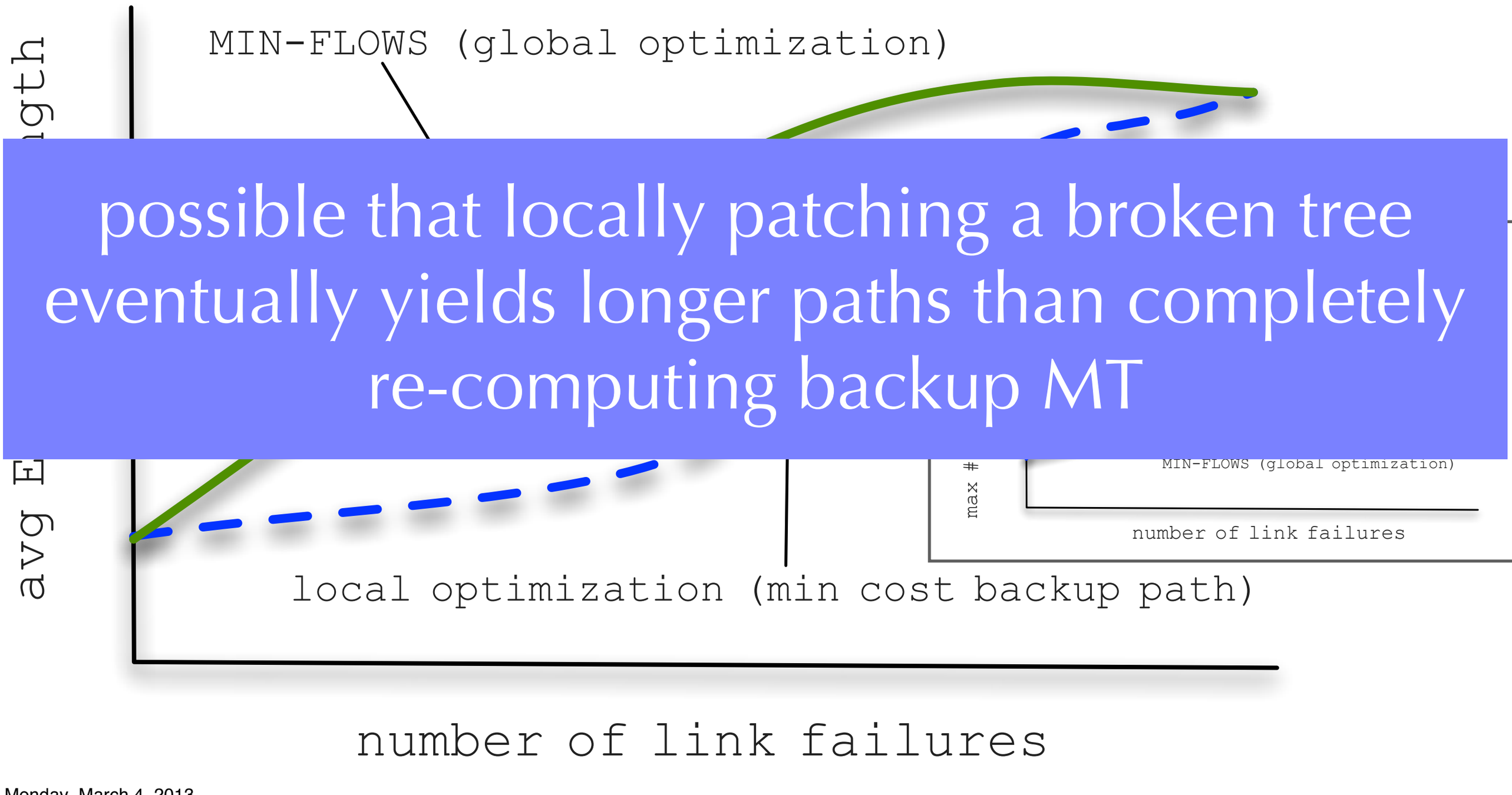
Ch 2: Research Questions

Q3: How much more does global optimization increase robustness than doing localized opt.?



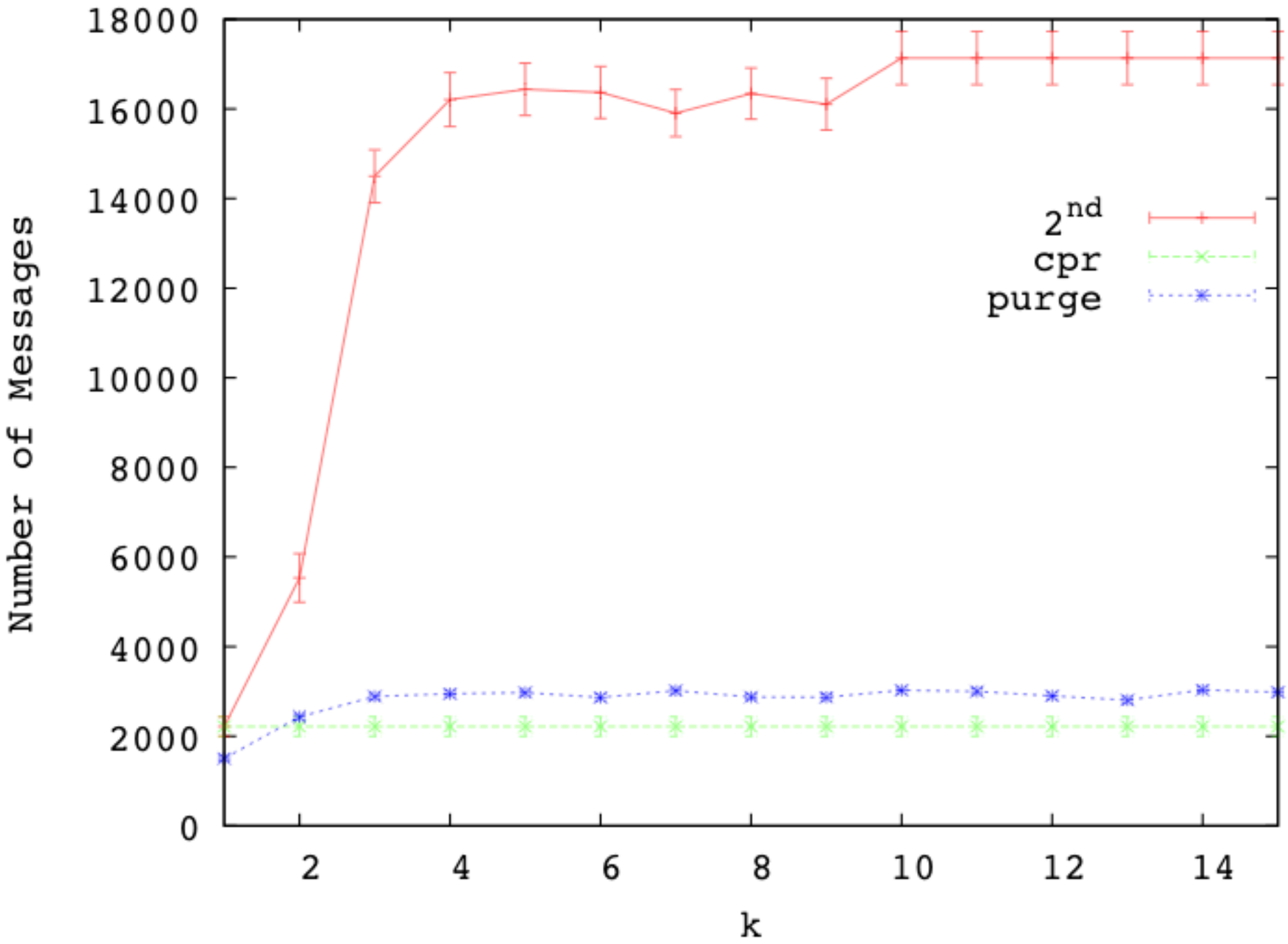
Ch 2: Research Questions

Q3: How much more does global optimization increase robustness than doing localized opt.?



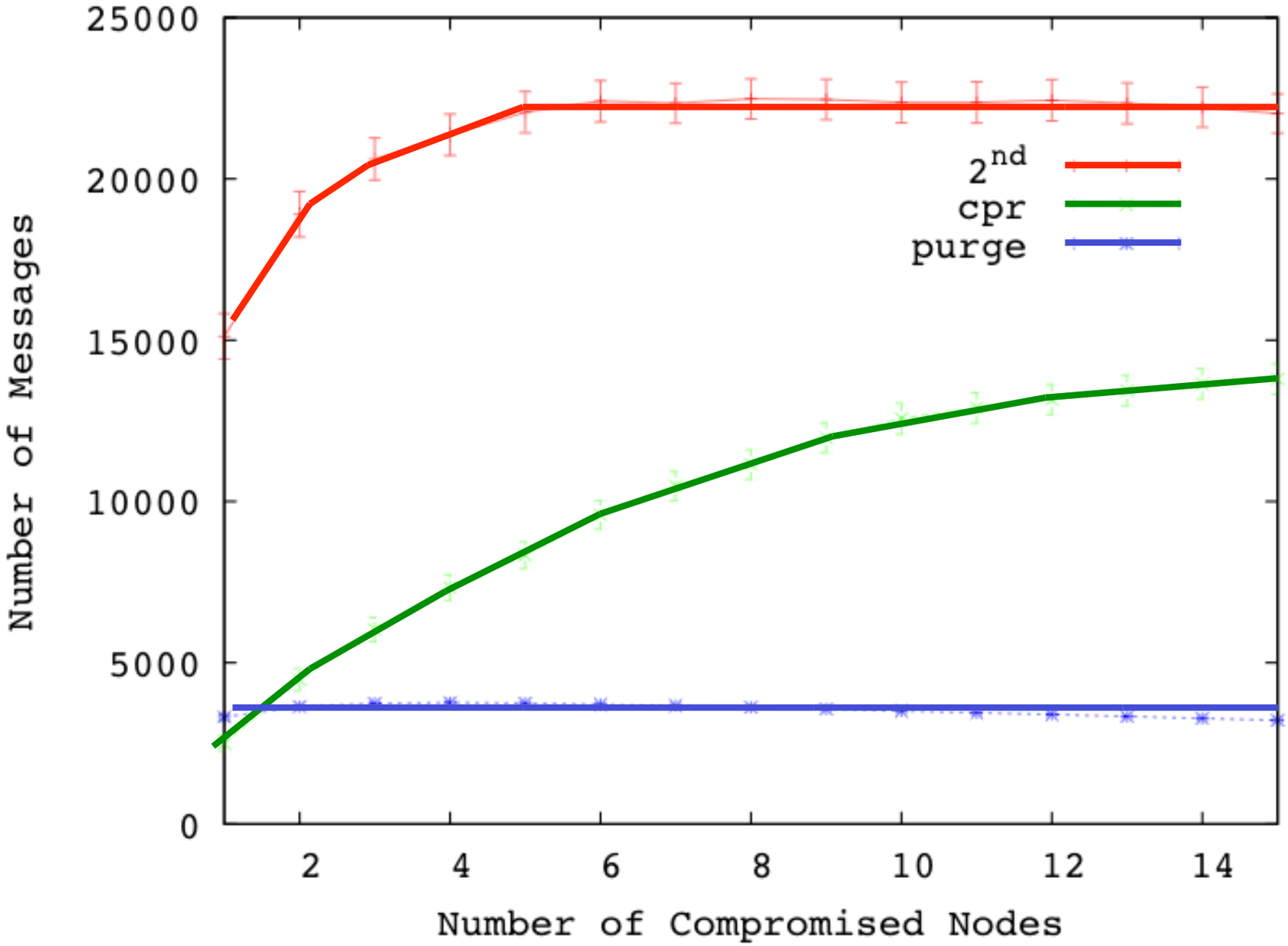
possible that locally patching a broken tree eventually yields longer paths than completely re-computing backup MT

Ch 3: Fixed Link Cost Graphs



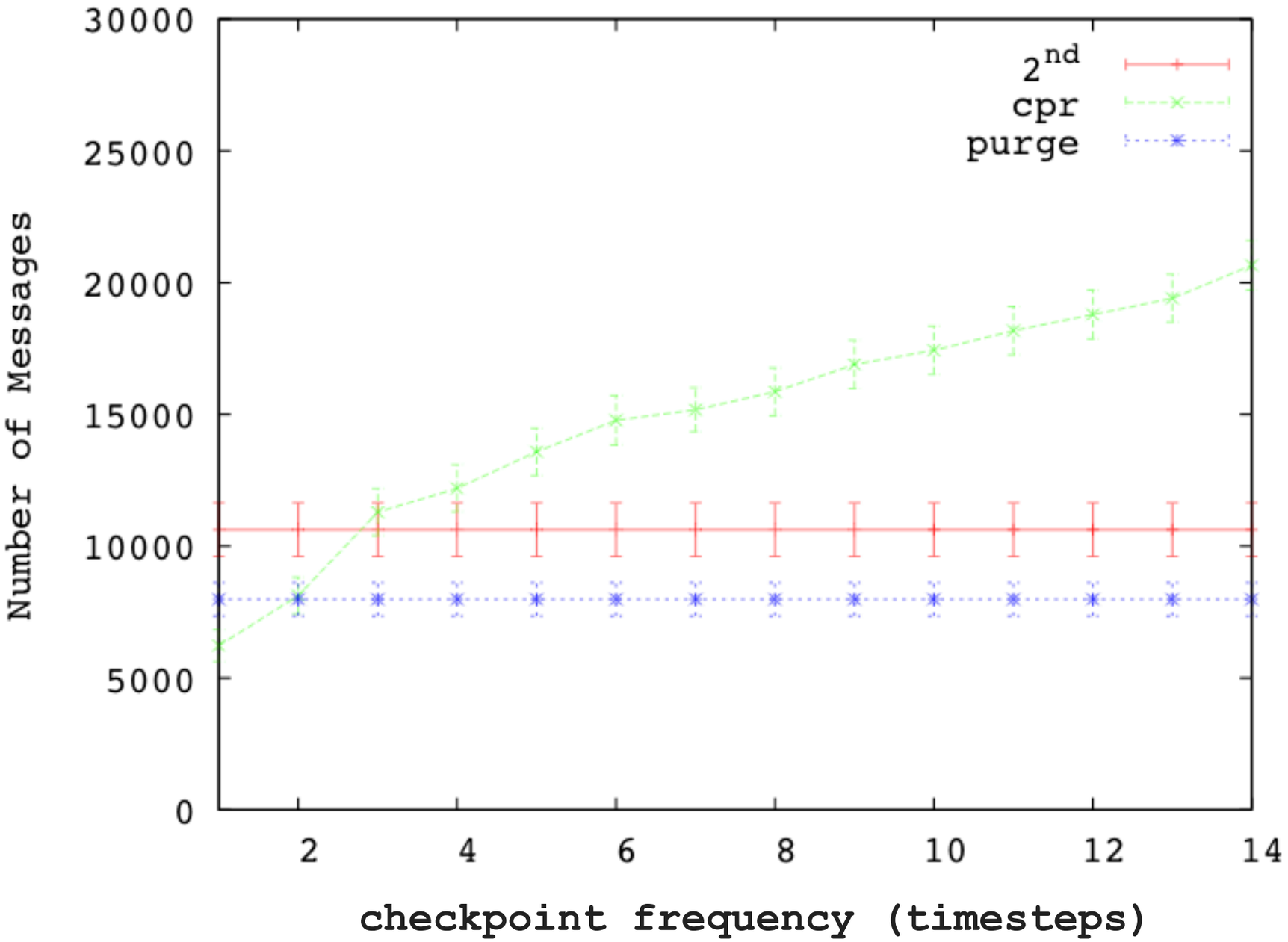
- 2ND BEST: many routing loops
- CPR: removes state with checkpoint and rollback
- 2ND BEST and PURGE: use iterative distance vector

Ch 3: Multiple Compromised Nodes



- 2ND BEST: many routing loops
- CPR: much stale state after rollback => routing loops
- PURGE: no routing loops + no stale state

Ch 3: Setting Checkpoint Frequency



- CPR: less frequent checkpoints => more overhead
- 2ND BEST and PURGE: constant overhead b/c neither algorithm checkpoints

Ch 3: Detecting Compromised Node

- detection is out of scope
  - output of detection algorithm is input to our recovery algorithms
- our algorithms work independent of detection algorithm

- some compromised node detection algorithms named in the report [22,23,25,46]
  - Feamster et al. "Detecting BGP Configuration Faults with Static Analysis" [23]
  - K. School et al. "Context Aware Detection of Selfish Nodes in DSR based Ad-hoc Networks" [46]

Ch 3: Open Questions

1. what is the “worst” false routing state
   - considered case of vector of all `0′ distances
     - easy to detect
   - false state that is easier to disguise?
   - state that maximizes count-to-infinity problem
2. algorithm performance over IEEE Bus Systems