Randomized Gossip Methods

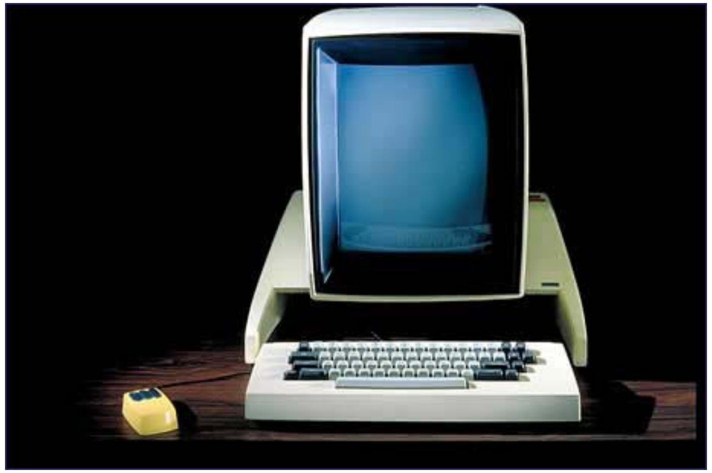
from Grapevine to SWIM

Dahlia Malkhi

principal researcher
VMware Research
http://research.vmware.com

vmware


10.-22.-38
ASTORIA



1938: first successful xerographic image
1979: first PC, Alto

Operating Systems — Anita K. Jones, Editor

Grapevine: An Exercise in Distributed Computing

Andrew D. Birrell, Roy Levin,
Roger M. Needham, and Michael D. Schroeder

Xerox Palo Alto Research Center

1982, CACM

Grapevine is a multicomputer system on the Xerox research internet. It provides facilities for the delivery of digital messages such as computer mail; for naming people, machines, and services; for authenticating people

Epidemic Algorithms for Replicated Database Maintenance

Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry

Xerox Palo Alto Research Center

1987, PODC

se is replicated at many sites, maintaining among the sites in the face of updates is a

This paper describes several randomized buting updates and driving the replicas to-

In this paper we present analyses, si practical experience using several strategi dates. The methods examined include:

1. *Direct mail:* each new update is immedi entry site to all other sites. This is ti

Randomized Rumor Spreading

R. Karp* C. Schindelhauer† S. Shenker‡ B. Vöcking§

2000, FOCS

Abstract

We investigate the class of so-called epidemic algorithms that are commonly used for the lazy transmission of updates to distributed copies of a database. These algorithms use a simple randomized communication mechanism to ensure robustness. Suppose n players communicate in parallel rounds in each of which every player calls a randomly

1 Introduction

We investigate the problem of spreading rumors in a distributed environment using randomized communication. Suppose n players exchange information in parallel communication rounds over an indefinite time. In each round t, the players are connected by a communication graph G_t generated by *random phone calls* as follows: each player u

Motivation

1. *Direct mail:* each new update is immediately mailed from its entry site to all other sites. This is timely and reasonably efficient but not entirely reliable since individual sites do not always know about all other sites and since mail is sometimes lost.

2. *Anti-entropy:* every site regularly chooses another site at random and by exchanging database contents with it resolves any differences between the two. Anti-entropy is extremely reliable but requires examining the contents of the database and so cannot be used too frequently. Analysis and simulation show that anti-entropy, while reliable, propagates updates much more slowly than direct mail.

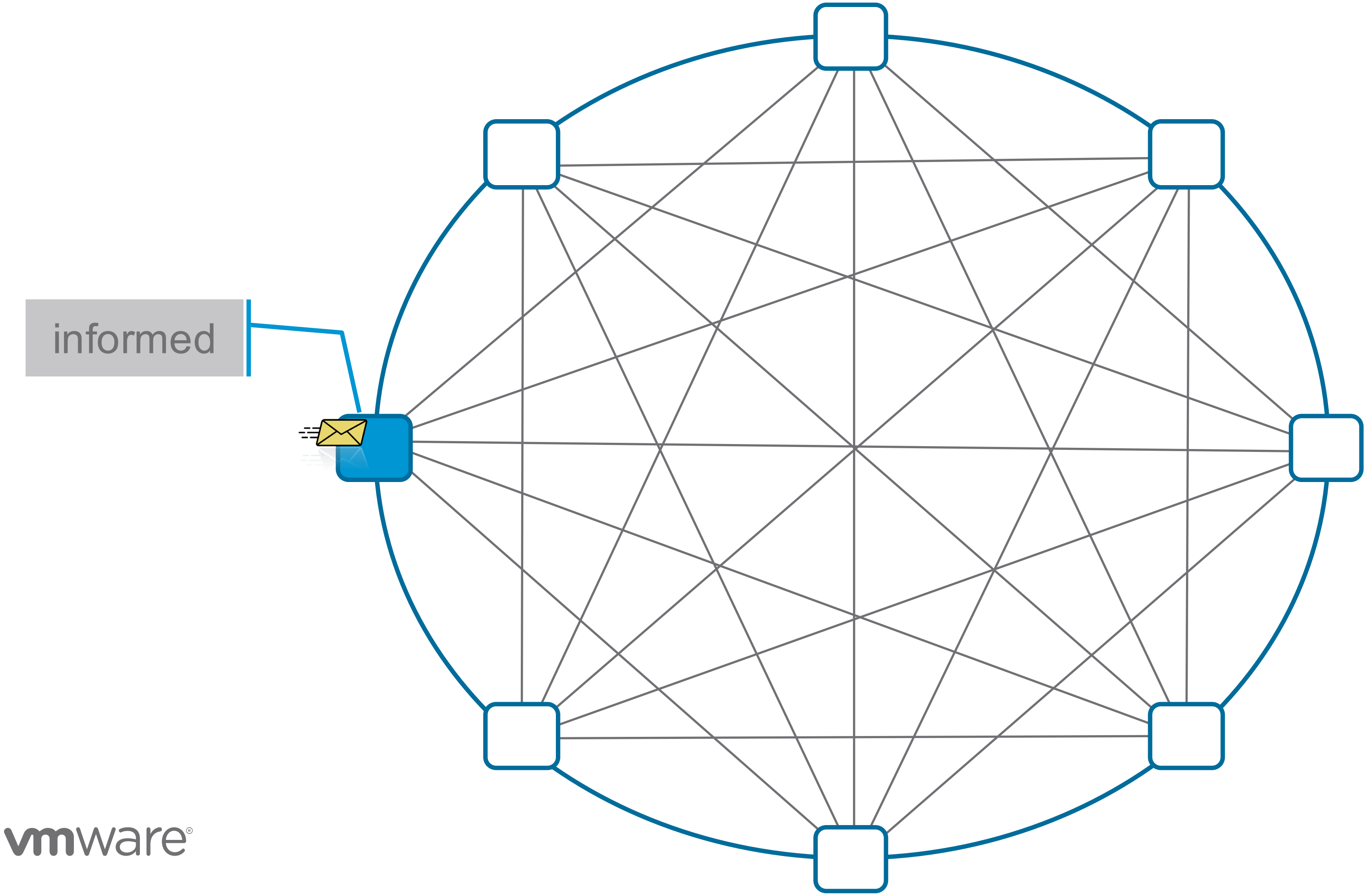
3. *Rumor mongering:* sites are initially "ignorant"; when a site receives a new update it becomes a "hot rumor"; while a site holds a hot rumor, it periodically chooses another site at random and ensures that the other site has seen the update; when a site has tried to share a hot rumor with too many sites that have already seen it, the site stops treating the rumor as hot and retains the update without propagating it further. Rumor cycles can be more frequent than anti-entropy cycles because they require fewer resources at each site, but there is some chance that an update will not reach all sites.

Enters: ”Random Phone Call” Framework

- Framework definition:
  - synchronous rounds
  - each node initiates one connection
  - full network, accurate membership, choose partners at random

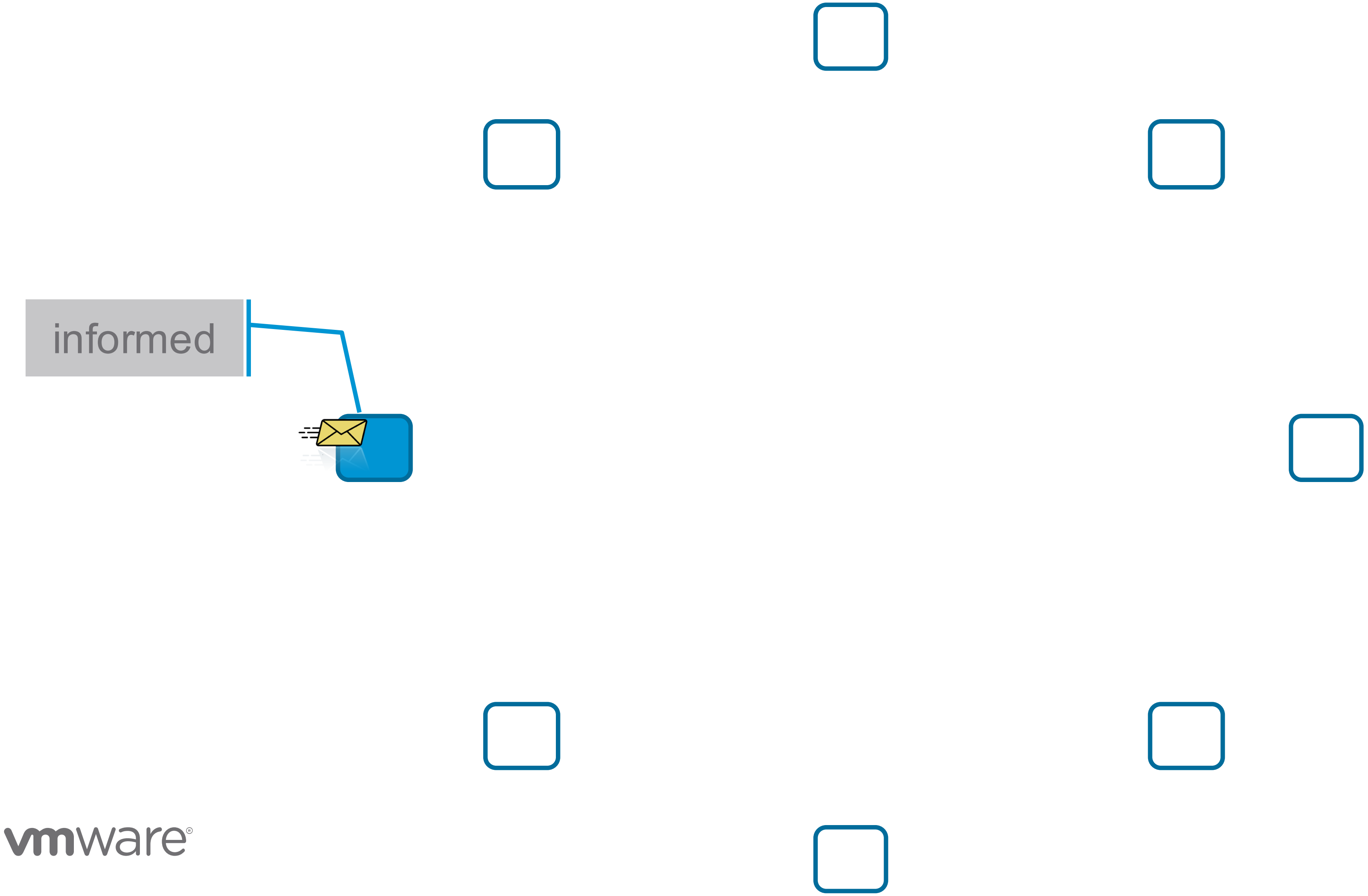
- Protocols in this talk:
  - rumor mongering
  - failure detection
  - network discovery

Demonstration of Randomized Gossip

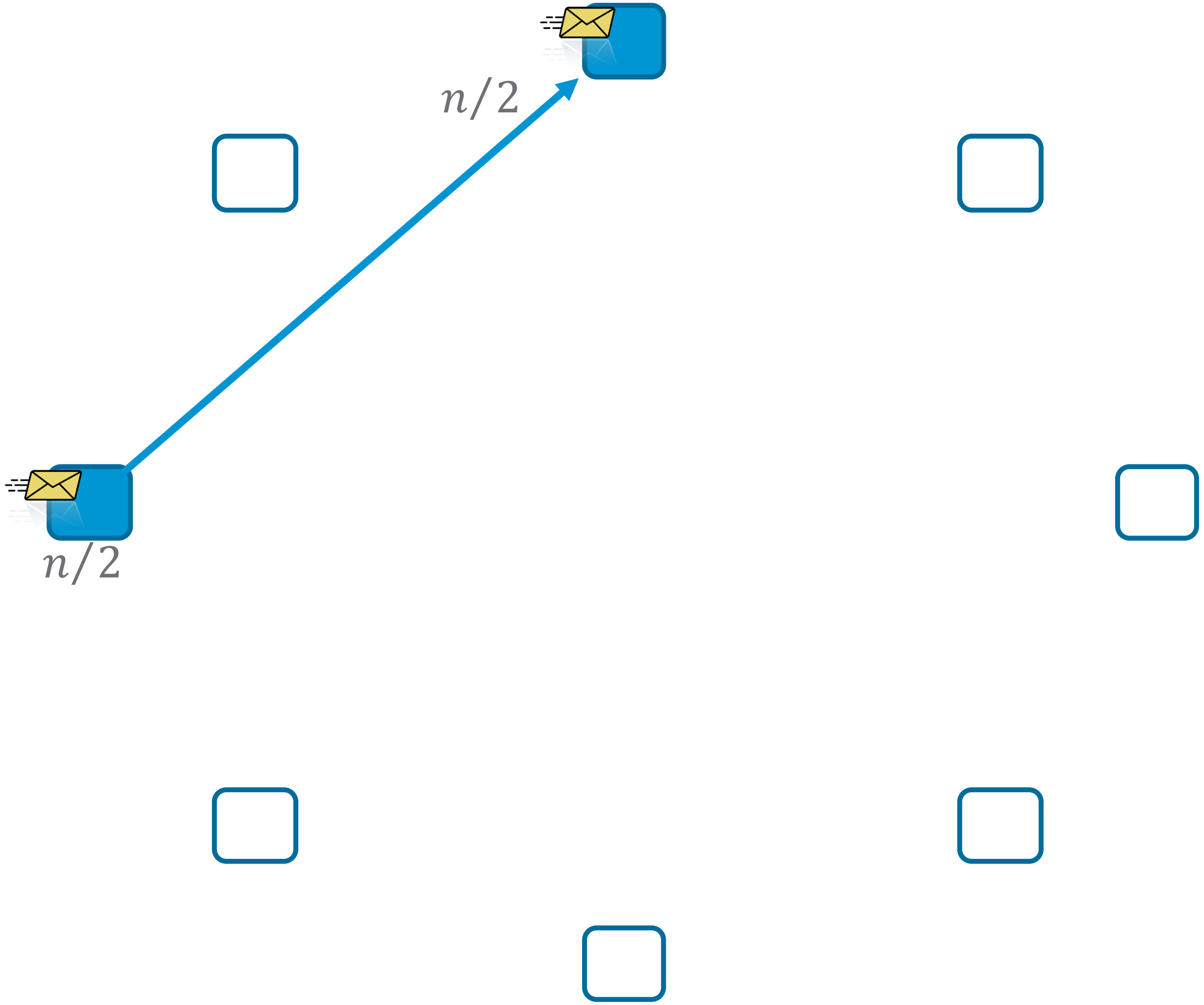


vmware

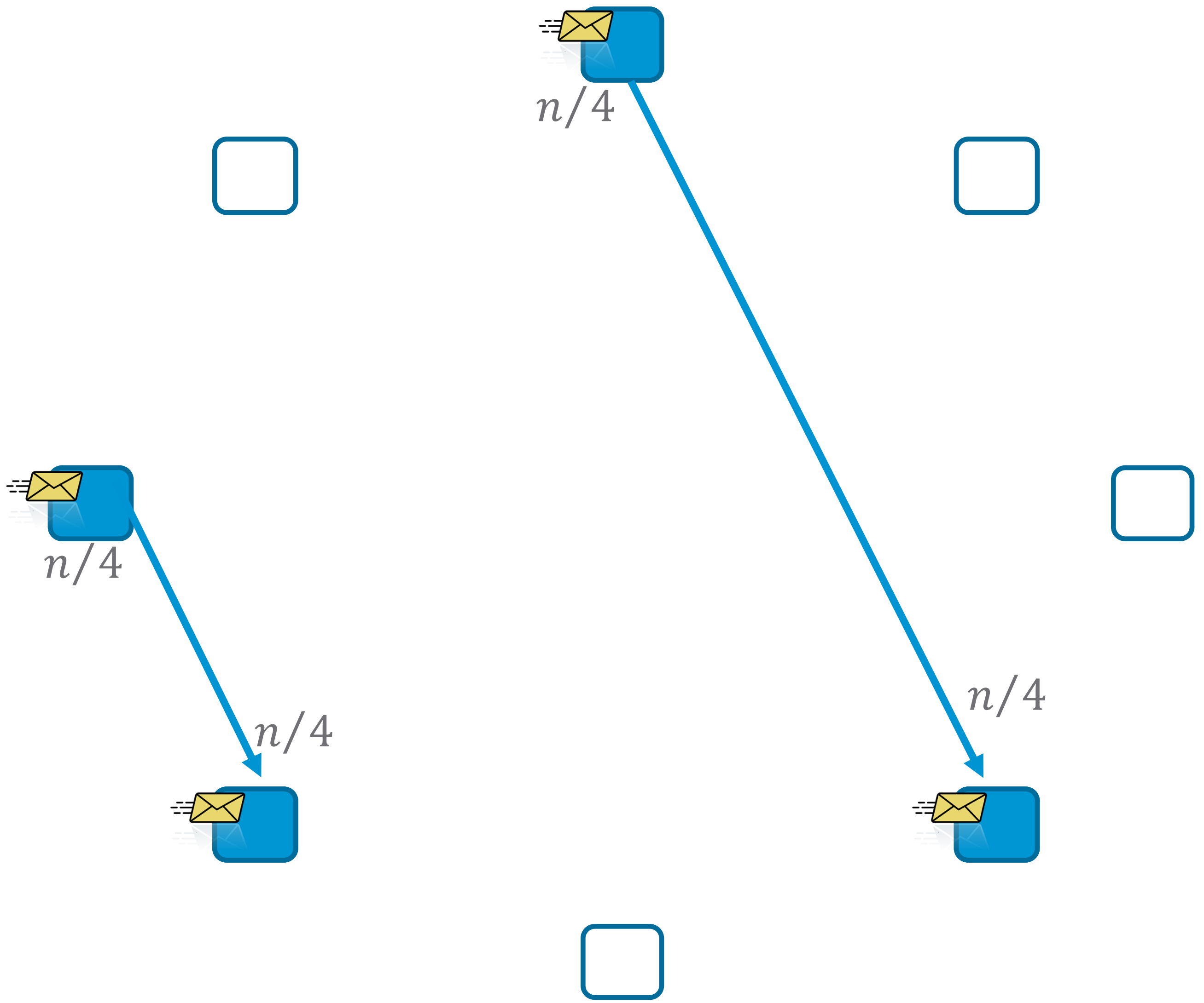
Demonstration of Randomized Gossip



Demonstration of Randomized Gossip

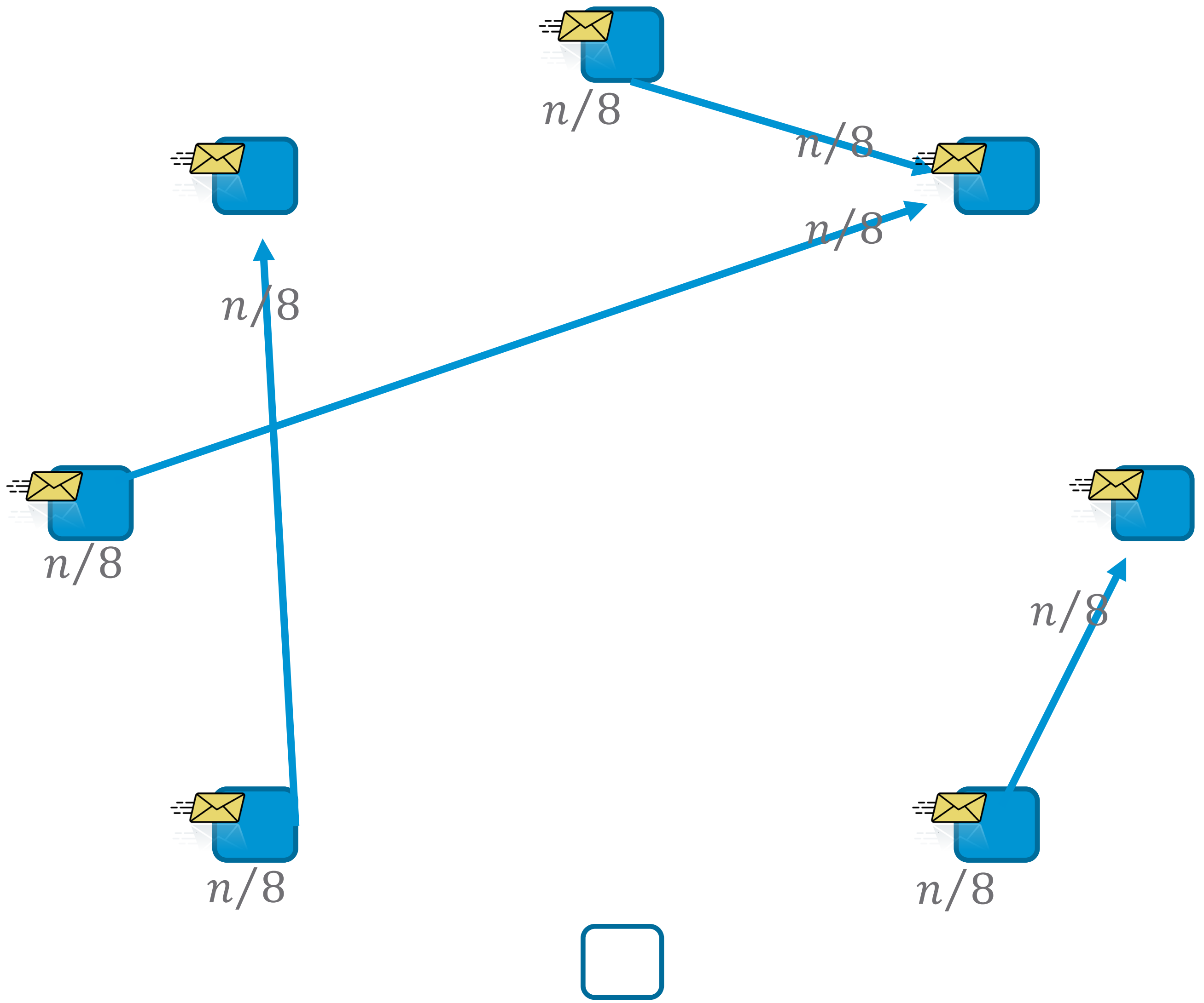


Demonstration of Randomized Gossip

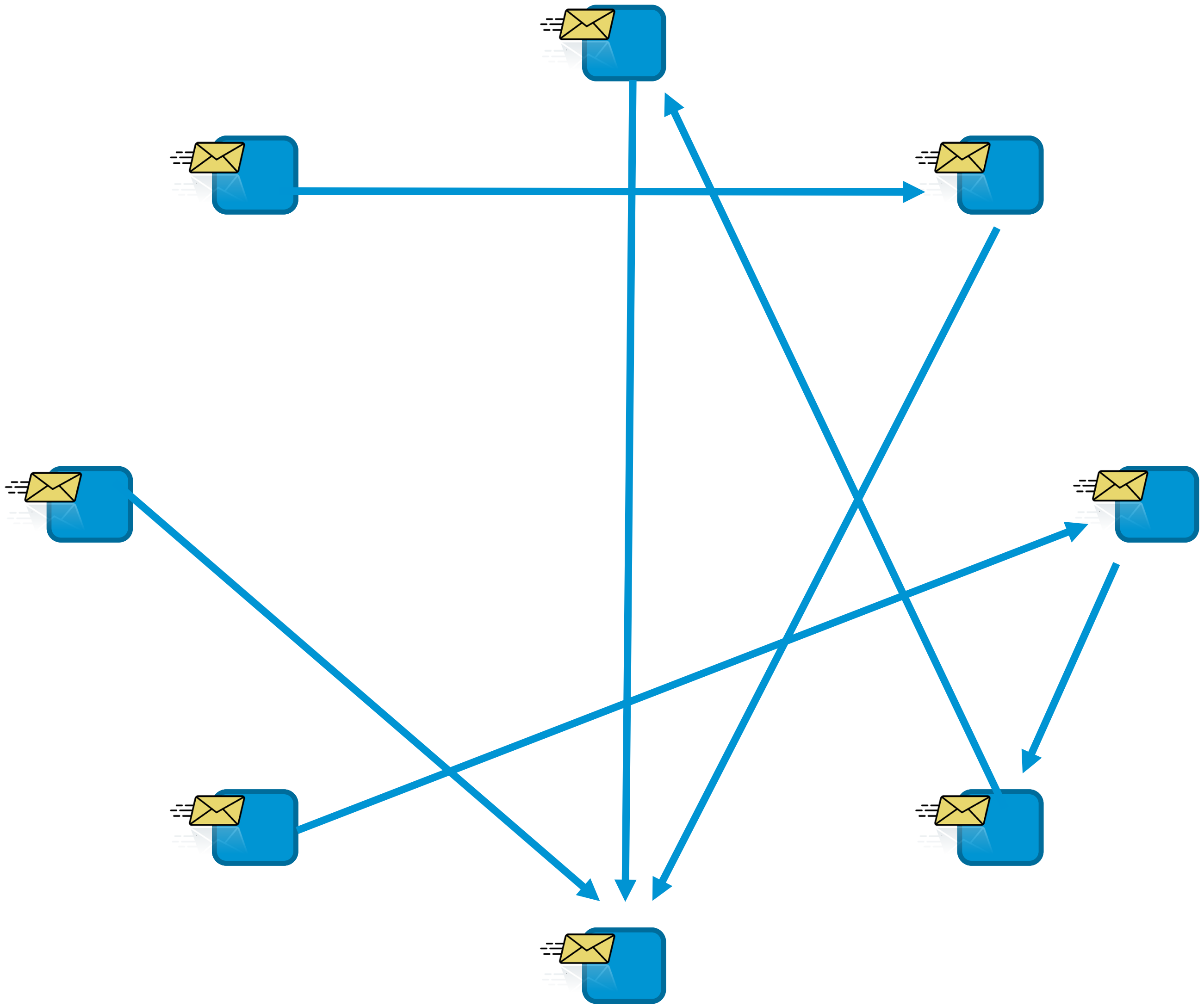


vmware

Demonstration of Randomized Gossip



Demonstration of Randomized Gossip



vmware

Complexity of Gossip Processes in Full Networks

- *n* nodes
- rounds to completion
  - lower bound
  - upper bound
- PUSH message complexity
- PULL message complexity
- PUSH-PULL message complexity

Complexity of Gossip Processes in Full Networks

Round complexity lower bound:

- in a round, informed has *informed* number of nodes
- in a round, every node interacts with expected 1 node
- in a round where number of informed > *log n*
  - with high probability, informed nodes interact with < 2x *informed* nodes
  - *informed* grows by at most factor 3
- *rounds* = $\Omega(\log n)$

Complexity of Gossip Processes in Full Networks

Round complexity PUSH upper bound, first phase:

- in a round where fraction of informed nodes < ½ :
  - probability of successful PUSH of an informed node > ½
  - expected number of successful PUSH's > *informed* / 2
  - with probability > 1-1/n , number of successful PUSH's > *informed* / 3
- in O(*log n*) rounds, *informed* grows to *n/2*

Complexity of Gossip Processes in Full Networks

Round complexity PUSH upper bound, second phase (coupon collector):

- in a round where fraction of informed nodes $\geq$ ½ :
  - probability of successful PUSH to an uninformed node $\geq$ 1- $\left(1-\frac{1}{n}\right)^{n/2} \cong 1- e^{-1/2}$
  - expected number of successful PUSH's to uninformed nodes $\geq$ *uninformed* / 2
  - with probability > 1-1/n , number of successful PUSH's > *uninformed* / 3
- in O(*log n*) rounds, *uninformed* shrinks to zero

Complexity of Gossip Processes in Full Networks

Round complexity PULL upper bound, first phase:

– in a round where *informed* is a small constant:

- non-negligible probability of no successful PULL from informed node

$$\left(\left(1-\frac{1}{n}\right)^{n-1}\right)^{informed} \cong e^{-informed}$$

– in a round where number of informed $\geq$ *log n :*

- with high probability *informed* grows by constant factor

– in O(*log n*) rounds *informed* grows to *n/2*

- but slow start!

Complexity of Gossip Processes in Full Networks

Round complexity PULL upper bound, second phase:

- in a round where *uninformed* = *cn :*
  - probability of unsuccessful PULL by an uninformed node = *c*
  - expected number of unsuccessful PULLs by uninformed nodes = nc^2
    - $.9nc^2$ with high probability
  - in R rounds, *uninformed* shrinks to $n \; x \; c^{(2^R)}$
- in O(*loglog n*) rounds *uninformed* shrinks to *zero*

Complexity of Gossip Processes in Full Networks

- $O(log(n))$ rounds to completion [Demers et al. PODC 1987]
  - lower bound
  - upper bound
- $\text{message complexity}$
  - connections
  - transmissions
- $O\left(nlog(n)\right)$ PUSH message complexity
- $O\left(nlog(n)\right)$ PULL connection complexity
- $O\left(nloglog(n)\right)$ PUSH-PULL message complexity [Karp et al. FOCS 2000]

vmware

Scalability of Randomized Gossip

- fault tolerant
- simple
- round complexity
  - $O(log(n))$ rounds sufficient and necessary for completion
  - synchronous rounds
- message complexity
  - number of interactions
  - number of transmissions
  - size of transmissions
- full network
- precise membership knowledge

Failure Detection

via Randomized Gossip Methods

vmware

A Gossip-Style Failure Detection Service

Robbert van Renesse, Yaron Minsky, and Mark Hayden*

Dept. of Computer Science, Cornell University
4118 Upson Hall, Ithaca, NY 14853

Abstract

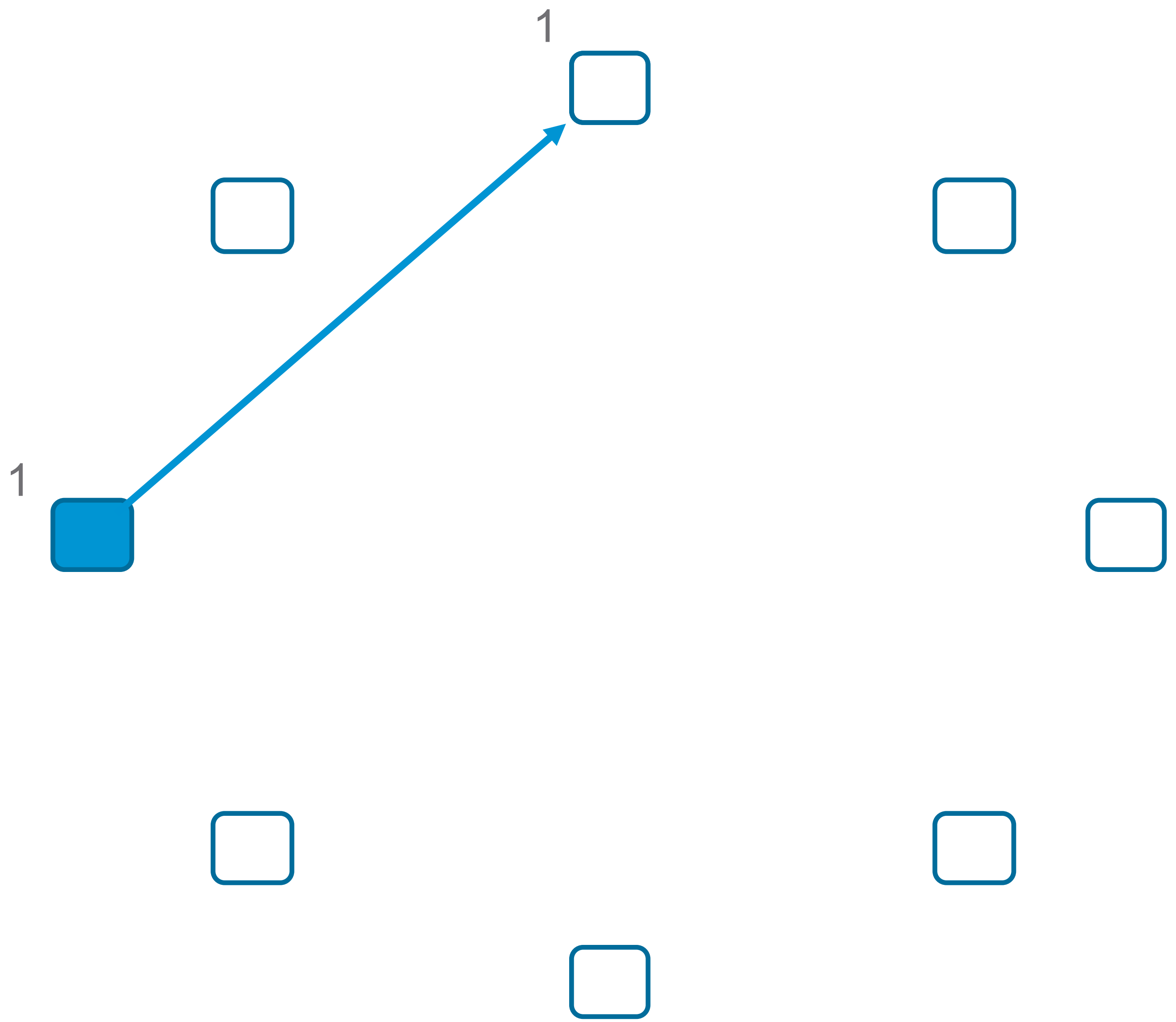
Failure Detection is valuable for system management, replication, load balancing, and other distributed services. To date, Failure Detection Services scale badly in the number of members that are being monitored. This paper describes a new protocol based on gossiping that does scale well and provides timely detection. We analyze the protocol, and then extend it to discover and leverage the underlying network topology for much improved resource utilization. We then combine it with another protocol, based on broadcast, that is used to handle partition failures.

IFIP Middleware 1998

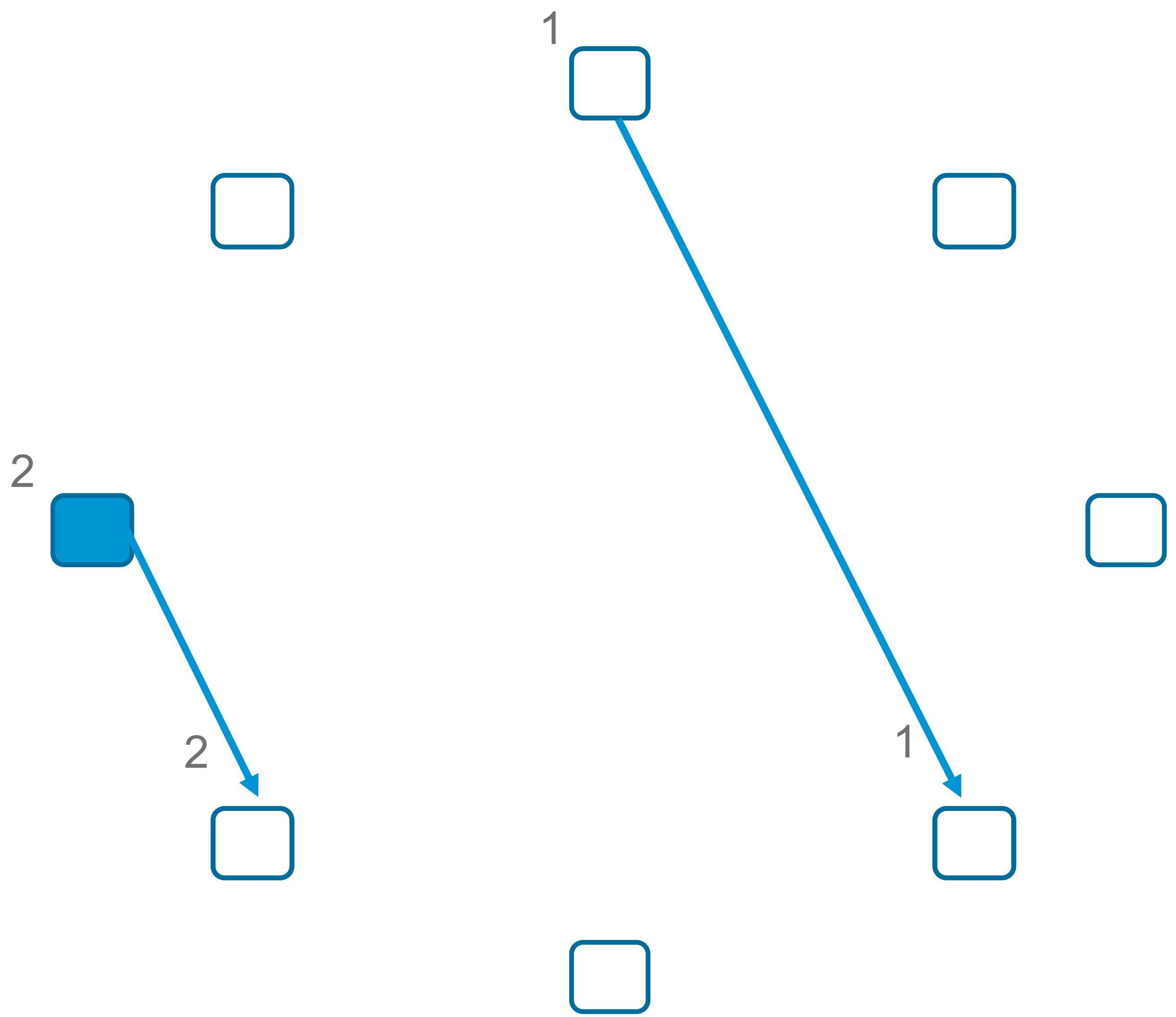
Gossip-Style FD

- Scale heartbeats:
  - Use gossip instead of multicast
  - Each node generates new heartbeat counter in every round

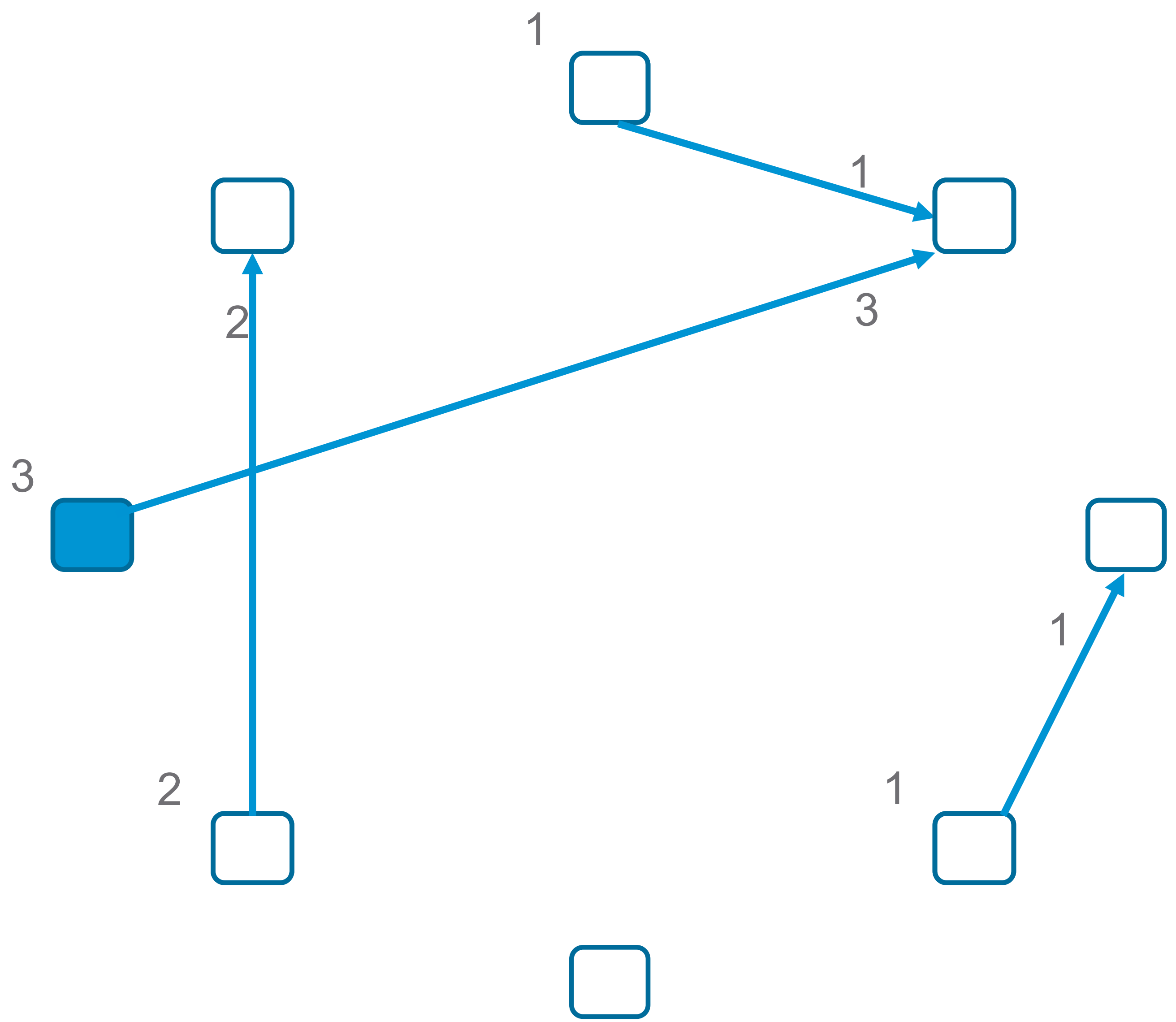
Demonstration of Gossip-style FD



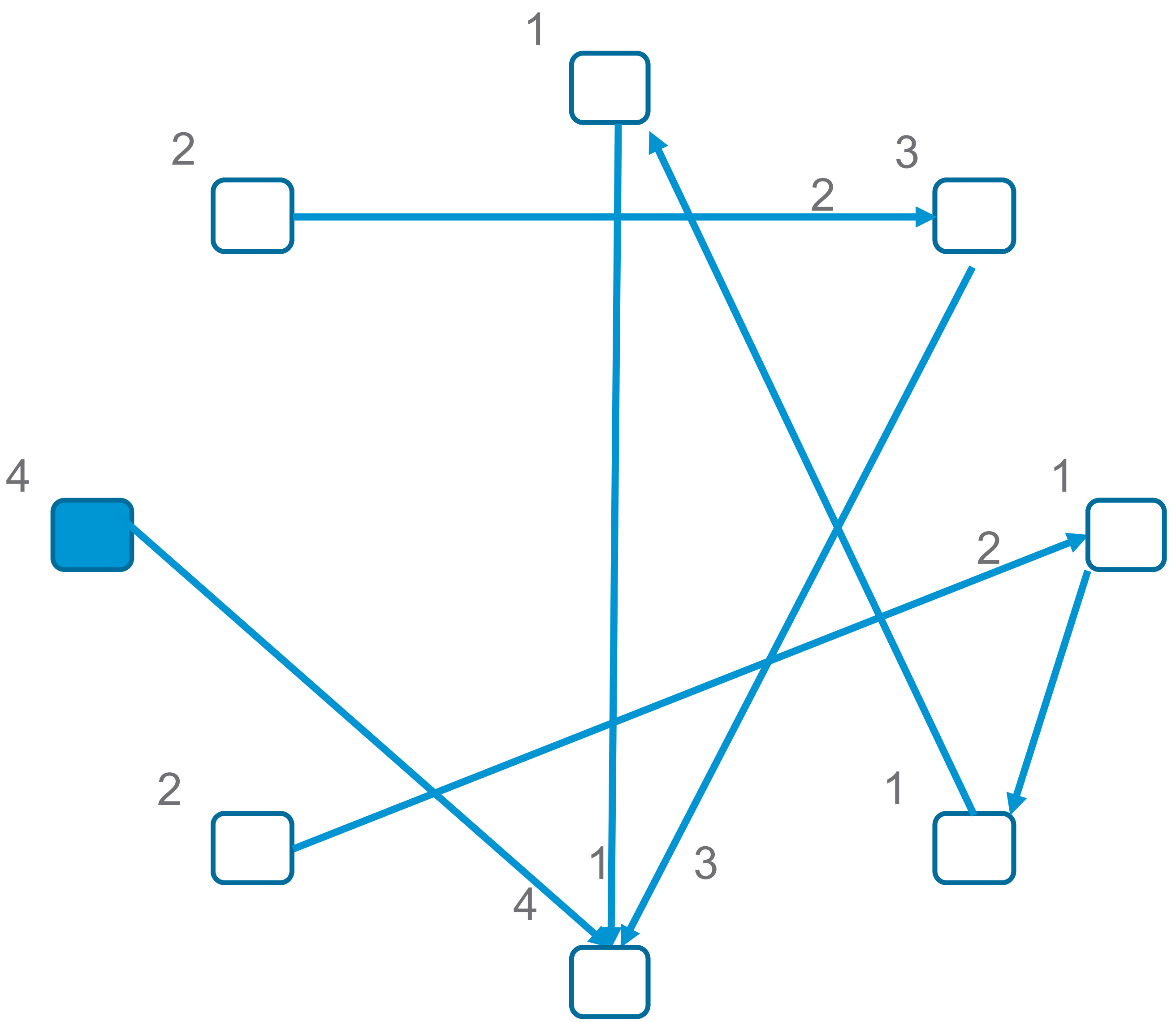
Demonstration of Gossip-style FD



Demonstration of Gossip-style FD



Demonstration of Gossip-style FD



vmware

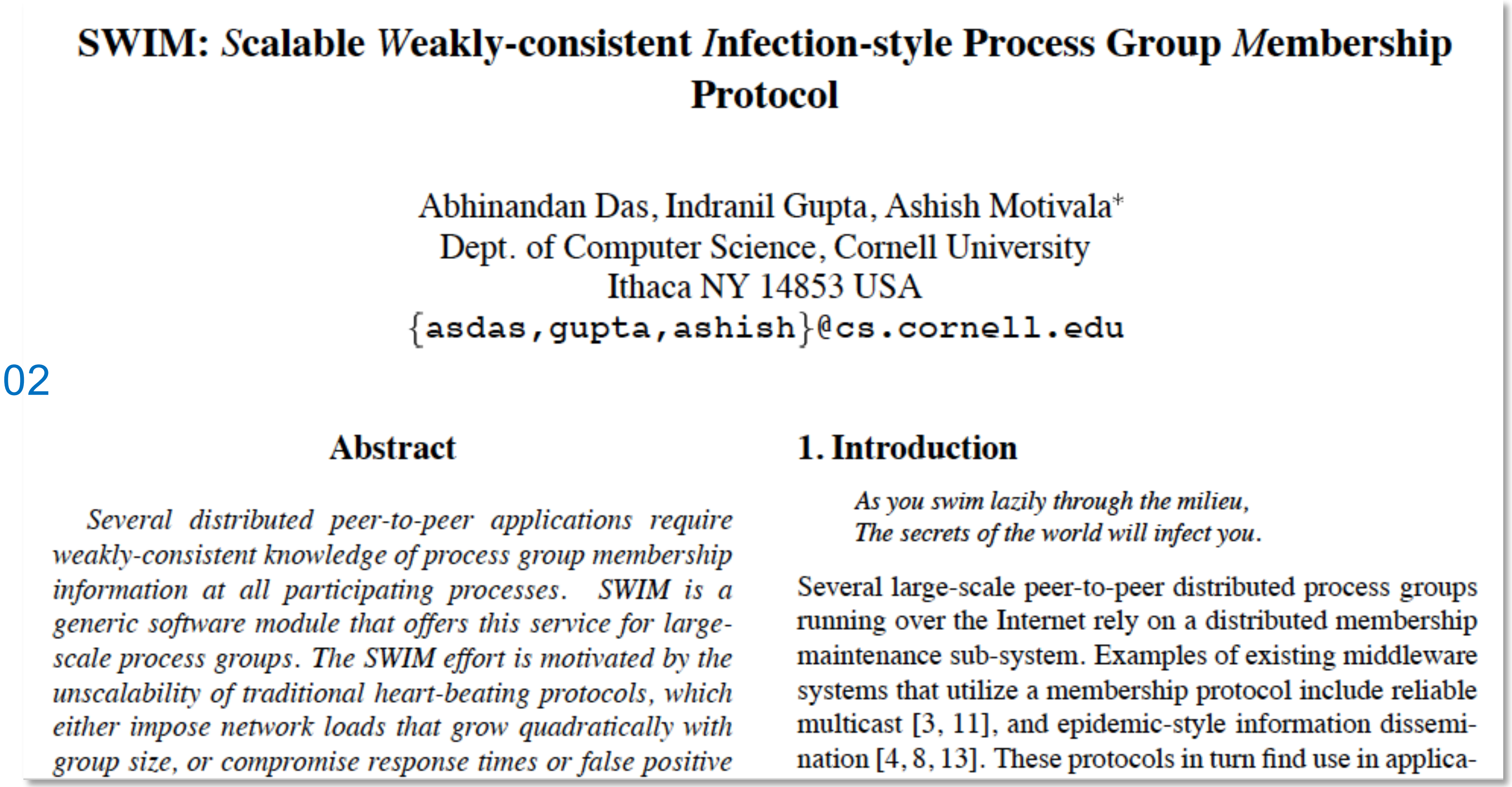
Gossip-Style FD

- each node generates new heartbeat counter in every round
- heartbeat j expected to arrive everywhere by round j + T_{fail}
  - from each node, heartbeat j+1 to follow heartbeat j in
    - expected one round
    - worst case T_{fail} rounds
- stopped heartbeat at round j expected be noticed everywhere by round j + T_{fail}
  - from failed node, stopped heartbeat at round j noticed in
    - gap of up to T_{fail} rounds ; keep tombstone for 2x T_{fail} rounds

Scalability of Gossip-Style FD

- Periodic multicast is hard to scale
  - Every node sends heartbeats by everyone
  - Much of the gossip is redundant or stale
  - Dividing gossip into packets leads to slow failure detection and error-prone convergence

DSN 2002



SWIM: *S*calable *W*eakly-consistent *I*nfection-style Process Group *M*embership Protocol

Abhinandan Das, Indranil Gupta, Ashish Motivala*
Dept. of Computer Science, Cornell University
Ithaca NY 14853 USA
{asdas,gupta,ashish}@cs.cornell.edu

Abstract

Several distributed peer-to-peer applications require weakly-consistent knowledge of process group membership information at all participating processes. SWIM is a generic software module that offers this service for large-scale process groups. The SWIM effort is motivated by the unscalability of traditional heart-beating protocols, which either impose network loads that grow quadratically with group size, or compromise response times or false positive

1. Introduction

As you swim lazily through the milieu,
The secrets of the world will infect you.

Several large-scale peer-to-peer distributed process groups running over the Internet rely on a distributed membership maintenance sub-system. Examples of existing middleware systems that utilize a membership protocol include reliable multicast [3, 11], and epidemic-style information dissemination [4, 8, 13]. These protocols in turn find use in applica-

Scalable Failure Detection

Gossip-Style FD gossips a lot of redundant and stale information

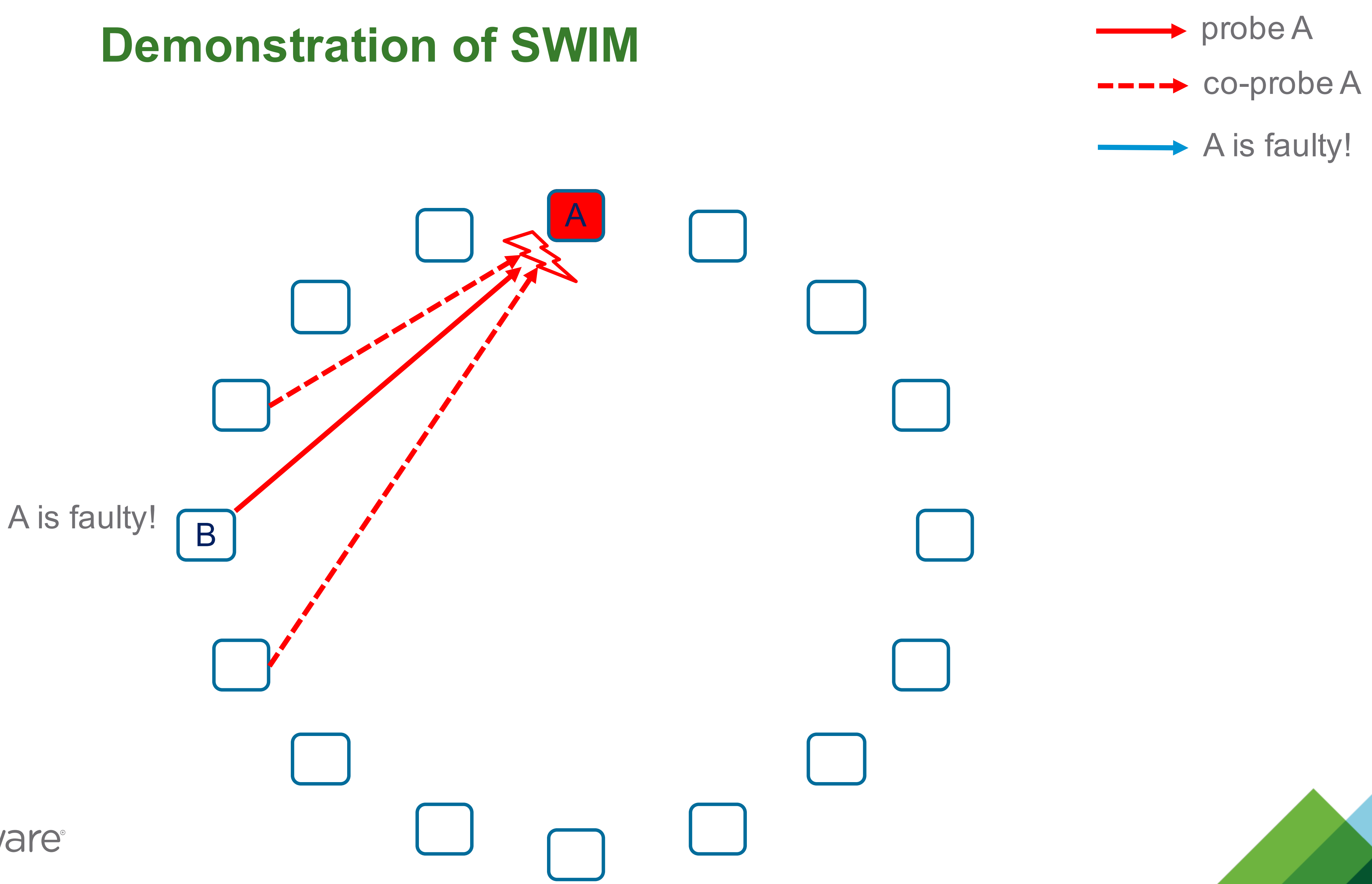
Instead, gossip only alerts!

SWIM

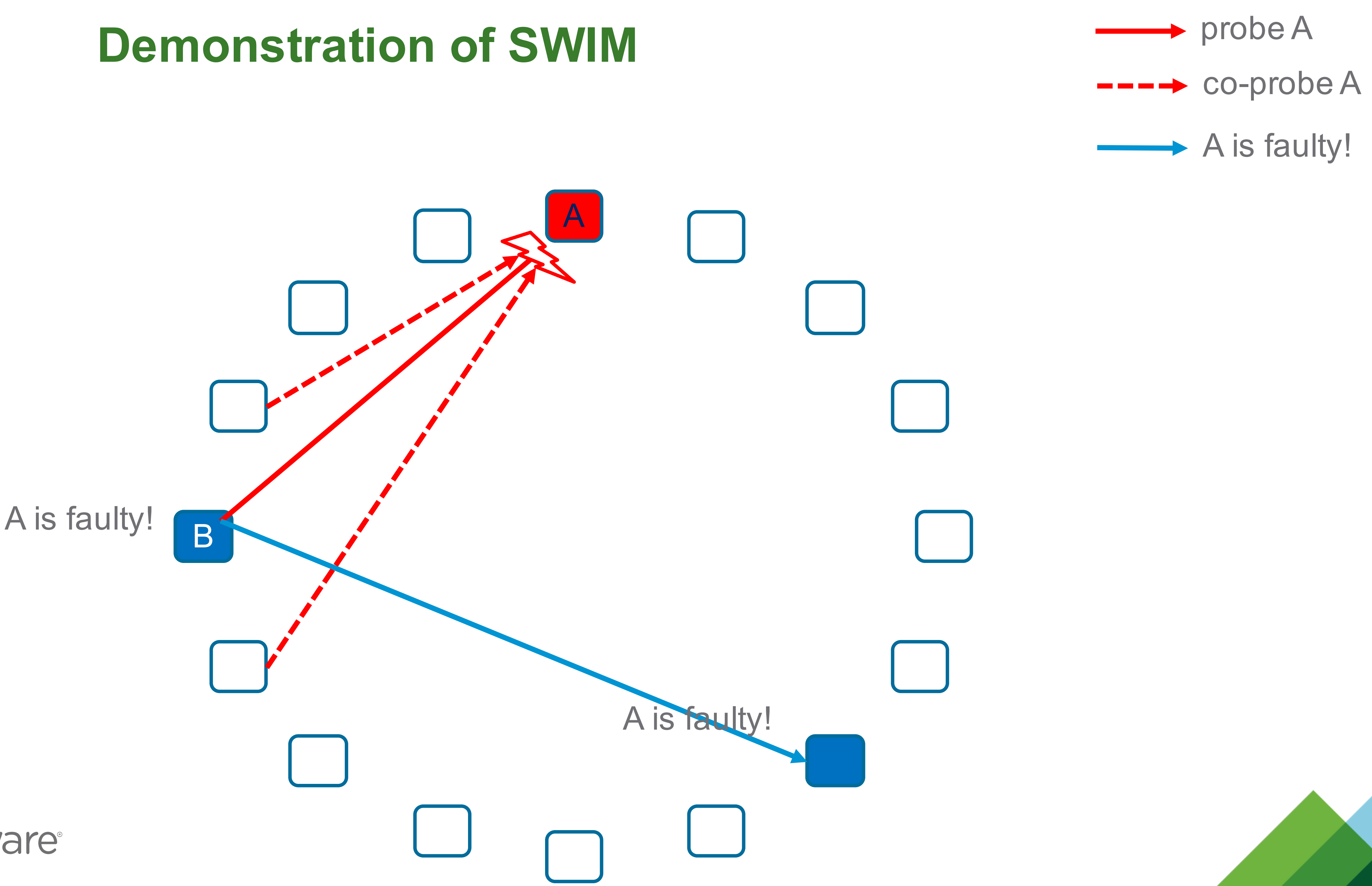
- Failure Detector:
  - In a round, every node probes another node at random
  - a failed probe reinforced by peers
- Weak Membership Service:
  - Spread alerts via gossip

- Every faulty node detected (completeness)
- Constant connection and message overhead per node per round
- Separate failure detection from alert dissemination

Demonstration of SWIM

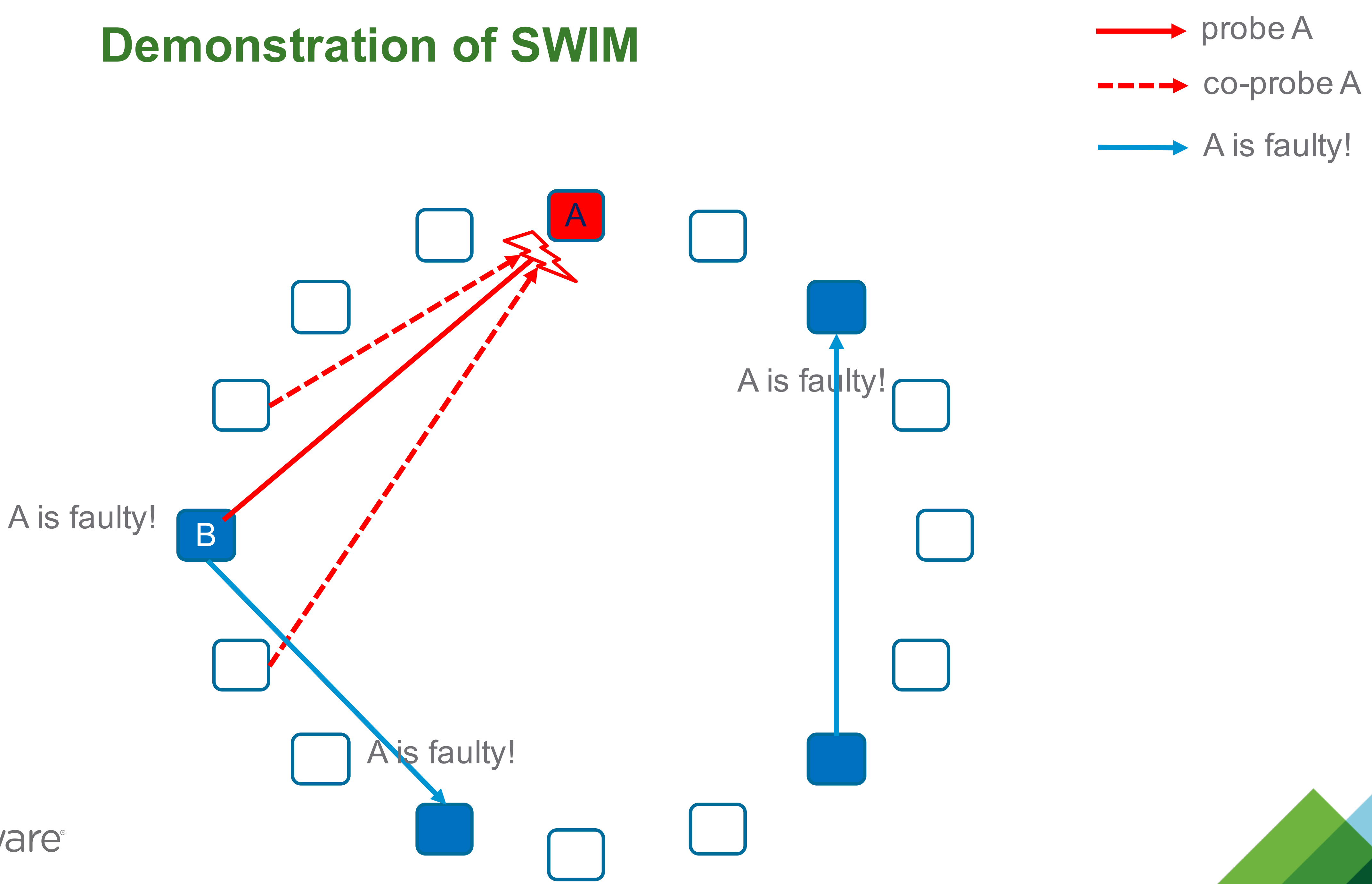


Demonstration of SWIM

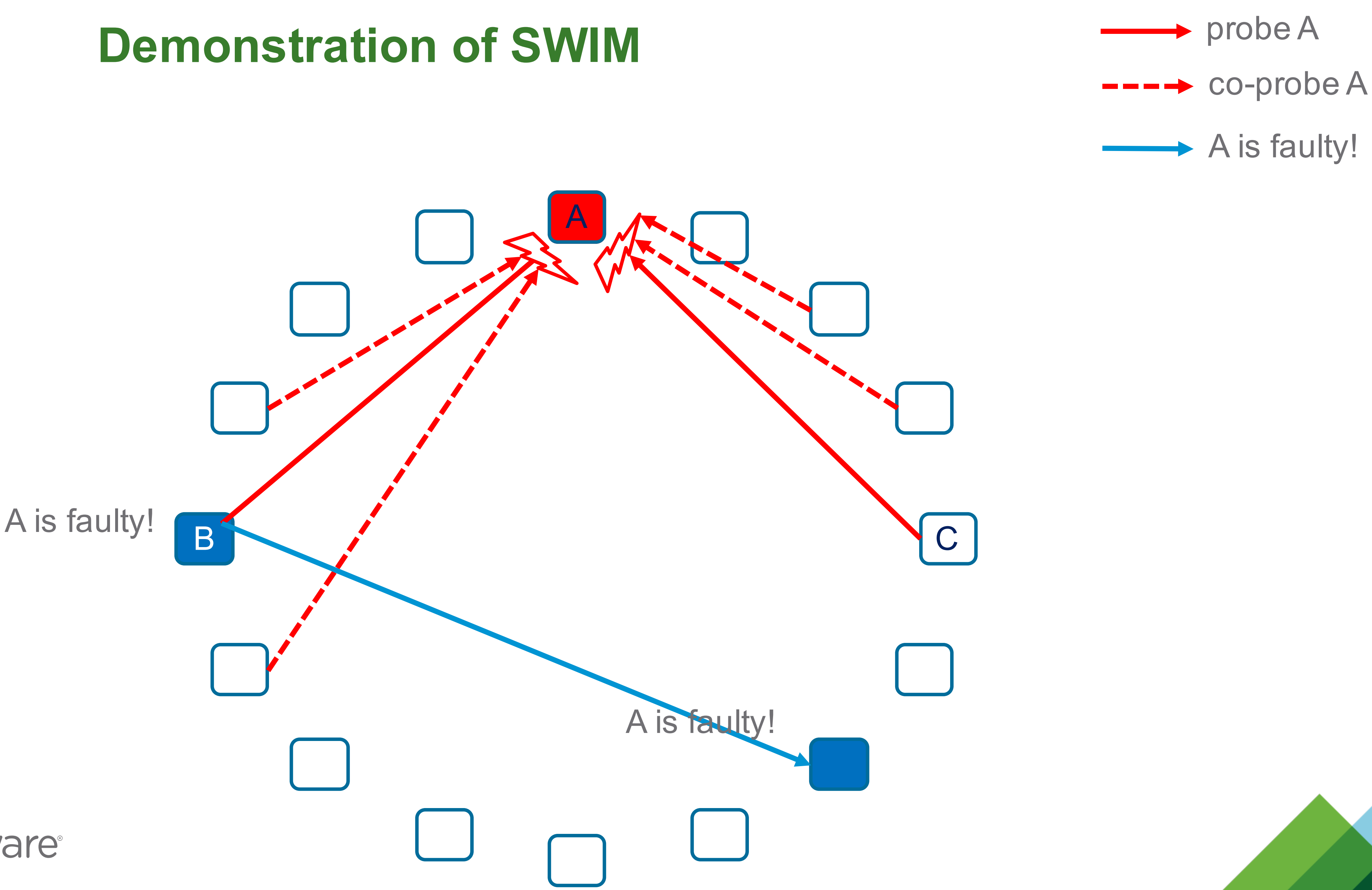


vmware

Demonstration of SWIM



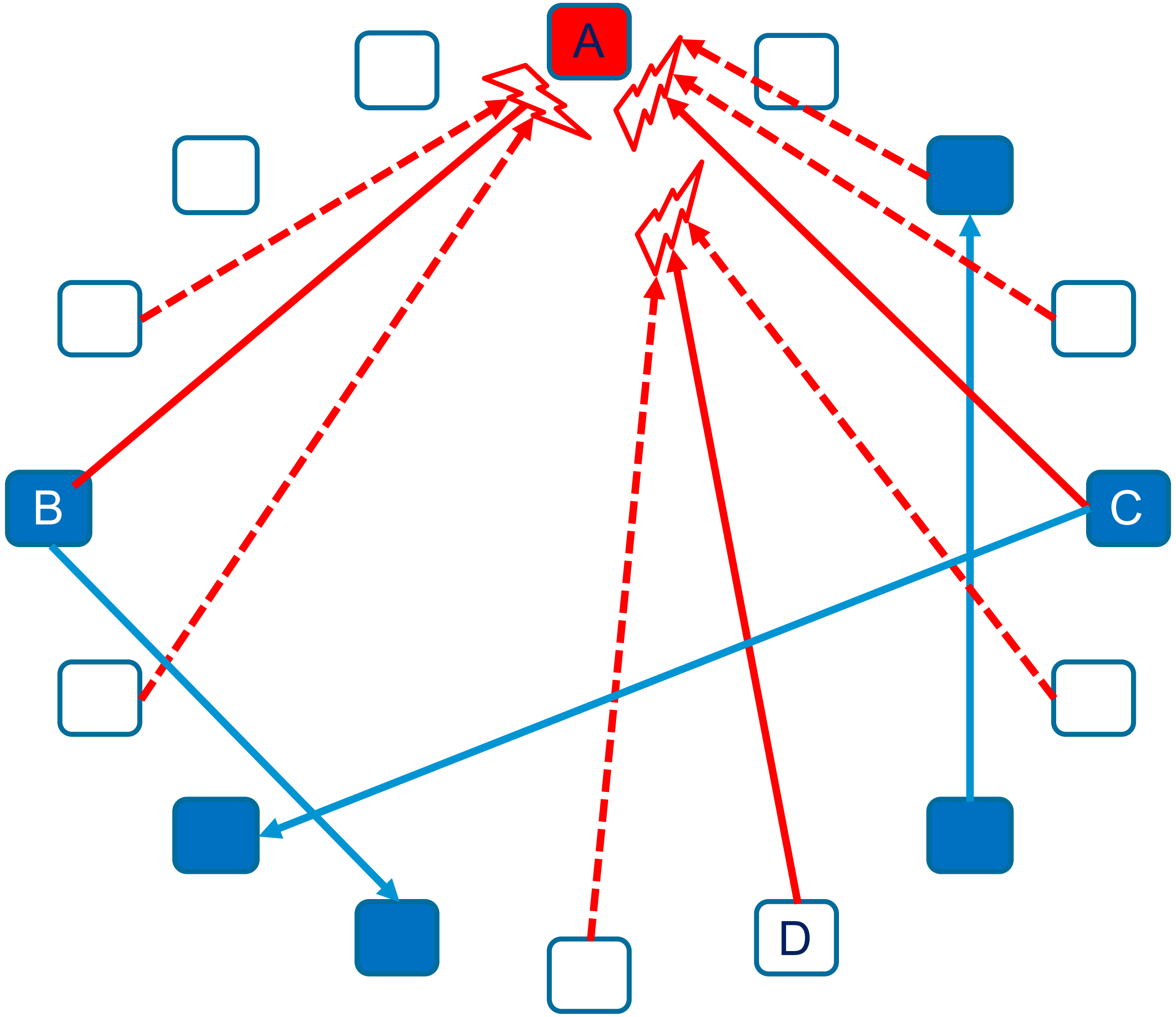
Demonstration of SWIM



vmware

Demonstration of SWIM

probe A
co-probe A
A is faulty!

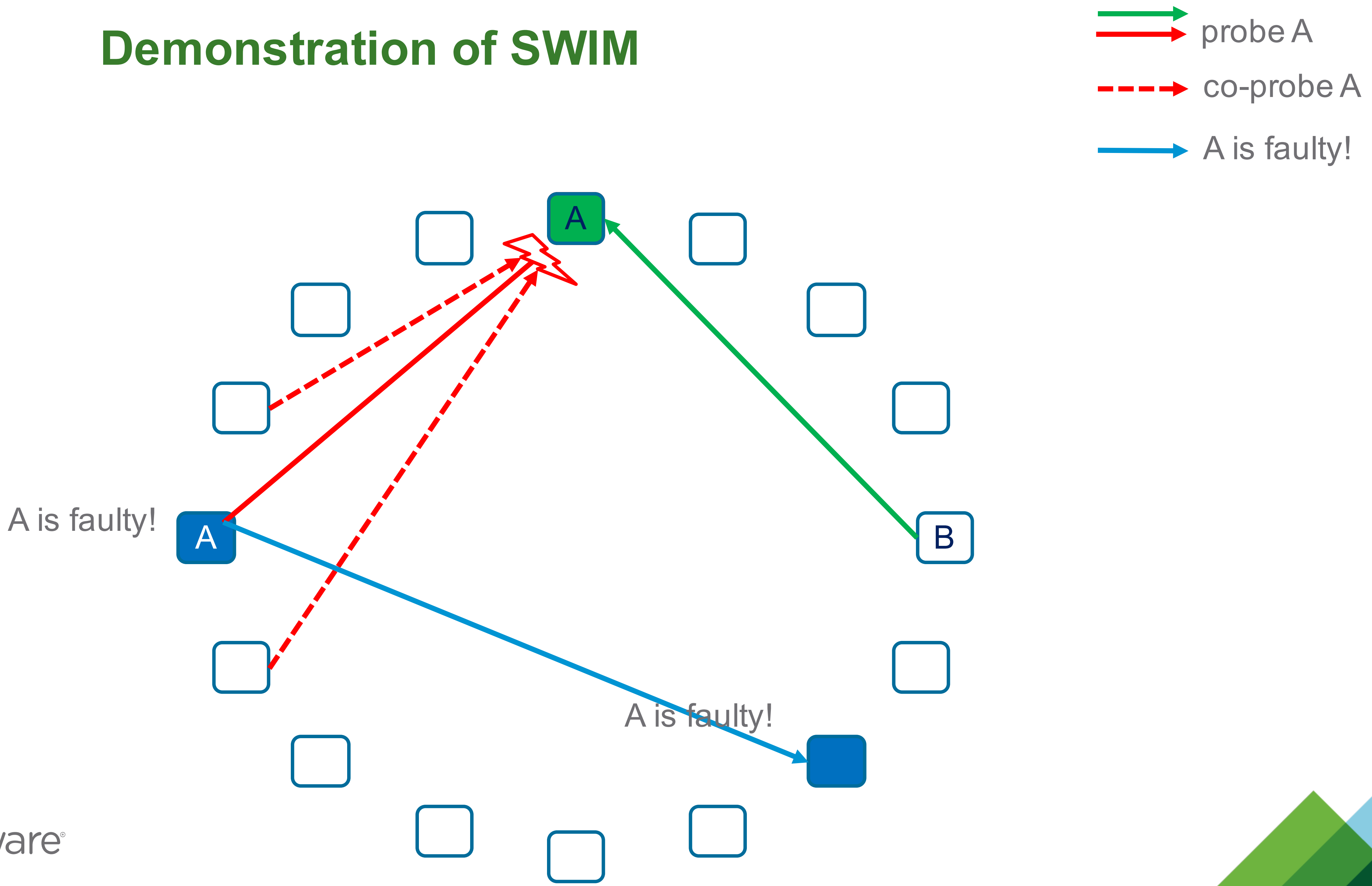


Detection Completeness in SWIM

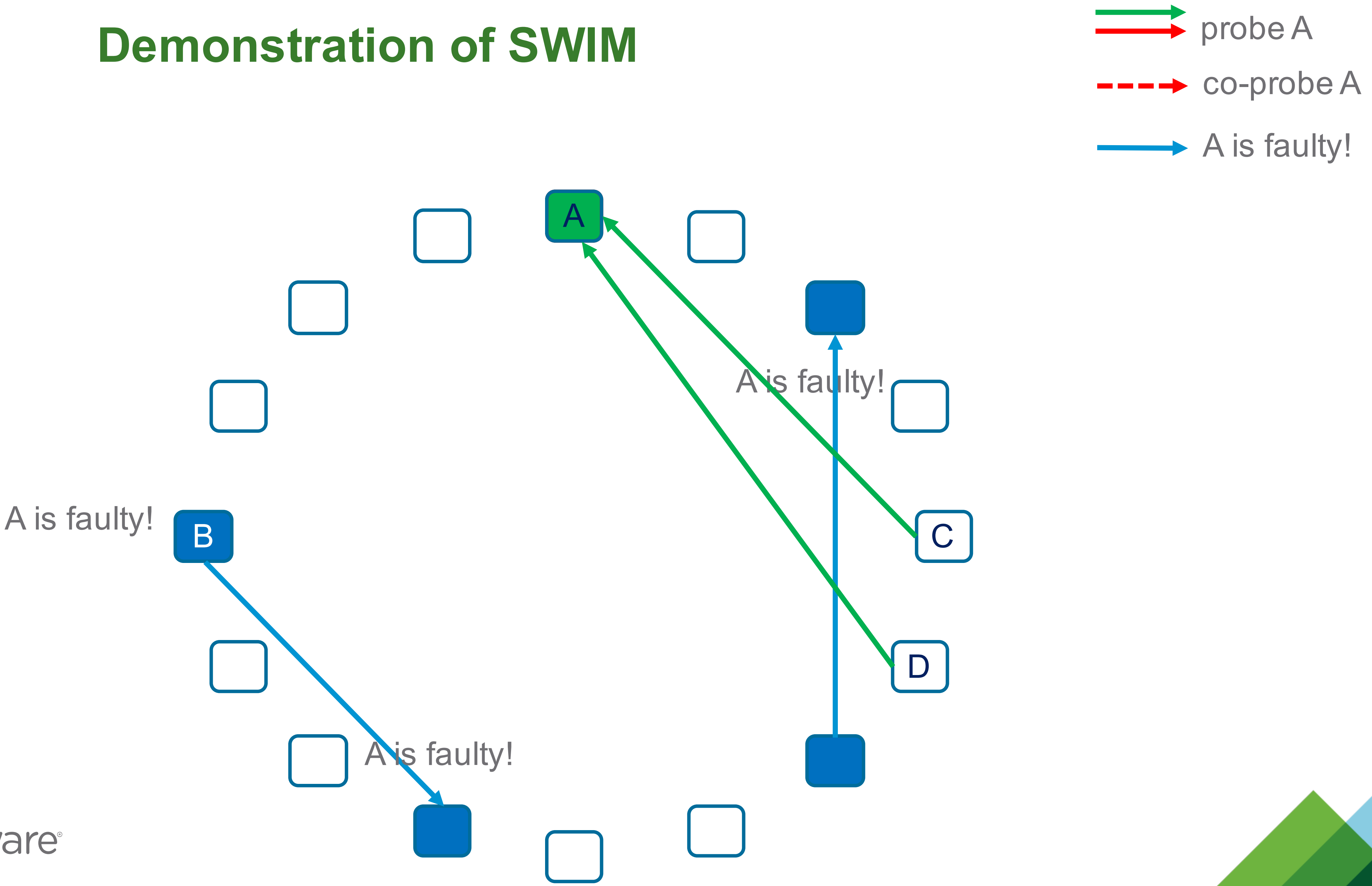
- A failure is detected in expected one SWIM round
  - with non-negligible probability, only in *log n* !
- A failure detection alert is disseminated in O(*log n*) rounds
  - half of the system will learn only after in O(*log n*) rounds
- The same failure will cause repeated co-probing and detection

- Is randomized probing better than
  - Steady heartbeat (e.g., along a ring) ?
  - Stable connection (e.g., TCP/IP) ?
  - Leases ?

Demonstration of SWIM

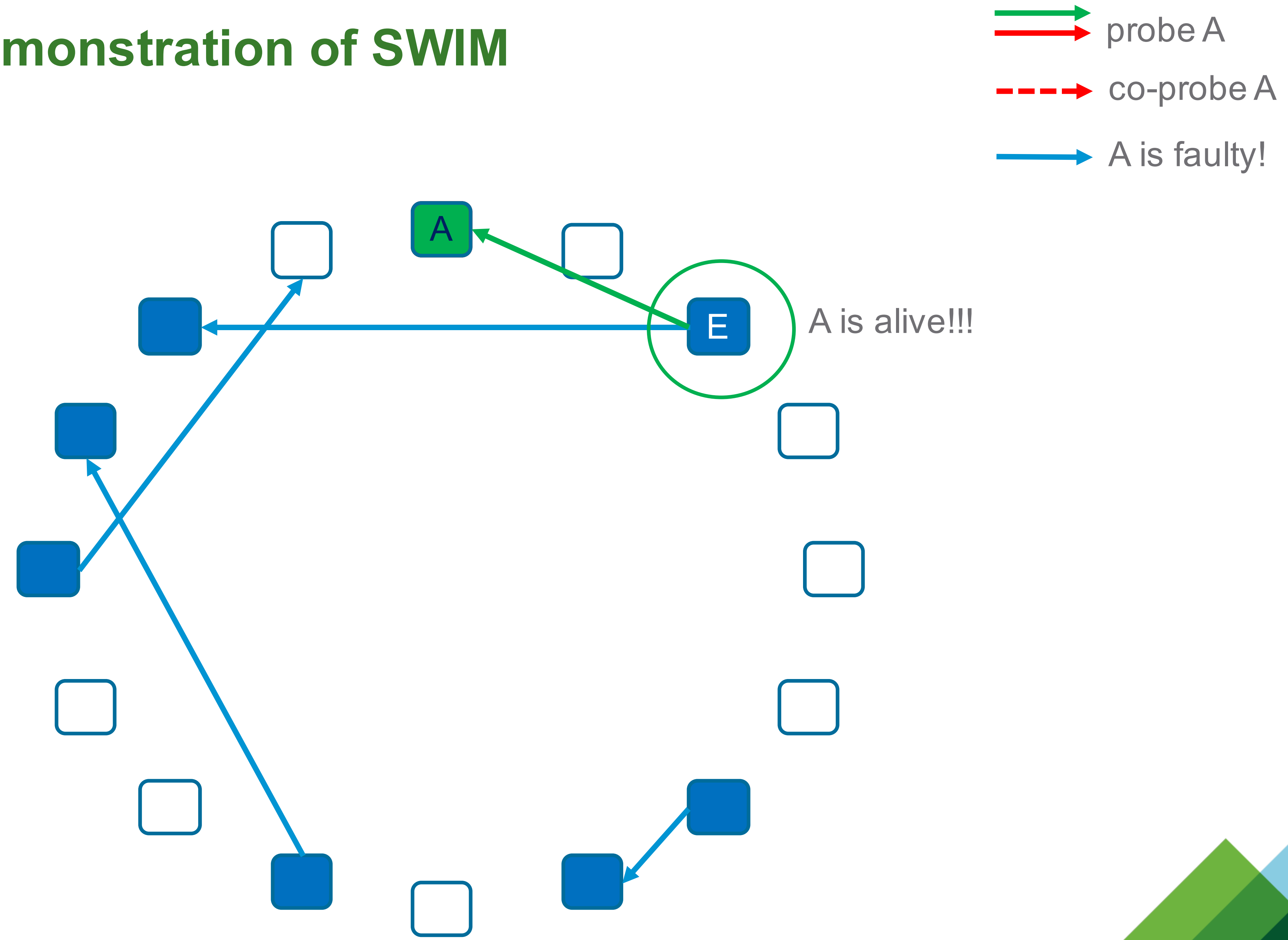


Demonstration of SWIM



vmware

Demonstration of SWIM

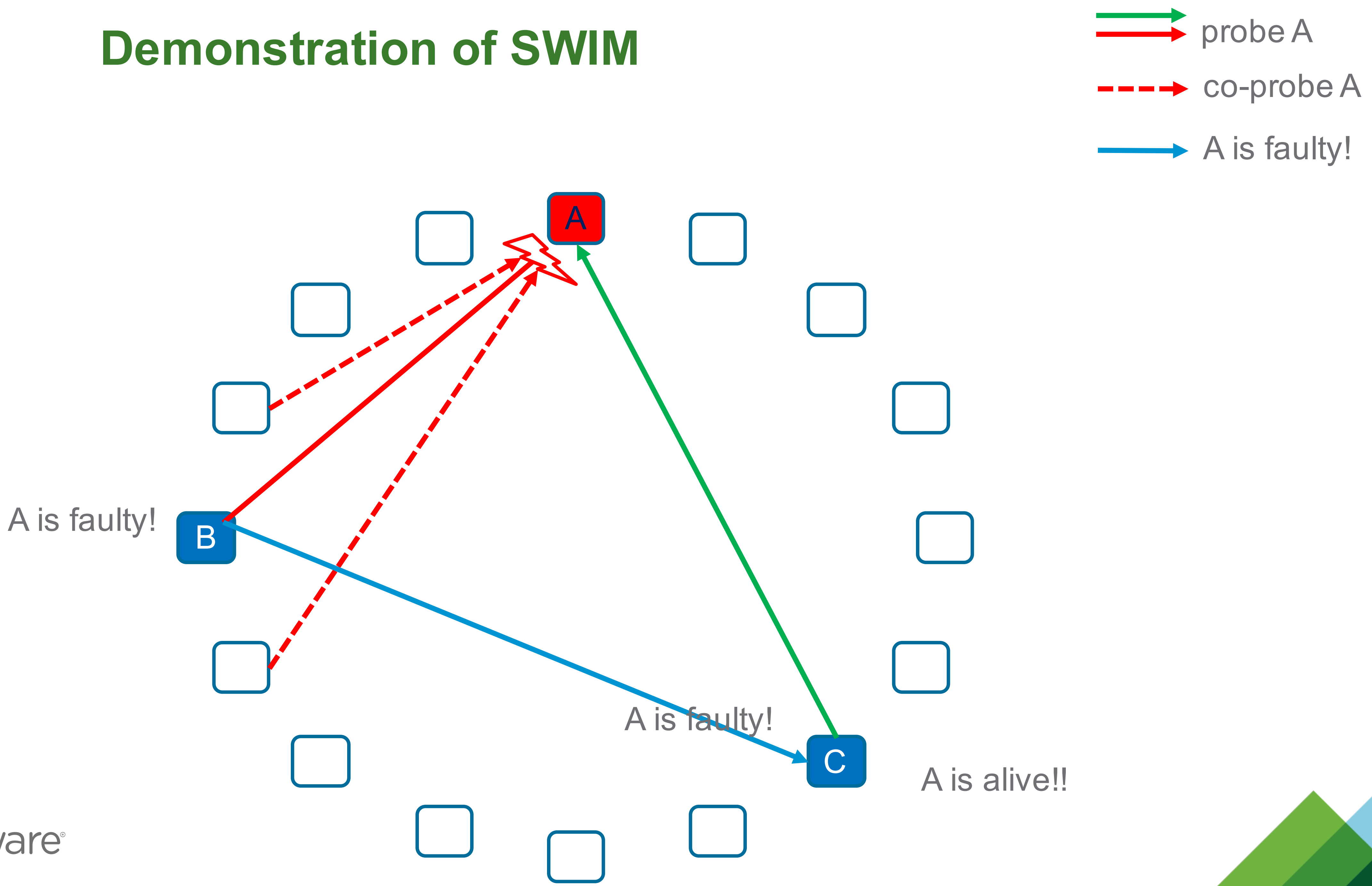


vmware

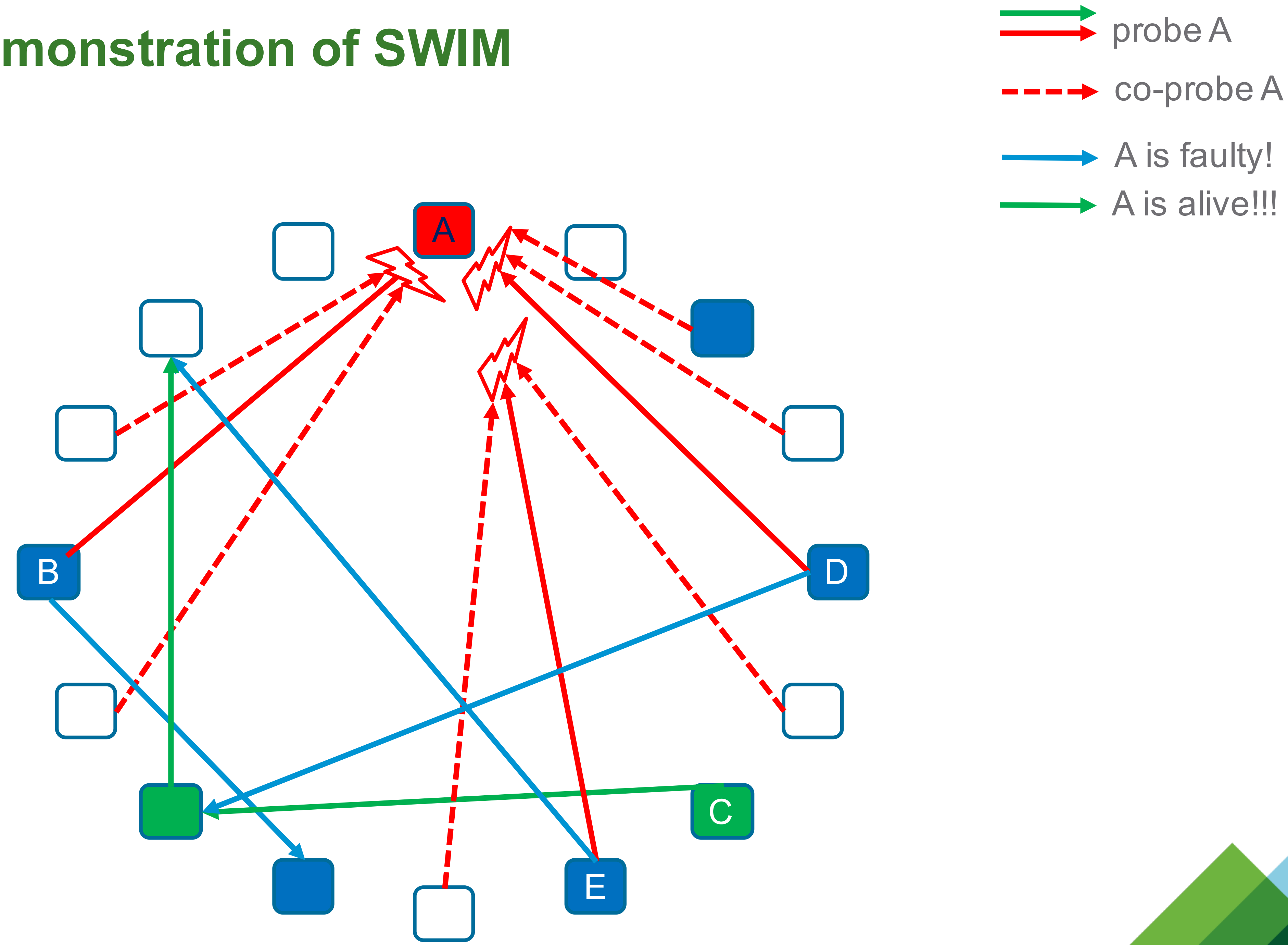
False Failure Detection in SWIM

- Suspicion – Alive -- Confirmation
- Competing gossips may arrive in arbitrary order
- Node incarnation
  - incremented by suspicion
  - suspicion overridden by alive overridden by confirmation

Demonstration of SWIM



Demonstration of SWIM



SWIM Patches

- Suspicion – Alive -- Confirmation
- Incarnations
- Deterministic probing
- Deterministic gossiping
- Weak membership service

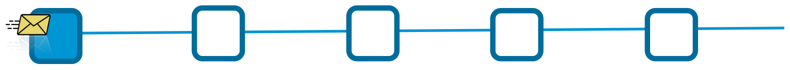
vmware

Gossip in Arbitrary Graphs

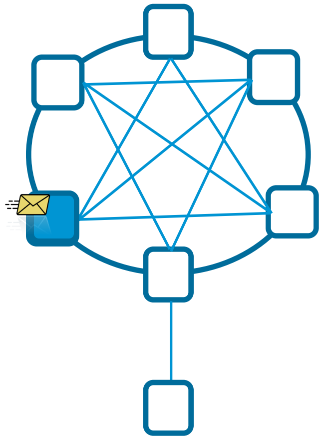
With and Without Network Discovery

Gossip in Arbitrary Graphs

- clearly not always $O(\log(n))$



- might even be super-linear

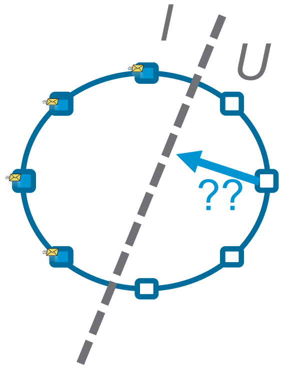


- $O(\log n/\Phi)$ rounds to completion [Giakkoupis STACS 2011]

Gossip Process in Arbitrary Graphs

- $I \stackrel{\text{def}}{=}$ set of informed vertices at beginning of round
- $U \stackrel{\text{def}}{=}$ uninformed vertices
- using PULL, each uniformed vertex v in U
  - has $deg_I(v)$ informed neighbors
  - has $deg(v)$ neighbors
  - learns with probability $(deg_I(v)/\deg(v))$
  - increases volume of edges in I by expected

$$\deg(v) * (deg_I(v)/\deg(v))$$



- total increase in edge-volume of I equals |E(I, U)|, the volume of edge-cut:

$$\sum_{v\ in\ U} deg_I(v)$$

vmware

Gossip Process in Arbitrary Graphs

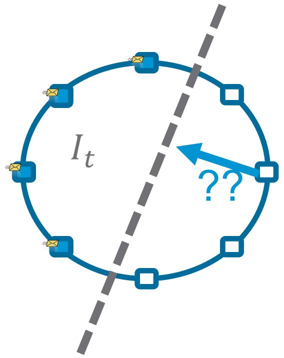
by definition, edge-cut is larger than graph's conductance

$$\Phi \overset{\text{def}}{=} \min_{S \subseteq V} \frac{E(S, V-S)}{vol(V)}$$

this means

$$\text{vol}(I_{t+1}) \geq \text{vol}(I_t) + E(I_t, U_t) \geq \text{vol}(I_t) * (1+\Phi)$$

finish in $O(\log n / \Phi)$



Network Discovery

Resource Discovery in Distributed Networks

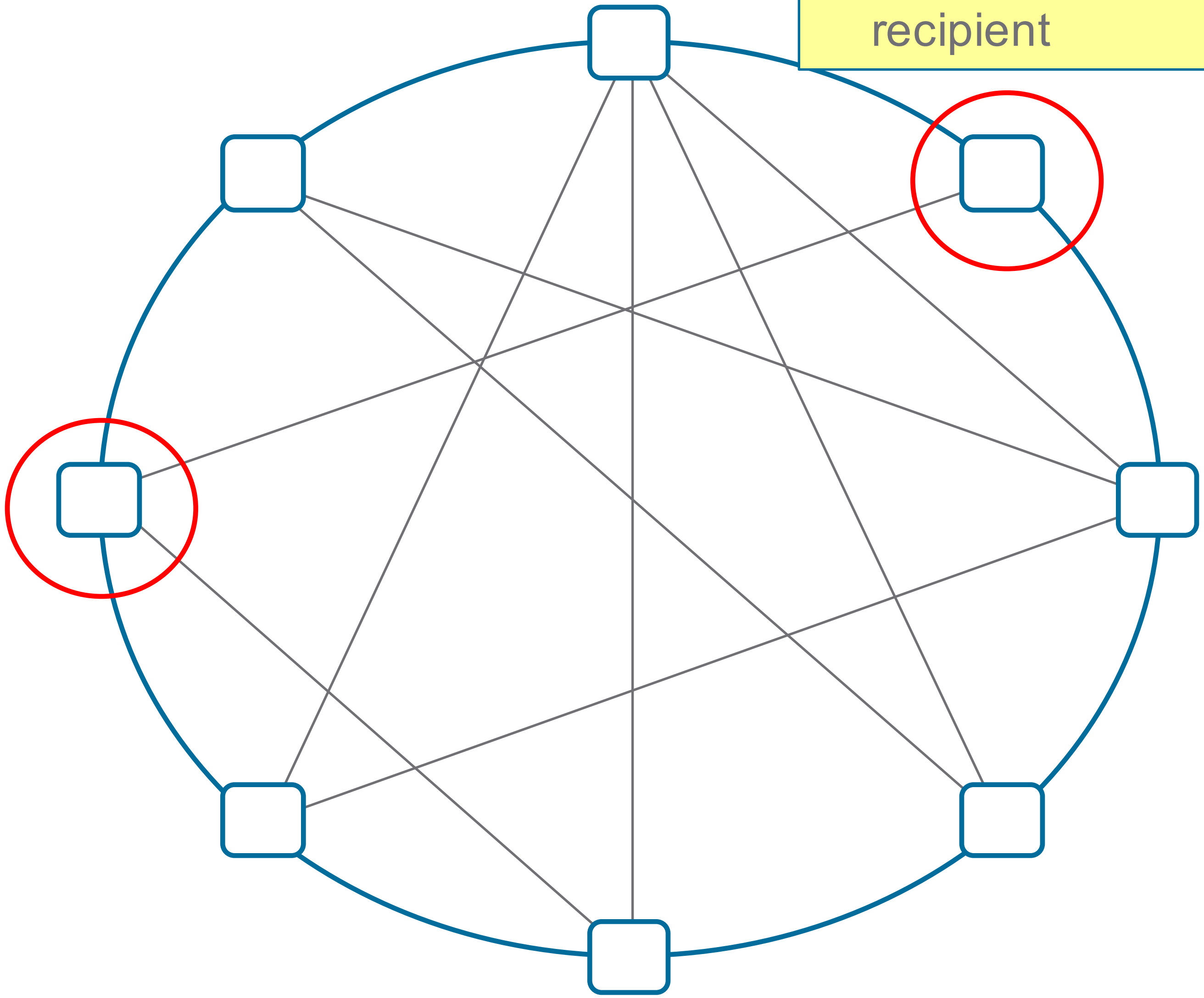
Mor Harchol-Balter* Tom Leighton† Daniel Lewin‡

1999, PODC

erate in caching information. In order for these machines to cooperate they must first locate each other, which is where the Name-Dropper resource discovery algorithm is used. The Name-Dropper algorithm has been licensed to an LCS startup company, Akamai Technologies, which is building an Internet-wide content-distribution system.

Definition of Network Discovery Problem

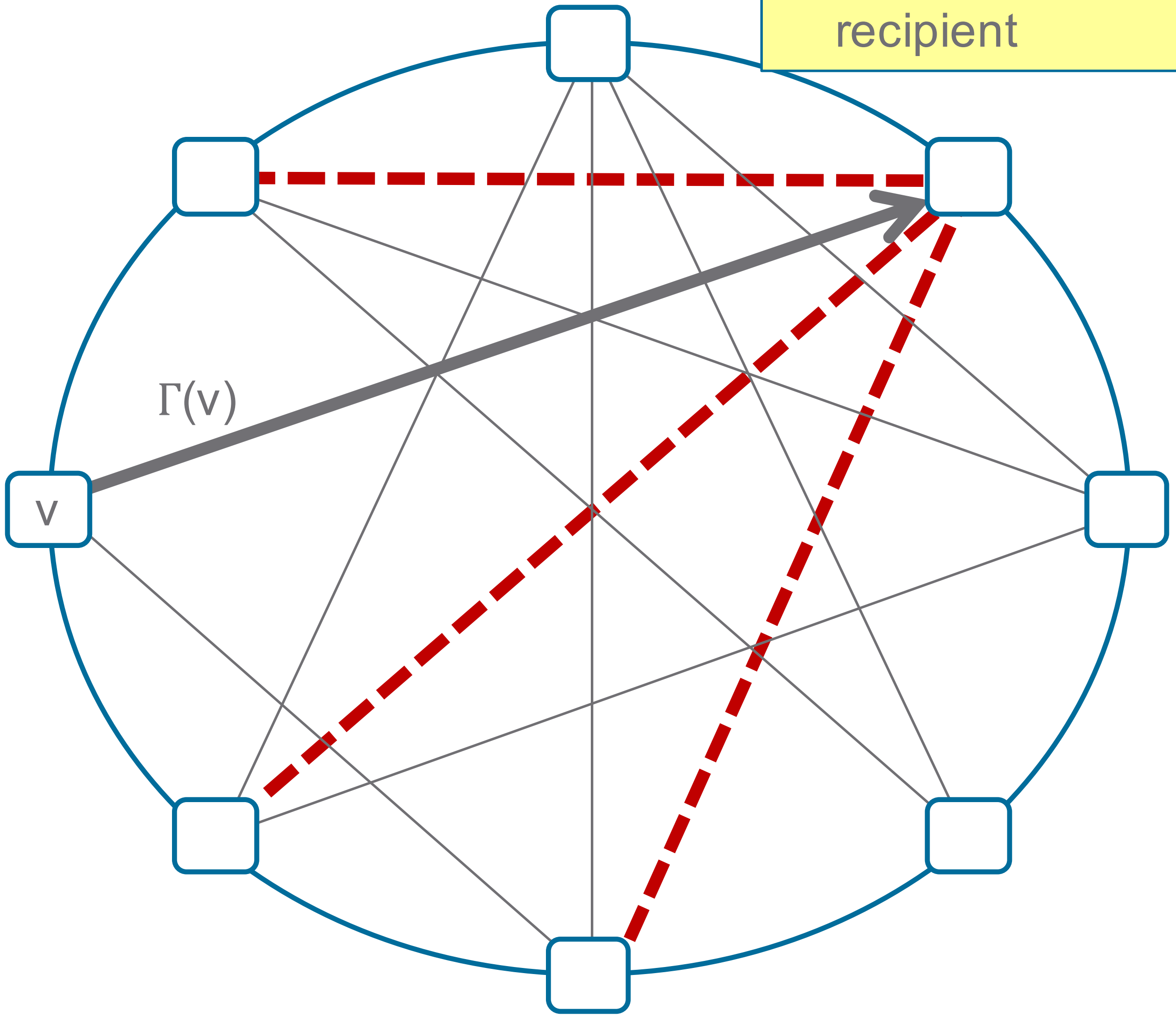
- send information about neighbors
- they become new neighbors of recipient



vmware

Definition of Network Discovery Problem

- send information about neighbors
- they become new neighbors of recipient

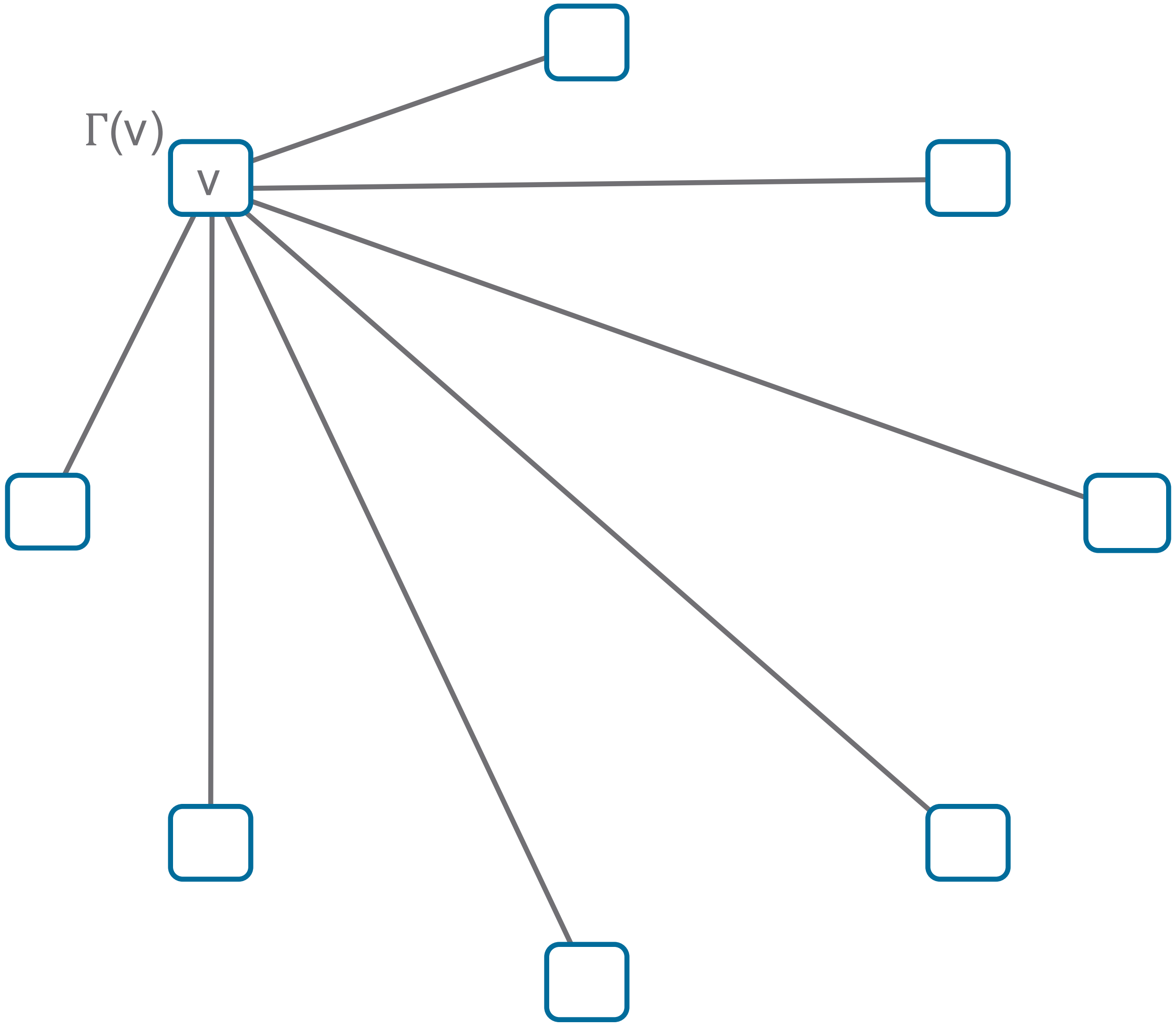


vmware

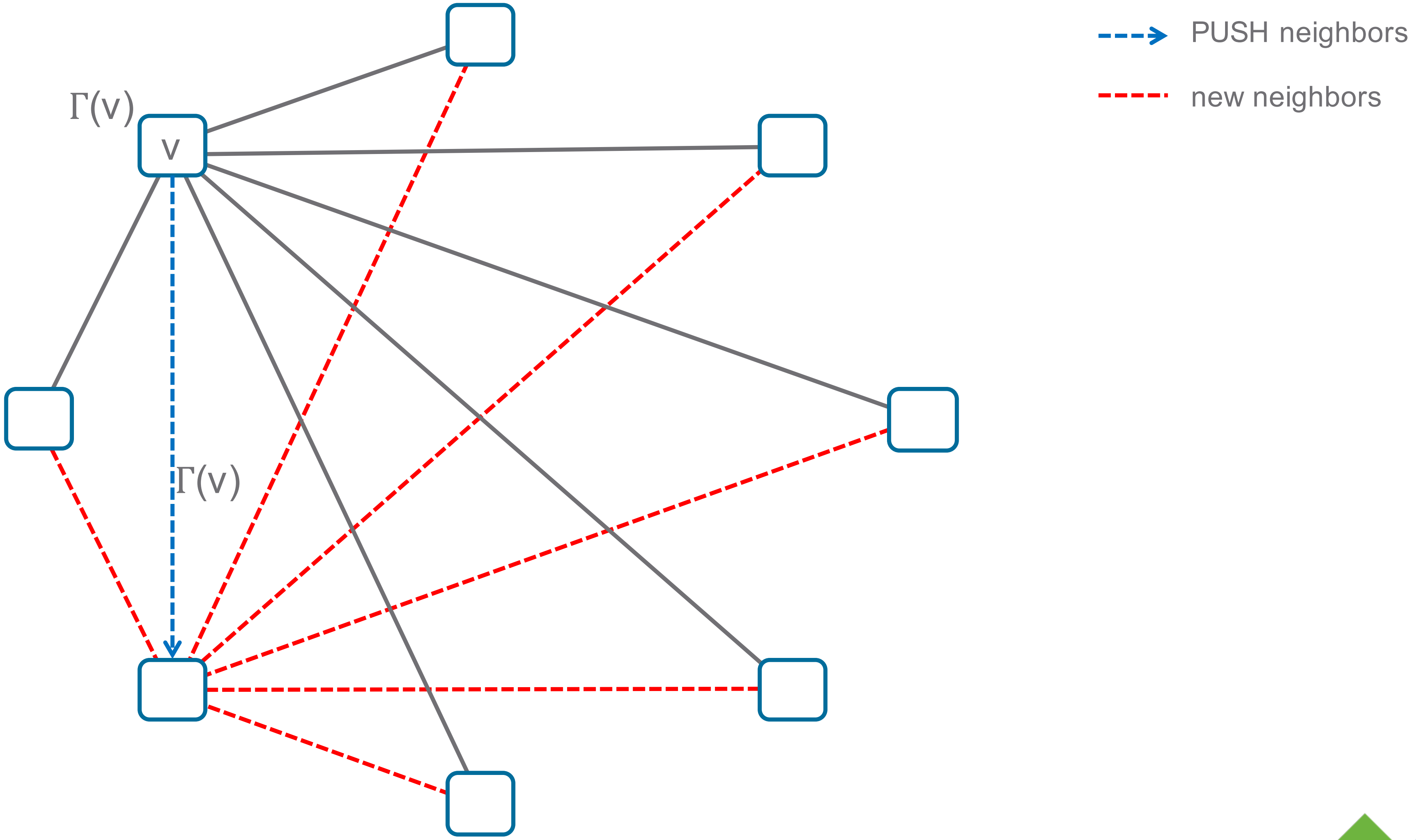
Complexity of Network Discovery

- Flood: send information about new neighbors to all initial neighbors
  - completes in diameter rounds
  - sends *diameter x degree x n* messages
- Swamp: send information about all neighbors to all neighbors
  - completes in O(*log n*) rounds
  - sends O(*n x n*) large messages
- Pointer jump: PULL from one random neighbor information about all its neighbors
  - may take O(*n*) rounds
- Name Dropper: PUSH to one random neighbor information about all neighbors
  - completes in O(log^2 *n*) rounds

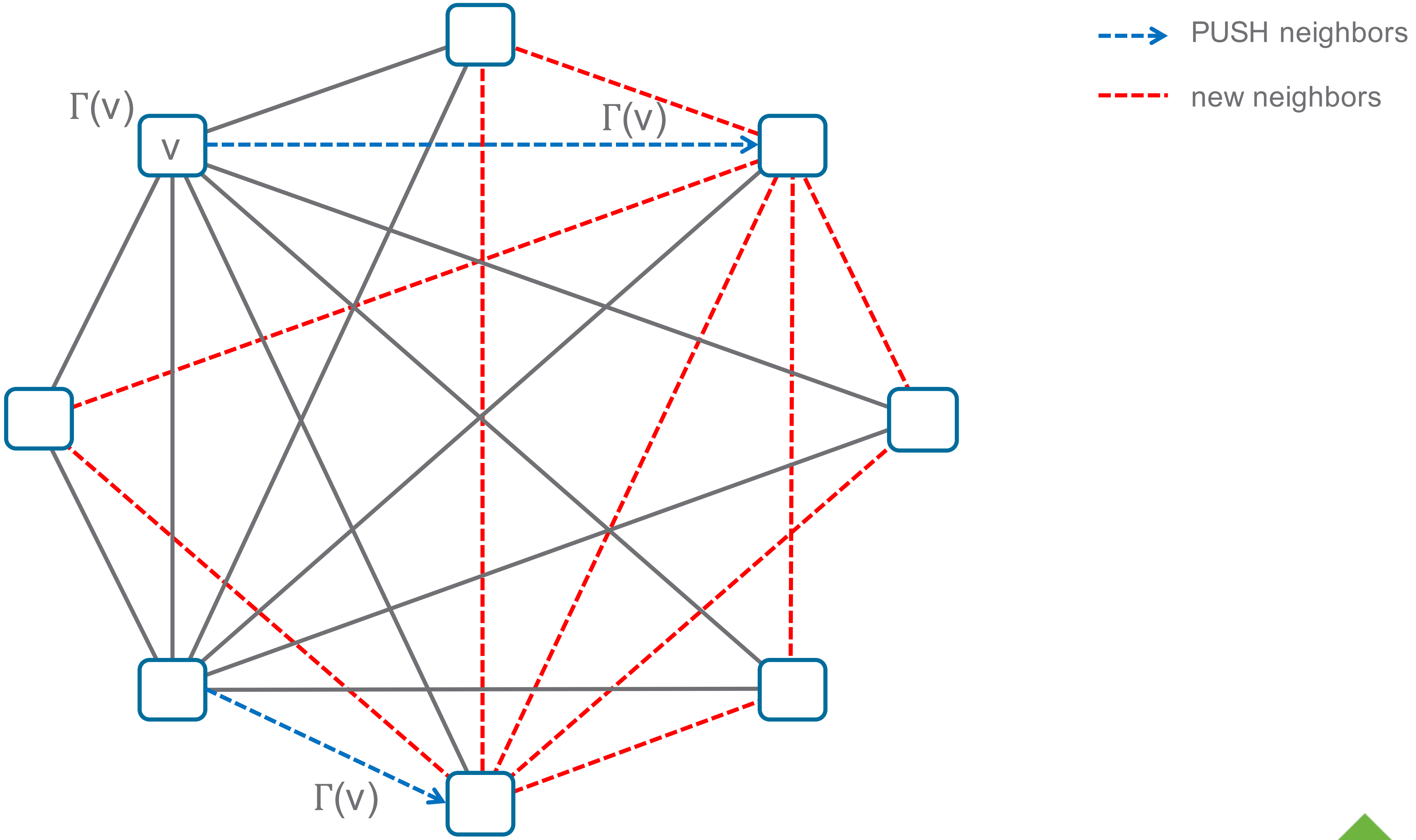
Demonstration of Network Discovery



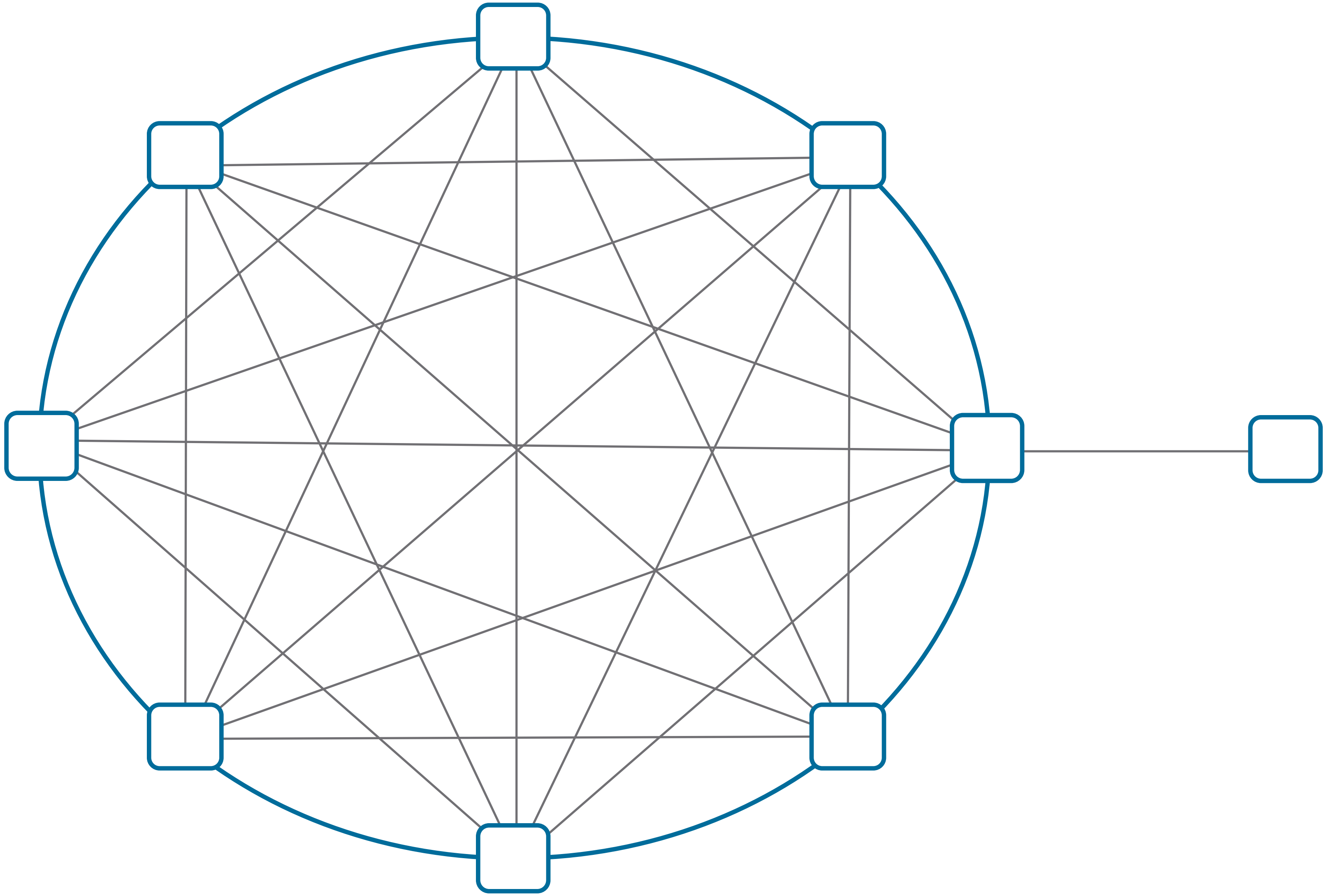
Demonstration of Network Discovery



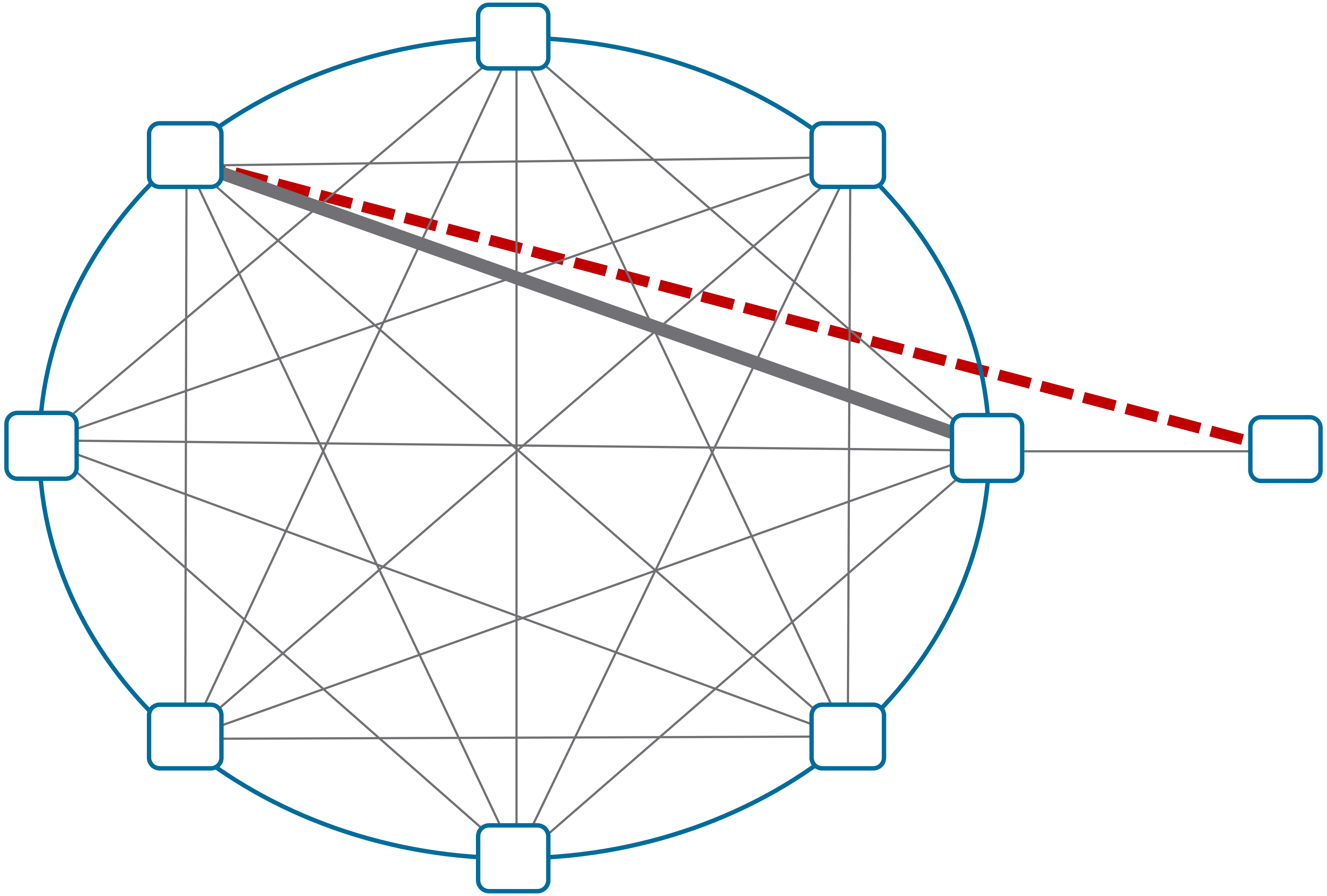
Demonstration of Network Discovery



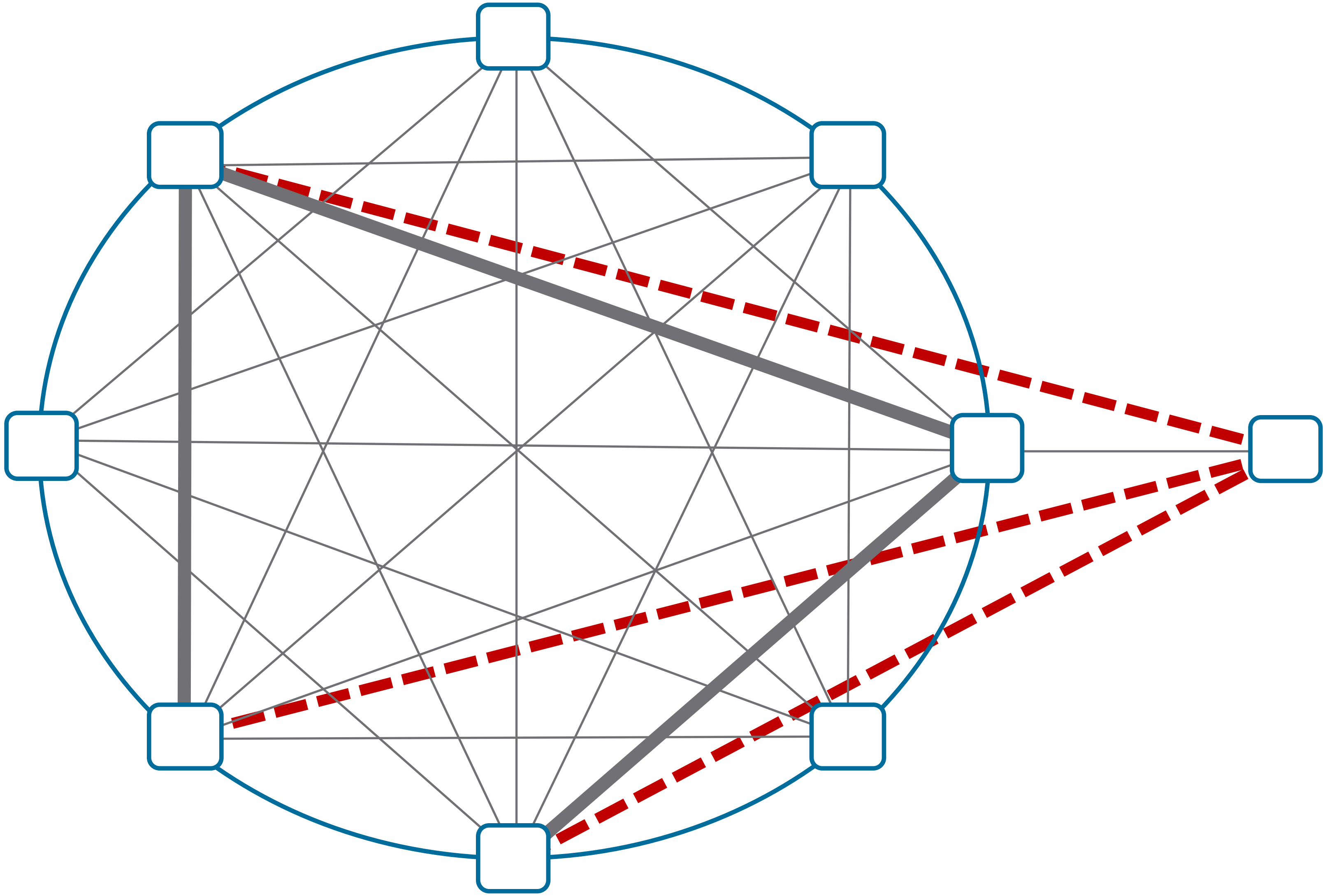
Demonstration of Network Discovery (2)



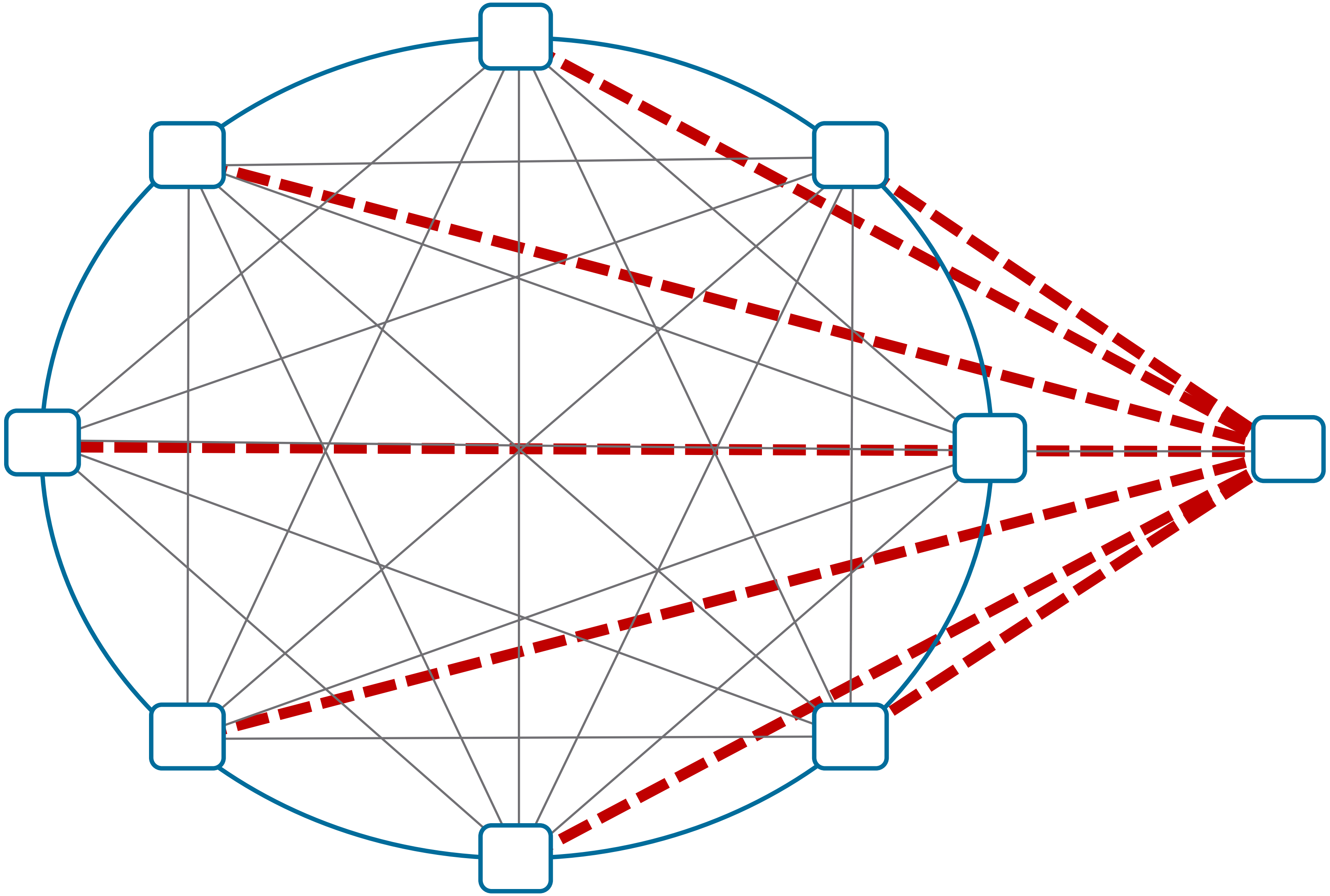
Demonstration of Network Discovery (2)



Demonstration of Network Discovery (2)

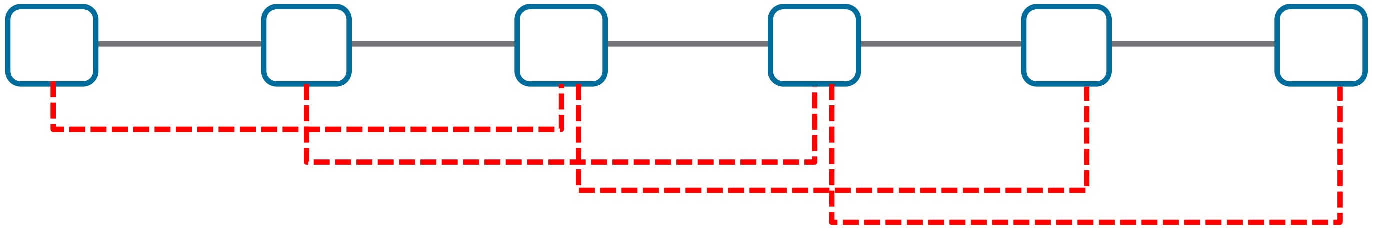


Demonstration of Network Discovery (2)

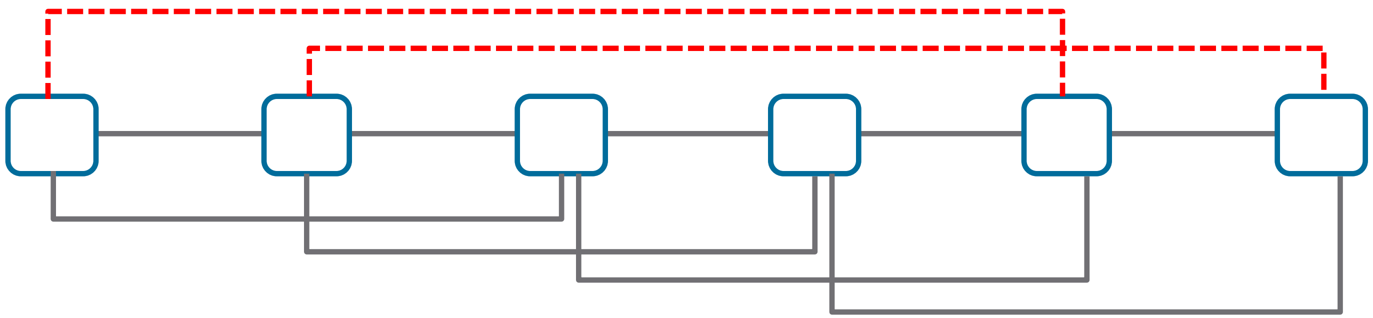


Demonstration of Network Discovery (3)

Demonstration of Network Discovery (3)



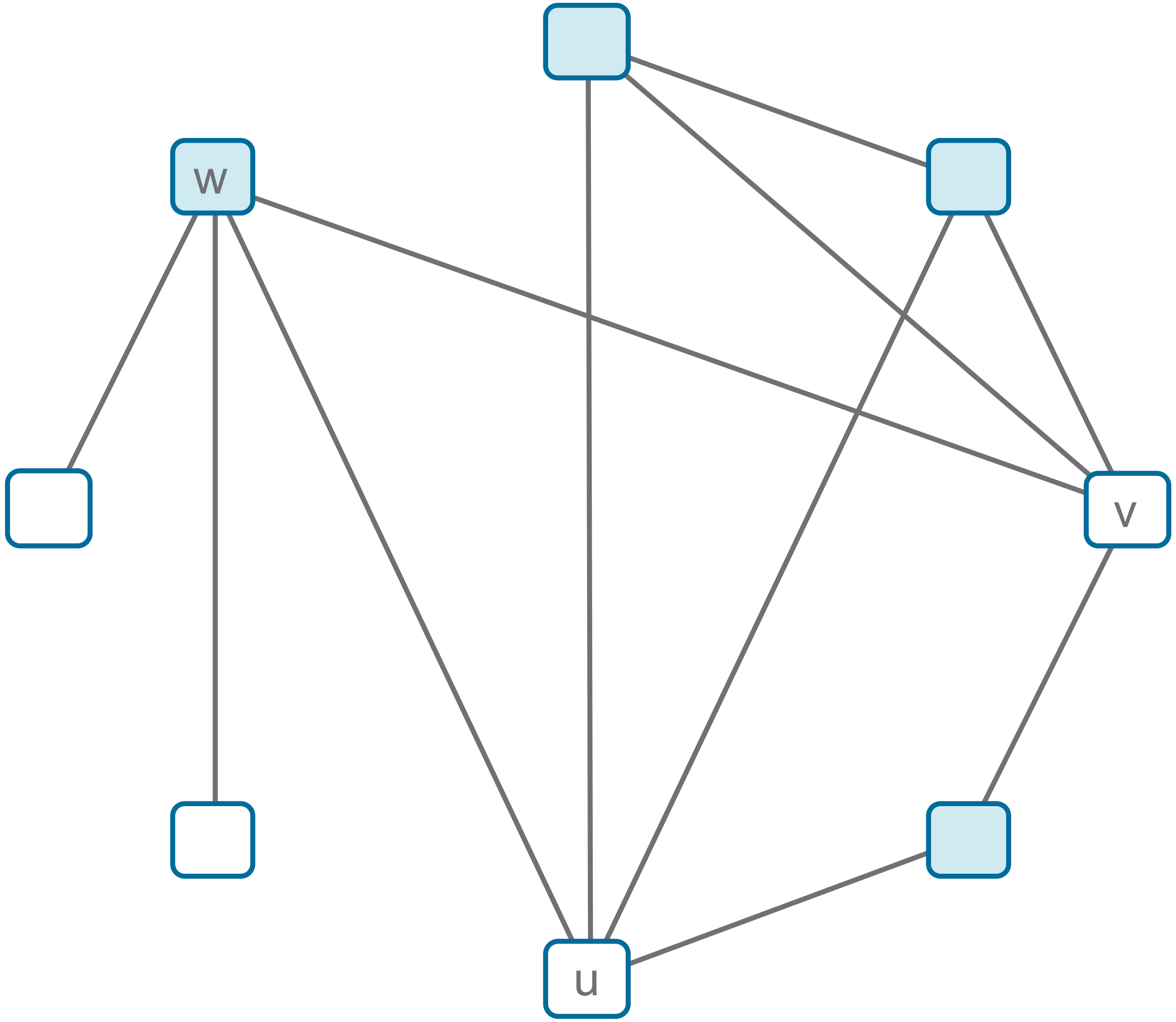
Demonstration of Network Discovery (3)



Analyzing Name Dropper

X

- for every path u – w – v
  - X: the set of neighbors of v, u
  - with constant probability
    - Either X grows by constant factor
      - if half of X has 2|X| neighbors
    - or some node in X contacts u



Complexity of Network Discovery

- Name-Dropper
  [Harchol-Balter, Leighton, Lewin, PODC 1999]:
  - O(*log n log D*) rounds
  - Arbitrary directed graphs
- Improvement
  [Kutten, Peleg and Vishkin, TCS 2003]
  - O(log n) rounds

Gossip by Network Discovery?

teaser: $O(\log n) + O(\log n) = ?$

Breaking the Logarithmic Bound

- synchronous rounds
- each node initiates one connection
- choose partners at random

Breaking the Logarithmic Bound

- synchronous rounds

- each node initiates one connection
  - number of informed nodes at most doubles each round
  - a ha! but a node may **respond** to any number of requests!
- choose partners at random

Breaking the Logarithmic Bound

- synchronous rounds

- each node initiates one connection
  - number of informed nodes at most doubles each round
  - a ha! but a node may **respond** to any number of requests!
- choose partners at random
  - graph of all interactions has $O(log(n))$ diameter
  - a ha! allow nodes to learn addresses and use them (e.g., TCP/IP)

A Hybrid Gossip Model

- synchronous rounds
- each node initiates one connection
- choose partners at random or by direct addressing
- learn addresses from interaction

Breaking the Logarithmic Bound

- GOSSIP with DA in $O(\sqrt{\log n})$ rounds [Avin and Elsässer, DISC 2013]
- Improved to $O(\log\log n)$ rounds [Haeupler and Malkhi, PODC 2014]
- Use for Network Discovery
  - choose a leader in $O(\log\log n)$ rounds
  - One push/pull via leader to share topology information
  - $O(\log D \log\log n)$ rounds [Haeupler and Malkhi, PODC 2015]


questions?