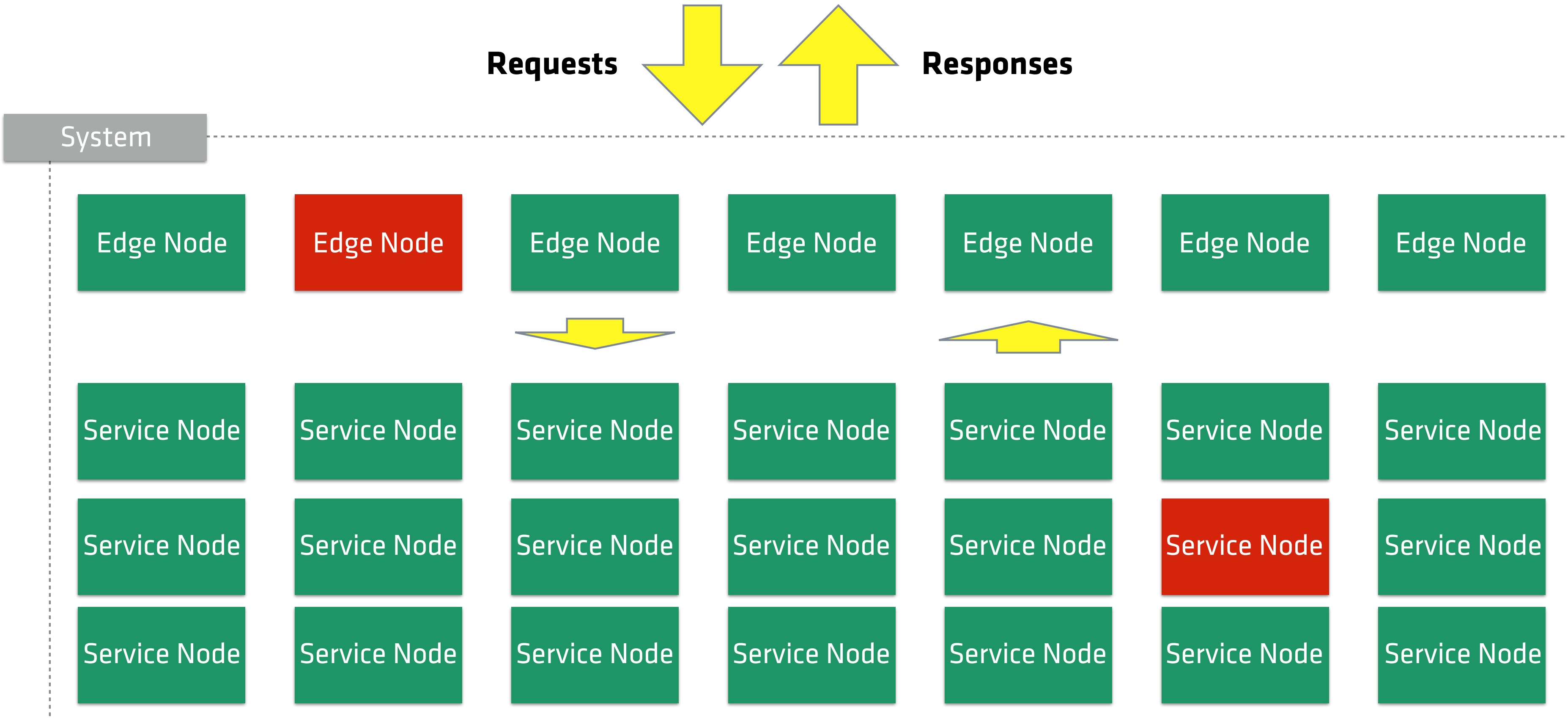
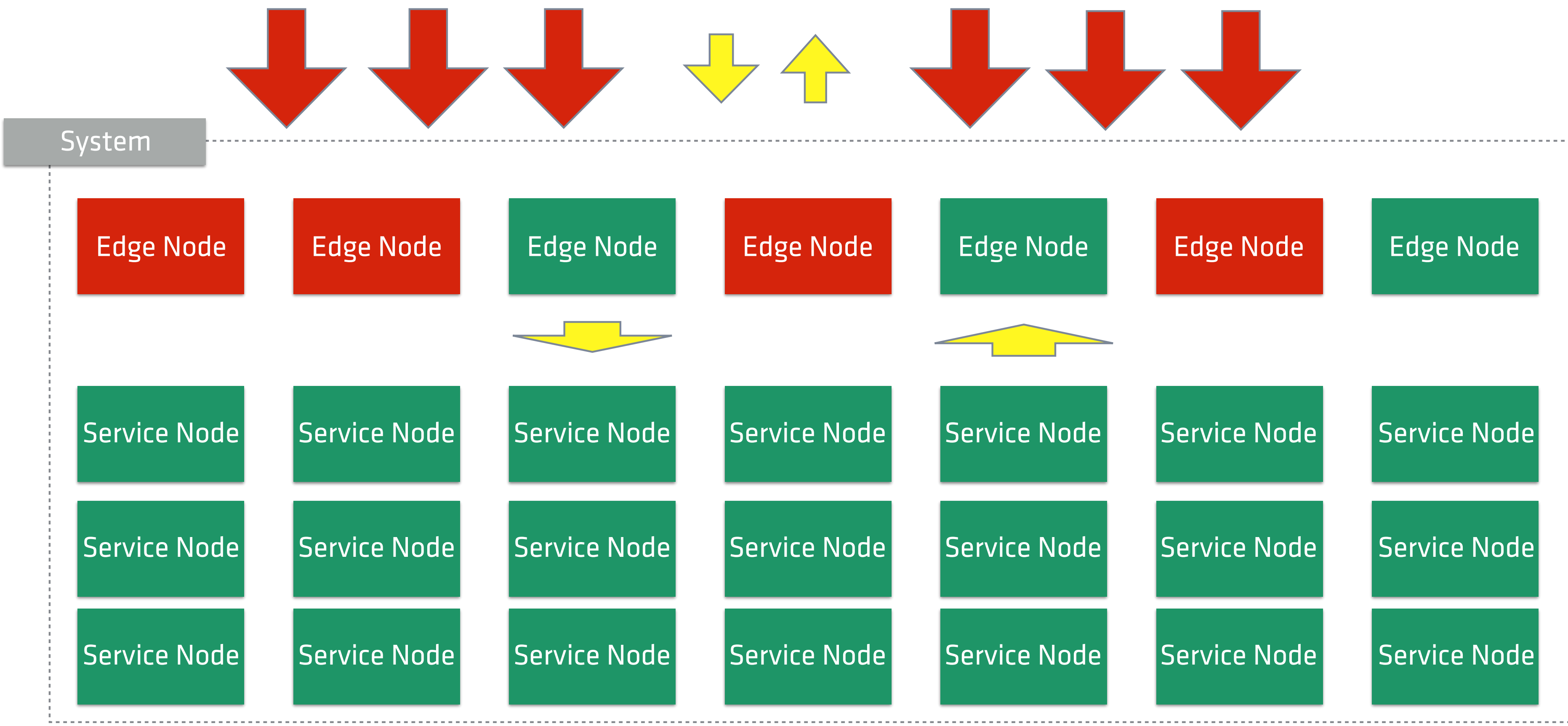
Using Randomized Communication for Robust, Scalable Systems

Jon Currey, HashiCorp

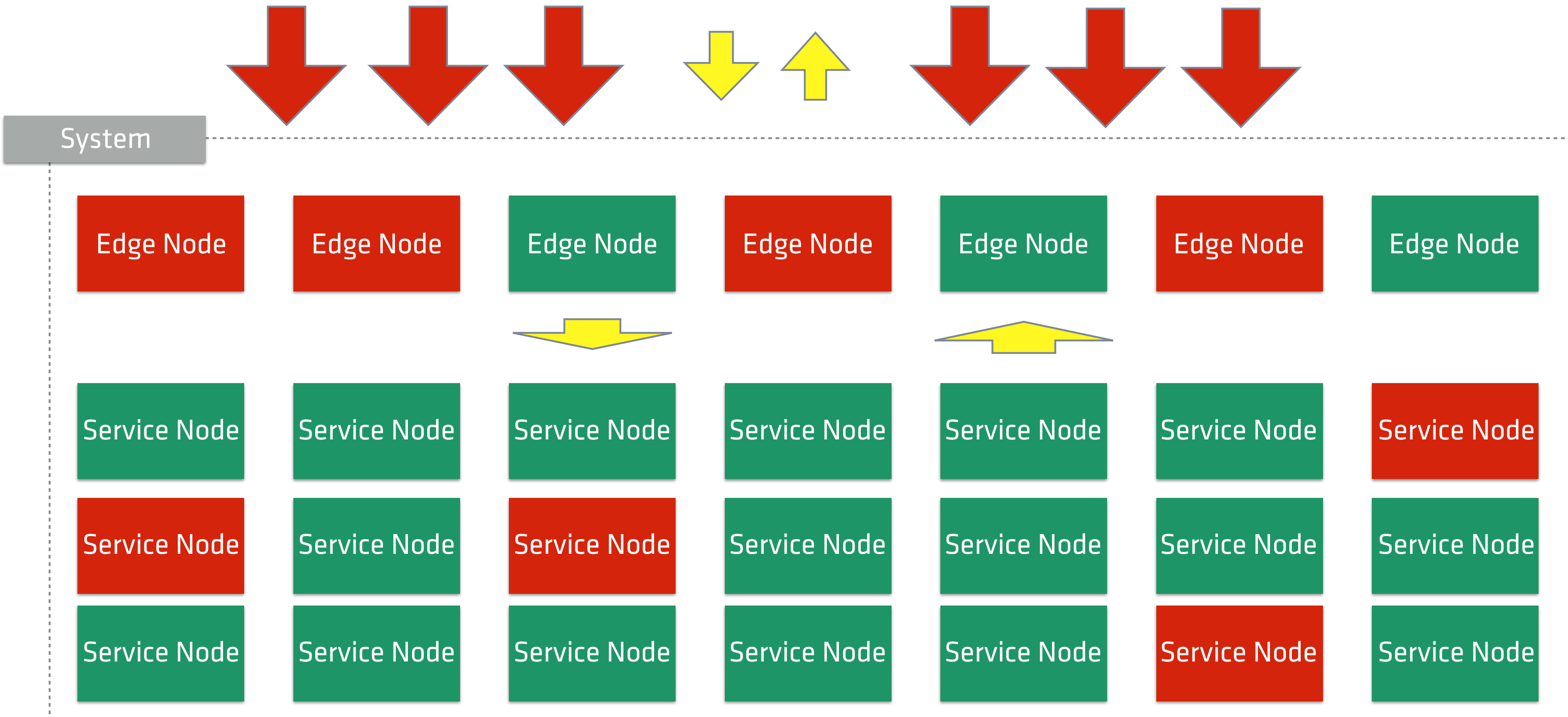
Service Discovery and Failure Detection



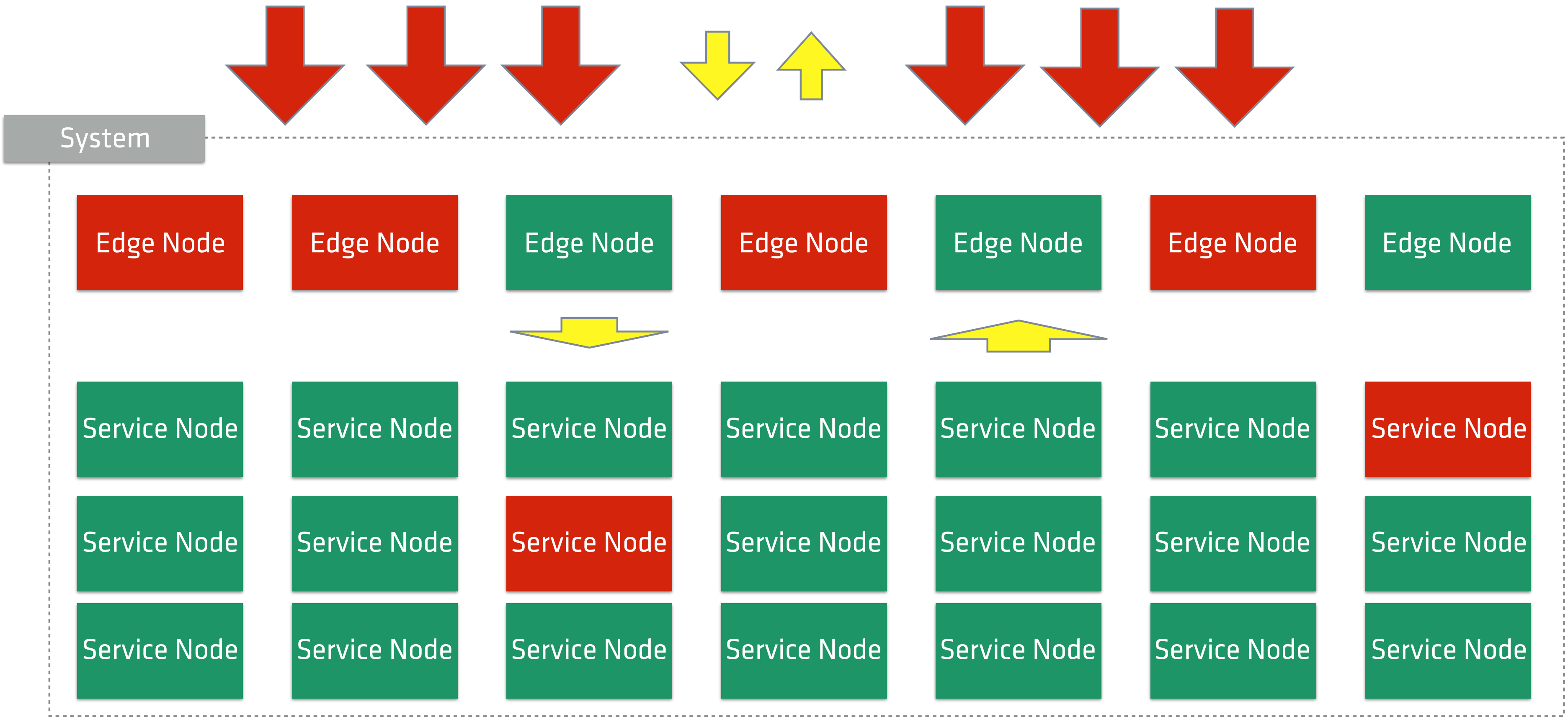
DDoS Attack


System
Edge Node
Edge Node
Edge Node
Edge Node
Edge Node
Edge Node
Edge Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node
Service Node

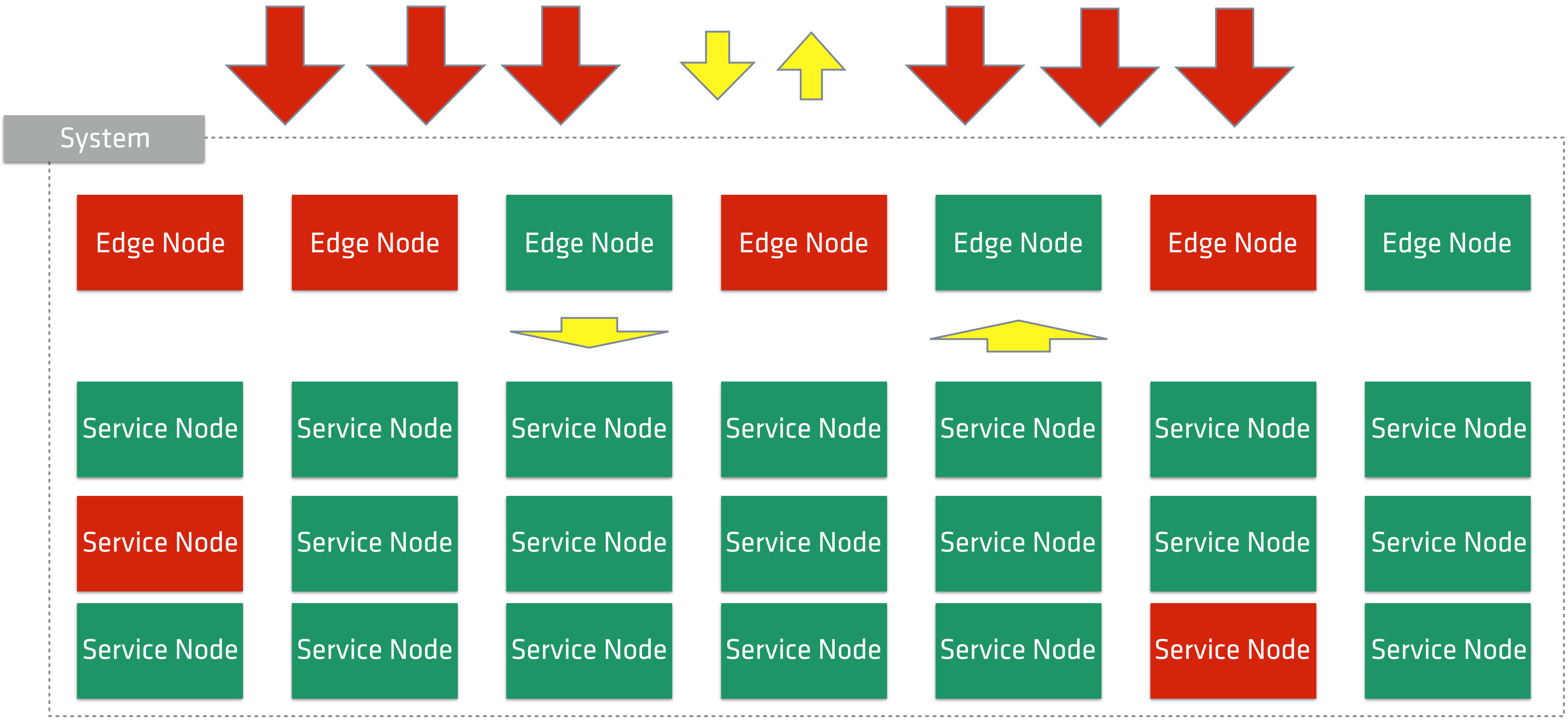
Unexpected Behaviour



Unexpected Behaviour

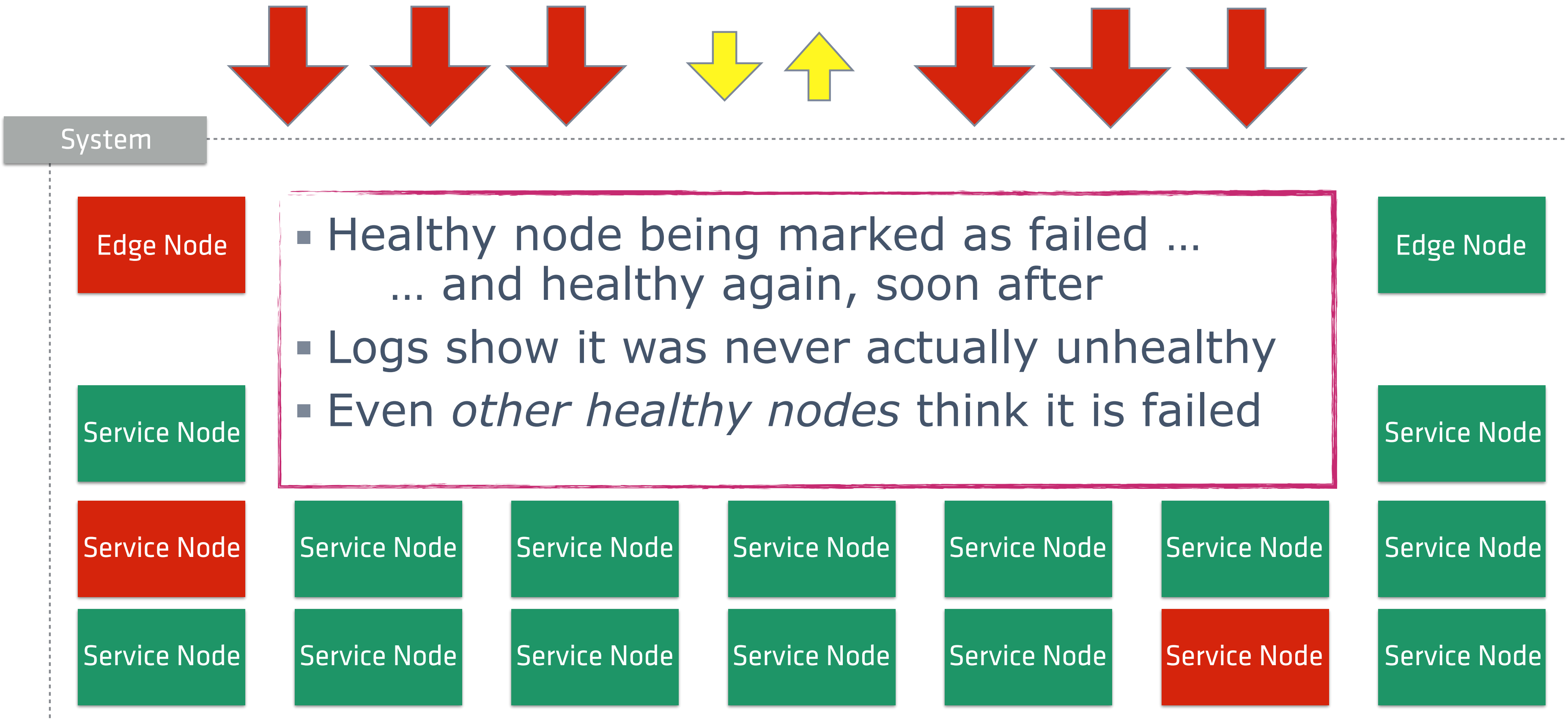


Unexpected Behaviour

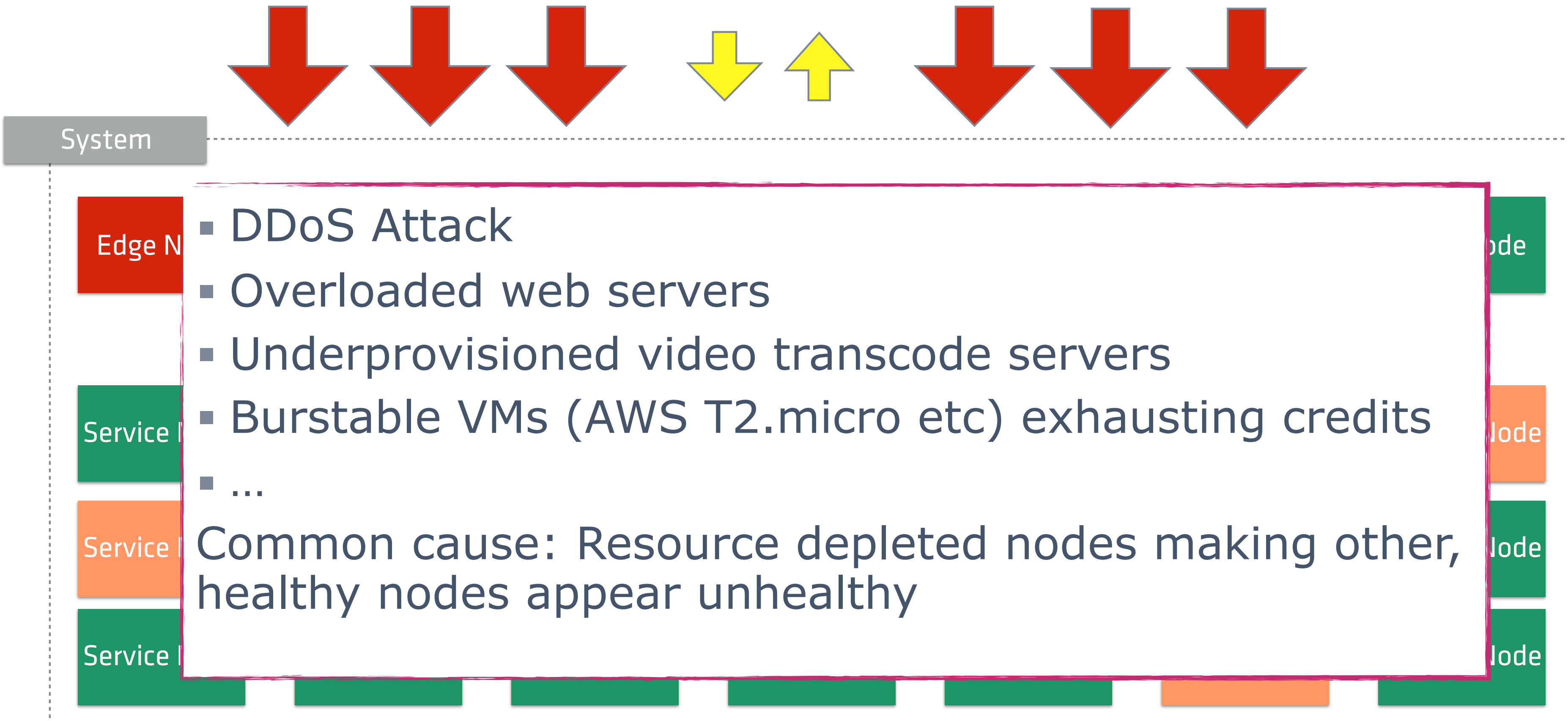


‘Flapping’ Nodes

- Healthy node being marked as failed …
  … and healthy again, soon after
- Logs show it was never actually unhealthy
- Even *other healthy nodes* think it is failed



Many Scenarios ... Common Cause



- DDoS Attack
- Overloaded web servers
- Underprovisioned video transcode servers
- Burstable VMs (AWS T2.micro etc) exhausting credits
- ...

Common cause: Resource depleted nodes making other, healthy nodes appear unhealthy

Agenda

Using randomization to build robust and scalable systems

Agenda + Meta-Agenda

Using randomization to build robust and scalable systems

Leveraging academic research in production systems

Jon Currey - Industrial Researcher

Google Scholar

Jon Currey

HashiCorp Research
Verified email at hashicorp.com
Distributed Systems and M…

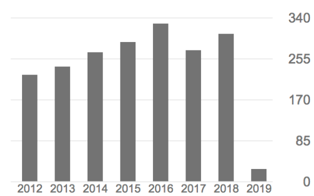
FOLLOW

TITLE	CITED BY	YEAR
DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language YYMID Fetterly, M Budiu, Ú Erlingsson, PKGJ Currey Proc. LSDS-IR 8	891	2009
Quincy: fair scheduling for distributed computing clusters M Isard, V Prabhakaran, J Currey, U Wieder, K Talwar, A Goldberg Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles …	830	2009
PTask: operating system abstractions to manage GPUs as compute devices CJ Rossbach, J Currey, M Silberstein, B Ray, E Witchel Proceedings of the Twenty-Third ACM Symposium on Operating Systems …	222	2011
An introduction to computational networks and the computational network toolkit D Yu, A Eversole, M Seltzer, K Yao, Z Huang, B Guenter, O Kuchaiev, … Microsoft Technical Report MSR-TR-2014–112	215	2014
Dandelion: a compiler and runtime for heterogeneous systems CJ Rossbach, Y Yu, J Currey, JP Martin, D Fetterly Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems …	110	2013
System and method for logging messages in an embedded computer system JJ Currey, JE Rodriquez, RM Jensen, KR Black US Patent 6,769,079	35	2004
System and method for adding transport protocols in distributed middleware applications KR Black, RM Jensen, JE Rodriguez, JJ Currey US Patent 7,010,609	17	2006
Some sample programs written in DryadLINQ Y Yu, M Isard, D Fetterly, M Budiu, Ú Erlingsson, PK Gunda, J Currey, … Tech. Rep. MSR-TR-2008-74, Microsoft Research	15	2008

GET MY OWN PROFILE

Cited by — VIEW ALL

	All	Since 2014
Citations	2384	1495
h-index	9	7
i10-index	9	5

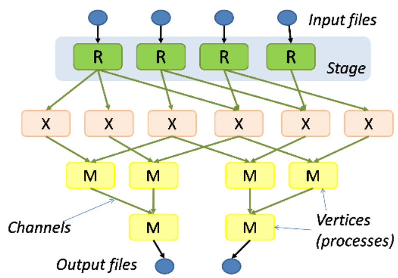


Co-authors

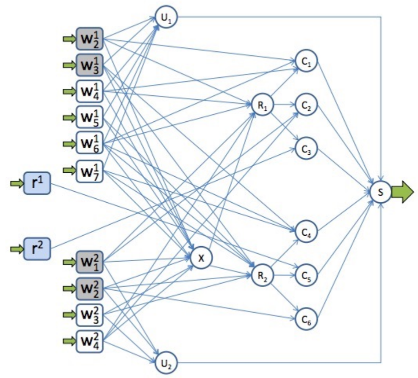
- CHRISTOPHER J. ROSSBACH — University of Texas at Austin and…
- Dennis Fetterly — Software Engineer, Google
- Úlfar Erlingsson — Research Scientist, Google Brain
- Michael Isard — Research Scientist, Google
- Andrew V. Goldberg — Senior Principal Scientist, Amaz…

Microsoft Research

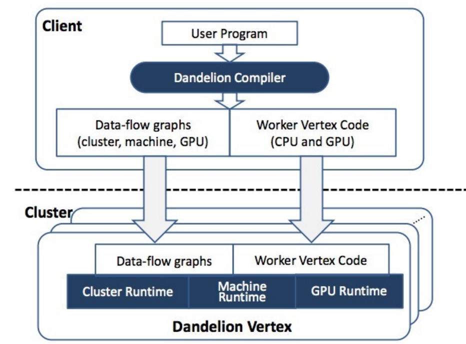
Dryad/DryadLINQ Distributed Dataflow



Quincy Scheduler



PTask/Dandelion GPU Cluster Computation



Used by

bing

Influenced

Apache Spark

TensorFlow

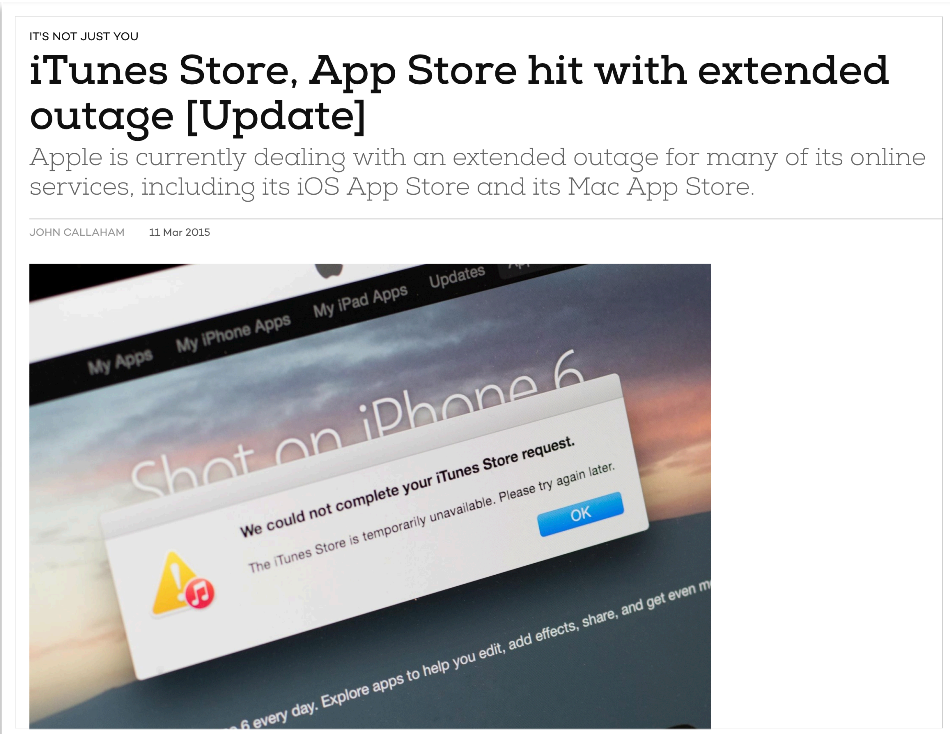
Production Engineering

ORACLE

NORTEL NETWORKS

iris financial

School of Hard Knocks

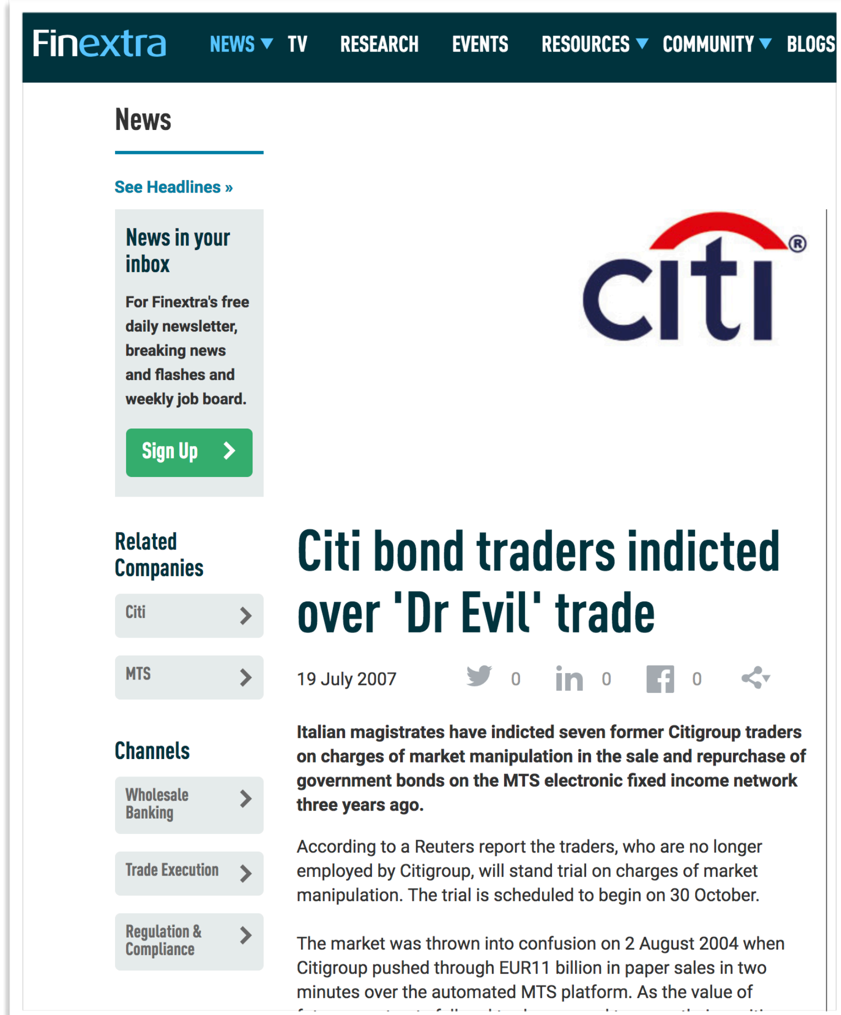


IT'S NOT JUST YOU

iTunes Store, App Store hit with extended outage [Update]

Apple is currently dealing with an extended outage for many of its online services, including its iOS App Store and its Mac App Store.

JOHN CALLAHAM 11 Mar 2015



Finextra NEWS TV RESEARCH EVENTS RESOURCES COMMUNITY BLOGS

News

See Headlines »

News in your inbox

For Finextra's free daily newsletter, breaking news and flashes and weekly job board.

Sign Up

Related Companies

Citi

MTS

Channels

Wholesale Banking

Trade Execution

Regulation & Compliance

Citi bond traders indicted over 'Dr Evil' trade

19 July 2007 0 0 0

Italian magistrates have indicted seven former Citigroup traders on charges of market manipulation in the sale and repurchase of government bonds on the MTS electronic fixed income network three years ago.

According to a Reuters report the traders, who are no longer employed by Citigroup, will stand trial on charges of market manipulation. The trial is scheduled to begin on 30 October.

The market was thrown into confusion on 2 August 2004 when Citigroup pushed through EUR11 billion in paper sales in two minutes over the automated MTS platform. As the value of

HashiCorp

HashiCorp Vagrant

HashiCorp Packer

HashiCorp Terraform

HashiCorp Vault

HashiCorp Nomad

HashiCorp Consul

Consul, Serf ... and memberlist too

hashicorp / **memberlist**

- Consul
  - KV store
  - *Strong consistency* via Raft protocol
  - mesh networking ...
- Serf
  - **Service discovery (*weakly consistent*)**
  - **Service and node health checks**
  - **Network distance**
  - ...
- memberlist
  - **Group membership**
  - **Failure detection**

Discovery and Failure-Detection Requirements

- **Robust**
  - To both node and network failures
- **Scalable**
- **Simple**
  - Easier to implement >> Less likelihood of bugs
  - Easier to manage >> Less likelihood of misconfiguration

Product Requirements and Research

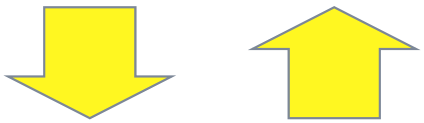
Product requirements

↓

Criteria for research discovery and evaluation

Product Requirements and Research

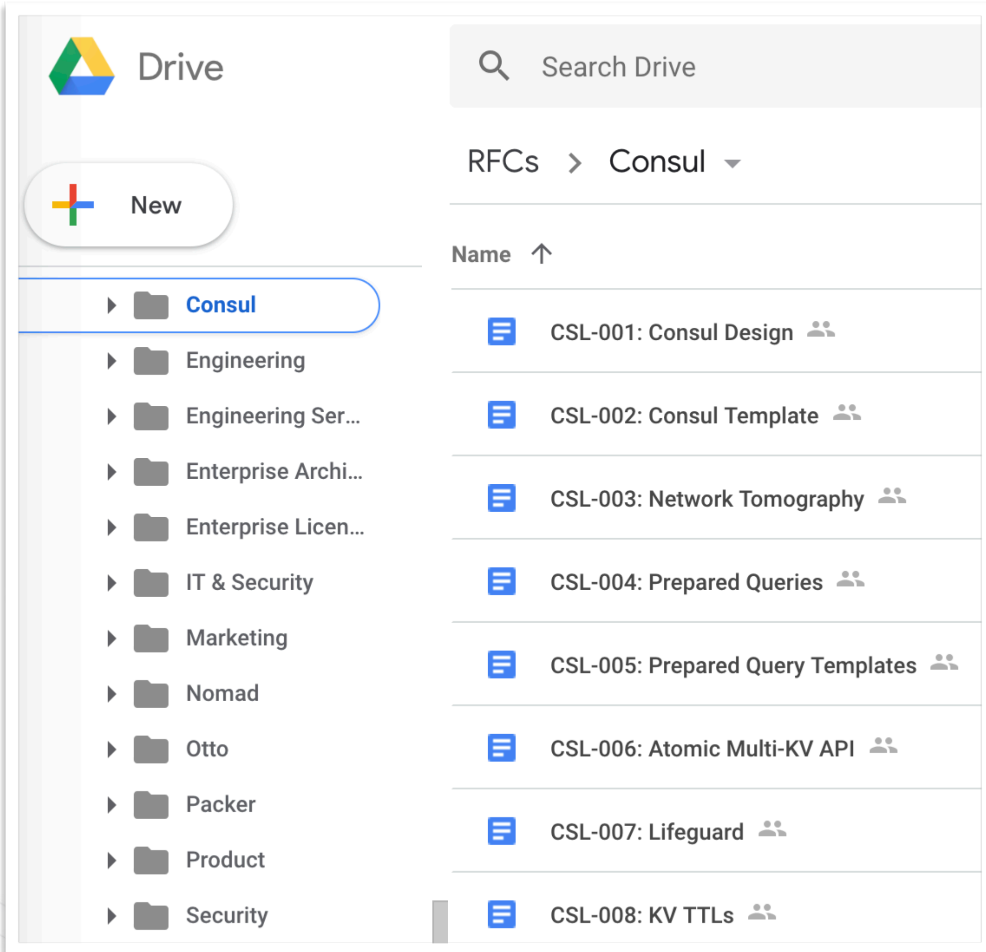
Product requirements



Criteria for research discovery and evaluation

Consuming research will inform your product decisions

Research-Aware Design Process

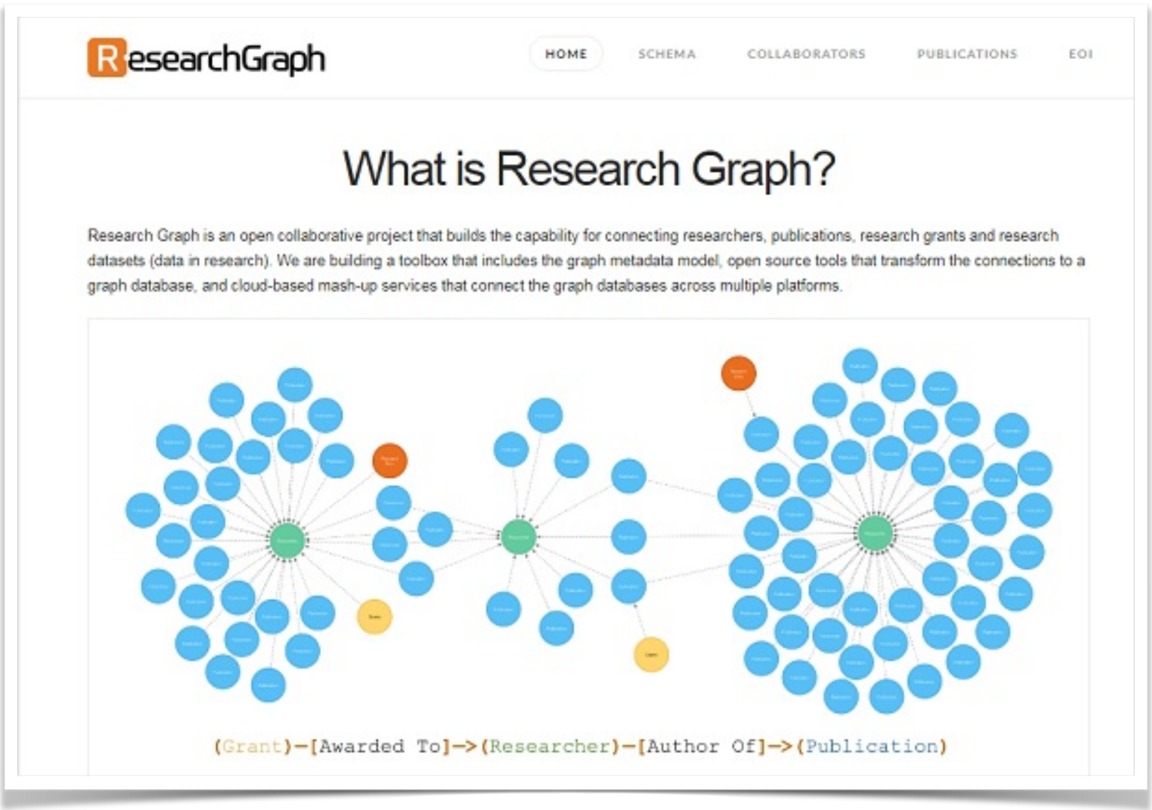


- Research section in design documents (if relevant)
  - Collate papers - *with backlog*
  - Summarize
  - Pros and cons
  - Trade-offs made
- Evaluate against product requirements
- c.f. competitive and market analysis

Research on the Internet

- **Google**
- **Research Databases** (many with *recommendation systems*)
  - arXiv, DBLP, Google Scholar, ResearchGate, Semantic Scholar ...
- **Websites**
  - Associations and publishers: ACM, IEEE, USENIX, ...
  - Labs + professors + students
  - Personal blogs
- **Twitter**

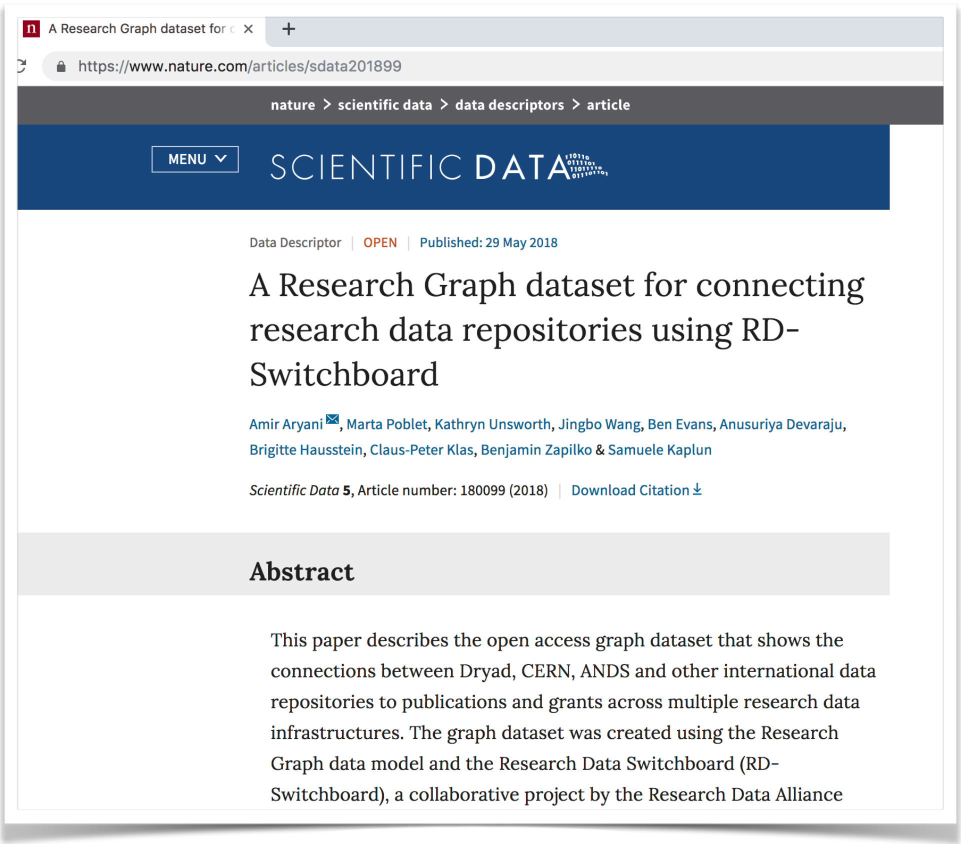
Research as a Knowledge Graph



researchgraph.org

- **Entities** (graph nodes)
  - Person
    - Advisor, Student, ...
  - Institution
    - University, Company, ...
  - Paper
  - Conference
- **Relationships** (graph edges)
  - Advised by
  - Published at
  - **Cites (reference)**

Research as a Knowledge Graph


A Research Graph dataset for c ×

https://www.nature.com/articles/sdata201899

nature > scientific data > data descriptors > article

MENU

SCIENTIFIC DATA

Data Descriptor | OPEN | Published: 29 May 2018

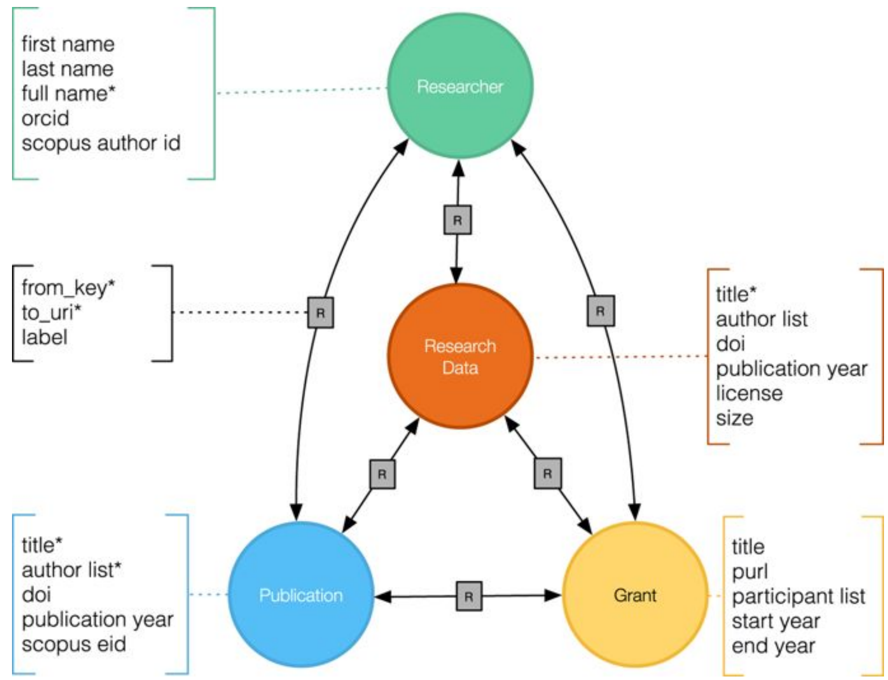
A Research Graph dataset for connecting research data repositories using RD-Switchboard

Amir Aryani, Marta Poblet, Kathryn Unsworth, Jingbo Wang, Ben Evans, Anusuriya Devaraju, Brigitte Hausstein, Claus-Peter Klas, Benjamin Zapilko & Samuele Kaplun

Scientific Data **5**, Article number: 180099 (2018) | Download Citation

Abstract

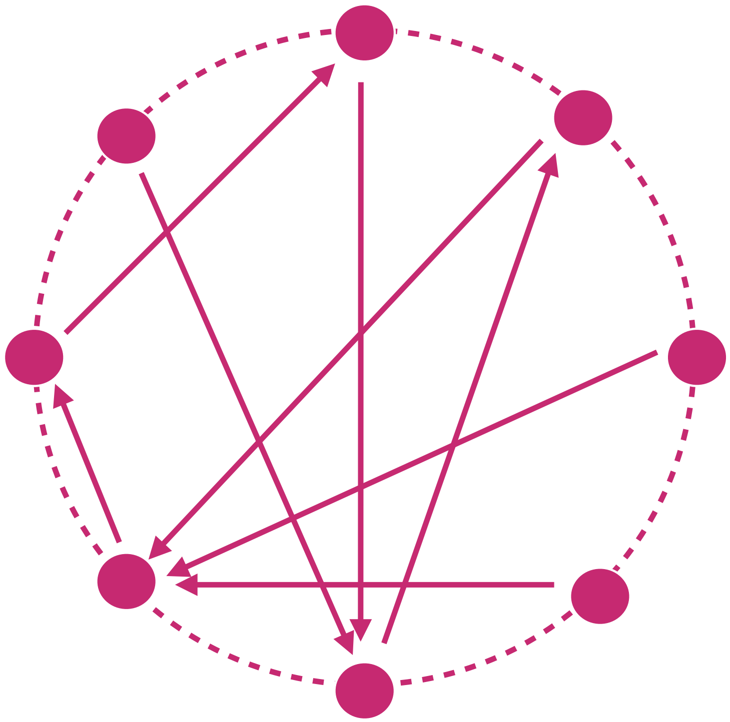
This paper describes the open access graph dataset that shows the connections between Dryad, CERN, ANDS and other international data repositories to publications and grants across multiple research data infrastructures. The graph dataset was created using the Research Graph data model and the Research Data Switchboard (RD-Switchboard), a collaborative project by the Research Data Alliance



Group Membership Protocols

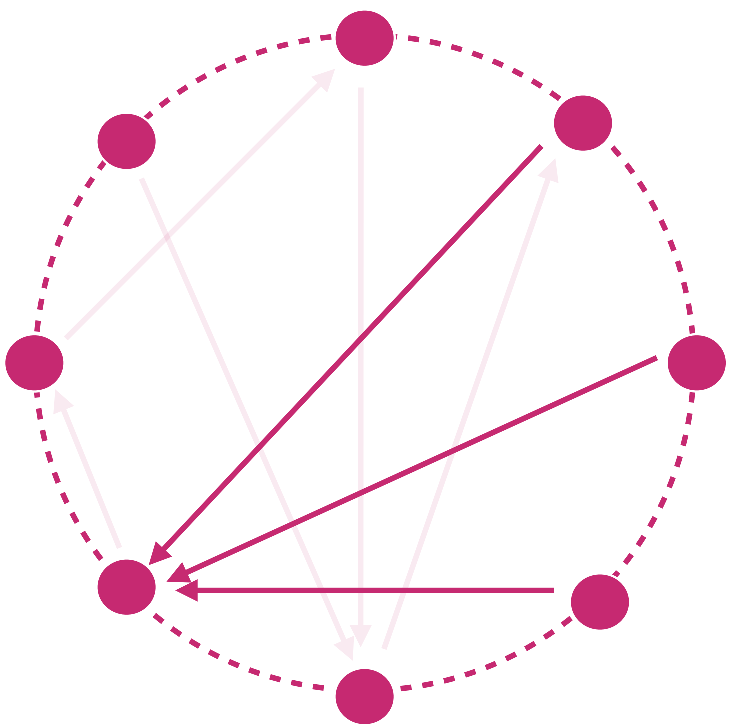
- Dynamic group of members ('processes')
- **Discovery**
  - New member joins group...
    - *Discovers* other members of the group
    - *Discovered* by other members
- **Failure Detection**

Peer Failure Detection



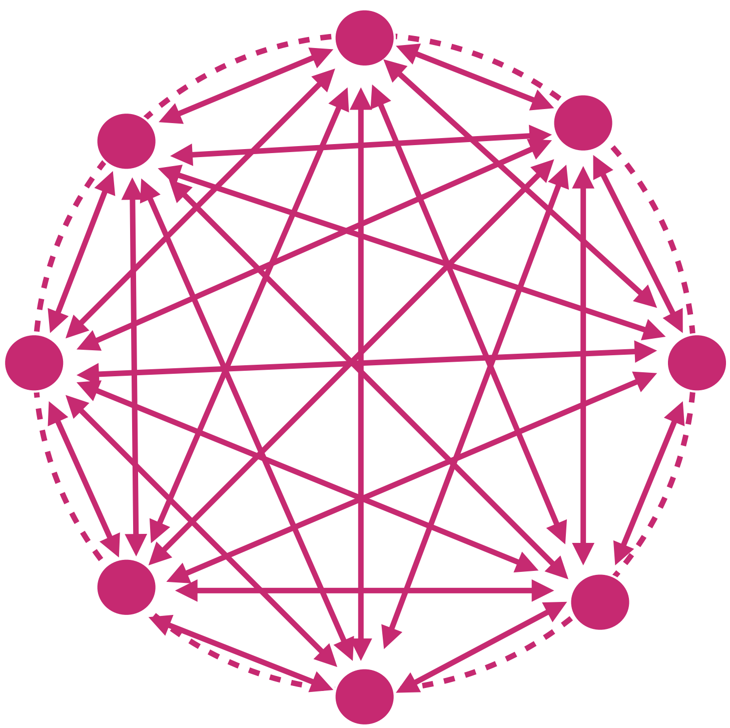
- *Processes monitor one another*
  - *No special nodes to administer*

Peer Failure Detection



- *Processes monitor one another*
  - *No special nodes to administer*
- ***Redundant monitoring***

Simple But Not Scalable



Heartbeat Membership and Failure Detection (circa 1996)

- Every node sends a regular heartbeat message ... *to every other node* ('full mesh')
- Receive a heartbeat from you?
  - "You're alive"
- Miss (a few?) heartbeats
  - "You're dead!"
- Message load **O(n²)**

SWIM (IEEE 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 frequency w.r.t. detecting process crashes. This paper reports on the design, implementation and performance of the SWIM sub-system on a large cluster of commodity PCs.

1. Introduction

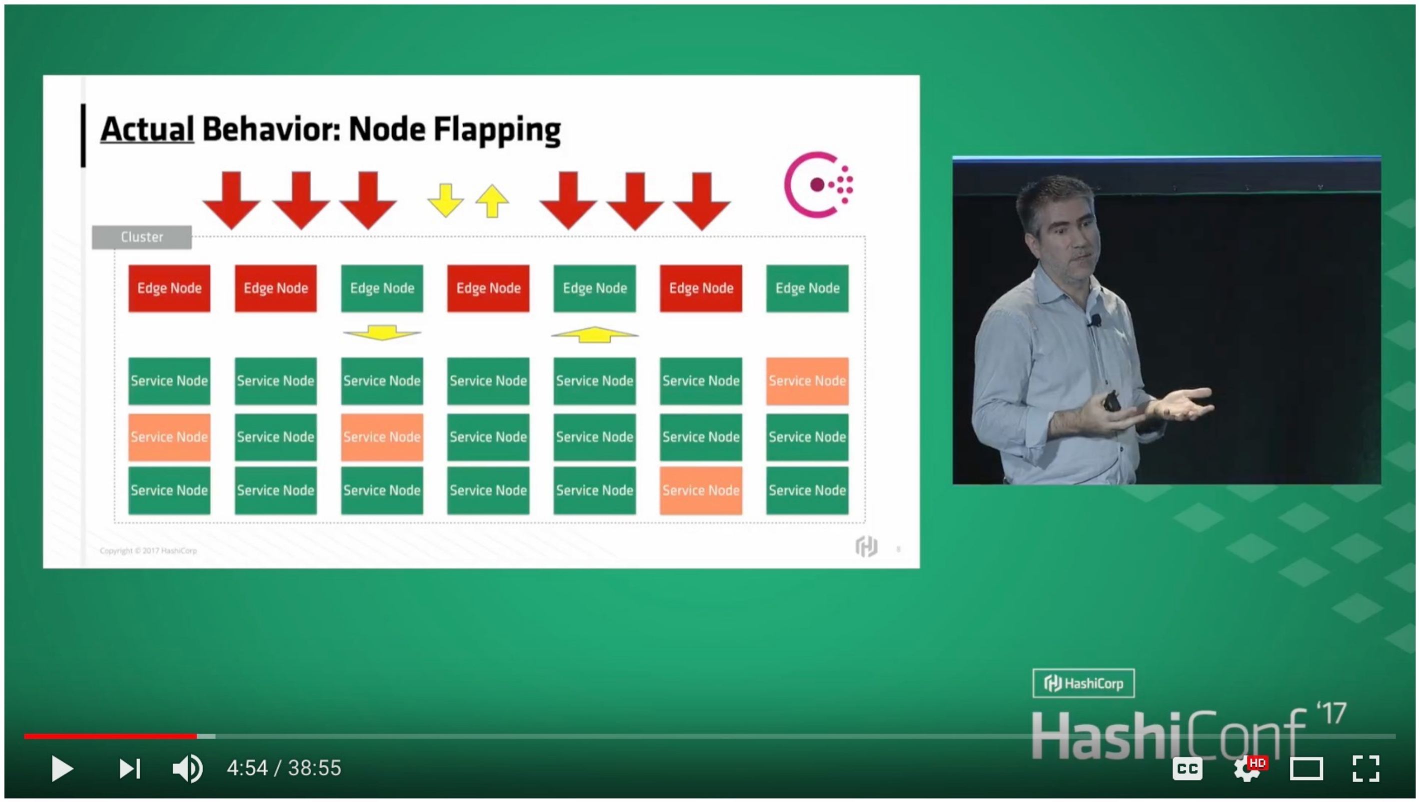
As you swim lazily through the milieu,
The secrets of the world will infect you.

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 applications such as distributed databases that need to reconcile recent disconnected updates [14], publish-subscribe systems, and large-scale peer-to-peer systems[15]. The performance

SWIM: What's In A Name?

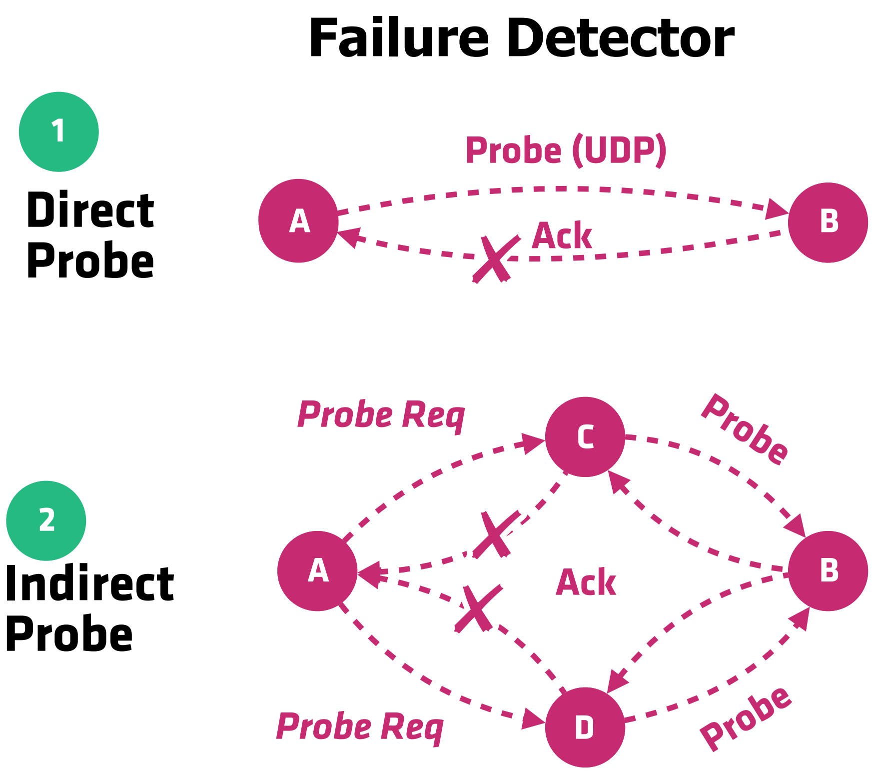
- **S**calable
  - *Fixed* per node number of messages, not ∝ # nodes
  - Message load O(n), not O(n^2)
- **W**eakly-consistent
  - Nodes don't all have to see same state simultaneously
  - Converge to same view (quickly)
- **I**nfection-style
  - Gossip (aka 'epidemic') spread of information
- **M**embership

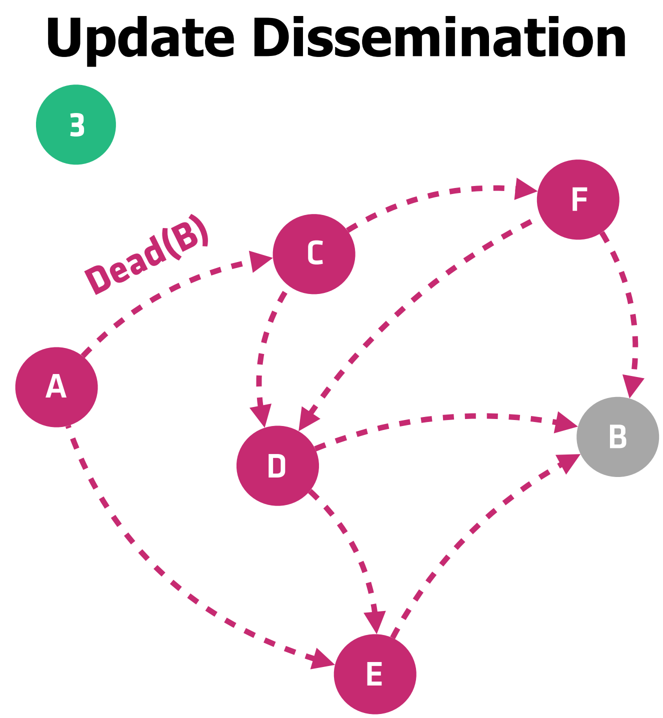
For a Detailed Explanation ...



Making Gossip More Robust with Lifeguard

SWIM Protocol Components





SWIM's Use of Randomization

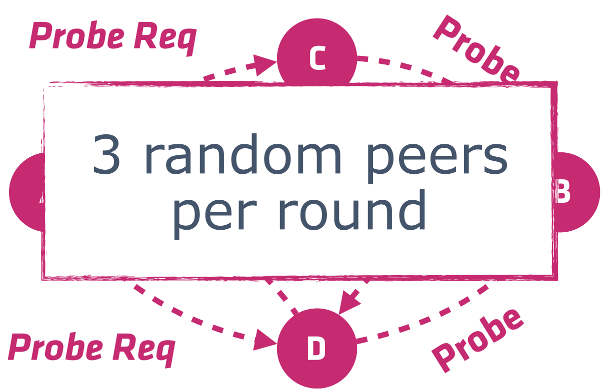
Failure Detector

1

Direct Probe

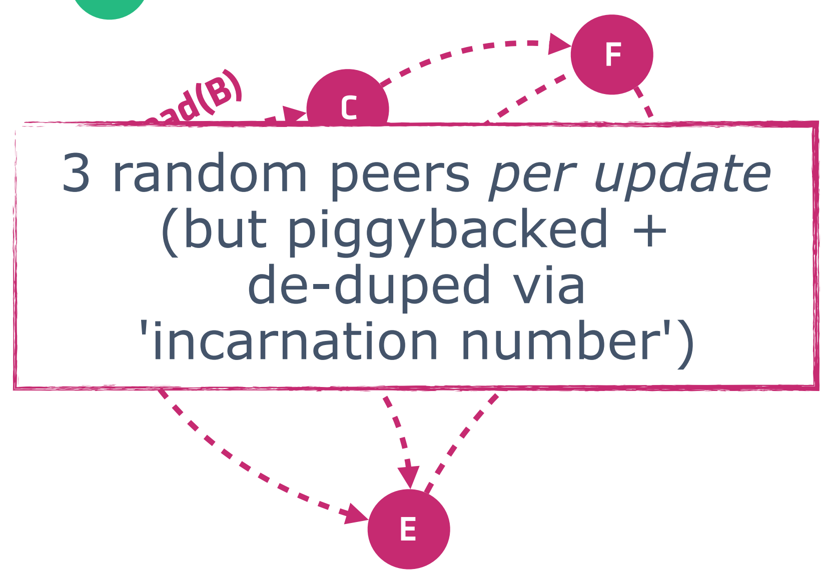
2

Indirect Probe

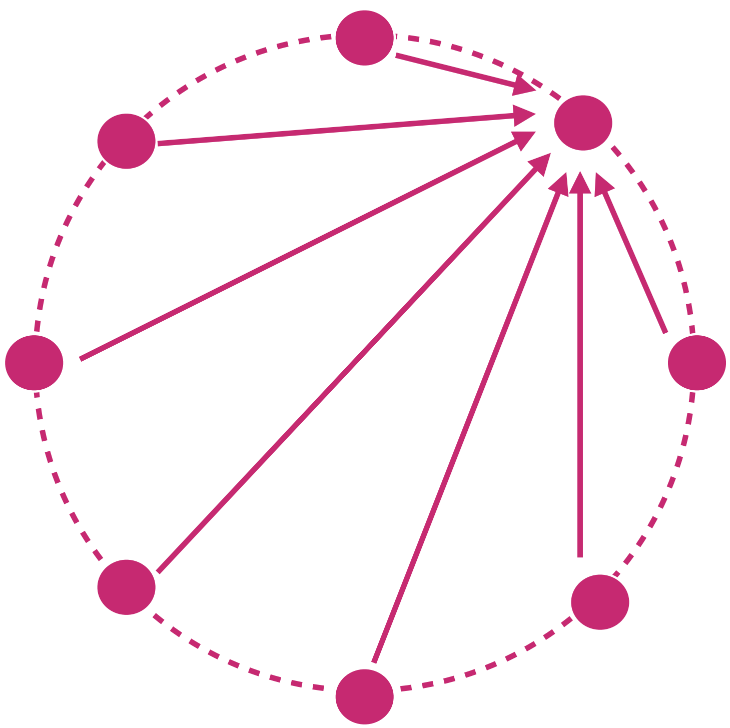


Update Dissemination

3

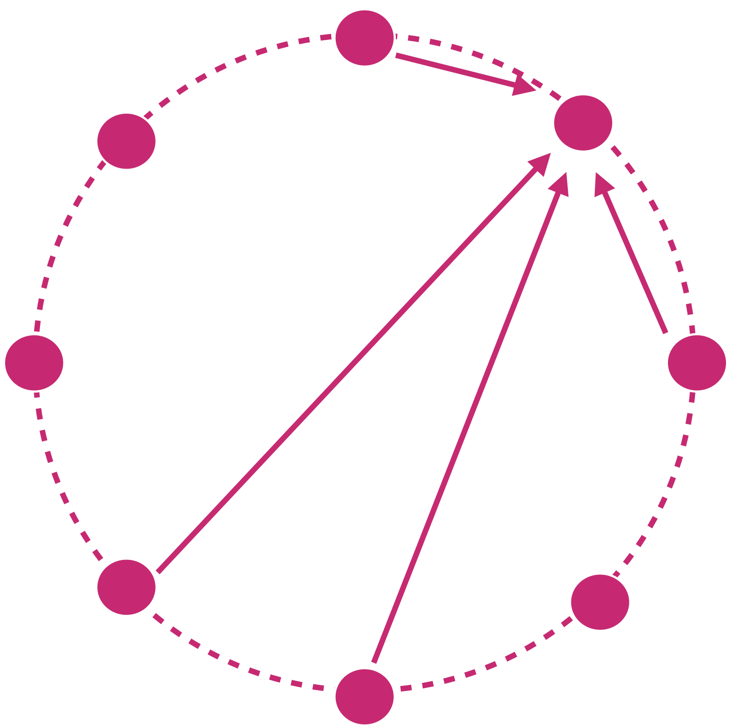


Randomization Keeps SWIM Robust (and Simple)



Reduce communication (for scalability) without randomization?

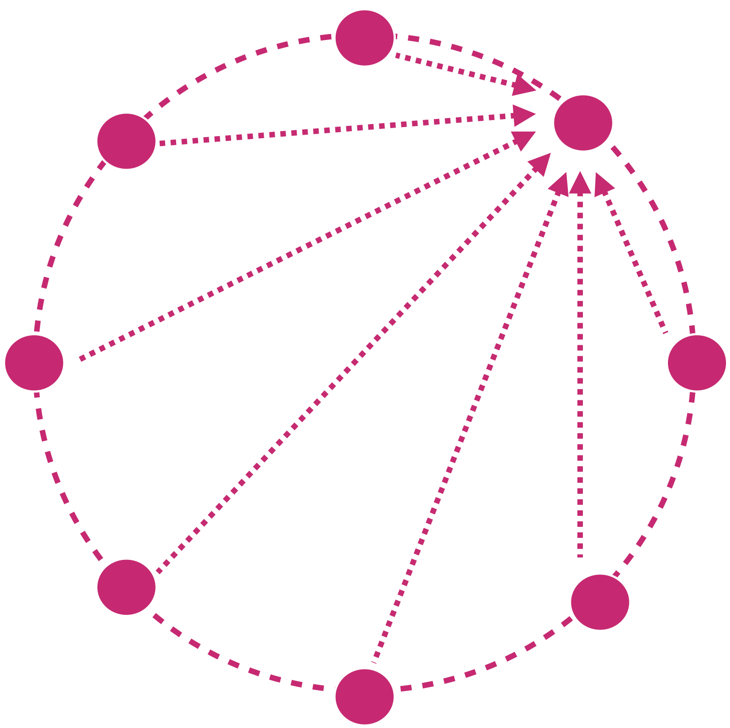
Randomization Keeps SWIM Robust (and Simple)



Reduce communication (for scalability) without randomization?

- Greatly increased chance of missed failures (aka 'false negatives')

Randomization Keeps SWIM Robust (and Simple)



Reduce communication (for scalability) without randomization?

- Greatly increased chance of missed failures (aka 'false negatives')

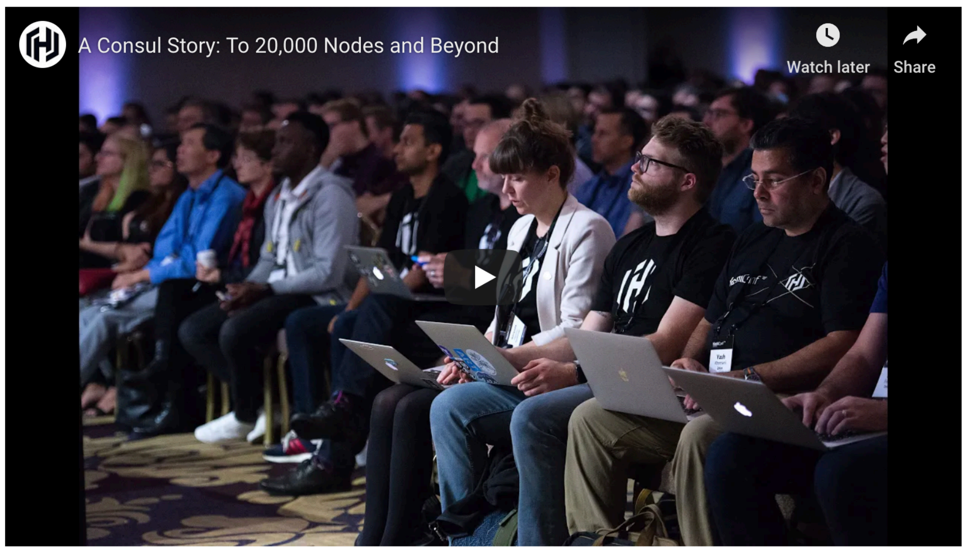
With randomization

- Each node is ***still checked by every other node*** ... just not so often
  - Much less chance of false negatives
  - No special nodes or node hierarchies to maintain after node failures

SWIM Has Worked Out

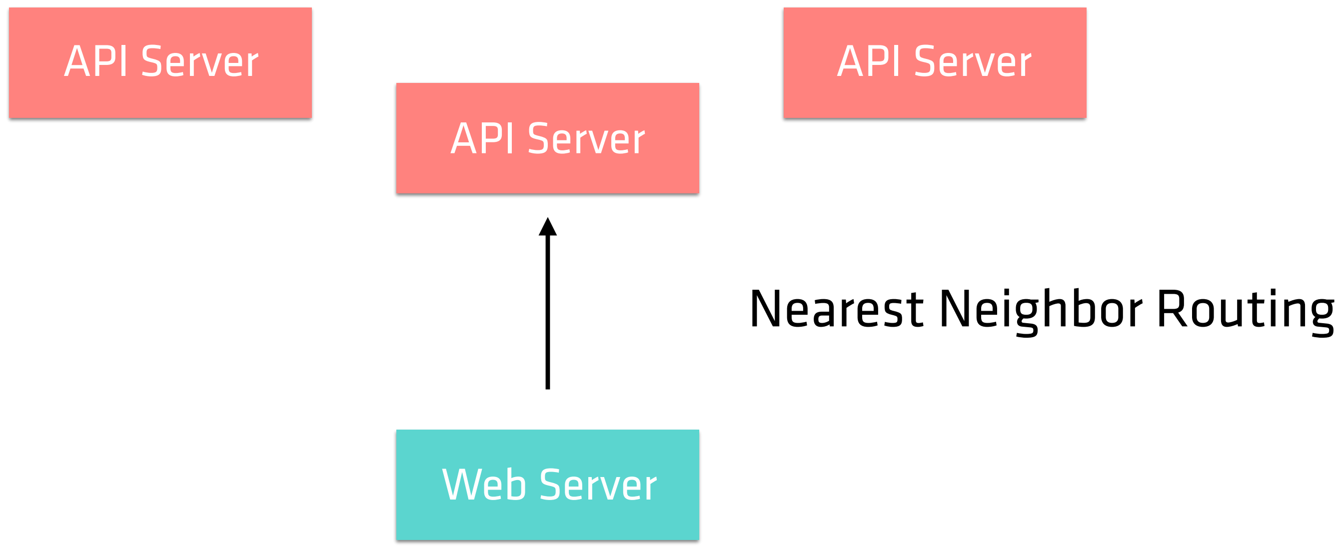
Bloomberg's Consul Story: To 20,000 Nodes and Beyond

Hear the story of how Bloomberg used Consul to create their own service discovery SaaS for their heterogeneous IT environment.

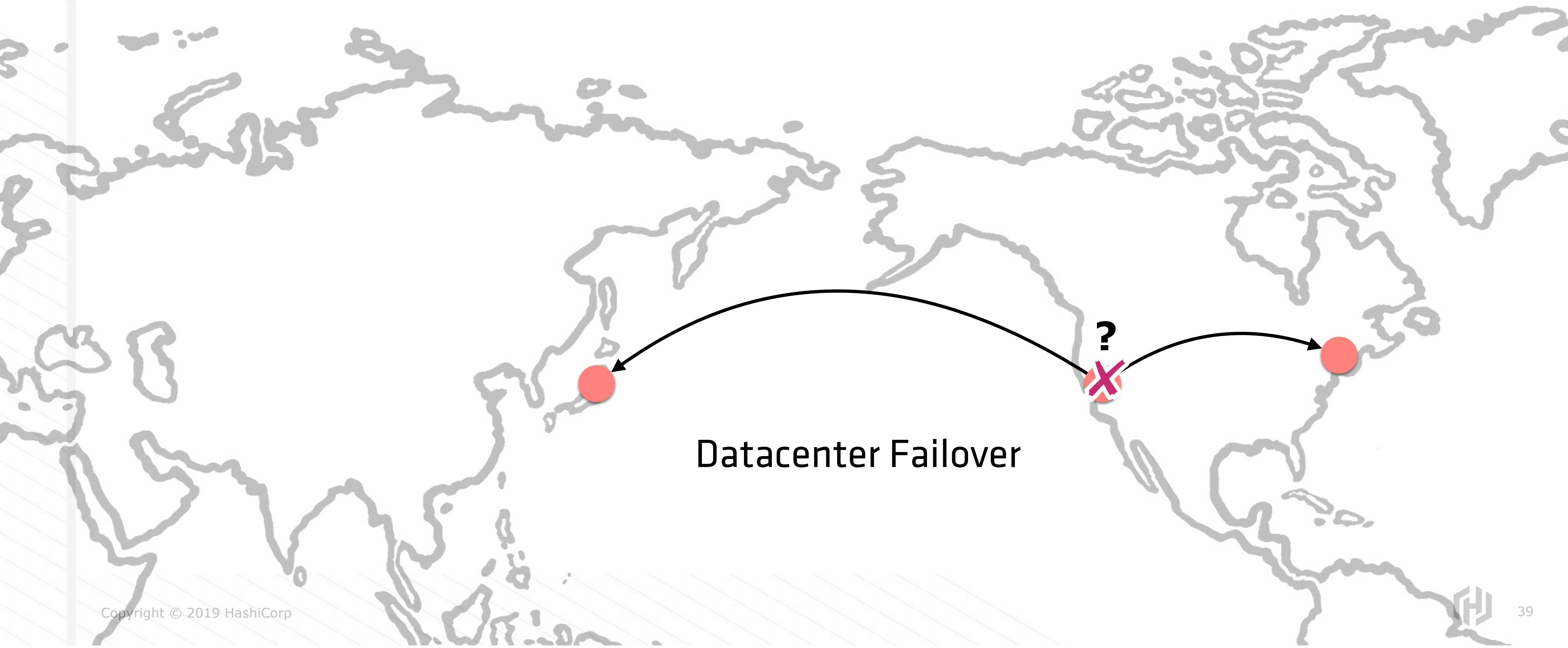


18 Months Later ...

Network Distance for Load Balancing



Network Distance for Disaster Recovery



Network Distance

Consul RTT

Command: `consul rtt`

The `rtt` command estimates the network round trip time between two nodes using Consul's network coordinate model of the cluster.

See the Network Coordinates internals guide for more information on how these coordinates are computed.

Usage

Usage: `consul rtt [options] node1 [node2]`

At least one node name is required. If the second node name isn't given, it is set to the agent's node name. These are the node names as known to Consul as the `consul members` command would show, not IP addresses.

Vivaldi: Network Coordinates

Vivaldi: A Decentralized Network Coordinate System

Frank Dabek, Russ Cox, Frans Kaashoek, Robert Morris
MIT CSAIL
Cambridge, MA
fdabek,rsc,kaashoek,rtm@mit.edu

ABSTRACT

Large-scale Internet applications can benefit from an ability to predict round-trip times to other hosts without having to contact them first. Explicit measurements are often unattractive because the cost of measurement can outweigh the benefits of exploiting proximity information. Vivaldi is a simple, light-weight algorithm that assigns synthetic coordinates to hosts such that the distance between the coordinates of two hosts accurately predicts the communication latency between the hosts.

Vivaldi is fully distributed, requiring no fixed network infrastructure and no distinguished hosts. It is also efficient: a new host can compute good coordinates for itself after collecting latency information from only a few other hosts. Because it requires little communication, Vivaldi can piggy-back on the communication patterns of the application using it and scale to a large number of hosts.

An evaluation of Vivaldi using a simulated network whose latencies are based on measurements among 1740 Internet hosts shows that a 2-dimensional Euclidean model with *height vectors* embeds these hosts with low error (the median relative error in round-trip time prediction is 11 percent)

synthetic coordinates in some coordinate space such that distance between two hosts' synthetic coordinates predicts the RTT between them in the Internet. Thus, if a host x learns the coordinates of a host y, x doesn't have to perform an explicit measurement to determine the RTT to y; instead, the distance between x and y in the coordinate space is an accurate predictor of the RTT.

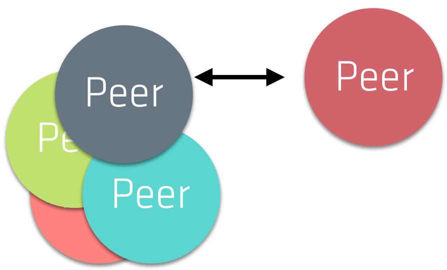
The Internet's properties determine whether synthetic coordinates are likely to work well. For example, if Internet latency is dominated by speed-of-light delay over links, and the Internet is well-enough connected that there is a roughly direct physical path between every pair of hosts, and the Internet routing system finds these direct paths, then synthetic coordinates that mimic latitude and longitude are likely to predict latency well.

Unfortunately, these properties are only approximate. Packets often deviate from great-circle routes because few site pairs are directly connected, because different ISPs peer at a limited number of locations, and because transmission time and router electronics delay packets. The resulting distorted latencies make it impossible to choose two-dimensional host coordinates that predict latency perfectly, so a synthetic coordinate system must have a strategy for

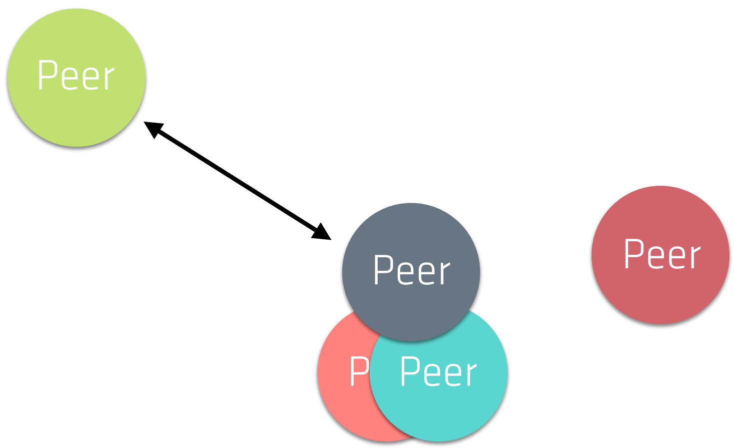
Spring 'Dynamic Relaxation' Model

- Imagine each pair of peers connected by a spring...
- Compressed together ...
- But natural length of each spring is RTT between those two peers...

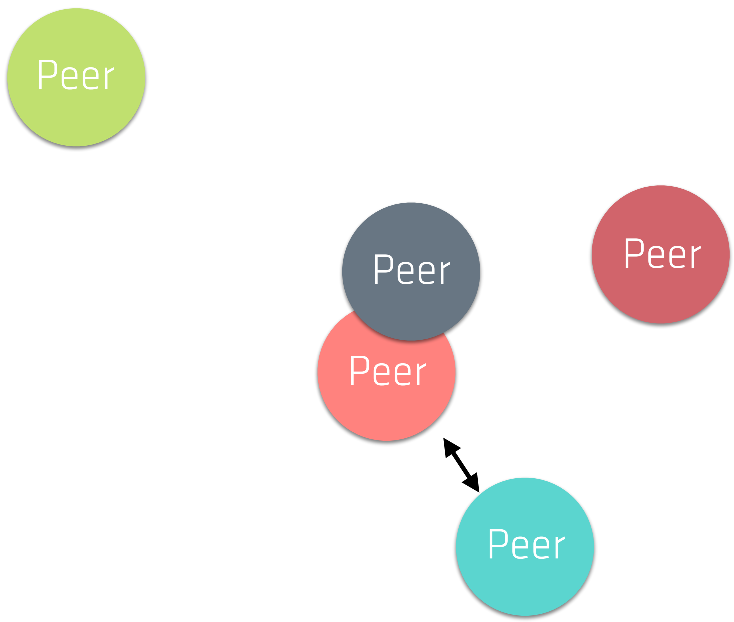
Spring 'Dynamic Relaxation' Model



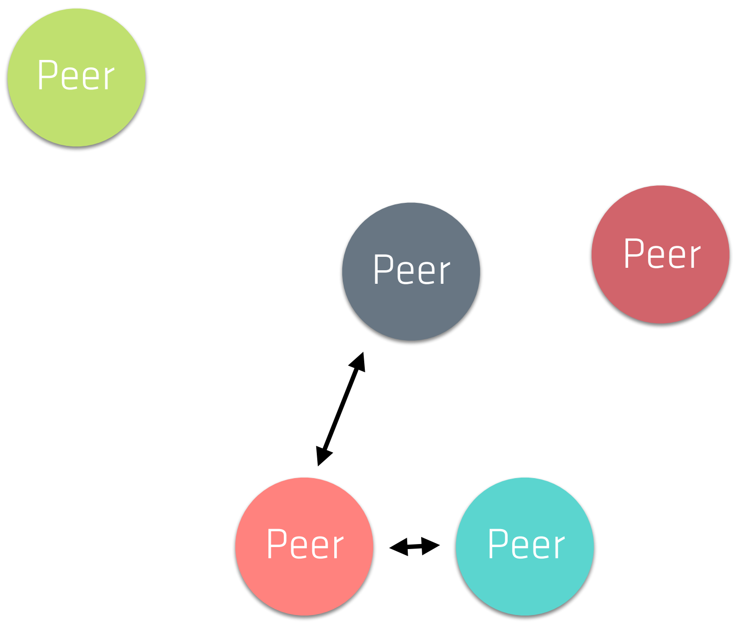
Spring 'Dynamic Relaxation' Model



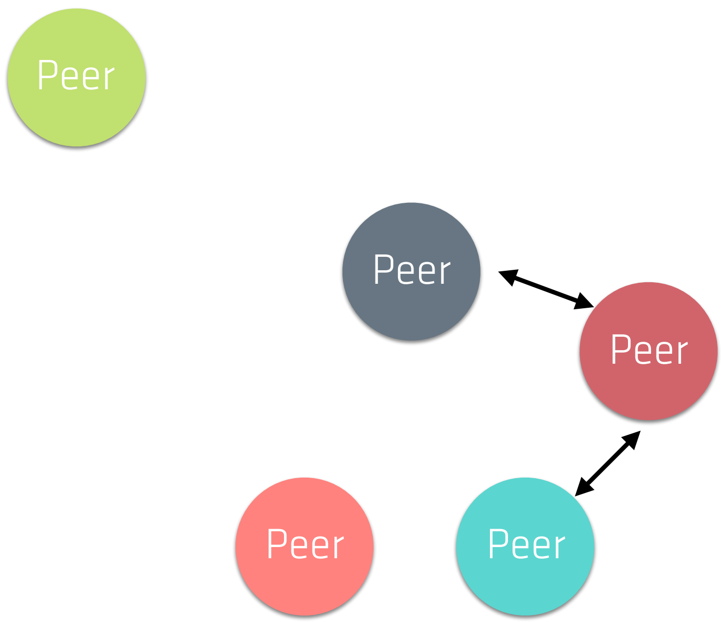
Spring 'Dynamic Relaxation' Model



Spring 'Dynamic Relaxation' Model



Spring 'Dynamic Relaxation' Model



SWIM Probe Messages Give Us RTT For Free

- Random pattern of communication really helps
  - Ring or tree topology would measure only a fraction of the paths

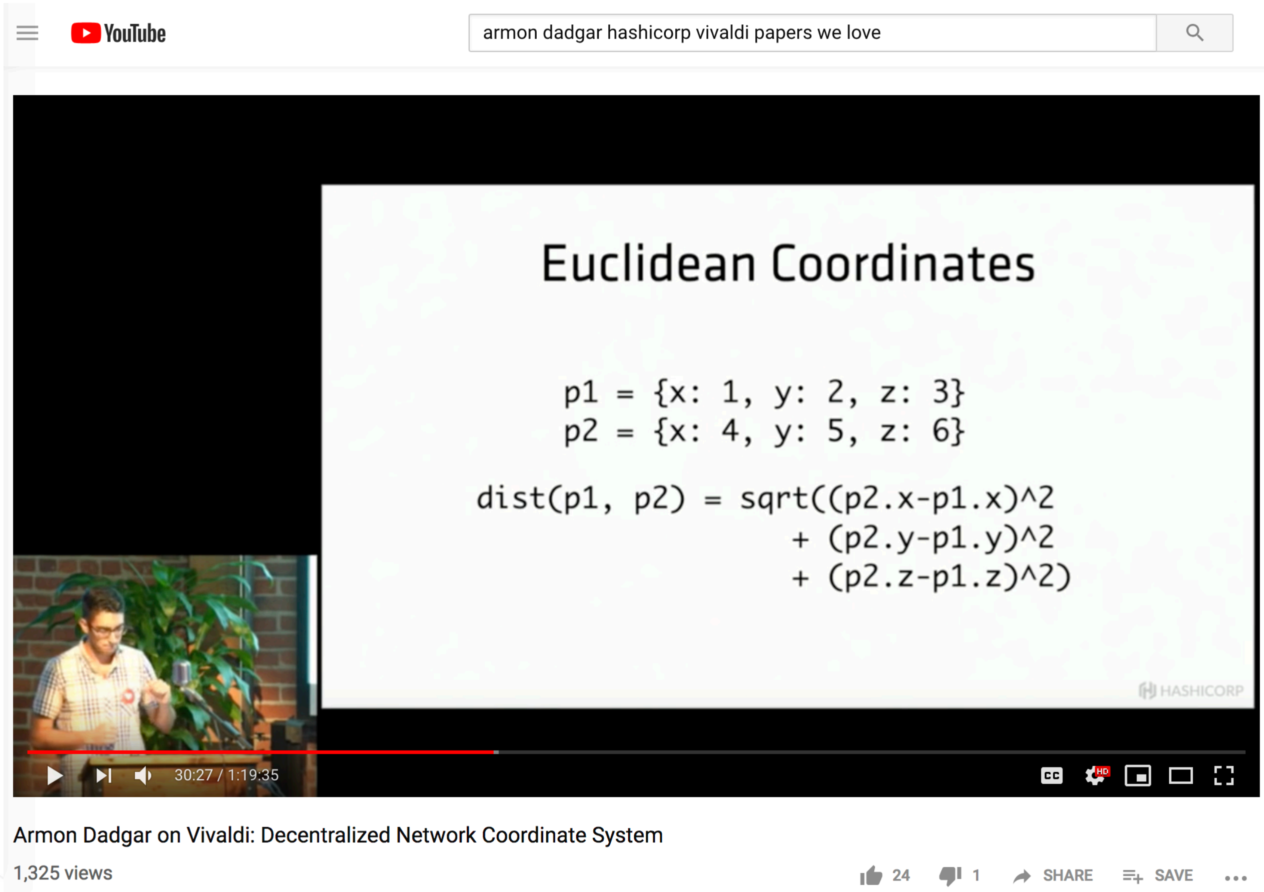
Randomization: Two Ways We Win

- SWIM's *simple*, robust scalability
- RTT's for Network Distance for free

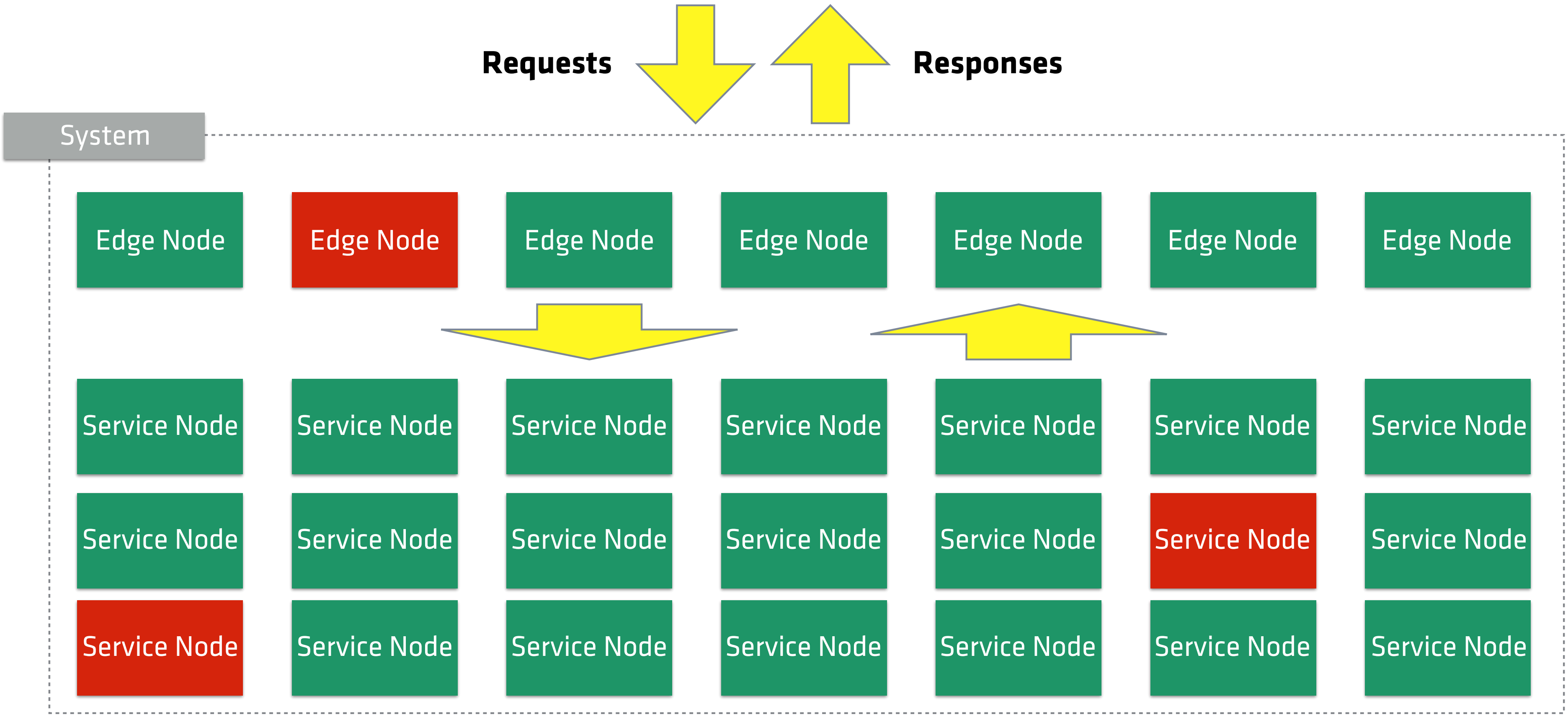
Mining the Research Graph

- Found multiple alternative solutions
- Some papers were follow up work ('responses') to Vivaldi
  - Found via *citations* (Google Scholar)
  - Helped identify issues Vivaldi originally missed
  - Provided a toolkit of possible extensions
  - Defined metrics we could use to evaluate the alternatives

Vivaldi In Consul/Serf In Depth

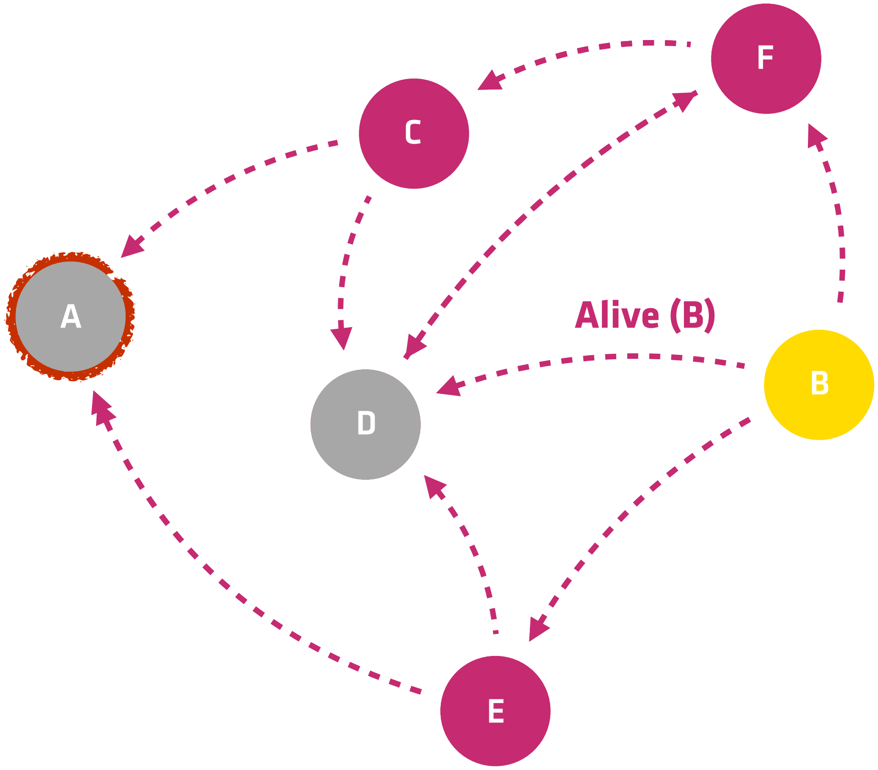


Wait … What About the Flappers?



SWIM's Achilles Heel

Works around slow/dead intermediaries...

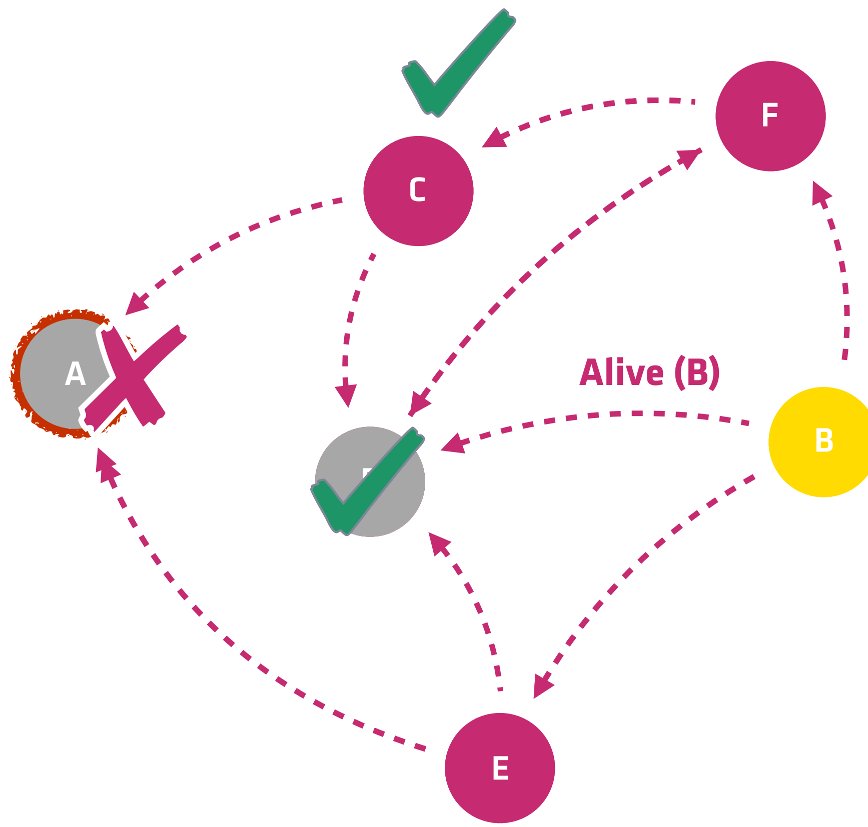


SWIM's Achilles Heel

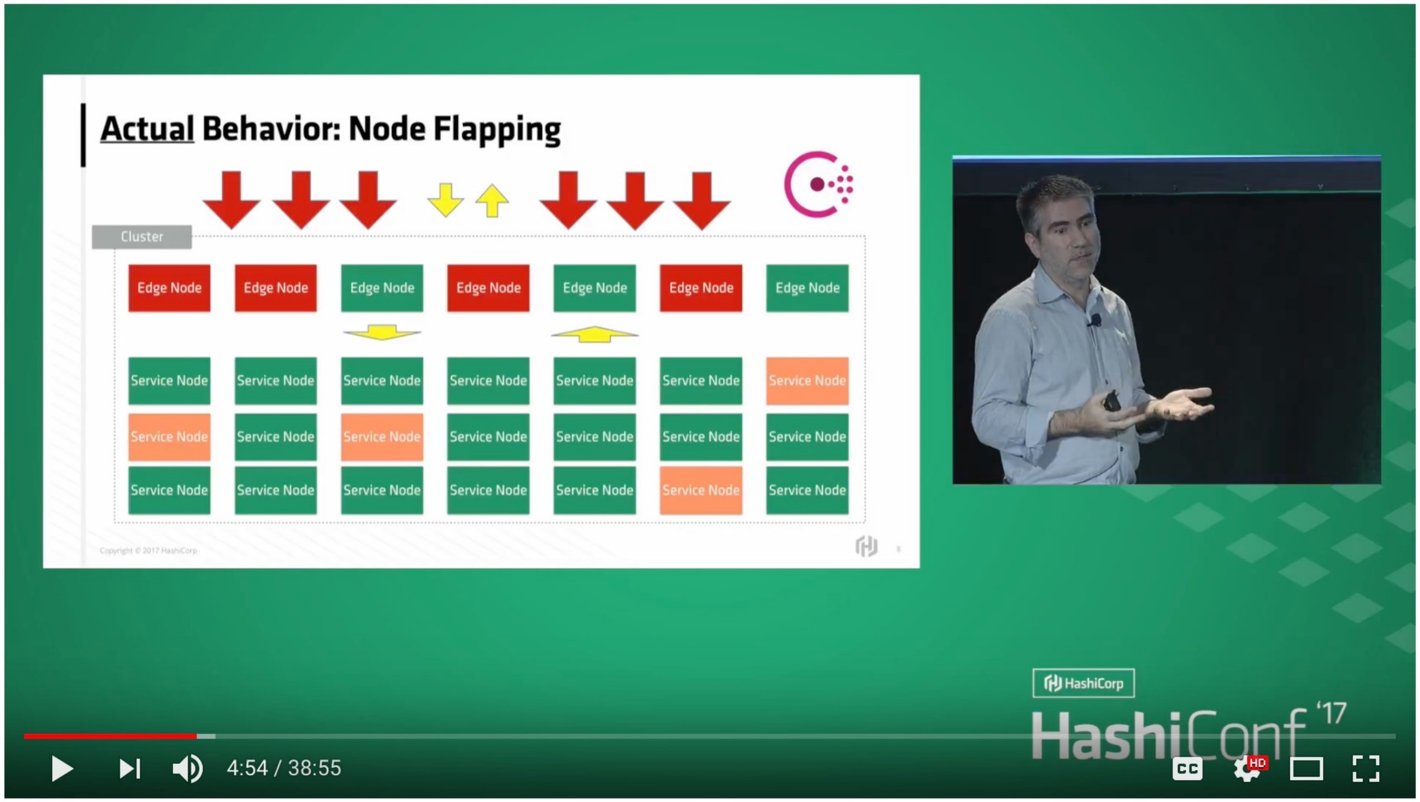
Works around slow/dead intermediaries...

But **still assumes some messages are processed in a timely manner**

- No slow node *originating* a probe or suspicion
  - Must process Ack and Alive messages



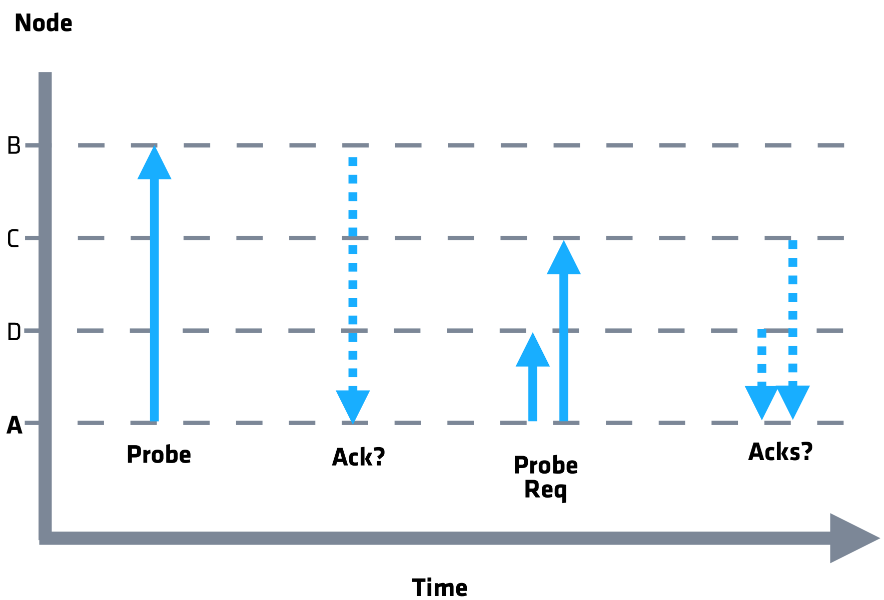
For a Detailed Explanation ...

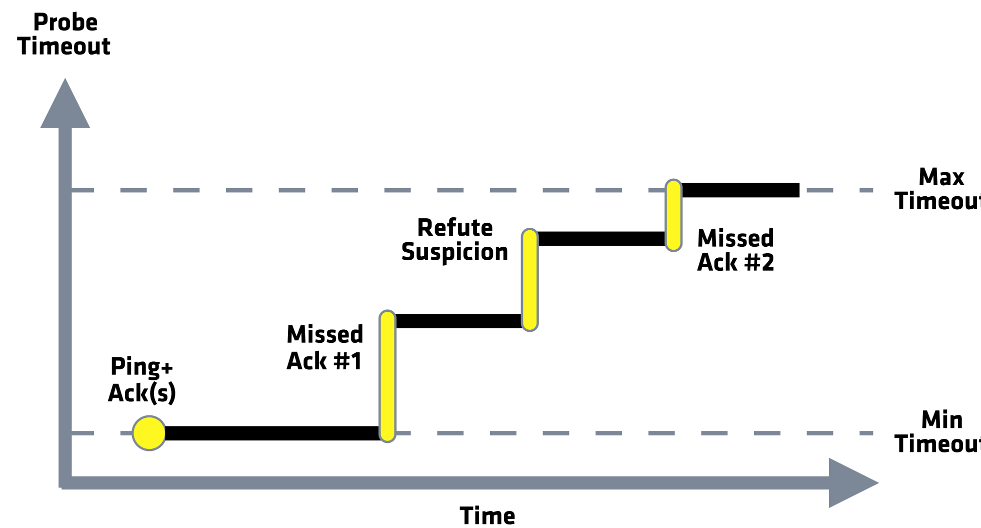


Making Gossip More Robust with Lifeguard

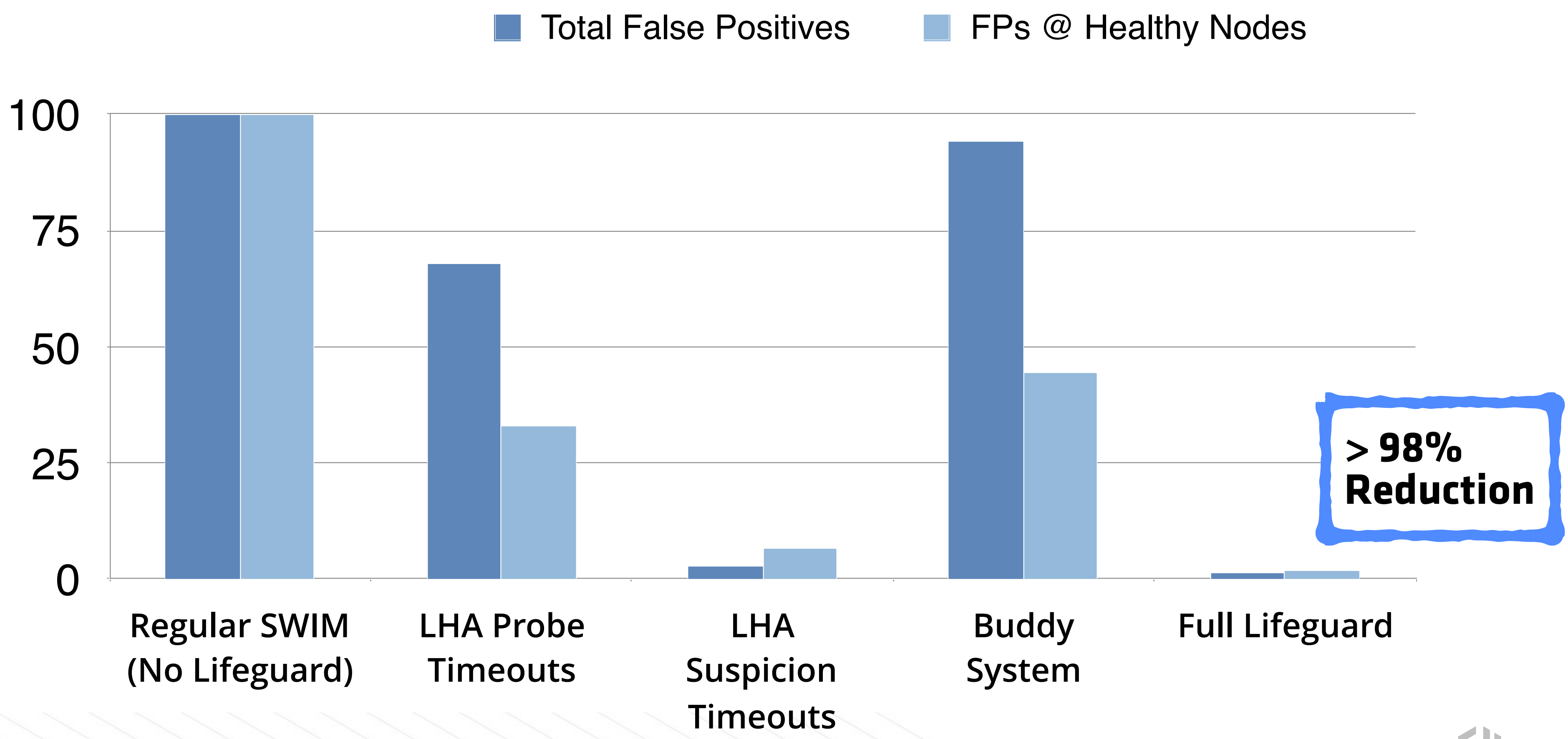
Lifeguard Heuristics Based On 'Local Health'

- Failure Detector has *expectations* about messages it will receive
- Use *absence of expected messages* to increase timeouts at slow members

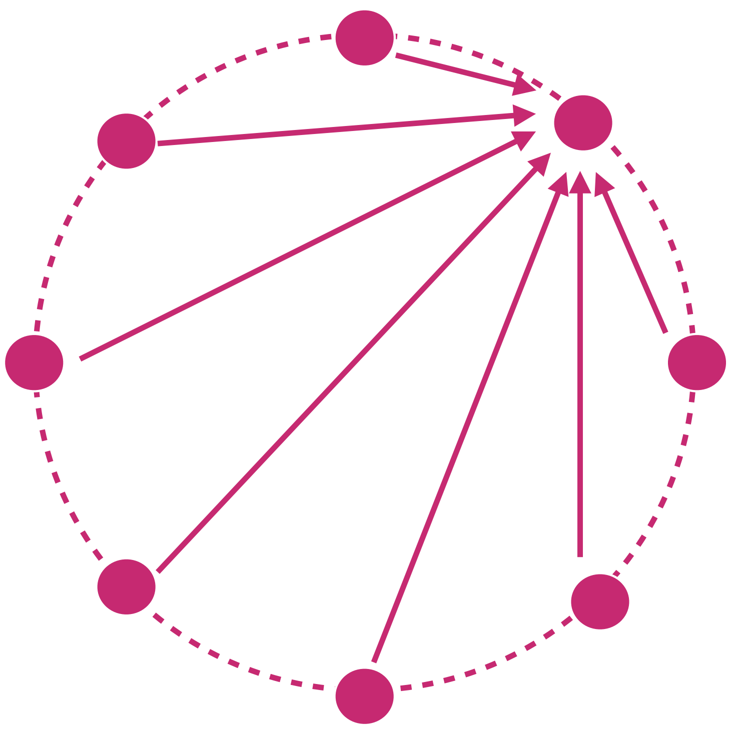




Reduction in False Positives

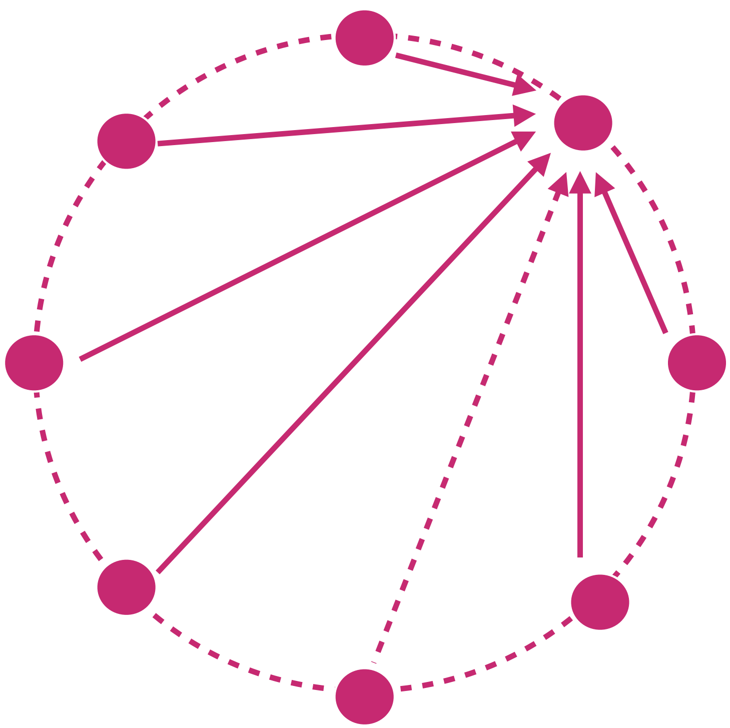


Randomization Is Key To Lifeguard Too



- Every node checked by every other node... just not so often

Randomization Is Key To Lifeguard Too



- Every node checked by every other node... just not so often
- If we **slow down 3 out of 100** nodes, **97 nodes are still checking all 100** of them
  - Graceful degradation
  - Gets *better* as group size increases

Randomization: ~~Two~~ Three Ways We Win

- SWIM's *simple*, robust scalability
- RTT's for Network Distance for free
- Lifeguard's defense against SWIM's weak spot

Research Also Helped With Lifeguard ... Eventually

- Picked up literature search thanks to backlog

- No one had reported on this issue but...
  - **Gave us the metrics** to use to investigate
  - **Showed us good benchmarking** experiments

- Benchmarks have persistent value
  - Regression testing, competitive analysis, ...
  - Backbone of a paper

Lifeguard (arXiv and IEEE DSN 2018)

Lifeguard: Local Health Awareness for More Accurate Failure Detection

Armon Dadgar James Phillips Jon Currey
HashiCorp Inc.
{armon,james,jc}@hashicorp.com

***Abstract*—SWIM is a peer-to-peer group membership protocol, with attractive scaling and robustness properties. However, our experience supporting an implementation of SWIM shows that a high rate of false positive failure detections (healthy members being marked as failed) is possible in certain real world scenarios, and that this is due to SWIM's sensitivity to slow message processing. To address this we propose a set of extensions to SWIM (together called Lifeguard), which employ heuristic measures of a failure detector's *local health*. In controlled tests, Lifeguard is able to reduce the false positive rate by more than 50x. Real world deployment of the extensions has significantly reduced support requests and observed instability. The need for this work points to the fail-stop failure model being overly simplistic for large datacenters, where the likelihood of some nodes experiencing transient CPU starvation, IO flakiness, random packet loss, or other non-crash problems becomes high. With increasing attention being given to these *gray failures*, we believe the local health abstraction may be applicable in a broad range of settings, including other kinds of distributed failure detectors.**

However, our experience supporting Consul and Nomad shows that, even with the Suspicion subprotocol, slow message processing can still lead healthy members being marked as failed in certain circumstances. When the slow processing occurs intermittently, a healthy member can oscillate repeatedly between being marked as failed and healthy. This 'flapping' can be very costly if it induces repeated failover operations, such as provisioning members or re-balancing data.

Debugging these scenarios led us to insights regarding both a deficiency in SWIM's handling of slow message processing, and a way to address that deficiency. The approach used is to make each instance of SWIM's failure detector consider its own health, which we refer to as *local health*. We implement this via a set of extensions to SWIM, which we call Lifeguard. Lifeguard is able to significatly reduce the false positive rate, in both controlled and real-world scenarios.

Looking at the bigger picture, we see that SWIM follows

Lifeguard (arXiv and IEEE DSN 2018)

Lifeguard: Local Health Awareness
for More Accurate Failure Detection

Developed iteratively over 18 months...

- Internal benchmarking report
- Internal white paper (engineering, sales, ...)
- arXiv.org 'pre-print'
- Published version
  - Blog posts, Tweets, conference talks ...

some nodes experiencing transient CPU starvation, IO flakiness, random packet loss, or other non-crash problems becomes high. With increasing attention being given to these *gray failures*, we believe the local health abstraction may be applicable in a broad range of settings, including other kinds of distributed failure detectors.

own health, which we refer to as *local health*. We implement this via a set of extensions to SWIM, which we call Lifeguard. Lifeguard is able to significatly reduce the false positive rate, in both controlled and real-world scenarios.

Looking at the bigger picture, we see that SWIM follows

Publication Has Embedded Us In the Graph

Lifeguard: Swim-ing with situational awareness

☐ Search within citing articles

[PDF] Multisource Rumor Spreading with Network Coding — [PDF] inria.fr

YD Bromberg, Q Dufour, D Frey - INFOCOM 2019, 2019 - hal.inria.fr

The last decade has witnessed of a rising surge interest in Gossip protocols in distributed systems. In particular, as soon as there is a need to disseminate events, they become a key functional building block due to their scalability, robustness and fault tolerance under high …

Related articles All 8 versions

[PDF] Membership Service for Large-Scale Edge Systems — [PDF] unl.pt

FM Magalhães - 2018 - asc.di.fct.unl.pt

Distributed systems are increasing in scale and complexity. Cloud-based applications need more resources to be able to continue satisfying the user requirements. There are more users and devices with Internet access and therefore, more data to process and store. There …

Related articles

Benefits of Research (Tell Your Boss ...)

- **Better algorithms**
- **Relevant metrics**
- **Talent**
  - Interns and full-time
- **Reputation**
  - Customers and potential customers
  - Internal >> employee satisfaction

Where to Begin?

- *Papers We Love* (PWL)
  - Github repo + Meetups

- *The Morning Paper* by Adrian Colyer
  - Blog + email

- At work
  - Reading group
  - Brownbag PWL
  - Colleagues with research experience?

Getting Involved in Research

- Ask questions
  - Email, Twitter
- Attend conferences
- Discuss your open problems
  - Blog
  - Twitter
- PhD candidate interns
- Employees with research experience
  - PhD, Masters or started graduate school?

PhD Candidate Interns

- Mutual benefit
  - Your (relevant) problem/data == **goldmine** for their work
  - Work done during internship is your Intellectual Property

- Approach students and advisors
  - Poster sessions at conferences
  - Email

Researchers are People Too!

Sarah Knowles @dr_know
Follow

Is it genuinely a thing that ppl don't email academics for their papers because they think we get paid for them?

We actually pay to make them free (I'd try to explain this but it's... a mess.) And we're happy and allowed to distribute copies. Hit us up!

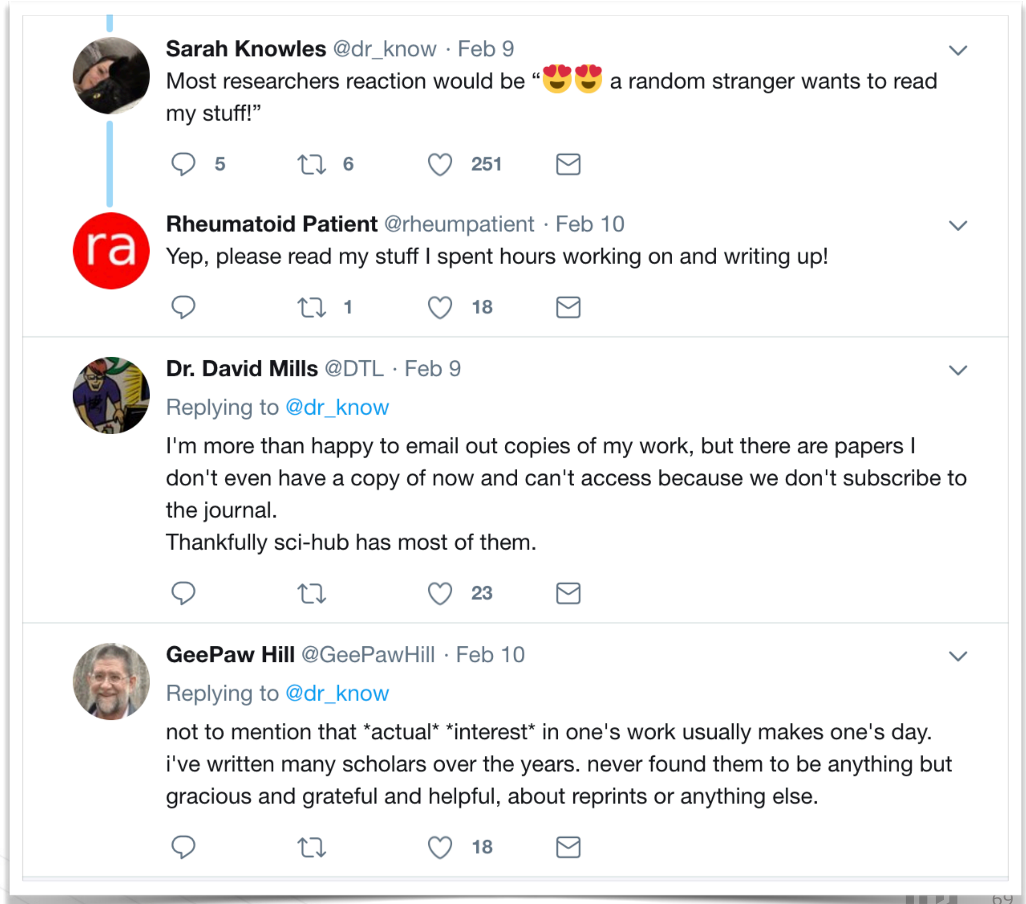
5:31 AM - 9 Feb 2019

1,540 Retweets 4,772 Likes

88 1.5K 4.8K

Tweet your reply

Madelen @m4delen · Feb 9
Replying to @dr_know @BipolarBlogger
Yes. I am always too shy to email researchers when it says email researchers for



Sarah Knowles @dr_know · Feb 9
Most researchers reaction would be "😍😍 a random stranger wants to read my stuff!"
5 6 251

Rheumatoid Patient @rheumpatient · Feb 10
Yep, please read my stuff I spent hours working on and writing up!
1 18

Dr. David Mills @DTL · Feb 9
Replying to @dr_know
I'm more than happy to email out copies of my work, but there are papers I don't even have a copy of now and can't access because we don't subscribe to the journal.
Thankfully sci-hub has most of them.
23

GeePaw Hill @GeePawHill · Feb 10
Replying to @dr_know
not to mention that *actual* *interest* in one's work usually makes one's day. i've written many scholars over the years. never found them to be anything but gracious and grateful and helpful, about reprints or anything else.
18

Thank you.

HashiCorp

www.hashicorp.com hello@hashicorp.com

Controlled Use of Randomization

- Pseudo-randomization
  - Go math/rand
  - Seeded from /dev/urandom

sean- / **seed**

- Each node *cycles through all the nodes it knows about*
  - Only then talk to same node a second time

- New nodes *inserted into its memberlist at random position*

Weak Consistency Is Sometimes Enough