Group Therapy for Systems:

Using link attestations to manage failure

Michael J. Freedman
NYU / Stanford

Ion Stoica, David Mazieres, Scott Shenker

A little background…

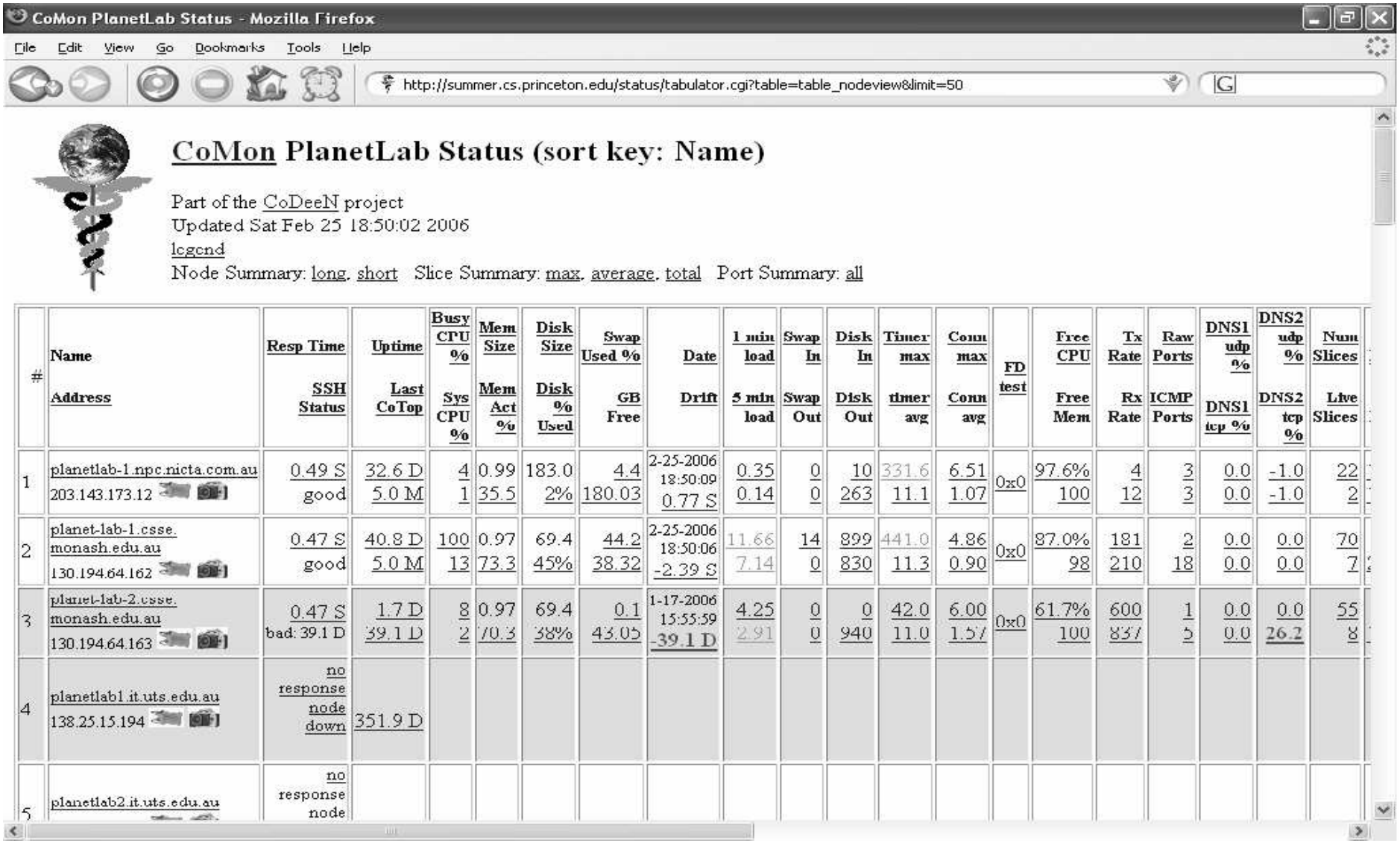
Powered By PLANETLAB

- I built and manage CORAL
  - CoralCDN is an open, P2P content distribution network
  - http://cnn.com/ → http://cnn.com.***nyud.net:8080***/
  - Publicly deployed for 2 years on PlanetLab
  - 25 M requests from 1 M clients for 2-3 TB daily
- Nodes rarely crash
- Nodes often don't behave "correctly"
- How do I cope with this problem?

Problems running CoralCDN

- Non-transitive or asymmetric routing
  - Interdomain routing failures, I2-only peering, firewalls, egress filtering, proxies, …
- Performance faults
  - Network queuing and high packet loss, slow disks, long context switches, memory leaks, …
- Buggy code
  - File-descriptor leaks, race conditions, versioning issues, …
- File-system errors
  - Disk quota exceeded, disk corruption, wrong file perms, …
- Problem: Failures are not fail stop!

How do we manage today?



CoMon PlanetLab Status (sort key: Name)

Part of the CoDeeN project
Updated Sat Feb 25 18:50:02 2006
legend
Node Summary: long, short Slice Summary: max, average, total Port Summary: all

#	Name Address	Resp Time SSH Status	Uptime Last CoTop	Busy CPU % Sys CPU %	Mem Size Mem Act %	Disk Size Disk % Used	Swap Used % GB Free	Date Drift	1 min load 5 min load	Swap In Swap Out	Disk In Disk Out	Timer max timer avg	Conn max Conn avg	FD test	Free CPU Free Mem	Tx Rate Rx Rate	Raw Ports ICMP Ports	DNS1 udp % DNS1 tcp %	DNS2 udp % DNS2 tcp %	Num Slices Live Slices
1	planetlab-1.npc.nicta.com.au 203.143.173.12	0.49 S good	32.6 D 5.0 M	4 1	0.99 35.5	183.0 2%	4.4 180.03	2-25-2006 18:50:09 0.77 S	0.35 0.14	0 0	10 263	331.6 11.1	6.51 1.07	0x0	97.6% 100	4 12	3 3	0.0 0.0	-1.0 -1.0	22 2
2	planet-lab-1.csse.monash.edu.au 130.194.64.162	0.47 S good	40.8 D 5.0 M	100 13	0.97 73.3	69.4 45%	44.2 38.32	2-25-2006 18:50:06 -2.39 S	11.66 7.14	14 0	899 830	441.0 11.3	4.86 0.90	0x0	87.0% 98	181 210	2 18	0.0 0.0	0.0 0.0	70 7
3	planet-lab-2.csse.monash.edu.au 130.194.64.163	0.47 S bad: 39.1 D	1.7 D 39.1 D	8 2	0.97 70.3	69.4 38%	0.1 43.05	1-17-2006 15:55:59 -39.1 D	4.25 2.91	0 0	0 940	42.0 11.0	6.00 1.57	0x0	61.7% 100	600 837	1 5	0.0 0.0	0.0 26.2	55 8
4	planetlab1.it.uts.edu.au 138.25.15.194	no response node down	351.9 D																	
5	planetlab2.it.uts.edu.au	no response node																		

How do we manage today?

CoMon Slice Status - mit_dht (sort key: Num Procs)

Updated Sat Feb 25 15:50:02 2006
Node Summary: long, short Slice Summary: max, average, total Port Summary: all

#	Node Name	1-min transmit	15-min transmit	1-min receive	15-min receive	Num Procs	Phys Mem MB	Virt Mem MB	CPU %
1	planetlab1.ifi.uio.no	0	0	0	0	1669	841.1	3759.8	29.5
2	zju1.6planetlab.edu.cn	10	7	7	6	15	38.9	87.8	1.8
3	planetlab2.unl.edu	11	9	10	8	12	98.9	196.1	0.0
4	planetlab2.simula.no	6	7	6	8	12	46.4	98.6	5.4
5	planetlab2.cs.virginia.edu	7	7	8	8	11	68.9	159.2	0.0
6	planetlab2.koganei.wide.ad.jp	7	7	7	7	11	95.3	149.4	13.0
7	planetlab3.nbgisp.com	8	24	5	5	11	66.1	143.3	3.2
8	planet2.calgary.canet4.nodes.planet-lab.org	7	7	6	6	10	190.7	281.6	0.0
9	thu2.6planetlab.edu.cn	0	0	0	0	9	11.9	53.3	0.0
10	planetlab1.een.orst.edu	9	10	7	8	8	92.2	163.1	1.7

How do we manage today?

```
Transaction: Ticket created by kyoungso@cs.princeton.edu

Subject: read-only fs on planetlab2.cnds.jhu.edu


[kyoungso@opus ~/codeen]$ ssh princeton_comon@planetlab2.cnds.jhu.edu
'cat > a'
/bin/bash: line 1: a: Read-only file system

Thanks,
--KyoungSoo
```

How do we manage today?

A maze of twisty little passages, all *different*

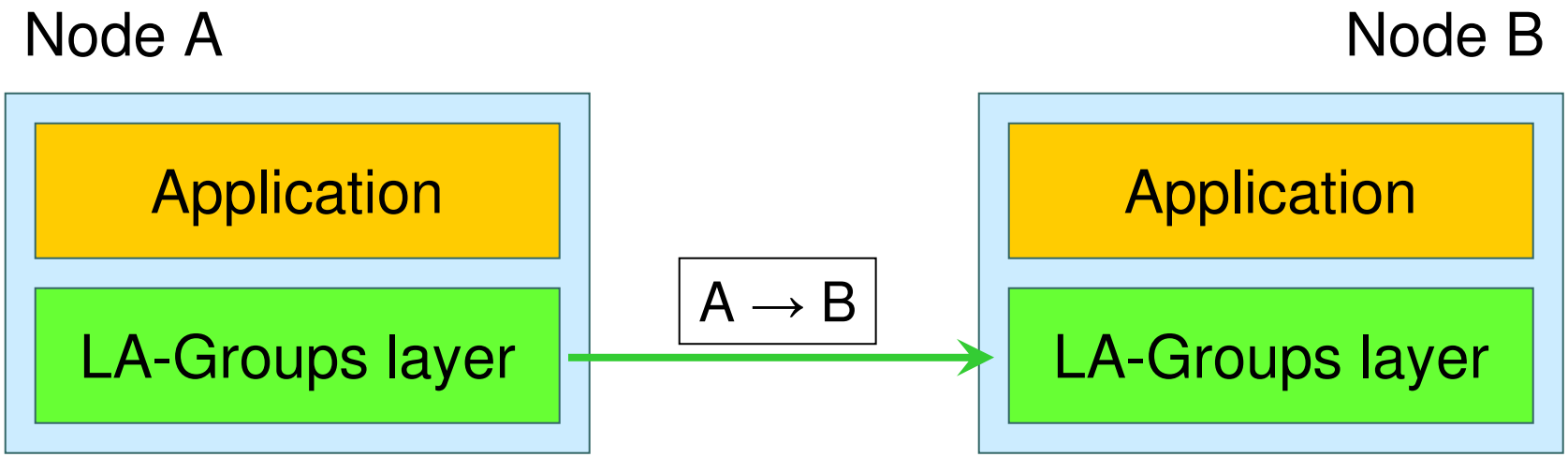
Something is needed…

- When running systems, weird stuff happens
- Once identify class of problems, write tests for them
- Give application more information →
  System makes more intelligent decision to work around
  - Graceful degradation
  - Give us time to go back and fix problem
  - Right now we don't utilize info systematically
- Today: Abstraction that collects and exposes information in structured way
- Goal: Simplify application design & implementation

Towards better system manageability

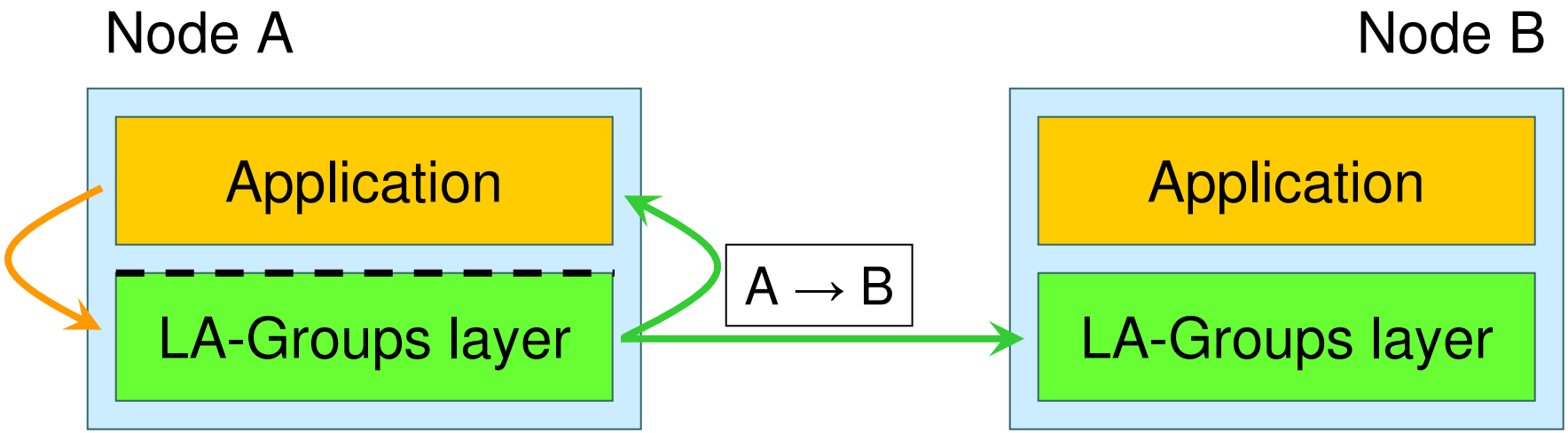
- Propose Link-Attestation Groups abstraction
  - Software abstraction to aid in management
  - “Group membership” subsystem
- Applying LA-Groups
  - DHTs
  - Multicast
  - File-sharing
- Only one point in design space

Link attestations



- Attestation: "A.app says B.app is correct"
  - Group identifier
  - Identities of attester (A) and attestee (B)
  - Expiration time (now + *t* secs)
  - Signed by attester (A)

The LA-Groups API

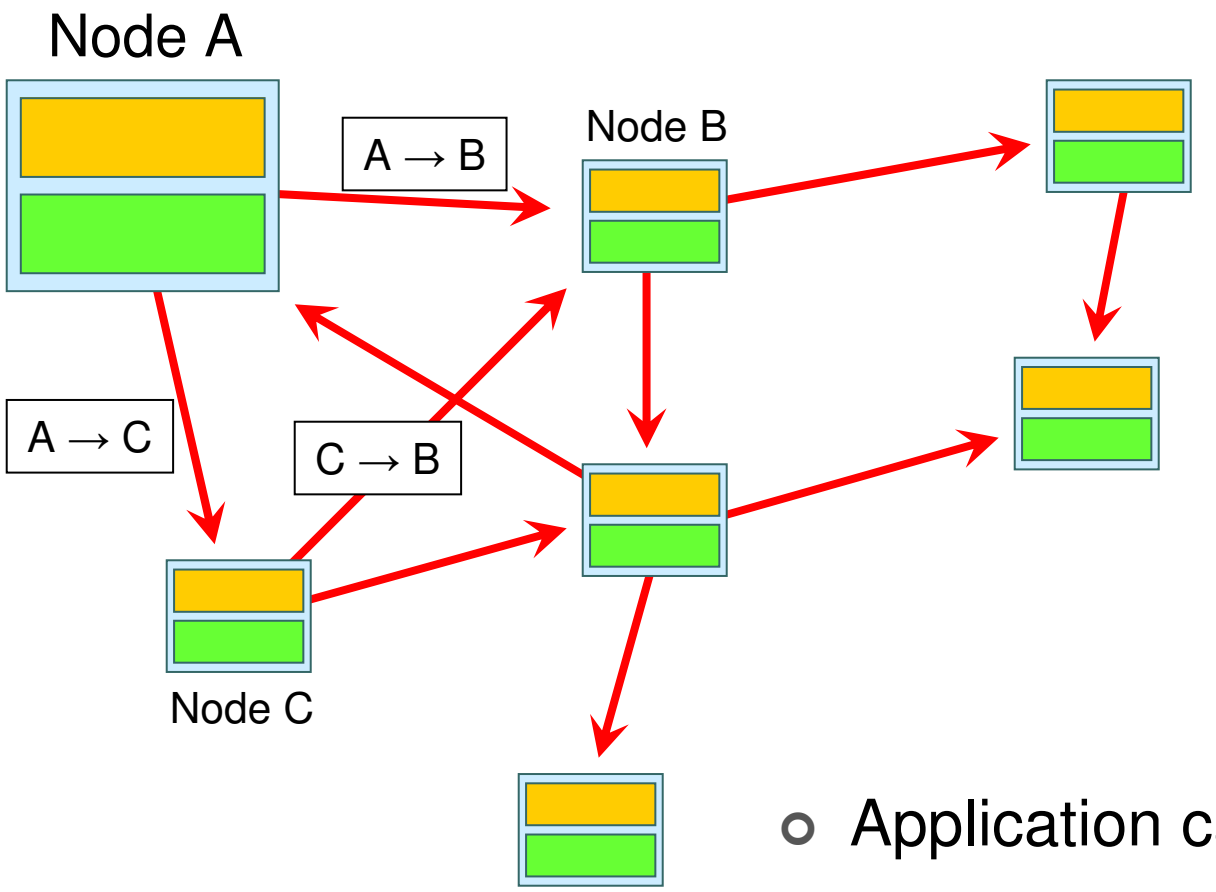


GID create()
void join(GID, nodeID[ ])

GID[ ] groups()
Graph attestations (GID)

void startAttest(GID, nodeID, info)
void stopAttest(GID, nodeID)

Graph of link attestations



A knows for GID:

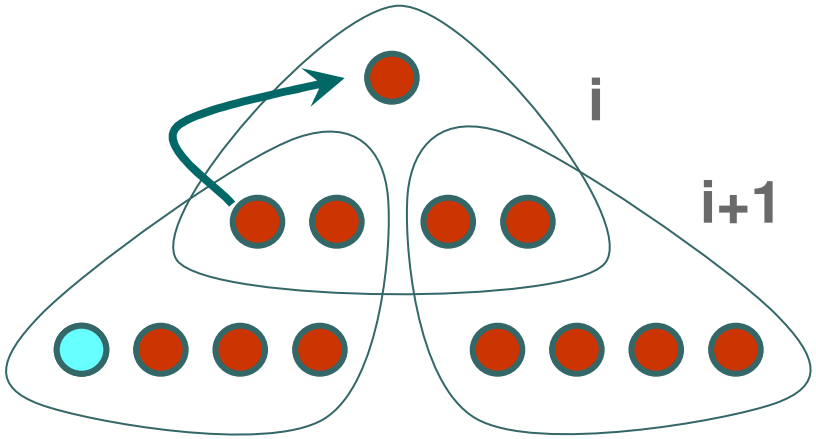
A → B
A → C
C → B
…

Think link-state

- Application calls startAttest()
- Subsystem generates, gossips, periodically refreshes attestations

LA-Groups for robust multicast

- Build fat multicast tree
- Goal:
  - Good nodes towards root



- LA-Group for parents and children
  - Correctness property:

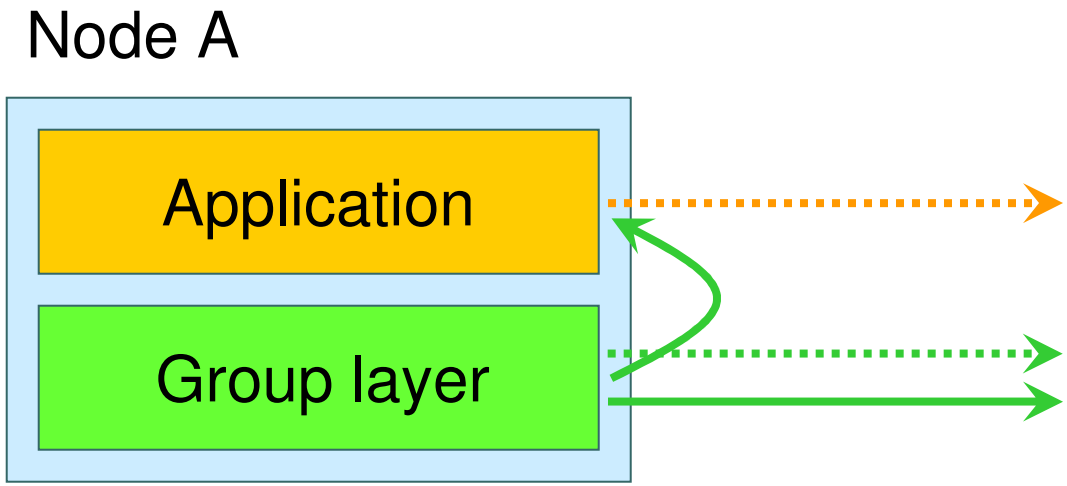
    Child says “Parent sent traffic at sufficient rate”
  - Level-i requires membership transcript from level i+1
  - If children fail to forward, must restart at bottom

When to startAttest() ?

- Unreliable failure detectors
  - Answers heartbeat: startAttest()
  - Fail to respond: stopAttest()
  - Yet applications aren't fail-stop!

- Application performs own battery of tests
  - Stateful anomaly detection
    - Network latency, application thruput, DoS attacks
  - Voting-based verification
    - Name resolution (DNS, pub keys), HTTP responses

vs. traditional membership systems



Group membership

- Layer tests liveness
- Uses failure reports
- Exports membership list

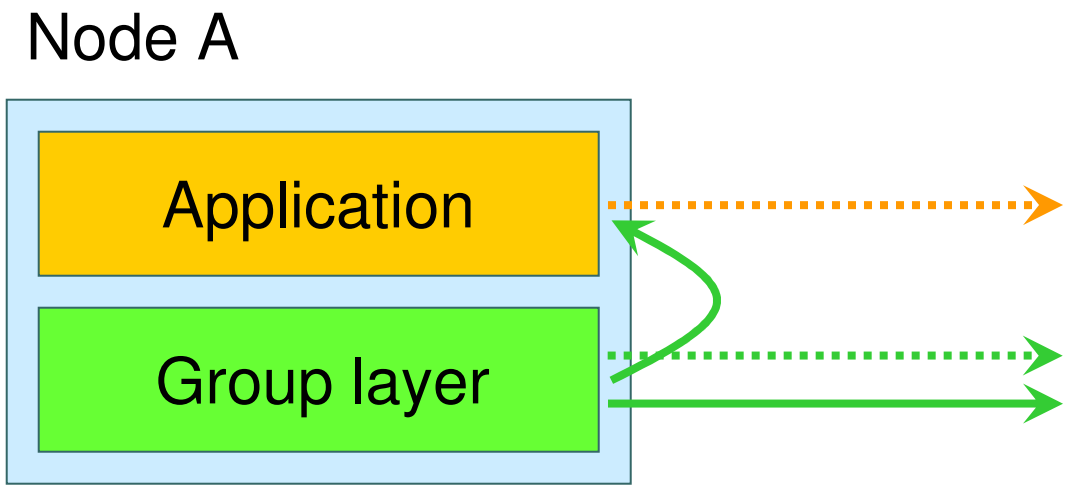
LA-Groups approach

- Application tests “correctness”
- Uses correctness attestations
- Exports attestation graph

Correctness, not failure, attestations

- Correctness attestations
  - Either both are correct or both are failed
  - More explicit that failure reports
    - Are failures per-link or global?
    - Either one or both are failed, but can't differentiate
    - Failure to receive report does not imply correctness
- Attestations form membership transcript
  - Node can show membership to non-group member
  - Crypto optimizations for aggregating signatures

vs. traditional membership systems



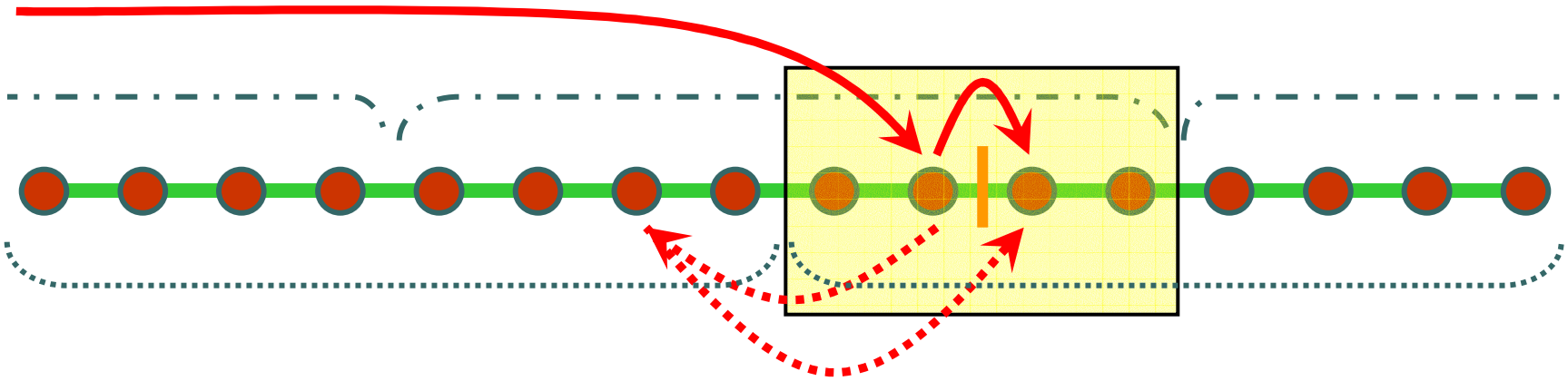
Group membership

- Layer tests liveness
- Uses failure reports
- Exports membership list

LA-Groups approach

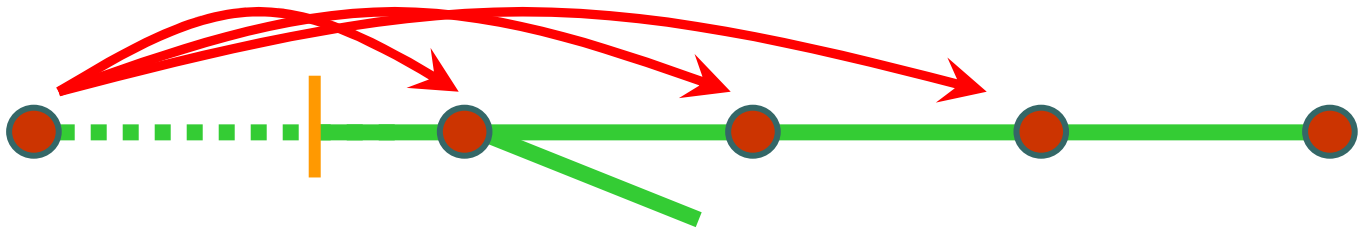
- Application tests “correctness”
- Uses correctness attestations
- Exports attestation graph

LA-Groups for robust routing



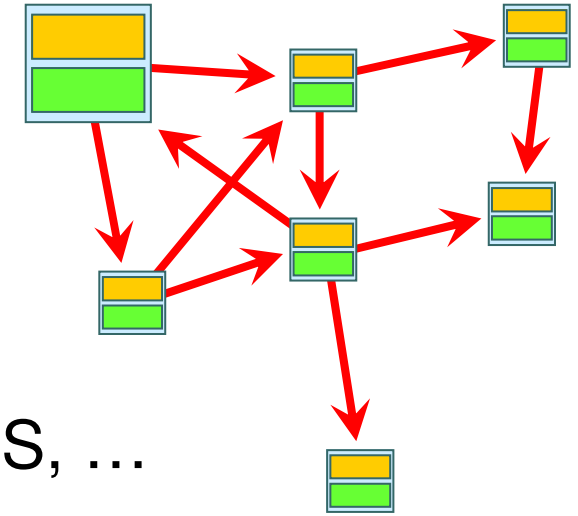
- Partition flat DHT ring into overlapping groups
  - Correctness test: heartbeats for link-level connectivity
  - Attestation graph gives topology at minimum
- Solves: Non-transitive routing
  - Use indirect hop to continue routing

LA-Groups for robust storage



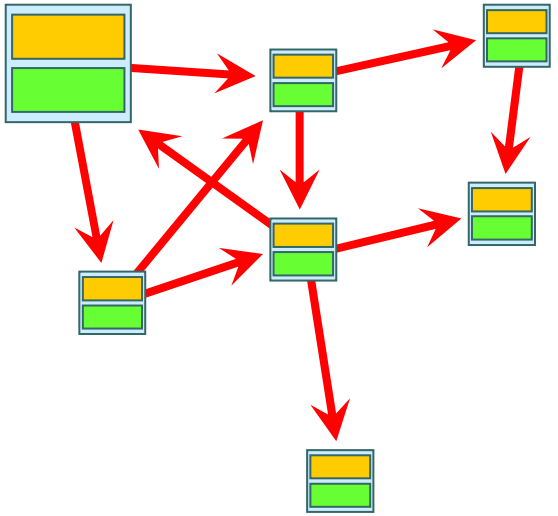
- DHTs store key-values on multiple successors
- Say only reachable via
  - If fails, key-value is lost
  - Replicas experience correlated failures
- Attestation graph captures correlation
  - Tune replication for desired fault-tolerance

LA-Groups for f2f



- Trust in partitionable systems
  - Backup, file sharing, cooperative IDS, …
  - “Trust, but verify”
- Correctness test: successfully returns content
- Use attestation graph to:
  - Tune replication
  - Verify result from *k* disjoint paths upon failures

Using graph properties…



- Multiple vertex-disjoint paths
  - Secure gossiping protocols
  - Decentralized key distribution
- Minimum vertex cut
  - Quorum systems
- Strongly-connected components
  - Structured routing overlays
  - Multi-hop wireless protocols
- Shortest path or max-flow on link capacity
  - Optimizing multicast transmission
  - Handling selfish peers in BitTorrent swarms
- LA-Groups makes these properties explicit

What's been traditional proposals?

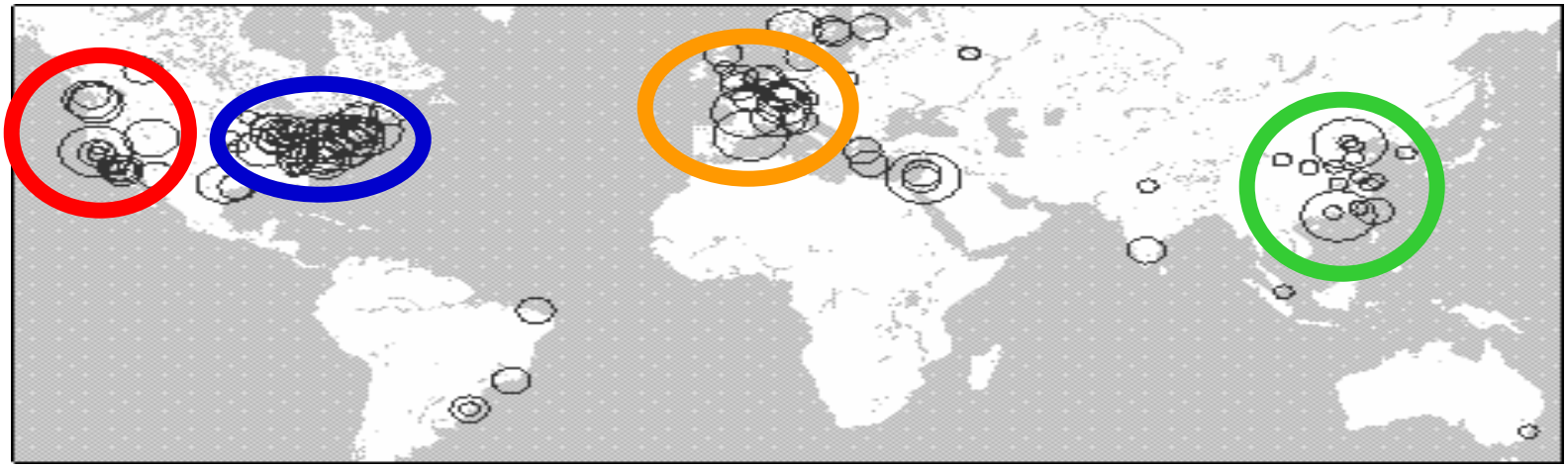
- Mask arbitrary failures
  - Virtual synchrony [Birman, …]
  - Replicated quorum systems [Malkhi/Reiter,…]
  - BFT replicated state machines [Liskov, …]

\+ abstraction generality and correctness

– systems don't experience uncorrelated failure:
  > f nodes can fail simultaneously

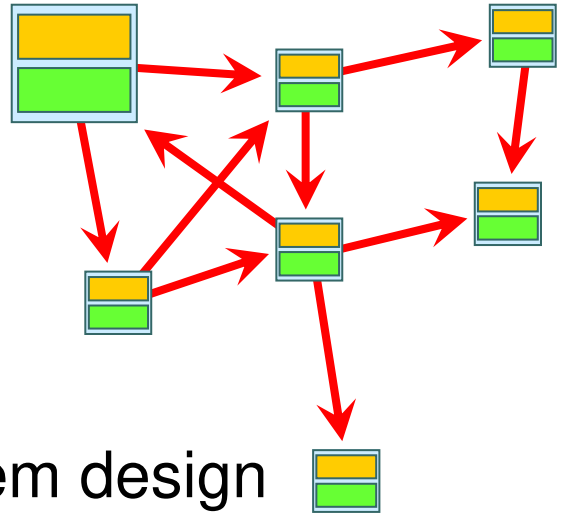
– often no global notion of failure

Future work: LA-Groups for CoralCDN



- Move all testing code to testing module, e.g.,
  - Receives incoming and sends outgoing relevant pkts
  - Compare GET responses with others' responses
- Group clusters of nearby proxies
- Redirect clients only to nodes with valid membership

Summary



- Presented LA-Groups
  - Software abstraction to simplify system design
  - Supports application-level notion of correctness
  - Exposes attestation graphs
  - Reason about system function vis-à-vis graph properties