CS 525 Advanced Distributed Systems Spring 2010

Indranil Gupta (Indy)

Failure Detectors and Membership Protocols

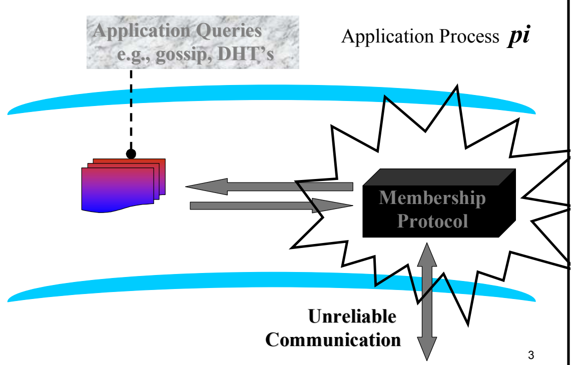
March 18, 2010

Acknowledgments: Dongyun Jin and Thuy Nguyen

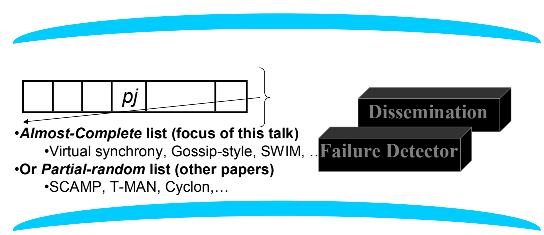
Target Settings

- Process 'group'-based systems
  - Clouds/Datacenters
  - Replicated servers
  - Distributed databases
- Crash-stop/Fail-stop process failures

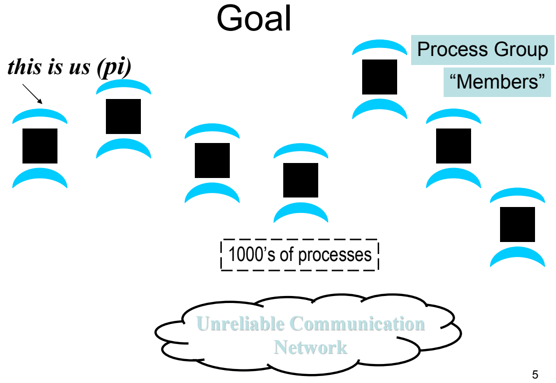
Group Membership Service



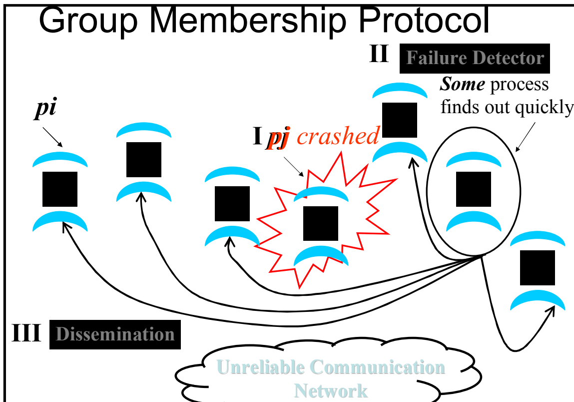
Two sub-protocols



Large Group: Scalability A Goal



Group Membership Protocol



I. *pj* crashes

- Nothing we can do about it!
- A frequent occurrence
- Common case rather than exception

II. Distributed Failure Detectors: Properties

- Completeness = each failure is detected
- Accuracy = there is no mistaken detection
- Speed
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

Distributed Failure Detectors: Properties

- Completeness
- Accuracy
- Speed
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

Impossible together in lossy networks [Chandra and Toueg]

Can then solve consensus!

What Real Failure Detectors Prefer

- Completeness → Guaranteed
- Accuracy → Partial/Probabilistic guarantee
- Speed
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

Failure Detector Properties

- Completeness → Guaranteed
- Accuracy → Partial/Probabilistic guarantee
- Speed → Time until ***some*** process detects the failure
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

Failure Detector Properties

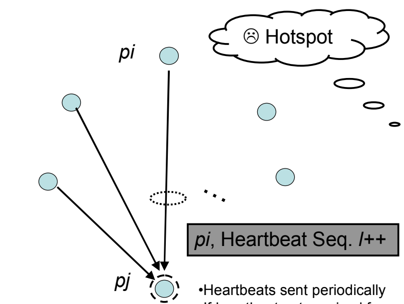
- Completeness → Guaranteed
- Accuracy → Partial/Probabilistic guarantee
- Speed → Time until ***some*** process detects the failure
  - Time to first detection of a failure
- Scale → No bottlenecks/single failure point
  - Equal Load on each member
  - Network Message Load

Failure Detector Properties

- Completeness
- Accuracy
- Speed
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

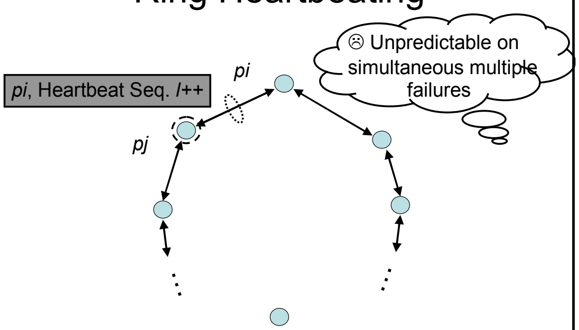
In spite of arbitrary simultaneous process failures

Centralized Heartbeating

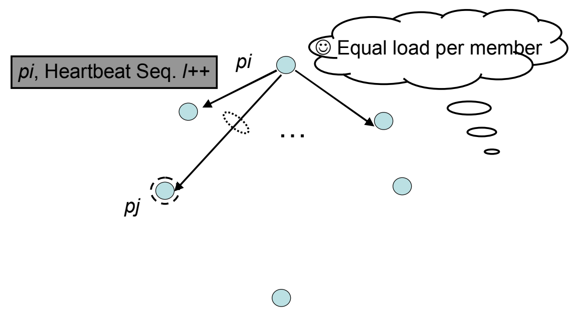


•Heartbeats sent periodically
•If heartbeat not received from *pi* within timeout, mark *pi* as failed

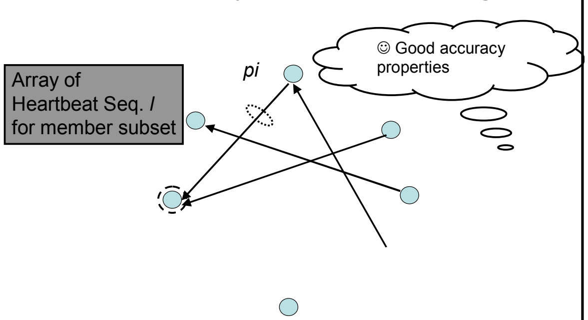
Ring Heartbeating



All-to-All Heartbeating



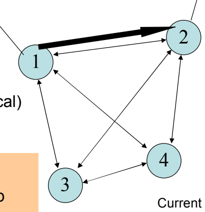
Gossip-style Heartbeating



Gossip-Style Failure Detection

1	10120	66
2	10103	62
3	10098	63
4	10111	65

Address — Heartbeat Counter — Time (local)



1	10118	64
2	10110	64
3	10090	58
4	10111	65

1	10120	70
2	10110	64
3	10098	70
4	10111	65

Protocol:

•Nodes periodically gossip their membership list

•On receipt, the local membership list is updated

Current time : 70 at node 2

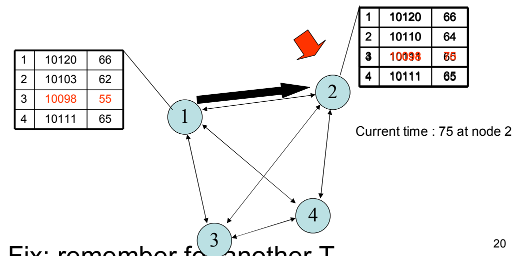
(asynchronous clocks)

Gossip-Style Failure Detection

- If the heartbeat has not increased for more than T_{fail} seconds, the member is considered failed
- And after $T_{cleanup}$ seconds, it will delete the member from the list
- Why two different timeouts?

Gossip-Style Failure Detection

- What if an entry pointing to a failed node is deleted right after T_{fail} seconds?

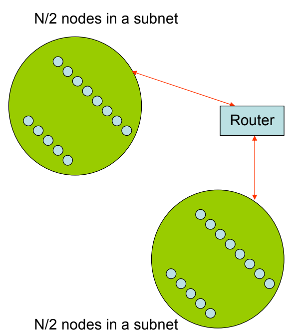


- Fix: remember for another T_{fail}

Multi-level Gossiping

- Network topology is hierarchical
- Random gossip target selection => core routers face O(N) load (Why?)
- Fix: Select gossip target in subnet I, which contains n_i nodes, with probability $1/n_i$
- Router load=O(1)
- Dissemination time=O(log(N))
  - Why?
- What about latency for multi-level topologies?

[Gupta et al, TPDS 06]

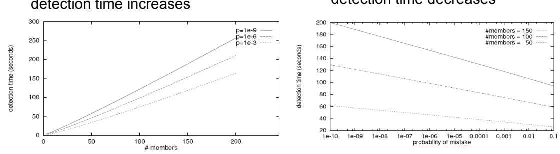


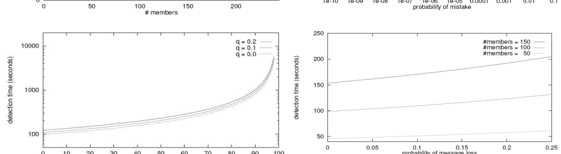
Analysis/Discussion

- What happens if gossip period T_{gossip} is decreased?
- A single heartbeat takes O(log(N)) time to propagate. So: N heartbeats take:
  - O(log(N)) time to propagate, if bandwidth allowed per node are allowed to be O(N)
  - O(N.log(N)) time to propagate, if bandwidth allowed per node is only O(1)
  - What about O(k) bandwidth?
- What happens to $P_{mistake}$ (false positive rate) as T_{fail} , $T_{cleanup}$ is increased?
- Tradeoff: False positive rate vs. detection time

Simulations

- As # members increases, the detection time increases
- As requirement is loosened, the detection time decreases





- As # failed members increases, the detection time increases significantly
- The algorithm is resilient to message loss

Failure Detector Properties …

- Completeness
- Accuracy
- Speed
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

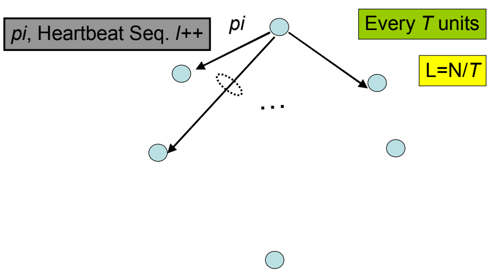
…Are application-defined Requirements

- Completeness → Guarantee always
- Accuracy → Probability *PM(T)*
- Speed → *T* time units
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

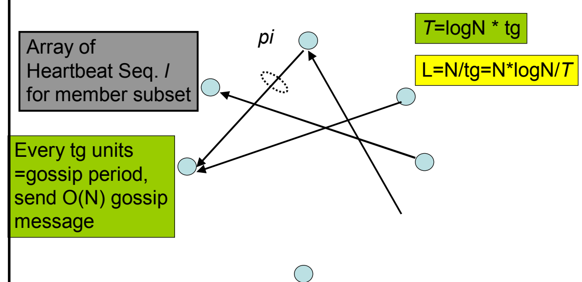
…Are application-defined Requirements

- Completeness → Guarantee always
- Accuracy → Probability *PM(T)*
- Speed → *T* time units
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load → N*L: Compare this across protocols

All-to-All Heartbeating



Gossip-style Heartbeating



What's the Best/Optimal we can do?

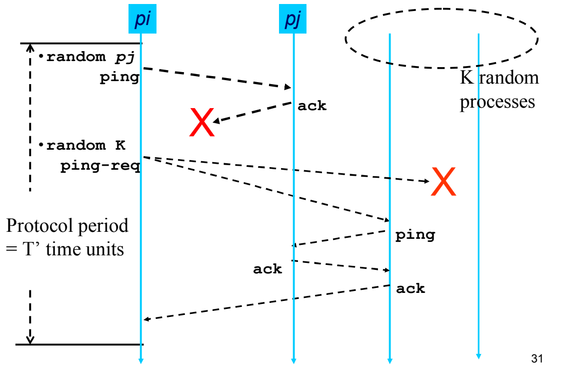
- *Worst case* load L*
  - as a function of *T*, *PM(T)*, N
  - Independent Message Loss probability p_{ml}
- $L^* = \frac{\log(PM(T))}{\log(p_{ml})} \cdot \frac{1}{T}$ (proof in PODC 01 paper)

Heartbeating

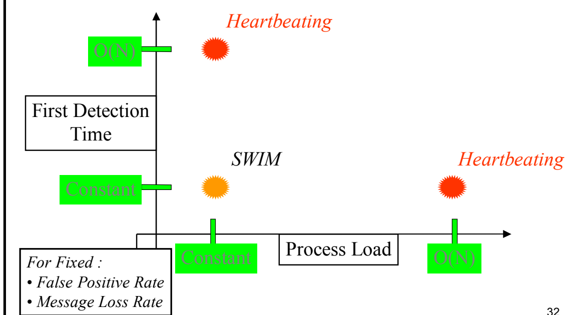
- Optimal L is independent of N
- All-to-all and gossip-based: sub-optimal
  - L=O(N/T)
  - try to achieve simultaneous detection at ***all*** processes
  - fail to distinguish *Failure Detection* and *Dissemination* components

Key:
- Separate the two components
- Use a non heartbeat-based Failure Detection Component

SWIM Failure Detector Protocol



SWIM versus Heartbeating



SWIM Failure Detector

Parameter	SWIM
First Detection Time	• Expected $\left[\frac{e}{e-1}\right]$ periods • Constant (independent of group size)
Process Load	• Constant per period • < 8 L* for 15% loss
False Positive Rate	• Tunable • Falls exponentially as load is scaled
Completeness	• Deterministic time-bounded • Within O(log(N)) periods w.h.p.

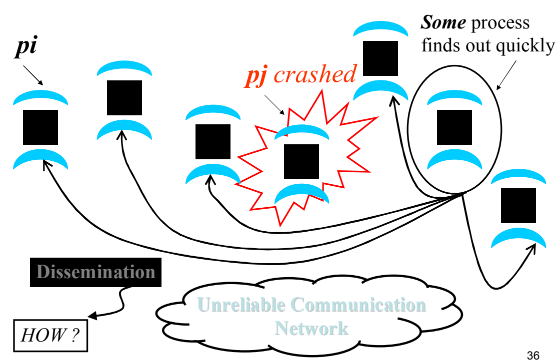
Accuracy, Load

- *PM(T)* is exponential in *K*. Also depends on *pml* (and *pf*)
  - See paper
- $\frac{L}{L^*} < 28$ $\frac{E[L]}{L^*} < 8$ for up to 15 % loss rates

Detection Time

- Prob. of being pinged in T'= $1-(1-\frac{1}{N})^{N-1} = 1-e^{-1}$
- E[*T*] = $T'.\frac{e}{e-1}$
- Completeness: *Any* alive member detects failure
  - Within worst case *O(N)* protocol periods

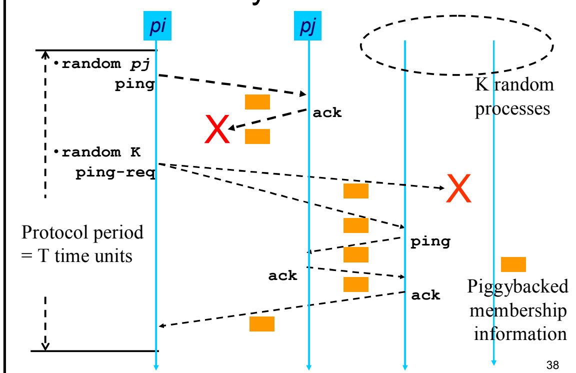
III. Dissemination



Dissemination Options

- Multicast (Hardware / IP)
  - unreliable
  - multiple simultaneous multicasts
- Point-to-point (TCP / UDP)
  - expensive
- Zero extra messages: Piggyback on Failure Detector messages
  - Infection-style Dissemination

Infection-style Dissemination



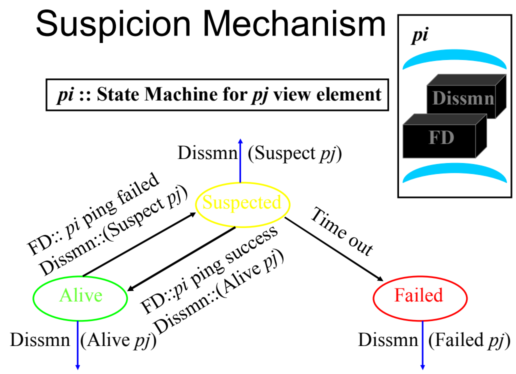
Infection-style Dissemination

- Epidemic style dissemination
  - After $\lambda.\log(N)$ protocol periods, $N^{-(2\lambda-2)}$ processes would not have heard about an update
- Maintain a buffer of recently joined/evicted processes
  - Piggyback from this buffer
  - Prefer recent updates
- Buffer elements are garbage collected after a while
  - After $\lambda.\log(N)$ protocol periods; this defines weak consistency

Suspicion Mechanism

- False detections, due to
  - Perturbed processes
  - Packet losses, e.g., from congestion
- Indirect pinging may not solve the problem
  - e.g., correlated message losses near pinged host
- Key: *suspect* a process before *declaring* it as failed in the group

Suspicion Mechanism



Suspicion Mechanism

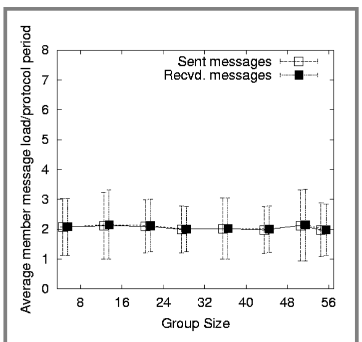
- Distinguish multiple suspicions of a process
  - Per-process *incarnation number*
  - *Inc #* for *pi* can be incremented only by *pi*
    - e.g., when it receives a (Suspect, *pi*) message
  - Somewhat similar to DSDV
- Precedence rules for (Alive, inc #), (Suspect inc #), (Failed, inc #)
  - See paper

Time-bounded Completeness

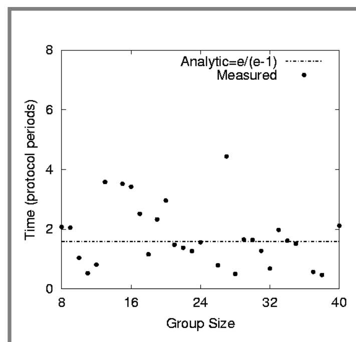
- Key: select each membership element once as a ping target in a traversal
  - Round-robin pinging
  - Random permutation of list after each traversal
- Each failure is detected in worst case 2N-1 (local) protocol periods
- Preserves FD properties

Results from an Implementation

- Current implementation
  - Win2K, uses Winsock 2
  - Uses only UDP messaging
  - 900 semicolons of code (including testing)
- Experimental platform
  - Galaxy cluster: diverse collection of commodity PCs
  - 100 Mbps Ethernet
- Default protocol settings
  - Protocol period=2 s; K=1; G.C. and Suspicion timeouts=3*ceil[log(N+1)]
- No partial membership lists observed in experiments



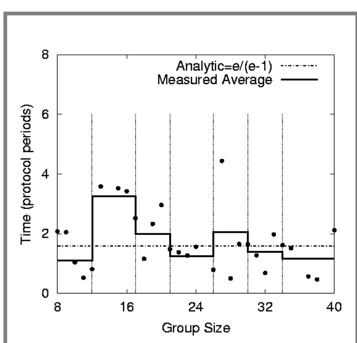
Per-process Send and Receive Loads are independent of group size



T1

T1+T2+T3

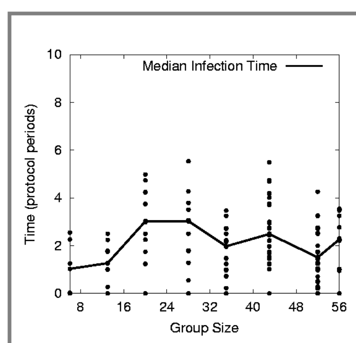
Time to First Detection of a process failure



T1

T1+T2+T3

Time to First Detection of a process failure apparently uncorrelated to group size

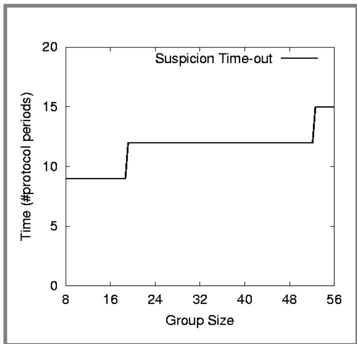


\+

T2

T1+T2+T3

Membership Update Dissemination Time is low at high group sizes

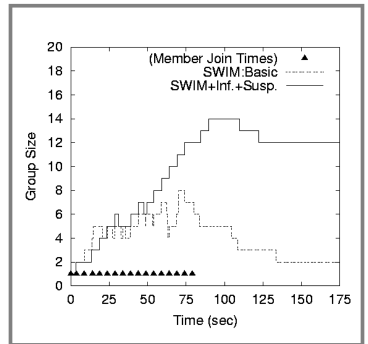


+

T3

T1+T2+T3

Excess time taken by
Suspicion Mechanism



Benefit of Suspicion Mechanism:
Per-process 10% synthetic packet loss

More discussion points

- It turns out that with a partial list that is *uniformly random,* gossiping retains same properties as with complete lists
  - Why?
  - Partial membership protocols
    - SCAMP, Cyclon, TMAN, …
- Gossip-style failure detection underlies
  - Astrolabe
  - Amazon EC2/S3 (rumored!)
- SWIM used in
  - CoralCDN/Oasis anycast service: http://oasis.coralcdn.org
  - Mike Freedman used suspicion mechanism to blackmark frequently-failing nodes

Questions