

CS 525
Advanced Distributed
Systems
Spring 2011

Indranil Gupta (Indy)

Membership Protocols (and Failure
Detectors)

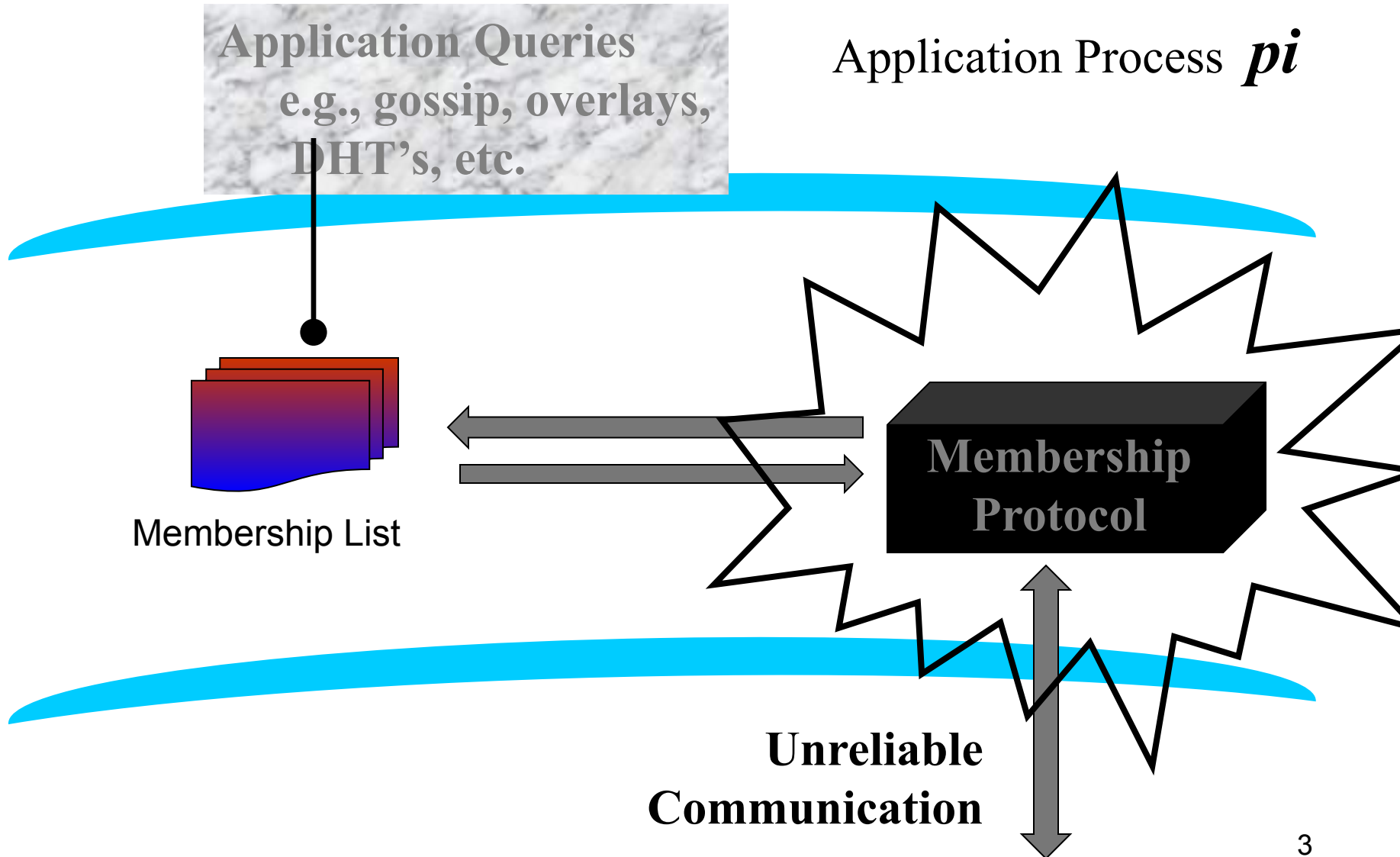
March 31, 2011

Target Settings

- Process 'group'-based systems
 - Clouds/Datacenters
 - Replicated servers
 - Distributed databases

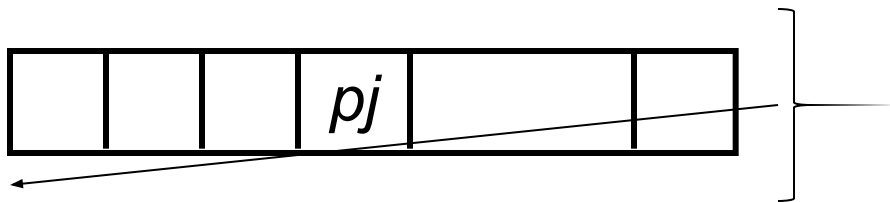
- Crash-stop/Fail-stop process failures

Group Membership Service



Two sub-protocols

Application Process pi



• **Almost-Complete list (focus of this talk)**

• Gossip-style, SWIM, Virtual synchrony, ...

• **Or Partial-random list (other papers)**

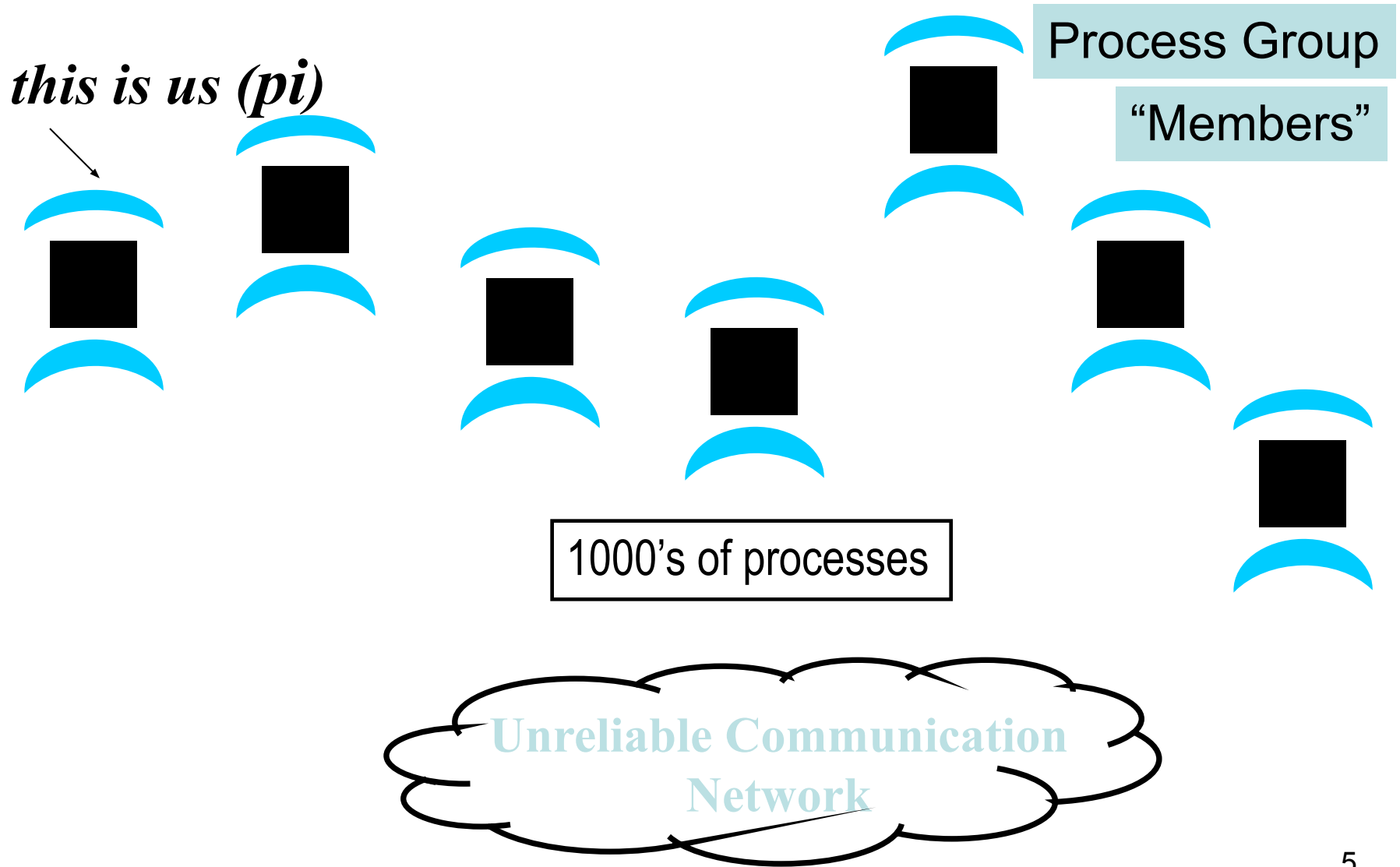
• SCAMP, T-MAN, Cyclon, ...



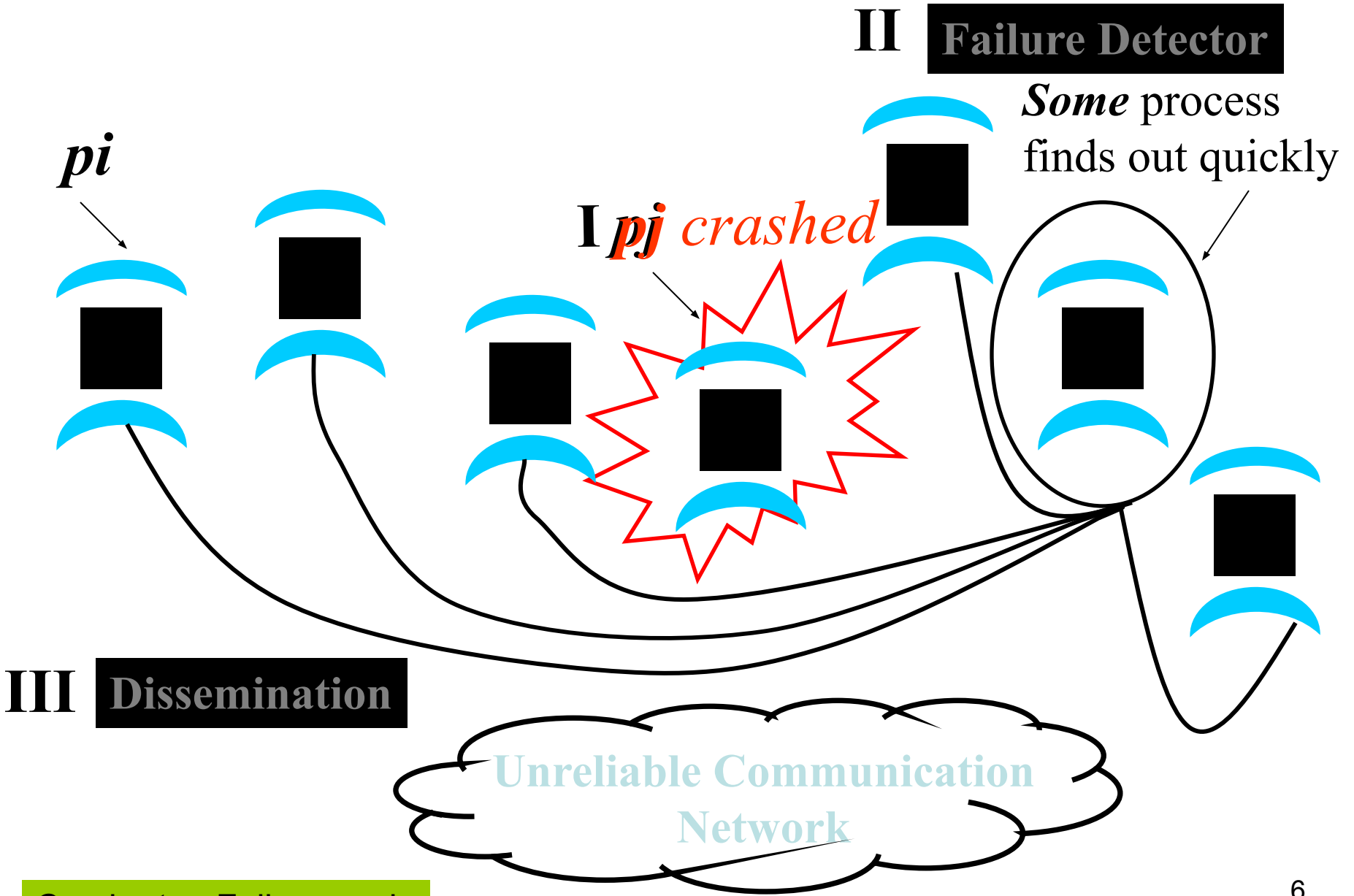
**Unreliable
Communication**

Large Group: Scalability A

Goal



Group Membership Protocol



Crash-stop Failures only

I. *pj* crashes

- Nothing we can do about it!
- A frequent occurrence
- Common case rather than exception
- Frequency goes up at least linearly with size of datacenter

II. Distributed Failure Detectors: Desirable 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

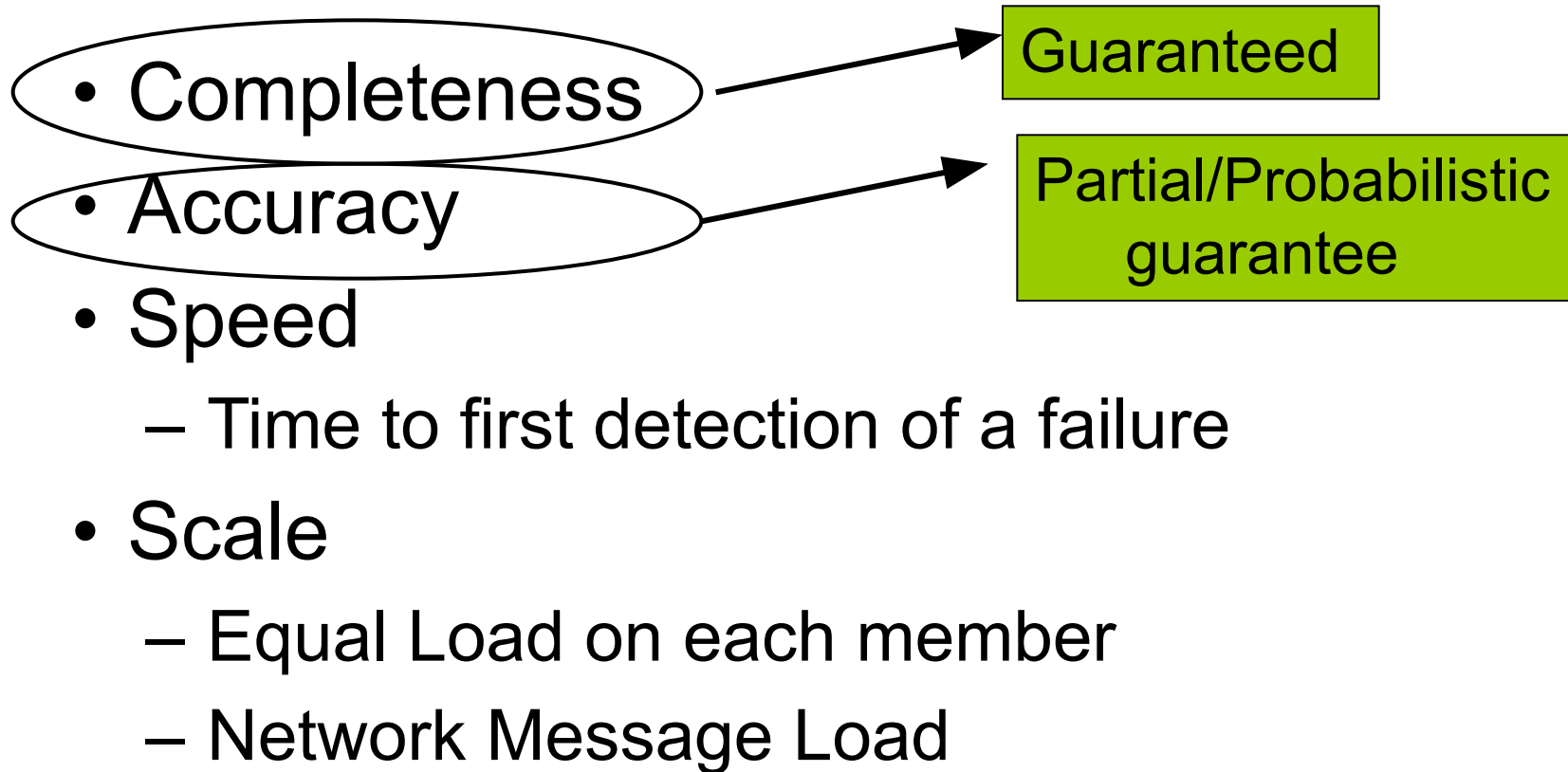
- Completeness
 - Accuracy
- 

Impossible together in lossy networks [Chandra and Toueg]

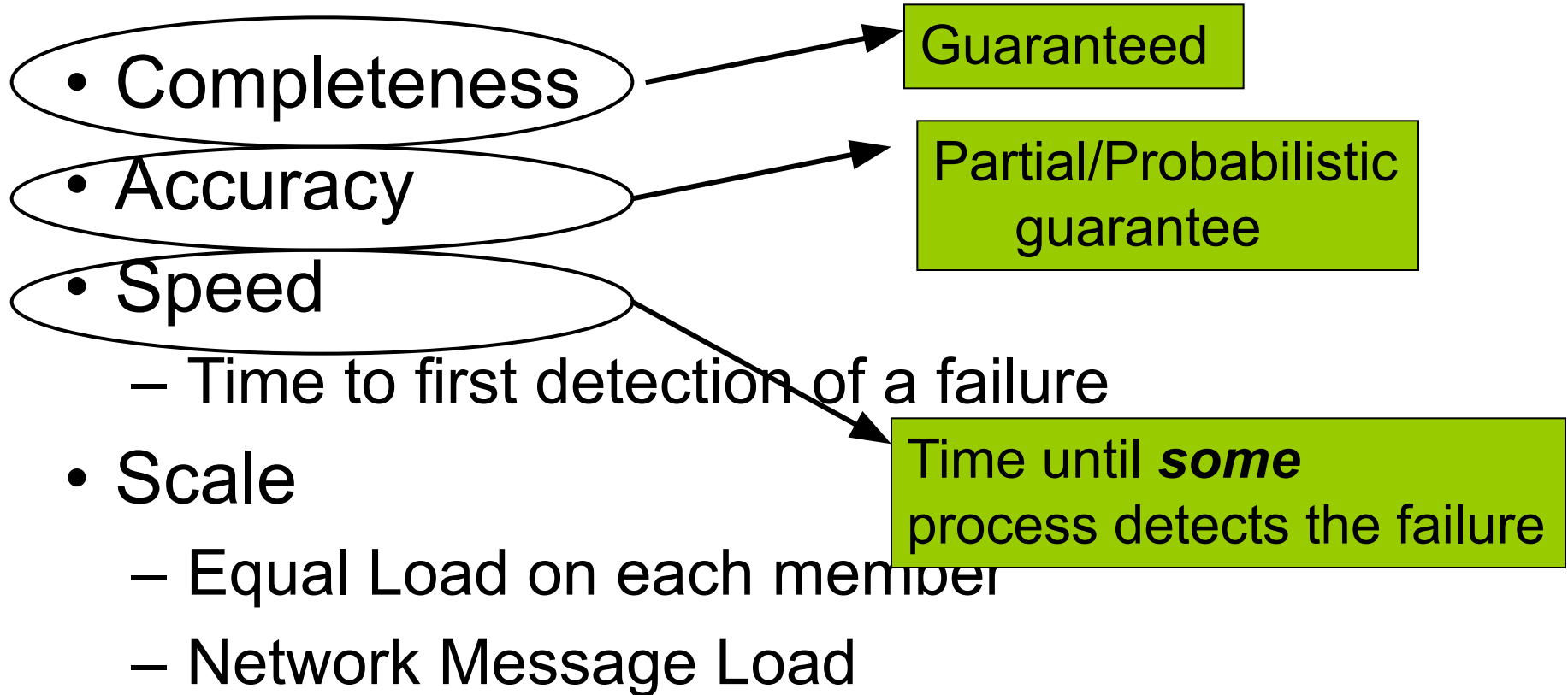
If possible, then can solve consensus!

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

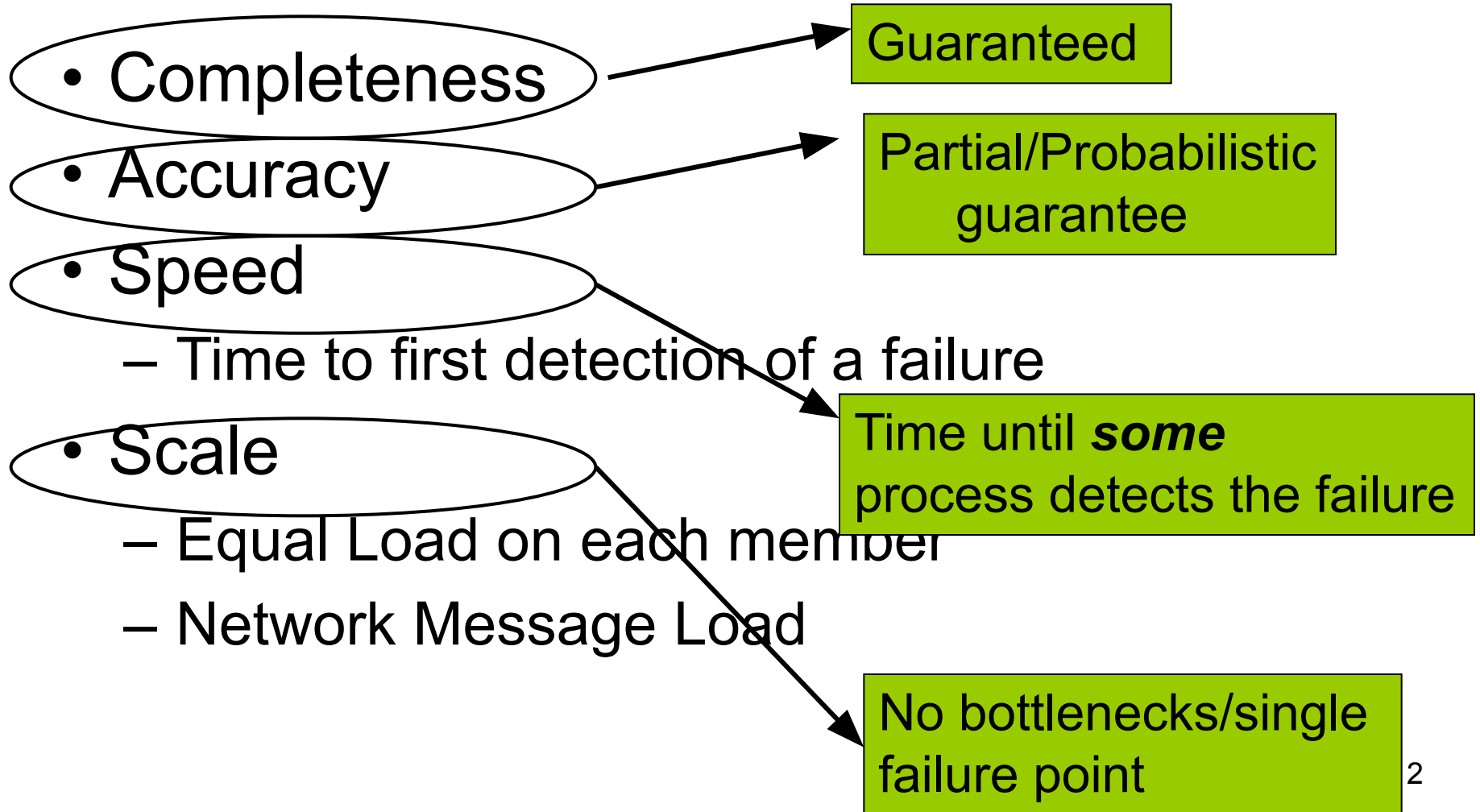
What Real Failure Detectors Prefer



Failure Detector Properties

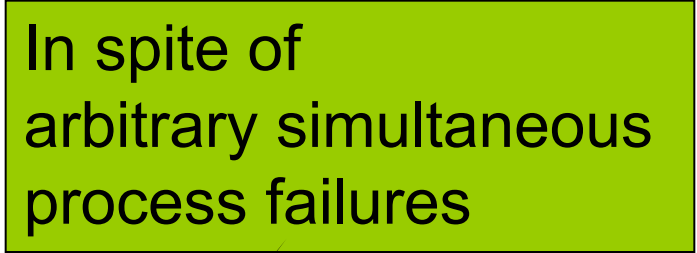


Failure Detector Properties



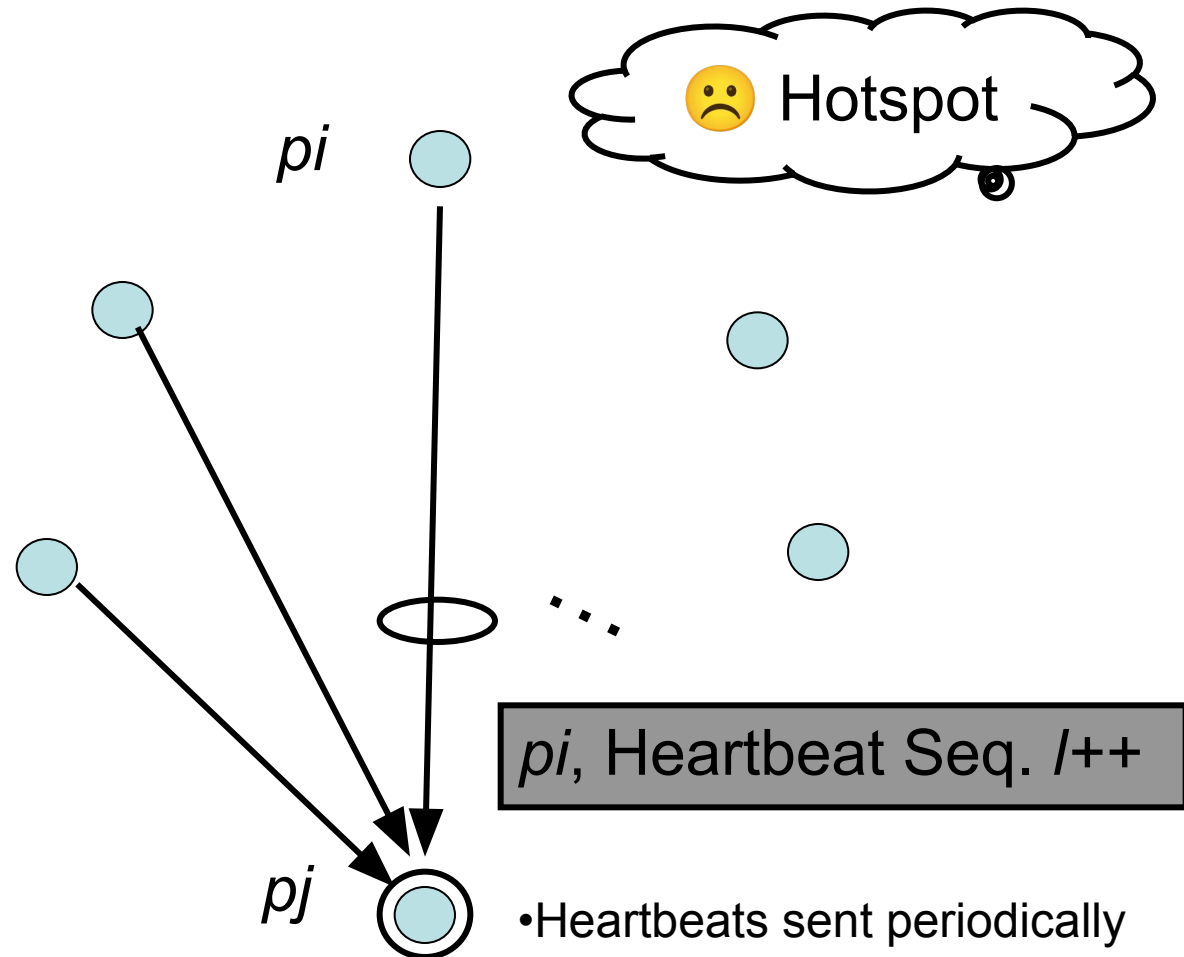
Failure Detector Properties

- 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

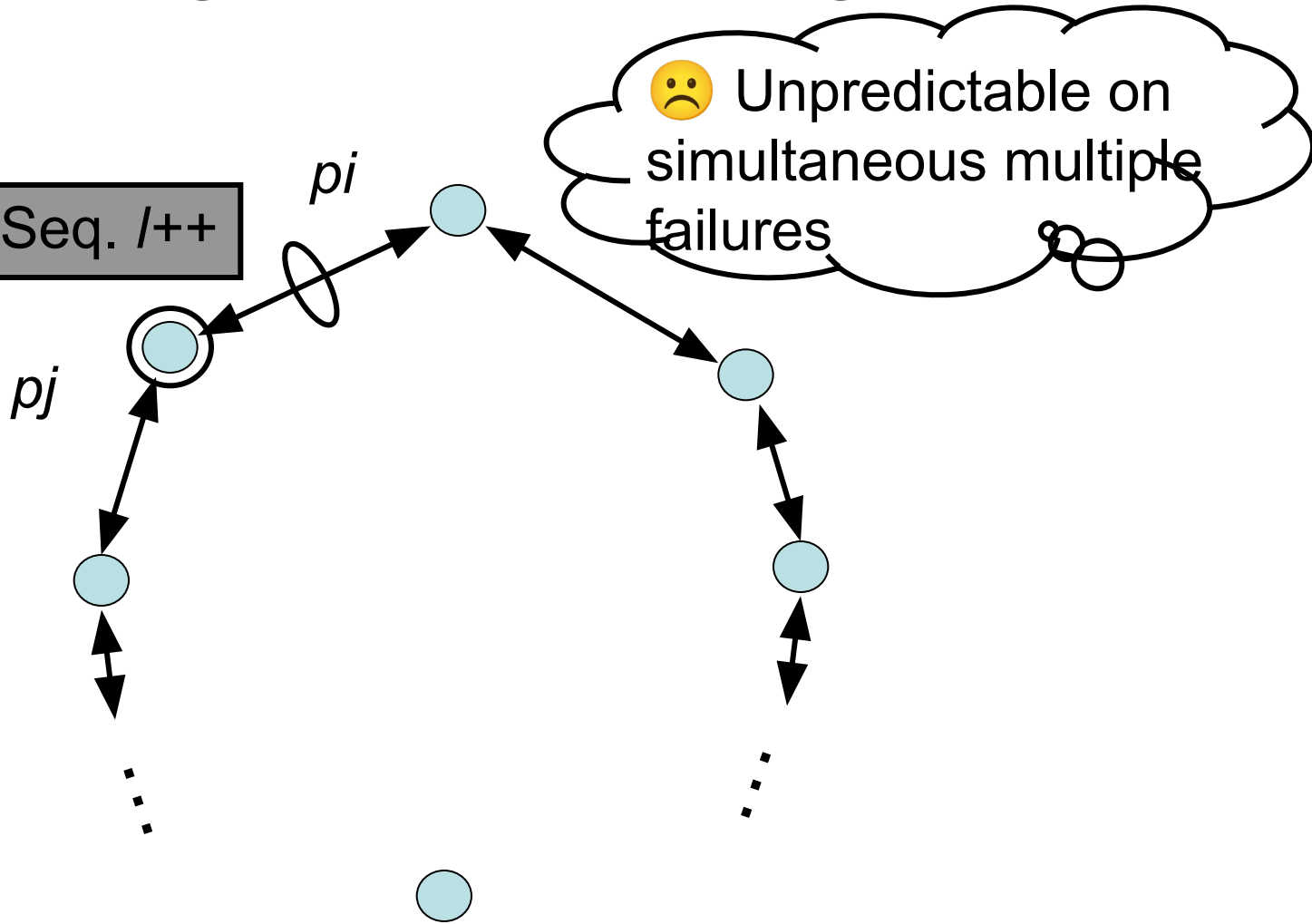
Centralized Heartbeating



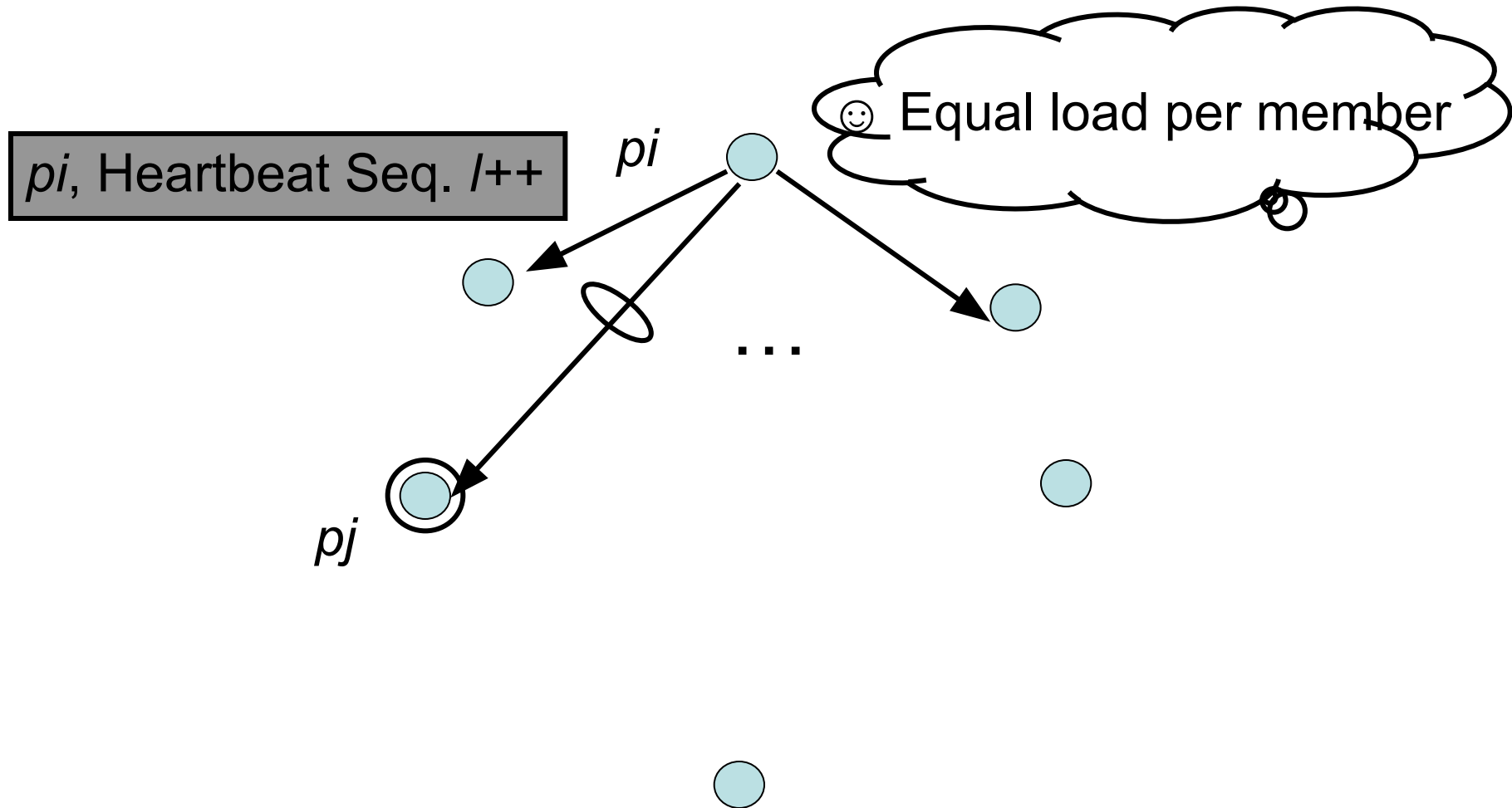
- Heartbeats sent periodically
- If heartbeat not received from p_i within timeout, mark p_i as failed

Ring Heartbeating

p_i , Heartbeat Seq. $l++$

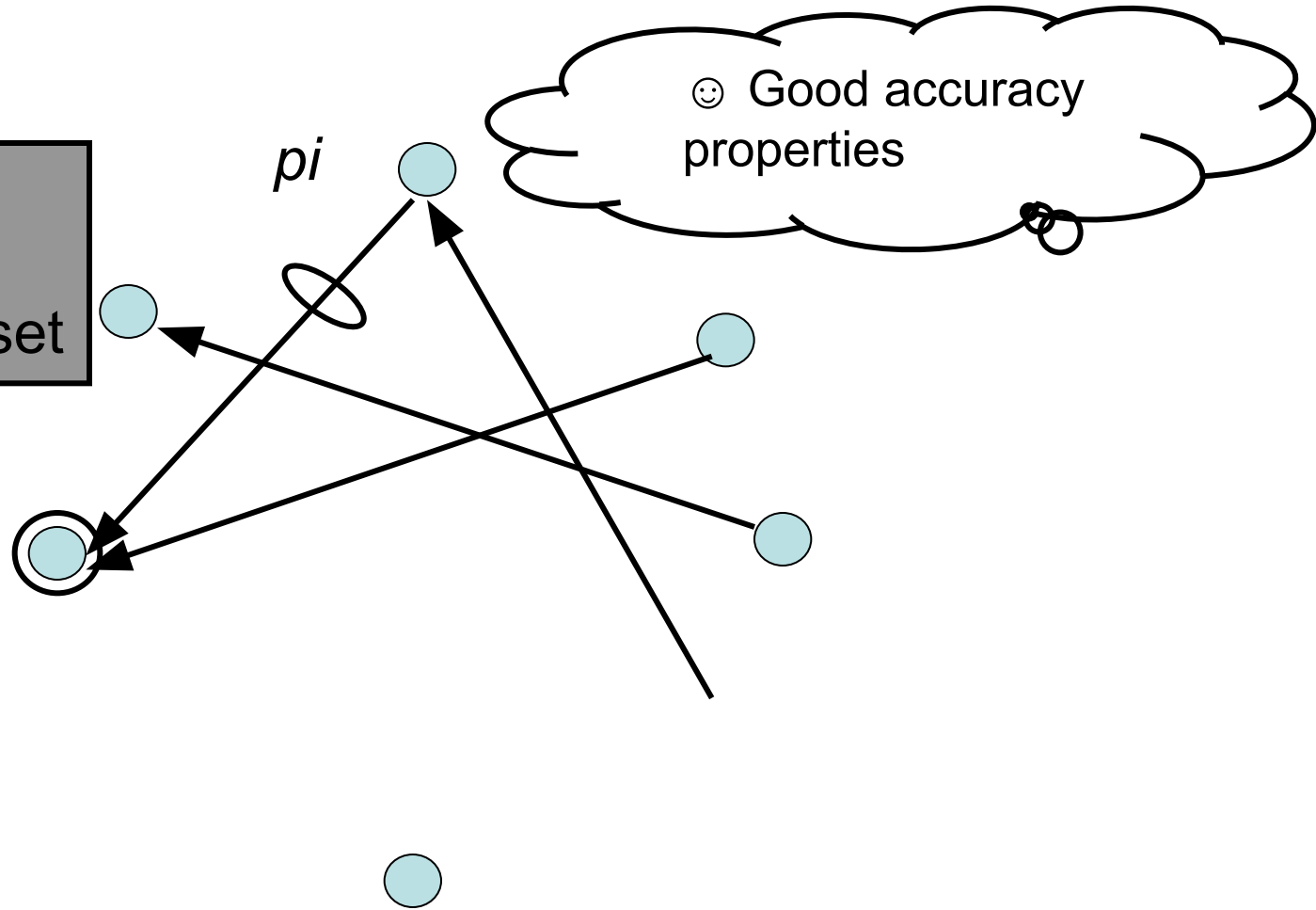


All-to-All Heartbeating

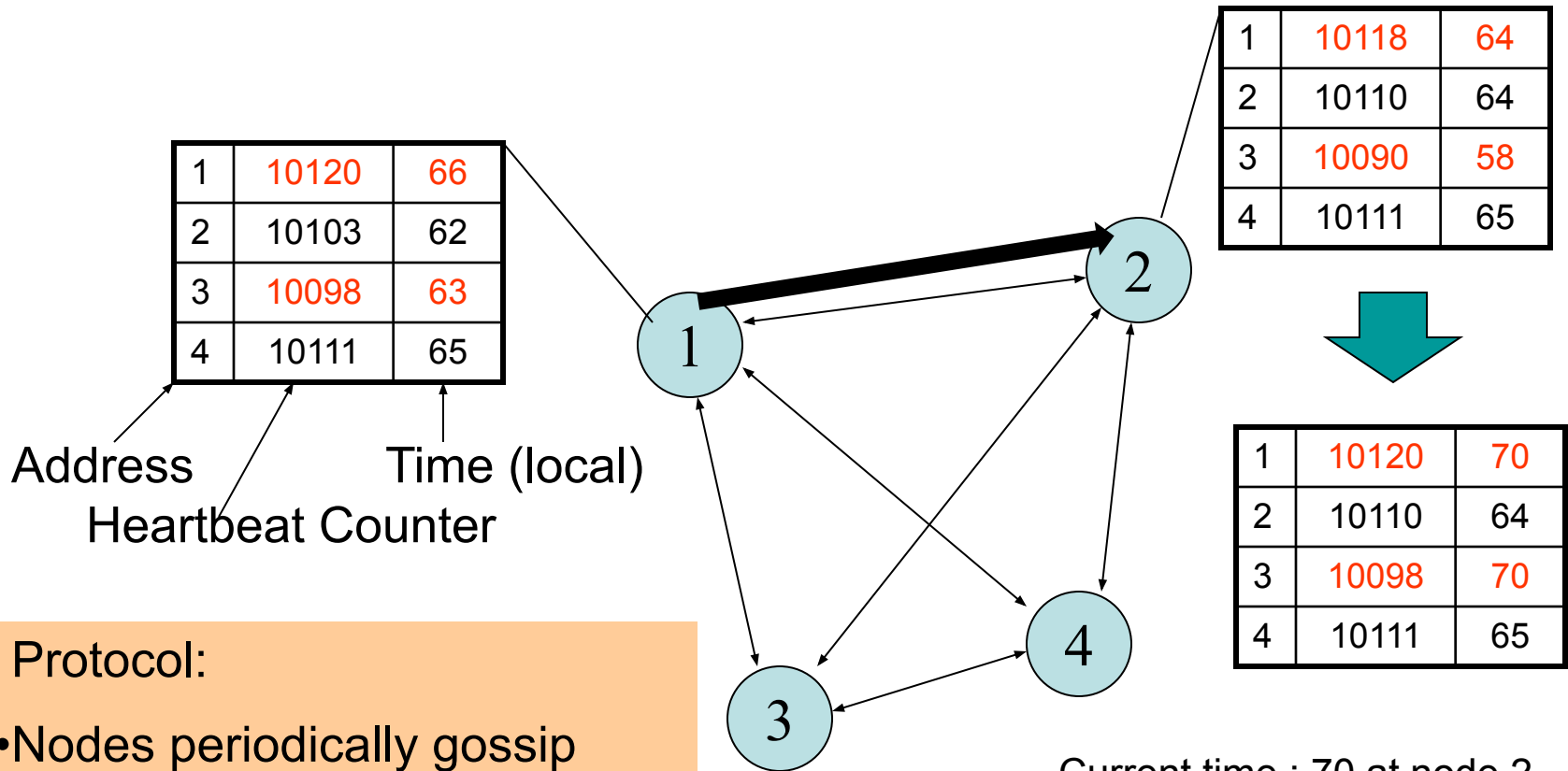


Gossip-style Heartbeating

Array of
Heartbeat Seq. l
for member subset



Gossip-Style Failure Detection



Protocol:

- Nodes periodically gossip their membership list
- On receipt, the local membership list is updated

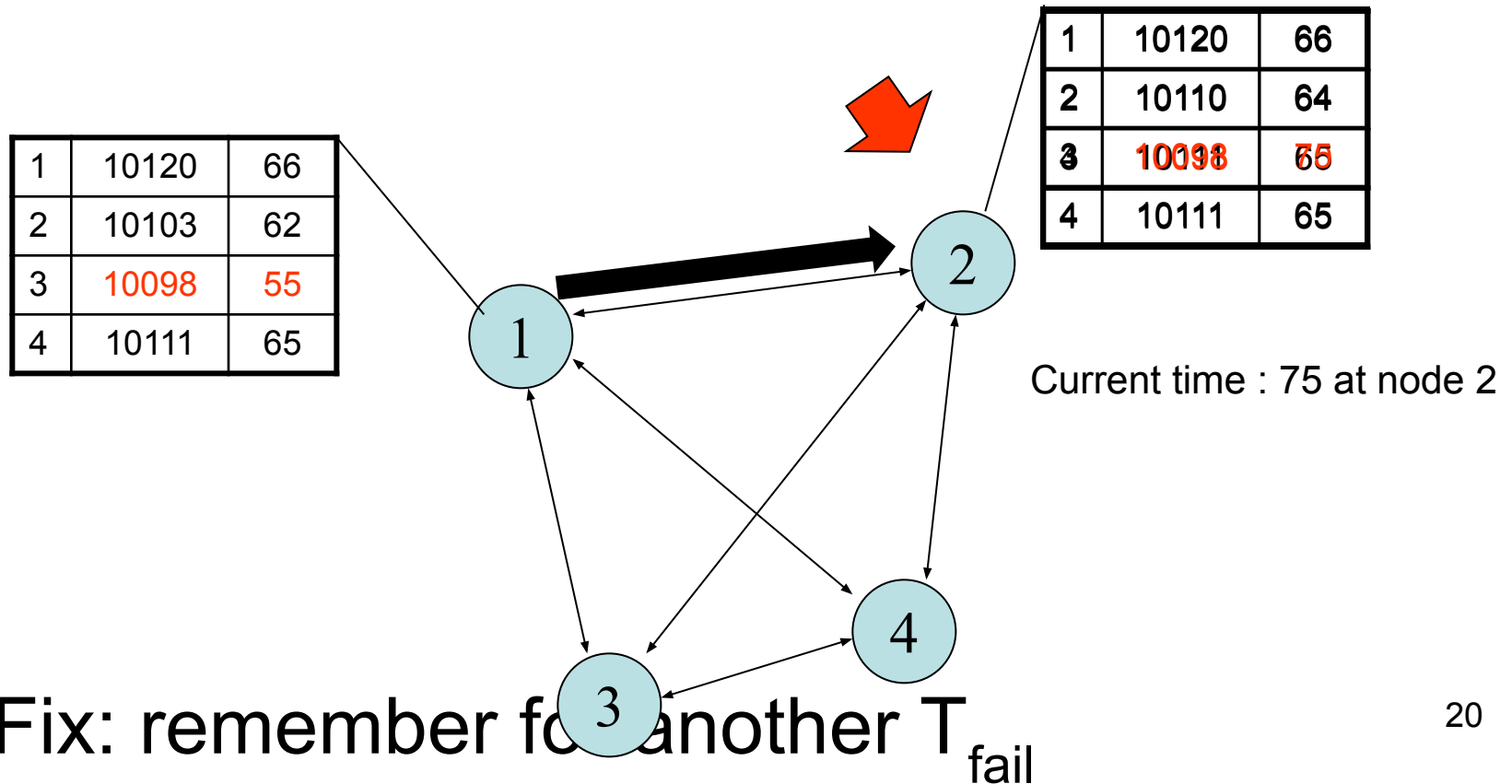
Current time : 70 at node 2
(asynchronous clocks)

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?

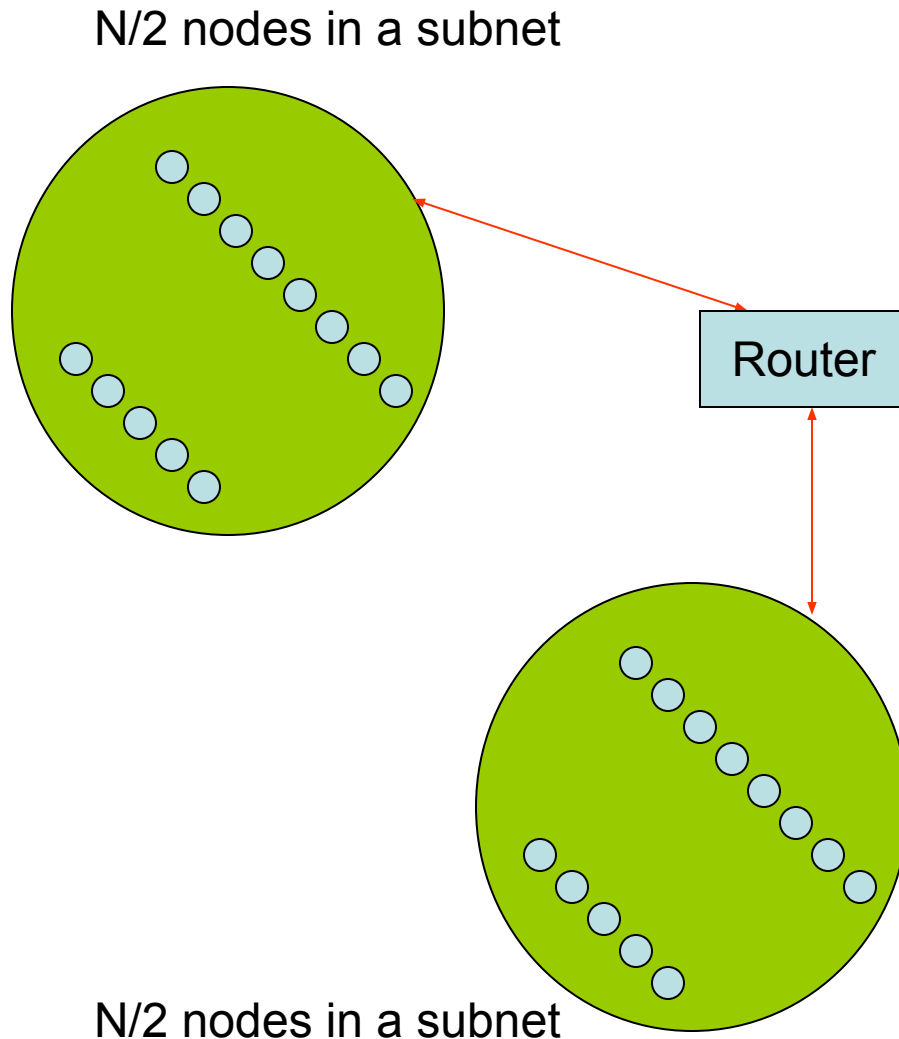
Gossip-Style Failure Detection

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



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]

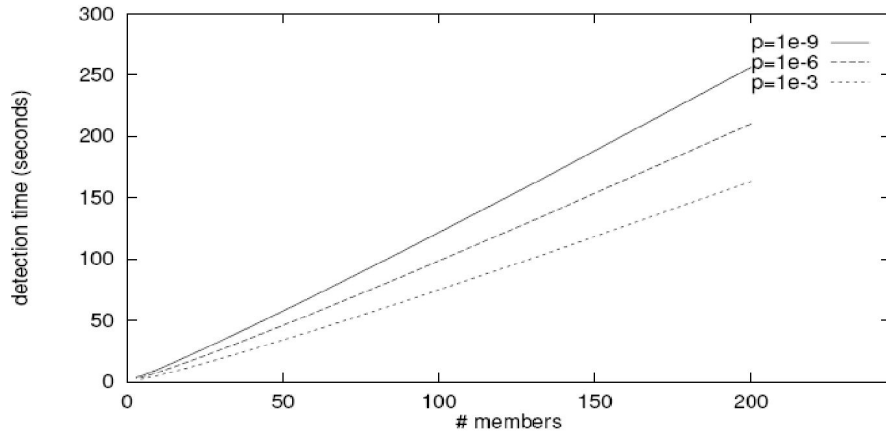


Analysis/Discussion

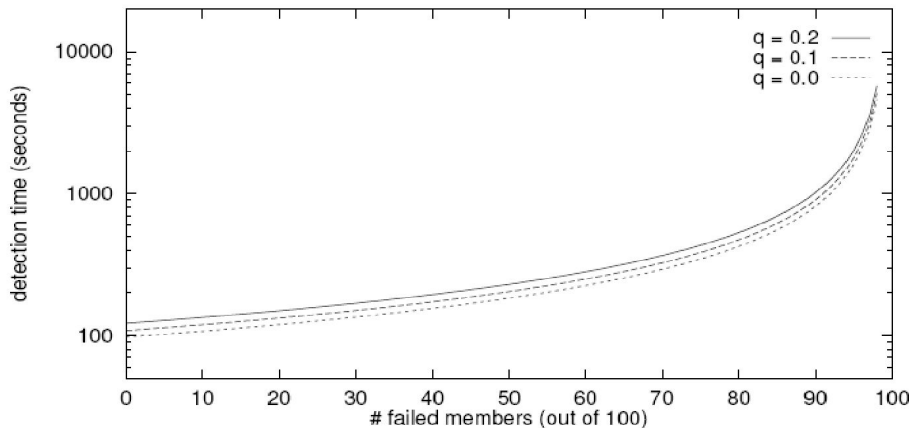
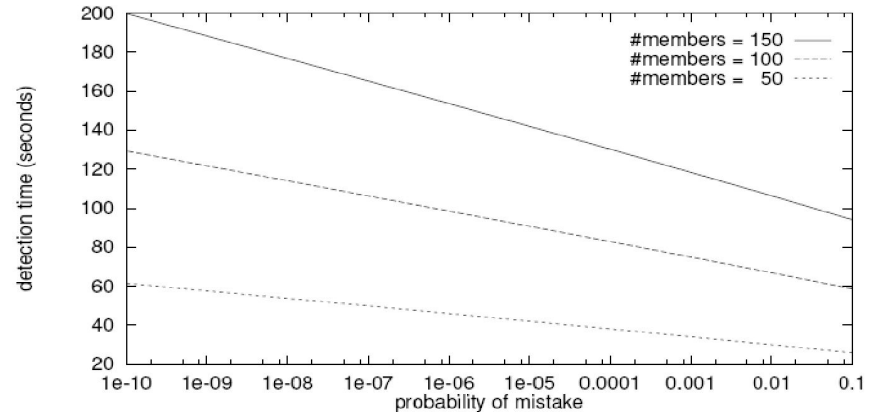
- What happens if gossip period T_{gossip} is decreased?
- A single heartbeat takes $O(\log(N))$ time to propagate. So: N heartbeats take:
 - $O(\log(N))$ time to propagate, if bandwidth allowed per node is allowed to be $O(N)$
 - $O(N \cdot \log(N))$ time to propagate, if bandwidth allowed per node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

Simulations

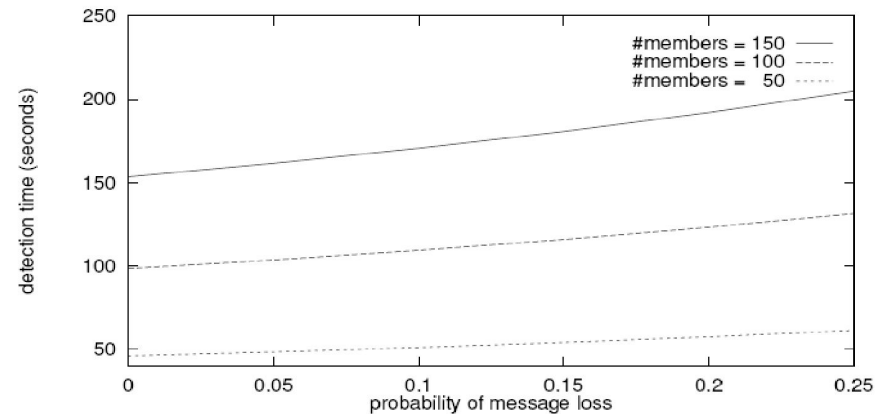
- As # members increases, the detection time increases



- As requirement is loosened, the detection time decreases



- As # failed members increases, the detection time increases significantly

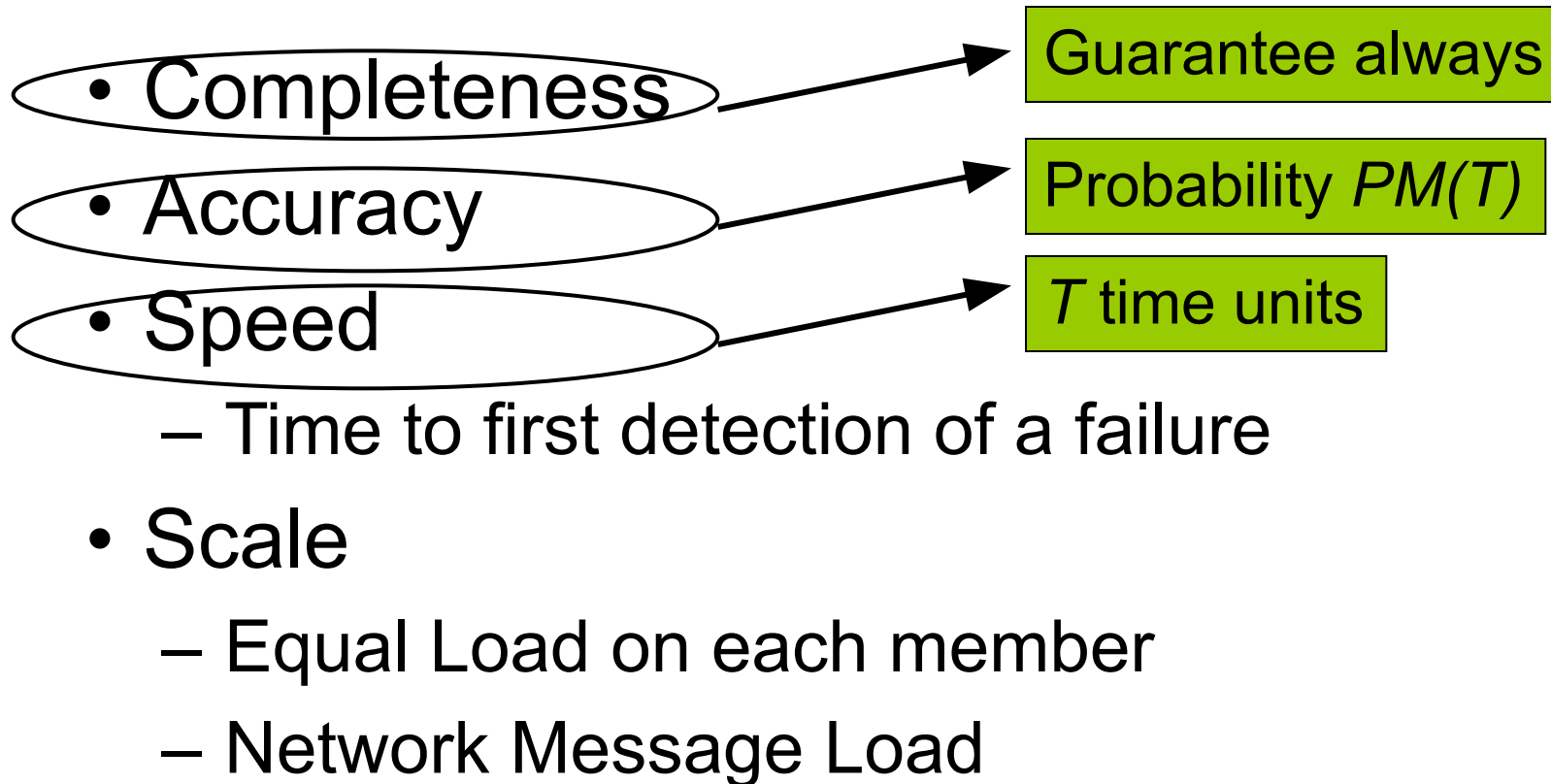


- The algorithm is resilient to message loss

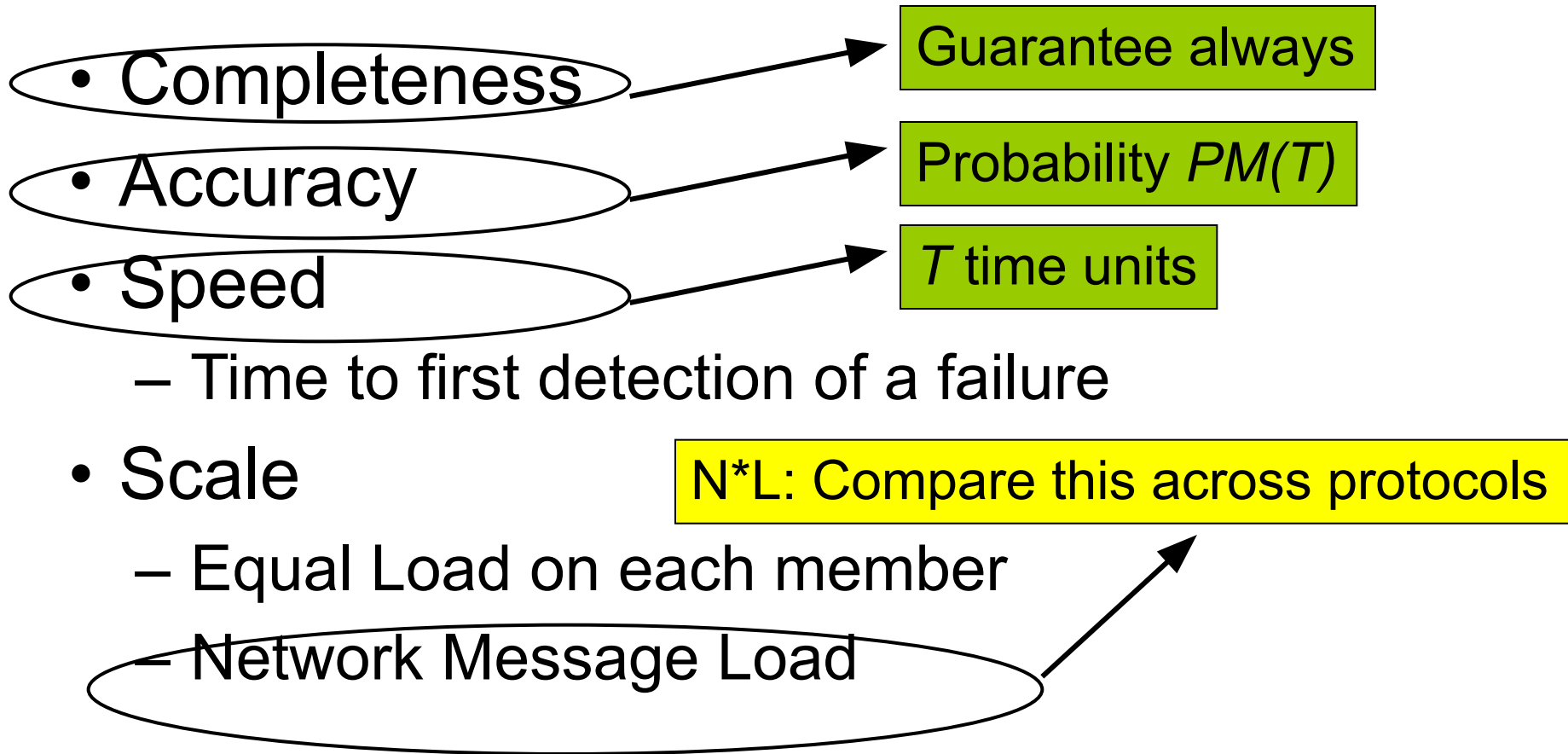
Failure Detector Properties ...

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

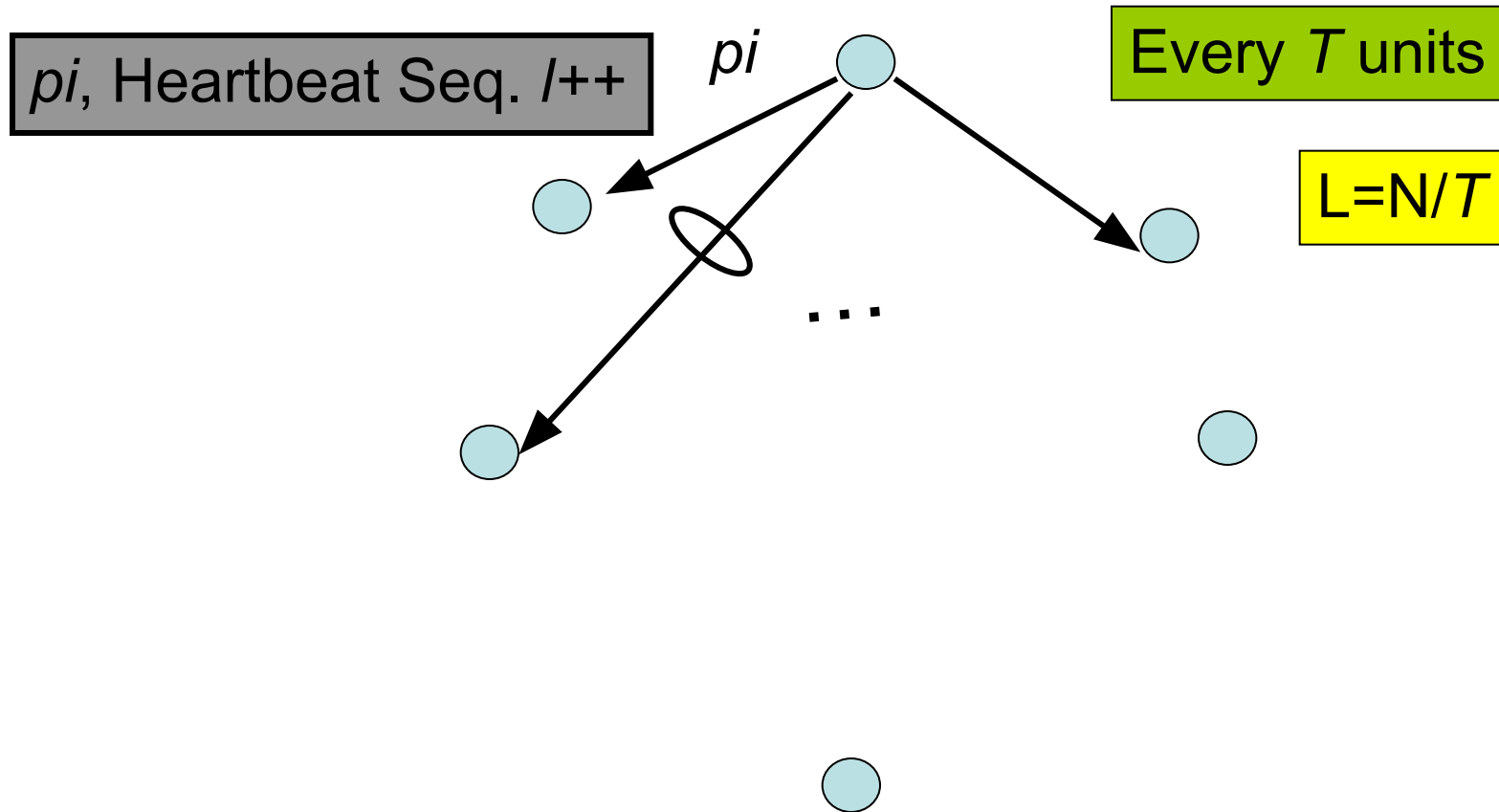
...Are application-defined Requirements



...Are application-defined Requirements



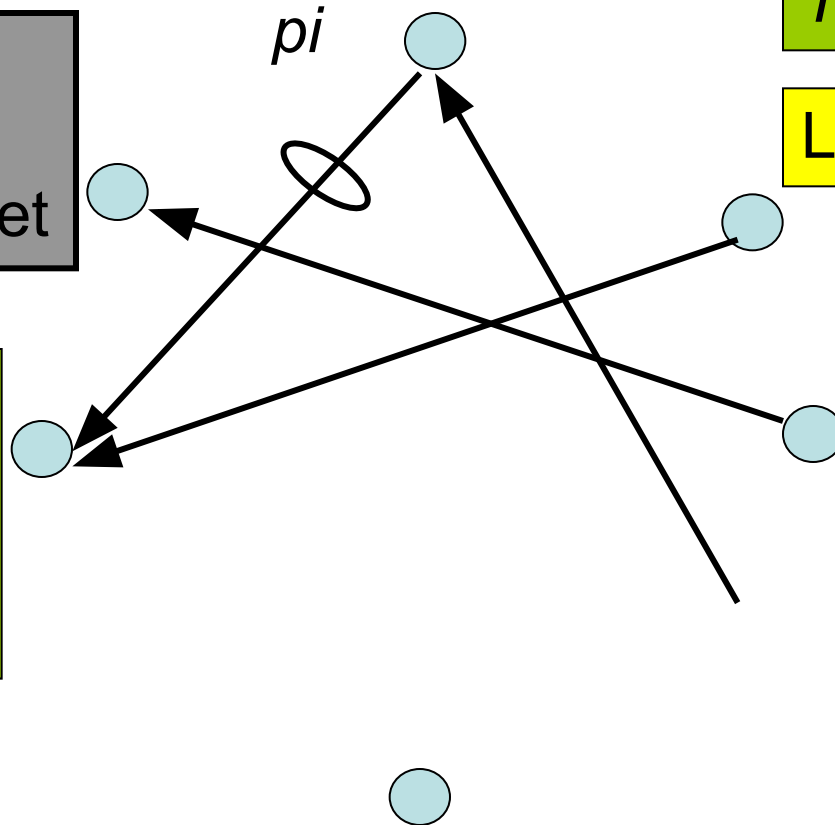
All-to-All Heartbeating



Gossip-style Heartbeating

Array of
Heartbeat Seq. l
for member subset

Every t_g units
=gossip period,
send $O(N)$ gossip
message



$$T = \log N * t_g$$

$$L = N/t_g = N * \log N / T$$

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)

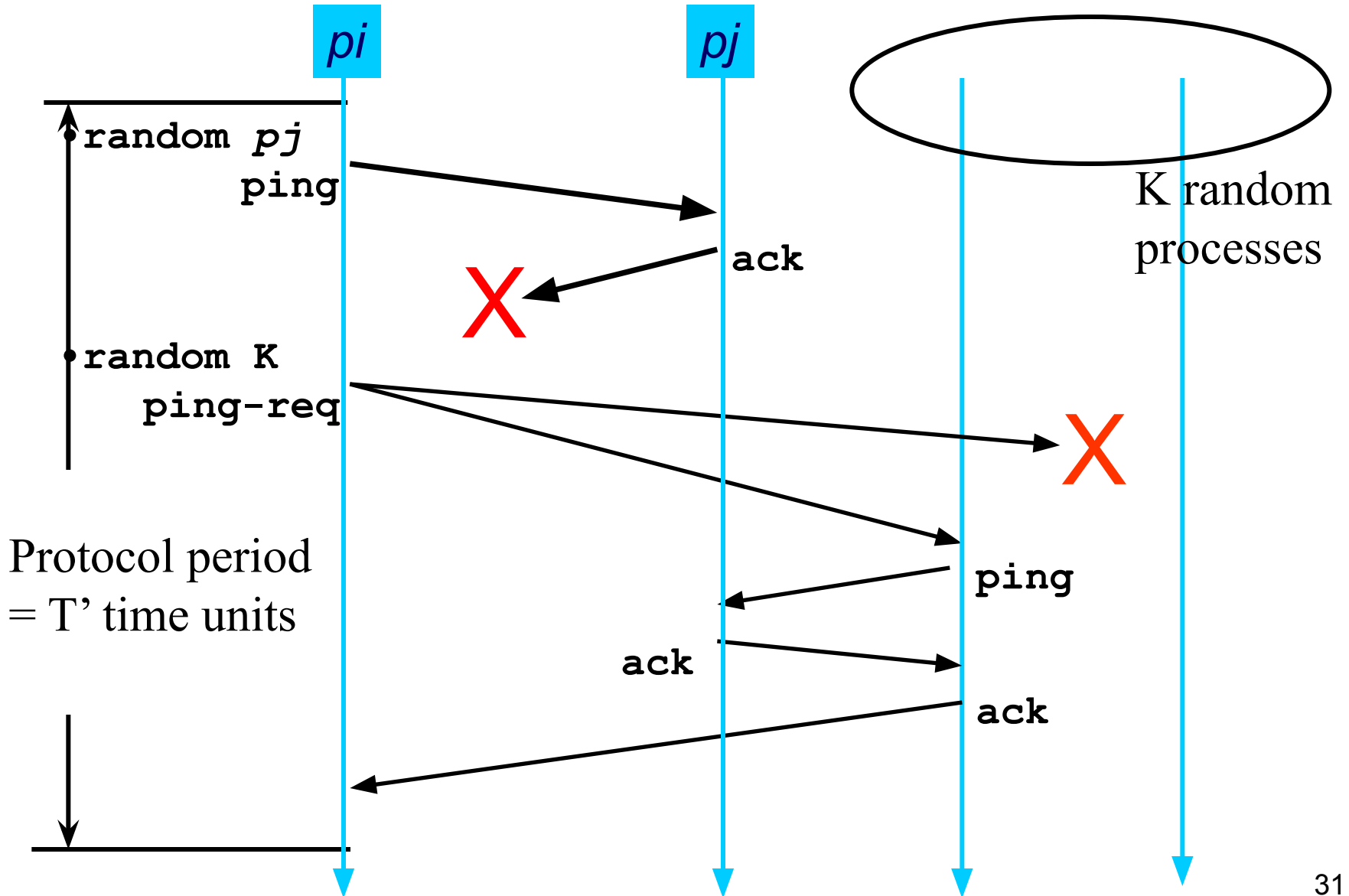
Heartbeating

- 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

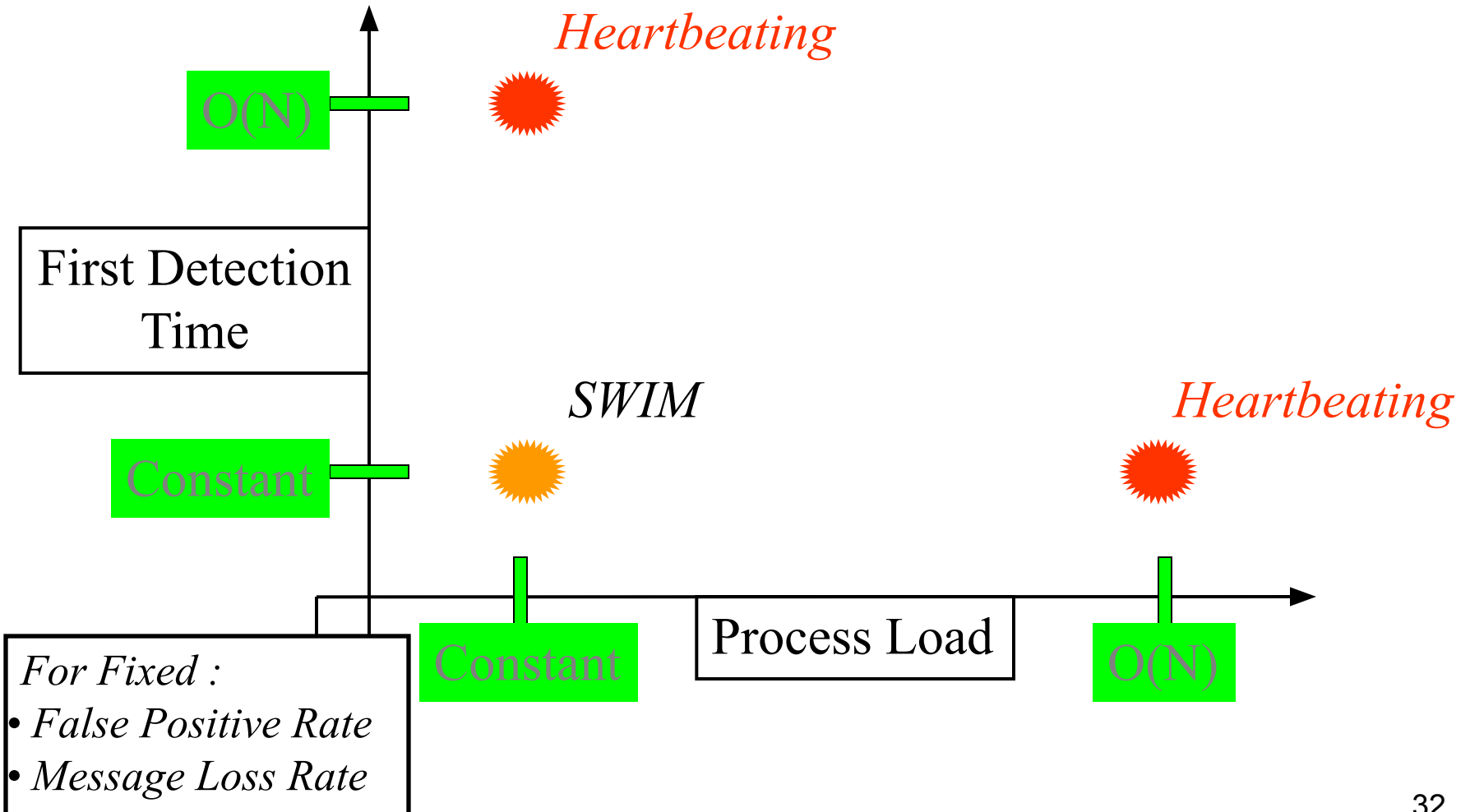
Key:

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

SWIM Failure Detector Protocol



SWIM versus Heartbeating



SWIM Failure Detector

$$1 - \left(1 - \frac{1}{N}\right)^{N-1} = 1 - e^{-1}$$

Parameter	SWIM
First Detection Time	<ul style="list-style-type: none"> • Expected $\left[\frac{e}{e-1} \right]$ periods • Constant (independent of group size)
Process Load	<ul style="list-style-type: none"> • Constant per period • $< 8 L^*$ for 15% loss
False Positive Rate	<ul style="list-style-type: none"> • Tunable (via K) • Falls exponentially as load is scaled
Completeness	<ul style="list-style-type: none"> • Deterministic time-bounded • Within $O(\log(N))$ periods w.h.p.

Accuracy, Load

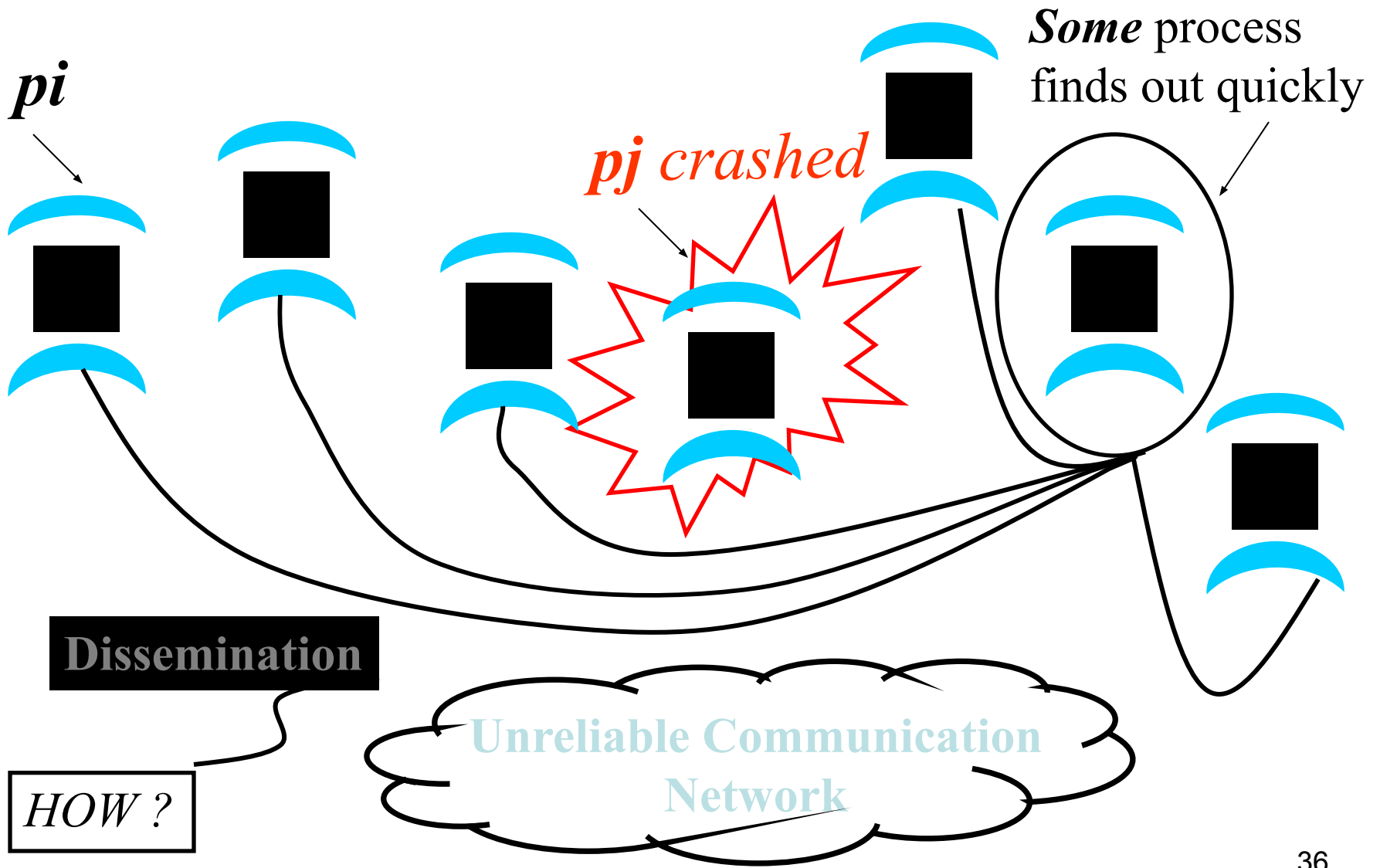
- $PM(T)$ is exponential in $-K$. Also depends on pml (and pf)
 - See paper

- $\frac{L}{L^*} < 28$ $\frac{E[L]}{L^*} < 8$ for up to 15 % loss rates

Detection Time

- Prob. of being pinged in $T' = 1 - \left(1 - \frac{1}{N}\right)^{N-1} = 1 - e^{-1}$
- $E[T] = T' \cdot \frac{e}{e-1}$
- Completeness: *Any* alive member detects failure
 - Eventually
 - By using a trick: within worst case $O(N)$ protocol periods

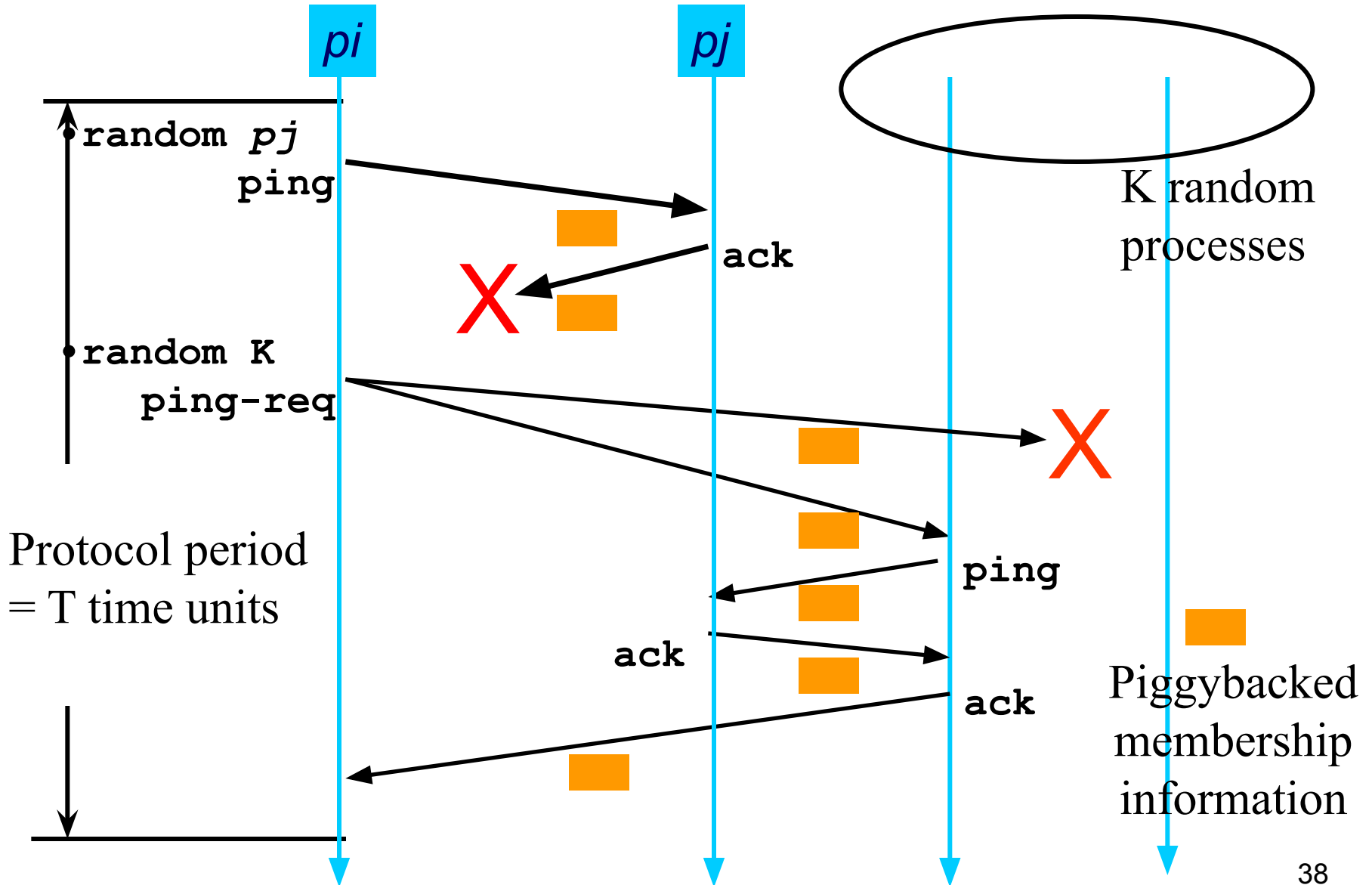
III. Dissemination



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

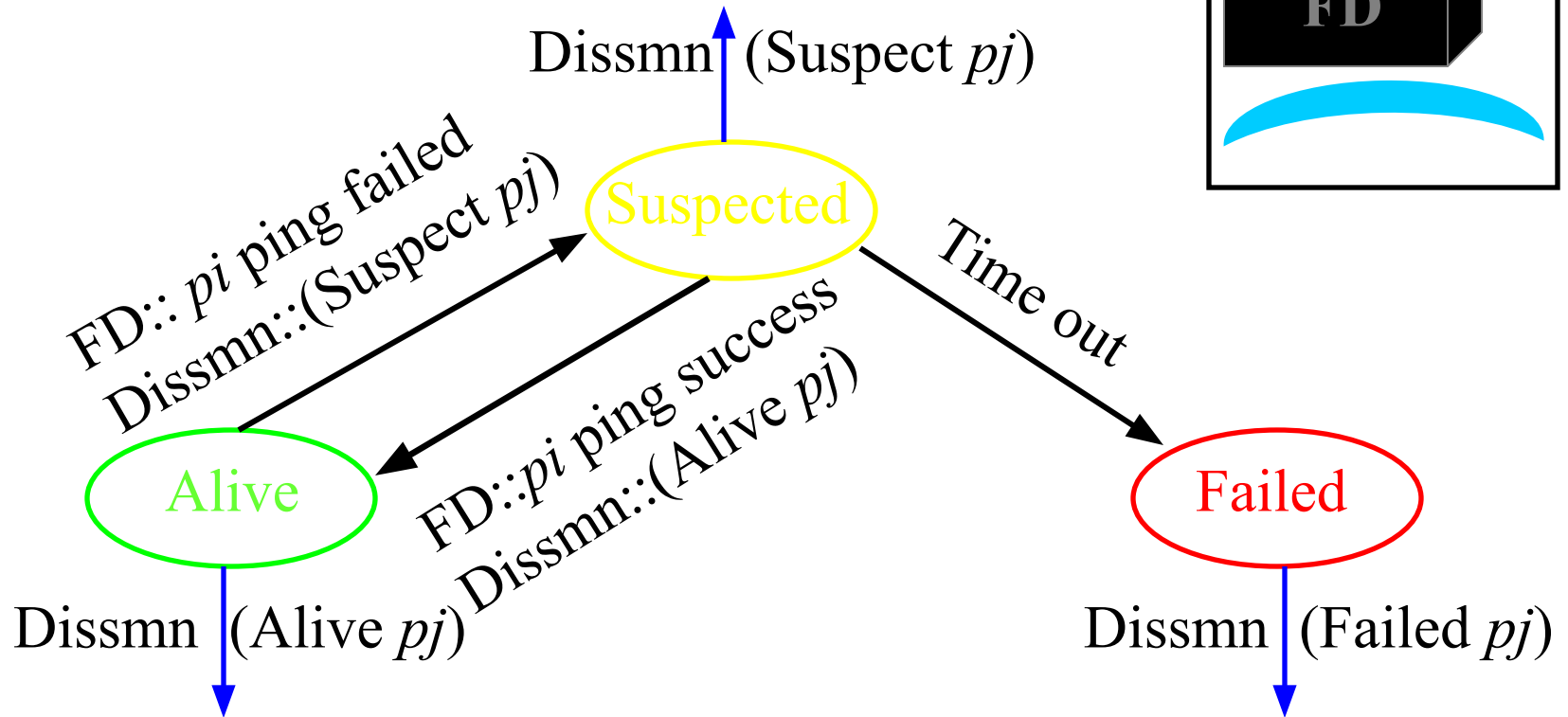
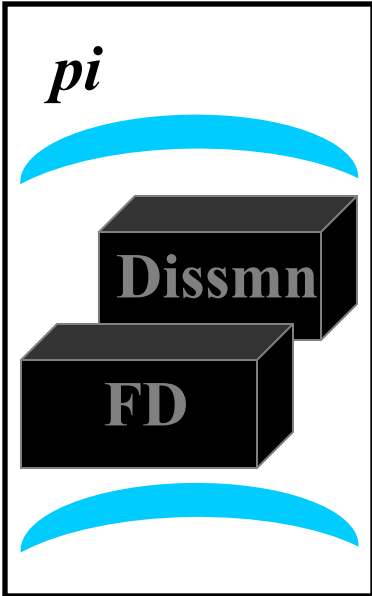
- Epidemic style dissemination
 - After $\lambda \cdot \log(N)$ protocol periods, $N^{(2^{\lambda}-2)}$ processes would not have heard about an update
- Maintain a buffer of recently joined/evicted processes
 - Piggyback from this buffer
 - Prefer recent updates
- Buffer elements are garbage collected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

$pi ::$ State Machine for pj view element



Suspicion Mechanism

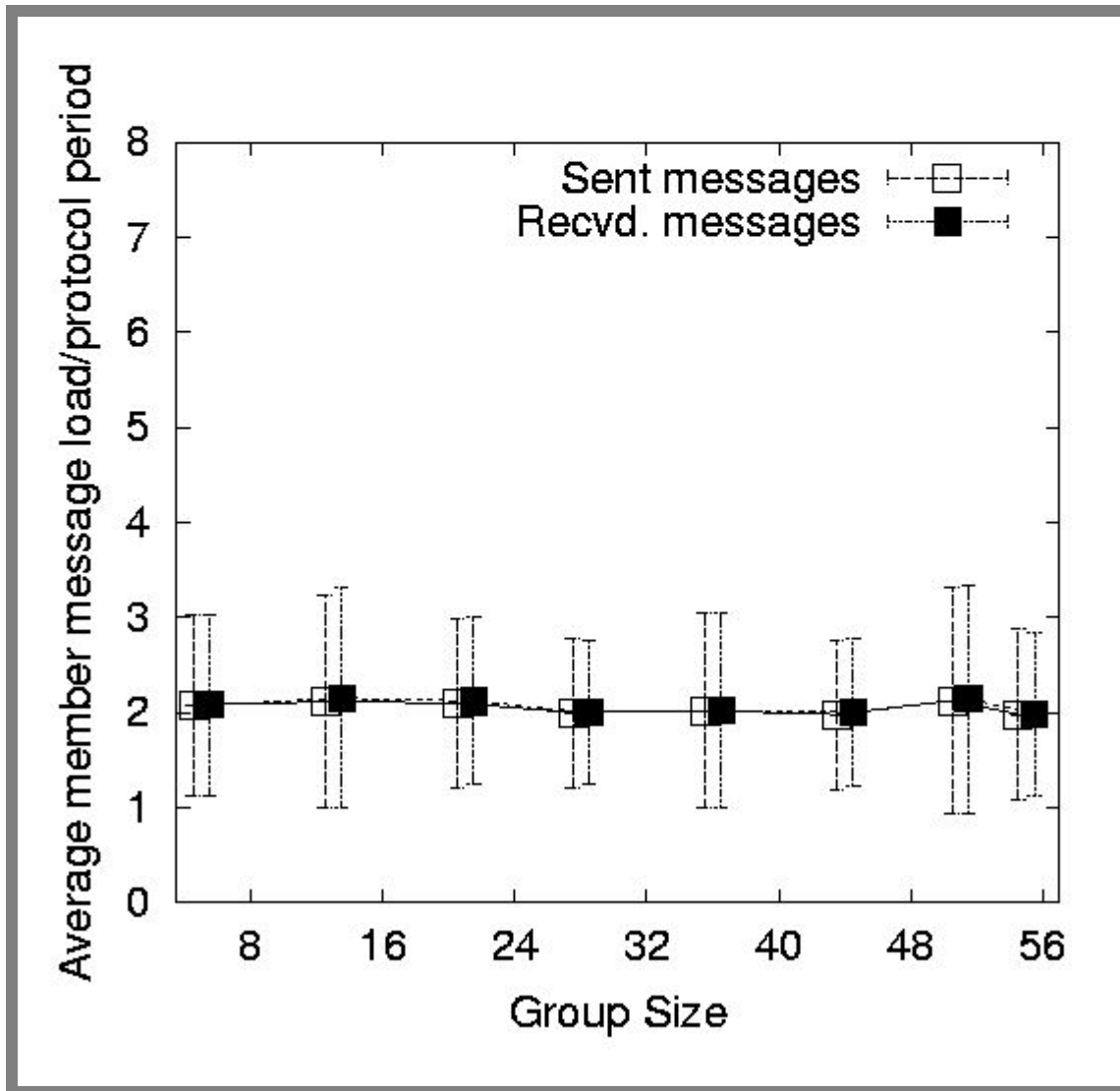
- Distinguish multiple suspicions of a process
 - Per-process *incarnation number*
 - *Inc #* for pi can be incremented only by pi
 - e.g., when it receives a (Suspect, pi) message
 - Somewhat similar to DSDV
- Higher *inc#* notifications over-ride lower *inc#*'s
- Within an *inc#*: (Suspect *inc #*) > (Alive, *inc #*)
- Nothing overrides a (Failed, *inc #*)
 - See paper

Time-bounded Completeness

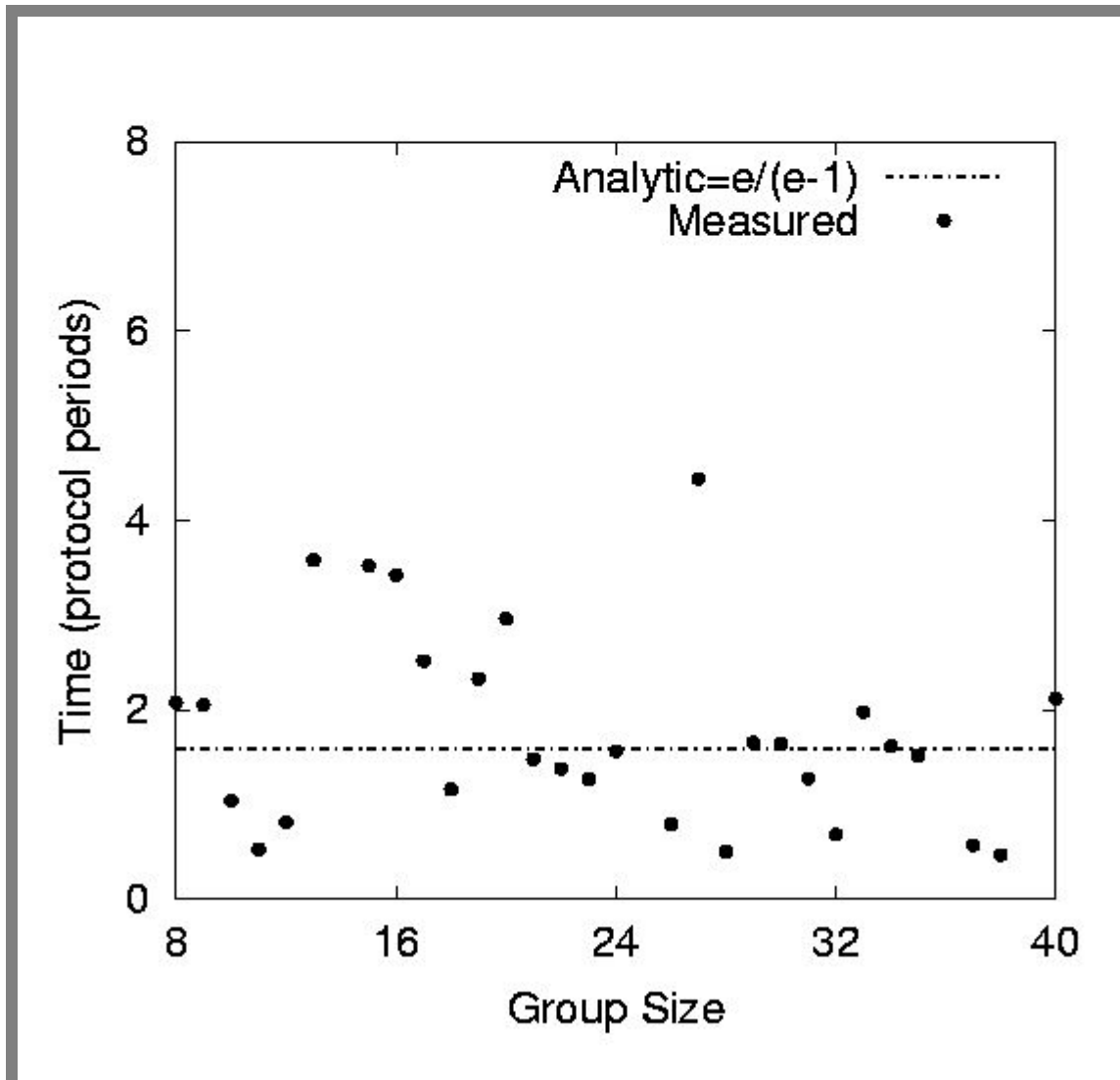
- Key: select each membership element once as a ping target in a traversal
 - Round-robin pinging
 - Random permutation of list after each traversal
- Each failure is detected in worst case $2N-1$ (local) protocol periods
- Preserves FD properties

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= $3 * \text{ceil}[\log(N+1)]$
- No partial membership lists observed in experiments



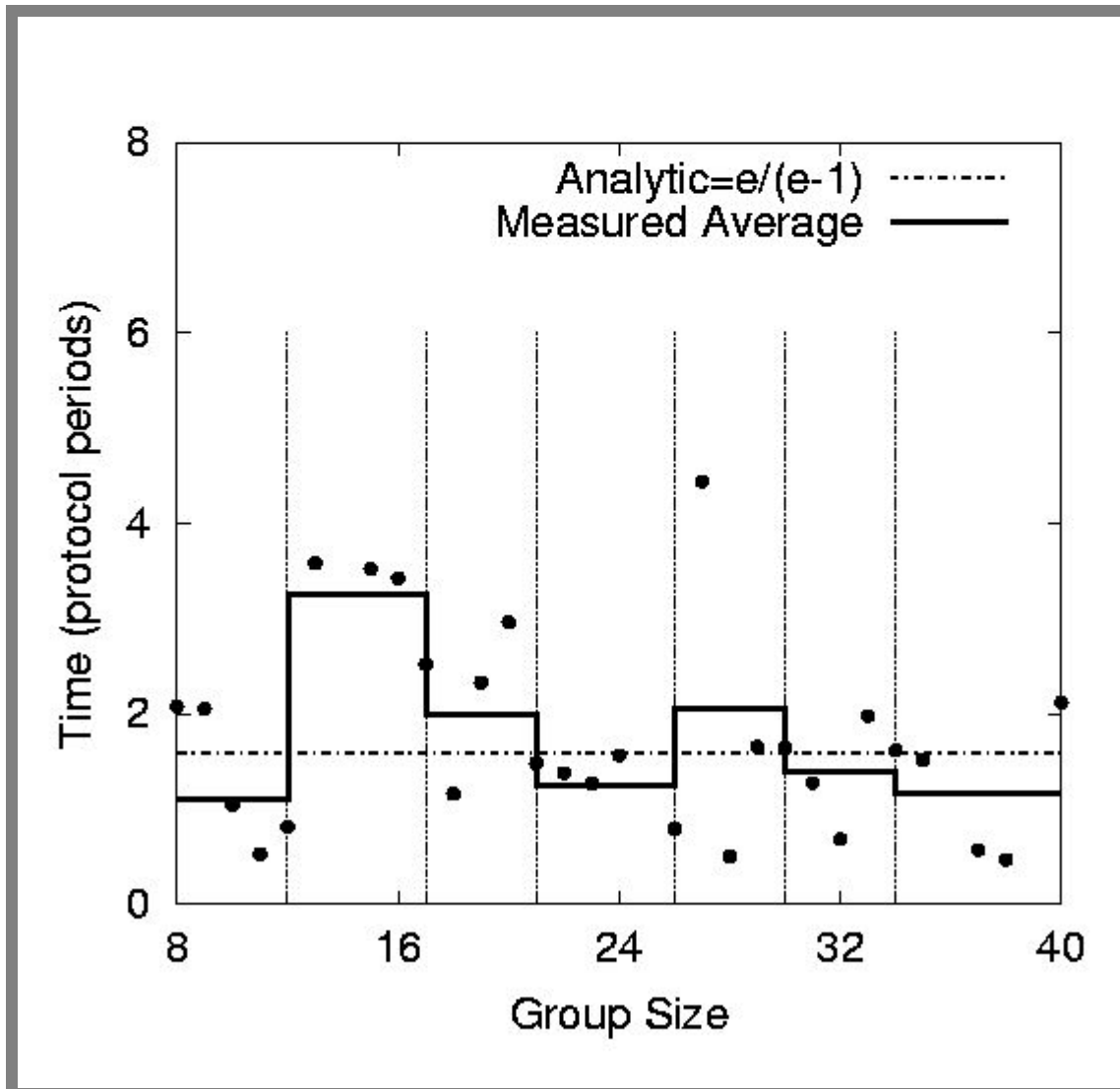
Per-process Send and Receive Loads are independent of group size



T1

T1+T2+T3

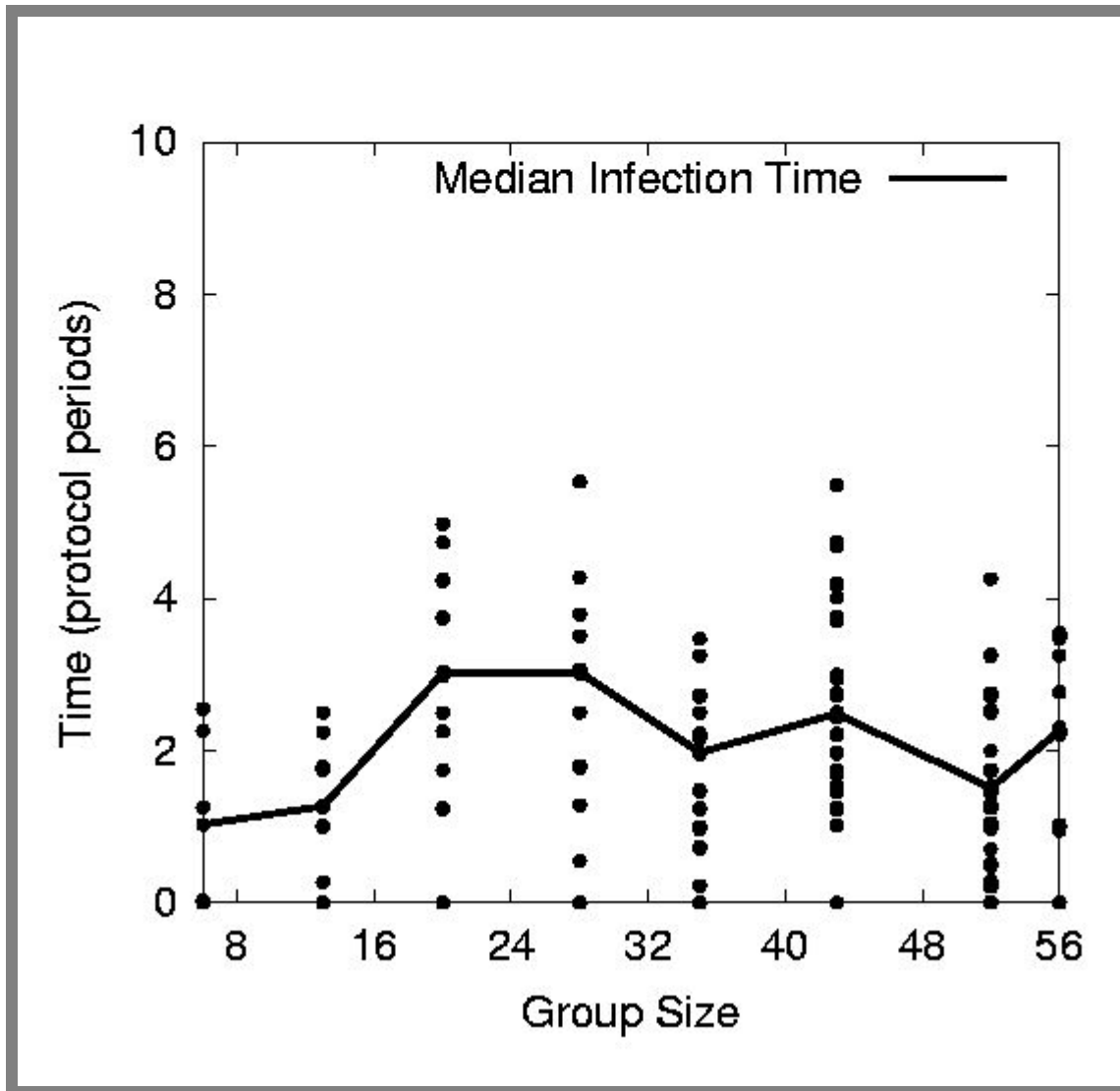
Time to First Detection of a process failure



T1

T1+T2+T3

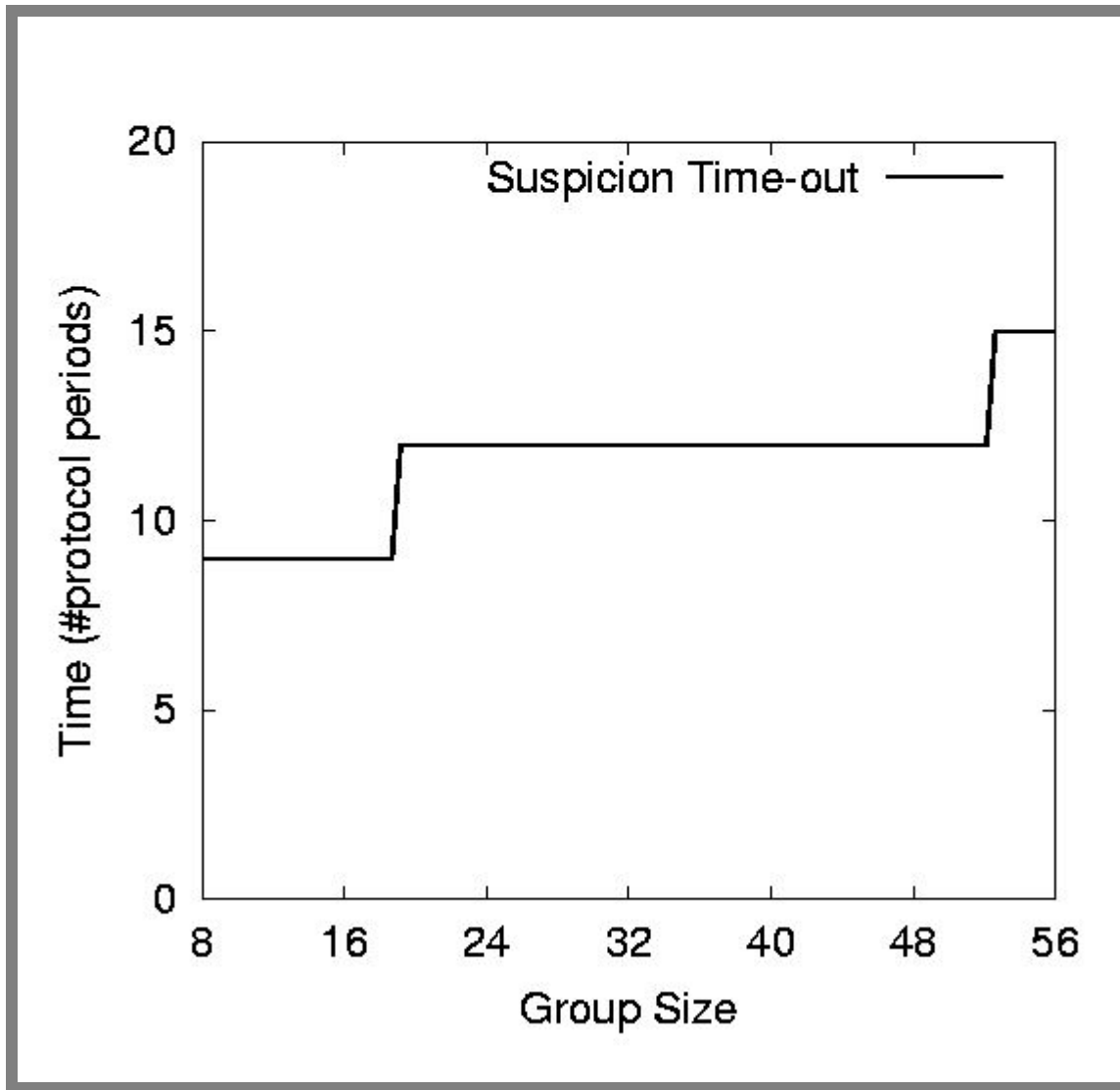
Time to First Detection of a process failure
apparently uncorrelated to group size



+
T2

T1+T2+T3

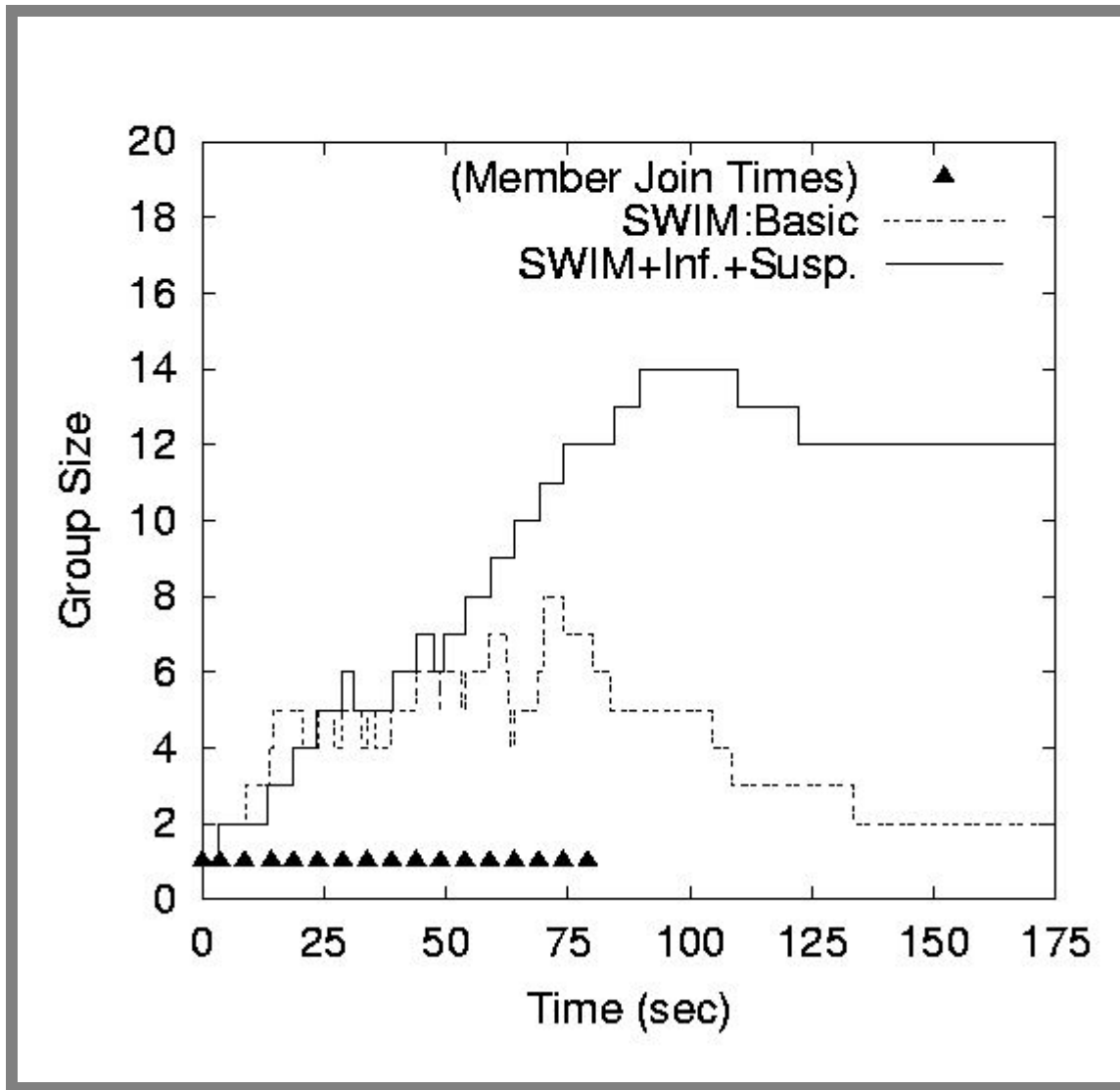
Membership Update Dissemination Time
is low at high group sizes



+
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 membership list that is *uniformly random*, gossiping retains same properties as with complete membership lists
 - Why? (Think of the equation)
 - 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

Reminder – Due this Sunday April 3rd at 11.59 PM

- **Project Midterm Report due, 11.59 pm [12pt font, single-sided, 8 + 1 page Business Plan max]**
- **Wiki Term Paper - Second Draft Due (Individual)**
- **Reviews – you only have to submit reviews for 15 sessions (any 15 sessions) from 2/10 to 4/28. Keep track of your count! Take a breather!**

Questions