OSU Oregon State UNIVERSITY

COLLEGE OF ENGINEERING

# Failure Detection and Membership I

ECE 599 / CS 519 – SPRING 2015

# Why Failure Detectors?

- Things Fail!!

- What happens if we don't detect failures in distributed systems?
  - slow or no progress
  - incorrect operation
  - …

# Failures are the Norm

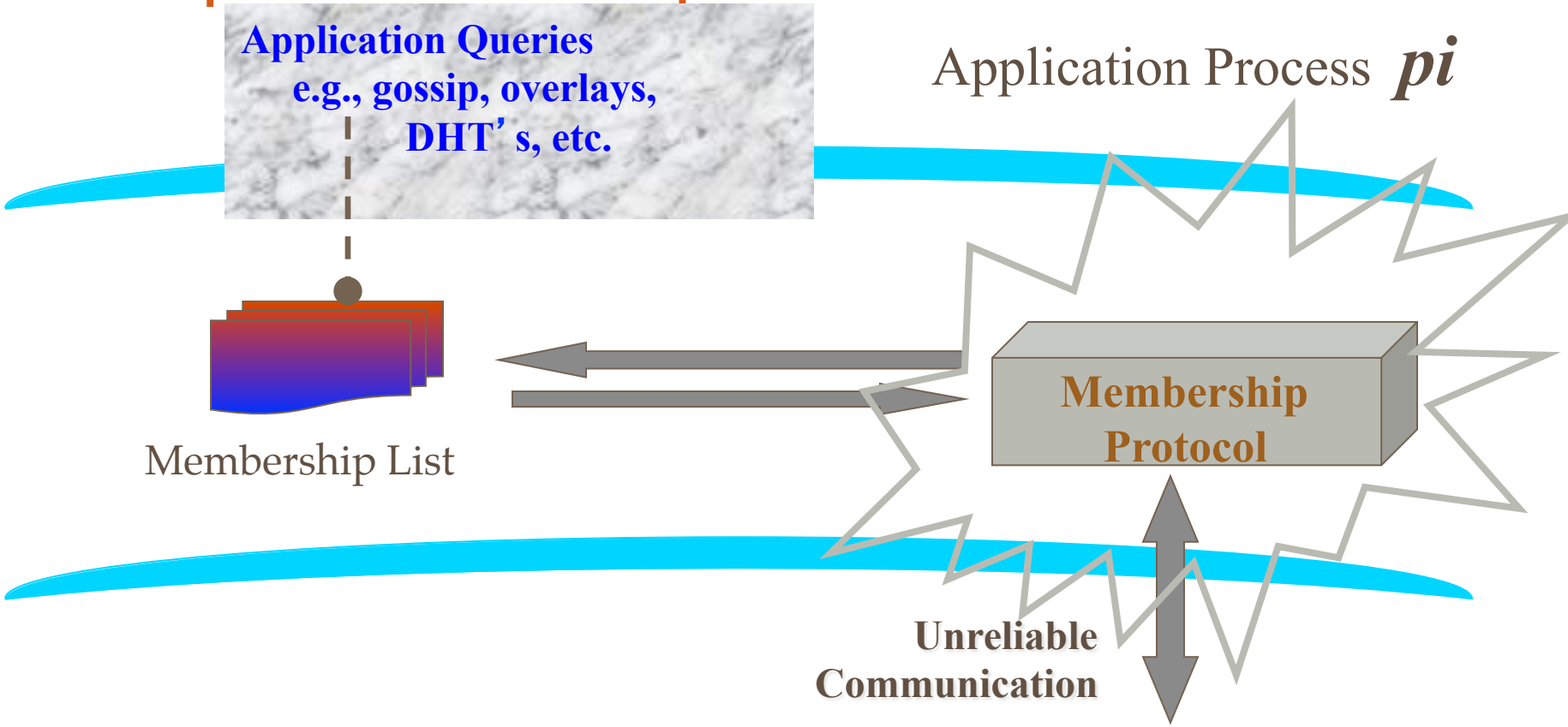
- … not the exception, in datacenters.
- Say, the rate of failure of one machine (OS/disk/motherboard/network, etc.) is once every 10 years (120 months) on average.
- When you have 120 servers in the DC, the mean time to failure (MTTF) of the next machine is 1 month.
- When you have 12,000 servers in the DC, the MTTF is about once every 7.2 hours!
- Soft crashes and failures are even more frequent!

# 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***

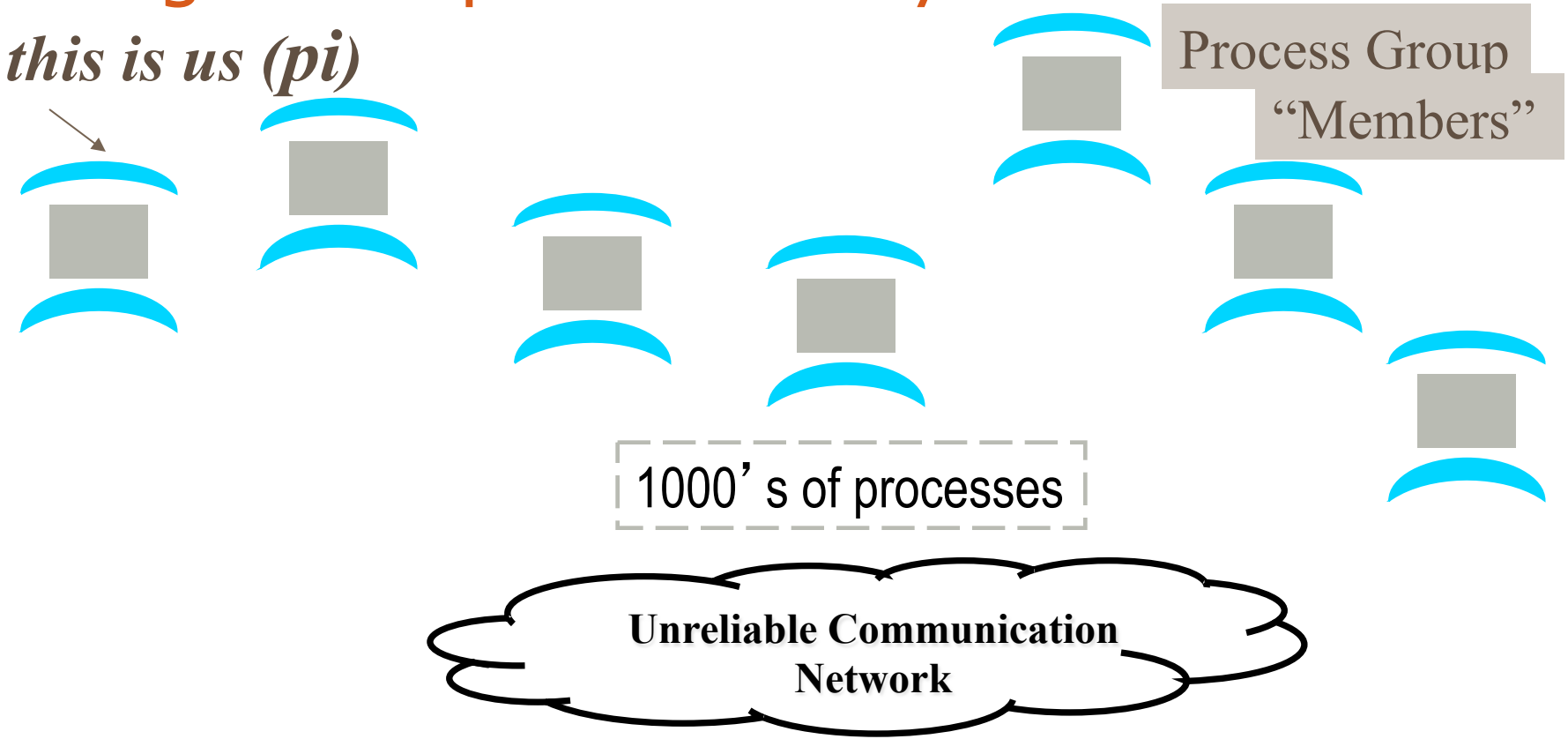
*Group Membership List*


- ***Complete list all the time*** **(Strongly consistent)**
  - Virtual synchrony
- ***Almost-Complete*** **list (Weakly consistent)**
  - Gossip-style, SWIM, …
- **Or *Partial-random* list (other systems)**
  - SCAMP, T-MAN, Cyclon,…

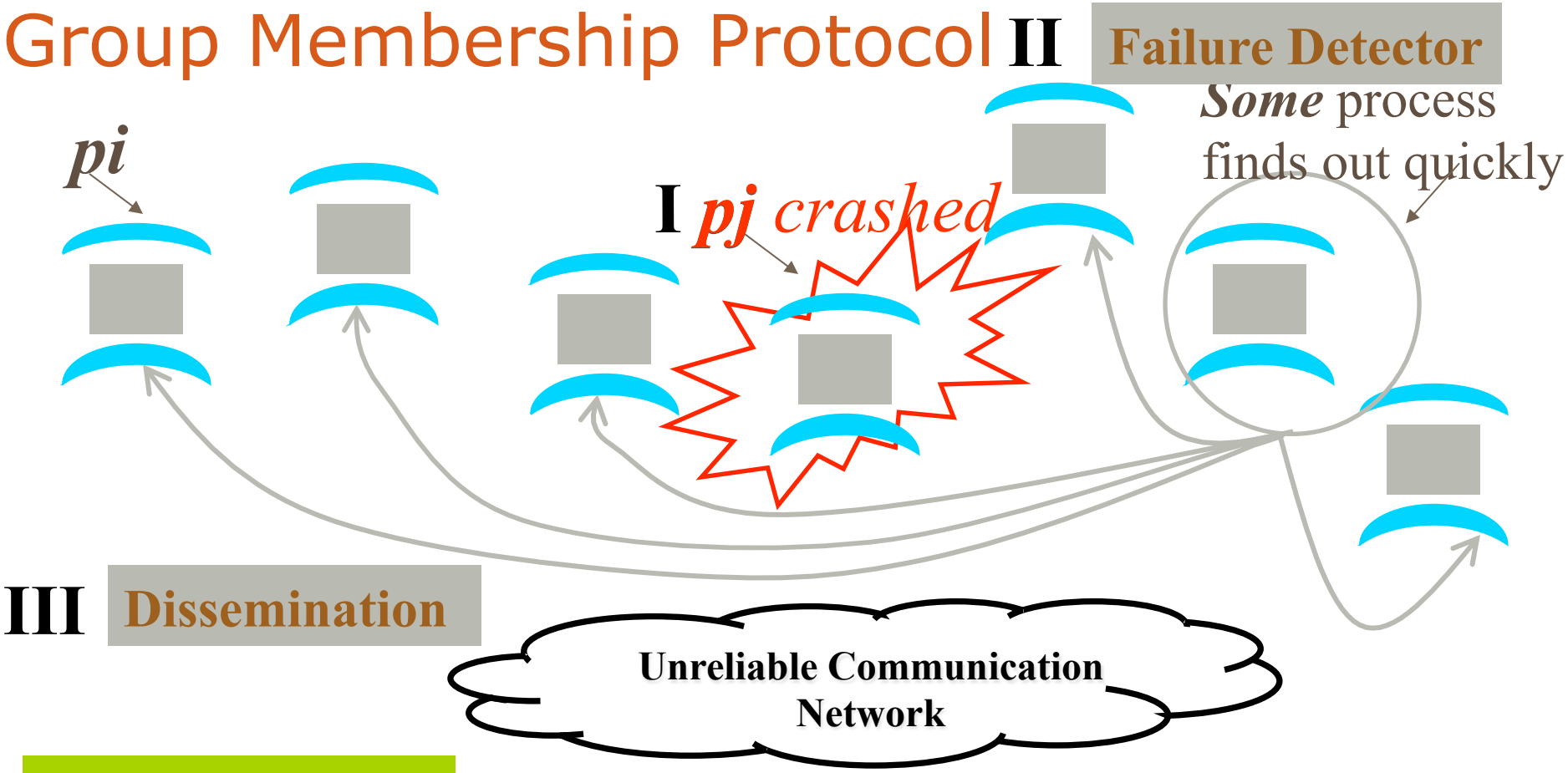
**Focus of this series of lecture**


**Unreliable Communication**

# Large Group: Scalability A Goal



# Group Membership Protocol




Crash-stop Failures only

# Next

- How do you design a group membership protocol?

# I. *pj* crashes

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

If possible, then can solve consensus!

# What Real Failure Detectors Prefer

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

Guaranteed

Partial/Probabilistic guarantee

# What Real Failure Detectors Prefer

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

Guaranteed

Partial/Probabilistic guarantee

Time until ***some*** process detects the failure

# What Real Failure Detectors Prefer

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

Guaranteed

Partial/Probabilistic guarantee

Time until ***some*** process detects the failure

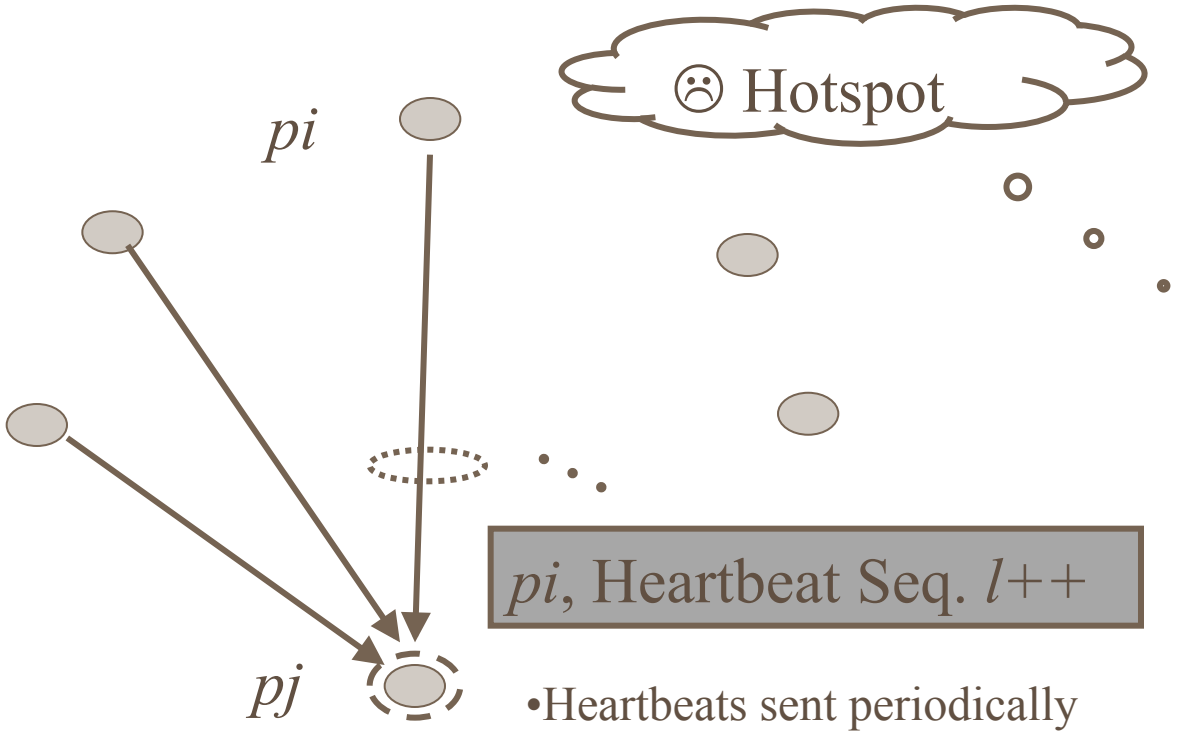
No bottlenecks/single failure point

# Failure Detector Properties

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

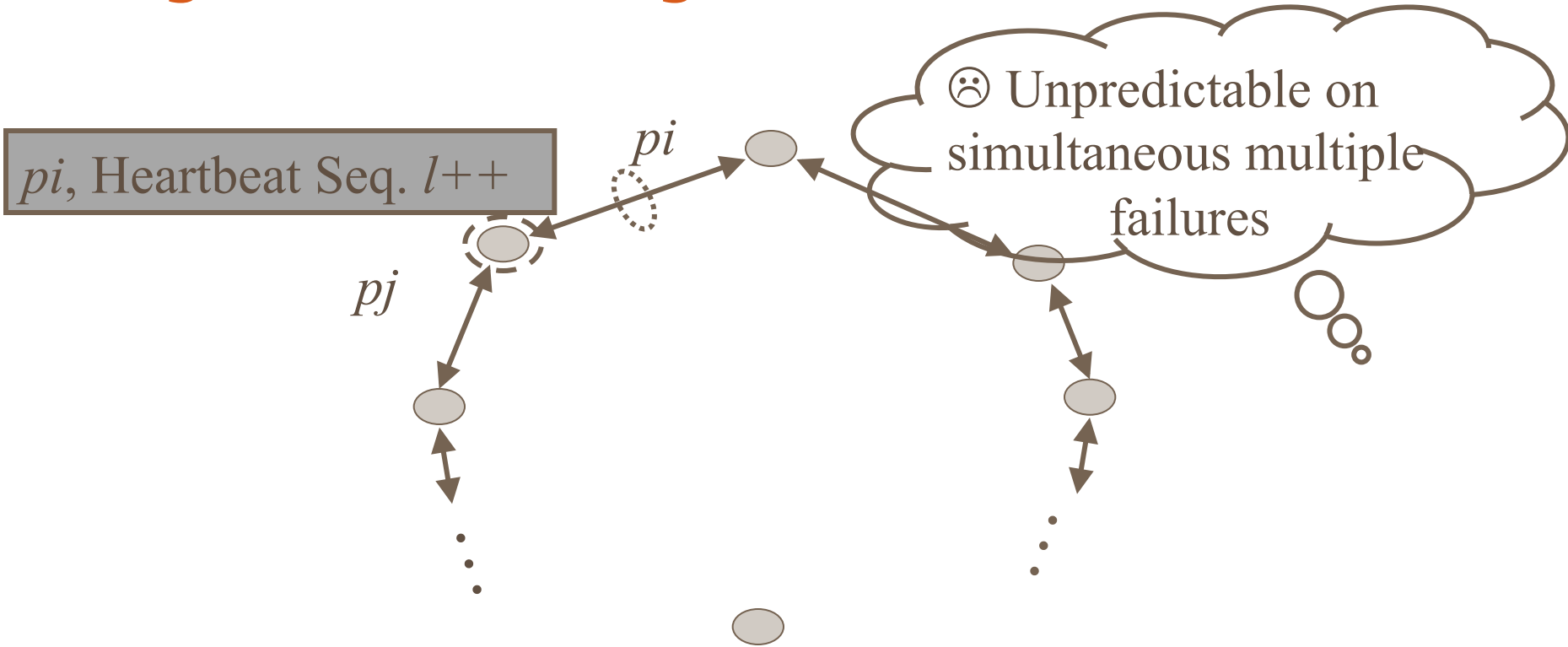
In spite of arbitrary simultaneous process failures

# Centralized Heartbeating

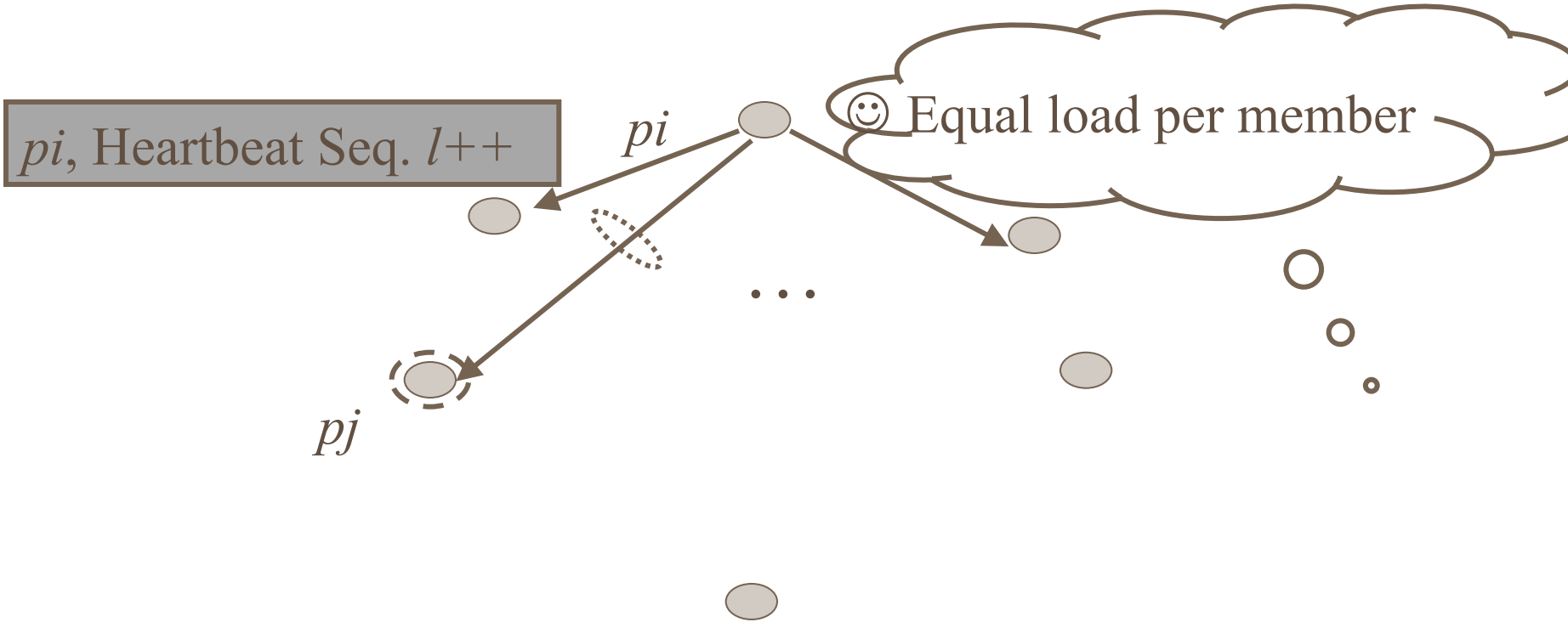



- Heartbeats sent periodically
- If heartbeat not received from *pi* within timeout, mark *pi* as failed

# Ring Heartbeating



# All-to-All Heartbeating

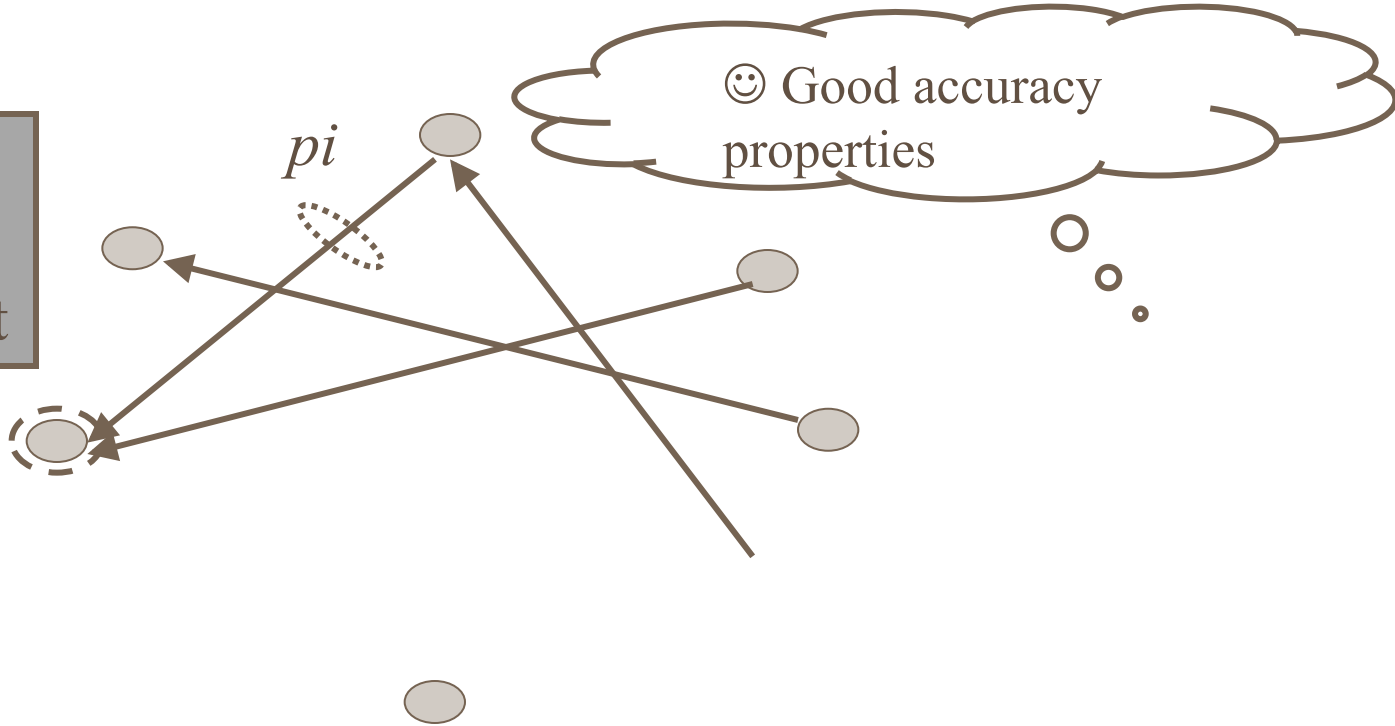


# Next

How do we increase the robustness of all-to-all heartbeating?

# Gossip-style Heartbeating

Array of Heartbeat Seq. *l* for member subset



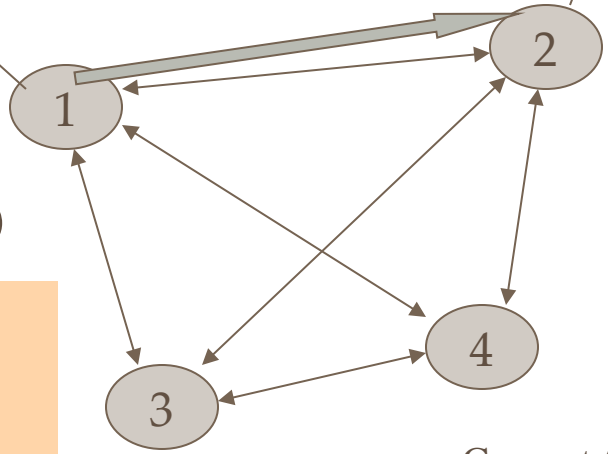
# Gossip-Style Failure Detection

| | | |
|---|---|---|
| 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 |

Current time : 70 at node 2

(asynchronous clocks)

**Protocol**:

- Nodes periodically gossip their membership list: pick random nodes, send it list
- On receipt, it is *merged* with local membership list
- When an entry times out, member is marked as failed

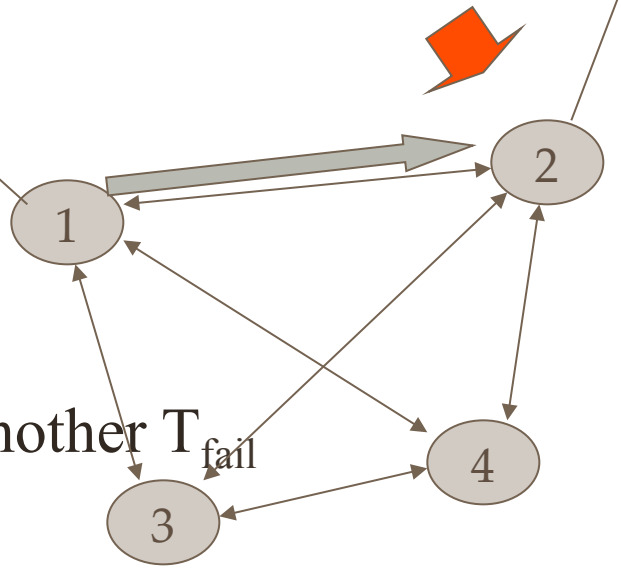
# Gossip-Style Failure Detection

- 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 $T_{fail}$ (=24) seconds?

| | | |
|---|---|---|
| 1 | 10120 | 66 |
| 2 | 10103 | 62 |
| 3 | 10098 | 55 |
| 4 | 10111 | 65 |




| | | |
|---|---|---|
| 1 | 10120 | 66 |
| 2 | 10110 | 64 |
| ~~3~~ | ~~10098~~ | ~~50~~ |
| 4 | 10111 | 65 |

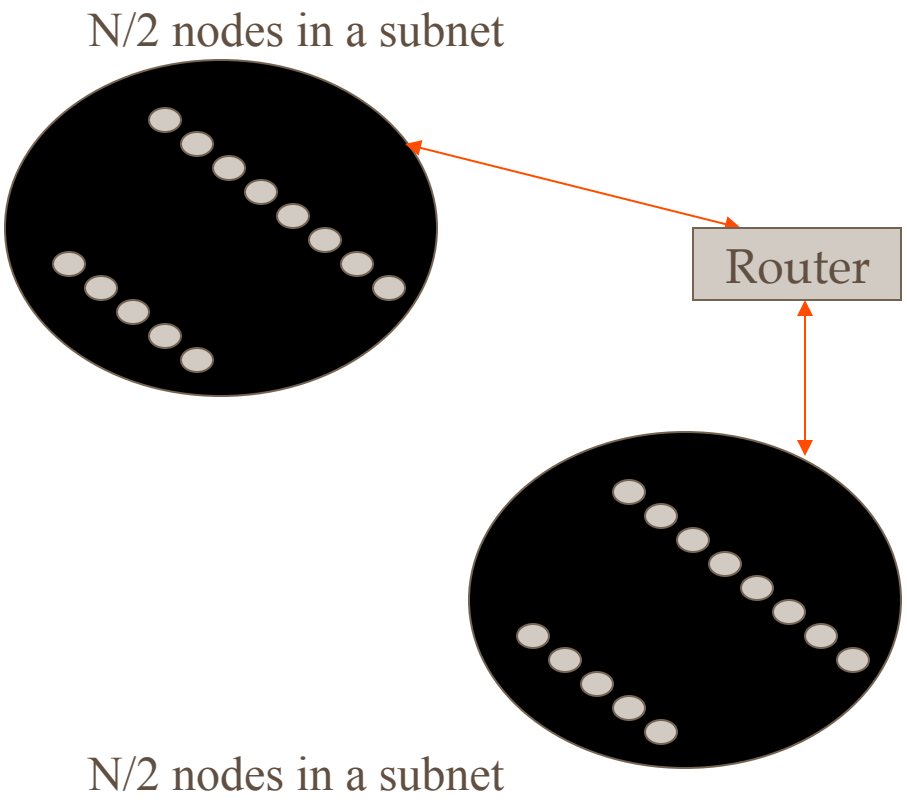
Current time : 75 at node 2

- 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: In subnet *i*, which contains $n_i$ nodes, pick gossip target in your subnet with probability $(1-1/n_i)$
- Router load=O(1)
- Dissemination time=O(log(N))
- What about latency for multi-level topologies?

[Gupta et al, TPDS 06]



# Analysis/Discussion (Not covered in class)

- 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.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 vs. bandwidth

# Next

- So, is this the best we can do? What is the best we can do?