USP - Universidade
de São Paulo

UFPE

A Group Membership Service for Large-Scale Grids*

Fernando Castor Filho[1,4], Raphael Y. Camargo[2],
Fabio Kon[3], and Augusta Marques[4]

[1]Informatics Center, Federal University of Pernambuco
[2]School of Arts, Sciences, and Humanities, University of São Paulo
[3]Department of Computer Science, University of São Paulo
[4]Department of Computing and Systems, University of Pernambuco

*Supported by CNPq/Brazil, grants #481147/2007-1 and #550895/2007-8

Faults in Grids

- Important problem
  - Waste computing and network resources
  - Waste time (resources might need to be reserved again)
- Scale worsens matters
  - Failures become common events
- Opportunistic grids
  - Shared grid infrastructure
  - Nodes leave/fail frequently

- Fault tolerance can allow for more efficient use of the grid

Achieving Fault Tolerance

- Fist step: detecting failures...
- And then doing something about them
- Other grid nodes must also be aware
  - Otherwise, progress might be hindered
- More generally: each node should have an up-to-date view of group membership
  - In terms of correct and faulty processes

Requirements for Group Membership in Grids

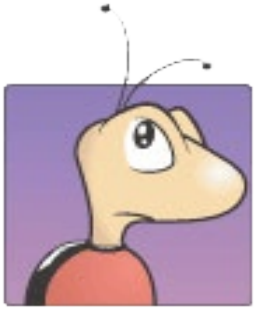
1. Scalability
2. Autonomy
3. Efficiency
4. Capacity of handling dynamism
5. Platform-independence
6. Distribution (decentralization)
7. Ease of use

Our Proposal

- A group membership service that addresses the aforementioned requirements
  - Very lightweight
  - Assuming a crash-recovery fault model
  - Deployable in any platform that has an ANSI C compiler
  - Leveraging recent advances in
    - Gossip/infection-style information dissemination
    - Accrual failure detectors

Gossip/Infection-Style Information Dissemination

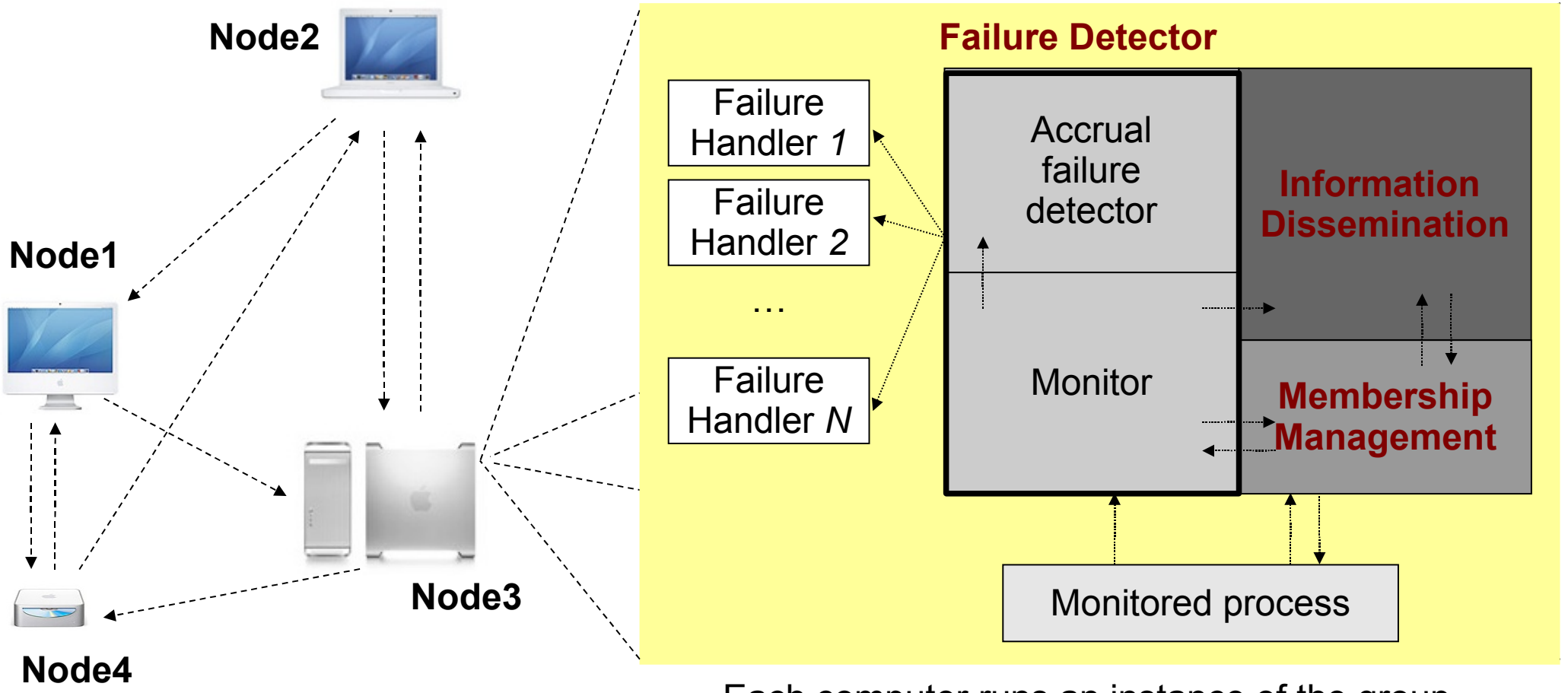
- Based on the way infectious diseases spread
  - Or, alternatively, on how gossip is disseminated
- Periodically, each participant randomly infects some of its neighbors
  - Infects = passes information that (potentially) modifies its state
- Weakly-consistent protocols
  - Sufficient for several practical applications
- Highly scalable and robust



Accrual Failure Detectors

- Decouple monitoring and interpretation
- Output values on a continuous scale
  - Suspicion level
  - Eventually strongly accurate failure detectors
- Heartbeat interarrival times define a probability distribution function
- Several thresholds can be set
  - Each triggering different actions
- As good as “regular” adaptive FDs
  - More flexible and easier to use

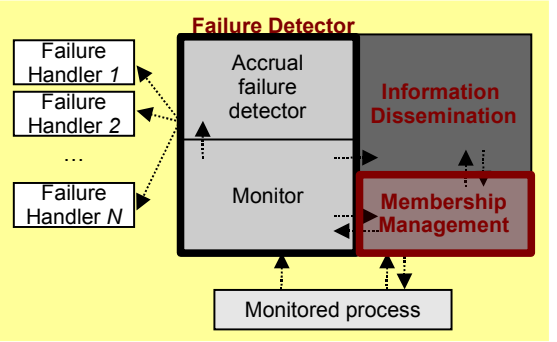
Architecture of the Group Membership Service



Each computer runs an instance of the group membership service

Membership Management

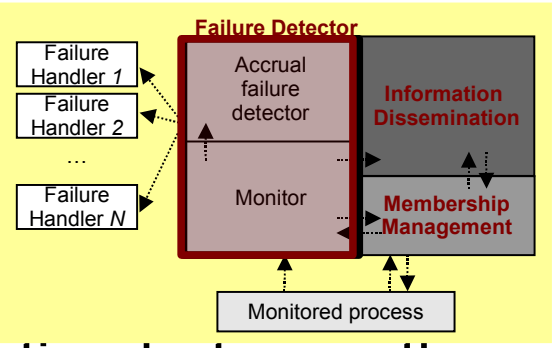
- Handles membership requests
- Disseminates information about new members



- Informs them about existing members
- Removes failed members from the group
- Failed processes can also rejoin
  - **Epoch** mechanism
  - Only 32 extra bits in each heartbeat message

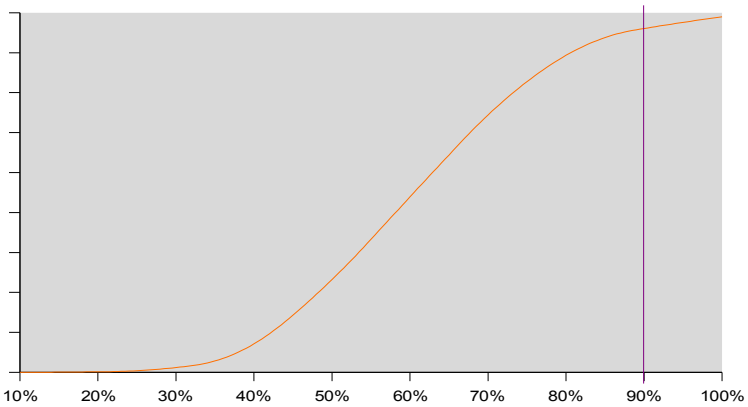
Failure Detector

- Collects data about k processes
  - *Push* heartbeats
  - Gossiped periodically (T_{hb})
  - if p_1 monitors p_2 then there is a TCP connection between them



- Accrual Failure Detector
  - Keeps track of the last m interarrival times for a given process
  - Derives a probability that a process has failed

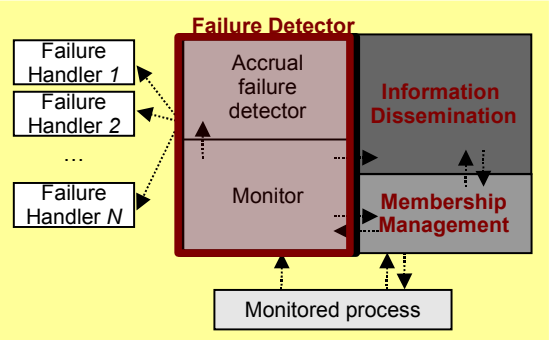
$$P_{fail,S}(t_\Delta) = \frac{|S^{t_\Delta * \alpha}|}{|S|}$$



  - Calculation is performed in $O(\log|S|)$ steps

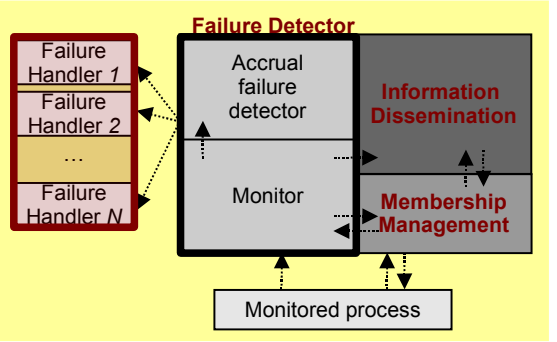
Collecting Enough Information

- Adaptive FDs need to receive information about monitored processes **regularly**
  - Also applies to accrual FDs
  - Traditional gossip protocols are not regular
- Solution: persistent monitoring relationships between processes
  - Established randomly
  - Exhibit the desired properties of gossip protocols



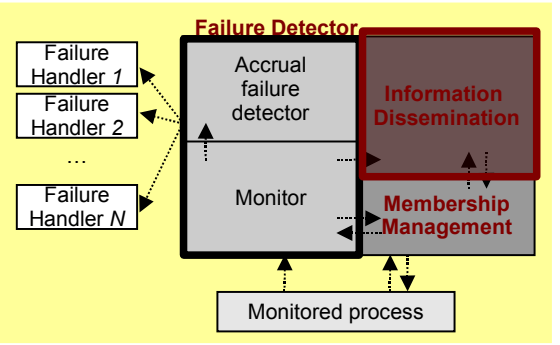
Failure Handlers

- For each monitored process, a set of thresholds is set
  - For example: 85, 90, and 95%
  - A handler is associated to each one
- Several handling strategies are possible
  - Each executed when the corresponding threshold is reached
  - It is easy to define application-specific handlers



Information Dissemination

- Responsible for gossiping information
  - About failed nodes (specific messages)
    - Important for failure handling
  - About correct members (piggybacked in heartbeat messages)
- Dissemination speed is based on parameter *j*
  - *j* should be $O(\log(N))$



Implementation

- Written in Lua
  - Compact, efficient, extensible, and platform-independent
  - The service is packaged as a reusable Lua module
- Uses a lightweight CORBA ORB (OiL) for IPC
  - Also written in Lua
- Approximately 80KB of souce code

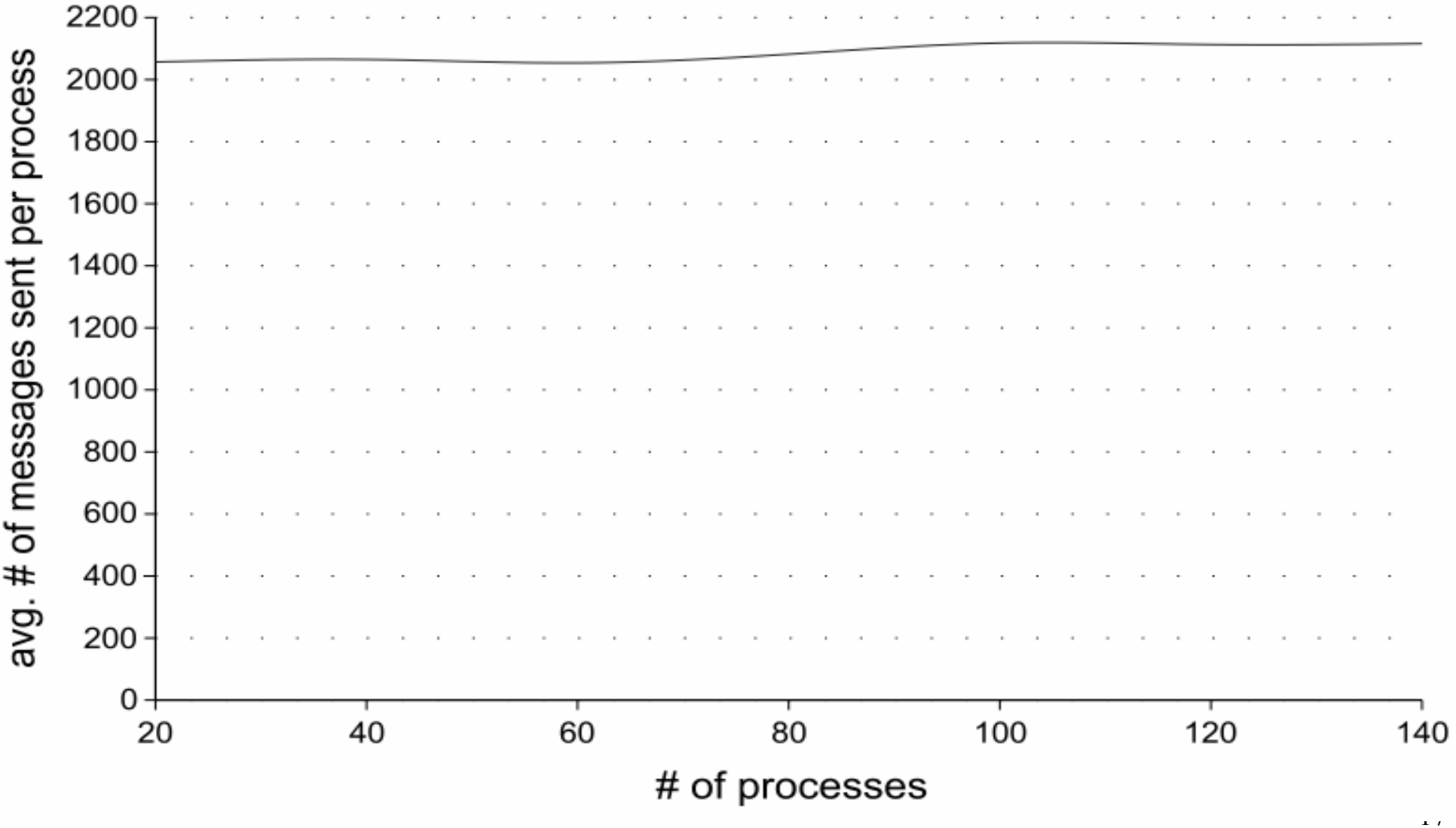
Initial Evaluation

- Main goal: to assess scalability and resilience to failures
- 20-140 concurrent nodes
  - Distributed accross three machines equipped with 1GB RAM
- 100Mbps Fast Ethernet Network
- Emulated WAN
  - *latency* = 500ms and *jitter* = 250ms
- Parameters T_{hb} = 2s, k = 4, j = 6,

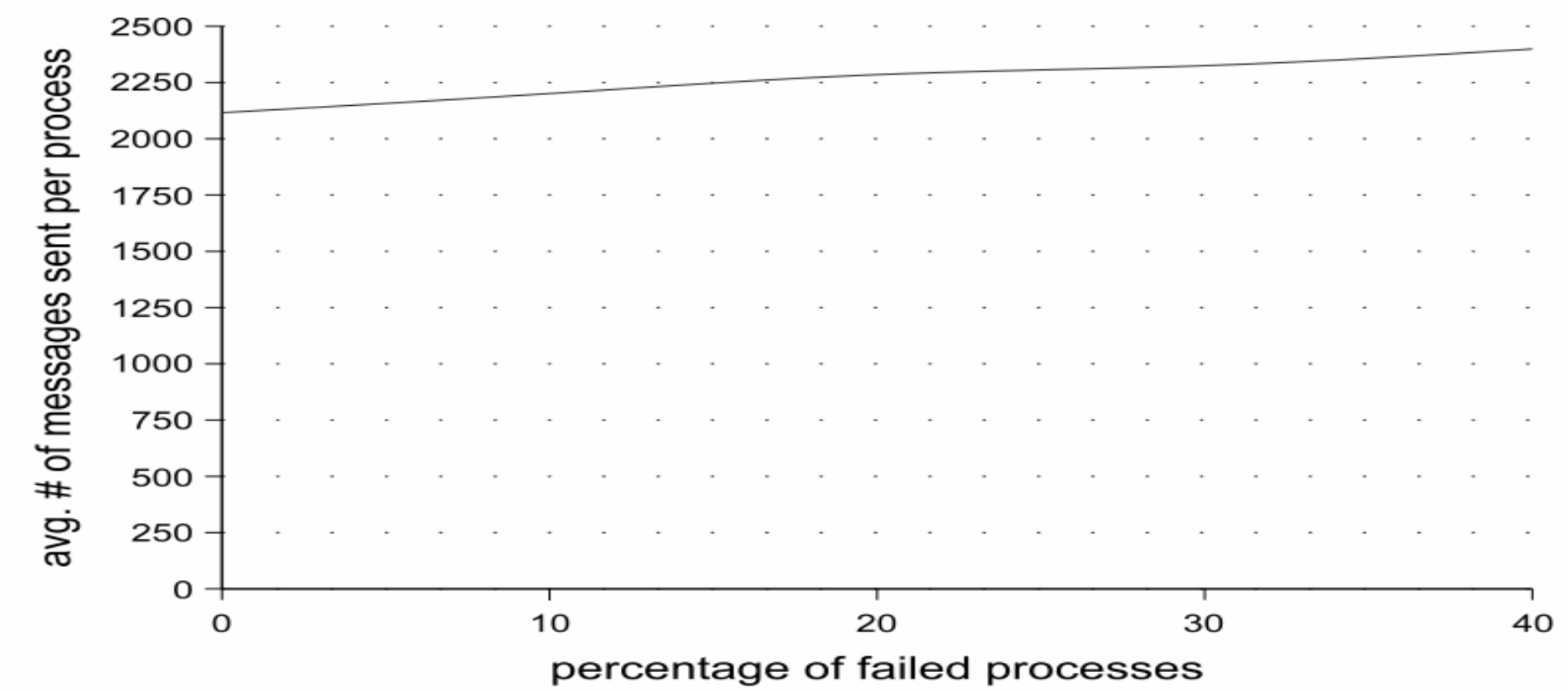
Initial Evaluation

- Two situations:
  - When no failures occur
    - 20, 40, 60, 80, 100, 120, 140 processes
  - When processes fail, including realistically large numbers of simultaneous failures
    - 140 processes
    - 10, 20, 30, and 40% of failures
- Number of sent messages per process as a measure of scalability

Scenario 1: No failures

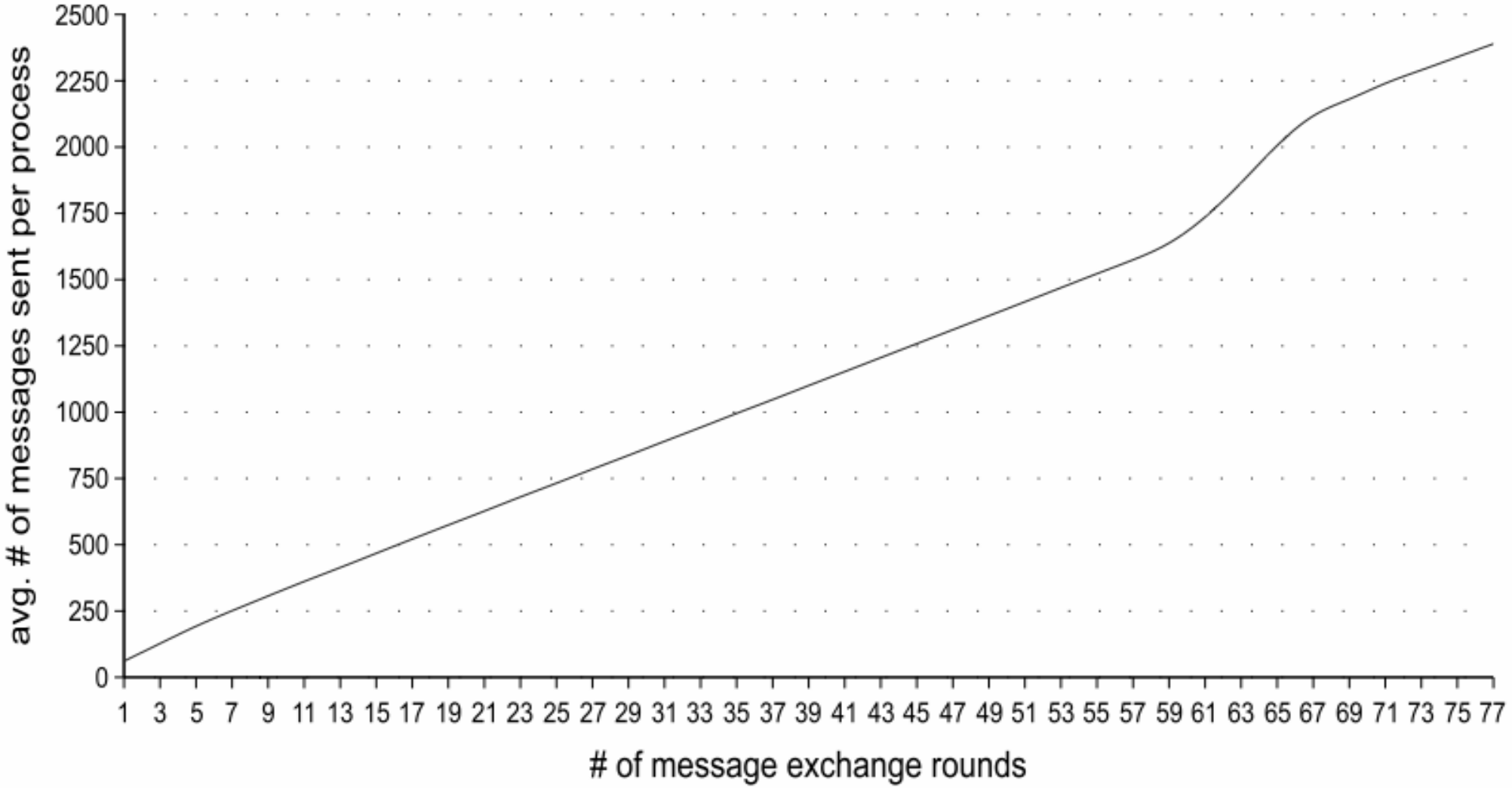


Scenario 2: 10-40% of process failures



- No process became isolated.
- Almost 95% were still monitored by at least k – 1 processes

Scenario 2: 40% of process failures



Concluding Remarks

- Main contribution: to combine gossip-based information dissemination and accrual FDs
  - while guaranteeing that the AFD collects enough information ;
  - scalably; and
  - in a timely and fault-tolerant way
- Ongoing work:
  - More experiments
  - Self-organizing for better resilience and better scalability
  - Periodic dissemination of failure information

Thank You!

Contact:

Fernando Castor

fcastor@acm.org