Distributed Resource-Allocation With Optimal Failure Locality

Paolo A. G. Sivilotti, Scott M. Pike and Nigamanth Sridhar

Computer and Information Science

The Ohio State University

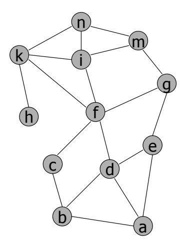
Motivation

- Process failures should have limited impact
  - Robust systems require algorithms that mask remote failures
  - One metric of impact: *failure locality*
- A new algorithm for resource allocation
  - Optimal worst-case failure locality
  - Configurable to improve expected failure locality

Dining Philosophers Problem

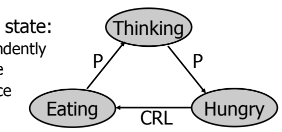
An abstraction for resource-allocation problems

- A **conflict graph** models a set of resources shared among competing processes
  - Each node represents a process
  - Each edge represents a *potential* conflict



Dining Philosophers Problem

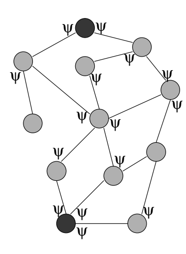
- A process is modeled by its state:
  - **Thinking**: executing independently
  - **Hungry**: requesting resource
  - **Eating**: using shared resource



- Restriction: Eating is always finite
- Conflict-resolution layer must satisfy:
  - **Safety:** no two neighbors eat simultaneously
  - **Progress:** every hungry process eats eventually

Safety

- Safety can be ensured by using **forks**
  - A fork is a token shared between two neighbors
  - Exactly one fork per edge
- A process can eat only if it holds all of its forks

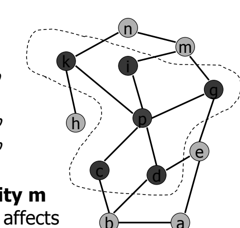


A Metric: Failure Locality

- **m-neighborhood of *p*:** the set of processes reachable along at most *m* edges from *p*

0-neighborhood of *p*

1-neighborhood of *p*



An algorithm has **Failure Locality m** if the failure of any process only affects processes within its *m-neighborhood*

Model of Computation

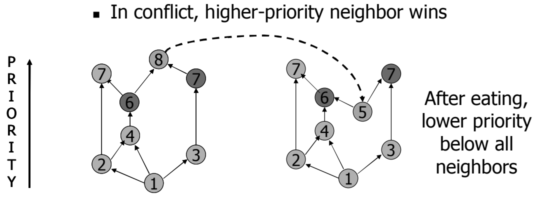
- Processes are **distributed**, communicating only by asynchronous message passing
- Channels are unordered, but messages are delivered reliably without loss, duplication, or corruption
- Process failures are **fail-stop**
  - Execution stops without warning
  - Failed processes remain stopped forever
  - Failures cannot be detected by neighbors

Algorithm Comparison

	Hygienic	Double Doorways	Bounded Doorways	Dynamic Thresholds
Safety	YES	YES	YES	YES
Progress	YES	YES	YES	YES
Failure Locality	n	4	2	2
FIFO Channels	✘	✘	✔	✘
Broadcast Messages	✘	✘	✔	✘
Interrupt Mechanism	✘	✘	✔	✘

The Hygienic Algorithm

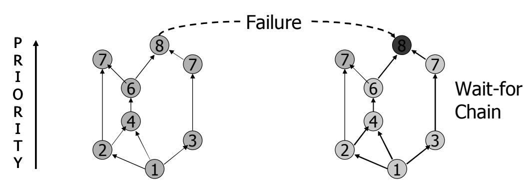
- Each process has a priority
  - Neighbors have distinct priorities
  - In conflict, higher-priority neighbor wins



Before Eating   After Eating

Hygienic Solution: Poor Failure Locality

- A hungry process **never yields** to a lower-priority neighbor, so long dependency chains may form
- Worst-case locality is linear in the number of nodes

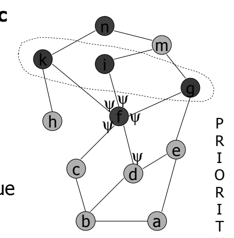


Impossibility Result

- Failure locality is ≥ 2
- Algorithms with constant failure locality:
  - Styer and Petterson, PODC 1988
  - Choy and Singh, TPDS 7(7), 1996
- To improve the failure locality of the Hygienic algorithm, we need a mechanism for breaking long dependency chains
- We borrow the notion of **thresholds** from Choi and Singh to allow lower-priority hungry neighbors to overtake higher-priority neighbors in some cases

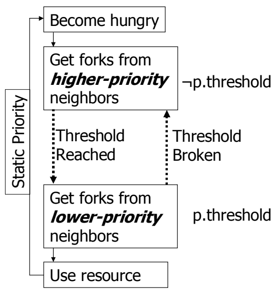
Thresholds: Improving Failure Locality

- Process priorities are **static**
- The **threshold set** of a process is the set of its higher-priority neighbors
- **p.threshold** ≡ *p* holds the fork from every process in its threshold set
- *p*.threshold is vacuously true if *p* has no higher-priority neighbors



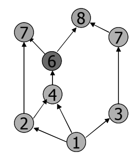
A Fork-Collection Scheme

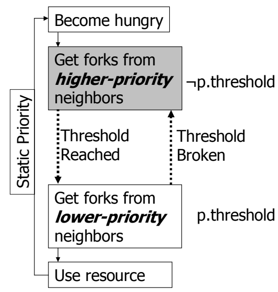
- *p* always yields forks to higher-priority neighbors
- Before *p* reaches its threshold, *p* **also** yields to lower-priority neighbors
- Failure locality is 2
- *p*.threshold is not stable
  - Yielding a fork to a higher-priority neighbor breaks *p*'s threshold



Static Priorities: Problems with Progress

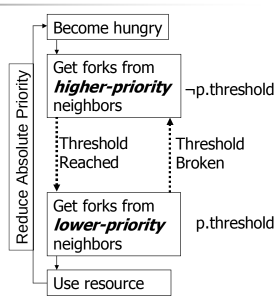
- Higher-priority processes can starve lower-priority neighbors





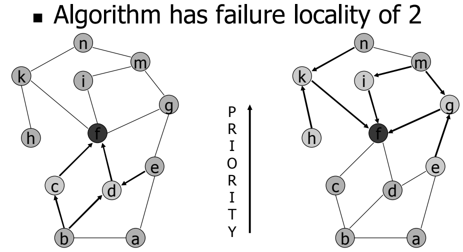
A New Algorithm: Dynamic Thresholds

- **Dynamic Thresholds**: A composition of the fork-collection scheme and dynamic process priorities
- A process at its threshold
  - can be overtaken by higher-priority neighbors
  - **but** at most once by each



Performance Analysis

- Algorithm has failure locality of 2



Lower-priority neighbors

Higher-priority neighbors

Algorithm Comparison

	Hygienic	Double Doorways	Bounded Doorways	Dynamic Thresholds
Safety	YES	YES	YES	YES
Progress	YES	YES	YES	YES
Failure Locality	n	4	2	2
FIFO Channels	✘	✘	✔	✘
Broadcast Messages	✘	✘	✔	✘
Interrupt Mechanism	✘	✘	✔	✘

A New Metric: Failure Sets

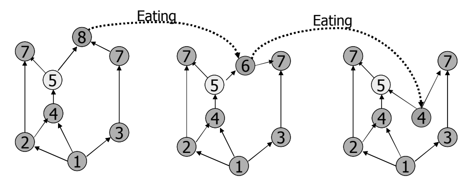
- We care about the *number* of impacted nodes, not their distance from the failure
- **Failure set of *p*** : the set of processes that starve if and when *p* fails
- Failure set $\subseteq$ *m*-neighborhood, where *m* is the failure locality of the algorithm
- Metric: cardinality of failure set
  - Depends on network topology

Minimizing Failure Sets

- **Observation:** High-priority processes that fail tend to have smaller failure sets
- *Why?* A high-priority process *p* has relatively more lower-priority neighbors
- These neighbors cannot reach their threshold without the fork from *p*
  - They yield forks to all requesting neighbors
  - This shields the rest of the network from *p*'s failure
- **Goal:** keep unreliable processes high in priority

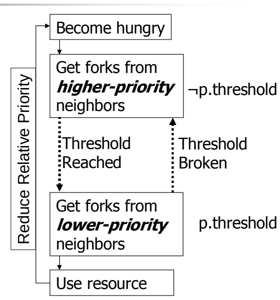
Refining Dynamic Thresholds

- After eating, reduce priority by an arbitrary amount
- Refined algorithm is still correct
  - Hungry processes can be overtaken a bounded number of times per neighbor



Refining Dynamic Thresholds

- Parameterize algorithm by a failure model
- Unreliable processes reduce priority less than reliable processes
- This keeps unreliable processes higher in priority



Contributions

- New algorithm: Dynamic Thresholds
  - Optimal failure locality of 2
  - Weaker assumptions on model
- New metric: Failure-set cardinality
- Parametric algorithm:
  - Incorporates failure model
  - Reduces *expected* cardinality of failure set

References

- The fault-tolerant fork-collection scheme
  - Choi and Singh, ACM TOPLAS 17(3), 1995
- Dynamic priorities in hygienic algorithm
  - Chandy and Misra, UNITY book, 1988
- Proof that 2 is optimal failure locality
  - Choi and Singh, IEEE TPDS 7(7), 1996

- {paolo,nsridhar,pike}@cis.ohio-state.edu