Characterizing Asynchronous Message-Passing Models Through Rounds

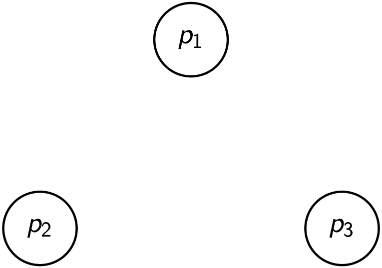
Adam Shimi, Aurélie Hurault, Philippe Quéinnec

IRIT, University of Toulouse

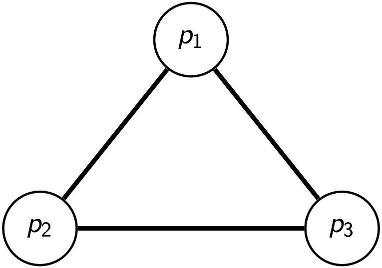
OPODIS, December 18, 2018

Table Of Contents

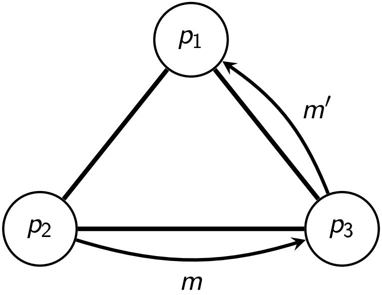
Message-passing Models



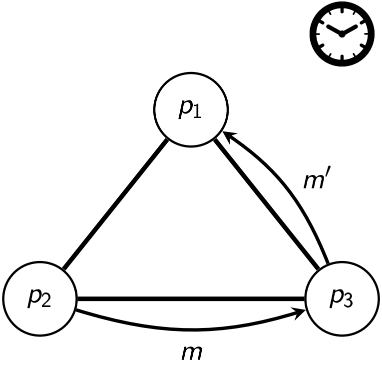
Message-passing Models



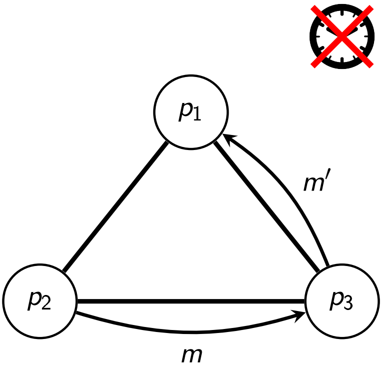
Message-passing Models



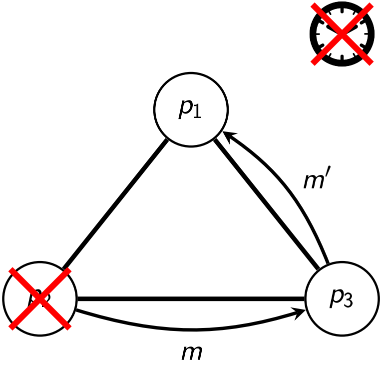
Message-passing Models



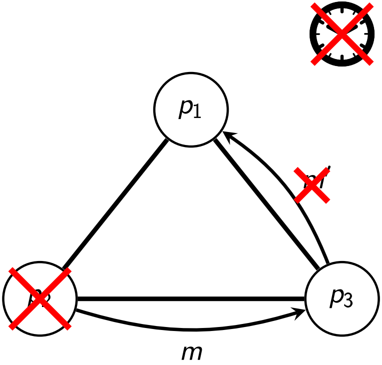
Message-passing Models



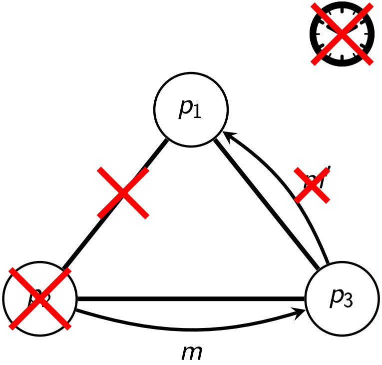
Message-passing Models



Message-passing Models



Message-passing Models

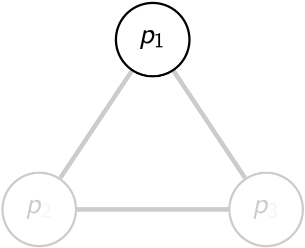


There are just too many incomparable models!

Plethora of parameters to consider

- Degree of synchrony: synchronous, asynchronous, partially synchronous...
- Kind of faults: crash-failure, crash-recovery, send-omission, message loss...
- Patterns of faults: upper bound on the number of faults, dependencies between faults...
- And more: static/dynamic topology of network, existence of global identities, upper bound on network size...

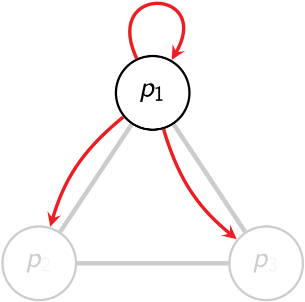
The Solution? Rounds!



Process behavior

- Broadcast messages tagged with its round number.
- Wait for messages with the same round number.
- Compute the next state and messages to send, then change round.

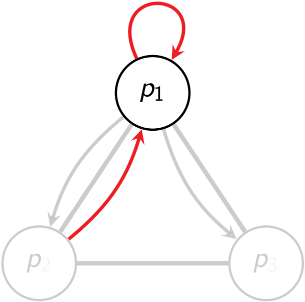
The Solution? Rounds!



Process behavior

- Broadcast messages tagged with its round number.
- Wait for messages with the same round number.
- Compute the next state and messages to send, then change round.

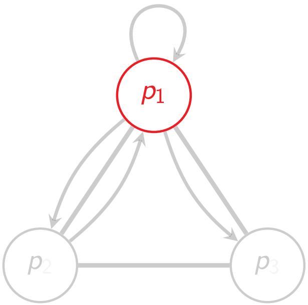
The Solution? Rounds!



Process behavior

- Broadcast messages tagged with its round number.
- Wait for messages with the same round number.
- Compute the next state and messages to send, then change round.

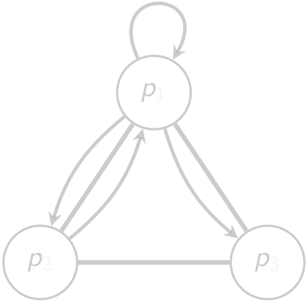
The Solution? Rounds!



Process behavior

- Broadcast messages tagged with its round number.
- Wait for messages with the same round number.
- Compute the next state and messages to send, then change round.

The Solution? Rounds!



Process behavior

- Broadcast messages tagged with its round number.
- Wait for messages with the same round number.
- Compute the next state and messages to send, then change round.

Rounds numbers can be different

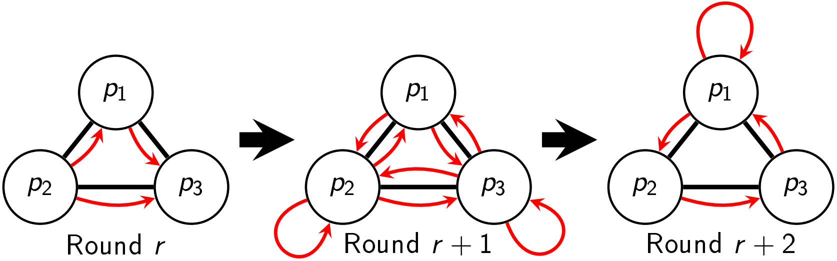
Asynchrony $\Longrightarrow$ Different processes at different rounds.
Messages received with a greater round number are buffered.
Late messages are discarded.

Heard-Of Predicate [Charron-Bost and Schiper 2009]

Definitions

- Heard-Of collection $\triangleq$ an infinite sequence of communication graphs, one for each round.
- Heard-Of predicate $\triangleq$ a predicate on Heard-Of collections.

A round is represented as a single communication graph, even if rounds might be asynchronous.



The Issue

On the value of rounds

We want to use rounds to study message-passing models:

- Design algorithms
- Prove impossibility results and lower bounds
- ...

But given a message-passing model, which Heard-Of predicate to use for this study?

Example: asynchronous model with at most F crashes

Let Π the set of n processes.
Then $\forall r > 0, \forall j \in \Pi : |HO(r, j)| \geq n - F$. [Charron-Bost and Schiper]

Why is this predicate the right one to study this model?

The Difficulty of Asynchrony

The question we address

What is the Heard-Of predicate one should use for studying a given message-passing model ?
Equivalent to asking what is the optimal way to implement rounds.

Asynchrony is harder than synchrony

- Synchrony $\implies$ **on-time delivery**.
  Implementation of rounds = wait for communication bound.
  Optimal way to implement rounds $\implies$ optimal Heard-Of.
- Asynchrony $\implies$ **unbounded delay**.
  Implementation of rounds = ???
  Comparing implementations of rounds is more involved.

The Difficulty of Asynchrony

The question we address

What is the Heard-Of predicate one should use for studying a given message-passing model ?
Equivalent to asking what is the optimal way to implement rounds.

Asynchrony is harder than synchrony

- Synchrony $\implies$ **on-time delivery**.
  Implementation of rounds = wait for communication bound.
  Optimal way to implement rounds $\implies$ optimal Heard-Of.
- Asynchrony $\implies$ **unbounded delay**.
  Implementation of rounds = ???
  Comparing implementations of rounds is more involved.

The Difficulty of Asynchrony

The question we address

What is the Heard-Of predicate one should use for studying a given message-passing model ?
Equivalent to asking what is the optimal way to implement rounds.

Asynchrony is harder than synchrony

- Synchrony $\implies$ **on-time delivery**.
  Implementation of rounds = wait for communication bound.
  Optimal way to implement rounds $\implies$ optimal Heard-Of.
- Asynchrony $\implies$ **unbounded delay**.
  Implementation of rounds = ???
  Comparing implementations of rounds is more involved.

Table Of Contents

Delivered Collections and Predicates

Definitions

A **Delivered collection** is an infinite sequence of communication graphs, and a **Delivered predicate** is a predicate on such collections.

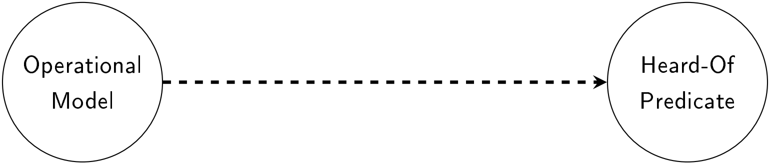
Same as Heard-Of, except all delivered messages are in the collection, even the late ones.

Delivered Collections and Predicates

Definitions

A **Delivered collection** is an infinite sequence of communication graphs, and a **Delivered predicate** is a predicate on such collections.

Same as Heard-Of, except all delivered messages are in the collection, even the late ones.

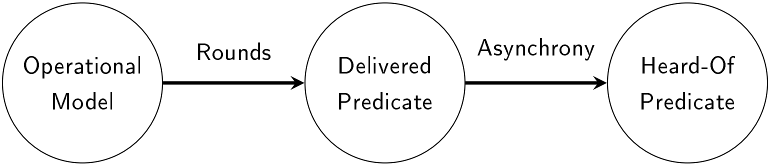


Delivered Collections and Predicates

Definitions

A **Delivered collection** is an infinite sequence of communication graphs, and a **Delivered predicate** is a predicate on such collections.

Same as Heard-Of, except all delivered messages are in the collection, even the late ones.



Rephrasing the Problem

How to go from Delivered predicate to Heard-Of one?

We have to consider all possible ways to implement rounds.
The only freedom of processes = decide whether they can change rounds or not.
This is captured by strategies: sets of the local states from which processes can change round. + some fairness on strategies.

Properties of strategies to look for

- **Correctness**: correct strategy for a Delivered Predicate implements a Heard-Of predicate.
- **Optimality**: Ordering on correct strategies. Maximal elements are the strategies implementing the right Heard-Of predicate for studying the model.

Validity

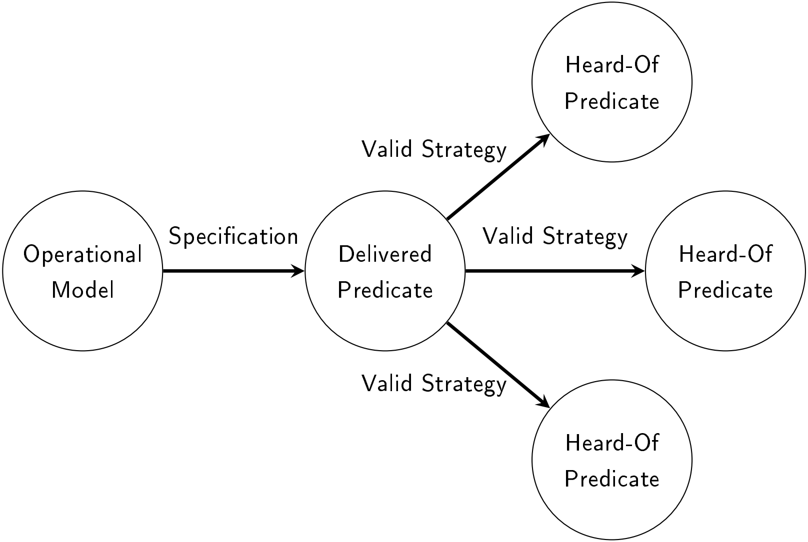
Valid run

A valid run t of *PDel* is a run of *PDel* where no process is blocked forever in a round.
An infinite number of rounds $\implies$ Defines a HO collection CHO_t by taking the received messages from the round just before going to the next one.

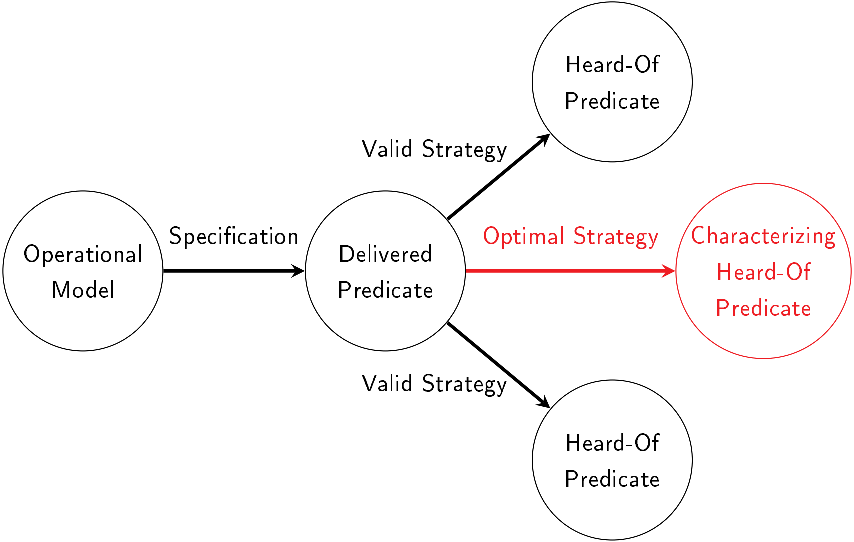
Valid strategy

A valid strategy f for *PDel* is a strategy such that all runs where all processes use f are valid runs.
It thus generates a HO predicate $PHO_f(PDel)$.

Comparing Valid Strategies



Comparing Valid Strategies



Domination

Domination order

A valid strategy f for $PDel$ dominates another valid strategy f' iff $CHO_f(PDel) \subseteq CHO_{f'}(PDel)$. A strategy dominating every valid one for $PDel$ is a dominating strategy for $PDel$.
The characterizing Heard-Of predicate is the one generated by a dominating strategy.

Domination

Domination order

A valid strategy f for *PDel* dominates another valid strategy f' iff $CHO_f(PDel) \subseteq CHO_{f'}(PDel)$. A strategy dominating every valid one for *PDel* is a dominating strategy for *PDel*.
The characterizing Heard-Of predicate is the one generated by a dominating strategy.

Intuition

- It is the most constrained predicate implementable by valid strategies
- All dominating strategies for *PDel* generate the same Heard-Of predicate.
- This dominating Heard-Of predicate implies all predicates generated by valid strategies.

Domination

Domination order

A valid strategy f for $PDel$ dominates another valid strategy f' iff $CHO_f(PDel) \subseteq CHO_{f'}(PDel)$. A strategy dominating every valid one for $PDel$ is a dominating strategy for $PDel$.
The characterizing Heard-Of predicate is the one generated by a dominating strategy.

Intuition

- It is the most constrained predicate implementable by valid strategies
- All dominating strategies for $PDel$ generate the same Heard-Of predicate.
- This dominating Heard-Of predicate implies all predicates generated by valid strategies.

Domination

Domination order

A valid strategy f for $PDel$ dominates another valid strategy f' iff $CHO_f(PDel) \subseteq CHO_{f'}(PDel)$. A strategy dominating every valid one for $PDel$ is a dominating strategy for $PDel$.
The characterizing Heard-Of predicate is the one generated by a dominating strategy.

Intuition

- It is the most constrained predicate implementable by valid strategies
- All dominating strategies for $PDel$ generate the same Heard-Of predicate.
- This dominating Heard-Of predicate implies all predicates generated by valid strategies.

Domination

Domination order

A valid strategy f for $PDel$ dominates another valid strategy f' iff $CHO_f(PDel) \subseteq CHO_{f'}(PDel)$. A strategy dominating every valid one for $PDel$ is a dominating strategy for $PDel$.
The characterizing Heard-Of predicate is the one generated by a dominating strategy.

Intuition

- It is the most constrained predicate implementable by valid strategies
- All dominating strategies for $PDel$ generate the same Heard-Of predicate.
- This dominating Heard-Of predicate implies all predicates generated by valid strategies.

Examples of dominating strategies

Asynchronous and at most F permanent crashes

Wait for at least $n - F$ messages from the current round.

At most B failed broadcasts by round

Wait for at least $n - B$ messages from the current round.

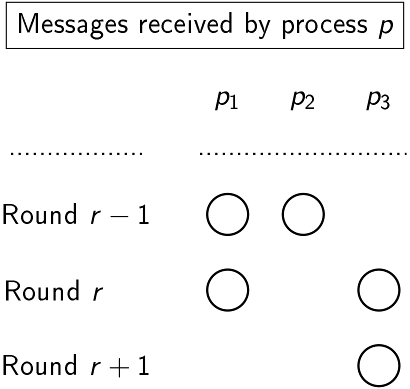
Asynchronous and at most F initial crashes

Wait for at least $n - F$ messages from the current round, and for messages from all processes from which a message was ever received.

At most L message losses in the whole run (Conjecture)

Wait for n messages from the current round, or for $n - 1$ messages from the current round and $n - 1$ messages from the next one.

Carefree and Reactionary Strategies



Carefree strategies

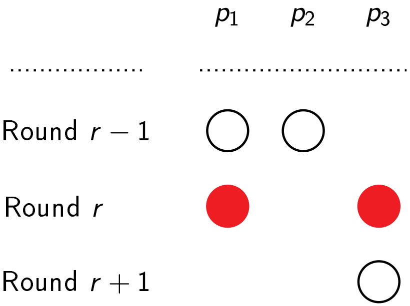
A carefree strategy depends only on messages from the current round.

Reactionary strategies

A reactionary strategy depends only on messages from past and current round, as well as the round number. It does not consider messages from future rounds.

Carefree and Reactionary Strategies

Messages received by process p



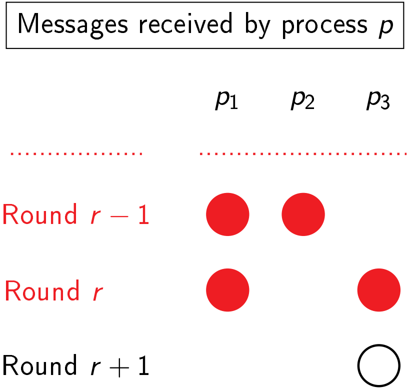
Carefree strategies

A carefree strategy depends only on messages from the current round.

Reactionary strategies

A reactionary strategy depends only on messages from past and current round, as well as the round number. It does not consider messages from future rounds.

Carefree and Reactionary Strategies



Carefree strategies

A carefree strategy depends only on messages from the current round.

Reactionary strategies

A reactionary strategy depends only on messages from past and current round, as well as the round number. It does not consider messages from future rounds.

Results for Carefree and Reactionary Strategies

Results for carefree strategies

- Necessary and sufficient condition for validity of carefree strategy.
- Existence, for any Delivered predicate, of a carefree strategy dominating all carefree strategies.
- Sufficient condition on Delivered predicate to be dominated by a carefree strategy.

Results for reactionary strategies

- Necessary and sufficient condition for validity of reactionary strategy.
- Existence, for any Delivered predicate, of a reactionary strategy dominating all reactionary strategies for this predicate.

Table Of Contents

Conclusion

The issue

Without synchrony, how to characterize the Heard-Of predicates implementable by asynchronous models?

Our solution

The right Heard-Of predicate = the one generated by dominating strategies for the corresponding Delivered predicate.

Methodology

- Represent asynchronous models by Delivered predicates.
- Find dominating strategies.
- Characterize the Heard-Of predicate implemented by these strategies.

Perspectives

The future can be useful

In the model with at most 1 message loss in the whole run, using messages from the next round implements a better predicate.

f_{asym} is the strategy containing all states with

- Either n messages from the current round
- Or $n-1$ messages from the current round and $n-1$ messages from the next round.

Other perspectives

- Consequences of the certainty of failures.
- Consequences of partial synchrony.
- Consequences of oracles.