Distributed Context Petri Nets: Managing Context Interaction in Distributed Environments

Juan Sebastian Sosa
José Daniel Fandiño

Promotor: Nicolás Cardozo

FLAG LAB

Background: Dynamically Adaptive Software Systems

- Systems that can automatically trigger predefined behavioral adaptations of the system as a result of information acquired via sensors or monitors.
- A dynamic adaptation is a modification that impacts a systems behaviour that occur during execution
- Eg: Low battery on a cellphone

Background: Context Oriented Programming

- Paradigm that allows dynamic adaptations
- Allows software modularization through adaptations that are activated and deactivated at run time
- Contexts as meaningful situations sensed in the systems surrounding environment
- Possible conflicting or contradicting adaptations

[17] Salvaneschi et al. Context-oriented programming a software engineering perspective

Background: Context Petri Nets

- Run time programming model that allows to ensure the consistency of adaptations in COP, since they serve as a formal model of the execution of the system
- Can represent one (singleton) or more contexts

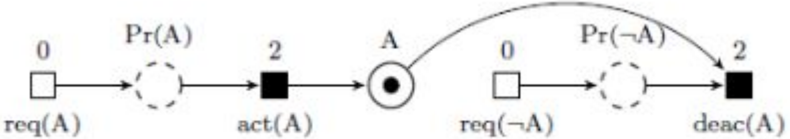


Figure 2.1: CoPN Context Representation

[4] Cardozo et al. Semantics for consistent activation in context-oriented systems. 2015

Background: Context Petri Nets

- Interaction between contexts is defined by means of *context dependency relations*
  - Exclusion (A ▫—▫ B)
  - Causality (A —▷ B)
  - Implication (A —▶️ B)
  - Requirement (A —◀️ B)
  - Suggestion (A - -▷B)

[5] Cardozo et al. Enabling consistent composition of context-dependent behavior. 2012

Background: Context Petri Nets

- **ext** and **cons** functions add transitions and arcs to a Petri net given the relations
- Eg: Consider the PN with contexts A and B, where A −▷ B

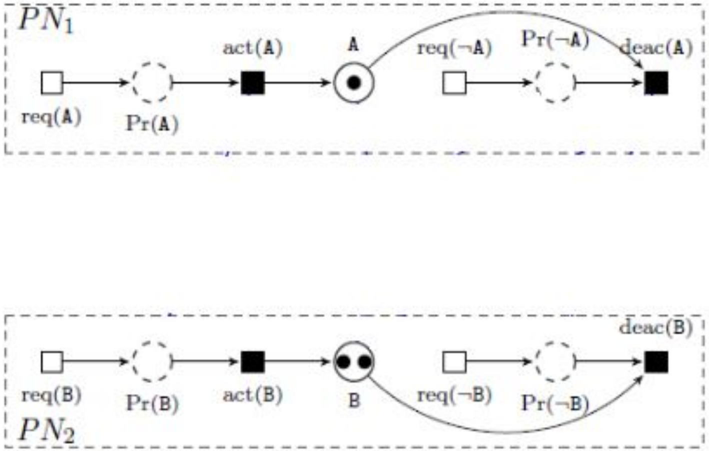


Figure 5.3: Causality relation.

Background: Context Petri Nets

- **ext** and **cons** functions add transitions and arcs to a Petri net given the relations
- Eg: The resulting CoPN of **ext**(**cons**(PN, ❬−▷, A, B❭), ❬−▷, A, B❭)

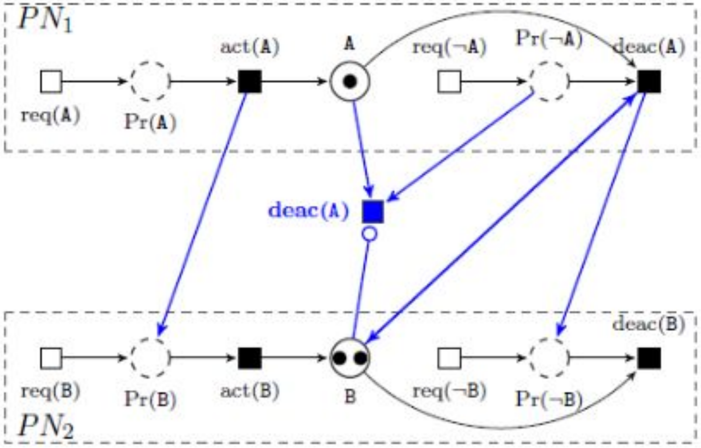


Figure 5.3: Causality relation.

Problem

- What about distributed systems?

- CoPN per device
- If a device joins, a single CoPN is unable to represent the context associated with the device
- If a device leaves the environment, the system is partitioned into two new different systems

Distributed Context Petri Nets (DCoPN)[1]

[1] Submitted to the International Workshop on context-oriented programming

DCoPN: Model

- 5-tuple $\langle PN, UT, F, F_\circ, R\rangle$
- UT, F, and $F_\circ$ are modified dynamically with connections and disconnections

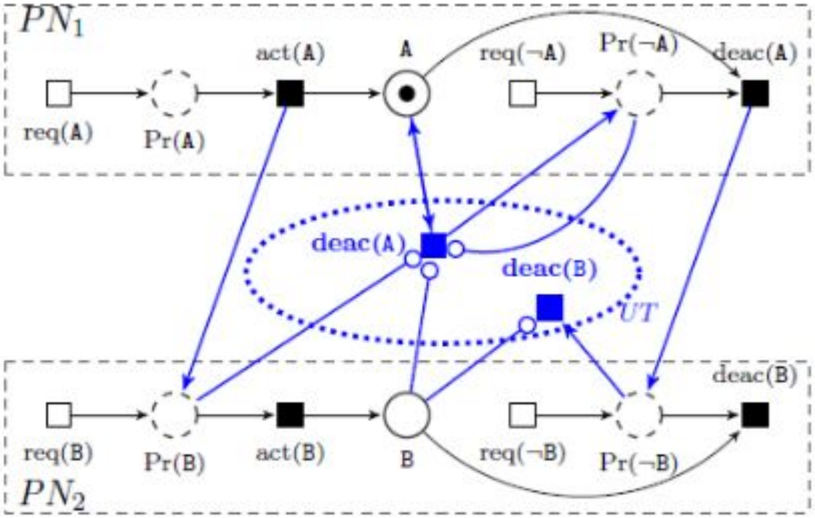


Figure 3.1: DCoPN example of A–▶B.

DCoPN: Inconsistency Management

Due to the connections and disconnections common in distributed systems the DCoPN model presents two main problems to deal with when managing contexts:

- Unstable states
- Conflicting states

DCoPN: Inconsistency Management - Unstable states

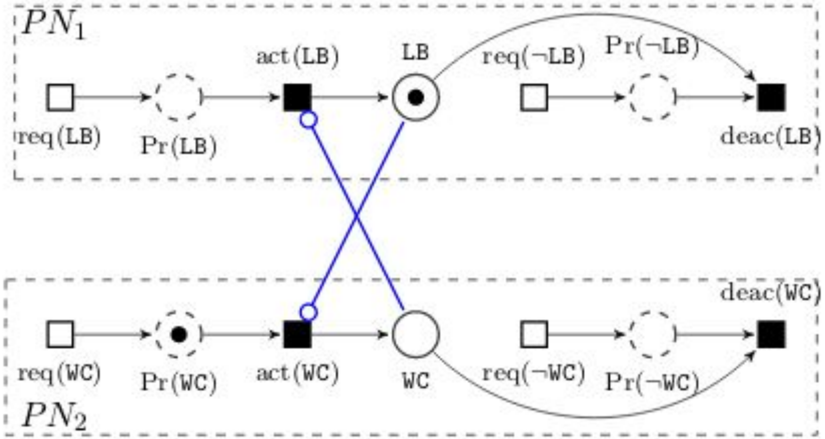
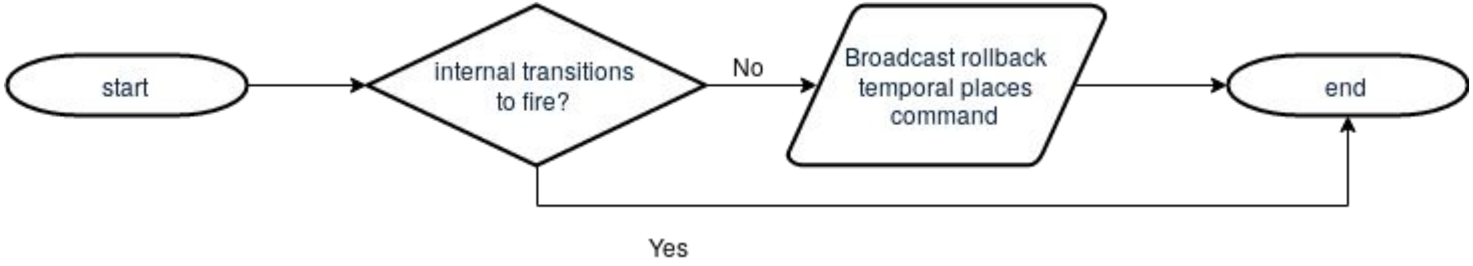


Figure 5.2: Example of a unstable state in a exclusion relation.

DCoPN: Inconsistency Management - Unstable states


start
internal transitions to fire?
No
Broadcast rollback temporal places command
end
Yes

DCoPN: Inconsistency Management - Conflicting states

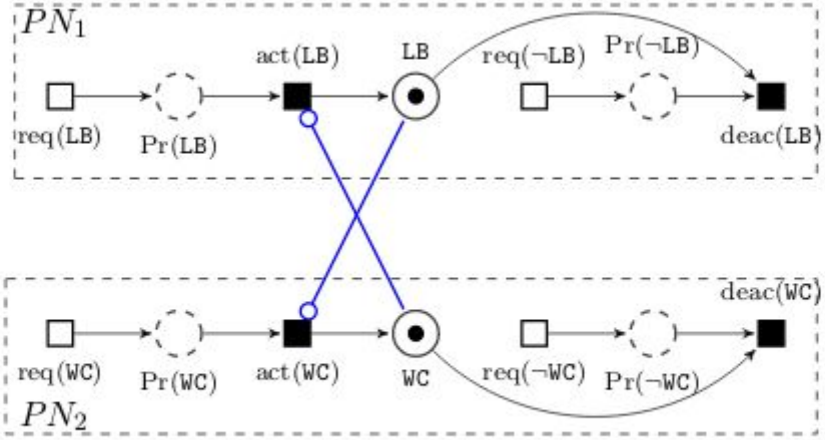
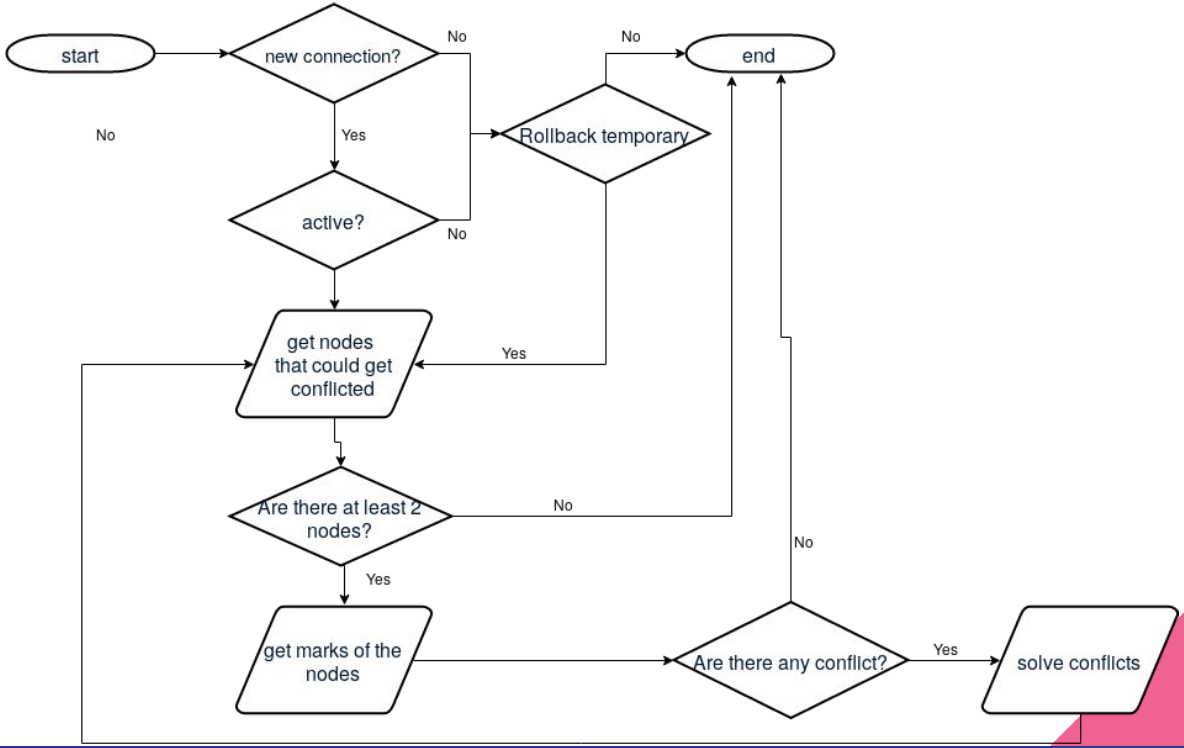


Figure 5.1: Example of a conflict in a exclusion relation.

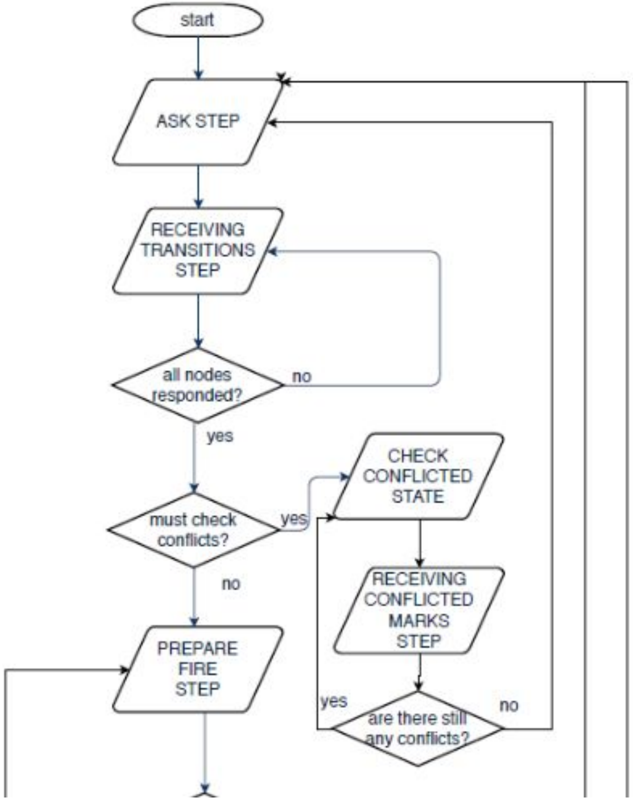
DCoPN: Inconsistency Management - Conflicting states



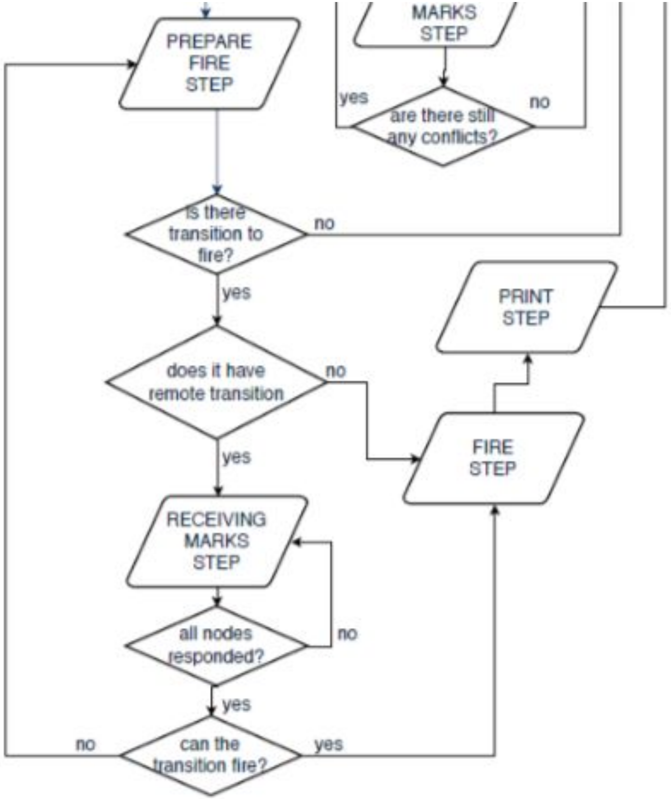
DCoPN: Implementation

- Implemented on Go using Noise library for p2p communication
  Available at: https://github.com/FLAGlab/DCoPN

DCoPN: Implementation - Protocol

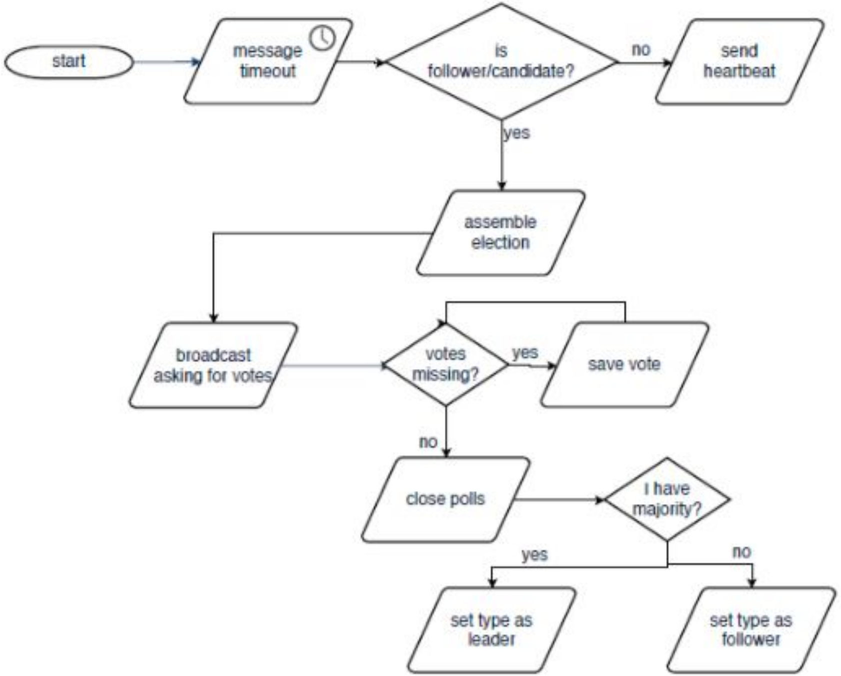


DCoPN: Implementation - Protocol

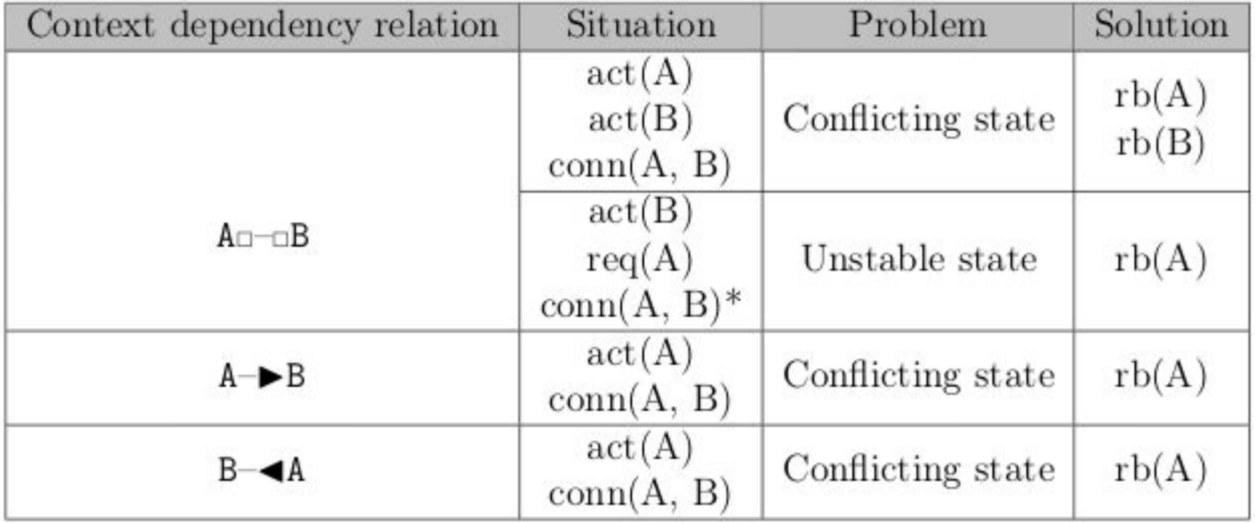


DCoPN: Implementation - Leader election

- Raft algorithm adaptation for leader election



DCoPN: Validation



Context dependency relation	Situation	Problem	Solution

A□–□B	act(A)
	act(B)
	conn(A, B)
		Conflicting state	rb(A)
			rb(B)

	act(B)
	req(A)
	conn(A, B)*
		Unstable state	rb(A)

A–▶B	act(A)
	conn(A, B)
		Conflicting state	rb(A)

B–◀A	act(A)
	conn(A, B)
		Conflicting state	rb(A)

Table 5.1: Context dependency relation inconsistency situations

Conclusion and Future Work

➔ We successfully propose a model able to maintain consistency for connections and disconnections in dynamically adaptive distributed systems

★ Currently we solve inconsistencies with a single rollback, could be improved to follow different heuristics

★ We depend on manual specification of conflicts, could be improved to automatic detection

★ We developed a specific case of distributed Petri nets, this field could be explored further

Bibliography

[4] Nicolás Cardozo, Sebastián González, Kim Mens, Ragnhild Van Der Straeten, and Theo D'Hondt. Modeling and analyzing self-adaptive systems with context petri nets. In Proceedings of the Symposium on Theoretical Aspects of Software Engineering , TASE'13, pages 191 – 198. IEEE Computer Society, July 2013.

[13] Tadao Murata. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4):541 – 580, April 1989.

[14] Diego Ongaro and John Ousterhout. In Search of an Understandable Consensus Algorithm. page 18

[11] Marvin L. Minsky. Computation: Finite and infinite machines. The American Mathematical Monthly , 75, 04 1968