# Observer – Formal Online Validation of Distributed Systems

Michel Diaz,
Guy Juanole and Jean-Pierre Courtiat

## Agenda

- Problem Definition
- Background work
- Approach in the paper
- Evaluation of solution
- Take aways
- Shortcomings

## Problem Definition

- Designing self checking distributed systems
- Build system whose online behavior is checked against formal model
- Approach that relates two design phases
  - Reliability with which software specifications are described and correctly implemented in the implementation
  - Runtime checking of correct behavior in actual environments including hardware failure, software bugs and human errors
- Concept of global behavior

## Background Work

## Definitions

- Let S be well defined set of admissible values.
- What is fault secure system? (safety property)
- What is self-testing system? (liveness property)
- What is self-checking system?
- What is distinctness concept?
- What is an observer-worker system?

## Observer Principle

- Redundancy
- Reference
- Visibility – Cooperative worker, spying observer
- In O-W system, if F is the set of faults occurring in only one subsystem, and if set of fault detection sequences are applied during runtime in each subsystem, then entire system is self-checking.

## Spying Observer - Advantages

- Observer & worker design and implementations can be independent
- Same observer can be used for checking different implementations of worker or family of distributed systems.

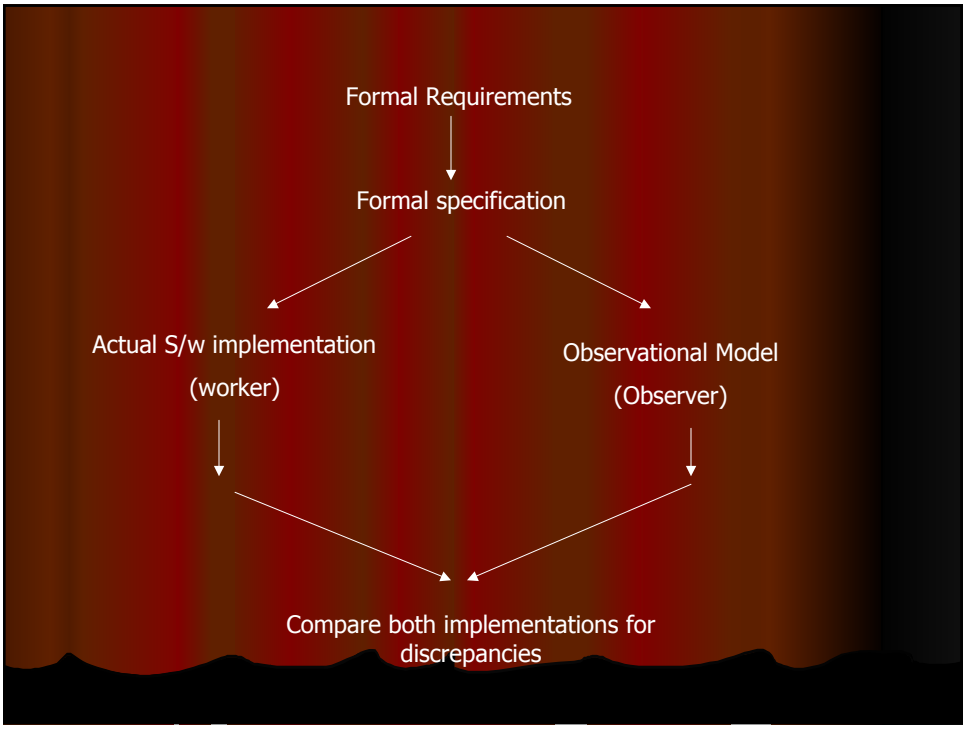
## Initial Problems faced…

- Can all possible faults in each component be tested during runtime?
- Can this set of tests be applied before fault occurs in other component?
- Bt, set of test behavior, must belong to the set of sequences that occur during runtime behavior.

# Approach in the Paper

## Concept of Formal Observer

- Formally proved run-time checked design not considered.
- Similarly, specific test based observer design not considered.
- Worker can be any distributed system implementation.
- Concept of quasi-self checking observer, based on formal model & exhaustive verification of this formal model.
- Formal observer is
  - software independent of specific protocols
  - Data defined by protocol to be checked, which can be formally specified and verified.
- System is quasi-self checking is it is an observer-worker system and observer is a formal observer.



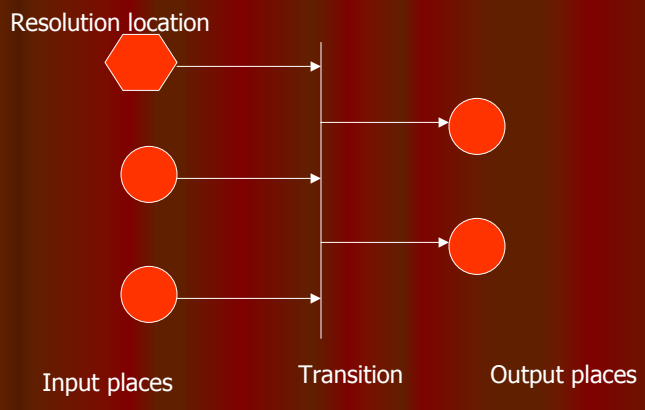

# Observational Model

- Observational models do not observe complete behavior, but only partial behavior of interest.
- Can neglect certain internal events in other processors not observable during runtime.
- Should be able to express simplified specifications of distributed systems
- Should support verification procedures.
- Act as basis for implementing observer.

# Petri Nets – A brief introduction

- Used to study interconnection of parallel activities
- Consists of places (conditions) and transitions (events) connected by directed arcs
- Transition is fireable if each input place contains atleast one token.
- Firing transfers tokens from each input to each output.
- Firing of enabled transition takes zero time
- Can use *resolution location* to extend this concept to evaluation nets, where resolution procedures and transition procedures can be evaluated when there is an input/firing of transition occurring – the model can be complicated to deal with data.

# Illustration of Petri Net



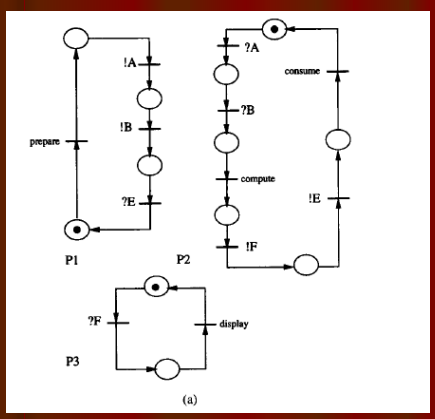
# Implementing Observational model

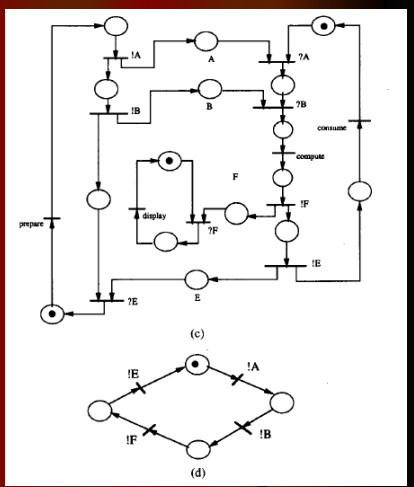
- Simulate behavior of global Petri net.
- Starting from initial marking, model evolves by firing firable not observable transitions until the model reaches a fireable observable transition.
- System stops at this stage and waits till

  - Corresponding observable event occurs, in which case transition is fired and subsequently starts again with the same procedure

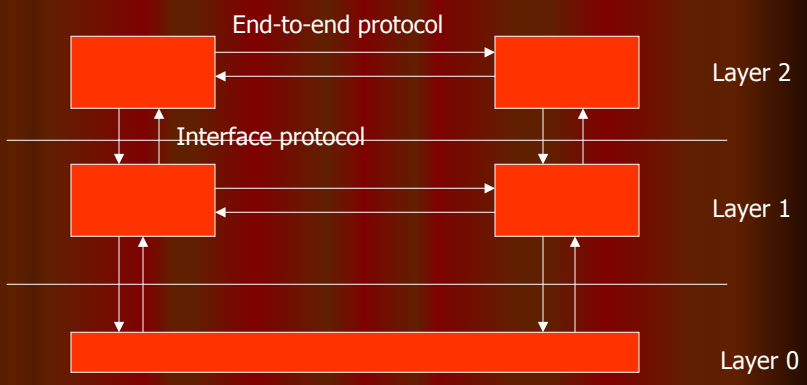
  - An observable event occurs which is not expected, error is detected. State information is saved and error is recorded.

# Complex observational models



# Layered Distribution Systems




# Layered Distribution Systems

- At each layer, each protocol entity is represented by Petri net, hence hierarchy of local models is obtained.
- Models of peer entities are connect together using Petri net description of service provided by lower layer .Hence global model of layer is constructed.
- Global models of interacting entities is given as data input to the observer
- One protocol of one layer to be represented as one model?
- Behavior of one layer is set of connections between protocol entities, each connection in a layer being called a dialog.
- All dialogs run the same protocol, so only one Petri net model of layer is required.
- However, owing to concurrent dialogs, dynamic process management has to be done by observer.
- One process, simulating one Petri net runs one dialog.
- Creating and deletion of dialogs governed by observer based on incoming protocol data units.

## Evaluation

- Observer deployed to check online a token bus protocol, and testing / performance evaluation of multiple layers in layered protocol.
- Observer has been found useful in performance measurement during system life cycle.
- Observer has been found useful in debugging communication software during development phase.
- Helpful in online detection of faults coming from hardware faults or software errors during runtime.

## Take Aways

- Global behavior checking (validation functions not in individual processes).
- Separation of implementation of observer and worker.
- Same observer can be used for validating family of distributed systems.
- Validates the reliability of formal specification, implementation as well as enforces online self-checking.

# Negatives

- How to implement spying observer with encrypted packets?
- How to implement spying observer in environment with tunneling facilities?
- How to deal with systems with no broadcast services?
- What happens when the observer itself loses packets (observer recovery)?