AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Test of Distributed Systems
Lecture 1

Introduction:

Course outline

Course plan

Starting with Spin

29-03-2012

Today's lecture

- Course outline – what's in and what's out
- Course contents – looking at the plan
- Test – elaboration on some of the concepts
- Starting with Spin
- Next time

Today's lesson - context

- Learning goals
  - Define and describe fundamental problems associated with test of distributed systems.
  - Use a tool for abstract description and verification of qualities of distributed systems, including explaining advantages and weaknesses of using such a tool.
- Previous lessons
  - None ☺

Today's lecture

- Course outline – what's in and what's out
- Course contents – looking at the plan
- Test – elaboration on some of the concepts
- Starting with Spin
- Next time

Course outline – the terms: “test”, “distributed” & “system”

We need narrow down the meaning of these concepts …..

- System
  - The good (easy): a software *system* put together with smaller, cooperating software *subsystems*.
  - In practice, we’ll often mean *system of independent processes working together to solve a larger task*

Course outline – the terms: “test”, “distributed” & “system”

- Test
  - The bad (things gets a bit more murky here) – what do we mean by test?
  - We normally group test activities in two types of activities (two views for different purposes, really)
    - Verification
    - Validation
  - Verification = are we building the thing right?
  - Validation = are we building the right thing

Course outline – the terms: “test”, “distributed” & “system”

- Test (contd.)
  - No matter which type of test (verification or validation), there are generally two ways of checking things – what are they?
  - State
    - In state based testing we compare the system’s state to what we expect
  - Behavior
    - In behavioral testing we compare the system’s behavior to what we expect

Course outline – the terms: “test”, “distributed” & “system”

- Test (contd.)
  - In this course we will put some extra meaning into the notion of test
  - Rather than test the actual system, we will be more concerned with verifying *a model* of the system we are making, and trying to specify desired qualities of the system in terms of formal properties of the system’s model
  - Think of it as verification/validation of the design

Course outline – the terms: “test”, “distributed” & “system”

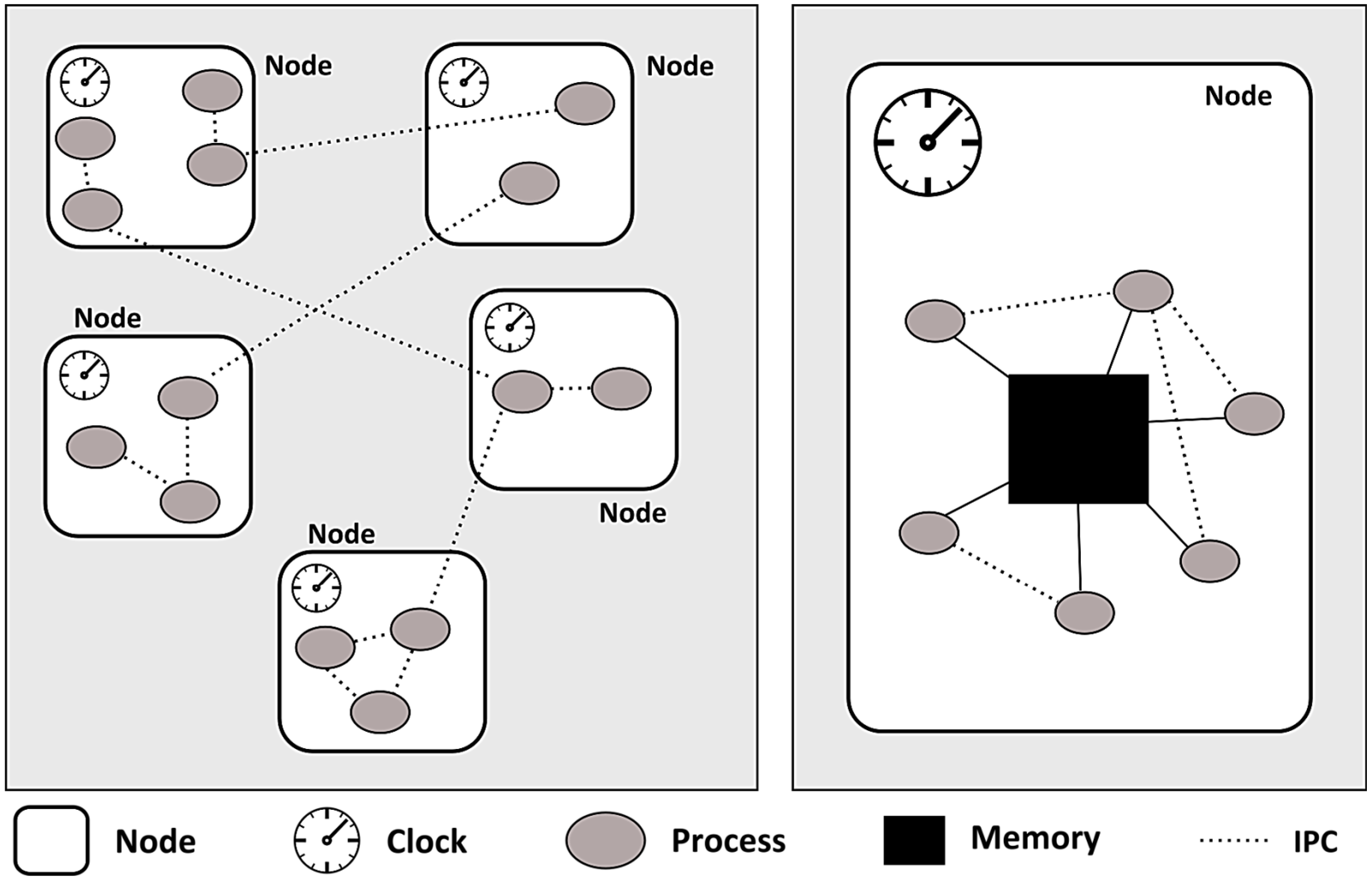
- Distributed
  - The ugly (we need to be very precise) – what is a distributed (sw) system?
  - It turns out that the key (most nasty) problem is distributed time, and not so much distributed state
  - Due to the relativistic nature of time, things get difficult (at least testing does) since we can’t establish a single, all-powerful process that can see all state as it unfolds during the computation

Course outline – the dimensions and problems

- Global/local state
- State/behaviour

Difficulty	Scope	
	Local	Global
State Based Test	Easy	Hard
Behavioral Test	Easy	Hard

Course outline – our model of a distributed system



Course outline – our model of a distributed system

- Our general model of a distributed system is an *asynchronous distributed system:*
  - No bounds on relative process speeds
  - No bounds on message delays

Today's lecture

- Outlining – what's in and what's out
- Course contents – looking at the plan
- Test – elaboration on some of the concepts
- Starting with Spin
- Next time

Course contents

- Course plan – let's have a look
- Learning goals
  - Define and describe fundamental problems associated with test of distributed systems.
  - Analyze and explain fundamental problems associated with verification of predicates on global and temporal qualities in distributed systems.
  - Compare and assess different possible schemes for verifying predicates on global and temporal qualities in distributed systems.
  - Use a tool for abstract description and verification of qualities of distributed systems, including explaining advantages and weaknesses of using such a tool.

Course contents

- The materials – let's have a look
- The book
  "Principles of the Spin Model Checker"
- The tool
  - Spin

Today's lecture

- Outlining – what's in and what's out
- Course contents – looking at the plan
- Test – elaboration on some of the concepts
- Starting with Spin
- Next time

Elaboration on test - verification

- Are we building the thing right?
- During **design** and implementation
- Should generate early and timely feedback
- Test as design driver (Test Driven Development)
- Remember the modern test mantra: Test early, test often, test enough
- So: Verify early, verify often, verify enough

Elaboration on test - validation

- Are we building the right thing?
- While validation should also be performed iteratively, it is primarily a post-implementation activity (*did* we build the right thing?)
- So it also generates feedback, but late!
- But if we validate our *design* early on, we will at least know that our design works before we start coding it

Elaboration on test - strategies and patterns

- *Strategy:* reduce things to the local scope – divide and conquer
- Approach (integration test):
  - Test each sub-part in isolation
  - Integrate and verify compound system
  - Keep on until all parts tested and integrated, i.e. until full system assembled
- But at some point we'll end up doing global testing (state or behavior) – and it's still hard

Elaboration on test - strategies and patterns

- *State based test*
- Really: using predicates (on local or global state) to test things
- $\forall$ list<int> ilist : ilist.size() = $|$ilist$|$
- $\forall$ Processes P_x in system with $Output_x$: $\Sigma(Output_x) < 100$
- Easy on local state, much harder on global state

Elaboration on test - strategies and patterns

- *Behavioral test*
- Inherently coupled to time, played out in time (*first* we expect event-a and *then* event-b)
- So we need formalisms that takes time into account
- LTL (Linear Temporal Logic (other types exist))
  - Views time as a single, linear dimension
  - Allows us to formulate conditions for correct behavior in time

Elaboration on test - strategies and patterns

- Realtime or post-run test
- Either:
  - Test state/behavior as it unfolds
  - Log information (“take notes”) during the computation and analyze it afterwards
- Realtime testing will always impact the system (i.e. we can’t observe a system without changing its behavior – now, where have we heard that before?)
- Post-run testing may also impact system behavior (due to logging)

Elaboration on test
- so where does this course fit in?

- We'll focus on
  - The hard parts, looking at global state or behavior
  - Verification (rather than validation)
  - Design (rather than implementation (for more than one reason))
- We'll use LTL to work with behavioral aspects

Today's lecture

- Outlining – what's in and what's out
- Course contents – looking at the plan
- Programming language poll
- Test – elaboration on some of the concepts
- Starting with Spin
- Next time

Starting with Spin

- The article (“A Primer om Model Checking”)
- Installation
- Demo
- A select few exercises

Starting with Spin - what does it do

- Several things
- The four “modes” of Spin
  - Random simulation (example program run)
  - Interactive simulation (guided by user)
  - Verification (full state space exploration)
  - Guided simulation (using trail from verification)
- Verification of the full state space is the interesting part (and more or less the only one)

Starting with Spin - how does it work

- Won't go into small nitty-gritty details
- The most important aspect from a user perspective (at least initially) is understanding what ProMeLa is and how to use it
- **Pro**cess **Me**ta **La**nguage
  - For modelling systems/protocols
  - Not for hard-core programming
  - Abstraction is the key

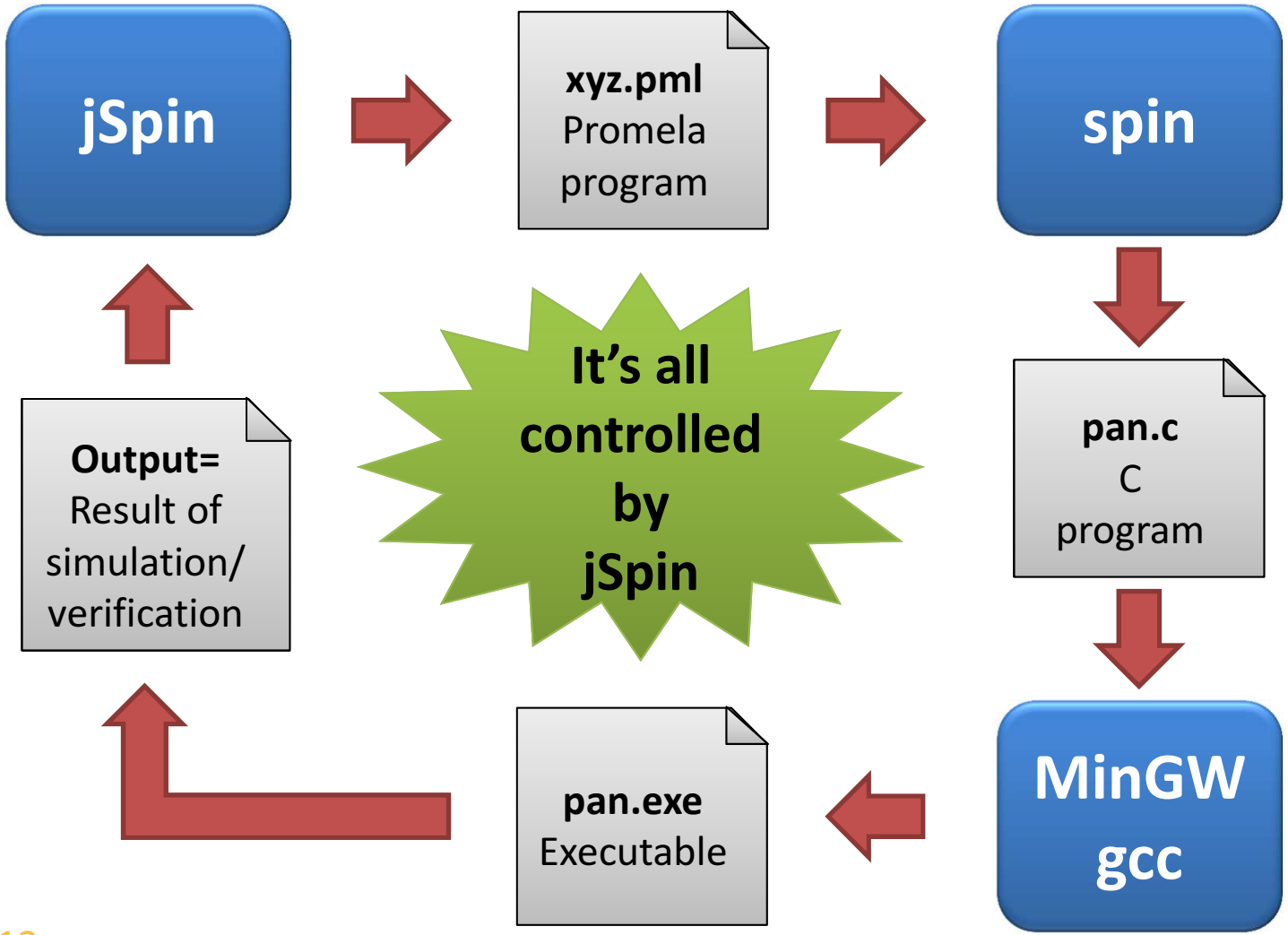
Starting with Spin - installation

- It’s a small footprint installation
- We’ll use the jSpin frontend
- Entire directory can simply be deleted for un-installing jSpin/Spin
- Java is assumed
- Use the installation guide
- Let’s go …

Starting with Spin - demo

- Running Spin via jSpin
- Looking at a few small programs to see ProMeLa at work

(j)Spin – behind the scenes



Starting with Spin
- a few small exercises

- Put on the hard hat and get to work …

Starting with Spin
- a few small exercises

- Exercise 1.1
  - Make a small Promela program with 3 processes.
  - ProcessA should start ProcessB and ProcessC
  - When started, processB should increment a global counter from 0 to 10, and then terminate
  - When started, ProcessC should wait for the global counter to reach 10, and then count it down to 0 before terminating
  - ProcessA should wait for ProcessB and ProcessC to terminate, write the value of the global counter and then terminate

Starting with Spin
- a few small exercises

- Exercise 1.2
  - Find the example tds2.pml on the course web page
  - Modify the example so that the deadlock problem disappears
  - Demonstrate that the deadlock problem is gone

Today's lecture

- Outlining – what's in and what's out
- Course contents – looking at the plan
- Programming language poll
- Test – elaboration on some of the concepts
- Starting with Spin
- Next time

Next time

- Exercise(s) to complete
  - Spin exercises
- Reading material
  - POTSMC Ch. 3-4 (concurrency & synchronization)
- Instructions, guidance
  - Stefan??