

Session 2: Shawn Configuration

Claudia Becker

Institute of Telematics, University of Lübeck

November 17th, 2008

- 1 A short overview: Shawn Simulation Environment and Models
Simulation Environment
Models
- 2 Step by Step Simulation
Simulation Parameters
Step by Step Simulation Configuration
- 3 Config files
- 4 JShawn
- 5 Converting a JShawn configuration file into a .conf file
- 6 A more complex example
- 7 Exercises

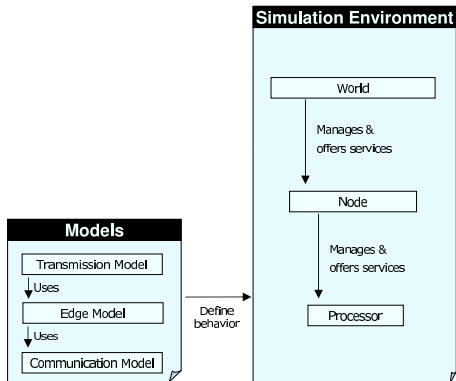
Outline

- 1 A short overview: Shawn Simulation Environment and Models
Simulation Environment
Models
- 2 Step by Step Simulation
Simulation Parameters
Step by Step Simulation Configuration
- 3 Config files
- 4 JShawn
- 5 Converting a JShawn configuration file into a .conf file
- 6 A more complex example
- 7 Exercises

Shawn Simulation Environment and Models (1/2)

The Shawn *simulation environment* is the home of

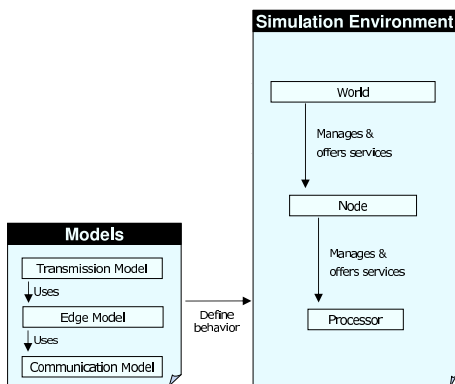
- the virtual *world* which contains the simulation objects like
 - *nodes*. Nodes represent simulated sensor nodes and act as containers for
 - *processors* which contain the application logic and hence specify the behavior of the nodes.



Shawn Simulation Environment and Models (2/2)

The Shawn simulation environment is affected by the chosen Shawn models.

- A *Communication Model* specifies whether two nodes can communicate or not.
- An *Edge Model* provides a graph representation of a network by querying the Communication Model.
- A *Transmission Model* defines message transmission properties like delay, drop, and modification.



Outline

- 1 A short overview: Shawn Simulation Environment and Models
Simulation Environment
Models
- 2 Step by Step Simulation**
Simulation Parameters
Step by Step Simulation Configuration
- 3 Config files
- 4 JShawn
- 5 Converting a JShawn configuration file into a .conf file
- 6 A more complex example
- 7 Exercises

Simulation Parameters

Simulation Parameters

- are defined as:

key=value

- can be accessed in Shawn through the simulation environment:

```
const shawn::SimulationEnvironment& se =  
    owner().world().simulation_controller().environment();
```

```
int send_round_  
    se.optional_int_param("send_round", 0);
```

- can be

- required:

```
required_string_param("KEY");
```

- or optional:

```
optional_string_param("KEY", DEFAULT_VALUE);
```

Step by Step Simulation (1/4)

1 Go to *Shawn/buildfiles* in cygwin shell.

2 Enter

```
./shawn
```

Something like

```
$ ./shawn
rufe init_apps auf
init_apps: init_examples(sc); init_reading(sc); init_topology(sc); init_vis(sc);
```

```
Initialising examples
init_topology_elevation
init_topology_generator
init_topology_node_gen
init_topology_node_mod
init_topology_point_gen
init_topology_point_mod
init_topology_topology
init_legacyapps: init_ws08(sc);
Initialising WinterSchool 2008 module (WS08)
[ NOW READING CONFIGURATION AND COMMANDS FROM STDIN --      ]
[ IF YOU ARE CONFUSED WHAT'S GOING ON HERE, TRY PIPING THE  ]
[ FILE shawn/apps/helloworld/randomhello.conf INTO THIS BINARY. ]
```

should appear.

Step by Step Simulation (2/4)

3 Pass the desired configuration settings to Shawn by typing e.g.:

```
prepare_world edge_model=simple comm_model=disk_graph  
range=1
```

The command *prepare_world* creates a new simulated world. The example above contains the simulation parameters

- `edge_model=simple`,
- `comm_model=disk_graph`,
- `range=1`

which affect the world's properties.

Step by Step Simulation (3/4)

- 4 After creating a world it has to be filled with life. Therefore we e.g. need
- a specification of the desired topology,
 - a number of nodes,
 - to specify the kind of processors that should define the behavior of the nodes.

Pass these settings to Shawn by typing:

```
rect_world width=25 height=25 count=800  
processors=helloworld
```

Step by Step Simulation (4/4)

- 5 Now we are ready to start a simulation by typing:

```
simulation max_iterations=10
```

The parameter *max_iterations* determines the simulation's operating time. That period of time is to be defined by a number of simulation rounds (here: 10 rounds).

Outline

- 1 A short overview: Shawn Simulation Environment and Models
Simulation Environment
Models
- 2 Step by Step Simulation
Simulation Parameters
Step by Step Simulation Configuration
- 3 Config files**
- 4 JShawn
- 5 Converting a JShawn configuration file into a .conf file
- 6 A more complex example
- 7 Exercises

.conf-files (1/2)

To facilitate running simulations you can store all needed configuration instructions in so-called *Config-Files* (.conf-files), instead of typing each instruction one after the other in a terminal for every new simulation.

.conf-files (2/2)

- 1 A *Config-File* could look like this:

```
# This is a comment that is not processed.  
# Comments must be written in a separate  
# line, and are not allowed to be attached  
# to a line that contains a command for Shawn.  
prepare_world edge_model=simple  
  comm_model=disk_graph range=1  
rect_world width=25 height=25 count=800  
  processors=helloworld  
simulation max_iterations=10
```

- 2 Now you can easily start a simulation by typing

```
./shawn -f path_to_my.conf
```

in *Shawn/buildfiles*.

Outline

- 1 A short overview: Shawn Simulation Environment and Models
Simulation Environment
Models
- 2 Step by Step Simulation
Simulation Parameters
Step by Step Simulation Configuration
- 3 Config files
- 4 JShawn**
- 5 Converting a JShawn configuration file into a .conf file
- 6 A more complex example
- 7 Exercises

JShawn (1/4)

Another way of configuring Shawn simulations is given by the use of JShawn.

- Language: Java
- Requires at least Java 1.6.
- Allows the execution of the full Java syntax. E.g. allows
 - the use of loops
 - to print debug messages:

```
System.out.println( "my debug message" );
```
 - the declaration of variables:

```
int variable = value;
```


JShawn (2/4)

The main commands to access Shawn from a JShawn configuration file are:

- Declaration of global variables:

```
shawn.setGlobalVariable( "variable", "value" );
```

- Invoking a given task with the possibility to set local variables that can only be seen by the task:

```
shawn.runCommand( "task-name", "settings of required"  
                  "and optional parameters"  
                  "of the named task" );
```

JShawn (3/4)

A simple JShawn-file could look like this:

```
shawn.runCommand("prepare_world", "edge_model=simple "  
+ "comm_model=disk_graph range=10" );
```

```
shawn.runCommand("rect_world", "width=25 height=25 "  
+ "count=800 processors=helloworld");
```

```
shawn.runCommand("simulation", "max_iterations=10");
```

JShawn (4/4)

To start a simulation you need to know

- the path to *jshawn-allinone.jar*,
- the path to *shawn.exe*,
- and the path to your JShawn file.

The start instruction could look like this:

```
java -jar C:\Shawn\jshawn-allinone.jar  
-s C:\Shawn\buildfiles\shawn.exe  
-b ws08.jshawn  
| C:\cygwin\tee.exe > out.txt
```

Run a simulation by typing that instruction in your cygwin shell.

Outline

- 1 A short overview: Shawn Simulation Environment and Models
Simulation Environment
Models
- 2 Step by Step Simulation
Simulation Parameters
Step by Step Simulation Configuration
- 3 Config files
- 4 JShawn
- 5 Converting a JShawn configuration file into a .conf file**
- 6 A more complex example
- 7 Exercises

Converting a JShawn configuration file into a .conf file

Sometimes it is helpful to be able to simply convert an existing JShawn configuration file into a .conf file, for instance for the purpose of debugging an application with *gdb*.

Converting can be easily accomplished by adding the following line at the beginning of your JShawn-file:

```
shawn.traceHistory("file.conf");
```

Then invoke JShawn with the command line option "--dryrun", e.g.:

```
java -jar jshawn-allinone.jar --dryrun -b yoursript.jshawn
```

This will create "file.conf" with the sequence of commands that would have been issued to Shawn.

Outline

- 1 A short overview: Shawn Simulation Environment and Models
Simulation Environment
Models
- 2 Step by Step Simulation
Simulation Parameters
Step by Step Simulation Configuration
- 3 Config files
- 4 JShawn
- 5 Converting a JShawn configuration file into a .conf file
- 6 A more complex example**
- 7 Exercises

A more complex example

```
1 shawn.setGlobalVariable( "processors", "helloworld" );
2 int number_of_simulations = 15;
3 int number_of_nodes = 0;
4
5 for ( int i = 1; i <= number_of_simulations; i++ )
6 {
7     System.out.println( "simulation nr = " + i );
8
9     shawn.runCommand( "prepare_world", "edge_model=simple " + " comm_model=disk_graph
10     range=10 "+ " transm_model=stats_chain " );
11     shawn.runCommand( "chain_transm_model name=random_drop_chain probability=0.15" );
12     shawn.runCommand( "chain_transm_model name=reliable " );
13
14     if ( number_of_nodes < 100 ) {
15         number_of_nodes = number_of_nodes + 10;
16     }
17
18     System.out.println( "number of nodes = " + number_of_nodes );
19
20     shawn.runCommand("rect_world", "width=25 height=25 " + " count=" + number_of_nodes );
21
22     shawn.runCommand( "simulation", "max_iterations=15" );
23
24     shawn.runCommand( "connectivity" );
25     shawn.runCommand( "dump_transmission_stats" );
26 }
```

Outline

- 1 A short overview: Shawn Simulation Environment and Models
Simulation Environment
Models
- 2 Step by Step Simulation
Simulation Parameters
Step by Step Simulation Configuration
- 3 Config files
- 4 JShawn
- 5 Converting a JShawn configuration file into a .conf file
- 6 A more complex example
- 7 Exercises

Exercises

- 1 Write your own configuration-files (using .conf-files and JShawn) for the application ws08.
- 2 Add and change parameters and have a look at the effects of simulations with different Shawn configurations.