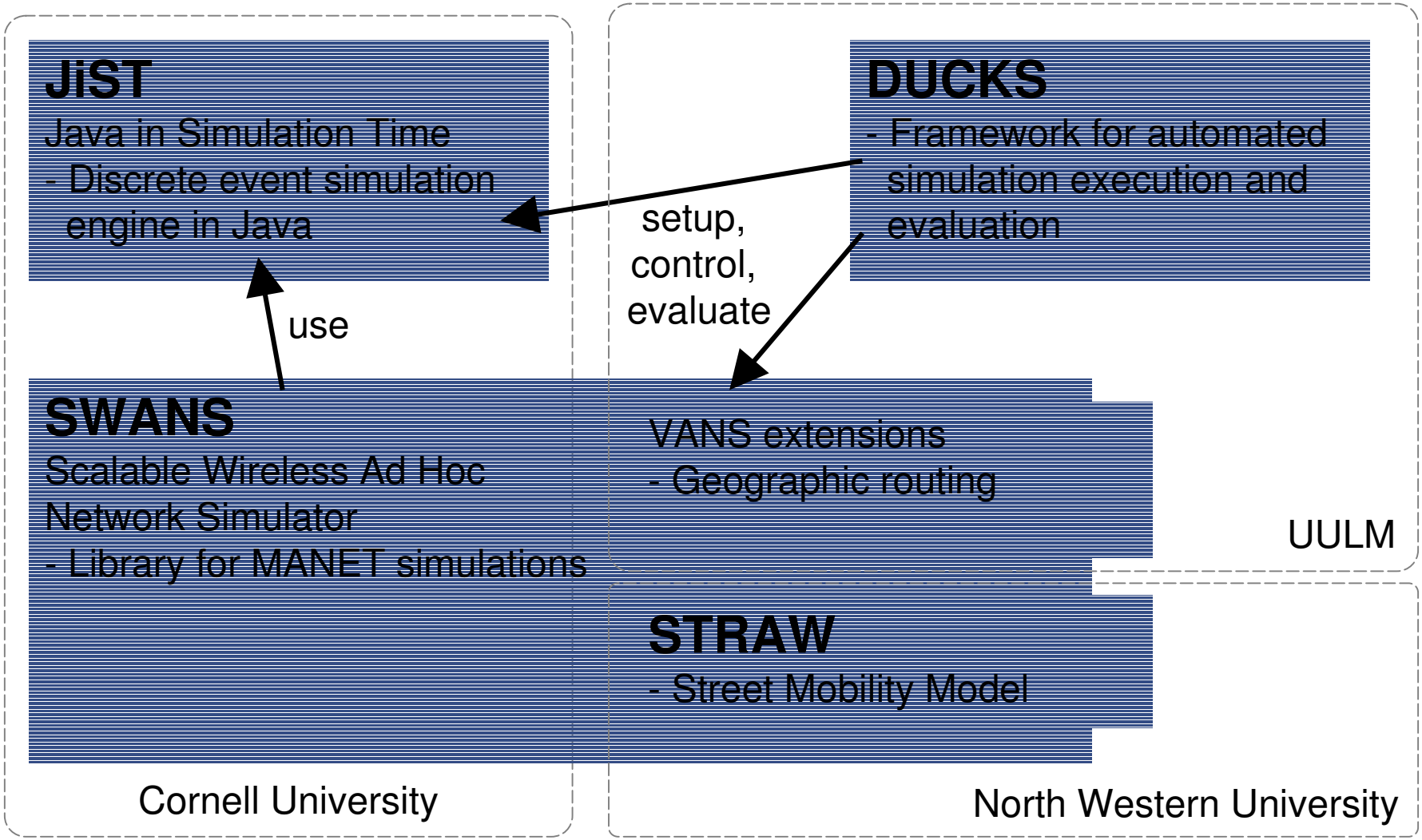
ulm university universität ulm

VANET Simulations with JiST/ SWANS

SEVECOM Kick-off Workshop

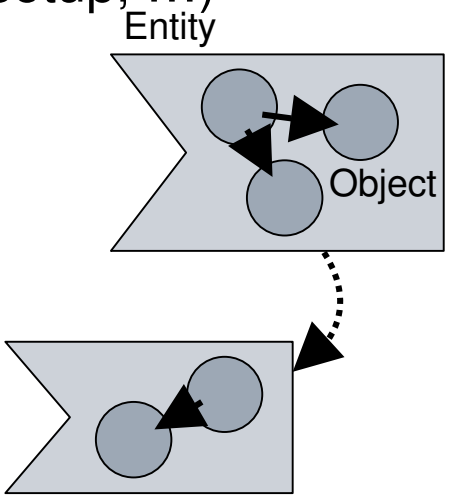
Elmar Schoch • elmar.schoch@uni-ulm.de

Overview


JiST
Java in Simulation Time
- Discrete event simulation engine in Java
DUCKS
- Framework for automated simulation execution and evaluation
setup, control, evaluate
use
SWANS
Scalable Wireless Ad Hoc Network Simulator
- Library for MANET simulations
VANS extensions
- Geographic routing
UULM
STRAW
- Street Mobility Model
Cornell University
North Western University

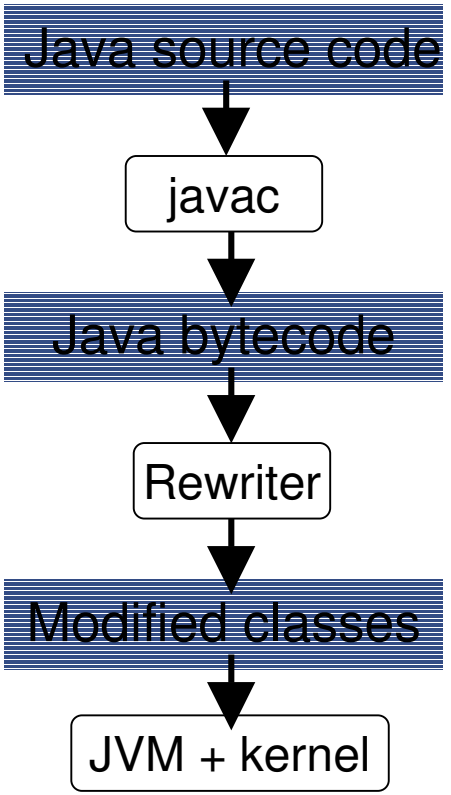
JiST – Simulation Kernel

- Basic idea: Convert virtual machine into simulation platform
  - Introduce virtual time
  - Make use of modern language concepts
- Base: Java and JVM
  - All components are pure Java
    (Rewriter, simulation kernel, library, simulation setup, …)
  - Reuse Java: reflection, interfaces, libraries, …
- Kernel
  - Strict partitioning of a simulation into entities
  - Method invocations on objects marked as entities represent simulation events
  - No explicit event queue, but virtual, explicit time progress



JiST – System architecture

- Java source files are compiled with regular Java compiler
- Running JiST invokes Rewriter
  - Rewriter modifies Java bytecode to introduce simulation time semantics
- JiST invokes simulation program
  - Rewritten program interacts with simu kernel
  - Virtual time progress independent of program progress (instructions take zero virtual time)
  - Time is advanced explicitly via `JistAPI.sleep()`
  - Time synchronization between Entities on method invocation (each Entity runs at own simulation time)
- Classes may be used without underlying JiST Kernel



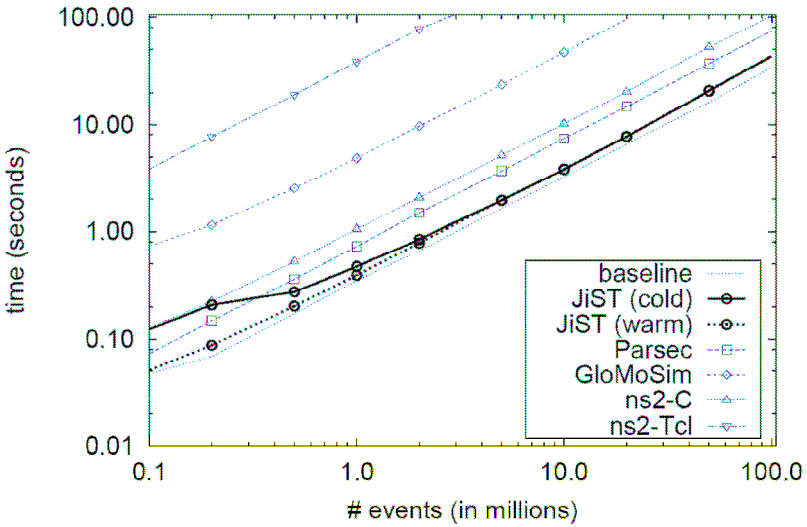
JiST example

```
import jist.runtime.JistAPI;

class Hello implements JistAPI.Entity {

   public static void main(String[] args) {
      System.out.println("Simulation start");
      Hello h = new Hello();
      h.doSequence(3);
   }

   public void doSequence(int count) {
      while( count > 0 ) {
         JistAPI.sleep(1);
         System.out.println("Hello t="+JistAPI.getTime());
         count--;
      }
   }
}
```

```
# java jist.runtime.Main Hello
> Simulation start
> Hello t=1
> Hello t=2
> Hello t=3
```

JiST – Performance

- Event troughput
  - ~ three times faster than ns2-C
  - ns2-Tcl shows extreme performance degradation
  - JiST shows kink in the first simulation second due to JIT compiler



5×10^6 events	time (sec)	vs. baseline	vs. JiST
baseline	1.640	1.0x	0.8x
JiST	**1.957**	**1.2x**	**1.0x**
Parsec	3.705	2.3x	1.9x
ns2-C	5.151	3.1x	2.6x
GloMoSim	23.720	14.5x	12.1x
ns2-Tcl	160.514	97.9x	82.0x

- Memory footprint

memory	entity	event	10K nodes sim.
JiST	**36 B**	**36 B**	**21 MB**
GloMoSim	36 B	64 B	35 MB
ns2	544 B	36 B*	72 MB*
Parsec	28536 B	64 B	2885 MB

Source: JiST User Guide

SWANS

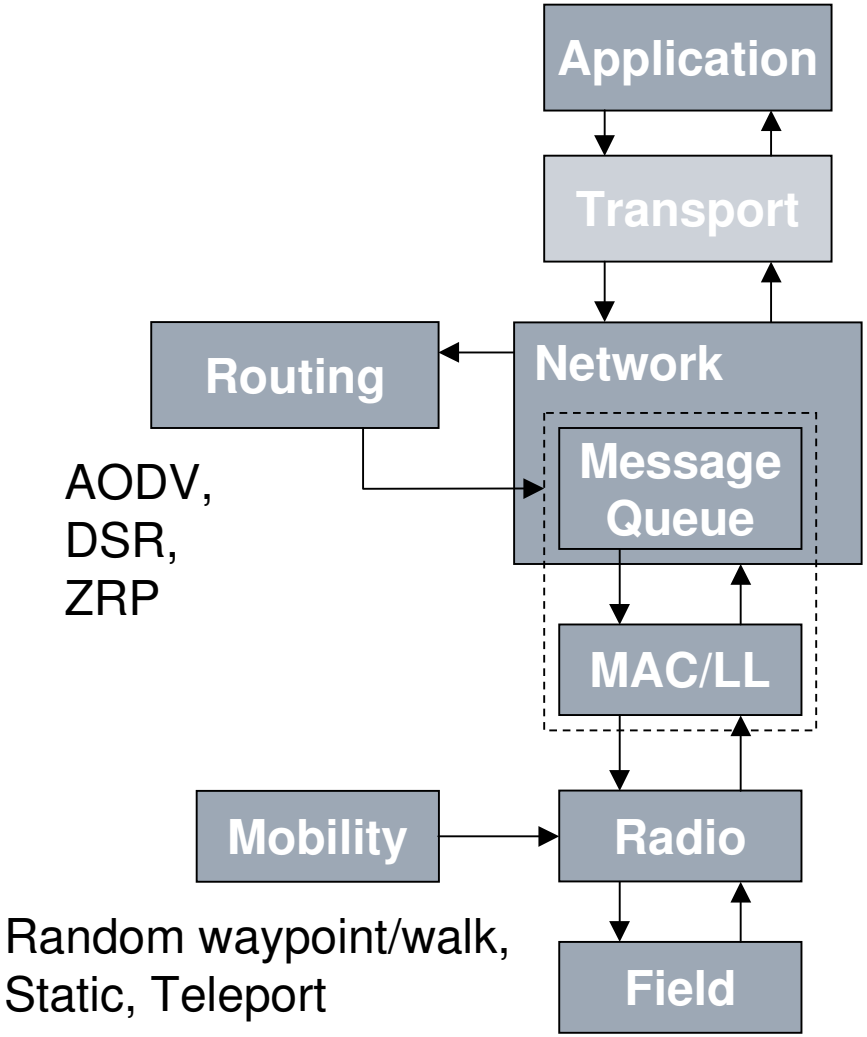
- Library for MANET simulations
- SWANS is an application of JiST
- Special properties:
  - Promises to handle huge node numbers with reasonable time/memory requirements
    Example: Neighbor Discovery Protocol, 15 minutes

t=15m	ns2		GloMoSim		SWANS		SWANS-hier	
nodes	time	memory	time	memory	time	memory	time	memory
500	7136.3 s	58761 KB	81.6 s	5759 KB	53.5 s	700 KB	*43.1 s*	1101 KB
5000			6191.4 s	27570 KB	3249.6 s	4887 KB	*433.0 s*	5284 KB
50000						47717 KB	*4377.0 s*	49262 KB

Source: http://jist.ece.cornell.edu/docs/031112-ece2.pdf

  - Efficient signal propagation by hierarchical binning
  - Allows running standard Java network apps over simulated networks

SWANS overview



CBR, heartbeat, legacy Java networking

UDP, TCP

IPv4

IEEE 802.11b, Naive MAC

Indep. & additive noise

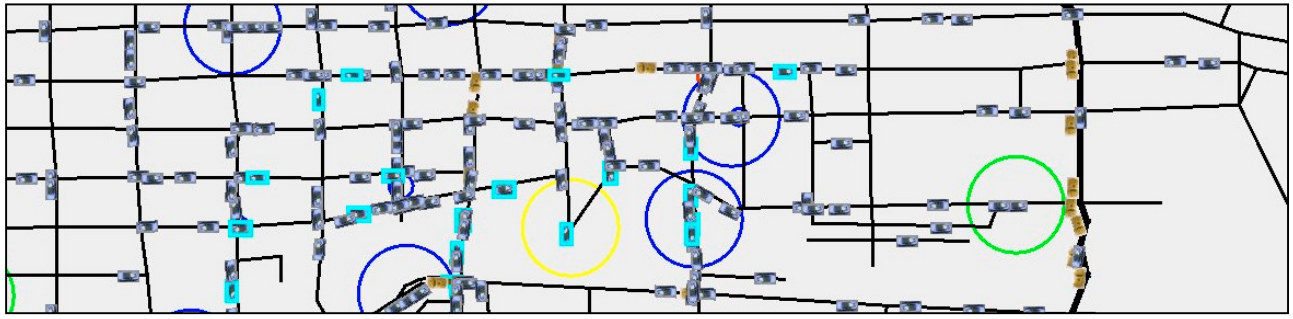
Zero, Rayleigh or Rician fading, Free-space or Two-Ray path-loss

VANET simulations

- Node mobility model — STRAW
  - Vehicle movements
    - High velocities
    - Quasi one-dimensional movements on highways
    - Short encounters of oncoming traffic
  - Vehicle Behavior
- Radio/Medium Access
  - Decentralized medium access, bandwidth allocation
  - Realistic radio propagation in urban environments
- Routing — Currently implemented by UULM
  - Position-based routing particularly suitable
- Applications
  - Extreme variety of application ideas
  - Other C2C projects?

STRAW Mobility Model

- By Northwestern University, Aqualab
  - http://www.aqualab.cs.northwestern.edu/projects/STRAW/index.php
- Written for JiST/SWANS
- Models vehicular node movements on streets
  - Structures: segments, ramps, intersections
  - Movements: acceleration, deceleration, …
- Uses TIGER® street maps
  - By U.S. Census Bureau



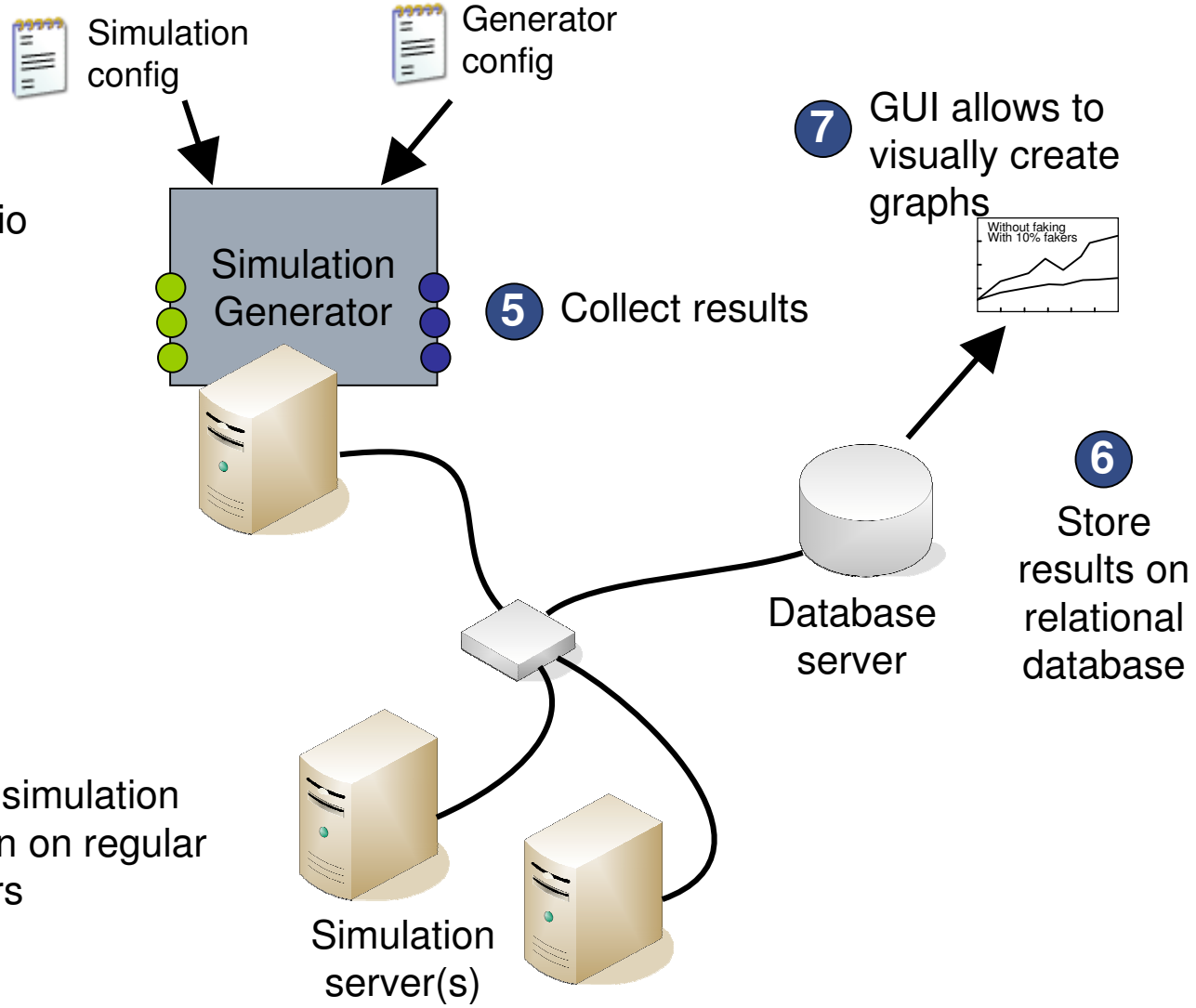
DUCKS simulation framework

- Problem:
  - JiST/SWANS lacks tools for handling simulation parameters and output
- Solutions by DUCKS:
  - Easily define complete simulation scenarios by config files (e.g. various simulation setup parameters like field size, node number, ...)
  - Automated execution of simulations
  - Easy distribution of simulation on multiple servers
  - Structured statistics collection, storage and evaluation
- Implementation nearly finished
  (still fixing some issues and extending usability)

DUCKS architecture

ulm university universität ulm

1. Defines complete simulation scenario
2. Demultiplexes scenario to single simulations
3. Distributes jobs to simulation servers
4. Distributed simulation computation on regular JiST-servers
5. Collect results
6. Store results on relational database
7. GUI allows to visually create graphs



DUCKS config file example

```
ducks.config.runs=20

ducks.general.fieldsize.x=500,1000
ducks.general.fieldsize.y=(ducks.general.fieldsize.x <-> 500,1000)
ducks.general.nodes=50-200/50,500

ducks.general.duration=120
ducks.general.waittime.start=10
ducks.general.waittime.end=(-> ducks.general.waittime.start)

ducks.mobility.movement=waypoint
ducks.mobility.waypoint.speed.min=1
ducks.mobility.waypoint.speed.max=5,20
ducks.mobility.waypoint.pausetime=(ducks.mobility.waypoint.speed.max == 5 ? 0 : 10)
ducks.mobility.waypoint.precision=100

ducks.traffic.type=cbr
ducks.traffic.cbr.rate=20
ducks.traffic.cbr.packetspercon=1
ducks.traffic.cbr.waittime=0

ducks.routing.protocol=aodv

ducks.mac.protocol=802.11
```

JiST/SWANS Pros & Cons

ulm university universität ulm

Pros

- Fast & scalable
- Completely Java-based approach
  - Advantages of Java (Garbage collection, type-safety, reflection, library, …)
  - No other language needed
  - Portability of JVM
- Interesting virtual time concept
- Usage of legacy, socket-based Java applications is possible

Cons

- Maturity unproven
  - Still little attention in research community
  - Correctness of implementation
  - Minor bugs/deficiencies
  - Issues regarding platform independence
- Sparse tool support
  - No GUI modeling/output
  - No framework for automated simulation execution
    - leveraged by DUCKS

Further work

- Ongoing activities
  - Implementation of geographic routing
  - Implementation of positioning and position verification
  - DUCKS consolidation
  - Performance tests
- Planned activities
  - More realistic signal propagation models
    e.g. including obstacles like buildings
  - Abstraction layer to be able to switch between
    simulator and real hardware
- Scientific plans
  - Qualitative comparison of SWANS and ns-2
    (e.g. regarding delivery ratio, delay, …)
  - Use for SEVECOM protocol validation