

A Candidate Platform for Simulation of Cognitive Radio Protocols.

Adam Wolisz, Daniel Willkomm (TUB/TKN)

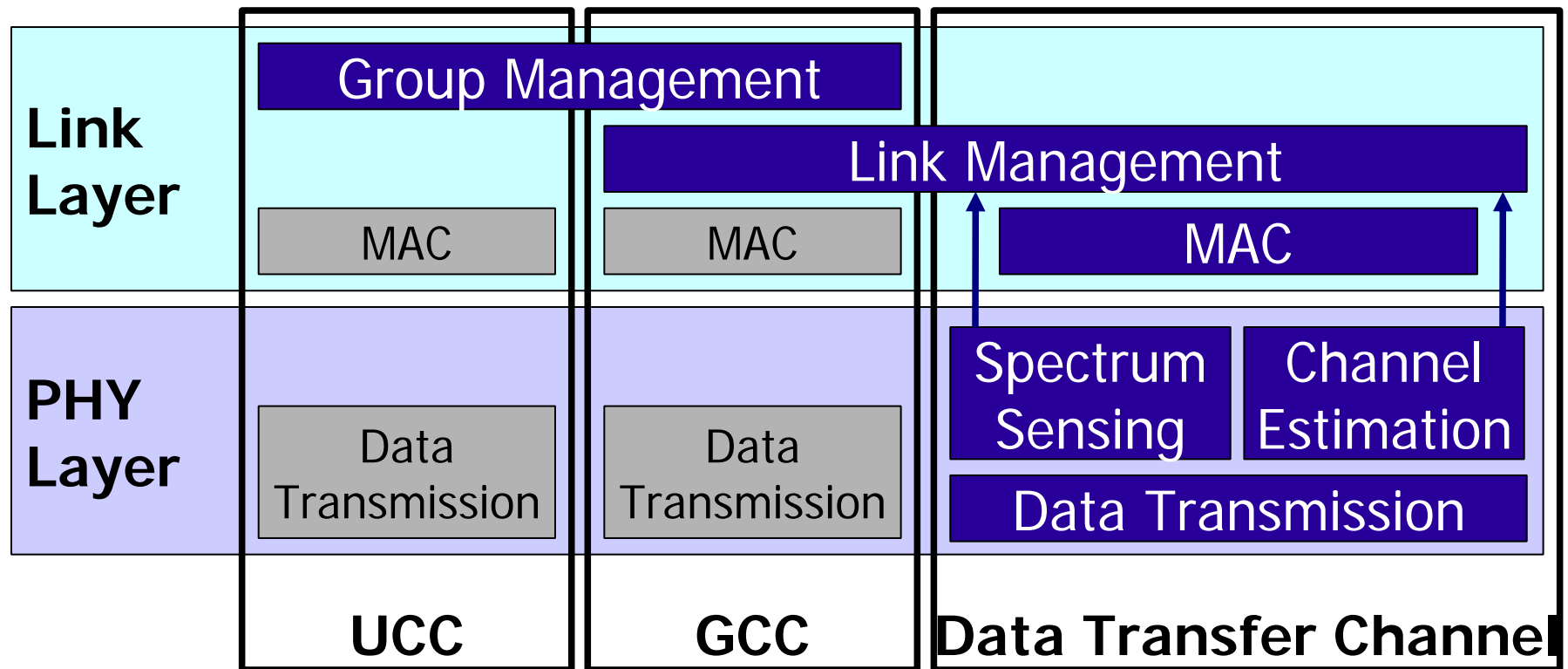


TKN Telecommunication
Networks Group

<http://www.tkn.tu-berlin.de>

CR Communication Stack *(from the white paper)*

- Only physical and link layer are CR specific
- Higher layers are initially not considered



UCC = Universal Control Channel
GCC = Group Control Channel

Protocol design aspects

Supported by discrete event simulation

- Functional correctness (by step-by-step analysis of traces):
 - Support of the desired functionality under border conditions.
- Vulnerability against identified (or assumed):
errors, attacks, malicious code modification.
- Performance evaluation (including scaling issues)

Not supported by simulation (but important!):

- Implementation validation and testing
- Implementation timing

Candidate Tools:

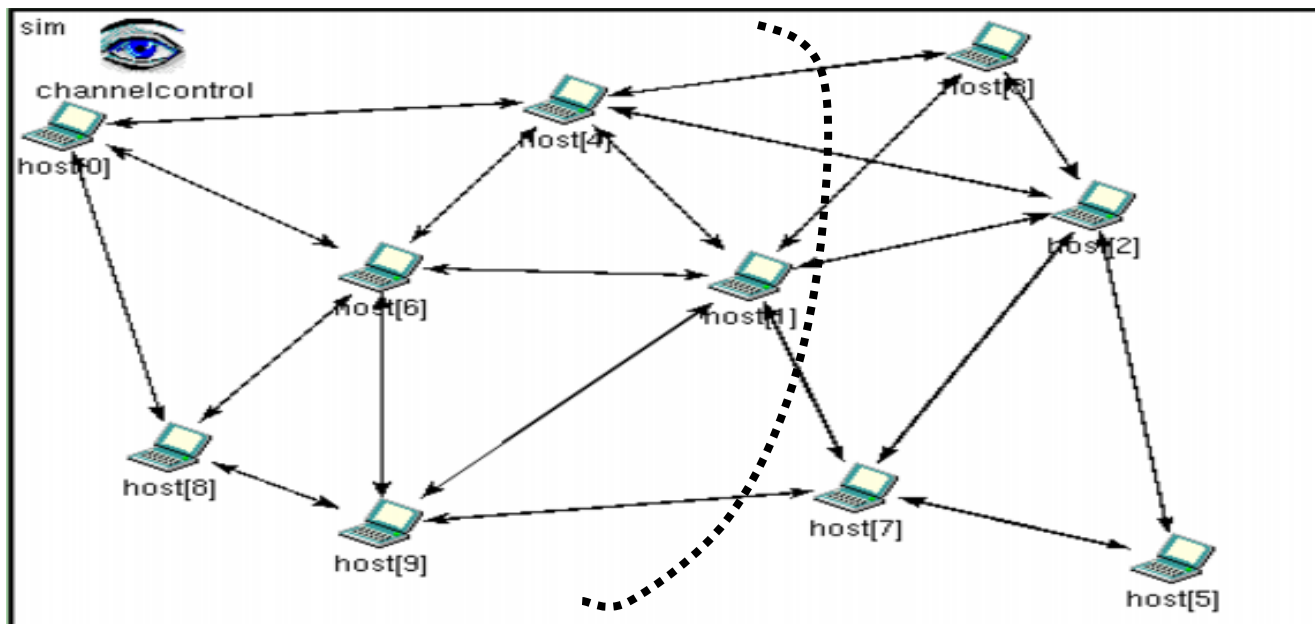
- Requirement: Open source
 - As condition for scientific assessment of the model
 - For sharing models
- What remains?
 - ns-2
 - OMNeT++
 - ???

Modeling concepts of OMNeT++

- Object Oriented Discrete Event Simulator
- Hierarchically nested modules with
 - Modules (functionality defined in C++)
 - In/Out gates, connections
 - Topology Description Language (NED) for configuration definition:
Alphanumeric but supported by a graphical representation
- Enhancement for Communication definition:
 - Channel modules
 - Modeling delays, bit errors, throughput
- GUI + inspector windows: for step-by-step analysis
- Excellent documentation
- Good performance of execution

TKN mobility framework.

- HOSTs: Moving, Communicating and/or interfering.
 - Communication subsystem, blackboard, mobility module
- Channel Control: For finding interference candidates (reducing the N^2 interference problem)



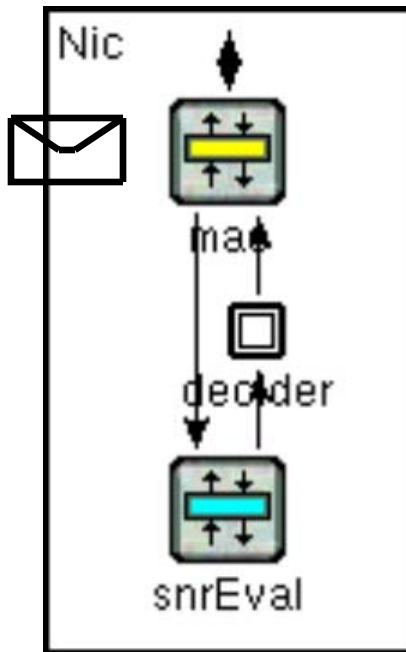
Blackboard & Mobility

- Blackboard
 - Every module can publish information
 - Every module can explicitly read information
 - Every module can subscribe to information
 - i.e. it gets informed every time the information it subscribed for has changed
- Mobility module
 - Keeps track of current position and publishes it on the blackboard.
 - Moving: Constantly recomputing a new position (according to certain rules) and re-publishing it.
 - Informs Channel Control about position changes.

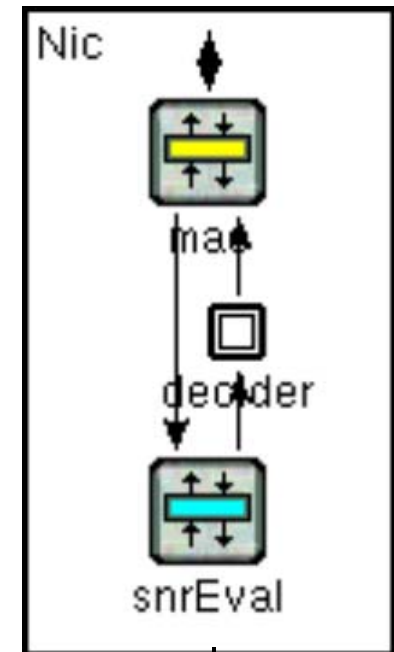
Communication Subsystem Model (NIC)

- Transmitter
 - Add sending power, modulation and position
 - Send message
- Receiver
 - Start receiving: calc. initial SNR & delay
 - Calc. additional SNR every time interference situation changes (new messages arrive), incl. fading models!
 - End receiving: pass message + calculated SNR information to decider
 - Decide whether a message is lost, contains bit errors, etc. based on the SNR information

Transmitter



Receiver



Platform for Cognitive Radio Protocol Simulation

- Goals:
 - Generally accepted open platform (like ns-2 for TCP ☺)
 - Library of modules (reusage, exchange)
 - Comparison of solutions.
- OMNeT++ with TKN Mobility Framework so far used for
 - wireless sensor networks
 - ... ad hoc routing protocols
 - ... 802.11 enhancementsseems to be able to support most of these goals
- Further extensions needed (among others):
 - Support of multiple channels
 - CR specific modules (e.g. licensed user interference models)

Ongoing/Planned Investigations at TKN

Internet Access using centralized approach:

Secondary user joins an **access group**, using a single **access point** (AP) for getting internet connectivity

- Group Management
 - Access point selection based on preferences/negotiations
 - Group creation/joining
 - Parameter exchange

- Link Management
 - Link setup (sub-channel negotiations)
 - Link maintenance (sub-channel update)

Some already investigated aspects...

- Secondary user might NEED a different number of Sub-Channels dependent on the expected QoS, quality of the channel, etc.
- A “reappearing” primary user X “hits” only a small part of the secondary user bandwidth (preferably one Sub-Channel)
- In order to keep continuous QOS in spite of primary user recurrence, secondary user should have a redundant amount of Sub-Channels; proper coding is required

Web-sites

- OMNeT++ and Mobility Framework are available at
 - OMNeT++
 - www.omnetpp.org
 - Mobility Framework
 - <http://mobility-fw.sourceforge.net>
- Both websites include manuals and API references

Thank You !



Questions?

Ask now!

Alternatively :

Talk to me after this talk
or drop me an
e-mail: awo@ieee.org

Example: OMNeT++ Simulator (I)

Network

Node A



Node B



```
simple Node
  gates:
    in: inPort;
    out: outPort;
endsimple
```

```
module Network
  submodules:
    nodeA: Node;
    nodeB: Node;
  connections:
    nodeA.outPort --> nodeB.inPort;
    nodeA.inPort <-- nodeB.outPort;
endmodule
```

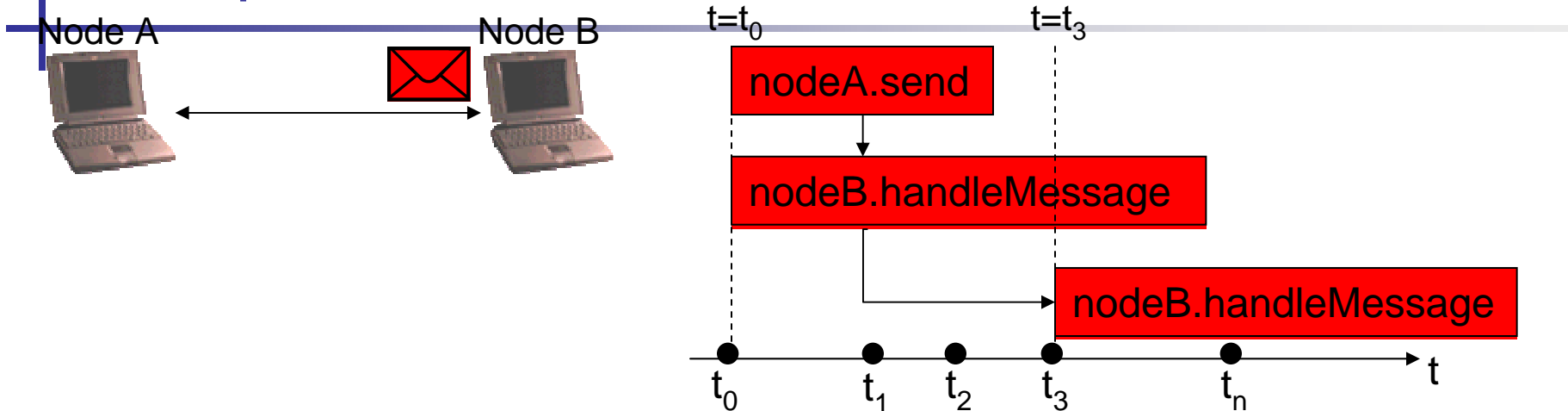
Functions:

`send(msg, gate)`: send 'msg' on specified 'gate'

`handleMessage(msg)`: called every time a message arrives

`scheduleAt(time, msg)`: (re)schedule 'msg' to the specified 'time'

Example: OMNeT++ Simulator (II)

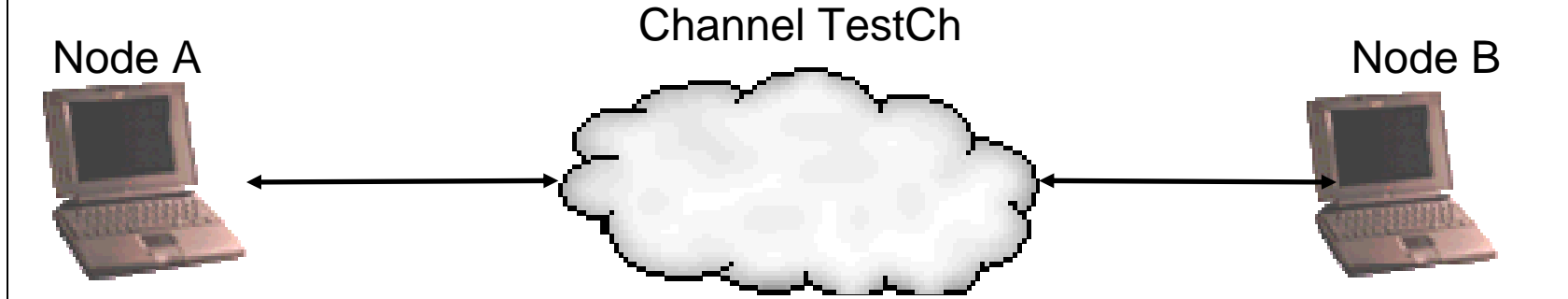


```
//set name and position of the
//node in the message
msg.setName('testMsg');
msg.setPos(myPos);
//send msg to Node B
send(msg, outPort);
```

```
handleMessage(msg) {
  if (msg.getName() == 'testMsg') {
    //calculate delay
    delay=calcDelay(msg.getPos(), myPos);
    //reschedule msg to 'now+delay'
    msg.setName('delayedMsg');
    scheduleAt(now+delay, msg);
  }
  else if (msg.getName() == 'delayedMsg') {
    //process message
  }
}
```

Example: OMNeT++ Simulator (III)

Network

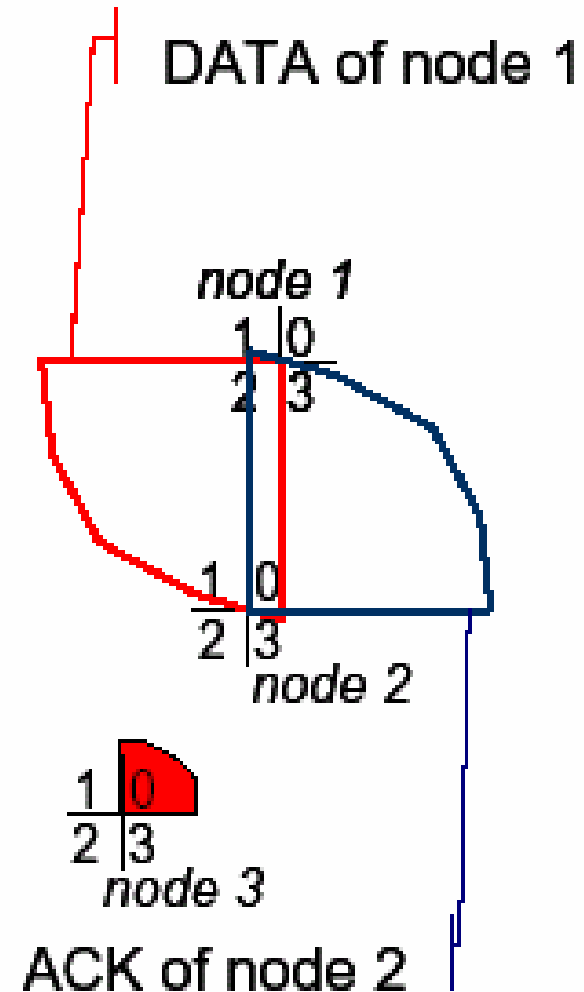


```
channel TestCh
  delay 0.0015
  error 0.000001
  datarate 10000000
endchannel
```

```
module Network
  submodules:
    nodeA: Node;
    nodeB: Node;
  connections:
    nodeA.outPort --> TestCh --> nodeB.inPort;
    nodeA.inPort <-- TestCh <-- nodeB.outPort;
endmodule
```

Extensions: 802.11 + directed antennas

- Modified FW-ChannelControl:
 - Interference range
 - Gate-List for antennas
 - only 1 connection between 2 nodes
 - but separate antenna control
- Channel-dependent frame, 1 SNR list at Physical Layer
- MAC sub layer controls:
 - Every antenna
 - State changes
 - ⇔ and needs indications about state changes



Performance of execution

- Coroutines \Leftrightarrow messages
- Speed of simulations mainly depends on:
 - Node density
 - Number of messages
 - Level of detail
 - Snr computation for every symbol \Leftrightarrow once / packet
 - Bit \Leftrightarrow packet errors
- According to Omnet++-Manual:
 - Important factor: programming language
 - "Omnet++ simulation was only 1.3 slower than its counterpart implemented in plain C"