Simulative Leistungsbewertung des IEEE 802.16 Systems im *open*WNS

19. ComNets-Workshop Mobil- und Telekommunikation

Dipl.-Ing. Karsten Klagges

ComNets Research Group
RWTH Aachen University

11. März 2011

Outline

Simulator Platform
- Simulator Architecture
- Functional Units

Continuous Integration
- Source Control System
- Grid/Cluster Computing
- Overview

Wrowser

WiMAC module
- IEEE 802.16 Protocol Overview
- WiMAC Component

Simulator Platform

***open*WNS** is witten in C++ and provides a rich set of common software design patterns

- Design patterns from Gamma et. al.
- Modern Software Design A. Alexandrescu

***open*WNS** is flexible

- Functional units allow flexible configuration of protocol layer
- Modular composition of protocol stack
- Simulation model is exchangeable → even non-event driven simulations are possible

***open*WNS** performs data post processing

- On-line data concentration (Histogram, PDF, CDF)
- Sorting of data by context
- Statistical data is accessible in database

Simulator Platform

***open*WNS** is witten in C++ and provides a rich set of common software design patterns

- Design patterns from Gamma et. al.
- Modern Software Design A. Alexandrescu

***open*WNS** is flexible

- Functional units allow flexible configuration of protocol layer
- Modular composition of protocol stack
- Simulation model is exchangeable $\rightarrow$ even non-event driven simulations are possible

***open*WNS** performs data post processing

- On-line data concentration (Histogram, PDF, CDF)
- Sorting of data by context
- Statistical data is accessible in database

Simulator Platform

***open*WNS** is witten in C++ and provides a rich set of common software design patterns

- Design patterns from Gamma et. al.
- Modern Software Design A. Alexandrescu

***open*WNS** is flexible

- Functional units allow flexible configuration of protocol layer
- Modular composition of protocol stack
- Simulation model is exchangeable $\rightarrow$ even non-event driven simulations are possible

***open*WNS** performs data post processing

- On-line data concentration (Histogram, PDF, CDF)
- Sorting of data by context
- Statistical data is accessible in database

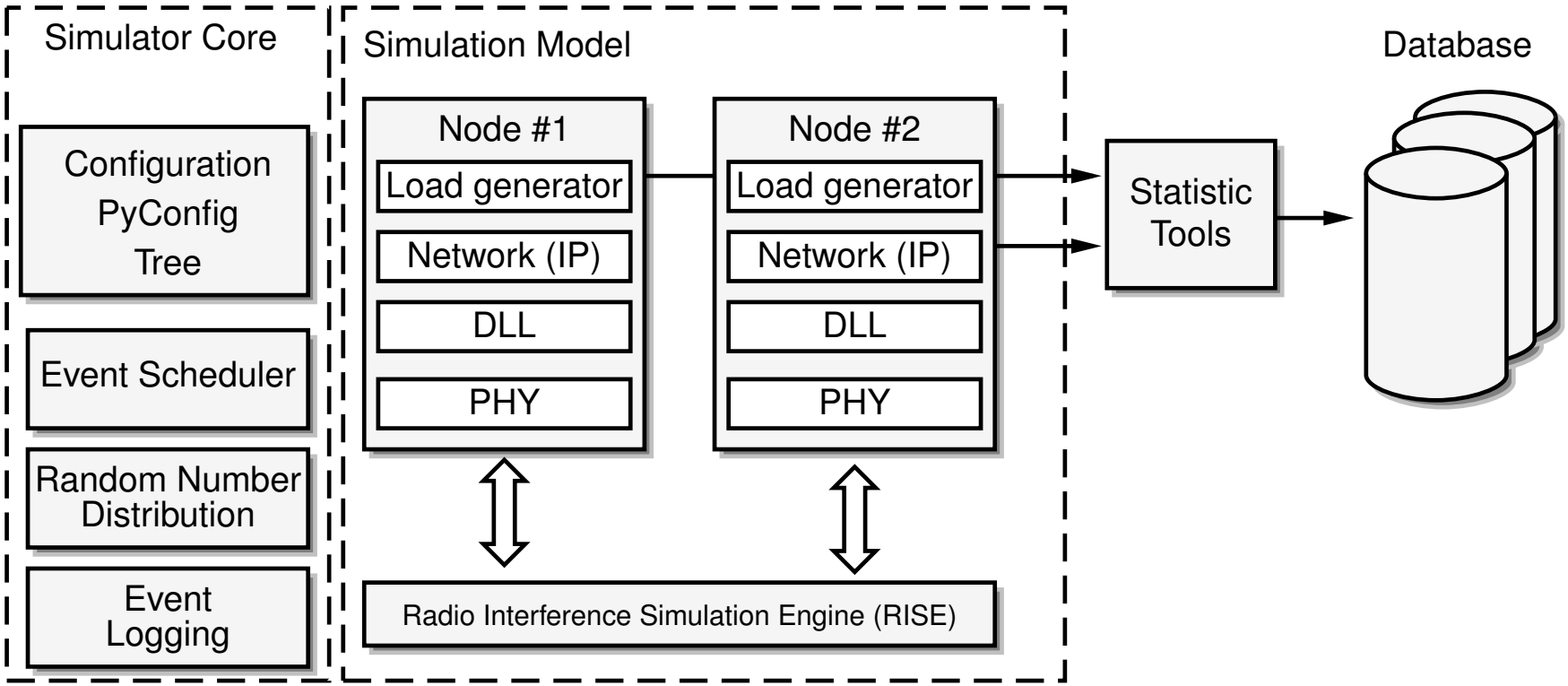


Figure: Simulator Architecture

Functional Units

Functional units have been invented [1] to provide a flexible configuration of protocol layers by reusing atomic protocol functionality in multiple layers.

- Definition of common functional unit interface to connect functional units
- Data is aggregated in compounds of functional unit commands
- Inter functional unit flow control

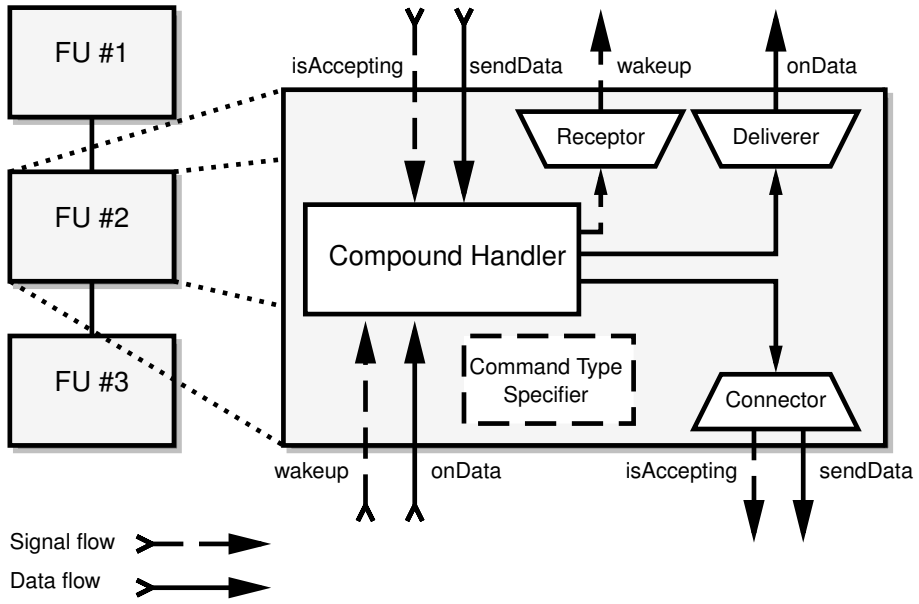


Figure: Functional Unit Aspects

[1] Schinnenburg et. al., A Framework for Reconfigurable Functions of a Multi-Mode Protocol Layer, SDR Forum 2005

Outgoing direction

1. Incoming wakeup from lower FU
2. Forward wakeup to next higher FU
3. isAccepting request from higher FU
4. Forward isAccepting request to next lower FU
5. sendData request from higher FU
6. Handle compound and forward to next lower FU

Incoming Direction

1. onData call from lower FU
2. Handle data and send to next higher FU

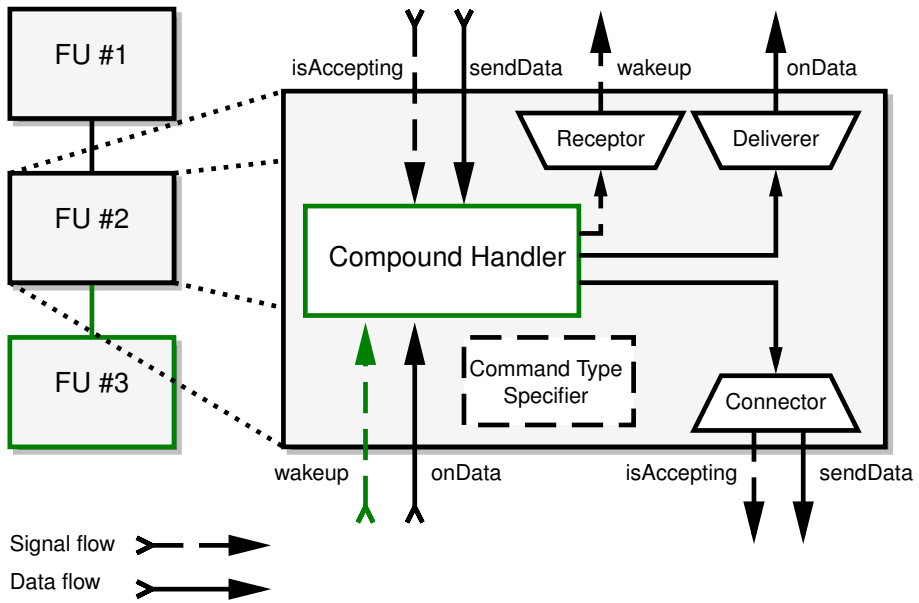


Figure: Functional Unit Flow Control

Outgoing direction

1. Incoming wakeup from lower FU
2. Forward wakeup to next higher FU
3. isAccepting request from higher FU
4. Forward isAccepting request to next lower FU
5. sendData request from higher FU
6. Handle compound and forward to next lower FU

Incoming Direction

1. onData call from lower FU
2. Handle data and send to next higher FU

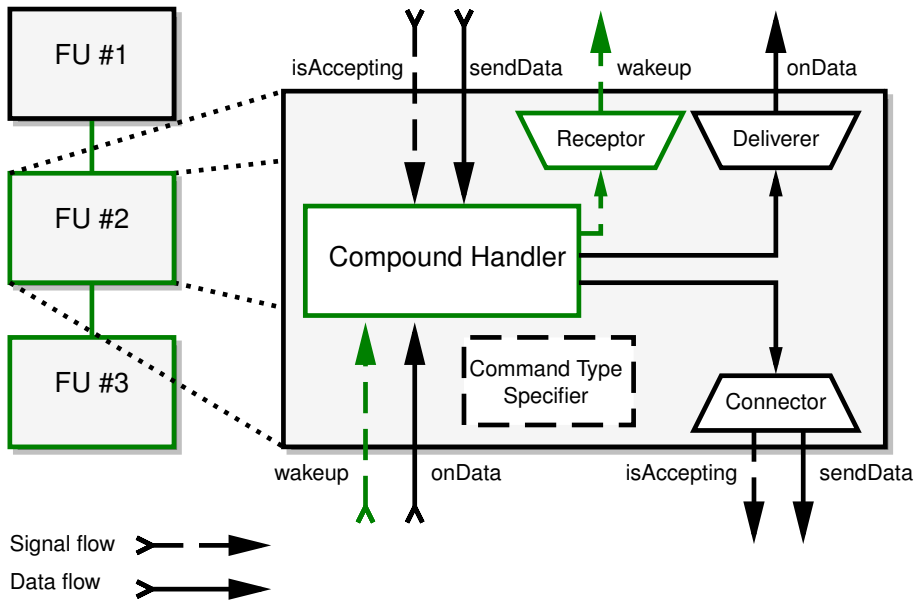


Figure: Functional Unit Flow Control

Outgoing direction

1. Incoming wakeup from lower FU
2. Forward wakeup to next higher FU
3. isAccepting request from higher FU
4. Forward isAccepting request to next lower FU
5. sendData request from higher FU
6. Handle compound and forward to next lower FU

Incoming Direction

1. onData call from lower FU
2. Handle data and send to next higher FU

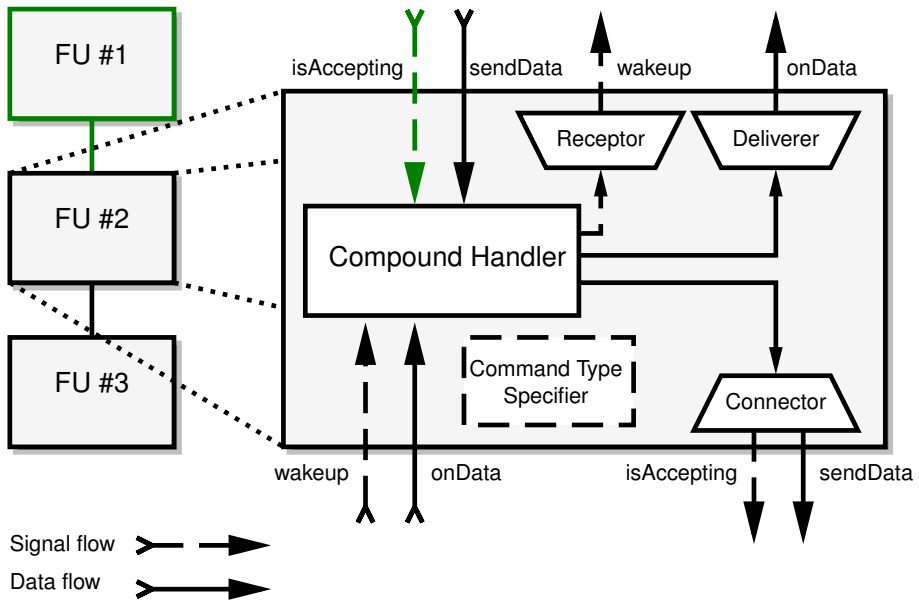


Figure: Functional Unit Flow Control

Outgoing direction

1. Incoming wakeup from lower FU
2. Forward wakeup to next higher FU
3. isAccepting request from higher FU
4. Forward isAccepting request to next lower FU
5. sendData request from higher FU
6. Handle compound and forward to next lower FU

Incoming Direction

1. onData call from lower FU
2. Handle data and send to next higher FU

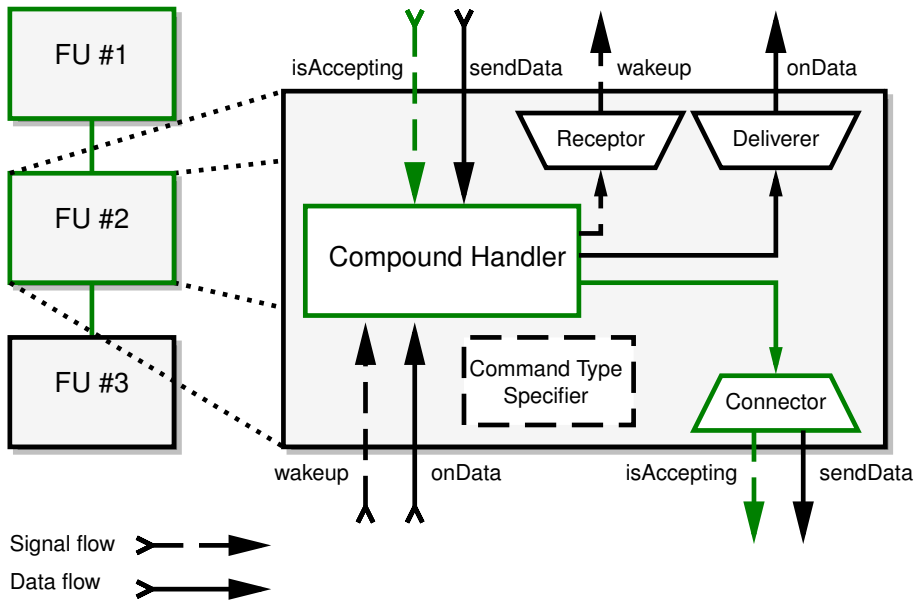


Figure: Functional Unit Flow Control

Outgoing direction

1. Incoming wakeup from lower FU
2. Forward wakeup to next higher FU
3. isAccepting request from higher FU
4. Forward isAccepting request to next lower FU
5. sendData request from higher FU
6. Handle compound and forward to next lower FU

Incoming Direction

1. onData call from lower FU
2. Handle data and send to next higher FU

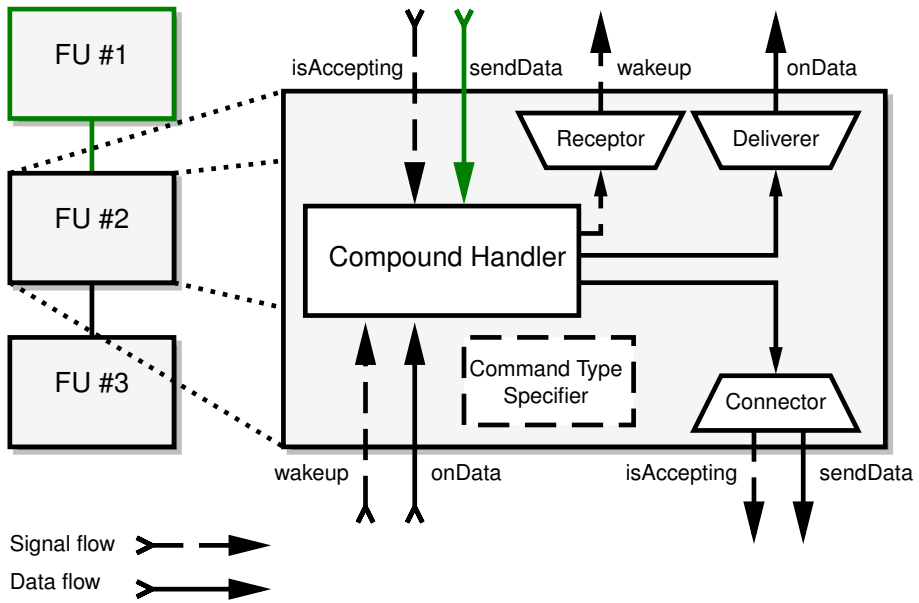


Figure: Functional Unit Flow Control

Outgoing direction

1. Incoming wakeup from lower FU
2. Forward wakeup to next higher FU
3. isAccepting request from higher FU
4. Forward isAccepting request to next lower FU
5. sendData request from higher FU
6. Handle compound and forward to next lower FU

Incoming Direction

1. onData call from lower FU
2. Handle data and send to next higher FU

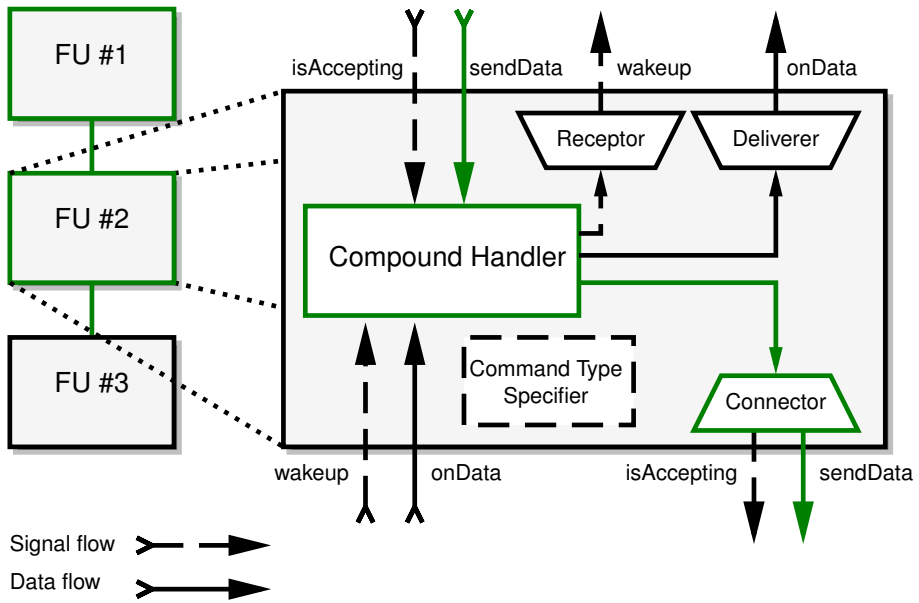


Figure: Functional Unit Flow Control

Outgoing direction

1. Incoming wakeup from lower FU
2. Forward wakeup to next higher FU
3. isAccepting request from higher FU
4. Forward isAccepting request to next lower FU
5. sendData request from higher FU
6. Handle compound and forward to next lower FU

Incoming Direction

1. onData call from lower FU
2. Handle data and send to next higher FU

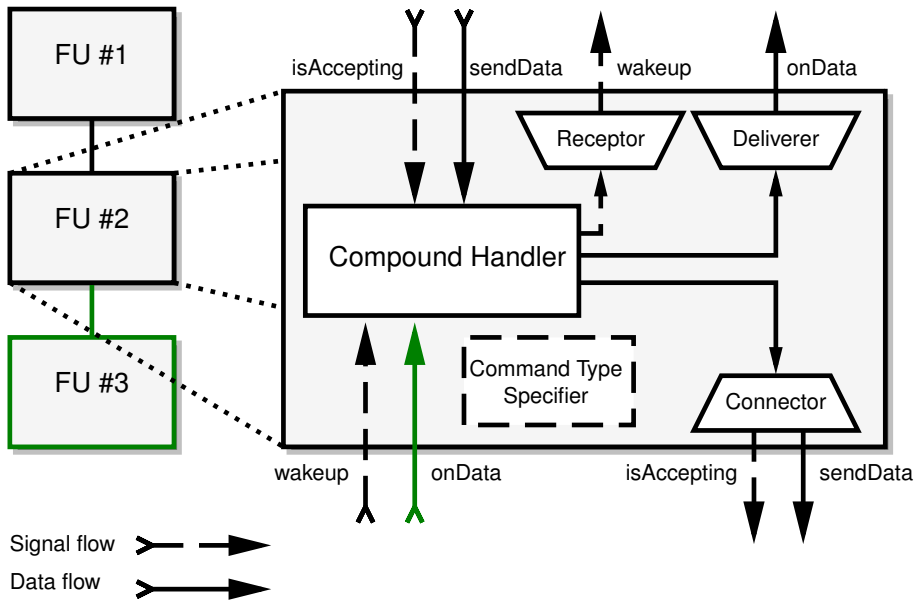


Figure: Functional Unit Flow Control

Outgoing direction

1. Incoming wakeup from lower FU
2. Forward wakeup to next higher FU
3. isAccepting request from higher FU
4. Forward isAccepting request to next lower FU
5. sendData request from higher FU
6. Handle compound and forward to next lower FU

Incoming Direction

1. onData call from lower FU
2. Handle data and send to next higher FU

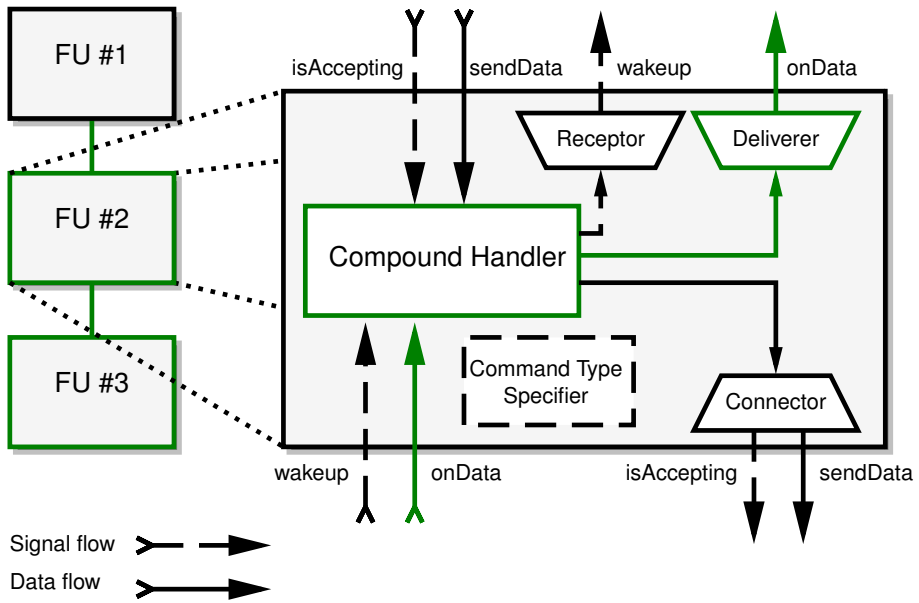


Figure: Functional Unit Flow Control

Simulator Platform
- Simulator Architecture
- Functional Units

Continuous Integration
- Source Control System
- Grid/Cluster Computing
- Overview

Wrowser

WiMAC module
- IEEE 802.16 Protocol Overview
- WiMAC Component

Continuous Integration with *buildbot*

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. *Martin Fowler*

- Development branch is always functional
- Unit tests on functional unit level with CppUnit testing suite
- System tests of full protocol stacks in selected scenarios
- Subsequent patches must not interfere with existing functionality

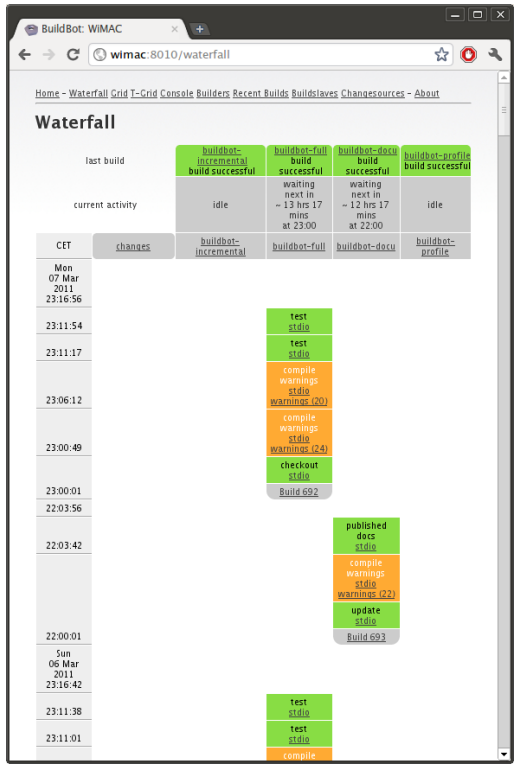
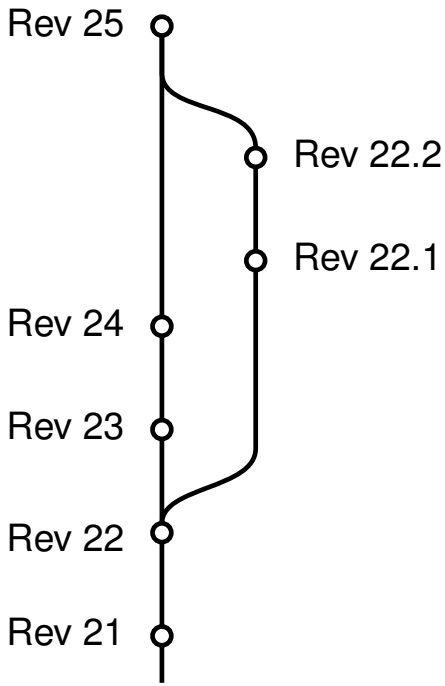


Figure: Buildbot web interface

Source Control System

Revision control provides both, collaboration of developers and archiving of achieved work.

- Based on Canonical's *bzr* also known as bazaar
- Everybody develops on the same codebase
- Small changes are developed on frequent commit base
- Big inventions are developed on separated branches
- Branches are reintegrated after successful testing



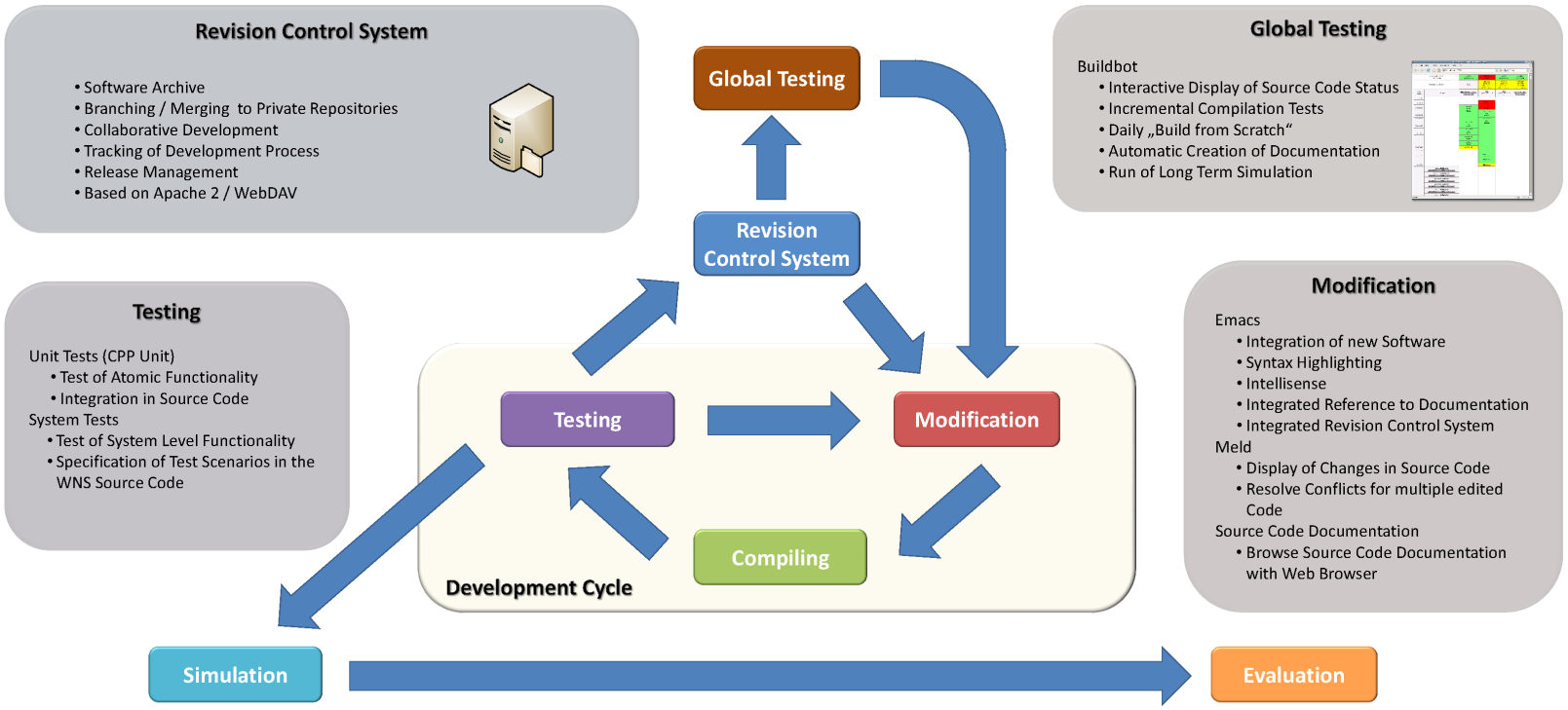
Grid/Cluster Computing

Grid computing

Grid computing combines distributed computer resources that are *loosely* coupled to compute programs in parallel.

- Simulation campaigns are simulated on the compute cluster, driven by the SUN gridengine
- Gridengine runs on Dell Power-Edge blades
- 16 blades with 2×Quad-Core Intel Xeon E5420 @ 2.5GHz
  → up to 128 simulations run in parallel
- Gridengine may also run on desktop computers

openWNS development cycle at a glance



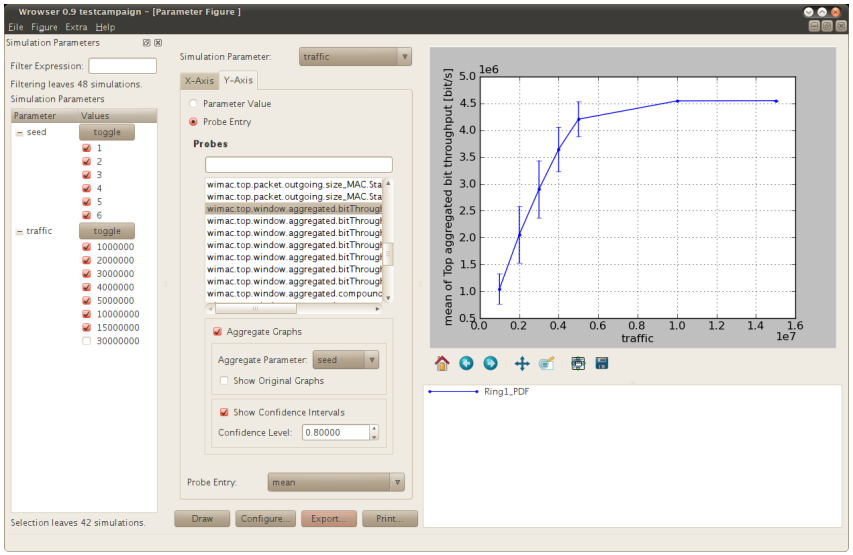


Figure: Wrowser data viewer

Simulation results are viewed with ***open*WNS** data browser: *Wrowser*

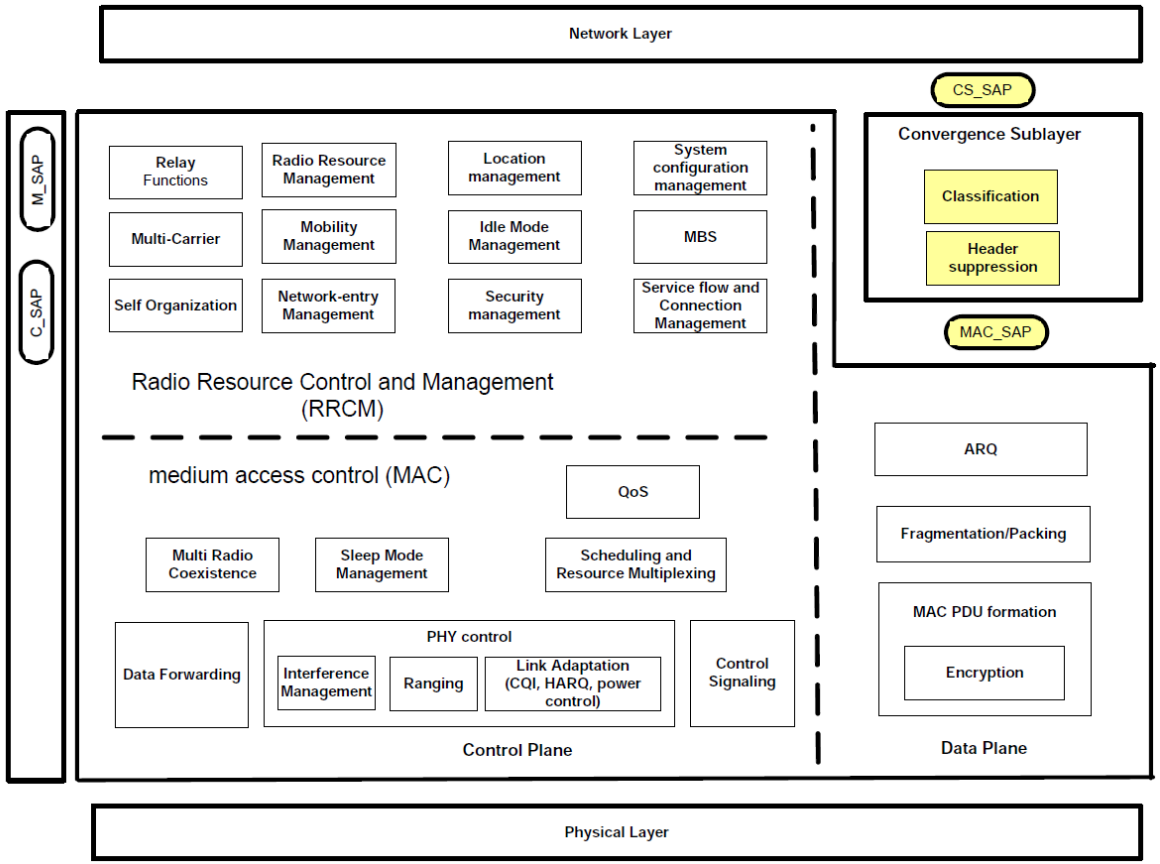
- Aggregation of simulation results of multiple runs
- Statistical confidence of results shown as confidence interval
- Export of data as png, pdf, emf, MATLAB data and many more

Simulator Platform
- Simulator Architecture
- Functional Units

Continuous Integration
- Source Control System
- Grid/Cluster Computing
- Overview

Wrowser

WiMAC module
- IEEE 802.16 Protocol Overview
- WiMAC Component



- Convergence Sublayer
- Common Part Sublayer
  - Radio Resource Control and Management (RRCM)
  - Medium Access Control (MAC)
  - Data Plane

WiMAC module

WiMAC (WIMAX MAC) is the IEEE 802.16 MAC and PHY model in the ***open*WNS**. WiMAC makes heavy use of the FU concept

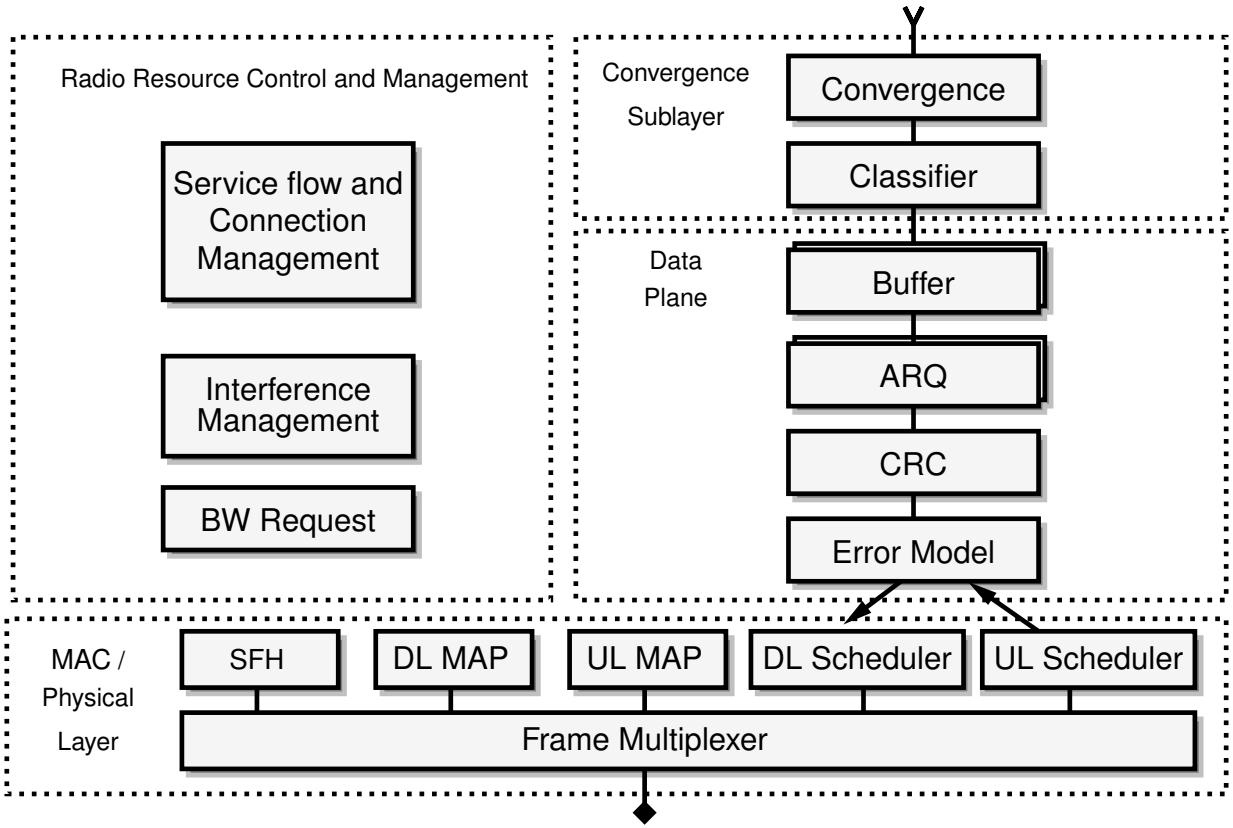


Figure: WiMAC Functional Unit Network

Thank you

kks@comnets.rwth-aachen.de