Master's Thesis Presentation

A Rule-Based Middleware Architecture for Wireless Sensor Networks

Georg Wittenburg, B.Sc.
Freie Universität Berlin

14. November 2005

Wireless Sensor Networks (WSN)

- Viele kostengünstige Sensorknoten bilden ein drahtloses ad-hoc Netzwerk.
- Einzelne Sensorknoten sind bestückt mit:
  - Micro-Controller und Speicher
  - Radio-Transceiver und Sensoren
- Anwendungsszenarien:
  - Gebäudesicherheit
  - Lebensraumüberwachung

Warum eine Middleware?

- Herausforderungen des WSN Konzeptes:
  - Knappe Resourcen (Speicher, Energie, ...)
  - Hardware-nahe Programmierung
  - Verteilte Algorithmen
- Eine Middleware kann
  - von Hardware und sogar einzelnen Sensorknoten abstrahieren,
  - zusätzliche Dienste anbieten,
  - und Programmier-Konzepte unterstützen.

Warum basierend auf Regeln?

- Energieersparnis durch ereignis-orientierte Programmierung:
  - Sensorknoten reagieren auf Ereignisse
  - Verbleiben ansonsten im Schlafzustand
- Verteilte Algorithmen arbeiten anhand von lokal vorhandenen Daten:
  - Verhaltensweisen der einzelnen Sensorknoten leiten sich von dem verfügbahren Wissen ab
  - Fokussierung auf Verarbeitung von Daten

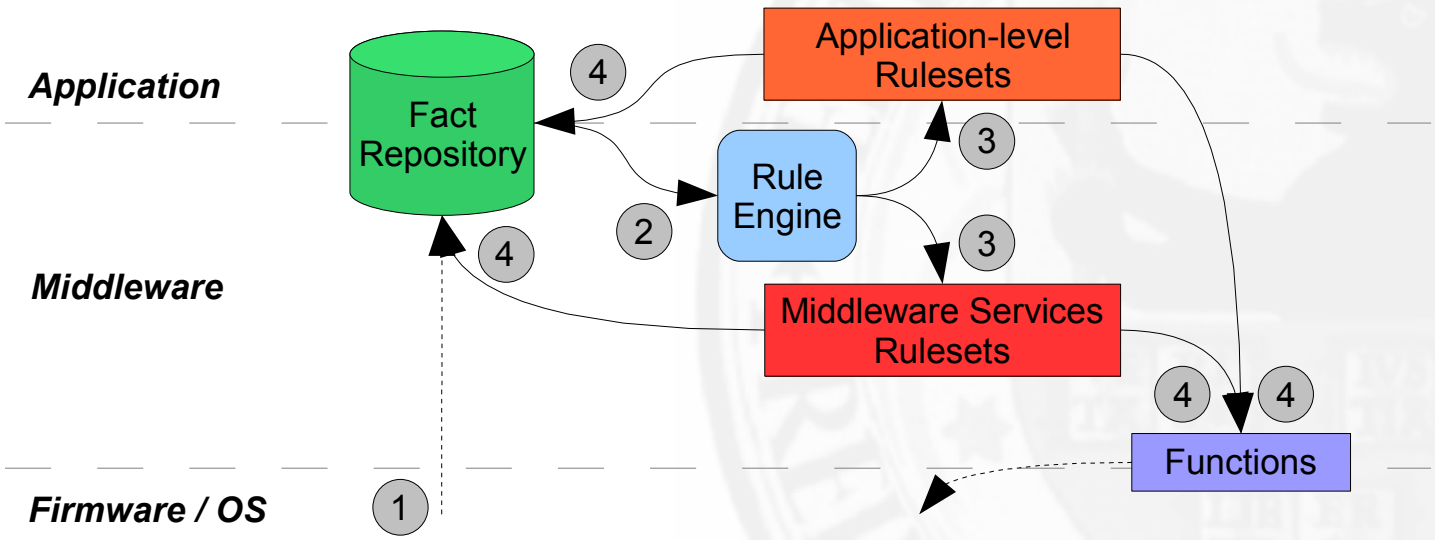
Warum basierend auf Regeln?

- *"In general, this novel feature of directed diffusion is achieved by data driven **local rules**."*
  - Estrin et al. Directed Diffusion. UCLA, 2000. [1]
- *"We will refer to [this step in the procedure of role assignment] as **local rule evaluation** as the node does not involve any additional remote data apart from its own cache."*
  - Frank und Römer. Generic Role Assignment. ETH Zürich, 2005. [2]
- *"To support mobility, the LIME model [...] defines **rules** for the sharing of [the content of multiple tuple spaces] when components are able to communicate."*
  - Picco et al. TinyLIME. Politecnico di Milano, 2005. [3]

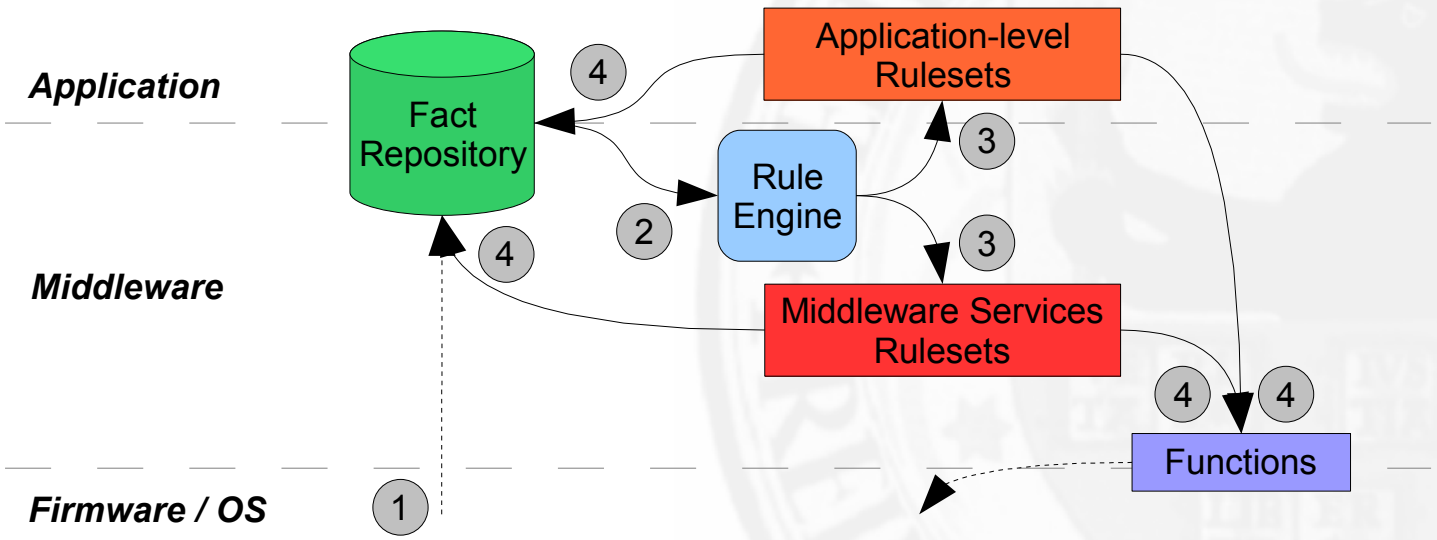
Beispiel: Messung als Regel

Wenn die lokalen Sensoren eine neue Temperatur-Messung erfasst haben **und** der Wert über 20° C liegt, **dann** schicke diesen Messwert an alle Nachbarknoten.

```
rule readAndSend
<- exists {temperature
    <- eval ({this owner} == nodeID)
    <- eval ({this value} > 20)
}
-> send BROADCAST {temperature}
```



FACTS – Eine regel-basierte Middleware Architektur für Wireless Sensor Networks



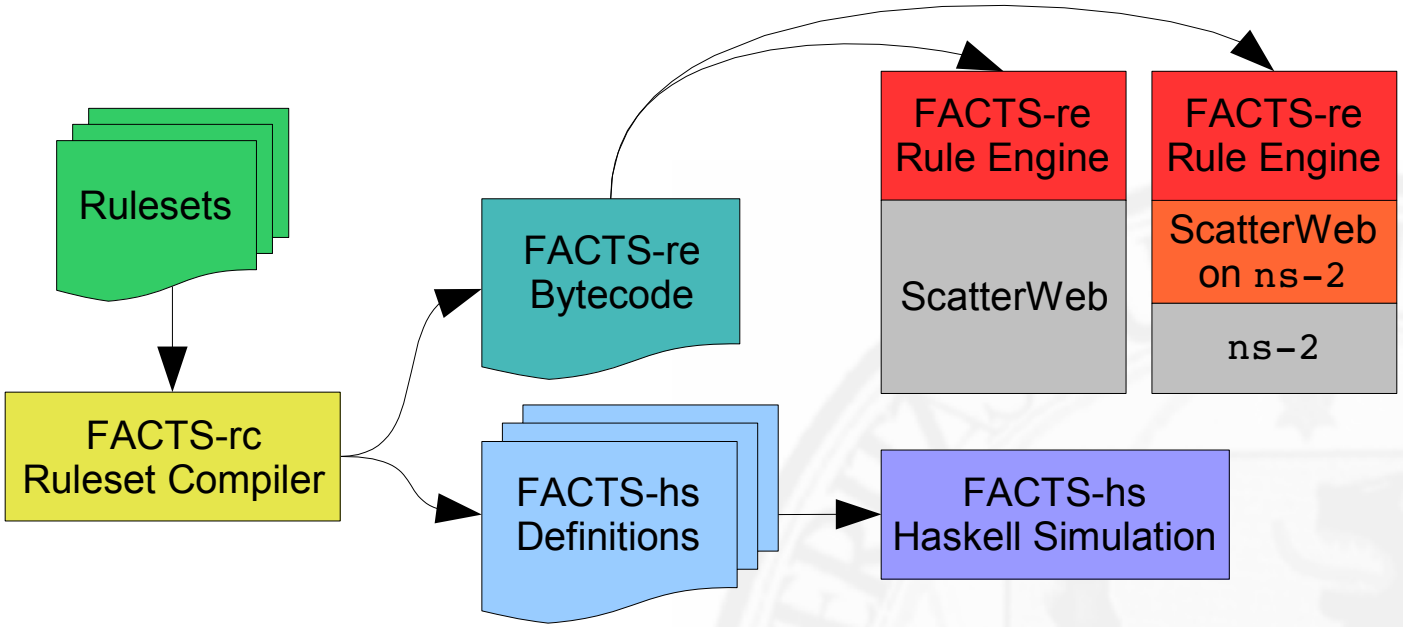
Die FACTS Middleware Architektur

- Fakten:
  - Benannte Menge von Schlüssel / Wert – Tupeln
  - Datenhaltung im *Fact Repository*
  - Versand von Fakten zwischen Sensorknoten
- Regeln:
  - Bestehen aus *Conditions* und *Statements*
  - *Rule Engine* führt *Statements* aus, wenn *Conditions* in Bezug auf das *Fact Repository* zutreffen
- Funktionen:
  - Ermöglichen Interaktion mit der Hardware

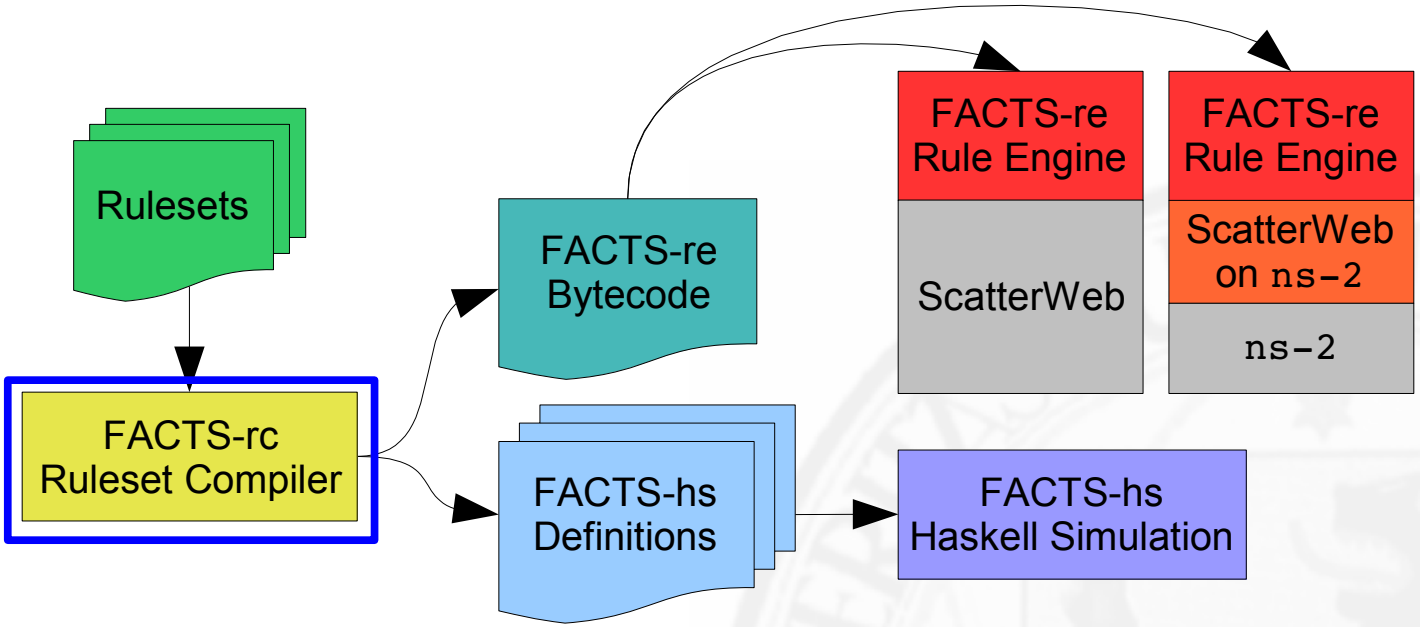
Die FACTS Middleware Architektur

- *Slots*:
  - Addressierung von Fakten im *Fact Repository*
  - Vermeidung von lokalen Variablen in Regeln
- *Rulesets*:
  - Menge von zusammenhängenden Fakten, Regeln und Slots
  - Gekapselte Implementierung eines Dienstes oder einer Anwendung
- Implementierung in der Ruleset Definition Language

FACTS Komponenten

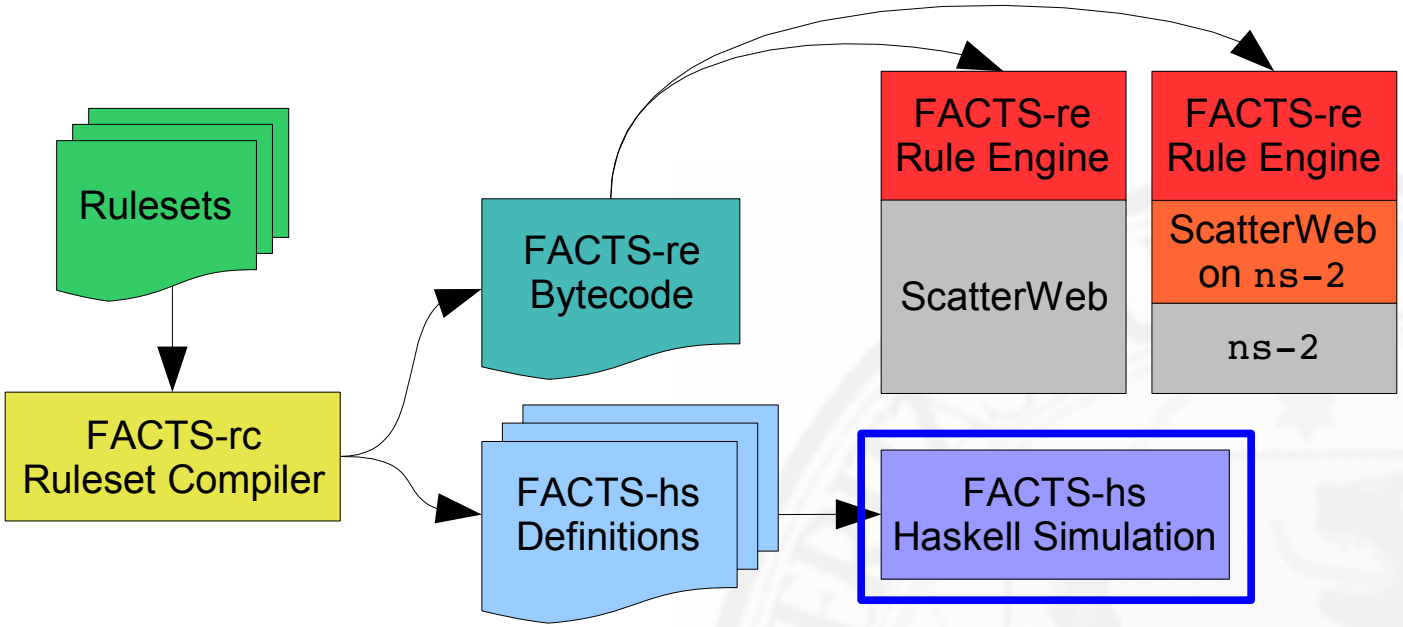


FACTS Komponenten: FACTS-rc



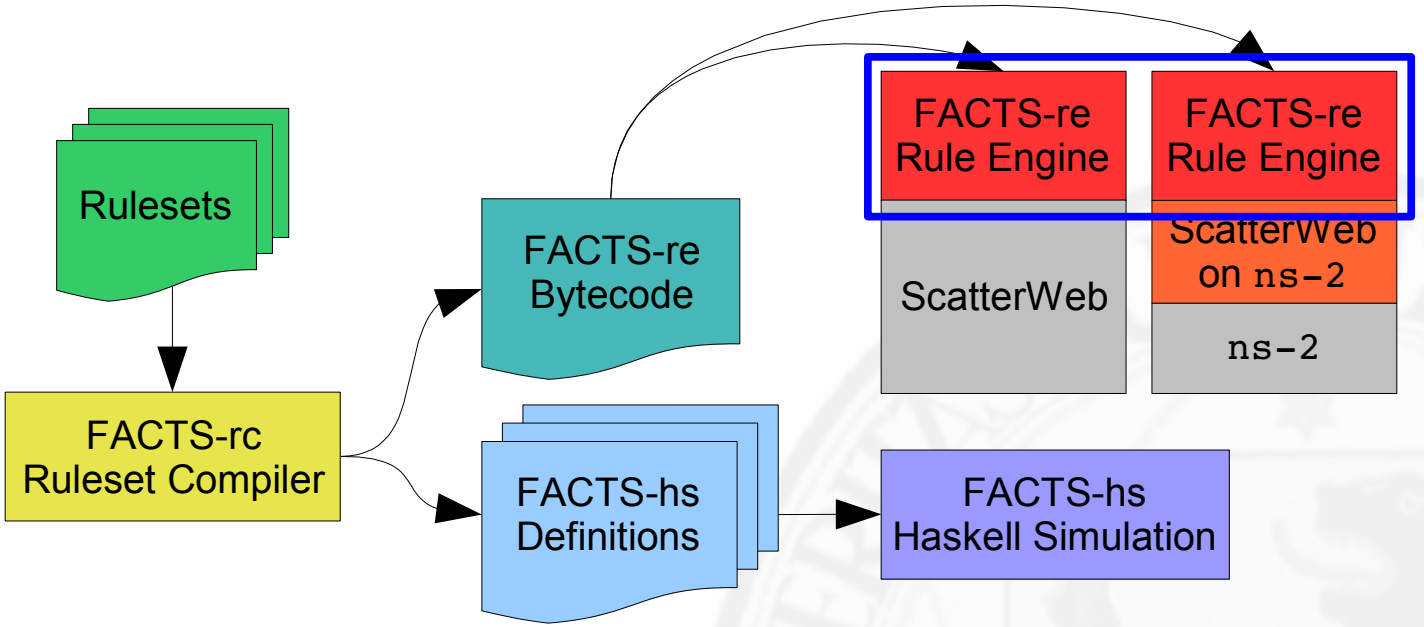
- FACTS-rc Ruleset Compiler:
  - `lex` / `yacc` Frontend für Ruleset Definition Language
  - Optimierendes Bytecode-Backend für FACTS-re
  - Haskell-Backend für FACTS-hs

FACTS Komponenten: FACTS-hs



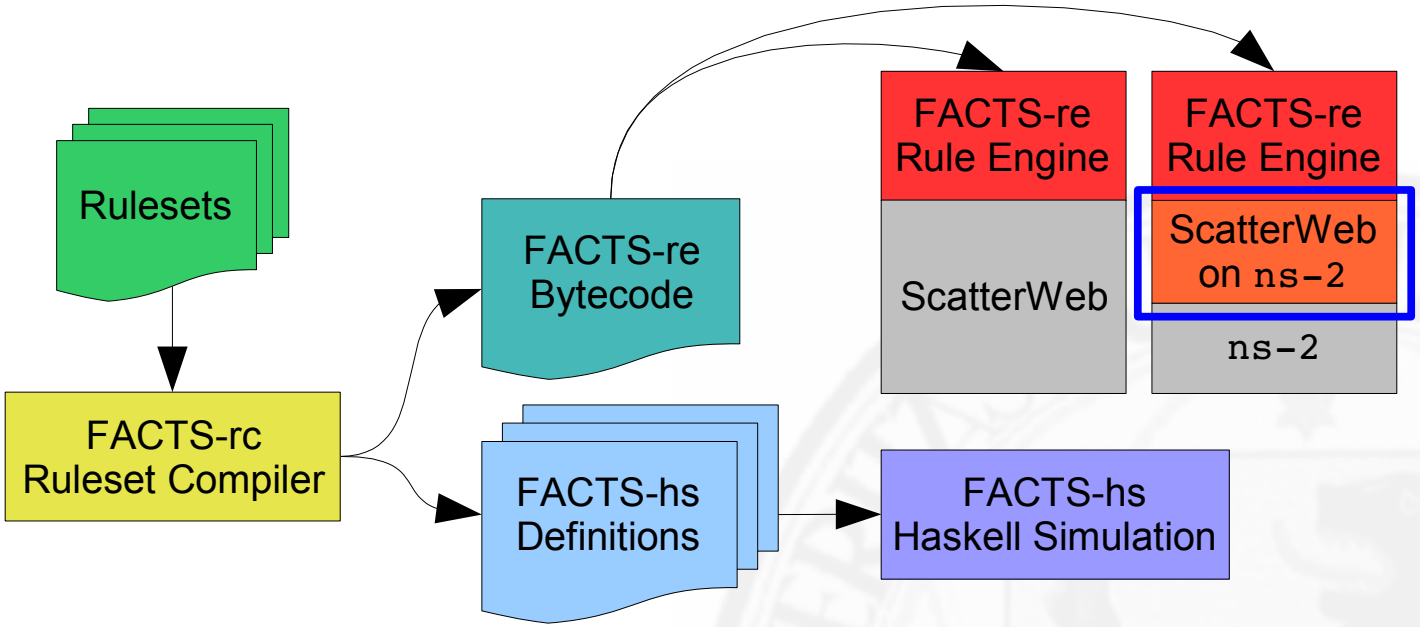
- FACTS-hs Haskell Simulation:
  - Funktionale Implemtierung der FACTS Middleware
  - Rudimentärer Simulator für WSN
  - Basis für Entwicklung der Ruleset Definition Language

FACTS Komponenten: FACTS-re



- FACTS-re Rule Engine:
  - Interpreter für FACTS Bytecode
  - Implementiert als ScatterWeb User Application
  - Speicherverbrauch: 2.536 Byte (zur Zeit)

ScatterWeb on ns-2



- ScatterWeb on ns-2:
  - Re-Implementierung der ScatterWeb Firmware API
  - ScatterWeb UserApp als C++ Objekt in ns-2
  - Ermöglicht Simulation und Entwicklung auf ns-2

Vergleich mit anderen Ansätzen

- Directed Diffusion [1]:
  - Baumstruktur zum Routen von Messwerten
  - Implementiert in FACTS: 7 Regeln, 916 Byte
- Generic Role Assignment [2]:
  - Rollenzuweisung für einzelne Sensorknoten
  - Implementiert in FACTS: 14 Regeln, 1.950 Byte
- Maté – TinyOS VM [4]:
  - Virtuelle Maschine und Bytecode-Pakete
  - FACTS Turing Maschine: 4 Regeln, 1.140 Byte

Weitere Informationen

- FACTS Middleware Architektur:
  - `http://page.mi.fu-berlin.de/~wittenbu/uni/facts/`
- ScatterWeb on ns-2:
  - `http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/tools/ns2.shtml`
- Kirsten Terfloth, Georg Wittenburg, and Jochen Schiller. FACTS - A Rule-Based Middleware Architecture for Wireless Sensor Networks. [5]

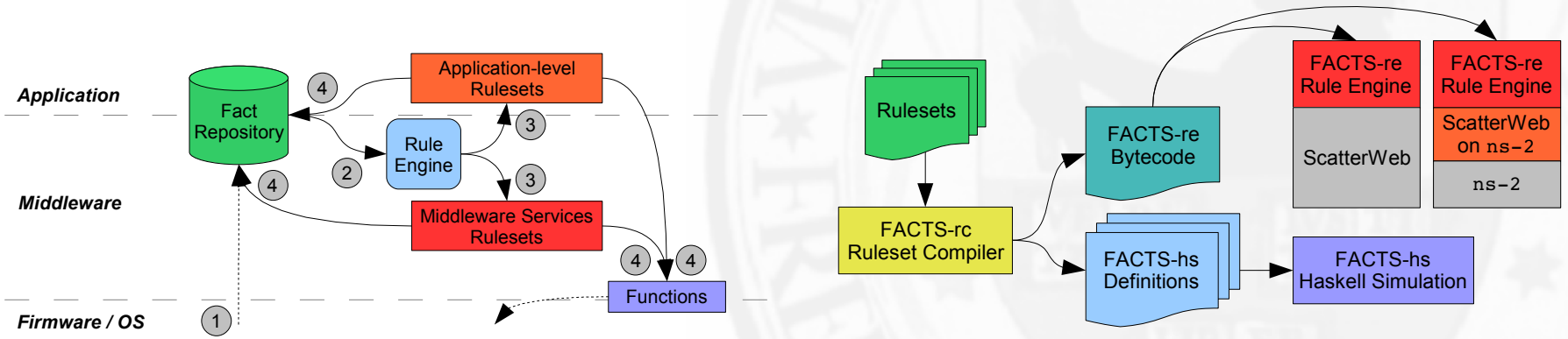
Zusammenfassung

- Es besteht eine begründete Nachfrage nach einer regel-basierten Middleware für WSNs.
- Die FACTS Middleware Architektur vereint regel-basiertes Programmieren mit geringem Verbrauch von Resourcen.
- Die Konzepte von FACTS sind erwiesenermaßen mächtig genug, um andere gängige WSN Middleware Ansätze auszudrücken.

Ende

Vielen Dank für Ihre Aufmerksamkeit!

Gibt es Fragen?



References

[1] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, Boston, Massachussetts, August 2000.

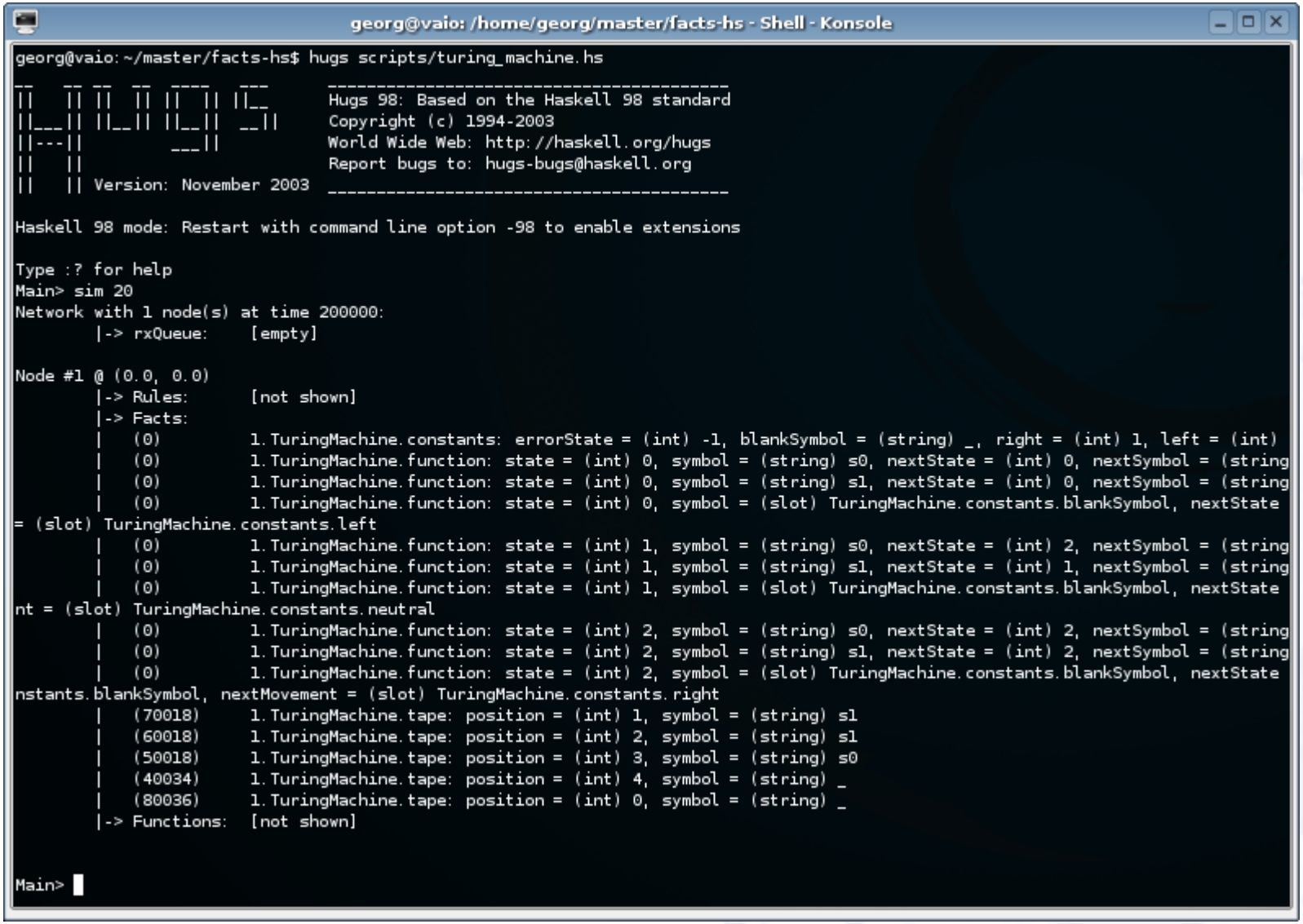
[2] Christian Frank and Kay Römer. Algorithms for Generic Role Assignment in Wireless Sensor Networks. In *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, USA, November 2005.

[3] Carlo Curino, Matteo Giani, Marco Giorgetta, Alessandro Giusti, Amy L. Murphy, and Gian Pietro Picco. TinyLIME: Bridging Mobile and Sensor Networks through Middleware. In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, pages 61-72, Kauai Island, Hawaii, 2005. IEEE Computer Society Press.

[4] Philip Levis and David Culler. Maté: A Tiny Virtual Machine for Sensor Networks. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, San Jose, California, October 2002.

[5] Kirsten Terfloth, Georg Wittenburg, and Jochen Schiller. FACTS - A Rule-Based Middleware Architecture for Wireless Sensor Networks. In *Proceedings of the First International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE)*, New Delhi, India, January 2006.

Demonstration: FACTS-hs

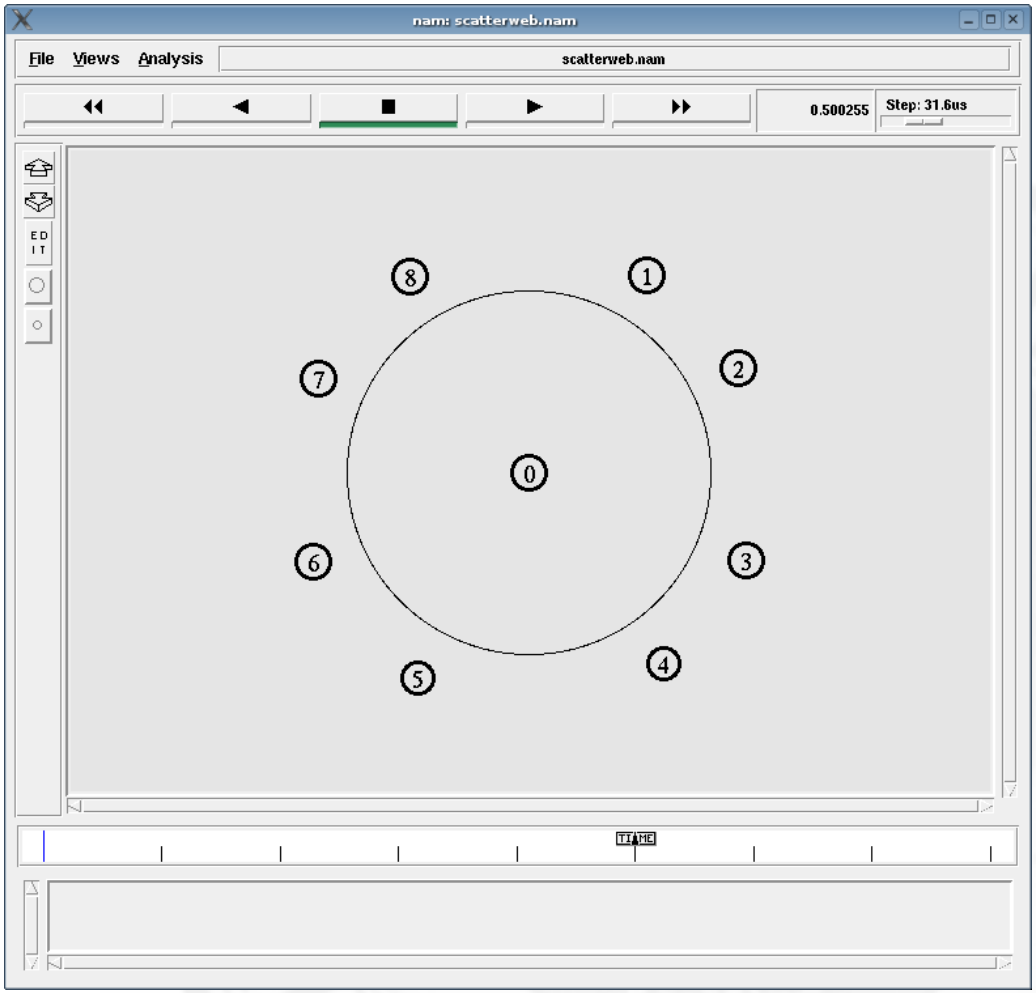


Demonstration: FACTS-re (1)

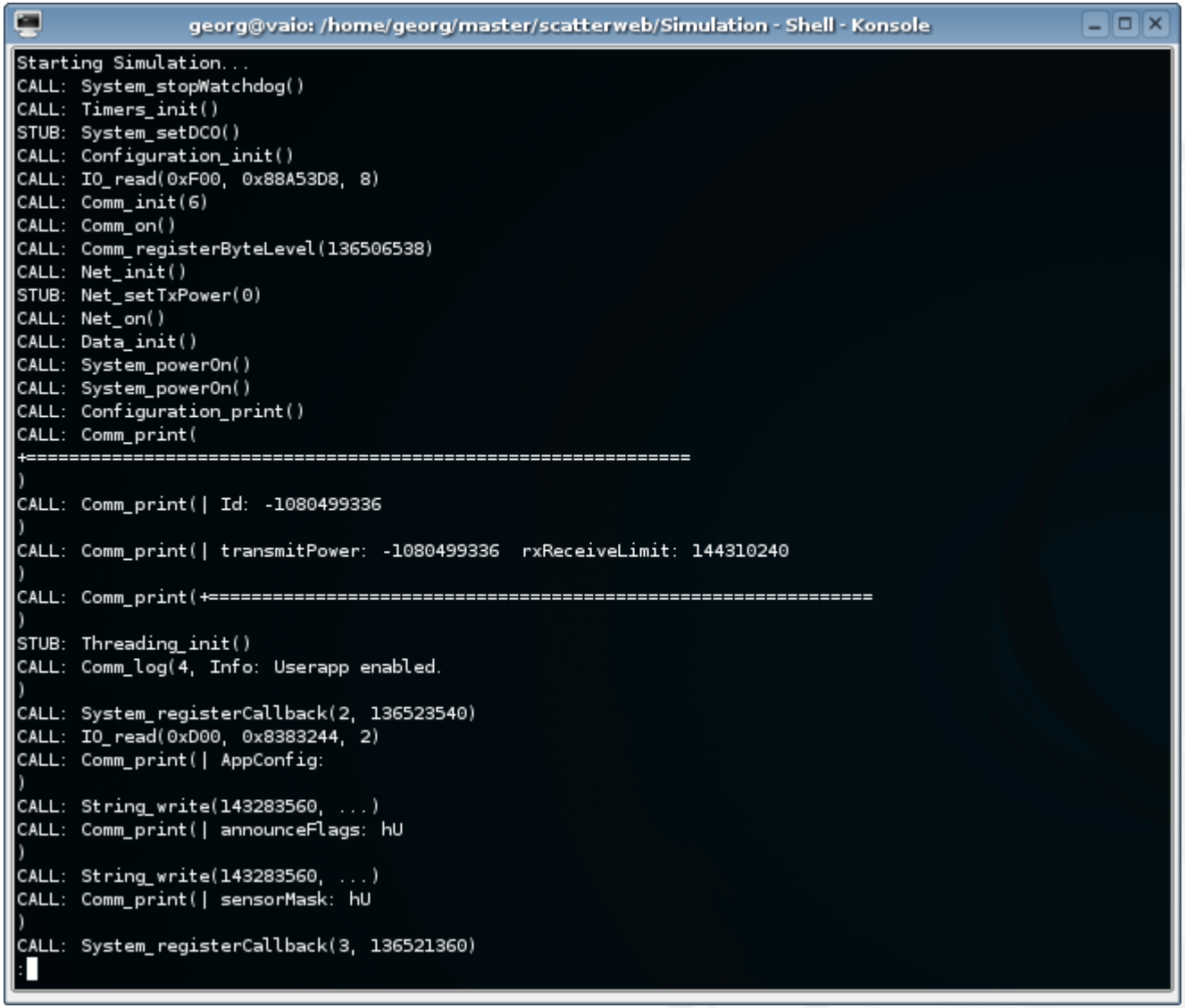
```
ruleset PingPong

[...]

rule button
<- exists {button}
-> retract {button}
-> define ping
-> send BROADCAST {ping}
-> retract {ping}
-> call greenToggle

rule ping
<- exists {ping}
-> retract {ping}
-> define pong
-> send BROADCAST {pong}
-> retract {pong}
-> call yellowToggle

rule pong
<- exists {pong}
-> retract {pong}
-> call redToggle
```

Demonstration: FACTS-re (2)

```
ruleset PingPong

[...]

rule button
<- exists {button}
-> retract {button}
-> define ping
-> send BROADCAST {ping}
-> retract {ping}
-> call greenToggle

rule ping
<- exists {ping}
-> retract {ping}
-> define pong
-> send BROADCAST {pong}
-> retract {pong}
-> call yellowToggle

rule pong
<- exists {pong}
-> retract {pong}
-> call redToggle
```



Debugging mit ScatterWeb on ns-2



Regeln im Detail

- Conditions:
  - **exists**: Existiert ein Fakt im Repository?
  - **eval**: Ist ein boolescher Ausdruck wahr?
- Statements:
  - **define**: Fügt neuen Fakt ins Repository ein.
  - **retract**: Entfernt Fakt aus Repository.
  - **send**: Versendet Fakt an andere Knoten.
  - **set**: Verändert die Werte von Fakten.
  - **call**: Ruft eine Funktion in der Firmware auf.

Beispiel: Das TickTack Ruleset

```
ruleset Timer

slot counterTicks = {"counter" ticks}

fact "counter" [ticks = 0]

rule tick 100
<- exists {"tack"}
-> retract {"tack"}
-> define "tick"
-> set counterTicks = (counterTicks + 1)

rule tack 100
<- exists {"tick"}
-> retract {"tick"}
-> define "tack"
```