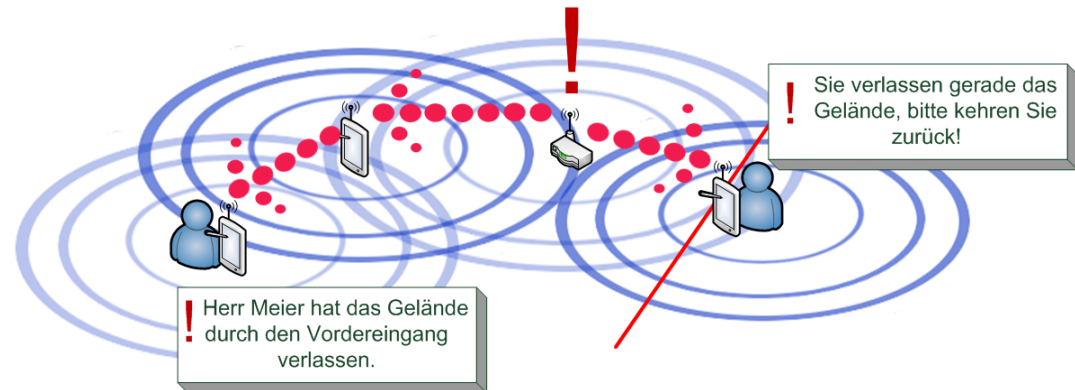


# The Potsdam Wireless Testbed

Sebastian Fudickar  
Institut für Informatik, Universität Potsdam

### KopAL Projekt

- Assistenzsystem für Senioren (in Pflegeheimen und Daheim)
- Funktionalität
  - Notruf / Sturzerkennung
  - Lokalisierung
  - Terminerinnerung

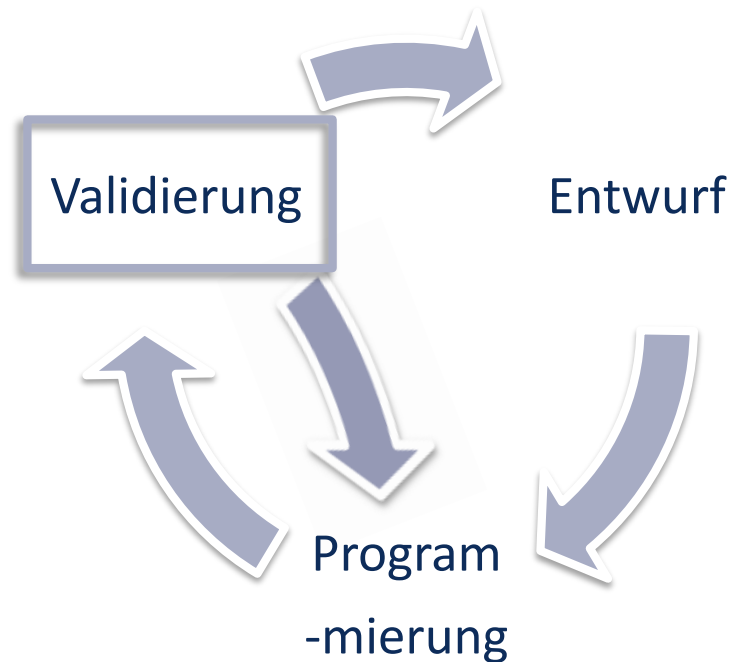


### Entwicklung zuverlässiger MAC und ad-hoc Routingprotokolle

→ Entwicklungsprozess erfordert häufige Validierung der Software

### Entwicklungsschritte mobiler Anwendungen / Funkprotokolle

- Entwurf
- Programmierung
- Validierung via Simulation oder Hardwaretests



# Potsdam Wireless Testbed

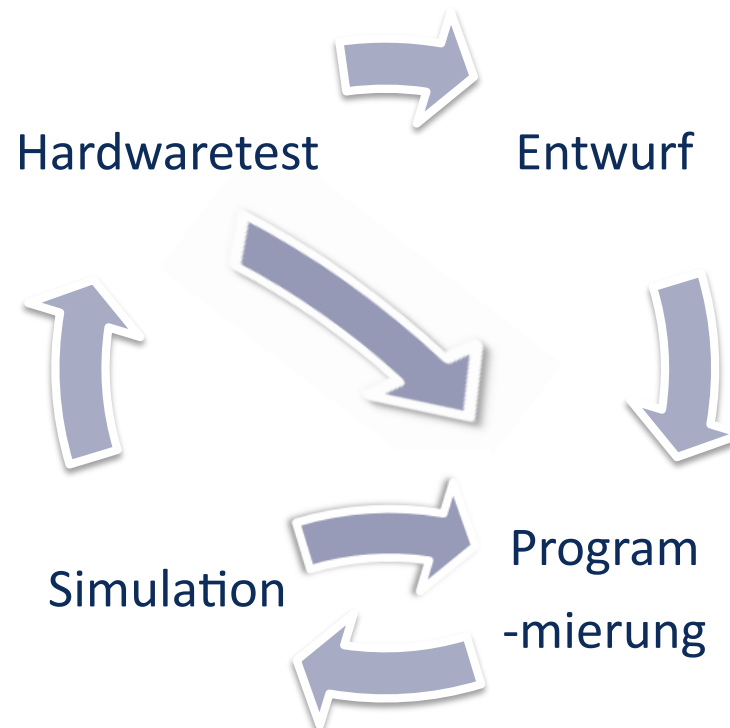
## Vergleich Simulation und Hardwaretests



	Simulation	Hardwaretests
Anschaffungskosten	Gering	- Hoch
Installationsaufwand	Gering	- Häufig manuell
Reproduzierbarkeit der Ergebnisse	Hoch	- Eingeschränkt
Aussagekraft der Ergebnisse	Näherungsweise (Modelle)	Realistisch

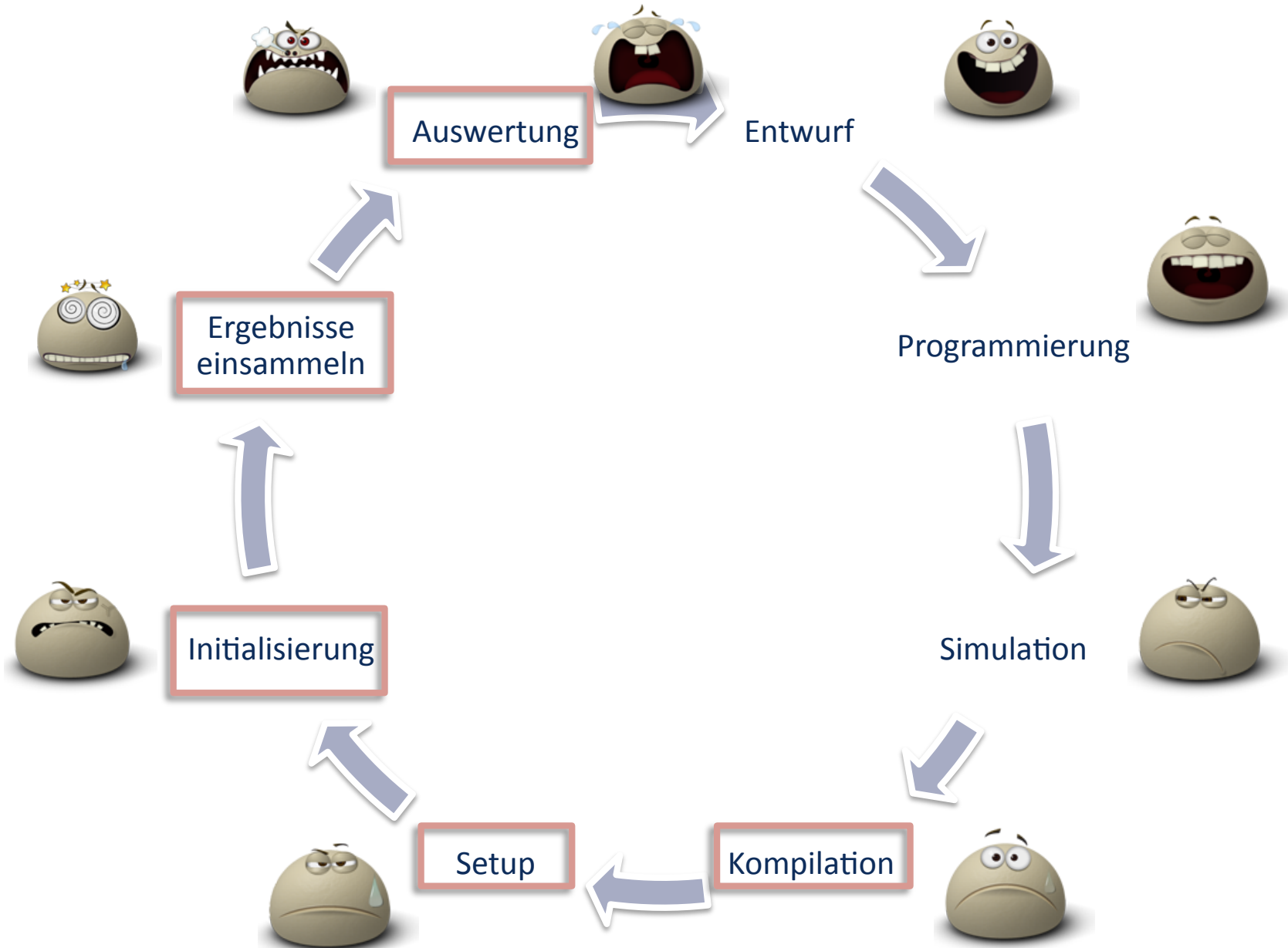
- Hardwaretests sind essentiell
- Mehrphasige Validierung
  1. Simulation (NS 2, Omnetpp, ...)
  2. Hardwaretests

- Entwurf
- Programmierung
- Simulation
- Hardwaretest



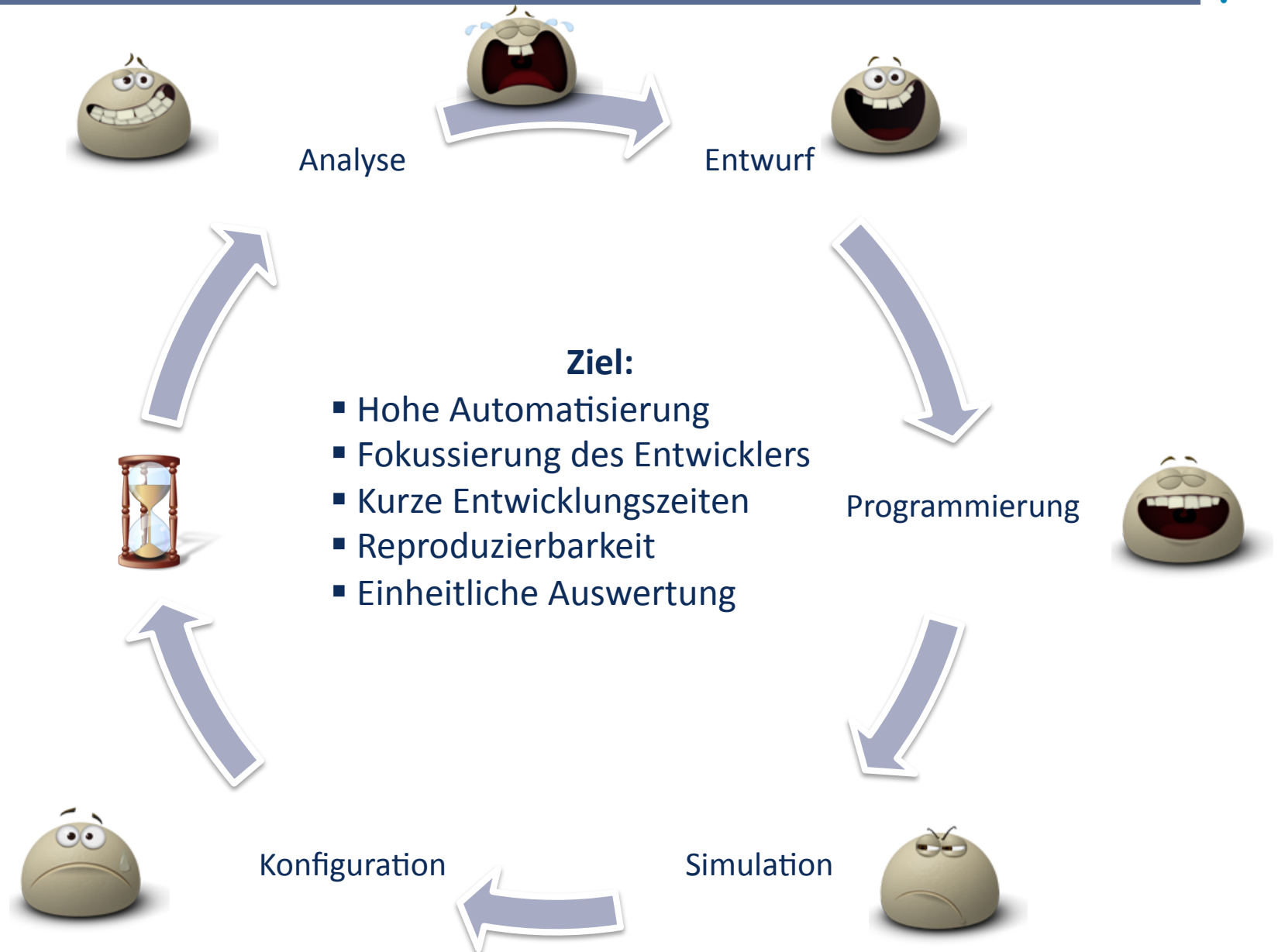
# Potsdam Wireless Testbed

## Entwicklungsschritte



# Potsdam Wireless Testbed

## Entwicklungsschritte



## Potsdam Wireless Testbed

- Hohe Automatisierung
- Entlastung der Entwickler
- Fokussierung auf die Entwicklung
- Kurze Entwicklungszeiten
- Reproduzierbarkeit
- Einheitliche Auswertung
  
- Wiederverwendbarkeit von
  - Konfigurationen
  - Software
- Kurze Einarbeitungszeiten
- Unterstützung vielfältiger Geräte

- Linksys WRT54GL Router (60€)
  - 200MHz MIPS Prozessor
  - 4 MB Flash, 16 MB RAM
  - WiFi, Ethernet
- Nokia N8x0 - N900 (350€)
  - 400+ MHz ARM CPU
  - 256+ MB Flash, 128+ MB RAM
  - WiFi, USB Serialport
- Openmoko Freerunner (200€)
  - 400 MHz ARM CPU
  - 256 MB Flash, 256 MB RAM
  - WiFi, USB Serialport
- X86 Computer

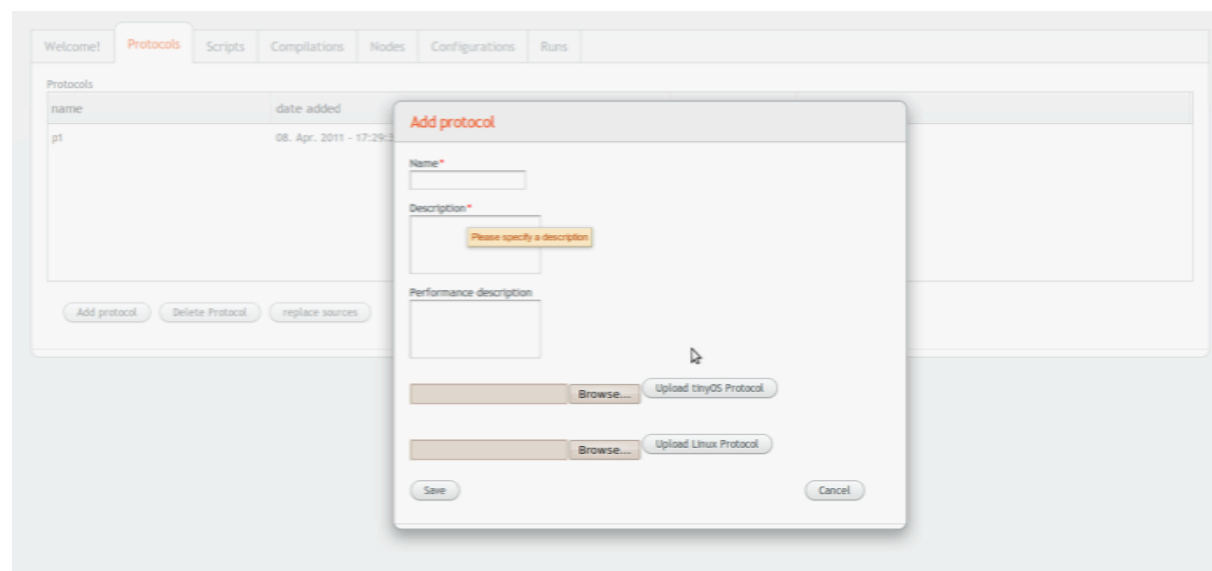


- Prozessorarchitekturen (MIPS, ARM, x86)
- Eingeschränkte Rechenleistung, Speicherkapazität
- Verschiedene Netzwerkschnittstellen (WiFi, Ethernet, USB)

- Erstellen der Software
  - Routingprotokoll oder Zusatzsoftware ( c )
  - Messskripte (c oder Skriptsprachen)
- Konfiguration
  - Softwareupload
    - Autom. Kompilation für alle Prozessorarchitekturen
  - Knoten & Knotengruppen (wiederverwendbar)
  - Testlauf (Routingprotokoll, Messskript, Laufzeit, Knotengruppe)
- Autom. Testlauf
  - Setup
  - Initialisierung
  - Abrufen und Zusammenfassen der Messergebnisse und Logdateien
- Analyse der Ergebnisse Interpretation



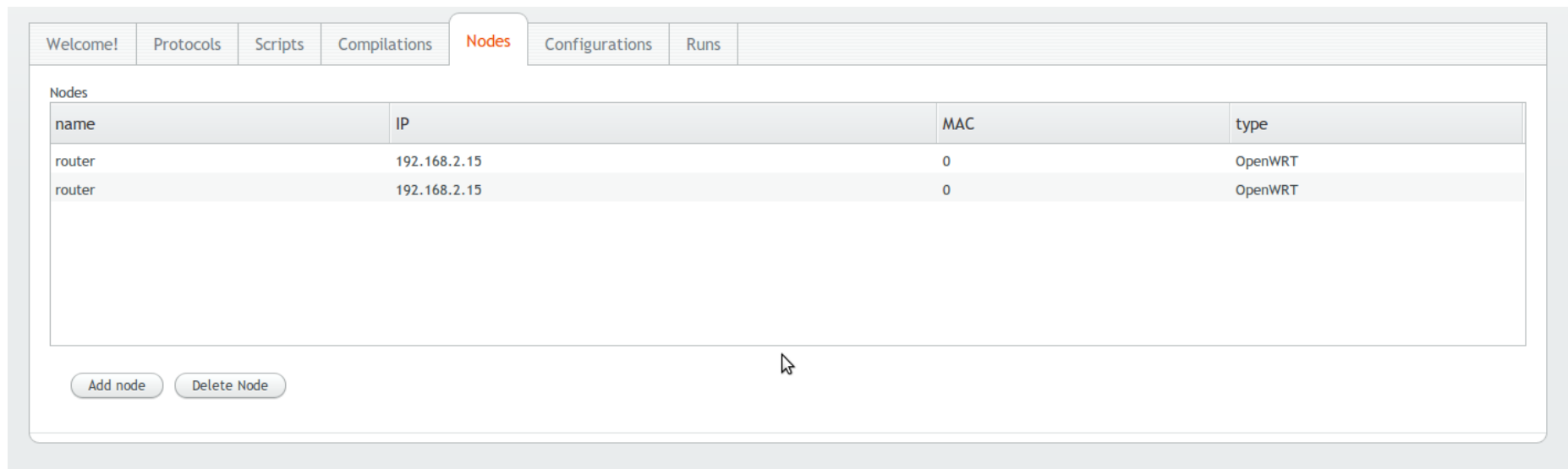
- Softwareupload
  - Aller Dateien als .tar.gz (zzgl. eindeutigem Name und Beschreibung)
- Kompilation
  - Hochgeladene Software wird automatisch Kompiliert
  - Cross-Compiler für ARM, MIPS und x86 Architekturen
  - Im Fehlerfall
    - Nutzer Benachrichtigung
    - Logdateien einsehbar



- Mess-Skripte
  - Universell einsetzbar
  - Programmierung via C oder Skriptsprache (z.B. Shellskript
    - Nutzung aller standardmäßigen Linux – programme (ping, iwlist, wget, awk, date,...) möglich
    - Messergebnisse in */www/scripting/results* speichern
  - Shellsript (messscript.sh)
  - C-Programm (messscript.c)

## Spezifikation Knoten und Knotengruppen

- Knoten: Name, IP oder MAC, Art
- Knotengruppen können einmalig definiert werden

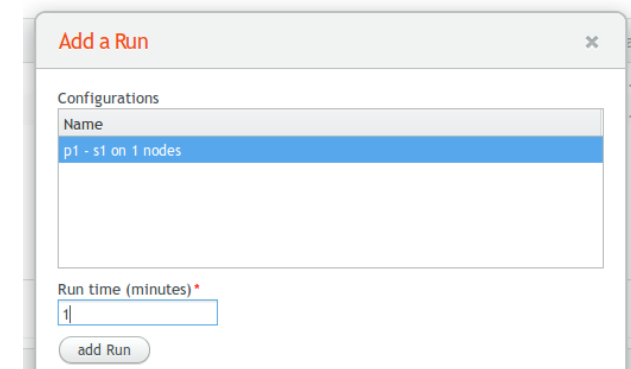


Nodes

name	IP	MAC	type
router	192.168.2.15	0	OpenWRT
router	192.168.2.15	0	OpenWRT

Add node Delete Node

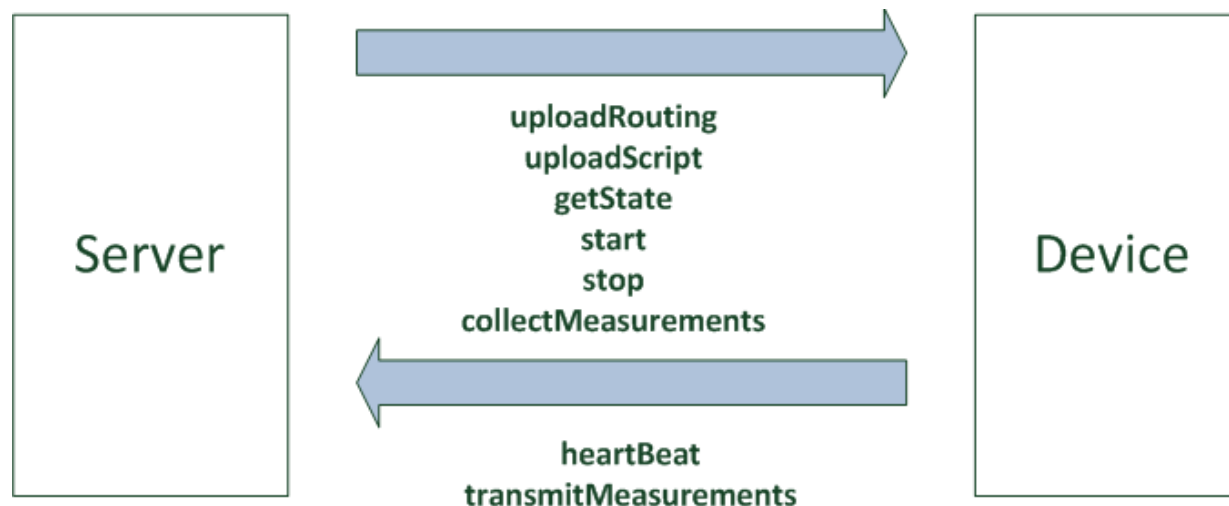
- Initialisieren eines Testlaufs
  - Selektion einer Konfiguration (Routingprotokoll, Messskripts, Knotengruppe)
  - Laufzeit
  
- Start des Laufs wird „gescheduled“
  - Scheduler verwaltet anstehende Läufe und führt diese sequentiell aus



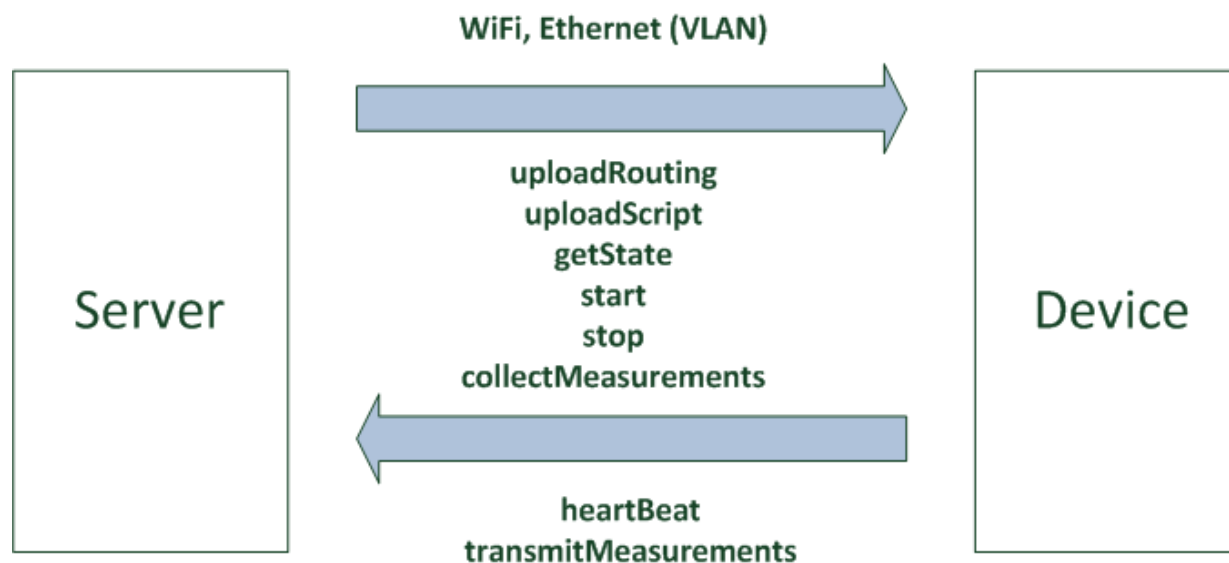
Welcome!   Protocols   Scripts   Compilations   Nodes   Configurations   <b>Runs</b>					
Runs					
name	duration	status	start	end	
p1 - s1 on 1 nodes	2	finished	08. Apr. 2011 - 17:32:28	08. Apr. 2011 - 17:34:30	
p1 - s1 on 1 nodes	1	finished	08. Apr. 2011 - 17:37:21	08. Apr. 2011 - 17:38:25	
p1 - s1 on 1 nodes	1	finished		08. Apr. 2011 - 17:50:06	
p1 - s1 on 1 nodes	1	scheduled			

add Run | up | down | show details | stop run

- Ausführung eines Testlaufs
  - Upload des Routing Protokolls
  - Upload des Mess-Skriptes
  - Validierung des Knotenzustandes
  - Starten des Testlaufs
  - Knoten prüfen Erreichbarkeit des Servers
  - Stoppen des Testlaufs
  - Abrufen der Mess-Ergebnisse



- Ziel:
  - Interoperabilität und Portabilität des Mechanismus
  - Ressourcen schonend
- HTTP get/post Requests
- Webserver mit php:
  - Apache / lighttpd / hiawatha
  - sh-skripte für interne Verarbeitung



## Sonderfälle

- Nicht Erreichbarkeit des Servers:
  - Knoten prüfen regelmäßig die Erreichbarkeit des Servers mittels heartbeat Nachrichten
  - Bei Misserfolg abschalten der Messung und zurücksetzen des Routingprotokolls auf Fallbacklösung
- Knoten-Speicher voll:
  - Knoten übertragen während der Messung Messergebnisse an den Server

- Vorverarbeitung der Messergebnisse
  - Abrufen von den Knoten
  - Parsen
  - Übertragung in Datenbank
  - Berechnung der Werte
- Visualisierung der Messergebnisse (JavaScript und Gnuplot)
- Logdateien
  - Abrufen von den Knoten
  - abrufbar

## Ausblick

- Erweiterung unterstützter Hardware
- Integration Benachrichtigungssystem (e-Mail)
- Integration "reproduzierbare" Mobilität
- Verfeinerte Auswertungsfunktionalität
- Zeitsynchronisation
- Langzeittests
  
- Opensource / Kooperationen: Nutzung und Erweiterung

**Danke für Ihre Aufmerksamkeit!**

**Fragen,  
Vorschläge,  
Ideen?**

**Kontakt:**

M.Sc. Sebastian Fudickar  
+49 331 977 3123  
fudickar@cs.uni-potsdam.de  
[www.cs.uni-potsdam.de/bs](http://www.cs.uni-potsdam.de/bs)