An Optimized WS-Eventing for Large-Scale Networks

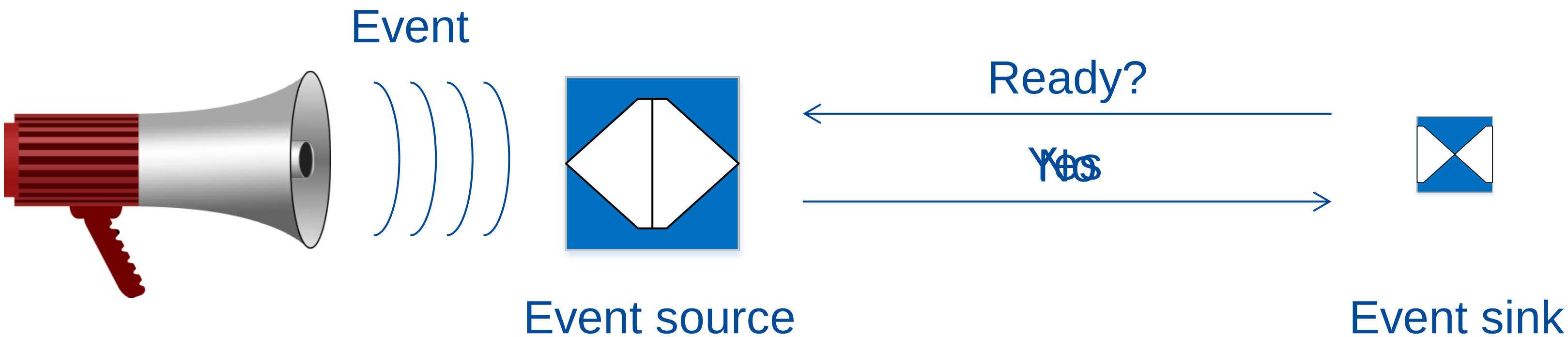
Jan Skodzik, Vlado Altmann, Peter Danielis,
Moritz Koal, Dirk Timmermann

University of Rostock, Germany

Institute of Applied Microelectronics
and Computer Engineering

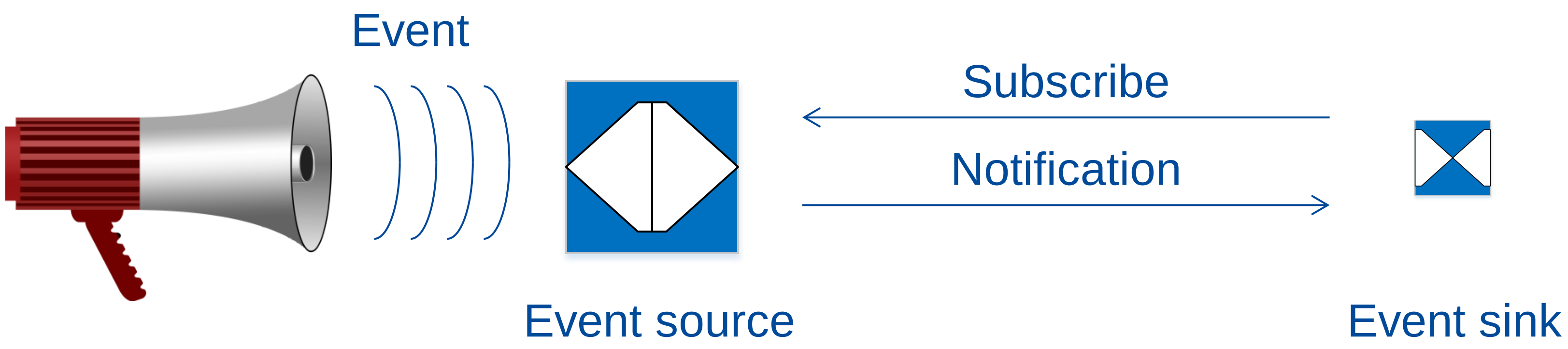
Eventing

- Easiest way to detect an event is polling
- Very ineffective



Eventing

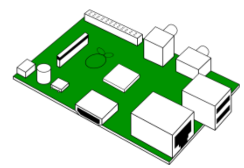
- Avoiding polling
- Eventing (realized e.g. by publish/subscribe pattern)



Eventing for automation

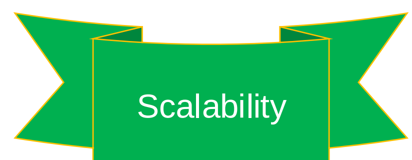
- Additional requirement for automation:
  - Standardized

  - Constrained resources



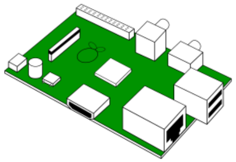
  - Hard real-time behavior

  - High scalability

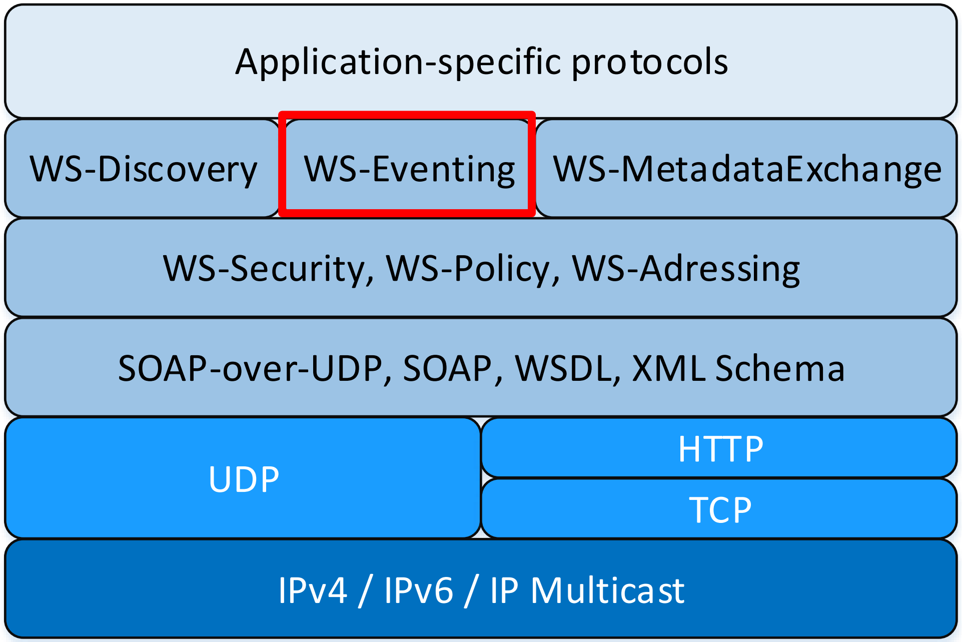


  - High reliability

State of the art

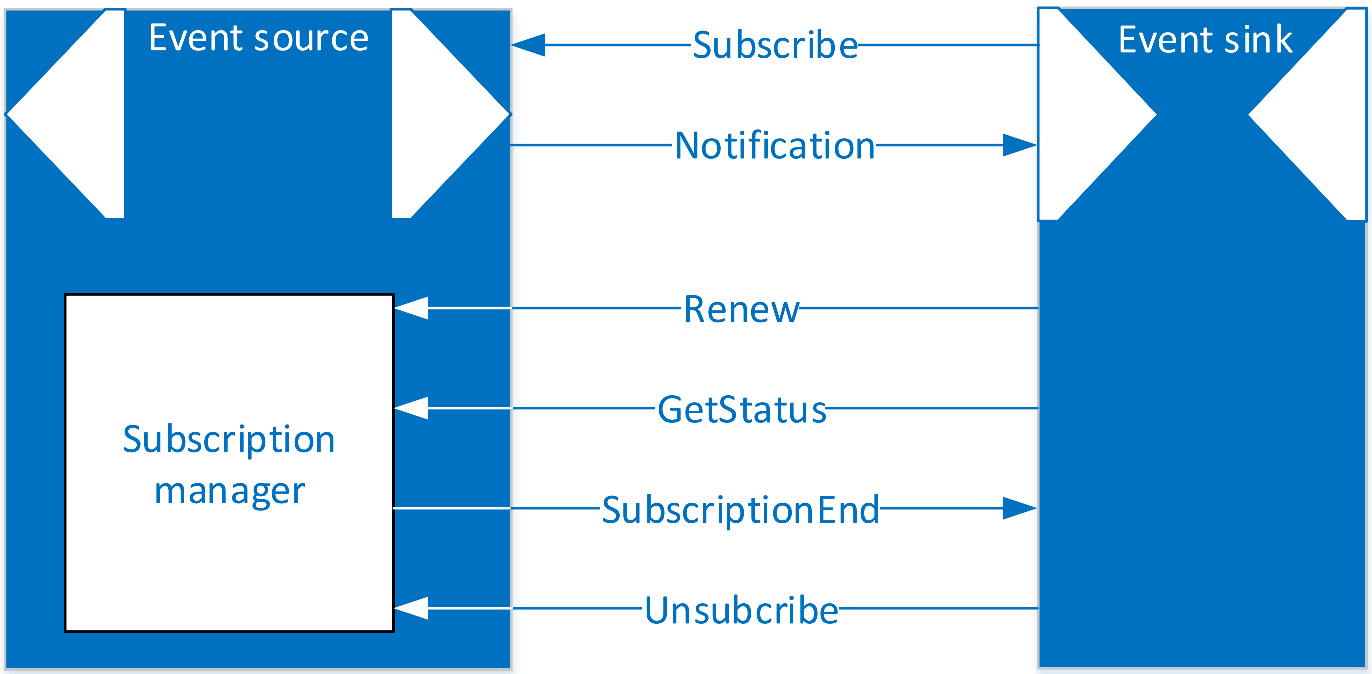


- We focus on Devices Profile for Web Services (DPWS)
  - Designed for devices with constrained resources
  - Intended domain (automation)



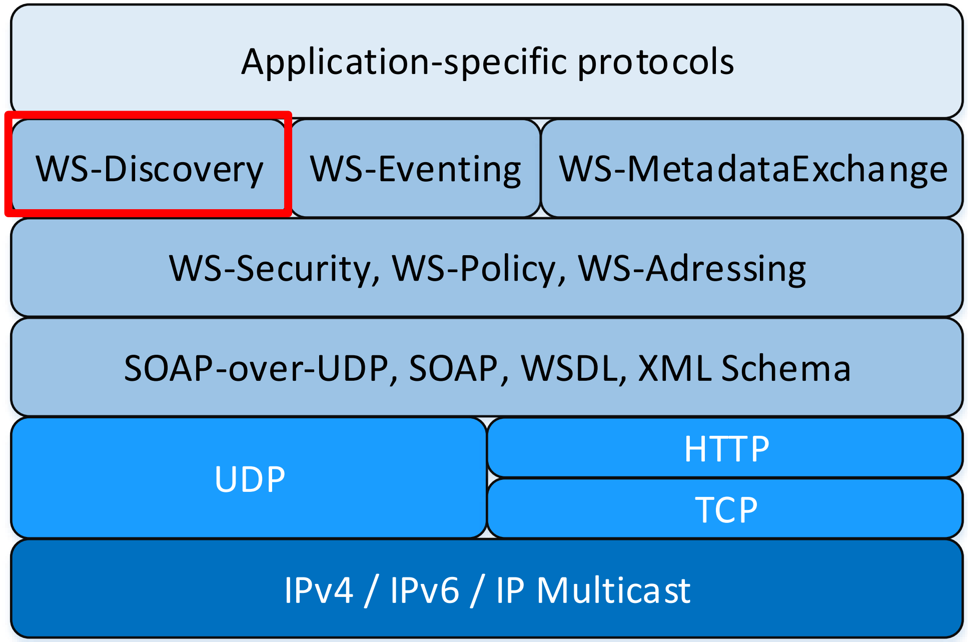
- WS-Eventing included as a standard (by OASIS)

State of the art : WS-Eventing basics



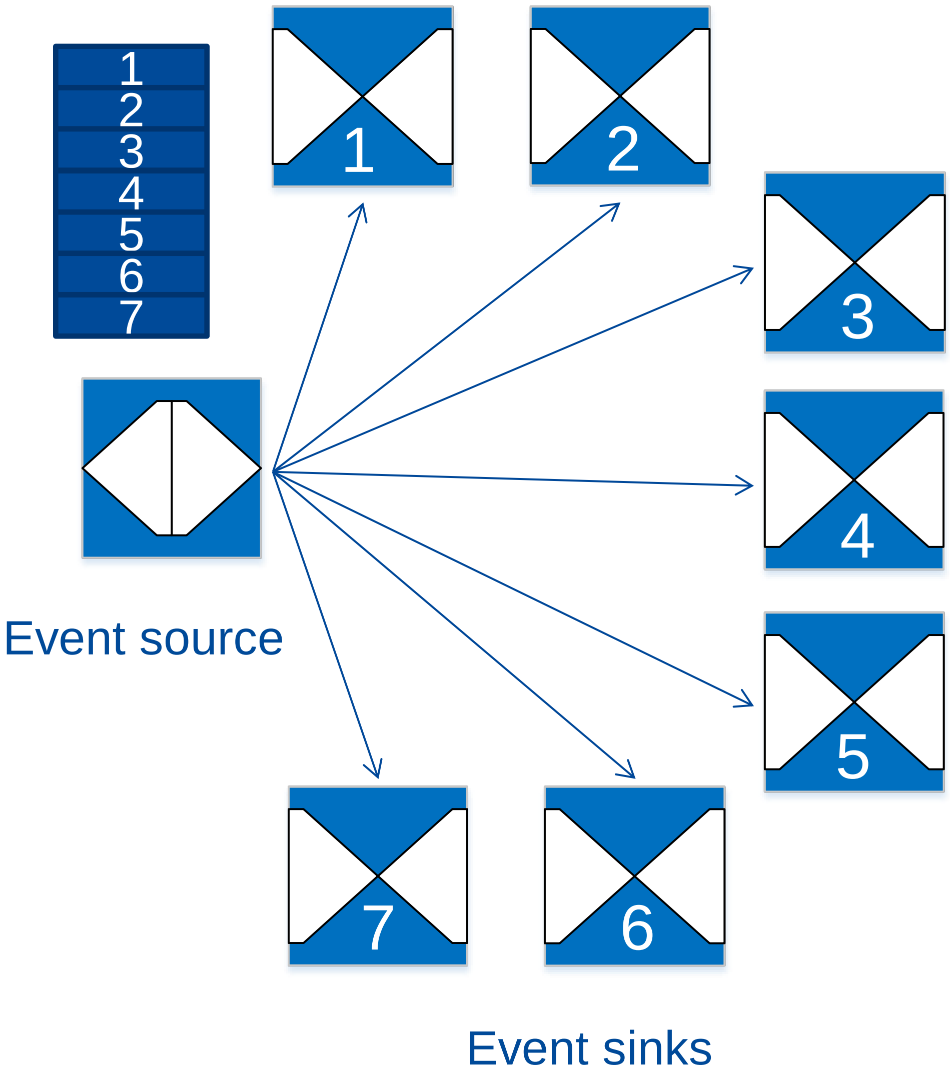
How do we find an event source?

- Usage of WS-Discovery
- Already optimized in a highly scalable manner*
- Based on the P2P network Kad(emlia)



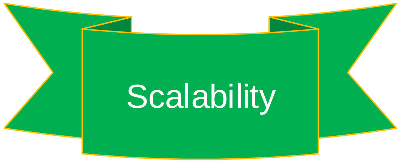
*Vlado Altmann et al.: "**A DHT-based Scalable Approach for Device and Service Discovery"**
12th IEEE International Conference on Embedded and Ubiquitous Computing (EUC14), August 2014

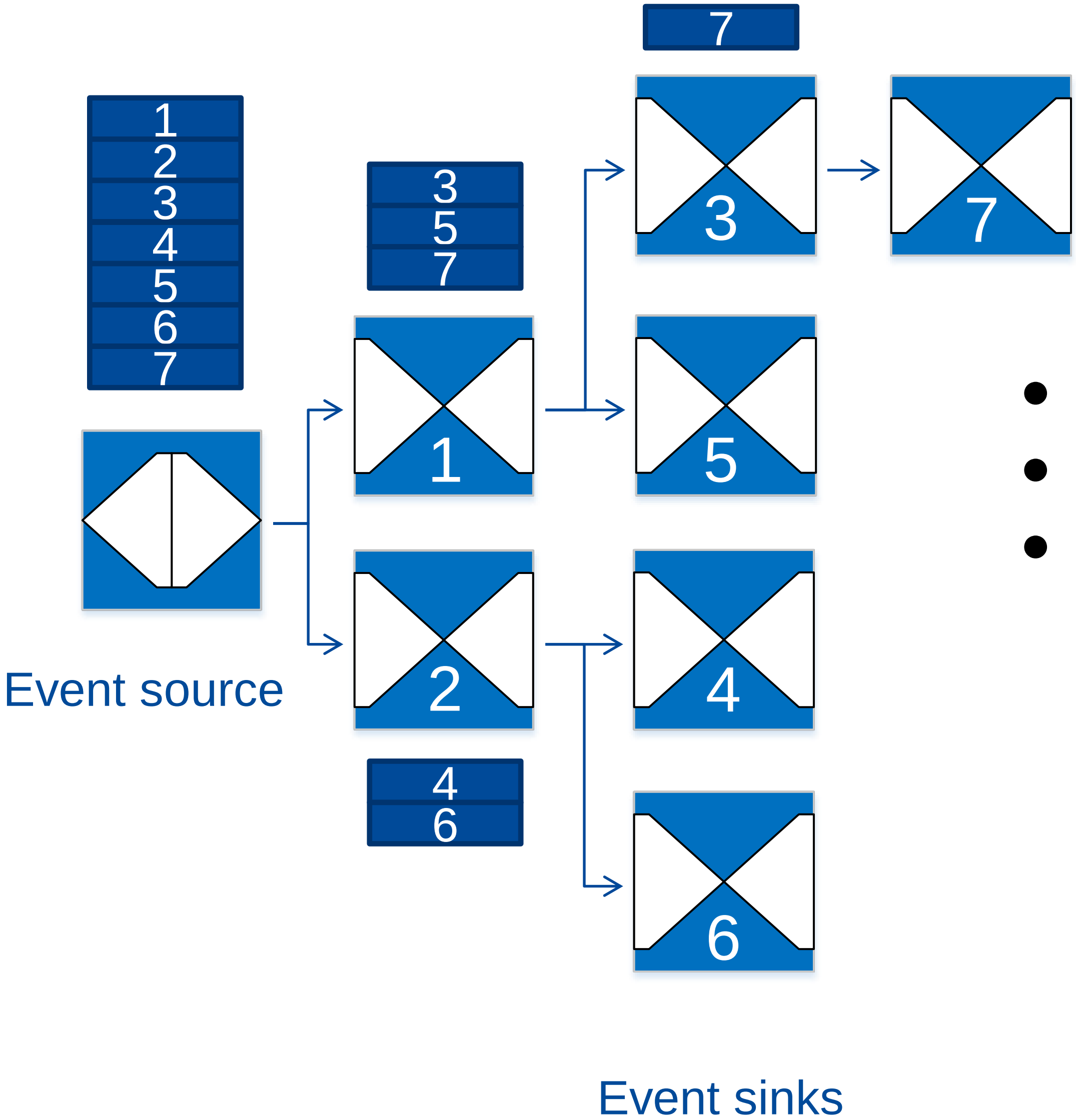
Standard notification procedure



- What happens if an event occurs?
- Notification transmitted sequentially
- Scales badly
- Overloading of the source

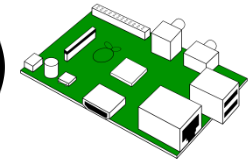
Highly scalable notification procedure



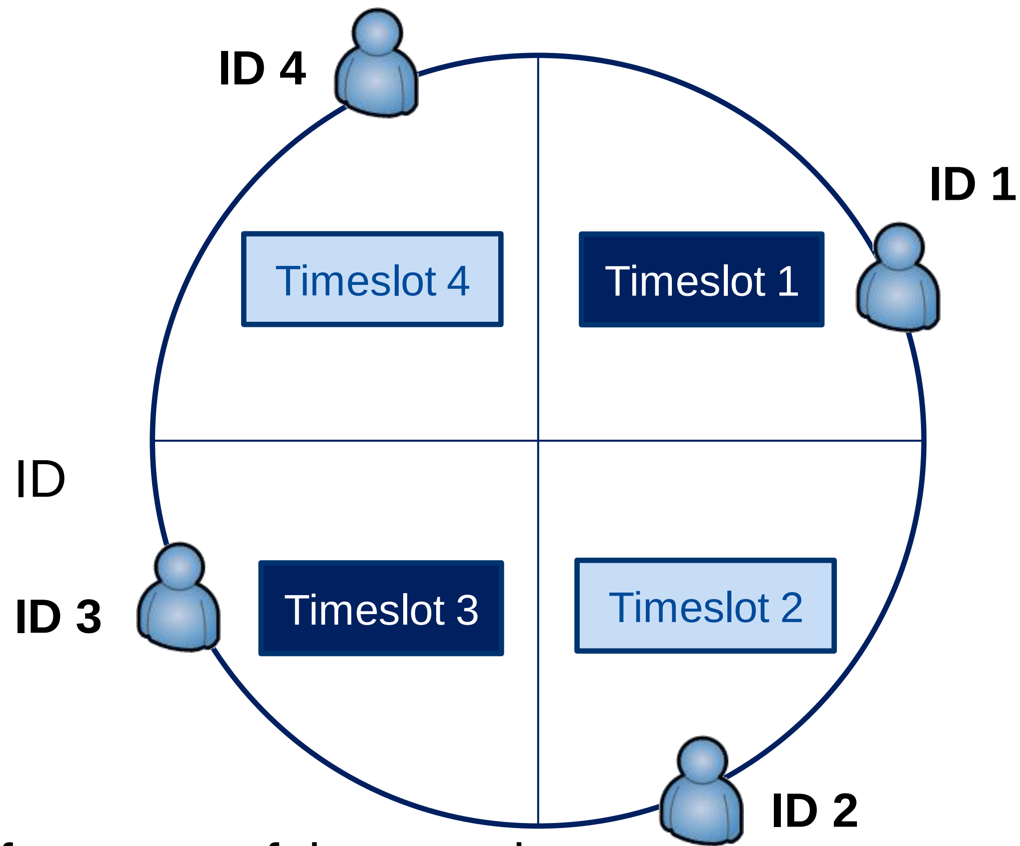


- Utilizing the subscribed sinks
- High scalable approach
- Forwarding the subscription list

How to achieve real-time



- For a Real-time System by choosing a platform and OS
- For communication a TDMA-based Peer-to-Peer approach has been developed called HaRTKad*
  - Controlled media access
  - Node has unique ID
  - Timeslot determination via ID



- We need to determine the performance of the eventing

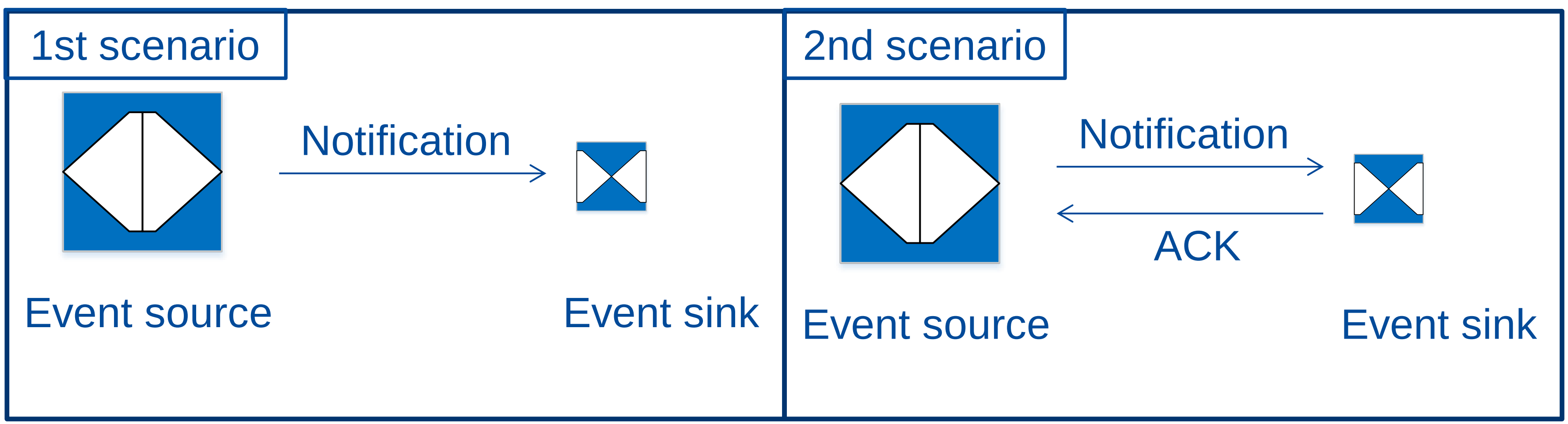
*Jan Skodzik et al.: "**HaRTKad: A hard real-time Kademlia approach**"
11th IEEE Consumer Communications and Networking Conference (CCNC)

Two scenarios

Reliability

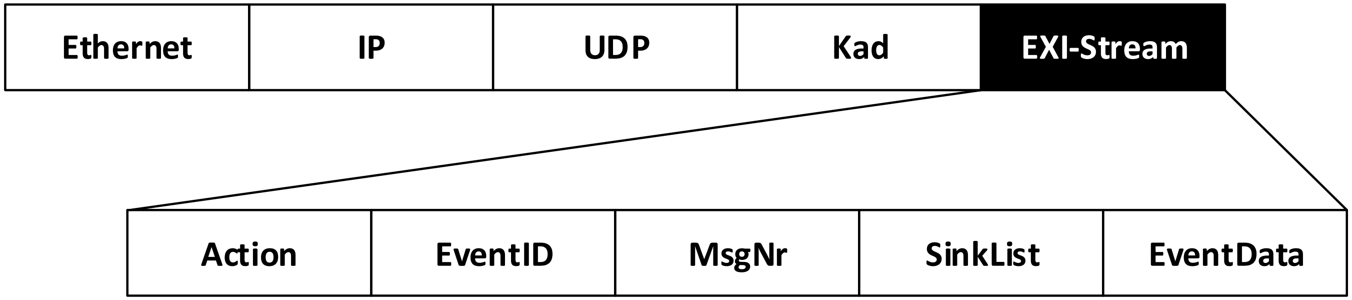
- 1st scenario:
  - Sending the notification without any acknowledge (ACK)
  - Less reliability, less complexity, less data volume

- 2nd scenario:
  - A response from every notified event sinks is required
  - Higher reliability

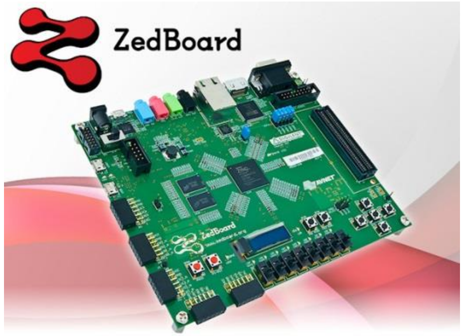


EXI: Efficient XML Interchange

- In WS the data are usually XML coded
  - Not very efficient
- Usage of Efficient XML Interchange (EXI)
- EXI allows a high lossless compression
  - Binary notation of the data
- W3C standard



Prototype



- ZedBoard as target platform
  - ARM Dual Core @ 667 MHz
  - 1-GBit Ethernet connection
- Runs the Kad software (HaRTKad application)

WS-Eventing

HaRTKad

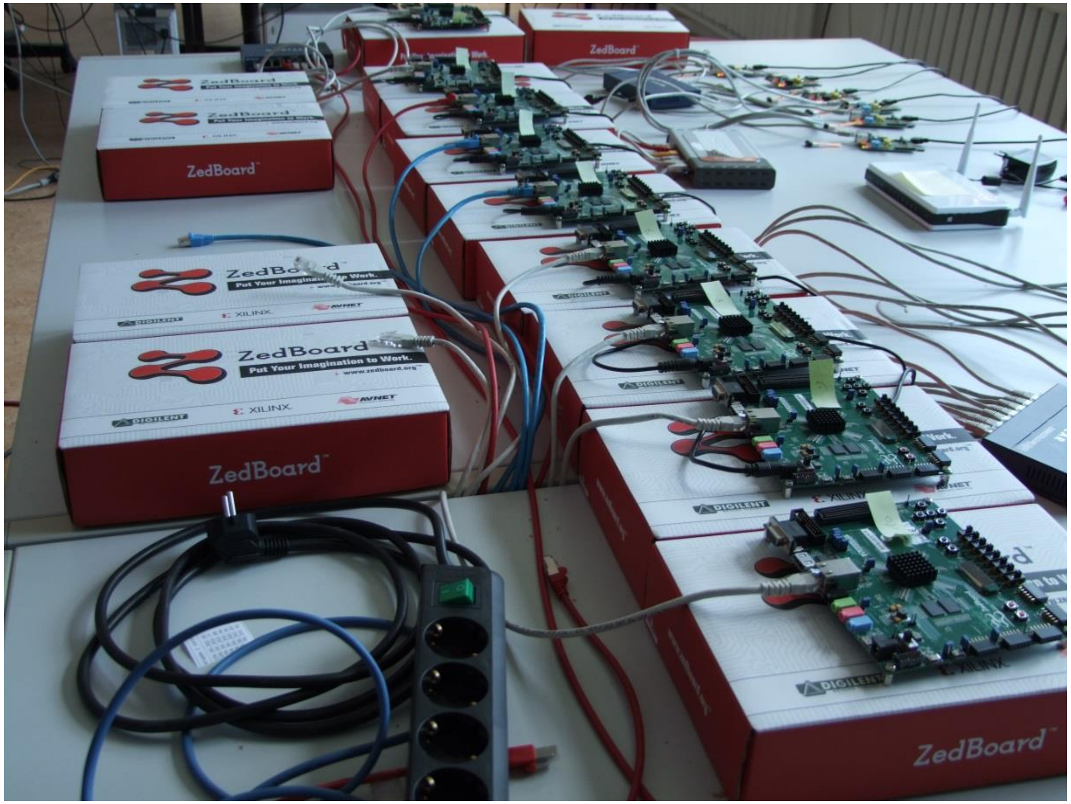
lwIP (UDP/IP)

FreeRTOS

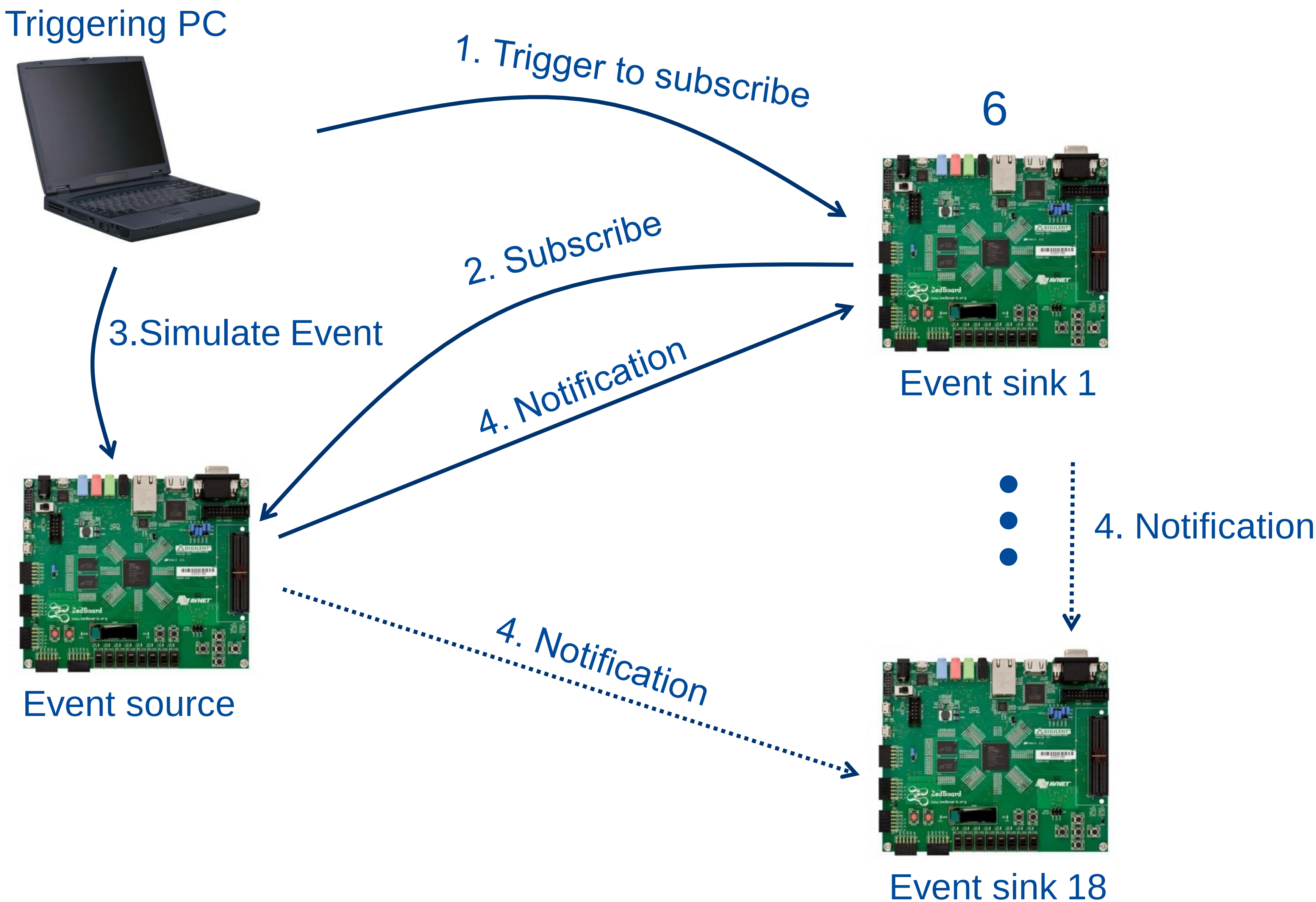
ARM-based Hardware

Prototype setup

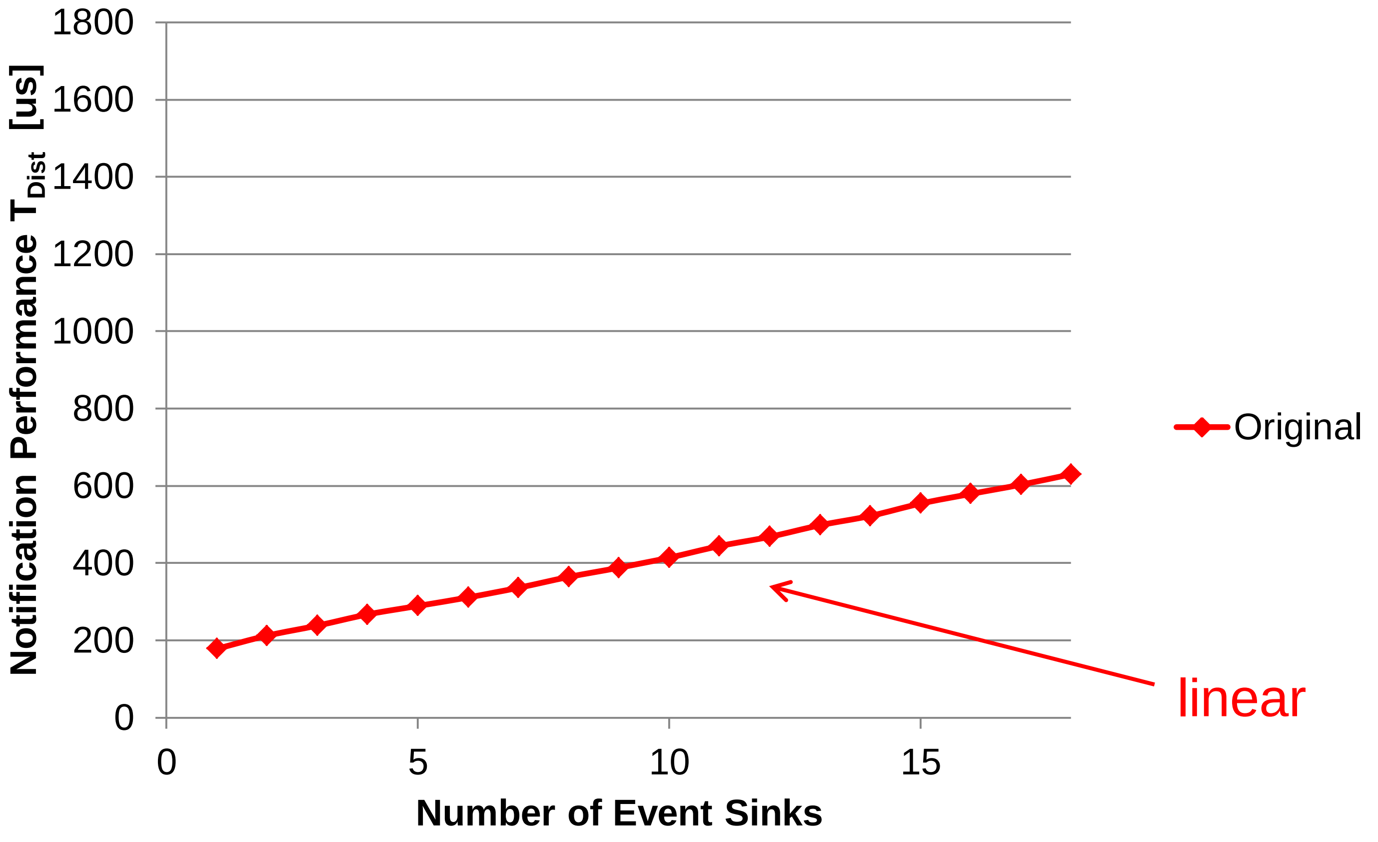
- One event source and 18 event sinks
- Connected via a 1-GBit switches



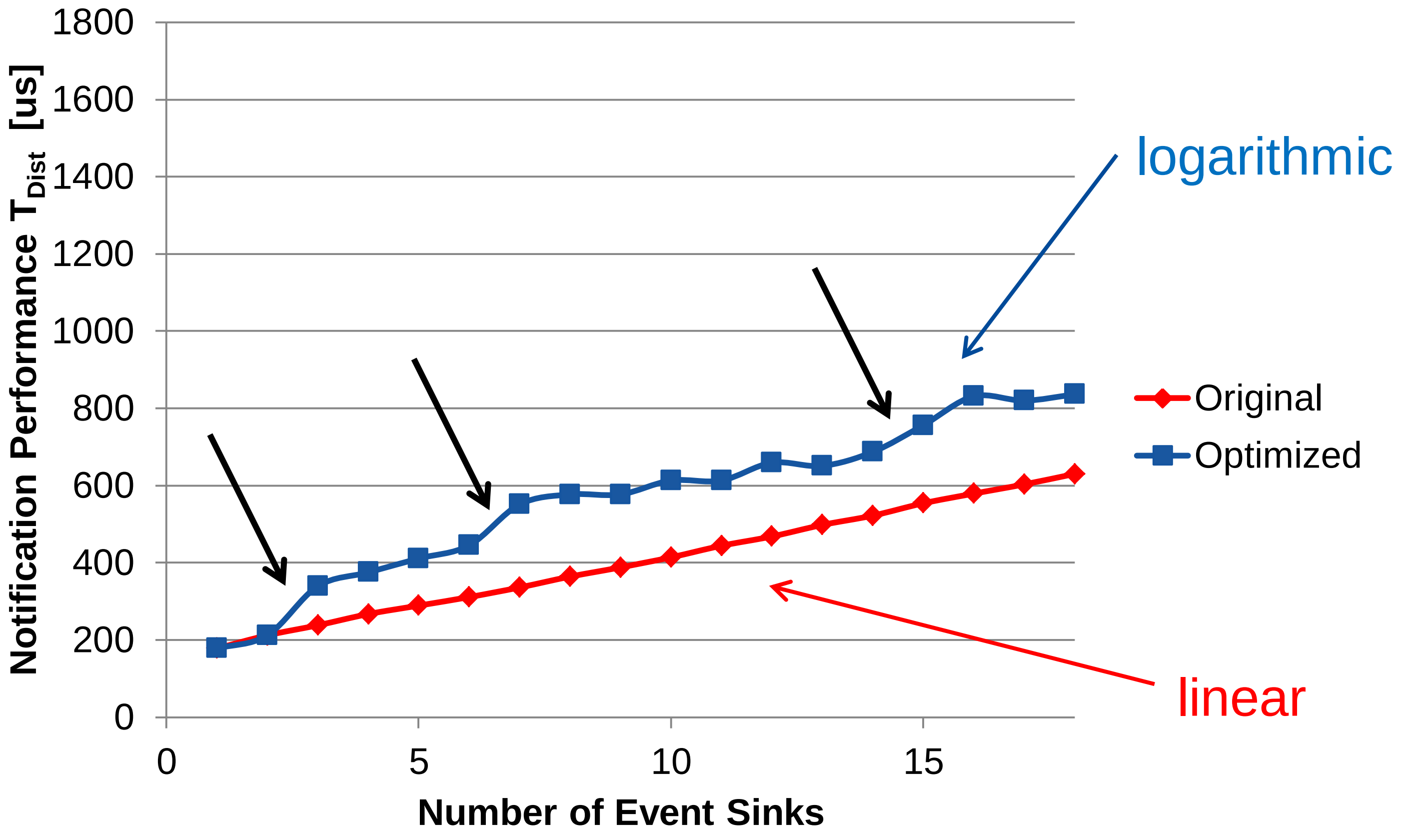
Prototype setup



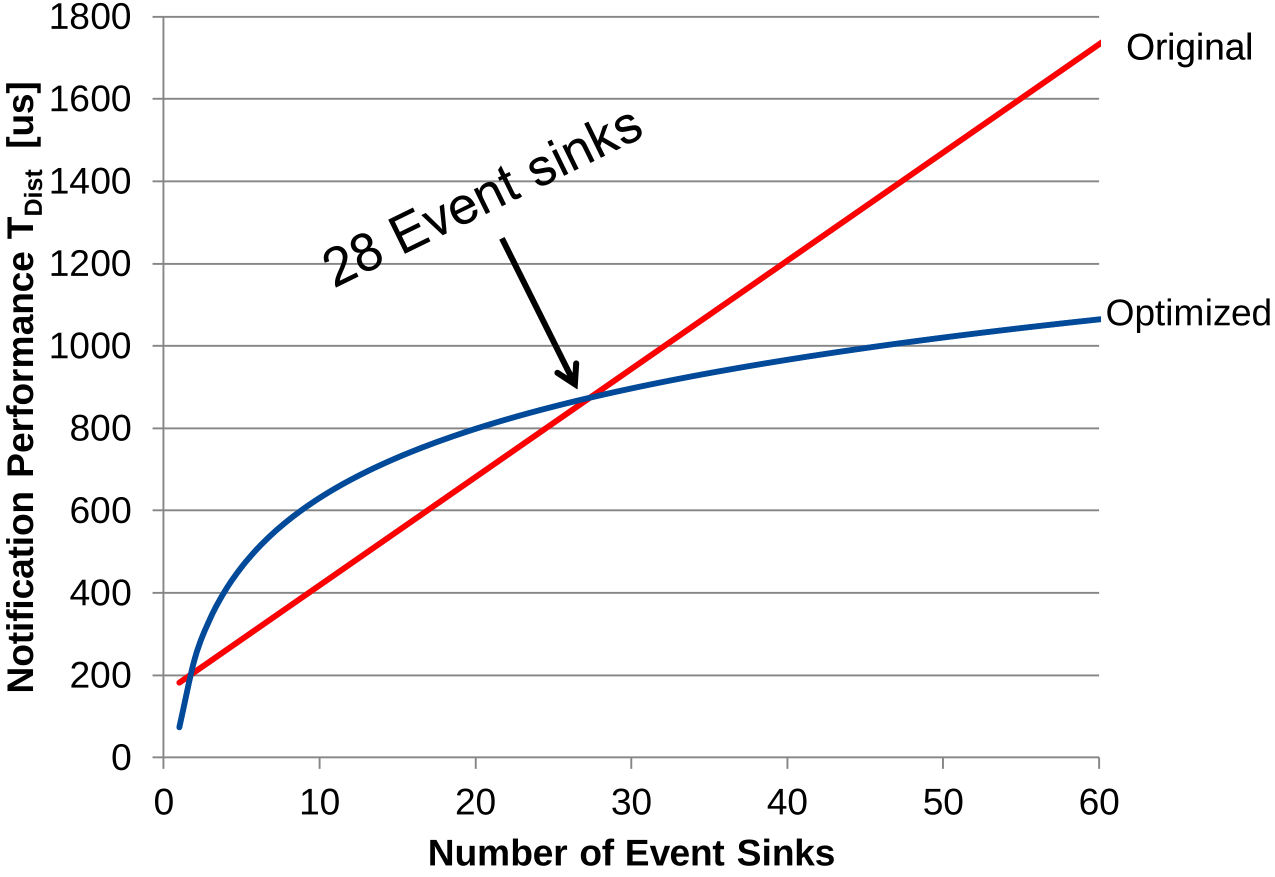
Results: Scenario 1 - Without ACK



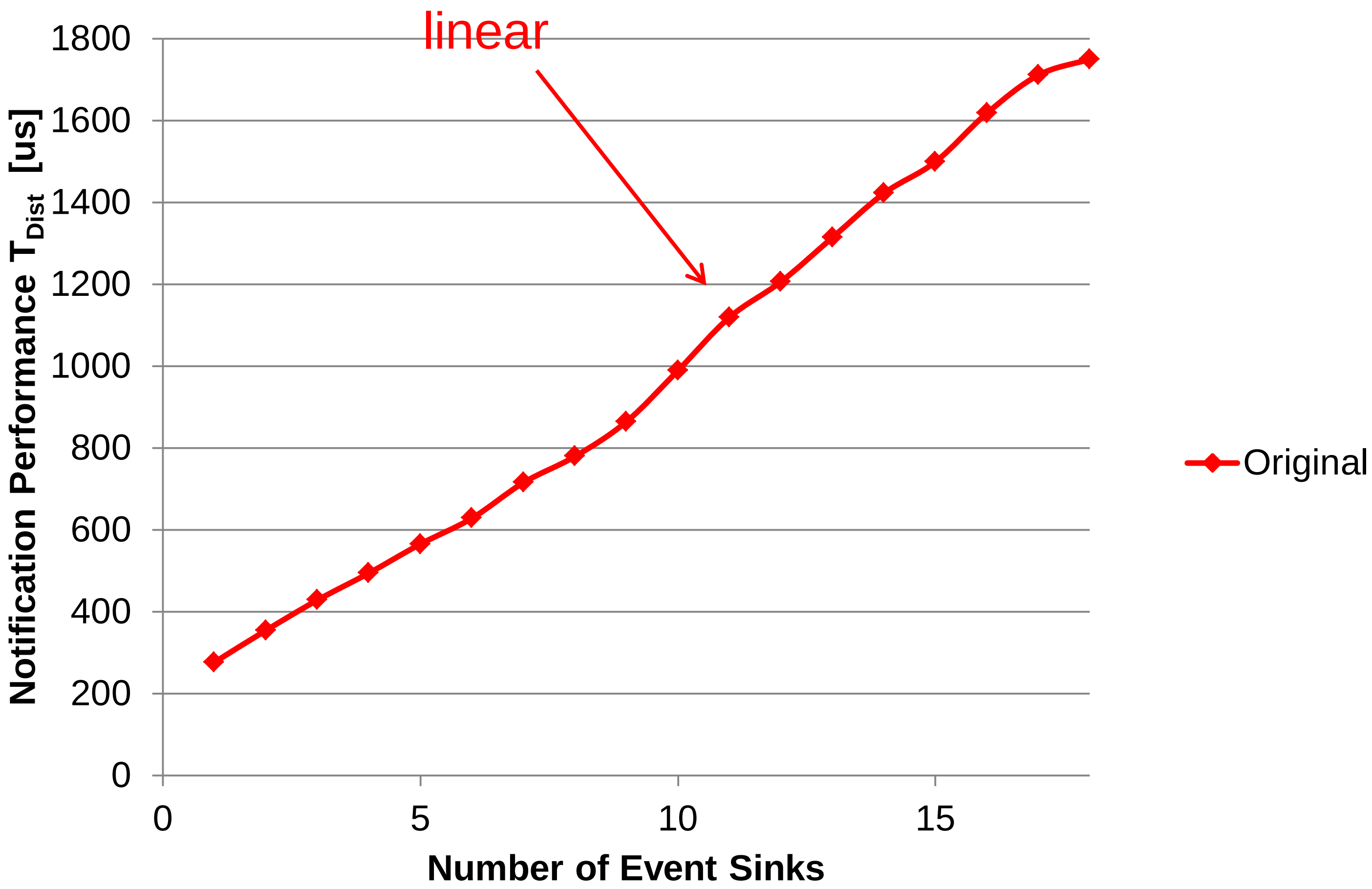
Results: Scenario 1 - Without ACK



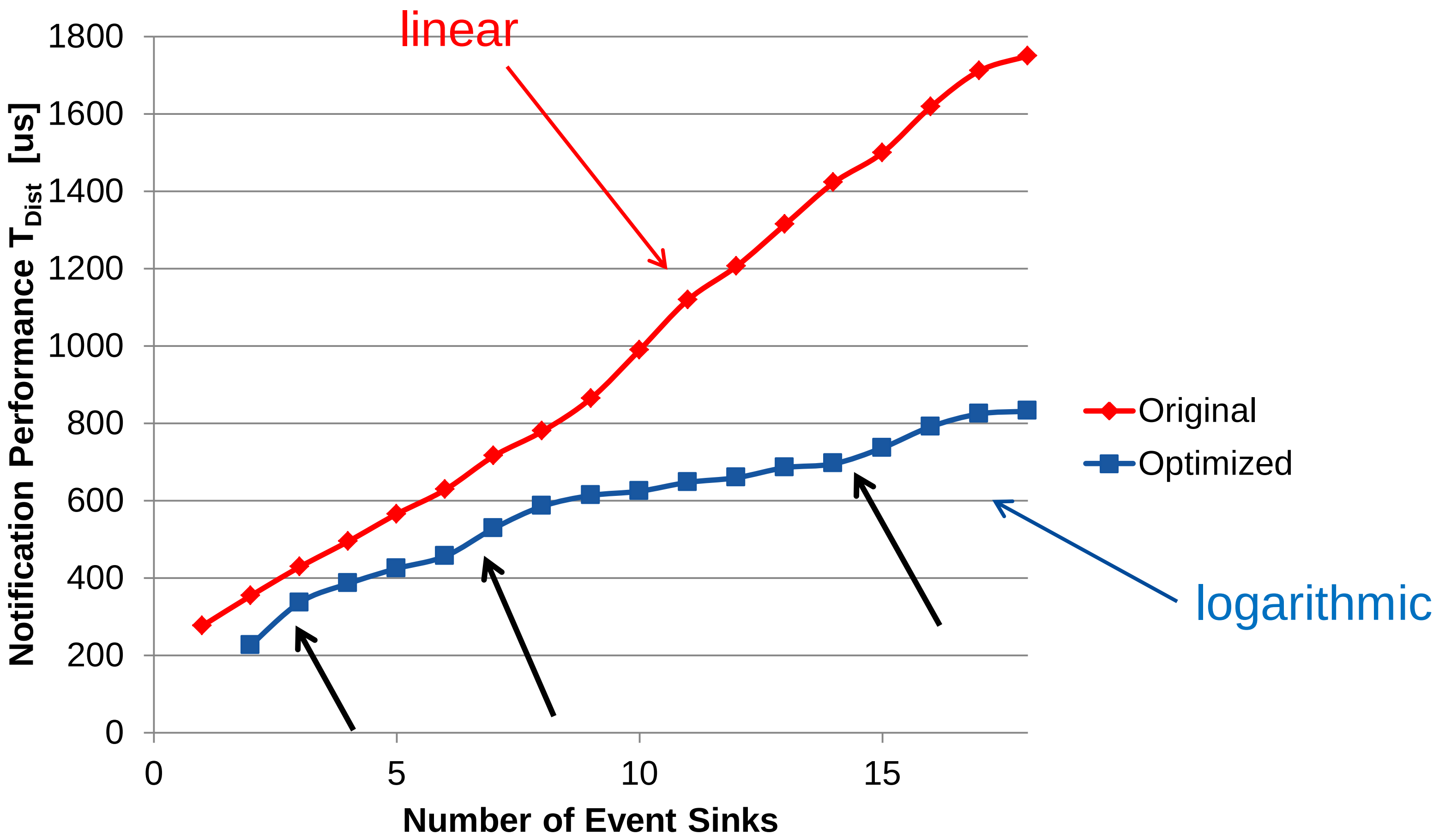
Results: Scenario 1 – Trend line without ACK



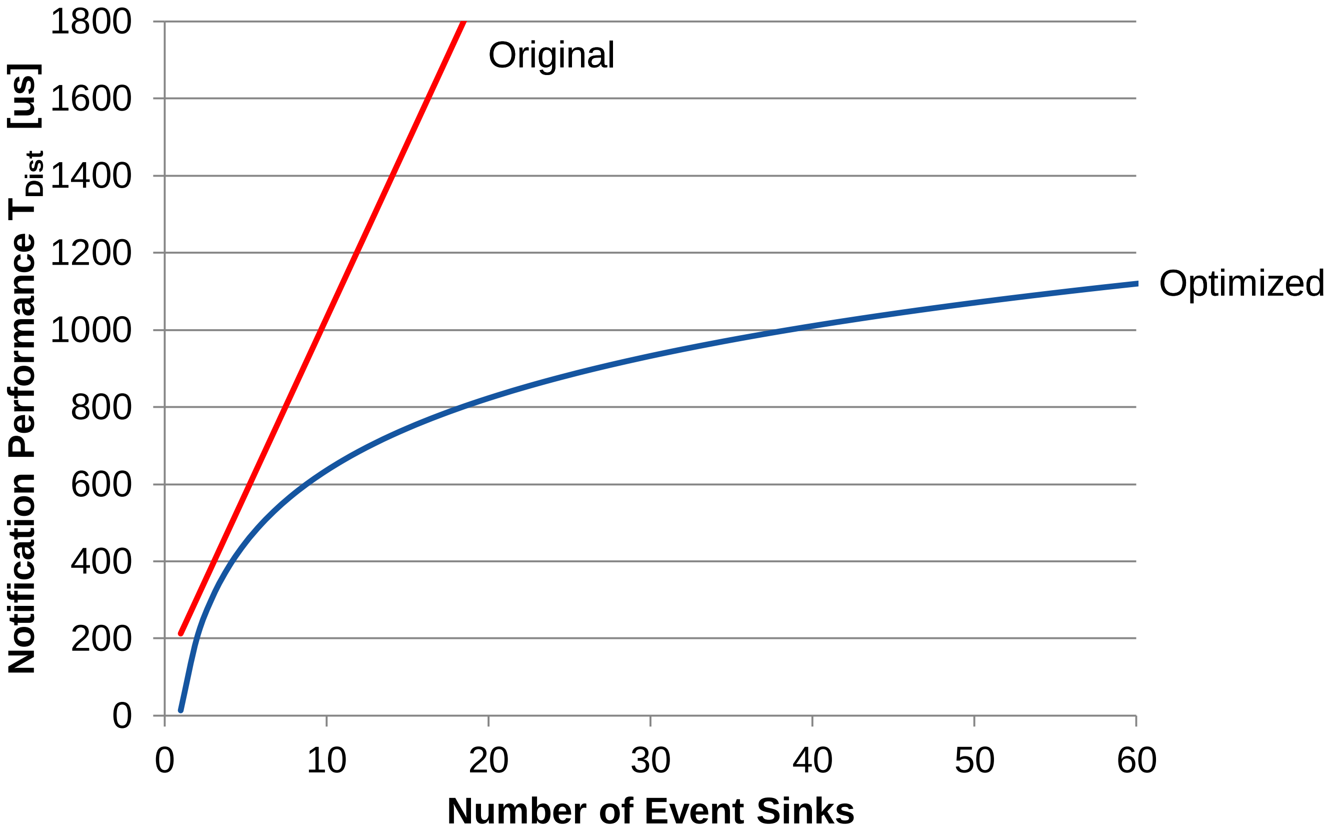
Results: Scenario 2 - With ACK



Results: Scenario 2 - With ACK



Results: Scenario 2 – Trend lines with ACK



Supported nodes at different automation scenarios

Profile	Human	Process
Cycle time	100 ms	10 ms
#Event sinks per event source	200	
Data amount per event source	34,240 Byte	
#Event sources	381	38
#Event sinks total	76200	7600

How to sort the lists?

Reliability

- Direct influence on who receives notification first

- 1st option:
  - First devices with high priority in timing
  - Bad results if a node close to source fails
    - As all following nodes are affected

- 2nd option:
  - First devices with highest reliability
  - Better as no following nodes are affected if a node fails
  - Better reliability and average timing

Summary

- Presentation of a scalable WS-Eventing approach
- Results from a prototype
  - An comparison with standard notification

- High efficiency due to utilization of EXI
- Two approaches sorting the lists

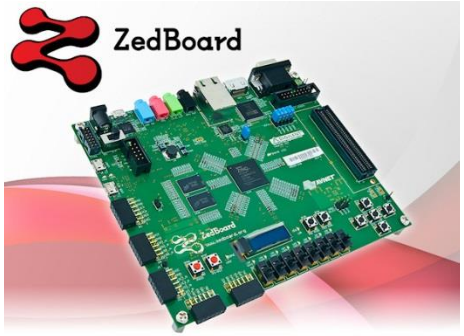
- Future Work:
  - Create simulation to verify behavior for thousand of nodes

Thank you for your attention!

Any questions?

Backup

Thread priorization

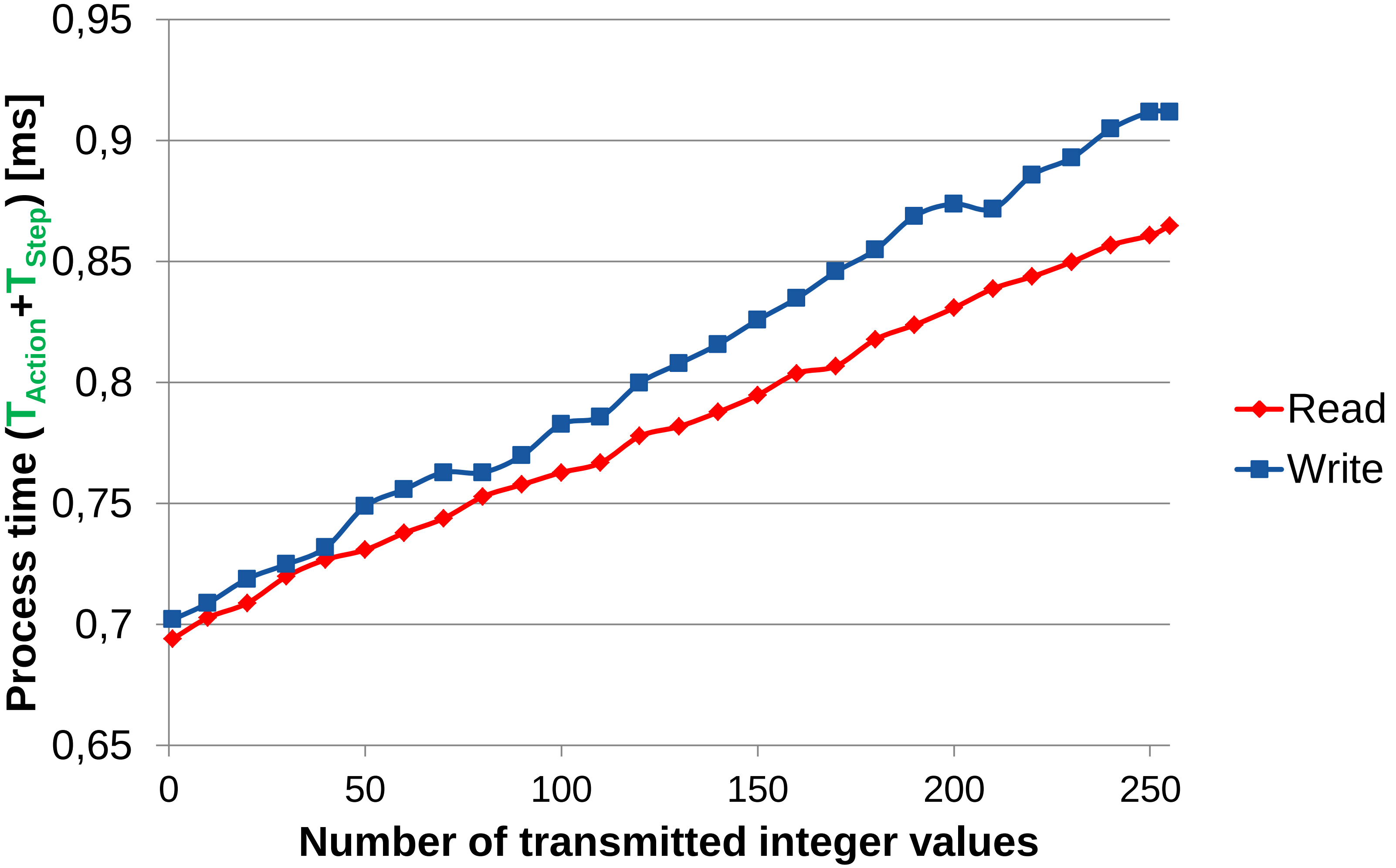


Thread
Main
External control
Kad communication
Search
Network
Maintenance
Idle

Program Flow

	Description
I. Initial Kad Operations	• Bootstrapping and Maintenance;
II. Search Tolerance Determination	• Determine the search tolerance ○ Max. one node for each hash value
III. Initial Synchronization	• First synchronization of the Kad network
IV. Application	• Application on top of HaRTKad
V. Maintenance	• Enable maintenance of Kad network ○ Also Bootstrapping
VI. Re-synchronization	• Re-synchronize the network ○ Due to clock drift of nodes

Data Transmission in HaRTKad



Supported Network Size by HaRTKad

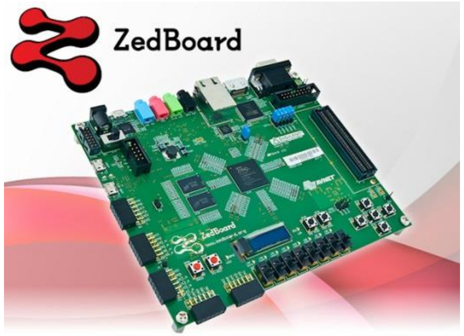
Scenario	Human	Process	Motion
Delivery constraint T_{Del} [ms]	100	10	1

$$Nodes = \frac{T_{Del}}{T_{Action} + \left(log_2(Nodes) * T_{Step}\right)}$$

$$Nodes_{Max} = \left\lfloor \frac{T_{Del} * log(2)}{T_{Step} * W\left(\frac{2^{\frac{T_{Action}}{T_{Step}}} * T_{Del} * log(2)}{T_{Step}}\right)} \right\rfloor$$

T_{Del}	Delivery constraint
$T_{\_Action}$	Time for interacting with node
$T_{\_Step}$	Time for further search step
W	Lambert W-function

HaRTKad Prototype



- Zedboard as target platform
  - ARM Dual Core @ 667 MHz
  - 1 Gbit/s Ethernet connection
- Allows to determine T_{Step} and T_{Action}
- Runs the Kad software (HaRTKad application)