Evaluating the Performance of DPWS in a Body Area Network

Dipl. Inf. Robert Hilbrich

Fraunhofer FIRST

Agenda

1. **Introduction**
   - Background
   - Problem description

2. **Experiments**
   - Setup
   - Hardware and software

3. **Results**
   - Time to reach "Operational Readiness" and Service Invocation
   - Effect of MTU-settings
   - DPWS's share of the total latency

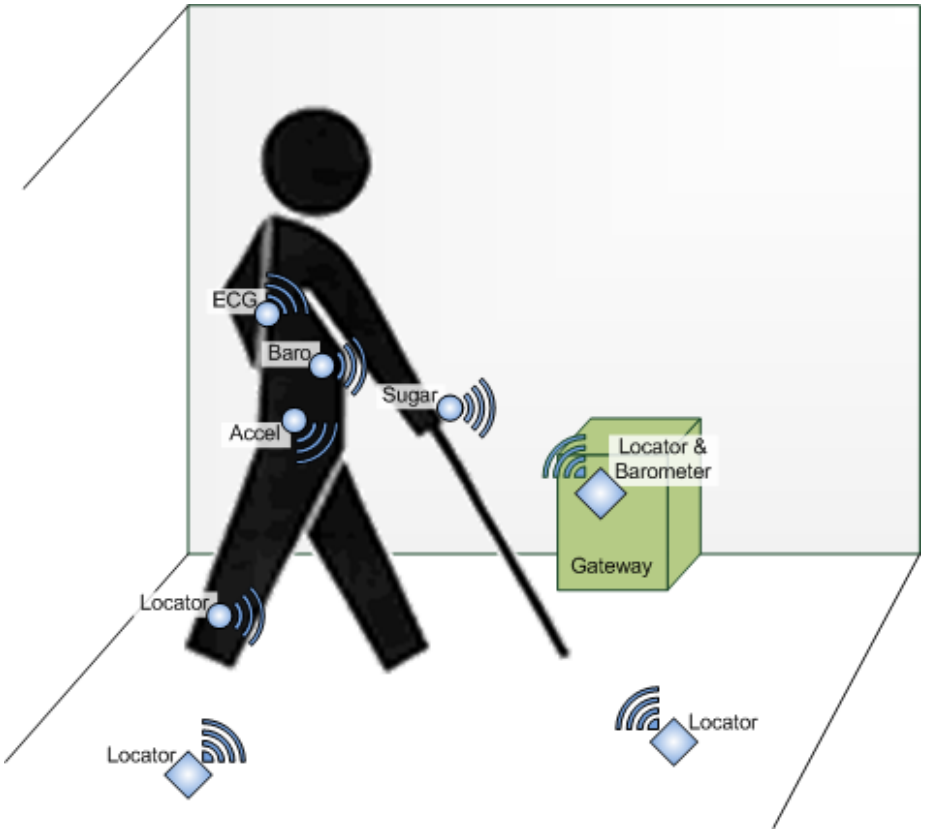
4. **Conclusion**

Evaluating the Performance of DPWS in a Body Area Network

INTRODUCTION

Introduction – Body Area Network

- "Live at home while being under continuous medical observation"
- Solution:
  wireless body area network
- Consisting of:
  - vital sensors
  - location sensors
  - gateway nodes

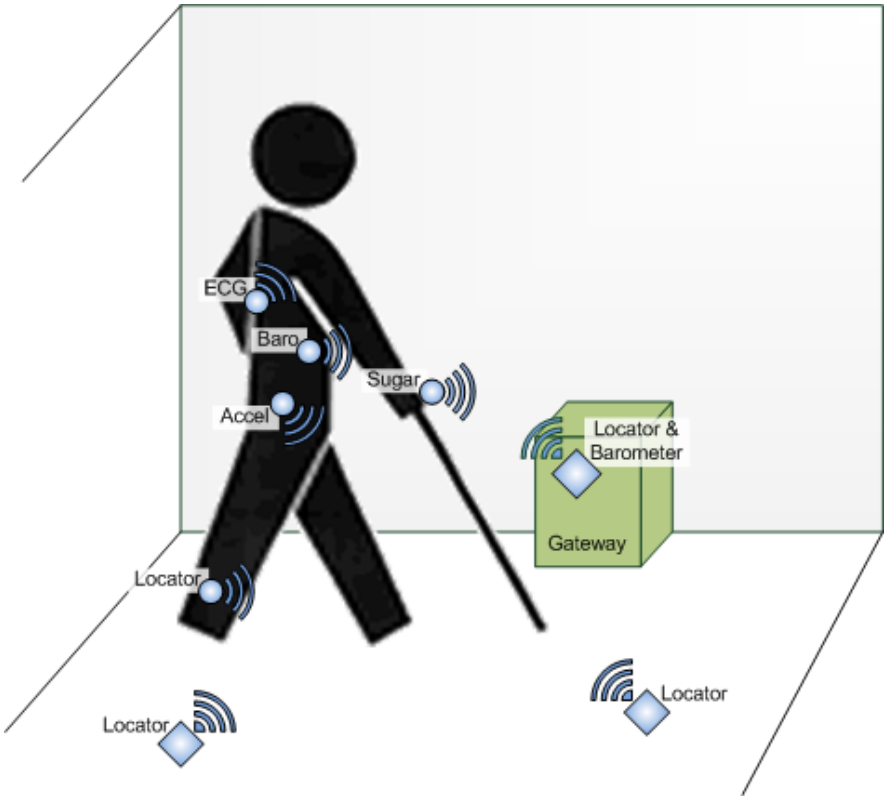


Fraunhofer FIRST

Introduction – Feasibility Study

- Major requirements:
  - Hide **hardware heterogeneity** to facilitate software development of a smart software layer
  - Support for **dynamic** discovery of nodes and a **dynamic** establishment of a workflow (zero configuration for ease of use)

- Our choice:
  - Web Service middleware
  - Devices Profile for Web Services **(DPWS)**



Fraunhofer FIRST

Introduction – Problem Description

- **Trade-Off** in software engineering for resource-constraint embedded systems:
  - Development effort at design time vs. processing effort at run-time

- Our decision for a middleware is influenced by:
  - Increasing availability of resources in embedded systems
  - General need for faster time-to-market by vendors

- Our **feasibility study**:
  - Analyze performance impact
  - Analyze bottle-necks (optimization potentials)

Fraunhofer FIRST

Evaluating the Performance of DPWS in a Body Area Network

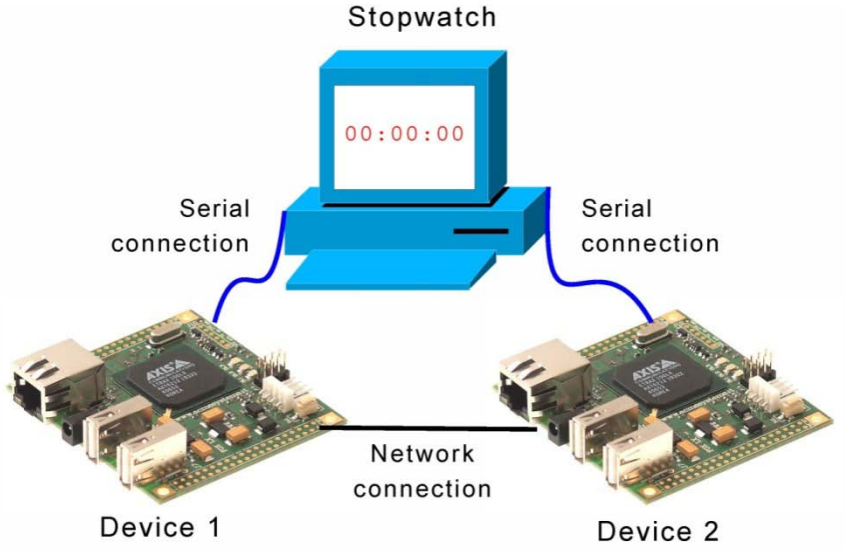
EXPERIMENTS

Experiments – Overview

- **Goal**
  - Measure latencies in the communication between DPWS-enabled devices

- **Use case** ("3-step interaction pattern")
  1. Discovery of a node
  2. Retrieval of metadata from the node
  3. Usage of a node

- **Setup**
  - Connect two embedded devices of the same type directly via Ethernet
  - Use a "Stopwatch" to measure latencies



Fraunhofer FIRST

Experiments – Software (DPWS frameworks)

– There are several DPWS frameworks / libraries available

– We chose:

  - DPWS implementation from the **Web Services for Devices Initiative (WS4D)**
    - Plug-in for gSOAP
    - Open Source, GPLv2
    - Development in C/C++

  - DPWS implementation in the **Microsoft .NET Micro Framework (NETMF)**
    - Assembly/library in the NETMF
    - Proprietary License
    - Development in C#

Experiments – Hardware (embedded devices)

- **FOX Board LX832**
  - ETRAX CPU 100MHz, 32MB RAM, 8MB Flash, 100 Mbit/s Ethernet, Linux 2.6

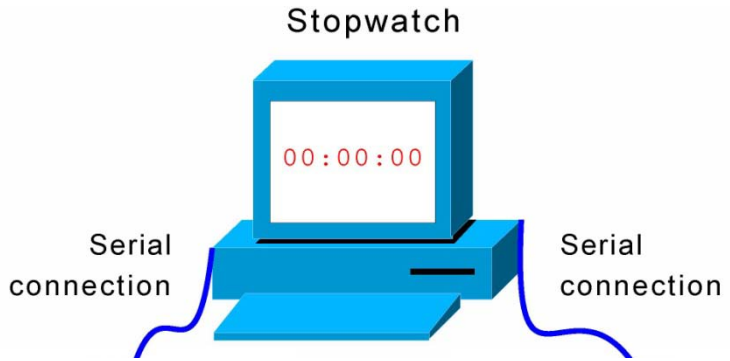
- **Emtrion HiCO.ARM9**
  - ARM9 CPU 180MHz, 64MB RAM, 32MB Flash, 100 Mbit/s Ethernet, Linux 2.6, .NETMF

- **Tahoe Development Platform I**
  - ARM9 CPU 100MHz, 8MB RAM, 4MB Flash, 10 Mbit/s Ethernet, .NETMF

- To compare: **Laptops**, 1GHz – 2.4GHz, 2GB RAM, 100MBit/s – 1GBit/s Ethernet, Linux 2.6

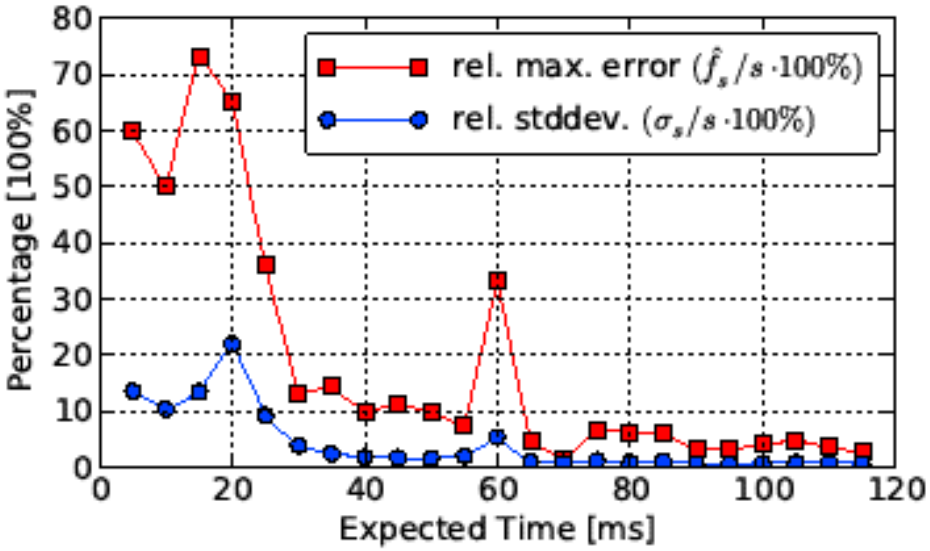
Experiments - Stopwatch

- Need fine grained measurement of the latency in the communication between devices
- No clock synchronization
- Low overhead



- Developed a **stopwatch** which is triggered via serial inputs
- Empirically determined the impact of measurement errors

- **Time delays >65ms** can be measured with an accuracy of:
  - **+/-1% typically**
  - +/-10% worst case



Fraunhofer FIRST

Evaluating the Performance of DPWS in a Body Area Network

RESULTS

Results – Time to reach "Operational Readiness"

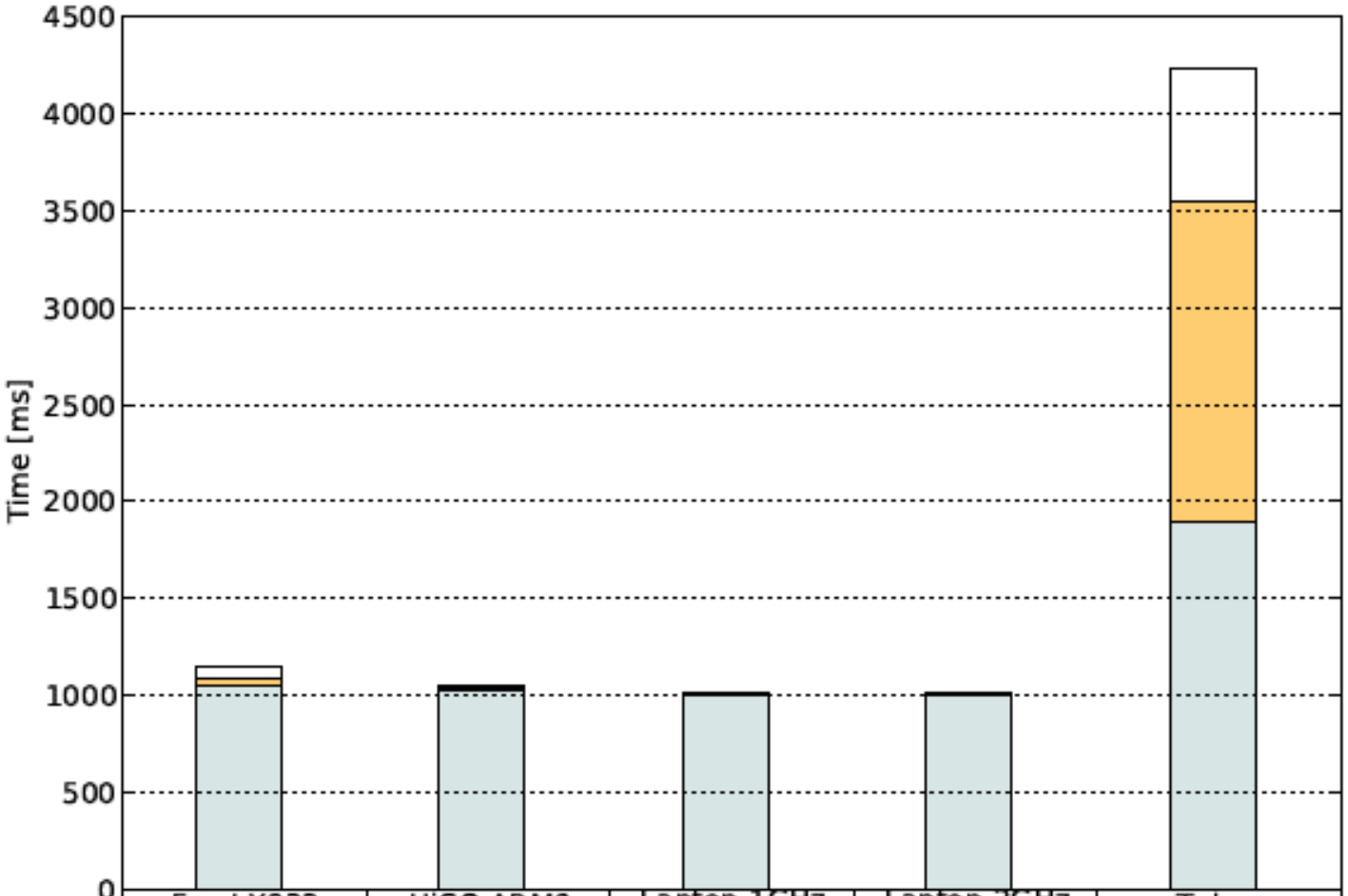
– Operational Readiness (after boot-up):

1. **Probe** the environment for nodes of a „type“ (IP multicast)
2. **Resolve** names to IP addresses (IP multicast)
3. **Retrieve** metadata (TCP/IP transfer)

– Nodes are now ready to collaborate

Fraunhofer FIRST

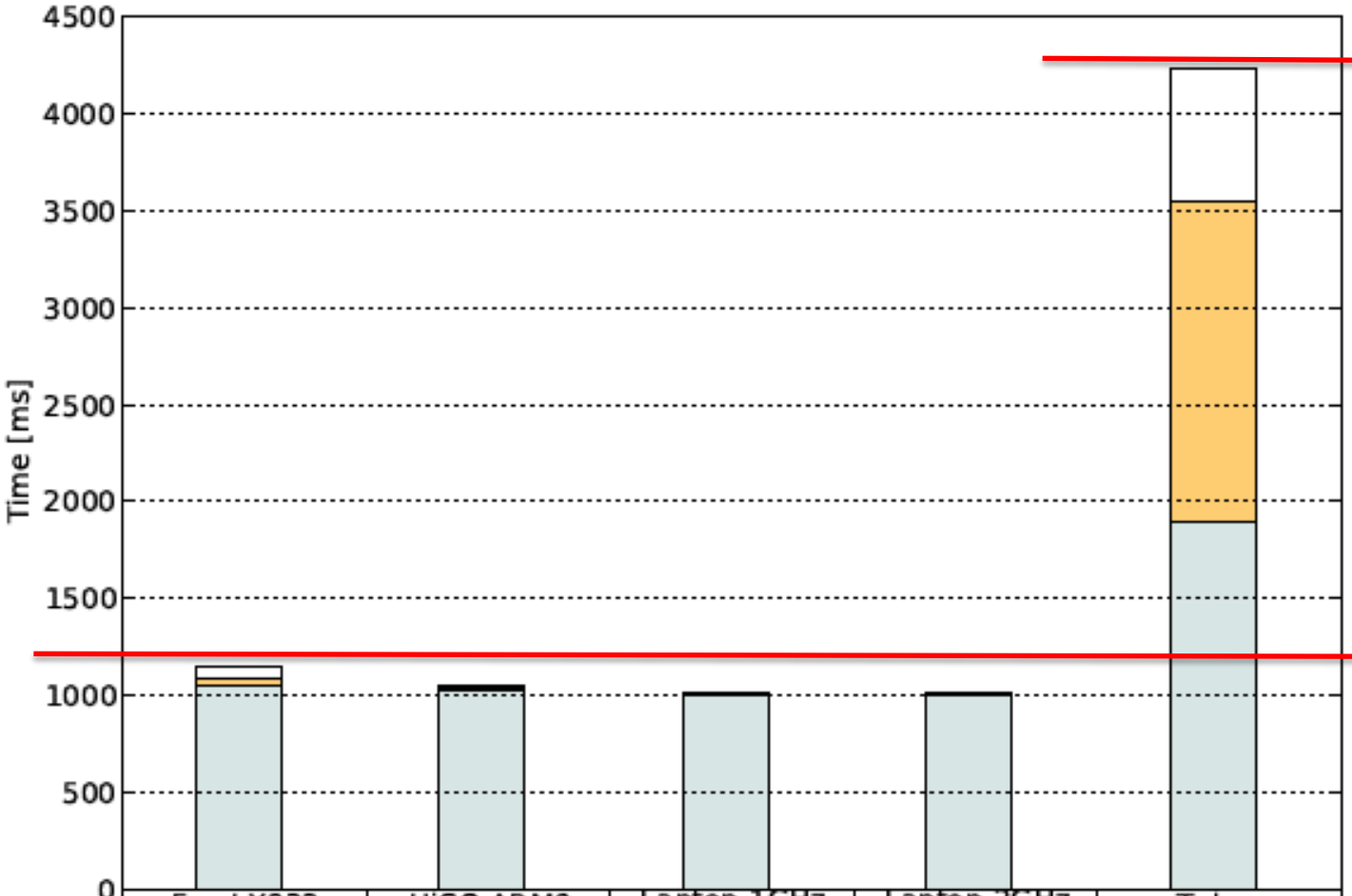
Results – Time to reach "Operational Readiness"



	Fox LX832	HiCO.ARM9	Laptop-1GHz	Laptop-2GHz	Tahoe
Probe	1047.24	1021.93	1002.44	1001.43	1890.84
Resolve	43.74	14.99	12.31	6.44	1663.45
Get	52.05	17.97	2.32	1.11	680.81

Fraunhofer FIRST

Results – Time to reach "Operational Readiness"



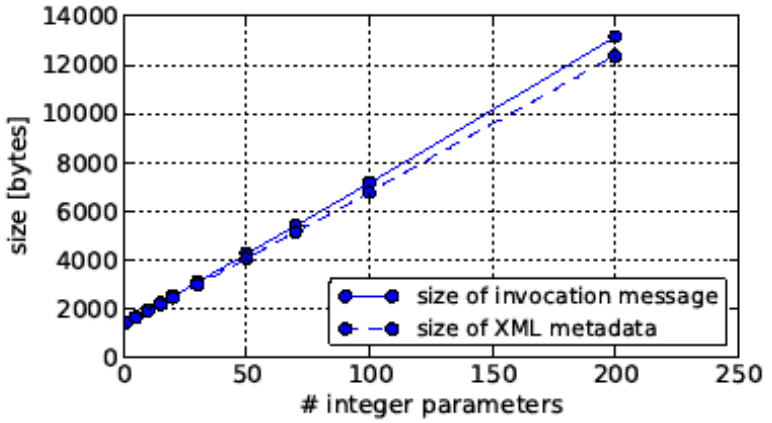
	Fox LX832	HiCO.ARM9	Laptop-1GHz	Laptop-2GHz	Tahoe
Probe	1047.24	1021.93	1002.44	1001.43	1890.84
Resolve	43.74	14.99	12.31	6.44	1663.45
Get	52.05	17.97	2.32	1.11	680.81

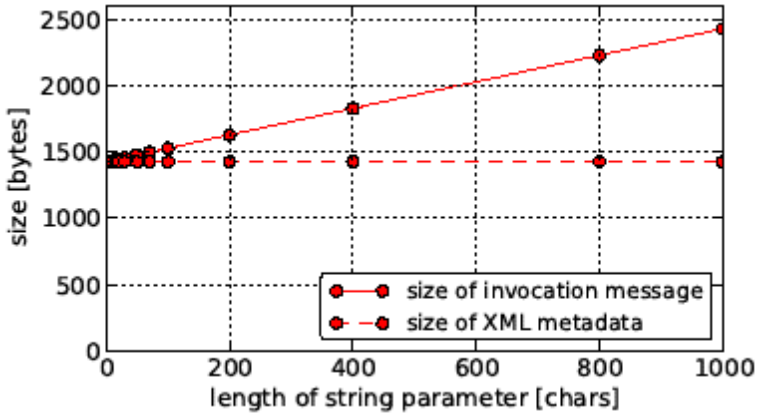
Fraunhofer FIRST

Results – Service Invocation

- Latency for service invocation depends on
  - **Processing effort** for XML data
    → **(De-)Serialization** effort
    → **„Structuredness“** of parameters

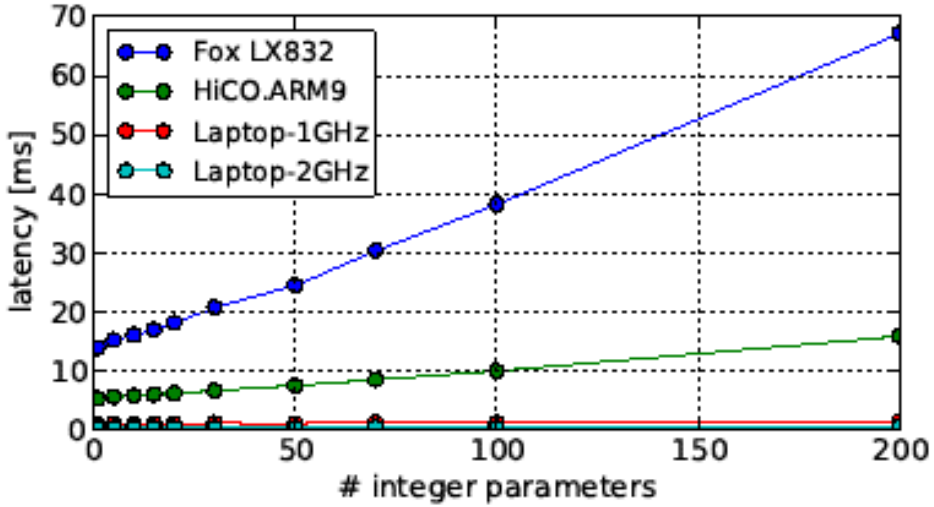
- We look at two extremes:
  - Heavily structured parameters (top)
    - Variable # of integer parameters
    - Ratio:
      „real“-data / XML-metadata → low
  - Lightly structured parameters (bottom)
    - single string parameter w. variable length
    - Ratio:
      „real“-data / XML-metadata → high

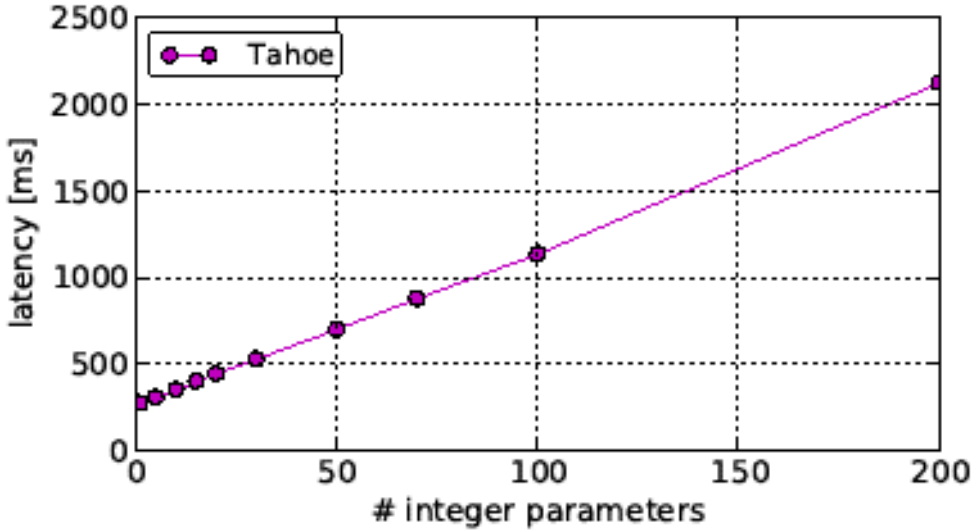




Fraunhofer FIRST

Results – Service Invocation – Heavily Structured Params.

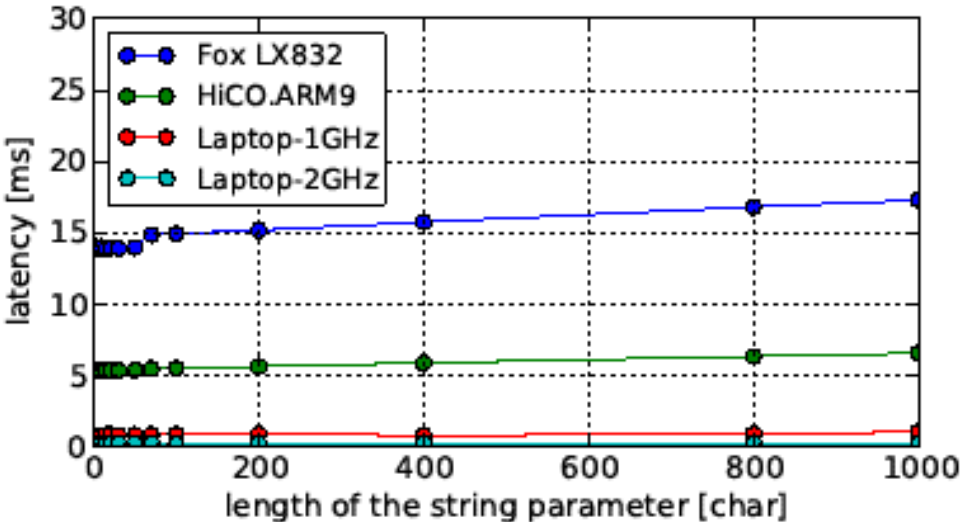


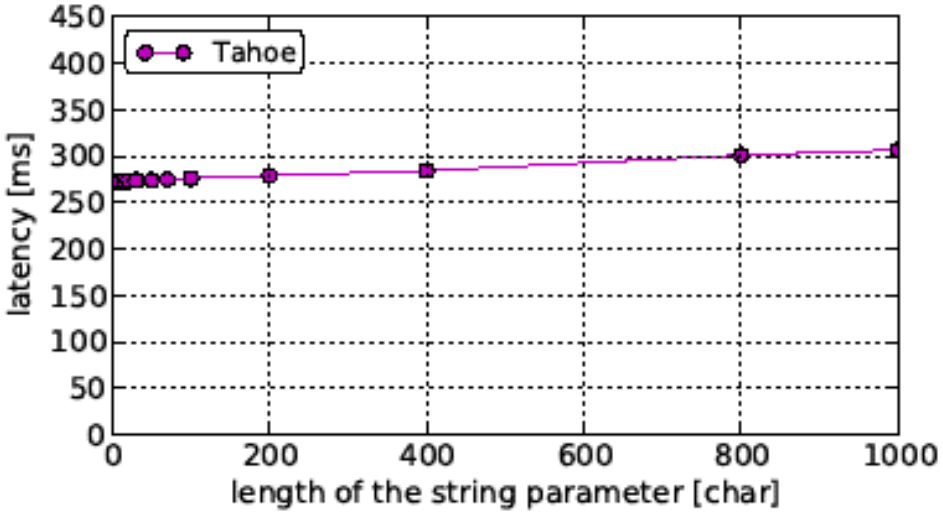


– Absolute latencies of a **one-way** service invocation with a varying amount of integer parameters

Fraunhofer FIRST

Results – Service Invocation – Lightly Structured Params.



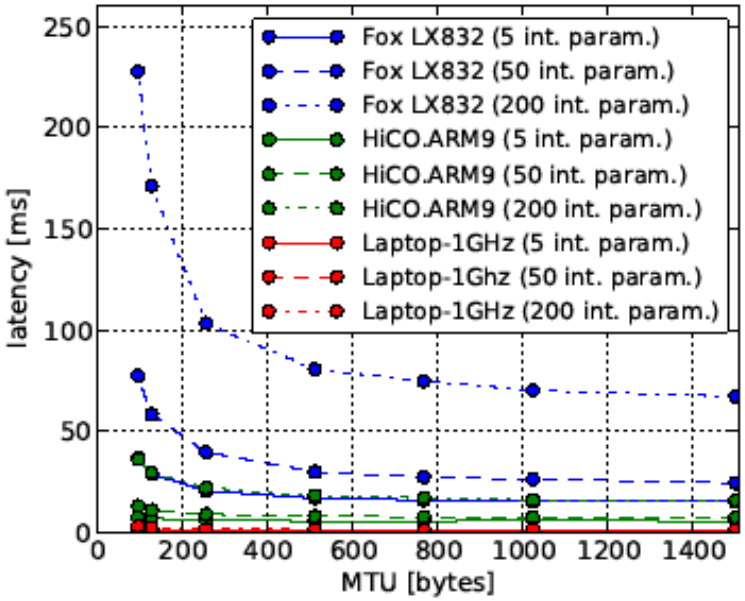


- Absolute latencies of a **one-way** service invocation with a single string parameters (variable length)
- Structuredness of call-parameters has a significant influence on performance!

Fraunhofer FIRST

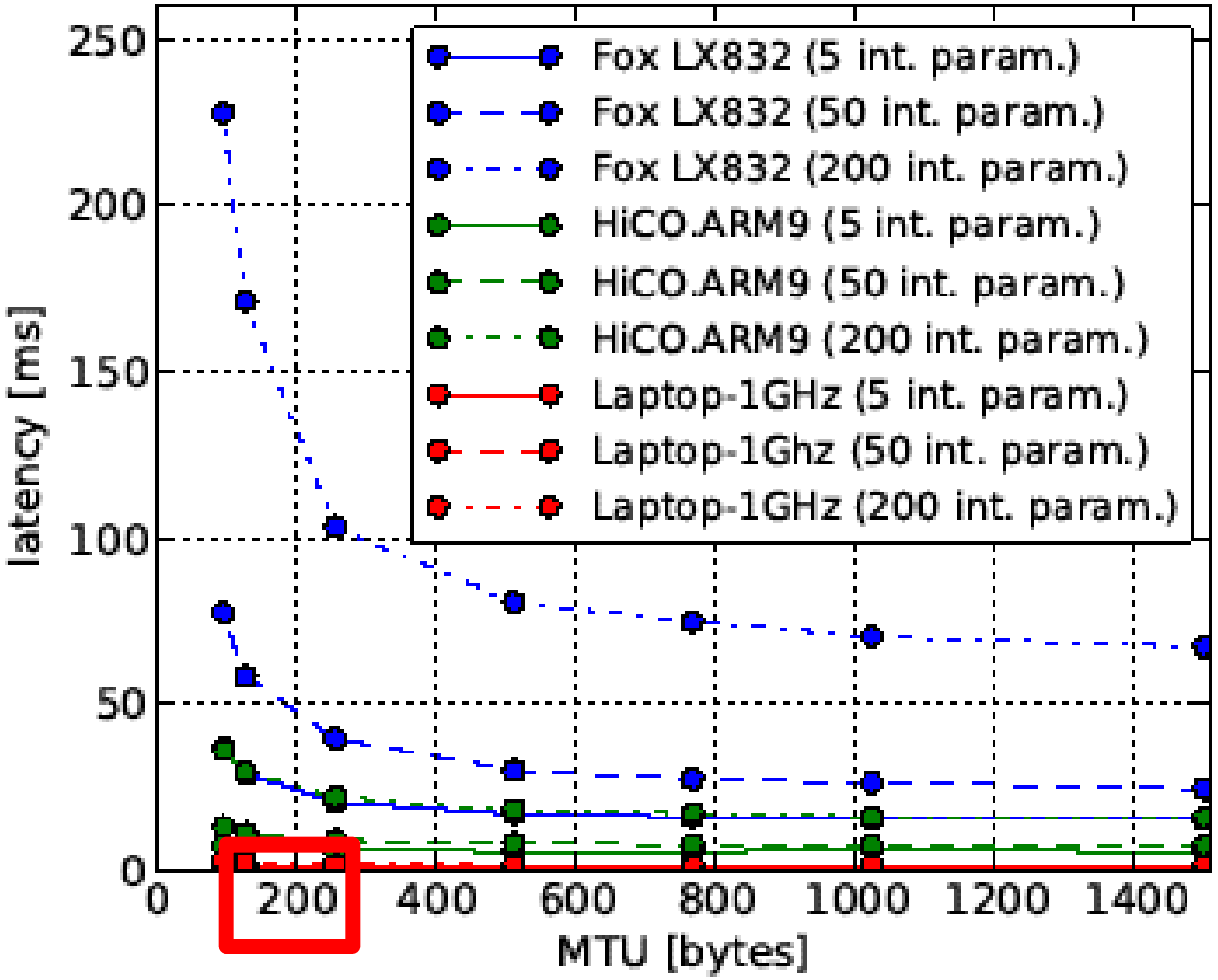
Results – Influence of MTU setting on Service Invocation

- Maximum Transfer Unit (MTU) =
  upper bound on the size of a **fragmented IP packet**
- Low power wireless sensor networks feature lower MTU settings
  (6LoWPAN: ca. 150bytes)
- Web Service data (XML) is typically larger than 1500bytes



Fraunhofer FIRST

Results – Influence of MTU setting on Service Invocation



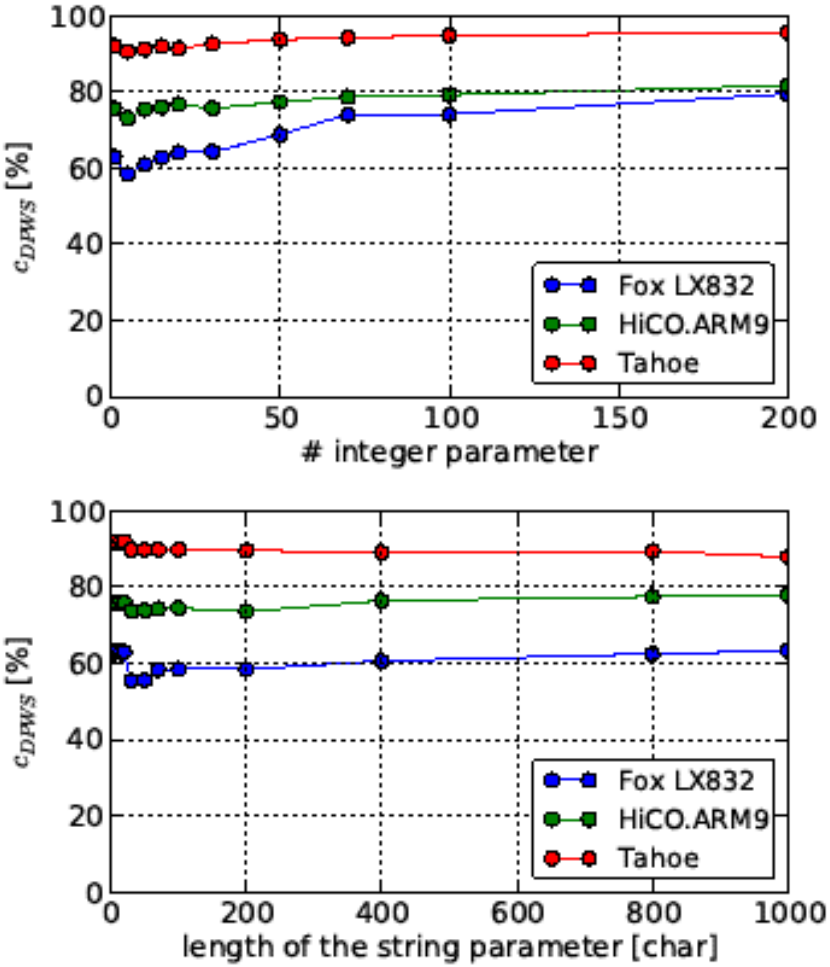
Fraunhofer FIRST

Results – DPWS's share of total latency

- **Total latency =**
  - packet transmission,
  - packet encoding,
  - processing in the operating system
  - DPWS layer

- How much of it is **caused by the DPWS** layer for (de-) serialization? (→ c_DPWS)

- Results for **one-way service invocation** with heavily and lightly structured parameters



Fraunhofer FIRST

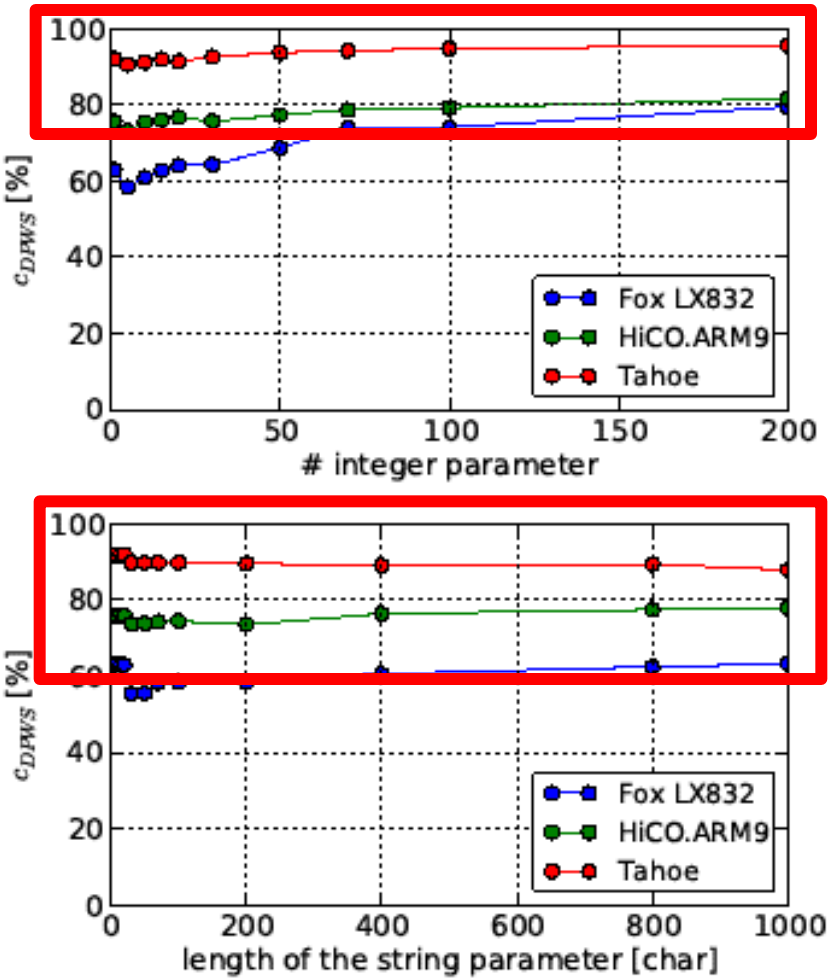
Results – DPWS's share of total latency

- **Total latency =**
  - packet transmission,
  - packet encoding,
  - processing in the operating system
  - DPWS layer

- How much of it is **caused by the DPWS** layer for (de-) serialization? (→ approx. **60-95%!**)

- Results for **one-way service invocation** with heavily and lightly structured parameters

- Boards benefit significantly from **faster CPU** instead of faster network



Fraunhofer FIRST

Evaluating the Performance of DPWS in a Body Area Network

CONCLUSION

Conclusion

- DPWS on embedded devices is a promising approach to achieve plug-and-play behavior with (almost) zero configuration
- Drawback: increased processing effort and prolonged latencies

- Reaching "**operational readiness**" takes about 1s
- Latency for a service invocation is significantly influenced by the "**structuredness**" of the parameters (API design is significant!)
- Majority of the total latency is "produced" within the DPWS layer (→ devices benefit significantly from faster processor!)

- Is DPWS **a viable choice** for a Body Area Network?
  - Well suited for heterogeneous devices with rare transmissions and infrequent topology changes
  - Unsuitable for rapid topology changes, low latency requirements and heavily structured data