Technische Universität München

Chair for Network Architectures and Services
Prof. Dr.-Ing. Georg Carle

Corinna Schmitt
Contact: schmitt@net.in.tum.de · http://www.net.in.tum.de

TinyIPFIX for Home Network Application

Motivation

Current situation in home monitoring scenarios:

- Manual input for control and management issues is needed.
  → Reduction of manual inputs by implementation of an **autonomic network**.
- Monitoring networks usually consists of Wireless Sensor Networks (WSNs).
  → Challenges caused by **limited resources** (e.g. energy, memory)
- In WSNs the nodes report autonomous in time intervals
  → **PUSH-protocol** is a good solution to optimize transmissions.

Aims:

- Saving resources (e.g. memory, energy)
- Reduction of energy consuming processes (e.g. transmissions)
  → Ensure long life of the WSN

Solution:

- Implementation of an efficient data transmission protocol called **TinyIPFIX** based on the IP Flow Information Export (IPFIX) protocol
- Combination with aggregation functionality

Design Decisions

1. IP-communication within the WSN:

- Integration of 6LoWPAN
- Characterization:
  - NanoStack size only 4kb (ZigBee Stack size 8kb)
  - Provision for data fragmentation and header compression mechanisms
  - Payload size up to 110 Bytes

2. Modified packet structure for transmissions:

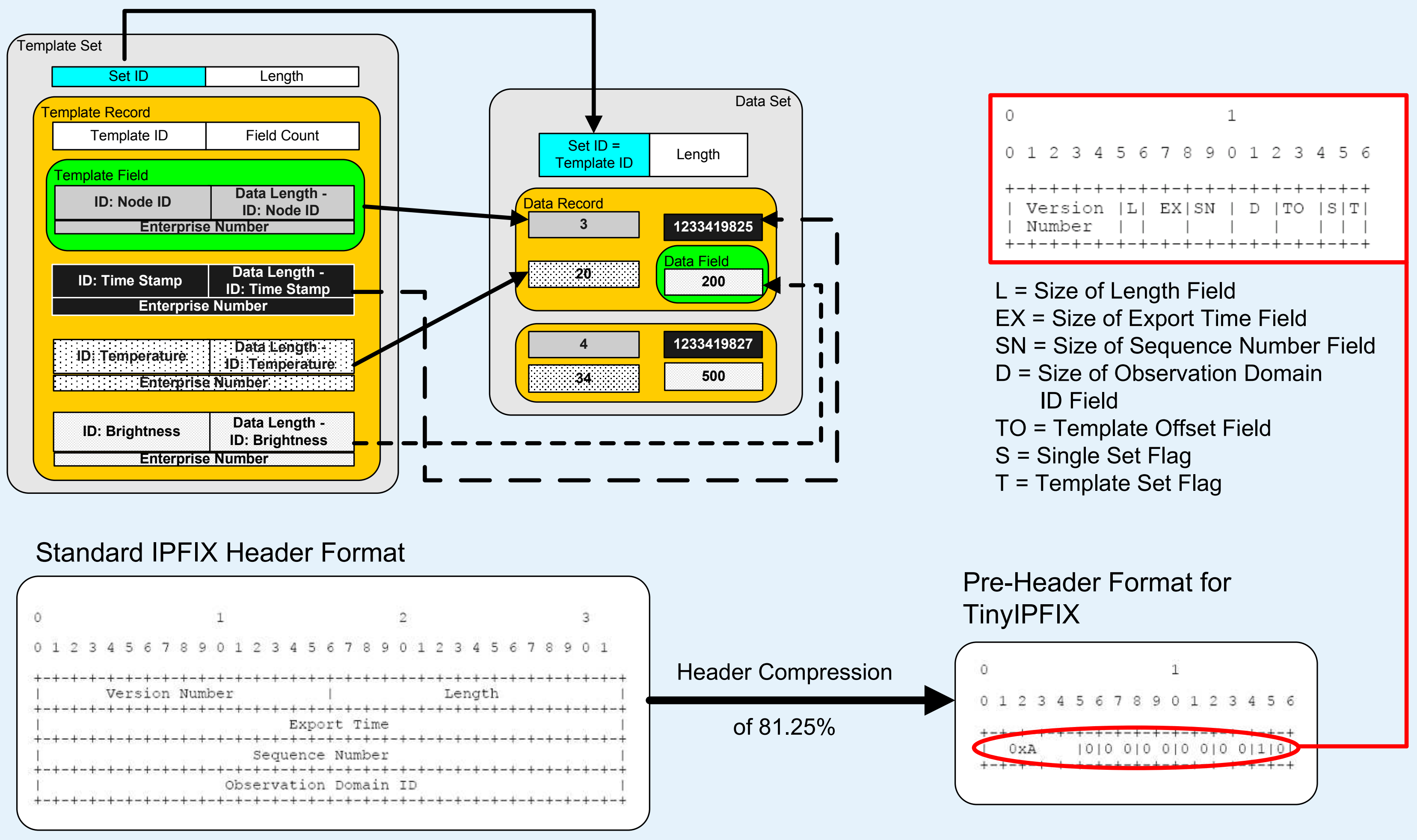
- Implementation of TinyIPFIX
- Characterization:
  - PUSH-protocol: Exporter periodically transmits data to Collector
  - Separation between meta information and data during transmission
  - Support for header compression and aggregation mechanisms
  - Reduction of transmission size

Overview of occuring events if a new node A enters the WSN:

1. New node A (= **Exporter**) boots up.
2. Node A transmits its Template which is stored by the other nodes within the WSN.
3. Node A starts its measurements and transmits its data refering to its Template.
4. Receiving nodes (= **Collectors**) decode the data using the announced Template.

TinyIPFIX

Structure of a Template Set and a Data Set showing decoding using pointers.



```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version |L| EX|SN | D |TO |S|T|
| Number  | |   |   |   |   | | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

L = Size of Length Field
EX = Size of Export Time Field
SN = Size of Sequence Number Field
D = Size of Observation Domain ID Field
TO = Template Offset Field
S = Single Set Flag
T = Template Set Flag

Standard IPFIX Header Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Version Number          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Export Time                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Sequence Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Observation Domain ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Pre-Header Format for TinyIPFIX

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  0xA    |0|0 0|0 0|0 0|0 0|1|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Demonstrator – Phase 1: Preparation

Step 1:
Program each sensor node with individual settings

- Connect programming board and sensor node to the USB-Port of the PC
- Install program using the command
  make iris install.1024 mib510./dev/ttyUSB2

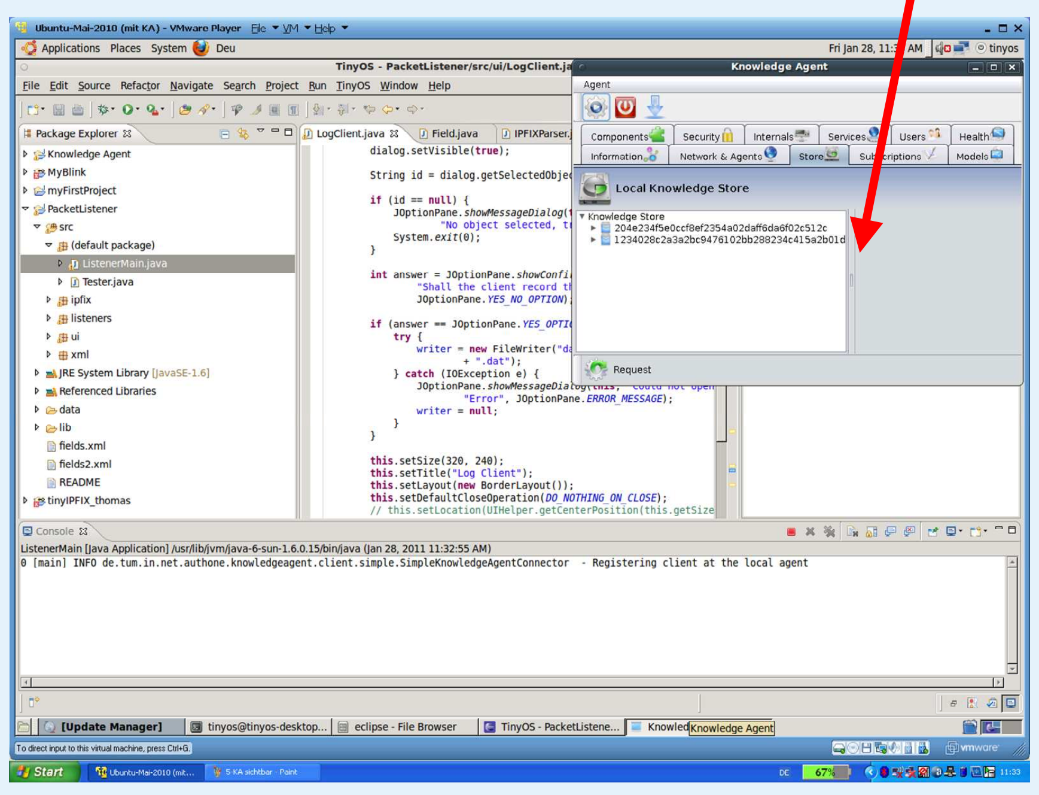
sensor node platform
adviced ID

Step 2:
Start Tunnel

Consequences:

- Tunnel cuts off not needed headers.
- Fired signals are displayed.
- Transmitted messages are displayed.

Step 3:
Start Knowledge Agent



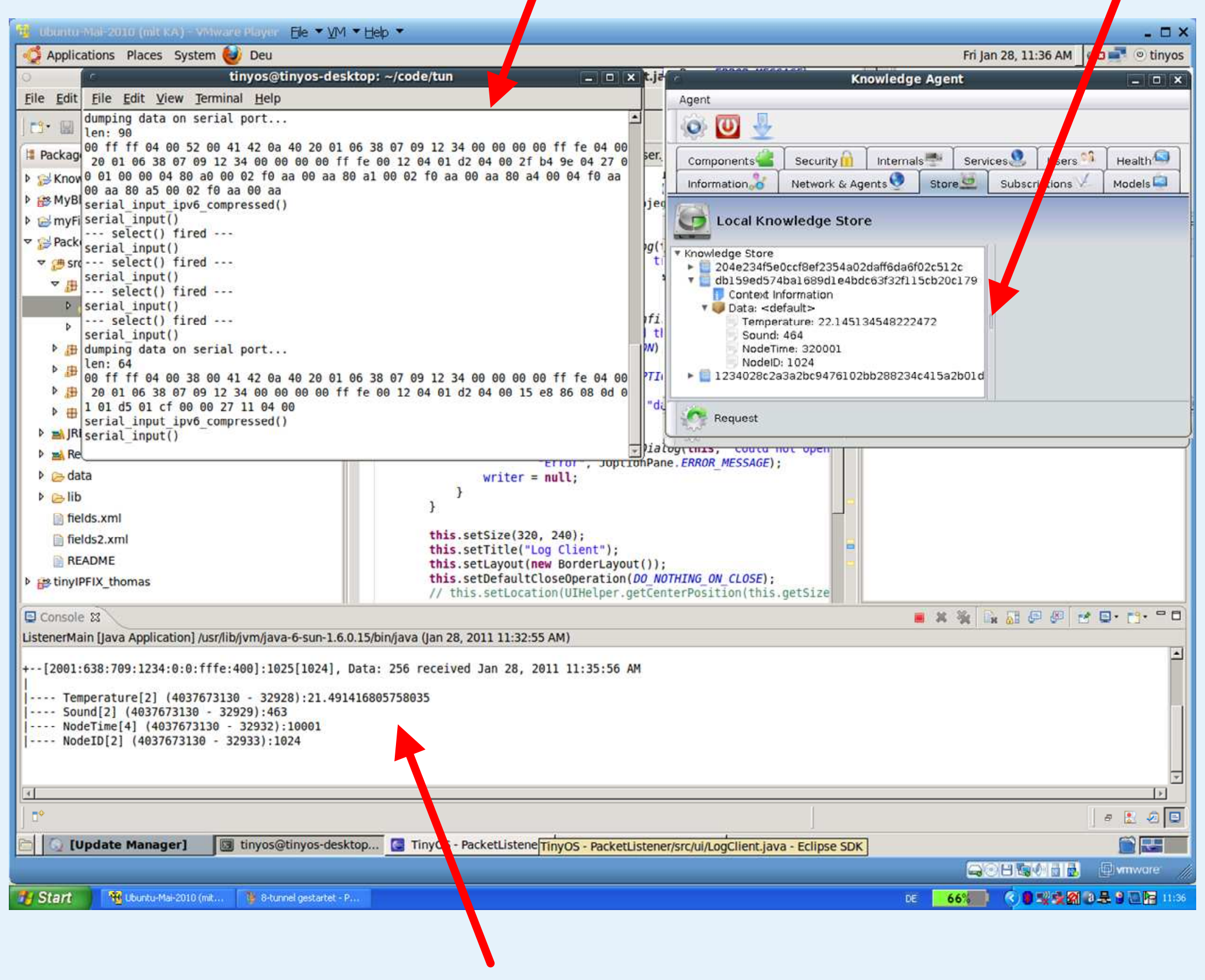
Demonstrator – Phase 2: Running WSN

Step 4:
Start sensor nodes
→ Template announcement
→ Measures envionmental Data
→ Transmitts Data

Consequences:

- Knowledge Agent registers sensor node.
- Knowledge Agent advices individual ID to each sensor node.
- Knowledge Agent receives pakets (Template/Data).

Step 5:
Start LogClient
→ Select sensor node for recording
→ Record data is stored in txt-file



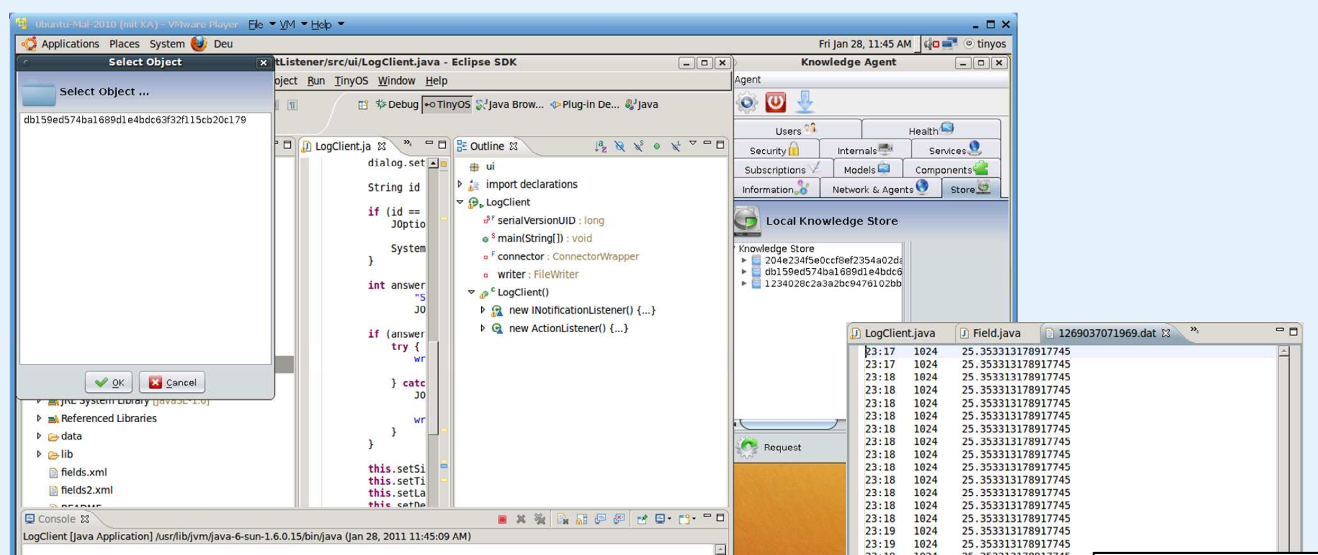
```
--- select() fired ---
serial_input()
--- select() fired ---
serial_input()
dumping data on serial port...
len: 64
00 ff ff 04 00 38 00 41 42 0a 40 20 01 06 38 07 09 12 34 00 00 00 00 ff fe 04 00
 20 01 06 38 07 09 12 34 00 00 00 00 ff fe 00 12 04 01 d2 04 00 15 e8 86 08 0d 0
1 01 d5 01 cf 00 00 27 11 04 00
serial_input_ipv6_compressed()
serial_input()
```

```
Context Information
Data: <default>
  Temperature: 22.145134548222472
  Sound: 464
  NodeTime: 320001
  NodeID: 1024
1234028c2a3a2bc9476102bb288234c415a2b01d
```

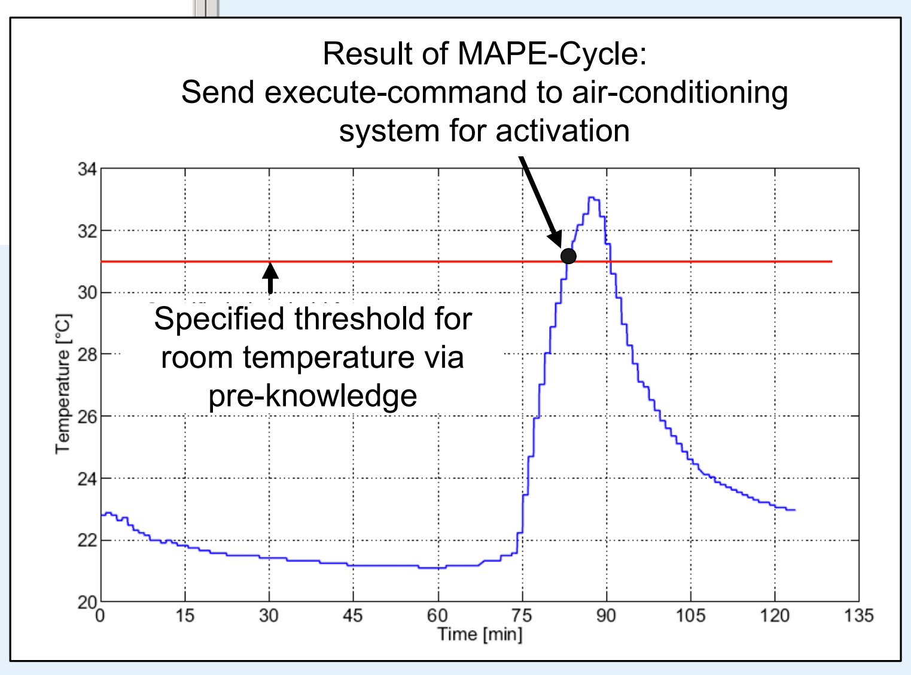
```
ListenerMain [Java Application] /usr/lib/jvm/java-6-sun-1.6.0.15/bin/java (Jan 28, 2011 11:32:55 AM)

+--[2001:638:709:1234:0:0:fffe:400]:1025[1024], Data: 256 received Jan 28, 2011 11:35:56 AM
|
|---- Temperature[2] (4037673130 - 32928):21.491416805758035
|---- Sound[2] (4037673130 - 32929):463
|---- NodeTime[4] (4037673130 - 32932):10001
|---- NodeID[2] (4037673130 - 32933):1024
```

Demonstrator – Phase 3: Log Data



Step 6:
Start LogClient
→ Select sensor node for recording
→ Record data is stored in txt-file



Consequences:

- Each sensor node is recorded individually.
- Data is stored in separated files.
- Data can be transmitted to analysis tools.
- Data is transferred to MAPE-Cycle.