6lowapp
Internet Draft
Intended status: Informational
Expires: March 2010

C.Schmitt
L.Braun
G.Carle
TU Muenchen
October 15, 2009

IPFIX for Wireless Sensors
draft-schmitt-6lowapp-ipfix-ws-00.txt

Abstract

In this draft we want to introduce an idea to develop a protocol for efficient data transmission for Wireless Sensors using the ideas of IPFIX. We will call this protocol IPFIX-WS.

Table of Contents

1. Introduction

Today everyone calls for new approaches for data acquisition in real-time. Different challenging requirements must be solved, such as efficient collection of environmental data using Wireless Sensors and efficient data transmission due to hardware limitations. Constraints on size, energy consumption and price lead to very limited memory, computational and communications resources. The desire for sensor nodes to be operational for a long time without external intervention, such as exchange of the batteries, which are often the only source of energy, leads to additional restrictions in the usage of the resources. Some research is done to address the issue of the sole dependency on battery power, but for now it remains the

prevalent source of energy for WSNs. The physical size of a sensor node is another limiting factor. The IRIS mote which is used in our setup has the dimensions of 58 x 32 x 7 mm, without the battery pack. Thus it does not leave much room for the micro controller, flash memory (128 kb) and RF transceiver, all of which are located on this board.

To satisfy those needs, standard protocols for efficient data transmission from common networks, such as the IP Flow Information Export (IPFIX) protocol, should be adapted to the equipment of Wireless Sensors. Another possibility is to reduce the data amount by implementing in-network data aggregation functionality. The most important thing is to develop these approaches while still providing interoperability between devices of different vendors.

1.1. Document structure

This draft will describe the idea to adapt the common IPFIX protocol to the requirements of Wireless Sensor technology. The resources are limited and the most important aspect is saving energy. Thus, the data should be transmitted efficiently to save energy costs. In the upcoming sections we want to introduce the IPFIX protocol for common networks followed by modifications for wireless sensors. A description of current implementation approaches and planned add-ons will follow. Finally, security and IANA considerations are mentioned as well as a conclusion.

2. Conventions used in this document

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in RFC-2119 [RFC2119].

3. Hardware Requirements

As mentioned before, Wireless Sensors should be used for collecting environmental data. They have many hardware requirements themselves, but the requirements of the application scenario also have to be kept in mind. On the hardware side, energy and computing capacity, as well as the space on the platforms are limited [1]. According to the application scenarios the sensors will be equipped with a different

amount and types of sensors (e.g. for temperature, humidity, seismic data).

4. IPFIX Protocol for Wireless Sensors

4.1. IPFIX Standard - RFC 5101

The IP Flow Information export (IPFIX) protocol was standardized under the RFC 5101 [RFC5101] and developed for common networks to transmit data efficiently. IPFIX itself is a PUSH-protocol. That means an exporter periodically transmits data to one or more collectors. The communication is template based. Before the data itself is transmitted a template is sent to declare how upcoming data has to be decoded. Thus, meta information is sent only once and for decoding there only need to be moved some pointers over the data. Finally, the processing needs are low. To make the transmission more efficient, data aggregation can be added.

Today the IPFIX protocol is used for network monitoring in common networks. These networks have different observation points and the hardware has no limited resources. In such networks IP flows pass through the different elements of the observed network. Due to security reasons it is interesting and useful to observe these flows. In this case the IPFIX Collecting process allows the user to verify different observation points of interest in the network. At this point the user is able to observe irregularities in traffic [RFC5101].

4.2. IPFIX-WS

As mentioned before, Wireless Sensors have different requirements. The common network protocol IPFIX is not yet adapted to fulfil the requirements of Wireless Sensors. In the first step, type and enterprise IDs must be defined to transmit sensor data measurements using IPFIX. Also new templates must be developed. An example is shown in Figure 1. As a consequence the data amount during transmission will be minimized and energy will be saved. The most important solutions are the data aggregation itself and a data compression [3].

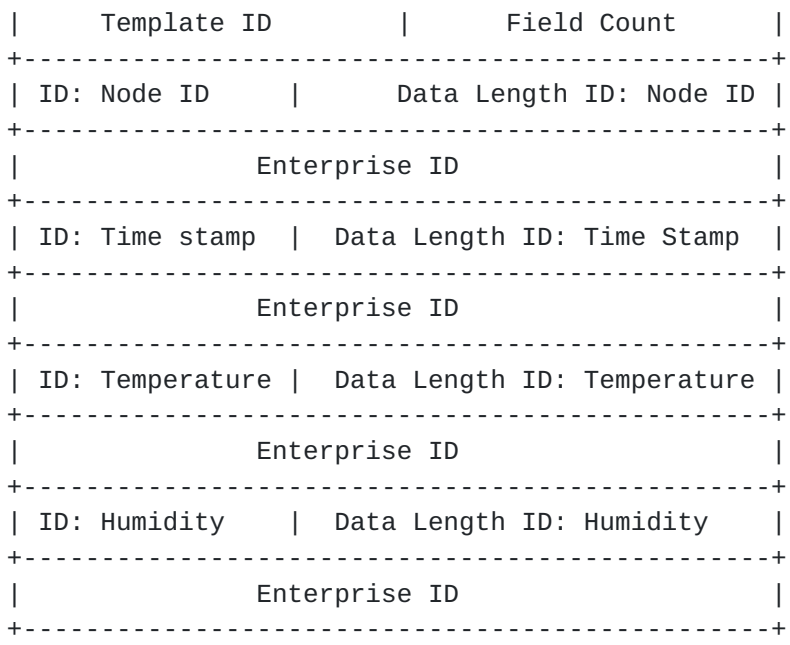


Figure 1: IPFIX-WS Template Set for sensor data

4.3. Technical details for IPFIX-WS

To apply IPFIX to Wireless Sensors the following tasks should be fulfilled:

- gather data from the sensors attached to the node

- generate IPFIX packets and transmit them to the base station of the network

- perform in-network aggregation to reduce the amount of network traffic and preserve energy

- receive and parse IPFIX packets on a Gateway server

- Transfer the acquired data to a home network infrastructure.

To achieve these goals, different steps must be executed. The mote's sensors are queried periodically to generate new sensor data. These raw values are transmitted to a specially developed tinyIPFIX library. It writes the values to the location specified by an IPFIX template, which was generated automatically when the mote booted. The template specifies the needed raw values. If all of these values are collected and written to the correct position in the respective IPFIX

data message, then the IPFIX packet is ready for transmission and sent to the base station via a multihop network. The base station is a special node in the network which is the connection point between a wireless and wired infrastructure. It listens for incoming packets from nodes in the network and transmits their payload to the gateway server over an USB connection. On the gateway a listening program waits for transmissions. The IPFIX messages sent via USB are parsed according to the matching IPFIX templates; their sensor values are extracted and transferred to the home network infrastructure. Here the raw values are converted from their platform specific representation to a platform independent value. For example a 14bit Integer raw value coming from a temperature sensor is converted to a Double whose value is in Celsius.

5. Integration of Wireless Sensors in Home Networks

Today common Home Networks work with IP addressing. Thus, the Wireless Sensors must be organised in a mesh network using also IPs for communication. For this situation the 6LoWPAN protocol was developed [4]. Additional requirements for connecting both system parts and a seamless integration of the Wireless Sensors into an existing infrastructure must be provided as well as support for devices of different vendors.

5.1. Hardware setting for test scenario

For our approach we are using IRIS motes from Crossbow Technology Inc. [10]. These motes have limited resources and can cooperate with different sensor boards. Crossbow Technology Inc. offers different sensor boards which include sensors for light, temperature, humidity, barometric pressure, seismic, GPS and others. The motes run TinyOS [11], an open-source and component-based operating system. Current versions of TinyOS are 1.x and 2.x which provided different applications and driver support.

5.2. Testbed results

Currently the IPFIX-approach was implemented in a network with 3 sensor nodes and one base station, and successfully tested in simulations with more nodes. Practical tests are running with the sensor boards MTS300CB and MTS400CB at the moment. The transmitted

data in such IPFIX packets consists of values shown in Figure 1. The packet sizes ranges from 34 bytes (data packet) to 72 bytes (template packet), including the IPFIX header. For simplicity, only one data set is sent per transmission. Figure 2 shows an example of an IPFIX-WS Template message and Figure 3 an IPFIX-WS Data message.

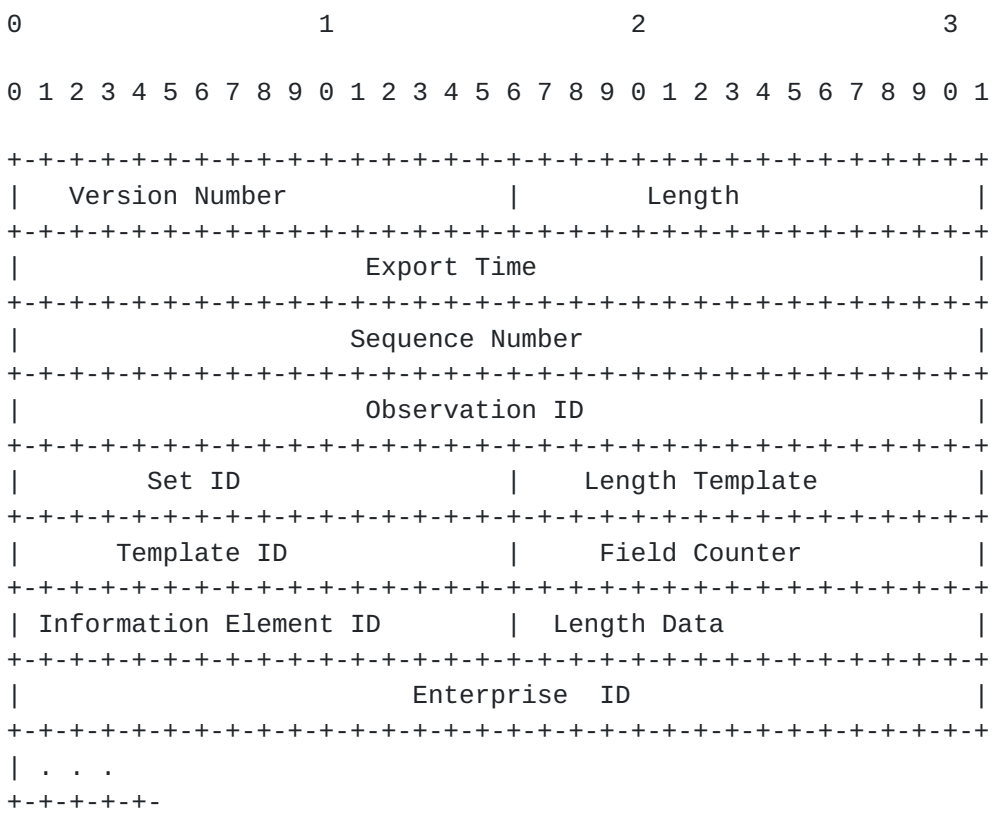
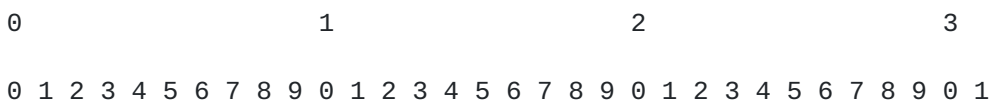


Figure 2: IPFIX-WS Template Format



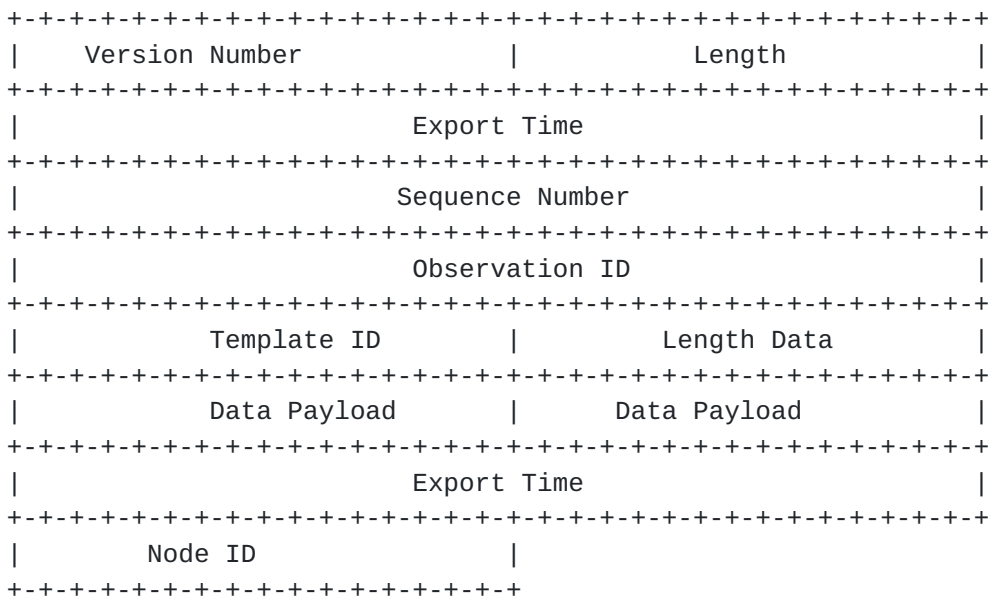
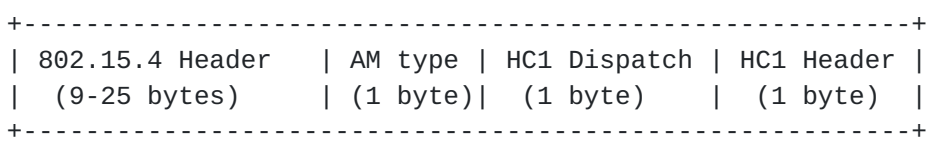


Figure 3: IPFIX-WS Data Format example (Node ID = Observation ID)

Parallel, 6LoWPAN was implemented for the platform IRIS using TinyOS 2.x based on the work of Harvan and Schoenwaelder [2]. They implemented a 6lowpan/IPv6 stack. The standard 802.15.4 provides two types of addresses (16-bit or 64-bit). Depending on the hardware used, the transmitted payload with 6LoWPAN can be up to 127 bytes. Optimal transmission is gained with 102 bytes payload. With the help of a fragmentation mechanism below the IP layer at least 1280 bytes can be transmitted. In the worst case, the IPv6 header has a size of 40 bytes. But the goal is to have as much space as possible for the payload. Thus, a header compression was implemented to compress the header down to 2 bytes. This compression mechanism can also be used for the transmission header like UDP which follows the IPv6 header. It can be compressed from 8 bytes down to 4 bytes. The IPv6 compression mechanism is called HC1 and the UDP compression mechanism is called HC_UDP. Figure 4 shows the worst case of transmission with no compression.



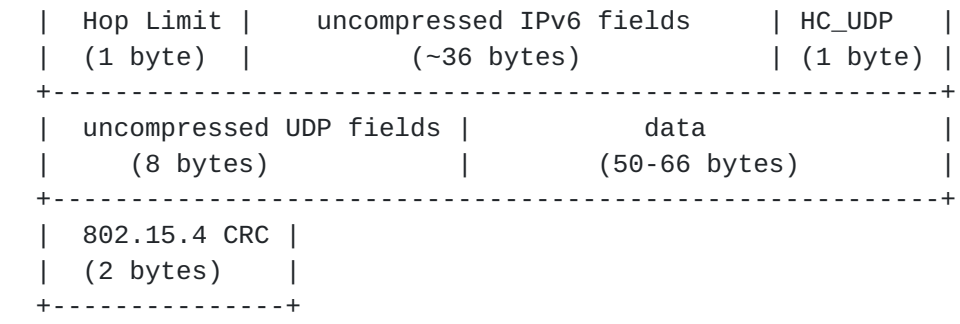


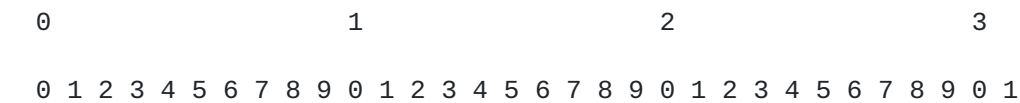
Figure 4: 6LoWPAN packet format uncompressed [RFC4944]

In the worst case, as shown in Figure 4, only 50-66 bytes are left for the data payload depending on the chosen address types in the 802.15.4 Header. Thus, compression is very important and in optimal case allows a data payload of 94-110 bytes depending on address types [RFC4944].

The 6LoWPAN approach was also successfully tested in real scenarios using IRIS motes. Currently the payload is simple without using sensor measurements due to driver problems.

5.3. Planned add-ons for IPFIX-WS

Currently we try to develop a header compression to reduce the header size of each IPFIX message which consists of the IPFIX Message Header and the Set Header as shown in Figure 5.



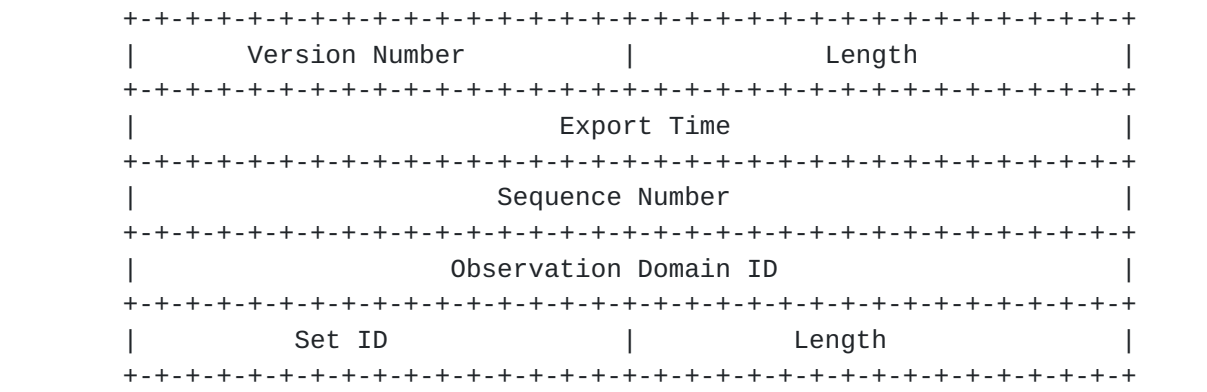


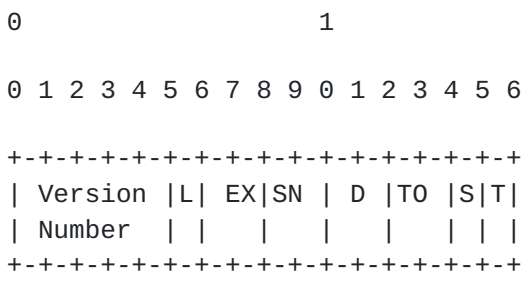
Figure 5: IPFIX Header Format including Message and Set Header

Since IPFIX was designed for conventional networks, some extensions and changes have to be introduced to increase it's efficiency in WSNs. One of the problems when deploying IPFIX in sensor networks is the overhead introduced by the relatively large header which is at least 32 bytes in size (24 bytes from the Message Header + 4 bytes from the Set header) as is shown in Figure 5. Remember, that the maximum size of a packet being transferred with an IEEE 802.15.4 network is 127 bytes. To address this issue, a header compression scheme was devised, which will be introduced in this section.

The idea behind our approach to header compression is to define the length of the fields separately in a pre header which is shown in Figure 6. First the Version field from the original IPFIX header is shortened to 5 bit, this leaves room for the IPFIX version to increase from version 10 to version 31. The definition of the length of the fields Message Length, Export Time, Sequence Number and Observation Domain ID follows. A value of 0 in the designated bit(s) means that the field is allowed 1 byte in the subsequent header, a value of 1 means 2 bytes, etc.. The next two bits are designated for the template offset. Decoders of IPFIX Messages are expected to keep track of the sequence in which they received templates from the IPFIX exporters. A value of 0 in the template offset bits means that the decoder should use the template it has received last, a value of 1 means the template before that and a value of 2 means two templates before the last one. If this offset is given for a data message, two

bytes for the Set ID can be saved. If template offset is set to 3 (both bits are one) it is ignored and a proper statement of the template ID is expected in the header. The next bit is called the Single Set Flag. It indicates whether the message contains only a single IPFIX set. If this is the case, the explicit statement of set

length in the header can be omitted since this value can be computed from the total message length. The last bit in the pre header is the Template Set Flag. If it is set to one, the first set in the message is a template set which is defined to have Set ID = 2. Thus the two bytes for definition of the set ID can be omitted.



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

Figure 6: IPFIX pre header defining the length of the subsequent header

In the best case scenario all header fields can be fitted to one byte and the set header can be fully omitted. The possibility to shorten the message length and Observation Domain ID to one byte is fairly obvious. Most messages will be shorter than 255 bytes, in fact if they are to be transmitted in a single packet they have to be smaller than 127 byte with our hardware. Since the Observation Domain usually refers to the node ID a value of one byte can accommodate 256 nodes which represent a WSN of medium scale. The sequence number can also be shortened to one byte, since a rollover after 255 messages is non problematic due to the low data sampling rate of typical WSNs. For the time stamp, a value of one byte could refer to the time that has passed since the last full UTC time stamp has been sent. Since the field length can be different with every package sent, it is possible to only transmit a full 4 bytes time stamp periodically and suffice with a delta value in between. So, for the best case this method can achieve a reduction in header size from 32 bytes to 6 bytes, or a compression of 81.25%. Figure 7 gives an example of the best case, which is actually fairly common since it shows the transmission of a data record referencing the last sent template set. In the worst case

however, header size may increase to 33 byte when all header fields are defined to be their original length.

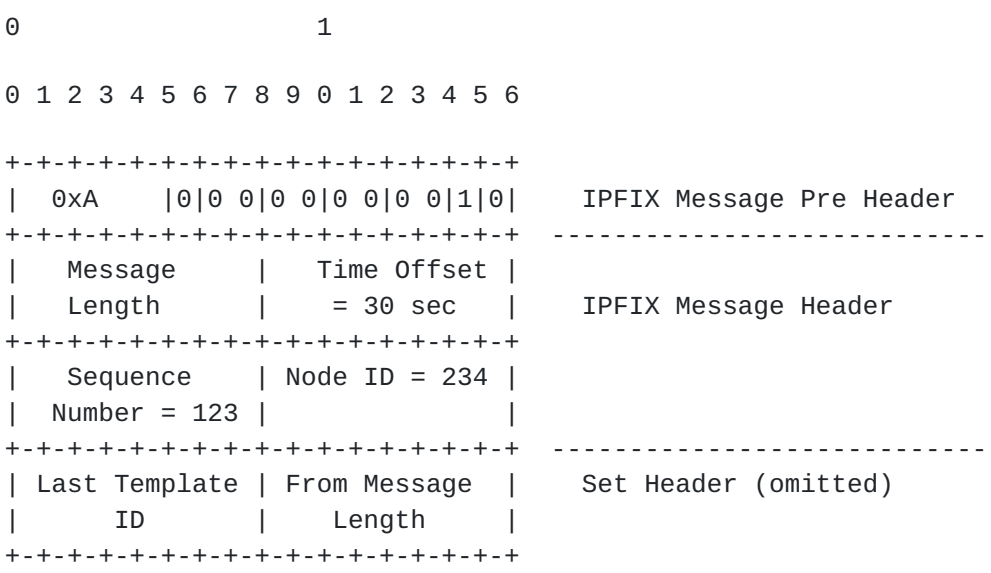


Figure 7: Best case header for the IPFIX header compression

A parallel approach is to integrate the IPFIX messages in the communication with 6LoWPAN. Here we are faced with some missing drivers for our hardware. We plan to port drivers from TinyOS 1.x to TinyOS 2.x which is needed for 6LoWPAN. Currently the only supported sensor board for IRIS motes under TinyOS 2.x is MTS300.

If the integration with 6LoWPAN is done the IPFIX messages may get a more compressed format due to redundancy. For example the source address mentioned in the IP header is the same as the node ID in the IPFIX packets. Thus, the whole packet size might become smaller.

6. Formal Syntax

IPFIX - - IP Flow Information Export protocol based on RFC 5101

IPFIX-WS - - IP Flow Information Export protocol for Wireless Sensors

7. Security Considerations

Measurement data security and data integrity can be integrated as well. IPFIX copes with these security issues by specifying that every IPFIX device needs to support TLS (on stream based transport protocols) or DTLS (on datagram based transport protocols). Fouladgar et al. [5] developed Tiny 3-TLS, a TLS handshake sub-protocol for sensor nodes, which can be used for securing IPFIX data transmission. TinySec [6] can be used instead to achieve data link security and message authentication. It offers an authentication encryption mode where data payload is encrypted and the packet itself is authenticated by a MAC. Another approach using the same idea as TinySec was developed by Luk et al. [7], called MiniSec. It is a secure sensor network communication architecture which modifies the common packet structure of TinyOS and combines features from TinySec and ZigBee [8] to perform low energy consumption and high security.

8. IANA Considerations

Vendors may specify their own IDs that are located above ID 32767. All these IDs have the most significant bit set to 1. If a collector recognizes an ID with this bit set, he can determine that this ID is a non standard ID. The collector will then check the enterprise ID (EID). Each vendor has to register an enterprise ID with Internet Assigned Numbers Authority (IANA) [9] which will ensure that any vendor can be identified uniquely.

9. Conclusions

In this draft we introduced a concept for integrating IPFIX in wireless sensor networks. We showed that IPFIX is suitable for deploying and integrating WSNs into home networks. With a connection to 6LoWPAN this approach can also be used for other applications using IP-addresses for communication.

IPFIX defines an efficient data format for transmitting sensor measurement data using low bandwidth. Generating and parsing IPFIX data can be performed with little processing power, thus saving energy on the nodes. Arbitrary aggregation techniques can be deployed to further reduce the transmitted data. If standard template IDs are issued, interoperability between different devices from different

manufacturers can be ensured. At the same time, vendors can register their own enterprise and type IDs to build custom devices. These devices can still interoperate with other devices. By using IP on the network layer below IPFIX, wireless sensor networks can easily be integrated in existing home networks. Therefore, new sensor nodes can be easily deployed and new functionality to the autonomic network can be added in an automatic fashion.

10. References

10.1. Normative References

[1] C.Schmitt and G.Carle: 'Applications for Wireless Sensor Networks', Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications, Edited by N.Antonopulus, G.Exarchakos, M.Li and A.Liotto, ISBN: 1-61520-686-8, Information Science Publishing, 2009 (in printing process)

[2] M.Harvan and J.Schoenwaelder, 'TinyOS Motes on the Internet: IPv6 over 802.15.4 (6lowpan)', PIK - Praxis der Informations-verarbeitung und Kommunikation, 2008, vol.31, pp. 244-251.

[4] J.W.Hui and D.E.Culler, 'IP is dead, ling live IP for wireless sensor networks', in SenSys 2008: Proceedings of the 6th ACM conference on Embedded network sensor systems, ACM New York, NY, USA, 2008, pp.15-28.

[5] S. Fouladgar, B. Mainaud, K. Masmoudi, and H. Afifi, 'Tiny 3-tls: A trust delegation protocol for wireless sensor networks', Lecture Notes in Computer Science, 2006, vol. 4357, p. 32-42.

[6] C. Karlof, N. Sastry, and D. Wagner, 'TinySec: a link layer security architecture for wireless sensor networks', in Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM New York, NY, USA, 2004, pp. 162-175.

[7] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, 'MiniSec: a secure sensor network communication architecture', in IPSN 2007: Proceedings of the 6th international conference on Information processing in sensor networks. ACM New York, NY, USA, 2007, pp. 479-488.

[8] ZigBee Alliance, 'ZigBee specification. Technical Report', ZigBee Alliance, Document 053474r06 Version 1.0, June 2005.

[9] Internet Assigned Numbers Authority, http://www.iana.org/, 2009.

[10] Crossbow Technologies Inc., http://www.xbow.com/, 2009.

[11] TinyOS Homepage, http://www.tinyos.net, 2009.

[RFC4919] N.Kushalnagar et al., 'IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals', 2007

[RFC4944] N.Kushalnagar et al., 'Transmission of IPv6 Packets over IEEE 802.15.4 Networks', 2007

[RFC5101] B.Claise et al., 'Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information', 2008

[RFC2119] S.Bradner et al., 'Key words for use in RFCs to Indicate Requirement Levels', BCP 14, RFC 2119, 1997.

10.2. Informative References

[3] G.Muenz and L.Braun, 'Lossless Compression for IP Flow Information Export (IPFIX)', The Internet Engineering Task Force (IETF), Internet-Draft (work in progress), 2008

Authors' Addresses

Corinna Schmitt
TU Muenchen
Chair for Network Architectures and Services
Boltzmannstr. 3
85748 Garching, Germany
Email: schmitt@net.in.tum.de

Lothar Braun
TU Muenchen
Chair for Network Architectures and Services
Boltzmannstr. 3
85748 Garching, Germany
Email: braun@net.in.tum.de

Georg Carle
TU Muenchen
Chair for Network Architectures and Services
Boltzmannstr. 3
85748 Garching, Germany
Email: carle@net.in.tum.de