UPnP Compression for IP based Field Devices in Building Automation

S. Knauth, D. Käslin, R. Kistler and A. Klapproth
University of Applied Sciences of Central Switzerland – Lucerne
HTA Luzern - CEESAR Embedded System Applied Research
Technikumstrasse 21
CH-6048 Horw, Switzerland
sknauth@hta.fhz.ch

Abstract

IP based field bus networks enable the usage of common IP protocols for example for security or automatic configuration, on the field level. A drawback of this protocol deployment is the higher required network bandwidth due to enormous high level protocol overhead, especially on XML-based schemes. We investigate UPnP datagram size reduction on an experimental IP based field bus with Ethernet and wireless IEEE802.15.4 devices, for building automation and control applications. UPnP is used for configuration and operation. In order to reduce data rates, we suggest a transparent bidirectional XML/SOAP proxy for example on the Ethernet/IEE802.15.4 hub and on the wireless devices itself. The proxy uses cache-based tokenizing of the SOAP messages and feedback, eventually deflating subsequent similar messages by orders of magnitude.

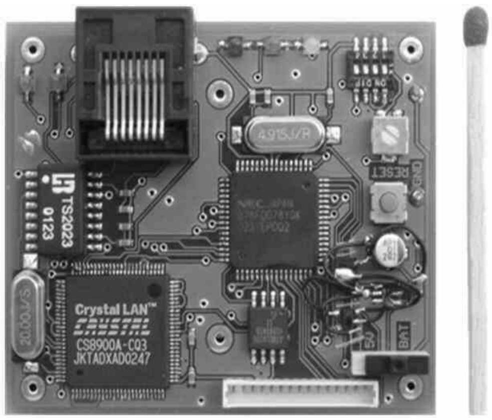
Figure 1. experimental low cost building automation field device [7], capable of communication via high level IP protocols.

1. Background

It is likely that next generation building control and automation networks (also referred to as “BAU” Networks) will migrate from current fieldbus solutions to IP protocol based field level solutions using wired and wireless connections, ousting legacy technologies to special application areas. For common building automation field bus systems (LON, EIB/KNX, PROFIBUS [1, 2, 3]), connectivity to IP networks is typically realized with an OPC [4] server. The field devices themselves do not communicate via IP. There exist solutions, where devices of, for example, an EIB field bus system are presented on an IP network in a UPnP [5] facade [6]. This approach is somewhat superior to an OPC XML presentation, but still the BAU devices are operating on their native field bus protocols. IP as fieldbus protocol allows adopting common IT network technologies to the field devices provided that the devices offer sufficient computational resources. Using highly optimized stacks, it is possible to integrate TCP/IP and high level protocols like the chain HTTP-SOAP-UPnP on low resource devices [7, 8].

Figure 1 shows a device we built and use for research applications in building automation. The device is running on a low resource 8 bit platform offering 60 kB of flash memory and 4 kB of RAM [7]. For building automation, ease of device and network commissioning is very important, in order to keep the network manageable and reliable and keep maintenance efforts and costs on a reasonable level [9]. The used protocols should be standardized or widely– and vendor–independently used. Therefore we chose UPnP for configuration and operation. UPnP (Universal Plug & Play, (C) Microsoft Corp.) [5] is a protocol for automatic device integration into IP networks. It covers addressing, device discovery, description, control, eventing, and presentation. The initial target of UPnP technology was, among others, home multimedia and office applications for PCs, but nowadays there are also templates defined for HVAC (Heating, Ventilation and Air-Conditioning), and yet proprietary device types can be modelled in the basic device template.

One drawback of using UPnP as operation protocol is its ”talkativeness”. Since it is based on SOAP [10], XML and HTTP, a basic command easily can have a length of several hundreds of bytes, which causes congestion or command propagation delays when IP communication is carried out via low- or mid rate physical layers (PHY), in our example the wireless standard IEEE 802.15.4. In this paper we report on our investigations on XML/SOAP compression and decompression for UPnP application in BAU networks

2. UPnP compression

2.1. Overview

UPnP communication for command transfer is done via SOAP messages transferred as XML datasets. A SOAP message for a typical command like ”switch light on” will have a length of some 100 Bytes. Besides addressing, the transferred information contains the desired state of the lamp, in the particular case 1 bit. Typically, in building automation such messages are transferred from one target to another with a rate well below 1 Hz. So in a typical 10 or 100 MBit Ethernet-based IP network, the traffic overhead due to the utilization of SOAP is not impacting the network in any way and the advantages of having a clearly defined command transfer mechanism and syntax justifies the overhead.

The situation changes when regarding two-wire or wireless fieldbus systems with rates down to 9600 bps. To apply UPnP in such scenarios, it is necessary to reduce the size of a command datagram. Known general approaches to the topic of XML compression are for example ASN.1 mapping or binary XML. Using Abstract Syntax Notation One (ASN.1), various encoding rules may be applied, therefore data may be transferred using highly compact binary formats and compression. Since XML formatted data and XML schema can be modelled in ASN.1 to some extent [11], it is possible to implement bidirectional converters between the two formats. In [12] a typical compression ratio of 4 ($\mathrm{Size}_{\mathrm{XML}}/\mathrm{Size}_{\mathrm{ASN}}$) has been reported for ASN.1 BER encoding. The chapter “control” of the UPnP description [5] also suggests the possibility to transfer data “out of band” i.e. by other mechanisms like direct binary transfer. Recently a W3C XML Workgroup “Binary XML” has been formed on this issue [13].

2.2. Tokenization

Tokenizing is here meant to be the replacement of a string with a key, where the key allows retrieval of the string from a dictionary. It is probably among the oldest data compression methods in information technology. Fundamental algorithms are the LZ77 [14] and LZ78 [15] stream compressors [16], where the dictionary is continuously updated and reconstructed from the decoded message on the receiver side. These algorithms are widely used in communication technology for example as compressors on the serial IP protocol PPP (Peer to Peer Protocol), as MPPC (Microsoft Point to Point Compression) [17] on CCP (RFC1962: Compression Control Protocol) [18]. These methods are very good in lossless compression of streams with local correlation, and do not use or need any prior knowledge of the structure of the data to be processed. On the other hand their horizon is defined by a sliding window so that no quasi–static parts in the dictionary are foreseen.

UPnP action messages in BAU applications normally have a length of several hundreds of bytes, imposed by the textual formulation of the HTTP headers and the identification of services and addressing by URLs, and the transmission of lengthy attribute names, which mostly occur twice. A typical operation in a BAU network might be a light switch, sending a “switch on” command to a lamp, or a thermometer which cyclic transmits a measurement to a heating controller. When regarding several subsequent occurrences of these events between two specific devices, the differential information is only 1 bit, in the case of the lamp, and a few bytes in the case of the thermometer. It is straightforward to replace the common parts of the messages by tokens.

A common approach, compression here meant as suppression of repeated transmission, is for example the TCP header compression [19] used in the SLIP (Serial Line IP). Here the headers have a strict block structure of constant length and no semantic information for reconstruction is needed, at the receiver side. Also the communication is one-way, the coder always knows the dictionary state of the encoder. The mentioned CCP protocol allows a misalignment message to be sent from the encoder to the coder.

If the configuration of the network was fixed and all to–be–transferred messages known in advance, the tokenizing mechanism could use a fixed dictionary. The sender of a SOAP command datagram would replace all occurrences of known words by tokens, and the receiver would replace the tokens back to the full string. The disadvantage of this approach is its static behavior. If devices are brought up after the dictionary is defined or are updated to have new commands, their messages may not be included in the common token dictionary. More sophisticated methods like mobile radio WAP XML compression [20] and the XMill [21] and XGRIND [22] projects use a variety of technologies for decomposing and tokenization, but do not especially encounter for repeatedly sent datagrams, and the latter two are not suitable for low resource devices.

2.3. Bidirectional Proxy

Since we want to use devices implementing standard UPnP, the tokenizing shall occur in proxies, which intercept the connection between sensor and actor. The dictionaries are dynamic and the proxies learn by performing command transfers.

More detailed, if the sender does not find a message part in his dictionary, he detects which parts of the message are likely to occur repeatedly. For this analysis some knowledge of the protocols to be transferred (UPnP, SOAP, XML, HTTP) is used. Some HTTP headers like for example LENGTH are not transferred but recreated at Proxy II. It is expected that also the SOAP envelope does not change when regarding transfers between two particular devices. By such predefined behavior the message can be stripped down to the attribute values and some tokens.

Principally, in contrast to Lempel-Ziv [14, 15] methods, the transmitted data does not contain the dictionary information. Also, by requirement, the encoder shall not rely on the dictionary state at the decoder. Therefore there will be the possibility for a feedback for the encoder to request missing information from the coder. The lifetime of the dictionary is beyond the lifetime of a single datagram transmission or the corresponding TCP connection.

In order to minimize the synchronization requirements of participating proxies, the tokens consist of hashes which are calculated from the to–be–tokenized text blocks. The sender and the receiver store the hash-values and the corresponding strings. Hashes is here meant as a relation of the full text into a number within a predefined range. A very easy hash function would just be the sum of the ASCII-values of the message characters modulo the desired range. Principally, several text inputs may give the same hash value, and in the case of the above defined function, all letter–permutations will give the same value. We are testing 16 bit and 24 bit hashes generated by shift registers with XOR-feedback. Collision-avoidance (two different strings generate the same hash) is carried out by the sender. If a hash collision is detected, the string is broken in two parts. In a scenario where a target device listens to several sending proxies, care must be taken to avoid cross-site collisions. The solutions under investigation are communication between the two sending proxies or alternatively handling of hashes of different coders separately on the encoder side. A second method under investigation for coding of the tokens is enumeration. Here, the problem of counter overflow is addressed by a count window and all entries outside of the window have to be deleted, on the coder as well as on the encoder.

Figure 2 shows three scenarios for command transfer. Command A is transferred directly from sensor to actor. This is the normal UPnP scenario. Command B is transferred through Proxy I - Proxy II. Proxy I replaces some parts of the message with tokens, and stores this compression information in his cache. Proxy II does not yet know about these tokens and requests their full text. After that, the command can be reconstructed and transmitted to the actor. In the case of Command B, the overall amount of transmitted data is higher as it would be in the uncompressed scenario A, and also the command delivery will be delayed by the duration of the additional packet transfers. Depending on available memory and entropy of the transferred data, this will only happen very occasional. Command C is also transferred via the proxy chain. In the case C, all tokens which have been used in the command during transfer from Proxy I to Proxy II, are known to proxy II, and the UPnP command can be reconstructed without further communication.

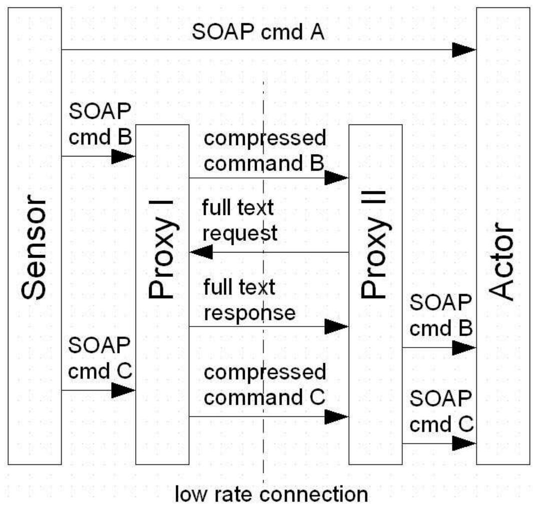

Figure 2. Compression scenarios. (A) direct transmission (B) learning scenario (C) compressed command transfer.

2.4. Results

We are currently implementing the proposed UPnP proxy system, and the proxies themselves. The intelligent discovery of unchanged parts in the UPnP messages is a main challenge of the project. By saving interlaced hashes over fixed text passage lengths, comparison time between stored and new commands is optimized. Depending on cache size and entropy of the uncompressed UPnP data, compression ratios of 1:100 have been achieved for special configurations, for example operation of illumination where the actual information is 1 bit only. In this case complete envelopes can be replaced by their hash values.

3. Conclusion

3.1. Summary

We developed a compression / suppression scheme for UPnP messages. This is motivated by the wish to use UPnP for configuration and operation of field level devices for building automation and control, on low transfer capacity physical layers. We suggest the usage of bidirectional tokenizing proxies for UPnP compression. Our ongoing research indicates that using of such proxies allows the UPnP implementation on the devices and the management respective automation nodes to be standards compatible, but still operate with highly reduced data rates on the IP network. This enables the deployment on networks with data transfer rates considerably lower that those of Ethernet, as it is the case in common two-wire field bus networks standards or on, for example, IEEE802.15.4 wireless connections.

3.2. Outlook

We believe that IP on the field level of building automation networks will play an increasing role besides other field level protocols in most of BAU domains, because of advantages like infrastructure sharing, standardization, connectivity, and software solutions available, to name but a few of them. In IP based building automation and control field bus systems, we propose UPnP to be a good solution for easy configuration and setup. UPnP proxy technology for data reduction is an important prerequisite in this concept.

References

[1] (1990), LON: Local Operating Network [Online]. Available: http://www.echelon.com

[2] (1995), EN 50090 - EIB/KNX open Standard for Home and Building Control [Online]. Available: http://www.konnex.org/

[3] (1995), PROFIBUS International [Online]. Available: http://www.profibus.com/pb/

[4] OPC Foundation, (1996), Openess, Productivity, Collaboration [Online]. Available: http://www.opcfoundation.org/

[5] (Dec. 2003), UPnP Forum,UPnP^{TM} Device Architecture 1.0, Version 1.0.1 [Online]. Available: http://www.w3.org/TR/wsdl20/

[6] W. Kastner and H. Scheichelbauer, *UPnP Connectivity for Home and Building Automation*, Proc IASTED (PDCN) 2004, Hamza M. H. (Ed.), IASTED Anaheim, Calgary, Zürich.

[7] A. Klapproth, D. Käslin, and T. Bürkli, (Apr. 2005), ZigBee Entwicklerforum 2005 Munich: Lowcost Wireless Webserver [Online]. Available: http://www.ceesar.ch/cms/upload/pdf/Paper%20Wireless %20Webserver%20HTA%20Luzern.pdf

[8] A. Klapproth and T. Bürkli, (Apr. 2005), ZigBee Entwicklerforum 2006 Munich: TCP/IP über IEEE 802.15.4 [Online]. Available: http://www.ceesar.ch/cms/upload/pdf/FH-Luzern_TCPIP-over-IEEE802%2015%204.pdf

[9] M. R. Brambley and S. Katipamula, *Beyond Commissioning: The Role of Automation*, U.S. Department of Energy, Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161, 2005.

[10] (Mar. 2006), Web Services Description Language (WSDL) [Online]. Available: http://www.w3.org/TR/wsdl20/

[11] (2002), Abstract Syntax Notation One (ASN.1) Specifications, ITU-T Rec. X.680 X.683 and X690 X.693 (2002) ISO/IEC 8824-1:2002 to 8824-4:2002 and ISO/IEC 8825-1:2002 to 8825-4:2002 [Online]. Available: http://asn1.elibel.tm.fr/xml/

[12] O. N. Inc, "Alternative binary representations of the XML Information Set based on ASN.1", in *W3C Workshop on Binary Interchange of XML Information Item Sets, Santa Clara, California, USA*, Sept. 2003.

[13] (2004), XML Binary Characterization Working Group Public Page [Online]. Available: http://www.w3.org/XML/Binary/

[14] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE-Transactions-on-Information-Theory*, vol. IT-23, no. 3, pp. 337–343, May 1977.

[15] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding", *IEEE-Transactions-on-Information-Theory*, vol. IT-24, no. 5, pp. 530–536, Sept. 1978.

[16] D. Sheinwald, A. Lempel, and J. Ziv, "On encoding and decoding with two-way head machines", *Information-and-Computation*, vol. 116, no. 1, pp. 128–133, Jan. 1995.

[17] Cisco Systems Inc., (Jan. 2000), Microsoft Point-to-Point Compression (MPPC) [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/product/soft ware/ios113ed/113t/113t_3/mppc.htm

[18] D. Rand, (June 1996), RFC1962: The PPP Compression Control Protocol (CCP) [Online]. Available: http://www.ietf.org/rfc/rfc1962.txt

[19] V. Jacobson, (Feb. 1990), Compressing TCP/IP Headers for Low-Speed Serial Links [Online]. Available: http://tools.ietf.org/html?rfc=1144

[20] (June 1999), WAP tokenization [Online]. Available: http://www.w3.org/TR/wbxml/

[21] H. Liefke and D. Suciu, "XMill: An efficient compressor for XML data", in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 153–164.

[22] P. Tolani and R. H. Jayant, "XGRIND: A Query-friendly XML Compressor", in *Proceedings of the 18th IEEE International Conference on Data Engineering (ICDE)*, 2002.