4.3 Common Protocols

This section introduces protocols that are commonly used or have good potential for use over 6LoWPAN. These include web service protocols, MQTT-S, ZigBee CAP, service discovery protocols, SNMP, RTP/RTCP, SIP and industry-specific protocols.

Web service protocols

The web service concept is hugely successful on the Internet, especially in enterprise machine-to-machine Internet systems. As many back-end systems incorporating information from LoWPAN devices will already be using existing web service principles and protocols, it is expected that 6LoWPAN will be integrated into the web service architecture. The use of XML, HTTP and TCP makes the adaptation of web services challenging for LoWPAN Nodes and networks. In this section we look inside web service protocols, and the technologies that can adapt them for use with 6LoWPAN.

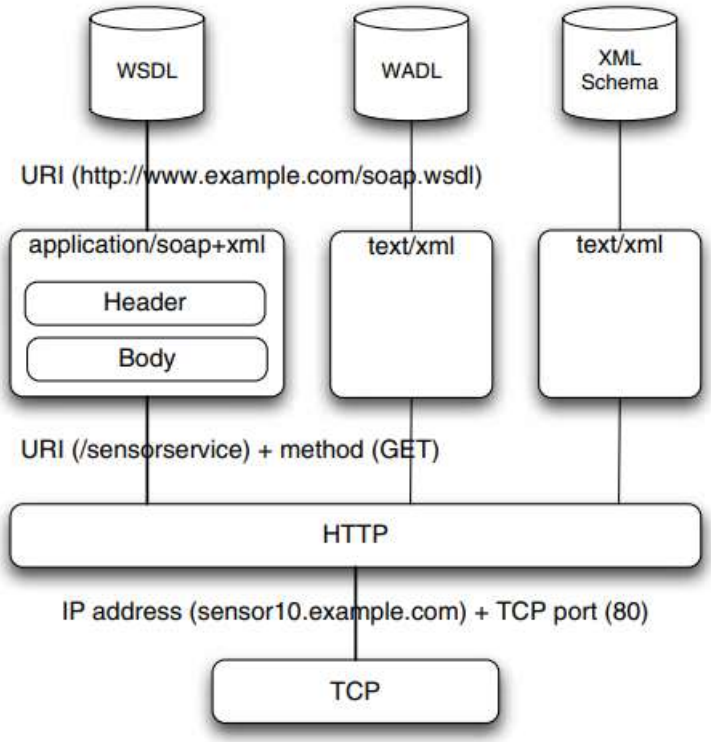

Fig 4.3.1 Typical structure of web service content over HTTP/TCP

Figure 4.3.1 shows the typical structure of web service content, which is always built upon HTTP and TCP as used today on the Internet. Web services are simply URLs available on an HTTP server with services or resources accessible behind them. In the SOAP model a URL identifies a service, e.g. /sensorservice in the figure. This service may support any number of methods (e.g. GetSensor) with corresponding responses that are described by a WSDL document. SOAP (application/soap+xml) is an XML format consisting of a header and a body, in which the body carries any number of messages.

Resource-based web services can also be realized using a REST design. Figure 5.5 also shows text/xml content used directly over HTTP. In this model formal message sequences are not used, instead each resource is identified by a URL. By using different HTTP methods on that URL, the resource can be accessed. For example sending an HTTP GET for /sensors/temp might return a text/xml body with the temperature of the sensor. REST designs make use of well-known XML or other formats to give meaning to the content that can be understood by all parties. All web services have the same basic problems for 6LoWPAN use. XML is typically too large for marking up content in the payload space available, HTTP headers have high overhead and are difficult to parse, and finally TCP has limitations of its own. A simplified HTTP header with application/soap+xml content for the example above may look like:

POST /sensorservice HTTP/1.1

Host: sensor10.example.com

Content-Type: application/soap+xml; charset=utf-8

Content-Length: nnn 0x1a

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.com/soap.wsdl">
  <m:GetSensor>
    <m:SensorID>0x1a</m:SensorID>
  </m:GetSensor>
</soap:Body>

</soap:Envelope>
```

This simple example has a length of 424 bytes, which may require up to six fragments to transmit it over a 6LoWPAN network. Next we look at different technologies to allow web services to be used with 6LoWPAN. There are two fundamental ways to integrate 6LoWPAN into a web service architecture: using a gateway approach or a compression approach.

Gateway approach: In the gateway approach, a web service gateway is implemented at the edge of the LoWPAN, often on a local server or the edge router. Inside the LoWPAN a proprietary protocol is used to request data, perform configuration etc. The gateway then makes the content and control of the devices available through a web service interface. In this approach web services actually end at the gateway. This technique is widely used with non-IP wireless embedded networks such as ZigBee and vendor-specific solutions. One downside of this approach is that a proprietary or 6LoWPAN-specific protocol is needed between nodes and the gateway. Furthermore, the gateway is dependent on the content of the application protocols. This creates scalability and evolvability problems where each time a new use of the LoWPAN is added or the application format is modified, all gateways need to be upgraded.

Compression approach: In the compression approach, the web service format and protocols are compressed to a size suitable for use over 6LoWPAN. This can be achieved using standards, and has two forms: end-to-end and proxy. In the end-to-end approach the compressed format is supported by both application end-points. In the proxy approach an intermediate node performs transparent compression so that the Internet end-point can use standard web services.

Several technologies exist for performing XML compression. The WAP Binary XML (WBXML) format was developed for mobile phone browsers [WBXML]. Binary XML (BXML) from the Open Geospatial Consortium (OGC) was designed to compress large sets of geospatial data and is currently a draft proposal [BXML]. General compression schemes like Fast Infoset (ISO/IEC 24824-1) work like zip for XML [FI]. Finally, the W3C is currently completing standardization of the efficient XML interchange (EXI) format, which performs compact binary encoding of XML [EXI]. One suitable technology for use with 6LoWPAN is the proposed EXI standard, as it supports out-of-band schema knowledge with a sufficiently compact representation. LoWPAN Nodes do not actually perform compression; instead they directly make use of the binary encoding for content, which keeps node complexity low.

XML compression alone only solves part of the problem. HTTP and TCP are still not suitable for use over 6LoWPAN. One commercial protocol solution called Nano Web Services (NanoWS) from Sensinode applies XML compression in an efficient binary transfer protocol over UDP, which has been specifically designed for 6LoWPAN use [Sensinode].

The SENSEI project is also researching the use of embedded web services inside Internetbased sensor networks [SENSEI]. The ideal long-term solution will be the standardization of a combination of XML binary encoding bound to a suitable UDP-based protocol.

The namespace and schema used with 6LoWPAN devices must also be carefully designed. Standard schemas such as SensorML from the OGC [SensorML] are often much too large and complicated for efficient use over 6LoWPAN even with compression. The most efficient result is achieved using a simple schema or one designed specifically for use with embedded devices.

1. **MQ telemetry transport for sensor networks (MQTT-S)**

The MQ telemetry transport (MQTT) is a lightweight publish/subscribe protocol designed for use in enterprise applications over low-bandwidth wide area network (WAN) links such as ISDN or GSM [MQTT]. The protocol was designed by IBM and is used in commercial products such as Websphere and Lotus, enjoying widespread use in M2M applications. MQTT uses a broker-based pub/sub architecture, to which clients publish data based on matching topic names. Subscribers then request data from the broker based on topic names. Although MQTT was designed to be lightweight, it requires the use of TCP, and the format is inefficient over 6LoWPAN networks.

In order to allow for MQTT to be used also in sensor networks, MQ telemetry transport for sensor networks (MQTT-S) was developed [MQTT-S]. This optimized protocol can be used over ZigBee, UDP/6LoWPAN or any other simple network providing a bi-directional datagram service. MQTT-S is optimized for low-bandwidth wireless networks with small frame sizes and simple devices. It is still compatible with MQTT and can be seamlessly integrated with MQTT brokers using what is called an MQTT-S gateway.

The MQTT-S architecture is shown in Figure 5.6. It is made up of four different elements: MQTT brokers, MQTT-S gateways, MQTT-S forwarders and MQTT-S clients. Clients connect themselves to a broker through a gateway using the MQTT-S protocol. The gateway may be located e.g. on the LoWPAN Edge Router, or it may be integrated in the broker itself, in which case MQTT-S messages are simply carried end-to-end over UDP. Gateways translate between MQTT-S and MQTT. In case a gateway is not directly available, forwarders are used to forward (unmodified) messages between clients and brokers. Forwarders may not be needed with 6LoWPAN as UDP datagrams can be sent directly to a gateway.

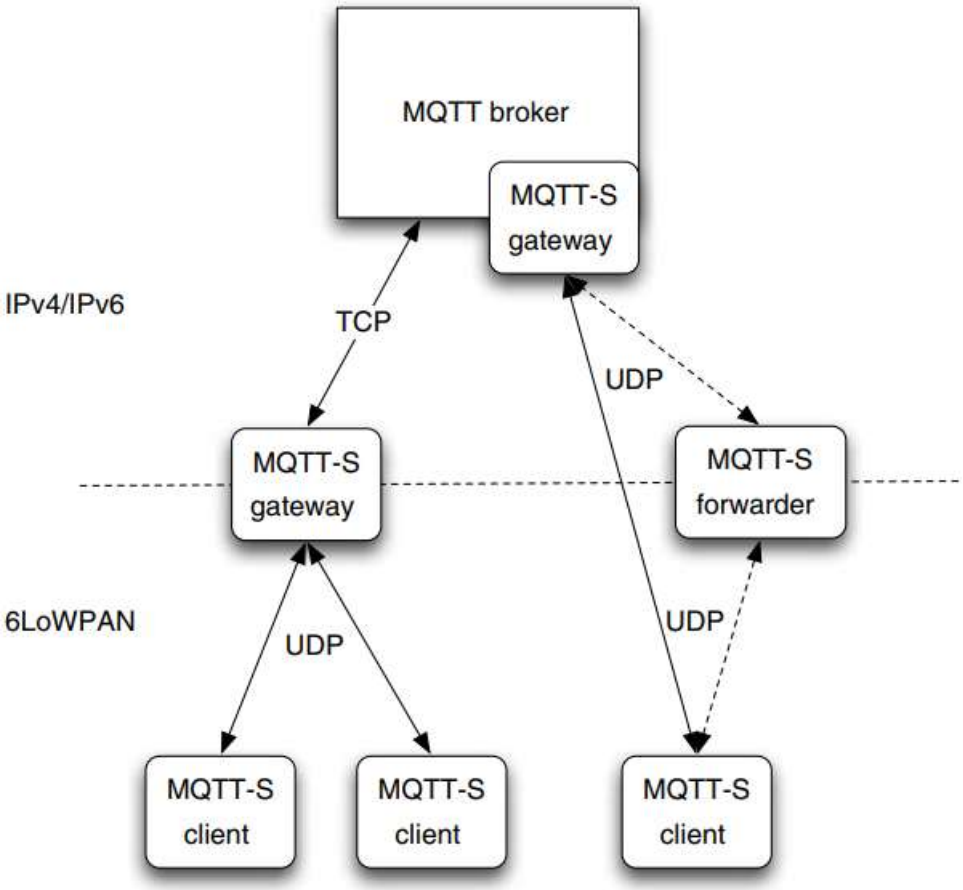

Fig 4.3.2 The MQTT-S architecture used over 6LoWPAN.

Figure 4.3.2 shows the MQTT-S message structure, which consists of a length field, a message type field and then a variable-length message part. Next the basic functionality of MQTT-S is described shortly. Readers should consult the MQTT-S specification for full protocol details [MQTT-S].

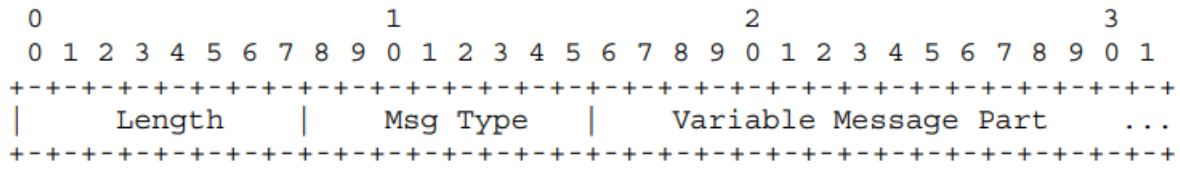

Fig 4.3.3 The MQTT-S message structure

Protocol operation: MQTT-S includes a gateway discovery procedure, which does not exist in MQTT. Alternatively clients can be pre-configured with the gateway location to avoid discovery overhead. Gateways send periodic ADVERTISE messages, and clients may send SEARCHGW messages. A GWINFO message is sent to a client in response to SEARCHGW with basic information about the gateway.

Clients CONNECT to a gateway which responds with an ACK. DISCONNECT is used to end a connection or to indicate a sleep period. Clients can connect with multiple gateways (and thus brokers) which are able to perform load balancing. Gateways can function in either transparent mode, where a connection to the broker is maintained for each client, or in aggregation mode, where the gateway aggregates messages from all clients into a single broker connection. Aggregation mode can considerably improve scalability.

MQTT-S makes use of two-byte topic IDs and short topic namesto optimize the long topic name strings normally used in MQTT. MQTT-S includes a registration message REGISTER, which explicitly indicates the topics that a client is

publishing. This reduces the bandwidth compared to MQTT where the topics and data are published in the same message. A client can send a REGISTER message with the topic name, which is acknowledged with a REGACK indicating the assigned topic ID. Topic names and IDs can also be pre-configured to avoid the need for registrations in certain cases. The client then publishes its data with PUBLISH messages including the topic ID and possible QoS information.

Clients subscribe by sending SUBSCRIBE to the gateway including the topic name of interest, which is acknowledged with SUBACK including an assigned topic ID. UNSUBSCRIBE is used to remove a subscription from the gateway.

2. ZigBee compact application protocol (CAP)

The ZigBee application layer (ZAL) [ZigBee], and the ZigBee cluster library (ZCL) [ZigBeeCL] specify an application protocol enabling interoperability between ZigBee devices at the application layer. The ZigBee Alliance maintains a series of specifications for ad hoc networking between embedded devices using a single radio, IEEE 802.15.4. Typical applications for ZigBee include home automation, energy applications and similar localarea wireless control applications. ZigBee makes use of a vertical profile approach over the ZAL and ZCL, with profiles for different industry applications such as the ZigBee home automation profile [ZigBeeHA] or the ZigBee smart energy profile [ZigBeeSE]. The ZAL and ZCL provide the key application protocol functionality in ZigBee, enabling the exchange of commands and data, service discovery, binding and security along with profile support. These protocols use compact binary formats with the goal of fitting in small IEEE 802.15.4 frames.

The ZigBee application protocol solution would have benefits used over standard UDP/IP communications as well, especially over 6LoWPAN, as the ZigBee application protocol has been designed with similar requirements. This would allow much wider use of ZigBee profiles, also end-to-end over the Internet eliminating the need for gateways. However the ZAL had been originally designed with only the ZigBee network layer primitives and IEEE 802.15.4 in mind.

A solution for using ZigBee application protocols and profiles over UDP/IP has been proposed in [ID-tolle-cap], which is an IETF Internet draft. This specification defines how the ZAL is mapped to standard UDP/IP primitives, enabling the use of any ZigBee profile over 6LoWPAN or standard IP stacks. This adaptation of the ZAL for use with UDP/IP is called the compact application protocol (CAP). The CAP protocol stack is shown in Figure 5.8. The functions of the ZAL and ZCL are implemented by the CAP. The data protocol corresponds to the ZigBee cluster library. The management protocol corresponds to the ZigBee device profile handling binding and discovery. Finally the security protocol implements ZigBee application sublayer (APS) security. Any ZigBee public or private application profile can be implemented over CAP in the same way that it would use the native ZigBee ZAL/ZCL. This allows for ZigBee application profiles to directly be applied to IP networks.

The main modification to the ZAL has to do with using IP hosts and IP addresses instead of IEEE 802.15.4 hosts and IEEE 802.15.4 addresses. ZigBee application layer messages are placed inside UDP datagrams using the CAP data protocol instead of ZigBee network layer frames. To receive unsolicited notifications CAP listens to a well-known UDP port. The ZAL identifies nodes by their 64-bit or 16-bit IEEE 802.15.4 MAC address. In CAP this is replaced by a CAP address record which can contain an IPv4 address plus UDP port, IPv6 address plus UDP port, or a fully qualified domain name plus UDP port.

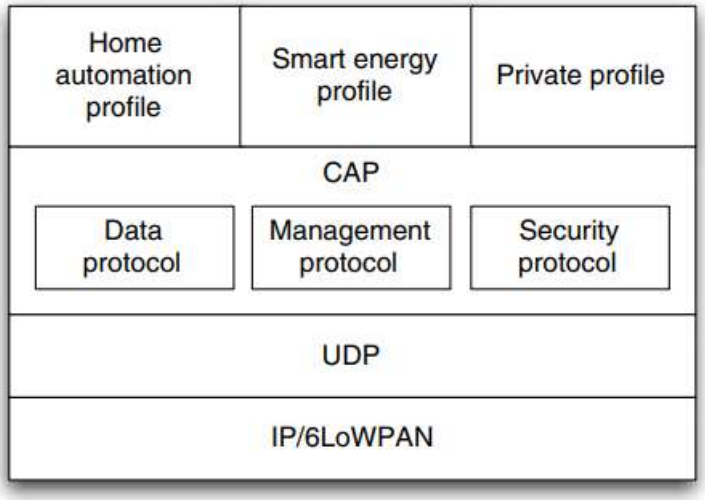

Fig 4.3.4 The CAP protocol stack.

It is assumed that the CAP is configured with the IP address and port of the discovery cache, trust center, binding coordinator and binding cache during bootstrapping. These can be configured manually, using DHCP, a special DNS entry or using a CAP server discovery message.

The CAP protocol is simply ZigBee application layer APS frames placed in UDP, with all the standard options and extensions. The APS delivery modes are mapped to IP unicast and broadcast delivery, and groupcast is reduced down to broadcast. CAP supports secure transmission and the use of APS acknowledgments, which provide limited application protocol reliability. The CAP data protocol is contained within the APS payload and it contains the ZCL command frame, with support for all ZCL command types. None of the ZCL commands require modification for use with CAP. The CAP management protocol modifies the ZigBee device profile command frame to remove IEEE 802.15.4 specific frames or to modify the address where possible. The ZigBee security and key management features are implemented by the CAP security protocol.

3. Service discovery

Service discovery is an important issue in Wireless Embedded Internet applications, where devices are autonomic – also requiring the autoconfiguration of applications. Service discovery is used to find which services are offered, what application protocol settings they use, and at what IP address they are located. Typical protocols used for service discovery on embedded devices includes the service location protocol (SLP), universal plug-n-play (UPnP) and devices profile for web services (DPWS). Some application protocols such as ZigBee CAP or MQTT-S have their own built-in discovery features. Frameworks such as OGC or SENSEI also have built-in service discovery and description mechanisms.

The service location protocol is used for general service discovery over IP networks. SLP needs optimizations in order to be effectively used with 6LoWPAN because of the size of typical messages. There has been a proposal for a simple service location protocol (SSLP) [ID-6lowpan-sslp], which provides a simple, lightweight protocol for service discovery in 6LoWPAN networks. Such a protocol could be easily interconnected with SLP running on IP networks by an SSLP translation agent located on an edge router – thus allowing 6LoWPAN services to be discovered from outside the LoWPAN and vice versa. SSLP supports most of the features of SLP, including the optional use of directory agents. The SSLP header format consists of a four-byte base header followed by specific message fields. As in SLP, service types, scopes and URLs are carried as strings. Strings used with a scheme like SSLP should be kept as short as possible.

UPnP is a protocol aimed at making home devices automatically recognizable and controllable as specified in [UPnP]. UPnP makes use of three protocols: the simple service discovery protocol (SSDP) for discovering devices, the generic event notification architecture (GENA) for event notification and SOAP for controlling devices. Devices descriptions are stored as XML and are retrieved using HTTP after initial discovery using SSDP. UPnP is not directly applicable to

6LoWPAN devices because of its dependence on broadcast along with XML- and HTTP-based descriptions and protocols. SSDP may be applicable directly over 6LoWPAN as it is similar to SSLP. It may be possible to use UPnP, or a subset of it, over 6LoWPAN with web service compression and binding applied to UPnP descriptions and protocols. However, this would require a special version of UPnP to be specified for use over 6LoWPAN and similar networks.

The devices profile for web service (DPWS) describes a basic set of functionality to enable embedded IP devices with web-service-based discovery, device descriptions, messaging and events [DPWS]. The objectives of DPWS are similar to those of UPnP, but DPWS uses a pure web-service approach. DPWS has recently been standardized under OASIS. As DPWS descriptions are XML and all messaging is based on XML/HTTP/TCP it would require web service compression and binding, along with simplification, in order to be used over 6LoWPAN. DPWS has been gaining in popularity for use in enterprise and industrial systems as devices using DPWS can be automatically integrated into back-end systems based on web services.

4. Simple network management protocol (SNMP)

Network management is an important feature of any network deployment, and a certain amount of management is necessary even for autonomous wireless embedded devices. There are several ways of performing management in IP networks, e.g. the simple network management protocol (SNMP), web services or proprietary protocols. SNMP is a standard for the management of the network infrastructure and devices in IP networks. It includes an application protocol, a database schema and data objects. The current version is SNMPv3, specified in [RFC3411]–[RFC3418]. SNMP exposes variables to a management system which can be GET or in some cases SET in order to configure or control a device. The variables exposed by SNMP are organized in hierarchies called management information bases (MIBs).

The polling approach used by SNMP (GET messages) is the biggest drawback of the approach. Polling approaches don't work for battery-powered LoWPAN Nodes which use sleep schedules, and blindly polling for statistics (which may not have changed) creates unnecessary overhead. An event-based approach would need to be added to SNMP for applicability to 6LoWPAN management.

The suitability of using SNMPv3 with 6LoWPAN has also been analyzed in [ID-snmp-optimizations], which found that optimizations are needed to reduce the packet size and memory cost. Furthermore a MIB has been specified for 6LoWPAN in [ID-6lowpan-mib]. The following optimizations for SNMPv3 have been identified in these drafts:

- Currently SNMPv3 requires the handling of payload sizes up to 484 bytes, which creates too much overhead for managing large 6LoWPANs.
- The SNMPv3 header is variable in size, and needs to be optimized for 6LoWPAN. Only a minimal subset of functionality should be supported and the header size should be limited.
- The binary encoding rules (BER) of the payload use variable length fields. For 6LoWPAN, fixed length fields or more compact encoding may be necessary.
- Payload compression and aggregation may be needed for 6LoWPAN.
- To reduce memory requirements the maximum size of SNMP messages should be limited. Furthermore MIB support should be carefully chosen.

5. Real-time transport and sessions

The real-time transport protocol (RTP) [RFC3550] is used for the end-to-end delivery of real-time data. RTP is designed to be IP-version- and transport-independent, and can be used over both UDP and TCP. The base RTP header provides basic features for end-toend delivery: payload-type identification, a sequence number and a timestamp. The RTP header format is shown in Figure 5.9. RTP does not by itself provide any kind of QoS, but it is able to help deal with out-of-order packets and jitter with the sequence number and timestamp fields, respectively. The accompanying real-time control protocol (RTCP) is used during an RTP session to provide feedback on the QoS of RTP data delivery, to identify the RTP source, adjust the RTCP report interval and to carry session control information. RTP uses the concept of

profiles, which define possible additional headers, features and payload formats for a particular class of application. The RTP audio video profile (AVP) specifies the profile for common audio and video applications.

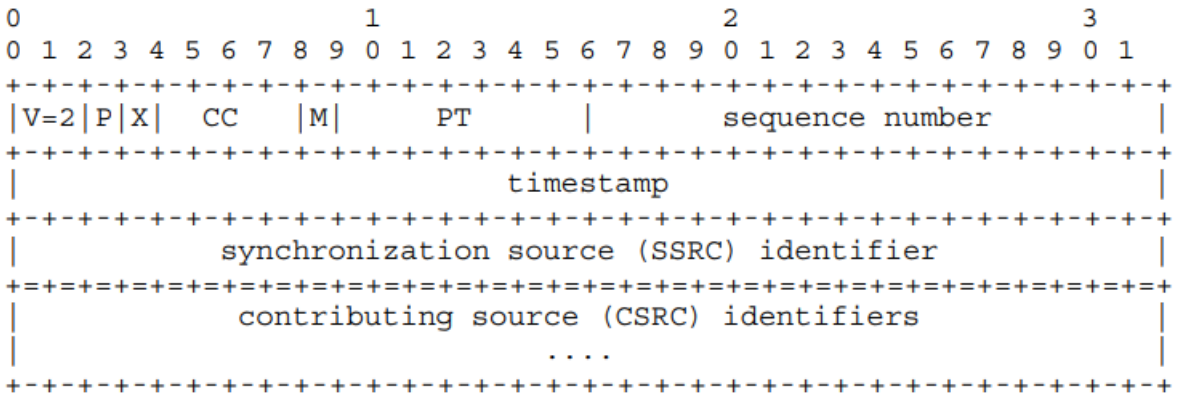

Fig 4.3.5 The RTP base header.

RTP is applicable also over 6LoWPAN for the transport of real-time data. As RTP makes use of UDP, is IP-version-independent and has a fairly compact header format, it is directly usable without modification. Existing profiles such as AVP are useful for the streaming of low-rate audio and video using the most efficient codecs, and custom profiles can be created for the transmission of e.g. sensor data streams.

Although RTP can be used to delivery and monitor real-time data streams, it requires that the sender and receiver somehow know about and find each other. Although this may be the case in specialized embedded applications of RTP over 6LoWPAN, the automatic negotiations of real-time sessions or other messaging may be very useful. The session initiation protocol (SIP) [RFC3261] was designed for establishing, modifying and tearing down multimedia sessions over IP. SIP is widely used for Voice-over-IP (VoIP) applications, and forms the backbone for the IP Multimedia System (IMS) upon which future cellular.

services will be built. The SIP design is similar to that of HTTP, in that it uses a humanreadable header format. SIP can be used over either UDP or TCP, can be handled by intermediate proxies, and provides identifiers for dealing with mobility. SIP exchanges are typically performed between SIP user agents (e.g. embedded devices) and servers. Typical methods include REGISTER, INVITE, ACK and BYE. SIP uses a separate session description protocol (SDP) to negotiate media types. An example SIP header for an INVITE from is shown below:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

The SIP header and body format is typically too large for efficient use over 6LoWPAN. But as SIP can be applied to session setup, alarms, events and IMS integration, it has valuable use in low-power embedded networks. One solution for using SIP with sensor networks is TinySIP, which defined alternative messages for use with TinyOS networking that were then mapped to SIP by a gateway application [TinySIP].

6. Industry-specific protocols

This section gives an overview of industry-specific application protocols that can be used over IP, and are relevant for Wireless Embedded Internet applications using 6LoWPAN. Building automation and energy are good examples of industries that have traditionally specified their own application protocols and formats. These are enterprise applications where system integrators make use of equipment from multiple vendors together with backend computer systems to achieve large deployments. The need for common application protocols and formats is obvious in such an environment. As communication technology has evolved, industry-specific protocols have steadily evolved to enable use over IP. Many industry-specific protocols may be used over 6LoWPAN, whereas others may require the addition of compression, IPv6 support or UDP support for example. Next we examine some common building automation and energy industry application protocol standards.

BACnet: BACnet is a network and application protocol format with support for a wide range of communication technologies including Ethernet, RS232, RS-485 and LonTalk. BACnet includes support for use over UDP/IP known as BACnet/IP. The standard BACnet network and application protocol frames are carried over UDP by encapsulating them in a BACnet virtual link layer (BVLL). This adaptation binds BACnet to an underlying communication technology. Currently there is only a BVLL defined for use with IPv4, but an extension of that for IPv6 is straightforward. IPv6 support is current under design in the BACnet IP working group. BACnet makes use of unicast, broadcast and optionally multicast IP communications, and is based on an object-oriented design. BACnet objects have properties that are acted upon using protocol services. These protocol services include Who-Is, I-Am, Who-Has and I-Have used for device and object discovery along with read-property and write-property used for data access.

As BACnet was designed for a whole range of low-bandwidth links, and has native support for IPv4, it will be a useful protocol over 6LoWPAN for building automation applications. The adaptation of BACnet/IP for use with 6LoWPAN will require IPv6 support, and should make careful use of multicast – keeping in mind its overhead on wireless multihop mesh networks. The performance and possible optimization of BACnet service protocol traffic over low-power wireless mesh networks should also be studied as BACnet currently assumes wired links.

KNX: The KNX protocol supports several different communication media: twisted pair, powerline, radio frequency (RF) and IP. Twisted pair is the most common KNX medium and is typically installed when a building or house is constructed. Powerline and RF are often used when retrofitting existing buildings. The KNX RF specification uses its own framing over an 868 MHz radio at 16 kbit/s. KNX networks can theoretically support up to 64k devices using twisted pair, power-line or RF communications.

KNX has some support for IP, also known as KNXnet/IP. KNX IP support is part of a framework called ANubis (advanced network for unified building integration and services) and among many other things provides a way to encapsulate KNX frames over IP. The purpose of this is currently to interconnect KNX networks using IP networks, to enable remote monitoring and to interconnect with other systems such as BACnet. This IP encapsulation technique could also be usefully applied over low-power IP and 6LoWPAN networks to KNX devices themselves. This would need a considerable amount of development and standardization.

oBIX: oBIX provides a web service interface, which can be used to interact with any building automation network including BACnet, KNX, Modbus, Lontalk or proprietary networks using the oBIX XML format. The format provides normalized representation of constructs common to building automation protocols: points (scalar value, status), alarms and histories. It has an extensible meta-format which can be used to describe any system. oBIX provides a low-level object model for working with these constructs. Usually these are accessed by using generic oBIX constructs, for example by an enterprise developer.

oBIX is web service binding agnostic. It can be used over both SOAP and directly over HTTP in a REST style. It represents objects with URLs and object state with XML. oBIX may be applicable also for use directly in building

automation networks thanks to 6LoWPAN. Instead of running a control network specific building automation protocol such as BACnet/IP or KNX over 6LoWPAN, oBIX together with compression and UDP/IP binding may be a solution. Careful design of the oBIX objects and elements used would be important to keep packet sizes reasonable.

ANSI C12.19

The ANSI C12.18 standard defines a point-to-point optical interface along with the protocol specification for electric metering (PSEM). The ANSI C12.19 standard defines utility industry end device data tables, which are accessible using PSEM. ANSI C12.21 specifies a communication protocol over modem lines. Finally a new standard ANSI C12.22 specified the interfacing of devices to any data communications network including Internet protocols. This furthermore specifies the extended protocol specification for advanced metering (EPSEM).

The ANSI C12 PSEM protocol and device data formats (tables) were designed with simple embedded electric meters and low-bandwidth point-to-point links in mind. Therefore PSEM uses a compact binary encoding for all its protocol and data fields. The typical frame size expected by point-to-point links is 64 bytes in the ANSI C12.22 specification. The ANSI C12.22 specification is therefore well suited for use over 6LoWPAN. The specification includes a simple example of using the application layers with a TCP/IPv4 communication stack. The standard may also be used with a UDP/IPv6 stack, as the protocol includes acknowledgment features for reliability along with application-layer segmentation and reassembly. Application layer security features are built in elements of the protocol.

APPLICATION PROTOCOLS

DLMS/COSEM

The device language message specification (DLMS) is a European model of communication exchange used to interact with utility end devices for meter reading, tariff and load control. It uses the companion specification for energy metering (COSEM) as a data exchange format and protocol. Together, they are standardized by the IEC under the 62056 series [IEC62056]. The DLMS Association [DLMS] actively promotes the maintenance and use of the standards. IEC 62056 is similar in function to its American counterpart, ANSI C12. It makes use of local optical or current loops to meters, point-to-point serial modems, or IP networks for communication. The COSEM protocol defines the application layer in IEC 62056-53. As with ANSI C12, an object model is used to access information as defined in IEC 62056- 61. The use of COSEM over IPv4 is described in detail in IEC 62056-47. This specification describes transport using both UDP and TCP, which enables use over 6LoWPAN. Although the specification describes the use of IPv4, the changes needed for IPv6 are minor. It is unclear if any changes would be needed to the standard to allow the use of IPv6. Even though the data representation format is not as compact as in ANSI C12, it is still suitable for the frame size and bandwidth limitations of 6LoWPAN.