4.2 Protocol Paradigms

There is a basic set of paradigms by which most Internet application protocols function. These include the end-to-end paradigm, streaming, sessions, publish/subscribe and finally web services.

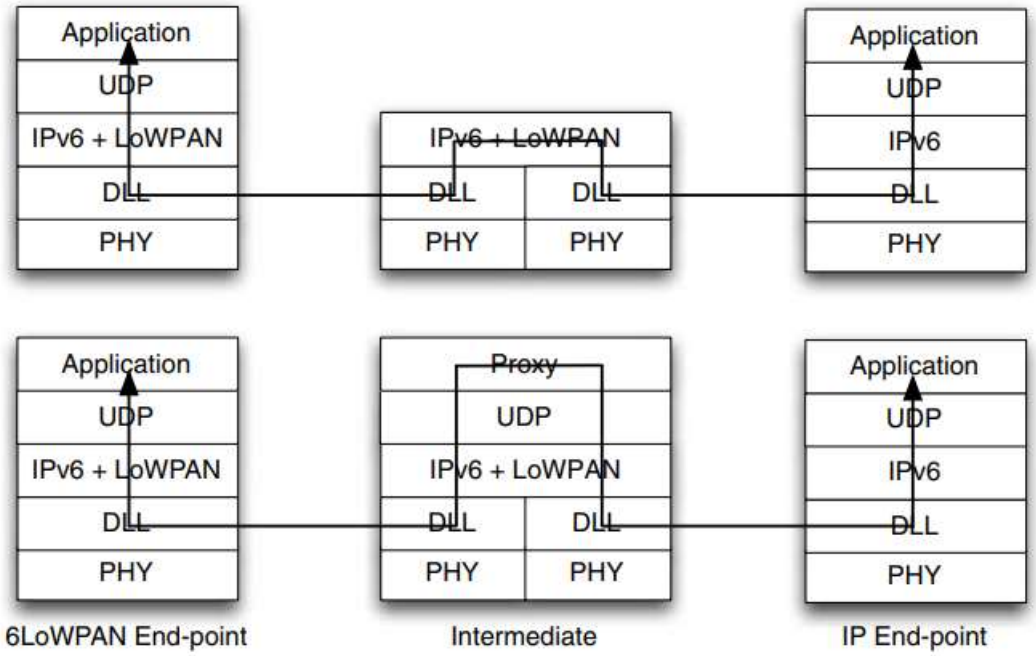

Fig 4.2.1 End-to-end (above) and proxied (below) application protocol paradigms.

- **End-to-end**: The Internet socket model is based on the use of the underlying transport layer to provide a transparent datagram or byte stream service between application processes, or so-called application end-points. When considering the application layer this can be called an end-toend paradigm where only the end-points participate in the application protocol exchanges. Some application protocols also include the possibility for intermediate nodes to inspect, cache or modify application protocols. Here we refer to this as proxying. An example would be an HTTP proxy that performs web-page caching.
- **Real-time streaming and sessions**: Internet protocols already provide a good framework for working with real-time streams, which can be employed by 6LoWPAN applications as well. The real-time transport protocol (RTP) encapsulates streams with appropriate timestamp and sequence information, while the companion RTP control protocol (RTCP) is used to control the stream. If a relationship between the sender(s) and receiver(s) of a stream needs to be automatically setup and configured, the session initiation protocol (SIP) can be employed.
- **Publish/subscribe**: Publish/subscribe (also known as pub/sub) is an asynchronous messaging paradigm in which publishers send data without knowing who the receiver is, and receivers subscribe to data based on the topic or content of the data. Pub/sub can be implemented using centralized brokers that match publishers and subscribers in a store-and-forward fashion, or in a distributed manner where subscribers filter messages directly from publishers. This decoupling of the application end-points allows for scalability and flexibility. For the Internet of Things, pub/sub plays an important role as most applications are data-centric, i.e. it is not so important who sends data, but rather what the data is. One good example of a pub/sub protocol is the MQ telemetric transport (MQTT), which is a broker-based enterprise pub/sub protocol for telemetry, used widely by IBM
- **Web service paradigms**: Web services are defined by the W3C as a software system designed to support interoperable machine-to-machine communications over a network [WS]. Web services as a whole commonly work between clients and servers over HTTP. There are two different forms of web services: service-based

(SOAP) web services and resource-based (REST) web services. Both forms of web services will play an important roll in 6LoWPAN applications.

Service-based web services use XML following the SOAP format to provide remote procedure-calls (RPCs) between clients and servers [SOAP]. These SOAP messages and sequences can be described using the web services description language (WSDL) [WSDL]. This paradigm is widely used in enterprise machine-to-machine systems. A SOAP interface is typically designed with a single URL that implements several RPCs called methods(a good analogy is a verb) as in the following example:

The representational state transfer (REST) paradigm instead models objects as HTTP resources (a good analogy is a noun), each with a URL accessible using standard HTTP methods [REST]. These interfaces can be described using the web application description language (WADL). With the release of WSDL 2.0, REST-based interfaces can alternatively be defined in a similar way to SOAP interfaces. This REST paradigm is widely used on the Internet between web sites. The content of REST HTTP messages can be of any MIME content, although XML is common in machine-to-machine applications. An example of a REST design follows, where objects are accessible using standard HTTP GET, POST, PUT and DELETE methods. In this example GET would be used on all resources to request the value, and POST would be used to set a new value for a parameter:

http://sensor10.example.com/sensors/temp http://sensor10.example.com/sensors/light
http://sensor10.example.com/sensors/acc-x http://sensor10.example.com/sensors/acc-y
http://sensor10.example.com/sensors/acc-z http://sensor10.example.com/config/sleeptime
http://sensor10.example.com/config/waketime http://sensor10.example.com/config/enabled
http://sensor10.example.com/config/samplerate