IOT Multiprotocols in Application layer

Dev Bhattacharya
dev_bhattacharya@ieee.org

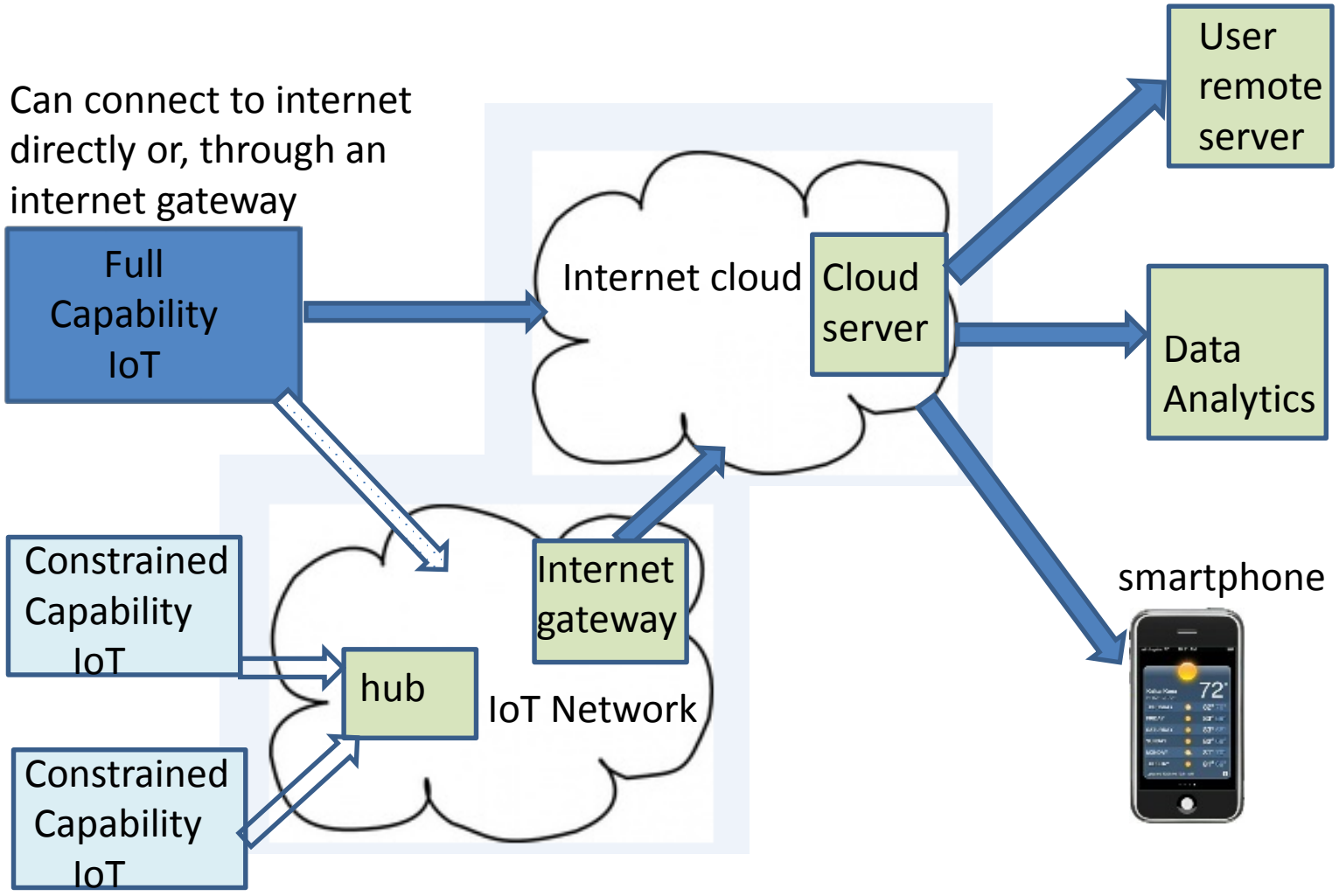
Outline

- Architecture of Internet of Things(IoT)
- Simplified IoT System Architecture
- Multiple layers of IoT
- Communication Problems in IoT
- Solution of IoT communication problems
- HTTP/REST protocol & architectural diagram
- MQTT protocol & architectural diagram
- XMPP protocol & architectural diagram
- DDS protocol & architectural diagram
- Scalability of IoT communication
- Conclusion

Architecture of IoT

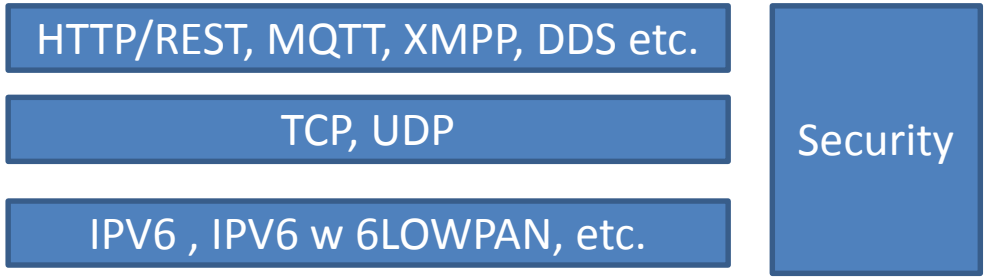
- Full capability IoT
  - With MCU & Network
  - The IOT application runs on the device. example: Smart Thermostat
- Constrained capability IoT
  - Various combinations of MCU & Networking
    - With MCU (various processing capabilities)
    - Without MCU (goes through a smart hub that provides MCU processing)
    - With networking
    - Without networking (goes through a hub that provides networking)
  - The IOT application runs as a web application on the cloud or, on the cloud connected dedicated server

Simplified IoT System Architecture

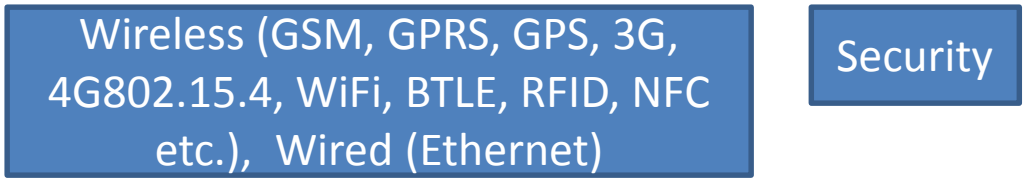


Multiple layers of IoT

- Higher layer protocols
  - Application
  - Transport
  - Network



- Lower layer protocols
  - Link layer



Communication problems in IoT

Following are some key communication problems we encounter when we connect devices, servers, data centers in a distribution network (LAN or, WAN) via a range of wired and wireless networks

- Inter Device Communication
  - Message exchanges between devices on a LAN, WAN
- Device to Cloud Communication
  - Message exchanges between devices via the internet
  - Message exchanges between device and data centers
- Inter Data Center Communication
  - Message exchanges between data centers

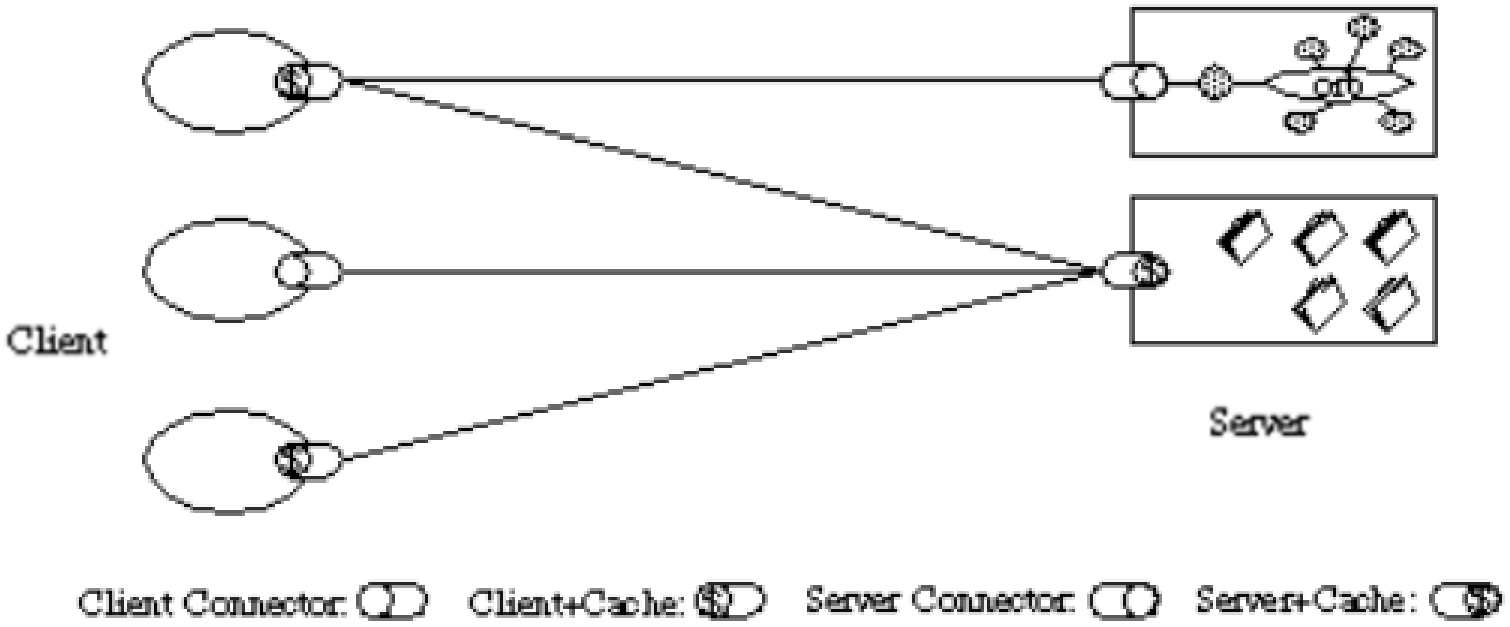
Solutions of IoT Communication problems

- Application layer protocols with underlying TCP and UDP transport protocols solves the problem
  - XMPP, MQTT, REST/HTTP runs over TCP transport
  - DDS normally runs over UDP transport as underlying transport used by DDSI protocol, however there are implementations that support DDSI over TCP from some vendors
- Device to data center and inter data center communication
  - XMPP, MQTT use communication via brokers through TCP/IP connection
  - RESTful applications can implement request/reply message exchanges from a client to server in a data center using HTTP
  - DDS implementations can support a broker based or, broker less architecture, communications can happen between one publisher device with many subscriber devices or, even data centers

HTTP/REST Application layer Protocol

- HTTP/REST
  - RESTful style architectures conventionally consists of clients and servers
  - Clients initiate requests to servers, servers process requests and returns responses
  - The protocol is primarily client/server, stateless, layered, and supports caching
  - REST was initially described in the context of HTTP but it may not be limited to HTTP
  - Universal availability of HTTP stacks for various platforms has allowed REST to emerge as a predominant web API design model
  - Backend processing ability of servers determine response time
  - Message latencies of several seconds
  - RESTful HTTP over TCP is used mainly for connecting consumer premise devices (example: home energy management)

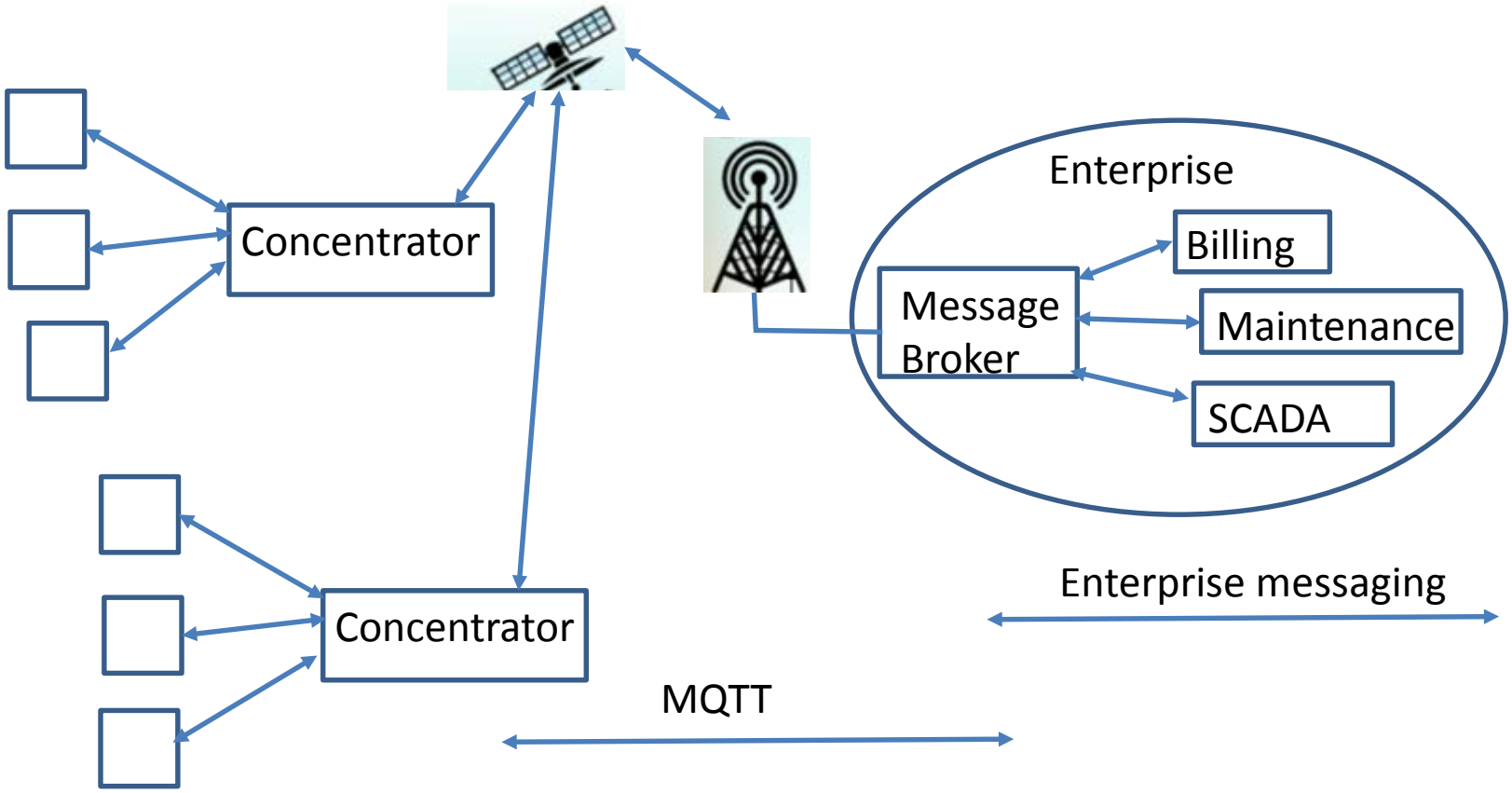
RESTful architecture diagram



MQTT Application layer Protocols

- MQTT (Message Queue Telemetry Transport)
  - MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol.
  - It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.
  - For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios. It is also ideal for mobile applications because of its small size, low power usage, minimised data packets, and efficient distribution of information to one or many receivers.
  - A hub-and-spoke architecture is natural for MQTT. All the devices connect to a data concentrator server, like IBM's new MessageSight appliance. Protocol works on top of TCP to avoid data loss and provides a simple, reliable stream. Since the IT infrastructure uses the data, the entire system is designed to easily transport data into enterprise technologies like ActiveMQ and enterprise service buses (ESBs).
  - Client/server model, where every sensor is a client and connects to a server, known as a broker, over TCP. Telemetry or, remote monitoring of large number of constrained devices.
  - MQTT is message oriented. Every message is a discrete chunk of data (small as 2 byte), opaque to the broker.
  - The publisher subscriber model allows MQTT clients to communicate one-to-one, one-to-many and many-to-one..
  - Publish/subscribe messaging protocol designed for lightweight M2M (constrained or, high latency) network communications.

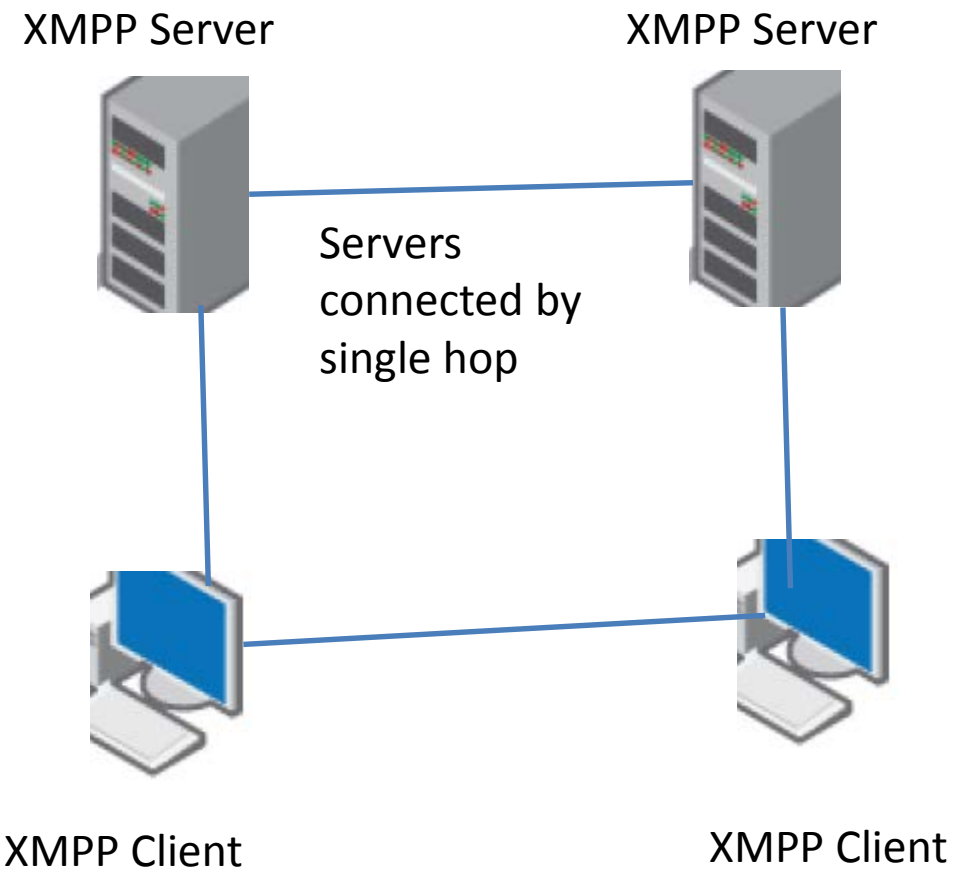
MQTT architecture diagram



XMPP Application layer Protocol

- XMPP (Extensible Messaging and Presence Protocol)
  - XMPP was developed for instant messaging (IM) to connect people to other people via text messages.
  - XMPP uses the XML text format as its native type, making person-to-person communications natural.
  - XMPP runs over HTTP on top of TCP. Its key strength is a name@domain.com addressing scheme that helps connect the needles in the huge Internet haystack.
  - In the IoT context, XMPP offers an easy way to address a device. This is especially handy if that data is going between distant, mostly unrelated points, just like the person-to-person case. It's not designed to be fast. In fact, most implementations use polling, or checking for updates only on demand.
  - A protocol called BOSH (Bidirectional streams over Synchronous HTTP) lets severs push messages.
  - XMPP response time is in seconds and it is good for people to people communication
  - XMPP provides a great way, for instance, to connect your home thermostat to a Web server so you can access it from your phone. Its strengths in addressing, security, and scalability make it ideal for consumer-oriented IoT applications.

XMPP architecture diagram



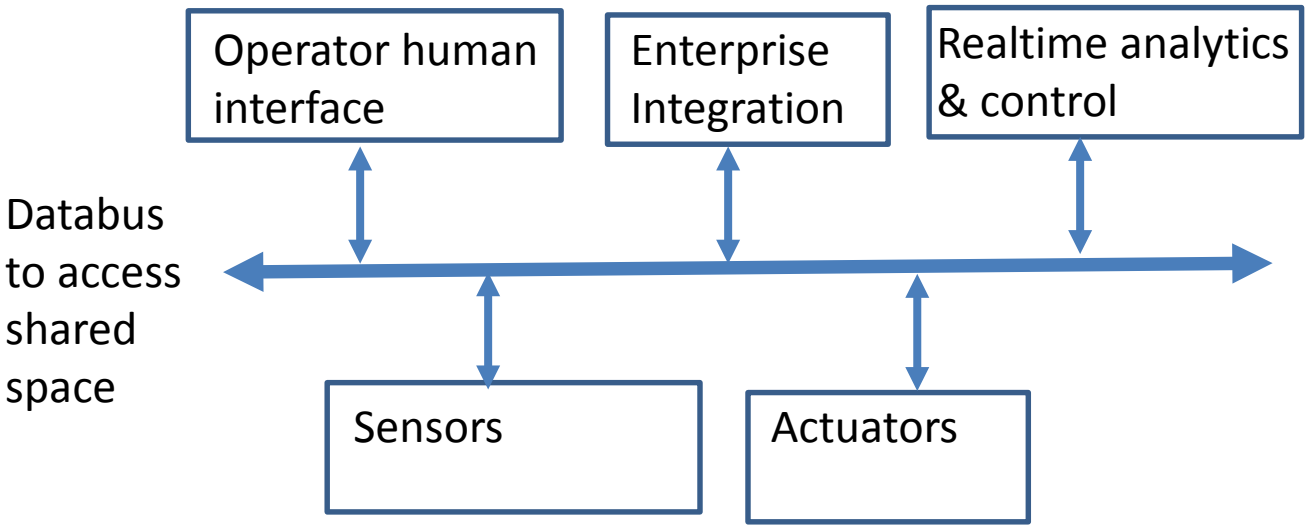
DDS Application layer Protocol

- DDS (Data Distribution Service for Real time Systems)
  - DDS implements direct device-to-device “bus” communication with a relational data model. Some companies call this a “DataBus” because it is the networking analog to a database. Similar to the way a database controls access to stored data, a data bus controls data access and updates by many simultaneous users. This is exactly what many high-performance devices need to work together as a single system.
  - High-performance integrated device systems use DDS. It is the only technology that delivers the flexibility, reliability, and speed necessary to build complex, real-time applications. Applications include military systems, wind farms, hospital integration, medical imaging, asset-tracking systems, and automotive test and safety. DDS connects devices together into working, distributed applications
  - Object management group’s DDS is a data centric publish and subscribe technology to address the data distribution requirements of mission critical systems such as financial trading, air traffic control etc.
  - DDS targets devices that directly use device data. It distributes data to other devices While interfacing with the IT infrastructure is supported, DDS’s main purpose is to connect devices to other devices
  - It enables scalable, real-time, reliable and high performance and interoperable data exchanges (by DDS interoperability wire protocol) between publishers and subscribers.

DDS Application layer Protocol (Contd.)

- DDS (Data Distribution Service for Real time Systems)
  - DDS standard
    - Data centric publish and subscribe(DCPS) layer allows configuration from small scale to large scale systems
      - Use APIs that present a set of standardized “profiles” targeting real time information availability from small scale to large scale systems
    - A DDS interoperability wire protocol (dynamic DSI)
    - An extensible and dynamic topic types for DDS standard.
  - DDS is both language and OS independent
    - DCPS APIs has been implemented in a range of different programming languages (C, C++, Java, Javascript etc.
    - Using standardized APIs helps ensure that DDS applications can be ported easily between different vendor’s implementations
  - DDS specifies a DDS interoperability(DDSI) wire protocol that helps multiple DDS applications to interoperate
    - A wire level protocol refers to the mechanism of transmitting data from point to point
    - Wire protocol is used to describe a common way to represent information at application level
    - The protocol also supports automatic “Discovery“ that allows DDS participants to declare the information they can provide or, receive including data type & QoS
    - Protocol automatically connect publishers to subscribers and this simplifies process of configuring systems with many nodes and many devices exchanging data

DDS architecture



Scalability of IoT Communication

- DDS
  - DDS's connectionless architecture scales well when number of applications on the node producing and consuming the data increases
  - DDS's shared memory based deployment architecture provides low latency intercore communication and this allows more nodes to efficiently share data in a deterministic manner.
- XMPP
  - XMPP application can tolerate latency and hence can be scaled to large number of nodes by having more servers in the system.
- MQTT
  - MQTT is a large network of small devices over satellite and cellular networks. MQTT has a latency in seconds and data packets are small and hence can be scaled as long as bandwidth is available.
- REST/HTTP
  - RESTful implementation has latency in seconds and can be scaled with more servers

Conclusion

- Protocol choice depends on deployment requirement
  - Low bandwidth satellite communication for large number of nodes calls for MQTT
  - Fast real time data driven application will require DDS
  - Need to use TCP transport for reliable communication where you cannot tolerate loss of data
- Some protocols are more suited for constraint configuration
  - MQTT allows concentrators to gather data from constraint devices with low memory and low code footprint
- Complex system might use a combination of protocols