Middleware architecture of SCADA

The concept of MAN (M2M area network) was introduced in 3GPP/ETSI's MTC (Machine Type Communication) specification. This concept also applies to other pillar segments of IoT. However, not all IoT applications will use a cellular network. In fact, most of the traditional SCADA (Supervisory Control And Data Acquisition) applications have been using local wireline networks for communications. The remote terminal units (RTUs), programmable logic controllers (PLCs), or even process control systems (PCSs) communicate to the SCADA middleware server via gateways (similar to MAN but all wired) that aggregate data from different wired field buses. The SCADA system is accessed in a LAN environment (sometimes xDSL, cable, WiFi, or WiMax can be used) before it is integrated into the corporate back office system.

Figure 2.6 depicts the role of SCADA middleware in such a scenario in more detail. Companies providing such SCADA middleware products include the following:

Central Data Control: CDC provides the software platform Integra, which utilizes data agents to translate protocols from different building system components into single management system.

Elutions: Its Control Maestro product has a SCADA heritage. SCADA may be best known for industrial processes but is also deployed for infrastructure (water treatment plants, gas pipelines, etc.) as well as facility systems. Control Maestro is web-based, uses human–machine interfaces (HMI), and is able to deliver real-time and historical information.

Richards Zeta: RZ's middleware solution is a combination of system controllers and software.

Tridium: It provides the Niagara Java-based middleware framework and JACE hardware controllers. The Niagara platform provides protocol translation for a range of systems and the tools to build applications

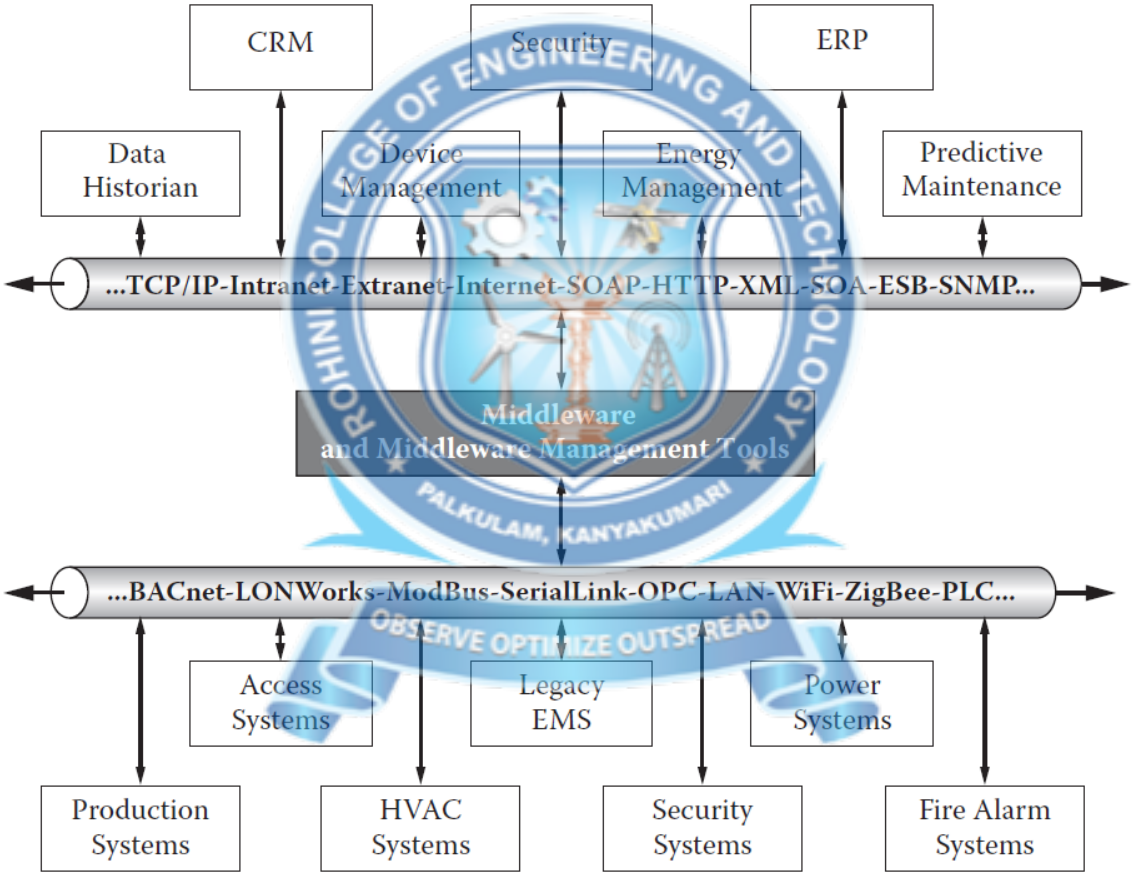

Fig.2.8 SCADA middleware architecture

[Ref: Honbo Zhou, "Internet of Things in the cloud: A middleware perspective", CRC press, 2012 *]*

With the development of wireless technologies, systems have been developed that blend wireless with wired communication in SCADA applications. SensiLink™ is a middleware and software suite from MeshNetics that links wireless sensor networks with SCADA systems. Sensor data collected from the nodes is channeled through RS232, RS485, USB, Ethernet, or GPRS gateway to the SensiLink server.

RFID Middleware:

RFID networking shares a similar three-tiered communication architecture as shown in Figure 2.9. RFID readers are the gateways similar to MAN. Data from the readers go to the corporate LAN and then are transmitted to the Internet as needed. However, just like the scenarios of M2M and SCADA, most current RFID systems stop at the corporate LAN level and are IoT systems only.

RFID middleware (including the edge middleware or edge ware) is currently no doubt the most well-defined, comprehensive, standardized middleware compared with the other three pillar segments of IoT. Before 2004, an RFID middleware-based system was defined by EPC global, which included:

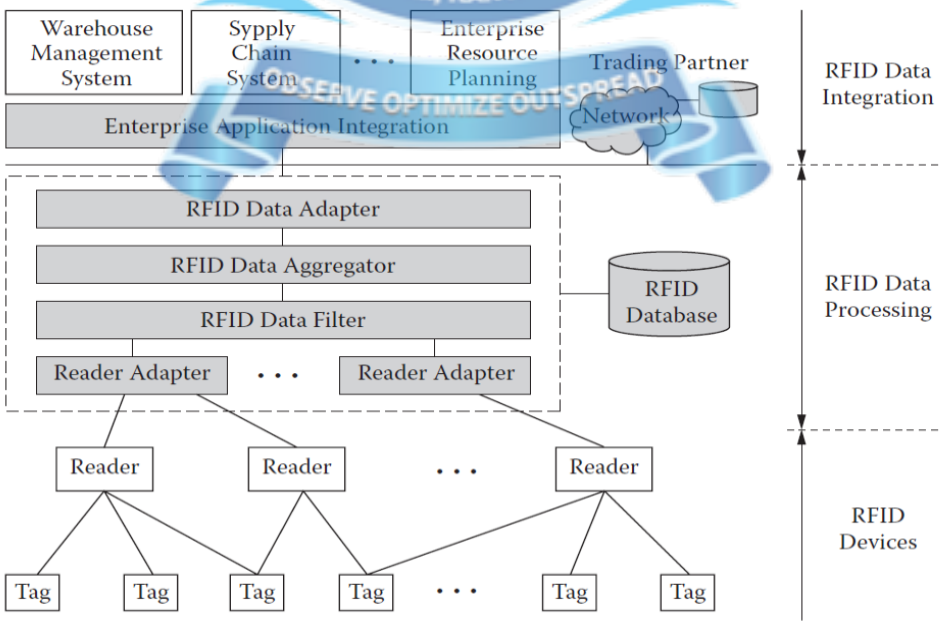

Fig.2.9 RFID architecture

[Ref: Quan Z. Sheng et.al, "RFID Data Management: Issues, Solutions, and Directions," in Lu Yan, Yan Zhang, Laurence T. Yang, and Huansheng Ning (Eds.), The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems, New York: Auerbach Publications, 2008.)

- A format for the data called physical markup language (PML), based on XML
- An interface to the servers containing PML records
- A directory service called ONS (object naming service), analogous to the DNS. Given a tag's EPC, the ONS will provide pointers to the PML servers containing records related to that tag.

An example of commercial RFID middleware product is IBM's WebSphere Sensor Events. WebSphere Sensor Events delivers new and enhanced capabilities to create a robust, flexible,
and scalable platform for capturing new business value from sensor data. WebSphere Sensor Events is the platform for integrating new sensor data, identifying the relevant business events from that data using situational event processing, and then integrating and acting upon those events with SOA business processes.

WSN Middleware

WSN middleware is a kind of middleware providing the desired services for sensor-based pervasive computing applications that make use of a WSN and the related embedded operating system or firmware of the sensor nodes. In most cases, WSN middleware is implemented as embedded middleware on the node. It should be noted that while most existing distributed system middleware techniques aim at providing

transparency abstractions by hiding the context information, WSN-based applications are usually required to be context aware.

A complete WSN middleware solution should include four major components: programming abstractions, system services, runtime support, and quality of service (QoS) mechanisms. Programming abstractions define the interface of the middleware to the application programmer. System services provide implementations to achieve the abstractions. Runtime support serves as an extension of the embedded operating system to support the middleware services. QoS mechanisms define the QoS constraints of the system. The system architecture of WSN middleware is shown in Figure 2.10.

Middleware for WSN should also facilitate development, maintenance, deployment, and execution of sensing-based applications. Many challenges arise in designing middleware for WSN due to the following reasons:

- Limited power and resources, e.g., battery issues
- Mobile and dynamic network topology
- Heterogeneity, various kinds of hardware and network protocols
- Dynamic network organization, ad-hoc capability

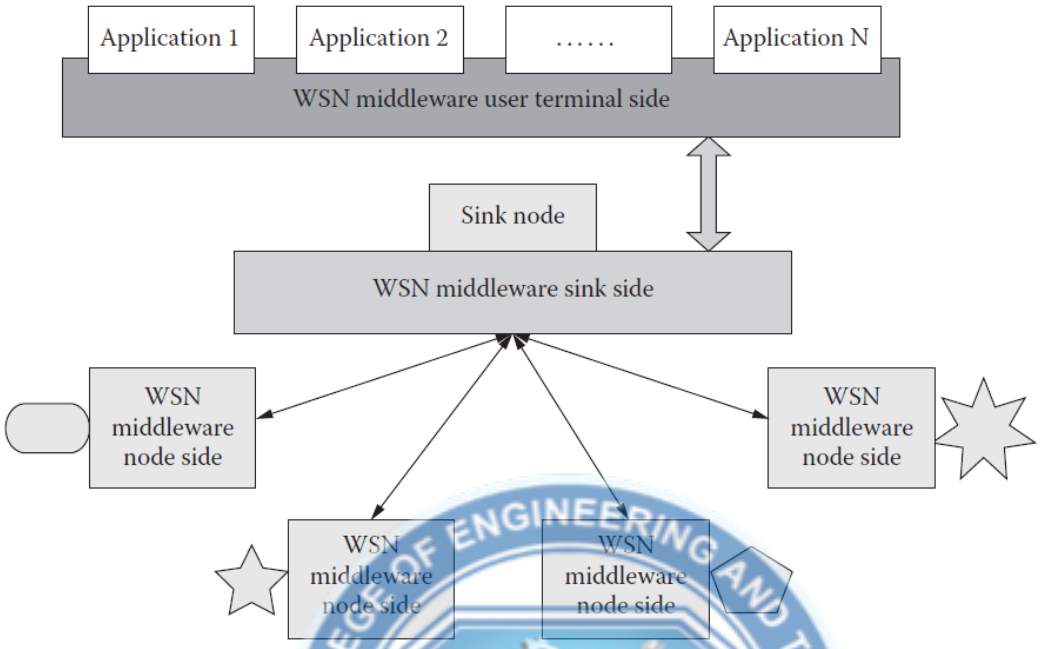

Fig.2.10 WSN middleware architecture

[Ref: Honbo Zhou, "Internet of Things in the cloud: A middleware perspective", CRC press, 2012]

WSN middleware is designed using a number of approaches such as virtual machine, mobile agents, database based, message-oriented, and more. The WSN middleware is considered to be "proactive" middleware in the middleware family. Example middleware are as follows:

MagnetOS : power-aware, adaptive; the whole network appears as a single JVM, standard Java programs are rewritten by MAGNET as network components, and components may then be "injected" into the network using a power-optimized scheme.

IMPALA: modular; efficiency of updates and support dynamic applications; application adaption with different profiles possible; energy efficient; used in the ZebraNet project for wildlife monitoring.

Cougar: represents all sensors and sensor data in a relational database; control of sensors and extracting data occurs through special SQL-like queries; decentralized implementation; message passing based on controlled flooding.

SINA (system information networking architecture): based on a spreadsheet database wherein the network is a collection of data sheets and cells are attributes; attribute-based naming; queries performed in an SQL-like language; decentralized implementation based on clustering.

MQTT-S (Message Queue Telemetry Transport for Sensors, IBM): a publish/subscribe messaging protocol for WSN, with the aim of extending the MQTT protocol beyond the reach of TCP/IP infrastructures (non-TCP/ IP networks, such as Zigbee) for sensor and actuator solutions; a commercial product.

MiLAN: This provides a mechanism that allows for the adaptation of different routing protocols; sits on top of multiple physical networks; acts as a layer that allows network-specific plug-ins

to convert MiLAN commands to protocol-specific ones that are passed through the usual

network protocol stack; can continuously adapt to the specific features of whichever network is being used in the communication.