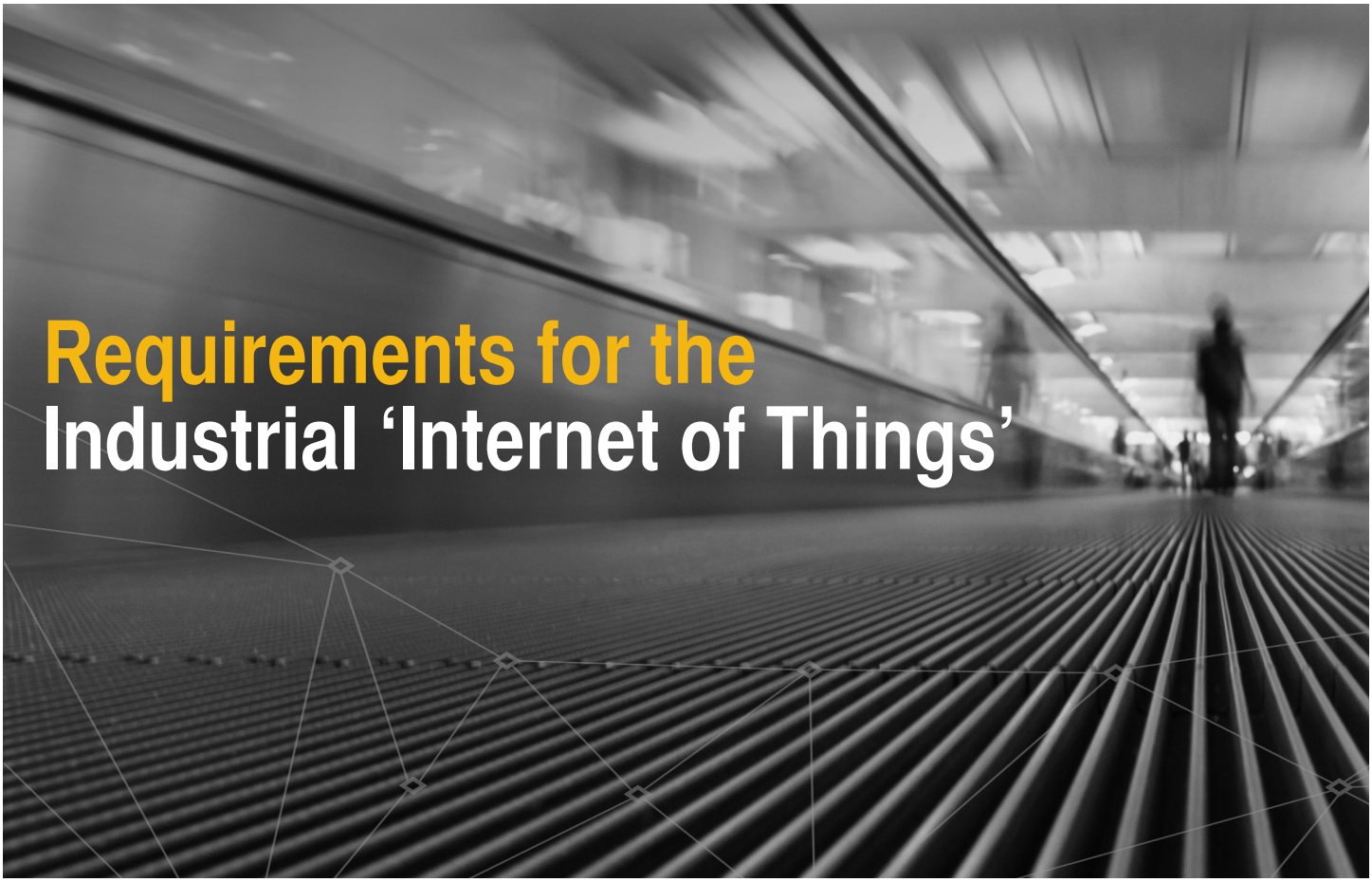
ECHELON®



Requirements for the Industrial 'Internet of Things'

Introduction

The set of applications we have envisioned for the Internet of Things (IoT) is very diverse. Internet Protocol (IP) will provide a single communications technology that can span all device-based applications: Smart Meters, Smart Grids, home energy networks, building automation networks, process control networks, factory automation networks, vehicle tracking systems, inventory management systems, lighting systems, home entertainment systems and many more.

We see three main classes of potential applications for the IoT, each with different communication requirements.

The first class of applications is the consumer IoT. These applications are characterized by a human interacting with a device. Viewing a video on a cell phone, or starting up an exercise monitor to send your statistics to your account in the cloud are examples of consumer IoT applications. In case of failure, a human is there to recover or restart the application. In the consumer IoT, communications run between client/server and are often streaming large amounts of data.

The next class of applications is a segment of the Industrial IoT, based on application monitoring. A vehicle tracking system; a system that monitors a building's mechanical systems for signs of wear, or a system that tracks mobile hospital equipment are all examples of the Industrial IoT. These applications are often referred to as Machine to Machine (M2M). This class of applications uses client/server communications and sends smaller amounts of data. For example a data record might include the device identifier, position coordinates, and a time stamp. Most important in these communications is the reliability of communications because there is no human operator or user to aid in any error recovery. That reliability is also crucial for a second reason: the items the data locates are valuable (so, too, is the knowledge of where they are at any given point). Cost is incurred when the information is not available or unreliable.

There is a third and more demanding class of Industrial IoT applications: a communications-emphasis class of Machines to Machines (M2M) communications where the application uses autonomous, peer-to-peer distributed control. In these applications, groups of nodes work together to accomplish a single task. For example, a baggage handler in an airport that senses luggage moving on a conveyor belt identifies the luggage by reading a bar code and then nudges them to the correct next conveyer belt based on the bar code. Then, further away, another node makes a routing decision as multiple conveyor belts converge. For these systems, the communications requirements are not merely client/server. Instead, the nodes act as peers on the network, each making decisions and reporting status to the other nodes. Data transfers are frequent, but typically do not convey large amounts of data. A message may merely convey temperature or the pressure or the status of a switch. Often, these systems process their tasks at rates greater than a human could, so they must run reliably and safely without human intervention. Communication failures risk added costs or can even threaten human safety. Besides performing their main task, these systems also connect to an enterprise system to issue alarms, archive historical data, and store a basis of performing analytics on the data. This connection can work via a local IP connection or could be hosted in the cloud. When communicating with the enterprise system, the communications model reverts to the client/server M2M model discussed above.

Systems performing these demanding industrial applications have been available for some time. However, they have often depended on hard wiring and purpose-built communications protocols for exchanging data and status between nodes. But now recent advances in Low Power Wireless, Power Line, and high speed multi-drop twisted pair communications technologies coupled with more compact implementations of IP are enabling a migration to IP-based nodes for even the smallest and most cost sensitive nodes within these systems.

A Look at Industrial Peer-to-Peer Distributed Control

Most distributed control systems do their work behind walls and in ceilings in buildings, or within factories where only the employees can go, so what they do, and how they do it is not well understood. However, there are more visible control systems that nearly everyone has heard of. One is the famous Bellagio fountain in Las Vegas.

The Bellagio fountain is a huge pool of water with jets that can shoot water up high and in many different directions. Each jet is actually a node on a peer-to-peer network. Colored theatrical lighting is synchronized to run at the same pace as the jets, and each light is a node on the distributed network. The water jets and lights are synchronized to music so that as the music changes, the lights move and water jets are unleashed, shooting water off at different heights and varying directions — creating a very dramatic show.

The fountain is a distributed control system, not very different from what any industrial or building control system does. Things are turned on and off in real time, motion is timed and controlled, and moving parts are synchronized. A great many of the applications we and developers are envisioning for the Internet of Things are distributed control systems just like this one. Each of these systems depends on reliability, timing accuracy, and synchronization across the network. A single late packet at the Bellagio fountain would ruin the show by causing the lights, music and water jets to lose synchronization in a very visible way, just as a late packet in a material handling system could result in a product dropping onto the factory floor and losing a company money.

A good example of the fountain's operation can be seen at: www.5min.com/Video/Learn-About-the-Bellagio-Fountains-278834664.

Today's Industrial Internet

Industrial automation today is very sophisticated. Complex manufacturing processes can run without human intervention in a "lights-out" environment — if something goes wrong, no human intervention is even possible, because automation speeds are so rapid. This high productivity and accuracy of our industrial environment is made possible because devices interact with one other at network speeds without human intervention or direct supervision.

Today, these industrial systems connect to the Internet and internal IP networks through gateways. These gateways require custom provisioning and programming to expose the necessary data to the enterprise systems. Invariably, the gateway constrains what information can pass back and forth and its configuration is difficult to evolve to support new requirements.

To provide the benefits of a common IP communications infrastructure from the simplest device all the way to the enterprise, the advanced communications requirements of these systems must be addressed.

Requirements for the Industrial Internet of Things

Requirements for the Industrial IoT are a superset of the main IoT requirements. Just like the total IoT market, the Industrial IoT market needs inexpensive nodes that work on easy to install links like wireless, power line, and simple twisted pair. These links do not always have the same reliability as is found with traditional data communication links, so there are the various problems of error detection and reliability common to the entire IoT. But the consequences for communications failures are much worse in the Industrial IoT due to the investment returns expected from flawless operation.

The stakes are high when industry is involved.

The requirements below do not apply to every Industrial IoT application. However, having a rich set of services in an IP based protocol stack allows that IP protocol stack to be used across the entire Industrial IoT application space, and application developers can depend on a common set of communications services as they implement Industrial IoT applications.

The requirements are categorized in four main areas:

- Resilience in the face of failures
- Physical Connectivity Requirements
- Security
- Control Services

Resilience in the Face of Failures

The protocol stack must recover from intermittent packet loss quickly via packet retransmission or it must report a message failure to the application.

Rationale: The new Low Power Wireless and Power line links that these networks run on have very low bandwidth compared to Ethernet connections; and, unlike Ethernet, the links are not nearly as reliable. Packets can be lost due to interference and noise or even collisions. When these events happen, and if they happen frequently, more bandwidth is needed to recover from the loss in the form of a packet retransmission. Secondly, because these systems typically have real time constraints, delivering the packet well beyond the application's timing constraints is not important or even desirable.

The communications network must be engineered such that the real-time requirements of the application are met. This involves being able to:

a) Design the network to meet response-time criteria by limiting the number of nodes per link, and tune the communications such that the network will not become overloaded.

b) Specify that a given communications transaction will either succeed or fail within a specified time, as well as guarantee that the success or failure of that transaction will be known to the application.

Rationale: In the Bellagio fountain example, a late packet would result in some node not performing its synchronizing function. On a factory floor, a material handler might drop something; in a semiconductor fab line, a wafer handler might fail to place a wafer on a probe station. Late packets mean communication failures in most control systems — there is no benefit in delivering a packet late.

There must be no single failure that can take down the communications for an entire link.

Rationale: One main purpose for distributing control is to make it nearly impossible for the entire system to fail. Building single points of failure into the communications infrastructure — such as non-redundant routers or switches or communication transceivers that can fail in such a way as to take down the entire link — defeats an important purpose of a distributed system.

Confirmed, network-wide (spanning all links within the system) multicast must be supported.

Rationale: in applications where the message must get through or a major equipment shutdown is required, for example for safety reasons, the sending node must be able to have confirmation that its message was received by all the members of the multicast group.

The protocol stack must support duplicate packet detection and resend the previously generated response without reprocessing or regenerating it.

Rationale: There are some transactions that are inherently not idempotent. Let's say that electricity customer is on a pre-pay contract with the utility, and the customer adds money to her/his account. The additional credit is transferred to the customer's meter, but the meter acknowledgement is lost. The utility re-sends the "add credit" message. Correct behavior would dictate that the meter only add the credit one time.

The protocol stack must support a mechanism that allows emergency messages to be routed in an expedited manner to overcome queuing delays within the nodes as well as queuing delays in routers between links.

Rationale: in control systems, sometimes nodes are synchronized to an external event that causes a flood of messages. Not all those messages are important in dealing with the external event (for instance: the oil refinery is about to catch fire), but some messages that could help avoid the impending problem must be propagated quickly across the network.

The protocol stack must support a sender node communicating with its peers in sequence without waiting for the response from one node to arrive before going on to the next one.

Rationale: Most control systems have supervisory nodes that ping the status of all nodes in the network, and drive an operator display of the system health. In this operation, if a node is down, the update of the entire display will halt until communication with the down node finally times out after some number of retries unless the protocol supports having multiple responses outstanding and a means to correlate those responses to original requests.

Physical Connectivity Requirements

The protocol stack must be independent of the underlying MAC/PHY.

Rationale: No single link meets all communication needs for the Industrial Internet of Things. Today multiple RF links, multiple power line links and a variety of wired solutions are needed to implement the various applications. Furthermore, transceiver development is an area of active research and investment, so the protocol stack must be able to take advantage of new technologies as they become available.

Security

Today, the vast majority of control networks are not secured. Protocol must be able to support strong encryption, mutual authentication, and protection against record/playback attacks. To sell networks to the U.S. government, the requirement is that the algorithms used be NSA Suite B approved.

Rationale: as the world moves from hardwired control systems and closed, unconnected networks to networks that can be connected to the Internet without gateways, strategies for perfecting on the Internet can be repurposed to attack problems with the Internet of Things.

It must be possible to secure the communications system to FIPS 140-2 level 1 at a minimum.

Rationale: NIST has created a widely accepted set of best practices for securing systems from cyber-attacks. These best practices have been vetted world-wide and form the basis for effective security policies management.

Control Services

The protocol stack must implement services that are needed in all nodes. Without a uniform set of communications services, nodes could not be successfully integrated into a cohesive network. Multiple, conflicting subsets of capabilities would make it cost-prohibitive to integrate nodes from multiple sources. Therefore, the required services must fit within the memory and performance capabilities of all devices. To meet this requirement, memory, and in particular, RAM consumption for the protocol stack must be limited to provide the application with adequate RAM as well. When they pick a platform, most application developers expect to be able to use RAM for their applications and not have it all devoted to the needs of the stack.

Rationale: In the world of low cost systems on a chip (SOCs), RAM is the most precious resource. In communications applications, RAM is needed to know when to retry a packet, to detect a duplicate packet, to put packets in correct order for delivery to the application, etc. But these memory requirements are all in direct competition with what the application needs for its memory — the more memory is used by the communications, the less the application can use. Given that these nodes are in a cost-sensitive environment, the use of SOCs can meet the cost constraints of the application.

The protocol stack must scale to thousands of nodes and multiple links of different speeds in a single logical network.

Rationale: Many building and factory systems today are composed of well over 1000 nodes spread out among multiple links with a high-speed backbone. As IP is pushed to every device, these networks will only get larger.

Network-wide (spanning all links within the system) multicast must be supported. Multicast group membership must be supported in the stack so that all applications do not see all multicasts and consume the node's resources discarding packets that are not addressed to them.

Rationale: multicast conserves bandwidth and improves response time over multiple, serial unicast messages. When closing a control loop over a network, all nodes that subscribe to a sensor value should get that value at the same time. Applications cannot be constrained to have all the nodes in their multicast group on their link, since some content, like emergency messages, must go to most or even all of the nodes on the network. Applications do not have the memory or cycles to process all multicasts just to discover that they do not apply to their node.

The protocol stack must support peer-to-peer communications.

Rationale: For response-time reasons, nodes cannot wait to be granted access to the network by a server, nor can they go through a lengthy session set-up sequence with a server. For finely distributed systems, the nodes must interact as groups performing a function across the network, the way light dimmers control lights.

It must be possible to provision timers in the protocol stack to indicate when to re-send a packet that has not been confirmed. These timers should be individually provisiond according to the destination address in the packet.

Rationale: This can improve response time and limit bandwidth consumption. How quickly a node should retry a message is a function of the round trip delay to/from the destination address. Retrying too early wastes bandwidth and may cause network congestion and packet collisions. Retrying too late makes response time suffer when a packet is lost. In a network composed of multiple links of differing speeds, a resource constrained node cannot be expected to know the round trip delays to all the subscribers of its data. Therefore, for large networks, a node with topology knowledge needs to provision these parameters.

It must be possible to discover the application-level information that a node can publish over the network. In this way, a network can self-organize and begin to perform an application function with minimal, if any, human involvement.

Rationale: Some networks must be ad-hoc in their formation and allow nodes to come and go without a management station.

The protocol must have a means to transfer a sequence of packets as a logical unit, like a firmware upgrade, a data log, or provisioning information.

Rationale: Data logging is a common control network function. Systems need to upgrade firmware, and provisioning such as linearization tables for sensors, calibration data, etc. is often needed.

Future versions of the protocol must work with prior versions and provide all the same capabilities as prior versions.

Rationale: Control networks are long-lived, as long as 20+ years in many cases, yet they are networks, so additional nodes and additional applications are added overtime. It is unreasonable, and often prohibitively costly to upgrade all the existing nodes in a network to the new version. The existing nodes may not have enough memory, or face other constraints.

There must be some lightweight application interoperability model to facilitate the exchange of data between publishers and subscribers.

Rationale: Resource constrained nodes cannot support parsers of data streams. Today's XML parsers are code space and RAM intensive.

While Industrial IoT systems operate autonomously, they also need to present data and status to system operators. To facilitate this with modern Web UI interface technology, the stack must support a RESTful API to provide system data.

Rationale: The most common interface to the Web today is RESTful APIs. This paradigm is well-supported by tools and developers and allows system access through ordinary browsers.

Introducing the IzoT™ Platform for the Industrial Internet of Things

Echelon introduced the IzoT platform specifically to address the demanding requirements of the Industrial Internet of Things. The IzoT stack addresses all of the listed requirements above. It is also available as free source code. A version of the IzoT stack runs on the Raspberry Pi platform, and allows developers to prototype on the Pi using either the Ethernet or WiFi interface for the communications. Once the prototype meets the requirements, the developer can then productize using one of Echelon's several IzoT SoCs and modules.

Overview of the IzoT Stack

Echelon has introduced the IzoT stack to meet the communication needs of the peer-to-peer Industrial Internet of Things. This stack meets the requirements listed above and is available in multiple forms from source code to SoCs for a variety of applications. To better understand how IzoT works, we present a brief overview of the stack, and then, requirement by requirement, outline how the IzoT stack meets the requirements.

The IzoT stack is a set of higher level protocol services that can run on top of any IPv4 or IPv6 UDP socket interface. When running on top of IPv4, the features of IPv6 such as stateless auto-configuration, and neighbor discovery are not available. If possible, use IPv6 for its ease of installation.

Resilience in the Face of Failures

Rapid Detection of Packet Failures. The IzoT stack supports end-to-end acknowledgements and responses so that if a packet is lost in traversing the network across multiple links, the loss is discovered and the packet is retransmitted. This feature is available with both unicast and multicast services. When using acknowledged multicast or request/response multicast services, retry messages are encoded with only the nodes that must respond. This limits congestion by informing the nodes whose responses or acknowledgements have already been received by the sender.

No Single Point of Failure. IzoT supports multiple PHYs that all have the properties of, if they fail, only the node connected to the PHY will fail. For example, twisted pair PHYs are transformer coupled, so that if the drivers fail in a state where they are stuck high or low, it will have no effect on the other nodes on the link. The twisted pair PHYs are true multi-drop so there are no single components that could fail and take down the link. In the case where IzoT uses Ethernet, this level of reliability may be achieved with redundant switches. When using IEEE 802.15.4 or IEEE 802.11 b/g/n, the PHY protects itself from denial of service attacks via a process of white listing MAC IDs. When using Power Line communications, the interfaces may be transformer-coupled and a watchdog timer can protect a node from having the transmitter constantly energized.

In the IzoT network, the devices are all autonomous with no central controller dictating their actions. Fully distributed systems such as the IzoT network, can survive multiple, individual node outages and still operate, or gracefully shut down with operator notification. A system consisting of clients and a server does not have this property, because when the server goes down, the entire system fails.

Duplicate Detection. When an IzoT packet is sent, it is assigned a transaction ID number. Nodes that receive the packet create a record of the communications transaction based upon the source, destination, priority attribute and transaction ID. At the time of initial creation of the record, a timer for the maximum time of the transaction is started. This timer is based on information encoded in the packet or provisioned in the receiver node. If a subsequent packet is received that matches this record, that packet is detected as a duplicate, and the previously sent response gets re-sent. If a new packet arrives with a transaction ID greater than the currently active transaction, but matching in all the address and priority attributes, the receiver assumes that a new transaction has begun, and re-initializes the receive timer. A packet that arrives with a previously used transaction ID — smaller value than the currently active transaction, but matching in the address and priority attributes — is considered stale, and is discarded.

Priority Messaging. A sender of an IzoT packet can mark the packet as high priority. This can give it priority at the MAC layer if that feature is supported, and will also give it priority through any intervening routers to its destination. This allows emergency messages to be propagated through the network ahead of non-emergency messages in router queues.

Multiple, Simultaneous, Communications Transactions. Nodes with more memory can use a version of the IzoT stack with the ability to support many outgoing transactions and correlate the responses. This is done by having a larger outgoing transaction record memory pool, and with some additional logic to correlate responses to the correct initial transaction.

Physical Connectivity Requirements

Multiple Link Support per Network. The IzoT stack has been proven on many different links: low power wireless (IEEE 802.15.4)/ 6LoWPAN, WiFi,Ethernet, multi-drop, free topology twisted pair (ISO/IEC 14908-2), consumer band power line (ISO/IEC 14908-3), a new standard for power line communications (IEEE P1901.2), and can be ported to any MAC/PHY that supports a UDP socket. IzoT routers can seamlessly route packets between these links to create a network spanning multiple physical media. There are many examples in the Industrial IoT market where support of multiple physical media on a single network is needed. For example, in a building the communications backbone is often Ethernet, while lighting may be wireless for the convenience of placing switches, and the HVAC system uses multi-drop twisted pair for speed and reliability. Similarly in an process control system, multiple, battery powered, wireless sensors can be placed without regard for local power or hardwired network access to provide additional inputs to a high speed, wired control system.

Security

End to End Security. The IzoT stack supports NSA Suite B algorithms for public and private key cryptography. Specifically, the stack uses AES-GCM for symmetric key encryption and authentication with 128 bit keys. Elliptic Curve cryptography is used for public key encryption and authentication with key agreement using ECHD and authentication using ECDSA. The IzoT stack can secure the application portion of the packets, so that only the receiver, and not any intervening infrastructure, need have the credentials to decrypt and authenticate the packet. IzoT security is available for multicast communications transactions as well as unicast, making the use of security equivalent to the IzoT application weather a packet must be sent using unicast or multicast services.
FIPs 140-2 Level 1 Certification. Echelon's new generation of the IzoT SoCs will be certified by NIST to FIPs 140-2 level 1 allowing developers to more easily incorporate state of the art security into their networks.

Control Services

Complete Implementation. The IzoT stack is fully implemented in all nodes with the one exception of the multiple, simultaneous outgoing transactions feature. This feature consumes more memory, and is not needed by nodes that are acting as peers on a network. It is available optionally and typically used for a supervisory node that must communicate rapidly with most or all other nodes on the network. With complete implementation, compatibility between the communications services in each node is ensured leading to simpler integration of the network. Scalability. Since the IzoT stack runs on top of the Internet Protocol, it inherits the scalability of IP. However, a key decision to base the stack on UDP rather than TCP reduces memory and communication bandwidth requirements allowing more constrained nodes to work on slower, more error prone links. This extra scalability allows system designers to optimize the solution so that IP can be economically extended to the simplest of control nodes.

Complete Multicast Support. Both confirmed and non-confirmed multicast is supported on the IzoT stack. This allows IzoT applications to work the same whether they are communicating to a single device or to many devices with a single message.

Peer-to-Peer Operation. In the classic Internet world, there are clients and servers. Servers tend to be larger, more capable nodes and can maintain state for thin clients. In a peer-to-peer network all devices are both clients and servers at once and the relationship between clients and servers is many-to-many rather than one-to-one for a given communications operation. IzoT supports peer-to-peer communications as well as client server communications in a way that is transparent to the IzoT application. So, IzoT applications do not need modification to run in either model.

Tuning Response Time to the Topology and Application. The timers for retrying packets, the retry counts and the time of the upper bound for a communications transaction consisting of the initial message and the receipt of all replies to that message are all configurable. This allows the timers to be tuned not only for the application needs but also for the network topology and bandwidth of the links that a message must traverse.

Node and Application Discovery. In the Industrial Internet of Things market, nodes will come from many suppliers. Integrating these nodes onto a network will require that the capabilities as well as the addresses of the nodes be discoverable over the network. The IzoT stack natively supports probing the network to get not only node addresses but also application interfaces, e.g. what data is published; what data a node may subscribe to, and what a node's type is. For example: is a node a switch? A temperature sensor? A supervisor controller? In this way, networks that embed the IzoT stack in each node may be created from multiple suppliers.

Support for File Transfer & Firmware Upgrade. The IzoT stack may be upgraded over the network, so as the IoT market evolves; nodes can be upgraded with new stack features. Since IzoT is built on top of the Internet Protocol, support for large data transfers is supported.

Backward Compatibility. The IzoT stack encodes a version number in every packet. This allows the protocol to evolve in a backward-compatible manner. Additionally, IzoT supports multiple adaptation layers that reside just above the MAC interface. This allows IzoT packets to be compressed and reformatted to fully interoperate with existing ISO/IEC 14908-1 networks. IzoT routers can then seamlessly route IzoT packets and ISO/IEC 14908-1 packets on the network. In cases where the IzoT MAC and PHY selection is the same as the MAC/PHY selection for the ISO/IEC 14908-1 nodes, both nodes can share the link and exchange application information transparently, even though the IzoT nodes are IP compliant while the ISO/IEC 14908-1 nodes are not.

Application Level Interoperability. The IzoT stack uses and builds upon the Application Interoperability model developed by Echelon and the LONMARK® trade association. Based upon groupings of standard data into functions, called Functional Profiles, nodes from many sources have been proven to integrate and work together as peers on a network. RESTful API Support. IzoT servers support a RESTful API for easy integration with web based user interfaces running on PCs, tablets and smart phones. This API allows an IzoT network and its data to be visualized, provisioned and managed using standard browsers.

Summary

The IzoT stack capabilities are unique in the market:

- IzoT unifies multiple communication links under a common application model
- IzoT provides a proven interoperability model for applications
- IzoT supports legacy ISO/IEC 14908 networks transparently
- IzoT can work with commonly available UDP interfaces for low cost, low footprint devices

IzoT is the most flexible and complete choice for implementing a network of devices in the realm of the Industrial Internet of Things.

ECHELON®