



A R R A Y E N T

White Paper:

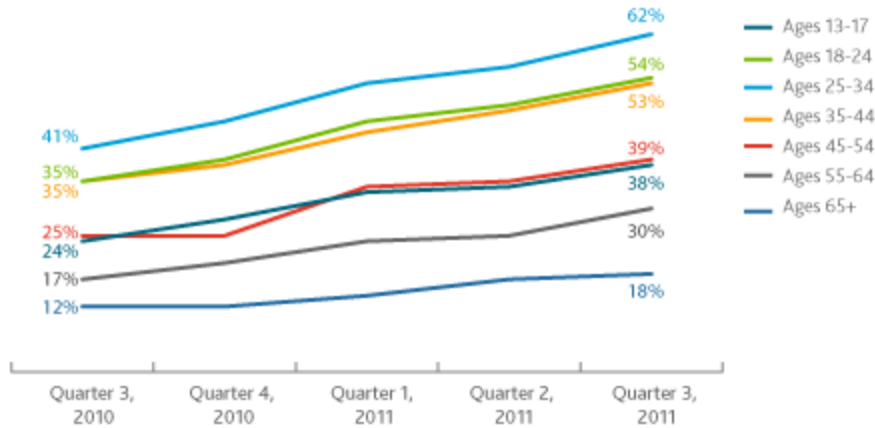
**System Requirements for Connecting
Consumer Products to Smartphone &
Web Applications**

Purpose

Smartphone adoption is accelerating. Nielsen research estimates that over 50% of Americans aged 13 to 44 now own a smartphone, see Figure 1.

Smartphone Penetration By Age Group

Q3 2010 - Q3 2011, U.S.



Source: Nielsen

nielsen

Figure 1. Nielsen research estimates that over 50% of Americans aged 13 to 44 now own a smartphone.

Consumers are becoming increasingly accustomed to being connected to any person they know, anytime, from anywhere. It is only natural for consumers to want to be able to monitor and control every “thing” in their lives, anytime from anywhere. It is this connectedness expectation that is stimulating consumer interest for internet-connected products. A representative Internet connected example is the text messaging alert flood sensor application shown in Figure 2 below:



Figure 2. Representative Internet connected product example.

To implement this Internet connect system, there are three implementation choices to make the magic happen:

- 1) Develop a solution from scratch using in-house programmers,
- 2) Build a system using outside contractors, or
- 3) Buy a proven software solution from a company who specializes in connecting low cost products to smartphone applications.

Whichever approach is chosen, there are six critical requirements that an internet of things connection system has to satisfy. This white paper describes the Internet connect system requirements that will enable consumers to monitor and control and their “things” (products) from web browsers or smartphones. It concludes with a summary of how the Arrayent Internet-Connect System™ satisfies these requirements.

Internet Connect System Requirements

Below are six system requirements required to connect consumer and SMB products to smartphone and web applications:

1. **Enable low BOM cost.** Price elasticity is a huge factor that must be taken into account in consumer products. There will be more product sales if Internet connectivity cost adder is \$25 than if it is a \$100 or \$200 Price adder. To achieve high volumes the hardware BOM cost adder to connect a product to smartphone applications has to be low.
2. **Just works installation.** If the product is difficult to install, most consumers will just return the product to the retailer. For an Internet connected product to stay in the home, it must be very simple to install.
3. **Secure.** Brand owners are concerned with hackers attacking their websites to extract their customer's financial information or privacy data, and consumers want to be sure that they are protected from hackers.
4. **Reliable.** High communication reliability is essential to ensure that your brand's reputation is not tarnished. Redundant servers mirrored at physically separated data centers are required to ensure high system uptime.
5. **Scalable.** Consumer product volumes can run in the hundreds of thousands to tens of millions of endpoints. The system has to be designed from the very beginning to scale, because there is no time to redesign the system if the product becomes a hit. The Internet history is littered with dead companies that failed due to lack of scalability (e.g. Friendster.)
6. **Efficient application development.** Smartphone and web application developers program at the web services, i.e. XML and HTTP, level of abstraction. But there is a problem because most consumer products are powered by low cost 8-bit MCUs that don't have the horsepower to support web service overhead. This system needs to be cleverly designed in order for web application developers to be productive and consumers to see low prices. In addition, there are common must have services such as real time alert notification, firmware updates, account management, customer support portal that are common across all product applications that have to be developed.

What follows is a more detailed description of these six system requirements.

Low Cost BOM

To grow beyond the hobbyist market, the Internet connect price adder has to be as low as can be possible, \$25 to \$30. Most internet connected products today subscribe to the "Honey, I shrunk the computer" connection paradigm. This approach leverages conventional Internet connectivity paradigm that was developed in the mini-computer era of the 1980's, and requires extensive memory and CPU resources to support a full 32-bit OS and TCP/IP stack. Most home area network (HAN) data protocols such as ZigBee, R4CE, and Z-Wave, have small packet sizes. So using TCP/IP that assumes very large packet sizes supported by IP, is overkill for telemetry applications. In addition, most HANs were not designed for global addressability. This limitation makes it impossible to carry HAN protocols over the Internet. This packet size versus

addressing mismatch requires a conversion step in the home gateway. The gateway cost drives up Internet connection implementation cost.

The protocols for devices to communicate with each other over a network have a major effect on endpoint BOM cost too. Object-oriented message protocols (e.g. CORBA, RMI, and SOAP) are a good match for modern object-oriented architectures, and reduce design and implementation time for product teams. However, these protocols require large support code libraries which increase memory in endpoint devices, and the BOM cost. Communication technologies based on XML (e.g. SOAP, XML-RPC, REST, JMS) require unwieldy code libraries adding memory requirements and BOM cost.

The alternative to the mini-computer communication paradigm is a cloud computing design pattern, where web server functionality and complex communication tasks (NAT fire wall traversal, state memory, etc.) are managed cost effectively by a server in the cloud. This means that the product can make use of an 8-bit MCU, with small amount of memory that most importantly means manageable verification problem. In essence, a cloud computing based communication infrastructure can be used to extract hardware cost out of the product endpoints and residential gateway. Therefore, the overall system cost to the consumer is minimized. Lower cost mean higher volumes, which is good for everyone.

Just Works Product Installation

For Internet connected products to stay in the home after purchase, the product has to be extremely simple to install. It is well recognized that consumers struggle to install Wi-Fi enabled products in their homes. In 2009, [ABI Research](#) found that 30% of consumers reported difficulty installing Wi-Fi equipment, and returned Wi-Fi products to the retailer to the tune of 11%, about twice as much as retailers will accept. The problem lies in that most consumers don't remember their Wi-Fi SSID or password that the Geek Squad set up when the consumer purchased their two laptop PCs and consumer Wi-Fi router.

One of most vexing product installation problems to solve in the home or SMB setting is making the connection between a smartphone and an endpoint product located behind the home owner's firewall (built into their Wi-Fi router). Most protocols used by endpoints to communicate with each other over a network (e.g. RPC, SOAP, XML-RPC, and REST) utilize an endpoint's IP address for addressing data or locating endpoints. There are two compounding problems. First, the consumer's home router will assign a dynamic IP address to the endpoint. This means that remote device such as a smartphone does not know the current address for the device it was connected just the day before. Second, there is no industry standard policy for the way consumer router firewall NAT (network address translation) tables handle incoming traffic. This means that a device behind the firewall must know the address to which it wants to connect to authorize a return message. An automated solution to keeping track of the endpoint device's current IP address is required. Consumers either do not have the technical expertise to modify their firewall setting to open ports, and those that do, will not want to be exposed to security risks that opening a firewall port can cause.

Any Internet connected system will need a well architected way to address endpoint products through a variety of dynamic IP address schemes. The system has to be tested against the large number of consumer routers that are sold at retail to ensure that a reliable connection can be

obtained right out of the box. A cloud computing based communication scheme, outside the consumer's firewall, is ideally suited to solving this problem.

Secure

Not a day goes by that we read about yet another computer system that was hacked. For the most part hackers exploit well known mistakes by owners of the hacked system. While no computer networking system is foolproof, steps can be taken to avoid being an easy target. Classic computer to computer secure communication schemes rely on securing the communication channel using SSL or VPN technologies. These techniques are overkill for things like telemetry-like data communication. Instead of encrypting a communication channel, you can encrypt the payload within a communication frame. 128-bit AES encryption used for Wi-Fi WPA2, and SSL, is the industry standard in banking industry. Many 8-bit CPUs today used in low cost endpoints have 128-bit AES encryption in hardware. The bottom line is that industrial strength security is already built into standard product silicon, and comes at no extra charge.

Reliable

The consumer expectation is already set: the things in their lives simply work. Consumer products are more reliable than websites, and they do not ever have to be rebooted like PCs and smartphones. While consumers tolerate Twitter going down for hours, they won't tolerate not knowing that their garage is up or down when they are away on vacation. High consumer satisfaction requires high system reliability. If isn't, they will return the product, and your brand name will be tarnished. In 2008, Apple's 20,000 MobileMe customers were experiencing misplaced e-mails, misplaced address book contacts, difficulty in syncing their calendars and service outages. Steve Jobs wrote in an e-mail to Apple employees "[the launch of MobileMe was not our finest hour](#)... The MobileMe launch clearly demonstrates that we have more to learn about Internet services." If Apple with all of its vast resources had difficulty getting Internet services flawlessly, it makes sense to scrutinize at what it takes to deliver a reliable connection between products and the web applications to which they connect.

Reliability must be designed into an Internet connected system from the very beginning; it cannot be designed in later. There are two components to Internet connect reliability: the endpoint itself and the server. Most consumer product endpoints are powered by 8-bit MCUs. These MCUS operate with operating systems and communication stacks on the order of 20KB, compared to Linux-based systems running on 32-bit CPUs with communication stacks that can be 200KB or more. The more software code there is in an endpoint, the more things can go wrong. Essentially reliability comes down to a system verification challenge, and having fewer lines of code to verify makes for a much more manageable problem to solve.

Placing a communication server in between a consumer product in the home and the smartphone introduces a new point of failure. A single server failure is equivalent to all your products failing in the field simultaneously. Operating redundant servers within a data center guards against a system wide failure when a server crashes. However dual servers in a rack are not good enough; the system has to be designed to guard against data center failure. The Internet connect system must be designed to support fast failover between geographical redundant data centers.

Scalable

The bottom line is that consumer product volumes run in the hundreds of thousands to tens of millions of endpoints. Considerable consideration to scalability deserves up front attention.

[Friendster](#) was an early pioneer in social networking. It was financed by top tier venture capitalists from Kleiner Perkins. Unfortunately they launched their service using Web 1.0 architecture (middleware on top of a relational database, and TCP/IP dependency.) Their service [melted under its fast growing popularity](#). Unfortunately, Friendster could not re-architect their infrastructure fast enough. Their members fled to MySpace and later to Facebook. More recently Radio Thermostat Corporation of America the maker of the 3M Filtrete Wi-Fi Thermostat sold at The Home Depot was a victim of their own success. Once they had over 100,000 thermostats their traditional web-server-on-top-of-a-SQL-database wasn't scaling. Their solution was to [transition their customers to EnergyHub](#), a company specializing in helping consumers understand and reduce their home power consumption.

Friendster stumbled unsuccessfully because the team made used SQL database backed application server architecture. SQL database architectures do not scale because a server can only handle 100 to 1000 SQL operations per second. When the operational limit is reached, system performance does not degrade gracefully. Friendster members experienced hung web pages and timeout messages. A common Band-Aid strategy is to place a caching layer between the application (i.e. website / device manager) and the database. Now the cache mechanism becomes mission critical, for which the caching is not designed. So when (not if) the caching layer fails, transactions fall back to direct database access, which then results in users getting locked out and operations failing. It is inevitable that a Band-Aid system will require a redesign.

Serving 20,000 accounts was the point where Apple's MobileMe broke down. So it does not take very many endpoints to crash inadequately designed server architectures. It is critically important that you examine your server architecture against each database transaction, such as:

- Checking secured login credentials. Credential checking is handled automatically in web high-level languages (e.g. php), and less sophisticated programmers are unaware that database queries are occurring.
- Sending a command to a device requires looking up that device's address.
- Checking the state of an endpoint device.
- Receiving a status update from an endpoint device and storing it for future display.

Large-scale websites (EBay, Facebook, Google, Twitter, etc.) have learned the hard way how to build storage systems that are not bottlenecked by a database when dealing with hundreds of thousands of users (or endpoints). They employ a modern day switch-like architecture that uses a persistent caching scheme.

A second factor affecting scalability is the communication protocol selected. There are number protocols that have been used for machine to machine (M2M) communication, including CORBA, RMI, SOAP, XML-RPC, and REST. These technologies tie up TCP/IP resources on the endpoints. TCP/IP becomes problematic because an Internet server designed for TCP/IP processing efficiency can handle 10,000 to 50,000 TCP/IP connections. For a server system to scale beyond

that limit the servers must be interconnected in a cluster to share data. Clustering raises costs beyond the obvious cost of the servers alone. The interconnection network quickly becomes complicated and mission critical, necessitating additional cost to ensure reliability and throughput. If you start with a system that is based on TCP, and you become commercially successful, you will need to switch to more efficient communication method, or suffer customer defections.

Message bus architectures e.g. Java Message Service (JMS) and competitors, are used for high-performance applications such as futures and derivatives trading. They do not rely on TCP/IP. A message bus allows peer-to-peer communication, has strong security mechanisms, and robust delivery models including both asynchronous and guaranteed modes. However existing commercial message bus implementations depend on large code stacks, and are designed for enterprise systems funded by financial institutions, where money to buy expensive hardware is easily available.

Efficient Web Application Development

Web services interfaces, i.e. XML and HTTP, have become the de facto web application coding interface. But there are two problems. First, consumer products are usually powered by low cost 8-bit MCUs, and web service overhead is too heavy weight for 8-bit MCUs to support. Second, web application developers have little knowledge of embedded system programming (e.g. assembly language programming). To make software development easy, one approach is to build a web application server on the endpoint. This requires 32-bit processing, large OS, expensive memory, and extensive verification cycles.

A more modern approach is to abstract away the details of wireless connectivity and embedded communications software on your endpoint product, and create a virtual instance in the cloud that has a web services programming interface. By employing such a virtualization technology (patent pending by Arrayent), web application developers can efficiently program and support applications at the web services level of abstraction. The server then communicates to the low cost 8-bit MCU powered endpoint in a lightweight way. The result is low cost end products, and low cost application development and manageable verification efforts.

There are system services that are common among every smartphone connected product ecosystem. These services include real time alert notification, firmware updates, customer account management, and customer support portal. Having these utilities in place reduce the cost of ongoing application development across your product line.

The Arrayent Internet-Connect System

The Arrayent founders saw that the mini-computer communication design pattern was too costly for low cost products sold at retail. Consequently they asked themselves: if the Internet could be redesigned from scratch to serve things (telemetry data), what would it look like? The Arrayent team made three fundamental observations:

1. Low cost 8-bit CPUs are more than sufficient for most consumer product applications like toys, thermostats, sprinkler controllers and home awareness systems.
2. The cloud computing paradigm affords the most cost effective way to spread communication cost over the network.
3. Web services (XML & HTTP) has become the de facto communication standard on the web. Web application developers know how to program to a web services interface, but they do not know embedded system programming that is required to program 8-bit CPUs.

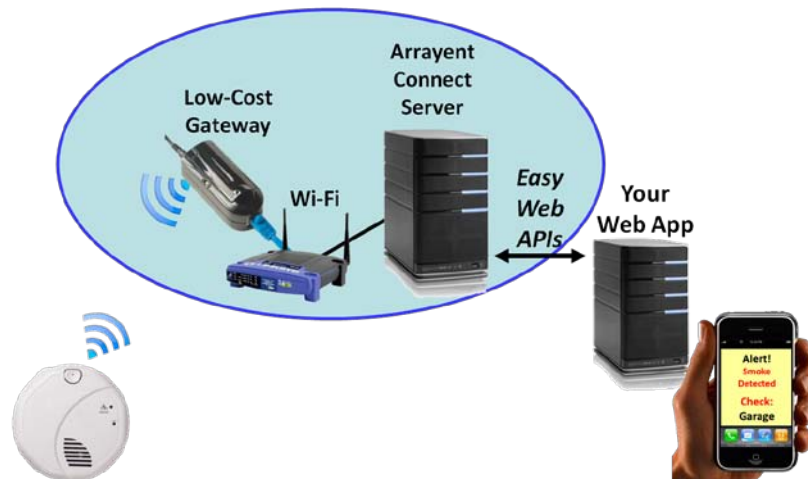


Figure 3. Elements in the Arrayent Internet-Connect System.

The Arrayent team made four design decisions around its cloud computing architecture:

1. Arrayent's software is to be portable across silicon implementations (CPU and RF radio.) Hardware reference designs will be available and the customer can select which CPU and RF radio fits the price/performance objectives.
2. Choose an IP communication protocol that is designed for things (telemetry.) TCP/IP is too heavy weight for telemetry applications and imposes unnecessary cost for light media applications. Arrayent built an end-to-end communication system using a message-like protocol. The traffic is managed by a message bus architecture optimized for low cost embedded device connections.
3. Place expensive communication processing tasks on servers located in the Internet cloud. Applications hosted in the Internet cloud have become mainstream. Witness the popularity of web based e-mail, massive multiplayer online games (e.g. World of Warcraft) among consumers, and SaaS based applications like Salesforce.com,

- SuccessFactors (acquired by SAP for over \$3B) and Concur with the enterprise customers. Cloud computing resources have been proven to be a scalable, cost efficient way enterprise customers can expand their computing capacity.
4. Arrayent developed patent pending “web services virtualization” technology to enable web application developers to communicate with an endpoint using 8-bit CPU, as if it were a fully supported web server.

Low Cost BOM

The Arrayent system has been designed to enable low cost 8-bit MCUs in the endpoint devices. Core to the Arrayent product philosophy is that “things should be things”, and expensive communication processing tasks are moved to communication servers hosted in the Internet cloud. For example, the Arrayent end-point and gateway software are relieved of having to translate between HAN and WAN protocols; instead, translation is performed by the Arrayent server in the cloud. A non-XML data format allows very small code footprint for generating and consuming data. This reduces device BOM cost.

Arrayent delivers a low cost product BOM. You can add a RF radio to your product for less than \$2 and Ethernet gateway for less than \$5 parts cost. The total costs less than half the cost of PC-like internet connectivity solutions.

Just works installation

Arrayent delivers just works installation procedure that ensures an easy to install and user friendly, which reduce customer support calls and product returns. For example, Mattel’s customer support organization reports that the Mattel IM-Me text messaging toy for 8-year girls required “customer support levels equivalent to any battery operated toy.” See Arrayent example install video [here](#). See Chamberlain’s LiftMaster gateway installation process [here](#).

The Arrayent Connect Server is designed to traverse the consumer’s router’s network address translation (NAT) firewall rules, and manage global device addressing, which ensures plug-and-play installation. The consumers do not need to make any changes to their firewall/router. Arrayent supports star topology networks (900MHz proprietary network, and Wi-Fi.) Star topology is the dominant LAN topology in use today (e.g. Ethernet, 3G, and Wi-Fi) because it is a straight forward topology to install and add additional endpoints.

Secure

Arrayent reference designs are based on 8-bit CPUs that support 128-bit AES encryption in hardware. Industrial strength security is already built into standard product silicon, so it comes at no extra charge. Communication between your application servers and Arrayent is done using SSL.

70% of Americans enter their homes through their garage. Chamberlain, the world’s largest garage door opener company with an installed base over 25 million homes, run its MyQ Internet connected garage door openers over the Arrayent system. They hired IBM Internet Security Services to independently assess the security of the Arrayent system. Arrayent invites all its customers to do the same, as it results in a more secure system for all.

Reliable

Network reliability is a function of the endpoint, gateway and the server. The RF module and gateway reliability is mostly a function of communication stack complexity. Arrayent's communication stack runs on industry standard 8-bit MCUs, and resides in 17KB, one tenth the code size of TCP/IP communication stacks. Less lines of code means fewer things can go wrong, and higher reliability.

Servers, the third link in the reliability chain, need to be designed for high reliability too. Arrayent's persistent cache architecture (see scalability description below) is redundant so that when a hardware failure takes down one server, the data is still located on another server. A distributed persistent cache is much more reliable than a SQL database. If a single SQL database machine is lost, all operations cease. If a backup SQL database is used, all data since the last backup/snapshot is lost. If continuous (i.e. synchronous) database replication is used, the database transaction rate is severely impacted, reducing performance and scalability even more. Arrayent uses an SQL database for maintaining long term data. A background process moves data from the persistent cache to the SQL database. This procedure provides the simplicity of an SQL database in case an entire system needs to be restarted from scratch; however, since the database is not in the normal operational path, the entire system is not slowed down. Arrayent operates redundant servers at each data center co-location. Additionally, each co-location is backed up by a geographical separated co-location installation that mirrors the primary location.

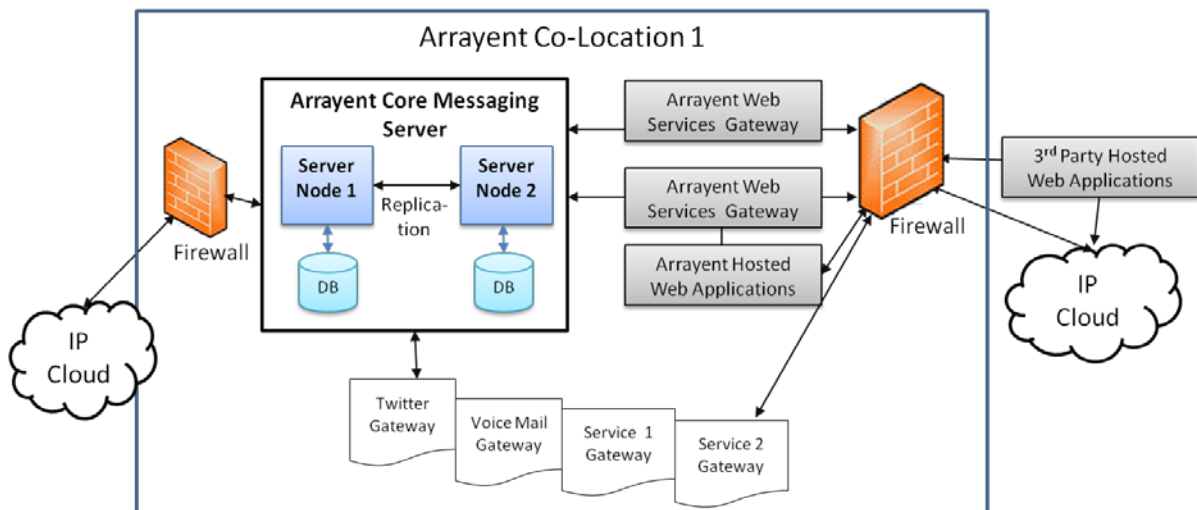


Figure 4. Arrayent's high reliable and scalable server architecture.

Arrayent delivers reliability. IM-Me servers have experienced 99.9% uptime since 2007.

Scalable

Arrayent uses modern switch-like server architecture, the same architecture used by E-Bay, Facebook and Google. Specifically Arrayent has implemented a distributed and redundant persistent caching mechanism to eliminate the SQL transaction bottleneck problem described in the previous section. To ensure high scalability, Arrayent has implemented the following features:

1. User and device credentials are stored in the persistent cache. All devices go through a login process to ensure that attackers cannot impersonate a device to steal user data. Only after credentialing both the device and the user are they allowed to communicate. All operations are continuously authenticated as the overhead is very low when using the persistent cache. Since the persistent cache is distributed, adding more servers automatically increases the performance of the cache. Therefore the number of users and devices that can be served can easily be scaled.
2. End user information is stored in both the persistent cache as well as an SQL database. This provides the speed and scalability of the persistent cache, as well as the manageability of an SQL database for long term storage.
3. The persistent cache operates out of server memory, not from disk. This results in two orders of magnitude faster access than a database. Transactions run at about 10,000 to 100,000 per second depending on complexity.

Arrayent delivers scalability because it uses a modern switch like architecture, the same proven architecture that has been shown to be scalable with E-BAY, Facebook and Google.

The Arrayent Connect communication protocol is implemented as a message bus architecture designed to communicate to low cost embedded devices, with the additional requirement of reducing server resources. Features and advantages:

1. A message bus protocol does not require TCP/IP resources on the server. This allows Arrayent servers to support 100,000 to 1,000,000 endpoints per server. Reduced server cost means small or no subscription fee to the end-user.
2. High performance message switching ensures sub-second latency for Internet round trip time. Latency is actually dominated by the endpoint device polling time, rather than the Internet.

Efficient Web Application Development

It is highly unusual for web application programmer to know everything about wireless communications and embedded system programming, and the converse is also true. Letting web application developers program at the web services abstraction level ensures that application development will be fast, robust, and easy to maintain. It is for coding efficiency reasons why most Internet of things endpoints are implemented running a web application server software (that require 32-bit CPUs and lots of memory.)

Arrayent's patent pending "web service virtualization" technology virtualizes a web server for each endpoint in the cloud. Web developers do not have to become embedded system programmers; they can program at their efficient web services level of abstraction. Most importantly, they can use their favorite web development tools.

Arrayent's web service virtualization technology delivers the best of both worlds: efficient web application development and low device cost, because the endpoints use 8-bit MCUs with low memory requirements.

The Arrayent Internet-connect service has utilities including as real time alert notification (SMS and e-mail), firmware updates, and customer account management. These services save application developers the time to develop and verify these common utilities.

Arrayent Is a Proven Solution

Connecting millions of devices at low cost and with easy installation process is Arrayent's single business mission, not a side show. Re-creating a Version 1.0 Arrayent Internet Connect technology from scratch would take a multiple person development team years two years to duplicate. On top of that, multiple man-years are required to work out all the bugs that can only be uncovered in real world use of the system. Nothing replaces large-scale real world deployment over a long time across different brand owner applications, and consumer routers/firewalls to uncover latent bugs that do not appear in artificial pre-production testing environments.

In October 2007, Mattel announced the IM-Me, a walled garden text messaging device for girls. Before the product was shipped, Arrayent simulated traffic of 100,000 IM-Mes pinging the servers to ensure system reliability. On Christmas morning, 27,000 IM-Mes logged into the Arrayent servers without failure. Since the IM-Me's product launch, 100,000 IM-Mes have been sold both domestically and in Europe. IM-Me servers have experienced 99.9% uptime thanks to Arrayent Internet-Connect redundant servers located in two co-locations. Arrayent's attention to plug-and-play installation was tested as well. Fisher Price does all Mattel customer support, and reported that the IM-Me customer support load was "the same as for any battery operated Mattel toy."

In 2010 Monster chose Arrayent's Internet-Connect™ turnkey communication system to deliver low cost and easy to install remote management to home theater installers deploying Home Theater HTUPS 3700 PowerCenter™ in their customer installations. Installers are on the hook to service their customer's home theater installation after the sale. By adding Monster Cable's [MP RA 100](#) card to a client's HTUPS 3700, installers gain remote access to Monster Cable's Control Suite web application that the installer can monitor all their installations from any computer or smart phone. It features an easy-to-read dashboard, see screen shot [here](#), that enables installers to monitor and control home theater components all as if the Installer were on site. In addition, Installers can optimize their client's energy use, reboot frozen components, and set up email or SMS notifications to alert customers of energy changes while they're away. Remote management adds up to big savings because the average installer truck roll costs \$125, and delivers service faster to keep customers satisfied.

In June, 2011 Chamberlain announced its new line of MyQ intelligently networked family of garage door openers. 70 percent of all Americans use their garage door as the primary home entrance. LiftMaster is the number one brand of professionally installed garage door openers

and access solutions. Mark Karasek, Ph.D., executive vice president engineering & chief technology officer said, “After exhaustive investigation we determined that Arrayent’s technology was the perfect match for meeting our cost and reliability goals for connecting LiftMaster products to the Internet, and at the same time enable an [easy installation process](#).” These products are now available through 4,000 LiftMaster dealers and retail.

Further Reading

The following papers describe the architectural concepts behind Arrayent’s scalable application server.

1. [Architectural Styles and the Design of Network-based Software Architectures](#). Roy Thomas Fielding, PhD Dissertation, University of California Irvine, 2000.
2. [Cassandra: A video presentation](#). Talk given by Prashant Malik, Karthnik Ranganathan, Avinash Lakshman at SIGMOD, July 2008.
3. [Dynamo: Amazon’s Highly Available Key-value Store](#). Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels. SOSP’07, October 14–17, 2007.
4. [Eventually Consistent](#). Blog post by Werner Vogels (CTO Amazon), December 2008.
5. [On Interprocess Communication](#). Leslie Lamport. Distributed Computing 1, 77–101. 1986. Springer-Verlag.
6. [PHP Web Services without SOAP](#). Adam Trachtenberg. OReilly.com, October 2003.
7. [Towards Robust Distributed Systems](#). Keynote talk by Eric Brewer at PODC July 2000.
8. [TreeCache: a Tree Structured Replicated Transactional Cache](#). Bela Ban, Ben Wong. Jboss.org, December 2004.
9. [Second Generation Web Services](#) and [REST and the Real World](#). Paul Prescod. XML.com, February 2002.
10. [Underneath the Covers at Google: Current Systems and Future Directions](#) (Talk gave by Jeff Dean at *Google I/O* conference in May 2008.)
11. [Web Search for a Planet: the Google Cluster Architecture](#). Luiz André Barroso, Jeffrey Dean, Urs Hölzle. IEEE Micro. March-April 2003. pp. 24-28.