GALIXSYS NETWORKS

White Paper

Real Time Control Over the Web

An Enhancement to M2M and Smart Devices Networks

Overview

Machine-to-Machine (M2M) operation, or any “smart device” in general, is characterized by allowing any device that has a network connection to be accessible on the Internet. With the increases in wireless capability and coverage (both cellular, wireless LAN and even near-field communication (NFC) based) and the rapid reduction of both hardware and connection costs, the number of machines that can be connected is increasing significantly. These machines make up a range of devices, from smart thermostats that are accessible by the home owner from any device with a web browser, to smart gas meters that automatically report the gas usage to the utility company, to consumer devices such as smart picture frames that automatically stream pictures stored from a web site to the frame.

Current capabilities of M2M operation is effectively the same as any other computer operation; devices make use of the Web to send and receive data, usually from a cloud service or back-end server. Data is stored on these servers and then viewed by operators using general web browsers. Mobile applications may be created that allows the access and sending of data or commands from a smart phone or tablet to the smart device, with the cloud service or back-end server being the control point in these operations.

Although these capabilities allow the M2M devices to have essentially PC-like capability in sending and receiving data, this limited functionality does not represent the full potential of these devices. Real-time control of devices, either by a user or via another machine, is desired for a variety of applications. How real-time control may be added over existing web technologies is explored.

Why the Web?

System cracking and the Internet have become nearly indistinguishable from each other. The history and sophistication of Internet cracking tools have become so high that it is nearly impossible to create a new Internet Application (IA) server and have it stand up to the automated cracking tools available. For this reason, most modern systems will only allow a single IA server be visible to the wider Internet, with the thought process being that it is much better to harden a single IA server than to have to validate and harden multiple servers. Since web pages are the most common Internet application, for nearly a decade now the web server has been the main server used, with other IA functions now being ported over to as a “web service”. A common example is document downloads over HTTP, replacing the IA File-Transport Protocol (FTP) in the process.

Security is not the only reason why web services have become so popular. Most of the data that now resides on the Internet actual resides in back-end functions that have direct Web accessibility. If a desired device or system wants any piece of data, it is generally directly accessible through a web-client. Internet applications built directly on top of a custom TCP/IP protocol will need to go through various translations or services to be able to store its data or request data from existing web services. This not only has a performance impact, but also a cost impact as maintaining not just the custom Internet application needs to be done, but also any translations tools used to access web data also needs to be maintained.

For these reasons (security and cost/system maintenance), any new Internet-enabled device or system must have a means to use the web directly, and not be reliant upon a separate IA protocol. However, the HTTP protocol that drives the Web is not without issues, and these issues need to be addressed if a real-time control over the web is

to be achieved.

Control over the Web

With the web server now becoming the single point of entry to any system from the wider Internet, a means needs to be provided that allows functions other than those required for web browser to occur. As the web is inherently a streaming protocol with text-based documents as the primary means to pass data, XML document passing has fulfilled this role. With an XML document (and its appropriate creators on the client side and parsers on the server side), a general tag describing the type of data being sent may be used, allowing the web server itself to fulfill nearly any function.

However, as XML documents are text documents, the size of the payloads is generally very high. Further, significant processing time must be dedicated to creating the XML document on the client device side, along with similar processing times in XML parsing to occur on the server side. This is in addition to generally a file I/O action on both sides as well as the document will be sent in a streamed (i.e., non-packet based) format. This overhead, both in the processing time on both the client and server, along with the amount of data transferred for even simple commands, generally adds significant network latency to the system. This processor network latency will have a direct impact on how responsive a real-time control application can be implemented. Historically, these latencies can be overcome with a custom Internet application, but for the reason listed above, these custom Internet applications are no longer feasible. An ideal solution would re-use the existing web infrastructure and protocols but also allow for a low overhead, low latency means to transmit generic commands to devices.

Binary-over-HTTP

Most of the inherent latency and processing problems of the web can be removed if XML document passing is not used as the means to transmit generic commands. The web already supports binary transfers if certain objects are tagged as a specific data type. These data types are identified in the HTML header and if identified as such, any payload associated with that tag is assumed to be binary, not text, data. The use of this binary mechanism can give custom IA-like performance and network latency to even web based services. Further, sending data over in packets instead of streamed will remove the file I/O operations as well, allowing all network command operations on both the device and server to be done in-memory.

The web, however, has not been designed to handle data in packet formats, and as web servers are stateless in nature, any dropped connection during a packet transfer will need to be recovered. Further, a general protocol will also need to be developed specifically for control applications, and embedded into the HTTP header. The use of binary-over-HTTP, along with solutions to the stateless web-server and the creation of a general control protocol over the web is the foundation of the IP and software created by Galixsys Networks.

About the Author

Steve Jahnke is a Founder and CTO at Galixsys Networks (http://www.galixsysnetworks.com). He has over 15 years of embedded engineering experience, having worked in silicon and software architecture, strategic marketing and business development roles. He is the lead or sole inventor on over 20 issued patents, and holds a Master in Electrical Engineering degree from Rice University (Houston, TX) and a BSEE from Northwestern University (Evanston, IL).