Contextual Compression for Constrained Networks

Whitepaper Sponsored by WIndSpring
June 14, 2016

Contextual Data Compression Technology Saves Time and Money

WindSpring offers up to 20:1 Compression

The deployment of widely distributed Internet of Things (IoT) devices is a challenge to maintain – on-going costs can include data connectivity. While some IoT devices can use short range wireless networks like Bluetooth and WiFi, many applications cannot rely on local networks to be present, requiring a dedicated cellular or other long range communications network. These long range connections, such as Sigfox, have restriction of packet traffic. As such, gathering data from a remote device, or providing updates, can be a challenge in these constrained networks.

The WindSpring Intelligent Compression API technology was developed for embedded applications, such as automotive, remote M2M, and other IoT solutions, and has been deployed for several years. The growth of IoT has meant sensors are being placed into remote locations where cellular and other bandwidth-constrained networks are the primary communications link. The main benefit of WindSpring's contextual compression is to get the "most" out of these constrained networks. Getting the most means reducing costs, improving performance and responsiveness, and doing it with limited device resources. It can also mean faster time to market because using WindSpring saves the time to develop a do-it-yourself (DIY) solution.

The WindSpring technology can become an essential solution when an IoT device needs high-ratio compression because the connection has limited bandwidth, limited power, or high cost-per-bit transmission costs. The total amount of data needs to be considered when scaling IoT deployments (many end node devices sending bursty data). Using a narrow data channel can reduce carrier fees and allow the use of cheaper radios. WindSpring technology not only compresses the data, it also compresses the protocol overhead as well, which standard data compression algorithms do not.

The need for WindSpring's technology may also be driven by changes in the cellular infrastructure that drive changes to the communications channel between the devices and the base. One example is that there may be designs that rely on the older 2G inexpensive infrastructure. As 2G sunsets, the communications channel may be switched to Sigfox, 4G LTE Cat 1, or future 5G narrow-band solutions. These new channels may be lower bandwidth, smaller data payloads, or higher cost per bit. For example, Sigfox has a 12-byte message limit per transaction. With the WindSpring tools, these variations between networks can also be hidden from the enterprise application.

Lossless compression

All mission-critical transfers must be lossless. That is, data integrity must be maintained throughout the transfer and must also be 100% accurate and reversible. Using general compression algorithms, it is possible to get 2:1 or 4:1 compressions levels which is not bad, but it's pretty standard fare and often traditional approaches have cases where compression doesn't work. Lossy compression can get higher ratios, but lose data integrity and are best used for media data where some loss of fidelity can be tolerated for the benefit of reduced bandwidth and storage requirements.

Lossless data compression relies on algorithms that look for statistical redundancies in the original data to replace it with smaller patterns. The process also has to be reversible – so the original data can be extracted from the compressed format. A basic example is run-length encoding that replaces long runs of identical data by a single data value and item count. More sophisticated methods include Lempel-Ziv (LZ) and the DEFLATE LZ variation. DEFLATE is used in PKZIP, Gzip, and PNG. Lempel-Ziv-Welch (LZW) compression is used in GIF. The various LZ methods use symbol tables where symbols are substitutes for repeated strings of data.

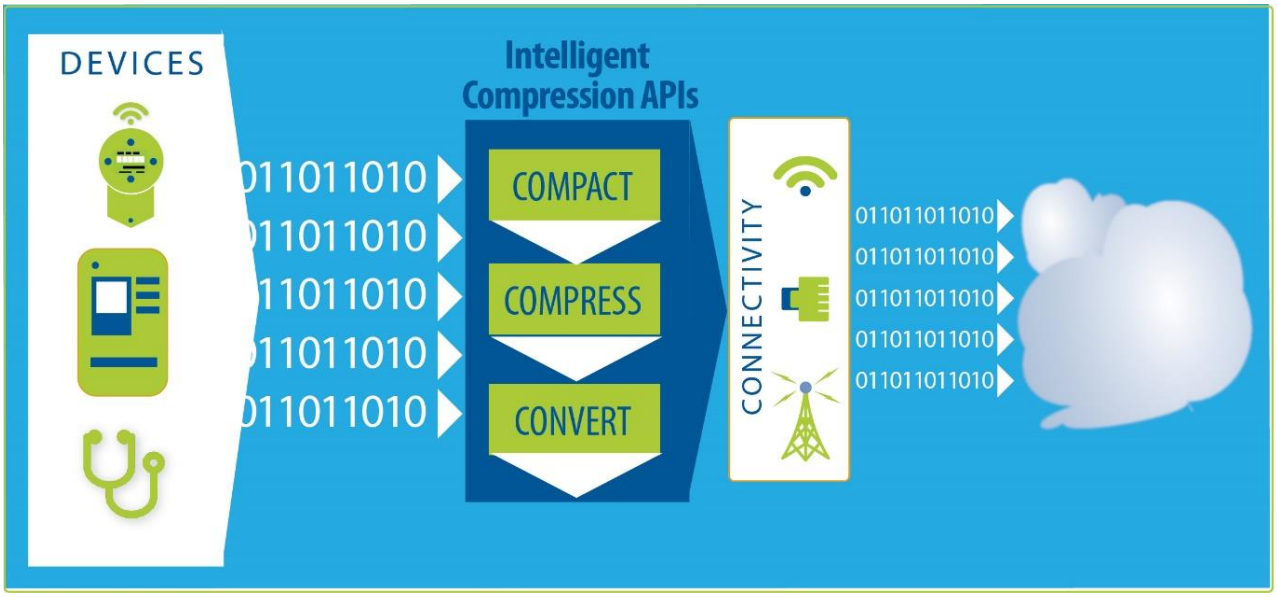
The WindSpring technology goes far beyond those standard algorithms to provide compression ratios up to 20:1. WindSpring's solution accomplishes this by applying a multi-phase compression process to the entire context of the data and protocol.

WindSpring SpringBoard API changed the process of compression by including a "Compaction" function, which reviews how to present the data more efficiently even prior to compression, as seen in Figure 1. The data is next passed to an adaptive compression engine that selects the best of several compression algorithms based upon the type of data. As one example of the process, if the content is a program update for the end device, the algorithm identifies which part of the code is new and the changes between the versions and compresses those changes into the minimal possible transfer.

Finally, the data goes through the conversion stage which performs 2 functions: first, it reviews the protocol being used, and it can replace it with a more efficient one that assists in overall data reduction removing unnecessary overhead; second, it reviews network traffic restrictions and automatically sends data at the optimum size for the network.

At best, the compression algorithm achieves approximately 20:1 compression, but even the 10:1 compression ratios typical for common examples is a great improvement over standard algorithms. If you use traditional compression algorithms you address only part of the problem.

Figure 1. Using Contextual Knowledge, Compressed Data Can Be Sent Over Limited Channels



IoT – what does data cost?

In the development of IoT networks, it is critical that device data and code updates be sent in the most cost effective manner possible. As IoT devices proliferate, with many millions, eventually billions of devices in the field, it is important to consider the aggregate amount of data that needs to be sent from the device to the cloud, and the amount of data and software updates back out to the devices. In addition, IoT devices can be constrained by remote location, limited power, constrained radio connections and other limitations. The most efficient way to handle these interactions is through data compression. WindSpring offers a solution that allows the greatest amount of data to be sent both to and from a device using the least amount of bandwidth while using a minimal amount of memory.

But each IoT deployment is unique and one size design does not fit all devices. Each deployment should be developed with the specific device in mind. IoT devices are also often constrained by limited processing capability and memory size. Using a standard GZip algorithm could take 100K-200K bytes of code to implement. The WindSpring implementation can fit in under 1K bytes, making it more appropriate for IoT applications.

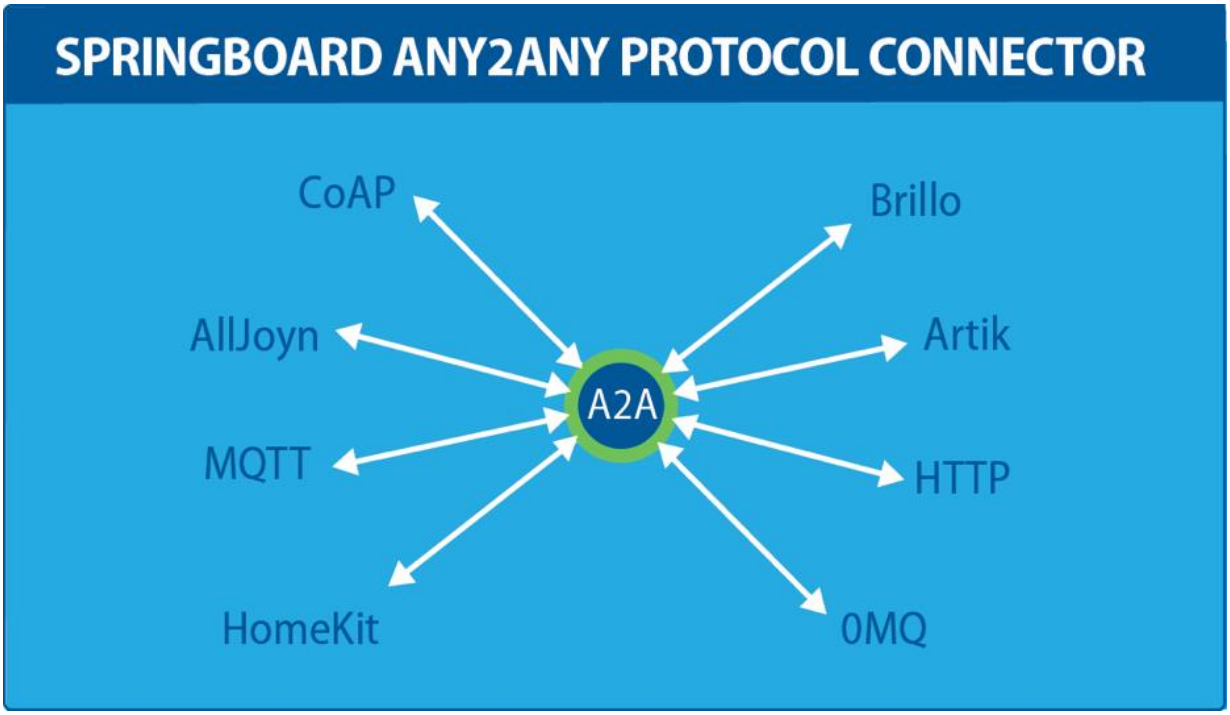
IoT data is also often transmitted in very short packets. Usual run-length encoding compression schemes don't work well on short data runs as it's harder to find long symbol strings. Often the protocol overhead is greater than the data packet being sent.

A IoT characteristic that works well with WindSpring's solution is when there's control of both ends of connection.

Radio power can also be an issue – longer transmission times result in the radio using more power. In some cases, the radio may be constrained by cost, power, and space that can limit radio efficiency. In either case, the channel bandwidth may be severely limited and the design must make the best use of the limited resource. With these constraints, data compression can make a big difference by shortening transmissions.

Using the server or gateway side A2A SpringBoard software (Figure 2) adds the additional benefit of being able to isolate applications from multiple data and communications protocols. The WindSpring SpringBoard A2A (Any-to-Any) Protocol Connector enables the device to transmit data in its native protocol, but the A2A then translates the data to a compressed intermediate wire protocol. On the receiving end, the data is reassembled into the desired network protocol. Using the SpringBoard A2A Protocol Connector, future protocols are easily supported, future proofing the connectivity platform. SpringBoard uses under 1k of memory on the device, without any performance degradation, making it seamless to the end user.

Figure 2. SpringBoard Tools Include Any2Any Protocol Connector

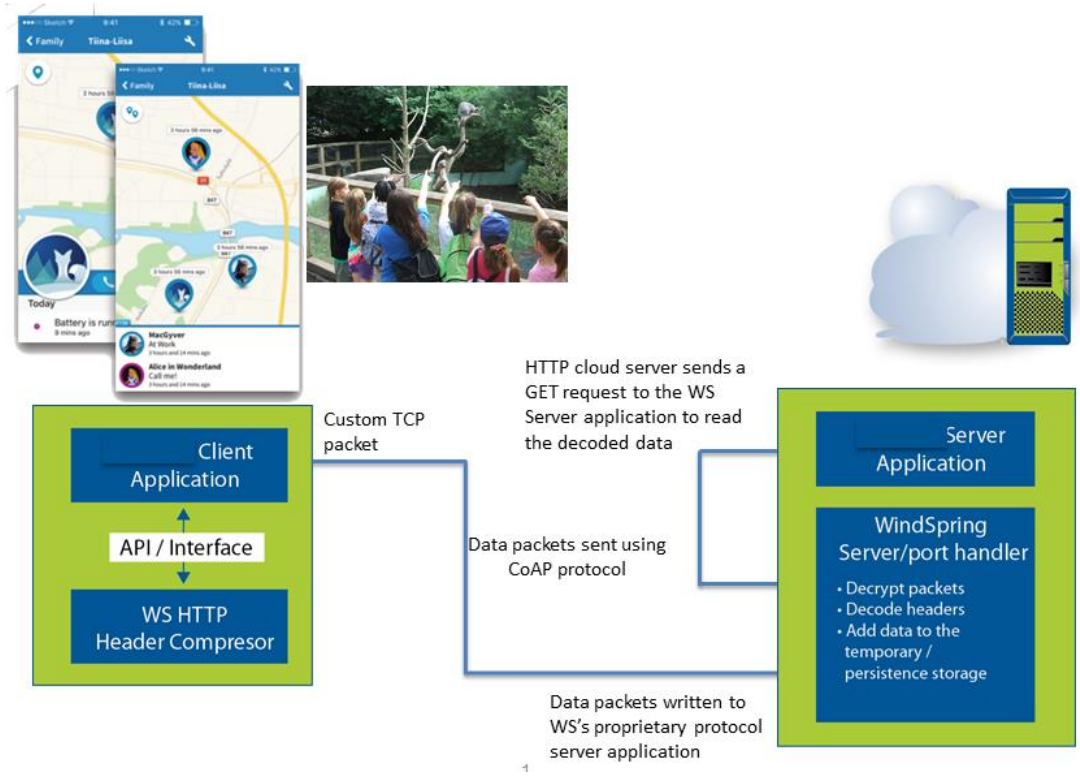


Another advantage of the SpringBoard A2A Protocol Connector is that it strips out unnecessary protocol overhead, sending only the bits that are needed. In addition, it can be used to break down the protocol Towel-of-Babel to open communications between protocols, while reducing payload size to save on constrained end node resources.

Example Deployments

A representative example of how WindSpring technology can be used in a location tracker product for children, as seen in Figure 3. Each location massage from the tracking device to the cloud can be as much as 403 bytes when the data payload (166 bytes) and protocol overhead (HTTP header needed is 237 bytes) are considered. One field trip can result in 960 messages sent over the network.

Figure 3. SpringBoard Child Location Tracker Example



The WindSpring's custom HTTP client compresses the headers and compresses and encrypts the data. At the server end, the WindSpring port handler decrypts the packets, decodes the headers, and new data is reintegrated with temporary/persistent storage. After reintegration, the full data packet is available to the server application to read. Using a compressed CoAP (RFC 7252

Constrained Application Protocol) with the WindSpring protocol, the application was able to reduce the data payload from 166 bytes to only 15 bytes. The HTTP header can be compressed form 237 bytes to 43 bytes. The overall reduction is from 403 bytes to 58 bytes, for a 11:1 compression ratio. The shorter packets save radio power at the client device and data costs over the long term.

Another example of the use of the technology is utilities that use 3G and 4G wireless technology to transmit data from smart meters, eliminating the need to build out an expensive infrastructure. However, moving larger numbers of small amounts of data through wireless networks can be incredibly costly, especially given that a typical smart meter generates as much as 400MB of data per day because the meter was not optimized for cellular communications. In most cases, the protocol to transmit the data exceeds the size of the data and is not bandwidth efficient.

Smart meters use unusual protocols like wireless MBus, or DLSM/Cosem (which support backend systems with billing) and use UDP versus TCP. Meters are also moving into new low-powered networks that have "traffic" and or "frame" limitations, which are simple to support in WindSpring APIs. Additionally, WindSpring can replace "heavy" protocols, with very lightweight alternatives, or standard efficient protocols like COAP, and then at the server side simply turn that to wireless mbus or DLSM/Cosem or any other protocol. By compressing the data and the simplifying the protocol, the utility can minimize the cost of the data transfers.

A smart meter customer worked with WindSpring to deploy the SpringBoard M2M Solution and cut data usage by a factor of 10. The 90% reduction in data transfers resulted in a huge reduction in operating costs, and helped transform the deployment into a viable solution.

With the WindSpring Springboard technology you can get your IoT product to market faster, with low recurring network costs, more securely (using DTLS), without the loss of data integrity. The final product is more efficient and robust, being able to handle larger data transfers then the network would normally allow.

Fundamentally, DIY doesn't work because most data compression has been designed for PCs and larger data files. Compressing in a constrained environment is significantly more difficult. As mentioned earlier, the data packet sizes are often dwarfed by the protocol overhead, and traditional compression techniques don't help reduce the overhead. Springboard is specifically designed for these environments.

Summary and Conclusions

IoT communications links aren't always robust or cheap. Communicating data and updates use precious power, device resources, and network bandwidth, all which are key expenses in both

deployment and maintenance. There are other factors to take into consideration, such as a faster time to market and a more capable product. While there is no single solution to eliminating network bandwidth constraints, leveraging the best applicable compression technology is one of the best alternatives that can also reduce transmissions costs and power consumption. Today, WindSpring's SpringBoard offers the best compression technologies we've seen.

When an IoT data channel is metered or has limited bandwidth or an end device has limited resources, compressing data transmission can be a critical and cost effective solution. The best option is to license compression technology from a company that has the expertise and the technology to do it right. In the end, you will save on operating costs, reduce both end-node memory requirements, and channel bandwidth requirements. WindSpring is a company that can deliver the best technology and understands how deliver data for the lowest possible cost and fast time to market.