LIFE.net over Web: an Advanced Monitoring Protocol for UPS Systems

Paolo Mistroni*, Carlo Giannelli**, Paolo Bellavista**, Letizia Ghirardello*
Chloride UPS Systems*
Via Fornace, 30 - 40023 Castel Guelfo (BO) - ITALY
Phone: +39-0542-632358; Fax: +39-0542-632251
[paolo.mistroni, letizia.ghirardello]@chloridepower.com
Dip. Elettronica, Informatica e Sistemistica - Università di Bologna**
Viale Risorgimento, 2 - 40136 Bologna - ITALY
Phone: +39-051-2093001; Fax: +39-051-2093073
[cgiannelli, pbellavista]@deis.unibo.it

***Abstract* - The remote monitoring and control of UPS devices performed remotely by control centers is a commonly adopted solution to limit required human interventions on site. Nowadays, the most usually adopted communication link is still represented by Public Switched Telephone Network (PSTN) lines, which offer relatively limited bandwidth at relatively high economic costs. The paper proposes a novel control center communication solution for UPS devices that exploits Internet-based connectivity instead of PSTN lines. The Internet represents a widely accepted standard; its adoption as communication infrastructure relevantly simplifies the management and integration of UPS monitoring solutions with other Internet-based systems and tools. For instance, it facilitates the provisioning of UPS information via standard Web browsers, possibly integrated with other customer-related data, such as billing information. In addition, the Internet exploitation allows the transmission of great amounts of monitoring data, thus enabling fine-grained control, at reasonable economic costs. In particular, the proposed solution is based on the encapsulation of standard UPS data and commands inside Hyper Text Transfer Protocol (HTTP) packets, exchanged between UPS devices residing inside enterprise networks and a HTTP server located in France at the edge of the Chloride private network. By exploiting HTTP as encapsulating protocol, there is the positive side effect of overcoming usually adopted enterprise security policies, which limit network traffic between local and remote nodes, thus facilitating the deployment in enterprise scenarios with no intervention on usual security settings. The paper reports implementation details that point out the feasibility of the proposed solution, in relation both to the client side capability to perform on microcontrollers with limited resources and to the server side to properly scale when managing large numbers of simultaneous clients.**

I. INTRODUCTION

The monitoring and control of UPS critical power supplies are commonly adopted to simplify (and increase the efficiency of) several management operations, e.g., to decrease site interventions and maintenance, to reduce human error risk, and to gather data for statistical analysis. To this purpose, there is the need for suitable mechanisms and tools for UPS information monitoring and system control. Common monitoring data are alarms, events, measurements, data recording, and event logs, while UPS control operations include equipment commands such as alarm acknowledgements, configuration, and settings.

Offering UPS monitoring and control remotely performed by experts requires an advanced and flexible management system and a suitable service organization. However, the complexity of such a monitoring/control system largely increases when data are gathered not only from a single UPS installation but from a large UPS population installed in different and geographically distributed sites, belonging to different customers within different organizations and management constraints.

Our current LIFE.net system is able to provide these challenging functions specialized for Chloride UPS equipments. The data gathering system exploits the Public Switched Telephone Network (PSTN) to receive UPS information from heterogeneous sites in a data center accessible for expert analysis. However, the continuous growth of monitored UPS devices, the constant demand of new services and monitoring capabilities, and the relative difficulty in having available PSTN lines in UPS installations are pushing the development of novel communication modes, both to improve monitoring/control capabilities and to reduce communication costs.

The paper describes our experience in migrating from the PSTN lines used by the current LIFE.net system to the standard IP-based network infrastructure exploited by the novel LIFE.net over Web solution, thus allowing the Internet-based interconnection of already installed UPS devices with our data center. Let us additionally note that there is not a single centralized data center; instead, it is distributed in different localities, thus allowing to allocate the monitoring load to several control sub-centers, also in relation with the geographic distribution of UPS devices. In particular, the proposed novel LIFE.net over Web protocol is

based on the Hyper Text Transfer Protocol (HTTP) [1] encapsulation of the LIFE.net protocol. The exploitation of HTTP as encapsulating protocol permits to achieve the non-negligible benefit of the conformance to standard Internet solutions, such as Web server technologies to manage HTTP traffic. In addition, adopting HTTP as transmission protocol permits not to be filtered out by most common security policies installed at enterprise firewalls, without requiring any direct manual intervention of enterprise network administrators. The drawback is the necessity of reproducing both a multi-step session abstraction and the capability to start the interaction from the server-side, which are features that standard HTTP tends not to provide but the LIFE.net protocol strictly requires.

The rest of the paper is structured as follows. Section 2 motivates the adoption and presents the main characteristics of HTTP as tunneling protocol for the standard LIFE.net protocol. Section 3 presents the general architecture of the proposed solution, by depicting how the HTTP encapsulation of LIFE.net is actually performed. Section 4 provides most relevant implementation insights and experimental evidences of the feasibility of the proposed solution. Section 5 compares the adopted HTTP tunneling solution specifically implemented for Chloride LIFE.net with other general-purpose ones. Conclusive remarks end the paper.

II. MOTIVATIONS

Traditional monitoring and control solutions for UPS systems mainly rely on limited bandwidth PSTN lines at not negligible economic costs. The current version of LIFE.net is based on the idea of buffering every event transition on local memory and periodically, usually once a day, transmitting them to a dedicated LIFE.net Watch Station (LWS). At every communication the UPS device provides the minimum and maximum values of locally performed analogue measures, e.g., battery voltage or load currents, gathered after the last communication with the LWS. In particular, the LIFE.net protocol is half-duplex and consists of three different layers. Each layer provides services for the upper layer and performs specific actions: Layer 3 is the physical layer, Layer 2 is in charge of fragmenting and eventually retransmitting packets, Layer 1 exchanges protocol commands and replies at a higher level of abstraction, with LWS that plays the role of master and the UPS device acting as slave.

The requirement of richer monitoring/control capabilities and lower costs push for novel and more powerful solutions taking advantage of the widely available Internet-based connectivity. In fact, in order to achieve a more fine-grained UPS monitoring and control, it is required to gather amounts of data largely greater than the currently connected, possibly more frequently, e.g., the battery voltage every 10 minutes instead of once a day. However, that is not currently feasible because the adopted PSTN lines limits the available link bandwidth and increases communication costs. In order both to get a greater bandwidth and to limit communication costs, there is the need to exploit the IP-based network infrastructure commonly available in almost all the served companies.

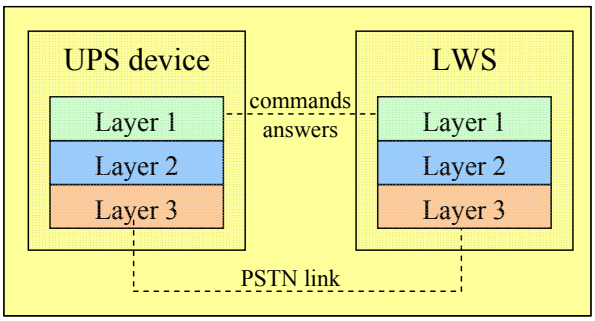

Figure 1: UPS - LIFE Watch Station protocol stack.

Besides the greater bandwidth and the limited costs, the adoption of IP-based communication protocols provides other valuable advantages. First of all, it permits the easy integration of our monitoring solution with other IP-based systems. For instance, it could be possible to easily integrate UPS data with other customer-related information, such as current billing conditions dynamically gathered on the Web. In addition, there is the valuable possibility of exploiting standard software and hardware solutions to offer access to the collected and managed monitoring information. For instance, Web servers installed on Internet-based enterprise networks can enable the access to UPS gathered information via standard Web browsers.

However, common enterprise network configurations do not allow a direct communication between UPS devices located inside the local network and control centers reachable via the Internet. In fact, enterprises generally adopt firewall-based network security policies to block data traffic from the Internet to their internal network and to limit the data traffic from the internal network to the external Internet. Generally, standard services of wide interest such as Web browsing are allowed, thus enabling HTTP clients within a corporate internal network to communicate with HTTP servers over the Internet.

Therefore, due to the wide adoption of HTTP and the availability of several mechanisms to manage HTTP traffic, HTTP seems to be an excellent candidate for our purpose of porting the standard LIFE.net protocol to the Internet, with the non-negligible benefit of not requiring any modification to the network security policies commonly adopted in most enterprises. The basic HTTP is rather simple. First of all, it is request-response, i.e., the client always starts the communication by performing a request to a server, and it is the contacted server that always ends the communication, immediately after returning a response. In addition, HTTP is one-shot, i.e., a communication act consists of only one request and only one response, after which the client-server connection is closed. Finally, HTTP is stateless: there is no

correlation between successive requests from the same client to the same server, which is not required to maintain any state related to the communication stage with that client. Let us rapidly observe that, while their primary objective is to ask for information, HTTP clients are even able to send data to servers. For example, HTTP clients can transmit data to HTTP servers as POST requests, sending information as key-value pairs. The simplicity of HTTP, based on few straightforward interactions between a client and a server, made possible the widespread and rapid availability of scalable Web applications on heterogeneous distributed nodes.

Fig. 2 shows the communication schema that the current PSTN-based LIFE.net protocol adopts. The UPS device directly communicates with LWS via a PSTN line, without any other intermediate component between them. Both UPS device and LWS can start the communication by performing a PSTN call.

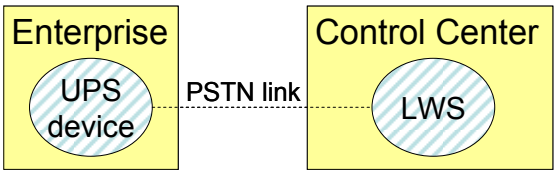

Figure 2: Traditional LIFE.net components and protocol link.

Fig. 3 considers the alternative scenario that LIFE.net over Web aims to support. The UPS device and LWS are not connected directly via PSTN lines, but instead via the Internet through intermediate components (described in the following), adopting HTTP as communication protocol. In particular, Fig. 3 depicts the most common case of enterprise network architecture: enterprise network nodes (and also the UPS device) access the Internet via a proxy server, acting as a firewall, that allows only HTTP traffic with requests generated inside the enterprise network. LWS is connected via the Internet directly, acting as an HTTP server able to manage HTTP packets.

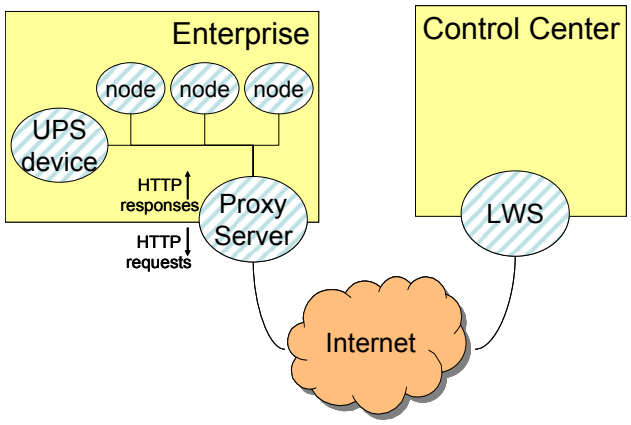

Figure 3: Most diffused enterprise network configuration.

LIFE.net over Web carefully takes into account these common network topologies to facilitate its immediate deployment in enterprise networks without requiring any modifications to commonly adopted network architectures and security policies. In particular, LIFE.net over Web encapsulates the traditional LIFE.net protocol inside HTTP, by performing UPS-specialized HTTP tunneling: the primary idea is to realize communications from the UPS device to the control center as HTTP requests, from the control center to UPS device as HTTP responses. The HTTP adoption is required to communicate without any policy modification between a UPS device residing inside an enterprise network and a control center located in the Internet. However, the simple adoption of HTTP is not sufficient: UPS monitoring and control protocols, such as LIFE.net, require features that the standard and simple HTTP typically does not provide, such as control center-driven start of communications and maintenance of connection state. For instance, a typical LIFE.net communication consists of a connection phase, the UPS device providing its identification details, a conversation phase, allowing the actual transmission of data and commands, and a disconnection phase, when the connection is closed.

To this purpose, we have designed and implemented the LIFE.net over Web protocol, which exploits HTTP, thus achieving the benefit of adopting a widely accepted communication standard, while overcoming its major UPS-related drawbacks. In particular, LIFE.net over Web exploits HTTP by adding the capabilities of both starting the communication from outside the enterprise network and maintaining connection states.

III. LIFE.NET OVER WEB

As emerged in the previous section, the standard LIFE.net protocol is an articulated monitoring solution providing several features. First of all, each LIFE.net connection is composed by several steps, one related to the state reached by the previous ones. Secondly, both the UPS device and LWS can start the communication interaction, in order to send data and commands respectively. Thirdly, while the standard LIFE.net protocol is based on a pre-defined sequence of commands sent by LWS, during a LIFE.net connection it is also possible to switch to an on-line mode: in this case, LWS stops its pre-defined procedure and a human operator can directly specify other commands manually via a shell-based textual interface. At the end of the on-line phase, the standard procedure continues as usual, until it is complete.

The main objectives of the proposed LIFE.net over Web solution are i) the HTTP tunneling of the LIFE.net protocol to allow the communications between UPS devices and control centers via Internet through enterprise proxies, and ii) the maintenance of the communication state to allow a multiple step communication as required by LIFE.net protocol. At the same time, the proposed solution must allow

both the UPS device and LWS to start the communication and must admit also a human controlled command sequence.

Let us stress that Fig. 3 represents only an ideal deployment scenario: both to minimize on site expensive intervention on already deployed UPS devices and to maintain compatibility with the standard LIFE.net LWS, on the one hand, it is not possible to directly connect UPS devices to the enterprise network and, on the other hand, LWS cannot autonomously perform as Web server. In fact, already deployed UPS devices and LWS are not equipped with HTTP capabilities; for this reason we decided to develop and deploy additional components with the purpose of making possible the UPS-LWS proper communication via HTTP.

As Fig. 4 shows, the adopted architecture, similarly to [2], consists of multiple components, where already deployed LIFE.net ones are not assumed to change:

- UPS is the actual UPS device;
- HTTP Client (HC) communicates with both the UPS device (via standard LIFE.net protocol) and server side (via HTTP);
- Proxy Server (PS) is the enterprise proxy server;
- HTTP Server (HS) communicates with HC via HTTP and with the server side via standard sockets;
- LIFE Gate (LG) forwards communications to the appropriate local control center;
- Life Watch Stations (LWSs) are multiple control centers geographically distributed in different locations.

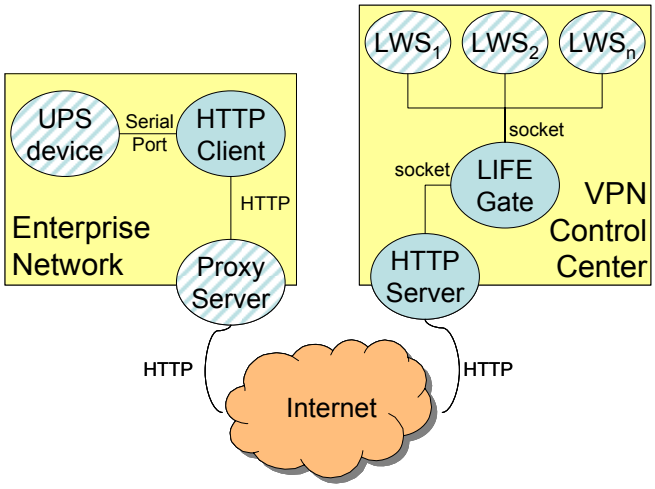

Figure 4: LIFE.net over Web architecture (striped ovals represent already deployed LIFE.net components, while filled ovals are the components originally added by LIFE.net over Web).

Delving into finer details, HC is the component encapsulating the LIFE.net protocol in HTTP packets: it communicates directly with the UPS device, emulating the role of a PSTN modem to gather data and to send commands; it sends data to HS as HTTP POST requests via the enterprise PS and Internet, by possibly providing HTTP authentication credentials to PS if required. HC starts a LIFE.net standard communication at every fixed time interval, e.g., every hour, by communicating, on the one hand with the UPS device via a serial port to gather data and to send commands, and on the other hand with HS via HTTP to provide gathered data and receive commands. In addition, at a higher frequency if compared with the standard LIFE.net, HC starts a heartbeat connection with the purpose of notifying to the server side it is still alive and able to correctly communicate. Note that the usual HTTP request-response process is inverted in this case: HC sends information via HTTP requests, HS sends commands via HTTP responses. This behavior is mandatory due to commonly adopted enterprise network security policies. In fact, HC is the only component which can start a HTTP communication, while HS can send data to HC only via HTTP responses and only after a previously received HTTP request.

HS is the component that provides a LIFE.net connection abstraction maintaining the connection state. In particular, it is implemented as a standard Web server: it accepts HC HTTP requests, communicates with LG via sockets by sending HC data and receiving LWS commands, and sends commands to HC as HTTP responses. Let us note that HS maintains the connection state to allow multi-step communications, by associating any on-going session of communications with a unique identifier. The state mainly includes the identifier of the socket towards LG, opened at the beginning of each LIFE.net connection and maintained until its completion. Delving into finer details, at the beginning of each LIFE.net connection, HS opens a new socket to LG and associates it to a unique identifier, namely the connection identifier. At each HTTP response, HS appends the connection identifier to the other commands related to the LIFE.net protocol; in this manner, HC gathers the connection identifier and can specify it in the following HTTP requests, thus permitting to HS to retrieve the already available LIFE.net connection state.

As better detailed in Section 5, HTTP tunneling is a commonly adopted solution to overcome the limitations that HTTP proxies impose. However, the proposed solution not only provides the capability to maintain the state of the communication, but also is able to invert the standard request-response HTTP communication model, i.e., sending commands via HTTP responses and data via HTTP requests.

LG and LWS are completely unaware of the adoption of HTTP. The former is in charge of forwarding connection data to the correct LWS, the latter is the component that actually manages UPS device data and sends commands. Different LWSs may be deployed in different geographical locations, in order to distribute the monitoring and control procedures close to the actual location of managed UPS devices. LWSs are not able to directly start a communication with a UPS device, as it was possible with the previous LIFE.net protocol based on PSTN lines, due to usually adopted enterprise

network traffic limitations. However, each LWS can start the LIFE.net over Web connection, making manifest its desire to communicate with a given UPS device by providing LG with the UPS device serial number. At the following UPS device heartbeat, LG will notify to the UPS device that there is a pending reservation, and the UPS will start the communication immediately, even if the given time interval is not yet elapsed.

IV. Implementations Insights and Experimental Details

The presented architecture has been developed and deployed in order to test both its effectiveness and efficiency. In particular, HC has been implemented in the C language, by exploiting standard library functions provided by the POSIX interface, e.g., the *pthread* library for multithreading purposes and the *termios* functions for serial port interaction and control. The exploitation of the standard POSIX API enables the portability of our HC implementation on any Linux-based platform, included μCLinux [3], the Linux operating system specifically designed for devices with limited capabilities. In particular, HC has been actually deployed and tested on a MOXA UC-7112 [4] device, a microcontroller equipped with a serial port and an Ethernet end point and provided with a μCLinux environment. Let us stress that our solution is completely transparent from the point of view of the UPS device, requiring no modifications to already installed components. In fact, HC communicates locally with the UPS device acting as a standard PSTN modem, interacting via the serial port, and communicates remotely via the Internet with the control center, encapsulating the LIFE.net protocol in HTTP packets.

HS has been implemented in terms of Java HTTP servlets, a server side technology to manage HTTP requests and provide HTTP responses. While it is possible to deploy Java servlets on both most spread Web servers, i.e., Apache HTTP server [5] and Microsoft IIS [6], for the sake of simplicity, we have adopted a standalone Tomcat [7] HTTP server (version 5.23) as servlet container, installed on a HP ProLiant DL380, 1Gbyte RAM.

Between HC and HS, we have alternatively deployed either a Squid proxy [8] (version stable.2.51) or a ISA Server 2004 [9]. Both proxy servers have been configured in order to allow only outgoing HTTP requests and ingoing HTTP responses, while discarding any other traffic category, e.g., not permitting FTP traffic and closing any network port different from 80 (the standard HTTP port). Moreover, both proxy servers have been alternatively configured to either request or not HTTP basic authentication: in the case of HTTP basic authentication, HTTP traffic is forwarded only if HC provides correct credentials as a user name/password pair.

In addition, we have performed some experimental tests with the twofold goal of i) evaluating HS scalability in terms of served clients and transferred data, and ii) of verifying the reliability of the HC implementation on the MOXA UC-7112 device. In particular, in order to verify the adopted solution performance, we have deployed a test prototype on a laptop configured with a Linux operating system (Ubuntu 6.10 distribution) equipped with Ethernet and RS-232 ports. The prototype does not actually interact with a single UPS device, but performs multiple LIFE.net over Web connections simultaneously (emulation of multiple HCs), by transferring a variable amount of data. In particular, we have tested a number of simulated clients which ranges from 1 to 100, transferring 100 byte, 1 Kbyte, or 50 Kbyte per connection, each connection repeated periodically (10 seconds). Let us stress that the emulated HCs are particularly challenging since successive HTTP requests are performed largely more frequently than in common industrial deployment scenarios. The main goal is to test the computational load imposed by HTTP servlets, to point out the scalability of the proposed solution with the number of served clients and the size of transmitted data.

Our experimental results about the performance of LIFE.net over Web demonstrate the feasibility of the approach even when considering cases that generate a huge amount of monitoring traffic with stringent delay requirements. In fact, the proposed solution provides great scalability, not presenting notable performance degradation, even when 100 clients transmit 50 kbyte per connection. In addition, our experiments have not exhibited any notable performance difference in the case of basic HTTP authentication or not. Let us additionally note that the server used in these experiments has limited capabilities if compared with a usual HTTP server host; thus, we expect even better performance in a real-world scenario.

Finally, we have tested our prototype on the MOXA UC-7112 device (see Fig. 5). In particular, the compliance with the POSIX standard has permitted to rapidly port our prototype on a μCLinux environment without any relevant coding issue. The main technical aspect to be re-considered was due to the absence of the memory management unit in μCLinux, which required specific care in dynamic memory allocation. In addition, as a further mechanism to provide reliability, we have included a watchdog in the HC implementation in order to avoid locking conditions. We have measured the system overall performance and found it in line with the results of the preliminary tests with the laptop-based deployment environment.

As a final remark, let us notice that the MOXA device is also connected to an external device, such as an operator laptop, via serial port for configuration purposes (as depicted in Fig. 5). In that way, it is possible to set the main configuration parameters of the HTTP client, such as IP addresses of PS and HS, and authentication credentials when required.

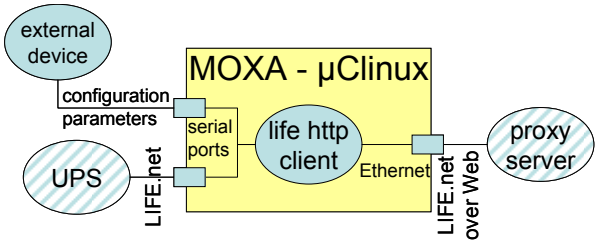

Figure 5: LIFE.net over Web deployment on the MOXA UC-7112 device.

V. Related Work

HTTP tunneling is a well-known mechanism to enable communications between nodes residing in different networks separated by proxy servers. This related work section briefly focuses on the differences among our HTTP tunneling solution and other already available ones, with the purpose of clarifying why an ad-hoc tunneling solution was required in our UPS-specific environment.

Many commercial projects provide general-purpose solutions specifically developed for Windows platforms [10], [11]. We have preferred to develop our own lightweight and portable HTTP client by limiting the set of supported features to only the strictly required ones, in order to impose only a limited computational load. [12] represents an interesting open source solution, available even for Linux operating system. However, it does not perform memory management efficiently; therefore, we claim that it is not suitable for critical applications, such as UPS monitoring and controlling, with strict availability requirements (24 hours per day, 7 days per week) and for running at devices with limited memory resources, e.g., MOXA UC-7112 devices.

In addition, it is not clear if the above products can effectively manage some aspects of the HTTP protocol (usually not addressed since not widely exploited) that are crucial in relation with our UPS-related application. In particular, LIFE.net over Web has requirements on URL safe base64 encoding (required to send arbitrary binary data via POST requests) and automatic client-to-proxy server socket re-establishment (required whenever a HTTP server closes a connection - *Connection: close* HTTP header).

VI. Conclusions

While UPS device monitoring and control are widely accepted as crucial aspects of power supplying, the lack in communication capabilities in terms of both bandwidth and costs has greatly limited its performance and wide applicability. The paper demonstrates how it is possible to greatly improve monitoring and control facilities by exploiting, as communication medium, the Internet in place of traditional PSTN lines. In particular, the developed solution permits the communication of UPS devices and control centers (geographically distributed in several locations) via the Internet, in a completely transparent manner, thus allowing its adoption even for already deployed UPS devices. LIFE.net over Web achieves the twofold goal of transmitting a much greater amount of data between UPS devices and control centers while minimizing economic costs. The preliminary experimental results reported in the paper have demonstrated that the proposed solution has great scalability, thus permitting its industrial adoption also in deployment scenarios with a large population of UPS devices.

Acknowledgments

Work supported by the MiUR PRIN MOMA, the MiUR FIRB TOCAI, and the CNR Strategic IS-MANET projects.

References

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", http://www.ietf.org/rfc/rfc2068.txt, January 1997.
[2] "Tunneling through the corporate network", http://www.ibm.com/developerworks/java/library/j-tunnel/
[3] µCLinux, http://www.uclinux.org/
[4] MOXA Embedded Computers, "UC-7112/UC-7110", http://www.moxa.com/product/download/UC-7112 _7110.pdf
[5] Apache, http://www.apache.org/
[6] Internet Information Services (IIS), http://www.microsoft.com/windowsserver2003/iis/default.mspx.
[7] Apache Tomcat, http://tomcat.apache.org/
[8] Squid Web Proxy Cache, http://www.squid-cache.org/
[9] Microsoft Internet Security and Acceleration Server (ISA), http://www.microsoft.com/isaserver/default.mspx
[10] HTTP-Tunnel, http://www.http-tunnel.com/
[11] Hopster, http://www.hopster.com/
[12] httptunnel, http://www.nocrew.org/software/httptunnel.html