Intelligent Devices for Appliances Control in Home Networks

A. Leventis, Th. Antonakopoulos, Ch. Stavroulopoulos
Th. Luckenbach and V. Makios

IEEE Transactions on Consumer Electronics

VOL. 49, NO. 2, MAY 2003, pp. 328-336

Intelligent Devices for Appliances Control in Home Networks

A. Leventis, Th. Antonakopoulos, *Senior Member, IEEE*, C. Stavroulopoulos, Th. Luckenbach and V. Makios, *Senior Member, IEEE*

Abstract — *In this work, the architecture and functionality of intelligent, reduced-complexity devices used in appliances control for networking applications are presented. These devices exploit the Ethernet infrastructure in order to offer network-based control functionality. The devices implement a hard-wired version of the IP-based protocol stack and use an embedded soft processor core for executing the functions of a control-oriented Application Layer. The protocol stack used is based on a limited functionality MAC layer, while packet filtering is performed at the IP layer. Simulation results prove the exceptional characteristics of the proposed solution in terms of response time under various traffic conditions.*[1]

Index Terms — Home Appliances, Home Automation, Local Area Networks.

I. INTRODUCTION

MANY consumer products extend their functionality by integrating networking functions and remote access capabilities. Examples include a digital refrigerator that utilizes on-line access for shopping and product price information update, the prototype of an internet microwave oven that is capable to download cooking recipes from the Internet and an internet connected washing machine that utilizes a web-server for controlling its operation and retrieving status information. Compared to existing facility management technologies, network operated control devices offer numerous advantages such as centralized control of appliances, remote sensor reading and status diagnostics.

Various approaches towards network integration of intelligent devices have been proposed [1], [2], [3] and many alternative solutions exist regarding the physical medium that can be utilized. Available choices include, but are not limited to, the use of wireless media, power-line communications, unshielded twisted-pair, Ethernet infrastructure and dedicated buses (e.g. IEEE1394). Due to the heterogeneous Quality of Service (QoS) requirements of applications used in SOHO (Small Office/Home) installations, no universal approach can be adopted, but each case is separately evaluated depending on its particular features. Undoubtedly, Ethernet is one of the most attractive networking solutions, in terms of cost, reliability and available bandwidth. Due to its popularity, many new buildings include pre-installed support for Ethernet networking. Since this infrastructure provides a viable solution for network-based management systems and automation services, many communication protocols have been developed to support such services. UpnP, Jini and HAVi are typical examples of such protocols that focus on developing reliable connectivity solutions among stand-alone devices.

Although these protocols can be used efficiently to handle control procedures in SOHO applications, they depend on services provided by the underlying protocol stack. For example, UPnP requires implementation of UDP/TCP, HTTPMU/HTTPU, GENA and SSDP/SOAP. Likewise, Jini is completely based on Java. Both UPnP and Jini, aim to provide services on devices in an ad-hoc manner. HAVi on the other hand, which is based on the IEEE1394 bus, is oriented towards the interconnection of Audio and Video devices. All protocols described above, use the full protocol stack in order to provide control services that may be adequate or ample, depending on the application requirements. However, most appliances control functions can be realized by using a limited communications protocol functionality and a combination of primitive transactions among the communicating entities.

In this paper, we present the implementation of intelligent devices for appliances control, using Ethernet as networking technology. The devices use a control-oriented Application Layer that utilizes the services of a minimized communications protocol stack. To retain the simplicity of the total system and provide a cost-effective solution, an approach towards minimal implementation of the IP-related protocols is adopted.

Field Programmable Gate Arrays (FPGAs) provide the ideal platform to prototype such devices [4] and similar approaches have been presented in the literature. In [5], a simple web-server was built by using the TCP and HTTP protocols in an FPGA supported by an external Ethernet MAC/PHY controller. That system performs sensor reading functions and remote control of electrical appliances. However, that design focuses its optimization approach on the protocols above the Network Layer. The system described herein, is based on the analysis of all communication layers and the exploitation of their features in order to build a system incorporating the minimum possible functionality and hardware complexity.

[1] This work was financially supported by the FhG/Fokus Institute under the framework of project *S2IPD*.

A. Leventis, Th. Antonakopoulos, C. Stavroulopoulos and V. Makios are with the University of Patras, Department of Electrical Engineering and Computers Technology, 26500 Rio - Patras, Greece (Tel +30 2610 997346, e-mail: antonako@ee.upatras.gr)

Th. Luckenbach is with the Fraunhofer Institute for Open Communication Systems (FOKUS), Berlin, Germany (Tel: +49 30 3463-7245, e-mail: luckenbach@fokus.fhg.de)

Contributed Paper
Original manuscript received January 10, 2003
Revised manuscript received February 25, 2003

Section II presents the total system design approach, the Application Layer functionality and the required protocol stack. Section III provides simulation results to demonstrate the response time of the proposed devices under various traffic conditions, while Section IV highlights the devices' architecture and provides implementation details.

II. SYSTEM FUNCTIONALITY

Control applications related to appliances and home automation, consist of multiple functions interacting in a distributed environment. An application may require a simple data exchange between two remotely located devices, or multiple transactions of information exchange between two or more devices, using one or more functions in each device. By analyzing different types of control applications, two types of basic functions have been defined, the '*transaction-initiator*' and the '*response-generator*'. The '*transaction-initiator*' function is triggered by external, application-related events (e.g. a user requires the activation of a remote device). '*Transaction-initiator*' functions generate requests and interact with associated '*response-generator*' functions, which are executed on remote devices, in order to perform the requested operation. The '*response-generator*' function is activated when a new request has been received, the respective command is executed and the response to the '*transaction-initiator*' is generated. Both functions are implemented using two types of logical devices. Each physical device may contain one or more logical devices of one or both types. When multiple logical devices are used in a single physical device, a '*function coordinator*' is used to support the multiple logical devices' functionality.

It is evident that the communication between a '*transaction-initiator*' and a '*response-generator*' forms a 'client-server'-like protocol. As it is shown in Fig. 1, the "client device" (implementing the *transaction-initiator* function) sends control commands to the "server device" (implementing the *response-generator* function); the "server device" accepts the incoming commands, updates the status of the appliance it controls and acknowledges the control command. This 'client-server' approach supports various applications typically found in control networks, as well as many functional operations, like having more than one clients per server device, or more than one servers per client device. Organization in logical groups for establishing sets of devices dedicated to specific control functions is also possible. The most attractive feature of the client-server approach however, is its scalability. Since complicated message transactions can be reduced to the exchange of simple message sets, minimal hardware architecture can adequately support the overall functionality and allow for possible upgrades. When a client device controls more than one server devices, it performs a separate transaction with each server device it controls, until all of them have executed the required command. Though broadcasting of the control command seems more appropriate in terms of response time, in fact it is not the most stable and reliable solution; In such a case, all server devices might try to transmit the acknowledgement response simultaneously, thus causing collisions and unpredictable delay or even loss of some acknowledgement messages. Simulations were performed to investigate the average response time under various traffic conditions. The results, presented in detail in the following section, indicate that even for a large number of network nodes, the average response time of the approach presented herein is in the order of a few milliseconds.

In the case where a server device is associated with more than one client devices, no client device is aware of this association. Each client device initiates a transaction when it is triggered by an external event while the server device, which only executes incoming commands, responds to the commands received regardless of their initiator.

The approach discussed, was incorporated in the Application Layer functionality in order to retain the simplicity of the total

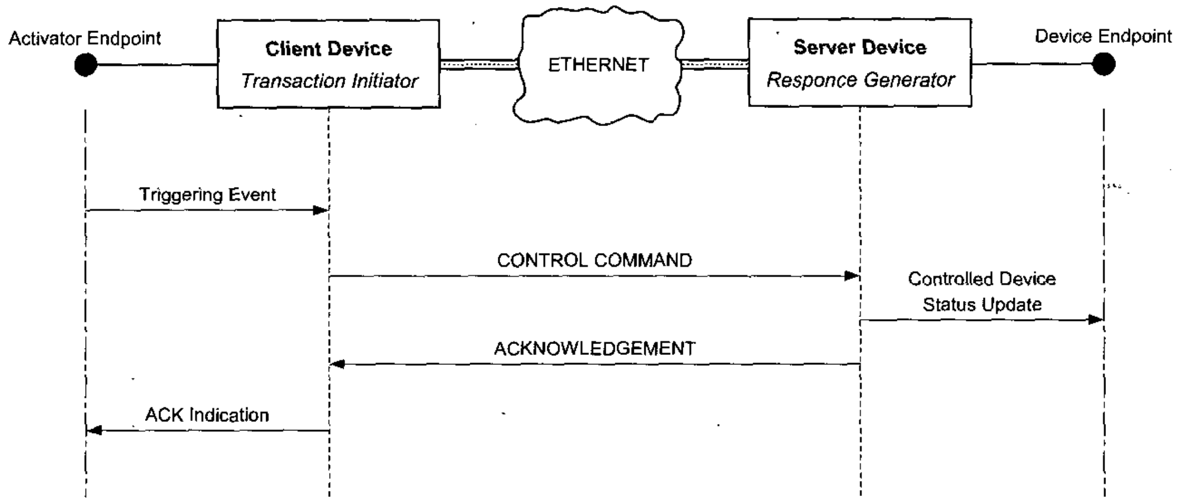

Fig. 1. Application layer functionality

system. Compared to the numerous existing, well-defined application layer protocols, this approach was considered more appropriate for control applications. In many cases, various protocols that support control functions were not primarily designed for that purpose. Such an example is the SIP protocol [6] that, despite being used for control applications [7], was aimed to support Internet telephony and multimedia functionality. In addition, application layer protocols depend on the implementation of lower-level protocols to support their functionality, which is not the case for the approach adopted.

In order to implement an extremely compact version of the communication functions included in the TCP/IP protocol stack, the various parts formulating the stack were analyzed in order to evaluate their importance and contribution to the client-server communication scheme described above. The most demanding part of the TCP/IP stack is the Transport Layer protocol. As already discussed, the control devices exchange messages of limited length. In addition, the inter-arrival time between successive activation commands varies from some milliseconds up to several seconds. Therefore, the flow control mechanism used in the TCP protocol is not necessary, while the connectionless UDP protocol is considered more appropriate. Since the UDP protocol cannot provide any flow control functionality, a limited amount of flow control has to be added into the Application Layer. Hence, the Application Layer performs retransmissions and timing control in order to re-issue any lost control messages.

In the Internet Layer, the IP protocol coordinates the correct exchange of datagrams among the communicating nodes. Since the IP datagrams exchanged among control devices have most of their header fields predefined (except run-time dependent fields, like source and destination addresses or checksums), the Internet Layer can safely discard incoming frames that have header field values not matching the expected ones. The same technique can be applied to the Network Access Layer: incoming frames can be filtered to exclude non-IP frames, or frames of wrong size.

In Ethernet networks, the ARP protocol is used to discover a node's physical address when its IP address is known. Given the limited size of control messages, in our approach, the implementation of the ARP protocol is completely avoided. Instead, the MAC broadcast mechanism is retained and used in all frames transmitted by the intelligent devices. Hence, destination address matching is based on IP rather than physical addresses. Although this is not the "standard" operation of the Ethernet MAC Layer, it is not opposed to the Ethernet protocol. Therefore, if a network node that implements the whole TCP/IP stack receives such a broadcast frame, the frame will be discarded since it contains a datagram, but is not addressed to that particular node. Therefore, only the node that has the same IP address will process that message. The benefit of that modification is the simplification of the MAC layer functionality and, in some cases, the reduction of the messages required to perform a control function.

All the modifications described above lead to a very compact stack, able of sending and receiving frames of a specific header format. A state machine at the receiver may constantly check each field in the header of all incoming frames: only frames complying with the predefined format will be accepted, while the rest will be discarded. Due to the limited functionality of each layer, the same state machine could successively check and compare the headers fields of the IP and the UDP layers too. Hence, Ethernet, Network and Transport Layers functions are eventually merged to one functional unit, responsible for sending and receiving frames having a specified frame header format.

Despite its minimal functionality, the protocol stack developed retains compatibility to existing UDP/IP over Ethernet implementations. Therefore, devices based on that protocol stack can communicate with devices implementing the full TCP/IP protocol stack, such as a Host PC. This allows development of software applications capable to assist the assignment of IP addresses to the devices, query the status of devices in a network and support network maintenance. By using these functions, erroneous network conditions, like having the same address assigned to more than one device can be prevented.

III. Network Performance

In order to evaluate the performance of the developed Application Layer protocol and the protocol stack described, a simulation model was created. Various simulation scenarios were executed, incorporating several tenths or hundredths of control devices sharing the same Ethernet medium. The devices are organized into groups having a client device controlling one or more server devices belonging to that client. All devices process activation type commands (ON/OFF). In this analysis, we assumed that a client device does not keep any information regarding the status of the server devices it controls. Hence, to change the status of a server device, the client has to issue two commands: the first command is used for learning the status of the server device and the second one for changing it. Fig. 2 presents the transactions taking place when a client device needs to update the status of the server device's output.

Since the status of all server devices controlled by a client is always the same, at any time the status of all server devices gets values from a predefined set. Therefore if a client device controls more than one server devices, it may be informed only once about the status of the server devices and then update the status of all devices accordingly. Hence, for changing the status of the server devices in a group consisting of 1 client and k servers, the client issues 1 frame to learn the status of the server devices and k frames to change it (one frame per server device). Consequently, the client device issues $(k+1)$-frames instead of $(2k)$-ones.

Based on the scenario described above, various simulation models were developed in order to investigate the network performance. The inter-arrival time of the triggering events was considered to follow exponential distribution with a mean value

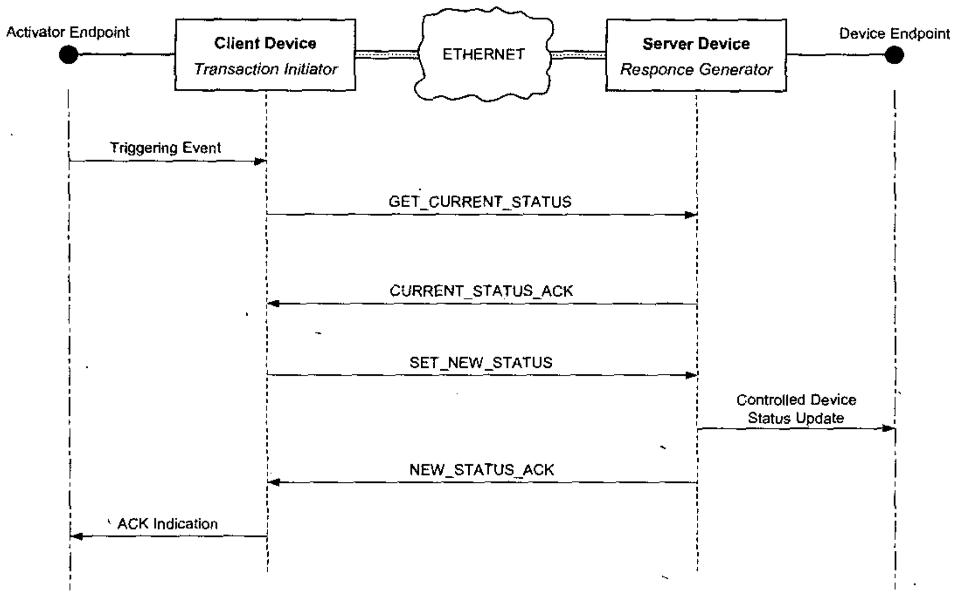

Fig. 1. Commands exchange flow

of 5 minutes. All simulations were run for a 2-hours simulation period. Initially, the response time versus the number of control devices in the network was evaluated, for different numbers of server devices per client. The measurements concern the time from the instance a client device is triggered until the last server device controlled by that client updates its output. A "dedicated" Ethernet network, where no other Ethernet nodes exist except the control devices, was assumed. In a network consisting of 1-client and k - server devices, the minimum time $T_{\min}$ required for updating the status of the k -th device is given by equation (1):

$$T_{min} = T_{setup} + T_{AccessMem} + (k+1) \cdot \left[(2 \cdot T_{tran}) + T_{Client} + T_{Server} \right] \quad (1)$$

where T_{setup} is the time necessary for the client device to acknowledge the triggering event, T_{tran} is the time required for transmitting a 64 bytes Ethernet frame, $T_{AccessMem}$ is the time required for the client device to retrieve the IP addresses of all k servers from the its storage area, while T_{Client} and T_{Server} are the packet processing times for client and server devices respectively, including the ACK packet processing time. Fig. 3 presents the average response time for a range of 1,000 to 5,000 devices, while Fig. 4 and Fig. 5 present the response time standard deviation and its maximum value, respectively. The results presented are normalized to the server devices count. It can be observed that in every case, the average response time is less than 40 msec. The transmission speed is 10 Mbps and all exchanged packets are 64 bytes long, the minimum packet length according to the Ethernet specifications. The packet processing times are based on the implementation details discussed in the next section. The case where the server/client ratio is 1:1 is of interest since this is the worst response time. When more than one server devices are controlled by one client, the client device needs to be informed only once about the status of the server devices. Therefore, the number of messages exchanged per client device decreases as the number of servers per client increases.

Furthermore, simulations were performed to investigate the impact of Ethernet background traffic to the performance of the control devices. Two scenarios, considered to be close to actual SOHO environments, were also studied. The first one uses 100 devices (server/client ratio equal to *4:1*), while the second scenario uses 1000 control devices (server/client ratio equal to *10:1*). The background traffic was created by nodes implementing the complete TCP/IP stack. The packet length of the messages exchanged by those nodes was set according to a bimodal distribution, consisting of minimum (64 bytes) and maximum (1536 bytes) length packets. This model provides a more accurate representation of the Ethernet network traffic [8], since the usually assumed exponentially distributed packet lengths do not match the characteristics of actual Ethernet traffic patterns. Fig. 6 presents the average response time as a function of background traffic, while Fig. 7 illustrates the response time standard deviation. The maximum response time is shown in Fig. 8. The simulation results show that in actual SOHO environments, where background traffic is present, the protocol described is still efficient and offers almost real time response characteristics. Even in heavily loaded network conditions, the average response time does not exceed 20ms for large networks. Although in the worst case the maximum response time is about 900ms, this is not considered a drawback of the approach proposed since the system described is targeted to SOHO control applications, tolerant to such response times.

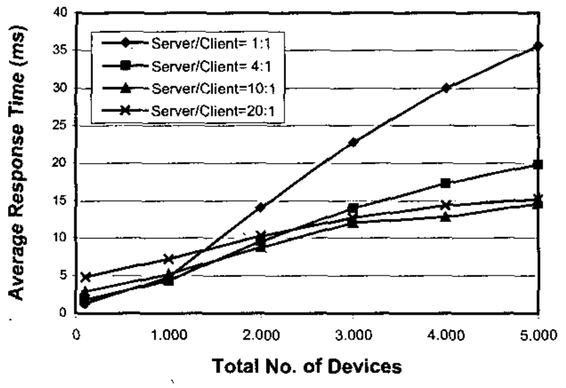

Fig. 3. Average Response Time versus the Number of Control Devices

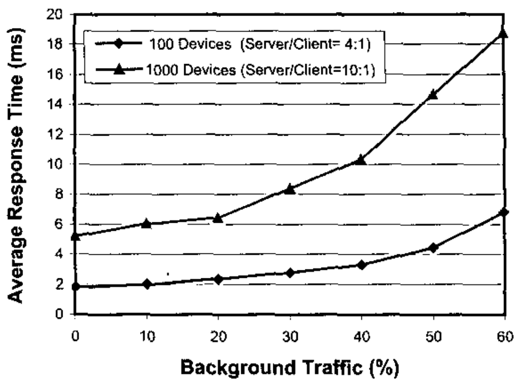

Fig. 6. Average Response Time versus Background Traffic

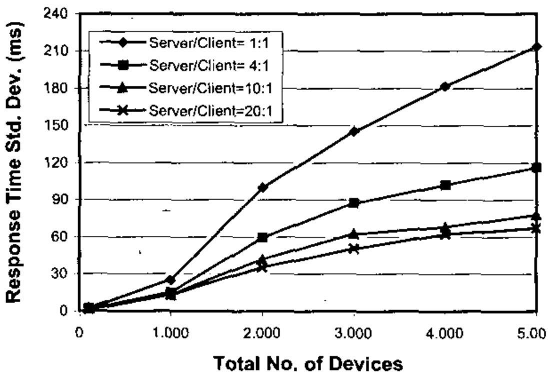

Fig. 4. Standard Deviation of Response Time versus the Number of Control Devices

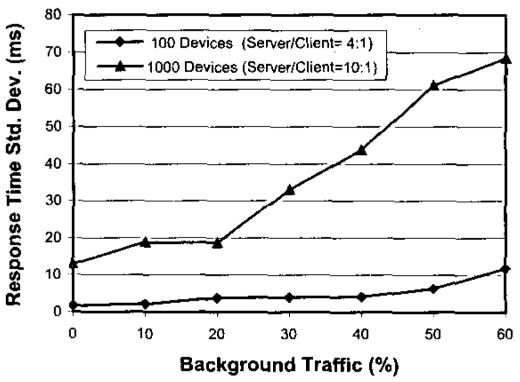

Fig. 7. Standard Deviation of Response Time versus Background Traffic

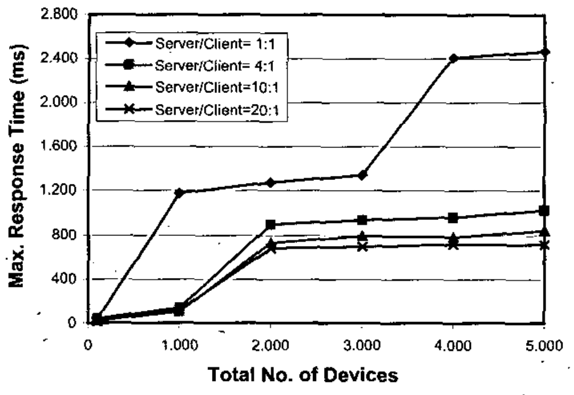

Fig. 5. Maximum Response Time versus the Number of Control Devices

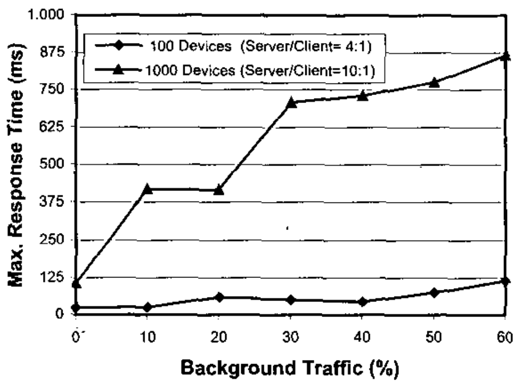

Fig. 8. Maximum Response Time versus Background Traffic

IV. System Architecture

For prototyping the intelligent device architecture, a single FPGA has been used along with an Ethernet Physical Layer interface controller. Fig. 9 presents the block diagram of the architecture employed. Both client and server devices share the same architecture. The device incorporates an 8-bits soft-IP processor core, for implementing the control-related functions. The soft-IP processor is a tiny-footprint core, having memory constraint of 256 bytes program space. A limited set of instructions is supported, which is still sufficient for performing the required functionality. For executing the communications protocol stack, custom logic is employed. This hybrid approach, incorporating an embedded processor core rather than considering an all-hardware (or a software only) implementation of the communication protocols, was proved to offer numerous advantages to the system design, including FPGA logic cells area reduction and development time shortening. To support this hybrid architecture, the system functionality was examined and partitioned among firmware executed in the processor core and dedicated hardware blocks. The processor core implements the Application Layer functions and assists the protocols' execution. Hardware logic was incorporated for implementing the interfacing to the Physical Layer device (using the MII bus), as well as the processing of incoming and outgoing frames. Furthermore, custom hardware was used to build all processor peripheral units, i.e. interrupt controller, timing modules, external I/O interfaces and a small RAM for run-time variables storage. The FPGA is supported by a 3-wire non-volatile memory, utilized for storing the device IP address as well as the addresses of all devices it controls (in the case of a client device). Although during the prototype development phase a soft processor core was used, in a final product licensed processor core can be used and only minor adaptations to the peripheral units' interfaces have to be performed.

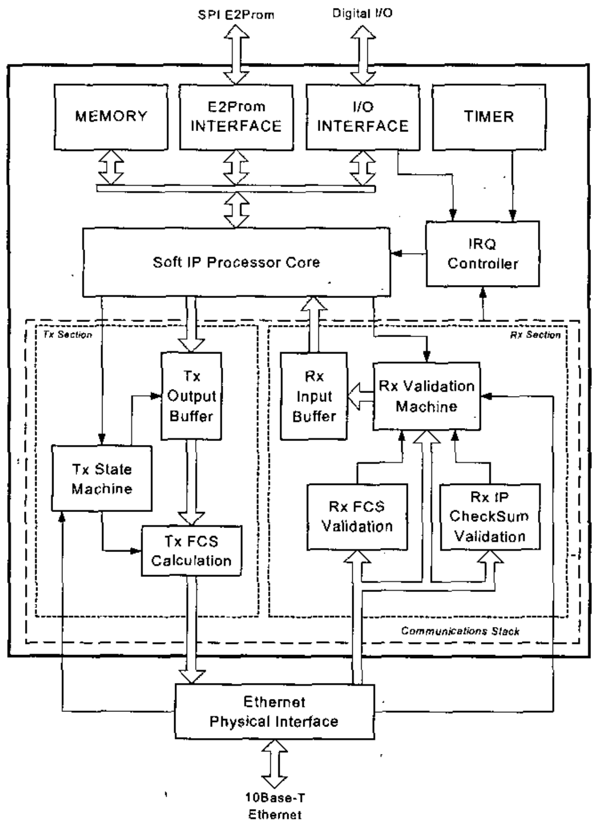

Fig. 9. Functional block diagram of the device architecture

On the receiving side, the Receiver Validation State Machine continuously monitors the incoming data, while the FCS and the IP header checksums are calculated. The state machine is initialized when a new frame is sensed in the Ethernet medium and its state changes progressively as the new frame is received. Each frame field received is either checked against a predefined value or ignored.

Fields that are checked include the Ethernet MAC addresses, the protocol type, IP layer related addresses, length and UDP ports. Header fields not critical to the application described, like TTL, IP fragment offset and IP flags, are ignored. This process is aborted when a header value mismatch is found; otherwise, it is continued until the headers of all layers are verified. When a node receives a frame having as IP destination address its own IP address, the FCS and IP header checksums contained in the frame are compared to the values calculated. Upon matching, the processor is informed to further process the data stored in the Rx Input Buffer, else the buffer is reset. The internal logic of the Receiver Validation State Machine is presented in Fig. 10.

On the transmitter side, the data to be transmitted are placed in the Tx Output Buffer. The processor core generates only a part of the response frame, while the rest frame fields are generated by the hardware Tx unit as the frame is transmitted into the network. During packet transmission, the TxFCS Calculation Module calculates the FCS checksum and appends the calculated value at the end of the outgoing frame. The Transmitter State Machine coordinates the transmission process and implements the MAC protocol functions, such as the collision detection process and the Exponential Backoff algorithm.

For the operation of the devices and the maintenance of the whole network, a minimal set of control commands and respective acknowledgements is incorporated. Table I outlines this set of messages and provides a short description regarding the use of each one.

The complete design was prototyped using a single Field Programmable Gate Array (FPGA). After finishing the FPGA implementation stage, the FPGA area coverage statistics confirmed the initial estimation concerning the area reduction that the embedded processor core would offer to the design. The processor covers less than 14% of the total area consumed by the whole design (250 FPGA CLBs). Since the processor's firmware mainly handles the execution of the Application Layer if the whole functionality was implemented in hardware,

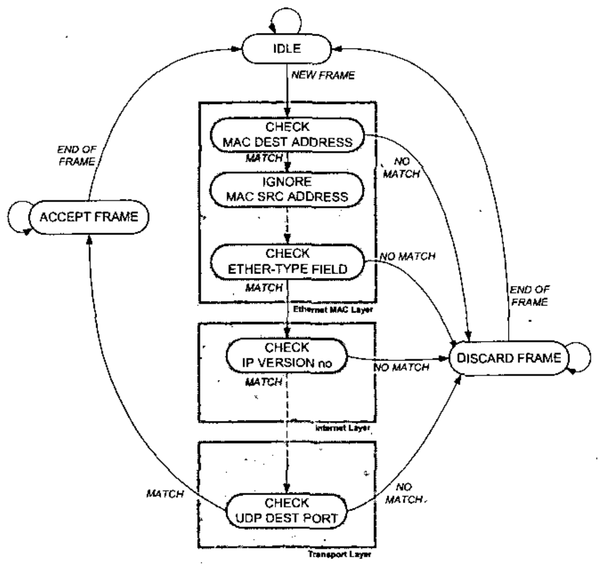

Fig. 10. Receiver Validation State Machine

much more silicon area would be necessary. On the other hand, if more parts of the system functionality were based on software rather than hardware blocks, the increased program memory requirements would urge to using a larger size FPGA. The maximum permitted clock frequency for the design is 25MHz. The prototype uses a 10 MHz external clock set for supporting the transmission speed of 10BaseT Ethernet networks, and its packet processing time is 25 μsec for server devices and 120 μsec for client devices. The difference in packet processing time is because server devices do not hold lists of associated client devices and hence respond to incoming commands regardless of the client that requested them. On the contrary, client devices have to access the external EEPROM in order to retrieve the server device's address prior to initiating a transaction with it.

Fig. 11 presents the prototype of a complete intelligent device. The power supply of the prototype system board is provided by the UTP Ethernet cable according to the IEEE P802.3af [9] draft proposal. The architecture of the device supports various extensions to the functionality offered. Such extensions could include encryption functions and authenticity checking mechanisms. Analog I/O could be supported as well, in order to allow for control functionality based on measurements of physical quantities (e.g. room temperature and light level). In addition, implementation of the IPv6 protocol is also possible. Since IPv6 offers many benefits to control systems, such as larger address allocation range and network auto-configuration functionality, its use would enhance the flexibility of control devices. Finally, remote functionality upgrade of the devices is possible through reprogramming of the software executed by the embedded processor. This can enhance the operation of the devices and allow various customization options.

V. CONCLUSIONS

In this work, the functionality and architecture of intelligent, reduced-complexity devices used for appliances control in networking applications was presented. Implementation of a low-complexity solution was the major design issue and hence the developed device consists of a very compact communications protocol stack combined with an efficient control-related application layer. Despite the optimizations performed to the functionality of the stack elements, the compatibility to the Ethernet protocol is retained. Therefore, host computer applications can be developed to monitor and manage the status of control devices based on the intelligent devices described.

The increasing use of Ethernet networking infrastructure inside buildings, allows such intelligent devices to be employed for remote control services. An important fact however, is the cost of the final product and its ability to be embedded into existing appliances. The system described in this work is believed to meet both requirements and therefore could be successfully adopted in various consumer products.

TABLE I
SET OF MESSAGES SUPPORTED BY PROTOTYPE DEVICES

Message Type	Message Name	Description	Usage
Command	GET_STATUS	Retrieve current Server Device Status	Operation
Response	CUR_STATUS	Current Server Status	Operation
Command	SET_STATUS	Force new Server Device Status	Operation
Response	STATUS_ACK	Updated Server Status Acknowledgement	Operation
Command	START_CONFIG	Force Configuration Mode of Operation	Maintenance
Response	START_CONFIG_ACK	Configuration Mode Acknowledgement	Maintenance
Command	STOP_CONFIG	Force Normal Mode of Operation	Maintenance
Response	STOP_CONFIG_ACK	Normal Operation Mode Acknowledgement	Maintenance
Command	SET_CONFDATA	Force New Device Operational Parameters	Maintenance
Response	CONFIG_DONE	Operational Parameters Set Acknowledgement	Maintenance
Command	GET_CONFDATA	Retrieve Device Operational Parameters	Maintenance
Response	CONFIG_DATA	Set of Device Operational Parameters	Maintenance

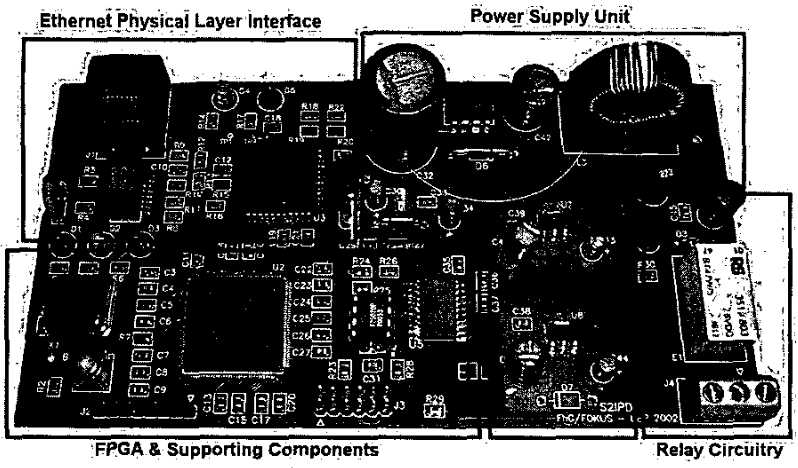
Ethernet Physical Layer Interface
Power Supply Unit
FPGA & Supporting Components
Relay Circuitry

Fig. 11. Prototype Demonstration Board

REFERENCES

[1] D.S. Kim, G.Y. Cho, W.H. Kwon, Y.I. Kwan and Y.H. Kim, "Home Network Message Specification of White Goods and Its Application", *IEEE Transactions on Consumer Electronics*, Vol.48, No.1, February 2002, pp. 1-10

[2] P. Corcoran, "Mapping Home-Network Appliances to TCP/IP Sockets using a Three-Tiered Home Gateway Architecture", *IEEE Transactions on Consumer Electronics*, Vol.44, No.3, August 1998, pp. 729-736

[3] T. Pfeifer, A. Micklei, H. Hartenthaler: "Internet-integrated Building Control: Leaving the Lab - Robust, Scalable and Secure", *26th IEEE Workshop on Local Computer Networks, LCN 2001 Tampa, Florida USA, November 14-16,* 2001 IEEE Computer Society Press, ISBN 0-7695-1321-2; pp. 306-315

[4] H. Fallside, and M. Smith, "Internet Connected FPL", *10th International Conference on Field Programmable Logic and Applications, Villach, Austria,* pp. 48-57, August 2000

[5] J. Riihijärvi, P., Mähönen, M.Saaranen, J. Roivainen, J. Soininen: "Providing Network Connectivity for Small Appliances: A Functionally Minimized Embedded Web Server", *IEEE Communications Magazine*, Vol 39, No. 10, p. 74-79, Oct 2001

[6] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg: "SIP: Session Initiation Protocol", IETF RFC 2543, June 2002

[7] St. Moyer, D. Marples, S. Tsang: "A protocol for Wide-Area Secure Networked Appliance Communication", *IEEE Communications Magazine*, Vol. 39, No. 10, p. 52-59, Oct 2001

[8] D. Boggs, J. Mogul, C. Kent "Measured Capacity of an Ethernet: Myths and Reality", *Proceedings of SIGCOM'88 Symposium: Communications Architectures and Protocols*, pp.222-234, 1988

[9] IEEE P802.3af, DTE Power via MDI Task Force, http://grouper.ieee.org/groups/802/3/af/ index.html

Apostolos Leventis was born in Athens, Greece in 1974. He received the Electrical Engineering Diploma degree in 1997 from the Department of Electrical and Computer Engineering at the University of Patras, Greece. In November 1997, he joined the Laboratory of Electromagnetics at the University of Patras participating in various R&D projects for the Greek Government, the European Union and private companies. His research interests are in the area of embedded systems design with emphasis in communications protocols development. Mr. Leventis participates as a research engineer in the Computer Technology Institute and is a member of the Technical Chamber of Greece.

Theodore Antonakopoulos was born in Patras, Greece in 1962. He received the Electrical Engineering Diploma degree in 1985, and his Ph.D. degree in 1989 from the Department of Electrical Engineering at the University of Patras, Patras, Greece. In September 1985, he joined the Laboratory of Electromagnetics at the University of Patras participating in various R&D projects for the Greek Government and the European Union, initially as a research staff member and subsequently as the senior researcher of the Communications Group. Since 1991 he has been on the faculty of the Electrical Engineering Department at the University of Patras, where he is currently an Associate Professor. His research interests are in the areas of data communication networks, LANs and wireless-networks, with emphasis on performance analysis, efficient hardware implementation and rapid prototyping. He has more than 80 publications in the above areas. Dr Antonakopoulos is a Senior member of the IEEE, and is a member of the Technical Chamber of Greece.

Chris Stavroulopoulos was born in Patras, Greece in 1965. He received the Electrical Engineering Diploma degree in 1989 from the Department of Electrical and Computer Engineering at the University of Patras, Greece. In 1990 he joined the Laboratory of Electromagnetics at the University of Patras participating in many R&D projects supported by the EU (ESPRIT, RACE, ACTS), the Greek Telecommunications Organization (OTE) and industrial companies. His research interests are in the areas of Data Communication Systems and Wireless Computer Networks. He has published three papers in international conferences and one in a scientific journal. Mr. Stavroulopoulos participates as a research engineer in the Computer Technology Institute and is a member of the Technical Chamber of Greece.

Thomas Luckenbach was born in Berlin, Germany, in 1956.He started studies of mathematics and computer science in April 1976, at the Technical University of Berlin.He graduated in 1981 at the TUB with a diploma degree in informatics and started work at the TUB as an assistant professor. Till 1987 he was working in various projects in the area of open communication systems and giving numerous lectures on X.25, ISDN, B-ISDN, and Open Systems Interconnection. In 1988, he joined the Fraunhofer Institut for Open Communication Systems, FOKUS (formerly GMD). He is working there as a research group leader on the development of new networking technologies for body, personal and home area networks including vehicular environments and ad hoc networks. He received his doctorate degree in computer science from the Technical University of Berlin in 1993 for the design and realization of a modular broadband ISDN gateway system. His current interests focus on seamless networking infrastructures for ambient intelligence.

Vassilios Makios was born in Kavala, Greece. He received his Electrical Engineering degree (Dipl.Ing) from the Technical University in Munich, Germany in 1962 and his Ph.D degree (Dr. Ing.) from the Max Planck Institute for Plasmaphysics and the Technical University in Munich in 1966. From 1962-67 he was a Research Associate in the Max Planck Institute for Plasmaphysics in Munich. He served as Assistant Professor in 1967-70, Associate Professor in 1970-73 and Full Professor in 1973-77 in the Department of Electronics, Carleton University of Ottawa, Canada, where he was involved in microwave communications, remote sensing and CO_2 laser development. From 1977 he is an honorary Research Professor of Carleton University. Since 1976 he has been Professor of Engineering and Director of Electromagnetics Laboratory in the Electrical Engineering Department of the University of Patras Greece. He has published over 140 papers and holds numerous patents in the above fields. He is the recipient of the silver medal of the German Electrical Engineering Society (VDE). He is a Senior member of the IEEE, member of the Canadian Association of Physicist, the German Physical Society and the VDE, Profession Engineer of the Province of Ontario and member of the Greek Technical Chamber.