Journal of Theoretical and Applied Information Technology
30th September 2011. Vol. 31 No.2

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

MINI TCP/IP FOR 8-BIT CONTROLLERS

R.MANIKANDAN
Assistant Professor, School of Computing , SASTRA University,Thanjavur-613401,India
E-mail: manikandan75@core.sastra.edu

ABSTRACT

Internet introduced sweeping changes in communication and information Technology. Information super highway linked every nook and corner of this world and flow of information brought down and almost eliminated the time spent in searching and sifting information. The benefits are endless. In recent years the interest of connecting small embedded devices to the IP networks has increased. In order to communicate over Internet the TCP/IP protocol stack is required. Unfortunately, for most electronic devices, implementing the technology to achieve this networking connectivity based on open Internet standards does not come easily. For example, most household appliances are based on very low-cost 8-bit controller technology and chances are that the host MCU includes neither a network port nor the hardware resources to support TCP/IP and other Internet protocols without disrupting their primary function. So, the normal TCP/IP way of implementation will fail. Here we have to incorporate certain optimization techniques for developing small TCP/IP stack. The usage of this stack is for Remote data acquisition and control systems, and many more utilities. [1],[2],[4]

Keywords: *TCP, IP, FITO buffer, RAM, Mini TCP/IP*

1. INTRODUCTION

With the success of the Internet , the TCP/IP protocol suite has become a global standard for communication. TCP/IP underlying protocol used for web page transfers, e-mail transmissions, file transfers, and peer-to-peer networking over the Internet. For embedded systems, being able to run native TCP/IP makes it possible to connect the system directly to an intranet or even the global Internet. Embedded devices with mini TCP/IP support will be first-class network citizens, thus being able to fully communicate with other hosts in the network. Traditional TCP/IP implementations have required far too much resources both in terms of code size and memory usage to be useful in small 8 or 16-bit Systems. Code size of a few hundred kilobytes and RAM requirements of several hundreds of kilobytes have made it impossible to fit the mini TCP/IP stack into systems with a few tens of kilobytes of RAM and room for less 100 kilobytes of code.

TCP is both the most complex and the most widely used of the transport protocols in the TCP/IP stack. TCP provides reliable full-duplex byte stream transmission on top of the best-effort IP layer. Because IP May reorder or drop packets between the sender and receiver, TCP has to implement sequence numbering and retrasmissions in order to achieve reliable, ordered data transfer. We have studied how the code size and RAM usage of a mini TCP/IP implementation affect the features of the TCP/IP implementation and the performance of the communication. We have limited our work to studying the implementation of TCP and IP protocols and the interaction between the mini TCP/IP stack and the application programs. Aspects such as address configuration, Security, and energy consumption are out of the scope of this work.[1],[2],[3]

The main contribution of our work is that we have shown that is possible to implement a mini TCP/IP stack that is small enough in terms of code size and memory usage to be useful even in limited 8-bit systems. Furthermore, Existing TCP/IP implementations for small systems assume that the embedded device always will communicate with a full-scale TCP./IP implementation running on a workstation-class machine. Under this assumption, it is possible to remove certain TCP/IP mechanisms that are very rarely used in such situations. Many of those mechanisms are essential, however, if the embedded device is to communicate with another equally limited device, e.g., when running distributed peer-to-peer services and protocols.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. TCP/IP OVERVIEW

From a high level view point, the TCP/IP stack can be seen as black box that takes incoming packets, and de-multiplexers them between the currently active connections. Before the date is delivered to the application, TCP sorts the packets so that they appear in the order they were sent. The TCP/IP stack will also send acknowledgements for the received packets. Data arrives asynchronously form both the network and the application, and the TCP/IP stack maintains queues in which packets are kept waiting for service. Because packets might be dropped or reordered by the network, incoming packets may arrive out of order. Such packets have be queued by the TCP/IP stack until a packet that fills the gap arrives. Furthermore, because TCP limits the rate at which data that might not be immediately sent out onto the network.[1],[2]

The mini TCP/IP suite consists of numerous protocols, ranging from low level protocols such as ARP which translates IP addresses to MAC addresses, to application level protocols such as SMTP that is used to transfer e-mail. We have concentrated our work on the TCP and IP protocols and will refer to upper layer protocols as "the application". Lower layer protocols are often implemented in hardware and will be referred to as " the network device" that are controlled by the network device driver. TCP provides a reliable byte stream to the upper layer protocols. It breaks the byte stream into appropriately sized segments and each segment is sent in its own IP packet. The IP packets are sent out on the network by the network device driver. If the destination is not on the physically connected network, the IP packet is forwarded onto another network by a router that is situated between the two networks. If the maximum packet size of the other network is smaller than the size of the IP packet, the packet is fragmented into smaller packets by the router. If possible, the size of the TCP segments are chosen so that fragmentation is minimized. The final recipient of the packet will have to reassemble any fragmented IP packets before they can be passed to higher layers.[1],[2].

3. RELATED WORK

The problem at hand was analysed in detail before implementing the stack. In various industrial applications where control and instrumentation is applied, there is a necessity of measuring the field point values of a changing physical parameter; for example measurement of temperature in a chemical plant where the contents in a bath is mixed at a constant temperature and pressure for a specified output. Monitoring temperature in olden days is done at the field point itself. But now there arises the need for monitoring and controlling field point values from a single control room. If the distance between the field point and control room is short, standard techniques like serial communication can be applied for measuring and controlling data. But if the distance is large these methods cannot be applied and thus one has to device alternate methods for this communication. The various possibilities can be [3],[5],[6]

1.Using simple RF communication at an unlicensed frequency. 2.Using GPS systems. 3.Using telephonic systems with the manipulation of analog data. 4. Using TCP/IP and Internet infrastructure.

Let us analyse these methods one by one. First, if we use the simple RF communication for transportation and controlling the data we will face the problem of the frequency to be used for modulation. Using an unlicensed frequency will definitely cause interference and there may be a permanent loss of data.In the second method using GPS systems we should necessarily have a GPS network. If it is not there then the proposed method will not work. If there is a GPS network, then the hardware involved for doing this communication will be very high and it will overshoot the sophistication of the problem.

The method of telephonic systems can be used effectively if remote data acquisition is not required. For example, a server in some other country wants to utilize the data already collected. The final method is more suited because the Internet infrastructure is well established and the hardware involved for this communication is very cheap. All that we have to do is to write a minimal version of the TCP/IP protocol which can run very well in a single 8-bit microcontroller.[2][4][5]

4. ARCHITECTURE AND FUNCTIONALITY

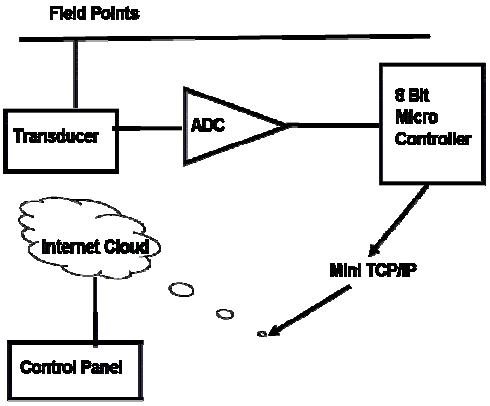
Mini TCP/IP can be seen as a code library that provides certain functions to the system. The figure III shows the relations between mini TCP/IP the underlying system and the application program. Mini TCP/IP provides three function to the underlying system, ip_init(), ip_input() and ip_periodic(). The application must provide a call back function to mini TCP/IP. The call back function is called when network or timer events

Journal of Theoretical and Applied Information Technology
30th September 2011. Vol. 31 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

occur. Mini TCP/IP provides the application with a number of function for interacting with the stack. Note that most of the functions provided by mini TCP/IP is implemented as C Language macros for speed, code size efficiency, and stack usage reasons.[1][2][3]

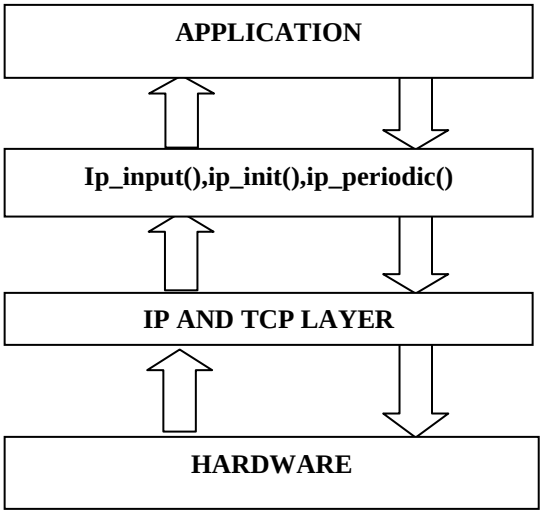
Figure-I:



The our proposed architecture is shown in the Figure-I.

The BSD socket interfaces used in most operating systems is not suitable for small systems. Since it forces a thread based programming model on the application programmer. A multithreaded environment is significantly more expensive to run only because of the increased code complexity involved in thread management, but also because of the extra memory needed for keeping per-thread state. The execution time overhead in task switching also contributes to this. Small systems may not have enough resources to implement such a multithread environment, and therefore an application interface, which requires this, would not be suitable for mini TCP/IP. Instead, mini TCP/IP uses an event based programming model where the application is implemented as a C language function that is called by mini TCP/IP in response to certain events.[2][3][4]

Figure-II:



Backbone functions in Mini TCP/IP Stack

Mini TCP/IP calls the application when data is received, when data has been successfully delivered to the other end of the connection, when a new connection has been setup, or when data has to be retransmitted. The application is also periodically polled for new data. The application program provides only one call back function; it is up to the application to deal with mapping different network services to different ports and connection.

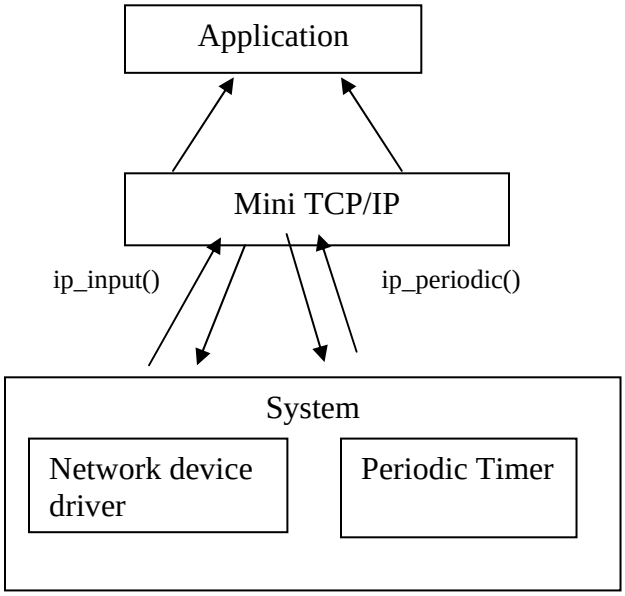
5. CODE SIZE REDUCTION

The 8-bit controllers have a restriction on the program memory and data memory, implementation of the above issues takes precedence. In realising the proposed mini TCP/IP, we have to incorporate the board support package, the device driver for the physical layer, the IP and TCP protocols and the application layer. Considering an Intel 8051 microcontroller, the memory available (64K data and 64K program memory) is less. Thus lot of optimization has to be done in memory usage. The features of the implementation includes Very small code size, Low memory usage, Implementation of full fledged IP and TCP Layer, Single threaded Application and No operating system is involved.[1],[2],[4],[5]

Journal of Theoretical and Applied Information Technology
30th September 2011. Vol. 31 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure III:



6. MEMORY MANAGEMENT AND BUFFER MANAGEMENT

Getting more help from the application layer thereby reducing the overhead for the stack. By retransmitting the data from the application layer and not keeping it in the buffer. This retransmission of the data involves only the application layer and not the stack. This reduces the source code and the memory usage to a maximum extent.
FITO buffer has been used for the implementation. It has one IN and two OUT pointers. Two OUT pointers are used to keep track of the acknowledgement signal.[2],[3]

7. CONCLUSION

A mini TCP/IP has been developed to meet the ever increasing demands of connecting small embedded devices to the Internet. It is highly portable across various platforms and is intended for single threaded applications.

8. FUTURE WORK

Security aspects: When computing systems to a global Internet the security of the system is very important. Identifying levels of security and mechanisms for implementing security for embedded devices is crucial for Connecting systems to the global Internet.

Performance Enhancing Proxy: It might be possible to increase the performance of communication with the embedded devices through the use of a proxy situated near the devices. Such a proxy would have more memory than the devices and could assume responsibility for buffering data.

REFERENCES

[1]. The Protocols (TCP/IP Illustrated, Volume I) by Richard Stevens, Addison-Wesley, 1994.

[2]. The Protocols (TCP/IP Illustrated, Volume II) by Richard Stevens, Addison-Wesley, 1995.

[3]. TCP/IP Application layer protocols for Embedded Systems by M.Tim Jones, Firewall Media,2003

[4]. Programming and Customizing the 8051 Microcontroller by Myke Predko, McGraw-Hill, 1998.

[5]. The 8051 microcontroller Architecture programming &Applications (2nd edition) by Kenneth J.Ayala Penram International

[6]. The 8051 Microcontroller and Embedded Systems by Muhammad Ali Mazidi Janice Gillispie Mazidi, Prentice-Hall India