IMPLEMENTATION OF TCP/IP STACK FOR 3L DIAMOND OPERATING SYSTEM

Marek Kváš
Doctoral Degree Programme (3), FEEC BUT
E-mail: marek.kvas@phd.feec.vutbr.cz

Petr Petyovský
Doctoral Degree Programme (7), FEEC BUT
E-mail: petyovsk@feec.vutbr.cz

Supervised by: Soběslav Valach
E-mail: valach@feec.vutbr.cz

ABSTRACT

This paper deals with implementation of TCP/IP stack for 3L Diamond operating system. After short introduction to multiprocessor operating system 3L Diamond, main implementation issues are described. Digital signal processor module based on TMS320C6455 is used as a target platform.

1. INTRODUCTION

Processor based digital systems are used in the broad spectrum of human activities. Applications impose many requirements on digital systems that are very often in conflict with each other. Computational capacity and low power consumption are typical example. One of the usual ways how to fulfill this kind of requirements is building system with multiple processors. It can be very advantageous to use general purpose processors (GPP) for user interface, digital signal processors for these parts of system that have to process signals in real-time and field programmable gate arrays (FPGA) for glue logic, communication interfaces or computational accelerators implementation. Replace such combination of technologies by only one more general platform would be very difficult and probably less efficient.

Parallel nature of such systems brings some problems also. Software development for multiprocessor systems is more complex than for single processor. This is the reason why the market with development tools for multiprocessor systems is growing too. 3L Diamond is one of these tools (tool suit and operating system).

Connecting devices through computer networks especially Internet is another trend in modern electronic. Data acquisition, remote control, distribution of computation for higher reliability or performance can be reasons for connecting devices to the network. Most often used technology is probably Ethernet based network in combination with TCP/IP protocol family. TCP/IP stack implementation for Diamond OS is main objective of this work.

2. TOOL-SUIT AND OPERATING SYSTEM DIAMOND

Diamond is a set of tools and operating system for multiprocessor system development [1]. It allows developer to separate implementation of application into two largely independent sections. The first one is development of SW as set of tasks that communicate through channels. Channels are only one way for communication between tasks that is allowed. They are also only way how tasks can be synchronized. SW to HW mapping is the section one. Every task has to be assigned to any processor in the system.

Very important tool from Diamond tool-suit is configurer. This tool creates final application from hardware description and compiled tasks. Hardware description provides configurer with information about types of available processors and connections between them. According to this information and task to processor assignment configurer adds to the application microkernels that implement system services and channels. This brings big flexibility to the application development. Topology and task to processor assignment can be change without any coding or even necessity to recompile tasks. This enables developer to experiment quickly, develop application on different HW than the final one (because it is under development, for example) or easily migrate to more powerful HW.

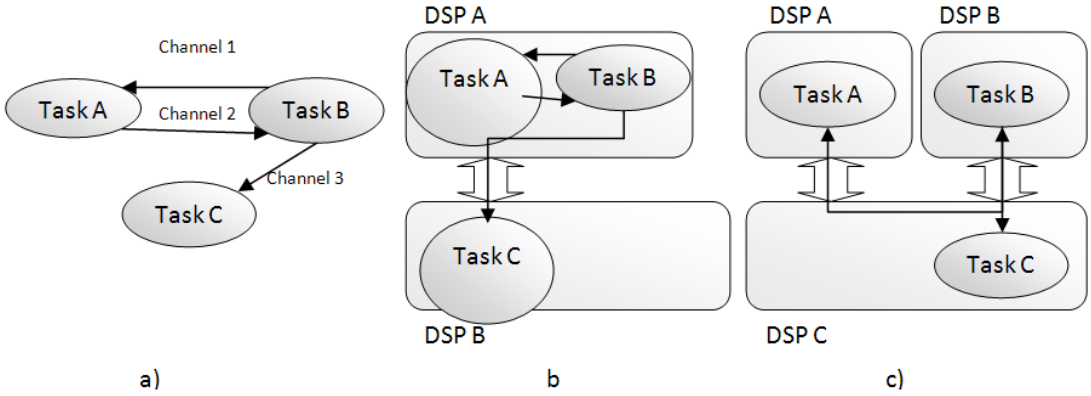

Fig. 1 Task to processor assignment: a) SW model of application. b) Mapping to two processors. c) Mapping to three processors.

Figure 1 demonstrates development process. Part (a) shows software model of application. Application consists of three independent tasks (A, B, C) that communicate with each other through channels (1, 2, 3). In this development stage actual implementation of channels or system topology is not important. Part (b) shows implementation example. Tasks are mapped into two signal processors (DSP A and B) connected through buses. Tasks A and B share processor DSP A and task C is placed in DSP C. Configurer adds to both DSPs such system modules that implement channel 3 as communication through physical bus and channels 1 and 2 as shared memory in ideal case.

Part (c) shows similar system with one more DSP. We can see that there is no physical connection between DSP A and B. Even in this case configurer automatically finds the way how to implement all channels. Data are transferred through DSP C. Because of standard channel interface this connection is completely transparent for tasks. Task A and B do not have to share processor time of DSP A and system can work more efficiently.

Every task can consist of several threads. All threads of one task are always placed on the same processor and share one address space. Threads do not have to communicate through channels only, but they can use shared memory also. For threads channels are not only synchronization object. They can use events, signals and semaphores also. Their functionality is similar to other operating systems.

Eclipse based integrated development environment is also part of diamond. Open API for communication with system through PCI bus and tool for system boot are provided by application called Diamond server.

3. TCP/IP STACK IMPLEMENTATION

TCP/IP protocol family is a base of worldwide network Internet. Description of most important internet protocols can be found in book [2] for example or as RFC (requests for comments) in [3].

Figure 2 shows comparison between ISO OSI model on part (a), TCP/IP model on part (b), assignment of actual protocols to layers in part (c) and model of stack for diamond on part (d).

Target HW for implementation is a multiprocessor module SMT362 [4] developed by Sundance company. There are two DSPs TMS320C6455 and FPGA Xilinx Virtex IV placed on this board. Module can be connected to PC via PCI bus using carrier board (e.g. SMT300Q). This connection makes SW development more comfortable. Module can be used either as PCI card or standalone module.

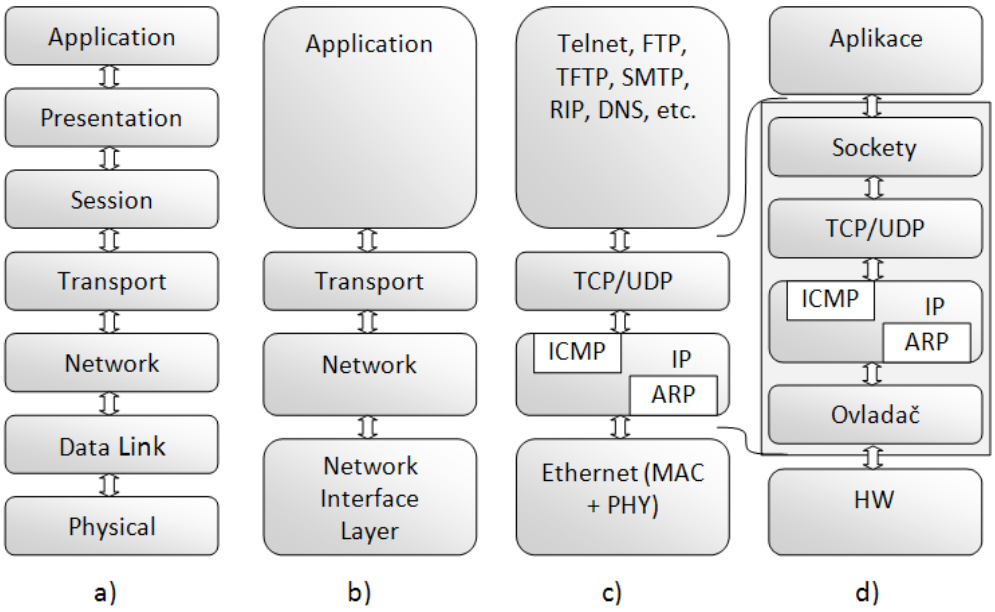

Fig. 2 a) ISO OSI model b) TCP/IP c) TCP/IP protocols d) Diamond TCP/IP stack

Network access layer is based on Ethernet in our case. This layer can be divided into MAC (Media Access Control) and PHY (Physical Layer). MAC is implemented on TMS320C6455 chip as EMAC unit. This unit is designed to support up to 1Gb/s ethernet. All other necessary circuits are placed on the SMT362 module.

Our project implements all layers between HW and application layer (figure 2d). There exist a lot of implementations for number of different platforms. Many of them are based on BSD (Berkeley Software Distribution) implementation that is open source with very tolerant license. We based our project on one of the available versions too.

The biggest advantage of this approach is that stack is well tested (used and being improved during long period). On the other hand, different target platform can be big disadvantage. Some architecture differences required significant changes in code. Main issues that had to be solved were memory management and process synchronization.

Original BSD implementation was designed for platforms equipped with MMU (Memory Management Unit). Using MMU can prevent system from dangerous memory fragmentation caused by dynamic allocation. Target HW does not have the MMU. Allocation from static memory pools was chosen as a solution. Buffers of two sizes are used for data exchange between layers, 128 B and 2 KB. Data are represented as chains of these small buffers. That allows easy manipulation like adding headers, fragmentation, reordering and so on. If we consider throughput of stack (>100 MB/s) we can see that dynamic allocation would be very inefficient.

Synchronization is much harder problem to solve. The stack consists of several threads running in the environment of preemptive multitasking. Stack threads can be divided into three groups. In the original BSD implementation priorities are used for synchronization.

Group with highest priority consists of driver threads. Driver has to respond on interrupts from HW to assure enough memory buffers for receiving and continuous feeding transmitter with data.

Protocol threads inside the stack are the next group. This one operates on middle priority, so it can be interrupted by the driver. These threads compose packets from data. That means mainly adding headers and computing checksums.

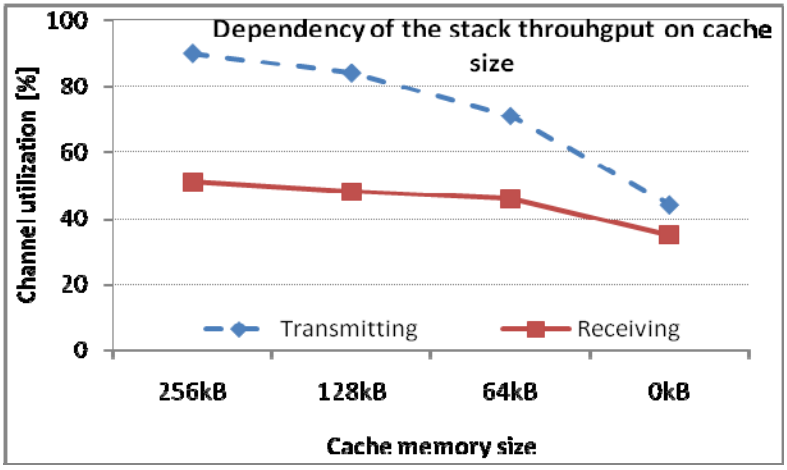

Figure 1. Dependency of the stack throughput on cache size

Group with the lowest priority consists of application layer threads and interface between application and the stack (socket layer).

Another difference between BSD environment and Diamond OS is that stack under BSD was part of kernel. The stack in diamond operates on the same level as applications. This together with different priority system and scheduling makes synchronization using priorities impossible. Synchronization has to be implemented using available synchronization objects instead.

Driver routines that service HW events runs on the highest priority and cannot be descheduled for any reason. They have to pass control to other processes on their own. Other

threads run on equal priority. Data consistency of shared structures is protected by critical sections. Most demanding task is to identify places that need protection and design protection in such way that deadlocks are avoided and processor time is not wasted for useless synchronization.

As mentioned above the stack works with memory very intensively. Hi-speed communication requires huge buffers that have to be placed in slow external memory. Therefore cache memory subsystem has big influence on stack throughput. Figure 3 documents this dependency. Data are transferred using DMA from and to HW. This together with using cache memory brings also problems with flushing cache and alignment of data in memory. As mentioned above buffers are allocated from static memory pools. That makes memory alignment problems easily solvable.

4. CONCLUSION

Development of this TCP/IP stack has not been finished yet. Development is in the stage when the stack is fully functional and it is being tested. There are performance data for TCP and UDP transfers in the table 2. Performance data are strongly dependent on parameters of communication subsystem of testing PC and network utilization. TCP/IP stack that has been implemented is currently offered as a component of OS Diamond.

Direction	PC -> DSP		DSP -> PC	
Protocol	Channel utilization [%]	Throughput [MB/s]	Channel utilization [%]	Throughput [MB/s]
TCP	90	112	51	64
UDP	80	100	80	100

Table 2. Diamond TCP/IP stack performance

ACKNOWLEDGEMENT

This work has been supported in part by Ministry of Education, Youth and Sports of the Czech Republic (Research Intent MSM0021630529 Intelligent systems in automation), Grant Agency of the Czech Republic (102/09/H081 SYNERGY - Mobile Sensoric Systems and Network) and by Brno University of Technology.

LITERATURA

[1] Websites of 3L company [online]. [2009-02-10]. Available on: <http://www.3l.com/what-is-3l-diamond>

[2] STEVENS, W. R.: TCP/IP Illustrated, Addison-Wesley 1994, ISBN 0-201-63346-9.

[3] Documents RFC [online]. [2009-02-10]. Available on: <http://rfc-ref.org/>

[4] Sundance SMT362 module datasheet [online]. [2009-02-10]. Available on: <http://sundance.com/web/files/productpage.asp?STRFilter=SMT362>