Data-Driven Implementation of Protocol Handling for Supporting Ad Hoc and Ubiquitous Networking Environment

Kazuhiro Aoki∗ Hiroshi Ishii† Hiroaki Nishikawa‡
∗ Information Infrastructure Laboratory, Inc.
Saiki 210-4, Tsukuba, Ibaraki 300-0028, Japan
† School of Information Science and Technology, Tokai University
Kitakaname 1117, Hiratsuka, Kanagawa 259-1292, Japan
‡ Institute of Information Sciences and Electronics, University of Tsukuba
Ten’nodai 1-1-1, Tsukuba, Ibaraki 305-8573, Japan

Abstract
Pervasive networking environment as social infrastructure has been going to become a reality, especially through the recent considerable progress of semiconductor technology. Considering the case where next generation of pervasive networking is realized over ad hoc network suitable for emergency and some tentative event, a project named “Ad hoc and Ubiquitous Communication Environment” has begun. It is very important to implement effective protocol handling and routing process in Ad hoc and ubiquitous networking environment especially. This paper describes the effectiveness of an implementation of protocol handling using networking-oriented data-driven processors CUE (Coordinating User’s requirements and Engineering constraints). In Ad hoc network, it is necessary to realize not only network protocol but also routing protocol such as DSR(dynamic source routing) protocol. Firstly, this paper discusses data-driven implementation of protocol handling on CUE networking board. Then this paper describes architecture of CUE-v2 which is available version in CUE processors. Furthermore, this paper proposes protocol handling off-loader which is implemented on the CUE-v2 networking board to minimize overheads in protocol handling. This paper finally discusses effectiveness of protocol handling off-loader using CUE processor system. it describes that client such as PC uses performance to process application which are video/audio processing without overhead in network communication.

Keywords: Ad hoc network, ubiquitous network, protocol handling, data-driven, off-loader

1 Introduction

So-called pervasive networking environment as social infrastructure has been widely studied[1]. And ubiquitous service is under realization, which enables us to access and utilize versatile computing and network resources and information wherever and whenever we are. Furthermore, there are many studies about mobile ad hoc network[2] which is an infrastructureless network and is a group of wireless devices that organize themselves in a mesh topology to find routes and relay packets from the hardware platform through the network layer to application. Considering the case where next generation of pervasive networking is realized over ad hoc network suitable for emergency and some tentative event, a project named “Ad hoc and Ubiquitous Communication Environment” has begun[3].

Then it is important how to design network processing which are protocol handling, routing, secure processing and so on in Ad hoc and ubiquitous communication environment because they can be overheads to application processing and be bottlenecks in communication. At the same time, as such an environment has to process multi-media stream in real-time, effective real-time multi-processing capability is essential for the processor to execute some processes with various granularity; media processing has inherent fine-grain parallelism in it.
and protocol processing has inherent coarse-grain parallelism in it.

On the other hand, user’s requirements will come more diversified and complex, while time to market will come shorter. It is the reason why effective prototyping environment is essential to support developing such systems[4].

The authors have been developed a super-integrated dynamic data-driven processor CUE-p (Coordinating Users’ requirements and Engineering constraints - prototype), CUE-v1 (CUE-version 1) and CUE-v2 (CUE-version 2)[5]-[7]. CUE processors can simultaneously execute parallel processing with various granularity such as instruction level, process level, processor level and so on. It has been reported that CUE processors can execute ideal multi-processing without runtime overheads like priority control mechanisms in conventional operating systems[8], [9]. One of the authors also has studied data-driven implementation of audio and movie compression[10]. The authors have also studied a prototyping environment RESCUE (Real-Time Execution System for CUE-series data-driven processors)[11]-[14] as a development environment to support implementing real-time systems using CUE processors.

This paper discusses the effectiveness of an implementation of network and routing protocol handling using networking-oriented data-driven processors CUE (Coordinating User’s requirements and Engineering constraints). In Ad hoc network, it is necessary to realize not only network protocol but also routing protocol such as DSR(dynamic source routing) protocol[15]. Firstly, this paper discusses data-driven implementation of protocol handling on CUE-v2. Then this paper describes architecture of CUE-v2 which is available version in CUE processors. This paper also shows an implementation of IP (Internet Protocol) on CUE-v2. Furthermore, this paper proposes protocol handling off-loader which is implemented using CUE-v2. Since data-driven implementation of protocol handling minimize overheads in protocol handling, client such as PC can achieve high throughput in application processing using data-driven protocol off-loader. This paper finally discusses effectiveness of protocol handling off-loader using CUE processor system. it describes that client such as PC uses performance to process application which are video/audio processing without overhead in network communication. Further works follow.

2 Data-Driven Implementation of Protocol Handling

2.1 A Networking Oriented Data-Driven Processor: CUE-v2

To create a streamlined architecture for future multimedia networking, we have studied a multimedia networking oriented data-driven processor. Data-driven architectures can naturally and efficiently exploit maximum and various levels of parallelism and spread it among processing elements and into their pipeline stages Furthermore, data-driven architectures have benefits for architectural level real-time processing[9] because they offer constant execution time of each process and deterministic guarantees for packet rate due to their fair multiprocessing at instruction level without context switching overheads. This means that data-driven architectures can perform real-time processing without any runtime scheduling overheads such as realtime OSs assuming the deterministic input rate of a network node, which is, for example, wire rate.

Also, we have evaluated CORBA(Common Object Request Broker Architecture) protocol offloading over OC-3 ATM(Asynchronous Transfer Mode) and realtime video compression us-
ing the CUE-v1 processors built in a 0.25µm CMOS process[5]. These studies demonstrated that the CUE-v1 achieved the realtime processing of actual applications for multimedia networking environment as we had expected.

These studies also gave us several issues to streamline the execution of the applications for multimedia networking. One of the issues is the inevitable inefficiency on serial codes, such as connection/port management in TCP and the serialization of parameters in video compression. Since data-driven architectures cannot exploit the locality of computation, it is not good at sequential processing. This is the compensation of exploiting fine-grained parallelism. To alleviate this issue with retaining the advantages of pure data-driven, we proposed an architecture which can simultaneously process data-driven and control-driven threads using common pipeline resource. The CUE-v2 performs both as data-driven and as out-of-order superscalar in order to alleviate the bottlenecks caused by sequential processing.

That is, a data-driven processor has benefits for realtime processing, such as instruction-level parallel processing without context switching overheads and fair resource scheduling among multiple threads. However, the following problems of data-driven architecture was pointed out in previous researchers:

1. the potential inefficiency on serial codes,
2. the overhead of firing control, and
3. low packet distribution/input rate.

Today’s VLSI technology can solve the problem 2. and 3. by integrating a large CAM (Content Addressable Memory) and non-blocking switches, respectively. But the VLSI progress cannot solve the problem 1. This is because data-driven principle ignores the locality of computation. That is, given N as the pipeline depth of a circular pipeline of a pure data-driven processor and p as the parallelism in a program, CPI (Cycle Per Instruction) corresponds to \( \max(1, N/p) \). Consequently, the instruction fetch rate or the front-end bandwidth of a data-driven processor is determined by the parallelism. Therefore the CPI of a serial code corresponds to N in data-driven architecture.

To address this problem, we proposed processor architecture for simultaneously processing data-driven and control-flow threads in a single pipeline. One kind, denoted “data-driven” is appropriate for the highly parallel parts; the other, “control-flow” is for the parts of the code with little parallelism. Unlike conventional data-drive/von Neumann hybrid architectures, exclusive execution of the control-flow thread is not performed in our proposed architecture. That is, the data-driven thread has priority to the control-flow thread in instruction fetching time. But, minimal issue opportunity is given to the control-flow thread to avoid blocking. Our architecture basically allocates empty slots, caused by the execution of the data-driven thread like in the failure of waiting-matching operation, to the control-flow thread. In addition, our architecture employs the forwarding paths for the control-flow thread for the speedup on the execution of dependent instructions. To do this, we introduce the extended the firing control unit, which also functions as a extended the firing control unit, which also functions as a reservation station in out-of-order superscalar processors.

The prototype chip was developed by employing standard-cell design, and it was implemented using timing-driven synthesis/layout. Crosstalk, antenna effect, and voltage drop were analyzed and validated using commercial EDA tools. The chip is built in a generic 0.18µm six-metal layer process, the die size of 5×5 mm², including 64kbyte SRAM, and is packaged in a ball grid array having
Checksum is sum of all header field values. Summation is realized using control-flow.

Checking header without "Checksum" are implemented using data-flow.

Figure 1: Data-Driven Implementation of IP Receiving

292pins. The chip is verified with several applications. It proves to work without any significant flaws in this chip. Most of its functional verification time was spent for an out-of-order scheduling of control-flow. The verification for basic operations and the simultaneous processing of data-flow and control-flow threads was not dominant. Thus, the CUE-v2 architecture is not more complex than superscalar processors. The CUE-v2 chip will be installed to PCI board[5], which was originally designed in the CUE project, to be examined for its potential capability as chip multi-processor core in the next generation CUE-v3.

2.2 An Implementation of Protocol Handling on CUE-v2

The authors have been studied data-driven implementation of protocol handling in CORBA to realize realtime multi-processing in networking environment. CORBA consists of inter-object protocol which are based on TCP/IP. These protocol handlers are implemented sequentially by using memory on sequential processors.

We have studied data-driven implementation of TCP/IP handling in parallel[8]. On the other hand, CUE-v2 is implemented as hybrid architecture because of optimizing sequential process in protocol handling as shown in section 2.1. So, we studied an implementation of protocol handling which are IP and DSR (Dynamic Source Routing) on CUE-v2.

IP handling consists of receiving process which sends data to upper layer and sending process which sends data and acknowledgments to the network in response to the data sent from the application.

Firstly, IP receiving process receives IP data-gram, initial signal and received interface indicator. Data-driven implementation treats IP datagram as a row of packet. A packet has 1 byte data. The order of packets is kept by generation. In IP receiving process, IP header check and buffering datagram are handled simultaneously. And IP receiving process outputs address of buffering data, IP address, data length and checksum.

Then, IP sending process generates IP header from pseudo header, and outputs frames or PDU(Protocol Data Unit) which is an IP data-gram, data length and trigger for network interface. Data-driven implementation realizes header check/generate concurrently because these can be independently processed each other except checksum. Checksum is realised using control-flow threads in CUE-v2 because checksum is sequential process as show in Fig.1.

In Ad hoc communication environment, it is necessary to implement routing protocol. we selected DSR as the routing protocol in our approach. Since DSR allows the discovery of multiple paths, it is suitable to apply DSR to our communication environment. We have studied extended version of DSR and data-driven implementation of DSR.

3 Data-Driven Protocol Off-loader

In this section, we propose data-driven propose off-loader to realize Ad hoc and ubiquitous com-
Communication environment. We have studied data-driven implementation of network protocol handling. It shows that data-driven implementation realizes real-time multi-processing without runtime overheads. So, we will apply data-driven protocol off-loader which handle network/routing protocol on CUE-v2 to client which process some applications to achieve high throughput in protocol handling and application processing.

Fig.2 shows patterns of off-loading processes onto data-driven protocol off-loader. Fig.2(0) is conventional system which is not off-loading any processes. In Fig.2(1), network protocol handling such as IP is off-loaded onto data-driven protocol off-loader.

The data-driven protocol off-loader consists of CUE-v2 to handle protocols, network interface to establish network, and USB (Universal Serial Bus) controller in order to connect to a client PC. In experimental study, it isn’t suitable to add complicated control to communicate between client and data-driven protocol off-loader because it is difficult to compare performance to conventional system. Then, it is relatively easy to output data to USB when client connects to data-driven protocol off-loader using USB.

We expect to show effectiveness of off-loading network protocol handling which is implemented on CUE-v2 in Fig.2(1). Furthermore, we study about off-loading routing protocol and network protocol as shown in Fig.2(2). Routing protocol such as DSR can be overhead on client. Therefore, we think that off-loading routing/network protocol achieve higher throughput than off-loading just network protocol.

Then, Fig.2(3) shows off-loading rout-
ing/network protocol and a part of application process such as media codec or secure process. We have studied data-driven implementation of media processing[9], so we expect that it is suitable to implement media processing on CUE-v2. However, we must estimate if data-driven protocol off-loader has enough resource to off-load media processing, routing protocol and network protocol. We will evaluate it in experimental study.

4 Conclusion

This paper firstly describes the effectiveness of an implementation of protocol handling using CUE-v2 which is available version in CUE processors. Since CUE-v2 is hybrid architecture which has data-flow and control-flow threads, its implementation is basically parallel, but utilizing control-flow to sequential process such as checksum. In Ad hoc network, it is necessary to realize not only network protocol much also routing protocol such as DSR. So we have studied data-driven implementation of routing protocol, and it is one of further works.

Then, this paper proposes data-driven protocol off-loader which is implemented using CUE-v2. We have studied some patterns about off-loading processes. We expect higher effect in off-loading routing and network protocol, but we also think that another pattern of off-loading will have some merit. It is necessary to estimate how many resources is needed to implement routing protocol and media processing on CUE-v2. It is also one of further works. As another further works, we will show effectiveness of data-driven protocol off-loader in experimental study. we have implemented IP and DSR on CUE-v2 networking board. Further works also include implementing device driver facility of network PHY device on CUE processors.

Through these studies, the authors will contribute to establish novel approach to develop an effective multi-processor system for Ad hoc and ubiquitous communication environment as a future social infrastructure.

Acknowledgments

Although it is impossible to give credit individually to all those who organized and supported the CUE project, the authors would like to express his sincere appreciation to all the colleagues in the project. This research is partially supported by SCOPE (Strategic Information and Communications R&D Promotion Programme), Ministry of Internal Affairs and Communications, Japan, the Scientific Grant of Japan Society for the Promotion of Science and Semiconductor Technology Academic Research Center.

References


