RESEARCH AND IMPLEMENTATION OF MPLS VPN PROTOCOL BASED ON NETWORK PROCESSOR

Wang YongJun, Huang QingYuan
School of Computer, National University of Defense Technology, ChangSha, HuNan, P.R.China

Keywords: Network processor, protocol development, software framework, picocode, MPLS, VPN.

Abstract: MPLS VPN is one of popular protocols in next generation internet. In general, it would be implemented in modern routers. In this paper, the implementation technology of MPLS VPN was studied in high performance router based on network processor. The programming view of NPs is studied and a flexible protocol development software framework is proposed, which considers function partition of protocol into two parts for specific NP and general-purpose processor. Making use of properties of flexible programming and high processing capability of network processor, software architecture of MPLS VPN was proposed, the key technology was designed and implemented, which shows the efficiency of protocol extension and exploits the method to software upgrade of network processor.

1 MOTIVATION

With the rapid development of internet, electronic business activities have become more and more popular. Many enterprises allow the partners to access their private network in order to simplify and to speed up information exchange. However, because of the distribution and open nature of internet, such business activities are threatened by security problems. Hence, virtual private network technology is brought to tackle with such security threat.

Traditional layer-2 VPN gets poor scalability because of full connections between different sites. MPLS VPN (Ivan, 2001) which is created on layer-3 can get over the problem while its deployment and management are very easy. Thus the security and privacy of layer-2 VPN can be assured through only delivering VPN route information which is handed out with BGP to VPN member routers.

MPLS VPN can overcome the limitations of traditional VPN (Rosen,1999). It can reduce the construction cost of enterprises' private network through utilizing the powerful transportation ability of public backbone network effectively while notably improving employment and management flexibility of user network. At the same time, through the isolation of routing information between users and public network as well different users, MPLS VPN can meet user's requirement for security, real time, broad bandwidth, and convenience. It will be one of the core protocols of next generation internet.

Network processors(Stephen, 2000) (NPs) are an attempt by hardware vendors to fulfill the growing need for low-priced specialized network hardware that is more future proof than conventional custom hardware or ASIC-based designs, and can be applied in a wide range of situations (e.g. in networked devices, as edge network routers and even in the network core). NPs are multiprocessor-based hardware units that support a number of network ports and provide software based packet processing facilities that can be programmed with the aid of a toolkit. So Network Processors (NPs) are emerging as cost effective networking elements that can be more readily updated and evolved than custom hardware or ASIC-based designs with high performance.

The aim of the research discussed in this paper is to design and implement MPLS VPN protocol based on NP. To support it, a flexible protocol development software framework for NPs is proposed that accommodates complex architectures and architectural heterogeneity while also supporting high performance. In this framework, software architecture of MPLS VPN is studied and proposed, the key technology was designed and implemented.

YongJun W. and QingYuan H. (2006).
RESEARCH AND IMPLEMENTATION OF MPLS VPN PROTOCOL BASED ON NETWORK PROCESSOR.
In *Proceedings of WEBIST 2006 - Second International Conference on Web Information Systems and Technologies - Internet Technology / Web Interface and Applications*, pages 156-163
DOI: 10.5220/0001252401560163
Copyright © SciTePress

The remainder of the paper is structured as follows. In section II, we characterize NP architecture as a basis for arguing, and also survey a number of existing NP software development support. In section III, we present our approach-flexible development framework to programming NPs and show how this improves on existing approaches. Then, in section IV, we describe the research and implementation of MPLS VPN protocol in detail. Finally, in section V we offer our conclusions.

2 RELATED WORK

2.1 Network Processor Architecture

The challenge in NP design is to provide fast and flexible processing capabilities that enable a variety of functions on the data path yet keep the simplicity of programming similar to that of a General-Purpose Processor (GPP). The NP system architecture plays a significant role in this respect: such architectures are primarily designed according to a serial model (or pipeline) or a parallel model.

In the parallel model, each thread in an NP core receives a different packet and executes the entire data-path code for this packet. In the serial model, each NP core receives each packet and executes a different portion of the data-path code in a thread.

From a programming point of view, the serial model requires that the code be partitioned such that the work is evenly distributed. In the parallel model, given that the same code can be performed by any NP Core and packets are assigned to the next available thread in any NP Core, the work is inherently evenly distributed. The Intel IXP (Matthew etc, 2002) architecture is designed primarily on a serial model, whereas the IBM PowerNP (James etc, 2003) is designed on a parallel model. For realistic, changing traffic mixes, the serial model would require a dynamic repartitioning of the code to maintain performance, unlike the parallel model for which no partitioning takes place.

2.2 Support of NP Software Development

The provision of software development environments for different NPs is almost as diverse as NP hardware architecture.

In terms of proprietary software, we focus on programming models and development environments for the IXP1200 and the IBM PowerNP. Information on the software environments used by other NPs is unfortunately hard to obtain without signing non-disclosure agreements.

Intel's MicroACE (Intel, 2001) is targeted at the IXP1200 and other Intel IXA products. In this model, proxy-like software elements (called active computing elements or ACEs) on the IXP1200's StrongARM control processor are 'mirrored' by blocks of code (called microblocks) that run on microengines. When the programmer loads a StrongARM element, the corresponding microblock is transparently loaded onto a microengine as a side effect. IBM's NPTool (James etc, 2003) includes such related toolkits as NPScope ,NPSim, which are loose coupled.

Turning to research-derived programming environments, NetBind (Campbell, etc. 2002) provides the abstraction of a set of packet processing components that can be bound into a data path. NetBind goes beyond MicroACE in supporting flexible composition of microblocks, but it offers no abstraction over the NP's interconnects or over different sorts of processors.

Apart from the work discussed above, additional research has focused on creating toolsets for specific NPs such as C compilers, simulators, debuggers and benchmarkers; some of this work is described in (Wagner, 2001), (Memik, 2001).

Above work can be classified into two kinds. The first focuses on making tools more usable, which has good performance but lack of development efficiency without efficient integration of those tools. The other one focus on providing programming model that promote design portability and transferable programming skills , which has good software development view, but would make performance influenced. We consider that most emergent problem of current NPs software development is how to provide a flexible framework and integrated toolkit to programmers to make complexity lower, improve the programming efficiency and shorten time-to-market actually.

3 FLEXIBLE SOFTWARE FRAMEWORK OF PROTOCOL DEVELOPMENT

3.1 General Architecture of NP Software

The architecture of NP-based communication device architecture and software are very different from that of traditional one. A general architecture of network processor software is shown as figure 1, which is divided into two part: network processor picocode, control point software.

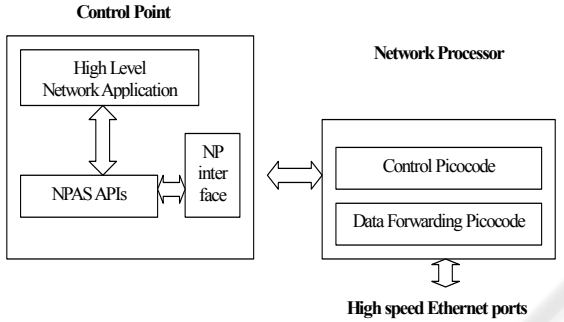

Figure 1: General Architecture of NP Software.

- Network processor Picocode

Embedded process chipset and co-processor of Network Processor core are responsible for executing Network processor picocode which is divided into two part: forwarding picocode and control picocode. Forwarding picocode is responsible for data plane process including packet classification, modification and forwarding. Complex packet process can be redirected to control point. Control picocode is responsible for control plane process including initialization of network processor, picocode downloading from control point, management of various classification tables and forwarding tables.

- Control Point (CP) Software

From NP's view, Control Point Software is composed of NPAS and network application.

- Network processor application service-NPAS

NPAS provides seamless interactions and application programming interfaces between control point software and network processor picocode, whose function involves: NP management, table management of kinds of protocols, traffic engineering management, redirection path of protocol processing, and so on.

- High Level Network Application

Except NPAS, other process running on control processor can be called high level network application , for example, TCP/IP, routing protocols and other signaling protocol . Network protocol processing would be initiated by registering its callback procedure to NPAS redirection path.

So, in general, complete processing of protocol includes two parts: the first is data plane on NP, the second is control plane on CP.

Figure 2 illustrates picocode view from programmer. Packet processing is divided into two stages: ingress and egress. Ingress refers to the data flow from the link towards the switch interface, and egress is the opposite. The same threads can perform processing at ingress or egress, thereby automatically balancing the processing power where needed. Along with the packet, additional context information can be transported from ingress to egress, such as the output port identifier of the egress NP obtained by the IP forwarding lookup previously executed on the ingress NP.

Ingress processing of packets steps from low layer to high layer, and the egress is opposite. Every layer acts according to corresponding tables of protocol, for example, multicast IP table, which is created and filled by control plane software, such as routing protocol.

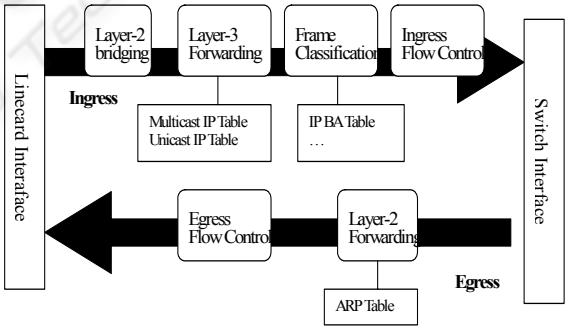

Figure 2: NP Picocode View From Programmer.

3.2 Integrated Coordinated Software Development Toolkit

We focus on the design of integrated coordinated software development toolkit, which would provide great convenience for programmer of NP software. The architecture of toolkit is illustrated as Figure.3.

The top level is NP software view from programmer, composed of information of picocode

flow, related table structure and packet processing performance.

The toolkit involves two main part integrated development environment: CP IDE and NP IDE, which would work coordinately.

CP IDE supports development of CP program. In general, two program frame are given to develop protocol processing callback, and NPAS extension. The main task of NPAS extension program is design of table structure, which also corresponds to table in NP, because entries of NP's one are written by CP. So NP Table Generation module can help create the right structure of NP table directly without manual edition of programmer.

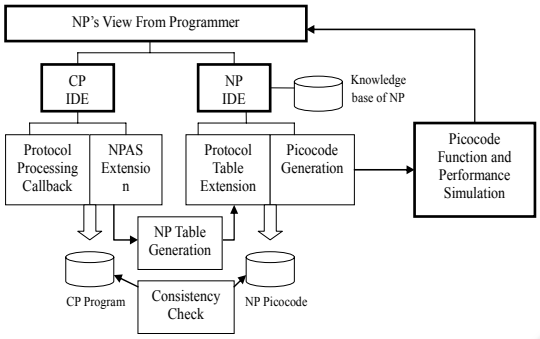

Figure 3: Architecture of Integrated Coordinated Development Toolkit of NP Software.

NP IDE supports development of CP program. In general, two program frame are given to develop protocol table extension, and picocode generation. Specialized picocode module will be programmed by human, but IDE will provide some useful facility to help integration to whole NP software, for example, check and suggestion for interface between different picocode module, allocation of public register and buffer, and so on, which will promote programming efficiency greatly.

After generation of picocode, module of picocode function and performance simulation will compile, debug and test the code. The test result can be shown in the view of programmer. If requirement can not be fulfilled, programmer will check and debug program again, even repartition protocol function. The simulation module will provide possible performance bottleneck and give some improvement suggestion intelligently.

Another useful coordinated working module is consistency check of both CP program and NP picocode, to keep corresponding structure and declaration consistency.

In order to program high-performance modular picocode, NP IDE provides a knowledge base of NP for programmer, which can be queried and sampled. The knowledge include structure of NP, mechanism of all kinds of coprocessors and their programming interface.

- Sample: Knowledge of Tree Search Engine mechanism of IBM PowerNP

Tree search engine (TSE) (IBM, 2001) is a hardware accelerated co-processor for tree search and management of table including access control table, forwarding table, security policy table and so on. Various tables are stored and indexed through Patricia tree which is a path-compressed binary tree. Data of tables is stored in leaves of the tree. Compressed data is stored in direct table (DT) which effectively improves tree search speed through hardware HASH algorithm.

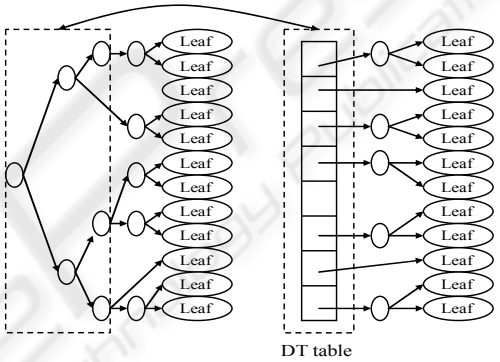

Figure 4: Patricia tree structure.

Tree search engine supports three kinds tree search algorithm: Full Match (FM), Longest Prefix Match (LPM), Software management Tree (SMT).

FM algorithm uses Patricia tree. Leaves are the only matching selection of key. LPM and SMT use extended Patricia tree. Data structure of SMT tree is similar to FM tree. The difference between them is that SMT may have many leave nodes which are organized as a chain. Key search ends on the chain until one match is found or there is no matching totally. FM is suitable for fix length key search while SMT is suitable for multidimensional key (as IP five tuples) search.

PowerNP can support 192-bits key search. The format of leave node can be defined freely in picocode. The extension of various forwarding tables is very easy for the flexible software programming feature of TSE co-processor and NPAS table management interface as the high performance table search speed can be assured.

4 IMPLEMENTATION OF MPLS VPN PROTOCOL

4.1 Function Partition

NP-based MPLS VPN software architecture is composed of control plane and forwarding plane software. As is shown in figure 5. Control plane software includes:

- User interface implements configuration command of MPLS and VPN.
- IP routing protocols implement IP route learning which is the basis of the construction of label switch path. Multiprotocols BGP protocol (MPBGP) (Bates, 2000) implements the delivering of VPN routing information in BGP domains. VRF table management implements the management of VPN routing information tables. It is equal to extract many independent tables from IP routing tables. Each VPN has one corresponding VRF table. VRF route is only managed by VRF manager which is responsible for delivering VRF routing information to MPGP for broadcast at the ingress.
- MPBGP will inform VRF manager about VPN route at the egress.
- Label distribution protocol (LDP) (Andersson, 2001) , implementing MPLS label switch path (LSP) messages;
- NPAS , implementing the management of network processor, including the configuration of various MPLS VPN tables on forwarding plane.
- Forwarding plane software is network processor picocode software:
- NP picocode , maintaining and searching various forwarding tables in network processor, including IP forwarding tables ,VRF forwarding tables, label forwarding tables. MPLS VPN forwarding flow is also implemented.

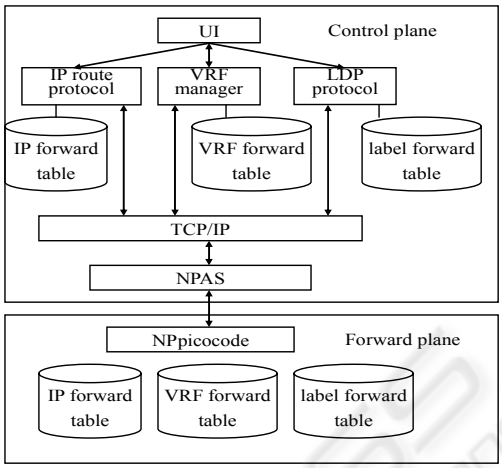

Figure 5: MPLS VPN software architecture.

4.2 Table Design

NP-based MPLS VPN is a function extended over standard network protocol suits, so it is better to minimize changes over primitive architecture while utilizing network processor's scalability effectively. In this section, two of the key issues will be discussed: (1) the design of VRF forwarding table on control point; (2) the implementation of VRF forwarding table and code on network processor.

4.2.1 VRF Table Design on CP

Actually, VRF forwarding table can be regarded as a subset of global table. It is organized according to the need of VPN route. Related properties are also stored in the table, the information of which is delivered by UI configuring module or VRF manager through MPBGP.

VRF should be assigned name and route distinguisher (RD) when it is created. At present, we only consider the scenario of one VRF with single RD in the overall network. RD and destination IP are the index of VPN member in VRF forwarding table. RD is eight bytes, including type, manager and sequence number. It supports autonomic system and IP.

There are two schemes to design VRF table:

- Extension scheme – multi-route table structure and multi-route table HASH table organization structure

To support MPLS VPN, the storage of VPN route information must be taken into account while keeping the primitive route table storage structure. Since VPN route address is private, route managing

unit shall create and maintain multi-route tables, including both original public network route tables which can be regarded as special private route tables whose RD is zero and VPN private route tables which are Radix Tree route tables indexed by corresponding VRF RD. Hence, all the route tables can be organized and stored as Radix Tree route tables indexed by RD. Lookup of these hash tables is very fast. The structure of hash tables is shown as figure 6:

- Modification scheme –single IP forwarding table

This method is implemented through directly modifying the data structure and management function of backbone routers. The index of forwarding tables is also changed as RD plus IPV4 address. New items of the tables such as bottom label are needed to support lookup process for datagram from VRF interfaces and general interfaces whose RD is zero. Mapping between interfaces and RD is needed because of pre-acquisition for RD before table lookup.

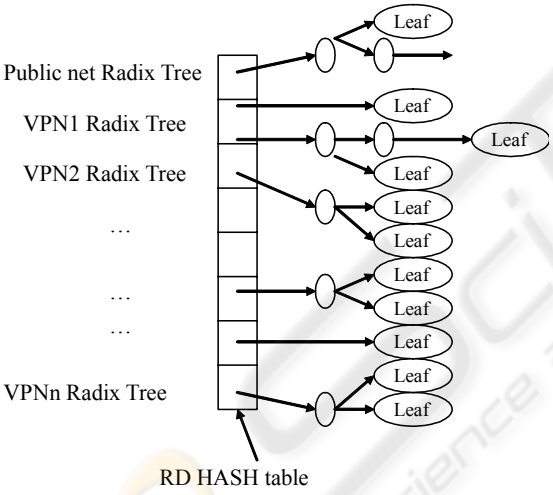

Figure 6: Multi-route table and HASH table structure.

Comparing the two schemes according to workload and performance, we choose the first scheme which needs minimal modification of current BGP and IP route tables ,that is easy for maintain system stability. The basic data structure of VRF table item can be shown as below:

```
; leaf data structure
Struct

np_signature_t   signature
;Signature
np_uint16      VPN_color
; needed for VPN
np_uint32         reserved_hdr
; Reserved variable
np_uint32  flags_counter
; Maint.flags&Count@BVNC
BLIx xxxx Sxxx cccc
;cccc cccc cccc cccc
np_ipps_ecmpActionData_s ecmpActionData
;16 bits
np_ipps_vrf_nextHopForwardingLeaf_s
nextHop[3]
np_uint32       bgp_nexthop
; BGP next hop
np_uint8      bgp_csi1
; BGP skip counter & csi
Sxxx cccc
np_uint16      bgp_csi2
;BGP csi, cccc cccc cccc cccc
np_uint8 bot_label_part1
; goes with insert_bot_label    bottom
label is 20 bits long we use bit 23 (0-
23) for insert bottom label flag
np_uint16   bot_label_part2
np_fwd_table_id_t    fwding_tableid

endstruct
```

4.2.2 VRF Forwarding Table Design and Picocode Flow in Network Processor

- Picocode VRF table design

VRF manager maintains a VRF route table of multi-table structure on both control point and network processor, as is similar to IP forwarding table. However, different from the multi route tables structure on control point, network processor which supports 192-bits key search only maintains one picocode VRF forwarding table.

There are two implementation schemes for relation between VRF forwarding table and IP forwarding table. The first scheme adopts independent table. This scheme defines a new VRF table which is indexed by RD plus IPv4. The second scheme adopts integral table. This scheme modifies IP forwarding table structure and extends key length to 96-bits for VPN searching. The second scheme affects general IP forwarding speed and needs modify IP table management function. So we select the first scheme.

- MPLS VPN picocode flow

The standard functions of edge routers based on network processor include: IP forwarding, MPLS forwarding and so on. The modification of picocode

software is needed to support MPLS VPN through pushing two labels at ingress and pulling them out at egress.

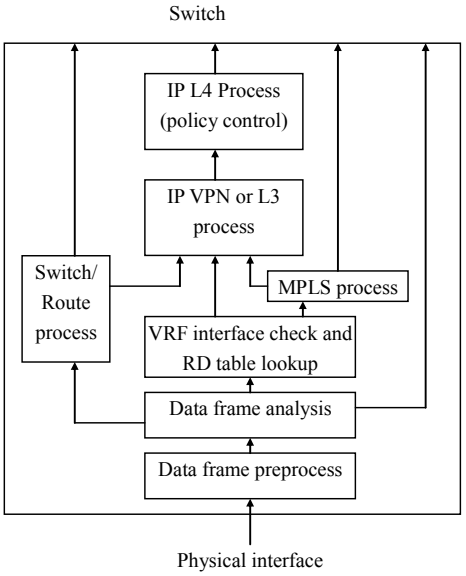

Figure 7: Ingress picocode flow.

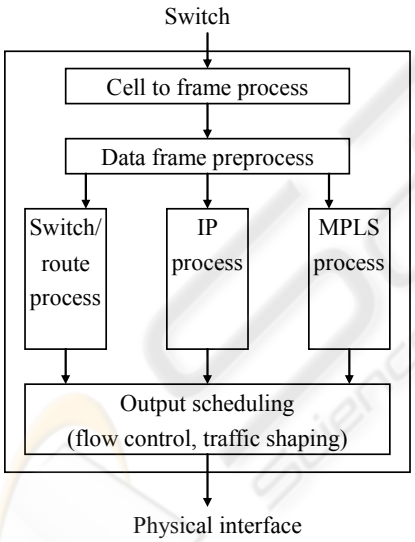

Figure 8: Egress picocode flow.

To construct key words of VPN route, We design RD table for mapping VRF interface to RD. Detailed flow is shown as figure 7,8. Data structure of VRF table item on NP which is corresponding to that is on CP can be shown as below:

```
STRUCT
byte  control_flags
; (BGP)(LV)(I=E_node)(x)  X X X
byte  counter_skip
; flag for enabling/disabling counting
on this entry
hword  counter_csi
; remaining bytes of counter set index
ecmpActionData  hword; ecmp thresholds
(thr1 and thr2)
word   next_hop_0
; IP address
hword encoded_target_blade_0
; compacted TB/TP values
hword action_flags_0
hword egress_Context_0
; egress context
word next_hop_1
; IP address
hword encoded_target_blade_1
; compacted TB/TP values
hword  action_flags_1
hword  egress_Context_1
; egress context

word next_hop_2
; IP address
hword encoded_target_blade_2
; compacted TB/TP values
hword action_flags_2
hword egress_Context_2
; egress context

word  BGP_next_hop
; IP address of BGP Next Hop
byte BGP_counter_skip
; flag for enabling/disabling BGP
counter on this entry
hword BGP_counter_csi
; remaining bytes of BGP counter set
index
byte insertBottomLabel
; MSb (bit 23) = insert bottom label
flag
hword bottom_Label
; bottom label itself is 20 bits long
hword next_lookup_table_id
; Table Id associated with 2nd lookup

endstruct
```

IP VPN process module is extended. VRF interfaces checking module and RD table lookup module are fresh modules. When RD table lookup for datagram from VRF interfaces succeed, IP VPN process is activated. OutSegment index is extracted

from MPLS InSegment table. Standard MPLS datagram are processed by standard MPLS procedure.

We implement two-layer label pushing function for MPLS process module in egress flow to support VPN application.

5 CONCLUSION

In this paper we have discussed a flexible integrated coordinated software development toolkit which has been proposed to support rapid development of protocol from programmer's view, and provides plenty of assistance functions to help NP programming more quickly and efficiently. Through this toolkit, MPLS VPN protocol is developed.

NPs will be adopted in more and more communication devices, especially edge devices for their flexibility of protocol function upgrading. One of our future work is to develop more new network protocols according to new requirement of network application.

ACKNOWLEDGMENTS

This work was supported in part by Chinese NSF grant 90104001/F0107.

REFERENCES

Stephen J. Sheafor.(2000). Network Processor: Using in a New Era of Performance and Flexibility. Retrieved November 20, 2005, from http://www.sitera.com.

C. Sauer K. Keutzer C. Kulkarni, M. Gries (October 2003). Programming Challenges in Network Processor Deployment. *In Int. Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*.

Agere Systems Proprietary. *The Challenge for Next Generation Network Processors*. April 2001.

IBM PowerNP NP4GS3 network processor datasheet. Retrieved December 20, 2005, from http://www.ibm.com/chips/techlib/techlib.nsf/products /PowerNP NP4GS3.

Matthew Adiletta, Mark Rosenbluth, Debra Bernstein, Gilbert Wolrich, and Hugh Wilkinson. (August 2002). The next generation of Intel IXP processors. *Intel Technology Journal,* 6(3):6-18.

James Allen, Brian Bass, Claude Basso, Rick Boivie, Jean Calvignac, Gordon Davis, Laurent Frelechoux, Marco Heddes, Andreas Herkersdorf, Andreas Kind, Joe Logan, Mohammad Peyravian, Mark Rinaldi, Ravi Sabhikhi, Michael Siegel, and Marcel Waldvogel. (2003). PowerNP network processor: Hardware, software and applications. *IBM Journal of Research and Development.*

Intel Press. 2001. *MicroACE, design document, revision 1.0*. Intel Press, Intel Corporation.

Campbell A.T., Kounavis M.E., Villela D.A., Vicente J.B., de Meer H.G.,Miki K., and Kalaichelvan K.S. (June 2002). NetBind: A Binding Tool for Constructing Data Paths in Network Processor-based Routers. *In 5th IEEE International Conference on Open Architectures and Network Programming (OPENARCH'02).*

Wagner J. Leupers R.(2001). C compiler design for an industrial network processor. *Proceedings of the 2001 ACM SIGPLAN workshop on Optimization of middleware and distributed systems*.

Memik G. Mangione-Smith W H. Hu W.(2001). Netbench: A benchmarking suite for network processors*. ICCAD.*

Network Processing Forum Working Group (Oct 2002). Network processing forum backgrounder. Retrieved November 20, 2005, from http://www.npforum.org/.

Ivan Pepelnjak, Jim Guichard, 2001. *MPLS and VPN Architectures*. Cisco Press.

Rosen, E., Rekhter, Y. (1999). BGP/MPLS VPNs, *IETF RFC 2547*.

Bates, T., Rekhter, Y., Chandra, R., Katz, D.(2000). Multiprotocol Extensions for BGP-4, *IETF RFC 2858*

Andersson, L., Doolan, P., Feldman, N. , Fredette, A. ,Thomas., B.(2001). LDP Specification., *IETF RFC 3036.*