

Understanding Network Processors

By Niraj Shah
niraj@eecs.berkeley.edu

VERSION 1.0

4 SEPTEMBER 2001

Table of Contents

0	Intended Audience	1
1	Introduction	2
1.1	What is a Network Processor?	5
1.2	A Brief History.....	5
2	A Profile of Network Applications.....	7
2.1	Network applications	7
2.2	Kernels	22
2.3	Summary	23
3	Network Processors	27
3.1	Agere (PayloadPlus)	27
3.2	Alchemy (Au1000).....	30
3.3	Applied Micro Circuits, formerly MMC Networks (nP7xxx)	31
3.4	Bay Microsystems	32
3.5	BRECIS Communications (MSP5000)	33
3.6	Broadcom, formerly SiByte (Mercurian SB-1250)	34
3.7	Cisco (PXF/Toaster 2)	35
3.8	ClearSpeed, formerly PixelFusion	36
3.9	Clearwater Networks, formerly XStream Logic Devices (CNP810SP).....	37
3.10	Cognigine	39
3.11	Conexant, formerly Maker (MXT4400 Traffic Stream Processor)	40
3.12	EZchip (NP-1)	41
3.13	IBM (PowerNP)	42
3.14	Intel, formerly Level-One (IXP1200)	44
3.15	Lexra (NetVortex & NVP)	46
3.16	Motorola, formerly C-Port (C-5 DCP).....	48
3.17	PMC-Sierra, formerly Quantum Effect Devices	50
3.18	Vitesse, formerly SiTera (PRISM IQ2000)	50
3.19	Xelerated Packet Devices (X40 & T40)	51
3.20	Summary	53
4	Analysis	55
4.1	Market Segmentation	55
4.2	Architecture	56
4.3	Programmability	63
4.4	Summary	65
5	Looking Forward.....	67
5.1	Applications.....	67
5.2	Architecture.....	67
5.3	Mapping Applications onto Architectures.....	69
6	Conclusions	71
7	Web Sites	72
8	Acronym Dictionary	74
9	References.....	75
	Appendix	79
A.	Detailed Network Processor Summary	79
B.	Applications/Architecture Mapping Table.....	87

List of Figures

Figure 1. Space of System Implementations.	3
Figure 2. Comparison of System Implementations.	4
Figure 3. The Solution Space of Network Processing.....	5
Figure 4. OSI Protocol Stack.	7
Figure 5. ATM cell header.	8
Figure 6. The protocol stack for IP over ATM.	9
Figure 7. Internet Protocol (IP) Header Format.	10
Figure 8. IPv6 Packet Header Format.	13
Figure 9. Type 0 Routing Extension.	14
Figure 10. Difference between Transport and Tunnel mode.....	14
Figure 11. AH Header Format.....	15
Figure 12. ESP Header Format.....	16
Figure 13. TCP header and optional data.....	17
Figure 14. Wireless TCP/IP Gateway.....	18
Figure 15. Logical view of a DiffServ node.	21
Figure 16. Agere PayloadPlus System.	28
Figure 17. FPP Block Diagram [24].	29
Figure 18. Architecture of the Agere Routing Switch Processor [26].	30
Figure 19. Alchemy's System Architecture [29].	31
Figure 20. Applied Micro Circuits' EPIF-200 Network Processor [30].	32
Figure 21. BRECIS Communications' MSP5000 [34].	34
Figure 22. Broadcom's Mercurian Architecture [37].	35
Figure 23. Example use of Cisco's PXF NP [39].	36
Figure 24. Macro-Architecture of Clearwater Networks' CNP810SP Network Processor. ...	38
Figure 25. Cognigine's RCU Architecture.	40
Figure 26. EZChip's NP-1 Architecture [45].	41
Figure 27. IBM's Network Processor [50].	43
Figure 28. Embedded Processor Complex Architecture [50].	43
Figure 29. Ingress and Egress Frame Flow [50].	44
Figure 30. Intel's IXP1200 Architecture.	45
Figure 31. An Example Use of Lexra's NetVortex System Architecture [56].	47
Figure 32. Motorola C-5 DCP Macro-architecture [60].	49
Figure 33. Vitesse's PRISM IQ2000 Architecture [64].	51
Figure 34. Macro-architecture of the X40.	52
Figure 35. Example use of Xelerated's X40 Packet Processor.	53
Figure 36. Varying Solutions of Network Processors.	54
Figure 37. Timeline of Network Processor Releases.	56
Figure 38. Issue Width per Processing Element Versus Number of Processing Elements....	58
Figure 39. Number of Processing Elements Versus MIPS (log scale).	59
Figure 40. Comparison of Multiple Thread Support among Network Processors.	63
Figure 41. Map of Network Processor Market.	65

List of Tables

Table 1. Applications and their kernels (part 1).	24
Table 2. Applications and their kernels (part 2).	25
Table 3. Applications and their kernels (part 3).	26
Table 4. Characteristics of the 3 major NP markets.	55
Table 5. Comparison of Network Processing Planes.	56
Table 6. Specialized Hardware Employed by Network Processors.....	62
Table 7. Micro-architectural Comparison of NPs.....	80
Table 8. Architectural Comparison of NPs.....	82
Table 9. Comparison of Software Support for NPs.	84
Table 10. Comparison of Memory for NPs.....	86
Table 11. Comparison of Physical Implementation for NPs.	87
Table 12. Mapping of Applications onto Architectures (part 1).....	88
Table 13. Mapping of Applications onto Architectures (part 2).....	89

0 Intended Audience

This document presents a survey and analysis of network processors. It is intended primarily for four major audiences:

- Network processor architects who want to know the technical details about current network processor offerings
- Network processor product managers who want to know the features, performance, and range of target applications of their competitors' products
- Users of network processors who want to incorporate them into their products but are having trouble choosing which device best suits them
- Developers and designers in network processor related fields, like network processing software, network co-processors, and network testing equipment

1 Introduction

The bandwidth explosion of the past couple years has impacted every part of our lives and this exponential growth will continue for many more years. The dropping cost of bandwidth allows the masses to take full advantage of the connectivity the Internet provides. This will result in more bandwidth-hungry and computationally intensive applications, like Voice over IP (VoIP), streaming audio and video, Peer-to-Peer (P2P) applications, Virtual Private Networks (VPNs), and many others that we have not even thought of yet.

For networks to effectively handle these new applications, they will need to support new protocols that include differentiated services, security, and various network management functions. While networks are demanding equipment with very high throughput, they also need the flexibility to support new protocols and applications. In addition, the ever-changing requirements of network equipment require solutions that can be brought to market quickly.

Today's legacy network implementations are based on Field Programmable Gate Arrays (FPGAs) for lower level processing and General Purpose Processors (GPPs) for higher layer processing. Neither of these solutions meets all the requirements that network processing demands. Consider the broad categories of alternatives for system implementation:

- ASIC (Application Specific Integrated Circuit) – any hardwired solution
- ASIP (Application Specific Instruction Processor) – an instruction set processor specialized for a particular application domain
- Co-processor – a hardwired, possibly configurable solution with a limited programming interface
- FPGA (Field Programmable Gate Array) – a device that can be reprogrammed at the gate level
- GPP (General Purpose Processor) – a programmable processor for general purpose computing

Figure 1 maps these categories on two axes: flexibility and performance. As shown, ASICs are the most hardwired (least flexible), but provide the highest performance. At the opposite end of the spectrum, GPPs are the most general (flexible) at the cost of the lowest performance. FPGAs provide an interesting value proposition: in the absence of ASIPs or co-processors, they are higher performance than GPPs with more flexibility than ASICs.

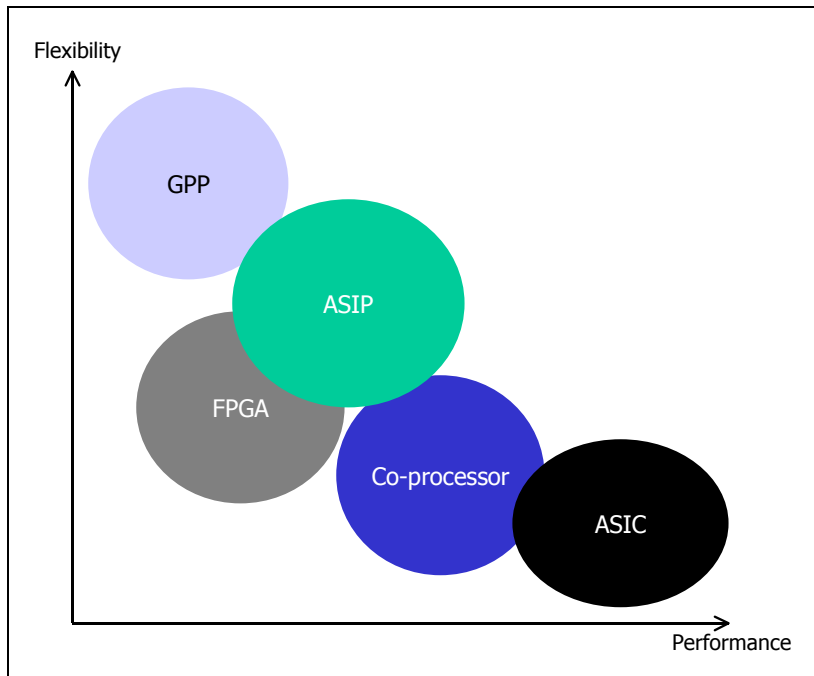


Figure 1. Space of System Implementations.

We further compare these system implementations in Figure 2. From this comparison, it is pretty clear that an ASIP is the best approach for most networking system implementations. An ASIP for networking, or Network Processor (NP), provides the right balance of hardware and software to meet all the requirements stated above:

- Performance – by executing key computational kernels in hardware, NPs are able to perform many applications at wire speed
- Flexibility – having software as a major part of the system allows network equipment to easily adapt to changing standards and applications
- Fast TTM – designing software is much faster (and cheaper) than designing hardware of equivalent functionality
- Power – while NPs may not be embedded in energy-sensitive devices (like handhelds), their power consumption is important for cost reasons (e.g. implications on packaging).

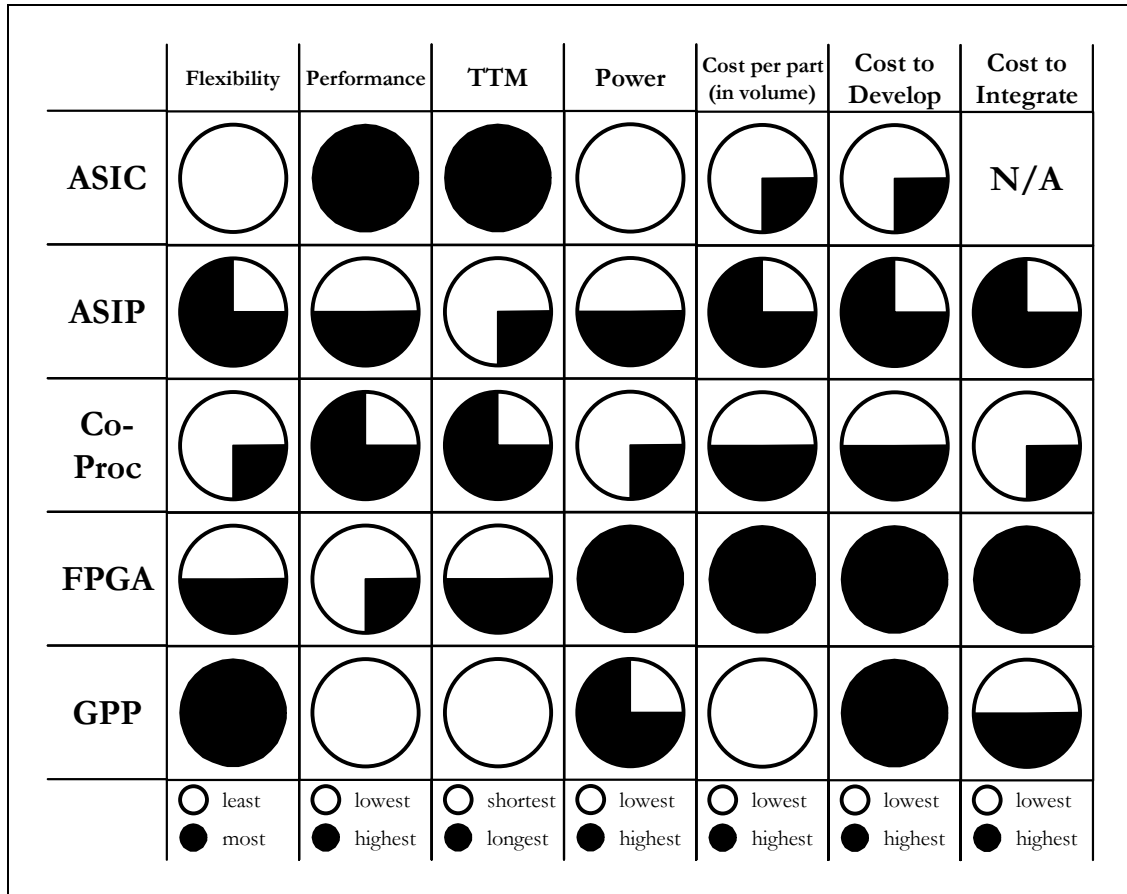


Figure 2. Comparison of System Implementations.

Network processors are part of a broader movement from ASICs to programmable system implementations. Numerous trends have come to light recently that are making the design of ASICs more difficult [1]:

- Deep submicron (DSM) effects are exacerbating circuit design difficulties
- Exponentially increasing number of devices on-chip
- On-chip integration of increasingly diverse elements
- Shrinking time-to-market

The combination of these pressures has resulted in a shift in system implementations to more programmable solutions. The recent explosion in network processor architectures supports this observation.

Twenty-four months ago, there were only a few network processors in development and only one shipping product (MMC Networks, now Applied Micro Circuits). Now, it seems every month a new network processor is announced. In an attempt to alleviate the bandwidth bottleneck, numerous solutions have emerged. They vary greatly in micro-architectural and architectural complexity, memory architecture, software support, and physical implementation. The purpose of this report is to survey and make sense of the fast growing network processing space.

We start with a definition of network processors and give a brief history of network processors. Then we profile common networking standards and applications and define their key characteristics. Next, we survey current network processors and detail their features. This allows us to analyze the network processor field and define market segments. Lastly, we identify some trends and provide some conclusions.

1.1 What is a Network Processor?

A network processor is an ASIP for the networking application domain – a software programmable device with architectural features and/or special circuitry for packet processing. While network processors do not cover all solutions to networking applications, we believe it covers the most exciting and high growth parts of the space. Our definition is broad to reflect the wide range of programmable architectures proposed for network processing. As a result, network processors share characteristics with many different implementation choices:

- network co-processors
- communication processors used for networking applications
- “programmable” state machines for routing
- reconfigurable fabrics (e.g. FPGAs)
- GPPs used for routing

Figure 3 shows the space of solutions for network processing. While network processors are not the only solution to network processing, they are the focus of this report.

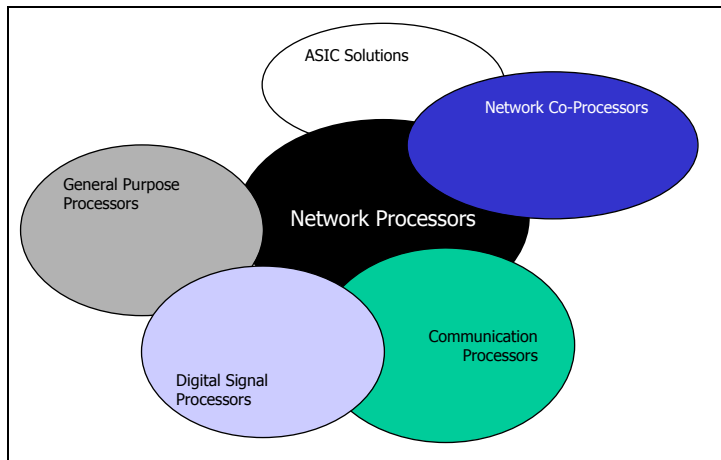


Figure 3. The Solution Space of Network Processing

1.2 A Brief History

Until recently, most networking functions above the physical layer have been implemented by software running on a general-purpose processor. The past few years have been witness to the exponential growth of the Internet. This super-Moore’s law growth has wreaked havoc on networking implementations. To cope with this traffic explosion, new solutions have emerged. First, many hardwired solutions appeared for layer 2 and 3 processing [2]. With the rapid change in lower layer protocols (e.g. MPLS, DiffServ) and higher layer

applications (e.g. Peer-to-Peer, streaming video), this solution will not scale. The need for customizability, in-the-field programmability, and shrinking time to market windows in network processing implementations has focused most of the activity on network processors.

2 A Profile of Network Applications

Before surveying network processors, we examine common networking applications. Since NPs are “application-specific” to networking, it is only natural to examine these applications first. We decompose each application into their computational kernels. By enumerating 1) the specific operations required for networking applications; and 2) the architectural features of network processors, we can identify the mapping of common applications on to target architectures. This mapping is the key to evaluating network processors.

2.1 *Network applications*

In this section, we describe the characteristics of various networking applications that would execute on a network processor. Our goal here is not to give a full tutorial of networking applications, rather give a flavor of them. The reader is referred to [3], [10], and [4] for an in-depth tutorial on networking applications.

For context, we first describe the OSI stack model [5]. We’ve classified different networking applications that could be found in a variety of network equipment (for example, core and edge routers, backbone switches, URL load balancers, traffic shapers, firewall appliances) into three categories: protocol standards, gateway applications, and Quality of Service (QoS) related applications.

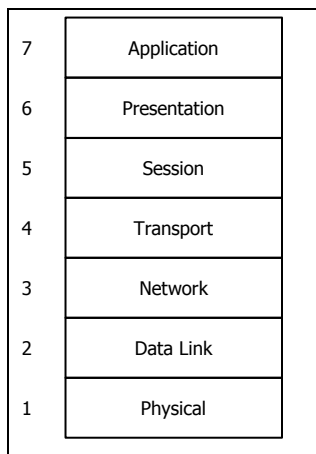


Figure 4. OSI Protocol Stack.

The following is a brief description of layers in the OSI stack as they relate to this report:

- **Layer 1:** The Physical layer defines the medium over which point-to-point links are established.
- **Layer 2:** The Data Link layer provides for a point-to-point link between two computers. It provides reliability on top of an otherwise unreliable physical link. While most Layer 2 operations have historically been performed in hardware, NPs are also attacking this task.
- **Layer 3:** The Network layer enables communication between any two computers on the network using the point-to-point communication facility provided by the Data Link layer.

- **Layer 4:** The Transport layer defines a socket, a point of access for a higher layer application to communicate with another end-station. Both TCP and UDP provide a port number to higher layers for applications to identify the endpoints of the communication. The combination of IP address and port number uniquely identify a socket. Since much of the Layer 4 functionality is only executed on an end-station, we describe only the parts of the protocol that are relevant to NPs.
- **Layers 5-7:** While network equipment in the fabric may access Layer 5-7 information, most of the Layer 5-7 tasks are executed on an end-station.

Protocol Standards

The following is a breakdown of applications related to protocol standards across different layers. We do not describe the details of each protocol in depth; rather, we highlight the operations that need to be performed within the fabric at wire speed (data-plane operations). Management- and control-plane applications are mostly control-dominated and best suited for a general-purpose processor.

Asynchronous Transfer Mode (ATM) Switching

Asynchronous Transfer Mode (ATM) [6] is a connection-oriented standard in which the end-stations determine a virtual circuit (VC), or path, through an ATM network. The VCs are made up of different virtual paths (VPs), or paths between switches. Once control-plane functions setup a VC, an ATM switch simply switches ATM cells from input ports to output ports. This switching is based on consulting a lookup table indexed by two fields in ATM cells:

- virtual circuit identifier (VCI): 8-bit VC identifier
- virtual path identifier (VPI): 16-bit VP identifier

A switch may then alter the VPI and VCI fields of the cell to update the new link the cell is traveling on.

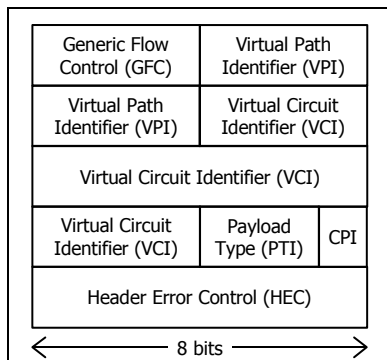


Figure 5. ATM cell header.

ATM Adaptation Layer 5 (AAL5)

The ATM Adaptation Layers (AALs) provide different ways for ATM to communicate with higher layer protocols (see

Figure 6). The most popular method is AAL5, which is often used for IP over ATM. Since IP packets are larger than ATM cells (48 byte payload), AAL5 provides a guideline by which to segment IP packets so they can travel over an ATM network and a facility to reassemble

ATM cells back into IP packets. To accomplish this, AAL5 defines its own Packet Data Unit (PDU) with the following contents [6]:

- Payload: Higher layer PDU, maximum 65,535 bytes
- Padding: for AAL5 PDUs to evenly fit into some number of ATM cells
- 8-byte trailer:
 - User-to-User (UU) field: 1 byte
 - Common Part Indicator (CPI) field: 1 byte
 - Length field: 2 bytes
 - CRC field: 4 bytes

After calculating the amount of padding needed, the length field, and the CRC field, the AAL5 PDU is simply sliced into 48 bytes chunks that are used as the payload for ATM cells.

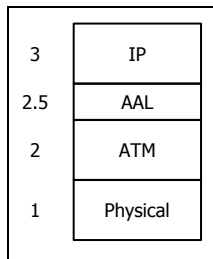


Figure 6. The protocol stack for IP over ATM.

Virtual Local Area Network (VLAN)

A VLAN is a group of end-stations, perhaps on multiple physical LAN segments, that communicate as if they were on one LAN. There are four major approaches to defining VLAN membership. While each of these approaches has their advantages and disadvantages, our goal is not evaluate them, but rather describe the operations needed to support VLANs [7] [8].

1. Port grouping: A set of ports on a switch defines a VLAN
2. MAC layer grouping: VLANs based on MAC layer addresses
This approach requires a switch or router to inspect the MAC address of each frame.
3. Network layer grouping: VLAN membership based on network layer address
This approach requires a switch to examine the network layer address (e.g. subnet address for TCP/IP) to determine VLAN membership.
4. IP multicast grouping: VLAN defined as an IP multicast group
This approach requires a router to simply support IP multicast.

Regardless of the approach used, once the VLAN group is determined for a frame, the switch is required to add a unique identifier to the header that designates VLAN membership.

Multi-Protocol Label Switching (MPLS)

MPLS [9] is a protocol for efficient routing, forwarding, and switching that is independent of layer 2 and layer 3 protocols. A router that supports MPLS is known as a Label Switch Router (LSR). The main task for an LSR is to switch packets based only upon an MPLS label. The knowledge of where to direct packets is either setup beforehand (via a control protocol, like ATM) or is determined based on packet flow. A Label Edge Router (LER) is a node that sits on the boundary of an MPLS network (and another network, like ATM or

Ethernet). Its primary function is to categorize and add labels to ingress packets, ensure enforce Service-Level Agreements (SLAs), and remove MPLS labels for egress packets.

Since an MPLS label is only 20 bits long, an LSR should be able to switch packets with high throughput. For each incoming packet, an LSR will lookup its label, determine the output port, decrement the Time-To-Live field and update the label (labels are of only local significance). An LER has considerably more work to do, as it must assign labels to incoming packets and ensure incoming flows conform to predetermined traffic patterns.

Internet Protocol Version 4 (IPv4)

Internet Protocol version 4 (IPv4) is the most widely used protocol for layer 3 communication. Figure 7 shows the header format of an IPv4 packet. The major processing steps for IPv4 packets are routing, fragmentation and reassembly, and address resolution protocol (ARP). Descriptions of the different algorithms and implementations are given below [10].

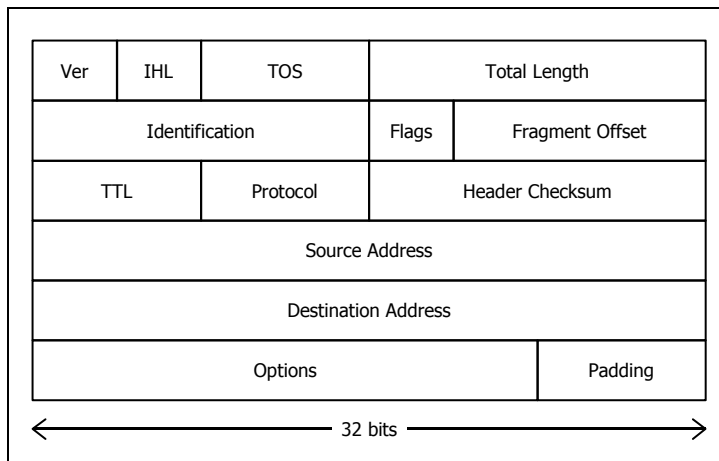


Figure 7. Internet Protocol (IP) Header Format.

Routing

The major steps in routing a packet are:

- Remove the packet from an input queue
- Check the version of the packet: verify the 4-bit Version field equals 4
- Check the Destination Address: make sure it is not in IP Address Class E (240.0.0.0 to 254.255.255.254), which is reserved for experimental use
- Verify checksum:
 - Store the Checksum field and clear it
 - Treat the header as a series of 16-bit integers
 - Compute the 16-bit one's complement of the one's complement sum these integers
 - Compare this sum to the Checksum field
- Lookup route:
 - Lookup the destination IP address in the routing table

There are many different algorithms for performing routing table lookups. However, they all use longest prefix matching, which allows entries to contain wildcards and find the entry that most specifically matches the input address. For example, all packets going to subnet 128.32.xxx.xxx may have the same next hop. While this significantly reduces the size of the routing table, multiple lookups may be required, depending on the data structures and algorithms used.

- Get the IP address of the next hop
- Update Time-To-Live (TTL): decrement the TTL field by one and correspondingly adjust the Checksum field
- Insert the packet in one of the output queues

Fragmentation & Reassembly

Fragmentation and Reassembly is the result of IP insulating higher layer protocols from the implementation of the Data Link layer. An IP packet may need to be fragmented if it is larger than the Maximum Transmission Unit (MTU) of the Data Link layer. For example, the maximum size of an Ethernet frame is 1518 bytes (12 bytes for the header, up to 1500 bytes for the payload, and 6 bytes for the trailer). For IP over Ethernet, all IP packets larger than 1500 bytes must be fragmented. The reverse of fragmentation is reassembly. This is required for any device on the network that wants to perform higher layer processing.

Fragmentation occurs just before a datagram is placed in the queue to a network interface. The major steps in fragmentation are:

- Verify that it's legal to fragment the datagram: check the Don't Fragment flag
- Determine how many datagrams are needed, based on the size of the original datagram and MTU
- Create each new fragment datagram:
 - Copy the header from the original datagram
 - Copy the appropriate section of the payload from the original datagram
 - Set the More Fragments flag to 1 (except for the last fragment datagram)
 - Set the Total Length field
 - Set the Fragment Offset field
 - Calculate the Checksum field

Reassembly requires some more data structures because IP does not guarantee ordered packet delivery. Since there are many different implementations of this, we only provide some requirements for reassembly:

- Quick location of the group fragments that comprise the original datagram
- Fast insertion of new fragment into a group
- Efficient test of whether a complete datagram has arrived

An example implementation of reassembly uses a table of queues. If an incoming packet is a fragment, attempt to match its source and destination addresses to an existing entry in the table. If there is a match, enqueue the fragment into the appropriate queue and check if the complete datagram has arrived. Otherwise, the fragment is a part of a new datagram; therefore, create a new entry in the table.

Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) [11] provides the data link layer with a mapping of network (IP) addresses to physical network addresses (e.g. MAC addresses). This insulates higher layer protocols from the physical implementation details. For a network interface to send a datagram, it needs the physical address of the destination. ARP support can be described as a cache: given an IP address, return the corresponding physical address. If the IP address does not exist in the cache, generate a request for it (using a control protocol) and queue the packet. This approach is similar to managing a routing table – some control-plane processing is required to manage the cache (populate entries, evict expired entries, etc.).

Internet Protocol Version 6 (IPv6)

Internet Protocol Version 6 (IPv6) is a next generation Internet protocol designed to overcome some of the limitations of the current protocol, IPv4. The major differences between the two protocols are:

- IPv6 uses 128-bit addresses (instead of 32-bit IPv4 addresses).
- IPv6 has a Flow Label field for Quality of Service support.
- IPv6 does not calculate a checksum, rather, it relies on other layers ensure data integrity.
- IPv6 routers do not perform fragmentation or reassembly; they rely on higher layer protocols to fragment large packets.
- IPv6 includes a security protocol, IPSec.

Figure 8 shows the format of IPv6 packets [12]. In addition to the standard header, IPv6 defines header extensions for specifying additional information about the packet (either to routers in the fabric or to end-stations). The headers defined to date include:

- Hop-by-Hop Options: examined by every router
- Routing: specifies one or more intermediate nodes the packet must visit
- Fragment: additional information for packets which have been fragmented by the source
- Destination Options: optional information for the destination node
- Authentication: for IPSec
- Encapsulating Security Payload (ESP): for IPSec

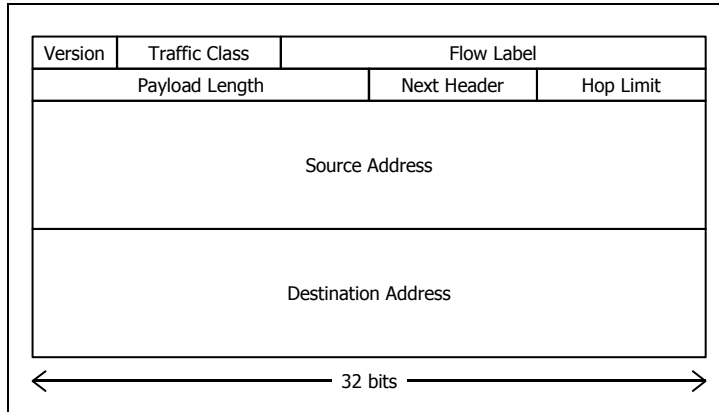


Figure 8. IPv6 Packet Header Format.

Routing

Basic IPv6 routing is similar to IPv4 (i.e. Longest Prefix Match), except source and destination address are 128 bits. One of the most common implementations of LPM routing tables is the Patricia Tree algorithm [13] – a path compressed binary trie algorithm. Another common implementation is a hash table-based approach that maps different bit lengths into separate hash tables. Binary search is then used on these hash tables to find the next hop address.

The Flow Label is a 20-bit field used by packets to request special handling from routers the packet encounters. A flow is uniquely identified by a source address and flow label. It can be used a number of ways: interaction with various control protocols (e.g. RSVP), traffic classes for differentiated services, treatment of TCP connections. For example, if a unique flow label is used for each TCP connection, a receiver could use it to de-multiplex connections [14]. However, this could impact routing caches because they would no longer be based on just the destination, but the flow label as well.

Hop-by-Hop Options

Hop-by-hop header extensions need to be examined by every router the packet encounters. The extensions, which are inserted after the main IPv6 header, have a separate header that specifies the presence of another extension field and the length of the extensions. An extension option is defined as a set of three values: type (8 bits), length (8 bits), and data (variable length).

Routing Header Extensions

The Routing Header Extensions define one or more intermediate nodes to be visited by a packet. The only type of routing header extension defined so far is Type 0 Routing, which has the following format:

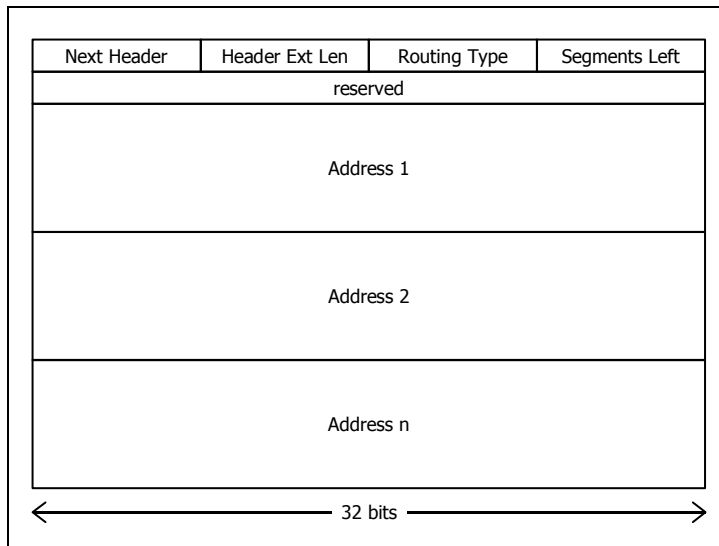


Figure 9. Type 0 Routing Extension.

Every router must examine the addresses not yet visited by the packet and base routing decisions on the remaining addresses.

IP Security (IPSec)

IP Security (IPSec) provides an extensible security platform at layer 3 for higher layer protocols. This relieves higher layer protocols from defining their own ad-hoc security measures. IPSec consists of two protocols [15]:

- Authentication Header (AH): proof-of-data origin, data integrity, and anti-replay protection
- Encapsulated Security Payload (ESP): AH plus data confidentiality, limited traffic flow confidentiality

Either of these protocols can be implemented in **transport mode** (protects higher layer protocols only) or **tunnel mode** (protects IP layer and higher layer protocols by encapsulating the original IP packet in another packet).

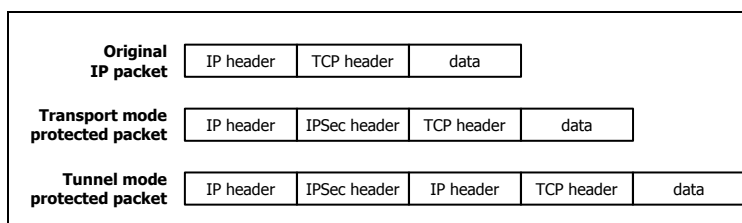


Figure 10. Difference between Transport and Tunnel mode.

To ensure all participating network equipment is consistent, some connection-related state (how to protect traffic, what traffic to protect, and with whom protection is performed) must be stored at each of the endpoints of a secure connection. This state is called a Security Association (SA). The SA is updated using various control protocols and is consulted for data-plane operations.

IPSec is implemented in IPv6 as an extension header. For IPv4, an IPSec header is inserted after the IP header.

Authentication Header (AH)

Authentication Header (AH) does not provide data confidentiality, but it does verify the sender and data integrity. The Security Parameters Index (SPI), along with the destination address, helps identify the Security Association (SA) used to authenticate the packet. The Sequence Number field is a monotonically increasing counter that is used for anti-replay protection, which protects against replay attacks. Anti-replay service is implemented by a sliding window of acceptable Sequence Numbers.

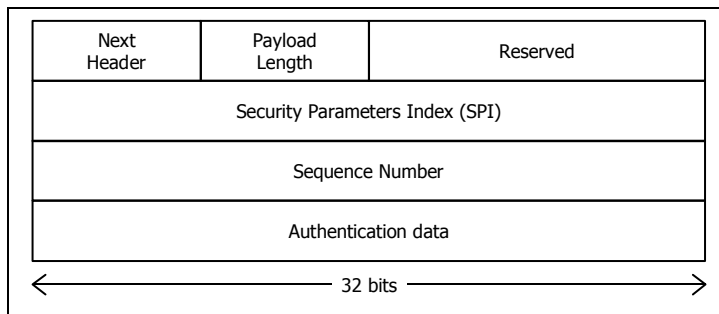


Figure 11. AH Header Format.

For ingress packets, a device that supports AH must execute the following operations:

1. If the packet is fragmented, wait for all fragments and reassemble
2. Find SA used to protect the packet (based on destination address and SPI)
3. Check validity of sequence number
4. Check Integrity Check Value (ICV)
 - a. Save authenticated data and clear authentication field
 - b. Clear all mutable fields
 - c. Pad packet, if necessary
 - d. Execute authenticator algorithm to compute digest
 - e. Compare this digest to the authenticated data field
5. Possibly increment window of acceptable sequence numbers

The following list enumerates the steps involved in supporting AH for egress packets:

1. Increment sequence number in SA
2. Populate fields in AH header
3. Clear mutable fields in IP header
4. Compute Integrity Check Value (ICV) using authentication algorithm and key defined in SA
5. Copy ICV to authentication data field

Encapsulating Security Payload (ESP)

Encapsulating Security Payload (ESP) provides data confidentiality and authentication. ESP defines a header and trailer that surround the protected payload. The presence of the trailer means that the payload may have to be padded (with zeros) to ensure 32-bit alignment. Some data encryption algorithms require a random initialization vector; if necessary, this is stored just before the protected data.

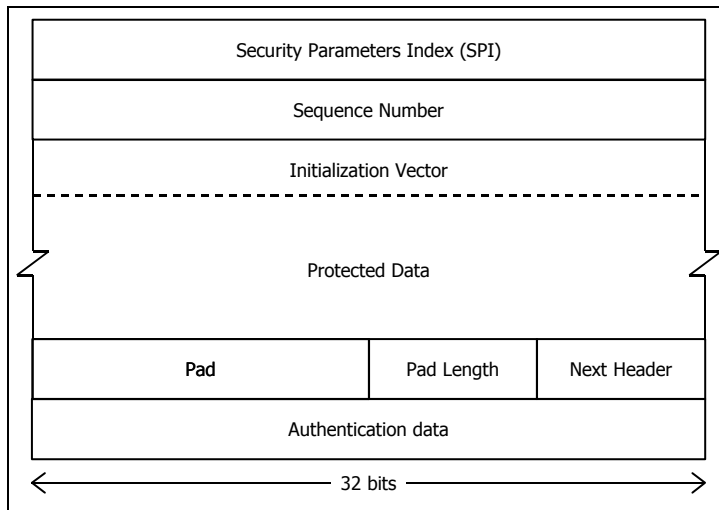


Figure 12. ESP Header Format.

The list below illustrates the major steps that are required to support ESP in ingress packets:

1. Wait for additional fragments, if applicable
2. Check for SA, drop packet if one does not exist
3. Check sequence number, drop if outside of window or duplicate
4. Authenticate packet (same as 4 in AH ingress support)
5. Decrypt payload using key and cipher from SA
6. Check validity of packet with mode (transport vs. tunnel)
7. Check address, port, and/or protocol, depending on SA

On the egress side, the following functions must be executed for each packet:

1. Insert ESP header and fill in fields
For transport mode, an ESP header just needs to be inserted. For tunnel mode, the original IP packet needs to be wrapped in another IP packet first, then the ESP header needs to be added.
2. Encrypt packet using cipher from SA
3. Authenticate packet using appropriate algorithm from SA and insert digest to authentication field in trailer
4. Recompute and populate checksum field

User Datagram Protocol (UDP)

User Datagram Protocol (UDP) is a layer 4 protocol that provides connectionless communication between applications. A UDP header is composed of four 16-bit fields:

- Source Port
- Destination Port
- UDP Length field
- UDP Checksum (optional, this field is zero if not set)

The steps involved in processing an ingress UDP packet at an end-station are as follows:

1. If Checksum field is non-zero (i.e. calculated by the sender), verify checksum
2. Search the set of datagram queues for the one that matches the UDP port

3. Check the status of the proper queue (e.g. overflow, etc.) and enqueue the datagram
4. Send a signal to the process/application (or operating system) indicating a packet has arrived

An NP does not execute most of these functions, but we include them here for completeness. An NP may use the source and destination port to make switching/routing decisions.

Transport Control Protocol (TCP)

Transport Control Protocol (TCP) provides a reliable layer 4 communication for higher layer applications over an otherwise unreliable medium (like IP, which may drop packets). As a result, it is one of the most complex protocols. While most of the TCP-related processing is rather involved (and executed on an end-station), only a small part of the protocol standard is relevant for fabric processing. For example, a web switch may examine TCP packets to determine the beginning and end of sessions.

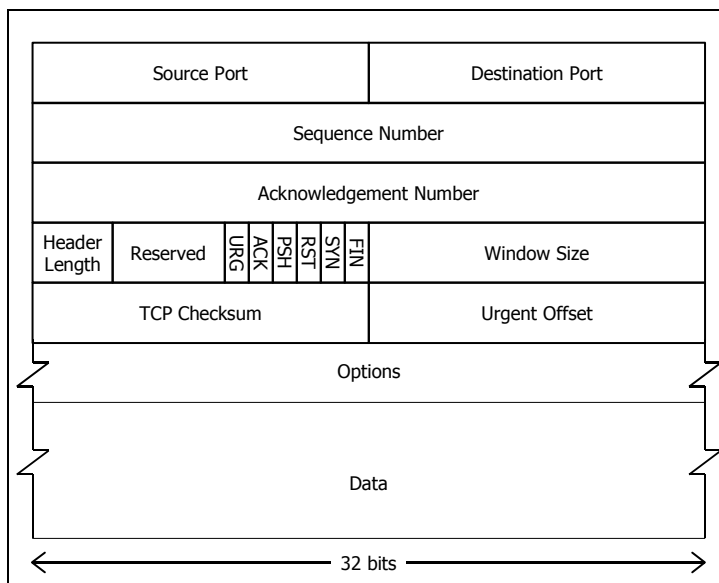


Figure 13. TCP header and optional data.

A TCP header consists of the following fields (see Figure 13) [10]:

- Source Port – port number of the session source
- Destination Port – port number of the session destination
- Sequence Number – unique number for all packets sent in a TCP connection
- Acknowledgement Number – the next sequence number that the sender of an Acknowledgement message expects to receive
- Header Length – length of header
- URG flag – the urgent offset valid
- ACK flag – the Acknowledgement number is valid
- PSH flag – receiver should pass data to application without delay
- RST flag – reset the connection
- SYN flag – establish connection

- FIN flag – sender is finished sending data
- Window Size – window size of acceptable data
- TCP Checksum
- Urgent Offset – offset to urgent data

Gateway Applications

Gateway applications occur near the edge of a network. The applications may alter packet headers, redirect packet flows, or cache packets, but they maintain the semantics of existing protocols.

Wireless TCP/IP

Accessing the Internet over a wireless medium violates some of the key assumptions of the TCP/IP protocol. For example, when TCP does not receive an acknowledgement for a packet that it sends, it assumes the packet did not reach the destination because of network congestion. This causes TCP to initiate an exponential back-off algorithm that waits to resend packets. While this may work well for wireline implementations, in the wireless domain, it is more likely the transmission failed due to an error on the physical layer. As a result, TCP waits exponentially longer to send packets even in the absence of traffic.

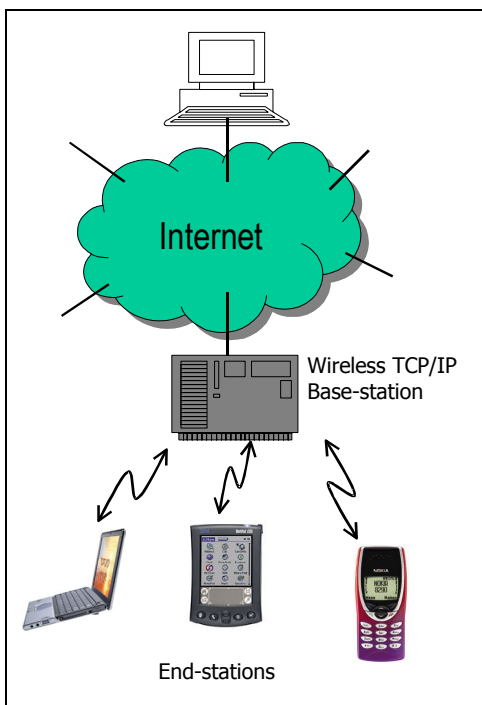


Figure 14. Wireless TCP/IP Gateway.

There are two broad approaches to solving this problem: have the sender alter its actions because the destination is a wireless node or have the destination gateway (base-station) resolve this problem. The former approach seems rather difficult to implement, because it requires all nodes that send to wireless end-stations are affected. The latter approach is transparent to the network, as the sender is not aware that the destination node is wireless.

There are a few main approaches to designing a base-station that connects wireless end-stations to the Internet [16].

In the first approach, the base-station may serve to break the connection from sender to wireless node. This requires the base-station to perform all the higher layer protocol processing and store the packets of the session. The base-station will then send these packets to the wireless handset (either using TCP or some other protocol). Since the base-station sends acknowledgements back to the sender, the sender is unaware the destination is wireless. This approach can place a large load on the base-station and incurs a severe overhead.

Another approach is for the base-station to cache packets that have not yet been acknowledged by the wireless handset. In the event a packet is dropped, the base-station can resend the packet, instead of having the sender resend the packet.

Both approaches require the base-station to intercept TCP packets to/from the wireless end-station. The first approach demands the base-station maintain a TCP protocol stack for each connection passing through it and another protocol stack for each connection to an end-station. The second approach requires ingress packets to be copied into a cache and egress packets to be examined for acknowledgements to evict those entries from the cache. The latter approach needs much less processing, but does require timers for each connection to determine whether or not to resend a packet to an end-station.

Network Address Translation (NAT)

Network Address Translation (NAT) [17] allows multiple end-stations to be represented by one IP address (the gateway's). This is used to alleviate the shortage of IPv4 address and also provides security for the end-stations behind the gateway. The end-stations use IP addresses reserved for local subnets. The gateway maps the TCP/IP address and port of all requests to outside hosts to addresses and ports of the gateway. To perform NAT, an edge router/gateway needs to carry out the following functions:

- Store a local (non-routable) IP address and port number in an address translation table
- Modify packet's source port with a port number that matches the information stored in the address translation table. The translation table now has a mapping of the computer's non-routable IP address and port number along with the router's IP address.
- When a packet comes back from the destination node, the router checks the destination port on the packet. It then searches the address translation table for the local TCP port and IP address the packet belongs to. It updates the destination address and destination port of the packet and sends it to that computer.
- Since the NAT router now has the computer's source address and source port saved to the address translation table, it will continue to use that same gateway port number for the duration of the connection. A timer is reset each time the router accesses an entry in the table. If the entry is not accessed again before the timer expires, the entry is removed from the table.

Web “Switch”

A web switch is a device that uses information from higher layer protocols to make lower layer routing/switching decisions. A web switch provides a point of contact on the Internet that clients can access. Clients' requests are then selectively directed to the most appropriate server. This makes web switches useful for a variety of applications including distributed web caching and load balancing.

A TCP/IP server load balancer can be implemented using a web switch as follows [18]:

1. The web switch recognizes a new TCP connection by identifying a TCP SYN (session initiation) packet.
2. The switch determines the most appropriate server to handle this request and binds the new TCP session to the IP address of that server in a table.
3. For all ingress packets belong to a new session, the web switch substitutes the switch's TCP port, IP address, and MAC address for the server's. This makes the packets appear as if they were directed to the server by the client.
4. Likewise, all egress packets that pass through the web switch are altered such that it appears as if the web switch responded the client's request.
5. When the web switch recognizes a TCP FIN (session teardown) packet, the web switch removes the session-server binding from its table.

A variety of network equipment can be realized by changing how a web switch directs traffic. For example, a URL load balancer can be implemented by examining all packets for HTTP requests and forwarding them to an appropriate server based on their content. The request-server mapping may be either static (e.g. have all images served by a separate machine, for example) or dynamic (e.g. based on real-time server load data). A logical extension of this can be used for web caches as well.

Quality of Service Related Applications

Many QoS-related applications have come to light recently. While most of these applications have a large control-plane component, they impact the data-plane operations as well. In this section, we examine the data-plane processing of three applications: usage-based accounting, Differentiated Services, and Integrated Services.

Usage-based Accounting

Collecting network usage information pertaining to flows and sessions is essential for billing and network analysis applications. Highly granular policy rules are required for associating bandwidth usage to specific users, selected applications, and distinct content. For example, it is necessary to track the download and bandwidth usage of a client when accessing a server, using RTSP (Real Time Transport Protocol) to play the latest rock video clip. The main functions and corresponding packet processing tasks include [44]:

- Recognize session initiation for specific server: layer 3 IP addresses and layer 4 port numbers
- Monitor login session to identify user name: layer 5-7 extraction of login information
- Recognize RTSP session and associate with user: layer 4 port numbers, layer 5 key words detection

- Identify desired file name (e.g. video clip) to download: layer 5-7 extraction of file name and matching to users and programs policy tables
- Recognize download session and associate with user: layer 4 port numbers and layer 5-7 key words detection

Differentiated Services (DiffServ)

Differentiated Services (DiffServ) enables a wide variety of services and provisioning policies, either end-to-end or within a particular set of networks [19]. When traffic enters a DiffServ network, it is classified and possibly conditioned at the boundary of the network and assigned a DiffServ codepoint (DSCP). The DSCP reflects a per-hop behavior (PHB) at each node. RFC 2474 [20] defines how DSCP overrides the previous definition of the Type of Service (ToS) field in IPv4. A logical view of the DiffServ node is shown in Figure 15 and the elements are described below [20]:

- Classifier
Classification can be performed based on the DSCP field only (a Behavior Aggregate (BA) classifier), or based on multiple header fields, like source/destination IP/TCP address/port (Multifield (MF) classifier).
- Meter
Traffic meters measure the temporal properties of the stream of packets selected by a classifier against a traffic profile specified in a Traffic Conditioning Agreement (TCA). A meter passes state information to other conditioning functions that may trigger a particular action for each packet that is either in- or out-of-profile (to some extent).
- Marker
Packet markers set/reset the DSCP of a packet to a particular codepoint, adding the marked packet to a particular differentiated service behavior aggregate. The marker may be configured to mark all packets that are steered to it with a single codepoint or mark a packet with one of a set of codepoints used to select a PHB, according to the state of a meter.
- Shaper/Dropper
Shapers delay some or all of the packets in a traffic stream to bring the stream into compliance with a traffic profile. A shaper usually has a finite-size buffer, and packets may be discarded if there is insufficient buffer space to hold them.

Droppers discard some or all of the packets in a traffic stream to bring the stream into compliance with a traffic profile. This process is known as "policing" the stream. Note that a dropper can be implemented as a special case of a shaper by setting the shaper buffer size to zero (or a few) packets.

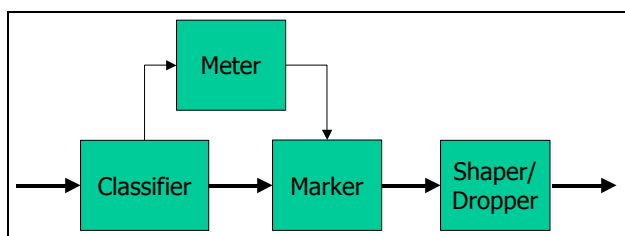


Figure 15. Logical view of a DiffServ node.

Integrated Services (IntServ)

Integrated Services (IntServ) uses traditional datagrams, but allows sources and receivers to exchange messages that establish additional packet classification and forwarding state on each node along the path [21]. There are three main components to providing IntServ (the first two must be performed at wirespeed):

- Classifier: map incoming packet to some class; all packets in the same class receive the same treatment. Classification can be based on the header fields (like the classification field or flow ID field) or looking deeper into the packet to identify application-layer fields (like video).
- Packet scheduler: control forwarding of different packet streams using a set of queues. The main function of a packet scheduler is to reorder the output queue using an algorithm like weighted fair queuing (WFQ) or round robin.
- Admission control: decision whether a new flow can be granted the requested QoS. This is implemented with a control-plane reservation setup protocol like RSVP.

Others applications

We have outlined many applications in this section. We believe this provides a good sample of applications to be implemented on network processors. However, the following applications are of importance to network processors also:

- VoIP gateway
- Internetworking – POS, already have AAL5
- Peer-to-peer
- SSL
- Different queuing algorithms
- Ethernet
- SONET

2.2 Kernels

To examine how the applications outlined in the previous section map to network processors, we decompose the applications into their computational kernels. These kernels broadly fall into six different categories: pattern matching, lookup, computation, data manipulation, queue management, and control processing.

While these application kernels are the basic operations for a particular packet, single packet processing makes poor utilization of network processor hardware (e.g. stalling on memory access). To reach the required data rates, a network processing system must simultaneously process multiple packets. We visit the many approaches network processors employ to solve this problem in Section 4.

Pattern matching

Pattern matching is the process of matching bits in packet fields (either header or payload). This kernel has two inputs: a regular expression pattern and the packet field. It outputs a

Boolean value reflecting whether or not the packet field matches the input pattern. Common algorithms used for this kernel are calculation and lookup tables.

Lookup

The lookup kernel is the actual action of looking up data based on a key. It is mostly used in conjunction with pattern matching to find a specific entry in a table. The data structures and algorithms used are dependent on the type of lookup (one-to-one or many-to-one) required and the size of the key. For ATM and MPLS, this field is quite small and the mapping is one-to-one, so often only one lookup is required. However, for IPv4 and IPv6 routing, the large address field and longest prefix matching (LPM) requirement make it impossible to find the destination address in one memory access. Therefore, trees are used to efficiently store the address table and multiple lookups are required.

Computation

The types of computation required for packet processing vary widely. To support IPSec, encryption, decryption, and authentication algorithms need to be applied over an entire packet. Most protocols require a checksum or CRC value be computed. Often, this value just needs to be updated (not recalculated) based on changes to header fields. Network equipment that implement protocols which support fragmentation (and reassembly) of PDUs require computation to determine if all fragments of a particular PDU have arrived.

Data manipulation

We consider any function that modifies a packet header to be data manipulation. For example, in IPv4 routing, the Time To Live (TTL) field must be decremented by one each hop. Additional instances of data manipulation include adding tags, header fields, and replacing fields. Other examples in this space include segmentation, reassembly, and fragmentation.

Queue management

Queue management is the scheduling and storage of ingress and egress PDUs. This includes coordination with fabric interfaces and elements of the network processor that need to access packets. The queue management kernel is responsible for enforcing dropping and traffic shaping policies and storing of packets for packet assembly, segmentation, and many Quality of Service (QoS) applications.

Control processing

Control processing encompasses a number of different tasks that don't need to be performed at wire speed, like exceptions, table updates, details of TCP protocols, and statistics gathering. While statistics are gathered on a per packet basis, this function is often executed in the background using polling or interrupt-driven approaches. Gathering this data requires examining incoming data and incrementing counters.

2.3 Summary

The following tables summarize how the applications described above can be decomposed into the five main types of processing.

	ATM		VLAN	MPLS
	Switching	AAL5		
Pattern Matching	VCI (8 bits) and VPI (16 bits)		MAC address (48 bits), IP subnet (8-24 bits)	MPLS label (20 bits)
Lookup	VCI (8 bits) and VPI (16 bits)	CPI field (for reassembly)	MAC address (48 bits) or IP subnet (8-24 bits)	MPLS label (20 bits)
Computation		check if all fragments have arrived (reassembly)	checksum	
Data Manipulation	TTL adjustment, update VCI/VPI	creating new packets and populating fields (segmentation), extracting payloads and combining them (reassembly)	insert unique identifier in VLAN field, checksum	popping or pushing labels to packet, TTL decrement
Queue Management	incoming cell management	organizing fragments (reassembly)		
Control Processing	VCI/VPI table update, path/circuit setup		VLAN group updates	path table updates

Table 1. Applications and their kernels (part 1).

	IPv4			IPv6	IPSec
	Routing	Frag/Reas	ARP		
Pattern Matching	version & address check	check flags		IP address (128 bits)	verify address (32/128 bits), port (16 bits), mode
Lookup	IP address (32 bits)	find other packets of a fragment (12-bit fragment offset)	IP address (32 bits)	IP address (128 bits), flow label (20 bits)	find SA (based on destination address (32/128 bits) and SPI (32 bits))
Computation	checksum	testing if all fragments have arrived (reassembly)			authenticator algorithm, decryption/encryption, checksum
Data Manipulation	insert next hop, TTL adjustment, checksum	creating new packets and populating fields (fragmentation), extracting payloads and combining them (reassembly)	insert MAC address	TTL adjustment	clear "mutable" fields, pad packet, insert headers and trailers, checksum
Queue Management	incoming packet management, to implement QoS/CoS	organizing fragments (reassembly)	queue packets waiting for MAC address	incoming packet management, based on flow label for QoS	organize fragments (reassembly)
Control Processing	routing table updates, group control		manage lookup table	routing table updates, RSVP	update SA based on connections

Table 2. Applications and their kernels (part 2).

	Wireless TCP/IP	NAT	Web Switch	Usage-based Accounting	DiffServ	IntServ
Pattern Matching	IP address (32/128 bits) and TCP port (16 bits)	IP address (32/128 bits) and TCP port (16 bits)	many different fields, depending on application (e.g. Ethernet, TCP/IP, HTTP requests, FTP)	many different fields, depending on application (e.g. Ethernet, TCP/IP, HTTP requests, FTP)	DSCP field (6 bits), Ethernet MAC, IP address, TCP port	Flow ID (20 bits), Ethernet MAC, IP address, TCP port, L5-7 field
Lookup	IP address (32/128 bits) and TCP port (16 bits)	IP address (32/128 bits) and TCP port (16 bits)	many different fields, depending on application (e.g. Ethernet, TCP/IP, HTTP requests, FTP)	many different fields, depending on application (e.g. Ethernet, TCP/IP, HTTP requests, FTP)	Ethernet MAC, IP address, TCP port (if multi- field classification)	Flow ID (20 bits), Ethernet MAC, IP address, TCP port
Computation		checksum update	checksum update		moving averages	
Data Manipulation	packet creation for packets to wireless endstations	TCP/IP address/port replacement	field rewriting, based on application		remarking DSCP fields	
Queue Management					RED, WFQ, traffic shapers for different classes	RED, WFQ, traffic shapers for different classes
Control Processing	timers for each connection to determine retransmission	timers to evict old entries in mapping table	update mappings	counters (packets & bandwidth), meters	meters	schedulers, resource reservation protocols

Table 3. Applications and their kernels (part 3).

3 Network Processors

This chapter surveys some of the current network processors in the market. We use the following outline to characterize the network processors identified in this section:

1. Intended data rate and applications
Before analyzing any technical details of a network processor, we must first understand its target uses. We present these target uses as data rates, types of layer processing (using OSI stack model), and plane of processing (management, control, or data).
2. Architecture
In this subsection, we describe the major elements of the device—processing elements (type, number, and layout of), description of specialized hardware (co-processors and functional units), on-chip communication scheme (bandwidth, layout, special features), and memory (amount and type of).
3. Interfaces
A network processor is not a stand-alone system, but a portion of a larger system. It is important to understand how it interfaces with other system components. This subsection describes the interfaces supported.
4. Programmability/Integrated Development Environment (IDE)/OS support
Software support is key part of network processors as it defines the interface for users of the device. In this subsection, we discuss the network processor's programming interface ("programming model"), included libraries, integrated development environment, and operating systems support.
5. Implementation (when available)
When available, we include implementation details of the device, like process technology, clock speed, die area, and availability.
6. Cost (when available)
We give the cost of the device whether available as a soft or hard-core.
7. Design wins (if applicable)
If this network processor has been used in any network equipment that has been made public, we list it in this subsection.

While many network processors are profiled here, there are numerous others for which we were unable to gather enough information about: Bay Microsystems, Entridia Corporation, IP Semiconductors A/S, Ishoni Networks, Navarro Networks, and Onex Communications.

3.1 Agere (PayloadPlus)

The Agere network processing solution consists of three separate chips: Fast Pattern Processor (FPP), Routing Switch Processor (RSP), and Agere System Interface (ASI). The main data pipeline is from the physical interface to the FPP to the RSP. The ASI is only used for exceptional cases and overall management. This solution is aimed at layer 2-4 processing and supports packet rates up to 2.5 Gbps [22].

Architecture

The system architecture consists of three chips: Fast Pattern Processor (FPP), Routing Switch Processor (RSP), and Agere System Interface (ASI). Figure 16 shows the PayloadPlus system and how it interfaces with the networking fabric, also note the main data path from the physical interface to the Fast Pattern Processor to the Routing Switch Processor and back to the fabric.

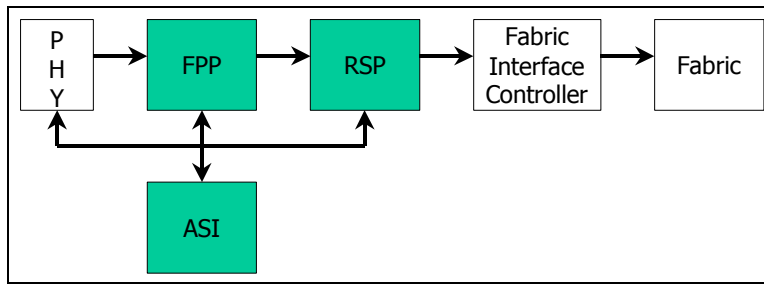


Figure 16. Agere PayloadPlus System.

Fast Pattern Processor (FPP)

The FPP performs pattern matching while receiving a frame or cell. The results of the pattern matching are passed to the RSP for packet manipulation or queuing [24] via a 32-bit POS-PHY Level 3 interface.

The FPP is a pipelined, multi-threaded processor with support for up to 64 threads. Each Packet Data Unit (PDU) that arrives from the UTOPIA bus is assigned to a new thread (context). Hardware support for fast context switching enables the FPP to process multiple PDUs in parallel. PDUs are processed in two passes: The first pass stores the PDU into an internal data format that consists of block data offsets and links blocks. The second pass processes the entire PDU, performing pattern matching and handing off to the downstream processor.

Figure 17 shows a block diagram of the FPP. The FPP includes an *Input Framer* that frames the input stream into 64-byte blocks and stores them in the *Data Buffer*. The *Context Memory* stores the blocks currently being processed in one of the 64 contexts. The *Pattern Processing Engine* performs pattern matching on PDUs. The *Checksum/CRC Engine* calculates their respective values for packets.

The FPP sends management frames to the ASI via the Management Path Interface (MPI). The Configuration Bus Interface (CBI) is used to configure the FPP and RSP. The PayloadPlus system supports a 32-bit UTOPIA Level 3/UTOPIA Level 2/POS-PHY Level 3 interface.

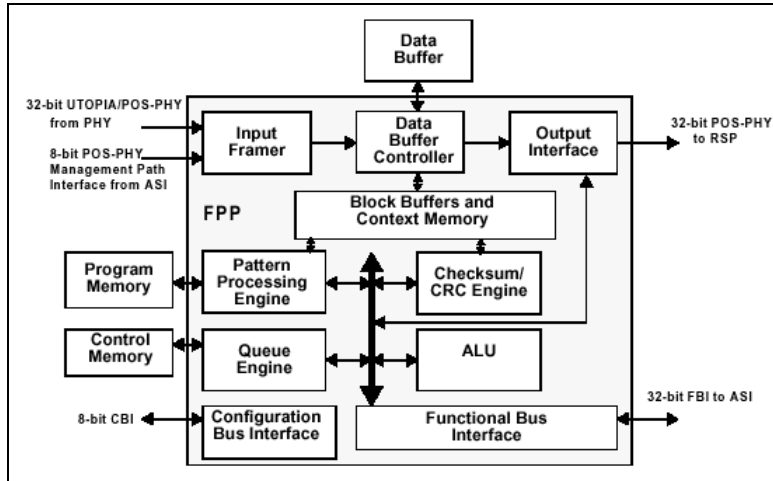


Figure 17. FPP Block Diagram [24].

The FPP uses a functional scripting language to specify protocol processing. The motivation behind this approach is similar to the motivation of SQL for databases – the programmer should only need to describe what to do, not how to do it. How this actually gets compiled remains a mystery. [23], [24], and [25] describe the motivation and advantages of this approach.

An *Application Code Library* is provided to support the IP protocol over ATM, Ethernet, and Frame Relay.

Routing Switch Processor (RSP)

The RSP takes classification data and protocol data units (PDUs) from the FPP and outputs PDUs to the fabric. It has four major functions: queuing, traffic management, traffic shaping, and packet modification [26].

The RSP receives packet handling “instructions” from the FPP and stores the PDU in SDRAM. Based on traffic management calculations, it either queues the PDU in one of 64k programmable queues or discards it. For each queue, QoS and CoS policies are used for traffic shaping. Once the PDU is chosen to be transmitted, it is fetched from SDRAM, modified, and transmitted.

There are 3 VLIW compute engines, each dedicated to a specific task:

- Traffic management compute engine: enforces discard policies and keeps queue statistics
- Traffic shaper compute engine: ensures QoS and CoS for each queue
- Stream editor compute engine: performs any necessary PDU modifications

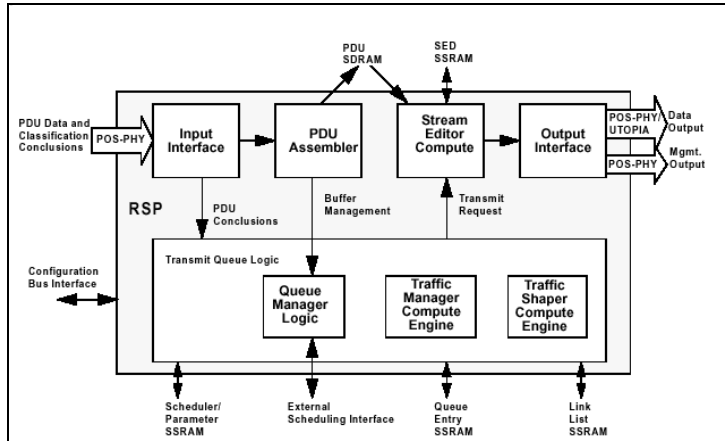


Figure 18. Architecture of the Agere Routing Switch Processor [26].

The RSP is programmed by configuring the ports and schedulers and defining the queues. There is a scripting language that is used to program the RSP as well as an API that “provides low-level access to the PayloadPlus chipset” [27].

Agere System Interface (ASI)

The main function of the ASI is to handle “slow path processing” – initialization, routing table updates, queue processing updates, exception handling, and statistics gathering [28].

There is a PCI interface for external management and a PC133 SDRAM for access to off-chip memory. The same scripting language used to program the RSP is used for the ASI.

Implementation

The FPP, RSP, and ASI are currently available in a 0.18 μ process with a typical power dissipation of 12W [22].

Cost

The cost for the PayloadPlus system is about \$750 [22].

3.2 Alchemy (Au1000)

The Alchemy Au1000 is best suited for access equipment.. However, the company claims its device is also targeted to edge routers and line cards. It is a low power MIPS core with a few new instructions and a wide variety of integrated peripheral support.

Architecture

The Au1000 [29] is based on a scalar 32-bit MIPS processor. The processor has a 5 stage pipeline optimized to reduce branch penalties. There is a 32x16 MAC (Multiply-Accumulate) that runs in parallel with the CPU pipeline. In addition, there are special instructions for conditional moves, counting leading ones/zeros, and prefetching memory.

With the core, there are two Ethernet controllers, an IrDA port, USB support, and four UARTs. There is a 16KB instruction and data cache. The system architecture of the AU1000 is shown in Figure 19.

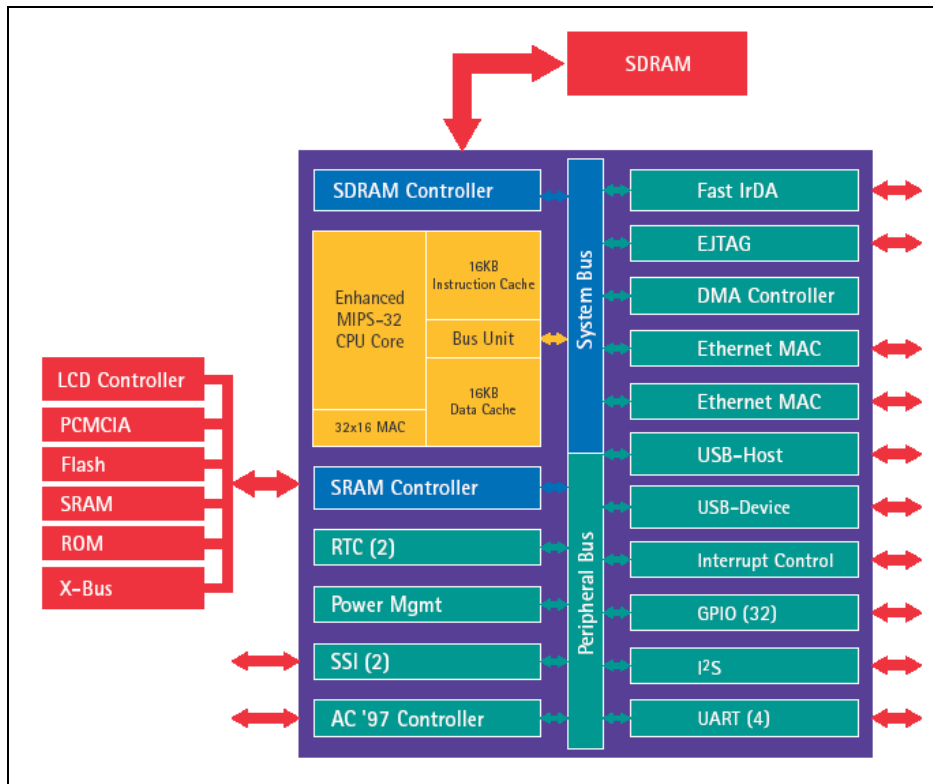


Figure 19. Alchemy's System Architecture [29].

Programmability

Since this device is based on a MIPS core, it can be programmed in C. The programmer can take advantage Alchemy's software development tools as well as various third-party tools. In addition, there is support for MS Windows CE, Linux, and VxWorks operating systems.

Implementation

The Au1000 is available as a soft core and can run at 266MHz, 400MHz, and 500MHz. At these speeds, the core consumes <300mW, 500mW, and 900mW, respectively.

3.3 Applied Micro Circuits, formerly MMC Networks (nP7xxx)

Applied Micro Circuits' nP7 network processor family [30] is built upon the EPIF-200 packet processor [31]. With six EPIF-200s on a single chip, the nP7XXX can support 10Gbps packet rates. The nP family is aimed at processing layers 2-7.

Architecture

The EPIF-200 is a 64-bit processor with a network-optimized instruction set and zero-overhead task switching among 8 threads. There is programmable Policy Engine for packet classification and a Search Engine for layer 2 VLAN bridging and layer 3 longest prefix match lookup. The Packet Transform Engines perform all the necessary packet

manipulation. In addition, there are Statistics Engines that collect RMON-compliant data. The EPIF-200 is designed to seamlessly work with other EPIF-200, MMC switch chips, and nP family co-processors. It also features an on-chip Fast Ethernet MAC. The macro-architecture of the EPIF-200 is shown in Figure 20.

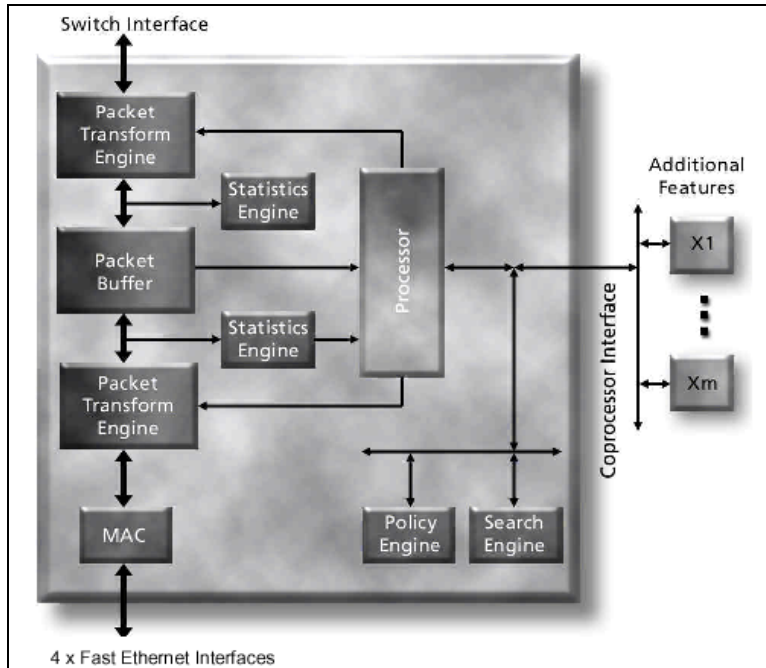


Figure 20. Applied Micro Circuits' EPIF-200 Network Processor [30].

Programmability

APPLIED MICRO CIRCUITS has simplified the multi-processor programming model by letting the programmer think of the device as a single logical CPU. In addition, they provide a C/C++ compiler, assembler, and debugger.

Implementation

The nPxxx family is implemented in a 0.18μ process and typically consumes 4W [22].

Cost

Applied Micro Circuits' np7xxx will cost around \$115 [22].

3.4 Bay Microsystems

Details of Bay Microsystems' device are a bit weak; the company has not released much information. All information here is from [32]. They claim deep packet analysis (layers 3-7) at 10Gbps.

Architecture

Bay's network processor uses a VLIW architecture with commodity DRAM. There is an "ultrawide-bus" standard that used standard DRAM. In addition, their engine is pipelined and superscalar.

Programmability

Not available

Implementation

Bay Microsystems' chip will run at 166MHz.

3.5 BRECIS Communications (MSP5000)

BRECIS Communications is developing a Multi-Service ProcessorTM aimed at connecting the enterprise to the edge of the network. Their MSP family of processors handles voice traffic from PBXs and data traffic from the network core to a LAN (layers 2-3) [33][34]. The heart of their solution lies in the Multi-Service Bus Architecture, which connects the main processing elements and inherently supports QoS at the bus transaction level. Their top-of-the-line product, the MSP5000, can simultaneously support 8-24 G711 voice channels, 4-10 G729 voice channels, and a 52Mbps data rate.

Architecture

BRECIS' MSP network processor consists of three processors, two DSPs (LSI ZSP400s) for packet and voice processing and a MIPS R4KM processor for control-plane operations, connected by a high bandwidth bus. The ZSP400 is a 4-issue superscalar processor with 80Kbytes of on-chip instruction and data memory. In addition, the DSP aimed at voice processing has a co-processor for ADPCM (adaptive pulse code modulation) acceleration, while the packet processor has a co-processor for efficient CRC generation. The ZSP400s each run at 160MHz. The control processor runs at 180MHz and has instruction and data caches of 16Kbytes. The MSP also has a shared co-processor for security-related operations (e.g. MD-5 authentication, 3DES encryption) that is connected directly to the Multi-Service Bus. Figure 21 shows a diagram of the system architecture.

The Multi-Service Bus Architecture has a 3.2Gbps peak bandwidth and connects the major devices of the network processor, including the DSPs, control processor, security co-processor, Ethernet MACs, and peripheral sub-system. The bus supports simultaneous transactions and dynamic priority switching among three priority levels. The bus interface for each sub-system (one for each processor) consists of a packet classifier and three packet queues, which map directly to the three types of traffic handled by this device (voice, data, and control). This enables efficient implementation of Quality of Service applications, a key to supporting voice and data on the same processor. Each sub-system also contains a context-aware DMA engine that offloads packet transferring duties from the data processors.

The MSP5000 has two 10/100 Ethernet MAC interfaces and a UTOPIA interface to support the networking side. For telephony support, the MSP5000 has dual TDM interfaces (each supporting 128 channels).

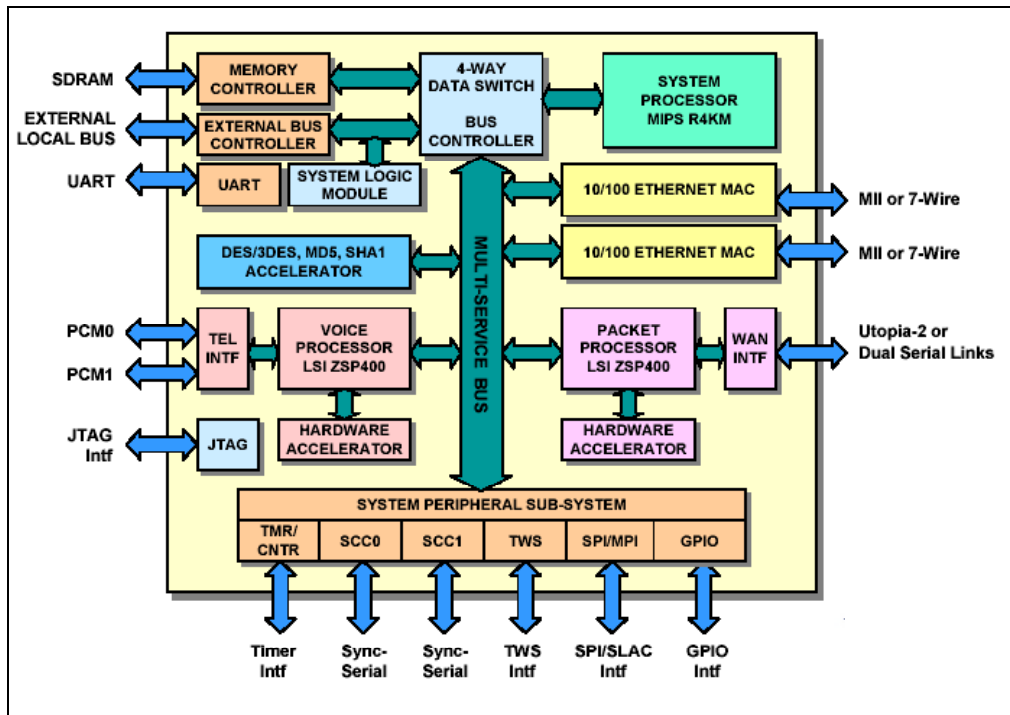


Figure 21. BRECIS Communications' MSP5000 [34].

Programming

BRECIS provides APIs for the application-specific engines, like the security co-processor and packet queues in the bus interface. Third party tool chains can be used to program the LSI ZSP400 and MIPS R4KM. In addition, they provide firmware for common networking applications (ATM AAL0/2/5, ATM SAR, Frame Relay encapsulation, and VoIP). The MSP family supports VxWorks, Linux, and BSD operating systems.

Implementation

The MSP5000 has been implemented in a 0.18 μ process technology. With 3.3Mbits of SRAM, it is approximately 27mm on a side and consumes 2W of power. It will be sampling in Summer 2001 [33].

Cost

The MSP5000 will cost less than \$50 if purchased in large volume [33].

3.6 Broadcom, formerly SiByte (Mercurian SB-1250)

The Broadcom Mercurian SB-1250 primarily consists of two 64-bit MIPS cores (Broadcom's own SB-1 cores), three Gigabit Ethernet MACs, and a 256-bit wide bus [35]. The processors don't have any special instructions or hardware for packet processing as most NPUs do. Instead of targeting data-plane operations, Broadcom is focusing on control-plane operations. They claim performance numbers of up to 2.5 Gbps and are aimed at processing layers 3-7.

Architecture

The SB-1250 has 2 64-bit MIPS CPUs (SB-1) running at up to 1 GHz. The SB-1s can execute 4 instructions per cycle (2 load/store, 2 ALU operations). They have a 9-stage integer ALU pipeline and a 12-stage floating-point pipeline [36][37]. Each processor has a 32kb L1 cache and the two cores share a 4-way associative 512kb L-2 cache. The architecture of the SB-1250 is shown in Figure 22.

The SB-1250 also includes 3 on-chip Ethernet MACs and 2 packet FIFOs. Their proprietary ZBbus (a 256-bit bus that runs at half of the processor speed) connects the major components of the chip.

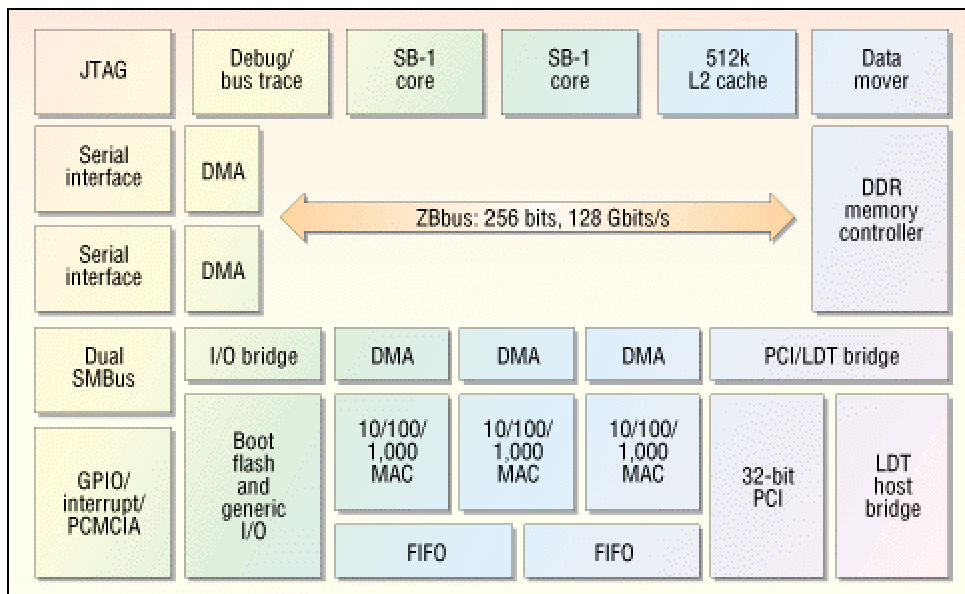


Figure 22. Broadcom's Mercurian Architecture [37].

Programmability

Since the SB-1 cores are MIPS-based and do not have any special instructions, the Mercurian uses the standard Gnu C/C++ tool chain with support for application specific extensions. It also has operating system support for FreeBSD, Linux, and VxWorks.

Implementation

The SB-1 CPU core is $\sim 25\text{mm}^2$ in a 0.15μ process. It runs at 1GHz and consumes only $\sim 2.5\text{W}$ [38]. Samples of the SB-1 are currently available.

3.7 Cisco (PXF/Toaster 2)

The Cisco PXF is an internal product for Cisco edge routers [39] [66]. Details are a bit sketchy, but it's a network processor that's used in routers, like the Cisco 10000 Edge Service Router (ESR). The PXF is intended to perform only layer 3 data path calculations. There is a separate route processor that handles network management tasks.

Architecture

The PXF consists of a pair of ICs, each comprised of 16 processors arranged in 4 pipelines. When used together, the pair of PXFs results in a 4x8 systolic array. Each of the 32 processors is a 2-issue VLIW with special instructions for packet processing. Each processor has an independent memory and each column of processors has access to its own separate memory (off-chip). Each of the 8 stages in the pipeline is responsible for a different packet forwarding function. Figure 23 shows an example of how the PXF could be used. They claim that the allocation of features to microprocessors is flexible, but it is unclear how that's possible since specialized hardware is likely used to accelerate these calculations.

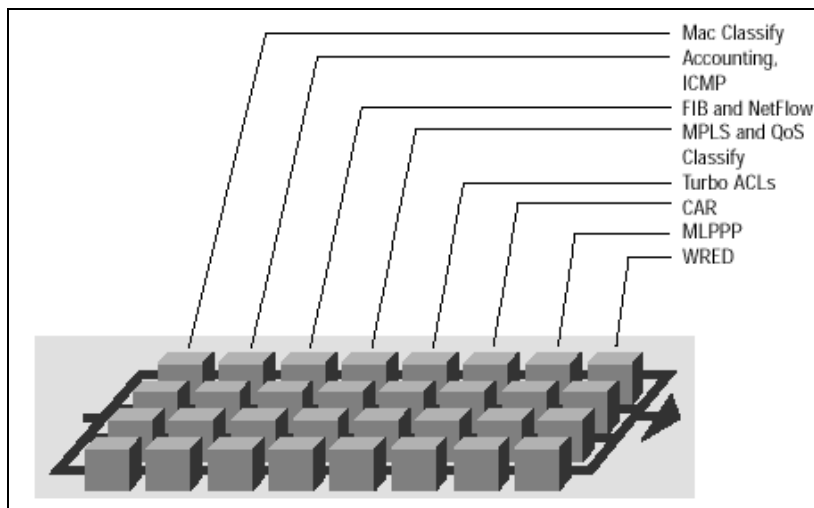


Figure 23. Example use of Cisco's PXF NP [39].

Programmability

The Connected Components Corp. has recently written a C-compiler for the PXF. In addition, the PXF supports Cisco's internal operating system, IOS.

Implementation

A version of the PXF is currently used in Cisco routers.

3.8 ClearSpeed, formerly PixelFusion

ClearSpeed's network processor is aimed at 40Gbps data rates for core routers, metro line cards, and edge routers. Their architecture consists of a multiple Multi-Threaded Array Processors (MTAPs) and shared co-processors connected by a high bandwidth bus. An MTAP consists of multiple 8-bit Processing Elements that all execute the same code. Since ClearSpeed is shipping their device as a soft core, many of the parameters are configurable (e.g. number of MTAPs, number of Processing Elements per MTAP).

Architecture

An MTAP processor mostly consists of multiple (100's to 1000's) of 8-bit Processing Elements. An MTAP's main function is to control the execution of its Processing Elements

and data delivery to/from other MTAPs. Since all Processing Elements execute a common instruction stream, an MTAP's central instruction fetch unit is shared among all the Processing Elements. An MTAP is capable of supporting up to 32 simultaneous threads.

A Processing Element is an 8-bit ALU with a small register file (configurable from 16-64 bytes) and a small amount of packet memory (1-16Kbytes). It also has memory controllers that are responsible for loading a packet (or partial packet) into its own memory. Since each Processing Element of an MTAP executes the same code, the execution of Processing Elements is based on the packet data currently loaded in the packet memory.

The ClearSpeed platform gives users the ability to add co-processors to accelerate common tasks. These user-defined co-processors are simply attached to the ClearConnect bus and accessed like any other MTAP. One such co-processor that ClearSpeed provides is a Table Lookup Engine (TLE). The TLE is able to perform multiple parallel searches on multiple tables with key sizes ranging from 32 bits to 128 bits. The TLE is composed of many parallel state machines, called Lookup Engines (LEs), and commodity memory. Since ClearSpeed sells their network processor as a soft core, the user is able to configure the size of the on-chip memory.

A scalable high-bandwidth bus called ClearConnect connects the MTAP processors, co-processors, and memory. The ClearConnect bus is composed of Nodes and T-Switches connected by Lanes. Nodes connect elements to the bus and function as repeaters, while T-Switches route bus requests to different Nodes. Nodes and T-Switches are connected by one or more Lanes, 50Gbps duplex links.

Programmability

ClearSpeed has an IDE in alpha release that includes a C compiler, assembler, debugger, and profiler. In addition, they have an application development kit that includes a visual tool for designing wire speed applications and a reference library of common networking functions.

Implementation

ClearSpeed's network processor will be available 1H03 as a soft core running at 400MHz. In a 0.13 μ process technology, it is 180mm² to 295 mm², depending on the number of PEs per MTAP and amount of on-chip memory.

3.9 Clearwater Networks, formerly XStream Logic Devices (CNP810SP)

Clearwater Networks applies simultaneous multi-threading to network routing [41]. Their CNP810SP processor issues 10 instructions per cycle and can simultaneously execute 8 threads. It is targeted at layers 4-7 processing for edge devices (edge routers, web switches, SSL accelerators) at packet rates of 10Gbps [40].

Architecture

Clearwater Networks uses simultaneous multi-threading, which is a hardware implementation of software multi-threading. Each thread has its own resources (register file,

PC, instruction prefetch buffer) and can conceptually be thought of as a separate superscalar processor [41].

Currently, Clearwater supports 8 simultaneous threads; there are 8 instruction queues, register files, and arithmetic units. The arithmetic functional units include some special instructions for network processing. In addition, there are two address generation units, which makes the CNP810SP a 10-issue machine. Clearwater uses superscalar techniques to dynamically determine how threads use these issue slots. There is an on-chip 64kB instruction and data cache and the ALU is a 9-stage pipeline.

The Packet Management Unit (PMU) handles packet I/O, leaving the CPU for deep packet classification. The PMU reads packets from the network interface, classifies them, and stores them in hardware-managed arrival queues. It also writes packets out to the network interface. The PMU has some control over the processor, as it can load contexts and context registers without processor intervention via the Register Transfer Unit (RTU). In addition, the PMU supports 24 global masks and 8 masks per thread for key extraction. Figure 24 shows the macro-architecture of Clearwater's network processor.

The CNP810SP has a dual ported on-chip packet memory of 256 Kbytes called the PacketCache. The structure of this memory allows for efficient implementation of packet manipulation, header/packet growth (e.g. MPLS over IP/Ethernet shim header), and packet memory allocation.

A high bandwidth interconnect, called XPress Switch connects the processor core, PMU, PacketCache, and peripherals. It has a peak throughput of 225Gbps. The CNP810SP supports numerous interfaces, including dual SPI-3, SPI-4 (Phase I), 64-bit PCI-X, and 2 serial ports.

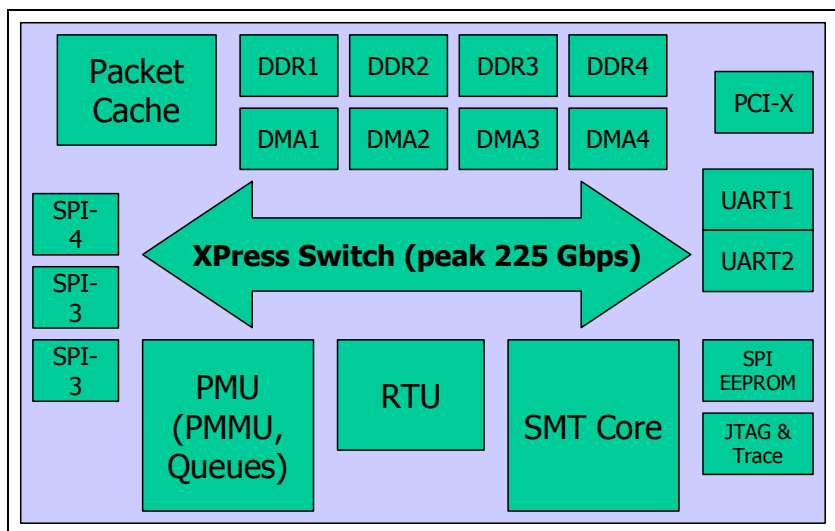


Figure 24. Macro-Architecture of Clearwater Networks' CNP810SP Network Processor.

Programmability

Green Hills is currently developing a C compiler for Clearwater Networks. Since simultaneous multi-threading makes each thread appear like a superscalar machine, the programmer can think of the device as 8 different superscalar processors. In addition, the programmer has some control over the resource sharing among those processors.

Implementation

The CNP810SP has been implemented in a 0.15 μ process. Running at 300MHz, it consumes 12W. Clearwater's network processor will be available as a soft core in 4Q2001.

3.10 Cognigine

Cognigine is one of the only network processors to use reconfigurable logic. Their Variable Instruction Set Computer (VISCTM) allows the execution units to be dynamically reconfigured. It is aimed at layer 2-7 processing at 10Gbps data rates [42].

Architecture

Cognigine's network processor is a distributed multi-processor machine – it has 16 processing elements, or Reconfigurable Communications Units (RCUs) connected by a crosspoint switch, called Routing Switch Fabric (RSF). Each RCU has four parallel execution units that operate on a 64-bit wide data path. The execution units are dynamically configurable by VISC instructions. A VISC instruction determines the major characteristics of an instruction, including operand sizes, operand routing, base operation, and predicates. The VISC instructions are stored in a Dictionary and decoded during the first stage of the pipeline. The RCU has a five-stage pipeline and has hardware support for four threads. Figure 25 shows the architecture of an RCU.

Each RCU has an associated RSF connector that serves as the RCU's interface to the RSF. This connector helps distribute arbitration of the fabric and schedule transactions. The RSF connects the RCUs in a hierarchical manner – a crossbar is used to connect groups of four RCUs and another RSF crossbar is then used to connect four groups of RCUs. The hierarchical nature of the communication fabric makes this solution scalable to a large number of RCUs. The RSF supports split transactions for hiding communication latency and is accessed by the RCUs via a memory map.

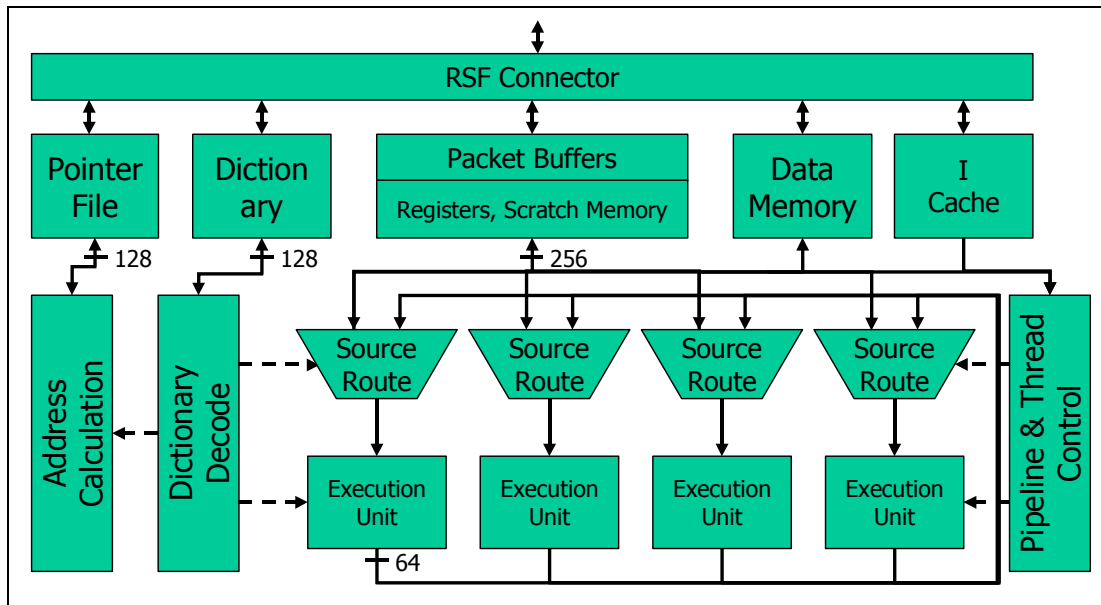


Figure 25. Cognigine's RCU Architecture.

Programmability

Cognigine provides a C/C++ compiler, assembler, and debugger for their network processor. They claim their tools can automatically determine VISC instructions from C/C++ application code. We doubt this can be done well, given the current state of compiler technology. Perhaps their tools are able to find some candidate VISC instructions, but to make efficient use of their architecture, programmers will likely have to determine and use their own VISC instructions, much like users of Tensilica's environment. Cognigine also provides an application level configuration tool that allows programmers to stitch together common networking elements. They also provide an applications library of common layer 2-7 functions.

Implementation

Cognigine's network processor has been manufactured in a 0.18 μ process running at 200MHz. It will be available in December 2001.

3.11 Conexant, formerly Maker (MXT4400 Traffic Stream Processor)

Conexant's Traffic Stream Processor (TSP) is aimed at Layer 2 processing – internetworking (AAL SAR, MPLS), buffer management, congestion control, bandwidth management, CRC & FCR error checking, traffic shaping. It can support packet rates up to 2.5Gbps [43].

Architecture

The core of the TSP is a 32-bit RISC processor (Octave) optimized for traffic stream processing. It has specialized instructions for internetworking and traffic management. In addition, the Octave processor efficiently dispatches parallel hardware operations and minimizes context-switching overhead by switching in the background. The Channel

Descriptor Look-up Engine examines packet headers, while the Packet/Command Engine maps traffic streams to processes on the Octave core. The Traffic Scheduling System handles controls protocols for bandwidth reservation and scheduling. The TSP supports UTOPIA 2 and has a 32-bit PCI bus interface.

Programmability

The development kit includes a C compiler, debugging tools, as well as simulation and analysis tools. Their PortMaker software provides a modular software architecture on top of the TSP.

Implementation

The TSP runs at 125 MHz and has a maximum power dissipation of 4.2W.

3.12 EZchip (NP-1)

EZchip uses specialized processors for different tasks required for network processing [44] [45]. These specialized processors, or Task Optimized Processors (TOPs), are superscalar processors arranged in a pipelined fashion. The NP-1 is designed for Layer 2-7 packet processing at 10Gbps. There is an interface for a separate control processor to handle control-plane operations.

Architecture

The EZchip NP-1 has many Task Optimized Processors (TOPs), each with their own customized instruction set and data path. These TOPs are arranged in a pipelined fashion (see Figure 26). There are four types of TOPs:

- TOPparse: identifies and extracts various packet headers and protocols
- TOPsearch: performs lookups at different levels (layer 2-7)
- TOPresolve: assign packet to appropriate queue and/or port
- TOPmodify: modifies packet contents

In addition, they claim to have patent-pending algorithms for leveraging embedded memory to search external memory to support line rates of 10Gbps. These algorithms and associated data structures enable long and variable-length string searching. Further details of their approach are not available.

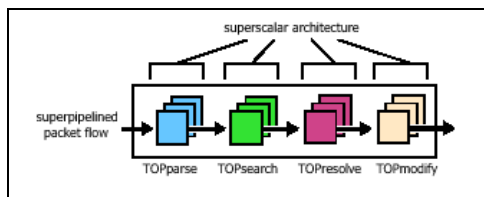


Figure 26. EZChip's NP-1 Architecture [45].

The NP-1 supports either eight 1 Gigabit Ethernet interfaces, one 10 Gigabit Ethernet interface, or one OC-192 interface.

Programmability

EZchip has a software development environment for the NP-1 that includes an assembler, debugger, and simulator [46]. A high-level language compiler is planned for future designs. In addition, they ship a library of common switching and routing applications. The cost for the software development environment is \$20,000 plus a 15% annual maintenance fee.

Implementation

The NP-1 should be available in August 2001 [47]. They recently announced a partnership with IBM, who will manufacture prototypes of the NP-1 in 0.18 μ and production parts in 0.13 μ [48].

Design Wins

Avaya plans to use the NP-1 in their next generation routing switches [49].

3.13 IBM (PowerNP)

The IBM PowerNP is multi-processor solution with 16 protocol processors, 7 specialized co-processors, and a PowerPC core. It supports Packet over SONET (POS) and Gigabit Ethernet at 2.5Gbps and is targeted for layer 2-5 processing.

Architecture

IBM's network processor consists of the Embedded Processor Complex (EPC), special frame processing hardware, and peripheral interfaces. The EPC has a PowerPC core and 16 programmable protocol processors (that make up the Embedded Processor Complex) [50]. Each pair of protocol processors shares a hardware co-processor to accelerate tree searching and frame manipulation. Each protocol processor has a 3-stage pipeline. Two of the protocol processors are specialized – one for “guided frames” (special Ethernet frames that allow one processor to communicate with other network processing devices) and one for building lookup tables. The seven co-processors execute the following functions:

- Data store: interfaces frame buffer to provide DMA capability
- Checksum: calculates header checksums
- Enqueue: interfaces with the Completion Unit to enqueue frames to the switch and target port queues
- Interface: provides all protocol processors access to internal registers, counters, and memory
- String Copy: enables efficient data movement within the EPC
- Counter: manages counter updates for the protocol processors
- Policy: examines flow control information and checks for conformance with preallocated bandwidth

Each Protocol Processor has an instruction memory of 8kb. There are various internal control memories sprinkled about the chip ranging from 8kb to 32kb. Figure 27 shows the macro-architecture of the PowerNP and Figure 28 illustrates the architecture of the Embedded Processor Core.

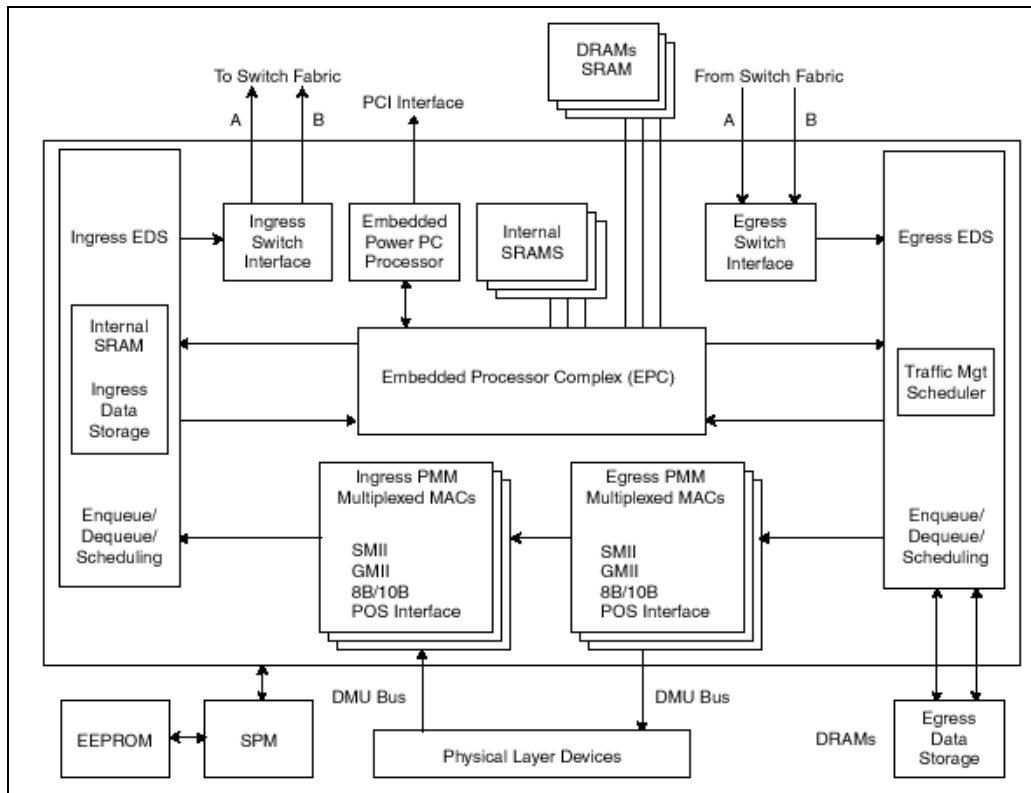


Figure 27. IBM's Network Processor [50].

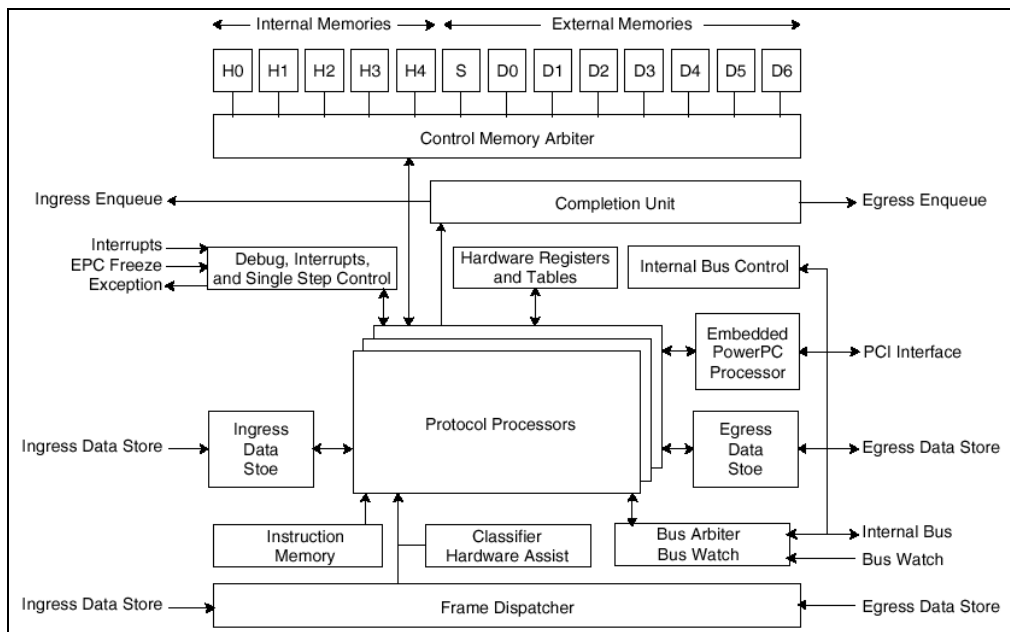


Figure 28. Embedded Processor Complex Architecture [50].

A large part of the packet processing actually occurs outside the EPC. An ingress frame first comes into the Ingress Physical MAC Multiplexer (PMM), which validates the frame (CRC check) and stores it in a buffer. It then passes part of the frame to the Protocol Processor

for frame lookups. The Classifier Hardware Assist helps identify the frame format, which is used by the Protocol Processor to perform lookups. The Tree Search Engine (TSE) aids in the search process by performing lookups in tables that reside in the Control Memory. The Control Memory Arbiter is used to manage multiple memory lookups from the different protocol processors. Once the lookup is completed, Ingress Switch Interface performs the necessary frame alterations (like adding a tag). The Completion Unit ensures the correct frame ordering before the frames are put back onto the switch fabric. Incoming egress frames are handled in the reverse manner (see Figure 29).

The PowerNP has on-chip support for 4 Gigabit Ethernet ports, 40 Fast Ethernet MACs, and Packet over SONET (POS) on-chip.

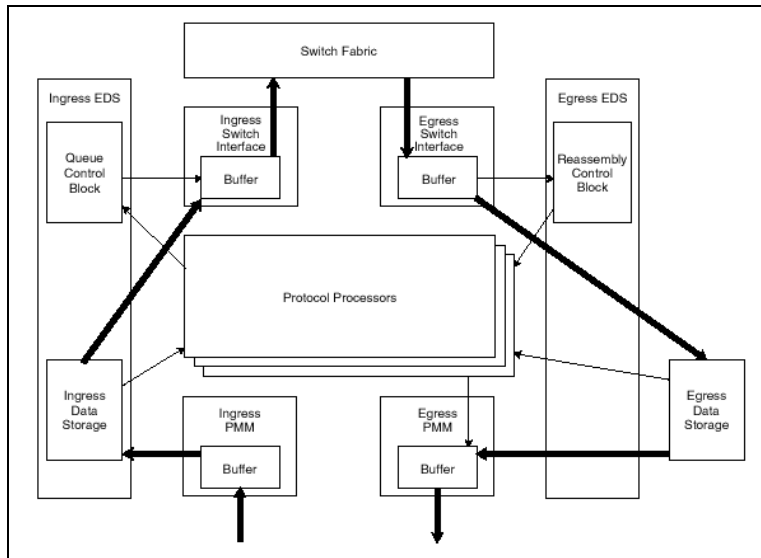


Figure 29. Ingress and Egress Frame Flow [50].

Programmability

The Code Development Suite includes a picocode assembler, debugger, and system simulator.

Implementation

The PowerNP is implemented in a 0.18 μ process, consumes about 20W, and runs at 133MHz [22].

Design Wins

Alcatel and Asante Technologies will use IBM's PowerNP devices in their IP core router and switch products [51].

3.14 Intel, formerly Level-One (IXP1200)

The Intel IXP1200 was one of the first network processors on the scene. Oddly enough, the design was actually done by DEC, which Intel acquired. The IXP1200 is mostly meant for layer 2-4 processing and can support a packet rate for 2.5Mpackets/s. As with other devices,

higher layers can be supported by connecting external processors to the PCI interface. The IXP consists of 6 “micro-engines” and a StrongARM controller. The micro-engines have hardware support for up to 4 threads each. In addition, there is special hardware to perform hash functions, queuing, and single cycle shifting and rotating.

Architecture

The IXP1200 consists of six programmable micro-engines and a 200 MHz StrongARM that coordinates system activities. The IX bus, a 64-bit bus, provides high bandwidth connectivity to the micro-engines, StrongArm, memory, and off-chip devices like a MAC device or another IXP1200. A PCI-bus interface allows integration with an external control processor.

The micro-engines perform all the packet processing tasks. They have hardware support (i.e. zero-overhead swapping) for four threads each, for a grand total of 24 threads on the chip. While the four threads on a micro-engine share a common register file, there is a software policy that splits the register file into four parts (one for each thread). This makes it possible for the device to swap contexts in a single cycle. The micro-engines also have special instructions for packet processing, like find first bit set, barrel shift, and extract byte/word.

In addition to the micro-engines, the IXP has some special hardware units that aid in packet processing. There is a programmable hash engine and specialized queues that are shared by all the micro-engines and the StrongARM. Receive (Transmit) FIFOs provide an interface to MAC-layer devices by reading (writing) packets into (out of) on-chip queues that can be accessed via the IX bus.

There is an on-chip data cache of 8KB, 16KB of instruction cache for the StrongARM, and 4kbyte of on-chip Scratchpad SRAM.

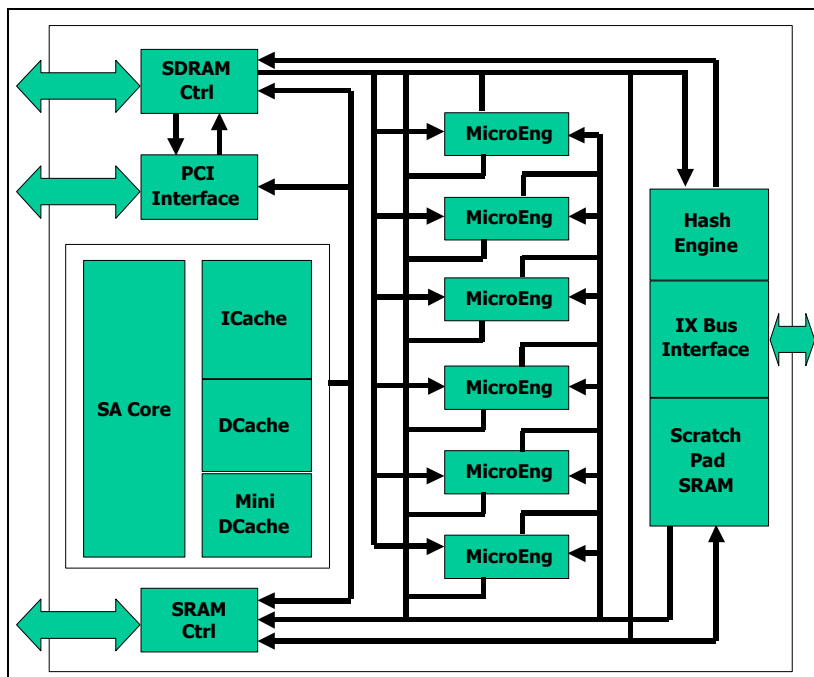


Figure 30. Intel's IXP1200 Architecture.

Programmability

Programming the IXP1200 is all done in macro-assembly, though they will release a C compiler in 2H01 [52]. Given that the 6 micro-engines running in parallel, programming the IXP proves to be a difficult task. This process is exacerbated by the assembly language including context switching and other unique features. However, their IDE (Integrated Development Environment) tremendously aids in programming the device. Their configurable simulation environment and visualizations clearly show all activities of the chip, making debugging much easier.

Implementation

The IXP is implemented in a 0.28 μ process, runs at 200 MHz, consumes about 5W and is available now [22].

Cost

The IXP1200 costs less than \$300 [22].

Design Wins

The IXP1200 is currently being used by Broadband Access Systems, Mayan Systems and NorthChurch [53]. Cloudshield Technologies is also using IXP1200s for their dedicated routing network device [54].

3.15 Lexra (NetVortex & NVP)

Lexra has two network processing products, the NetVortex and the NVP. The NetVortex uses multiple (up to 16) MIPS R3000 cores that are specialized for network processing. It is targeted for layer 2-4 processing at speeds of greater than 10 Gbps. Their second product, the NVP, is currently in development. It has an improved communication infrastructure and more co-processors to accelerate common networking tasks.

Architecture

NetVortex

The Lexra NetVortex strings together multiple LX8000 32-bit RISC network processor units (NPUs). Each NPU is a MIPS R-3000 core augmented with hardware support for single cycle context switch among 8 contexts – aside from PCs (program counters), there are separate register files for each context. The cores also have special instructions to speed up packet processing (e.g. ones complement add, insert and extract bit fields). They also have two level branch instructions for case statements often found in control-plane code [55] [56]. Lexra also has a unit to assist fetching packets from memory, called a Block Transfer Unit, that is accessible via the system bus. Since the NetVortex is available as a soft-core, the user is able to add co-processors and other engines. Figure 31 shows an example of this.

The NPUs are connected by a multi-channel DMA controller that transfers packets to/from local memories.

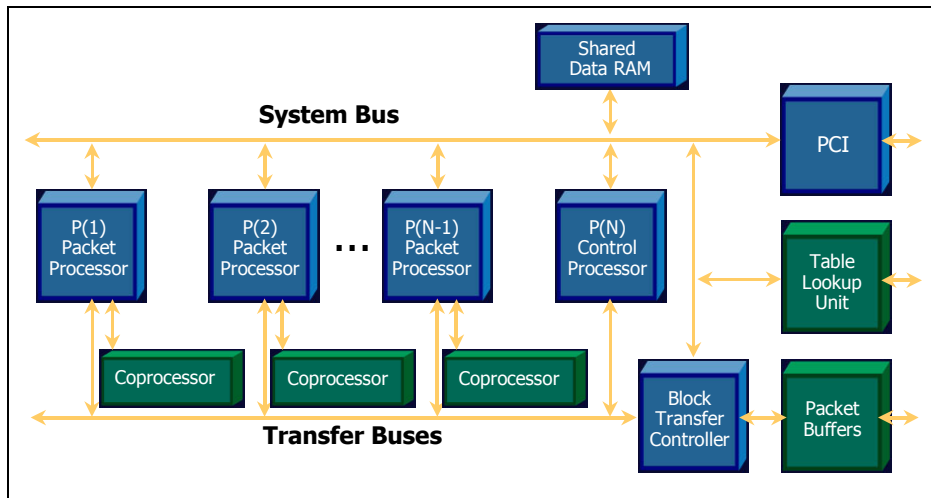


Figure 31. An Example Use of Lexra's NetVortex System Architecture [56].

NVP

The architecture of the NVP is similar to that of the NetVortex, however it has some key improvements. With the NVP, Lexra has removed the co-processors that were tied to individual packet processors and added a few shared co-processors for lookup, metering, and statistics [57]. Each of these has access to its own off-chip memory. The packet processors have a few more specialized instructions for network processing (checksum calculation, table lookup, hashing, and bit field extraction/manipulation). Each packet processor has 16Kbytes of instruction and data memory and supports eight contexts. A block transfer engine offloads the packet processors from packet I/O tasks by supporting block moves to/from shared memory. It also includes hardware management of 144 buffers.

In addition, a wide crossbar connects the co-processors and the 16 packet processors. This crossbar had a peak bandwidth of 270Gbps, supports internal queuing to prevent blocking, and is pipelined for greater throughput.

Programmability

Despite some minor architectural differences, the programmability for the NetVortex and NVP are very similar. While C is used for the standard portion of the MIPS processors, assembly code is required for Lexra-specific instructions. A graphical debugger with multi-processor and multi-thread support is also provided.

Implementation

NetVortex

The NetVortex is available as a synthesizable RTL macro. This will allow 1-16 packet processors running at 250 MHz in a 0.15 μ process [56].

A hard macro with 16 packet processors should be currently sampling; it runs at 450 MHz in a 0.15 μ process. It is estimated to be 64 mm² and consume 6.8W [56].

NVP

When implemented in a 0.13 μ process, the NVP is 134mm². Running at 420MHz, it consumes 12W. When released, the NVP will be available as a synthesizable RTL macro.

Cost

The standard up-front license fee for the NetVortex is \$645,000, plus per-chip royalties of \$1.00 to \$2.50 per core.

3.16 Motorola, formerly C-Port (C-5 DCP)

The Motorola C-5 DCP is a single chip multi-processor network processor [58] [59]. There are 16 *channel* processors (with 5 co-processors) and 1 general-purpose processor for system coordination. Each channel processor consists of a RISC core with two Serial Data Processors. The C-5 is targeted at layers 2-7 processing at 2.5Gbps rates.

Architecture

The C-5 DCP consists of 16 channel processors with 5 co-processors (executive processor, fabric processor, table lookup unit, queue management, buffer management). Each channel processor can be used individually, organized in banks to handle data streams in parallel, or organized serially with each processor handling a different task.

A channel processor consists of a RISC core plus two (one for send, one for receive) parallel Serial Data Processors (SDPs) that act as communication building blocks to talk to other channel processors. The RISC cores handle characterization and classification, policy enforcement, and traffic scheduling, while the SDPs handle programmable field parsing, header validation, extraction, insertion, deletion, CRC validation/calculation, framing and encoding/decoding.

There are 5 shared co-processors, each with a different function:

- executive processor: coordination with external processors
- fabric processor: for using multiple C-5's in a fabric
- table lookup unit: table lookup and update
- queue management: manage packet queues
- buffer management: fast, flexible memory management

In addition, there are three internal buses with an aggregate bandwidth of 60Gbps. The C-5 DCP supports Level 2/3 UTOPIA and PCI bus interfaces. The macro-architecture of the C-5 DCP is shown in Figure 19.

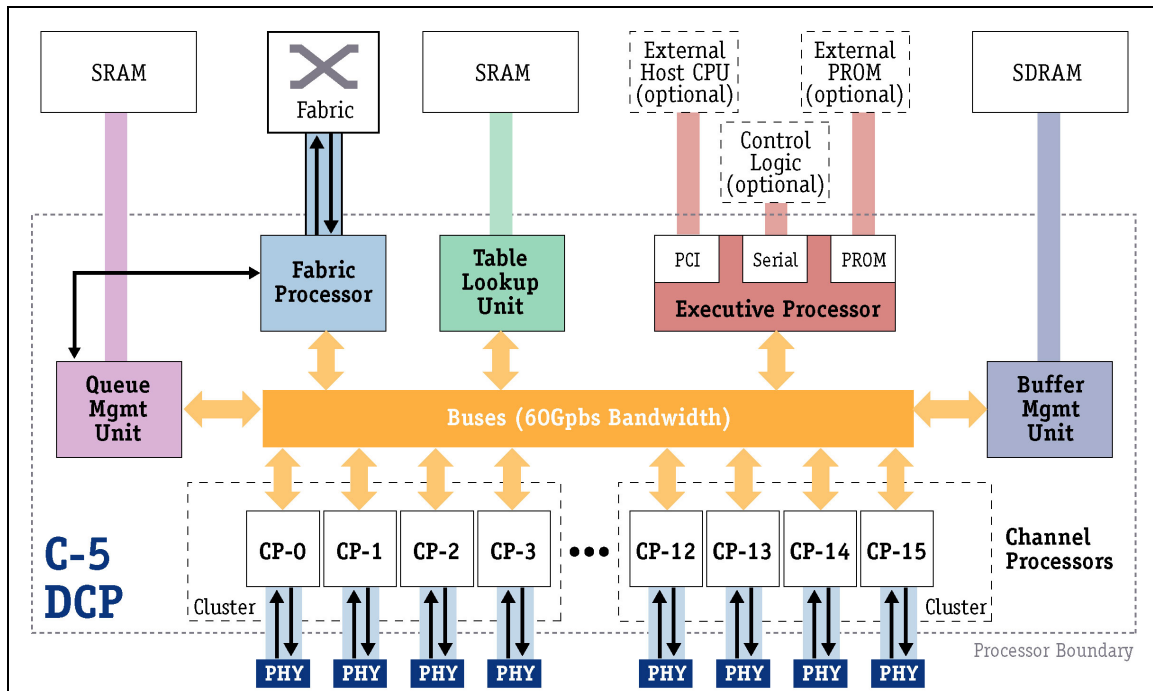


Figure 32. Motorola C-5 DCP Macro-architecture [60].

Programming

The C-5 DCP is C/C++ programmable and uses the C-Ware Communications Programming Interfaces (CPI) – an open set of standard interfaces that abstract “common network task building blocks” such as physical interface management, data forwarding, table lookups, buffer management, queuing operations, etc [60]. The CPI is mainly used to access the co-processors. There is also a C-Ware reference library for use in common applications [61].

Implementation

The C-5 is implemented in a 0.18 μ process and consumes 15W during typical operation. It is currently available.

Cost

The projected cost is \$400 [22].

Design Wins

Extreme Networks is using the Motorola C-5 in their BlackDiamond® 6800 series chassis for Packet over SONET (POS) applications. In April 2001, Atoga Systems announced they were using the C-5 in their Optical Application Router 5 (OAR 5), a router that uses software tunable lasers for on-demand bandwidth provisioning and dynamic optical scaling. Empirix is using the C-5 for their new network emulator that allows users to test IP networks in the face of jitter, delay, loss, duplication, and reordering at wire speeds.

3.17 PMC-Sierra, formerly Quantum Effect Devices

The PMC-Sierra RM7000 is more of a high-end digital signal processor than a network processor. It's a 64-bit MIPS-compatible superscalar microprocessor that has several specialized DSP instructions, a 256KB secondary cache, and a high-performance floating-point unit [62]. Since its architecture is not geared for network processing, we do not profile it. Like other MIPS-based processors, the RM7000 can take advantage of numerous 3rd party software development environments.

3.18 Vitesse, formerly SiTera (PRISM IQ2000)

The PRISM IQ2000 is a network processor that works with existing standard processors (from IDT, QED, and NEC) [63][64][65]. The standard processor does control-plane processing and system management, while the network co-processor does packet processing for classification, lookups, and QoS/CoS priority checking. The IQ2000 is aimed at aimed at layer 2-3 processing for edge routers and supports packet rates up to 2.5Gbps [66].

Architecture

The architecture of the IQ2000 consists of four 200MHz scalar RISC processor cores with co-processors for lookup, classification, packet order management, multi-cast support, DMA management, and context management. The RISC processors have an optimized instruction set for network operations. The co-processors and CPUs are arranged in a streaming fashion: As packet stream on to the chip, the Classification Engine classifies packets. The Order Management block assigns each packet to a thread on a particular CPU. After the packet has been processed, it is forwarded to the Queue Management co-processor, which queues the packet. The QoS engine handles packet priorities and transferring the packet to the output interface. The standard processor is used for route processing and system management. A block diagram of Vitesse's network processor is shown in Figure 33.

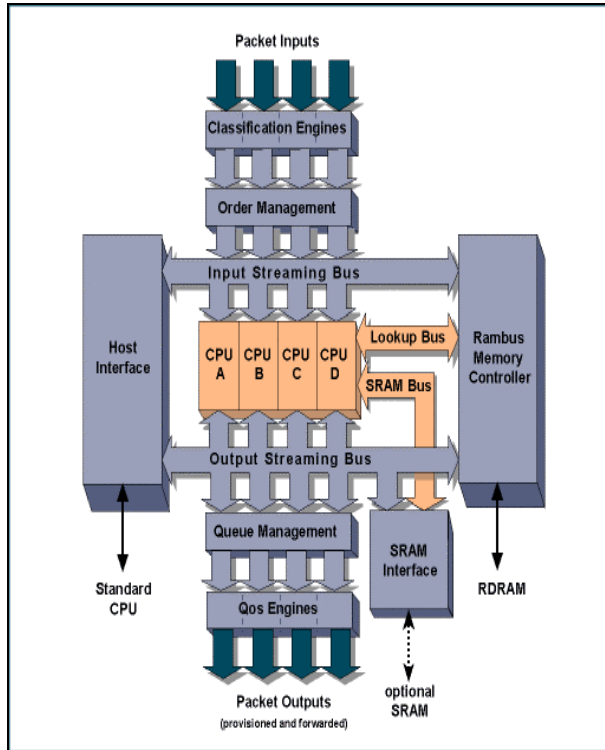


Figure 33. Vitesse's PRISM IQ2000 Architecture [64].

Programmability

Their software development environment uses standard high-level programming languages and development tools (GNU-based). The Vitesse development suite provides a graphical environment for system simulation and hardware-assisted debug, as well as an integrated environment for system regression and performance testing. In addition, there is network software library for common functions.

Implementation

The PRISM runs at 200 MHz and is implemented in a 0.25 μ process. It typically consumes 12W [22].

Cost

The PRISM IQ2000 sells for \$250 [22].

3.19 Xelerated Packet Devices (X40 & T40)

Xelerated Packet Devices provides a two chip solution to network processing – the X40 Packet Processor and the T40 Traffic Manager. Both chips are based on a packet-driven computation paradigm that Xelerated calls Packet Instruction Set Computing [67]. Xelerated's solution is geared for layer 2-7 processing at 10Gbps data rates.

Architecture

Packet Instruction Set Computing can be characterized as a programmable pipeline of processors where each pipeline stage executes a fixed function on a different packet. Each processor is composed of a classifier and an action block - the classifier identifies particular packets, while the action block alters or examines them. The action block (called a Packet Instruction Set Computer) is a processor with a specialized ISA for packet processing. Every clock cycle, a new packet enters and exits the pipeline, giving a deterministic per packet latency and throughput for a series of packet streams.

The X40 Packet Processor [68] is composed of 10 macro-pipeline stages. In addition to the classifier/PISC pairs, the X40 also includes 384k counters and 128k meters for making traffic metering and conditioning decisions. In addition there is a small internal CAM and an arbiter that controls access to an external CAM (for larger tables). The counters, meters, and CAMs are accessible to all pipeline stages. Figure 34 shows the micro-architecture of the X40.

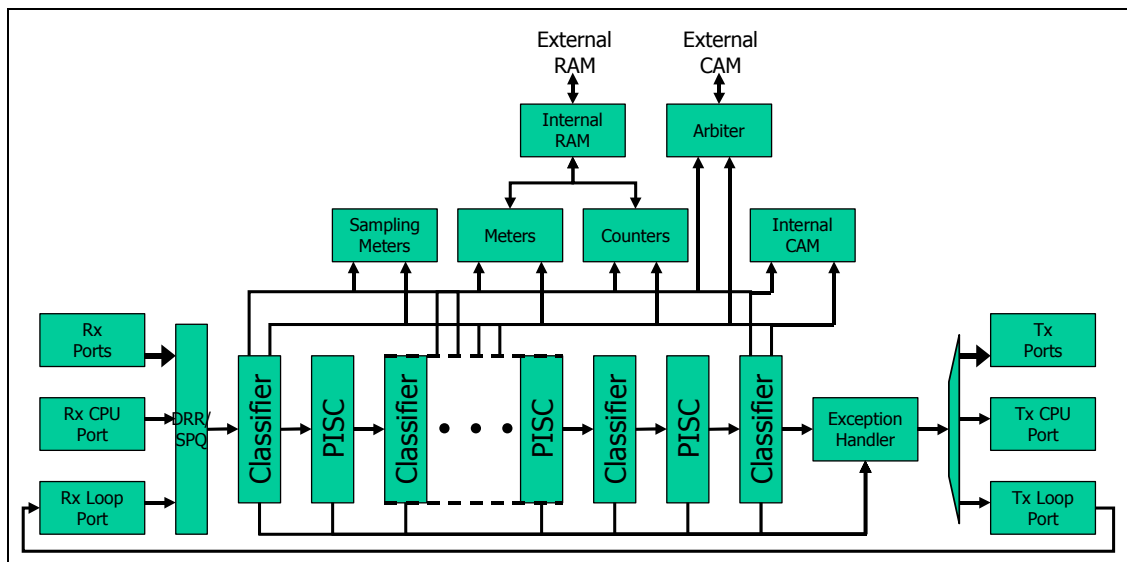


Figure 34. Macro-architecture of the X40.

Figure 35 shows an example configuration of the macro-pipeline for IP routing. For a single packet processing function that takes longer than one clock cycle to execute, multiple pipeline stages may be used (e.g. packet forwarding stage).

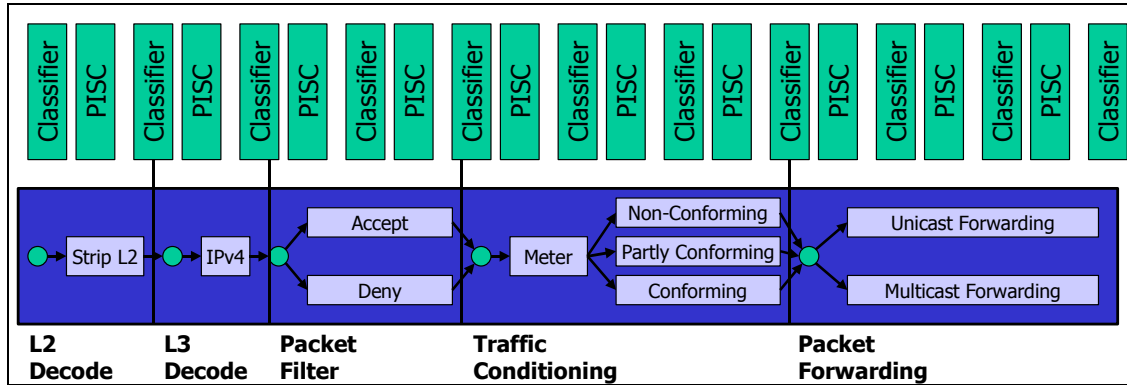


Figure 35. Example use of Xelerated's X40 Packet Processor.

The T40 Traffic Manager [69] is also based on the PISC paradigm, but includes more specialized hardware for traffic management. In particular, there is a classifier/PISC pair on ingress and egress of the pipeline with a Queue Engine in the middle. The Queue Engine is responsible for most of the traffic management tasks, like congestion control, traffic shaping, and fragmentation and reassembly. The T40 includes shared access to 64k queues, has hardware support for Weighted Random Early Detection (WRED) and fragmentation and reassembly, and has three levels of scheduling.

Programmability

To program the X40 and T40, users must program each of the pipeline stages separately, as each stage is executing a different program. Xelerated provides development tools for programming in an augmented version of C and simulating compiler code. In addition, they provide building blocks for implementing control-plane processing.

Implementation

The X40 and T40 have been implemented in a 0.13 μ process technology and will be available in April 2002.

3.20 Summary

Figure 36 shows the network processors profiled in this report mapped onto Figure 3. It is interesting to note that many of the original network processors (Applied Micro Circuits, Conexant, IBM, Intel, and Motorola) are architecturally similar: they are composed of one or more processing elements and a couple of co-processors for common networking applications. From this initial group, network processors architectures have spread out into a number of different solutions:

- General-purpose processor-like architectures (Alchemy, Broadcom)
- Dataflow processing architectures (Cisco, EZchip, and Xelerated Packet Devices)
- Simultaneous Multi-threading (Clearwater Networks)
- Digital Signal Processing (BRECIS Communications, PMC-Sierra)

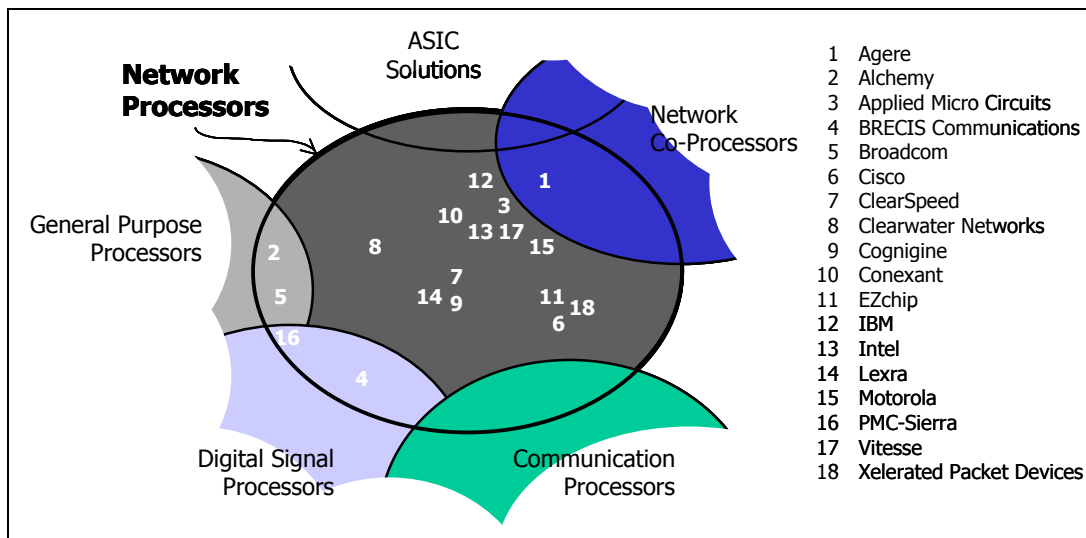


Figure 36. Varying Solutions of Network Processors.

Appendix A contains detailed summary tables of these network processors on many different axes: micro-architecture, architecture, software support, memory features, and physical implementation.

4 Analysis

In the last section, we introduced a number of network processors; in this section, we analyze their different approaches. In this section, we move from the technical specifications of network applications and capabilities of network processors to market segments. First, we describe the market segmentation occurring in this market to lay the foundation upon which to compare network processors. Then, we compare different architectural and programmability aspects of network processors. Lastly, we summarize the results of our analysis.

4.1 Market Segmentation

As the number of applications for network processors has grown, the market has begun to segment into three main network equipment areas: core, edge, and access. Each of these areas has different target applications and performance requirements. Core devices sit in the middle of the network. As a result, they are the most performance critical and least responsive to flexibility. Examples of these devices are gigabit and terabit routers. Edge devices sit in between the core network and access devices. Examples of edge devices include URL load balancers and firewalls. They are focused on medium-high data rates and higher layer processing, so a certain amount of flexibility is required. Access equipment provides various devices access to the network. Most of their computation relates to aggregating numerous traffic streams and forwarding them through the network. Examples of access devices include cable modem termination systems and base stations for wireless networks. Table 4 summarizes the characteristics of the three target markets of network processors.

	Performance	Flexibility	Examples
Core	High	Low	Gigabit/terabit router
Edge	Medium	Medium	URL load balancer
Access	Low	High	CMTS, wireless network basestation

Table 4. Characteristics of the 3 major NP markets.

Another consideration for network processors is the network processing functionality they perform: data-plane, control-plane, or management-plane. Each has different processing needs and requirements. Data-plane processing consists of forwarding packets or frames from the input ports to the output ports. This is the most performance hungry, as it must be executed at wire speed to avoid dropping packets. As a result, data-plane processing systems often take advantage of the packet independence by processing multiple packets in parallel. Control-plane processing refers to processing the control packets that aid network equipment in performing data-plane tasks. Examples of control-plane processing include routing table updates, ATM virtual circuit setup and teardown, and IPsec's Internet Key Exchange. These operations have little or no performance requirements and exhibit little data parallelism. As a result, they are often executed on a general-purpose processor. Management-plane operations refer to the processing of network management packets. Like control-plane processing, these operations have little or no performance requirements. Table 5 summarizes the salient characteristics of the different planes of computation.

	Performance Requirements	Data Parallelism	Examples
Data plane	High	High	Routing packets
Control plane	Low	Low	ATM VC setup/teardown
Management plane	Low	Low	SNMP processing

Table 5. Comparison of Network Processing Planes.

4.2 Architecture

In this section we compare and contrast the architectures of the various NPs described in the last section. We first present a timeline of network processor release dates. This is important for providing a reference to the different target markets and implementation details of NPs. Then, we examine their approaches to parallel processing, namely multiple processors and multiple issue. Third, we look at the specialized hardware employed to accelerate network processing, both at the co-processor level and at the functional unit level. Lastly, we examine techniques for hiding latency of various elements.

Timeline

Since many of the NPs profiled in this report have varying release dates, we first present a timeline of network processors. Because application requirements and process technology change rapidly, it is important to consider the release date when comparing architectural features of different devices. Figure 37 shows the timeline of network processor releases. The dates of releases indicate when devices began (or will begin) the sampling phase. On average, full-scale production occurs two to four quarters after sampling. Despite the many network processors profiled in this report, only seven are shipping – Agere, Applied Micro Circuits, Conexant, IBM, Intel, Motorola, and Vitesse. A large number of the NPs have yet to pass the testing phase (i.e. done sampling). This is important to note when comparing many metrics of network processors. For example, in Figure 39, the four highest MIPS devices are not in production yet; two of them will not begin sampling until the fourth quarter of this year.

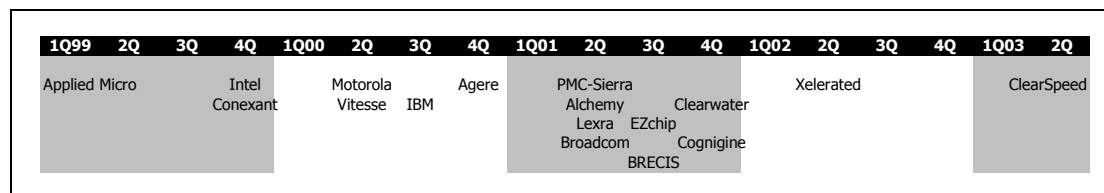


Figure 37. Timeline of Network Processor Releases.

Parallel Processing

Many architectures use parallel processing to increase the throughput of their device. This is enabled by the independent nature of traffic streams and increased per packet computation requirements of new applications. Two architectures for parallel processing are multiple processor and multiple issue.

Of the large number of NPs that use independently executing multiple processing elements (PEs), there are two prevalent configurations:

- Pipelined: each processor is designed for a particular packet processing task and communicates in a pipelined fashion
In this approach, inter-PE communication is very similar to data-flow processing – once a PE is finished processing a packet, it sends it to the next downstream element. Examples of this architectural style include Cisco’s PXF, Motorola’s C-5 DCP, and Xelerated Packet Devices.
- Parallel: each PE is performing similar functionality
This approach is commonly coupled with numerous co-processors to accelerate specific types of computation. Since these co-processors are shared across many PEs, an arbitration unit is often required. The Agere PayloadPlus, Intel IXP1200, IBM PowerNP, and Lexra NetVortex are examples of this type of macro-architecture.

At the processing element level, relatively few NP architects have embraced multiple issue architectures – those that issue multiple instructions per program counter (thread). The Agere Routing Switch Processor, Brecis’ MSP5000, and Cisco’s PXF use VLIW architectures; this allows them to take advantage of intra-thread instruction-level parallelism (ILP) at compile time by leveraging sophisticated compiler technology. Clearwater Networks takes another approach - they use a multiple issue superscalar architecture in which a hardware engine finds the available ILP at runtime. Cognigine also has multiple issue PEs (4-way), but they have a run-time configurable instruction set that defines data types, operations, and predicates.

To compare the different approaches to parallel processing, we attempt to characterize NPs by their computation power. To do this, we divide computational elements of NPs into two categories:

- Processing Elements (PEs): instruction set processors that decode their own instruction stream; and
- Functional Units (FUs): computational blocks that fit within the pipeline of a PE

Ideally, we would also include co-processors and special functional units (SFUs) as computational elements. This would allow the comparison to serve as a benchmark for network processors. However, the performance of co-processors and SFUs is impossible to evaluate without reference to a specific application. As a result, we separate the analysis of special hardware (see the next section) from the parallel processing comparison. For a detailed performance benchmarking effort of network processors, the reader is referred to Tsai’s work [70].

The diversity of the approaches to parallelism by different NPs is shown in Figure 38. By plotting issue width per PE versus number of PEs, we can graphically depict the trade-off various network processor architects have made between number of processing elements and number of functional units. Clearwater Networks, at one extreme, has a single PE with 10 issue slots, while EZchip has 64 scalar PEs. On this chart, we have also plotted iso-curves of issuing 8, 16, and 64 instructions per cycle. While the clock speed and specialized hardware employed by network processors are not represented in Figure 38, it does illustrate the trade-offs NPs have made between processing element and functional unit parallelism.

Figure 38 also illustrates the future scalability of various network processors. To handle increasing data rates, network processors will need to grow increasingly parallel. Current approaches by those in the lower right corner of Figure 38 (Alchemy, Conexant, PMC-Sierra) will not scale to higher data rate applications (like those required for the core market). Most likely, these devices will appear in access devices and low-end edge equipment.

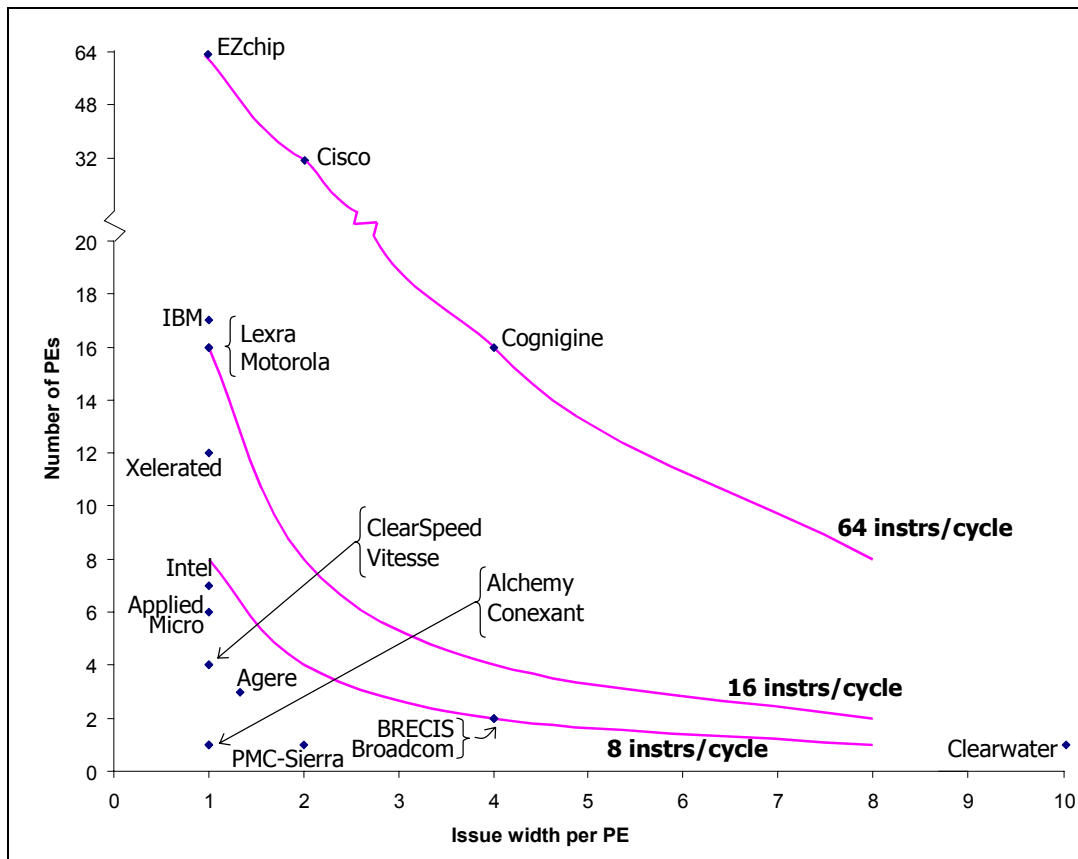


Figure 38. Issue Width per Processing Element Versus Number of Processing Elements.

To account for clock speed in this analysis, Figure 39 plots the number of processing elements versus MIPS (log scale). The MIPS of an NP is calculated by multiplying the number of PEs, issue rate per PE (number of FUs that can execute in parallel), and clock rate. Figure 39 shows a staggering two orders of magnitude range in MIPS among NPs. Note that some of the network processors are not represented due to the unavailability of their clock rate.

Figure 39 also shows how increased clock speed can make up for a lack of parallelism. For instance, Broadcom's network processor issues eight instructions per cycle, but runs at 1GHz, giving it 8000 MIPS. While Figure 39 does not account for specialized hardware, but it does hint at relative data rates different NPs can support. Therefore, caution must be taken in drawing too many conclusions from this analysis. For example, Vitesse's IQ2000 appears to be computationally underpowered, however, it has a variety of special hardware for packet processing, like classification, lookup, and queue management.

From Figure 39, we can draw some interesting conclusions. The parallelism and computational power provided by the NPs in the upper left corner (Cognigine, Lexra, Motorola, IBM) makes them good candidates for data processing for high-speed applications, like core networking equipment. The high processing element parallelism is necessary to simultaneously support multiple traffic streams. The high MIPS/low PE network processors like Broadcom and Clearwater Networks, are better suited for high data rate control-plane applications and higher layer data processing, both of which have limited amounts of task-level parallelism, but require a large amount of processing. The network processors in the lower left corner (Alchemy, Conexant, PMC-Sierra) have limited MIPS and processing element parallelism. These devices are best suited for access equipment, which has low data rate requirements. BRECIS' network processor is also geared toward access equipment, however, it is somewhat misrepresented in Figure 39 – about half of its computing resources (MIPS) are dedicated to telephony processing.

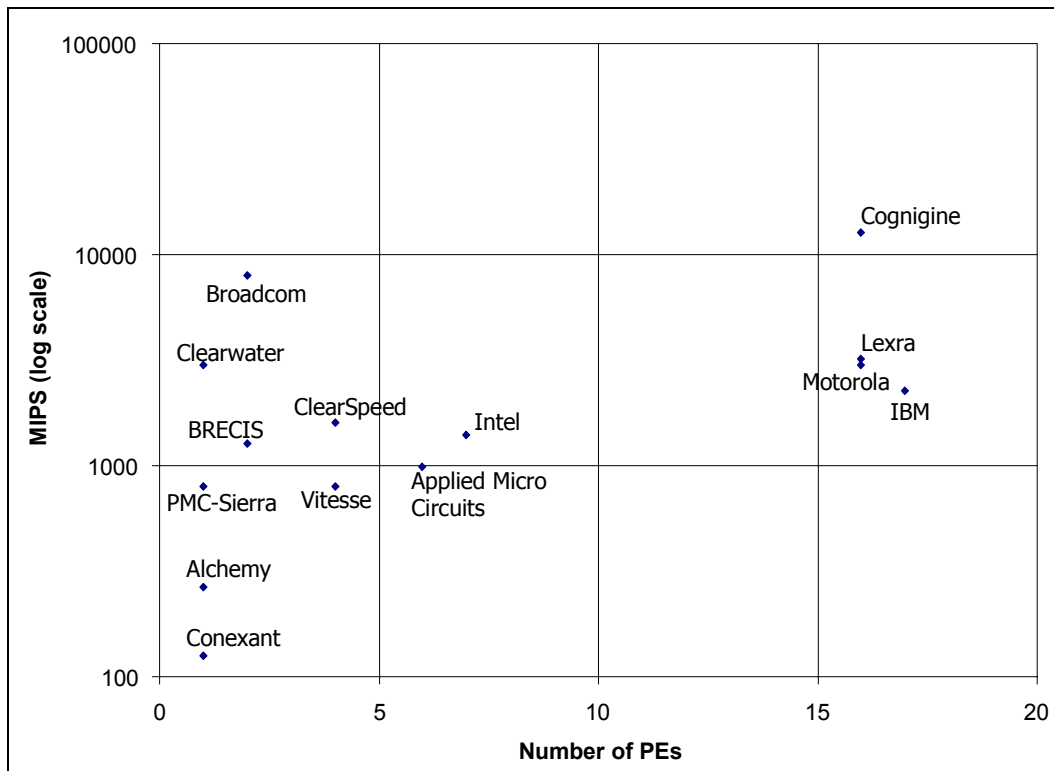


Figure 39. Number of Processing Elements Versus MIPS (log scale).

Special Hardware

Network processors have extensively used specialized hardware to accelerate common networking computational kernels. In this section, we examine the two major approaches used:

- Co-processor: a computational block that is triggered by a processing element (i.e. it does not have an instruction decode unit) and computes results asynchronously
- Special functional unit: a specialized computational block that computes a result within the pipeline stage of a processing element

In general, a co-processor is used for more complicated tasks, may store state, and may have direct access to memories and buses. On the other hand, a special functional unit is used for simpler operations, is usually stateless, and only writes back to registers. As a result of its increased complexity, a co-processor is more likely to be shared among multiple processing elements, while a special functional unit is not. Usually, interfaces to special functional units are instructions, while co-processors may be accessed via a memory map, special instructions, or bus. Given the orthogonality of these two approaches, a majority of NPs employ co-processors for some tasks and special functional units for others.

Integrated Co-processors

Of the 19 NPs profiled in this report, 14 of them have integrated co-processors for common networking tasks; nine NPs have more than one co-processor. Operations ideally suited for co-processor implementation are well defined, expensive and/or cumbersome to execute with an instruction set, and prohibitively expensive to implement as a special functional unit.

The most common integrated co-processors execute lookup and queue management functions. The functionality of lookup is clear – given a key, lookup a value in a mapping table. The main design parameter is the size of the key. For additional flexibility, some co-processors also support variable sized keys. Since lookup often references large memory blocks, it needs to operate asynchronously from a processing element. Common uses of lookup are for next hop addresses and for accessing connection state. The global aspect of lookup operations (with respect to the device) requires the co-processor be shared by all processing elements. Queue management is another good candidate for an integrated co-processor as the memory requirement for packet queues is large and queues are relatively cheap to implement in hardware. The small silicon overhead eliminates many memory read and write operations that would otherwise be required. Other common co-processors are for pattern matching, computing checksum/CRC fields, and encryption/authentication.

The functions of co-processors vary from algorithmic-dependent operations to entire kernels of network processing. For example, the Hash Engine in the Intel IXP1200 is only useful for lookup, if the algorithm employed requires hashing. For IP routing, the most common algorithms (trie table-based) do not use hash tables. Algorithm-specific co-processors limit the freedom of software implementation on network processors – the software programmer is forced to implement a task using a specific algorithm that can make use of the co-processor. While this may be desirable in some cases, the majority of customers will want to design their own algorithms for product differentiation reasons.

Special Functional Units

Most network processors have special functional units for common networking operations like pattern matching and bit manipulation. The computation required for these operations is cumbersome and error-prone to implement in software (with a standard instruction set), yet very easy to implement in hardware. For example, Intel's IXP1200 has an instruction to find the first bit set in a register in a single cycle. With a standard instruction set, this would be quite tedious and take numerous cycles. As with co-processor candidates, the transistor overhead is well worth the convenience and speedup.

Summary

Table 6 shows the types and applications of specialized hardware employed by various network processors. We measure different network processors by the six kernels identified in Section 2.2: pattern matching, lookup, computation, data manipulation, queue management, and control processing. We use shades of gray to give a rough indication of the amount of specialized hardware on the level of specialized hardware used by each network processor. In addition to functional units and co-processors, we also include application-specific bus and memory features and entire processors dedicated to networking kernels.

This analysis shows the diversity among different network processors. For example, IBM and Motorola have co-processors for most or all packet-processing kernels, while Cognigine relies solely on their reconfigurable functional units. EZchip has entire processors devoted to pattern matching, lookup, data manipulation, and queue management. On the other hand, PMC-Sierra's network processor has no specialized hardware; Broadcom and Alchemy have very little. A number of processors have interesting mixes of functional units, memory features, co-processors, and processors for various tasks. Agere's PayloadPlus system uses a special processor for pattern matching and data manipulation, a co-processor for checksum/CRC computation, and has memory features for queue management. Vitesse and Xelerated Packet Devices shy away from special bus or memory features and simply use a mix of co-processors and functional units. Intel and Lexra also include special memory and bus features and have a dedicated processor for the control-plane.

	pattern matching	lookup	computation	data manipulation	queue management	control processing
Agere (PayloadPlus)						
Alchemy						
Applied Micro Circuits (nP)						
BRECIS Communications (MSP5000)						
Broadcom (Mercurian SB-1250)						
Cisco (PXF/Toaster 2)						
ClearSpeed						
Clearwater Networks (CNP810SP)						
Cognigine						
Conexant (CX27470 Traffic Stream Processor)						
EZchip (NP-1)						
IBM (PowerNP)						
Intel (IXP1200)						
Lexra (NetVortex & NVP)						
Motorola (C-5 DCP)						
PMC-Sierra						
Vitesse (PRISM IQ2000)						
Xelerated Packet Devices (X40 & T40)						

Legend

Functional Unit	
Bus/memory features	
Co-processor	
Processor	

Table 6. Specialized Hardware Employed by Network Processors.

Appendix B provides an in-depth analysis of how networking applications map to network processor architectures.

Hiding Latency

Hiding latencies of various operations is a key aspect to efficiently using the hardware of a network processor. There are three ways NPs hide latency: multi-threading, memory prefetching, and split transaction buses. Multi-threading, by far the most common approach, is used to efficiently multiplex a processing element's hardware. The stalls associated with memory access are well known to waste many valuable cycles. Multi-threading allows the hardware to be used for processing other streams while another thread waits for a memory access (or a co-processor or another thread). Without dedicated hardware support, the cost of operating system multi-threading would dominate computation time, since the entire state of the machine would need to be stored and a new one loaded. As a result, many NPs have separate register banks for different threads and hardware units to schedule threads and swap them in one cycle. Clearwater Networks takes a slightly different approach – they have eight threads executing in parallel on the same processing element (which can issue 10 instructions per cycle). In addition, their processing element employs superscalar techniques to dynamically determine the available instruction-level parallelism and functional unit usage.

Figure 40 shows a chart of the number of threads per processing element different network processors support. At one extreme, Agere's PayloadPlus Fast Pattern Processor supports 64 simultaneous threads; at the other, we have six single-threaded network processors. Not surprisingly, a majority of the single-threaded NPs are either targeting the access market or control-plane processing. All of the multi-threaded architectures, except for Agere and Clearwater, have multiple processing elements and support multiple threads per processing element. This implies there are numerous (up to 128) threads running simultaneously on these network processors, which has serious implications on the difficulty of programming these devices.

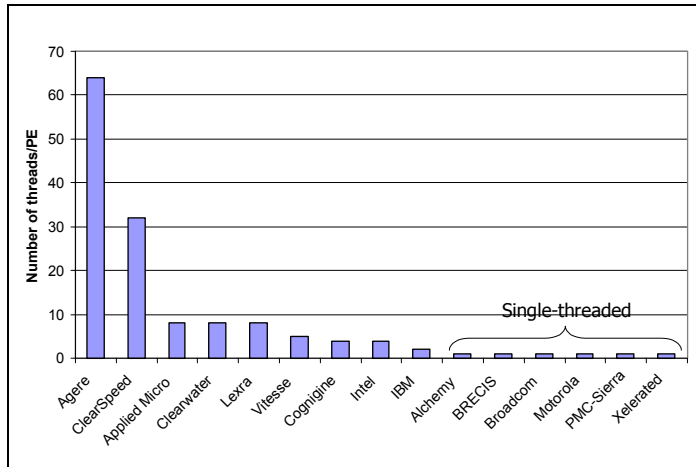


Figure 40. Comparison of Multiple Thread Support among Network Processors.

In some cases, the latency of a long memory access can be hidden by prefetching – accessing the memory location well before it is needed. For many streaming operations, like packet processing, this is not possible, but for control-plane operations, like routing table updates, this is an efficient way to hide memory access latency without the additional register or thread scheduling cost of a multi-threaded approach.

Another source of latency on multi-processor systems is communication across the bus. Increased integration in network processing systems will increase the communication time between processing elements, co-processors, memory, and physical and data link devices. Buses that support split transactions help hide this latency in the same way prefetching helps hide memory access latency, by splitting the issuance and completion of a bus request.

As memory and processor speeds continue to diverge and as communication architectures increase in complexity due to improved integration, the importance of hiding memory access time and communication latency will become more important. Pipelined multi-PE architectures are an example of this trend. By limiting the inter-PE communication, they are able to decrease communication latency. New architectural techniques to reduce this latency will also emerge.

4.3 Programmability

While parallel processing, specialized hardware, and hiding latency are important to executing applications efficiently, programming use these features is paramount to a

successful system implementation. We discuss three major topics related to the programmability of NPs: programming model, integrated development environment (IDE), and operating systems (OS) interaction.

Programming Model

The programming model for network processors is a difficult problem. Many NPs are composed of multiple processors, which has traditionally been a difficult programming problem. The specialized hardware present on these devices exacerbates the issue. NP companies have attacked the problem in various ways.

Clearwater Networks claims their “dynamic multistreaming” (i.e. simultaneous multi-threading) paradigm is natural to think about, as it is similar to programming on top of Windows or UNIX. Viewed from another angle, their processor can be thought of eight independent processors. The hardware schedules threads and their resource usage, but for fine-grain control, the programmer is able to guide the scheduler.

Agere has an interesting solution to programming their device. They use a declarative language for pattern matching, with patterns and associated actions, much like SQL [23]. While a scripting language raises the level programmer’s level of abstraction, it is unclear if the mapping of the program to the hardware is visible to the programmer. If it’s not, it will be difficult to improve the performance of an existing program (a common step in the embedded software development cycle).

Applied Micro Circuits has simplified the multi-processor programming model by letting the programmer think of their six processing element device as a single logical CPU. Further details on their approach were not available, but many similar past efforts were met with limited success.

Integrated Development Environment (IDE)

Increasingly, there has been more focus on the programmability of these devices. This has manifested itself in many ways. First, NP vendors are basing their products on standard processor cores to get the benefit of existing tool chains. For example, Broadcom’s Mercurian uses two MIPS processors augmented for network processing. This allows them to use the GNU C/C++ tool chain for (almost) free.

An increasing number of NPs have C compilers: Alchemy, Applied Micro Circuits, Broadcom, Cisco, Clearwater Networks, Conexant, EZchip, Intel (coming soon), Lexra, and Motorola [71]. Programming in C is only a first step; to take advantage of the specialized architectural features, pragmas and inlined assembly coding are still required.

Operating System (OS)

For NP operating systems support, there’s an increasing trend to implement more OS functionality in hardware and expose an interface to the application. As a result, there is only a small software component to the operating system. For example, the Intel IXP1200 provides instruction set extensions to perform fast context swapping (there is a hardware thread scheduler). Applications use these instructions to perform context swapping without

the need of an operating system. On the IXP1200, memory management is handled in a similar fashion: the SRAM queues can be used as freelists, thus obviating the need for a separate OS service routine. Some NPs have special hardware that handles the common I/O path (i.e. packet flow). Clearwater's Packet Management Unit copies data from a MAC device into a memory shared by the core. IBM, Motorola, Intel, EZchip have similar units. In fact, many network processors do not even run an OS on the data-plane processing elements. Operating systems will still be needed for control processors, some memory management, and limited I/O handling. Currently, most of the NPs that have OS support are based on a standard architecture (e.g. MIPS).

4.4 Summary

Based on a synthesis of the characteristics analyzed in this section, we can estimate the target markets of the network processors profiled in this report. Figure 41 presents a comparison of network processors along two axes, data rate and computational requirements. The x-axis represents increasing computational requirements: from control and management-plane support to low layer (layers 2-3) data-plane processing to higher layer (layers 4-7) data-plane processing. On the y-axis, we identify the three target markets for network processors in order of increasing data rate requirements: access, edge, and core. The labels on each axis represent points on a continuum, rather than separate categories, as a network processor targeted for high-end edge equipment can also be used for low-end core equipment, for example.

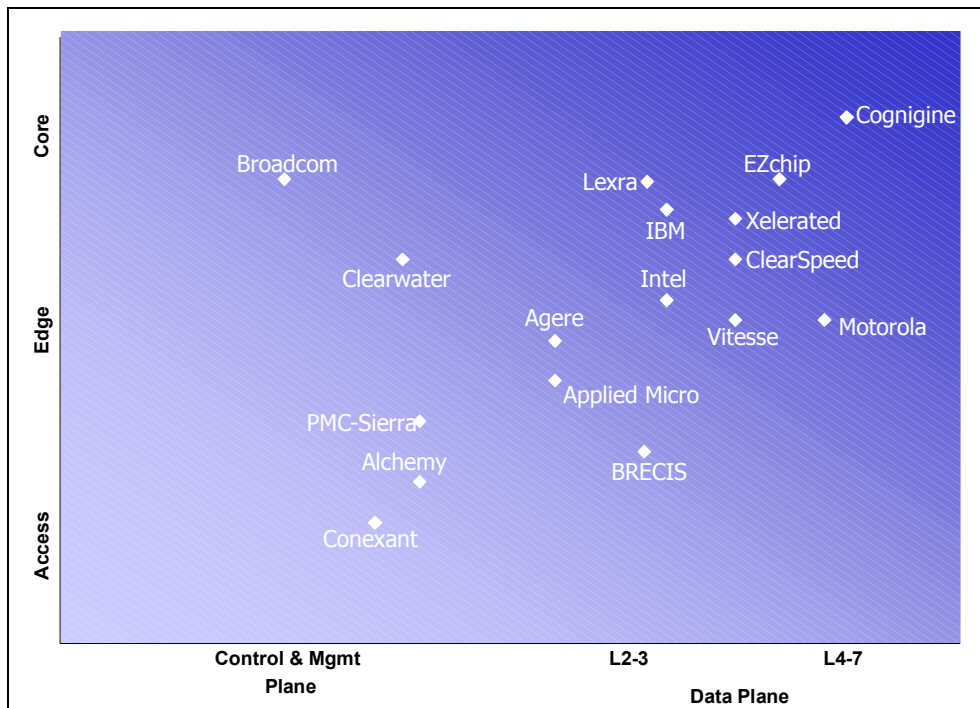


Figure 41. Map of Network Processor Market.

The analysis reflected in Figure 41 reveals a few groups of network processors. PMC-Sierra, Alchemy, and Conexant are focused on low computation (i.e. management, control, and possibly lower layer data-plane processing) for access and low-end edge equipment. Broadcom and Clearwater Networks have much more computational power, but lack the

support for data parallelism to perform core data-plane processing. As a result, they are targeted for control and management-plane processing for high data rate networking equipment. The bulk of the market is focused on data-plane operations for edge and core equipment, representing a response to the massive recent growth in this area. Agere, Applied Micro Circuits, and Intel have enough computational power and specialized hardware to perform layer 2-3 data-plane tasks for edge equipment, but not enough of either for the higher layers or the core market. The increased parallelism of the NPs from Lexra and IBM enable them to hit core equipment data rates performing lower-layer data-plane processing. The powerful co-processors of Motorola's C-5 DCP and the numerous 8-bit computational blocks of ClearSpeed's NP allow both of them to perform higher layer processing for edge equipment. Cognigine, EZchip, and Xelerated Packet Devices are best suited for higher layer processing at the highest data rates. Not coincidentally, none of these devices are currently even sampling, let alone shipping product.

5 Looking Forward

In this section, we explore some trends in networking applications and network processing architectures. We also speculate on the future of mapping networking applications onto network processing architectures.

5.1 *Applications*

The most salient trend with respect to applications is the increase in data rates. This impacts all segments of the network processor market as the hunger for bandwidth comes from the very edge of the network, individual nodes. As individual nodes demand more bandwidth, they push on all classes of network devices, access, edge, and core. With data rates increasing at a super-Moore's Law rate, every 12-18 months each segment must a data rate more than double what it previously supported. Today, the core devices support 2.5Gbps, the next generation will have to support 10Gbps, or even 40Gbps. While core devices must support the fastest data rates, edge devices usually lag them by one generation. Likewise, access devices lag edge devices by a generation.

Networking applications appear to be moving in a couple directions. First, they tend to break the traditional OSI stack model. Often, higher layer information is used to make lower layer decisions. For example, a "web switch" or a URL load balancer uses TCP ports and/or HTTP protocol information to determine which server to route requests to. Since higher layer protocols/applications change rapidly and are difficult to predict (e.g. the rise of peer-to-peer), it is important for network processors to not focus on supporting a single protocol, but rather be able to implement many different ones.

The move to IPv6 will have profound differences on lookup implementations, as IPv6 addresses are 128 bits long (compared to 32-bit IPv4 addresses). Since many applications require lookup on multiple fields, lookups of >300-bit keys will become common. Currently, the largest third-party co-processors can only handle 288 bits (that too, for only a small number of entries). IPv6 also requires supporting the IPSec protocol. This will require implementations for various encryption and authentication algorithms, which are extremely computationally intensive. These changes will require NPs to not only support higher data rates, but also an increasing amount of computation.

The accelerating adoption of MPLS as an interoperability standard between various core protocols like Ethernet, SONET, Frame Relay, and ATM will require network processing systems to support all the control protocol processing required for Label Switch Routers. It is unclear if a standard control processor (i.e. general-purpose processor) will be able to support this.

5.2 *Architecture*

To date, network processors have exhibited great architectural diversity. Based on changing application requirements, Moore's Law increases, and the analysis in Section 4.2, we predict architectural changes in network processing systems. Specifically, we examine the increasing importance of parallel processing, co-processors, and communication architectures.

Parallel Processing

While Figure 38 and Figure 39 may give the impression that many network processors are already highly parallel, we note that many of them have not yet been released. Consider the highly parallel network processors of Figure 38: EZchip's 64 processing element NP-1 and Cognigine's 16 PE 4-issue processor are scheduled to sample later this year. Xelerated Packet Device's 12 PE device is not due until the second quarter of 2002. Likewise, many of the highest MIPS network processors of Figure 39 are not in full production either.

A major aspect of the future of the parallel processing of network processors is their ability to scale to higher data rates of the future. Many of the older network processors will benefit more from technology scaling as they are currently achieving their performance with an old process technology. For example, Intel is using a 0.28 μ process technology for their IXP1200. Moving to a 0.13 μ process technology should enable them to double their clock rate (at least). On the other hand, Broadcom's Mercurian is already implemented in 0.15 μ process and runs at 1GHz. It is unlikely they will get much improvement moving to 0.13 μ . Since data rates continue to rise faster than Moore's Law, advances in process technology will not be enough. The scalability of architectures will also play a role. Scalar architectures like those of Alchemy and Conexant will require major redesign. To achieve more parallelism, wide issue architectures (like the 10-issue CNP810SP from Clearwater Networks) will struggle as adding extra functional units reaches the point of diminishing returns.

Rise of Co-processors

The rise of co-processors represents a shift from ASIPs towards ASICs (see Figure 1). In this section, we examine the impact of the two types of network co-processors, integrated and external. Integrated co-processors are tightly coupled to the processing elements of a network processor. Their functionality may be either algorithm-specific or task/kernel specific. External co-processors are third party developed blocks that are specific to particular networking tasks. Since these devices must work with a variety of NPs, they are interfaced through a bus or memory mapped. This is an exploding field that has yielded almost as network co-processor companies as NP companies.

The increasing use of co-processors raises a couple of issues. First, co-processors are less programmable than network processors. Since most of the functionality of a co-processor is hard-wired, the future use of the device is limited (for in the field software upgrades to support new protocols or applications, for example). In addition, the widespread use of external co-processors makes it difficult to develop software for networking processing systems. Since there is no standard application-level interface for co-processors, each of them has a different mode of interaction.

Second, the use of external network co-processors increases overall system cost. Since most network processors and co-processors are not available as IP, a network processing systems with external co-processors will likely be composed multiple chips. This not only increases system power consumption, but also requires a larger printed circuit board area (which is directly related to cost). The rise in adoption of external co-processors is not allowing network processing systems to take advantage of silicon integration that technology scaling provides.

By extrapolating from the current use of co-processors, we can imagine a network processing system consisting of co-processors for pattern matching, lookup, queue management, security, and packet manipulation controlled by a general-purpose processor. Writing a networking application for this device will mostly be a series of calls to these co-processors. The “network processor” will be little more than a controller. This solution also provides little flexibility to adapt to future changes in protocols or applications. We would expect these solutions to be used only for performance critical applications, like core devices.

Communication Architectures

While processing packets faster with increased parallelism will be important in the future, the increased integration of future network processors will place a larger burden on the communication fabric that connects various on-chip components. As more processing elements, co-processors, and memories are integrated on a chip, simple buses and local connection schemes that are used to connect a handful of elements will not be sufficient. Interconnect schemes that scale to 128 or 256 components will have a distinct advantage.

The complexity of these schemes will likely increase communication latency; effective means to hide this latency will also be important. BRECIS’s communication architecture has taken an interesting approach of mapping application characteristics directly to their bus architecture. Their bus supports three priority levels, which correspond to the three types of packets their device processes: voice, data, and control. This allows programmers to handle the different latency and throughput requirements of these packet types.

5.3 Mapping Applications onto Architectures

As more programmable solutions for networking emerge, the importance of software will only increase. The current method of ASIP design is for some architects to build a device for a particular application space using their own knowledge of that space. Then, they hand it off to the software team who writes (or attempts to write) a compiler for it. This hides many of the design decisions and relies on the judgment capabilities of the architect. Instead, an approach that couples hardware and software together and evaluates trade-offs of particular implementations will emerge. The result is an architecture for a particular application domain along with a method to map software to this architecture. Vissers et al [72] [73] have demonstrated this approach for multimedia processors at Trimedia. The MESCAL [74] research group is developing methodologies and tools to support this approach for future designs.

Recently, a few networking software companies have started targeting network processors, reflecting the importance of software in the end system. However, many of the network software companies deliver a solution in C and presume the presence of operating systems like Linux, FreeBSD, or VxWorks (e.g. Trillium, IPinfusion). Nortel’s Open IP Environment provides a framework and building blocks (modules) for developing networking applications. These modules are implemented in C/C++ and communicate via a well-defined APIs. Modules for most IP protocols has already been built and many network processor companies have partnered with Nortel on this effort including IBM, Intel, Motorola, and Vitesse [75]. In addition, the Network Processing Forum, an industry

consortium, is also developing standard software interfaces to network processors. Their API will not be released until the fourth quarter of 2001 [76]. It is unclear whether either of these will succeed, as the efficiency of the mapping these APIs onto various NPs must first be demonstrated.

Teja Technologies has taken an interesting angle by developing a “network processor operating system.” Their flagship product provides a higher level of abstraction for designing networking applications. This allows application designers to design and modify applications independent of the target architecture. The basic elements of this abstraction are *servers* (compute elements) and *queues* that interact using communicating finite state machines. In addition, they allow designers to specify memory layout of packets. From this abstract application representation, they generate scheduling and computational code for network processors. Currently, Teja only supports code generation for Intel’s IXP1200. Since a code generator of a network processor is a substantial effort, the manifold of network processors on the market severely hampers the scalability of this approach.

Companies that are known for developing DSP (Digital Signal Processor) compilers have also developed compilers for network processors. For example, Connected Components Corporation wrote a compiler for Applied Micro Circuits’ nP7000 and Cisco’s PXF. Green Hills is currently developing a compiler for Clearwater Network’s network processor.

6 Conclusions

In this report, we've surveyed and evaluated the diverse field of network processors. Since network processors are "application-specific," we first explored networking applications. After analyzing these applications, we extracted some common networking tasks. These kernels provided a sort of "benchmark" by which to compare the functionality of different network processors.

We then surveyed in detail many network processors on the market in addition to those scheduled to be released. For each network processor, we examined their architecture, programmability, implementation details, and announced design wins. To better understand the numerous NP offerings, we synthesized this "raw data" along many axes. We identified and described different market segments of networking equipment (access, edge, and core). We then plotted the timeline, parallel processing features, specialized hardware, and latency hiding capabilities of network processor architectures. Combining this with their various programmability aspects results in our conclusions regarding what target markets network processors are best suited for.

The analysis used in reaching these conclusions also helps us identify future trends of the network processor market. The increasing data rate requirements across all market segments will force higher throughput. The ever-changing networking applications will reinforce the programmability of network processors. The move to 128-bit IPv6 address will have profound effect on access to memory and table lookup functions. On the architecture front, our analysis raises more questions than it does answers:

- What is the right mix of processing element and functional unit parallelism?
- How will the increasing use of co-processors affect network processor architectures? Will future network processors be merely a collection of co-processors coordinated by a controller? What about the lack of flexibility of such an approach?
- How will on-chip communication architectures adapt to the increasing number of processing elements, co-processors, memories, and peripherals they must connect?

While much of the industry has focused on the hardware side of the system, what about the software side? The complexities of these architectures make them very difficult for programmers to think about, let alone to provide effective high-level language support for. While C/C++ compilers exist for many network processors, performance critical code will continue to be written in assembly. Is there a common programming model that can be used to target multiple network processors (much like C is to general-purpose processors)? Will this software difficulty force future architectures to be much more programmable?

As stated in the introduction, the emergence of network processors is part of a broader paradigm shift from ASICs to ASIPs. Networking is a great example of where programmable solutions have an advantage over hardwired solutions; the changing standards and applications require flexibility, and the increasing data rate requirements push for faster performance and for fast time-to-market. Being a leading example of the move to programmable systems, we can learn a great deal from their maturation process and apply this to other application areas beginning this shift.

7 Web Sites

The following is a list of relevant web sites as of July 23, 2001.

- Network Processors
 - [AGERE](http://www.agere.com/): <http://www.agere.com/>
 - [Alchemy Semiconductor, Inc.](http://www.alchemysemi.com/): <http://www.alchemysemi.com/>
 - [Allayer](http://www.allayer.com/): <http://www.allayer.com/>
 - [Bay Microsystems](http://www.baymicrosystems.com/): <http://www.baymicrosystems.com/>
 - [Brecis Communications](http://www.brecis.com/): <http://www.brecis.com/>
 - [C-Port Corporation, A Motorola Company](http://www.cportcorp.com/): <http://www.cportcorp.com/>
 - [Cisco](http://www.cisco.com/): <http://www.cisco.com/>
 - [ClearSpeed](http://www.clearspeed.com/) (formerly PixelFusion): <http://www.clearspeed.com/>
 - [Clearwater Networks](http://www.clearwaternetworks.com/) (formerly XStream Logic Devices):
<http://www.clearwaternetworks.com/>
 - [Cognigine](http://www.cognigine.com/home.html): <http://www.cognigine.com/home.html>
 - [Entridia Corporation](http://www.entridia.com/): <http://www.entridia.com/>
 - [EZchip](http://www.ezchip.com/): <http://www.ezchip.com/>
 - [IBM Networking Technology](http://www.chips.ibm.com/products/wired/communications/network_processors.html):
http://www.chips.ibm.com/products/wired/communications/network_processors.html
 - [IP Semiconductors A/S](http://www.ipsemiconductors.com): <http://www.ipsemiconductors.com>
 - [Intel\(R\) Networking and Communications Building Blocks](http://developer.intel.com/design/network/INDEX.HTM):
<http://developer.intel.com/design/network/INDEX.HTM>
 - [ishoni Networks](http://64.35.17.187/index.asp): <http://64.35.17.187/index.asp>
 - [Lexra](http://www.lexra.com/): <http://www.lexra.com/>
 - [Maker Communications, Inc.](http://www.maker.com/) (now Conexant): <http://www.maker.com/>
 - [MMC Networks, Inc.](http://www.mmcnet.com/) (now Applied Micro Circuits):
<http://www.mmcnet.com/>
 - [Navarro Networks](http://www.navarronetworks.com/): <http://www.navarronetworks.com/>
 - [Onex Communications](http://www.onexaco.com/): <http://www.onexaco.com/>
 - [PMC-Sierra TT1 Chip Set](http://www.pmcsierra.com/products/details/pm9311/):
<http://www.pmcsierra.com/products/details/pm9311/>
 - [SiByte Inc.](http://www.sibyte.com/) (now Broadcom): <http://www.sibyte.com/>
 - [Silicon Access Networks](http://www.siliconaccess.com/): <http://www.siliconaccess.com/>
 - [SiTera](http://www.sitera.com/) (now Vitesse): <http://www.sitera.com/>
 - [Xelerated Packet Devices](http://www.xelerated.com/): <http://www.xelerated.com/>
- Network Co-processors
 - [Acorn Networks](http://www.acornnetworks.com/): <http://www.acornnetworks.com/>
 - [Chrysalis-ITS](http://www.chrysalis-its.com/): <http://www.chrysalis-its.com/>
 - [Fast-Chip](http://www.fast-chip.com/): <http://www.fast-chip.com/>
 - [Hifn](http://www.hifn.com/): <http://www.hifn.com/>
 - [Lara Networks](http://www.laranetworks.com/home.html) (now Cypress Semiconductor):
<http://www.laranetworks.com/home.html>
 - [NetLogic Microsystems](http://www.netlogicmicro.com/): <http://www.netlogicmicro.com/>
 - [Orologic](http://www.oro-logic.com/) (now Vitesse): <http://www.oro-logic.com/>
 - [Solidum Systems](http://www.solidum.com/home.cfm): <http://www.solidum.com/home.cfm>

- [SwitchCore](http://www.switchcore.com/): <http://www.switchcore.com/>
 - [ZettaCom](http://www.zettacom.com/): <http://www.zettacom.com/>
- Embedded Network Software
 - [Connected Components Corp.](http://www.concmp.com/): <http://www.concmp.com/>
 - [GreenHills Software](http://www.ghs.com/): <http://www.ghs.com/>
 - [Teja Technologies](http://www.teja.com/): <http://www.teja.com/>
 - [Trillium](http://www.trillium.com/): <http://www.trillium.com/>
- Other Resources
 - [EE Times](http://www.eetimes.com/): <http://www.eetimes.com/>
 - [The Linley Group](http://www.linleygroup.com/): <http://www.linleygroup.com/>
 - [Network Processing Forum](http://www.npforum.org/): <http://www.npforum.org/>
 - [NPC 2001 Network Processors Conference](http://www.networkprocessors.com/):
<http://www.networkprocessors.com/>

8 Acronym Dictionary

AAL – <i>ATM Adaptation Layer</i>	OS – <i>Operating System</i>
AH – <i>Authentication Header</i>	OSI – <i>Open System Interconnect</i>
ARP – <i>Address Resolution Protocol</i>	P2P – <i>Peer-to-Peer</i>
ASIC – <i>Application Specific Integrated Circuit</i>	PDU – <i>Packet Data Unit</i>
ASIP – <i>Application Specific Instruction Processor</i>	PE – <i>Processing Element</i>
ATM – <i>Asynchronous Transfer Mode</i>	PHB – <i>Per-Hop Behavior</i>
BA – <i>Behavior Aggregate</i>	POS – <i>Packet over SONET</i>
CMTS – <i>Cable Modem Termination System</i>	QoS – <i>Quality of Service</i>
CoS – <i>Class of Service</i>	RFC – <i>Request For Comments</i>
CPI – <i>Common Part Indicator</i>	RSVP – <i>Resource Reservation Setup Protocol</i>
CRC – <i>Cyclic Redundancy Check</i>	RTSP – <i>Real Time Transport Protocol</i>
DSCP – <i>Differentiated Services codepoint</i>	SA – <i>Security Association</i>
DSM – <i>Deep Sub-Micron</i>	SAR – <i>Segmentation & Reassembly</i>
DSP – <i>Digital Signal Processor</i>	SFU – <i>Special Functional Unit</i>
ESP – <i>Encapsulating Security Payload</i>	SLA – <i>Service-Level Agreement</i>
FPGA – <i>Field Programmable Gate Array</i>	SMT – <i>Simultaneous Multi-Threading</i>
FU – <i>Functional Unit</i>	SPI – <i>Security Parameters Index</i>
GPP – <i>General-Purpose Processor</i>	SSL – <i>Security Socket Layer</i>
HTTP – <i>HyperText Transfer Protocol</i>	TCA – <i>Traffic Conditioning Agreement</i>
IDE – <i>Integrated Development Environment</i>	TCP – <i>Transmission Control Protocol</i>
ILP – <i>Instruction-Level Parallelism</i>	TOS – <i>Type Of Service</i>
IP – <i>Internet Protocol</i>	TTL – <i>Time-To-Live</i>
IPSec – <i>Internet Protocol Security</i>	TTM – <i>Time-To-Market</i>
IPv4 – <i>Internet Protocol Version 4</i>	UDP – <i>User Datagram Protocol</i>
IPv6 – <i>Internet Protocol version 6</i>	URL – <i>Uniform Resource Locator</i>
LAN – <i>Local Area Network</i>	UU – <i>User-to-User</i>
LER – <i>Label Edge Router</i>	VC – <i>Virtual Circuit</i>
LPM – <i>Longest Prefix Match</i>	VCI – <i>Virtual Circuit Identifier</i>
LSR – <i>Label Switch Router</i>	VLAN – <i>Virtual Local Area Network</i>
MAC – <i>Media Access Control</i>	VLIW – <i>Very Long Instruction Word</i>
MF – <i>Multi-Field</i>	VoIP – <i>Voice over IP</i>
MIPS – <i>Millions of Instructions Per Second</i>	VP – <i>Virtual Path</i>
MPLS – <i>Multi-protocol Label Switching</i>	VPI – <i>Virtual Path Identifier</i>
MTU – <i>Maximum Transmission Unit</i>	VPN – <i>Virtual Private Network</i>
NAT – <i>Network Address Translation</i>	WFQ – <i>Weighted Fair Queuing</i>
NP – <i>Network Processor</i>	

9 References

All URLs are as of July 23, 2001.

- [1] K. Keutzer. "Enabling Fully Programmable Embedded System Solutions." Presentation. Gigascale Silicon Research Center Annual Review. December 1999.
- [2] Agere, Inc. "The Challenge for Next Generation Network Processors." White Paper. Agere, Inc. September 10, 1999.
- [3] W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley. January 1994.
- [4] A. Tanenbaum. *Computer Networks*. Prentice Hall PTR. January 1996.
- [5] M. Rose. *The Open Book: A Practical Perspective on OSI*. Prentice-Hall. 1990.
- [6] P. Loshin. *Essential ATM Standards: RFCs and protocols made practical*. John Wiley & Sons. 2000.
- [7] Cisco Systems. "Cisco VLAN Roadmap." April 15, 1999. Available at <http://www.cisco.com/warp/public/538/7.html>.
- [8] D. Passmore and J. Freeman. "The Virtual LAN Technology Report." Available at <http://www.sunset.ch/~bro/vlan/3com/vlan.html>.
- [9] E. Rosen, A. Viswanathan, R. Callon. Multiprotocol Label Switching Architecture; RFC3031. *Internet Request for Comments*. January 2001.
- [10] D. Comer, D. Stevens. *Internetworking with TCP/IP Volume II*. Prentice Hall. 1994.
- [11] R. Braden. Requirements for Internet Hosts -- Communication Layers; RFC1122. *Internet Request for Comments*. October 1989.
- [12] S. Deering, R. Hinden. Internet Protocol, Version 6 (IPv6) Specification; RFC2460. *Internet Request for Comments*. December 1998.
- [13] J. C. Bays. "The Complete PATRICIA." PhD thesis, University of Oklahoma, 1974.
- [14] C. Partridge. Using the Flow Label Field in IPv6; RFC1809. *Internet Request for Comments*. June 1995.
- [15] N. Doraswamy and D. Harkins. *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall. 1999.
- [16] N. Deshpande. "TCP Extensions for Wireless Networks". Available at ftp://ftp.netlab.ohio-state.edu/pub/jain/courses/cis788-99/tcp_wireless.pdf
- [17] K. Egevang, P. Francis. The IP Network Address Translator (NAT); RFC1631. *Internet Request for Comments*. May 1994.
- [18] Alteon Web Systems. "The Next Steps In Server Load Balancing." White Paper. November 1999.
- [19] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss. An Architecture for Differentiated Services; RFC2475. *Internet Request for Comments*. December 1998.
- [20] K. Nichols, S. Blake, F. Baker, D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers; RFC2474. *Internet Request for Comments*. December 1998.
- [21] R. Braden, D. Clark, S. Shenker. Integrated Services in the Internet Architecture: an Overview; RFC1633. *Internet Request for Comments*. June 1994.
- [22] Linley Gwenap. "Net processor makers race toward 10-Gbit/s goal." EE Times. June 19, 2000. Available at <http://www.eetimes.com/story/OEG20000619S0011>.
- [23] Lucent Technologies. "PayloadPlus Functional Programming Language." Preliminary Product Brief. Lucent Technologies, Microelectronics Group. April 2000.

- [24] Lucent Technologies. "PayloadPlus Fast Pattern Processor." Preliminary Product Brief. Lucent Technologies, Microelectronics Group. April 2000.
- [25] Rothfus, Eric J. "The Case for a Classification Language." Agere White Paper. Sept 10, 1999.
- [26] Lucent Technologies. "PayloadPlus Routing Switch Processor." Preliminary Product Brief. Lucent Technologies, Microelectronics Group. April 2000.
- [27] Lucent Technologies. "PayloadPlus Software Development Environment", Preliminary Product Brief, Lucent Technologies, Microelectronics Group, April 2000.
- [28] Lucent Technologies. "PayloadPlus Agere System Interface", Preliminary Product Brief, Lucent Technologies, Microelectronics Group, April 2000.
- [29] Alchemy Semiconductor, Inc. "The Alchemy Au1000 Internet Edge Processor." Product brief. 2000.
- [30] MMC Networks. "nP Family." Data Sheet.
- [31] MMC Networks. "EPIF-200 Packet Processor Product Overview." Data Sheet.
- [32] Loring Wirbel. "Network processor handles mix of carrier services." EE Times. September 18, 2000.
- [33] Michael Ngo. "Introducing the BRECIS Multi-Service Processor™." Presentation. Network Processor Forum. June 14, 2001.
- [34] BRECIS Communications. "MSP5000 Multi-Service Processor." Product Brief. May 2001.
- [35] SiByte. "SB-1250." Data Sheet. October 2000.
- [36] SiByte. "SB-1 CPU." Fact Sheet. October 2000.
- [37] Linley Gwennap. "SiByte net processor shoots for control." EE Times. October 9, 2000. Available at <http://www.eetimes.com/story/OEG20001009S0031>.
- [38] Robert Stepanian. "SiByte SB-1 MIPS64 CPU Core." Presentation. June 13, 2000.
- [39] Cisco Systems. "Parallel eXpress Forwarding in the Cisco 10000 Edge Service Router." White Paper. October 2000.
- [40] Narendra Sankar. "CNP810™ Network Services Processor Family." Presentation. Network Processor Forum. June 14, 2001.
- [41] Mario Nemirovsky. "Simultaneous Multithreading Architectures: Enabling the Next-Generation Internet." Presentation. XStream Logic Devices. 2000.
- [42] Rupan Roy. "A Monolithic Packet Processing Architecture Monolithic Packet Processing Architecture." Presentation. Network Processor Forum. June 14, 2001.
- [43] Maker. "MXT4400: Traffic Stream Processor." Product Brief. 1999.
- [44] EZchip Technologies. "7-Layer Packet Processing: A Performance Analysis." White paper. July 2000.
- [45] EZchip Technologies. "Network Processor Designs for Next-Generation Networking Equipment" White paper. December 1999.
- [46] EZchip Technologies. "EZchip Technologies Software Development Suite Now Available For Its 10-Gigabit 7-Layer Network Processor." Press Release. January 17, 2001.
- [47] EZchip Technologies. "EZchip Technologies: 10-Gigabit 7-Layer Network Processors." Corporate Presentation. Available at <http://www.ezchip.com/>.
- [48] EZchip Technologies. "EZchip Technologies And IBM Sign Network Processor Technology Deal." Press Release. November 1, 2000.
- [49] EZchip Technologies. "Avaya Chooses EZchip's 10-Gigabit, 7-Layer Network Processor for its Next Generation Routing Switches." Press Release. May 7, 2001.

- [50] IBM Corp. "IBM Network Processor (IBM32NPR161EPXCAC100)." Product Overview. November 1999.
- [51] Jayant Mathew. "IBM Ropes In Partners for Network Processors." Electronic News Online. May 15, 2000.
- [52] Bernard Cole. "Intel net processor boosts clock, adds C compiler." EE Times. February, 20, 2001. Available at <http://www.eetimes.com/story/OEG20010220S0029>.
- [53] Jayant Mathew. "Network Processor Companies Face the Same Tough Issues." Electronic News Online. July 17, 2000. Available at <http://www.electronicnews.com/enews/Issue/RegisteredIssues/2000/07172000/z24f-1.asp>
- [54] Craig Matsumoto. "CloudShield pushes net processors to next performance level." EE Times. June 26, 2001. Available at <http://www.eet.com/story/OEG20010625S0088>.
- [55] Linley Gwennap. "Lexra offers NetVortex net processor as licensable core." EE Times. June 12, 2000. Available at <http://www.eetimes.com/story/OEG20000610S0001>.
- [56] Paul Alexander, Robert Gelin, W. Patrick Hays, Sol Katzman, William J. Dally. "NetVortex: A Scalable, Multiprocessor for Network Communications." Presentation. Embedded Processor Forum. June 14, 2000.
- [57] Bob Gelin, Paul Alexander, Charlie Cheng, W. Patrick Hays, Ken Virgile, William J. Dally. "NVP: A Programmable OC-192c Powerplant." Presentation. Network Processor Forum. June 14, 2001.
- [58] Motorola Corp. "C-5 Digital Communications Processor." Product Brief. May 4, 2000.
- [59] D. Husak. "Communication Processors: a definition and comparison." White paper. Motorola Corp. May 3, 2000.
- [60] David Husak & Robert Gohn. "Network Processor Programming Models: The Key to Achieving Faster Time-to-Market and Extending Product Life." White Paper. Motorola Corp. May 4, 2000.
- [61] Motorola Corp. "C-Ware Software Toolset." Product brief. May 2, 2000.
- [62] Quantum Effect Devices. "QED RISCMark." Product Sheet.
- [63] SiTera. "Intelligent Network Processing." Network Processor Brochure. 1999.
- [64] SiTera Corp. "PRISM IQ2000." Product Brief. February 2000.
- [65] SiTera Corp. "Ushering in a New Era of Performance & Flexibility." Presentation. April 2000.
- [66] Peter Glaskowsky. "Network Processors Multiply." Microprocessor Report. Jan 29, 2001.
- [67] Thomas Eklund. "The World's First 40Gbps (OC-768) Network Processor." Presentation. Network Processor Forum. June 14, 2001.
- [68] Xelerated Packet Devices. "Xelerator™ X40 Packet Processor." Preliminary Product Brief. June 2001.
- [69] Xelerated Packet Devices. "Xelerator™ T40 Traffic Manager." Preliminary Product Brief. June 2001.
- [70] Mel Tsai. Personal Communication. 2001.
- [71] Linley Group. "IXP1200 Enhancements Include Compiler." Article. February 19, 2001.
- [72] J.T.J. van Eijndhoven, F.W. Sijstermans, K.A. Vissers, E.J.D. Pol, M.J.A. Tromp, P. Struik, R.H.J. Bloks, P. van der Wolf, A.D. Pimentel, H.P.E. Vranken. "TriMedia CPU64 Architecture." ICCD 1999, *International Conference on Computer Design*, October 10-13, 1999, Austin Texas.

- [73] G.J. Hekstra, G.D. La Hei, P. Bingley, F.W. Sijstermans. "TriMedia CPU64 Design Space Exploration." *ICCD 1999, International Conference on Computer Design*, October 10-13, 1999, Austin Texas.
- [74] K. Keutzer, S. Malik, R. Newton, J. Rabaey and A. Sangiovanni-Vincentelli. "System Level Design: Orthogonalization of Concerns and Platform-Based Design." *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, Vol. 19, No. 12, December 2000.
- [75] Nortel Networks. "Open IP Environment." Product brochure. November 2000.
- [76] Network Processing Forum. "Network Processing Forum Unveils Specifications Roadmap." Press Release. April 30, 2001.

Appendix

A. Detailed Network Processor Summary

The following spreadsheets summarize the architectures of the network processors presented across multiple categories: micro-architecture, architecture, memory, software support, and physical implementation.

Note: many of these network processors are still in the development phase. As a result, some of the details of these products are not available, especially regarding software support and physical implementations.

Micro-Architecture

We summarize the micro-architectures of the NPs described in this section in Table 7. Definitions and implications of the categories used to compare micro-architectures are given below:

- Task-based – Does this NP have any hardware for specific tasks? These are often co-processors or special functional units (FUs).

Since many of the tasks involved in network processing are quite specific, specialized hardware can be used to speed up common operations (relative to a software implementation). The functions of task-based hardware employed is a good indicator for what types of processing an NP is targeted towards.

- Special instructions – What kind of special instructions does this NP have, if any? These instructions may be interfaces to special FUs, co-processors, or other pieces of hardware (e.g. CAMs).

Special instructions are an interface to specialized hardware. This gives an indication of how the software programmer can take advantage of this hardware.

- Number of active contexts – How many different contexts (either threads or processes) can be physically executing at a time?

This counts the number of program counters on the NP. The number of active contexts gives an idea for how many independent executing entities there are.

- Number of contexts per active contexts – For each active context, there is hardware support for how many contexts to exist?

Since many NPs have hardware support for multiplexing resources across contexts and fast switching between them, we only include the number of contexts that hardware can efficiently support. With the appropriate operating system software, many of these devices can support an infinite number of contexts. However, switching between them would be prohibitively slow, as the entire state of the machine would have to be stored to memory and another one loaded from memory.

	Task-based	Special Instructions	# of Active Contexts	# of Contexts per Active Context
Agere (PayloadPlus)	- FPP: pattern matching on a frame - RSP: packet manipulation, traffic management - ASI: "slow path" processing	Yes, for traffic management, QoS/CoS, and packet modifications	3	64 for the FPP
Alchemy	None	Yes, but very generic - conditional moves, count leading 0s/1s.	1	1
BRECIS Communications (MSP5000)	ADPCM, CRC, security co-processors	Yes, for DSP	3	1
Broadcom (Mercurian SB-1250)	2 packet FIFOs	None	2	1
Cisco (PXF/Toaster 2)	None	Yes	at least 32	
ClearSpeed	Table Lookup Engine	None	1 per MTAP; variable number of MTAPs	up to 32 per MTAP
Clearwater Networks (CNP810SP)	Packet Management Unit	Yes	8	1
Cognigine	None	Yes, variable (based on application)	16	4
Conexant (CX27470 Traffic Stream Processor)	Channel Descriptor Lookup Engine and Packet/Command Engine map packets to SW processes	Yes	1	multiple
EZchip (NP-1)	· Parse: identifies and extracts packet headers and protocols · Search: performs lookups · Resolve: assign packet to appropriate queue and/or port · Modify: modifies packet contents	Yes, the ISA of each TOP is customized to a set of tasks		
IBM (PowerNP)	Hardware coprocessor accelerates tree searching and frame manipulation	Yes	16	2
Intel (IXP1200)	Specialized functional units for hashing and queue management	Yes	6	4
Lexra (NetVortex & NVP)	None	Yes	2	configurable from 1-8
Applied Micro Circuits (nP)	Policy Engine and Search Engine	Yes	up to 8	8
Motorola (C-5 DCP)	· Fabric processor: for using multiple C-5's · Table lookup unit · Queue management · Buffer management: fast, flexible memory management	Yes, accessed via CPI	16	1
PMC-Sierra	None	Yes, for DSP and networking	2	1
Vitesse (PRISM IQ2000)	Network co-processors for packet processing for classification, lookups, and QoS/CoS priority checking	Yes	4	5
Xelerated Packet Devices (X40 & T40)	Meters, counters, WRED, Fragmentation/Reassembly	Yes, packet-based instruction set	10 for X40, 2 for T40	1

Table 7. Micro-architectural Comparison of NPs.

Architecture

To summarize the architectures of different NPs, we use the following axes:

- Central control – Does this NP have a central control processor? If so, what is it?
Control-plane operations are very diverse and often quite complicated. As a result, they are often executed on a general-purpose processor to take advantage of HLL tool support (i.e. compiler and debugger). Given the coupling between control and data-plane operations, it would be advantageous have a control processor integrated with the packet processors.
- Multi-PE – Does this NP employ multiple PEs? If so, what kinds?
Multiple processors can be used to take advantage of the inherent parallelism involved in datagram processing. How different NPs use multiple PEs has an impact on their overall performance.
- Inter-PE communication structure – What does the NP use to communicate with various PEs, co-processors, and memory.
Although there is a lot of inherent parallelism in network processing, communication between different hardware elements (especially memory) is paramount to an efficient implementation.
- Interfaces – What kind of interfaces does this NP support?
The interfaces an NP supports indicates the ease of integrating this device into an overall system.

	Central control	Multi-PE	Inter-PE communication structure	Interfaces
Agere (PayloadPlus)	ASI	3 (FPP, RSP - which has 3 VLIW compute engines, ASI)	Functional Bus Interface to connect FPP/ASI	UTOPIA Level 2/3, POS-PHY Level 3, MPI, FBI, CBI, PCI
Alchemy	Yes. Single 32-bit MIPS processor	No	N/A	2 Ethernet controllers, IrDA, USB
BRECIS Communications (MSP5000)	MIPS R4KM	2 LSI ZSP400s - 4 issue superscalars	Multi-Service Bus Architecture - 3.2Gbps b/w, 3 priorities	2 10/100 Ethernet MACs, UTOPIA 2, 128 channel TDM
Broadcom (Mercurian SB-1250)	None	2 64-bit MIPS CPUs; 4 issue	256 bit bus; runs at 1/2 CPU speed	3 On-chip Gigabit Ethernet MACs
Cisco (PXF/Toaster 2)	None	32 2-issue VLIWs arranged in a 4x8 systolic array		
ClearSpeed	Depends	variable number of MTAPs	ClearConnect bus connects MTAPs - 50+ 200Gbps peak b/w	
Clearwater Networks (CNP810SP)	Simultaneous MultiThreading, 8 issue superscalar	No	N/A	
Cognigine	None	16 RCUs	RSF - hierarchical crossbar	SPI-4, PCI
Conexant (CX27470 Traffic Stream Processor)	Yes, RISC	No	N/A	UTOPIA, PCI
EZchip (NP-1)	None	64 (TOPs)		
IBM (PowerNP)	On-chip Power PC core	16 programmable protocol processors		40 Fast Ethernet/4Gb MACs with SMII and GMII, POS
Intel (IXP1200)	on-chip 200MHz StrongARM coordinates system activities	6 programmable microengines	Microengines communicate via Fast Bus Interface (FBI)	4.2Gb/s 66MHz IX bus, PCI
Lexra (NetVortex & NVP)	1 control processor	modified RISCs with multi-threading support	64-bit Vortex bus running at chip speed	
Applied Micro Circuits (nP)	None	Yes	Designed to work with other nP's, MMC switch chips, nP co-processors	Fast Ethernet
Motorola (C-5 DCP)	1 executive processor	16 channel processors	3 internal buses connect CPs and co-processors (60Gb/s aggregate bandwidth)	33/66MHz PCI, UTOPIA (Level 2 and 3)
PMC-Sierra	None	2 64-bit MIPS compatible CPUs; Dual issue, superscalar	SysAD interface	
Vitesse (PRISM IQ2000)	None	4 CPUs for route processing and system management	Input and Output Streaming Busses	
Xelerated Packet Devices (X40 & T40)	None	- X40: 10 classifier/PISC pairs - T40: 2 classifier/PISC pairs		

Table 8. Architectural Comparison of NPs.

Software Support

With software becoming an increasingly important part of the overall system, it is important to compare the software support of NPs. We include the following comparisons in Table 9:

- Compilers – Is a compiler, interpreter, and/or assembler available for this device? What about other development tools, like a debugger, simulator, etc?
- Operating systems – What operating systems have been ported to this device? As an increasing amount of OS functionality is moved into hardware, the role of an operating system diminishes.
- Libraries – Many NPs include libraries for common networking applications as well as APIs for accessing specialized hardware.

	Compilers	Operating systems	Libraries
Agere (PayloadPlus)	Yes, for FPP, RSP, and ASI		Application Code Library with basic wire-speed classification and forwarding
Alchemy	C/C++ compiler	Windows CE, Linux, VxWorks	None
BRECIS Communications (MSP5000)	C/C++ compiler	VxWorks, Linux, BSD	firmware for common networking apps
Broadcom (Mercurian SB-1250)	Standard Gnu C/C++ tool chain	FreeBSD, Linux, and VxWorks OS support	
Cisco (PXF/Toaster 2)	C compiler	IOS	
ClearSpeed	C compiler		Application development kit, reference library of common networking apps
Clearwater Networks (CNP810SP)	C/C++ compiler		
Cognigine	C/C++ compiler		application library for common L2-7 functions
Conexant (CX27470 Traffic Stream Processor)	C Compiler and Assembler		Modular SW architecture; Libs for AAL5, AAL2 SAR, POS
EZchip (NP-1)	C compiler and Assembler		
IBM (PowerNP)	Assembler only		None
Intel (IXP1200)	C compiler and Assembler		None
Lexra (NetVortex & NVP)	C compiler		
Applied Micro Circuits (nP)	C/C++ compiler	Wind River	network software reference library
Motorola (C-5 DCP)	C/C++ compiler		CPI that abstracts common networking tasks
PMC-Sierra	C/C++ compiler	Many	
Vitesse (PRISM IQ2000)	yes		Network software library for common functions
Xelerated Packet Devices (X40 & T40)	C compiler		building blocks for control-plane processing

Table 9. Comparison of Software Support for NPs.

Memory

We profile the NPs with respect to their memory support in Table 10 based on the following categories:

- Shared/Distributed – Is the memory shared or distributed (or some combination) across multiple PEs?
- Size/Type – What is the size and type of any on-chip memory?

- Cache size/Associativity – What are the characteristics of any caches that are included on chip?
- Special features – Any other relevant details or features relating to memory

	Shared/ Distributed	Size/Type	Cache Size/ Associativity	Special Features
Agere (PayloadPlus)	Shared between the FPP and RSP	External	None	All memory is off chip; supports 64-bit interface to PC-133 SDRAM and 133MHz pipelined ZBT-style SSRAM
Alchemy			16KB instruction and data cache	Prefetch instructions
BRECIS Communications (MSP5000)	Shared	External	- 80KB I and D cache for each ZSP - 16KB I and D cache for MIPS	Packet queues in each bus interface; intelligent DMA engines
Broadcom (Mercurian SB-1250)	Shared main memory and L2 Cache; distributed L1 Cache	External	32KB L1 for each CPU; share 512KB L2 cache	2 Packet FIFOs
Cisco (PXF/Toaster 2)	Vertical slices of pipeline have access to shared memory	External	None	
ClearSpeed	Distributed among PEs	Per PE: 16-64 byte register files, 1-16KB packet memory	None	Memory controllers to load packets into packet memory
Clearwater Networks (CNP810SP)	N/A	External	64KB I and D Cache; Dual Ported	Packet Management Unit; 2 address generation units
Cognigine		External	I and D cache per RCU	
Conexant (CX27470 Traffic Stream Processor)	N/A	8KB scratchpad	4KB Instruction	Buffer management unit
EZchip (NP-1)				Algorithms that leverage embedded memory to search external memory
IBM (PowerNP)	Distributed	32KB SRAM/Protocol Processor (8KB for instruction memory)	None	Data store co-processor, control memory arbiter, ingress/egress data storage
Intel (IXP1200)	Shared	4KB SRAM (ScratchPad)	None	SRAM Queues; Xmit/Recv FIFOs
Lexra (NetVortex & NVP)	Shared	External		Block transfer controller, table lookup unit, packet buffers
Applied Micro Circuits (nP)				
Motorola (C-5 DCP)	Accessed by queue mgmt, table lookup, and buffer mgmt units	External	None	Table lookup unit, queue and buffer management co-processor
PMC-Sierra	N/A	External	16KB 4-way associative Instr & Data Cache; 256 KB L2 cache	None
Vitesse (PRISM IQ2000)	Shared			Separate bus for lookups; 256-bit wide bus
Xelerated Packet Devices (X40 & T40)	Distributed by function - counter, meter, CAM, pipeline buffer		None	

Table 10. Comparison of Memory for NPs.

Physical Implementation

Lastly, we summarize the NPs in this report regarding physical implementation. Some these categories are not relevant or indicative given that some of these NPs are available as soft cores.

- Process technology – What process technology is being used for this NP?
- Die size
- Core – Is this NP available as a core?
- Speed – Speed of the global clock
- Power – Average power consumption
- Availability – When is this NP available?

	Process technology	Die size	Core	Speed	Power	Availability
Agere (PayloadPlus)	0.18u				12W	Now
Alchemy			Yes	266MHz / 400MHz / 500MHz	<300mW / 500mW / 900mW	Sampling
BRECIS Communications (MSP5000)	0.18u			160MHz	2W	3Q01
Broadcom (Mercurian SB-1250)	0.15u	25mm ²	Yes	1GHz	2.5W	Sampling
Cisco (PXF/Toaster 2)						Cisco internal
ClearSpeed	0.13u	180-295mm ²	Yes	400MHz		1H03
Clearwater Networks (CNP810SP)	0.15u		Yes	300MHz	12W	4Q01
Cognigine	0.18u			200MHz		12/01
Conexant (CX27470 Traffic Stream Processor)				125MHz	4.2W	
EZchip (NP-1)						3Q01
IBM (PowerNP)	0.18u		Yes	133MHz	20W	Now
Intel (IXP1200)	0.28u			200MHz	5W	Now
Lexra (NetVortex & NVP)	0.28u			200MHz	5W	Sampling
Applied Micro Circuits (nP)	0.18u			165MHz	4W	Now
Motorola (C-5 DCP)	0.18u				15W	Now
PMC-Sierra	0.18u			400MHz		Now
Vitesse (PRISM IQ2000)	0.25u		No	200MHz	12W	Now
Xelerated Packet Devices (X40 & T40)	0.13u					4/02

Table 11. Comparison of Physical Implementation for NPs.

B. Applications/Architecture Mapping Table

Table 12 and Table 13 summarize the mapping of application kernels to the architectures examined in Section 3. The application kernels defined earlier (pattern matching, lookup, computation, data manipulation, queue management, and control processing) are the basic operations for a particular packet. However, single packet processing makes poor utilization

of network processor hardware (e.g. stalling on memory access). As a result, a network processing system must simultaneously process multiple packets. To account for this in our analysis of mapping applications onto architectures, we include network processor features for multiple packet processing.

	Features for Multiple Packet Processing	Single Packet Processing Features					
		pattern matching	lookup	computation	data manipulation	queue management	control processing
Agere (PayloadPlus)	64 contexts; 2 issue VLIW	FPP; Pattern Processing Engine		CRC/checksum engine	Stream editor compute engine	programmable queues; queue management engine	
Alchemy	prefetching	few basic special instructions			few basic special instructions		
BRECIS Communications (MSP5000)	Multi-Service Bus, 2 4-issue DSPs			security co-processor		bus interface has packet queues	MIPS
Broadcom (Mercurian SB-1250)	2 64-bit MIPS CPUs; 4 issue					2 packet FIFOs	device meant for control plane
Cisco (PXF/Toaster 2)	32 2-issue VLIWs in systolic array	special instructions		special instructions	special instructions		
ClearSpeed	multiple MTAPs; up to 32 threads per MTAP	many 8-bit PEs	Table Lookup Engine	many 8-bit PEs	many 8-bit PEs		
Clearwater Networks (CNP810SP)	SMT; 10 issue	packet management unit			special instructions	packet management unit	
Cognigine	16 4-issue RCUs, 4 threads each	special instructions (variable)	special instructions (variable)	special instructions (variable)	special instructions (variable)		
Conexant (CX27470 Traffic Stream Processor)	multiple contexts; background context swap	Channel Descriptor Look-up Engine			special instructions	Traffic Scheduling System; buffer management	

Table 12. Mapping of Applications onto Architectures (part 1).

	Features for Multiple Packet Processing	Single Packet Processing Features					
		pattern matching	lookup	computation	data manipulation	queue management	control processing
EZchip (NP-1)	4 TOPs, pipelined	TOPparse	TOPsearch; technique to leverage embedded memory to search		TOPmodify	TOPresolve	
IBM (PowerNP)	16 protocol processors	Classifier Hardware Assist	Data store co-processor; Tree Search Engine	Checksum co-processor	Ingress/Egress Switch Interface	Enqueue and Policy co-processors	Counter co-processor
Intel (IXP1200)	6 micro-engines; 4 threads each with 0-overhead context swapping	special instructions	Hash Engine		special instructions	Receive/Transmit FIFOs; queue interface to SRAM	StrongArm
Lexra (NetVortex & NVP)	multiple RISCs; 8 threads each with 0-overhead context swapping	special instructions	Table Lookup Unit		special instructions	Packet buffer	Control processor
Applied Micro Circuits (nP)	up to 6 processors; 0-overhead context swapping among 8 threads	programmable policy engine	search engine		packet transform engine		statistics engine
Motorola (C-5 DCP)	16 channel processors	SDP	Table Lookup Unit	SDP	SDP	Queue Management Unit	
PMC-Sierra							
Vitesse (PRISM IQ2000)	4 scalar RISCs; share threads	classification engine	lookup co-processor		special instructions	queue management co-processor	
Xelerated Packet Devices (X40 & T40)	10 stage macro-pipeline of classifier/PISC pairs	special instructions	internal CAM, interface to external CAM		special instructions, HW support for frag/reass	Queue Engine, HW support for WRED	meters, counters

Table 13. Mapping of Applications onto Architectures (part 2).