Introduction to Network Processors

Guest Lecture at UC Berkeley, 07Mar2002

Chuck Narad, Principal System Architect
Intel Network Processor Division

Outline

- Introduction
- Application Partitioning
- Generic Networking Equipment
- Network Processor Focus
- Network Processor Challenges
- Fitting the Architecture to the Problem Space

Introduction

- Overview of networking applications and processing systems that are tuned to address them
- Network Process*ing* vs. Network Process*ors*
- Discussion of Network Processors must be driven by what networking applications do
  - Moving data from here to there: Switching, Routing, Aggregation/Disaggregation, Bridging etc.
  - Providing services: Security, Monitoring, Traffic Shaping etc.
- Value proposition of NP's:
  - Improve TTM and to reduce investment by turning a silicon design problem into a programming problem
  - Provide flexibility and field upgradability in networking equipment

What is a Network Processor?

- Terminology emerged in the industry 1997-1998
  - Many startups competing for the network building-block market
- Broad variety of products are presented as an NP
- Some amount of integration and some amount of programmability
- Generally some characteristics that enable efficient processing of network headers in cells or packets
- Sometimes support for higher-level flow management
- Wide spectrum of capabilities and target markets

Motivations for using a Network Processor

- "Flexibility of a fully programmable processor with performance approaching that of a custom ASIC."
- Faster time to market (no ASIC lead time)
  - Instead you get software development time
- Field upgradability leading to longer lifetime for products in the field
  - Ability to adapt deployed equipment to evolving and emerging standards and new application spaces
- Enables multiple products using common hardware
- Allows the network equipment vendors to focus on their value-add

What Can an NP Be Used For?

- Highly dependent on user's application:

- Integrated uP + system controller + "acceleration"
- Fast forwarding engine with access to a "slow-path" control agent
- A smart DMA engine
- An intelligent NIC
- A highly integrated set of components to replace a bunch of ASICs and the blade control uP

Common Features in NPs

- Pool of multithreaded forwarding engines
- Integrated or attached GP uP
- High Bandwidth and High Capacity Memories
  - Embedded and external SRAM and DRAM
- Integrated media interface or media bus
- Interface to a switching fabric or backplane
- Interface to a "host" control processor
- Interface to coprocessors

NP Architectural Challenges

- Application-specific architecture
- Yet, covering a very broad space with varied (and ill-defined) requirements and no useful benchmarks
- The Swiss Army Knife challenge
  - Versatile but does a bad job at everything
- Need to understand the environment
- Need to understand network protocols
- Need to understand networking applications
- Have to provide solutions before the actual problem is defined
  - Decompose into the things you can know
  - Flows, bandwidths, "Life-of-Packet" scenarios, specific common functions

Problem Spaces Addressed by NP's

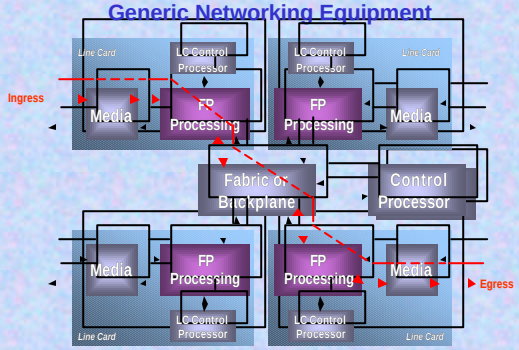
Network Application Partitioning

- Network processing is partitioned into *planes*
  - Forwarding Plane: Data movement, protocol conversion, etc
  - Control Plane: Flow management, (de)fragmentation, protocol stacks and signaling stacks, statistics gathering, management interface, routing protocols, spanning tree etc.
- Control Plane is sometimes divided into Connection and Management Planes
  - Connections/second is a driving metric
  - Often connection management is handled closer to the data plane to improve performance-critical connection setup/teardown
  - Control processing is often distributed and hierarchical

Network Processor Focus

- The NP is generally aimed at Forwarding Plane tasks
  - Data shovel
  - Light Touch: Framing, SAR'ing, Classification and Lookups, Mappings (port, path, tag, flow, etc.)
  - High Throughput
  - Queuing and Scheduling
  - Backplane encapsulation and decapsulation
- Packets requiring heavier work are offloaded to Control Plane or Coprocessor
- NP's usually provide a forwarding plane closely coupled with a uP.
  - The microprocessor may implement the entire control plane
  - May handle a portion of it locally (e.g. flow setup) and have an external host which provides the higher-level control plane

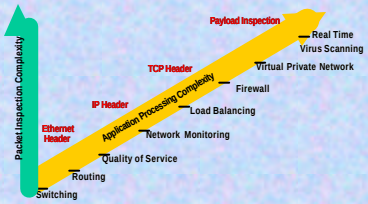
Generic Networking Equipment



L3-L7 Application Examples

- Some or all packets require involvement of a GPP
  - Handles exceptions, manages connections, or handles higher layer processing
- Examples of L3 processing:
  - IP Fragment reassembly, IP filtering, MPOA, LANE, Multicast Forwarding, Virtual Private Networks (IPSEC)
- Examples of L4 processing:
  - Proxying, Port Mapping (NAT), TCP stream following, stream reassembly, content-based routing, QoS/CoS, Rate Shaping, Load balancing (LB)
- Examples of L5-L7 processing:
  - Content-based load balancing (CBLB), RMON-2, traffic engineering, accounting, Intrusion Detection, Virus Detection
- Many/most higher-layer functions implicitly include forwarding (routing).

Oversimplified Categorization of Applications



Categorizing Application Types and Needs

- Applications can:
  - be high- or low-touch on packet data
  - be high- or low-touch on application state
  - span a spectrum of compute needs from low-compute to very compute-intensive
- Some applications are high touch for a percentage of packets:
  - where one or more packets require high {packet, state} touch and relatively high compute to establish flow state, and
  - subsequent packets in that flow require simple forwarding
- The simplest of L4 applications can be high-touch or -compute
  - TCP/UDP checksums require touch of entire packet
    - Some modernMAC's do this per datagram frag for you, minor math to combine
  - IP fragment reassembly, TCP flow assembly require creation and management of flow state, and copies or linked lists of buffers in order to do processing on streams of packets rather than per-pkt

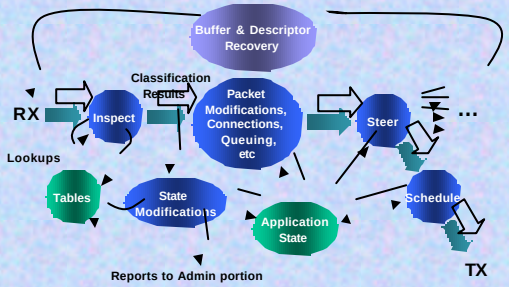
More Detailed Application Characteristics

Application	Data touch	State touch	Compute	CP touch
Switching	Low	Low/Med	Low/Med	Low
Routing	Low	Low/Med	Low/Med	Low
QoS	Low/Med	Low/Med	Low/Med	Low
Stateful Firewall	Low/Med	Low/Med	Low-High	Med/High
Proxy Firewall	Med/high	Med	Med	High
Load Balancing	Med	Med/High	Low/Med	Med/High
CB Load Balance	High	Med/High	Low/Med	High
VPN	High	Med	High	
Virus Detection	High	High	High	High
IDS	High	High	High	High

Basic Paradigm of Forwarding Plane Processing

- Examine header(s)
- Do lookup(s)
  - e.g. bridging tables, IPv4 LPM, flow identification table
- Select and Execute Actions
  - Packet (or cell) modifications
  - Application State modifications (tables, counters, flow records)
  - Queuing
  - Possibly heavy lifting such as connection management, crypto, RegEx string search…
- Transmit may also include scheduling/shaping
- Since *ingress* and *egress* are typically on different blades, "TX" and "RX" may be to/from the fabric
- Housekeeping: Buffers and descriptors must be allocated and recovered for each frame

Canonical Network Processing Flow



Challenges Faced by Network Processors

Challenges for NPs

- Infinitely variable problem space
- "Wire speed"; small time budgets per cell/packet
- Poor memory utilization; fragments, singles
  - Mismatched to burst-oriented memory
- Poor locality, sparse access patterns, indirections
  - Memory latency dominates processing time
  - New data, new descriptor per cell/packet. Caches don't help
  - Hash lookups and P-trie searches cascade indirections
- Random alignments due to encapsulation
  - 14-byte Ethernet headers, 5-byte ATM headers, etc.
  - Want to process multiple bytes/cycle
- Short-lived flows (esp. HTTP) => high rate of "exceptions"
- Sequentiality requirements within flows; sequencing overhead/locks
- Shared data structures -> locks; latency and contention costs

Processing Time Budgets

- Packet processing budgets are per packet, unless all of the packet is touched (e.g. checksum).
  - Average packet size in most apps is higher thus relaxing the budget
  - Voice applications are the pathological case
- Cell processing budgets are essentially a per-byte cost.

Media	Cells/ Packets	Size (bytes)	#/Sec	Time per unit
10 Mb Ethernet	Packets	64-1518	14.88K - 800	67.2 – 1,240 uS
100 Mb Ethernet	Packets	64-1518	148K – 8K	6.72 – 124 uS
Gb Ethernet	Packets	64-1518	1.48M – 80K	672 nS – 12.4 uS
OC3	Cells	53	~300K	~3.3 uS
OC12	Cells	53	~1.2M	~833 nS
OC48	Cells	53	~4.8M	~208 nS
OC192	Cells	53	~19.2M	~52 nS

Latency Challenges of Packet Processing

- Packet data and descriptors are new per packet
  - Inter-packet data locality is poor, so cache warmth is limited
  - Generates lots of *compelled* data cache misses on a uP
  - "Dataflow" processing does not benefit (much) from processor data caches
- Communications from cached uP to other units involves lots of cache flushing
  - Embedded processors are rarely Multi-Processor cache coherent
- Hash and LPM searches are pointer-chasing
  - Cache-thrashing if no inter-packet locality
  - uP stalls during these indirections
  - Specialized hardware or microcoded engines can help hide latency
- Techniques for hiding latency include pipelining, parallelism, and multi-threading

Locality Can Be Poor in Network Applications

- Packet data is essentially dataflow.
  - Per packet, new data arrives which is not in the uP cache; similarly any related descriptors are also new data.
  - FP engines filter packets so that only "exceptions" go the uP
  - A small % for L2/L3, up to 100% for L3-L7 go to the uP
- Data structures accessed per packet are likely to exhibit poor temporal (inter-packet) locality.
  - The relationship between a packet and other recently processed packets is likely to be poor; minor exceptions…
  - The uP data cache is not likely to have a working set for the data structures and tables accessed per packet.
  - Higher bandwidth means higher aggregation, & less locality
- The uP instruction cache may similarly thrash
  - Exception processing may require large code space

Cruelty to Memory (1): Access Size

- DRAM is desired for capacity; SRAM for speed
- Mem bursts provide best perf for 2^n-sized accesses
- Cells and packets rarely cooperate with memory
  - 53-byte ATM cells, or 48-byte payloads
  - Variable-sized packets mean possible tail fragments
  - Decapsulation and encapsulation means adding arbitrary starting alignment
    - 14-byte Ethernet headers benefit from a 2-byte prepad to align payloads
    - PPP headers and others have may byte-arbitrary length
  - SAR'ing 48-byte chunks on 64-byte mem system means 32,16,16,32-byte accesses for 2 cells
  - 65-byte packets cost as much bandwidth as 128-byte pkts
- Descriptors, queues, tables etc have lots of small accesses but may need huge capacity (e.g. flow tables.)

Cruelty to Memory (2): Life of a Packet

- Example comms per packet, simple forwarding case
  - Get buffer descriptor from allocation pool (freelist) LIFO — RD
  - Write hardware receive status *(and store pkt into buffer)* — N*WR + WR
  - Schedule ptr into queue for classification — WR
  - Read ptr from queue — RD
  - Read descriptor *(and fetch some of packet from buffer)* — RD + (2-3 RD)
  - Do 'n' accesses to a lookup table to get forwarding information — N*RD
  - *(modify L2/L3 header in buffer)* and update descriptor — WR +WR
  - Enqueue descriptor pointer into a transmit queue — WR
  - Fetch pointer from queue — RD
  - Fetch descriptor *(and fetch packet)* — RD +N*RD
  - Return descriptor to freelist LIFO — WR
- Raw bandwidth of these accesses can exceed data movement of small packets
- Number of accesses far exceeds packet data accesses

Cruelty to Memory (3): Stress and Contetion

- More complex applications may have more searches and more unit-to-unit communications per packet
- Often, scattered state is updated per packet for statistics, reassembly lists etc
- RegEx sting searches, done for content -based traffic management and for intrusion detection, are extremely memory -access intensive
- Control-plane use of shared memory makes forwarding-plane behavior less deterministic.

Solutions

- Approaches include:
  - General Purpose Processors
  - Fixed Function Si, sometimes with some programmability
  - Network Processors
- Choice is driven by many parameters
  - Speed of media
  - Complexity of the application(s)
  - Need for flexibility & field upgradability
  - Cost, Power, & Real Estate

Why a Network Processor?

- **Exploit parallelism**: Multiple processing elements
  - Packet stream contains independent flows aggregated together
  - Some amount of potential parallelism even within a flow
- **Hide Latency**: Multithreading & Pipelining increase throughput
  - Poor locality in network traffic handling
  - Lots of indirections
- Mix of SRAM and DRAM helps **improve utilization and latency**
- Optimize for **Header Inspection** and **Lookups**
  - Bit field extraction, comparisons, alignment
- Bind common special functions with lots of processing power
  - Integrated functions and/or attached coprocessors e.g. TCAM
- **Optimize housekeeping** functions and inter -unit comms
- High Integration

Special Functions Found in NPs

- Pool of latency -hiding processors
  - With header -parsing optimizations
- Lookup acceleration: hash index, hash search, TCAM
- Integrated and flexible media interfaces
- Queue Management
  - Internal communication, allocation, scheduling, WRED/RED
- CRC and Checksum acceleration
- Timer support for schedulers
- Alignment and merge support
- Crypto support: bulk encryption, key management
- "Magic memory" for fire-and-forget counters etc

Solutions Offered by Network Processors

NP's Use a Variety of Acceleration Techniques

- Keep uP stalls and cache traffic to a minimum, and provide specialized compute capabilities as appropriate
- A forwarding plane handles (ideally) most packets, uP(s) handle an (ideally) small percentage of traffic
- Offload high-touch portions of applications from the uP
  - Header parsing, checksums/CRCs, RegEx string search
- Offload latency-intensive portions to reduce uP stall time
  - Pointer-chasing in hash table lookups, tree traversals for e.g. routing LPM lookups, fetching of entire packet for high-touch work, fetch of candidate portion of packet for header parsing
- Offload compute-intensive portions with specialized engines
  - Crypto computation, RegEx string search computation, ATM CRC, packet classification (RegEx is mainly bandwidth and stall-intensive)
- Provide efficient system management
  - Buffer management, descriptor management, communications among units, timers, queues, freelists, etc.

Acceleration Techniques (2)

- Offload media management (device drivers) as much as possible
- Media processing (framing etc) is generally better done by specialized units
- Decouple hard real-time from budgeted-time processing
  - keep up with the media
  - meet per-packet/cell time budgets
  - higher level processing via buffering (e.g. IP frag reass'y, TCP stream assembly and processing etc.); **elasticity** allows for averaging of packet times
- Efficient communication among units is key
  - Easy to eat entire gain from coprocessors in communicating among units; hardware and software must be well architected and designed to avoid this. (keep **compute:communicate** ratio high.)

Acceleration via Pipelining

- Goal is to increase total processing time per packet/cell by providing a chain of pipelined processing units
  - May be specialized hardware functions
  - May be flexible programmable elements
  - Might be lockstep or elastic pipeline
  - Communication costs between units must be minimized to ensure a ***compute:communicate ratio*** that makes the extra stages a win
  - Possible to hide some memory latency by having a predecessor request data for a successor in the pipeline
  - If a successor can modify memory state seen by a predecessor then there is a "time-skew" consistency problem that must be addressed

Acceleration via Parallelism

- Goal is to increase total processing time per packet/cell by providing several processing units in parallel
  - Generally these are identical programmable units
  - May be symmetric (same program/microcode) or asymmetric
  - If asymmetric, an early stage *disaggregates* different packet types to the appropriate type of unit (visualize a pipeline stage before a parallel farm)
  - Keeping packets ordered within the same flow is a challenge
  - Dealing with shared state among parallel units requires some form of locking and/or sequential consistency control which can eat some of the benefit of parallelism
- Caveat; more parallel activity increases memory contention, thus latency

Latency Hiding via Hardware Multi-Threading

- Goal is to increase utilization of a hardware unit by sharing most of the unit, replicating some thread state, and switching to processing a different packet on a different thread while waiting for memory
  - Specialized case of parallel processing, with less hardware
  - Good utilization is under programmer control
  - Generally non-preemptable (explicit yield model instead)
  - As the ratio of memory latency to clock rate increases, more threads are needed to achieve the same utilization
  - Has all of the consistency challenges of parallelism plus a few more (e.g. spinlock hazards)
  - Opportunity for quick state sharing thread-to-thread, potentially enabling software pipelining within a group of threads on the same engine (threads may be asymmetric)

Coprocessors: NP's for NP's

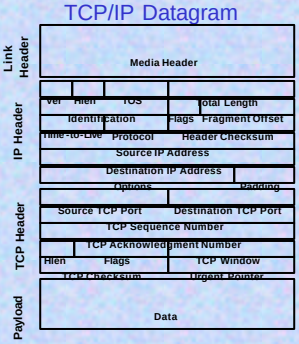
- Sometimes specialized hardware is the best way to get the required speed for certain functions
- Many NP's provide a fast path to external coproc's; sometimes slave devices, sometime masters.
- Variety of functions
  - Encryption and Key Management
  - Lookups, CAMs, Ternary CAMs
  - Classification
  - SAR functionality
  - RegEx string searches (often on reassembled frames)
  - Statistics gathering

Summary

- There are no typical applications
- Whole network processing solution partitions to forwarding plane, connection management plane, control plane
- Data rates and application complexity are increasing
- NP's address common needs and functions at a variety of performance points; exploiting the characteristics of the problem space to provide specific acceleration and integration

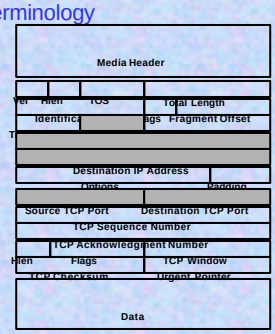
Supplementary Materials

TCP/IP Datagram



Basic Terminology

- Forwarding - Using the destination address to choose an output port for a given packet.
- Connection - Defines a single TCP/UDP connection.
- Connections are typically identified by 5 fields.



Why not just use a GHz+ Pentium?

- NP's always compete with general purpose uP's.
  - External uP speeds will continue to evolve rapidly, NP's less rapidly and with fewer releases
  - High-integration uP + sys ctrl may also compete at the low end where acceleration may be overkill
- The value proposition of NP's must more than overcome the delta between an NP's embedded uP and GP uP's.
  - Network processing is basically a dataflow problem; uP caches do not hide latency
  - Time budgets per cell/packet at high wire speeds
  - Get device drivers out of the loop

Variety of Workload Partitioning

- Pure Forwarding
  - Tables come from "outside"; all work is done in the forwarding plane
- Partitioned
  - Some percentage of "exception" packets require GP uP
    - To manage connections
    - To examine a portion of the byte stream, e.g. RegEx for CBLB
    - To handle tasks based on certain packets
  - The rest are handled in the forwarding plane
- Whole Data (e.g. encryption, string search, Proxy…)
  - All frames must be handled by a general purpose processor and/or a coprocessor; the Forwarding Plane assists

Tasks for Network Processors

- CPE, Edge, and Core applications
- L2, L2/L3, L4-L7…
- Forwarding (bridging/routing), Protocol Conversion
- In-system data movement (DMA+)
- Encapsulation/Decapsulation to fabric/backplane/custom devices
- Cell/packet conversion (SAR'ing)
- L4-L7 applications; content and/or flow-based
- Security and Traffic Engineering
  - Firewall, Encryption (IPSEC, SSL), Compression
  - Rate shaping, QoS/CoS
- Intrusion Detection (IDS) and RMON
  - Particularly challenging due to processing many state elements in parallel, unlike most other networking apps which are more likely single-path per packet/cell

A Little (Random) History

- Early entry was Protocol Engines Incorporated with their TCP/IP stack offload processor (~1989?)
- Port processors on switch chipsets (MMC, PMC-Sierra)
- Proprietary building blocks
- Smart NICs and switch chips circa 1995-96
  - e.g. Alteon
- Network building block startups ~1997…
  - Obtek/Agere, Sitera, NetBoost, C-Port, RapidStream, Solidum, Maker
- …and big guys
  - DEC (Intel), IBM, Motorola

Time Support

- Timestamps for tracking and sequencing
- Sequence numbers
- Calendar queue support
- Periodic timers
- QoS algorithm support
  - Leaky buckets
  - WRED, RED
  - WFQ
  - etc.

Acceleration Details

Accelerating Header Parsing

- Data fetch
- Field extraction
  - Byte alignment issues
- Comparisons
- Boolean computation
- Endianness issues
- Lookups using extracted fields as keys

Accelerating Lookups

- Hash table searches
  - Hash index computation
  - Table search (pointer chasing)
  - Latency hiding; speculative fetch, threads (parallelism)…
- Longest Prefix Match (LPM) searches
  - Traditional P-tries
  - Body of literature on methods of accelerating
  - Match the algorithm to the best characteristics of the hardware
- Coprocessors
  - Table search engines
  - Ternary CAMs

Accelerating Internal Communications

- Hidden costs in distributed processing systems
- Buffer allocation, buffer recovery
- Unit-to-unit communications
  - One-to-one
  - Many-to-one
  - One-to-many
  - Queues, mailboxes, magic status units
- Locking
  - Mutex
  - Counting locks/resource locks
  - Sequence locks
- Data pushing (latency reduction for consumer)
- Data copy/realign support

Partitioning on Network Processors

- At each layer, it is vital to minimize communications overheads; the **"compute:communications" ratio** is what determines if there is value in partitioning work across multiple units.

Reducing Media Management Overhead

- Standard NICs and SARs have a device driver layer
  - may require complex communications structures
  - high overhead for data movement/signalling
- Network processors don't need such a generic interface
- Many NP's provide fully integrated media
  - Fixed function (e.g. Ethernet MAC)
  - Programmable framer (microcoded)
  - Tightly coupled with the processing elements
- Many NP's provide a simple bus to connect simple external media
  - fifo-like interfaces
  - UTOPIA, IXBus etc

Fabrics

- Variety of styles
  - Packet, cell
- Many different uses
  - Interconnect many NP's for port count/switching (scaling fabric)
  - As a backplane with high bisection bandwidth
  - As a distribution medium among multi-NP compute farms
- Generally proprietary
  - Often the special sauce in an application
  - Sometimes provided as part of the family with an NP
  - Standards are emerging but slowly (e.g. CSIX)
- Fabrics generally incur an encapsulation overhead
- Convergence among NP's, fabrics, coprocessors
  - The distinctions are blurring among these...

Evaluating Network Processors for Your Needs

Benchmarking

- How do we measure and compare performance?
  - Apps are sensitive both to app. behavior and to specific traffic
  - Categories of apps is not sufficient; behavior is sensitive to specific implementation of each app.
  - No industry standard benchmarks or traffic flows
  - Measure of simple forwarding rates is *not* a good indicator of higher-level app performance
  - Simply selecting several apps and writing standard implementations and traffic is not sufficient; someone needs to recode completely for different environments/NP's/API's. Not like specmarks etc.
  - Developers need guidance on which platform is best for their needs; this is one of the biggest challenges in selling NP's

Metrics

- We can measure many different relevant metrics
- pps, pps/$, pps/Watt, pps/sq. in,
- behavior at core, at edge, inside of server farm, etc (traffic)
- type of traffic (e.g. webbench, ftp, email, telnet, voice…)
- Reality constraints ($, W, area)
- Ease of use; both software and board-level designs
- General-purpose processors may compete well on raw performance but fall down on one or more of these others
  - uP + sys ctrl can be expensive and/or high power
  - Dataflow portion (new packet contents, descriptors) cause cache-fill stalls. Also data locality may be poor in application data structures referenced per-packet. Both of these factors can slow the fastest uP down to the speed of (highly contended) DRAM.

Other Evaluation Criteria

- DRAM and SRAM bandwidth(s)
  - Not just 2* wire Gb/Sec. Account for all chatter about the packets, and alignment/fragmentation degradation of performance
- Memory arbitration policies and how it affects your usage of the device
- Availability of desired media interface/support chips
- Availability of family/3rd-party coprocessors
- Ease of attaching custom media and coprocessors
- Use of standard interfaces
- Access to detailed benchmark information
  - Also details of how the benchmarks are coded, traffic, etc
- Balance of forwarding, compute, and host communication resources for you application

Network Processor Software

Network Processor Value

- As the networking market matures, vendors are increasing value through software services.
- Time to market is increasingly important in the race to add features.
- New services often need to be added after the original platform was designed.

Network Applications

L2/L3 Forwarding

- Identify destinations of incoming addresses
- Lookup output port and next destination
  - Longest prefix match
  - Exact match
- Remove/add headers
- Update headers as appropriate
  - TTL and checksum for IP
- This application generates exceptions when the network is reconfigured.

Network Address Translation (NAT)

- Used to map private addresses to “real” addresses
- Can cause source/destination addresses to be modified as well as port numbers
- Connection lookup for existing state.
- New connections require resource allocation.
- Certain protocols require modifying the application data
  - FTP control connections
- Exceptions are generated for all new connections.
- For some connection types, all packets will go to the control plane.

IPSec VPN

- Packets must be associated with IPSec connections (1 or more table lookups).
- High speed bulk encryption is required
- Connection rates depend on usage model.
  - For remote access, each time a new user connects
  - For branch offices, at user configurable intervals.
- Connection setup requires fast public key operations.

Firewalls

- Data plane uses connection table to determine whether to allow or deny the packet.
- Each new connection generates an exception.
- Control plane may need to see a number of packets before it can cache a decision.
- Often used in conjunction with other applications (L3 forwarding, NAT, …).

Load Balancing

- Distribute load across multiple servers.
- Connection table is used for established connections.
- New connections require control processing.
- Simple load balancing can be determined with a single packet.
- Content load balancing requires TCP termination.

Quality of Service

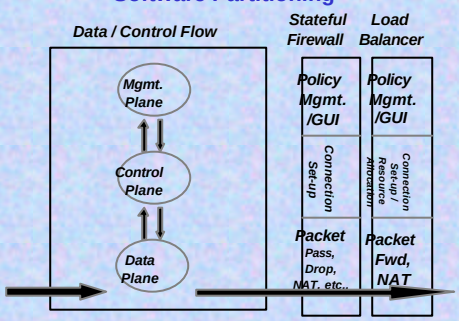
- Incoming packet must be identified and mapped into classes (1 or more table lookups).
- Policing and scheduling are performed on each of the classes.
  - Scheduling may require more queues than provided by hardware.
- Exceptions are generated by “users” (slow rates).

Protocol Conversion

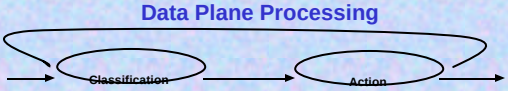
- Convert from one media type or encapsulation to another media type or encapsulation
  - Frame Relay to ATM
  - IPv4 to IPv6
- Incoming packets are identified through table lookups.
- Exception rates depend on the protocols being converted.
  - Frame Relay to ATM ----> low rates
  - IPv4 to IPv6 ----> high rates

Software Partitioning

Software Partitioning



Data Plane Processing



Classification
- Header parsing to extract fields
- Connection table lookups
- Route lookups

Action
- Encapsulation / decapsulation
- Header modification
- Update counters
- Queuing and scheduling operations

Control Plane

- Handles exceptions from data plane.
  - New connections
  - Packets of unknown data types
  - Packets needing additional (more complex) processing
- Some higher level protocols may run here.
  - ATM signaling
  - Frame Relay Signaling
  - RSVP
  - PPP Session negotiation

Management Plane

- Provides centralized control for device.
- Management plane typically provides the configuration agent for the applications.
- Higher level protocols may run here
  - Routing
  - Authentication Protocols
  - SNMP
- Many protocols can run either in the control or management plane. Developer must make tradeoffs

Network Application Characteristics

Exception Handling

- A key characteristic of network applications is how often exception packets arrive.
- The data plane passes exception packets to the control plane.
  - Processing is too complicated for data plane.
  - Processing would take too much time on data plane.
- Exceptions often results in the modification of state shared between the control and data planes (e.g. connection tables).
- Different applications have different exception processing needs.

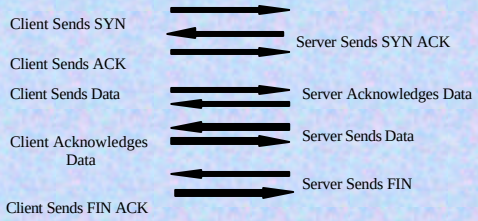
Network Reconfiguration Exceptions

- Applications that are doing forwarding need to respond to network reconfiguration events.
- These events can be routing updates, or detecting a link failure.
- Control/Management planes will update the forwarding tables.
- Data plane will continue forwarding with the updated tables.

Connection Exceptions

- Some applications needs to perform higher level processing when new connections are established..
- All packets on the same connection are sent to control plane until connection table is updated.
- Requires at least 1 exception packet per connection.
- Connections must be removed at a later time (through aging, or exceptions when connections are closed).
- The connection rate is at the mercy of traffic characteristics. Many applications can be limited by the connections per second.

TCP Handshake Example



TCP Endpoints

- Some applications must see TCP payload before making a decisions.
- Connection must be established before payload is sent.
- It may not be possible to put full TCP stack in the data plane.
  - Code space
  - Complexity and state management
- After control plane sees the payload, it may need to open a new TCP connection to another machine. (Control plane is a proxy.)

Connections per Second is a Driving Metric

- Most applications are connection oriented. This means that some number of packets require processing on the Control Processor before we can turn the rest into a forwarding problem (common optimization)
- The number of packets per connection that require Control Processor, combined with the lifetime of connections, drives this.
  - Simple switching/routing might require a single packet per connection, but likely zero since tables update glacially.
  - Proxying requires ~8/conn to sink and establish and eventually terminate flows, then TCP splicing to forward the rest
  - Most HTTP connections are quite short, ~10 packets total.
  - ftp connections (the data plane) are generally quite long.
  - VPN, IDS, and anything requiring reassembly followed by processing will send **all** packets to the uP