Using Ada95 to Build Software for a Gigabit Layer 7 IP Networking Device: Ada’s No Big Deal Anymore

Mike Kamrad
“senior” Software Engineer
kamrad@TopLayer.com
+1.508.870.1300x139
FAX +1.508.870.9797
www.TopLayer.com
What You’ll See

♦ A Really Bitchin’ Data-Comm Product
♦ Architecture
♦ Ada Software Development Environment
♦ Obstacles and Lessons Learned
♦ Restricted Usage RTOS
AppSwitch™ Family

The First Layer 7 Device with e-Application Control
Who is Top Layer Networks?

- Founded January, 1997
- 155+ Employees
- Headquarters in Westboro, MA, USA
- Strong executive management team
  - Chipcom, Fore Systems, DEC, Wellfleet/Bay Networks
- World-class engineering team
  - DEC, Fore, Chipcom, 3Com, etc.
  - Major experience in ASICs and switches
- Private Company - Raised over $30M in funding
- Shipping Products: AppSwitch 2500 - October ‘99; AppSwitch 3500 - June ‘00
- Over 125 customers… and growing!
What is the Top Layer AppSwitch?

- The AppSwitch is a high performance L4-L7 device with a powerful set of flexible features, called e-Application Control.

- It provides flexible solutions to today’s IP networking challenges including congestion, accounting, monitoring, high-availability, responsiveness, and security.
What is e-Application Control?

♦ A powerful set of flexible high-performance features...
  ♦ Security Augmentation
    – Packet Filter Firewall with Syslog Reporting
    – Filters for DoS attacks
    – Flow Mirroring for IDS
  ♦ Application Accounting/Monitoring
    – TopFlow protocol
    – TopFlow Data Collector
    – TopView Graphs
  ♦ Application Traffic Management
    – Application QoS
    – Application Balancing
    – Application Redirection

...built on a powerful policy-based management model which includes a unique Application Definition Library
What do we do with Applications

- Assign relative priorities to Applications
- Weighted Round Robin Queuing
What do we do with Applications

♦ Graduated Priorities

- Priorities are altered as size of flow increases
What do we do with Applications

- Assign guaranteed bandwidth to Applications
- Uses the static queues
- Will guarantee per flow bandwidth up to 8 Mbps per flow
- Up to 128 flows can be set

Bandwidth Guarantees
What do we do with Applications

*Firewall*

- A firewall policy can be set by application between zones
What do we do with Applications

♦ TopFlow displays message flow statistics

- Output stats on packets sent by application and user + L3/L2 data

- TopFlow output:

Background Engine

FastPath Process

Applications:
- Lotus Notes
- Oracle
- FTP
- Internet Explorer

K.N.A.N.
- AS2512 - 12 10/100 TX ports + 2 100FX ports
- 32,000 simultaneous flows
AppSwitch 3500

- AS3502 - 12 10/100 TX ports + 2 Gig ports (1 redundant only)
- 256,000 simultaneous flows
- Additional packet processing capability
Technical Differentiators

♦ **“Touch every packet”** switching engine
♦ Stateful packet inspection
♦ Datalink independent LAN/WAN switching
♦ Hierarchical Hybrid Queuing (HHQ)
♦ Customizable Application Policy Library
♦ “Automatic” operation
♦ Scaleable, extensible architecture
Architecture
Architecture Matters!

Key components
- TopFire™ Programmable Silicon Switching Engine
  - Architected, designed, and patented
  - Layer 7 at full wire speed at every port
  - Very scaleable and extensible
  - Same architecture/software in entire family
- TopPath™ Application Flow Switching
e-Application Control Architecture

Packet Processing

- Incoming Data
- Forwarding Tables
- Application Definition Library
- Session Data
- Application Policies

L2/L3

L4

Tx Queue

Priority Output Queues

Stateful Application Control
AppSwitch 2500

TopFire™ chipset

Architecture - A Closer Look

MOM Chip

RE Chip

QM Chip

Forwarding Engine

Lookup Tables Memory

SDRAM

Fast Path Code, System Data, Memory

Descriptor, Tables Memory

Queue Manager

Packet Memory

RAMbus DRAM

100 Mbps MII

100 Mbps MII

MII Octal MAC

MII Octal MAC

Background Engine

10/3/00

Top Layer - "Layers above the Rest"
Scalable Architecture

AppSwitch 3500

- RE Chip
- RE Chip
- SRAM
- RE Chip
- SDRAM
- GMOM Chip
- QM Chip
- SRAM
- QM SRAM
- RAMbus DRAM
- Packet Memory
- Fast Path Code, System Data, Memory
- Descriptors, Tables Memory
- 1000 Mbps
- 100 Mbps MII
- Dual Gig + 16 10/100 MAC

Forwarding Engine
Lookup Tables Memory
Queue Manager
Application Engine
Background Engine
Major Software Components

- **Forwarding Engine** - The main switching component to establish and maintain message flows, per network policies
- **Background Engine** - The management control component that interfaces with the network administrator
- **Application Engine(s)** - Protocol specific accelerator to handle complex protocols
Forwarding Engine (FE) Architecture

- Custom RISC Core (ARC)
- Extensive Proprietary HW Assist
- Tight Polling Loop - NO INTERRUPTS
- Application “wrapper” and Slowpath coded in Ada
- Fastpath and Application “policies” handcrafted in assembly
- Connection rates and throughput are key performance requirements
Background Engine Architecture

- Based on same ARC chip as FE
- Event-driven system where events are:
  - Arrival of message packets
  - Time expiration
- Multi-tasking application to respond to various events
- Division of application between
  - Core modules
  - Extension modules
- Throughput and event processing are key performance requirements
Background Engine Architecture

Extensions

Core

10/3/00  Top Layer - "Layers above the Rest"
Application Engine (FE) Architecture

♦ Based on same ARC chip as FE
♦ Tight Polling Loop - NO INTERRUPTS
♦ Basis infrastructure, written in C
♦ Application specific software written in Ada or C
♦ Throughput is key performance requirements
Why Ada?

- **Best combination** of language features for high reliability and portability
  - Strong typing
  - OOP
  - Multi-tasking
  - Exception handling
- **Founder’s familiarity** with Ada and frustration with traditional languages
- **Implementation based on GCC**, best chance for ARC target
Ada Tool Chain

- GNAT because GCC targeted ARC
- **Hosts:**
  - Solaris on SPARC
  - Linux on PC
- **Targets:**
  - Motorola MPC860 (PPC based) for WAN
  - ARC
  - Linux (for simulation)
Argonaut RISC Core (ARC)

- Argonaut, British producer of computer game technology
- Provided as a “soft macro” (in VHDL), configurable to customer needs
- Basis for Forwarding Engine, Background Engine and Application Engines
Modes of Ada Execution

♦ While a multiprocessor product, no distributed execution mode considered
♦ Instead,
  ♦ Forwarding Engine and Application Engines executes as single task, i.e. no RTOS
  ♦ Background Engine application executes with restricted tasking operations, per the Ravenscar Profile
  ♦ Control information is passed between Engines as special IPC protocol
Ravenscar Profile (+)

- No task hierarchy
- No dynamic creation of protected objects and tasks

Tasks
- No entries
- No abortion nor ATCs
- No select statements
- No user-defined attributes
- No dynamic priorities
- No requeue
- No formal termination

Protected Objects
- Limited to one entry
- Limited queues to one caller
- No requeue
- Barrier conditions limited to single Boolean variable

Interrupt handlers defined through protected procedures
Priority-based scheduling with time-slicing within priorities
Obstacles and Lessons Learned
Limitations on Ada Usage

- **Not used**
  - Real types
  - Goto
  - Annexes E-H
  - Functions returning unconstrained objects
  - Predefined I/O

- **Limited use**
  - Child generic units
  - Formal package parameters
  - Dynamic slices and aggregates
  - Length attribute
**Significant Features of Ada**

- **OOP** - Reinforced relationships among "modules", "interfaces" and "state block"
- **Root_Storage_Pool type** - Supports total memory management
- **Controlled and Limited_Controlled types** - Get "closure" on resource usage
- **Interfacing to C** - Because there’s lots of legacy and third-party software
- **Various Interfaces to hardware**
- **GNAT’s pragma Assert**
Obstacles/Solutions

- **Learning Curve**
  - Intensive internal training and advocacy

- **Incomplete Tool Chain for ARC**
  - GCC consultants to bridge the gap

- **Inadequate RTOS - GNAT RTS and RTEMS too excessive and non-unified for ARC**
  - Internal development of lightweight RTOS
Other Lessons

- Guidelines to guide user through “choices”
- Budget resources for tool support
- Tool chains reflect their legacy
- Be supportive to new user through the “niggling” period
- Watch out for generic foot print
- Pay attention to exception handling costs
Restricted Usage RTOS
Our Restricted Ada RTOS

- Built to work with GNAT
- Simplify the functionality of RTOS to only support Ravenscar-like profile
- Reduce the “thickness” of layers in GNAT RTOS
- Simply the “kernel” for bare machine execution
Restricted RTOS Organization

Ada application

Ada-specific runtime routines

C application

Ada-specific runtime routines

OS binding to Kernel

RT POSIX compatible Kernel

RT-POSIX binding to Kernel

Kernel binding to processor
Restricted RTOS Organization

Ada application

Ada-specific runtime routines

RT POSIX compatible Kernel

C application

Kernel binding to processor