

Portable Open Protocol Software

Tony Downes – Principal Technologist, NPD
Data Connection Ltd. (DCL)
October 16, 2007

Data Connection Ltd. (DCL)

Networking Protocols Division

Protocol software for
 OEMs



Internet Applications Division

Unified Communications
 software for SPs



Enterprise Connectivity Division

SNA software for OEMs
 and Enterprises



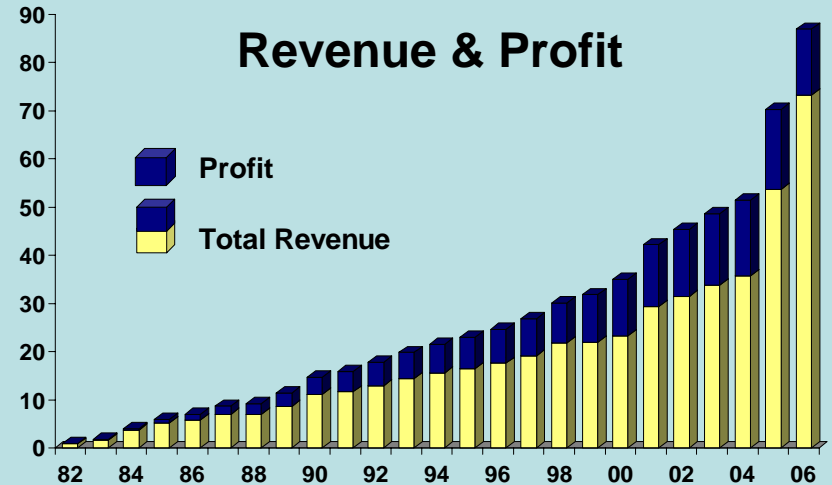
MetaSwitch

Class 4/5 softswitch
 solutions for IOCs /
 CLECs



300+ deployments
 3m subscriber capacity

Revenue & Profit



Employees



WW Leader in Networking Protocols

IP Routing

- BGP
- OSPF-TE
- ISIS-TE
- RIP
- PIM
- IGMP

MPLS

- MPLS
- RSVP-TE
- LDP
- VPNs
- VPLS

Optical

- GMPLS
- O-UNI/NNI
- E-NNI
- LMP

VoIP / IMS

- SIP
- Megaco/H.248
- MGCP
- Diameter
- SBC

ATM

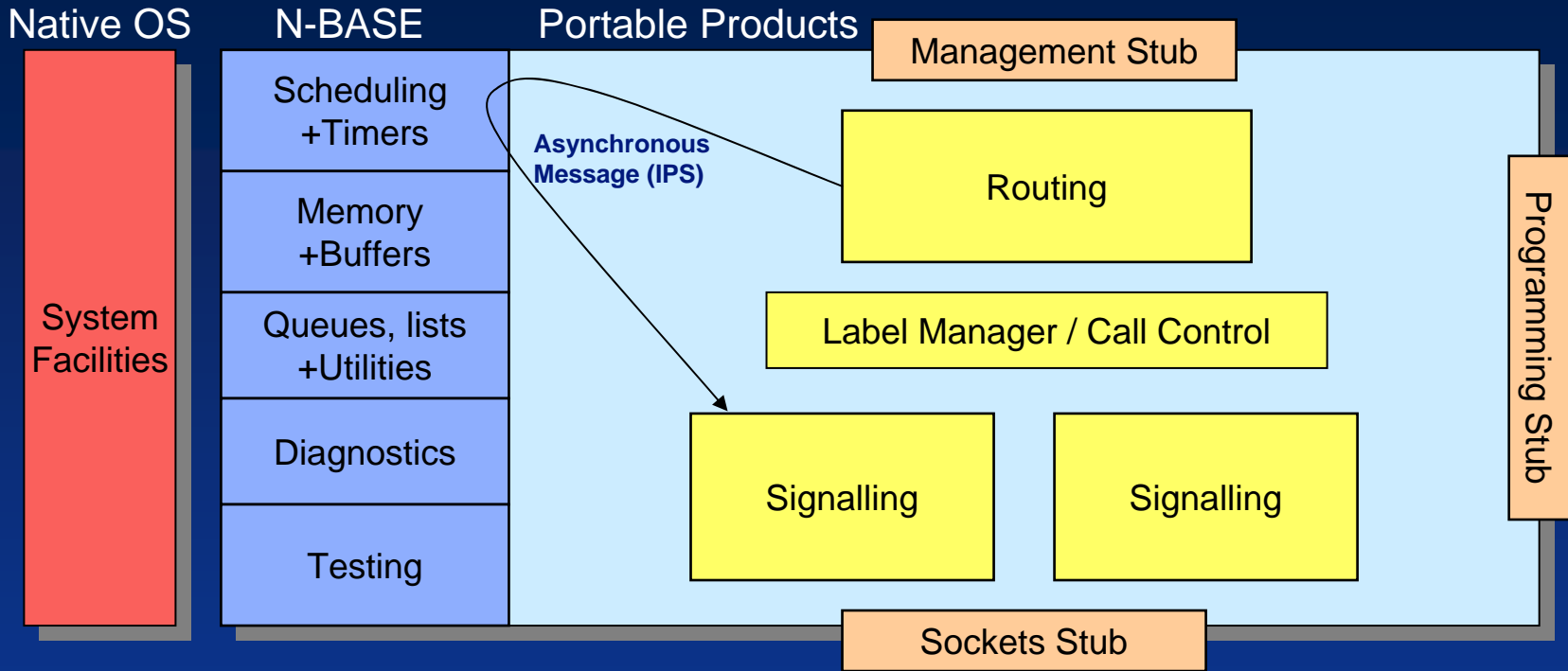
- PNNI
- SVCs
- UNI
- IISP
- ILMI



Key Protocol Software Requirements

- Portability
- Optimize CPU and memory performance
- Facilitate modularity
- Component-level testability
- Development processes and tools
 - Development productivity
 - Code quality
 - Maintainability
 - Remote debugging
- Flexibility
 - Distribution
 - Threading
- High Availability
 - Fault tolerance
 - Hot Software Upgrade

Data Connection Architecture



Key

- Core Protocol Code (100% portable)
- DCL API mapping (platform/OS)
- Stubs (platform/OS)

End Result:

- **Modularity**
- **Flexibility**
- **Scale**
- **High Availability**

- Diagnostics Requirements
 - Code must be easy to debug
 - Implementing diagnostics for a new N-BASE development should involve minimal/no additional effort
 - Diagnostics should allow remote debugging
 - Diagnostics should not impact performance in the field

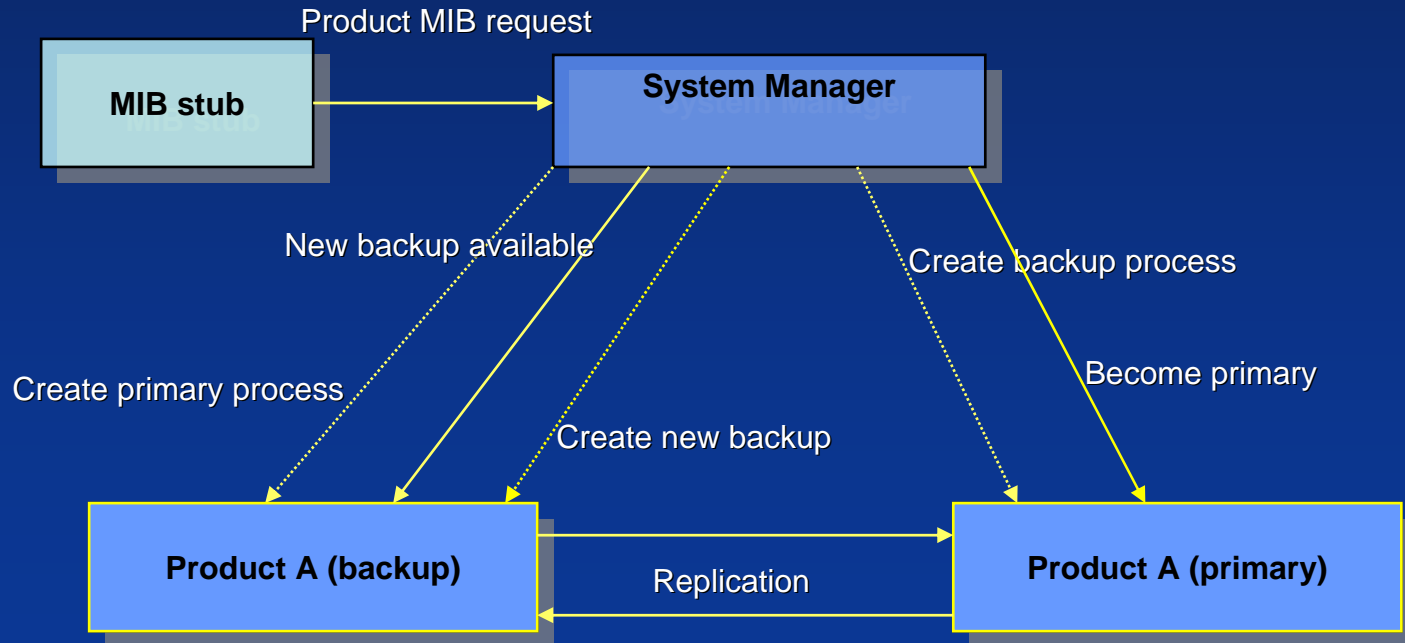
- DCL Diagnostics
 - Problem Determination logs
 - IPS trace
 - Memory diagnostics
 - Internal trace
 - FSM tracing
 - Memory dumping
 - Memory verification
 - Memory checking
 - More...

High Availability

- Reduce effort to implement High Availability for each protocol product
- High Availability == Fault tolerance with dynamic state replication to a backup (or persistent memory)
 - Stateful restart
 - Warm standby
 - Hot standby
- Two key parts to High Availability
 - State replication
 - System Management - coordination of primary and backup
- Hot software upgrade

High Availability Functional Overview

- System Manager manages the fault tolerant environment transparently
 - Creates backup process if required
 - Initiates replication procedures
 - Handles failovers



- Same underlying mechanism as for fault tolerance
 - Locations and location groups
 - Inter Location Transport (ILT)
 - Hardware Manager
- Permits distribution of N-BASE components across multiple processors or cards
 - Performance
 - Data locality
- Distributed system can be fault tolerant also

Hot Software Upgrade

- Run up-level of software in backup location, then initiate a failover
 - Similarly for Hot Software Downgrade
- Requires translation of IPs between the (down-level) primary and (up-level) backup locations
 - Processes implement translation modules for replicated CBs
 - Up-level always does translation
 - Often translation is not required at all!

- DCL has a uniquely successful networking protocol solution
 - Well proven business and software models
 - Portable, modular & open architecture
 - Effective tools and diagnostics

- Which allows system vendors to focus on
 - Application acceleration
 - Platform re-use
 - Time to market
 - Engineering efficiency
 - Product differentiation

BACK UP SLIDES

Elements of a Solution

- Isolation layer: The N-BASE API
 - Protocol code never makes OS calls
 - Implementation under API can change from platform to platform
- Asynchronous interfaces
 - As easy to write as synchronous interfaces
- Component modularity for reliability and distribution
- Flexible scheduling for any threading model
 - Including efficient use of timers
- Performance
 - Code is optimized for OS when actually compiled in a production build
- Diagnostics

Asynchronous Message Passing

- Asynchronous IPS* interfaces between N-BASE processes
 - Allows distribution between OS processes or hardware
 - Can be mapped to synchronous if desired
- Implemented with pointer assignment and call-back functions
 - Sender calls a process, passing in pointer to message
 - Message is put on “queue” of target process
 - N-BASE schedules target process, and calls its receive proc
 - No memory copies or context switches
 - Performance identical to function call APIs
 - Allows prioritization and flow control
- Can be replaced with OS messaging if desired
 - No assumptions about location of destination process (distribution)

*IPS= Inter Process Signal

- Work enters the N-BASE in the form of an IPS
 - Stub builds IPS and sends to target process – IPS is queued
- N-BASE scheduler is called by external code
 - N-BASE calls back those processes that have work to do (receive procedure)
- Provides total control over how much work the N-BASE does
 - Work can be prioritized
- Allows for any threading model
 - Multiple threads, multiple processors
 - Multiple threads, one processor
 - Just one thread doing all work in the system
- No assumptions on where N-BASE processes are
 - Hence allows distribution

Performance (Memory Manager)

- Message passing and N-BASE API do not impact performance
- Implementation of N-BASE memory manager can improve performance
 - In comparison with direct pass-through to OS
- Default memory manager implementation allocates large memory blocks from OS
 - Blocks then subdivided and allocated to processes
 - Controls frequency of requests to operating system
 - Lots of small requests from a process results in few calls to the OS
 - Reduced memory fragmentation
 - Separate storage pools for buffers and control blocks - buffers have short lifetimes
- Also allows for rich memory diagnostics for debugging

Performance (Buffer Management)

- **IPS is just a C-structure**
 - Default implementation of N-BASE is to allocate a piece of memory to hold an IPS – a buffer
- **Buffer memory has different characteristics to normal memory**
 - Transient nature
 - Allocated and freed more quickly
 - Use to pass data received on the wire into (and out of) the N-BASE
 - Memory copies should be avoided!
- **Solution:**
 - Separate N-BASE APIs for IPS buffers and normal memory
 - IPS buffers have separate “control” and “data” parts
 - Products only access “data” part through well-defined macros
 - Allows for non-contiguous memory, and different types of memory