



CloudShield®

**Integrating Applications with the
CloudShield ONSP™**

A CloudShield White Paper

August 2004

Integrating Applications with the ONSP

Table of Contents

I.	Executive Summary	3
II.	Introduction.....	3
III.	A Fresh Approach to Network Application Development - Open Network Services Platforms	3
	A. ONSP Platform Capabilities Set	4
	B. How Take Advantage of the ONSP	5
	C. Understanding ONSP Applications Architectures.....	5
IV.	CS-2000 Application Architectures.....	6
	A. Rule-Driven IDS-like Applications	6
	B. NetFlow Applications	7
	C. Libpcap Library Applications	7
	D. Direct Network Interface Card (NIC) Access.....	7
V.	Deployment Architectures	9
	A. Single Box Solution	10
	B. Distributed Multi-box Solution.....	10
VI.	Tools for the Integration / Applications Developer	10
	A. An IDE for Open Network Services	11
	B. RAVE – Packet Processing Programming Language.....	11
	C. Extensible Reference Utilities.....	13
	D. CloudShield Data Plane API.....	14
	E. Data Export Options	15
	1. ODBC / SQL.....	15
	2. SNMP.....	15
	3. XML or SOAP (via HTTP).....	16
	4. NetFlow Data Export	16
	5. File-based Communication (HTTP or NFS).....	16
	6. Raw Ethernet Packet Export	16
VII.	Conclusion and summary.....	16
	Arbor Networks®: Seizing a Market Opportunity	18

Integrating Applications with the ONSP

I. Executive Summary

CloudShield provides a platform that vendors of packet processing applications use to deliver multi-gigabit line-rate performance. Although line-rate performance for deep packet inspection at rates up to 2.5Gbps (OC-48) is necessary, it is not sufficient for marketplace success: in order to meet time-to-market goals, it must be easy to port existing packet processing applications to the CloudShield platform.

This white paper shows how to easily port applications to the CloudShield platform using the CloudShield software developer's kit (i.e. Open Network Services IDE™), the RAVE™ packet processing programming language, and example RAVE applications that provide key blocks of functionality (such as access control lists) that are ready to integrate.

Different packet processing application architectures are examined, and the paper shows how these are easily mapped into implementations on the CloudShield platform: as an integrated, single-box solution, or as a large-scale distributed solution.

The result is that the complete package of the CloudShield platform plus the developer's kit offers a unique solution to the problem of scaling packet processing applications to multi-gigabit line rates while achieving a time-to-market measured in weeks.

II. Introduction

Network bandwidth and application volume increases have outpaced most commonly used network platforms. The implication for solutions providers is that market opportunity of existing applications is diminished either because they cannot keep pace with their customers' bandwidth and service requirements, or the remedies to scaling the application are too expensive or too cumbersome to be viable. An additional implication is that solutions providers are not able to add the next set of required features because their chosen platform does not provide the horsepower to deliver the features at the required network speeds.

Using any number of available deployment scenarios, CloudShield provides a fast path to multi-gigabit solutions for a solution developer's current applications with minimum re-development. The CloudShield CS-2000™ can process packets up to 5 Gigabit per second and provides the tools and interfaces needed to allow your solutions to quickly reach new performance and functionality levels.

This paper explains the applications model for the CS-2000 Open Network Services Platform (ONSP™), and the various methods of taking existing applications and very quickly boosting their performance and/or add value using it.

Let's begin with a short but powerful example of the platform, the ease of integration and the potential business impact using the CloudShield ONSP

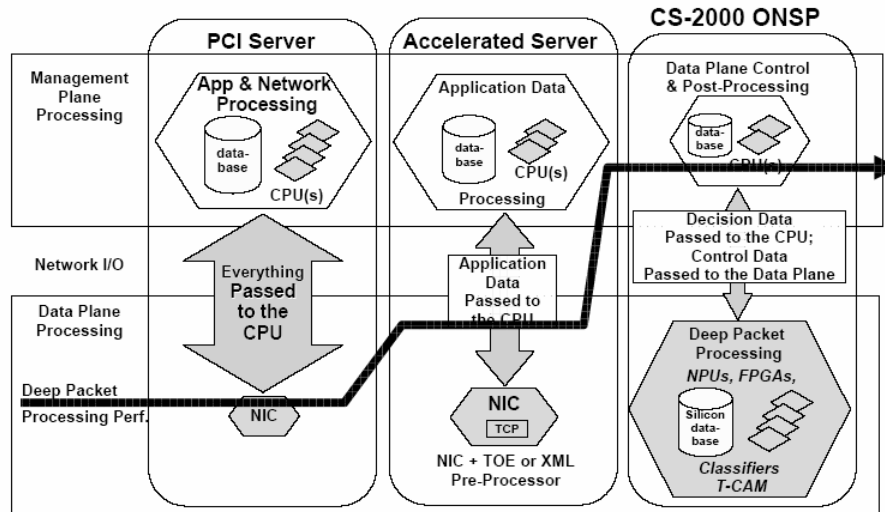
III. A Fresh Approach to Network Application Development - Open Network Services Platforms

CloudShield was founded on the premise that, with the explosion of bandwidth in both public and private networks, current CPU-based computer architectures are not viable platforms for network services and applications—a network-focused computing architecture was needed. Moreover, since business applications development does not require direct programming of the micro-processors, but instead use high-level programming languages and interfaces, network applications developers could be afforded the

Integrating Applications with the ONSP

same benefits. This laid the foundation for the Open Network Services Platform: a packet processing or network applications server.

The diagram below illustrates how the ONSP architecture differs from both a standard PCI-based server and a server with some accelerated hardware assist at the network level. Instead of passing packets or packet data – with or without some hardware accelerated processing - up to the server’s CPU for ‘per packet’ processing functions, the ONSP positions the processing resources directly in the data plane. The ONSP includes a Linux server for all the processes that are not explicitly for per packet processing (e.g. user interface, data base admin, configuration, heuristics, etc.). The result is the programming flexibility of the standard server together with ASIC-like per packet processing performance.



A. ONSP Platform Capability Set

1. Structured Packet Operations Programming

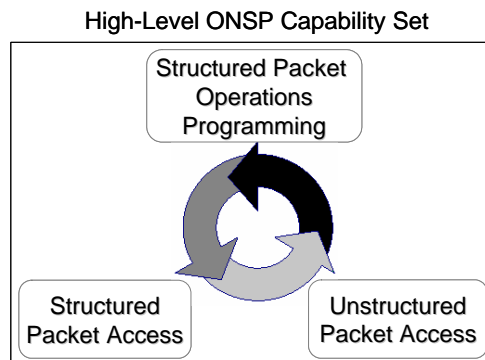
- Programming language for packet processing operations — RAVE
- Flexible, complete packet content control (read, write, create, modify)
- Interaction with management plane applications and databases
- Build flexible state machines
- Perform math operations

2. Structured Packet Access

- Up to 500,000 bi-directional ACLs
- Keep state for 500,000+ flows
- 512Mb database memory
- Bi-directional, shared memory access
- Dynamic-between packet-table updates

3. Unstructured Packet Access

- Regular expression / pattern match searches
- Search entire packet-any bit sequence
- Custom pre- and post-processors (e.g. Normalization, decanonicalization)
- 32,000 search strings @ 1024 bytes
- Up to 10 Gigabits per second of Deep Packet Inspection capacity



Integrating Applications with the ONSP

B. How Take Advantage of the ONSP

Vendors of packet processing applications typically consider the ONSP for one or more of the following advantages:

- Extending an existing application with line-rate gigabit packet processing (example: adding mitigation to a carrier DDoS identification platform)
- Scale an existing server application to multi-gigabit line rates (example: identifying and blocking port 80 based file sharing applications on a gigabit LAN segment)
- Developing a new (“greenfield”) application that requires multi-gigabit line rate packet processing

In all cases:

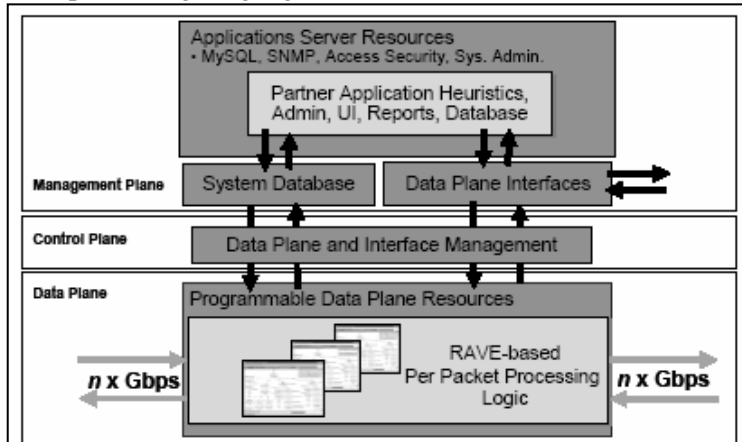
- The flexibility of software, the line-rate throughput of ASICs/hardware
- Re-programmable to adapt to rapidly changing market needs
- An architecture that meshes with existing Linux/Windows packet processing apps

C. Understanding ONSP Applications Architectures

The general architectural approach for deploying packet processing applications on the CloudShield platform is to split functionality into *management plane* and *data plane* functions. Data plane functions are then coded in CloudShield’s RAVE packet processing language which is discussed in more detail in section VI. Management plane functions continue to run on a general purpose computing platform with virtual memory, mass storage and device management provided by an off-the-shelf operating system such as Linux or Windows.

In the real world of packet processing application design, it can sometimes be difficult to identify data plane vs. management plane functions. In general, functions that belong in the data plane are those that perform any of the following:

- Examine every packet
- Manipulate a counter or compute a value based on packet contents
- Identify packets that belong to the same flow
- Modify any part of the packet
- Block, Filter, Forward, Copy, Capture packets
- Add or remove a layer of packet encapsulation (e.g. adding or removing a GRE tunnel header)
- Perform a string search on packet contents
- Perform an associative memory lookup on some fragment of a packet (e.g. an MPLS label)
- Access control list (ACL) functionality



The ONSP software architecture defines three distinct layers: a Data Plane for ‘per packet’ processing functions; a Control Plane for controlling and interfacing with the Data Plane; and a Management Plane, for off-line analysis, database, user interface, application and device administration. Within the ONSP both the Data and Management Planes are programmable layers. RAVE is the Data Plane programming language and the Management Plane is a hardened Red Hat Enterprise Linux. The Control Plane runs CPOS, CloudShield’s run-time operating system, which provides a bi-directional interface for getting data from, and delivering data and instructions to, the Data Plane.

Integrating Applications with the ONSP

- Any packet function that is stateless or requires a small amount of state (less than 255 bytes)

Functions that belong in the management plane include the following:

- User interaction
- Statistical reporting
- Database transactions (2-phase commit)
- Data mining
- Maintaining a network topology
- Analyzing aggregate packet information to determine policies or rule sets
- Archiving contents from several packets, or any function that requires a large amount of state
- Application Heuristics

Another way of summarizing the distinction is to say that management plane functions pertain to “what” needs to be done with packets (i.e., packet policy), and that the data plane functions are the “how”: the mechanism that implements the policy.

IV. CS-2000 Application Architectures

While understanding the distinction between data plane and control management plane is important, it remains a fact that many packet processing applications were never designed or coded with this distinction in mind. Even in those that were, it’s entirely possible that there was no original requirement that the application be designed such that it can easily be distributed. Classification, analysis, capture and forwarding functionality may be tightly coupled, often using in-memory data structures or objects.

Fortunately, even in these cases, the CloudShield system makes re-writing large portions of the application unnecessary by offering flexible deployment and porting options. There are a variety of approaches that can be used, depending on the needs of your application. In general, we’re going to provide rules of thumb based on the level of packet processing abstraction present within the application, as follows:

- Rule-driven IDS-like applications (similar to Snort or other IDS tools)
- Applications that use aggregated traffic information from NetFlow
- Applications that use *libpcap* to examine or capture specific packets of interest.
- Applications that directly access the network driver and don’t use any higher level libraries or aggregation.

A. Rule-Driven IDS-like Applications

The open source sniffer-come-IDS, Snort (<http://www.snort.org>) is the inspiration for or the basis of many packet processing applications. Its popularity is the result of its flexible packet inspection and classification that can be easily modified through creation of well understood rules. The ONSP’s structured programming language, RAVE, allows for the development of similar rules engines and the platform offers several mechanisms for importing rules. The sample RAVE program insert on page 8, illustrates a simple packet classification/selection engine where the rules are imported as a simple table. More sophisticated multi-tiered selection engines are feasible.

With packet identification and classification completed, the remaining effort is in exporting results and/or captured from the RAVE program. Automated support is available for exporting capture files (see section VIII) and the RAVE variable manager in the IDE makes short work of defining and controlling exported variables.

Integrating Applications with the ONSP

B. NetFlow Applications

Many applications are based on processing NetFlow records from routers. The key benefit is that the application receives aggregated information about all traffic flows in the network vs. having to process every single packet. This means that less processing is required to analyze network traffic, but the downsides are as follows:

- There is a typically a performance hit to routers when provide NetFlow information. This can be attenuated by using “sampled NetFlow” – i.e. examining a subset (sample) of packets passing through the router. But this may result in important information being lost.
- Your customers may not be willing or able to turn on NetFlow in their routers. It can be hard to find a version of the router operating system that has all the features the customer needs while avoiding specific bugs, and still have NetFlow working in that same release.

CloudShield solves both of these problems by providing a reference RAVE program that performs line-rate NetFlow record generation. The CloudShield platform runs at line rate, all packets are examined (i.e., there is no packet sampling required at high data rates). Also, because it is separate from the router, the router’s performance is not impacted and there is no need to use different versions of the router’s operating system.

To ensure NetFlow data collectors and network monitoring applications are not overwhelmed by the volume of NetFlow data generated by 100% packet visibility on multi-Gb/s links, the CloudShield NetFlow Data Export utility can be configured to collect and deliver only the NetFlow data of interest to your application or other functionality. *Flow_ID*, CloudShield’s NetFlow Data Export utility is discussed further in section VI.

C. *Libpcap* Library Applications

The packet capture library (*libpcap*, or simply *pcap*) has emerged as a de-facto API for applications that want to examine or capture packets from network interfaces¹. Applications that use this library typically use a rule-set to specify which packets to capture, and generate a packet capture file (in “pcap file format”) containing those packets.

Porting these applications to CloudShield is a straightforward process. The packet capture functions are translated into RAVE code, using the free examples provided in the CloudShield developer kit. These examples contain code to generate a file containing captured packets in exactly the same format as *libpcap*.

The *libpcap* file can be exported off the data plane using one of the file-based options outlined below (HTTP or NFS, for example). NFS offers the potential for *zero code changes* in the part of your application that analyzes packet capture files, though HTTP is more “firewall friendly” and easily secured via SSL / TLS.

D. Direct Network Interface Card (NIC) Access

Another way to utilize the CloudShield platform without rewriting large parts of your application is to use the platform as a highly efficient, large scale, line-rate packet redirecting device. This is especially useful for applications that must identify “interesting packets” of some kind and then analyze only those packets. The challenge is that, in order to be effective, the application must either sit inline between routers or switches, or hang off a span / “mirror” port of a gigabit LAN switch and look at every packet – even

¹ Developed at Lawrence Berkeley Laboratory, LBL (<http://ee.lbl.gov/>)

Integrating Applications with the ONSP

though only a small fraction of traffic volume may be “interesting.” Deep packet inspection is often required on all potentially interesting packets to complete identification – especially with so many applications running over port 80.

This presents a scaling challenge for any application, in that the application works well for lower data rates, but becomes overwhelmed with the large amount of traffic that it must sift through at Gigabit data rates.

The fastest time to market for higher data rates is to use the CloudShield platform to perform a line-rate scan of all the traffic, and capture just the potentially interesting packets. Captured packets are duplicated to an outbound Gigabit Ethernet interface on the CS-2000 platform, which is connected to the input Gigabit Ethernet interface of your application.

In this scenario the CS-2000 runs a simple RAVE program that identifies potentially interesting packets and passes only those to your existing application, ignoring all other traffic. Sample RAVE code that performs this function can be seen at right. This sample program, developed and compiled using CloudShield’s Open Network Services IDE, is a complete working packet selection application.

Identification can be as simple or as complicated as you like: it’s a trade-off between the time spent modifying the RAVE example to make it more discerning vs. the volume of potentially interesting traffic that your existing application has to process. The sample program here selects packets from specific flows by matching Source IP address and TCP Port plus Destination IP address and TCP Port. This complete program, comments and all, is just 50 lines. The data table defined and imported in the program at right can be easily built using a simple Excel spreadsheet – a topic covered in RAVE training.

Consider an application that looks for file sharing traffic that is masquerading as web traffic by using TCP on port 80. One possible approach is to have the RAVE code capture every packet destined for port 80, and pass these to the existing application. However, for a large enterprise or service provider, this may be a tremendous volume of traffic – too

// Packet Selection Application Using Table-based Packet Selection

// Define Per Packet Processing (Local) Variables Used for Packet/Flow Selection and Actions

```
.local
.var32 SourceIP
.var32 DestIP @SourceIP+4
.var16 SourcePort @DestIP+4
.var16 DestPort @SourcePort+2
// (For Finer Grained Packet Selection Criteria Additional Match Variables Can Be Defined)
```

```
.var32 MatchRow // Where to Store Table Lookup on Match
.var32 ReceivePort // Variable to get Receive Port of Current Packet
.var32 AnalysisAppPort = 4 // Port to Redirect Selected Packets to
.var32 FromAnalysisAppFWDPort = 1 // Port Traffic from the Partner App Should Exit System
```

// Define Counters to be Exported to ASM database for Presentation, Reporting, Analysis, etc.

```
.global
.var32 NotValidPacket EXPORT
.var32 RedirectedPacketCount EXPORT
.var32 NoMatchPacketCounter EXPORT
```

// Define and Imported Table with Packet Selection Criteria Data Used in Lookup Function

```
.database
.db128 Packet_Select 100 importfile='mydata\acltable.tbl'
```

// Program Logic Begins

```
.code
```

// Section Excludes Traffic Received From the Analysis Application Itself Prior to Lookup

```
:START
Packet_Info_Read (ReceivePort, INFO_INPUT_PORT)
Compare_Local_Local_32 (ReceivePort, ==, AnalysisAppPort):REDIRECTCANDIDATE
Packet_Info_Write (INFO_OUTPUT_PORT, FromAnalysisAppFWDPort)
Forward()
Terminate()
```

// Section Checks That Packet is Valid; Copies Valid Packet Data to Flow Identification Variables; Performs Table Lookup; On Match, Forwards Packet to Analysis Application Port; and Increments Redirected Packet Counter.

```
:REDIRECTCANDIDATE
PacketFlagCompare ( FLAG_L4_CHKSUM_VALID, FLAG_L4_CKSUM_VALID,
ALLSET):BADPKTEXTIT
```

```
Copy_Local_Packet_32( SourceIP, ipv4_SourceAddress )
Copy_Local_Packet_32( DestIP, ipv4_DestAddress )
Copy_Local_Packet_16( SourcePort, tcp_SourcePort )
Copy_Local_Packet_16( DestPort, tcp_DestPort )
```

```
Database_Select_ID ( TBLID, Packet_Select, MatchRow, SourceIP ):NOMATCH
Packet_Info_Write (INFO_OUTPUT_PORT, AnalysisAppPort)
Add_Global_Constant_32 ( RedirectedPacketCount, 1 )
Forward()
Terminate()
```

// Section Counts the Number of Non-Matching Packets, and Forwards the Packet

```
:NOMATCH
Add_Global_Constant_32 ( NoMatchPacketCounter, 1 )
Forward()
Terminate()
```

// Section Counts Bad Packets and Forward

```
:BADPKTEXTIT
Add_Global_Constant_32 ( NotValidPacket, 1 )
Forward()
Terminate()
```

Integrating Applications with the ONSP

much to process. If that's the case, the RAVE code can be extended to be more discerning. With a small modification to the RAVE program, packet capture can be made to occur for only TCP port 80 traffic *not* destined for IP addresses within the range used by the web server farm, web application firewall, or other IP addresses used for bona fide web traffic.

Packet selection can be made using the supported Regular Expression matching to locate a string in the payload itself. For example, the file-sharing application Kazaa uses modified HTTP 1.1 requests between clients that look like regular, bona-fide web requests to a stateful firewall or router. A typical Kazaa file sharing request looks like this²:

```
GET /.hash=d0633f1bfdd0fde48cf351ef8c541b67567426dd HTTP/1.1
Host: 123.52.193.31:1214
UserAgent: KazaaClient Oct 18 2002 01:57:14
X-Kazaa-Username: czarny
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 213.77.151.176:2647
X-Kazaa-SupernodeIP: 206.158.106.142:1715
Connection: close
X-Kazaa-XferId: 11312345
X-Kazaa-XferUid: ytCcDgo+3sTohNl2+1Y2jYkCY6NwCA==
```

A RAVE program can easily identify all HTTP flows that contain the “X-Kazaa” fields in the payload, and block them. In addition, with a little more work, the same RAVE program could easily block any return traffic from the Kazaa peer with which the user was attempting to communicate, and also block the Kazaa Supernodes (systems that serve as a directory of other Kazaa clients, so that users can find other people with whom to share files).

CloudShield gets you started with sample RAVE programs that you can modify to perform your specific packet identification function. Additionally, optional reference utilities include a complete RAVE program that identifies, classifies, and optionally blocks per-user instant messaging packet flows. Your developer license allows you to freely modify and incorporate this code in your own RAVE application without restrictions. In this way, you get a flying start on developing your own RAVE code to perform whatever traffic identification function you require.

V. Deployment Architectures

One of the key reasons for using the CloudShield platform is time to market. For existing applications, this means being able to extend or port your application as quickly as possible. In this section, we examine several different alternative approaches: the right approach for you will depend on the nature of your application.

To structure the discussion, consider that most packet processing applications iteratively perform the following sequence of actions:

1. (Optional) Examine the aggregate traffic profile (by aggregating statistical data about all packets) to indicate which traffic might be of interest. This is used in a denial of service detection application, for example, where unusual traffic volumes may indicate that there is a DOS attack.
2. Per-packet examination, classification and statistical data collection on specific packets or packet flows of interest.

² For more information, see <http://kazaasearch.narod.ru/KazaaHTTP.htm>

Integrating Applications with the ONSP

3. Apply some policy or other decision-making process to determine what to do with the packets or flows of interest.
4. Modify, drop or forward packets / packet flows.

All of these communication paths can be accomplished using a single box solution, or a distributed solution.

A. Single Box Solution

The CloudShield 2000 platform incorporates a Server Module which is standard Intel server platform with dual Pentium 3s (future versions will be accommodated – e.g., Xeon or Pentium 4's, etc.) that can be used to host your existing application in a “single box” solution. This server module runs the Red Hat Enterprise distribution of Linux and includes drivers and an API that allow it to communicate with the RAVE-based data plane of the CloudShield platform. The Server Module also contains its own Gigabit Ethernet interfaces, which can be used as a management interface and/or to perform processing on packets captured by the data plane (see direct NIC access above).

The single box solution only supports applications that can run on Red Hat Linux (or can be ported to it); the Application Server Module also performs device management functions that are Linux-based, and consequently onboard partner applications must also run in a Linux environment. This is not to say that only Linux applications are supported however. VMWare can be used to bring applications on board that use other Operating Systems. Application performance requirements should be well understood and tested when considering this scenario.

B. Distributed Multi-box Solution

Often, it is not possible or desirable to have a single box solution for a packet processing application. Your management application simply may not run on Linux or perform fast enough when using VMWare, for example, or packet processing may need to be carried out at geographically distinct locations but controlled by a central management console.

In these cases, a distributed solution is mandatory. Whenever application functionality is distributed, a key consideration is the method and protocols used to achieve the communications between the CloudShield platform and your main application.

In terms of communication flows between the ONSP data plane and the solution's management plane (your applications server) the following communication paths are required:

- Management plane to data plane: commands to initialize data plane
- Data plane to management plane: statistics and other aggregate packet data
- Management plane to data plane: instructions on packets / flows of interest
- Data plane to management plane: captured packets of interest
- Management plane to data plane: instructions on packet treatment (drop, modify etc.)

An application programming interface exists to facilitate precisely these communications paths and will be discussed in this next section (Section VII)

VI. Tools for the Integration / Applications Developer

CloudShield offers a wide set of tools to simplify integrating, porting, or developing new ONSP applications. An IDE offers all the tools for both data plane and management plane application development. RAVE offers a high-level interface for programming data plane operations, while the

© 2004 CloudShield Technologies Inc.

Integrating Applications with the ONSP

ONSP API allows programmatic access to and manipulation of the high-speed memory structures that are accessible to the data plane (AKA the Silicon Database – see below). There are also many methods used to export data from the data plane to the management plane, or off-board to a remote server process.

A. An IDE for Open Network Services

The CloudShield developer's kit ships with an integrated development environment (IDE) and software debugger for the CloudShield CS-2000. The Open Network Services (ONS) IDE is suitable for both individual developer use and team development. Based on the *Eclipse.org* open source project, the IDE framework supports any number of open source 3rd party 'plug-ins' including programming language compilers and debuggers for Java, C/C++, and PHP. Early releases of the ONS IDE includes only tools focused on assisting in CS-2000 data plane programming, but future versions can be expected to provide tools to help programmers build complete top-to-bottom applications. The goal is to enable developers to code, compile, execute, and debug their entire applications from their PCs.

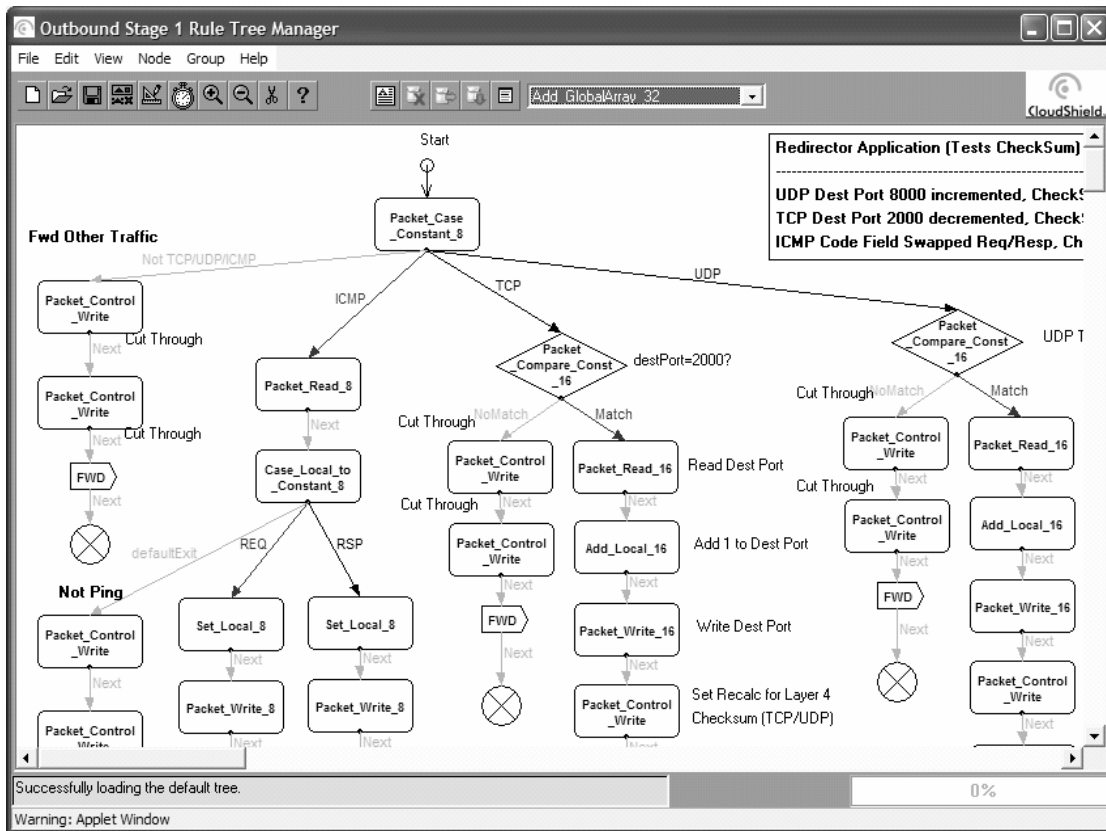
The IDE allows RAVE programs to be written as text and also via a "drag and drop" graphical editor, where nodes represent core RAVE functions and lines between nodes dictate control flow. Visual RAVE provides a simple way to quickly generate RAVE programs and visualize control flow. Textual RAVE programs allow standard development tools and practices to be used with RAVE code, such as version control using deltas, "diff"-like tools, standard text editing tools, etc.

The RAVE debugger means developers can work on RAVE code and do unit testing on their own development workstation without having to share test hardware. It provides comprehensive debugging of RAVE programs, including single-stepping of RAVE code and memory read/write during debugging. Packet input to the system is via libpcap format files for both input and output of the system. This dramatically simplifies the construction of standard, repeatable test cases and regression test suites. It also means that traffic generators and packet capture hardware is not required for development testing.

B. RAVE – Packet Processing Programming Language

CloudShield's Open Network Services Platforms use a wide range of state-of-the-art technologies to achieve very high-performance complete packet processing at very high data rates. A simple packet oriented programming language to access the packet processing capacity and functionality of the ONSP is provided called RAVE. Initially targeting the network engineer, RAVE enables programmers to implement packet processing logic as though mapping out on a white board. Visual RAVE offers a simple way to quickly generate RAVE programs and visualize control flow. Packet functions are selected from a menu and logic flow is determined by links between functions. The screen capture below is a Visual RAVE sample program.

Integrating Applications with the ONSP



Visual RAVE is designed for use by only one programmer for one application. It offers a simple interface for rapid modeling and prototyping of packet processing applications. For more sophisticated or team-based programming, a text form of the RAVE language. Using a programmatic syntax, programmers have access to the same high-level packet functions as Visual RAVE but in a more traditional programming environment. The ONS IDE mentioned above supports CVS-based version control systems so that team members can work independently while code tree integrity is maintained. RAVE program components created using the text editor can be joined with other components and compiled into a single executable application on the CS-2000

Here is a very simple packet counting RAVE program.

```
//Define variables
.global
.var32 TotalpacketCount EXPORT INTERVAL=5 TABLE=Counters ACTION=none
// Beginning of application
.code add_global_constant_32 (TotalPacketCount, 1) // Add Constant to Global
forward()
terminate()
.end
//End the program
```

In this program, a global variable called *TotalpacketCount* is defined indicating where it will be stored (Counters) and how often the data plane memory value is exported to the management plane for access by storage. Note how simple it is to make the output data of packet processing functions available to your

Integrating Applications with the ONSP

applications. Simply by declaring a variable with the EXPORT designation automatically moves the variable data, at a defined interval, to a database (MySQL) that resides on the ASM hard drive. Once there the data is available via well known access methods (e.g. ODBC or SNMP) for presentation or further analysis.

The table below is a sample of available packet processing functions in RAVE. For a more in-depth look at RAVE a programming guide is available as well as hands-on training.

Packet_Info_Read	Packet_Get_Offset_32	Database_Insert
Packet_Info_Write	Packet_Collapse	Database_Update
Packet_Flag_Compare	Drop	Database_Delete
Packet_Flag_Write	Forward	Database_Read
Packet_Search	Terminate	Database_Select
Log	Gosub	Database_Select_ID
Packet_Replicate	Get_XTime	Copy_Local_Constant_32
Packet_Expand	Get_Context	Variable_Transfer

C. Extensible Reference Utilities

CloudShield also offers four complete top-to-bottom programs called ONS Utilities to perform key packet processing functions and will serve to further shorten your development cycles. These programs include functional data plane program code for both Visual and Standard RAVE along with their required connections to a management plane database. The utilities also come with web-based user interfaces that plug into the ONSP Web Management Interface (WMI). These utilities serve as a model to help developers understand how to apply the ONS applications model to their needs, and all components are licensed for reuse by developers royalty free. As new utilities become available they will be added to the offering. The ONS Utilities available today are: ACL_Control, Flow_ID, Host_ID, and IM_Control. Read on for details on each.

- **ACL_Control: Putting Access Control Lists to Work Outside of the Router**

The ACL_Control utility supports well understood Cisco formatted ACLs and scales up to hundreds of thousands of ACLs. ACLs can be added or removed dynamically and programmatically. ACLs can be applied in between packets, thus providing higher security resiliency when adding new filters to block unwanted traffic.

- **Flow_ID: NetFlow Data Export on Multi-Gigabit Links**

Flow_ID, a NetFlow Version 5 data explore utility reports on 100% of the traffic—no data sampling. Hundreds of thousands of flows are supported with compliance with aging timers that can retire long application-type flows (FTP, etc.)

- **IM_Control: Instant Messenger Application Policy Control**

Instant messenger applications provide a means of communication across private / public boundaries that are often undetected and outside of network use policies. For operators dealing with communications audit and compliance regulations, this can pose a significant challenge. IM_Control taps into the ONSP's line-rate traffic classification, state tracking, and deep packet inspection capabilities to allow or deny IM

Integrating Applications with the ONSP

(AIM, MSN Messenger, and Yahoo Messenger) user logins, log authorized IM user message activity, and message capture and store message traffic.

- **Host_ID: Network Host Inventory and reporting**

Host_ID monitors user-specified subnets and sets a baseline network inventory. Among other features, it examines the streams for “information and inventory fingerprints” which indicate operating system and level, as well as application services and level, are active on that a given IP address. All common desktop environments, whether Microsoft, Unix, or MAC-based, are profiled.

D. CloudShield Data Plane API

The CloudShield data plane API provides for the control of RAVE programs running on the data plane, and the ability to read and write to variables, memory and associative memory in the data plane Silicon Database. Having said that, there are several other options for the bulk export of data and packets from the data plane that may be easier for your application to use including ODBC/SQL, SNMP, HTTP, NFS and others (detailed below). Therefore, the API is most useful for the loading, starting, and stopping of RAVE programs and for writing to data plane memory in order to control the execution of RAVE programs.

The CloudShield API is session-oriented, and has transactional semantics. This means that all changes that are made during a session are queued, and then executed as a single transaction when the session is “committed.” This provides two key benefits:

- Race conditions are avoided when making updates to data plane memory (the data plane is highly parallelized in order to provide line-rate performance and low latency).
- Memory writes can be optimized to avoid adversely impacting the performance of the data plane.

The API has a fixed syntax which means that future versions of the CloudShield platform (and API) will still use the same syntax, thereby ensuring that you will not have to keep “re-reporting” your application to new CloudShield hardware every time. Again, the concept is similar to SQL in the database world – a fixed syntax language for manipulating database structure, and reading and writing data.

The API runs over TELNET or SSH transport, and therefore the method of connection is the same regardless of whether you are building a single box or distributed solution. If your application can run in either mode, there are no API code changes required for the single box or distributed versions.

One approach that has already been used by a CloudShield partner is to build a small HTTP to CloudShield API translation “shim” that runs on the server module. Commands to the data plane can be sent using standard HTTP methods, and HTTP can also be used to retrieve the packet capture files that are analyzed by the control plane. This simplifies the design and coding of the management plane application.

Integrating Applications with the ONSP

E. Data Export Options

Given that your application must be modified in order to use the CloudShield API, several widely used data export options are available that are likely to be easier to integrate with an existing application.

The data export option that makes sense for your application usually depends on its architecture. The following table indicates the most appropriate data export option for each of the application architectures identified in section V above:

Architecture	Applicable data export options
Rules-based IDS-like	File-based, ODBC/SQL, XML, Ethernet
Cisco NetFlow	NetFlow data export, SNMP
Libpcap	File-based
Direct NIC access	Ethernet

Taking these in decreasing order of level of abstraction:

1. ODBC / SQL

The CloudShield platform allows RAVE programs running in the data plane to export values into table rows in a MySQL database on the Server Module's hard drive. This database can then be accessed through standard ODBC / SQL queries. The database schema used to contain the RAVE variables is user-definable.

This kind of approach is typically used in applications that collect and aggregate information in the data plane. Cisco NetFlow record generation is an example of an aggregation function – rather than capturing packets or sending information about each packet, individual flows are identified and aggregate information about all the packets in the flow is maintained. However, instead of sending NetFlow data records to a NetFlow collector, these statistics could be exported to a MySQL database and then queried using ODBC/SQL.

This approach is particularly useful for reporting or data mining functions, as standard database reporting tools such as Crystal Reports can be used.

2. SNMP

The CloudShield 2000 platform automatically makes the MySQL database tables available via SNMP MIBs. No additional programming is required – it is provided automatically by the CloudShield system. Original SNMP plus SNMPv2 and SNMPv3 MIBs are provided. MIB values are read only and accessible using SNMP "Get" or get bulk commands.

This option is particularly useful for integration with existing network management or fault management applications that know how to deal with SNMP MIBs and traps, such as HP OpenView and Micromuse Netcool.

Again, applications that make use of aggregated packet data may find this approach particularly suitable.

Integrating Applications with the ONSP

3. XML or SOAP (via HTTP)

Because the CloudShield platform contains an Apache server, it can also process XML messages, including those that use the SOAP methodology. There are Apache server extensions and toolkits available that mean you will not have to “roll your own” XML or SOAP parser or state machine.

In addition, it is code can be quickly written that parses HTTP POST requests to control the operation of the data plane RAVE program using the CloudShield API. This offers the possibility of using HTTP / XML as a single channel for both data export and control of the data plane.

For rules-based applications, this approach allows aggregate information to be retained on the data plane and then queried via XML or SOAP.

4. NetFlow Data Export

If your application processes NetFlow data today in order to identify interesting traffic flows, it can quickly and easily be integrated with the CloudShield platform. CloudShield offers Flow_ID, a NetFlow Data Export reference application that generates NetFlow records for all traffic. Note that this means all packets – the packet sampling approach (“NetFlow sampling”) that is typically used on routers for performance reasons is not required due to the line-rate performance of the CloudShield data plane.

5. File-based Communication (HTTP or NFS)

RAVE programs can also capture packet traces to files that can then be read by the server application using either HTTP or NFS. Files can be accessed using HTTP GET requests to the URL that specifies the file, and then processed / analyzed by the application on the server. Alternatively, NFS can be used to transport the data to your server application, providing the ability to read the files as if they were local.

This approach is typically used by applications interested in packet capture. For example, an application that uses libpcap to capture packets of interest will generate capture files, which can then be exported using HTTP or NFS.

6. Raw Ethernet Packet Export

With the Ethernet approach, the CloudShield client sends captured packets directly to the server application via a Gigabit Ethernet connection (see “the packet picker” above). The key advantages are:

- 1) Very few changes are required to the server application (sometimes none).
- 2) No coding of a communications protocol between the client and server is required
- 3) No RAVE code is required to extract, process, and export data values from the data plane, and process input from the server.

The key disadvantages are:

- 1) The client and server must be physically close enough that they can be linked together via Ethernet.
- 2) A limited number of *clients* can be connected to the *server* (due to physical port restrictions or throughput restrictions)

VII. Conclusion and summary

This paper began by segmenting packet processing applications into buckets:

- Rules-driven IDS-like applications
- Applications that use NetFlow or other aggregated traffic information

Integrating Applications with the ONSP

- Applications that use libpcap to examine or capture packets of interest
- Applications that directly access the network driver and don't use any higher level libraries or aggregation

For each of these, it was shown how the CloudShield platform could support that model of application, and the example code and tools provided in the CloudShield developers kit were described. Then, for each type of application, the required communication between management and data plane layers was examined, and the wide range of CloudShield-supported mechanisms mapped to each communication need.

The result is a roadmap for rapidly porting any kind of existing packet processing application to the CloudShield platform. The complete package of the CloudShield platform plus the developer's kit offers a unique solution to the problem of scaling packet processing applications to multi-gigabit line rates while achieving a time-to-market measured in weeks.

Appendix A: An Application Port Case Study

Arbor Networks®: Seizing a Market Opportunity

Arbor Networks of Lexington, MA is the leading supplier of network integrity tools. Arbor's Peakflow® is the most widely deployed Distributed Denial of Service (DDoS) detection product in the world. Service providers, based on demands from their subscribers, wanted to offer clean bandwidth services which required more than DDoS identification, it required *mitigation*. Arbor quickly realized that this could be a strong differentiator for their solution, but their current platform was unable to manage the workload and line rates. Developing a brand new hardware platform was expensive and technically risky, and Arbor didn't want its customers to start looking for alternative solutions in the meantime.

Peakflow processes NetFlow records generated by routers at various junctions in the service provider networks to identify potential denial of service attacks. While Arbor customers loved the ability to simplify the identification and sources of DDoS attacks, they also needed something that would be able to mitigate the attack and prevent the packets from thousands of "zombies" (Internet hosts compromised by a worm or other DDoS program) consuming bandwidth on their backbones and saturating peering points and customer links. They were looking for DDoS *mitigation* as well as identification.

The key challenge for DDoS mitigation is that there are large volumes of traffic entering a service provider's network at each interconnect point, only some of which is DOS traffic. By the very nature of this type of attack, the zombies are distributed across the Internet; simply blocking whole blocks of IP addresses would stop the zombie traffic but also cut off important legitimate IP traffic from hosts on the same network. A very fine-grained form of traffic filtering is required, blocking only the traffic from zombies identified by Arbor's DDoS product – but this has to be performed at multi-Gigabit, even OC-48, line rates (2.488Gigabits per second).

OC48 line rate is well beyond the capabilities of standard PC or Compact PCI hardware: at OC48, a 40-byte packet can arrive in 129ns. Given the business and technical risks associated with building a solution, Arbor looked to the marketplace for a better alternative and found the CloudShield ONSP offered the capacity and processing performance needed to service their service provider customers.

For their implementation, Arbor modified an existing RAVE reference utility, ACL_Control, a rules based traffic control engine. Running on CloudShield's ONSP, the Arbor program can support hundreds of thousands of ACL entries, and, because ACL_Control provides access to the ONSP's complete packet inspection and analysis, fined grained filtering rules can be implemented. Arbor wrote code that allowed their DDoS identification application to generate the ACL list and then post it to the CloudShield platform using HTTP. On the ONSP's Pentium/Linux-based Application Server Module (ASM), Arbor built a simple CGI script (invoked by the Apache web server) to process the list and load into the CS-2000's *Silicon Database* (memory resident database located in the data plane) for packet filtering. The Arbor solution periodically captures attack traffic for further analysis and finer attack filter creation.

In a matter of days, Arbor networks had a prototype DDoS mitigation application working in their labs. In just six weeks, Arbor had completed integration development and QA, and shipped to customer for field trials. Based on positive customer feedback additional functions have been added during the trial periods, including a periodic packet capture to help further define the attack and refine the filtering rules. Initial distributed multi-box implementations provided the shortest possible time-to-trial allowing for market testing and meaningful customer interaction before a larger application porting effort is taken. Taking this approach Arbor ensures the additional investment is secure and that the solution only gets better.