



**6WINDGate™**

-

**Architecture Overview**

-

**NE 6W-08-188 v1.0**

## TABLE OF CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	PURPOSE OF THE DOCUMENT	1
1.2	REFERENCE DOCUMENTS	1
1.3	ACRONYMS	1
<b>2</b>	<b>GENERAL CONCEPTS</b>	<b>4</b>
2.1	NETWORK EQUIPMENT ARCHITECTURE	4
2.1.1	General concepts	4
2.1.2	Data Plane	4
2.1.3	Control Plane	5
2.1.4	Management Plane	5
2.2	ARCHITECTURE OPTIONS	5
2.3	MULTI-CORE PROCESSOR TECHNOLOGY	6
2.4	MULTI-CORE PROCESSOR TECHNOLOGY FOR NETWORKING	7
2.5	ULTIMATE SOFTWARE ARCHITECTURE FOR MULTI-CORE	8
<b>3</b>	<b>6WINDGATE™ ARCHITECTURE OVERVIEW</b>	<b>9</b>
3.1	6WINDGATE™ ARCHITECTURE – MAIN CONCEPTS	9
3.2	6WINDGATE™ PROFILES	10
3.3	FAST PATH LESS SOLUTION - 6WINDGATE™ ADS	11
3.4	FAST PATH-BASED SOLUTIONS	12
3.4.1	6WINDGate™ EDS	13
3.4.2	6WINDGate™ SDS	14
3.5	COMPARISON BETWEEN 6WINDGATE™ PROFILES	15
3.6	6WINDGATE™ EXTENSIONS FOR DISTRIBUTED ARCHITECTURES	17
3.6.1	Application: High Availability	19
3.6.2	Application: Fast Path Extensions	19
<b>4</b>	<b>6WINDGATE™ BUILDING BLOCKS AND MODULES</b>	<b>20</b>
4.1	6WINDGATE™ FEATURE SUMMARY	20



---

<b>4.2</b>	<b>6WINDGATE™ MODULE DATA SHEET</b>	<b>21</b>
<b>4.3</b>	<b>6WINDGATE™ LINUX NETWORKING STACKS</b>	<b>21</b>
4.3.1	VNB framework	22
4.3.2	Others Enhancements in 6WINDGate™ Linux Networking Stack	23
<b>4.4</b>	<b>6WINDGATE™ FAST PATH</b>	<b>24</b>
4.4.1	6WINDGate™ Fast Path Modules	24
4.4.2	Fast Path Module Integration	25
4.4.3	IP Forwarding Example	26
<b>4.5</b>	<b>6WINDGATE™ CONTROL PLANE</b>	<b>27</b>
<b>4.6</b>	<b>6WINDGATE™ XML-BASED MANAGEMENT SYSTEM</b>	<b>27</b>
4.6.1	Introduction	27
4.6.2	XMS Configuration Manager	28
4.6.3	Core	29
4.6.4	Service Modules	29
4.6.5	XMS Clients	29
<b>5</b>	<b>6WINDGATE™ API</b>	<b>30</b>
<b>5.1</b>	<b>API LIST AND OVERVIEW</b>	<b>30</b>
5.1.1	API for 6WINDGate™ ADS	30
5.1.2	API for 6WINDGate™ EDS and SDS	31
<b>5.2</b>	<b>CACHE MANAGER</b>	<b>32</b>
<b>5.3</b>	<b>FAST PATH MANAGER</b>	<b>32</b>
<b>6</b>	<b>PACKET PROCESSING</b>	<b>33</b>
<b>6.1</b>	<b>FAST PATH LESS ARCHITECTURE (6WINDGATE™ ADS)</b>	<b>33</b>
<b>6.2</b>	<b>FAST PATH-BASED ARCHITECTURES</b>	<b>33</b>
6.2.1	6WINDGate™ EDS	33
6.2.2	6WINDGate™ SDS	35
<b>7</b>	<b>6WINDGATE™ EXTENSIBILITY</b>	<b>37</b>
<b>7.1</b>	<b>6WINDGATE™ OPEN ARCHITECTURE</b>	<b>37</b>
<b>7.2</b>	<b>INTEGRATION OF A VNB MODULE</b>	<b>37</b>
<b>7.3</b>	<b>CUSTOMER CONTROL PLANE MODULE WITH 6WINDGATE™ FAST PATH MODULES</b>	<b>38</b>
<b>7.4</b>	<b>CUSTOMIZING 6WINDGATE™ MANAGEMENT SYSTEM</b>	<b>39</b>
<b>8</b>	<b>6WINDGATE™ VIRTUAL FAST PATH SOLUTION</b>	<b>40</b>
<b>9</b>	<b>EVALUATION AND FURTHER READING</b>	<b>41</b>

## TABLE OF FIGURES

Figure 1: Network Equipment Architecture .....	4
Figure 2: Multi-Core Technology .....	7
Figure 3: 6WINDGate™ Profiles .....	10
Figure 4: 6WINDGate™ Profiles .....	11
Figure 5: 6WINDGate™ ADS architecture.....	12
Figure 6: 6WINDGate™ Fast Path exception strategy.....	13
Figure 7: 6WINDGate™ EDS architecture.....	14
Figure 8: 6WINDGate™ SDS architecture.....	15
Figure 9: Performance comparison between 6WINDGate™ Profiles (1 to 4 cores) .....	16
Figure 10: Performance comparison between 6WINDGate™ Profiles (1 to 16 cores) .....	16
Figure 11: Comparison between 6WINDGate™ Profiles .....	17
Figure 12: MC-based configurations .....	18
Figure 13: 6WINDGate™ Distributed Architecture .....	18
Figure 14: VNB Architecture .....	22
Figure 15: Fast Path architecture .....	25
Figure 16: IPv4 Forwarding Packet Processing .....	26
Figure 17: 6WINDGate™ eXtensible Management System (XMS) .....	28
Figure 18: 6WINDGate™ ADS Implementation Example .....	33
Figure 19: 6WINDGate™ EDS Implementation Example .....	34
Figure 20: 6WINDGate™ SDS Implementation Example .....	35
Figure 21: Integration of CoS to VLAN association function with VNB framework.....	38



### MODIFICATIONS TRACKING

Date	Author	Version Number	Modifications
May 5 <sup>th</sup> , 2008	Eric CARMES	V1.0	Initial version



# 1 INTRODUCTION

## 1.1 PURPOSE OF THE DOCUMENT

This document aims at giving an overview of the 6WINDGate™ software architecture.

- Section 2 gives a general reminder about network equipment architecture.
- Section 3 describes 6WINDGate™ software architecture including design concepts, profile definition and profile comparison.
- Section 4 details the different building blocks and modules of 6WINDGate™ architecture including Fast Path, Slow Path, Control Plane and Management modules.
- Section 5 details 6WINDGate™ API.
- Section 6 details how a packet is processed in 6WINDGate™ architecture.
- Section 7 explains how 6WINDGate™ can be used to extend feature set using some examples.
- Section 8 explains how 6WINDGate™ Virtual Fast Path solution can help to develop Fast Path modules.
- Section 9 introduces 6WINDGate™ evaluation process and lists documents that should be read in a further step.

## 1.2 REFERENCE DOCUMENTS

- [1] 6WINDGate™ Powerpoint presentations
- [2] 6WINDGate™ Modules – Data Sheets
- [3] 6WINDGate™ Software Architecture Documents and Software Design Documents
- [4] Application Note: Open VPN Integration

## 1.3 ACRONYMS

ADS	6WINDGate™ Advanced Dev Suite
API	Application Programming Interface
ARP	Address Resolution Protocol
ASIC	Application-Specific Integrated Circuit
BGP	Border Gateway Protocol
CoS	Class of Service
CLI	Command Line Interface
CP	Control Plane

---

CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
DP	Data Plane
EAP	Extensible Authentication Protocol
ECMP	Equal Cost Multi-Path
EDS	6WINDGate™ Enhanced Dev Suite for Multi-Core
FMIP	Fast Mobile IP
FP	Fast Path
FPC	Fast Path Control
FPM	Fast Path Manager
FPN	Fast Path Networking
FPS	Fast Path Statistics
FPVI	Fast Path Virtual Interface
GRE	Generic Routing Encapsulation
HA	High Availability
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
IKE	Internet Key Exchange
IPsec	IP Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IS-IS	Intermediate System to Intermediate System
L2	Layer 2
L3	Layer 3
MC	Multi Core
MCEE	Multi-Core Executive Environment
MIP	Mobile IP
MLD	Multicast Listener Discovery Protocol
NAT	Network Address Translation
NAT-PT	Network Address Translation – Protocol Translation
NDP	Neighbor Discovery Protocol
NEMO	NEtwork MObility
OCF	Open Cryptographic Framework
OLSR	Optimized Link State Routing Protocol
OS	Operating System
OSPF	Open Shortest Path First
PCI	Peripheral Component Interconnect

---

PIM-SM	Protocol-Independent Multicast – Sparse Mode
PPP	Point-to-Point Protocol
PPPoE	Point-to-Point Protocol over Ethernet
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RFC	Request For Comment
RFPVI	Remote Fast Path Virtual Interface
RIP	Routing Information Protocol
RIPng	Routing Internet Protocol for IPv6
ROHC	Robust Header Compression
SAD	Software Architecture Document
SDD	Software Design Document
SDK	Software Development Kit
SDS	6WINDGate™ Specialised Dev Suite for Multi-Core
SMP	Symmetric Multi Processing
SP	Slow Path
SPI-4	System Packet Interface Level 4
TCP	Transmission Control Protocol
TTL	Time To Live
VLAN	Virtual Local Area Network
VNB	Virtual Networking Blocks
VPN	Virtual Private Network
VR	Virtual Routing
VRRP	Virtual Router Redundancy Protocol
WiFi	Wireless Fidelity
XIPC	eXtended Inter Process Communication
XML	eXtensible Markup Language
XMS	XML-based Management System

## 2 GENERAL CONCEPTS

### 2.1 NETWORK EQUIPMENT ARCHITECTURE

#### 2.1.1 General concepts

Architecture of network equipment has been heavily studied over the years. Main concepts of an efficient architecture were defined some years ago to be able to design high-speed routers to face the explosion of Internet traffic. Now, this architecture has been extended to new services such as L2 protocols, security, mobility, multicast...

IP-based equipment can be partitioned into three basic elements: Data Plane, Control Plane and Management Plane (cf. Figure 1).

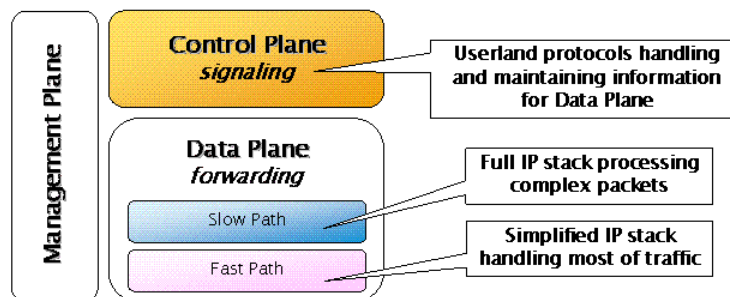


Figure 1: Network Equipment Architecture

#### 2.1.2 Data Plane

The Data Plane is a subsystem of a network node that receives and sends packets from an interface, processes them in some way required by the applicable protocol, and delivers, drops, or forwards them as appropriate.

For routing functions, it consists of a set of procedures (algorithms) that a router uses to make a forwarding decision on a packet. The algorithms define the information from a received packet to find a particular entry in its forwarding table, as well as the exact procedures that the routing function uses for finding the entry. It offloads packet forwarding from higher-level processors. For most or all of the

packets it receives and that are not addressed for delivery to the node itself, it performs all required processing.

Similarly, for IPsec functions, a security gateway checks if a Security Association is valid for an incoming flow and if so, Data Plane locally finds information to apply Security Association to a packet.

### 2.1.3 Control Plane

The Control Plane maintains information that can be used to change data used by Data Plane. Maintaining this information requires handling complex signalling protocols. Implementing these protocols in Data Plane would lead to poor forwarding performance. A common way to manage these protocols is to let Data Plane detect incoming signalling packets and locally forward them to Control Plane. Control plane signalling protocols can update Data Plane information and inject outgoing signalling packets in Data Plane. This architecture works because signalling traffic is a very small part of the global traffic.

For routing functions, Control Plane consists of one or more routing protocols that provide exchange of routing information between routers, as well as the procedures (algorithms) that a router uses to convert this information into the forwarding table. RIP, OSPF, BGP, for unicast and PIM for multicast are examples of such routing protocols. In case of virtual routing, multi-instances of forwarding tables are managed at the Data Plane level. As soon as Data Plane detects a routing packet, it forwards it to Control Plane to let routing protocol compute new routes, add or delete routes. Forwarding tables are updated with this new information. When a routing protocol has to send a packet (OSPF Hello packet for instance), it is injected in Data Plane to be sent in the outgoing flow.

For IPsec security functions, signalling protocols for key exchange management such as IKE or IKEv2 are located in Control Plane. Incoming IKE packets are locally forwarded to Control Plane. When key are renewed, Security Associations located in Data Plane are updated by Control Plane. Outgoing IKE packets are injected in Data Plane to be sent in the outgoing flow.

### 2.1.4 Management Plane

The Management Plane provides an administrative interface into the overall system. It contains processes that support operational administration, management or configuration/provisioning actions such as:

- Facilities for supporting statistics collection and aggregation,
- Support for the implementation of management protocols,
- Command Line Interface, Graphical User Configuration Interfaces through Web pages or traditional SNMP Management. More sophisticated solutions based on XML can also be proposed.

## 2.2 ARCHITECTURE OPTIONS

According to the targeted level of performance and cost of the network equipment, several architecture options can be considered.

For low-end equipment, having a single processor to manage both Control Plane and Data Plane is a viable option. It significantly reduces the bill of material of the equipment. In that case, Data Plane is generally managed at the OS kernel level while Control Plane is implemented in user land. The forwarding element of Data Plane is directly connected to network interfaces through drivers.

For equipment that has to process higher bandwidth and/or deeply inspect the incoming packets, a more sophisticated architecture is required. Data Plane and Control Plane are managed by different processors. Functions implemented at the Data Plane level are performance-oriented; algorithms are simple but have to be efficient. For that reason, ASIC, micro-coded processors are well suited for Data Plane implementation. Control Plane processors can be more generic as they are used to process complex finite state machines to implement signalling protocols.

In such architectures, Data Plane can be itself subdivided into two parts:

- **Fast Path:** The Fast Path handles most of the packets to determine to which port a packet should be forwarded. However, some complex packets are not processed at this level and are redirected to the Slow Path through exceptions. For instance, IP forwarding is managed by the Fast Path but not ICMP or ARP.
- **Slow Path:** The Slow Path handles the few packets which are too complex to be processed by the Fast Path. Generally, Slow Path implements a complete IP stack and handles most of the exceptions of the Fast Path. Slow Path and Control Plane can be co-localized on the same processor.

High end equipment may require more:

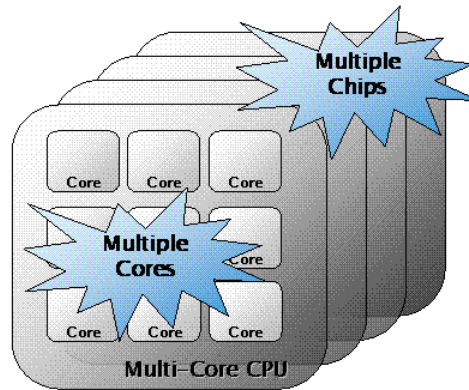
- A distributed Fast Path to handle a large number of interfaces,
- A redundant architecture to provide high availability services,
- Separated processors to handle Control Plane and Slow Path.

The architecture described in previous sections can be applied to all network protocols having in mind a simple rule. All mechanisms that require a per-packet processing should be implemented in Data Plane, while signalling protocols that require more complex processing on a limited part of the incoming flow should be implemented in Slow Path & Control Plane.

### 2.3 MULTI-CORE PROCESSOR TECHNOLOGY

Until now, running a processor at a higher frequency was the only solution to increase processing capabilities. Unfortunately, this approach has recently reached micro-electronics limits and the preferred solution to reach higher processing capabilities while keeping power consumption under acceptable limits is to have several processors running in parallel.

Multi-Core (MC) processor technology (cf. Figure 2) was introduced a few years ago and is bringing a smart solution to solve this problem; it is based on an instance of generic processing unit (core) that is duplicated to build a complete chip (Multi-Core).



*Figure 2: Multi-Core Technology*

Currently, the number of cores can reach 16 but it is likely that this number will grow up to several hundreds of cores in the future thanks to micro-electronics improvements. These cores are interconnected through ad hoc communication hardware such as message rings or switching fabrics to be able to handle parallel processing and exchange information between cores.

According to different implementation strategies, MC can run a single or several threads on a core.

Moreover, MC implements built-in specific hardware to speed up information processing such as network interfaces, crypto-processors, pattern matching processors, or dedicated queues for efficient QoS management.

MC architecture is by essence scalable as larger configurations can be built by interconnecting several MC CPUs.

The way cores can be used is very flexible. Several execution environments possibly relying on different Operating Systems can coexist on different cores of the same MC using virtualization concept. For optimal performance, some MC providers also provide a low-level executive environment (MCEE: Multi-Core Execution Environment) to optimise performance.

As the core is based on a standard processor (MIPS, PowerPC, ARM...), programming models are far simpler compared to the previous generation of micro-coded architectures.

## 2.4 MULTI-CORE PROCESSOR TECHNOLOGY FOR NETWORKING

MC technology can address many applications. Networking and telecoms markets are early adopters as they can benefit from significant improvements brought by MC technology to design efficient software architecture for network equipment:

- MC architecture allows a flexible distribution of cores between Data Plane and Control Plane and the coexistence of different execution environments (one for Fast Path, one for Slow Path and Control Plane for instance) on a single chip. A typical use of MC technology for network equipment, for instance on a 16-core processor, is to use several cores to implement an efficient Fast Path under MCEE, while the remaining number of cores are dedicated to Linux environment implementing Slow Path (IP stack) and Control Plane. The different functions are co-localized, meaning on the same MC chip, but distributed over the different cores.

- To implement efficient high performance packet processing such as those performed by a Fast Path, dedicated MCEE is provided that minimizes latency, lock contention... Although services provided by such dedicated environment are limited, the programming model is simpler compared to previous generation of Network Processors based on micro-coded architectures. It is therefore easier to provide complete features at the Data Plane level.
- Built-in hardware features (crypto engines, packet matching engines, and hardware queue for QoS management) can be used for an efficient implementation of time-consuming functions.
- Standard Operating Systems have also been ported on MC technology. Slow Path and Control Plane that implement more complex mechanisms can run under a standard Operating System. It however requires an efficient multi-processor implementation of the networking stacks to be able to use efficiently several cores at the same time.
- MC scalable architecture can also be used to interconnect different MC to have for instance a distributed Fast Path over several MC or High Availability features.

This being said, one of the key issues to be solved is the integration of Control Plane, Slow Path & Fast Path to benefit from the level of performance of the MC technology while providing a complete set of networking features.

## 2.5 ULTIMATE SOFTWARE ARCHITECTURE FOR MULTI-CORE

MC is a great technology and the only viable solution to significantly increase processing capabilities.

However, it completely changes the software development paradigm. Dedicated software architectures have to be proposed to use efficiently the MC platforms.

Requirements for ultimate networking software architecture for MC can be summarised as follow:

- Meeting performance requirements based on a specific software architecture
- Fully integrated with standard OS making MC complex architectures transparent for applications
- Providing hardware independent framework portable over various MC hardware
- Enabling differentiators for value-added customer extensions thanks to an open architecture

### 3 6WINDGATE™ ARCHITECTURE OVERVIEW

#### 3.1 6WINDGATE™ ARCHITECTURE – MAIN CONCEPTS

To answer the requirements listed in section 2.5, 6WINDGate™ architecture has been designed with five main concepts in mind listed in the following table.

Concept	6WINDGate™ Architecture Options
<b>6WINDGate™ has been specifically designed for Multi-Core</b>	<ul style="list-style-type: none"> <li>• Efficient Fast Path architecture to make the best use of MC performance according to the numbers of cores</li> <li>• Flexible distribution of Control Plane / Slow Path / Fast Path over the cores</li> <li>• Complete synchronization between Control Plane, Slow Path &amp; Fast Path</li> </ul>
<b>6WINDGate™ takes full benefit from MC built-in HW while providing a highly portable solution</b>	<ul style="list-style-type: none"> <li>• High-level APIs to interface HW features such as crypto-engines for security or queue management for QoS</li> <li>• Fully portable architecture (independence from MC) for generic features while taking benefit from HW specificities</li> </ul>
<b>6WINDGate™ integrates Multi-Core with standard OS</b>	<ul style="list-style-type: none"> <li>• Full integration of MC specific SW (Fast Path) with Linux</li> <li>• MC transparent solution for applications relying standard Linux APIs to maximize reuse of existing SW</li> </ul>
<b>6WINDGate™ provides a complete networking solution</b>	<ul style="list-style-type: none"> <li>• Comprehensive set of L2/L3 networking features, each one specifically designed for MC</li> <li>• Complete management solution based on XML</li> </ul>
<b>6WINDGate™ is open for extension at all levels</b>	<ul style="list-style-type: none"> <li>• Generic packet processing APIs for Fast Path extension</li> <li>• Preserved standard Linux APIs application integration and extension</li> <li>• Ready-to-use XML-based management for implementation of new services</li> </ul>

### 3.2 6WINDGATE™ PROFILES

6WINDGate™ architecture relies on a modular approach to provide adapted solutions to customer requirements. This modularity is based on three building blocks:

- Fast Path Modules
- Linux Networking Stack
- Control Plane Modules

6WINDGate™ building blocks can be assembled in 3 different profiles. These profiles are summarized in Figure 3.

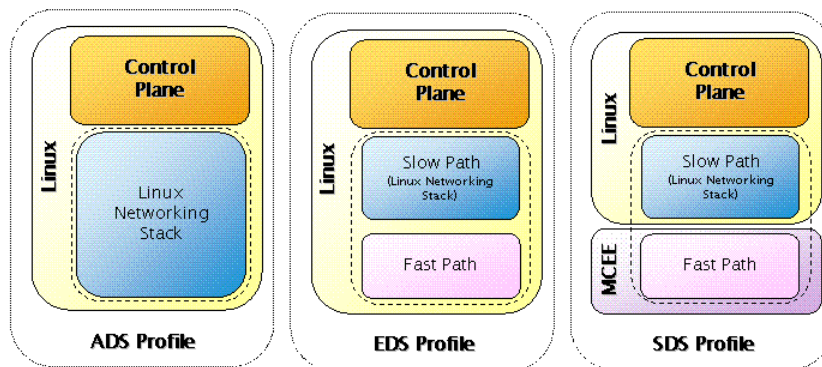


Figure 3: 6WINDGate™ Profiles

- **6WINDGate™ ADS** is targeted for middle range appliances and equipment. Control Plane and Data Plane are co-localized. For MC architectures, a SMP Linux kernel with an optimized SMP kernel networking stack is running on all the cores in order to process many packets simultaneously.
- **6WINDGate™ EDS** is a solution based on a Fast Path architecture. This Fast Path is implemented as a Linux kernel module between the Linux networking stack and the interface drivers. So, it does not require any specific MCEE (Multi-Core Executive Environment). Compared to a standard Linux architecture, forwarding is performed at the driver level. Only packets that cannot be processed by Fast Path are forwarded to the Linux Networking Stack (Slow Path). 6WINDGate™ EDS architecture relies on Cache Manager and Fast Path Manager modules to integrate and synchronise Fast Path processing and Slow Path / Control Plane in a transparent manner. 6WINDGate EDS™ delivers the best possible performance in pure Linux environment.
- **6WINDGate™ SDS** is targeted for high end equipment. Similarly to 6WINDGate™ EDS, it is also based on a Fast Path architecture but Fast Path is implemented in the MCEE (Multi-Core Executive Environment). A certain number of cores are dedicated to Fast Path; Fast Past modules run in a dedicated execution space outside of Linux kernel. Forwarding is performed at the Fast Path level. 6WINDGate™ SDS architecture relies on Cache Manager and Fast Path Manager

modules to integrate and synchronise Fast Path processing and Slow Path / Control Plane in a transparent manner. 6WINDGate™ SDS delivers the highest possible performance MC architectures can sustain.

Control Plane Modules are the same for all 6WINDGate™ Profiles.

6WINDGate™ EDS/SDS Linux Networking Stacks includes 6WINDGate™ ADS Linux Networking Stack plus the Cache Manager module required to synchronise Fast Path processing and Slow Path / Control Plane.

Figure 4 shows how the different 6WINDGate™ Profiles can be used to offer the required level of performance according to the number of available cores.

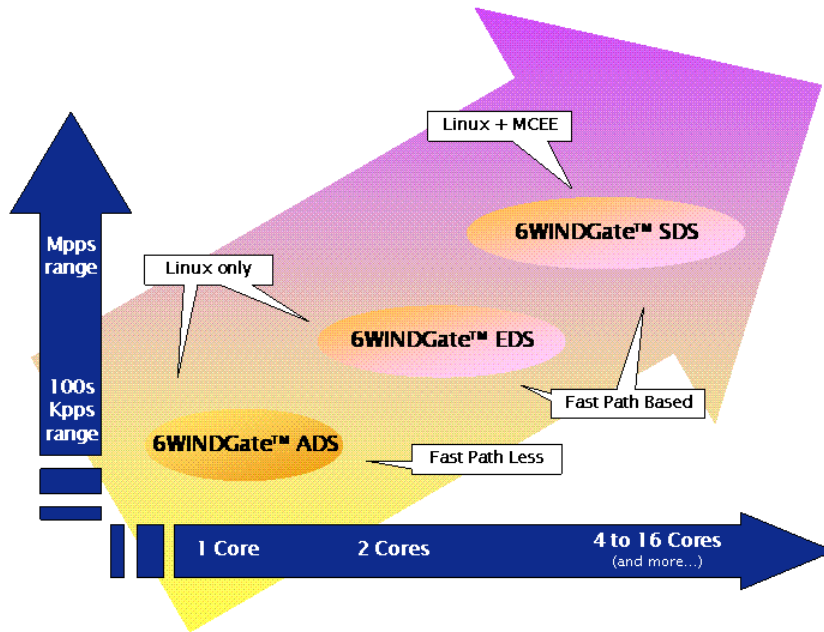


Figure 4: 6WINDGate™ Profiles

### 3.3 FAST PATH LESS SOLUTION - 6WINDGATE™ ADS

6WINDGate™ ADS (Figure 5) is targeted for low-end applications. Control Plane and Data Plane are co-localized and the entire 6WINDGate™ software runs in Linux environment. Forwarding is performed at the Linux Networking Stack level.

To build this profile, the following 6WINDGate™ building blocks are required:

- 6WINDGate™ ADS Linux Networking Stack
- 6WINDGate™ Control Plane Modules

6WINDGate™ ADS architecture can be used for single-core processors or MC processors. For MC architectures, it means that Linux kernel with a dedicated optimized SMP networking stack is running on all the cores according in order to process many packets simultaneously. 6WINDGate™ ADS Linux Networking Stack has been enhanced compared to a standard Linux stack (refer section 4.3).

6WINDGate™ Control Plane offering is modular and modules can be selected according to requirements.

6WINDGate™ XML-based Management Plane is provided to deliver a complete solution.

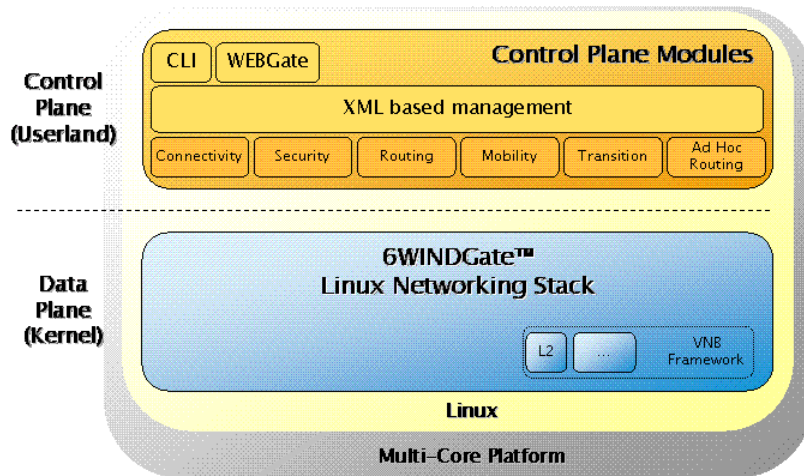


Figure 5: 6WINDGate™ ADS architecture

### 3.4 FAST PATH-BASED SOLUTIONS

Although an optimised networking stack for SMP architectures can significantly improve performance, 6WINDGate™ ADS is not able to fully benefit from the capabilities of a large number of cores. Even with an efficient implementation of the Linux networking stack in SMP mode, a Linux-based solution cannot scale because all the packets are managed by the stack and the Linux architecture is limited by OS latency and lock contention.

To go beyond this limit, software architecture shall provide a Fast Path implementation. Each protocol has to be redesigned to implement time-consuming and recurrent tasks at the Fast Path level while only complex packets are forwarded to the Slow Path.

6WINDGate™ is based on a Fast Path exception strategy that is described in Figure 6.

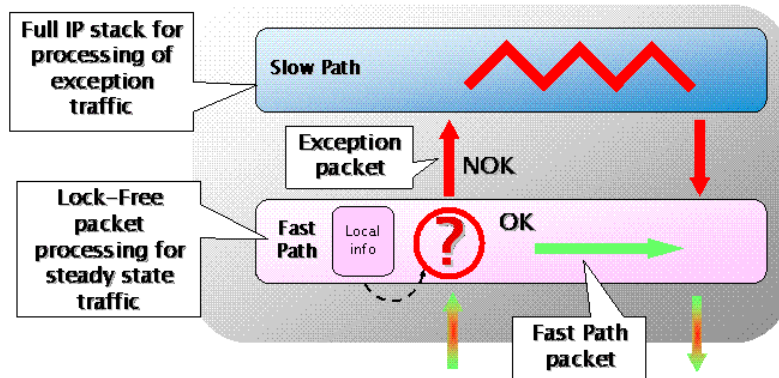


Figure 6: 6WINDGate™ Fast Path exception strategy

This concept applies to all protocols that have to be split in two parts:

- Fast Path only implements packet processing to be done on each packet. This is performed by a simplified IP stack that finds the necessary information in a local memory that has previously updated by high level protocols (signalling).
- When a received packet is too complex to be processed at the Fast Path level, it is forwarded to the Slow Path through an exception using a dedicated API called FPVI. This packet will be either managed at the Slow Path or at the Control Plane level. It can be noted that exception packets are only a few % of the traffic making useless to have a full and complex IP stack at the Fast Path level.
- Packets to be sent locally by the Slow Path or the Control Plane are directly injected in the outgoing flow.

Fast Path exception strategy is therefore the best trade-off between complexity and performance.

More, Fast Path exception concept can be used for a progressive approach to integrate smoothly new features in the Fast Path. For instance, if IPv6 is not yet implemented in the Fast Path, IPv6 packets are processed as exceptions to be managed at the Slow Path level. When you integrate 6WINDGate™ Fast Path - IPv6 forwarding and transition module, IPv6 packets can be processed at the Fast Path level delivering same level of performance as IPv4 forwarding without any change in the Control Plane and Management Plane.

### 3.4.1 6WINDGate™ EDS

6WINDGate™ EDS (Figure 7) is based on a Fast Path architecture targeted to mid-range applications. This Fast Path is implemented as a Linux kernel module between the Linux Networking Stack and the interface drivers. So, it does not require any MCEE. Compared to a standard Linux architecture, forwarding is performed at the driver level and not in the Linux Networking Stack. Only packets that cannot be processed by Fast Path are forwarded to the Linux Networking Stack (Slow Path).

6WINDGate™ EDS architecture relies on Cache Manager and Fast Path Manager modules to integrate and synchronise Fast Path processing and Slow Path / Control Plane in a transparent manner.

To build this profile, the following 6WINDGate™ building blocks are required:

- 6WINDGate™ Fast Path Modules (Fast Path Manager is part of the Fast Path - IP Forwarding module)
- 6WINDGate™ EDS Networking Linux Stack (including Cache Manager)
- 6WINDGate™ Control Plane Modules

6WINDGate™ Control Plane offering is modular and modules can be selected according to requirements.

6WINDGate™ XML-based Management Plane is provided to deliver a complete solution.

When using 6WINDGate™ EDS architecture on MC, Linux kernel in SMP mode with a dedicated optimized Fast Path is running on all the cores in order to process more packets simultaneously.

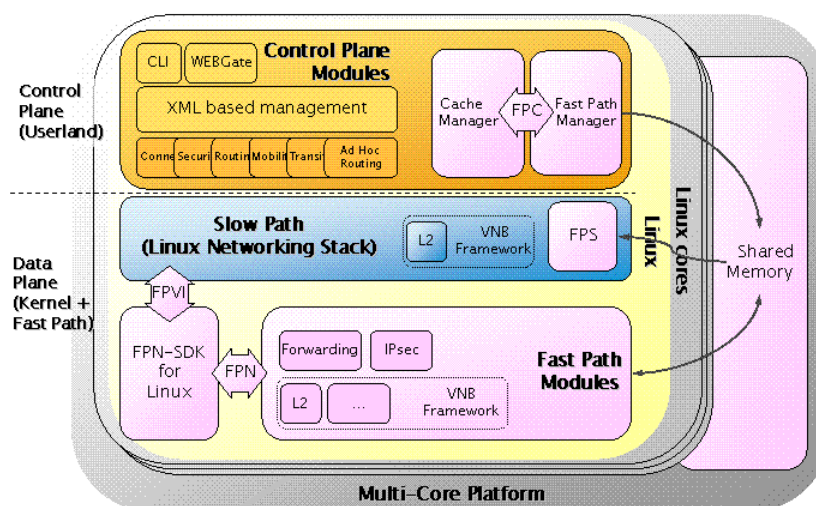


Figure 7: 6WINDGate™ EDS architecture

### 3.4.2 6WINDGate™ SDS

6WINDGate™ SDS (Figure 8) is based on a Fast Path architecture targeted to high performance applications. This Fast Path is identical to 6WINDGate™ EDS Fast Path but is implemented as a module running in the MCEE outside Linux environment. Forwarding is performed at Fast Path level for optimal performance. Only packets that cannot be processed by Fast Path are forwarded to the Linux Networking Stack (Slow Path). Similarly to 6WINDGate™ EDS, 6WINDGate™ SDS architecture relies on Cache Manager and Fast Path Manager modules to integrate and synchronise Fast Path processing and Slow Path / Control Plane in a transparent manner.

To build this profile, the following 6WINDGate™ building blocks are required:

- 6WINDGate™ Fast Path Modules (Fast Path Manager is part of the Fast Path IP Forwarding module)
- 6WINDGate™ SDS Linux Networking Stack (including Cache Manager)

- 6WINDGate™ Control Plane Modules

6WINDGate™ Control Plane offering is modular and modules can be selected according to requirements.

6WINDGate™ XML-based Management Plane is provided to deliver a complete solution.

6WINDGate™ SDS architecture is perfectly well suited to MC processors. Fast Path Modules are running on a certain number of dedicated cores in order to process many packets simultaneously with guaranteed performances. The Linux Networking Stack (Slow path) and Control Plane Modules are running on the rest of cores (possibly in SMP mode if more than one core is dedicated to Linux environment).

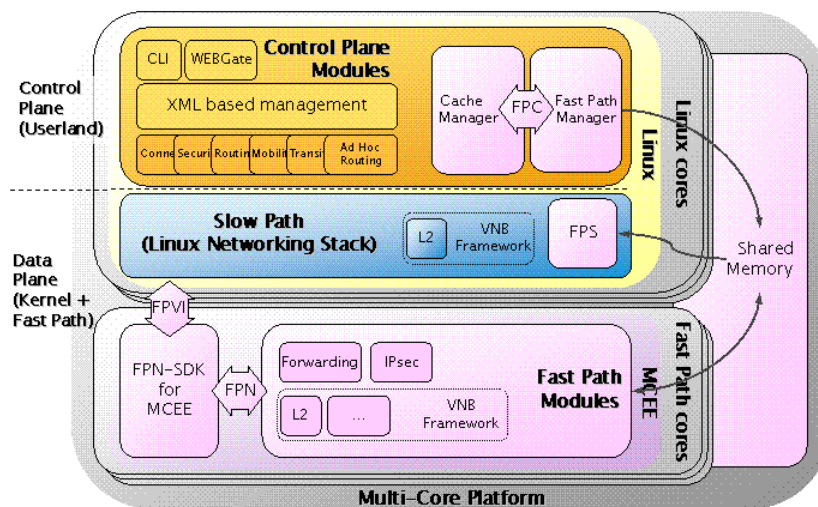


Figure 8: 6WINDGate™ SDS architecture

### 3.5 COMPARISON BETWEEN 6WINDGATE™ PROFILES

Figure 9 and Figure 10 compare the performance of the three 6WINDGate™ Profiles using IP forwarding (64 bytes) as an example.

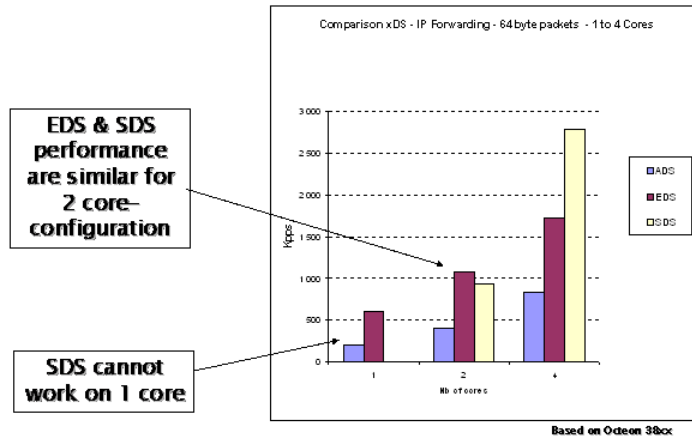


Figure 9: Performance comparison between 6WINDGate™ Profiles (1 to 4 cores)

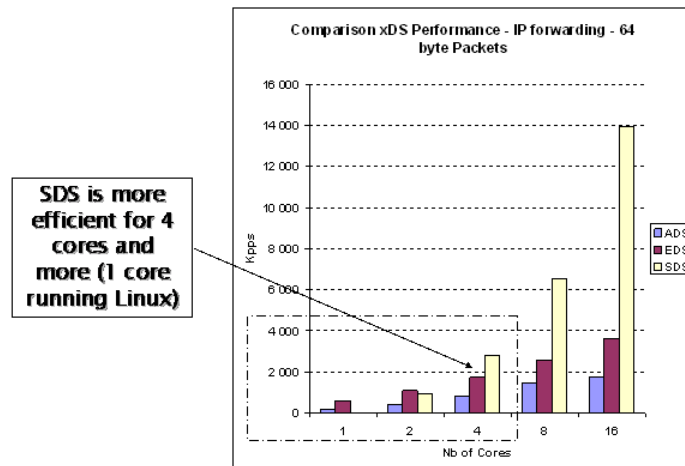


Figure 10: Performance comparison between 6WINDGate™ Profiles (1 to 16 cores)

Figure 11 summarises the different 6WINDGate™ Profiles according several criteria can be used to define the optimal configuration for each one.

	6WINDGate™ ADS	6WINDGate™ EDS	6WINDGate™ SDS
<b>Slow Path and Control Plane</b>	Linux SMP		
<b>Fast Path</b>	None	Linux Kernel Module	MCEE Application
<b>Protocol Implementation</b>	Control Plane + Slow Path	Extended with Fast Path support Each protocol is split between Fast Path / Slow Path / Control Plane	
<b>Performances</b>	Limited by Linux and Fast-Path less architecture	Application and Fast Path share cores	Fast Path performance is guaranteed
<b>Optimal configuration</b>	1 to 2 cores	2 cores More when no MCEE is available	2 cores and more (needs more than 1 core)

Figure 11: Comparison between 6WINDGate™ Profiles

- 6WINDGate™ ADS is a simple solution well adapted to small configurations (1 to 2 cores) with reasonable performance requirements but limited by Linux architecture and a standard Fast Path-less architecture.
- 6WINDGate™ EDS is well adapted to configurations between 2 to 4 cores. If we take the example of 2-core configuration, 6WINDGate™ EDS Fast Path can run on the 2 cores (SMP mode) while 6WINDGate™ SDS Fast Path can only run on 1 core as the other is dedicated for Linux. So, even if 6WINDGate™ SDS has a higher performance per core, 6WINDGate™ EDS could provide a more interesting solution. It has however to be noted that 6WINDGate™ EDS software and the application (if any) share the same cores. As a result, there could be an impact in terms of performance for each part. 6WINDGate™ EDS can also be an interesting solution for large configurations when no MCEE is available as performance reaches two times 6WINDGate™ ADS performance.
- 6WINDGate™ SDS is targeting configurations from 4 to n cores with a Fast Path running on a large number of these cores. It can be noted that although 6WINDGate™ EDS scales well on the number of cores, 6WINDGate™ SDS scales better as it does not rely on generic services of Linux Operating System and fully benefits from the limited but optimised services of the MC environment. As the Fast Path is using dedicated cores, performance is guaranteed. So, the difference in terms of performance is growing as the number of cores grows reaching 4 times for a 16-core configuration.

### 3.6 6WINDGATE™ EXTENSIONS FOR DISTRIBUTED ARCHITECTURES

As mentioned in section 2.3, MC technology is by essence scalable. To address larger configurations, higher performance or provide high availability features, hardware architectures can be extended by interconnecting several MC chips on a single board and / or by interconnecting several boards in a chassis as described in Figure 12.

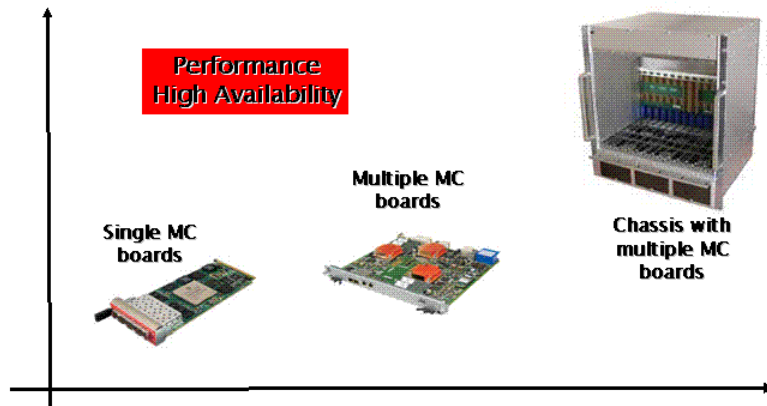


Figure 12: MC-based configurations

This section gives as an overview of 6WINDGate™ extensions for distributed architectures.

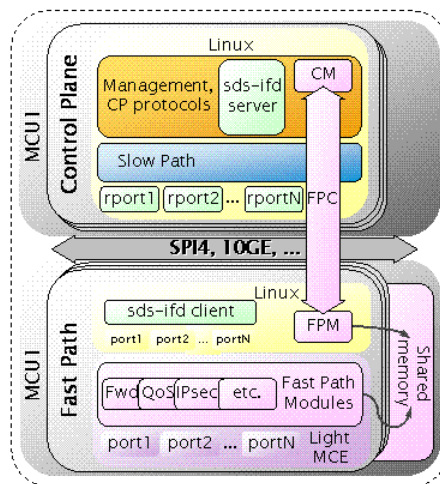


Figure 13: 6WINDGate™ Distributed Architecture

Figure 13 shows how 6WINDGate™ EDS or SDS API have been extended to support a distributed architecture.

The first MC chip is running Linux for the Control Plane and the Slow Path as the second one is running the Fast Path under Linux (6WINDGate™ EDS) or MCEE (6WINDGate™ SDS). To ensure communication between the two MC chips, Fast Path – Multi-instance module is required. It can rely on several media for

communication such as SPI-4 or Ethernet. It is based on client-server architecture; server is implemented on Cache Manager side and client on Fast Path Manager side.

6WINDGate™ APIs are extended as follows:

- FPVI: Fast Path ports appear as logical interfaces in Linux (RFPVI, Remote FPVI) and interface states are synchronized thanks to sds-ifd daemons
- FPC and FPS: Cache Manager and Fast Path Manager communicate over a TCP socket on the communication bus

### 3.6.1 Application: High Availability

To provide High Availability features, a common architecture is to use a board with two MC chips or two boards with a single MC. Each MC is running the same software, one active and the other in hot stand-by. If the first one fails, the second one can take over to provide the expected service without interruption. Monitoring of software health, duplication of information to let the system restart within the shortest period of time, detection of a problem, are part of an efficient solution.

6WINDGate™ architecture has been designed to meet these requirements thanks to extensions to 6WINDGate™ EDS and SDS and specific development to ensure a fast restart of 6WINDGate™ Control Plane protocols.

6WIND is developing such solutions to be available in a future road map. Should you be interested by such solutions, please contact 6WIND Support and Services team ([support@6wind.com](mailto:support@6wind.com)).

### 3.6.2 Application: Fast Path Extensions

To address large network configurations it can be necessary to use several MC chips to extend Fast Path features. It is possible to combine several MC chips and design a Fast Path managed by a set of cores belonging to different MC chips.

6WIND is developing such solutions to be available in a future road map. Should you be interested by such solutions, please contact 6WIND Support and Services team ([support@6wind.com](mailto:support@6wind.com)).

## 4 6WINDGATE™ BUILDING BLOCKS AND MODULES

### 4.1 6WINDGATE™ FEATURE SUMMARY

6WINDGate™ solution is a comprehensive and ready-to-use solution that provides many network protocol features (refer [www.6wind.com/RFC](http://www.6wind.com/RFC) or 6WINDGate™ Module Data Sheet for a detailed list of RFCs 6WINDGate™ complies with).

The following table summarises 6WINDGate™ features provided by 6WINDGate™ Profiles and based on the three different kinds of building blocks:

- 6WINDGate™ Linux Networking Stacks
- 6WINDGate™ Fast Path Modules
- 6WINDGate™ Control Plane Modules

Linux Networking Stacks (6WINDGate™ ADS, EDS & SDS)	Optimized stack for MC including Optimized SMP, built-in crypto engine management Optional library for ROHC Based on VNB framework (refer section 4.3.1) for L2 encapsulation Integrates Cache Manager for 6WINDGate™ EDS & SDS
--	--

Fast Path Modules (6WINDGate™ EDS & SDS)	IP Forwarding including Virtual Routing and ECMP
	IPsec
	L2 – VLAN, GRE, Bridging, Link Aggregation – based on VNB framework (refer section 4.3.1)
	QoS
	Traffic conditioning
	IP Reassembly
	IP tunnelling
	NAT and Filtering
	IPv6 Forwarding including Virtual Routing and ECMP
	IPv6 tunnelling and transition mechanism 6to4
	L2 Advanced including 802.1x, L2TP VPN
	ROHC
IPv4 Multicast	

	IPv6 Multicast
	Multi-Instance

Control Plane Modules (6WINDGate™ ADS, EDS & SDS)	Routing	Static RIP (IPv4-v6), RIPng, OSPFv2, OSPFv3, BGP-4, BGP-4+, IS-IS including Virtual Routing, ECMP (IPv4-v6), VRRP, PIMv4-SM, PIMv6-SM, IGMP/MLD snooping & proxy
	Security	IKE, IKEv2, EAP
	Connectivity	PPP, Multi-link PPP, PPPoE, CHDLC, VLAN, GRE, 6in6, 4in4, L2TP, DHCPv6-v4, DNS proxy, RADIUS Client
	Mobility	Home Agent, FMIP, Corresponding Node, Mobile Node, IPsec integration, NEMO, Proxy MIP
	IPv4-IPv6 Transition	NAT-PT
	Ad Hoc Routing	OLSR, OLSRv6

6WINDGate™ detailed roadmap is available upon request.

#### 4.2 6WINDGATE™ MODULE DATA SHEET

A detailed description of the different modules is provided in a data sheet (refer document [2]).

Each data sheet includes the following information:

- Module name
- Module features
- Management interfaces
- APIs
- 6WINDGate™ related modules
- Implementation details
- RFCs
- Supported hardware platforms

#### 4.3 6WINDGATE™ LINUX NETWORKING STACKS

Features provided by 6WINDGate™ Linux Networking Stacks differ according to the 6WINDGate™ profile.

- If the architecture is Fast Path-less, the Linux Networking Stack implements all the Data Plane (Slow Path and Fast Path) mechanisms. It directly interfaces network interfaces drivers.
- If the architecture includes a Fast Path, the Linux Networking Stack only implements the Slow Path part of the Data Plane. If so, it has to be integrated smoothly with 6WINDGate™ Fast Path

Modules. The 6WINDGate™ architecture has been designed to ease this integration. Fast Path-based architectures also raise the question of the integration of Control Plane with Fast Path Modules. 6WIND has chosen smart architecture design options to make this integration fully transparent, through the use of two modules called Cache Manager and Fast Path Manager that hide the complexity of the Fast Path to Control Plane. Control Plane does not have to be adapted to this new architecture.

As a summary, two Linux Networking Stacks are proposed:

- **6WINDGate™ ADS Linux Networking Stack** for Fast Path-less architectures. It has to be noted that this stack has been optimized to run on a mono-processor or in a multi-processor environment (such as a MC processor). It is based on VNB framework (refer section 4.3.1).
- **6WINDGate™ EDS and SDS Linux Networking Stacks** for Fast Path-based architectures. They include the Cache Manager module (refer section 5.2) and is based on VNB framework (refer section 4.3.1).

6WINDGate™ Linux Networking Stacks can be provided either as stand alone modules to be integrated in a Linux environment or integrated with other 6WINDGate™ building blocks. 6WIND has developed some specific enhancements to optimise Linux packet processing for MC architectures.

#### 4.3.1 VNB framework

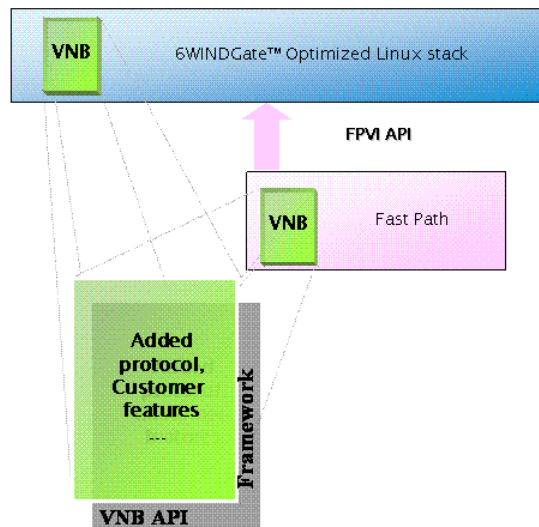


Figure 14: VNB Architecture

As described in Figure 14, VNB (Virtual Networking Blocks) is initially an in-kernel networking subsystem that follows the UNIX principle of achieving power and flexibility through combinations of simple modules (or tools). Each module is designed to perform a single and well defined task. The basic idea is straightforward: there are nodes and hooks that connect pairs of nodes. Data packets flow bi-directionally along the hooks from node to node. When a node receives a data packet, it performs some processing on it, and then (usually) forwards it to another node. The processing may be something as simple as adding/removing headers, or it may be more complicated or involve other parts of the system. VNB is derived from “Netgraph” technology.

6WIND has ported VNB framework at the Fast Path level to bring similar benefits for the integration of Fast Path Modules (refer section 4.4.2). Doing so, same source code for a VNB module can be used within the 6WINDGate™ ADS Data Plane or 6WINDGate™ EDS/SDS Fast Path.

VNB technology has various applications at each level of the software architecture. It is well-suited for especially for:

- Protocol encapsulation
- Feature performance optimization by separation of Kernel/Fast Path packet processing from Control Plane
- Assembly of several protocols in order to provide global solutions
- Driver integration

6WIND has extensively used the VNB framework to develop its own L2/L3 modules including:

- Layer 2
  - Bridging
  - VLAN
  - PPP / L2TP
  - 802.1x
  - Link Aggregation
  - ROHC
- Layer 3
  - GRE

For more details about VNB, refer document [3] for VNB.

#### 4.3.2 Others Enhancements in 6WINDGate™ Linux Networking Stack

- **Crypto Performances**

To improve IPsec performance, 6WIND has developed a high-level API to interface built-in crypto engines. This API is based on OCF interface that makes it possible to interface both synchronous and asynchronous crypto drivers. This OCF API is used by 6WINDGate™ Linux Networking Stack but is also available for application level modules (6WINDGate™ - Control Plane - Security including IKE protocol) as well as third parties that require efficient crypto mechanisms.

- **Locks**

6WIND has significantly improved the performances of the Linux stack in SMP mode by modifying locks on whole structures (giant locks) to make them atomic.

- **Feature Improvements**

6WINDGate™ Linux Networking Stack includes some major improvements such as:

- IP forwarding for virtual routing,
- ROHC library when no Fast Path is used,
- Multicast features (6WIND has contributed to the implementation of IPv6 multicast in the vanilla Linux distribution)

- Mobility features
- Etc.

## 4.4 6WINDGATE™ FAST PATH

### 4.4.1 6WINDGate™ Fast Path Modules

6WINDGate™ Fast Path Modules implement a large set of networking features for MC.

To achieve the highest level of performance, these modules can run in the MCEE (6WINDGate™ SDS). Thus they are therefore independent from the selected Operating System.

Fast Path Modules can also be implemented as Linux kernel modules located between the drivers and the Linux Networking Stack (6WINDGate™ EDS). This solution is well-suited for small MC configurations or if no specific MCEE is available for the selected MC (refer section 3.5).

For each networking feature, a 6WINDGate™ Fast Path Module has been designed, having in mind the following design rules:

- Fast Path Module shall be simple and efficient,
- Fast Path Module shall be easily integrated with a Linux-based Slow Path and Control Plane through simple APIs,
- Fast Path Module shall be portable.

The following 6WINDGate™ Fast Path Modules have been developed:

- Fast Path – IPv4 Forwarding (This module includes the Fast Path Manager module, refer section 5.3)
- Fast Path – IPsec
- Fast Path – L2 (This module is based on VNB framework, refer section 4.3.1)
- Fast Path – QoS
- Fast Path – Reassembly
- Fast Path – NAT & Filtering
- Fast Path – IPv6 Forwarding and Transition
- Fast Path – ROHC
- Fast-Path – IPv4 Multicast
- Fast-Path – IPv6 Multicast
- Fast Path – TCP Offload

To scale on distributed architectures, 6WINDGate™ Fast Path also includes a specific module:

- Fast Path – Multi-instance

### 4.4.2 Fast Path Module Integration

To provide a complete solution, the integration of the different Fast Path Modules is of key importance. Figure 15 describes how 6WINDGate™ Fast Path Modules are integrated.

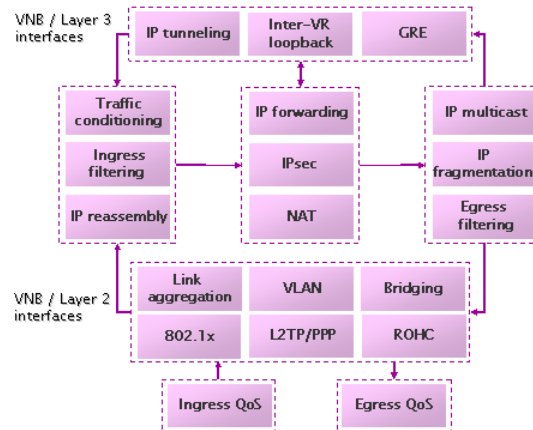


Figure 15: Fast Path architecture

As mentioned in section 4.3.1, VNB framework for Linux kernel has been ported at the Fast Past Level to provide same flexibility to integrate L2/L3 features. This VNB framework is used at both L2 and L3 levels.

Incoming packets are processed as follow (assuming all the functions are activated):

- Ingress QoS is applied first.
- Packet enters L2 processing; each 6WINDGate™ VNB module is dedicated to a specific function. If a packet has to be processed by several functions, it is done step by step within VNB framework.
- Packet is optionally reassembled and filtered (ingress); traffic conditioning (rate limitation) is applied.
- Layer 3 processing (forwarding, IPsec, NAT) are then applied.
- Packet is filtered, and if need be, fragmented.
- Packet enters again in L2 processing.
- Egress QoS is applied.

At each level, exception can happened if the packet cannot be handled at the Fast Path level.

In case L3 encapsulation is applied (IP tunnelling, GRE), this is considered as an interface and packet is processed by block named "VNB/layer 3 interfaces". If several encapsulations are applied, it is applied recursively.

### 4.4.3 IP Forwarding Example

Figure 16 details how a Fast Path module works using Fast Path IPv4 forwarding packet processing as an example.

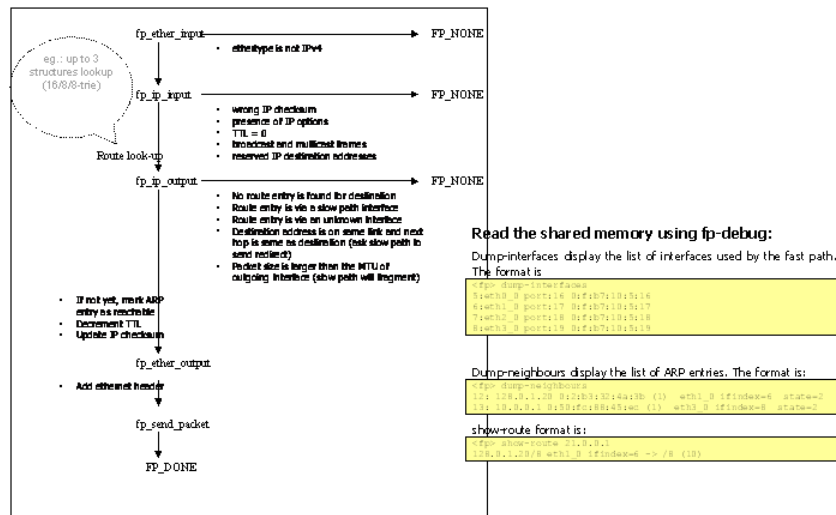


Figure 16: IPv4 Forwarding Packet Processing

Fast Path packet processing is responsible for parsing the incoming packet and checking if it can be forwarded with the information available in the local memory.

For that purpose, the following tests have to be performed to achieve the IPv4 forwarding function:

- Firstly, Fast Path module is checking if the incoming packet has an IPv4 ethertype. If the test is negative, the packet is considered as an exception and diverted to the Slow Path (6WINDGate™ Networking Linux Stack) to be handled.
- Then, the IPv4 packet is checked to detect packets that cannot be managed at the Fast Path level such as packets with wrong checksum, IP options or a TTL at 0. Broadcast/multicast packets as well as packets reserved IP destination addresses are also detected. If such an event occurs, the packet is considered as an exception and diverted to the Slow Path (6WINDGate™ Networking Linux Stack) to be handled.
- Once done, if the packet has not been diverted, IP lookup takes place to check if there is valid entry in the routing table. IP lookup in 6WINDGate™ IPv4 forwarding Fast Path module is implemented as M-trie 16/8/8 so that any flow is looked up within a fixed number of memory accesses to achieve the highest performance. As this stage, it is checked:
  - if a route is found for destination,
  - if the route found in the table is via a Slow Path interface. It can be for instance for a PCI or WiFi interface managed by the Slow Path,
  - if the route found in the table is via an unknown interface,

If such an event occurs, the packet is considered as an exception and diverted to the Slow Path (6WINDGate™ Networking Linux Stack) to be handled.

- At this stage, Fast Path has to perform some processing to mark ARP entry as reachable, decrement TTL, update IP packet checksum and add ethertype before enqueueing the packet to be forwarded.

This simple example shows how a Fast Path module works knowing that processing can be more complex when you add functions such as Virtual Routing.

Each Fast Path module has to be integrated within the complete 6WINDGate™ Fast Path architecture to provide a complete solution (refer Figure 15). For instance, ingress QoS (respectively egress QoS) mechanism can be added before (respectively after) IPv4 forwarding.

This example also illustrates the flexibility of 6WINDGate™ architecture and how it can be used to progressively develop Fast Path modules. For instance, the ethertype test in this example redirects IPv6 packets to the Slow Path. As soon as an IPv6 forwarding Fast Module is available, the IPv6 ethertype is identified and the packet stays at the Fast Path level for further processing.

At last, Figure 16 shows how 6WINDGate™ tools (fp-debug) can be used to dump the local shared memory and display useful information.

## 4.5 6WINDGATE™ CONTROL PLANE

6WINDGate™ Control Plane Modules provide Control Plane features. The following 6WINDGate™ Control Plane Modules are proposed:

- Control Plane – Routing
- Control Plane – Security
- Control Plane – Connectivity
- Control Plane – Mobility
- Control Plane – IPv4-IPv6 Transition
- Control Plane – Ad-hoc Routing

6WINDGate™ Control Plane Modules can be provided either as stand alone modules to be integrated in a Linux environment or integrated with other 6WINDGate™ building blocks.

## 4.6 6WINDGATE™ XML-BASED MANAGEMENT SYSTEM

6WINDGate™ modules come with their associated management modules. 6WINDGate™ management is built on an XML-based architecture.

### 4.6.1 Introduction

6WINDGate™ features are managed through the eXtensible Management System (XMS) depicted in Figure 17. Detailed architecture is described in document [3] for XMS.

The configurations are described as XML documents (refer document [3] for full details).

XMS is split into two parts:

- **XMS Configuration Manager:** this part is responsible for processing the XML configuration and configures services accordingly; this is xmsd daemon.

- **XMS Clients:** these are the applications responsible for XML configuration generation and formatting. 6WINDGate™ includes two full featured XMS clients: a CLI and a Web-based management interface. Other XMS clients can be developed.

The XMS Configuration Manager and XMS Clients use a standard UNIX stream socket for communication. XML messages are exchanged through the XIPC protocol.

As XML configuration processing is done in XMS Configuration Manager, adding another XMS Client is done through a simple interfacing, not a complete re-development.

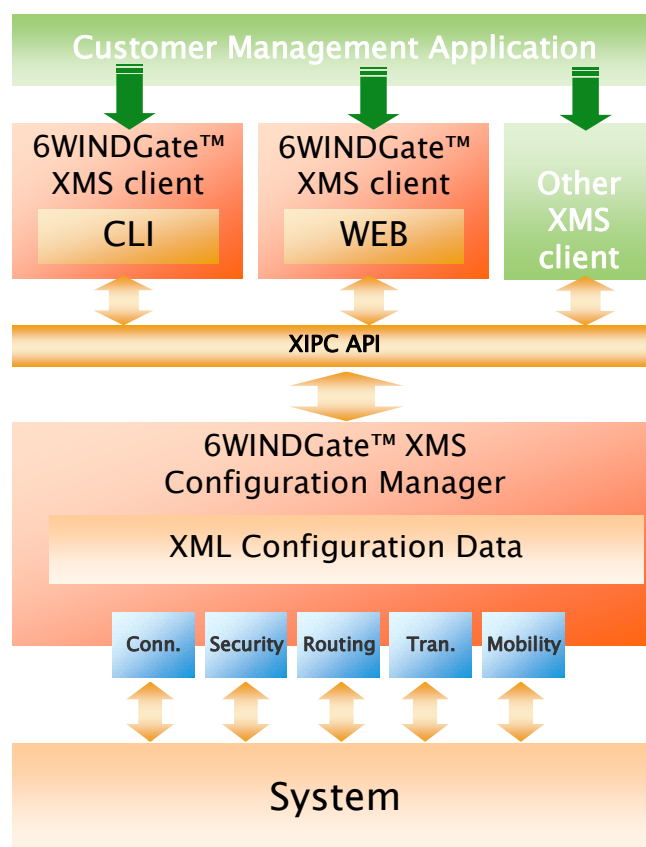


Figure 17: 6WINDGate™ extensible Management System (XMS)

#### 4.6.2 XMS Configuration Manager

The XMS Configuration Manager is designed to be as modular as possible; it is split into two parts:

- The **core** implementation, which is responsible for dispatching the XML configuration file to the relevant services. It also handles the integration of the different service modules.
- The **service modules**, which are responsible for configuring the system, according to the XML configuration received. They are also responsible for keeping track of the current service configuration.

These two parts are fully written in C99 and therefore fully portable.

### 4.6.3 Core

The XMS Configuration Manager core implementation relies on a standard UNIX stream socket to listen to the different XIPC requests sent by XMS Clients. These requests are configuration requests or commits, statistics requests or configuration editions.

The core implementation provides an API to the different software modules in order to enable XML processing and gives access to provided implemented facilities (error notification or netlink messages processing for example).

### 4.6.4 Service Modules

Service modules are integrated with the core implementation using a standard API enabling them to register and process the XML configuration of the service for which they are responsible.

This API also provides a way to feedback to the core implementation the current service status, as an XML configuration.

### 4.6.5 XMS Clients

XMS Clients are designed to be as simple as possible. They are just focused on XML configuration generation, XIPC messages generation and processing and configuration display.

Then, it is possible for these clients to implement advanced features according to customer needs.

The XMS clients will generate the XML configuration according to the configuration described. As 6WINDGate™ embeds libxml2, it is possible to use either XPATH or the tree API to generate the XML configuration.

## 5 6WINDGATE™ API

To interconnect 6WINDGate™ building blocks and modules, 6WINDGate™ relies on a certain number of API that are described in this section. These API can also be used to extend 6WINDGate™ feature set (refer section 7).

This section also describes two key components of Fast Path-based 6WINDGate™ architecture: Cache Manager and Fast Path Manager.

### 5.1 API LIST AND OVERVIEW

The following table identifies the different APIs between 6WINDGate™ modules for Fast Path less then Fast Path-based architectures.

#### 5.1.1 API for 6WINDGate™ ADS

API	Between	Purpose
<b>Control Plane / Linux Networking Stack API</b>		
Netlink	6WINDGate™ Linux Networking Stack and Control Plane	Netlink Linux API is used to exchange information between userland and kernel space. It is used for: <ul style="list-style-type: none"> <li>Interface Management</li> <li>Route management</li> <li>Asynchronous monitoring of kernel events</li> </ul>
PF_KEY	6WINDGate™ Linux Networking Stack and Control Plane (security protocols)	PF_KEY interface is used by IKE & IKEv2 to configure Security Policy Database and Security Association Database and receive notifications from Linux stack.
<b>Crypto API</b>		
OCF Open Cryptographic Framework	6WINDGate™ Linux Networking Stack and Control Plane interface for crypto-engines	OCF is an asynchronous API that provides Linux Networking Stack and Control Plane with a uniform access to one or more crypto devices. Detailed information about OCF can be found here: <a href="http://www.openbsd.org/papers/ocf.pdf">http://www.openbsd.org/papers/ocf.pdf</a> .
<b>Management API</b>		
XIPC eXtended Inter Process Communication	XMS and XMS clients	XIPC is used for sending configuration and monitoring XML messages to 6WINDGate™ XMS. It is implemented by 6WINDGate™ XMS clients, including CLI, WEBGate or available for third-party clients.

XMS system interfaces	XMS and 6WINDGate™ Control Plane and Slow Path	These interfaces configure 6WINDGate™ Control Plane and Slow Path through various, standard Linux APIs (Netlink, ioctl, system calls, etc.).
-----------------------	--	--

### 5.1.2 API for 6WINDGate™ EDS and SDS

API	Between	Purpose
<b>Packet processing API</b>		
FPN Fast Path Networking	MCEE and Fast Path Modules (6WINDGate™ SDS)	FPN API interfaces the MCEE (6WINDGate™ SDS) or the network driver (6WINDGate™ EDS) that helps sending and receiving packets with the 6WINDGate™ Fast Path software. This API is divided in two parts: <ul style="list-style-type: none"> <li>o A generic part</li> <li>o A MCEE-dependent or driver-dependent part for maximal portability</li> </ul>
	Network Driver and Fast Path Modules (6WINDGate™ EDS)	
FPVI Fast Path Virtual Interface	6WINDGate™ Fast Path Modules and Linux Networking Stack	FPVI API forwards exceptions (packets that are too complex to be processed at the Fast Path level) to 6WINDGate™ Linux Networking Stack.  Exceptions for instance include ARP packets, OSPF packets, ICMP, etc.
<b>Linux Networking Stack / Fast Path synchronization API</b>		
FPS Fast Path Statistics	6WINDGate™ Linux Networking Stack and Fast Path Modules	FPS API gathers counters from Fast Path Modules and builds global statistics for the system (Fast Path + Slow Path).  Global statistics are exported to unmodified Linux applications (CLI, Web, SNMP, Routing...) via Linux standard APIs.
FPC Fast Path Control	6WINDGate™ Linux Networking Stack and Fast Path Modules	FPC API is the communication API between Cache Manager and Fast Path Manager.  The goal of the FPC is to synchronize the states of the Slow Path with the Fast Path without any modification into the existing services to distribute the information. It means that you do not have to modify any existing daemons or kernel modules to interface Fast Path functions when you use the FPC and the Cache Manager.
Netlink	6WINDGate™ Linux Networking Stack and Cache Manager	Netlink API notifies Cache Manager of kernel events and state changes for interfaces, L2/L3 tables, etc.  Netlink is also used to interface 6WINDGate™ Linux Networking Stack and Control Plane (refer 5.1.1).

PF_KEY	6WINDGate™ Linux Networking Stack and Cache Manager	PF_KEY API notifies Cache Manager of kernel events and state changes for IPsec Security Associations.  PF-KEY is also used to interface 6WINDGate™ Linux Networking Stack and Control Plane (refer 5.1.1).
NETFPC	6WINDGate™ Control Plane Modules and Fast Path Modules	NETFPC triggers events in the Fast Path Executive from the 6WINDGate™ Control Plane.
<b>Crypto API</b>		
OCF Open Cryptographic Framework	6WINDGate™ Linux Networking Stack, Fast Path and Control Plane interface for crypto-engines	OCF is an asynchronous API that provides Fast Path, Linux Networking Stack and Control Plane with a uniform access to one or more crypto devices. Detailed information about OCF can be found here: <a href="http://www.openbsd.org/papers/ocf.pdf">http://www.openbsd.org/papers/ocf.pdf</a> .
<b>Management API</b>		
XIPC eXtended Inter Process Communication	XMS and XMS clients	XIPC is used for sending configuration and monitoring XML messages to 6WINDGate™ XMS.  It is implemented by 6WINDGate™ XMS clients, including CLI, WEBGate or available for third-party clients.
XMS system interfaces	XMS and 6WINDGate™ Control Plane and Slow Path	These interfaces configure 6WINDGate™ Control Plane and Slow Path through various, standard Linux APIs (Netlink, ioctl, system calls, etc.).

## 5.2 CACHE MANAGER

The Cache Manager is a Control Plane software module that performs synchronization between Slow Path and Fast Path. It listens to the Kernel updates realized by the Control Plane Modules (ARP and NDP entries, L3 routing tables, Security Associations...) and synchronizes the Fast Path with this information.

Thanks to the Cache Manager, no change is required in the Control Plane Modules to be integrated with Fast Path modules. It is hardware-independent.

## 5.3 FAST PATH MANAGER

The Fast Path Manager can be considered as a Fast Path Linux driver. The Fast Path Manager updates Fast Path tables (via shared memory, or using NETFPC) according to messages exchanged with Cache Manager through the FPC API.

## 6 PACKET PROCESSING

This section illustrates how the different 6WINDGate™ building blocks and modules interact to process a packet and which APIs are used taking the IP forwarding / routing as an example.

### 6.1 FAST PATH LESS ARCHITECTURE (6WINDGATE™ ADS)

Figure 18 describes an example of 6WINDGate™ ADS implementation.

All 6WINDGate™ ADS software is running under Linux and is using a certain number of cores.

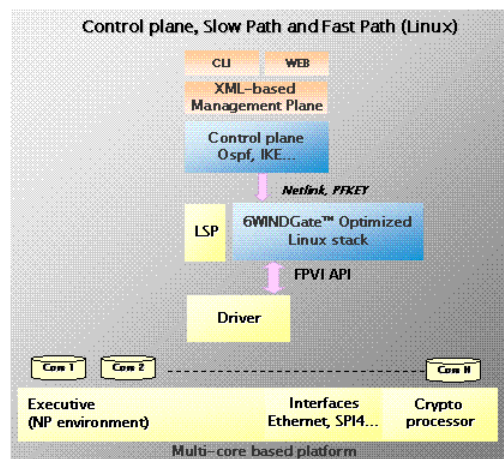


Figure 18: 6WINDGate™ ADS Implementation Example

6WINDGate™ ADS is based on a standard architecture. A complete IP stack (6WINDGate™ ADS Linux Networking Stack) is implemented in the kernel in SMP mode. Packets are processed at the Linux Networking Stack level either they are forwarded or processed locally (ARP packets for instance). More complex packets such as routing packets or IKE packets are forwarded to the relevant module in the Control Plane that use standard Linux API (Netlink, PF\_KEY) to update information in the Linux Networking Stack.

### 6.2 FAST PATH-BASED ARCHITECTURES

#### 6.2.1 6WINDGate™ EDS

Figure 19 describes an example of 6WINDGate™ EDS implementation.

All 6WINDGate™ EDS software is running under Linux and is using a certain number of cores in a flexible manner.

Figure 19 shows how Fast Path software is exchanging information with Linux software using 6WINDGate™ API and a shared memory architecture.

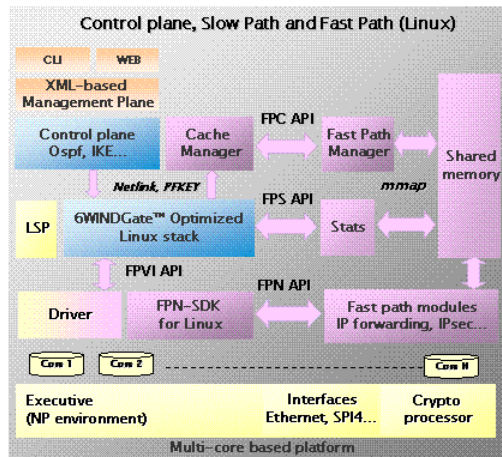


Figure 19: 6WINDGate™ EDS Implementation Example

6WINDGate™ EDS is based on a Fast Path architecture. Let us check how different types of packets are processed taking IP unicast forwarding and routing.

- Forwarded packet:
  - Packet is received through FPN interface from the network driver. Packet header is analyzed by Fast Path – IP forwarding module to check if it can be processed at the Fast Path level. It can be processed if a valid entry can be found in the L2/L3 forwarding table located in the shared memory. If so, Fast Path - IP forwarding module sends the packet over the FPN API to be transmitted by the network driver over the right network interface.
- Exception packet:
  - Packet is received through FPN interface. Packet header is analyzed by Fast Path – IP forwarding module that finds it is an exception packet (ARP packet, routing packet...). Fast Path – IP forwarding module forwards this packet to the Slow Path through the FPVI API. According to the type of packet, the Slow Path will process it (ARP packet for instance) or will forward it again to the Control Plane (routing protocol packet for instance). If the packet can be processed by the Slow Path, the Slow Path updates if necessary the L2 Forwarding Table located in the shared memory through the FPC API between the Cache Manager that intercepts the information and the Fast Path Manager.
- Received routing packet:
  - Received packet is a routing packet. Slow Path notifies the event to the relevant Control Plane Module (for instance OSPF in Control Plane - Routing Module) through the NetLink API. Packet is copied to the user land to be processed by the Control Plane Module. Once processed, the Control Plane Module updates if necessary the L3 Forwarding Table located in the shared memory through the FPC API between the Cache Manager that intercepts the information (for instance route addition or deletion) and the Fast Path

Manager.

- Transmitted routing packet:
  - In case a Control Plane Module has to transmit a packet (for instance an OSPF Hello packet), this packet is injected in the Fast Path by the Slow path.

## 6.2.2 6WINDGate™ SDS

Figure 20 describes an example of 6WINDGate™ SDS implementation.

6WINDGate™ SDS architecture is based on two parts:

- Fast Path is running in the MCEE.
- Slow Path, Control Plane and Management Plane are running under Linux

The distribution of cores is fully flexible but generally a large number of cores are dedicated to Fast Path to provide high forwarding performances. This distribution also depends on the applications running on the MC.

Figure 20 shows how Fast Path software is exchanging information with Linux software using 6WINDGate™ API and a shared memory architecture.

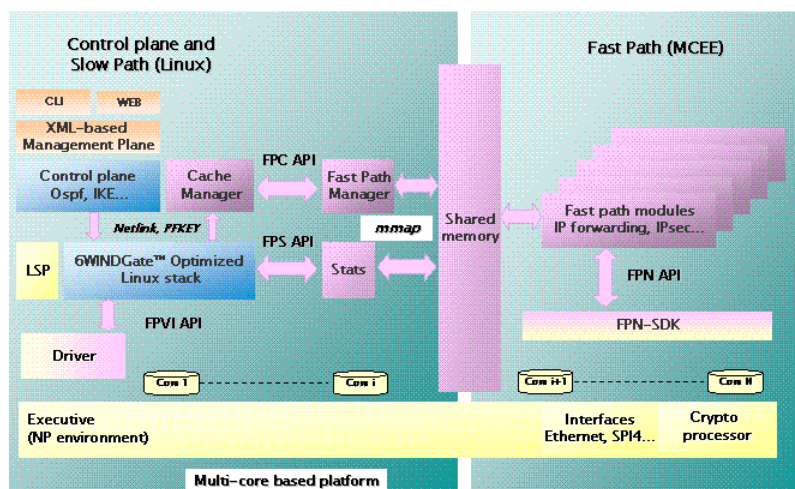


Figure 20: 6WINDGate™ SDS Implementation Example

Let us check how different types of packets are processed taking IP unicast forwarding and routing.

- Forwarded packet:
  - Packet is received through FPN interface from the MCEE. Packet header is analyzed by Fast Path – IP forwarding module to check if it can be processed at the Fast Path level. It can be processed if a valid entry can be found in the L2/L3 forwarding table located in the shared memory. If so, Fast Path - IP forwarding module sends the packet over the FPN API to be transmitted over the right network interface.
- Exception packet:

- Packet is received through FPN interface. Packet header is analyzed by Fast Path – IP forwarding module that finds it is an exception packet (ARP packet, routing packet...). Fast Path – IP forwarding module forwards this packet to the Slow Path through the FPVI API. According to the type of packet, the Slow Path will process it (ARP packet for instance) or will forward it again to the Control Plane (routing protocol packet for instance). If the packet can be processed by the Slow Path, the Slow Path updates if necessary the L2 Forwarding Table located in the shared memory through the FPC API between the Cache Manager that intercepts the information and the Fast Path Manager.
- Received routing packet:
  - Received packet is a routing packet. Slow Path notifies the event to the relevant Control Plane Module (for instance OSPF in Control Plane - Routing Module) through the NetLink API. Packet is copied to the user land to be processed by the Control Plane Module. Once processed, the Control Plane Module updates if necessary the L3 Forwarding Table located in the shared memory through the FPC API between the Cache Manager that intercepts the information (for instance route addition or deletion) and the Fast Path Manager.
- Transmitted routing packet:
  - In case a Control Plane Module has to transmit a packet (for instance an OSPF Hello packet), this packet is injected in the Fast Path by the Slow path.

## 7 6WINDGATE™ EXTENSIBILITY

This section explains how 6WINDGate™ feature set can be extended to develop customised applications.

### 7.1 6WINDGATE™ OPEN ARCHITECTURE

6WINDGate™ solution is a comprehensive and ready-to-use solution that provides many network protocol features (refer [www.6wind.com/RFC](http://www.6wind.com/RFC) for a detailed list of RFCs 6WINDGate™ complies with).

However, customers often need to add specific added value features that have to be smoothly integrated with 6WINDGate™ built-in ones. Integration includes functional and management integration.

This section describes how features can be integrated in 6WINDGate™ at different levels using some examples.

### 7.2 INTEGRATION OF A VNB MODULE

VNB framework is very well-suited to implement additional L2/L3 features (refer section 4.3.1). To illustrate such integration, implementation of "CoS to VLAN association" will be used; a CoS level is associated to each VLAN identifier.

Requirements for this function can be summarised as follow:

- **Packet reception**
  - packet arrives into ETHER node, is forwarded to VLAN node
  - VLAN node de-capsulates VLAN header and sends packet to link\_n hook, n being the VLAN id
  - CoS2VLAN node receives packet and possibly adds CoS information as ancillary data into mbuf
  - EIFACE receives packet, which seems to be received on a logical Ethernet interface (bnetX) in the Linux IP stack
- **Packet emission**
  - packet arrives from IP stack into EIFACE node, is forwarded to CoS2VLAN node
  - NSN CoS2VLAN node sends packet to
    - 'link\_n' hook, n being the VLAN id determined by CoS information
  - OR
    - 'nomatch' hook if packet does not need to be tagged
  - VLAN node receives packet, adds VLAN header and sends it to ETHER node
  - ETHER node receives packet and enqueues it for emission

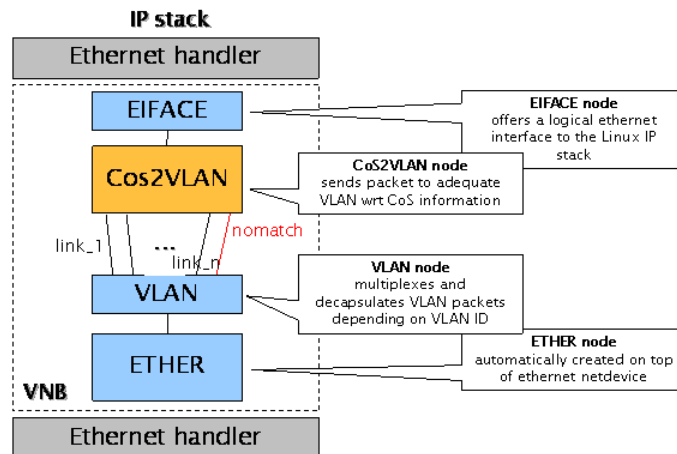


Figure 21: Integration of CoS to VLAN association function with VNB framework

Figure 21 shows how the “CoS to VLAN association” VNB module interacts with the existing 6WINDGate™ VNB modules (in blue).

Cos2VLAN module is inserted between VLAN module that manages VLAN multiplexing and de-capsulation and EIFACE module that offers a logical interface to the Linux IP stack. Cos2VLAN module remains very simple as it benefits from the services of VNB framework to hook the added module with the existing ones. All the existing features do not require any modification making improvements very simple to develop, integrate and test.

Such architecture can be implemented at both Fast Path and Linux kernel levels thanks to the portable VNB framework. Cos2VLAN software is identical in both cases.

6WIND provide some tools to help the development of Fast Path modules (refer section 8).

### 7.3 CUSTOMER CONTROL PLANE MODULE WITH 6WINDGATE™ FAST PATH MODULES

As 6WINDGate™ offering is fully modular, it is possible to easily integrate a customer Control Plane module with 6WINDGate™ Linux Networking Stack and Fast Past without making any modification in the Control plane assuming it is running smoothly on top a Linux stack on a standard processor architecture. This can be done thanks to 6WINDGate™ architecture that relies on the Cache Manager and Fast Past Manager modules.

Let us take the example of the integration of a unicast routing protocol such as OSPF. In a standard Linux architecture, its role is to receive protocol packets, analyse these packets and decide to update the Linux kernel forwarding table through the Linux netlink API. This API is also used to manage interfaces and addresses over these interfaces.

The main role of the Cache Manager is to listen to the updates sent by the OSPF module, to intercept them to synchronise the Fast Path forwarding table instead of the Linux forwarding table. This is done through the FPC API between the Cache Manager and the Fast Path Manager. Doing so, the OSPF module does not have any knowledge of the MC based architecture and is not modified at all.

As a result, the integration of the external routing protocol is straightforward and performance can be immediately and drastically improved benefiting from the highest possible performance MC architecture

can sustain. As 6WINDGate™ IP Forwarding Fast Path Module supports Virtual Routing, routing protocols inherit from this feature at the Fast Level.

This approach can be extended to all userland protocols using standard Linux APIs such as an IKE module using PF-KEY to interface an IPsec Fast Path or a PPP server module using netlink API to interface PPP tunnelling in the Fast Path.

## 7.4 CUSTOMIZING 6WINDGATE™ MANAGEMENT SYSTEM

6WINDGate™ comprehensive networking feature set is ideal to build new generation multi-services IP-centric equipment. 6WINDGate™ smart software architecture enables OEMs to perform unmatched customization to their end-product. Reflecting software development best practices, 6WINDGate™ software provides true openness at all level of its architecture, enabling an easy integration of additional software as described in previous sections. OEMs can then get quickly to market high-performing, full-featured multi-services equipments or systems.

It is very important to provide solutions to also integrate the added feature with 6WINDGate™ Management System in order to provide a consistent solution with the management of 6WINDGate™ built-in features.

6WINDGate™ eXtensible Management System (XMS) has been designed for that purpose and provides all configuration and management tools for a complete solution by providing standard interfaces like an industry-standard CLI and a Web-based management (WEBGate). On top of providing immediately deployable full-featured configuration tools, the XMS framework, a 6WIND-patented innovation, was engineered to provide high customization capabilities to those user interfaces, as well as to the system configuration manager.

You will find in document [4] a complete description of such an integration using as an example the integration of OpenVPN to add SSL tunnels over UDP and TCP functionality on top of 6WINDGate™.

This document will show you how to:

- Compile OpenVPN with 6WINDGate™ and build a complete binary image,
- Define a simple XML scheme for OpenVPN
- Extend 6WINDGate™ CLI and WEBGate to configure OpenVPN,
- Implement the XMS service for OpenVPN,
- Compile the whole software and build a complete binary image including the management extensions.

## 8 6WINDGATE™ VIRTUAL FAST PATH SOLUTION

Developing low-level software such as Fast Path modules in a MC environment is more complex compared to standard software development. Development tools are also less sophisticated knowing fixing a bug is more difficult. Fast Path development could be time-consuming and expensive as it could require a lot of hardware resources.

Thanks to its experience in MC software development, 6WIND proposes a development strategy based on a specific version of 6WINDGate™ using a Virtual Fast Path.

A Virtual Fast Path is a full-featured Fast Path running as an application in user land on an x86 hardware platform. It has not to be mistaken for 6WINDGate™ EDS that includes a kernel implementation of the Fast Path.

6WINDGate™ Virtual Fast Path allows software engineers to develop Fast Path modules in a user-friendly environment with standard development tools and using inexpensive x86 hardware platforms.

A first step of development can be done in such environment to check the behaviour of the developed code as well as the interface with 6WINDGate™ software.

Once done, a second step can be performed on the MC hardware to check the performances on the final hardware target.

In maintenance phase, 6WINDGate™ Virtual Fast Path can be used to reproduce hardware – independent bugs and reduce time to fix them.

Using 6WINDGate™ Virtual Fast Path solution has the following benefits:

- Significant reduction of software development time by using user-friendly environments and by paralleling development tasks on inexpensive x86 development platforms.
- Software quality improvement.
- Faster response time to fix hardware – independent bugs.
- Reduction of hardware costs as only a few MC hardware platforms are required to make final integration and tests.
- Fast Return on Investment on the acquisition of 6WINDGate™ Virtual Fast Path solution.

## 9 EVALUATION AND FURTHER READING

The purpose of this document is to give you:

- A better understanding of 6WINDGate™ architecture
- A first level of information about 6WINDGate™ modules
- A better understanding about the possible integration of 6WINDGate™ with your application

After this first reading, we suggest you should evaluate 6WINDGate™ software. 6WINDGate™ evaluation software is a full-featured and ready to use software with all management tools included.

Process for evaluating 6WIND software is very simple:

- A Non Disclosure Agreement and a Software Evaluation Agreement have to be signed. This second document defines the software version to be evaluated (6WINDGate™ profile, 6WINDGate™ version, processor, evaluation board).
- Evaluation is done in binary form and 6WINDGate™ binary can be downloaded once the Software Evaluation Agreement is signed.
- 6WIND Customer Support team will open a dedicated Customer Zone on 6WIND Web site and provide complete support during the evaluation period.
- Evaluation also gives you access to a large set of detailed documentation including:
  - Detailed test reports on hardware used for evaluation
  - Software Architecture Documents and Software Design Documents
  - Developer's Guide
  - User's Guide
  - 6WINDGate™ FAQ

### **COMPANY CONTACTS:**

France Headquarters  
33.1.39.30.92.10  
[6wind-contact@6wind.com](mailto:6wind-contact@6wind.com)

USA  
1.650.968.8768  
[6wind-us-sales@6wind.com](mailto:6wind-us-sales@6wind.com)

Asia  
82.2.6203.3088  
[6wind-asia-sales@6wind.com](mailto:6wind-asia-sales@6wind.com)