TEXAS INSTRUMENTS
20450 Century Boulevard
Germantown, MD 20874
Fax: (301) 515-7954

NETFP Proxy

Software Design Specification (SDS)

Revision A

QRSA000XXXX

May 30, 2015

Revision Record	
Document Title: **Software Design Specification**	
Revision	Description of Change
A	1. Initial Release

Note: Be sure the Revision of this document matches the QRSA record Revision letter. The revision letter increments only upon approval via the Quality Record System.

TABLE OF CONTENTS

TABLE OF FIGURES

1 Scope

This document describes the functionality, architecture, and operation of the Network Fast Path (NetFP) Proxy component (NetFP-Proxy) of NetFP library.

2 References

The following references are related to the feature described in this document and shall be consulted as necessary.

No	Referenced Document	Control Number	Description
1	Syslib User Guide		SYSLIB User Guide
2	NETFP API Documentation	N/A	NETFP Doxygen API documentation
3	SYSLIB Unit Test document		SYSLIB Unit Test documentation
4	NETFP SDS	SDOCM00117686	NETFP Software Design Specification

Table 1. Referenced Materials

3 Definitions

Acronym	Description
API	Application Programming Interface
DSP	Digital Signal Processor
NETFP	Network Fast Path Library
PA	Packet Accelerator
SA	Security Accelerator
NETCP	Network coprocessor
JOSH	Job Scheduling

Table 2. Definitions

4 NETFP Proxy

4.1 General

The NETFP Proxy is designed as a component of NetFP subsystem responsible for all interactions between Fast Path and Linux networking stack. It is instantiated in ARM processing realm. There can be multiple NetFP subsystems, but there is always one and only one NetFP proxy and one and only one NetFP server within a subsystem.

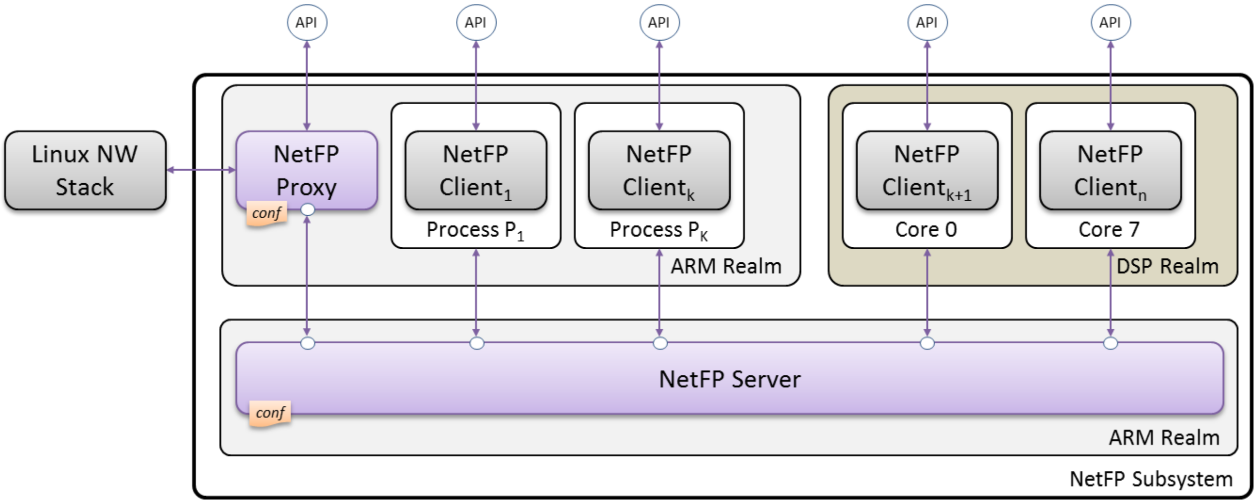


Figure 4-1. NetFP Subsystem

NetFP Proxy is delivered as a module within SYSLIB software package in source code, together with build environment, and with pre-built executable.
In addition, NetFP proxy is delivered with a customizable plug-in component. The example plugin code, build system and pre-build shared object provide an API for policy offload, interface and route management.

NetFP proxy supports multiple targets: K2H, K2K, and K2L.

NetFP proxy interacts with NetFP server using NetFP library. From that perspective, NetFP proxy behaves similar to “normal” NetFP Client in ARM realm.

NetFP proxy interface with Linux Networking stack consists of following parts:

- interfacing kernel’s IPSec manager module,
- maintaining interface and route caches for relevant fast path objects,
- managing neighbor relationships

4.2 NetFP Proxy Module Architecture

NetFP proxy architecture is shown in the figure below:

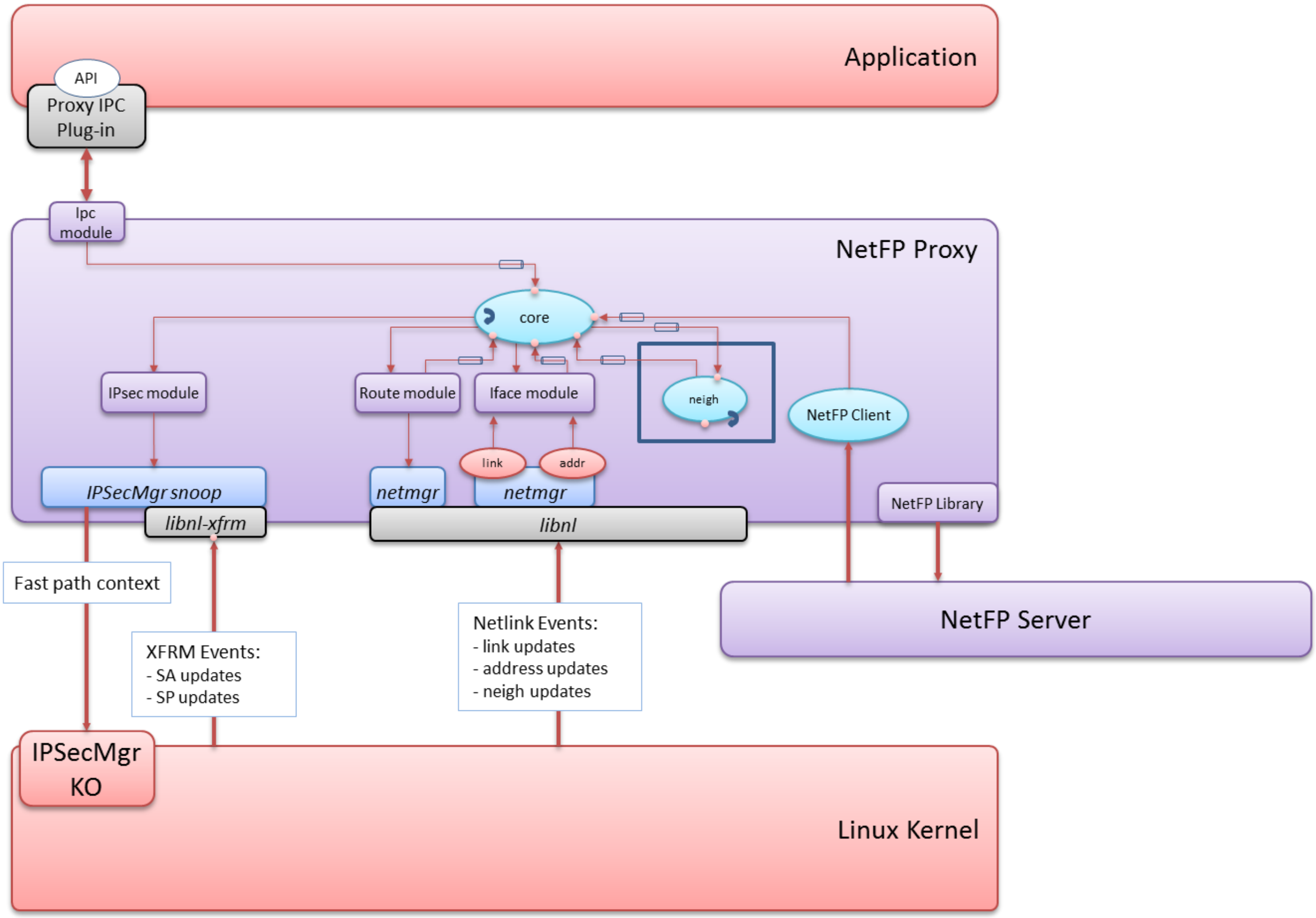


Figure 4-2. NetFP Proxy System Architecture

NetFP proxy consists of following components:

- ***Core module***. This is a proxy main thread which is blocked awaiting for the events coming from multiple modules
- ***Ipc module*** handles the interface with user Application
- ***Route module*** handles egress Fast Path routing requests by obtaining routing information from Linux routing system

- ***Neigh module*** provides address resolution for the egress Fast Paths. Monitors the changes, and updates the core module. Module runs a separate thread to monitor the ARP cache changes (v1), or raw sockets (v2)
- ***Iface module*** handles NetFP Proxy interfaces. Monitors for changes, and updates the core module
- ***IPsec module*** is used to interface Linux Kernel security policies and security association database. It uses ipsecmgr_snoop library to as wrapper, which uses libnl-xfrm library to maintain updated cache of policies and associations. In addition, ipsecmgr_snoop interfaces kernel module IPSecMgr, providing kernel with offloaded security context.
- ***NetFP client*** instance. This is a thread that handles receive interface with NetFP Server

Each module is explained in detail in the following sections.

4.3 Core module

Core module is main NetFP proxy module, which includes netfp proxy initialization functionality, main processing thread, and core module message processing.
Main thread creates a set of fds and blocks on selecting multiple events. Once any asynchronous event is detected, appropriate module's execute function is called.
Events from different sources will awake the main thread:

- Events triggered by Application command delivered via the plug-in
- Events triggered NetFP server, delivered via a pipe established between NetFP Client thread and Core thread
- Link and address change events subscribed via netmgr instance using libnl library
- Neighbor discovery events coming from the neigh module

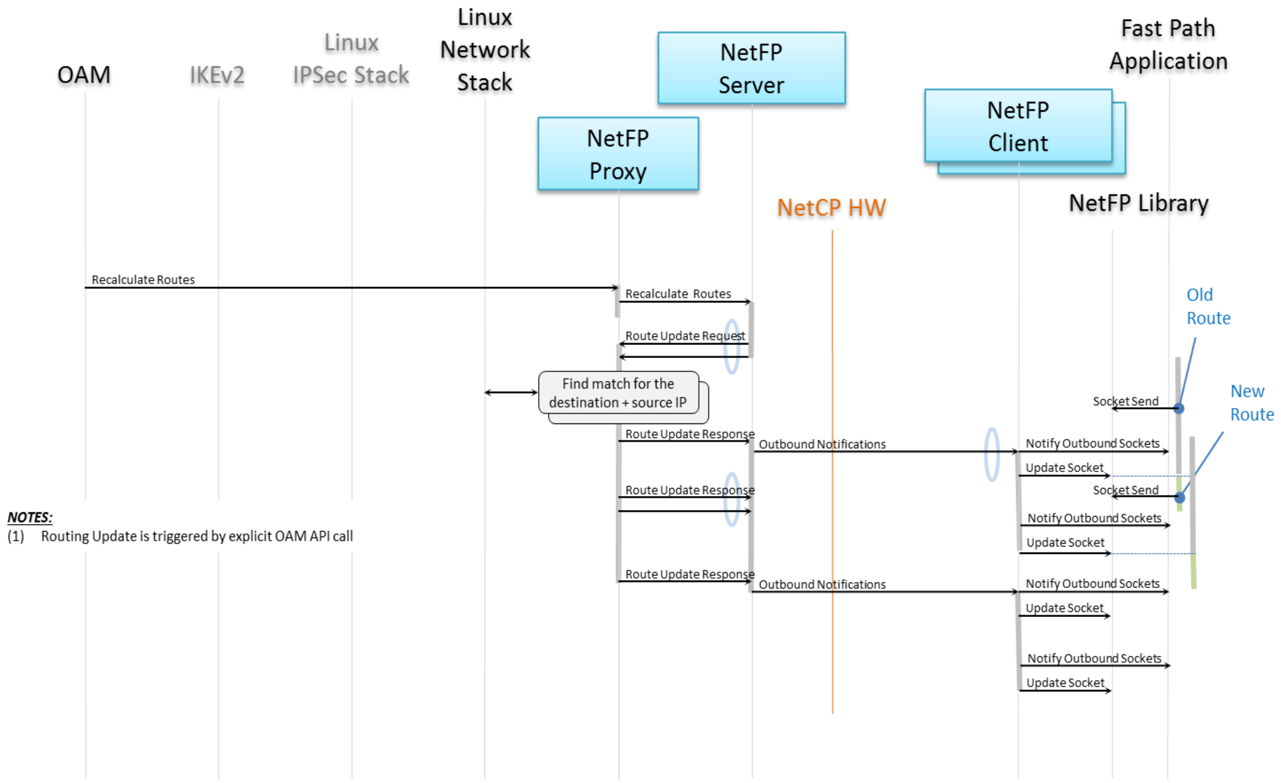
In addition, main thread periodically polls the IPSec module.

4.4 Route module

Route module executes in the context of the core thread. It performs route lookup and returns the next hop IP address based on IP destination/source. It maintains cached version of Linux Routing tables. To support variety of routing modes (multiple routing tables, source based routing), the design decision was made that:

a) routing updates MUST be explicitly triggered by an Application
b) once triggered, NetFP Proxy under NetFP server control will re-evaluate ALL routes that are used by Fast Path
c) resulted changes in interfaces/next hop MAC address are propagated to NetFP clients

Simplified route update procedure is illustrated below



NOTES:
(1) Routing Update is triggered by explicit OAM API call

Figure 4-3. Route Resolution Procedure

4.5 Negh module

Neigh module is responsible for next hop MAC address resolution of the routes managed by NetFP proxy. The module is managing its own neighbor database independently from Linux. It uses raw sockets to perform neighbor discovery and listens for gratuitous updates. Neigh module instantiate a separate thread that is blocked on messages from the core thread (neighbor resolution request) and raw socket receive (neighbor responses/gratuities ARPs).

4.6 Interface module

Interface module maintains cached database of the interfaces and addresses used by NetFP proxy. The interface to be used by the Fast Path is decided based on routing resolution decision.
Changes in interface/addresses are delivered via NetFp server event system to concerned NetFP clients.

4.7 IPsec module

IPSec module implements the APIs to offload security policies to the fast path. It initializes the snooper, which runs a state machine for each instance of the offloaded security policy. Snooper listens to potential updates in offloaded security policies or security associations via libnl/xfrm events. In addition IPSec module periodically collects packet/bytes counts for the traffic processed via offloaded tunnels from the NetFP server and uses snooper to update IPSec kernel module, which maintains life time packet/bytes limits.

4.8 NetFP client instance

NetFP proxy instantiates NetFP client thread to listen to the messages from the NetFP Server. Client wakes up on such event, terminates a Josh job and queues a message to a NetFP core thread.

4.9 Thread model

NetFP proxy uses three threads:

- Main Core thread,
- Neighbore module thread, and
- NetFP client thread

Picture below illustrates the proxy thread model showing simple case route resolution handling by netfp proxy.

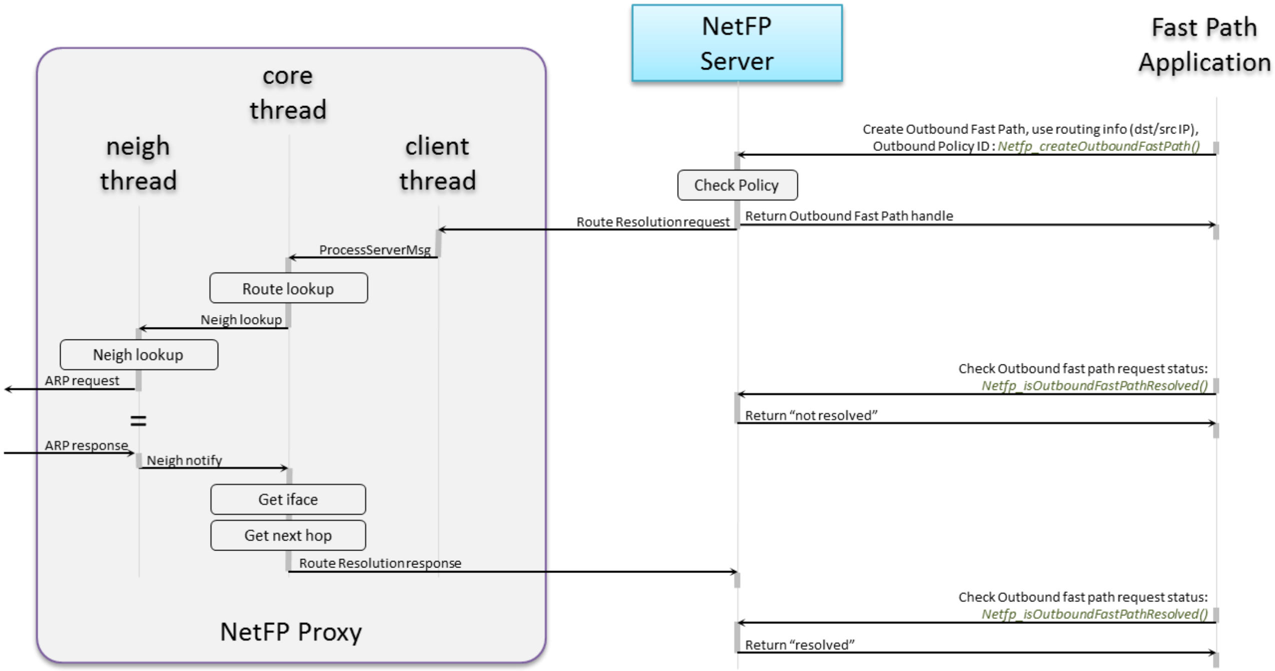

NetFP Server
Fast Path Application
core thread
neigh thread
client thread
Create Outbound Fast Path, use routing info (dst/src IP), Outbound Policy ID : Netfp_createOutboundFastPath()
Check Policy
Route Resolution request
Return Outbound Fast Path handle
ProcessServerMsg
Route lookup
Neigh lookup
Neigh lookup
ARP request
=
ARP response
Neigh notify
Check Outbound fast path request status: Netfp_isOutboundFastPathResolved()
Return "not resolved"
Get iface
Get next hop
Route Resolution response
Check Outbound fast path request status: Netfp_isOutboundFastPathResolved()
Return "resolved"
NetFP Proxy

Figure 4-4. Route resolution handling inside NetFP Proxy