NETRONOME

WHITE PAPER

Programming Netronome Agilio® SmartNICs

NFP-4000 AND NFP-6000 FAMILY: SUPPORTED PROGRAMMING MODELS

THE AGILIO SMARTNICS DELIVER HIGH-PERFORMANCE SERVER-BASED NETWORKING APPLICATIONS SUCH AS NETWORK VIRTUALIZATION, SECURITY, LOAD BALANCING, QUALITY OF SERVICE, AND TELEMETRY.

CONTENTS

INTRODUCTION TO AGILIO SMARTNICS AND NETWORK FLOW PROCESSORS (NFP)

The Agilio SmartNICs deliver high-performance server-based networking applications such as network virtualization, security, load balancing, quality of service, and telemetry. The Netronome Network Flow Processors (NFP-4000 and NFP-6000 family of devices) are used in the Agilio SmartNICs. Server-based networking deployments have become mainstream in COTS servers, and this includes implementations where networking functions are implemented inline between the network port on PCIe server adapters and host applications or virtual machines (VM) and containers implemented in servers. It also includes networking functions implemented in VMs (as in virtual network functions or VNFs).

The NFP-4000 and NFP-6000 family of devices (collectively called the NFPs in the rest of the document) are programmable flow processors that can perform a range of packet processing operations for different applications.

RELATED DOCUMENTS

DESCRIPTIVE NAME	DESCRIPTION
Netronome Agilio Software version 2.0 Getting Started Guide	A guide to new users of Netronome's Agilio Software for server-based networking applications
Agilio Software v2.0 Programmer's Reference Manual (PRM)	Describes the list of APIs supported by the Agilio Software

DESCRIPTIVE NAME	DESCRIPTION
Netronome Agilio Software version 2.0 Product Brief	Provides list of features and benefits offered by the Agilio Software for server-based networking functions such as network virtualization, security, load balancing and statistics/metering
Netronome Network Flow Processor 6000 Family: Databook	Contains detailed reference information on the Netronome Network Flow Processor NFP-6000 family
Netronome Network Flow Processor 4000 Family: Databook	Contains detailed reference information on the Netronome Network Flow Processor NFP-4000 family
Netronome Network Flow Processor Development Tools User's Guide	Describes the Programmer Studio and the development tools that can be accessed through the Programmer Studio
Netronome Agilio CX and Agilio LX SmartNIC Product Briefs	Provides list of features and benefits offered by these PCIe Gen3 server adapters suitable for use in x86 COTS servers

NETWORK FLOW PROCESSOR (NFP) PROGRAMMING BLOCKS

As shown in figure below, the Network Flow Processor (NFP) has the following internal blocks that allow for networking datapath configuration and programmability. The NFP processing elements are distributed across the chip in the form of "islands" with a high-speed CPP (command push pull) bus connecting the islands for transfer of data between the islands. The number of islands on a NFP depends on the SKU of the chip (NFP-4000 or NFP-6000). The NFP has the following main components:

THE NFP PROCESSING ELEMENTS ARE DISTRIBUTED ACROSS THE CHIP IN THE FORM OF "ISLANDS" WITH A HIGH-SPEED CPP (COMMAND PUSH PULL) BUS CONNECTING THE ISLANDS FOR TRANSFER OF DATA BETWEEN THE ISLANDS.

1. MAC island
2. Ingress and Egress processing islands (NBI)
3. FPC islands with 12 FPCs (Flow processing cores)
4. PCI, Arm, Crypto and Interlaken islands with 4 FPCs
5. Memory Units

All the islands with the FPCs are programmable using one or more of the following programming languages P4, C, or Microcode as explained in the future sections.

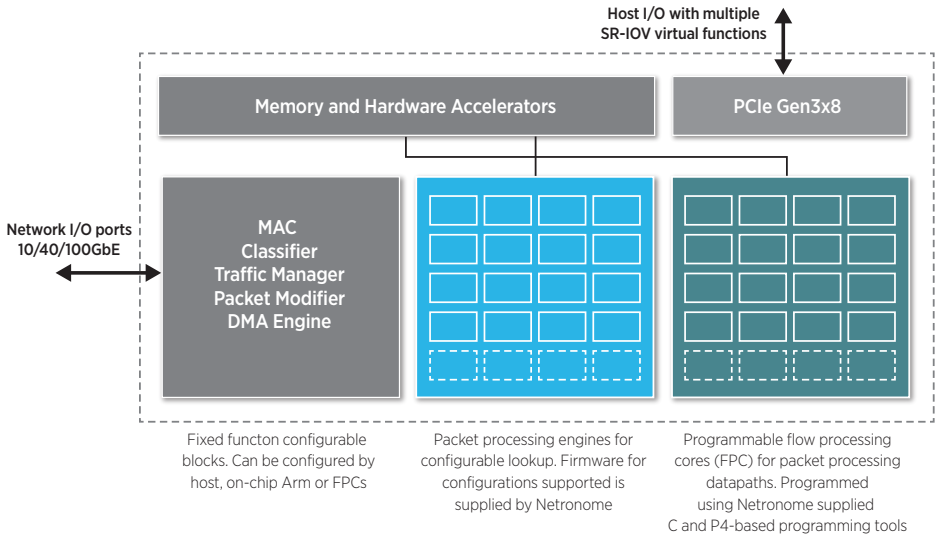

Figure 1. NFP Programming Architecture

Fixed Function Hardware Blocks

The following are fixed function hardware blocks in the NFP:

- I/O Interface blocks (10G/40G/100G MACs and 4xPCI Gen3x8)
- Packet Classifier
- DMAs between internal blocks
- Traffic Manager
- Packet Modifier
- Packet sequencer

The above blocks can be configured through the control and access registers accessible through the following:

- Host CPU via the PCI bus
- On Chip Arm (control processor)
- Flow processing cores (via the on chip configuration bus)

The NFP also has configurable packet classifier blocks referred as packet processing engines in above figure, these blocks are state machine based configurable engines, which are capable of lookup-based L2 and L3 classification of packets. Netronome provides firmware in binary form that configures these packet processing engines.

Programmable Flow Processing cores (FPCs)

The flow processing cores or FPCs constitute the main programmable blocks of the NFP. The FPCs can be used for packet classification and packet modification operations that can go well beyond 5-tuple classification.

Depending on the NFP SKU selected, there can be up to 120 of these FPCs in the device. FPCs are distributed across the Islands within the NFP. Each FPC is multi-threaded and has its own instruction memory, data memory and registers for program execution. FPCs also have access to a data bus called command-push-pull (CPP) bus and configuration bus called eXpansion Bus (XPB) for transferring data and accessing the control and status registers respectively.

FPC programming is also assisted by:

- Hierarchical memory structure (up to 30MB within the NFP and up to 24GB connected as off chip) accessed through the CPP commands.
- Hardware accelerators such as lookup engine, statistics engine, crypto engine, packet engine etc. (accessible via the commands from FPCs).

The FPCs can be programmed using high level languages such as P4 or C with the Netronome provided SDK tool chain. P4 programs are compiled using the open source P4 compiler from the P4 consortium. P4 complier is integrated with the Netronome C compiler to provide an integrated development environment where both P4 and C-based applications can be applied to the data-path. The SDK tool chain also provides the capability to simulate and debug the programs on the NFP software simulator and NFP based hardware targets such as the Agilio SmartNICs.

Detailed description of each of the above blocks is beyond the scope of this document and can be found in the NFP 4000/6000 Family Data Books.

NFP PROGRAMMING MODELS

Both high level and low level programming models are supported. For example P4 and API based programming are high level and when such models are used, the developer need not be aware of internal architectural blocks of the NFP such as the CPP bus, MAC and packet classifier. Other models support addition of C or P4-based applications as a sandbox or plug-in to the Netronome provided Agilio Software datapath.

NFP-based Agilio SmartNICs support the following programming models:

1. Host API-based Programming Model: Using Agilio Software supported APIs
2. User Datapath Programming Model: C-based programming with configuration APIs
3. User Datapath Programming Model: P4 and C-based programming with configuration APIs
4. User Datapath Programming Model: Programming a C (or P4) sandbox or plug-in application into the Agilio Software datapath

Each of the programming models above is described in detail in the next sections.

Host API-based Programming Model

This high level programming model is supported with Netronome Agilio SmartNICs. In this model, the SmartNIC is supplied with the production quality Agilio Software in binary form. The features supported includes:

- PCIe and Network I/O configuration (includes the host interface drivers)
- Standard datapath features (via Open vSwitch (OVS) offload, tunneling, match-action etc.) usable via supported API calls

The Agilio Software v2.0 that is currently available supports the OVS v2.3 and v2.4 datapaths – for further details please see the appropriate Agilio Software documentation. The host API programming model works with the Agilio Software datapath.

Netronome has developed the Agilio Software provided with the Agilio SmartNICs using the NFP internal blocks and hierarchical memory most efficiently. The users of the Agilio Software are expected to just call the Agilio Software APIs to meet the requirements of their use cases.

The Agilio Software datapath supports match-action processing along with support for a unique hardware-based flow cache implementation for fast path processing. The classification of the ingress packets is performed using the match fields as configured by the user and action is taken on individual packets based on the match entry. The Agilio Software is based on the OVS datapath that is offloaded from the kernel to the NFP, along with the additional features such as auto learning flow-cache, load balancer and tunneling. Additional features are planned for the Agilio Software with future releases.

The features supported by the Agilio Software can be used through the Agilio API calls described in the Agilio Software Programmer's Reference Manual (PRM). The user is not required to learn about the internal architecture of the NFP.

Figure 2 below explains the high level architecture of the Agilio Software. The Agilio Software blocks shown in the figure are implemented using the three NFP hardware blocks described above. Any configuration and FPC programming details are kept hidden from user. The kernel mode portion of the OVS software (repeated hash tables and OVS kernel actions) is replicated and offloaded in the NFP.

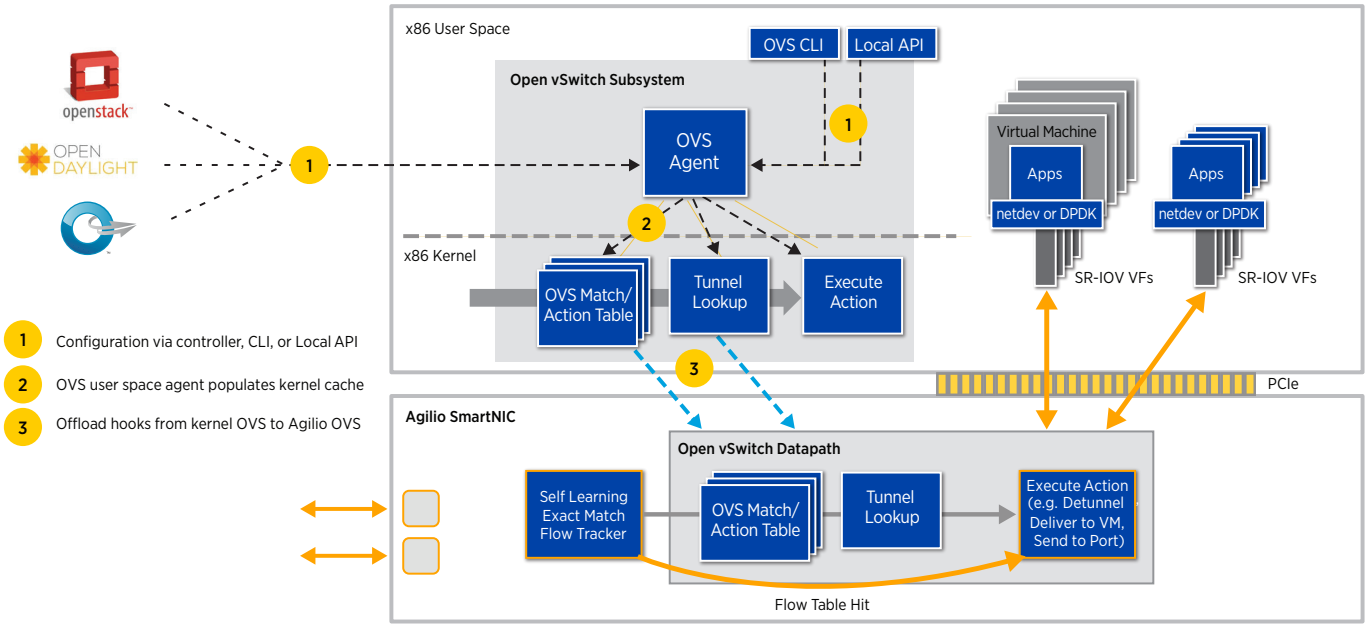

Figure 2. OVS offload and tunneling implemented in NFP programming blocks

The Agilio Software features are available to the user through API calls. These APIs can be called via the command line or integrated into the application software to configure the datapath in the Agilio SmartNIC.

The following set of API calls are available for use with the Agilio Software:

1. **Local Flow Table APIs** - Used to manipulate the Agilio Software flow tables. They are compatible with OpenFlow v1.3 specifications. The Agilio Software is provided with sample applications that demonstrate the use of these APIs.
2. **Local Packet APIs** – Allows the users to interact with the network packets on the host installed with Agilio SmartNIC and Software. Sample applications that demonstrate the use of these APIs are included in the Agilio Software package.
3. **Group APIs** – These APIs allow the users to manipulate (add, modify, delete) group entries in Agilio Software-supported group tables.
4. **Meter APIs** – Allows the user to manipulate (add, modify, delete) meter entries in the Agilio Software-supported meter tables.
5. **Health monitoring APIs** – Allows the user to monitor the system health on the host OS. Users can create their own health monitoring applications. Sample applications that demonstrate the use of these APIs are included with the Agilio Software.

Please refer to the Agilio Software Programmer's Reference Manual (PRM) for details on how to use the above API calls.

Using the above API calls the following example applications can be configured on the NFP-based Agilio SmartNIC:

- Traffic engineering and network virtualization
- Learning L2 bridge
- Layer-3 routing functionality
- Load balancing to physical and virtual ports.
- Wire – fast path configuration

- Adding and removing packet tags
- Running any host DPDK application with NFP virtual functions.
- Accelerated origination and termination of VXLAN and NVGRE tunnels on NFP

The above list just shows some example applications and does not cover the complete functionality of the Agilio Software. Please refer to complete documentation supplied with the Agilio Software for additional details.

User Datapath Programming Models

This model is targeted for users who want to program the datapath on the Agilio SmartNIC and NFP. Both simple and complex datapaths can be programmed utilizing the three methods described in this section. In general, while selecting which model is most suitable, the user should consider the characteristic of the desired datapath:

- Datapath is based on a free form pipeline where a developer can write any packet-processing pipeline.
- Datapath is based on the match-action paradigm, which is similar to the Open vSwitch (OVS) pipeline supported by the Agilio Software as described in above section. In this model, the match-action datapath for example, may be fixed by a P4 program while the custom actions are defined by a C program.

C-Based Programming with Configuration APIs

This model allows for the complete control of the programmable datapath using the FPCs. The user utilizes software drivers and configuration APIs that provide the needed functionality of the fixed function configurable blocks, configurable classifier and part of the programmable FPCs.

These API sets include the following:

- PCIe driver for the host – C code for receiving and sending packets from and to host respectively
- GRO – Global Re-Ordering software in C
- Sample NIC send/receive applications C
- 10GbE/40GbE network ports initialization APIs
- Hardware Classifier initialization APIs
- Hardware Traffic Manager configuration APIs
- Look up based pre classification firmware
- Buffer list Manager Code – For ingress and egress packet buffering and DMA to the Flow Processing Cores (FPCs)

Using the above APIs and source code written in C supported by the Netronome SDK tool chain, users can modify, add or write their own datapath application.

One example of this model is shown in Figure 3. In this model, users write the C code for the packet classification and actions functions in the datapath and compile using the Netronome SDN tool chain.

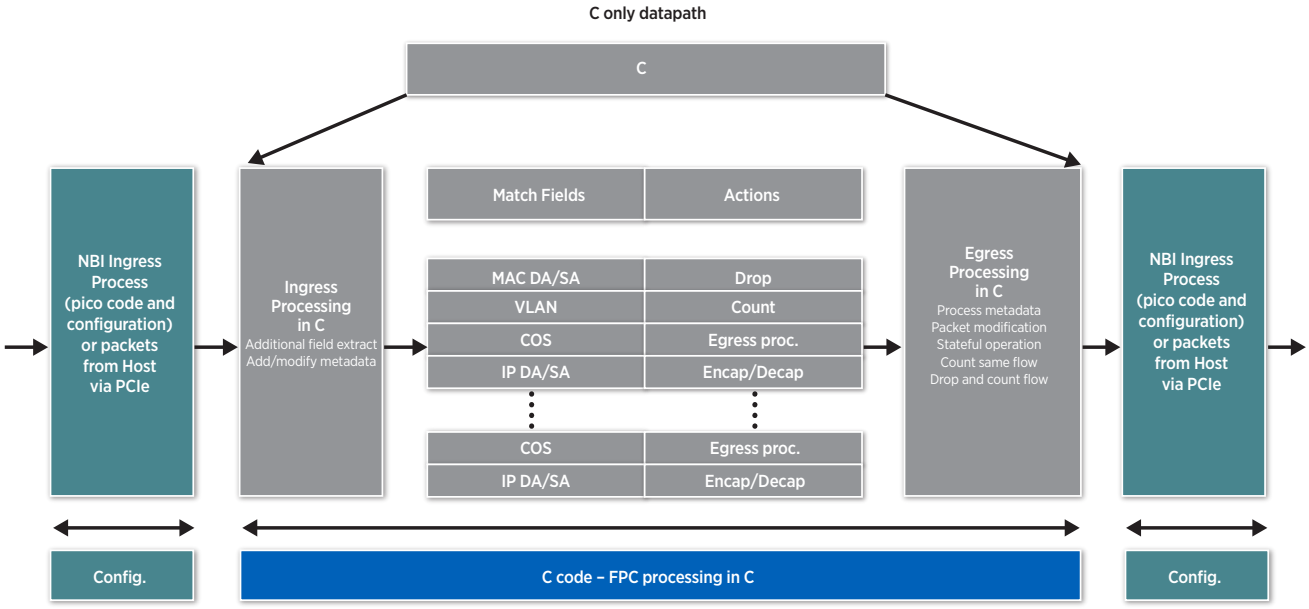

Figure 3. Example of NFP datapath written in C

This programming model is facilitated by sample C libraries and applications provided by Netronome. Figure 4 provides the list of programming resources provided for this model.

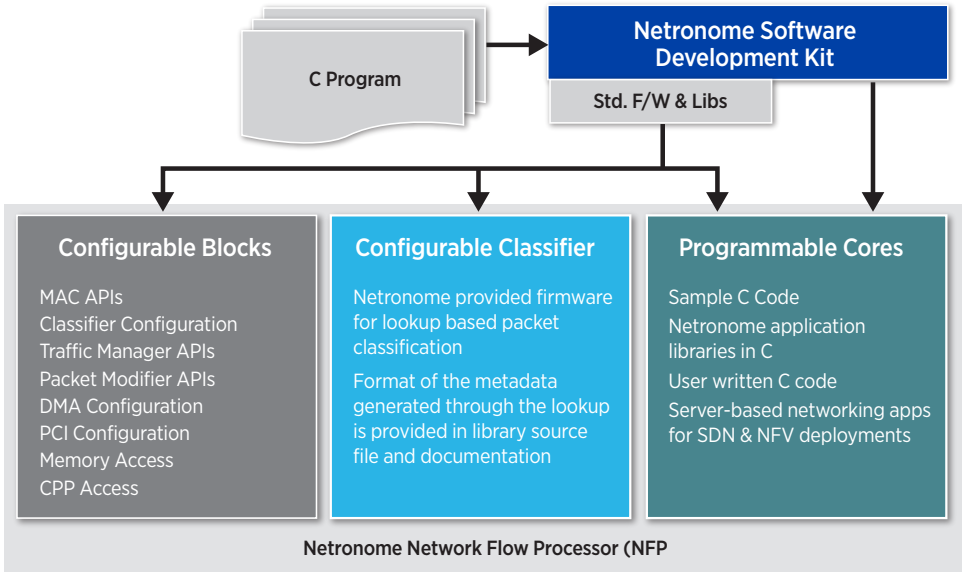

Figure 4. Resources for NFP Programming Model using C

Netronome C Libraries

This C-based Programming Model also comes with Netronome-provided C libraries for basic packet operations and low-level NFP access functions. These libraries are pre-verified and include the detailed documentation describing their functionality.

The following are some standard C libraries provided as sample code:

- Memory Lookups
  - Hash lookup
  - Index lookup
  - Lookup and add
- Header Parsing
  - Ethernet
  - ARP/IPv4/IPv6
  - TCP/UDP
  - GRE/NVGRE/VXLAN
- Common NFP Access and Engines
  - PCIe Read/Write/DMA
  - Memory Ring operation (push/pop)
  - TCAM lookups
  - CRC computations
  - Register configuration
- Packet Operations
  - Receiving/sending packet from/to Network ports
  - Reading number of bytes from memory
  - L3/L4 checksum calculation
  - Packet modification script
  - DMA packet between memories
  - Drop a packet and free buffer

P4 and C-based Programming with Configuration APIs

This NFP programming model is meant for users who want to program the datapath in a hardware-agnostic way. In this model, users do not need to know the details of the underlying NFP architecture.

P4 is target independent network programming language where users can write the forwarding behavior of the network devices (ASIC/NPUs/FPGAs) using the standard forwarding model defined in the P4 architecture. P4 allows the user to create their own packet headers and protocols along with their processing behavior in a networking device.

The packet-processing model proposed by the P4 language is shown in Figure 5. User writes the datapath of a network device in P4 language without any knowledge of the target hardware. The P4 tool chain developed by the device vendor converts the P4 program into the device specific firmware. P4 tool chain also generates a run time API (similar to open flow model) to allow the match-action table modification.

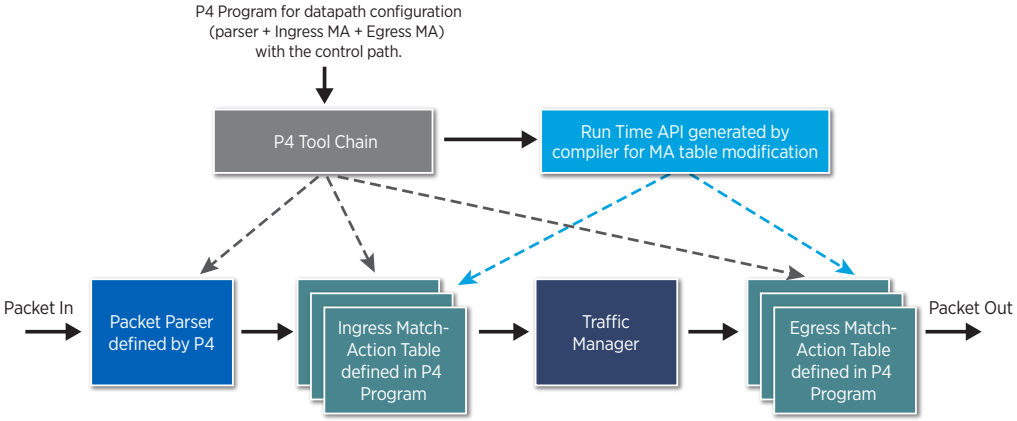

Figure 5. Packet processing model using the P4 language

While the P4 language enables hardware agnostic programming of the network device, there may still be a need for device specific customizations. Some examples of such custom features include:

- Stateful packet filtering
- Stateful statistics collection (based on the flows)
- Sending messages to Host (Control and Data)
- Hash Table modifications
- Atomic operations
- QoS implementation (Traffic Manager and buffering)

To enable such extensions to P4-based programmability, Netronome provides the ability to extend P4 datapath features with C-based custom applications. This is also referred to as application of C-based sandbox or plugins to a P4-defined datapath.

For example, the following suggested steps may be taken to implement this programming model:

- Use of Netronome provided API calls for PCIe and Network I/O configuration functions
- Use of Netronome provided classification firmware
- Use of sample programs that demonstrate the use of P4 and C datapath programming using this model
- Review of provided C libraries that can be used as a Sandbox or plugin application
- Use the SDK tool chain to compile the P4 and C programs to generate and install the needed datapath on the NFP

Figure 6 below shows an example of this programming model as implemented on the NFP.

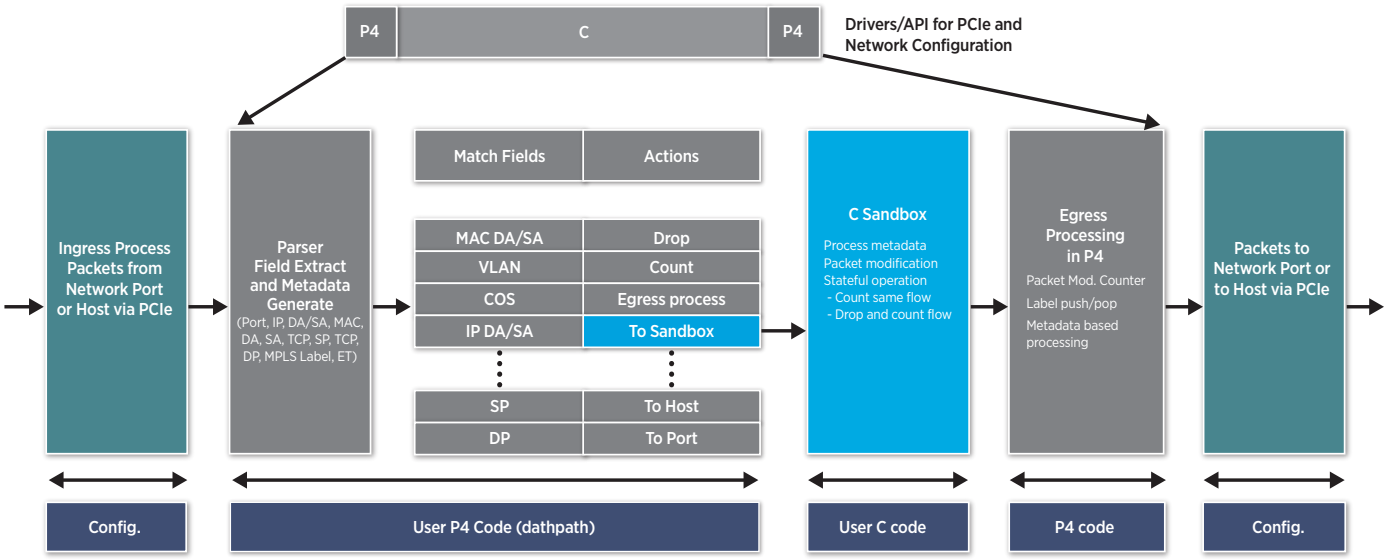

Figure 6. P4 datapath with C sandbox implemented on NFP

The implementation described above requires the Netronome SDK tool chain with the P4 programming support. The details of the SDK tool chain and features supported are described in a future section.

Programming a C (or P4) Sandbox App into the Agilio Software Datapath

This NFP programming model is still under planning however some initial architecture and high-level plans are discussed in this section Figure 7 represents the proposed C or P4 Sandbox application implementation along with the main datapath defined and supported by the Agilio Software.

Since Netronome has implemented Agilio Software by taking advantage of all the NFP hardware resources and accelerators, this method potentially represents one of the most feature-rich, performance and resource-optimized implementation of packet processing datapath on the NFP, which includes the applications that require network-to-host, host-to-network and network-to-network ingress and egress packet flows.

This programming model is offered with the following components:

- Agilio Software in binary form
- C Sandbox libraries
- SDK tool chain that supported integrated C and P4-based programming
- Sample sandbox programs in P4 and C

The sandbox (or plugin) application shown in Figure 7 can be implemented either in C or P4. Also the sandbox functionality can be replicated in the user/kernel space to allow for the fallback path of the sandbox implementation.

In the sandbox implementation example shown, an OVS match-action table implements one of the actions as “send-to-sandbox,” where the “send-to-sandbox” action can be one of the logical ports directing to the C sandbox code.

This sandbox can be implemented as

- Custom/Primitive action in the form of a plugin stateful function running on the same FPC.
- Extended match action functionality implemented in P4/C (as described in an earlier section) running on separate set of FPCs. In this case the packets from Agilio Software to the sandbox are transferred via the ring based transfer mechanism between the FPCs.

The proposed implementation of the P4/C sandbox is not designed to feed every packet into the sandbox code, as by doing so there may be performance implications.

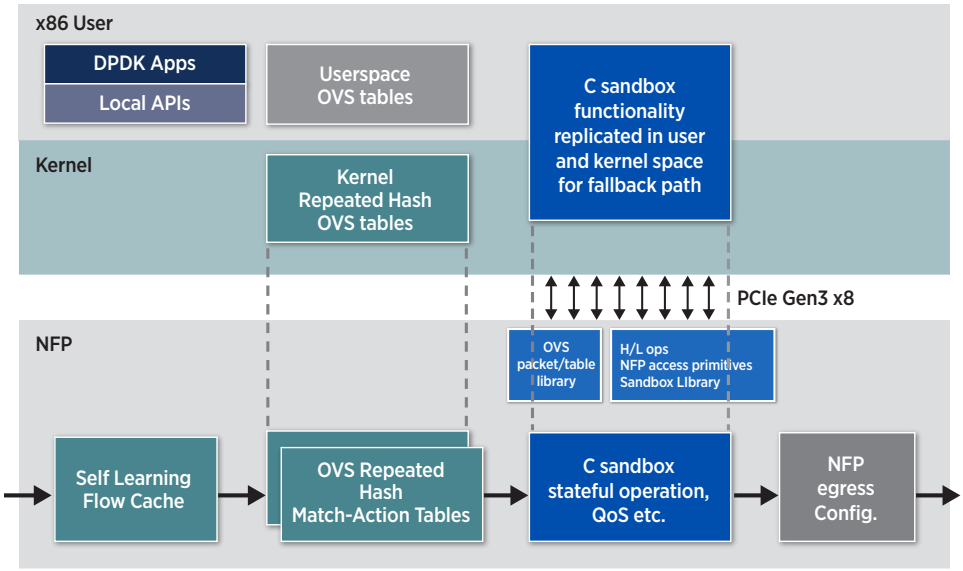

Figure 7. Agilio Software with the C app sandbox implementation on the NFP

The NFP datapath associated with the Agilio Software based sandbox model is shown in Figure 8. The details of this programming model such as sandbox functionality actions, number of FPCs available for the sandbox functionality, libraries etc. is still in the planning and development stage. More details will be available in future version of this and other related documents.

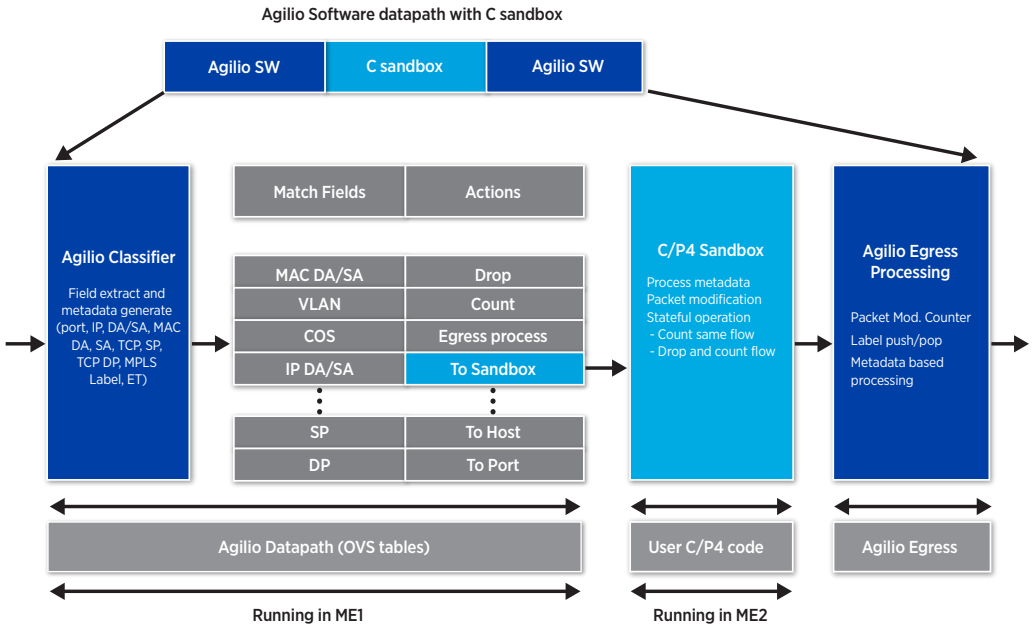

Figure 8. NFP datapath with Agilio Software and C/P4 sandbox

SDK TOOL CHAIN FOR NFP PROGRAMMING

This section provides a brief introduction to the SDK tool chain for programming the NFP device in Agilio SmartNICs. The SDK tool chain can be used to exercise the programming options described in user datapath programming models above.

Netronome's SDK tools can run on Windows and Linux platforms. The Windows version of the tool chain comes as an integrated development environment (IDE) that combines both C and P4 programming, and allows the full visibility of the chip features with a graphical user interface (GUI) that includes a simulator and a hardware debugger. The Linux version of the tool chain provides a similar set of features and can be exercised through the Linux command line interface (CLI).

Figure 9 below shows all the components of the SDK tool chain available in the Windows and Linux environments.

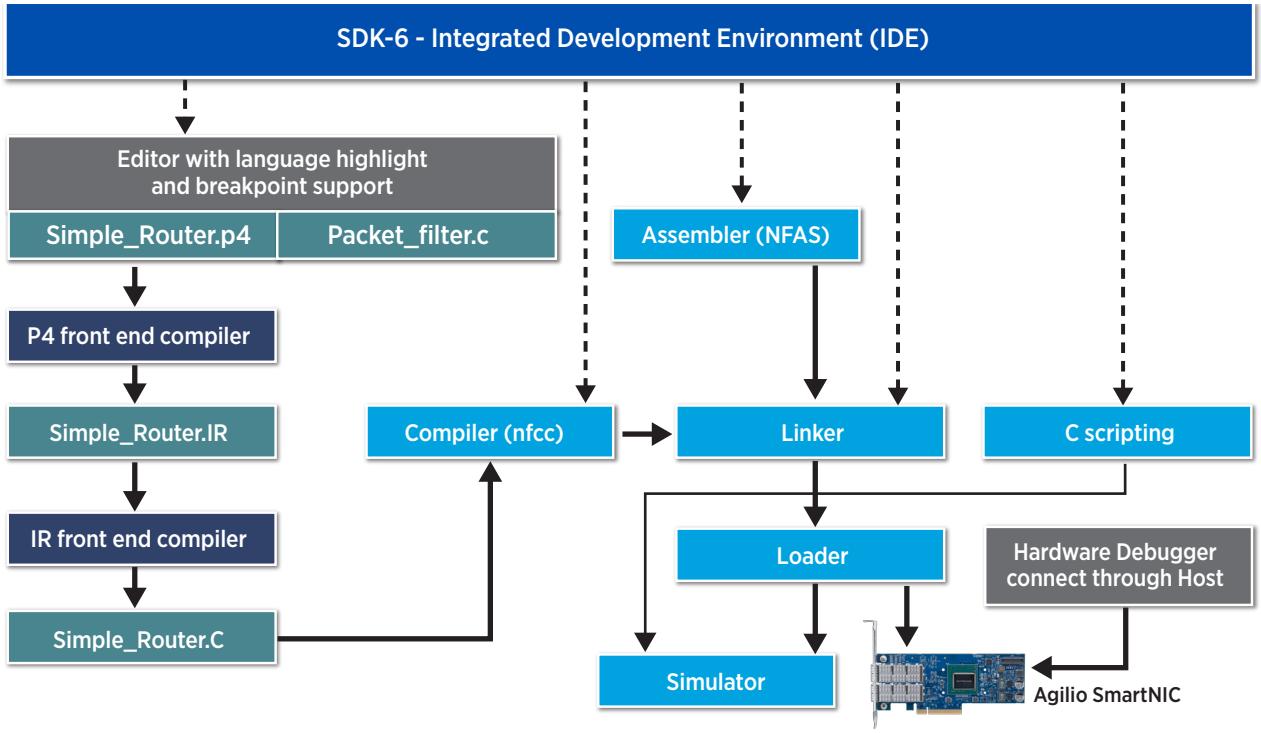

Figure 9. SDK tool chain components for NFP programming

The following is a summary of code development tools and features included in the SDK tool chain:

- Integrated development environment (IDE) in windows
- P4 front end compiler (see below for further information on P4 support)
- IR (intermediate representation) back end compiler
- Netronome C compiler
- C scripting for the NFP configuration
- Netronome microcode assembler (for any legacy microcode/assembly based applications and libraries)
- Linker (links the compiler code to generate the NFP firmware)
- Loader (loads the NFFW files to the NFP)

The following is a summary of profiler and debugger features included in the SDK tool chain:

- Cycle accurate event and queue history
- Cycle accurate history collection for FPC threads
- Performance profiling and bus bandwidth estimates
- Hardware debugger which runs on the Host and communicates to NFP via the PCIe bus to NFP to allow runtime debugging of Netronome SmartNICs

Detailed description of each of the SDK components mentioned above is available in the documentation supplied with the SDK package.

Netronome supports the P4 programming language as defined in the P4 Consortium (www.p4.org). Netronome has integrated the open source P4 compiler developed by the P4 Consortium to generate an intermediate representation (App.IR) of the P4 program in the yaml format, which is further compiled by the Netronome's back end compiler to generate target specific C implementation of the NFP datapath.

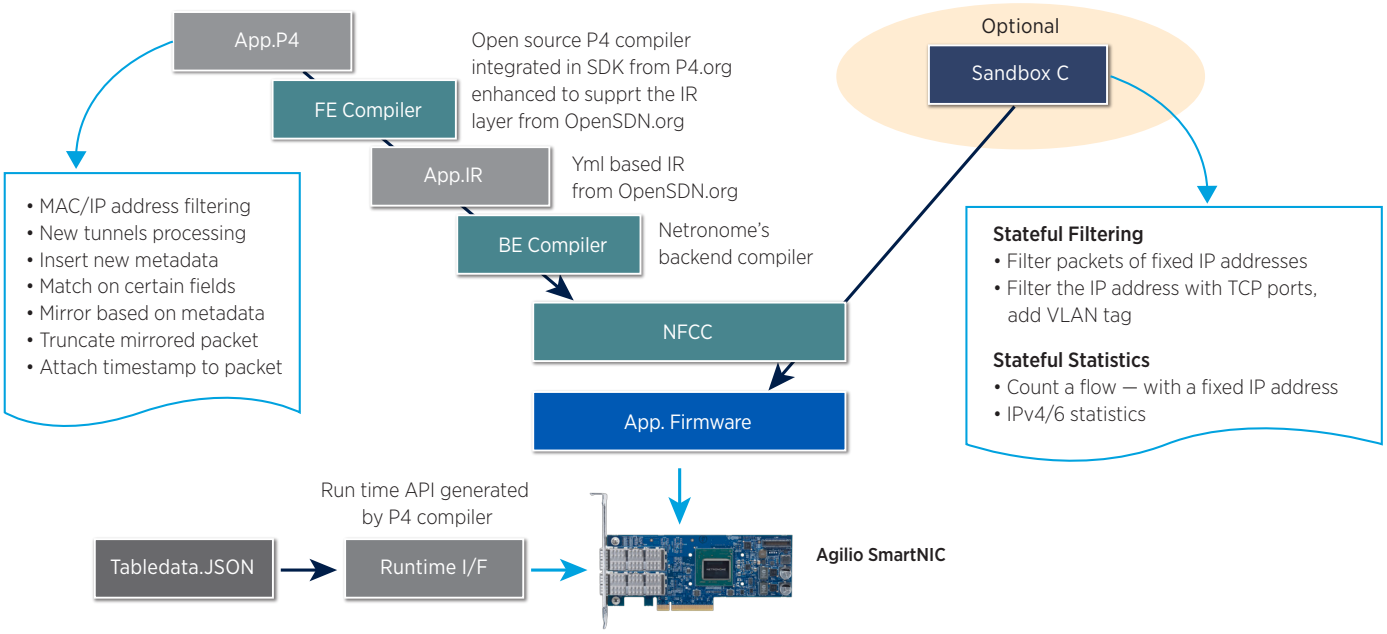

Figure 10. Implementation of P4 datapath with C sandbox on NFP using SDK

The back end compiler generated C files (from P4 code) are compiled together with the custom (sandbox) C files to generate the firmware to be downloaded on the NFP.

Table entries compiled in the JSON format are programmed into the NFP in the Agilio SmartNIC using run time APIs. These APIs support functions such as addition, modification or deletion of the NFP datapath match-action table entries.

CONCLUSION

Netronome's Agilio SmartNICs with Agilio Software provides high-performance networking for modern data center servers. The Agilio solution accelerates server-based networking functions such as network virtualization, security, load balancing and telemetry. The Agilio solution is built on the Netronome NFP which is a programmable device optimized for network datapath processing. The NFP along with the SDK tool chain supports a variety of datapath programming models where users can select one of the models based on their requirements.

The Host API model utilizes the Netronome-supplied production quality Agilio Software that implements standard datapaths such as in OVS. This model is suitable for users who want to use available features in the Agilio software datapath, and are not interested in any custom programming of the datapath or understanding architectural details of the NFP.

The C-based programming model can be used to program a new datapath. An example of this can be development of complete packet classification and processing pipeline for server-based networking applications such as telemetry or load balancing.

The P4 only or P4 with C sandbox model of programming is suitable for the users who want to program the NFP hardware but is not interested in learning the architectural details of the NFP device. The P4 language can be used to implement a hardware-agnostic match-action processing datapath. C-based sandbox applications may be added to such a pipeline for special features such as stateful filtering. This requires some familiarity with the Netronome NFP data structures and knowledge of the SDK tool chain.

The Agilio Software-based Host API model combined with the C sandbox model holds the promise of delivering high performance as well as faster go to market capability while allowing for datapath customizations, provided the features supported by the Agilio Software meets most of the users' needs.

The above programming models are enabled by Netronome's SDK tool chain described above. Netronome's SDK v6.0 will come with an integrated development environment that supports both P4 and C software development and debugging environment.

NETRONOME

Netronome Systems, Inc.
2903 Bunker Hill Lane, Suite 150 Santa Clara, CA 95054
Tel: 408.496.0022 | Fax: 408.586.0002
www.netronome.com