Internet Draft
Expiration: April 2002
File: draft-anderson-forces-model-00.txt
Working Group: ForCES

T. Anderson
Intel Labs
November 2001

ForCES Architectural Framework and FE Functional Model

draft-anderson-forces-framework-00.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-2119].

1. Abstract

This document defines an architecture for ForCES network elements and a functional model for ForCES forwarding elements. This model is used to describe the capabilities of ForCES forwarding elements within the context of the ForCES protocol. The architecture and forwarding element model defined herein is intended to satisfy the requirements specified in the ForCES requirements draft [FORCES-REQ].

2. Definitions

Most of these definitions are copied from the ForCES requirements document [FORCES-REQ].

Addressable Entity (AE) - A physical device that is directly addressable given some interconnect technology. For example, on Ethernet, an AE is a device to which we can communicate using an Ethernet MAC address; on IP networks, it is a device to which we can communicate using an IP address; and on a switch fabric, it is a device to which we can communicate using a switch fabric port number.

Physical Forwarding Element (PFE) - An AE that includes hardware used to provide per-packet processing and handling. This hardware may consist of (but is not limited to) network processors, ASIC's, or general-purpose processors. For example, line cards in a forwarding backplane are PFEs.

PFE Partition - A logical partition of a PFE consisting of some subset of each of the resources (e.g., ports, memory, forwarding table entries) available on the PFE. This concept is analogous to that of the resources assigned to a virtual router [REQ-PART].

Physical Control Element (PCE) - An AE that includes hardware used to provide control functionality. This hardware typically includes a general-purpose processor.

PCE Partition - A logical partition of a PCE consisting of some subset of each of the resources available on the PCE.

Forwarding Element (FE) - A logical entity that implements the ForCES protocol. FEs use the underlying hardware to provide per-packet processing and handling as directed by a CE via the ForCES protocol. FEs may use the hardware from PFE partitions, whole PFEs, or multiple PFEs.

Proxy FE - A name for a type of FE that cannot directly modify its underlying hardware but instead manipulates that hardware using some intermediate form of communication (e.g., a non-ForCES protocol or DMA). A proxy FE will typically be used in the case where a PFE cannot implement (e.g., due to the lack of a general purpose CPU) the ForCES protocol directly.

Control Element (CE) - A logical entity that implements the ForCES protocol and uses it to instruct one or more FEs as to how they should process packets. CEs handle functionality such as the execution of control and signaling protocols. CEs may use the hardware of PCE partitions or whole PCEs. (The use of multiple PCEs will usually be modeled as separate CEs.)

Pre-association Phase - The period of time during which a FE does not know which CE is to control it and vice versa.

Post-association Phase - The period of time during which a FE does know which CE is to control it and vice versa.

ForCES Protocol - While there may be multiple protocols used within a device supporting ForCES, the term "ForCES protocol" refers only to the ForCES post-association phase protocol (see below).

ForCES Post-Association Phase Protocol - The protocol used for post-association phase communication between CEs and FEs. This protocol does not apply to CE-to-CE communication, FE-to-FE communication, or to communication between FE and CE managers. The ForCES protocol is a master-slave protocol in which FEs are slaves and CEs are masters.

FE Model - A model that describes the logical processing functions of a FE.

FE Manager - A logical entity that operates only in the pre-association phase and is responsible for determining to which CE(s) a FE should communicate. This determination process is called CE discovery and may involve the FE manager learning the capabilities of available CEs. A FE manager may use anything from a static configuration to a pre-association phase protocol (see below) to determine which CE to use. Being a logical entity, a FE manager might be physically combined with any of the other logical entities mentioned in this section.

CE Manager - A logical entity that operates only in the pre-association phase and is responsible for determining to which FE(s) a CE should communicate. This determination process is called FE discovery and may involve the CE manager learning the capabilities of available FEs. A CE manager may use anything from a static configuration to a pre-association phase protocol (see below) to determine which FE to use. Being a logical entity, a CE manager might be physically combined with any of the other logical entities mentioned in this section.

Pre-association Phase Protocol - A protocol between FE managers and CE managers that helps them determine which CEs or FEs to use. A pre-association phase protocol may include a CE and/or FE capability discovery mechanism. It is important to note that this capability discovery process is wholly separate from (and does not replace) that used within the ForCES protocol. However, the two capability discovery mechanisms may utilize the same FE model (see Section 5). Pre-association phase protocols are not discussed further in this document.

ForCES Network Element (NE) - An entity composed of one or more CEs and one or more FEs. To entities outside a NE, the NE represents a

single point of management. Similarly, a NE usually hides its

internal organization from external entities. However, one exception to this rule is that CEs and FEs may be directly managed to transition them from the pre-association phase to the post-association phase.

ForCES Protocol Element - A FE or CE.

High Touch Capability - This term will be used to apply to the capabilities found in some forwarders to take action on the contents or headers of a packet based on content other than what is found in the IP header. Examples of these capabilities include NAT-PT, firewall, and L7 content recognition.

Bootstrap CE - The first CE that a FE connects to in a ForCES NE.

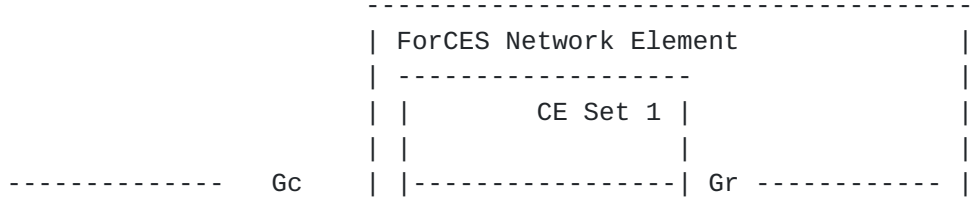
CE set - One or more equivalently capable CEs designed to operate concurrently (for load sharing) or in a 1+N failover mode (for redundancy).

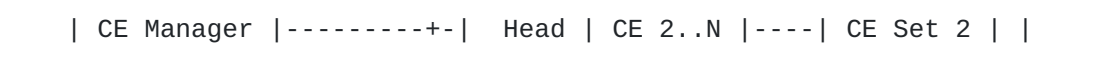
3. Introduction

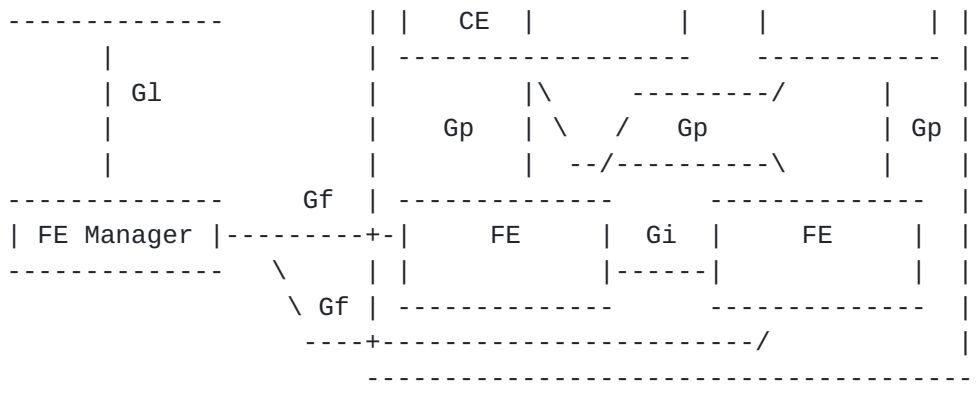
[TBD]

4. Architecture

This section defines a ForCES architectural framework. This ForCES framework consists primarily of ForCES NE's but also includes several ancillary components. ForCES NE's appear to external entities as monolithic pieces of network equipment, e.g., routers, NAT's, firewalls, or load balancers. (See [FORCESREQ], Section 5, Requirement 4.) Internally, however, ForCES NE's are composed of several logical components. By defining logical components and specifying the interactions between them, the ForCES architecture allows these components to be physically separated. This physical separation accrues several benefits to the ForCES architecture. For example, separate components would allow vendors to specialize in one component without having to become experts in all components. Scalability is also provided by this architecture in that additional forwarding or control capacity can be added to existing network elements without the need for forklift upgrades. The components of the ForCES architecture and their relationships are pictured in the following diagram. For convenience, the interactions between components are labeled by reference points Gp, Gc, Gf, Gr, Gl, and Gi.




| CE Manager |---------+-| Head | CE 2..N |----| CE Set 2 | |



4.1. Control Elements

This architecture permits multiple CEs to be present in a network element. These CEs may be used for any combination of redundancy, load sharing, or distributed control. Redundancy is the case where one or more CEs are prepared to take over should an active CE fail. Load sharing is the case where two or more CEs are concurrently active and where any request that can be serviced by one of the CEs can also be serviced by any of the other CEs. In both redundancy and load sharing, the CEs involved are equivalently capable. The only difference between these two cases is in terms of how many active CEs there are. Distributed control is the case where two or more CEs are concurrently active but where certain requests can only be serviced by certain CEs.

To enable multiple CEs, control in a ForCES NE is handled by one or more CE sets. Each CE set can specialize in handling a particular subset of the control functions of a NE. For example, one CE set may handle routing functions while another may handle firewall or QoS functions. Each CE set is itself composed of multiple CEs. All of the CEs in a CE set are equivalently capable, meaning that each is capable of performing the same set of functions albeit with possibly different performance. The remaining members of a CE set may be used for load sharing or redundancy purposes. Communication between members of a CE set or between CE sets is discussed in Section 4.10. CEs are wholly responsible for coordinating amongst themselves to provide redundancy, load sharing, or distributed control, if desired.

CEs are concerned with controlling the layer-3 and above capabilities of FEs. CEs are not concerned with controlling the layer-2 and below communication aspects of the FE.

While the ForCES model allows for multiple CEs, the coordination of those CEs is beyond the current scope of ForCES. In cases where an implementations uses multiple CEs or CE sets, it is still required that an implementation must maintain the invariant that a single NE MUST NOT appear as multiple NEs even in the presence of link failures between FEs and/or CEs.

4.2. Forwarding Elements

FEs are responsible for per-packet processing and handling as directed by its CEs. FEs have no initiative of their own. Instead, FEs are slaves to their CEs and only do as they are told (Section 4.9). FEs may communicate with one or more CEs, either from the same or different CE sets concurrently. However, FEs have no notion of CE redundancy, load sharing, or distributed control. Instead, FEs accept commands from any CE authorized to control them. This architecture mandates that a coarse grain mapping of requests to CE sets be possible but also allows finer grain mappings. For example, at a minimum, a CE must be able to specify a single CE set to which all requests generated by the FE should be sent. However, the architecture also allows different CE sets to be mapped to different types of requests if the FE is capable of differentiating between request types.

This architecture permits multiple FEs to be present in a NE. Each of these FEs may potentially have a different set of capabilities. FEs express these capabilities using the ForCES FE model described in Section 5. FEs are responsible for establishing and maintaining layer-2 connectivity with other FEs or with entities external to the NE. Thus, FEs are also responsible for any signaling required at layer-2.

4.3. CE Managers

CE managers are responsible for determining which FEs a CE should control. It is legitimate for CE managers to be hard-coded with the knowledge of with which FEs its CEs should communicate. Likewise, CE managers can communicate with any other entity or perform any kind of computation to make that determination.

4.4. FE Managers

FE managers are responsible for determining to which CE any particular FE should initially communicate. Like CE managers, no restrictions are placed on how a FE manager decides to which CEs its FEs should communicate. The FE manager can be hard-coded with this information or communicate with any other entity to make that determination.

4.5. Gl Reference Point

CE managers and FE managers may communicate with each other across the Gl reference point in order to help them decide which CEs and FEs should communicate with each other. Communication across the Gl reference point is entirely optional in this architecture. No requirements are placed on this reference point.

CE managers and FE managers may be operated by different entities.

The operator of the CE manager may not want to divulge, except to

specified FE managers, any characteristics of the CEs it manages. Similarly, the operator of the FE manager may not want to divulge FE characteristics, except to authorized entities. As such, CE managers and FE managers may need to authenticate one another. Subsequent communication between CE managers and FE managers may require other security functions such as privacy, non-repudiation, freshness, and integrity.

Once the necessary security functions have been performed, the CE and FE managers MAY communicate to determine which CEs and FEs should communicate with each other. In this process, the CE and FE managers will likely learn of the existence of available FEs and CEs respectively. This process is called discovery and will likely entail one or both managers learning the capabilities of the discovered ForCES protocol elements.

4.6. Gf Reference Point

The Gf reference point is used to inform forwarding elements of the decisions made by FE managers. Only authorized entities may instruct a FE with respect to which CE should control it. Therefore, authentication is a necessary between FE managers and FEs. Privacy, integrity, and freshness are also required. Once the appropriate security has been established, FE managers may instruct FEs across this reference point to join a new NE or to disconnect from an existing NE.

4.7. Gc Reference Point

The Gc reference point is used to inform control elements of the decisions made by CE managers. Only authorized entities may instruct a CE to control certain FEs. Privacy, integrity, and freshness are also required across this reference point. Once appropriate security has been established, the CE manager may instruct CEs as to which FEs they should control and how they should control them.

4.8. Gi Reference Point

Packets that enter the NE via one FE and leave the NE via a different FE are transferred between FEs across the Gi reference point. (See [FORCESREQ], Section 5, Requirement 3.)

4.9. Gp Reference Point

Based on the information acquired through CEs' control processing, CEs will frequently need to manipulate the packet-forwarding behaviors of their FE(s). This manipulation of the forwarding plane is performed across the Gp ("p" meaning protocol) reference point. In this architecture, the ForCES protocol is exclusively used for all communication across the Gp reference point.

4.10. Gr Reference Point

Varying degrees of synchronization are necessary to provide redundancy, load sharing or distributed control. However, in all cases, consistency protocols between CEs take place across the Gr reference point and are out of the scope of this document. Likewise, detecting the inability to synchronize due to a loss of connectivity between CEs is out of the scope of this document.

It is not necessary to define any protocols across the Gr reference point to enable simple control/forwarding separation (i.e., single CE and multiple FEs). However, to make it possible to define Gr at a later time, the concept of CE sets and the associated CE/FE behavior should be included in the first versions of the ForCES protocol. From the basic CE set building block concept, protocols across the Gr reference point can be defined to provide the desired effect.

5. FE Model

This section describes a model that can be used to express the capabilities of a ForCES FE. (As we will see, this model can also be used as the basis to control a FE's capabilities.) This model satisfies the requirements set forth in ForCES requirements document [FORCES-REQ] with respect to FE modeling. Our model is composed of two level hierarchy of detail. The higher level of the hierarchy expresses which logical data path elements exist in the FE and describes how these elements are interconnected. We call these logical data path elements "stages." The lower level of the hierarchy expresses the capabilities of each stage that the FE provides. In general, the lower level expresses these capabilities in terms of five categories: 1) what information the stage uses to classify packets, 2) once classified, the actions the stage can perform on the packet, 3) the statistics the stage collects in this process, 4) the asynchronous events the stage may send to the CE as part of this process, and 5) the parameters that the stage uses to control its overall behavior.

5.1. Introduction

The ForCES architecture allows Forwarding Elements (FEs) of varying functionality to participate in a ForCES network element. The implication of this varying functionality is that CEs can make only minimal assumptions about the functionality provided by its FEs. Instead, CEs discover the capabilities of their FEs. [FORCES-REQ] mandates that this capability information be expressed in the form of a FE model. [FORCES-REQ] further requires that this FE model describe which logical functions (i.e., stages) are present in the FE and in which order these stages are performed. See [FORCES-REQ] for types of logical functions that this model must support. For each logical function, [FORCES-REQ] also requires that the FE model

be able to describe each stageÆs "capabilities."

A stage's capabilities clarify what the stage does but not how it does it. (There is a small exception to this described later for the case where the FE allows the CE to choose which algorithm the stage should use.) For example, a forwarding function may perform a lookup on destination IP address and mask to find a next hop IP address and egress interface. However, the fact that the forwarding function uses a Patricia Trie or a CAM to accomplish this lookup is not relevant to the CE. Stage capabilities are best illustrated by the following description of the logical packet-processing model of a stage.

Stages logically process packets using the following process. First, the stage receives a packet and performs a classification step on the packet. This classification step finds the highest priority rule (i.e., filter) in the stage's rule set (i.e., classification or rule table) that matches the given packet. Next, the stage performs one or more actions associated with the matching rule. As part of this process, the stage may update certain statistics (e.g., number of packets processed, number of packets matching each filter rule) to reflect the types of packets it has processed. As one of the actions (or occasionally asynchronously), the stage may generate an event for further processing by the CE. For example, a stage may detect that the router alert IP option is present in a packet and would then generate a "packet redirection" event to send the packet to the CE. Finally, some stages may have tunable "knobs" that affect how they process packets. For example, a FE may provide various algorithms for performing a metering function (e.g., average rate, exponentially weighted moving average, token bucket).

From this process, we see that the capabilities of stages can be modeled by describing the five logical sets of data maintained by each stage. The first two sets of data are the filtering rules and associated actions that are applied to each packet as they pass through the stage. The third set of data is the statistics maintained by the stage. The fourth set is the current state of the stage's tunable "knobs." Finally, the fifth set is the set of events for which the CE has registered to receive notifications from the stage. Manipulation of these five logical databases can be used as a model for control of each stage.

5.2. Model Approach

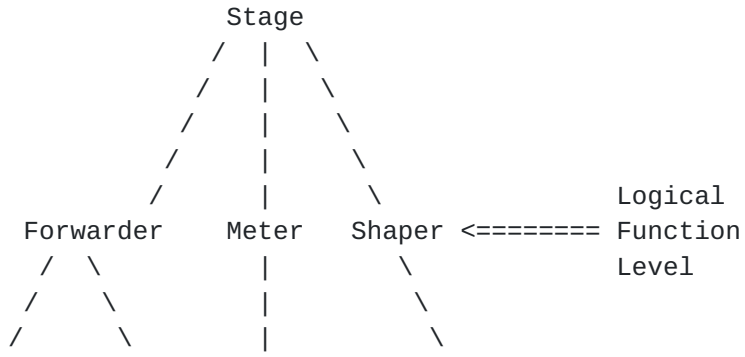
There are many ways that one could model the packet processing capabilities of a FE. However, as we shall see, there is often a tradeoff between the flexibility of a FE model and the ease with which the CE can interpret that model to provide services. One approach to this problem is to define a number of simple "device types." Each of these device types would have well-known components connected together in well-known ways. For example, we could define a RFC1812 router device type that does a longest prefix match on

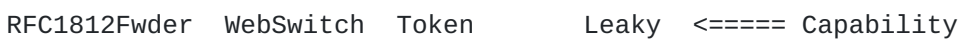
destination IP address and mask and forwards packets to the associated next hop IP address. However, since many services (e.g., QoS, firewall, intrusion detection) are being added to network devices, the number of possible device types would be exponential in the number of services. Writing a CE that understood exponentially many device types would be a daunting task. Therefore, one would likely want to restrict the number of devices types to a small set of "likely" devices. Coming up with this set would be difficult. Furthermore, restricting device types would seem to disallow vendors from creating interesting new devices. One could attempt to solve this problem by allowing vendors to define their own proprietary device types but this only leads to another explosion of device types and introduces interoperability problems for CE vendors who do not have access to the description of FE vendors' proprietary device types.

The FE model proposed in this document tries to strike a balance between flexibility of the model and ease of use by the CE. The model tries to strike this balance by describing packet processing in two levels of detail. The higher level of detail (Section 5.3) uses the concept of logical functions to make it easier for CEs to determine how to implement a service with a given model. The lower level of detail (Section 5.4) allows great flexibility to express the realization of a logical function chosen by a FE. The model allows arbitrary topologies to be described. While arbitrary topologies make it harder for the CE to understand the FE, it is asserted that static topology (or small set of topologies) is insufficient to describe the types of devices already in use.

5.3. Logical Functions and Topology

There are two largely orthogonal parts to the FE model proposed in this draft. The first part provides a way to describe which logical functions are present in a FE and how packets flow between these logical functions. The concept of a logical function is akin to that of an abstract base class in object-oriented terminology. By saying that a FE supports a logical function, what we are really saying is that the FE implements a specific concrete "derived class" version of the logical function. The following inheritance diagram illustrates this concept.




RFC1812Fwder WebSwitch Token Leaky <===== Capability

Bucket Bucket Level

By describing the FE at this high level, the FE model is able to give a broad overview of what processing a FE may perform on packets. The goal of this part of the FE model is to provide a way for the CE to know which stage(s) to modify to achieve a given service. As such, this model allocates a namespace for the specification of different logical functions. (We expect about 15 to 20 logical functions to be defined initially, e.g., ingress port, egress port, forwarder, meter, marker, shaper, scheduler, queue, encapsulator, decapsulator, encrypter, decrypter, NAT, mux, demux, and editor.) Each FE allocates a FE-unique stage identifier (USI) to each of its stages and passes the USI along with the corresponding logical function name as part of the FE capability description. This allows there to be multiple instances of the same logical function in each FE's model. We will start with a simple version of the model illustrating a capability exchange. In subsequent sections, we will expand the model and refine the same capability exchange. The following is the first version of the capability exchange that indicates which logical functions are present and how they are connected together.

- The number of stages supported.
- For each stage:
  - The USI.
  - The logical function name (from the namespace) that this stage implements.
  - The number of downstream stages to which this stage can send packets.
  - For each downstream stage:
    - The USI of the downstream stage.
    - A label for this exit point (i.e., target) from the stage.

This representation allows zero or more instances of each logical function to be present in a FE model. Furthermore, this representation encodes the topology of the provided stages. Since it is not possible to represent all possible FEs' processing models using a fixed topology, the model presented in this draft allows functions to be connected with largely arbitrary topologies. The only restrictions on topology relate to the source and sink natures of ingress and egress port functions respectively. For example, egress port functions must not have any downstream stages whereas no other stage may refer to an ingress port function as one of its downstream stages. Cycles in the topology are permitted.

5.4. Stage Capabilities

This section defines how the capabilities of all the stages in our model can be expressed using a single methodology. We achieve this uniformity by viewing all stages as acting according to the classification/action paradigm. In this paradigm, when a packet

logically enters a stage, the stage first performs a classification

on the packet. This classification is performed according to a logical database of classification entries maintained by the stage. Next, the stage performs one or more actions associated with the matching classification entry. Each classification entry contains this set of actions that the stage should perform for all packets that match the entry.

This paragraph provides several examples of how the stages identified in Section 3 can be viewed as acting according to the classification/action paradigm. This paradigm is most naturally applied to the generic filtering stages. In those stages, prioritized filters (e.g., ACLs) are installed in a stageÆs logical database. These filters specify which fields in the packet should be evaluated and which values should be present in those fields for the filter to match. In each filter, a pass or drop action is typically specified that determines the disposition of packets matching the filter. This paradigm maps to classical layer 3 forwarding in the following way. The logical database of classification/action entries corresponds to a forwarding table. The entries in this forwarding table have typically consisted of a network address, a network mask, a next hop IP address, and an egress interface number. The network address and mask make up the classification portion of this entry while the next hop IP address and egress interface correspond to a parameterized "forwarding decision" action. The typical longest-prefix match algorithms utilized by forwarding stages are nothing but classification algorithms optimized for a masked match against a packetÆs destination IP address. Finally, the metering stage can also be viewed in terms of classification and action. Meters take a flow specification and some rate limiting parameters (and optionally a rate limiting algorithm). This flow specification may be based on DSCP, 5-tuple or some other arbitrary packet contents. In any case, this flow specification essentially defines a classification entry. The rate limiting parameters are parameters to the specified rate limiting action (or to an assumed rate limiting algorithm when one is not explicitly specified).

While most of the functionality of a stage can be described according to the classification/action paradigm, some additional functions remain. These additional functions relate to how the stage as a whole operates (as opposed to how the stage handles individual flows), the kinds of asynchronous notifications that the stage can send to the CE and the types of statistics the stage maintains. While we will often have no control over the algorithm the stage uses to perform its function, there may be certain knobs and dials that we can adjust to control the algorithm. We call these knobs and dials "parameters" to the stage because they resemble parameters to algorithms. For example, one can view an ingress port stage as running an ARP algorithm that responds to ARP requests. In order for the ARP algorithm to know when to respond to an ARP request, the ARP algorithm needs to know the IP addresses of

each port. Thus, IP addresses can be viewed as parameters to the ingress port stage.

Next, some stages can be viewed as the originators of asynchronous notifications, i.e., events. These events correspond to occurrences that the CE cannot anticipate. For example, the ingress and egress port stages may be able to send the link up/down event when they detect that their port link state has changed. Likewise, one or more stages may support the packet redirection event for sending well-known control packets to the CE. Since CEs may not want to receive all the events that a FE may generate, the ForCES protocol SHOULD support a registration/deregistration mechanism where the CE can signal its interest in receiving the events that it has discovered via this FE model. Finally, stages may maintain certain statistics related to their packet processing.

In simplest terms, we describe the capabilities of each stage simply by listing the names of the items in each of the five categories that that stage supports. This approach is illustrated in the following updated capability exchange.

- The number of stages supported.
- For each stage:
  - The USI.
  - The logical function name (from the namespace) that this stage implements.

  - The number of properties supported by the stage.
  - For each property:
    - The name of the property from the property namespace.

  - The number of properties supported by the stage.
  - For each action:
    - The name of the action from the action namespace.

  - The number of parameters supported by the stage.
  - For each parameter:
    - The name of the parameter from the parameter namespace.

  - The number of events supported by the stage.
  - For each event:
    - The name of the event from the event namespace.

  - The number of statistics supported by the stage.
  - For each statistic:
    - The name of the statistic from the statistic namespace.

  - The number of downstream stages to which this stage can send packets.
  - For each downstream stage:
    - The USI of the downstream stage.

- A label for this exit point (i.e., target) from the stage.

The following paragraphs describe in more detail how the classification, action, parameter, event and statistics capabilities are expressed.

5.4.1. Classification Capabilities

The classification capabilities of a stage are expressed in our model through a variable length sequence of "properties." Each property in the sequence indicates that the stage is capable of including that property in any of the classification entries for that stage. Properties come in two varieties: packet properties and metadata (tag) properties. Packet properties are those protocol fields that occur explicitly in packets. For example, in the IP protocol, the version, type of service bits, fragment offset, time-to-live, protocol, source address, and destination address are potentially useful packet properties for classification. Other examples of useful packet properties include UDP source/destination port, TCP source/destination port, and ICMP type and code fields. Metadata (tag) properties are those values associated with a packet that do not occur explicitly in the packet. For example, the "ingress port" tag may be associated with a packet by the ingress stage. This tag indicates by which port the packet entered the FE. This tag may be useful to classify on in subsequent stages. For example, some stages may give preferential treatment to packets arriving on a certain port because that port is associated with a customer receiving premium service. Without the "ingress port" tag, subsequent stages would have no way of knowing on which port a packet entered the FE. As another example, if the forwarder stage is processing a multicast packet, that stage may need to know what port the packet came in on so that the forwarder does not send the packet back along the original link. In order to exchange property information, we must agree on how to represent the presence of absence of a property. This model allocates a property namespace for this purpose. This namespace is shared across all stages because many stages will classify on the same properties (e.g., ingress/egress port number or destination IP address).

5.4.2. Action Capabilities

Similarly, the action capabilities of a stage are represented by a logical sequence of "actions." Each action in the sequence indicates that the stage is capable of having that action associated with one of the stageÆs classification entries. Actions come in three varieties. The first type of action edits (e.g., changes a field, inserts/removes a header) the current packet being processed. The second type of action associates or dissociates a piece of metadata (tag) with the packet being processed. The third type of action selects a target (i.e., downstream stage) for the packet. For example, the action provided by the forwarder stage typically associates the "forwarding decision" tag with a packet. (The

forwarding decision tag is a parameterized tag that specifies which interface(s) the packet should be sent out and what the next hop IP address is of the next router(s).) The egress stage then logically classifies on this forwarding decision tag to determine which interface to send the packet out. As another example, the Meter stage may be configured to either drop packets exceeding a certain rate limit or it may be configured to simply "tag" those packets (e.g., with the "exceeding guaranteed rate" tag). A subsequent stage may be configured to drop or pass packets tagged this way depending on some other characteristic of the system. In contrast, NAT stages would use the first type of action to edit the current packet by rewriting the source or destination IP address. Some stages may be configured to drop packets matching certain classifiers. Drop may be seen as removing all the headers and payload from the packet and removing all associated metadata properties as well. Like properties, this model allocates a namespace for the identification of different actions. This namespace is shared across all stages because different stages may share the same action (e.g., drop).

5.4.3. Parameter Capabilities

The parameters supported by a stage are expressed by a logical sequence of "parameters." Each parameter in the sequence represents one of the knobs or dials used by the stage. A namespace is allocated for the identification of parameters. This namespace is shared across all stages because stages may share the same parameters.

5.4.4. Event Capabilities

The events supported by a stage are expressed by a logical sequence of "events." Each event in the sequence represents one of the events that the FE may be configured to send to the CE when the event happens. A namespace is allocated for the identification of events. This namespace is shared across all stages because stages may share the same events (e.g., packet redirection or link up/down).

5.4.5. Statistics Capabilities

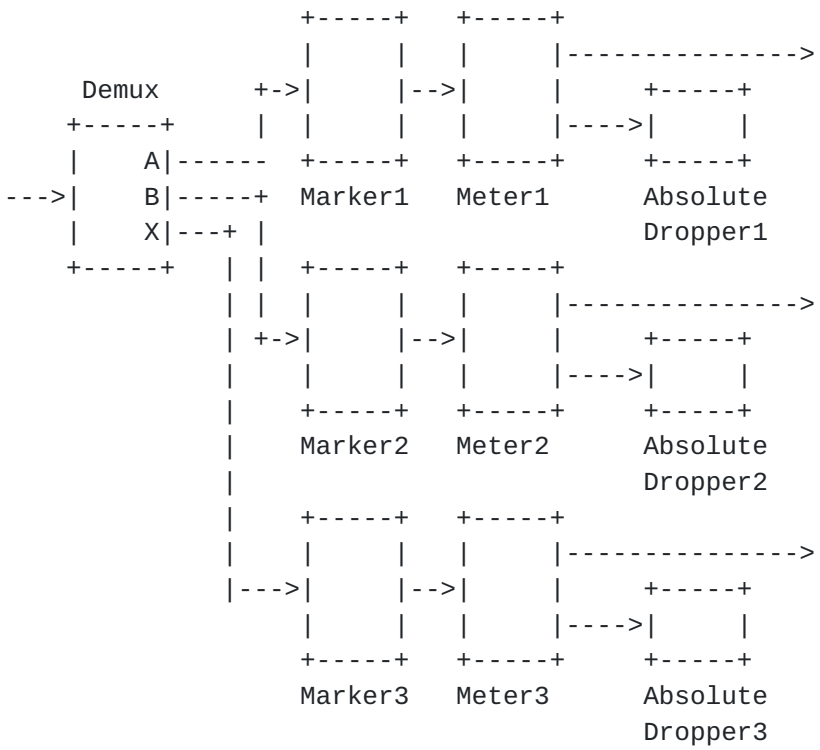
The statistics collected by a stage are expressed by a logical sequence of "statistics." Each statistic in the sequence represents one of the statistics maintained by the stage. A namespace is allocated for the identification of statistics. This namespace is shared across all stages because stages may share the same statistics (e.g., number of packets processed).

5.5. Read-only Stages

The FE model must be able to express that certain stages in a FE may

not be modifiable by a CE. However, the model cannot simply ignore

these stages, as it may be necessary to understand their functionality to predict the behavior of the FE. For example, consider the following subset of a FE model. While the FE may allow the Demux to be configured to select different kinds of traffic to be sent to the A, B, and X targets, the subsequent meters may not be programmable. However, the behavior of these meters must be known so that the CE can make decisions as to which traffic should be sent to which target (depending on the QoS desired for the traffic).



Two additions to the model are necessary to support read-only stages: first, a Boolean flag that indicates whether the stage is read-only or not, and second, an agreed upon way of expressing any static classification/action entries. (There may be static parameters as well, which will need a similar expression.) In each classification/action entry, there are zero or more properties and one or more actions. When multiple properties are present, the result is a logical AND of each property (e.g., if destination IP address==X AND IP protocol==TCP AND TCP destination port number==80). When multiple actions are present, all those actions are performed on matching packets. To represent each property or action, a type/length/value (TLV) approach is used. The names defined the property and action namespaces are suitable as the type in the TLV. The length of the TLV is an appropriately sized integer and represents the size of the "value" portion of the TLV. The value portion of the TLV may itself have some structure and it is therefore necessary to standardize a data structure that corresponds to each type in the namespace. Combining all these concepts together, the following model is used to express the static classification/action entries:

- The number of static classification/action entries.
- For each entry:
  - The number of properties.
  - The number of actions.
  - For each property:
    - The name of the property.
    - The length of the property.
    - The value of the property (using the data structure corresponding to the given name.)
  - For each action:
    - The name of the action.
    - The length of the action.
    - The value of the action (using the data structure corresponding to the given name.)

5.6. TLV Errata

The capability exchange shown in Section 5.4 represents an all-or-nothing approach to the five categories of capabilities. For example, either you support all types of classification (e.g., equal to, not equal to, range matching, inverse range matching) for all values of a property or you support no classification for that field. However, in practice, things are often not as simple. For example, some stages may be able to classify on specific values for certain fields but no others, or a stage may be able to match the IP protocol field for either TCP or UDP but nothing else. The FE model must therefore be capable of expressing these sorts of restrictions on the values associated with any of the five categories of capabilities. To express these restrictions, no longer can we describe capabilities by listing the names of supported items in each of the five namespaces. Instead, along with each supported item, the model must describe any restrictions associated with that item. The model describes these restrictions in the following way.

Like section 5.5, a TLV structure is used. However, each TLV contains two values instead of one. The first value represents the bottom of a range of allowable values for the item while the second value represents the top of a range of allowable values. It is important to note the difference between the ability to select one specific value in a range between A and B and the ability to select a range of values, C-D, between A and B (A < C < D < B). The two values in the TLV represent A and B but do not imply the ability to do range checking. In fact, several different kinds of matching are capable with the specific range of values. There is "equal to" matching (e.g., does field X have the value C, where A < C < B?), "not equal to" matching (e.g., is X not equal to C?), "less than" matching, "not less than" matching, "inside range" matching (e.g., is X in C-D?), and "not inside range" matching (e.g., is X not in C-D?). "Less than" and "not less than" matching are specialized forms of range matching and can be expressed in that form given an appropriate lower or upper bound. We therefore need four additional

flags associated with each specified range (i.e., A-B). These flags

indicate whether equal to, not equal to, inside range, or not inside range types of matching are allowed. Using the property category as an example, the capability expression model becomes the following:

- The number of properties supported by the stage.
- For each property:
  - The name of the property from the property namespace.
  - The length of the value portion associated with this property.
  - A flag indicating whether "equal to" classification is allowed.
  - A flag indicating whether "not equal to" classification is allowed.
  - A flag indicating whether "inside range" classification is allowed.
  - A flag indicating whether "not inside range" classification is allowed.
  - The bottom of a range of values, using the data structure associated with the given property.
  - The top of a range of values, using the data structure associated with the given property.

The previous paragraph describes capabilities inside one contiguous range. This paragraph describes how capabilities are represented in non-contiguous ranges, as in the one that motivated this section (i.e., matching the IP protocol field for TCP or UDP only). To express capabilities for non-contiguous ranges, multiple capabilities entries are used, each having the same name from the chosen namespace. For example, to express our motivating example, the following two entries are used.

- 2 properties entries to follow.
- Entry 1:
  - Name: IP protocol
  - Length: two octets.
  - Equal to: True
  - Not equal to: False
  - Inside range: False
  - Not inside range: False
  - Bottom: 6, TCP
  - Top: 6, TCP
- Entry 2:
  - Name: IP protocol
  - Length: two octets.
  - Equal to: True
  - Not equal to: False
  - Inside range: False
  - Not inside range: False
  - Bottom: 17, UDP
  - Top: 17, UDP

Unlike properties, the other four categories have no need for the flags indicating the four types of classification. However, the

other four categories still do need the bottom and top of range to

indicate the range of allowable values from which the CE can select only one.

5.7. Completed Capability Exchange

Having updated the capability exchange data model to express each stage's capabilities according to the five categories, the capability exchange consists of the following information:

- The number of stages supported.
- For each stage:
  - The USI.
  - The logical function name (from the namespace) that this stage implements.

  - The number of properties supported by the stage.
  - For each property:
    - The name of the property from the property namespace.
    - The length of the value portion associated with this property.
    - A flag indicating whether "equal to" classification is allowed.
    - A flag indicating whether "not equal to" classification is allowed.
    - A flag indicating whether "inside range" classification is allowed.
    - A flag indicating whether "not inside range" classification is allowed.
    - The bottom of a range of values, using the data structure associated with the given property.
    - The top of a range of values, using the data structure associated with the given property.

  - The number of actions supported by the stage.
  - For each action:
    - The name of the action from the action namespace.
    - The length of the value portion associated with this action.
    - The bottom of a range of values, using the data structure associated with the given action.
    - The top of a range of values, using the data structure associated with the given action.

  - The number of parameters supported by the stage.
  - For each parameter:
    - The name of the parameter from the parameter namespace.
    - The length of the value portion associated with this parameter.
    - The bottom of a range of values, using the data structure associated with the given parameter.
    - The top of a range of values, using the data structure associated with the given parameter.

- The number of events supported by the stage.

- For each event:
  - The name of the event from the event namespace.
  - The length of the value portion associated with this event.
  - The bottom of a range of values, using the data structure associated with the given event.
  - The top of a range of values, using the data structure associated with the given event.

- The number of statistics supported by the stage.
- For each statistic:
  - The name of the statistic from the statistic namespace.
  - The length of the value portion associated with this statistic.
  - The bottom of a range of values, using the data structure associated with the given statistic.
  - The top of a range of values, using the data structure associated with the given statistic.

- A flag indicating whether the stage is read-only.

- The number of static classification/action entries.
- For each static classification/action entry:
 - The number of properties.
 - The number of actions.
 - For each property:
   - The name of the property.
   - The length of the property.
   - The value of the property (using the data structure corresponding to the given name.)
 - For each action:
   - The name of the action.
   - The length of the action.
   - The value of the action (using the data structure corresponding to the given name.)

- The number of static parameters.
- For each static parameter:
   - The name of the parameter.
   - The length of the parameter.
   - The value of the parameter (using the data structure corresponding to the given name.)

- The number of downstream stages to which this stage can send packets.
- For each downstream stage:
  - The USI of the downstream stage.
  - A label for this exit point (i.e., target) from the stage.

6. Applicability to RFC1812

[To be done.]

7. Security Considerations

Significant security considerations need to be documented but were not done in time for submission. Next revision will begin to address these issues.

8. References

[FORCES-REQ] T. Anderson, et. al., "Requirements for Separation of IP Control and Forwarding", work in progress, September 2001, <draft-anderson-forces-req-02.txt>.

9. Authors' Addresses

Todd A. Anderson
Intel Labs
2111 NE 25th Avenue
Hillsboro, OR 97124 USA
Phone: +1 503 712 1760
Email: todd.a.anderson@intel.com