Some Possible Extensions of the Current LFB Model

<draft-blake-forces-attrib-00.txt>

Steven Blake
slblake@modularnet.com

Zsolt Haraszti
zsolt@modularnet.com

Presentation available at:

http://www.petri-meat.com/slblake/networking/drafts/forces-attrib.pdf

ForCES Model Schema

- Schema defines syntax for representing FE attributes and capabilities.
- Schema defines a syntax for representing LFB class attributes and capabilities:
  - Attributes include tables, flags, etc.
  - Capabilities include the max range of a particular attribute (e.g., max table size).
- Requirement to define LFB classes such that each represents “a fine-grained, logically separable and well-defined packet processing operation in the datapath”.
- Model should support a generalized set of LFB class definitions that can be used to model any functional implementation.

LFB Model

- Key characteristics:
  - Flat (non-hierarchical, non-nested) layer of interconnected LFBs within a particular FE.
  - Exclusive ownership of attributes:
    - Each LFB has its own resources/attributes, not accessible to others.
  - Attributes configured by ForCES messages addressed to that LFB only (or a subset of all LFBs of that class).

Problem Statement

- There is a problem between the current, simple LFB model, and the requirement to define LFB classes so that each is fine-grained.
- This is due to lack of support for one or more of the three following capabilities:
  - Efficient, bundled configuration of multiple LFBs
  - Attribute sharing between LFBs
  - Inter-LFB control

Example 1: Unicast LPM and RPF

- Unicast route lookup depends on an LPM Classification of the IP destination address, as well lookup of the appropriate next-hop info.
- RPF check for blocking source address spoofing depends on LPM classification of the IP source address, as well as comparison of the incoming interface to the next-hop info.
- Ideal LFB topology:
  - LPM (source IP) -> Next-hop (RPF) -> LPM (destination IP) -> Next-hop (normal)
- Ideally, the LPM and next-hop databases are shared between both sets of LFB instances.

Example 2: ARP and L2 Address Resolution

- ARP might be offloaded into a LFB in a FE.
- ARP LFB needs to update the L2 Address Resolution table, which would be an attribute of a L2 Address Resolution LFB.
- These two LFBs are not necessarily adjacent:
  - ARP function might sit on the “ingress” side of a FE.
  - L2 address resolution function might sit on the “egress” side.

Example 3: Interface MIB Counters

- Some Interface MIB counters are incremented exclusively of others:
  - Ex/ ifInUcastPkts or ifInErrors
  - There must be a common decision point where one or the other of these counters is incremented.
  - Ideally there is only one LFB that needs to be queried to retrieve all Interface MIB counters.
  - Solutions:
    - One big IP Interface LFB which performs all header verification tests to ensure that ifInErrors should not be incremented.
    - Split LFBs (e.g., IP Interface and IP header verification) that share the counter table.

Bundled Configuration

- There is a conflict between the goals of:
  - Good functional separation between LFBs (e.g., high-granularity), and
  - Efficient, one-step configuration of a certain forwarding operation, which begs for a single, complex LFB.
- In the absence of LFB attribute sharing, it may be desirable for the model and protocol to support the simultaneous configuration of attributes of multiple LFBs, to ensure consistency, and to simplify the CE software.
- Support for atomic transactions in the ForCES protocol may be sufficient to cover these cases (TBD).

Attribute Sharing

- In an implementation, two functions that are not adjacent in a LFB topology graph may share tables for efficiency:
  - The LFB model should not impose additional configuration operations that are not required in the implementation.
  - The LFB model should not introduce indeterminate states in the FE (e.g., two LFB tables configured differently that are shared in an implementation).
- This problem could be solved by:
  - Extending the schema to permit sharing of attributes between LFB classes, or instances of the same class.
  - Defining some attributes as belonging to the FE instead of individual LFB classes.

Inter-LFB Control

- One LFB may need to configure the attributes of another LFB (Example 2). This could be achieved by:
  - LFB attribute sharing
  - Special control metadata forwarded with packets between the LFBs
  - Inter-LFB control paths in the LFB topology

Possible Model Extensions

- Ideally, one or two extensions to the model schema would be sufficient to solve all three problems.
- Requirements:
  - Model extensions should not impose any extra burden on the definition of LFB classes that do not benefit from them.
  - Use of these extensions in a LFB class definition, or in LFB instances within a particular FE, should be elective.
  - Non-explicit extensions (e.g., hidden sharing or control paths) should be avoided.

Possible Model Extensions (2)

- Dispatcher (proxy) LFB:
  - Not in the datapath, but used as a protocol proxy to configure two or more LFBs.
  - Solves: Bundled configuration
- Protocol messages addressing multiple LFBs:
  - Solves: Bundled configuration
- Nested LFBs:
  - A LFB could be decomposed into a graph of simpler LFBs, sharing attributes of the parent.
  - Solves: Bundled configuration, Shared attributes

Possible Model Extensions (3)

- FE-level attributes:
  - Some attributes accessed by LFBs could be defined at the FE level.
  - Solves: Shared attributes
- Resource *soft-links*/LFBs exporting certain attributes to other LFBs:
  - Solves: Bundled configuration, Shared attributes, Inter-LFB control
- Decoupling nodes in the LFB topology from LFB instances:
  - One LFB instance may show up in the topology as two or more nodes.
  - Still leads to complex LFBs
  - Solves: Shared attributes, Inter-LFB control

Possible Model Extensions (4)

- Configuration/control channels between LFBs:
  - Show up in the LFB topology as special (non-packet) paths.
  - Solves: Bundled configuration, Inter-LFB control
- List above is not exhaustive.

Summary

- The ForCES model will likely need one or more extensions.
- Extensions should be:
  - Backwards compatible
  - Little or no effect on mature LFB definitions (not many at the moment, however).
  - Optional (in LFB class definition or use in LFB instances)
  - Explicit
- Preferably only one or two extensions are needed.
- Further analysis is needed.