Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: October 05, 2013

J. Hadi Salim
Mojatatu Networks

April 03, 2013

ForCES Protocol Extensions
draft-jhs-forces-protoextenstion-00

Abstract

Experience in implementing and deploying ForCES architecture has demonstrated need for a few small extensions both to ease programmability and to improve wire efficiency of some transactions. This document describes a few extensions to the ForCES Protocol Specification [RFC5810] semantics to achieve that end goal.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on October 05, 2013.

Table of Contents

1. Terminology and Conventions

1.1. Requirements Language

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 [RFC2119].

1.2. Definitions

This document reiterates the terminology defined by the ForCES architecture in various documents for the sake of clarity.

FE Model - The FE model is designed to model the logical processing functions of an FE. The FE model proposed in this document includes three components; the LFB modeling of individual Logical Functional Block (LFB model), the logical interconnection between LFBs (LFB topology), and the FE-level attributes, including FE capabilities. The FE model provides the basis to define the information elements exchanged between the CE and the FE in the ForCES protocol [RFC5810].

LFB (Logical Functional Block) Class (or type) - A template that represents a fine-grained, logically separable aspect of FE

processing. Most LFBs relate to packet processing in the data path. LFB classes are the basic building blocks of the FE model.

LFB Instance - As a packet flows through an FE along a data path, it flows through one or multiple LFB instances, where each LFB is an instance of a specific LFB class. Multiple instances of the same LFB class can be present in an FE's data path. Note that we often refer to LFBs without distinguishing between an LFB class and LFB instance when we believe the implied reference is obvious for the given context.

LFB Model - The LFB model describes the content and structures in an LFB, plus the associated data definition. XML is used to provide a formal definition of the necessary structures for the modeling. Four types of information are defined in the LFB model. The core part of the LFB model is the LFB class definitions; the other three types of information define constructs associated with and used by the class definition. These are reusable data types, supported frame (packet) formats, and metadata.

LFB Metadata - Metadata is used to communicate per-packet state from one LFB to another, but is not sent across the network. The FE model defines how such metadata is identified, produced, and consumed by the LFBs, but not how the per-packet state is implemented within actual hardware. Metadata is sent between the FE and the CE on redirect packets.

ForCES Component - A ForCES Component is a well-defined, uniquely identifiable and addressable ForCES model building block. A component has a 32-bit ID, name, type, and an optional synopsis description. These are often referred to simply as components.

LFB Component - An LFB component is a ForCES component that defines the Operational parameters of the LFBs that must be visible to the CEs.

ForCES Protocol - Protocol that runs in the Fp reference points in the ForCES Framework [RFC3746].

ForCES Protocol Layer (ForCES PL) - A layer in the ForCES protocol architecture that defines the ForCES protocol messages, the protocol state transfer scheme, and the ForCES protocol architecture itself as defined in the ForCES Protocol Specification [RFC5810].

ForCES Protocol Transport Mapping Layer (ForCES TML) - A layer in ForCES protocol architecture that uses the capabilities of existing transport protocols to specifically address protocol

message transportation issues, such as how the protocol messages are mapped to different transport media (like TCP, IP, ATM, Ethernet, etc.), and how to achieve and implement reliability, ordering, etc. the ForCES SCTP TMLP [RFC5811] describes a TML that is mandated for ForCES.

2. Introduction

Experience in implementing and deploying ForCES architecture has demonstrated need for a few small extensions both to ease programmability and to improve wire efficiency of some transactions. This document describes a few extensions to the ForCES Protocol Specification [RFC5810] semantics to achieve that end goal.

This document describes and justifies the need for 3 small extensions which are backward compatible.

1. A table range query to allow a controller or control application to request for a range of table rows.

2. A table append operation to allow a controller to add a new table row using the next available table index

3. Additional Error codes returned to the controller (or control application) to improve granularity of existing defined error codes.

3. Problem Overview

In this section we present sample use cases to illustrate the challenge being addressed.

3.1. Table Ranges

Consider, for the sake of illustration, an FE table with 1 million table rows which are sparsely populated.

ForCES GET requests from a controller (or control app) are prepended with a path to a component and sent to the FE. In the case of indexed tables, the component path can either be a table or a table row index. A control application desiring to receive the first 2000 table rows appearing between row indices 23 and 10023 can achieve its goal in one of:

- Dump the whole table and filter for the needed 2000 table rows.

- Send upto 10000 ForCES PL requests with monotonically incrementing indices and stop when the first 2000 entries are retrieved.

- Use ForCES batching to send fewer large messages (2000 path requests at a time until you hit the require number of entries).

All of these approaches can be seen as programmatically unfriendly, tedious, and both compute and bandwidth abuse.

3.2. Table Append

For the sake of illustration, assume that a newly spawned controller application wishes to install a table row but it has no apriori knowledge of which table index to use.

ForCES allows a controller/control app to request for the next available table index as demonstrated in (Figure 1) (refer to [RFC5810] section 4.8.2 for details of table properties).

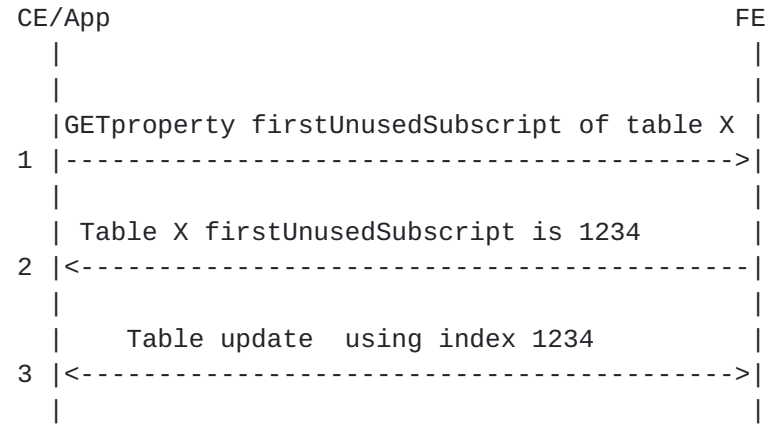

Figure 1: ForCES table property request

The problem with the above setup is the application requires one roundtrip time to figure out the index to insert into. Moreover, depending on implementation (and in presence of multiple possible control applications), there is no guarantee that the next unused subscript in the above example would be 1235; which means if the application wishes to insert more than one entry, it will have to incur the roundtrip time for every to-be-inserted table row. XXX: talk about possibly enforcing reservations for multi-app etc.

3.3. Error codes

TBA

3.4. Bitmap Datatype

TBA

4. Protocol Update Proposal

4.1. Table Ranges

We propose to add a Table-range TLV (type ID 0x117) that will be associated with the PATH-DATA TLV in the same manner the KEYINFO-TLV is. Path flag of F_SELTABRANGE (0x2) is set to indicate the presence of the Table-range TLV. XXX: This pathflag can only be used in a GET and is mutually exclusive with F_SELKEY.

The Table-range TLV contents constitute:

- A 32 bit start index. XXX: May need a wild-card.

- A 32 bit end index. XXX: May need a wild-card.

The response for a table range query will either be:

- When referenced table rows exist, then a response with a path pointing to the table and whose data content contain the rows will be sent to the CE. The encapsulating sparse data will have the "I" (in ILV) indicating the table indices.

- When data is absent, a return code indicating absence of content in the specified range is sent back to the CE.

4.2. Table Append

We propose using a path flag, F_TABAPPEND(0x4) to achieve this goal.

When a CE application wishes to append to the table, it will set the path to a desired table index and set the path flag to F_TABAPPEND. The FE will first attempt to use the specified index and when unsuccessful will use an available table row index.

When successful, an E_SUCCESS return code is sent back to the CE. The path piece of the response will contain the table row index where the table row was inserted.

4.3. Error Codes

4.4. Bitmap Datatype

TBA

5. IANA Considerations

TBA: request to IANA for Table-range TLV, F_SELTABRANGE etc.

6. Security Considerations

TBD

7. References

7.1. Normative References

[RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004.

[RFC5810] Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, March 2010.

[RFC5811] Hadi Salim, J. and K. Ogawa, "SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol", RFC 5811, March 2010.

[RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, March 2010.

7.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Author's Address

Jamal Hadi Salim
Mojatatu Networks
Suite 400, 303 Moodie Dr.
Ottawa, Ontario K2H 9R4
Canada

Email: hadi@mojatatu.com