Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 06, 2014

J. Hadi Salim
Mojatatu Networks
July 05, 2013

ForCES Protocol Extensions
draft-jhs-forces-protoextenstion-01

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.

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 TML [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 4 small extensions which are backward compatible.

1. A table range operation to allow a controller or control application to request or delete an arbitrary 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. Improved Error codes returned to the controller (or control application) to improve granularity of existing defined error codes.

4. Optimization to packing and addressing commonly used bitmap structure.

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 reasonably sized table rows which are sparsely populated.

ForCES GET requests sent 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 to a table or a table row index. A control application attempting to retrieve the

first 2000 table rows appearing between row indices 23 and 10023 can achieve its goal in one of:

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

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

o Use ForCES batching to send fewer large messages (several path requests at a time with incrementing indices until you hit the require number of entries).

All of these approaches are programmatically (from an application point of view) unfriendly, tedious, and are seen as abuse of both compute and bandwidth resources.

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).

```
    CE/App                                       FE
      |                                           |
      |                                           |
      |GETproperty firstUnusedSubscript of table X |
    1 |------------------------------------------->|
      |                                           |
      | Table X firstUnusedSubscript is 1234      |
    2 |<-------------------------------------------|
      |                                           |
      |    Table update  using index 1234         |
    3 |<------------------------------------------>|
      |                                           |
```

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 control applications):

1. there is no guarantee that the next available subscript in the above example would stay at 1234 at the moment an application chooses to do the update; this will entirely depend on implementation at the FE and/or available holes in the table.

2. In case of multiple apps wishing to insert rows to the same table concurently, all contending apps will be returned the same value for unused subscript; however, if all the contending apps try to insert at the same time, only the first one to reach the FE row will succeed. A solution involving a reservation mechanism to ask for an index will contribute complexity.

We conclude that even in the best case scenario, 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. This greatly affects table add latencies and update rates.

3.3. Error codes

[RFC5810] has defined a generic set of error codes that are to be returned to the CE from an FE. Deployment experience has shown that it would be useful to have more fine grained error codes. As an example, the error code E_NOT_SUPPORTED could be mapped to many FE error source possibilities that need to be then interpreted by the caller based on some understanding of the nature of the sent request. This makes debugging more time consuming.

3.4. Bitmap Datatype

TBA

4. Protocol Update Proposal

This section describes proposals to update the protocol for issues discussed in Section 3

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.

```
OPER = GET
       PATH-DATA:
       flags = F_SELTABRANGE,  IDCount = 2, IDs = {1,6}
       TABLERANGE-TLV = {11,23}
```

Figure 2: ForCES table range request

Figure 2 illustrates a GET request for a a table range for rows 11 to 23 of a table with component path of 1/6.

Path flag of F_SELTABRANGE (0x2 i.e bit 1, where bit 0 is F_SELKEY as defined in RFC 5810) is set to indicate the presence of the Table-range TLV. The pathflag bit F_SELTABRANGE can only be used in a GET and is mutually exclusive with F_SELKEY. The FE MUST enforce those constraints and reject a request with an error code of E_INVALID_FLAGS with an english description of what the problem is (refer to Section 4.3).

The Table-range TLV contents constitute:

- A 32 bit start index. An index of 0 implies the beggining of the table row.

- A 32 bit end index. A value of 0xFFFFFFFFFFFFFFFF implies the last entry. XXX: Do we need to define the "end wildcard"?

The response for a table range query will either be:

- The requested table data returned (when at least one referenced row is available); in such a case, a response with a path pointing to the table and whose data content contain the row(s) will be sent to the CE. The data content MUST be encapsulated in sparsedata TLV. The sparse data TLV content will have the "I" (in ILV) for each table row indicating the table indices.

- A result TLV when:

  * data is absent where the result code of E_NOT_SUPPORTED (typically returned in current implementations when accessing an empty table entry) with an english message describing the nature of the error (refer to Section 4.3).

  * When both a path key and path table range are reflected on the the pathflags, an error code of E_INVALID_FLAGS with an english message describing the nature of the erro (refer to Section 4.3).

  * other standard ForCES errors (such as ACL constraints trying to retrieve contents of an unreadable table), accessing unknown components etc.

4.2. Table Append

We propose using a path flag, F_TABAPPEND(0x4, bit 2) 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.

On success or failure to insert the table row, a result TLV will be returned with the appropriate code. Alternatively a the new EXTENDED-RESULT-TLV (refer to Section 4.3) maybe returned. The path of the response will contain the table row index where the table row was inserted (which the application can then learn).

When successful, an E_SUCCESS return code is sent back to the CE.

Upon failure to append the table row, an appropriate error code is sent back to the CE.

4.3. Error Codes

We propose a new TLV, EXTENDED-RESULT-TLV (0x118) that will carry both a result code as currently specified but also a string[N] cause. This is illustrated in Figure 3.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Type = EXTENDED-RESULT-TLV |               Length          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Result Value  |                   Reserved                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          Cause string                         |
 .                                                               .
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Extended Result TLV

- o Like all other ForCES TLVs, the Extended Result TLV is expected to be 32 bit aligned.
- o The Result Value is derived from the same current namespace as specified in RFC 5810, section 7.1.7.
- o It is recommended that the maximum size of the cause string should not exceed 32 bytes. We do not propose the cause string be standardized.

XXX: Backward compatibility may require that we add a FEPO capability to advertise ability to do extended results so that the CE is able to interpret the results.

4.4. Bitmap Datatype

TBA

5. IANA Considerations

This document registers two new top Level TLVs and two new path flags.

The following new TLVs are defined:

- Table-range TLV (type ID 0x117)

- EXTENDED-RESULT-TLV (type ID 0x118)

The following new path flags are defined:

- F_SELTABRANGE (value 0x2 i.e bit 1)

- F_TABAPPEND (value 0x4 i.e bit 2)

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