        eCoMP:a usecase of the unified radio and optical control architecture
                       draft-zhang-ccamp-ecomp-00

Abstract

   This document specifies a usecase, called enhanced coordinated multi-
   point (eCoMP), for integreting radio and optical networks. It focus
   on the fronthaul flexibility that allows "any-RRH_A to any-BBU_A"
   connection to improve the eCoMP service performance. The procedure
   of the usecase is based on the unified radio and optical control
   archtecture, and an extended OpenFlow protocol is introduced to
   realize the procedure.

Status of This Memo

Copyright Notice

to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

   The eCoMP exploits the fronthaul flexibility by dynamically
   reconfiguring the lightpath, which is to reassociate coordinated
   RRH_As (connected to different BBU_As) with a single BBU_A. With the
   benefits of flexibility, an RRH_A can choose a proper BBU_A for a
   specific purpose. In order to achieve agile RRU_A and BBU_A mapping,
   a flexible optical fronthaul network is required, in which the
   lightpath between the BBU_A and RRU_As can be reconfigured.

   To realize the eCoMP service, radio and optical resources should be
   jointly allocated, and radio and optical network devises should to be
   simutaneusly controlled. This memo introduces a mechanism to achieve
   agile RRU_A and BBU_A mapping in a SDN-enabled radio and optical
   control architecture. The procedure of eCoMP contains the following
   steps: 1) Radio controller(Radio-C) obtains the current radio
   resource information via an extended OFP message, and calculate new
   RRH_A and BBU_A mappings for maximizing the intra-BBU eCoMP ratio. 2)
   Transport controller (Transport-C) obtains the current transport
   resource information via an extended OFP message, and calculate the
   corresponding lightpaths for the new RRH_A and BBU_A mappings. 3)
   eCoMP reconfigures the BBU_A according to the result of new RRH_A and
   BBU_A mappings. 4) eCoMP reconfigures lightpath between the new RRU_A
   and BBU_A pairs.

2. Requirements Language

   The key words are "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL
   NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL".

3. Terminology

   This memo uses the following terms :RRU_A, BBU_A, TN_A, Radio-C,
   Transport-C.

4. Motivation

   The RAN architecture towards mobile 5G and beyond is undergoing a
   fundamental evolution, which brings optics into the radio world.
   Fronthaul is a new segment that leverages on the advantages of

optical communication for RAN transport. However, the current
fronthaul architecture shows a fixed connection between an RRH_A and
a BBU_A, which leads to inefficient resource utilization.

To solve these problem, the "any-to-any" correspondence between
RRU_As and BBU_As is becomming more and more important. The agile
RRU_A and BBU_A mapping can be reached through lightpath
reconfiguration. However, it is hard to realize dynamic RRU-BBU
reassociation because of the independent control of radio and optical
networks.

Therefore, this memo give a usecase to realize the eCoMP by joint
allocation radio and optical resources in an unified control plane.

5. Overview of eCoMP

5.1. Problem

Because the eCoMP service is realized by the lightpath
reconfiguration, a communication protocol between BBU_A and RRU_A is
needed. The communication protocol is designed specifically for
communications between a BBU_A and a controller.A controller may use
the communication protocol to send a lightpath reconfiguration
request to a BBU_A, and the BBU_A may reply with a set of
reconfigured lightpaths if the lightpaths satisfying the set of
constraints.

5.2. New Messages

The communication protocol operates over TCP, which fulfills the
requirements for reliable messaging and flow control without further
protocol work.

This memo define the following new communication protocol messages
for eCoMP:

Controller Request Message for RRU Feature (RRU_Feature_Req): A
message sent by a Controller to a RRU to request RRU Feature which
contains RRU_ID and RRU_IP. A Controller MUST send RRU Feature
request message to a RRU at initialization phase to get the
information about traffic load,wavelength and corresponding BBUs.
The details of RRU_Feature_Req message is described in Section 6.1.

RRU Reply Message for RRU Feature (RRU_Feature_Rep): A message sent
by a RRU to a Controller to reply specific RRU_Feature_Req message,
which contains the features of the RRUs, such as traffic load and
corresponding BBUs. A RRU sends RRU Feature reply message if and

only if it received related RRU_Feature_Req message. The details of
RRU_Feature_Rep message is described in Section 6.2.

Controller Request Message for TN Feature (TN_Feature_Req): A message
sent by a Controller to a TN to request TN Feature which contains
node_ID and node_IP. A Controller MUST send TN Feature request
message to a TN at initialization phase to get the information about
port ,wavelength and switch status. The details of TN_Feature_Req
message is described in Section 6.3.

TN Reply Message for RRU Feature (TN_Feature_Rep): A message sent by
a TN to a Controller to reply specific TN_Feature_Req message, which
contains the features of the TNs, such as port ,wavelength and switch
status. A TN sends TN Feature reply message if and only if it
received related TN_Feature_Req message. The details of
TN_Feature_Rep message is described in Section 6.4.

Controller Request Message to BBU_A for Lightpath Reconfiguration
(BBU_A_Mod): A message sent by a Controller to a BBU_A to request
lightpath reconfiguration which contains BBU_ID, node_IP and BBU
status. A Controller MAY send lightpath reconfiguration request
message to a BBU_A at any time as long as it consideres this
operation necessary. The details of BBU_A_Mod message is described
in Section 6.11.

BBU_A Reply Message for Lightpath Reconfiguration (BBU_A_Rep): A
message sent by a BBU_A to a Controller to reply specific BBU_A_Mod
message, which contains the configuration results of BBU_As. A BBU_A
sends lightpath reconfiguration reply message if and only if it
received related BBU_A_Mod message. A BBU_A_Rep message can contain
either a set of reconfigurated lightpaths if the request can be
satisfied, or a negative reply if not. The negative reply may
indicate the reason why the lightpaths can not be reconfigurated.
The details of BBU_A_Rep message is described in Section 6.12.

Controller Request Message to TN_A for Lightpath Reconfiguration
(TN_A_Mod): A message sent by a Controller to a TN_A to request
lightpath reconfiguration which contains BBU_ID, node_IP , port,
wavelength and switch status. A Controller MAY send lightpath
reconfiguration request message to a TN_A at any time as long as it
consideres this operation necessary. The details of TN_A_Mod message
is described in Section 6.9.

TN_A Reply Message for Lightpath Reconfiguration (TN_A_Rep): A
message sent by a TN_A to a Controller to reply specific TN_A_Mod
message, which contains the configuration results of TN_As. A TN_A
sends lightpath reconfiguration reply message if and only if it
received related TN_A_Mod message. A TN_A_Rep message can contain

either a set of reconfigured lightpaths if the request can be
satisfied, or a negative reply if not. The negative reply may
indicate the reason why the lightpaths can not be reconfigured.
The details of TN_A_Rep message is described in Section 6.10.

5.3. Normal Communication Procedure

5.3.1. Initialization Phase

   The initialization phase consists of two subphases and each subphase
   two successive steps.

   In the first subphase, the two steps are (described in a schematic
   form in Figure 1:

   1) Establishment of a TCP connection (3-way handshake) between the
   RRU_A and the Controller.

   2) Establishment of a session over the TCP connection.

```
                   +-+-+-+              +-+-+-+-+-+
                   |RRU_A|              |Controller|
                   +-+-+-+              +-+-+-+-+-+
                      |                      |
                      |                      |
                      |--------------------->|
     (Initialize      |                      |
      Scop of         |    RRU_Feature_Req   |
      Attributes)     |<---------------------|
                      |                      |
                      |    RRU_Feature_Rep   |
                      |--------------------->|
                      |                      |
```
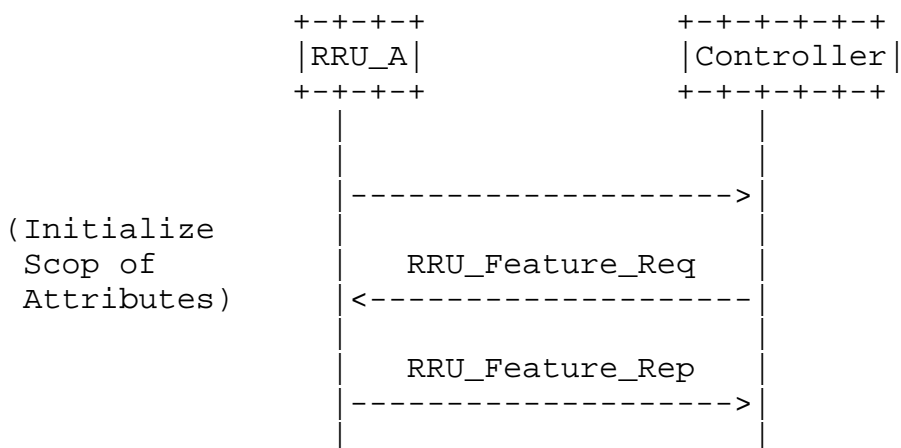
   Figure 1: Initialization Phase between the RRU_A and the Controller

   Once the TCP connection is established, the Controller and the RRU_A
   initiate session establishment during which various session
   parameters are negotiated. The Controller sends a RRU_A Feature
   request to the RRU(RRU_Feature_Req message).

   Details about the RRU_Feature_Req message can be found in
   Section 6.1.

   After received the RRU_Feature_Req message, the RRU send a
   RRU_Feature_Rep message including the features of the RRUs, such as
   traffic load, corresponding BBU_As and potentially other detailed

capabilities and policy rules that specify the conditions under which
path computation requests may be sent to the Controller.

Details about the RRU_Feature_Rep message can be found in
Section 6.2.

Similarly, in the second subphase, the two steps are (described in a
schematic form in Figure 2:

1) Establishment of a TCP connection (3-way handshake) between the
TN_A and the Controller.

2) Establishment of a session over the TCP connection.

```
                  +-+-+-+                +-+-+-+-+-+
                  | TN_A|                |Controller|
                  +-+-+-+                +-+-+-+-+-+
                     |                        |
                     |                        |
                     |----------------------->|
 (Initialize         |                        |
  Scop of            |     TN_Feature_Req     |
  Attributes)        |<-----------------------|
                     |                        |
                     |     TN_Feature_Rep     |
                     |----------------------->|
                     |                        |
```
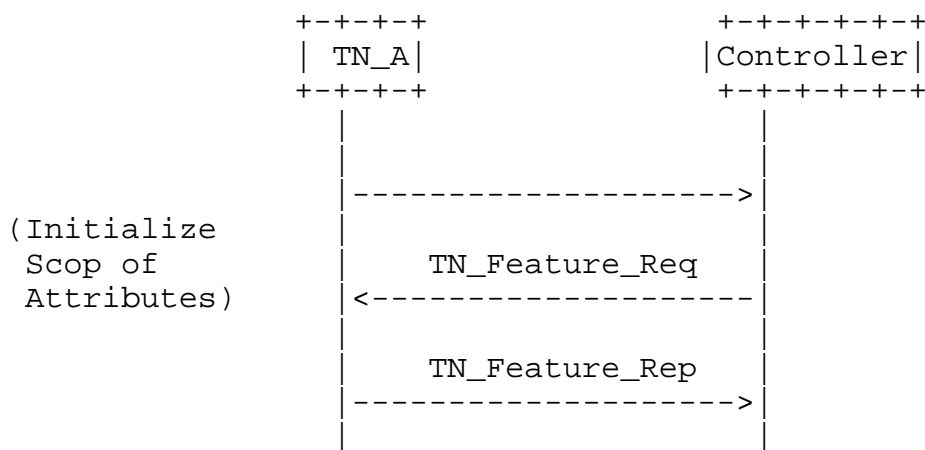
        Figure 2: Initialization Phasebetween the TN_A and the Controller

Once the TCP connection is established, the Controller and the TN_A
initiate session establishment during which various session
parameters are negotiated. The Controller sends a RRU Feature
request to the TN_A (TN_Feature_Req message).

Details about the TN_Feature_Req message can be found in Section 6.3.

After received the TN_Feature_Req message, the TN_A send a
TN_Feature_Rep message including the features of the TN_As, such as
traffic load, corresponding TN_A and potentially other detailed
capabilities and policy rules that specify the conditions under which
path computation requests may be sent to the Controller.

Details about the TN_Feature_Rep message can be found in Section 6.4.

5.3.2. Lightpath Reconfiguration Request Sent by a Controller to a
        BBU_A

   Once a Controller has sucessfully established a session with one or
   more BBU_As, if a lightpath reconfiguration event is triggered that
   requires the reconfiguration of a set of lightpaths, the controller
   first selects one or more BBU_As.

   Once the Controller has selected a BBU_A, it sends a BBU_A_Mod
   message to the BBU_A. Each request is uniquely identified by a tp-id
   number and Controller-BBU_A address pair. The process is shown in
   Figure 3.

   Details about the BBU_A_Mod message can be found in Section 6.11.

```
      +-+-+-+                 +-+-+-+-+-+
      |BBU_A|                 |Controller|
      +-+-+-+                 +-+-+-+-+-+
         |                       | 1)Lightpath Reconfiguration Event
         |                       | 2)Lightpath Reconfiguration Request
         |                       |    Sent to the BBU_A
         |<------BBU_A_Mod------|
         |                       |
         |                       |
         |                       |
```
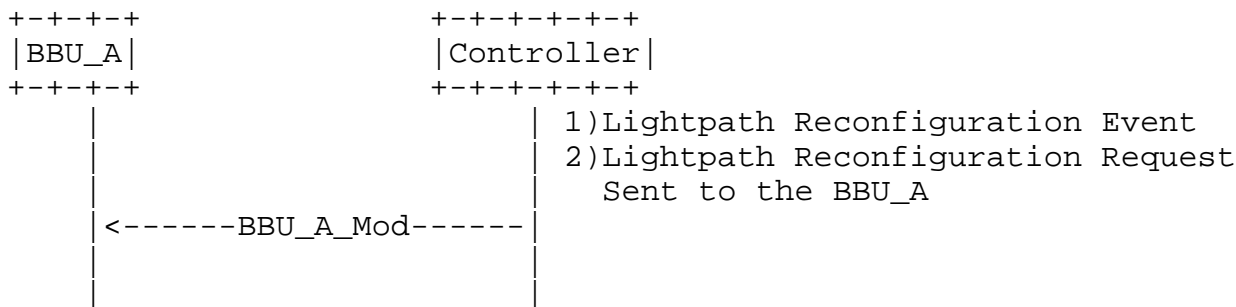
             Figure 3: Lightpath Reconfiguration Request to BBU_A

5.3.3. Lightpath Reconfiguration Request Sent by a Controller to a TN_A

   Once a Controller has sucessfully established a session with one or
   more TN_As, if a lightpath reconfiguration event is triggered that
   requires the reconfiguration of a set of lightpaths, the controller
   first selects one or more TNs.

   Once the Controller has selected a BBU_A, it sends a TN_Mod message
   to the TN_A. Each request is uniquely identified by a tp-id number
   and Controller-TN address pair. The process is shown in Figure 4.

   Details about the TN_Mod message can be found in Section 6.9.

```
        +-+-+-+                    +-+-+-+-+-+
        |TN_A |                    |Controller|
        +-+-+-+                    +-+-+-+-+-+
           |                          | 1)Lightpath Reconfiguration Event
           |                          | 2)Lightpath Reconfiguration Request
           |                          |   Sent to the TN_A
           |<------ TN_Mod -------|
           |                          |
           |                          |
           |                          |
```
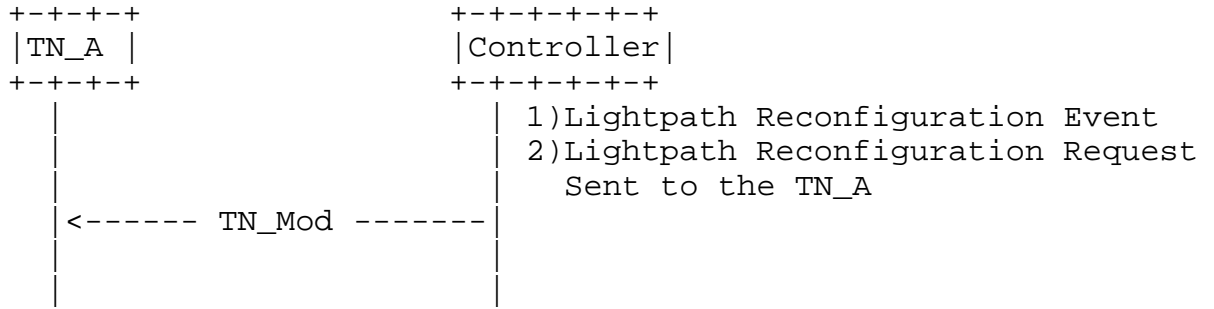
            Figure 4: Lightpath Reconfiguration Request to TN_A

5.3.4. Lightpath Reconfiguration Reply Sent by a BBU_A to a Controller

   After receiving a lightpath reconfiguration request from a
   Controller, the BBU_A triggers a lightpath reconfiguration, If the
   BBU_A manages to reconfigure a lightpath satisfies the set of
   required constraints, the BBU_A returns the result to the requesting
   Controller. The process is shown in Figure 5.

```
                        +-+-+-+              +-+-+-+-+-+
                        |BBU_A|              |Controller|
                        +-+-+-+              +-+-+-+-+-+
                           |                    |
                           |                    |
                           |                    |
                           |<------BBU_A_Mod------|
     1) Lightpath Reconfiguration |                    |
        Request Received          |                    |
     2) Reconfiguration Succefully |                    |
     3) Reconfiguration Result     |                    |
        Sent to the Controller     |                    |
                           |------BBU_A_Rep------>|
```
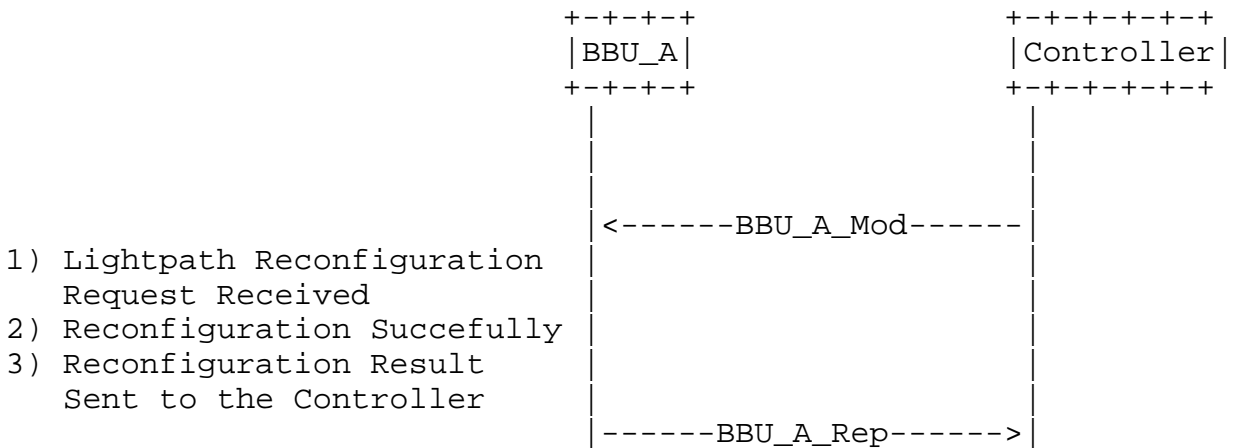
       Figure 5: Lightpath Reconfiguration Reply (Success) from BBU_A

   However, if no lightpath could be found that satisfies the set of
   constraints. In this case, a BBU_A may provide the set of
   constraints that led to the lightpath reconfiguration failure. Upon
   receiving a negative reply, a Controller may decide to resend a
   modified request or take any other appropriate action. The process
   is shown in Figure 6.

```
                        +-+-+-+                +-+-+-+-+-+
                        |BBU_A|                |Controller|
                        +-+-+-+                +-+-+-+-+-+
                           |                        |
                           |                        |
                           |                        |
                           |<-----BBU_A_Mod-------|
     1) Lightpath Reconfiguration |                |
        Request Received          |               |
     2) econfiguration Unsuccefully|              |
     3) Cause of Failure           |              |
        Sent to the Controller     |              |
                           |------BBU_A_Rep------>|
```
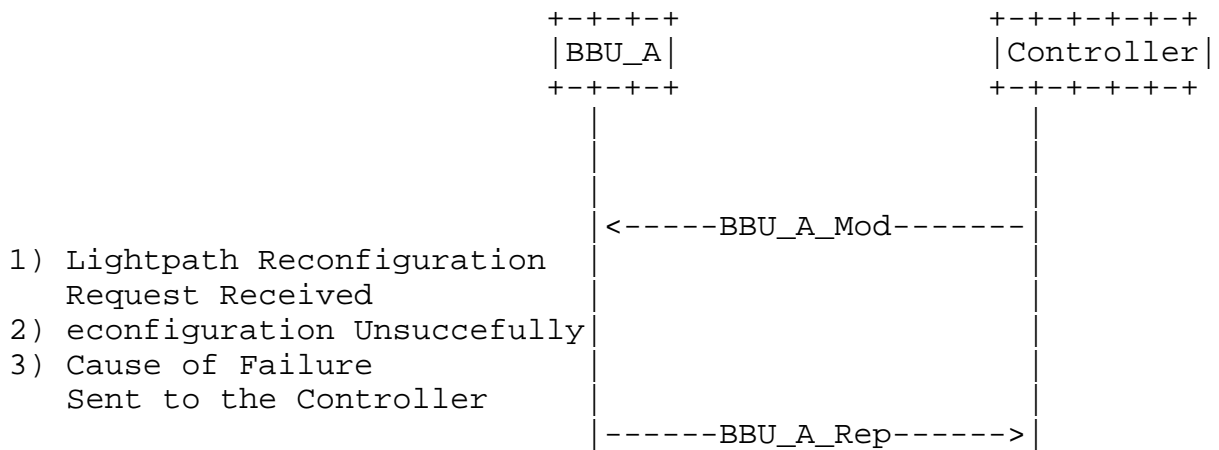
          Figure 6: Lightpath Reconfiguration Reply (Failure) from BBU_A

     Details about the BBU_A_Rep message can be found in Section 6.12.

5.3.5. Lightpath Reconfiguration Reply Sent by a TN_A to a Controller

     After receiving a lightpath reconfiguration request from a
     Controller, the TN_A triggers a lightpath reconfiguration, If the
     TN_A manages to reconfigure a lightpath satisfies the set of required
     constraints, the TN_A returns the result to the requesting
     Controller. The process is shown in Figure 7.

```
                        +-+-+-+                +-+-+-+-+-+
                        |TN_A |                |Controller|
                        +-+-+-+                +-+-+-+-+-+
                           |                        |
                           |                        |
                           |                        |
                           |<---- TN_A_Mod -------|
     1) Lightpath Reconfiguration |                |
        Request Received          |               |
     2) Reconfiguration Succefully |              |
     3) Reconfiguration Result     |              |
        Sent to the Controller     |              |
                           |----- TN_A_Rep ------>|
```
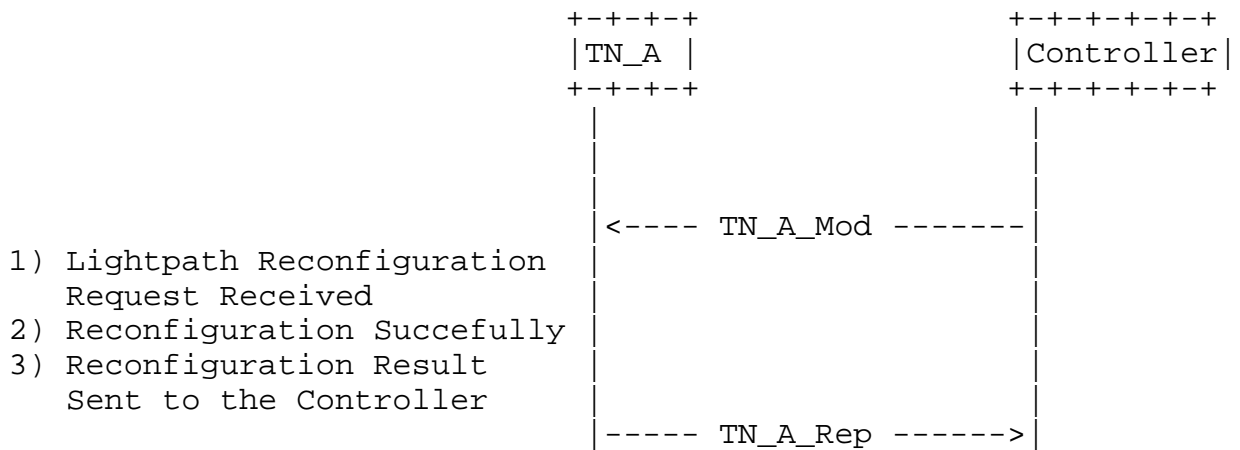
          Figure 7: Lightpath Reconfiguration Reply (Success) from TN_A

     However, if no lightpath could be found that satisfies the set of
     constraints. In this case, a TN_A may provide the set of constraints
     that led to the lightpath reconfiguration failure. Upon receiving a
     negative reply, a Controller may decide to resend a modified request
     or take any other appropriate action.' The process is shown in
     Figure 8

```
                        +-+-+-+                  +-+-+-+-+-+
                        |TN_A |                  |Controller|
                        +-+-+-+                  +-+-+-+-+-+
                           |                         |
                           |                         |
                           |                         |
                           |<----- TN_A_Mod ------|
        1) Lightpath Reconfiguration |            |
           Request Received          |            |
        2) econfiguration Unsuccefully|           |
        3) Cause of Failure           |           |
           Sent to the Controller     |           |
                           |------ TN_A_Rep ----->|
```
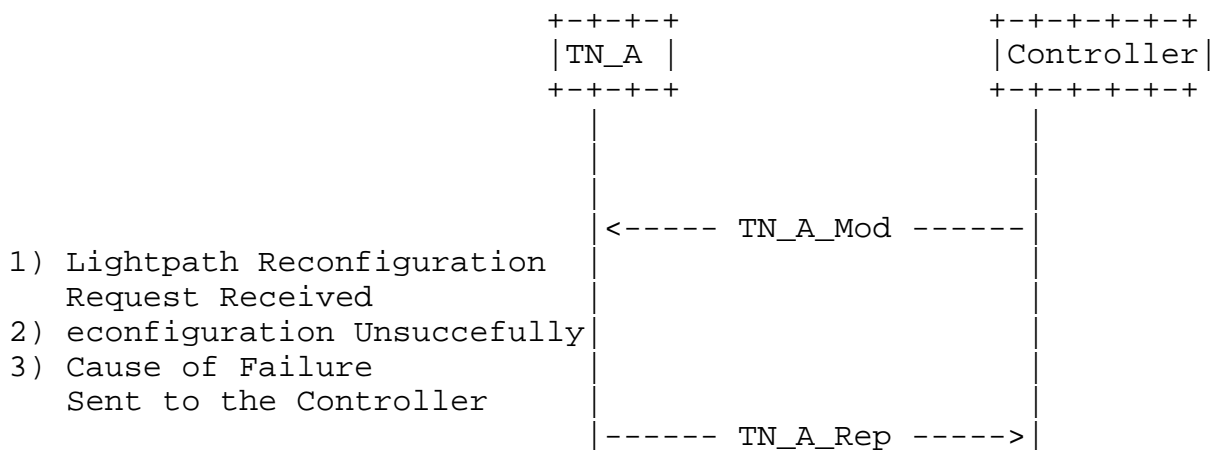
Figure 8: Lightpath Reconfiguration Reply (Failure) from TN_A

Details about the TN_A_Rep message can be found in Section 6.10.

6. the communication protocol Messages for eCoMP

The communication protocol Messages for eCoMP consists of a common
header followed by a variable-length body made of a set of objects.
For each message type, rules are defined that specify the set of
objects that the message can carry.

6.1. The RRU_Feature_Req message

```
       <RRU_Feature_Req message> ::= <Common Header>
                                     <RRU-information>
```
   Where:

```
       <RRU_information> ::= <RRU-ID>
                            <RRU-IP>
```

6.2. The RRU_Feature_Rep message

```
    <RRU_Feature_Rep Message> ::= <Common Header>
                                  <RRU-feature-reply>
Where:

    <RRU-feature-reply> ::= <RRU-information>
                            <RRU-feature>

    <RRU_information> ::= <RRU-ID>
                          <RRU-IP>

    <RRU-feature> ::= <target_BBU_ID>
                      <eCoMP_status>
                      <eCoMP_flow>
                      <total_flow>
                      <wavelength>
```

6.3. The TN_Feature_Req message

```
    <TN_Feature_Req message> ::= <Common Header>
                                 <TN-information>
Where:

    <TN_information> ::= <node-ID>
                         <node-IP>
```

6.4. The TN_Feature_Rep message

```
    <TN_Feature_Rep Message> ::= <Common Header>
                                 <TN-feature-reply>
Where:

    <TN-feature-reply> ::= <TN-information>
                           <TN-feature>

    <TN_information> ::= <node-ID>
                         <node-IP>

    <TN-feature> ::= <port>
                     <switch-status>
                     <wavelength>
```

6.5. The BBU_Feature_Req message

```
    <BBU_Feature_Req message> ::= <Common Header>
                                  <BBU-information>
  Where:

    <BBU_information> ::= <BBU-ID>
                         <BBU-IP>
```

6.6.   The BBU_Feature_Rep message

```
    <BBU_Feature_Rep Message> ::= <Common Header>
                                  <BBU-feature-reply>
  Where:

    <BBU-feature-reply> ::= <BBU-information>
                            <BBU-feature>

    <BBU_information> ::= <BBU-ID>
                         <BBU-IP>

    <BBU-feature> ::= <load-status>
                      <wavelength>
                      <eCoMP-status>
```

6.7.   The RRU_A_Mod message

```
    <RRU_A_Mod message> ::= <Common Header>
                            <Controller-reconfiguration-request>
  Where:
    <Controller-reconfiguration-request> ::= <RRU-ID>
                                             <node-IP>
                                             <eCoMP_status>
```

6.8.   The RRU_A_Rep message

```
    <RRU_A_Rep message> ::= <Common Header>
                            <Controller-reconfiguration-reply>
  Where:
    <Controller-reconfiguration-reply> ::= <RRU-ID>
                                           <node-IP>
                                           <RRU-config-reply>
```

6.9.   The TN_A_Mod message

```
    <TN_A_Mod message> ::= <Common Header>
                           <Controller-reconfiguration-request>
 Where:
    <Controller-reconfiguration-request> ::= <TN-ID>
                                             <node-IP>
                                             <TN-feature>


    <TN-feature> ::= <port>
                     <switch-status>
                     <wavelength>
```

6.10. The TN_A_Rep message

```
    <TN_A_Rep message> ::= <Common Header>
                           <Controller-reconfiguration-reply>
 Where:
    <Controller-reconfiguration-reply> ::= <TN-ID>
                                           <node-IP>
                                           <TN-config-reply>
```

6.11. The BBU_A_Mod message

```
    <BBU_A_Mod message> ::= <Common Header>
                            <Controller-reconfiguration-request>
 Where:
    <Controller-reconfiguration-request> ::= <BBU-ID>
                                             <node-IP>
                                             <eCoMP-status>
```

6.12. The BBU_A_Rep message

```
    <BBU_A_Rep message> ::= <Common Header>
                            <Controller-reconfiguration-reply>
 Where:
    <Controller-reconfiguration-reply> ::= <BBU-ID>
                                           <node-IP>
                                           <BBU-config-reply>
```

7. Object Formats

   A object carried within a communication protocol messages for eCoMP,
   which consists of one or more 32-bit words with a common header.

7.1. Initialization Phase Object

7.1.1. RRU feature request TLV

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      version   |     type=40    |         Message Length      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              xid                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            RRU_ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            RRU_IP                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
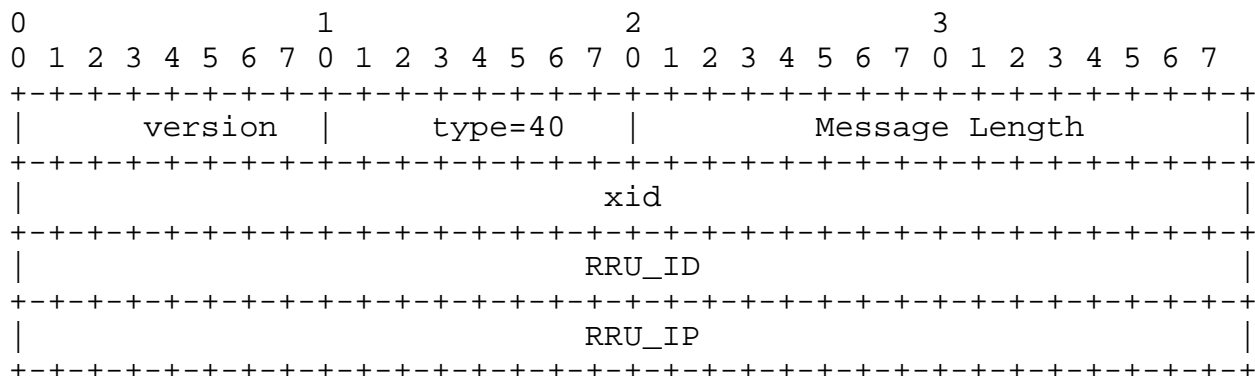
                Figure 9: RRU feature request TLV format

   The common header consists of version, type and message length.

   Version (8 bits): The version number. Current version is version 1.

   Type (8 bits): A number indicates the message type. The
   "RRU_Feature_Req" message is the type 40.

   Message Length (16 bits): Total length of the message including the
   common header, expressed in bytes.

   The RRU_Feature_Req object body consists of the hardware id (xid),
   RRU id(RRU_ID) and RRU ip (RRU_IP).

7.1.2. TN feature request TLV

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      version   |     type=41    |         Message Length      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              xid                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            node_ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            node_IP                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                Figure 10: TN feature request TLV format

The common header is similar with the RRU feature request object.
The "TN_Feature_Req" message is the type 41.

The TN_Feature_Req object body consists of the hardware id (xid),
node id(node_ID) and node ip (node_IP).

7.1.3. BBU feature request TLV

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     version    |    type=42     |         Message Length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             xid                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            BBU_ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            BBU_IP                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: BBU feature request TLV format

The common header is similar with the BBU feature request object.
The "BBU_Feature_Req" message is the type 42.

The TN_Feature_Req object body consists of the hardware id (xid), BBU
id(BBU_ID) and BBU ip (BBU_IP).

7.1.4. RRU feature reply TLV

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     version   |    type=43    |          Message Length       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              xid                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            RRU_ID                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            RRU_IP                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         target_BBU_ID                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         eCoMP_status                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          eCoMP_flow                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          total_flow                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          wavelength                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                 Figure 12: RRU feature reply TLV format

   The common header is similar with the RRU feature request object.
   The "RRU_Feature_Rep" message is the type 43.

   The RRU_Feature_Rep object body consists of the hardware id (xid),
   RRU id(RRU_ID), RRU ip (RRU_IP),target BBU ID, eCoMP status,
   eCoMP_flow, total flow, wavelength.

   The eCoMP_status is used to report the status of eCoMP. Two values
   are currently defined: "1" is a intra-BBU eCoMP state while "0" is a
   inter-BBU eCoMP state.

7.1.5. TN feature reply TLV

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       version  |      type=44   |         Message Length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            xid                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          node_ID                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          node_IP                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           port                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        switch_status                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         wavelength                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                Figure 13: TN feature reply TLV format

   The common header is similar with the RRU feature request object.
   The "TN_Feature_Rep" message is the type 44.

   The TN_Feature_Rep object body consists of the hardware id (xid),
   node id(node_ID), node ip (node_IP),port, switch_status and
   wavelength.

7.2. BBU feature reply TLV

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     version   |    type=45    |          Message Length       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                             xid                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            BBU_ID                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            BBU_IP                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          load_status                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          wavelength                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         eCoMP_status                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
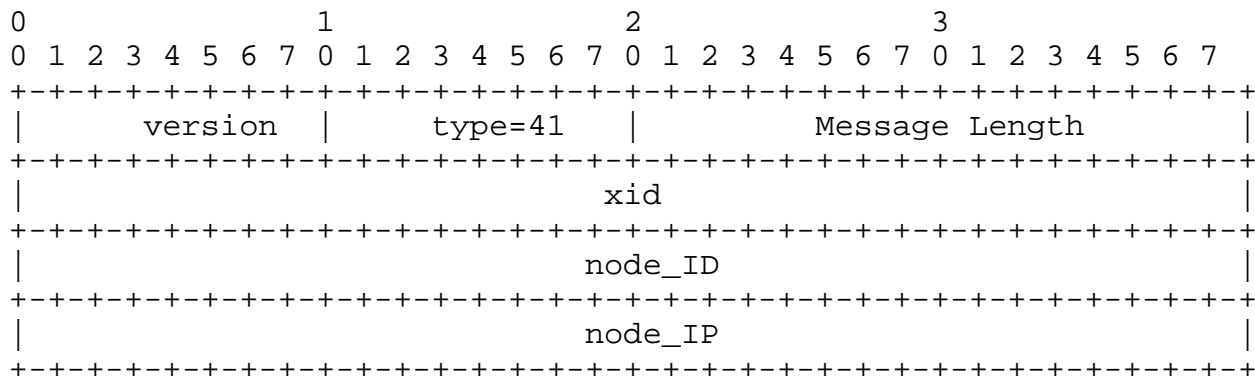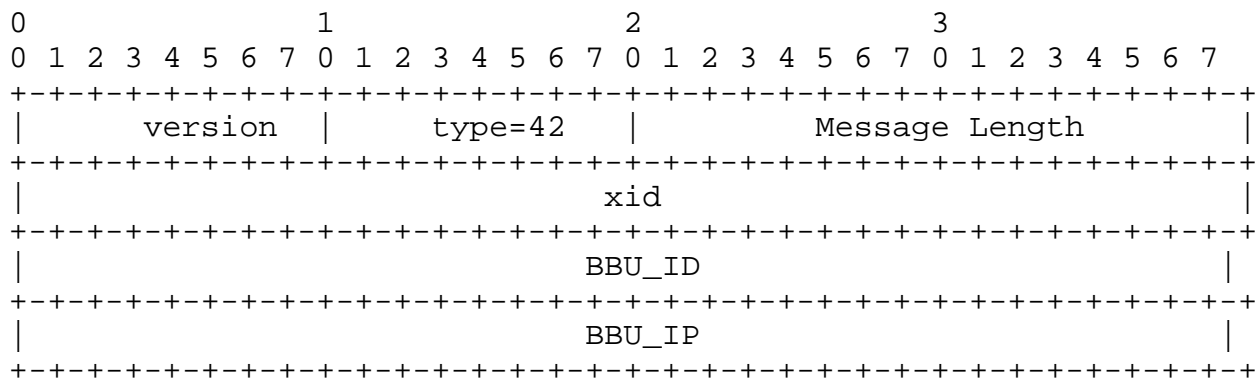
Figure 14: BBU feature reply TLV format

The common header is similar with the RRU feature request object.
The "RRU_Feature_Rep" message is the type 45.

The RRU_Feature_Rep object body consists of the hardware id (xid),
BBU id(BBU_ID), BBU ip (BBU_IP), load status, wavelength, eCoMP
status.

The eCoMP_status is used to report the status of eCoMP. Two values
are currently defined: "1" is a intra-BBU eCoMP state while "0" is a
inter-BBU eCoMP state.

7.3. Lightpath Reconfiguration Phase Object

7.3.1. RRU modification request TLV

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      version   |     type=46    |         Message Length      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            xid                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          RRU_ID                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          node_IP                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        eCoMP_status                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
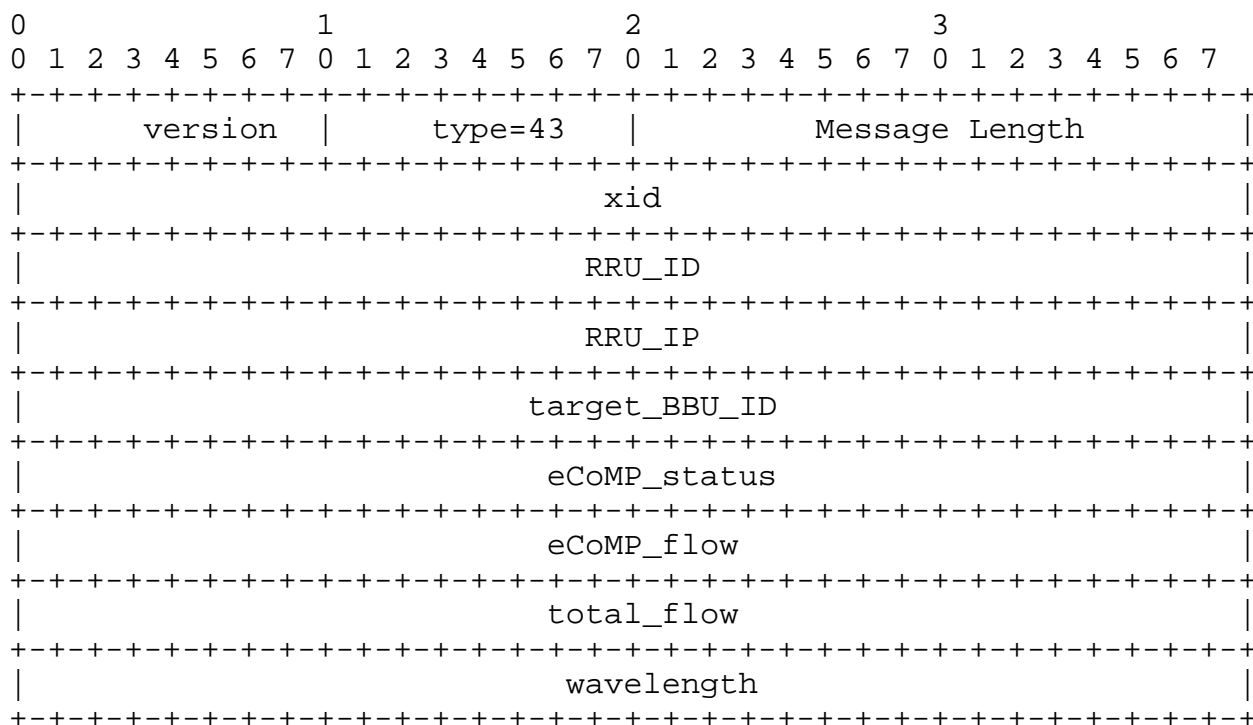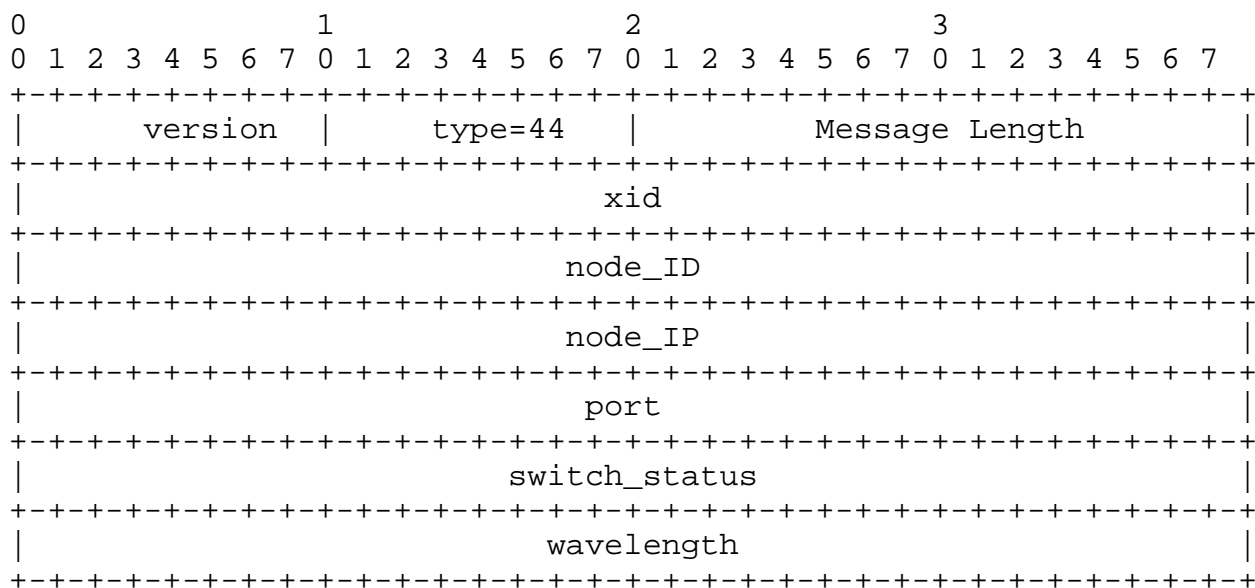
Figure 15: RRU modification request TLV format

The common header is similar with the RRU feature request object.
The "RRU_A_Mod" message is the type 44.

The RRU_A_Mod object body consists of the hardware id (xid), RRU
id(RRU_ID), RRU ip (node_IP) and eCoMP status(eCoMP_status).

7.3.2. TN modification request TLV

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      version   |     type=47    |         Message Length      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            xid                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          TN_ID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          node_IP                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            port                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        switch_status                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          wavelength                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: TN modification request TLV format

The common header is similar with the RRU feature request object.
The "TN_A_Mod" message is the type 47.

   The TN_A_Mod object body consists of the hardware id (xid), TN
   id(TN_ID), node ip (node_IP), port, switch_status and wavelength.

7.3.3. BBU modification request TLV
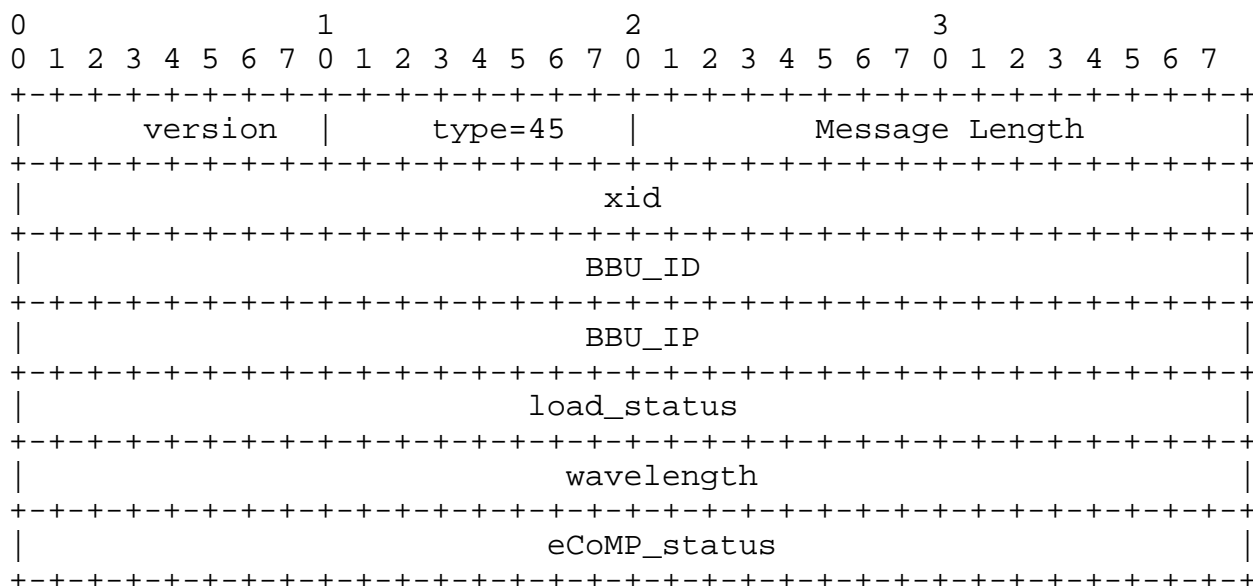
```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      version    |    type=48    |        Message Length        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             xid                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           BBU_ID                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          node_IP                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        eCoMP_status                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
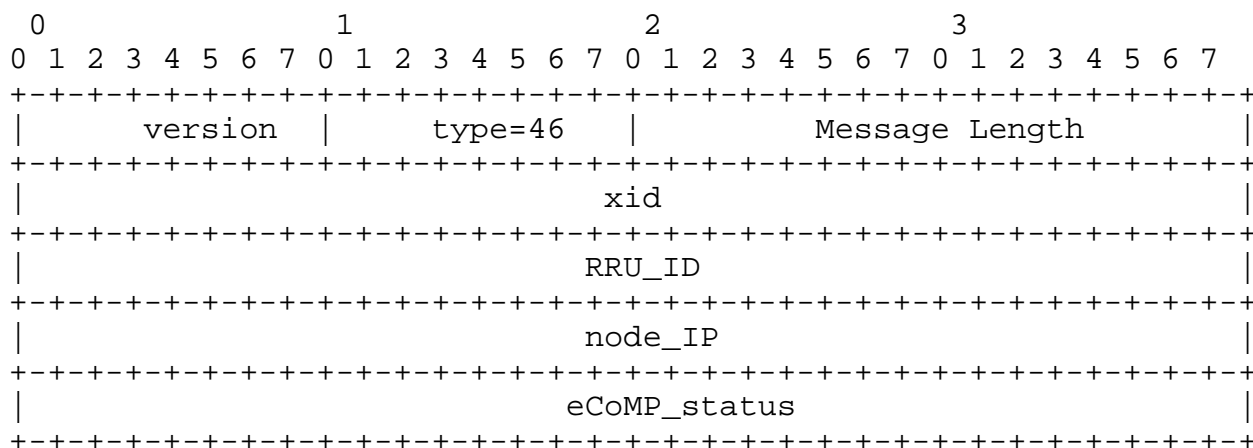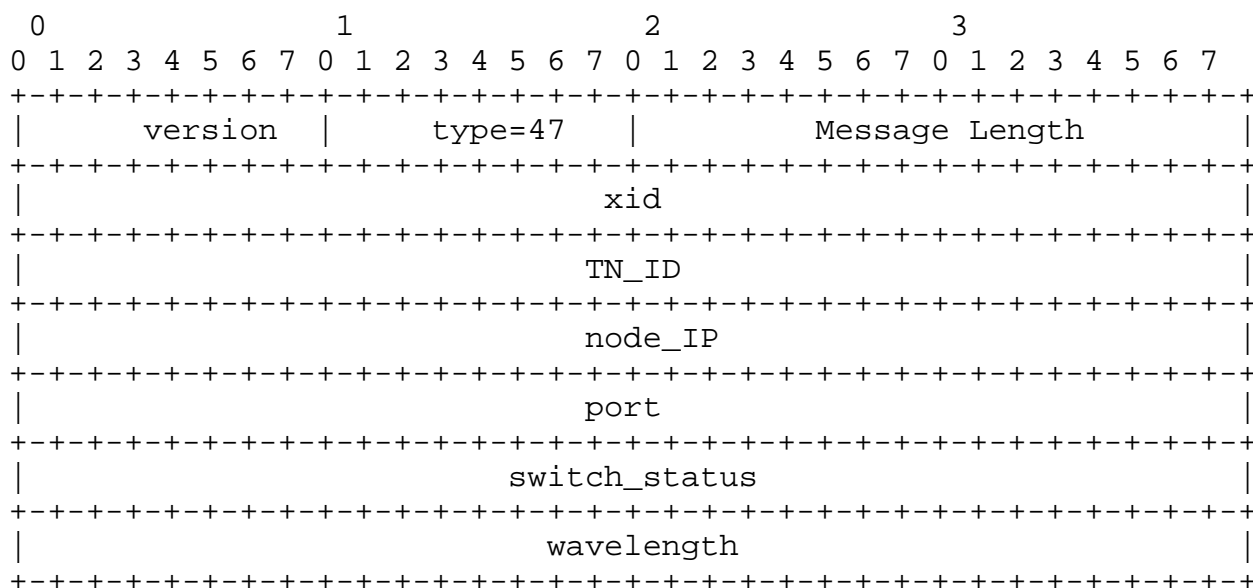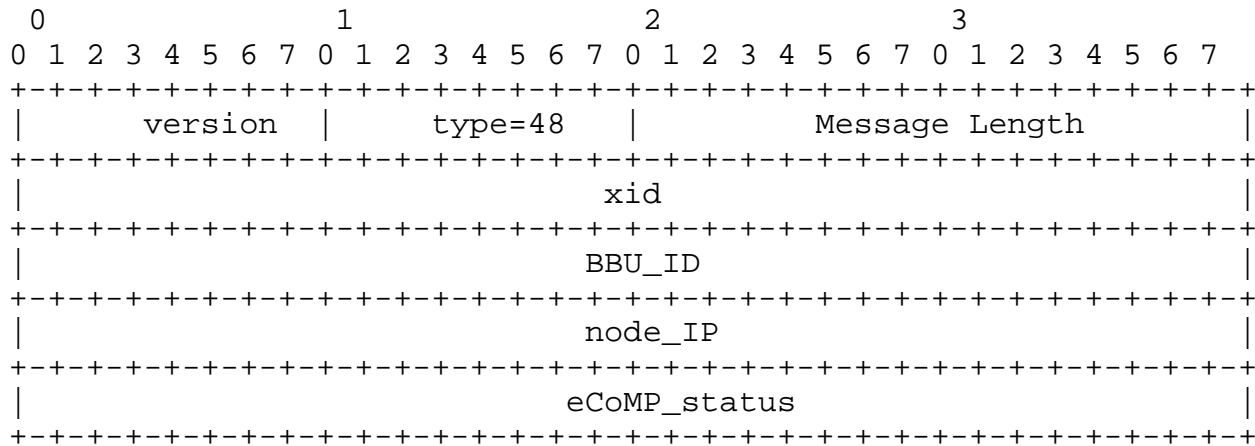
              Figure 17: BBU modification request TLV format

   The common header is similar with the RRU feature request object.
   The "BBU_A_Mod" message is the type 48.

   The BBU_A_Mod object body consists of the hardware id (xid), BBU
   id(BBU_ID), node ip (node_IP) and eCoMP status(eCoMP_status).

7.3.4. RRU modification reply TLV

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      version    |    type=49    |        Message Length        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             xid                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           RRU_ID                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          node_IP                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       RRU_config_reply                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

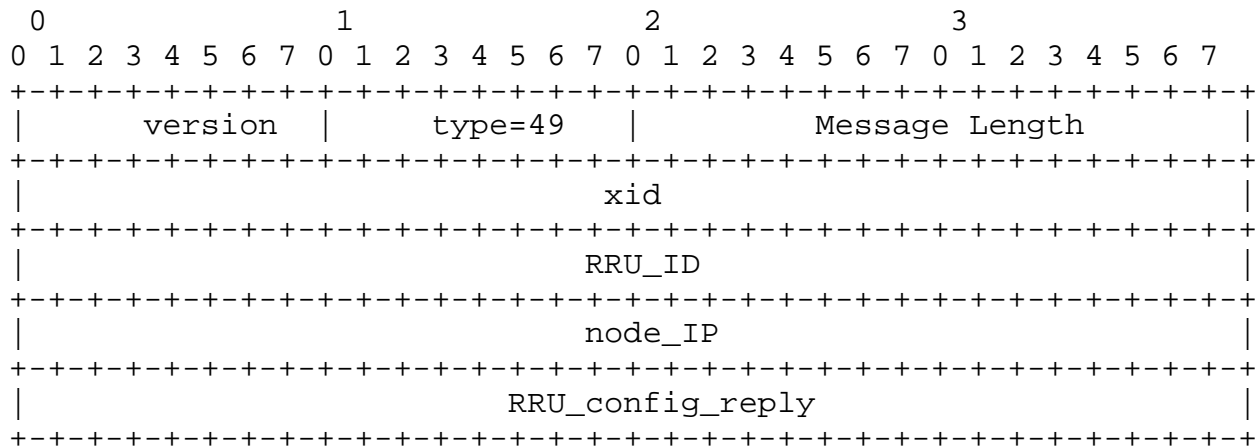   Figure 18: RRU modification reply TLV format

   The common header is similar with the RRU feature request object.
   The "RRU_A_Rep" message is the type 49.

The BBU_A_Mod object body consists of the hardware id (xid), RRU id(RRU_ID), node ip (node_IP) and BBU configuration reply(RRU_config_reply).

The RRU_config_reply is used to report the result of RRU configuration. Two values are currently defined: "1" is a successful state while "0" is a failure state.

7.3.5. TN modification reply TLV

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      version    |     type=50     |          Message Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              xid                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            TN_ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            node_IP                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         TN_config_reply                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
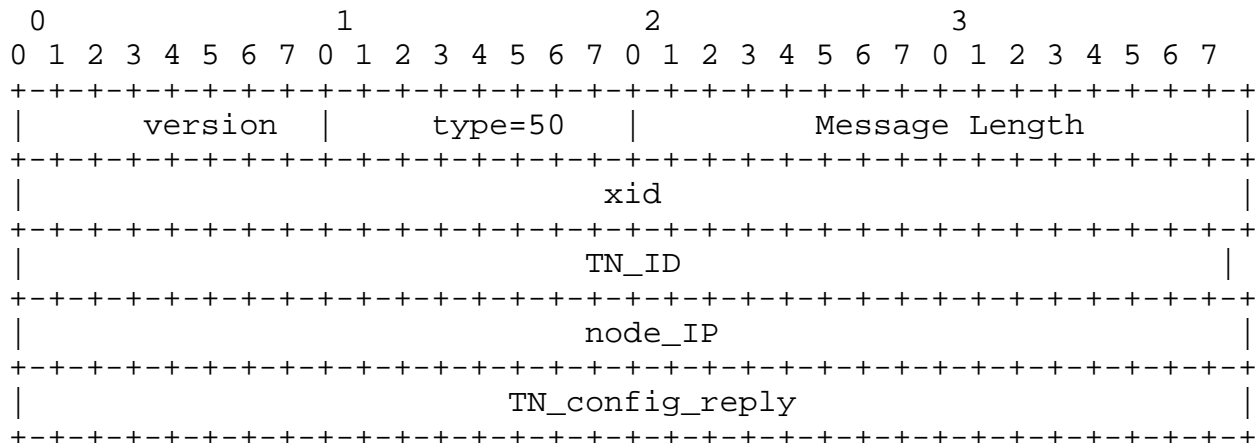
Figure 19: TN modification reply TLV format

The common header is similar with the RRU feature request object. The "TN_A_Rep" message is the type 50.

The TN_A_Mod object body consists of the hardware id (xid), TN id(TN_ID), node ip (node_IP) and TN configuration reply(TN_config_reply).

The TN_config_reply is used to report the result of TN configuration. Two values are currently defined: "1" is a successful state while "0" is a failure state.
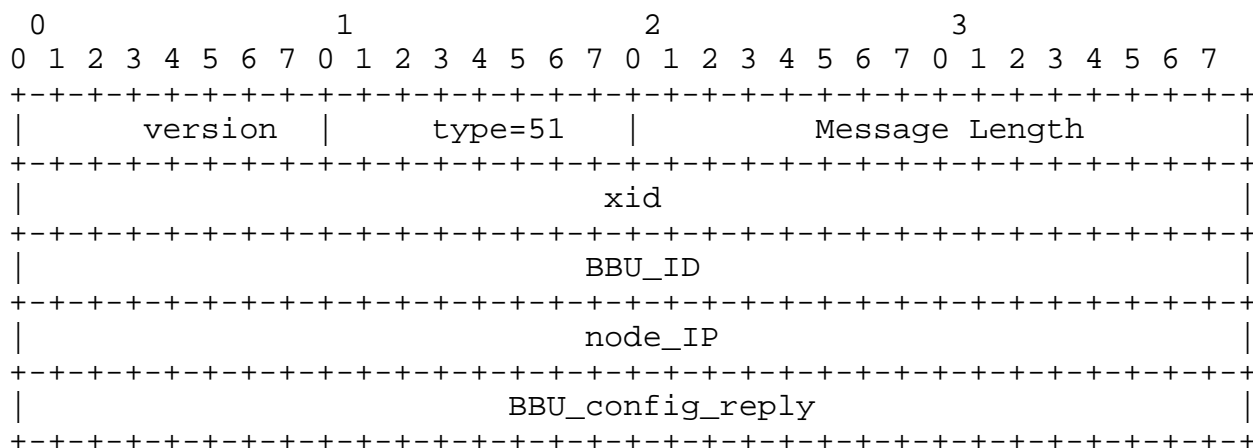
7.3.6. BBU modification reply TLV

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      version    |     type=51     |         Message Length    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                             xid                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                            BBU_ID                             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                            node_IP                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        BBU_config_reply                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

               Figure 20: BBU modification reply TLV format

   The common header is similar with the RRU feature request object.
   The "BBU_A_Rep" message is the type 51.

   The BBU_A_Mod object body consists of the hardware id (xid), BBU
   id(BBU_ID), node ip (node_IP) and BBU configuration
   reply(BBU_config_reply).

   The BBU_config_reply is used to report the result of BBU
   configuration. Two values are currently defined: "1" is a successful
   state while "0" is a failure state.

8. Acknowledgments

9. Contributors

Authors' Addresses

   Zhen Liu (editor)
   Beijing University of Posts and Telecommunications
   Xitucheng Road
   Beijing, Haidian Dist 100876
   China

   Phone: +86-15832040790
   Email: liuzhen207@bupt.edu.cn

Jiawei Zhang (editor)
Beijing University of Posts and Telecommunications
Xitucheng Road
Beijing, Haidian Dist 100876
China

Phone: +86-18600582335
Email: zjw@bupt.edu.cn

Sainan Liu (editor)
Beijing University of Posts and Telecommunications
Xitucheng Road
Beijing, Haidian Dist 100876
China

Phone: +86-010-61198422
Email: lsn0429@bupt.edu.cn

Yuefeng Ji (editor)
Beijing University of Posts and Telecommunications
Xitucheng Road
Beijing, Haidian Dist 100876
China

Phone: +86-13701131345
Email: jyf@bupt.edu.cn