Resource Computation Element (RCE)

Design and User Manual

Version 2.1b
April 2008

DRAGON

University of Southern California (USC)
Information Sciences Institute (ISI)

http://dragon.east.isi.edu

Table of Contents

1. Introduction

Resource Computation Element (RCE) is a control-plane element developed by the Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON) Project to perform resource information management and path computation in multi-layer multi-domain heterogeneous GMPLS networks. This section provides a brief background of the DRAGON architecture and the RCE functionality. An example network configuration will be described to illustrate the RCE usage within the DRAGON networking context.

1.1. Background

In the DRAGON network, RCE is mainly used to provide raw resource database (a.k.a. Traffic Engineering Database (TEDB)) and path computation services to NARB and therefore is often referred to as an integrated part of NARB. However, via an Application Programming Interface (API) RCE cannot only provide such services to NARB but also to any control-plane module that has an embedded RCE API client. RCE is a standalone software that maintains an self-contained resource database that is synchronized to external routing protocol daemons and an autonomous Path Computation Engine (PCEN) that utilizes the raw resource database to perform constraint based path computation. RCE uses the so-called 3D Resource Computation Module (RCM) to incorporate three broad dimensions of constraints, the Traffic Engineering (TE) constraints, the time schedule constraints, and the AAA policy constraints, into its path computation algorithms. These constraints are available in the resource TE database or fed from other control-plane elements. Details on this topic can found in the “DRAGON NARB and RCE Architecture Design” document.

1.2. RCE Software Design

RCE is designed as a stand-alone software that can provide simultaneous resource/path computation services to multiple RCE clients. Every client connects a common RCE server port and interacts with the RCE server through the RCE API. Unlike NARB, RCE is a passive listener to the inter-domain and intra-domain OSPF-TE and does not communicate with other RCE directly. The remaining section provides an overview of the RCE software structure and the RCE path computation process.

1.2.1. Software Structure

Due to the complexity of resource computation, it is desirable to have all the three dimensions of resource and policy information available in a single computation space, i.e., inside RCE. The RCE has the functionalities of both a resource database, including the TEDB, and a PCEN. The PCEN features are similar or comparable to those described in the IETF Path Computation Element Architecture. By eliminating the need for repeated information retrieval from other network elements, path computation cannot only run fast but also avoid some inconsistent resource states. A dedicated, deterministic LSP can be obtained from RCE by a single request. Additional routing constraints carried in the LSP request, e.g., some Service Level Agreement (SLA) requirements and user specified restrictions, are also incorporated into each path computation.

Figure 1 shows the diagrammatic RCE architecture. RCE supports a variety of protocol API in order to collect resource information from other control plane entities. For instance, a specific OSPF API allows RCE to collect OSPF link state information from an OSPF daemon that supports the API. Resource information, such as OSPF LSA, is parsed and stored into TEDB. In addition, local and global AAA rules and LSP schedule information can be exchanged through the RCE API. By cross-referencing to the constraints in the AAA Rules Table and the LSP Schedule Table, the TEDB becomes a 3D TEDB, which provides the input for the 3D CSPF PCEN module to perform 3D path computation. In addition to 3D path computation, RCE provides the computation functions for resource management, LSP scheduling and policy management.

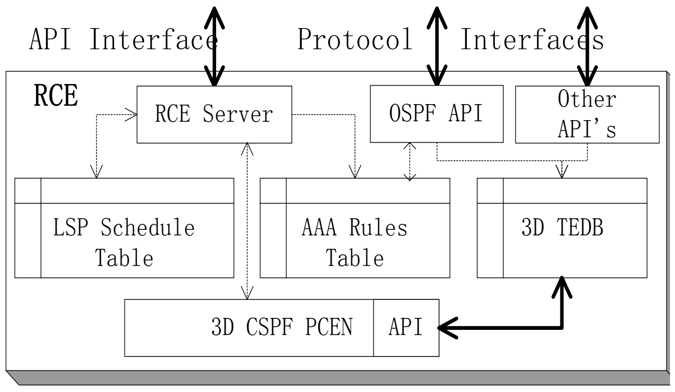

Figure 1: Resource Computation Element (RCE) software structure.

1.2.2. 3D Constrained Path Computation

The diagram in Figure 2 shows our 3D constrained path computation process. Upon the LSP request, RCE checks out all the AAA rules related to this request from the AAA rules table. These AAA rules, plus some user-specified rules carried by the LSP request, are parsed into constraints that instruct RCE to only retrieve the related resources from the TEDB and create a snapshot of the retrieved resource data in the memory.

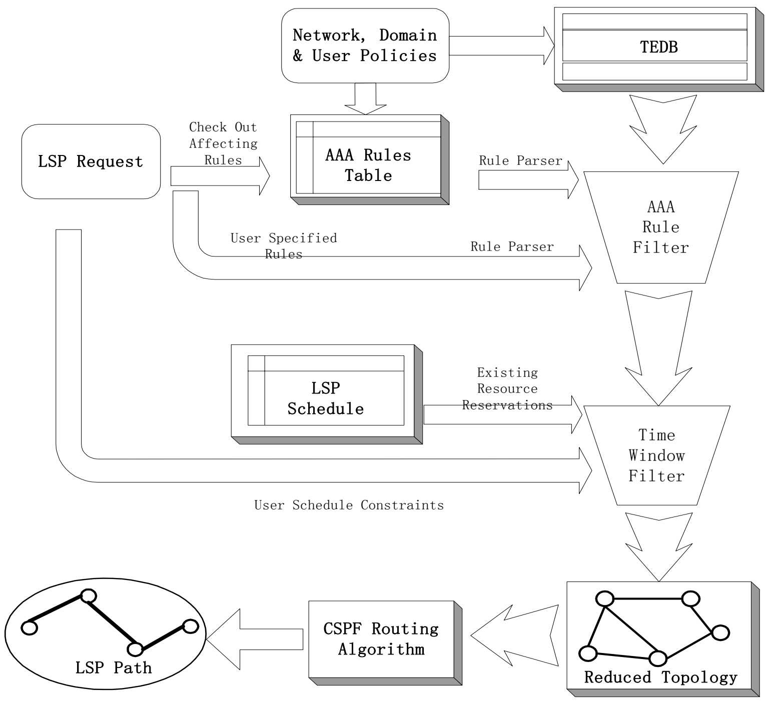

Figure 2: 3D constrained path computation process.

As resource availability is measured by time slots, the retrieved resource information is further constrained by the existing LSP schedule and therefore those time slots already reserved shall be removed. Next, to satisfy the user specified time schedule in the LSP request, RCE filters the retrieved resources using the time schedule constraints as requested. The resulting resources are used to create a reduced network topology. CSPF computation is carried out on this reduced topology to select a routing path. If a fixed time schedule is requested, time filtering and CSPF computation are carried out only once. If a flexible schedule in a given a time window is requested, RCE needs to scan the resources in the whole time window for W times, where W is the window size counted by the number of time slots in that time window. After each scanning, RCE increments the LSP uptime by one time slot, creates an instance of topology and carries out the CSPF computation. This procedure is repeated up to W times until a path is found.

1.2.3. Incorporation of Subnet Control Model

As explained in Appendix A of the NARB and RCE Architecture document, a domain may contain a subnet that is controlled via OIF UNI or vendor-specific API. RCE can obtain dynamic subnet topology information via standard mechanisms such as OSPF-TE, OIF ENNI etc, or via vendor-specific API such as those using TL1, CORBO etc. When none of the dynamic mechanisms is available, DRAGON provides a static topology configuration method that initially loads the subnet topology from configuration files and performs provisioning-related link state updates in the memory. During each path computation process, RCE creates the so-called ‘jump links’ to bridge the subnet to the rest of the domain and/or to inter-domain links. Layer-crossing criteria are observed on these ‘jump links’ such that a path search procedure can enters and leaves the subnet. One may also query for a path between two edge interfaces of the subnet, in which case the ‘jump links’ that correspond to the source and destination *subnet-interface* local IDs will be identified and path constraints will be applied to the ‘jump links.’ When a path with a subnet segment is selected, the VLSRs controlling the switches along the subnet segment will be located and a VLSR-level path segment will be generated to replace the subnet segment to compose an ERO that directs the signaling flow across the subnet controlling VLSRs. The subnet path segment will be stored into a subnet ERO that can be returned together with the main ERO if requested by the client/user.

1.2.4. Inter-domain Path Computation Relay in Multi-Region Networks

NARB/RCE uses the RPD scheme (as described in Section 4.3.2) to compute an inter-domain path. In a Multi-Region (or Multi-Layer) network (MRN/MLN) the process is different than in a single-region network. The path computation conditions and TE constraints on a recursive path segment are usually different from those on the original path. For example, in the two-domain Ethernet over Optical network shown in Figure 3, the original path request to Domain 1 starts from the Ethernet switch A with the bandwidth 300 Mbps, switching type L2SC and encoding type Ethernet. The destination is Ethernet switch F in Domain 2. The recursive path request to Domain 2 changes the starting point into the optical switch LSR2.1 with the new ingress bandwidth 10 Gbps, switching type LSC and encoding type lambda. The egress conditions remain the same.

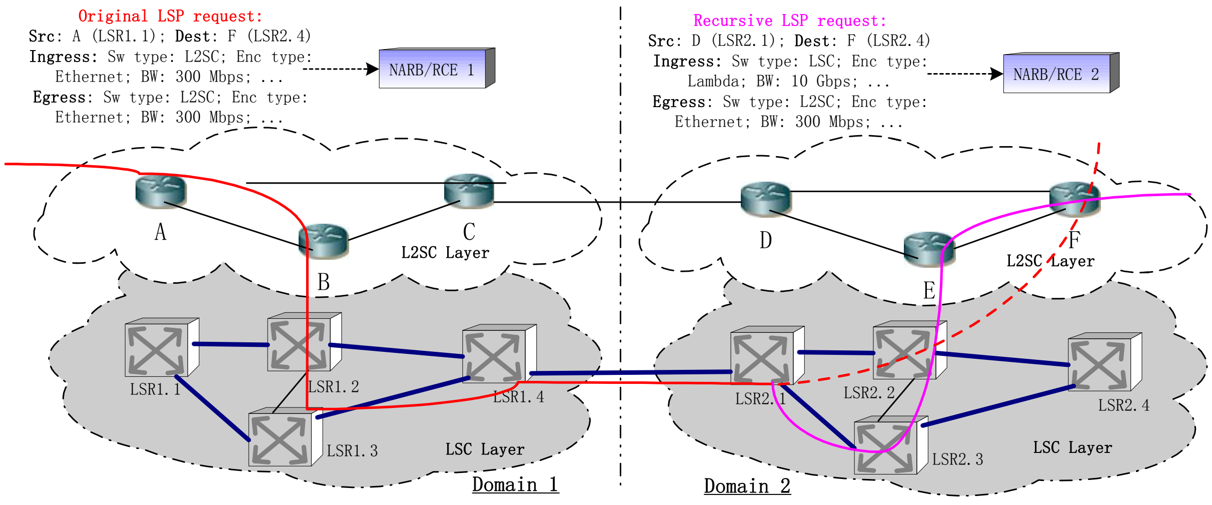
Original LSP request:
Src: A (LSR1.1); Dest: F (LSR2.4)
Ingress: Sw type: L2SC; Enc type: Ethernet; BW: 300 Mbps; ...
Egress: Sw type: L2SC; Enc type: Ethernet; BW: 300 Mbps; ...
NARB/RCE 1
Recursive LSP request:
Src: D (LSR2.1); Dest: F (LSR2.4)
Ingress: Sw type: LSC; Enc type: Lambda; BW: 10 Gbps; ...
Egress: Sw type: L2SC; Enc type: Ethernet; BW: 300 Mbps; ...
NARB/RCE 2
A
B
C
L2SC Layer
D
E
F
L2SC Layer
LSR1.1
LSR1.2
LSR1.3
LSR1.4
LSC Layer
Domain 1
LSR2.1
LSR2.2
LSR2.3
LSR2.4
LSC Layer
Domain 2

Figure 3: Illustration of RCE multi-domain path computation relay in MRN.

To handle the above situation, the RCE path computation in a previous domain will return the constraint context at the point where a recursive path request starts. The RCE in a next domain will use the context parameters sent from the previous domain to relay the path computation process (via NARB). In so doing, the distributed RCEs can collaborate in a multi-domain, multi-region path computation progressively to produce consistent and efficient results.

1.2.5. Resource Holding upon Query

In the Query Holding (Q-Hold) mode, resources will be held for a predefined period of time after a path is successfully returned. Q-Hold allows resource holding and query combined as an atomic operation in order to avoid contention. To implement this feature, the resource *Delta* (Δ) is introduced. A Delta represents a piece/portion of resource data being held for a path. A common Delta data structure is used to describe a variety of resources, including bandwidth, VLAN tag(s), TDM timeslot(s) and wavelength. Each Delta has time attributes for RCE to determine when it is created and whether it has expired. In RCE, each TE link has an associated Delta list to track the resources being held on this link. Under the Q-Hold mode, an LSP query will create a new Delta on every link along the path. The available resources on the links are decremented by the Delta. In other events, e.g. LSA updates, LSP reservation and release, a state-machine based logic will determine how to handle the link resources and associated corresponding Delta lists.

1.3. Example Network Configuration

Figure 4 depicts an example two-domain network configuration. This is a typical configuration that illustrates how the RCE is used in a multi-domain networking environment. There are one NARB, one RCE and two OSPFd's running on the NARB/RCE hosts of both network domains. Each intra-domain OSPFd has adjacency with one of the network node in its local domain via a GRE tunnel. The two inter-domain OSPFd's have adjacency with each other via another GRE tunnel. There are three isolated OSPF flooding areas, generating two intra-domain physical topologies and one inter-domain summarized topology. RCE listens to both the intra-domain OSPFd and the inter-domain OSPFd to maintain a resource database containing both the precise physical resource information in its local domain and the summarized resource information at a global level.

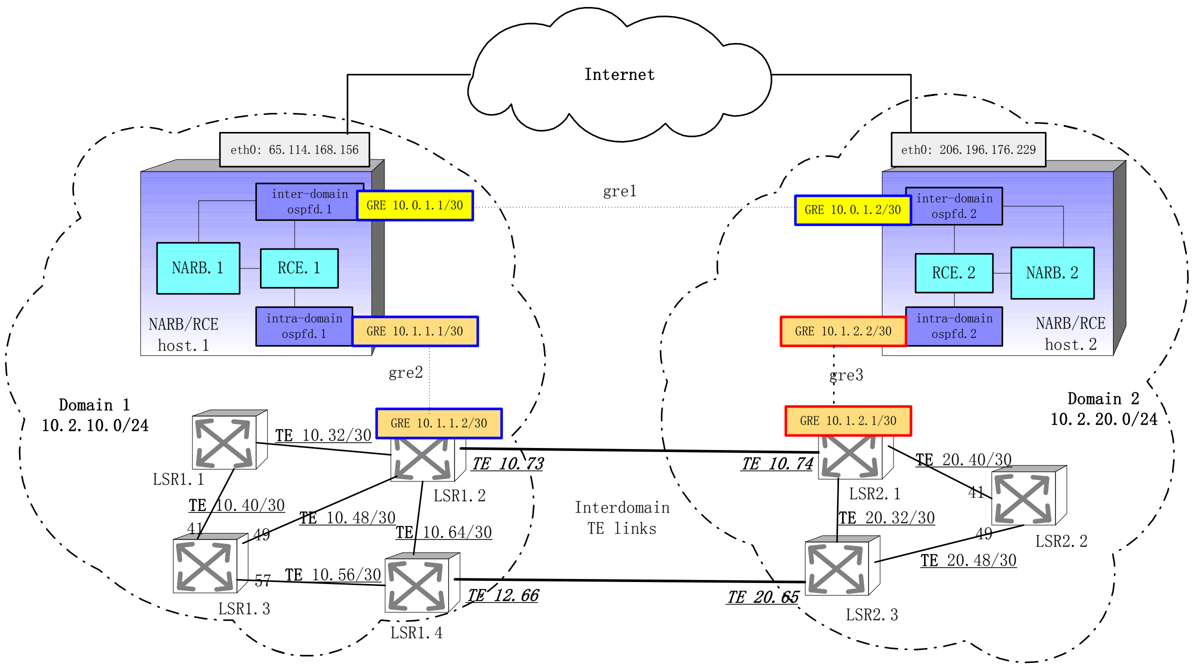

Figure 4: An example two-domain network configuration.

Suppose we want to provision an LSP from LSR 1.2 to LSR 2.2. The DRAGON end system agent (ESA) on the LSR 1.2 will send an LSP query request to NARB.1, which in turn sends a path computation request to RCE.1. RCE.1 may return a hybrid path with precise, physical hops in Domain 1 and rough, abstract hops in Domain 2. NARB.1 then communicates with RCE.2 (through NARB.2) to complement the initial path with precise routing information in Domain 2. In a network with more than two domains, this procedure is carried out recursively until the source ESA on LSR1.2 obtains a full explicit route for the requested end-to-end LSP.

2. Installation Guide

2.1. Pre-installation

2.1.1. Examining Hardware and Operating System Configurations

The RCE software runs on a Linux or FreeBSD operating systems. Hardware requirements include Pentium III CPU or above, at least 512 MB RAM and at least 1 GB free hard disk space. A Linux server or workstation with kernel version 2.4 or above (e.g., RedHat Linux v9) is recommended.

2.1.2. Installing GNU Zebra DRAON Extension

The DRAGON GNU Zebra software can be downloaded from:

http://dragon.east.isi.edu ====> VLSR

A minimum installation of the Zebra OSPFd module is required. Refer to the DRAGON VLSR Implementation Guide for general DRAGON Software installation and configuration instructions.

2.1.3. Downloading the NARB Software Package

The software package that contains both NARB and RCE is available at:

http://dragon.east.isi.edu ====> NARB

2.2. Compilation and Installation

GNU GCC 3.2 or above is required for compilation of the software. Extract the package to a working directory, e.g. /usr/local/narb. To compile and install,

```
#cd narb-sw
#./do_build.sh
#./do_install.sh
```

Read the INSTALL document under the narb directory for more installation information. To run the software, narb.conf, rce.conf and schema_combo.rsd must exist under the configuration directory, which is /usr/local/dragon/etc by default. Sample narb.conf, rce.conf and schema_combo.rsd files are installed under the configuration directory. More details on narb.conf are available in the next section. schema_combo.rsd can be literally copied from schema_combo.rsd.sample for most users to begin with. The installation is under /usr/local/dragon/. To run both NARB and RCE, use following command

```
#/usr/local/dragon/bin/run_narb.sh
```

To compile, install and run the RCE software alone, do the following.

```
#cd narb-sw/rce
#./configure
#make
#make install
#cp /usr/local/dragon/etc/rce.conf.sample /usr/local/dragon/etc/rce.conf
#cp /usr/local/dragon/etc/schema_combo.rsd.sample /usr/local/dragon/etc/schema_combo.rsd
#/usr/local/dragon/bin/rce -d -f /usr/local/dragon/etc/rce.conf
```

3. Configuration Guide

In a real network, the following configuration must be done before we actually run the RCE software. We use Figure 4 as the reference network configuration.

3.1. Preparing Control Plane Network

By default, both intra- and inter-domain instances of OSPFd are hosted on the RCE host. The inter-domain OSPFd listens to API port 2607 and the intra-domain OSPFd listens to API port 2617.

The OSPF adjacency between the intra-domain OSPFd on the RCE host and the OSPFd on one of operational network nodes in the local domain should be created. In addition, the OSPF adjacency between the inter-domain OSPFd and its peers in adjacent domains should be created. (Note that in the DRAGON architecture, NARB is needed to summarized and originate the higher-level topology to the inter-domain OSPFd.)

With reference to the configuration depicted in Figure 4, the following GRE tunnels are created to provide inter- and intra-domain OSPF adjacency.

gre1 (10.0.1.0/30): inter-domain OSPF adjacency between the two NARB/RCE peers.
gre2 (10.1.1.0/30): intra-domain OSPF adjacency between RCE.1 and Domain 1.
gre3 (10.1.2.0/30): inter-domain OSPF adjacency between RCE.2 and Domain 2.

3.2. Customizing Configuration File

The RCE must read rce.conf to start. By default rce.conf is located under /usr/local/dragon/etc/ or the current directory. rce.conf contains the following configuration blocks.

Domain ID Configuration

domain-id { ip *IP* | id *NUM* | asn *ASN*}

Domain ID is used to associate each advertised inter-domain router or link with a unique domain. A domain ID can be expressed as an IP address, an unsigned integer number or an AS number (single number or in A.B format). Note that the domain-id configured in rce.conf must match up with that configured in narb.conf.

TEDB Scheme Configuration

```
include-tedb-schema {path /usr/local/dragon/etc/schema_combo.rsd}
```

The above block is mandatory in rce.conf. The path may be changed if the schema file is relocated.

Subnet Topology Configuration

```
include-subnet-topology {path /usr/local/dragon/etc/subnet.conf}
```

This optional configuration block is used to read a static topology file that represents a subnet embedded within the domain. This configuration is not needed in most configuration cases. Details of the subnet topology configuration file are provided in Section 3.4.

Holding Times

```
holding-time {
    query-expire-seconds 30
    reserve-expire-seconds 0
    subnet-reserve-expire-seconds 86400
}
```

This optional configuration block set the holding times for path query, path reservation and subnet path reservation.

Ethernet-Over-SONET Bandwidth Mapping

```
ethernet-bandwidth-to-sonet-timeslots-mapping {
        map 100 to 2
        map 1000 to 21
        map 10000 to 191
}
```

By default every 50Mbps Ethernet bandwidth is mapped into one TDM STS-1 timeslot. This optional configuration block overrides the mappings for some Ethernet bandwidth values.

3.3. Reconfiguring TEDB Structure

As configured in rce.conf, RCE needs to read a TEDB schema file, which uses XML to define the resource (link) TE parameters that will be stored into the TEDB in RCE. A sample RCE TE schema file *schema_combo.rsd.sample* is included in the narb/rce package. The purpose of RCE TE schema is to provide the add-on flexibility that allows users to define their proprietary TE parameters. For example, some user needs to incorporate a physical-layer parameter, say optical filer dispersion, into the TE database. Without changing data structure in the code, one can simply do that by adding one line into *schema_combo.rsd*.

A sample RCE TE scheme file looks like the below.

```
<SCHEMA VERSION="1.0">
<RESOURCE DOMAIN="GLOBAL" TAG="LSA/OPAQUE/TE/LINK" RID="1.10.1.2">
   <TLV TYPE="1" LENGTH="1" DATA_TYPE="U_INT8" TAG="LINK_TYPE"/>
   <TLV TYPE="2" LENGTH="4" DATA_TYPE="IPV4" TAG="LINK_ID"/>
   <TLV TYPE="3" LENGTH="4" DATA_TYPE="IPV4" TAG="LOCAL_INTERFACE"/>
   <TLV TYPE="4" LENGTH="4" DATA_TYPE="IPV4" TAG="REMOTE_INTERFACE"/>
   <TLV TYPE="5" LENGTH="4" DATA_TYPE="NUMBER" TAG="METRIC"/>
   <TLV TYPE="6" LENGTH="4" DATA_TYPE="FLOAT" TAG="MAX_BANDWIDTH"/>
   <TLV TYPE="7" LENGTH="4" DATA_TYPE="FLOAT" TAG="MAX_RESV_BANDWIDTH"/>
   <TLV TYPE="8" LENGTH="32" DATA_TYPE="FLOAT" TAG="UNRESV_BANDWIDTH"/>
   <TLV TYPE="9" LENGTH="4" DATA_TYPE="NUMBER" TAG="ADMIN_GROUP"/>
   <TLV TYPE="11" LENGTH="8" DATA_TYPE="IPV4" TAG="LOCAL_REMOTE_IDS"/>
   <TLV TYPE="14" LENGTH="4" DATA_TYPE="NUMBER" TAG="PROTECTION_TYPE"/>
   <TLV TYPE="15" LENGTH="40" DATA_TYPE="LIST" TAG="ISCD"/>
   <TLV TYPE="16" LENGTH="4" DATA_TYPE="LIST" TAG="SRLG"/>
   <! Defining proprietary TE parameters - begin>
   <TLV TYPE="16641" LENGTH="4" DATA_TYPE="U_INT32" TAG="DRAGON_LAMBDA"/>
   <! Defining proprietary TE parameters - end>
</RESOURCE>
```

The TLV types 1 through 16 are standard TE parameters that are defined by RFC 3630 and RFC 4203. The standard TE parameters are sufficient for most users. The above example contains a DRAGON proprietary TLV *DRAGON_LAMBDA*. Handling such proprietary TE parameters needs to upgrade the path computation algorithm, which involves source code changes and is beyond the scope of this document.

3.4. Subnet Topology Configuration File

Optionally configured in the rce.conf is a path to the subnet topology configuration file. This file is used when the current domain contains a subnet and has to create a subnet topology from static configuration (when dynamic topology update mechanism is missing). The syntax of subnet topology configuration is very similar to that of NARB abstract topology configuration. An example one-link topology is described below.

```
!
router {id 149.165.129.17
      home_vlsr 10.100.70.223
      dtl_name SEAT
```

```
        link {id 129.79.216.69 type 1 dtl_id 129
                max_bw 10000.0 max_rsv_bw 10000.0
                unrsv_bw0 10000.0 unrsv_bw1 10000.0 unrsv_bw2 10000.0 unrsv_bw3 10000.0 unrsv_bw4
                        10000.0 unrsv_bw5 10000.0 unrsv_bw6 10000.0 unrsv_bw7 10000.0
                enc_type 5 sw_type 100
                metric 100
                local_if 140.173.101.37  remote_if 140.173.101.38
        }
}
!
router {id 129.79.216.69
        home_vlsr 10.100.70.233
        dtl_name GRNOC
        link {id 149.165.129.17 type 1  dtl_id 129
                max_bw 10000.0 max_rsv_bw 10000.0
                unrsv_bw0 10000.0 unrsv_bw1 10000.0 unrsv_bw2 10000.0 unrsv_bw3 10000.0 unrsv_bw4
                        10000.0 unrsv_bw5 10000.0 unrsv_bw6 10000.0 unrsv_bw7 10000.0
                enc_type 5 sw_type 100
                metric 100
                local_if 140.173.101.38  remote_if 140.173.101.37
        }
!
```

In the RCE subnet topology configuration, the additions to the NARB abstract topology configuration syntax are as follows.

router: home_vlsr:	specifies the router ID of the controlling VLSR for this switch node
router: dlt-name:	specifies the native name of the switch in the subnet
link: dlt-id:	specifies the native link ID of a link originating from a subnet switch

These fields are used by RCE to associate subnet path segment with the VLSR-level path and to generate vendor-specific form of subnet ERO, such as the Designated Transit List (DTL).

3.5. *Single-Domain Configuration*

Figure 4 has illustrated a multi-domain network configuration. RCE is also useful in a single domain to support sophisticated routing constraints in a multi-region/multi-layer network. Figure 5 shows two scenarios for RCE run in a single domain. In the first scenarios path computation is requested through NARB. In the second scenario, a third-party agent with RCE API client can direct request for a path computation from RCE. Description of RCE API will be presented in Section 5.

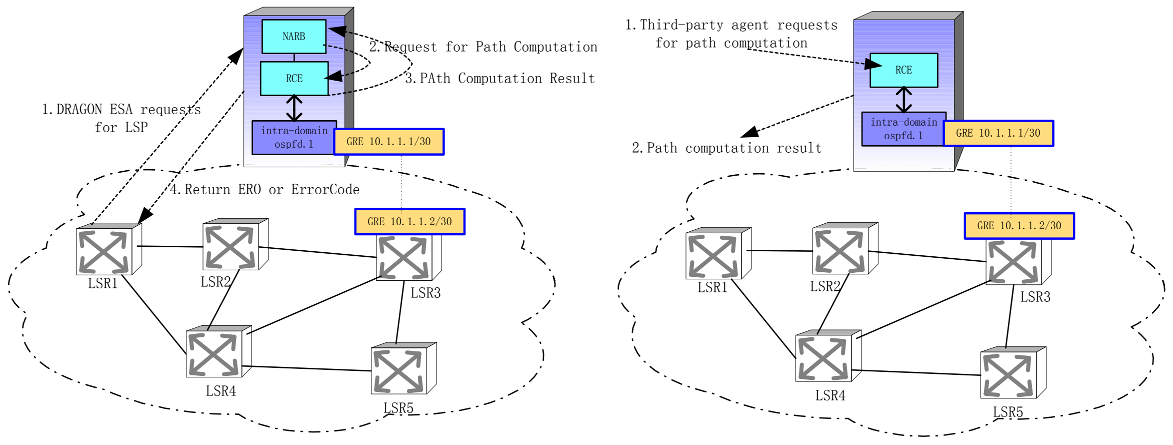
NARB
RCE
intra-domain ospfd.1
GRE 10.1.1.1/30
2.Request for Path Computation
3.PAth Computation Result
1.DRAGON ESA requests for LSP
4.Return ERO or ErrorCode
GRE 10.1.1.2/30
LSR1
LSR2
LSR3
LSR4
LSR5
1.Third-party agent requests for path computation
RCE
intra-domain ospfd.1
GRE 10.1.1.1/30
2.Path computation result
GRE 10.1.1.2/30
LSR1
LSR2
LSR3
LSR4
LSR5

Figure 5: Use of RCE in a single domain network.

4. Command Line Interface (CLI) Reference

RCE CLI server listens to port 2688 for telnet connections. After logged in to RCE CLI via telnet, one is under the View mode with "rce:cli>" as the shell prompt.

4.1. Commands in the View mode

rce:cli> *show module*

Displays addresses, ports and status of the inter- and intra-domain instances of OSPFd and the local NARB (if applicable). Status of OSPFd is two-fold: (a) whether the OSPFd is online; (b) whether the API client connection to OSPFd is alive.

rce:cli> *show topology (interdomain | intradomain)*

Displays all the RouterId and TE link LSAs in the inter- or intra-domain flooding areas respectively.

rce:cli> *show link (interdomain | intradomain) local_if_addr IP remote_if_addr IP*

Displays details of a TE link identified by the local and remote interface IP addresses. The information displayed includes bandwidth, switching capability, metric etc as well the resource holding *Delta* list.

rce:cli> *exit*

Logs off the RCE CLI.

rce:cli> *configure*

Enters the Configuration mode. The shell prompt changes into "rce:cli#".

4.2. Commands in the Configuration mode

rce:cli# *load-config FILE*

Load and execute arbitrary CLI commands saved in FILE.

rce:cli# *set ospfd (interdomain | intradomain) HOST LCL_PORT RMT_PORT ORI_IF AREA*

Reconfigure inter- or intra-domain OSPFd parameters.

rce:cli# *connect ospfd (interdomain | intradomain)*

(Re)Connect to an OSPF daemon. Upon success of connection, resource database will be immediately synchronized to the OSPFd.

rce:cli# *exit*

Exit from the topology configuration mode.

5. Application Programming Interface (API) Reference

RCE provides an API interface between the user application client and the RCE server. RCE API server listens to port 2678. A RCE API server can accommodate multiple RCE API clients simultaneously.

5.1. API Message Format

The RCE API message format is shown in Figure 6.

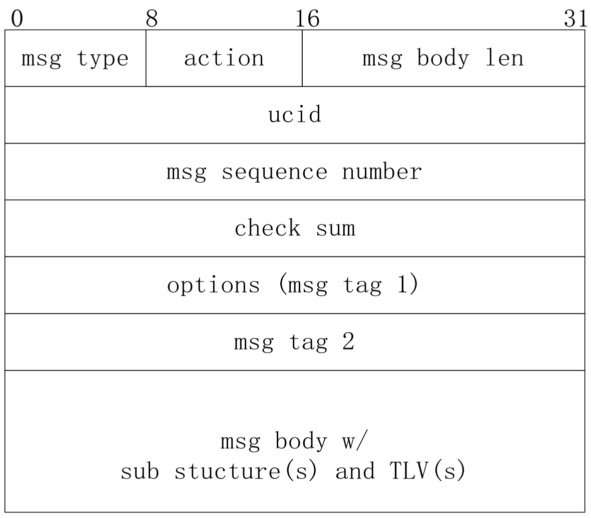

Figure 6: RCE API message format.

Message Type

RCE_MSG_LSP (0x01): LSP query, i.e., path computation request
RCE_MSG_LSA (0x02): Link state advertisement (LSA) update
RCE_MSG_AAA (0x03): Reserved for AAA related communication
RCE_MSG_RM (0x04): Reserved for resource management
RCE_MSG_CTRL (0x05): Reserved for other controlling activities

Action

This is the message sub-type to indicate the action the RCE is to perform.
ACT_NOOP (0x00): Empty message; doing nothing.
ACT_QUERY (0x01): Query an LSP, a resource, an AAA rule etc.
ACT_INSERT (0x02): Insert an LSA, a resource state, an AAA rule, etc.
ACT_DELETE (0x03): Delete an LSA, a resource state, an AAA rule, etc.
ACT_UPDATE (0x04): Update an LSA, a resource state, an AAA rule, etc.
ACT_ACK (0x05): Acknowledgement from RCE.
ACT_ACKDATA (0x06): Acknowledgement from RCE with data.
ACT_ERROR (0x07): Error code from RCE.
ACT_QUERY_MRN (0x10): Query an LSP in the multi-region network mode.

Message Body Length

The length of message body in number of octets and must be divided by 4.

Universal Client ID (UCID)

A 32-bit ID number to uniquely identify a RCE API client, which is used to track LSP owner.

Message Sequence Number

The number can be used to identify the LSP query context a message belongs to.

Check Sum

The arithmetic sum of the first three 32 bit words

Options (or Message Tag 1)

A bit mask to indicate client-specific requirements, here are explanations for some bits.
0x0000,0001: Using standard NARB CSPF request parameter format
0x0000,0002: Requesting all available VLAN tags.
0x0000,0004: Requesting all available VLAN wavelengths.
0x0000,0010: Excluding layer-1/optical-layer from path computation.
0x0000,0020: Excluding TDM layer from path computation.
0x0000,0040: Excluding layer-2/Ethernet-layer from path computation.

0x0000,0080: Excluding layer-3/IP-layer from path computation..
0x0001,0000: Request for *all-strict* hop path; otherwise, *loose*.
0x0002,0000: The above *all-strict* or *loose*-hop path is *preferred*; otherwise, a *must*
0x0010,0000: *Bidirectional* request; otherwise, *unidirectional* request
0x0020,0000: Using VLAN tag constraint for layer 2 LSP (*tagged E2E VLAN*); otherwise, an untagged E2E VLAN if applicable.
0x0080,0000: A VLAN tag bit mask is passed as extra constraint.
0x0100,0000: Client requesting Query Holding (Q-Hold) mode.
0x0400,0000: Returning Subnet ERO if any.
0x0800,0000: Returning Subnet DTL if any.

Message Tag 2
A 32-bit tag; when used with the 0x00200000 option, is the requested VLAN tag to constrain a layer-2 LSP (0: no tag, the default value; 0xffff: any tag; otherwise: a specified tag in [1, 4095])

5.2. LSA Update/Delelte Messages

The message type is RCE_MSG_LSA. The actions to add or update an LSA to the resource database are ACT_UPDATE and ACT_DELETE respective. The message body contains an OSPF LSA whose format is defined in the IETF OSPF standards.

5.3. LSP Computation Messages

The LSP computation request and replies are described as follows.

5.3.1. Client LSP Computation Request

The message type and action are RCE_MSG_LSP and ACT_QUERY respectively. The message body contains the following structure.

```
struct narb_lsp_request_tlv {
   u_int16_t type;
   u_int16_t length;
   struct in_addr src;
   struct in_addr dest;
   u_int8_t  encoding_type;
   u_int8_t  switching_type;
   u_int16_t gpid;
   float bandwidth;
};
```

5.3.2. Peer LSP Computation Request

The message type and action are RCE_MSG_LSP and ACT_QUERY_MRN respectively. The message body contains the two *narb_lsp_request_tlv* structures.

```
struct narb_lsp_request_tlv ori_spec_tlv;
struct narb_lsp_request_tlv mrn_spec_tlv;
```

This request is used in a multi-region multi-domain network. The *ori_spec_tlv* contains the original LSP request parameters from the source domain. The *mrn_spec_tlv* contains the LSP request parameters passed between adjacent domains to relay a recursive path computation process.

5.3.3. LSP Reply with ERO

The message type and action are RCE_MSG_LSP and ACT_ACK_DATA respectively. The message body contains the following TLV.

```
struct ero_reply_tlv {
        u_int16_t type;
        u_int16_t length;
        struct ero_subobj subobjs[N];
    };
```

The TLV body contains a sequence of *N* ERO subobjects, each being defined as follows.

```
    struct ero_subobj {
        struct in_addr addr;
        u_char hop_type;
        u_char prefix_len;
        u_char pad[2];
        u_char sw_type;
        u_char encoding;
        union {
            u_int16_t lsc_lambda;
            u_char tdm_indication;
            u_int16_t l2sc_vlantag;
            u_int16_t psc_mtu;
        };
        float bandwidth;
    };
```

This ERO subobject structure is a DRAGON-specific definition that allows RCE to return more information than the standard RFC one. This message may optionally return either a Subnet ERO TLV or a Subnet DTL TLV (Sections 5.4.4 and 5.4.5) when the domain has a subnet and the user/client has requested to see the subnet ERO or DTL.

5.3.4. LSP Reply with Error Code

The message type and action are RCE_MSG_LSP and ACT_ERROR respectively. The message body contains the following TLV.

```
    struct ero_reply_tlv {
        u_int16_t type;
        u_int16_t length;
        u_int32_t error_code;
    };
```

5.3.5. LSP Reservation, Release and Update Confirmation

These messages are of RCE_MSG_LSP type with ACT_CONFIRM, ACT_DELETE and ACT_UPDATE actions respectively. The messages are sent from client to RCE with the body contains an ERO TLV. The messages are mostly used for client to notify RCE about state change events in LSP provisioning. These events will trigger resource Delta state-machine operations.

This message may optionally carry either a Subnet ERO TLV or a Subnet DTL TLV (Sections 5.4.4 and 5.4.5), in which case RCE is also notified about link state changes in the subnet topology.

5.4. Optional TLVs

Several optional TLVs are used together with the above API messages.

5.4.1. LSP Broker ID TLV

The TLV type is TLV_TYPE_NARB_LSPB_ID (0x08). The value field contains a 32-bit unsigned number.

```
    struct narb_lsp_lspb_id_tlv
    {
        u_int16_t type;
        u_int16_t length;
        u_int32_t lspb_id;
    };
```

An LSP broker ID is a unique identifier for the RCE client, usually a NARB that is paired with the RCE. When NARB passed an LSP Broker ID TLV to RCE, the same TLV is returned to NARB along with the ERO or Error Reply Messages

5.4.2. VLAN Tag Mask TLV

The TLV type is TLV_TYPE_NARB_VTAG_MASK (0x05). The value field is a 4096 bit string stored in a 512 bytes data array.

```
struct narb_lsp_vtag_mask_tlv
{
    u_int16_t type;
    u_int16_t length;
    u_char bitmask[MAX_VLAN_NUM/8];
};
```

VLAN Tag Mask TLV is mandatory when the message option bit 0x00800000 is set. It is passed along with LSP Query messages to narrow down the range of VLAN tags for RCE path computation.

5.4.3. Hop Back Interface TLV

The TLV type is TLV_TYPE_NARB_HOP_BACK (0x06). The value field contains an IPv4 address.

```
struct narb_lsp_hop_back_tlv
{
    u_int16_t type;
    u_int16_t length;
    u_int32_t ipv4;
};
```

A Hop Back TLV is used when signaling proceeds to the next-domain border router and needs to expand a loose hop ERO by contacting the next-domain NARB. When NARB receives a request with the Hop Back TLV, it forwards the TLV to RCE. More explanation about this TLV can be found in the “NARB Design and User Manual” document.

5.4.4. Subnet ERO TLV

The TLV type is TLV_TYPE_NARB_SUBNET_ERO (0x09). The value field contains an ERO with the same format as that of the *ero_reply_tlv*. The Subnet ERO TLV is returned when RCE has computed a path across a subnet and the client/user wants to know the subnet physical hops (in addition to the VLSR-level signaling hops in the *ero_reply_tlv*.).

5.4.5. Subnet DTL TLV

The TLV type is TLV_TYPE_NARB_SUBNET_DTL (0x0B). The value field contains a Subnet Designated Transit List which is the equivalent of a subnet ERO.

```
struct narb_lsp_subnet_dtl
{
    u_int16_t type;
    u_int16_t length;
    dtl_hop hops[1]; //actual number of dtl_hops depends on length
};
```

Each DTL hop contains a node name and outgoing link id in the following format.

```
struct dtl_hop
{
   u_int8_t nodename[20]; //19-char C string;
   u_int32_t linkid;  //link ID number
};
```

Both Subnet ERO and DTL TLVs should not be requested and returned within the same message.

5.4.6. Local-ID TLV

The TLV type is TLV_TYPE_NARB_LOCAL_ID (0x0A). The value field contains a pair of 32-bit source and destination local IDs. When one is missing, its value is zero.

```
struct msg_narb_local_id
{
   u_int16_t type;
   u_int16_t length;
   u_int32_t lclid_src;
   u_int32_t lclid_dest;
};
```

RCE usually only apply constraints to intermediate links along a path. When Local IDs are passed, RCE will try to apply the constraints to the edge interfaces represented by the local ID(s). RCE will also perform resource holding on the edge interfaces if Q-Hold is requested.

6. Diagnostics and Troubleshooting

6.1. RCE_Test Client

The RCE test client is a utility software within the narb/rce package to perform some basic API client functions. It can send an LSP computation request to and receive the reply from a RCE server and display the path computation result. For instance, the following command sends a request to the RCE server on mcln-hopi-narb, at port 2678, for an LSP from the source 10.100.40.233 to the destination 10.100.10.233. The constraints include bandwidth 1000 Mbps (-b1000), L2SC switching type (-x51), Ethernet encoding type (-e2), bidirectional (-B) and VLAN tag 3020 (-v3010).

```
#/usr/local/narb/rce_test -H mcln-hopi-narb -P 2678 -S 10.100.40.233 -D 10.100.10.233 -b1000 -x51 -e2 -B -v3010
```

The result is displayed as follows, which shows a successful query reply with a full path.

```
@[2006/05/04 11:27:28] : Request successful! ERO returned...
 @[2006/05/04 11:27:28] : HOP-TYPE [strict]: 10.100.30.38
 @[2006/05/04 11:27:28] : HOP-TYPE [strict]: 10.100.30.37
 @[2006/05/04 11:27:28] : HOP-TYPE [strict]: 10.100.20.46
 @[2006/05/04 11:27:28] : HOP-TYPE [strict]: 10.100.20.45
 @[2006/05/04 11:27:28] : HOP-TYPE [strict]: 10.100.15.38
 @[2006/05/04 11:27:28] : HOP-TYPE [strict]: 10.100.15.37
 @[2006/05/04 11:27:28] : HOP-TYPE [strict]: 10.100.10.38
 @[2006/05/04 11:27:28] : HOP-TYPE [strict]: 10.100.10.37
 @[2006/05/04 11:27:28] : E2E VLAN TAG [ 3010 ]
```

For complete usage of the RCE_Test utility software, check out */usrlocal/dragon/sbin/rce_test.*

```
RCE Testing Client Usage:

    rce_test [-H host] [-P port] [-S source] [-D dest] [-U] [-b bandwidth] [-x switching type] [-e encoding type] [-v vtag] [-V] [-a] [-Q]
```

where [-S source] and [-D destination] are mandatory [-U]: unidirectional [-v]: E2E VLAN with specified tag [-V]: any E2E VLAN with tag picked by RCE [-a]: returning all the available VLAN tags [-Q]: client requesting Q-Hold

Upon failure, an error message will be displayed.

```
#./rce_test -H mcln-hopi-narb -P 2678 -S 10.100.70.233 -D 10.100.10.233 -b1000 -v3010
@[2006/05/04 11:33:30] :  Request failed : Unknown Source Address
```

The below is a list of possible error messages.

"Unknown Source Address"
"Unknown Destination Address"
"No Routing Path Found"
"Invalid Path Request"
"System Warming Up"
"Max. Retransmission of Request Exceeded"

The first three errors could happen when some constraint cannot be satisfied or some node or link is missing in the global or local topology.

"Invalid Path Request" usually means the request has unrealistic parameters/options or an API message packet is corrupt, i.e., wrong length, wrong checksum or wrong format.

When "System Warming Up" is present, just wait for up to one minute for the system to finish initiation or transition.

6.2. Log File

More information can be obtained from the system log file in /var/log/rce.log. From the log file, for example, one can tell whether an API client connection has been accepted and what TE parameters in an LSA are not recognized.

7. Conclusion

This document presents an installation and configuration guide to use the RCE and related software. It intends to provide necessary information for users to deploy a multi-domain, multi-layer GMPLS network control plane successfully. In this document, the RCE software is used in the context of the DRAGON network, particularly to provide path computation service to NARB. However, users can easily use RCE as a standalone module or integrate it into their control plane implementation to provide sophisticated path computation services. Those interested in deploying the NARB and RCE components in their networks or in the DRAGON control plane in general should further refer to the following documents available at http://dragon.east.isi.edu.

- DRAGON Control Plane Overview
- NARB and RCE Architecture
- NARB Design and User Manual
- RCE Design and User Manual
- VLSR Implementation Guide
- DRAGON and HOPI Network Control Plane Guide

8. DRAGON Project

This document was generated by the University of Southern California (USC) Information Sciences Institute (ISI) as part of the National Science Foundation funded Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON) project. Additional details regarding this project are available at http://dragon.east.isi.edu.