“Package” description

In the package we find the following files:

1. agent_rsvp
2. agent_client.py
3. conf.txt
4. the agent's management protocol description

File 1. - agent_rsvp

Let's get to the most important parts first. The first file is the RSVP Agent itself. It is a binary file – just run it and it works. But make sure you run it with root priviledges, otherwise it will not be able to create a raw socket, which is required to send RSVP packets.

The only keyboard commands you should know are:

- “Ctrl + \” - it makes the agent print the information about current paths. This info is printed every period specified in the conf.txt file, but sometimes you want to specify an hour in the file and have the information “on demand”. That's what it is for.
- “Ctrl + C” - just terminates the agent. However, the agent will NOT tear down the paths if you terminate it this way!

The agent can create and maintain a many working paths in parallel.

File 2. - agent_client.py

This is a small application which we use for debugging, experiments, etc. It is a simple, python implementation of the management protocol (the one described in Deliverable 2.4 along with information about RSVP Agent's requirements) which is capable of creating, monitoring and destroying the paths. If you run it without arguments, you will get a short output describing the possible parameters:

```
uni-mupbed:~/Projects/agent_client # ./agent_client.py
------------------------------------------------------------------------
The agent client usage:

1) Path creation:    ./agent_client.py create -tun_id <tunel_id> -dst <dest_addr>
[other parameters,...]
2) Path deletion:    ./agent_client.py delete -tun_id <tunel_id>
3) Status query:     ./agent_client.py status [-tun_id <tunel_id>]

Available parameters:
  -call_id_1      <int>         call id - part 1
  -call_id_2      <int>         call id - part 2
  -dst            <ip_addr>     rsvp session destination address
  -dst_tna        <ip_addr>     destination TNA
  -enc            <int>         lsp encoding type
```

```
  -ero            <ip_addr>     ERO address
  -ero_pref_len   <int>         ERO prefix length
  -ero_sl         <int>         ERO strict/loose
  -err_code       <int>         error code
  -err_flag       <int>         error flags
  -err_node       <ip_addr>     error node address
  -err_val        <int>         error value
  -eth_cbr        <int>         ethernet committed burst size
  -eth_cir        <int>         ethernet committed information rate
  -eth_ebs        <int>         ethernet excess burst size
  -eth_eir        <int>         ethernet excess information rate
  -eth_granuality <int>         type of ethernet link
  -eth_index      <int>         index of bandwidth allocated for given QoS
  -eth_profile    <int>         flags of color mode and coupling
  -gpid           <int>         lsp gpid
  -hold_prior     <int>         hold priority
  -hop            <ip_addr>     RSVP Hop address (deprecated)
  -interface_id   <int>         data forward/reverse interface id
  -lsp_id         <int>         sender LSP id
  -mpart_count    <int>         multiple part message - total packet count (N)
(STATUS ACK/NACK only)
  -mpart_number   <int>         multiple part message - packet number (0..N-1)
(STATUS ACK/NACK only)
  -name           <string>      path name
  -out_if_addr    <ip_addr>     agent's outgoing interface (HOP neighbor address)
  -path_state     <int>         path state (STATUS ACK only)
  -peak_rate      <int>         peak data rate
  -refresh        <int>         path refresh period
  -service_lvl    <int>         protection service level
  -setup_prior    <int>         setup priority
  -src            <ip_addr>     rsvp session source address (extended tunnel id)
  -src_tna        <ip_addr>     source TNA
  -switch_type    <int>         lsp switching type
  -tb_rate        <int>         token bucket rate
  -tb_size        <int>         token bucket size
  -traff_dst      <ip_addr>     traffic destination address (for data plane only)
  -traff_rev_dst  <ip_addr>     traffic reverse destination address (for data
plane only)
  -tun_id         <int>         tunnel id
  -up_label       <int>         upstream label (for bidirectional paths)
-------------------------------------------------------------------------
```

Of course, make sure you have the agent running when you run the client. Also be aware of the firewall issues, as they communicate with TCP/IP. This client always connects to 127.0.0.1 port 35000, so you should run both the agent_client and RSVP Agent on the same machine.

Example of usage:

```
./agent_client.py create -tun_id 1 -dst 10.131.12.9
./agent_client.py status
./agent_client.py status -tun_id 1
./agent_client.py delete -tun_id 1
```

Many agent clients can connect to the agent simultaneously but they should connect from different IP address. Each client can create and maintain his own paths separetly from each other. You can't see status of path of another clients. Be aware that all clients use the same tunnel_ID space so two clients can't create a path with the same tunnel_ID value.

File 3. - conf.txt

This is the configuration file for RSVP Agent. The agent reads some info about how it should behave from this file. It is a text file so feel free to look into it and changing parameters.

One section of config have the following fields:

```
/* a protocol implementaction version: */
protocol_vers           = OIFUNI20          # IETF_RSVP, OIFUNI10, OIFUNI20

/* an eth interface used by the agent_rsvp */
rsvp_local_if_addr      = 10.135.135.254    # ip_addr

/* an interface name used by the agent_rsvp */
rsvp_local_if_name      = eth0              # interface_name

* a cooperative gmpls router (UNI-N) */
rsvp_router_addr        = 10.131.12.9       # ip_addr

 /* a reporting interval of rsvp agent state to screen */
report_paths_interval   = 3600              # sec

/* an ip address of node to which RSVP message copies will be sended; nice to
have when IPSEC is used; uncoment with '#' if you do not want to use it */
debug_send_addr         = 150.254.160.219   # ip_addr

/* how long agent is waiting for RSVP RESV after sending initial RSVP PATH */
path_setup_timeout      = 3                 # sec

/* TCP listingin port for API management protocol
management_port         = 35000             # TCP port number
```

File 4. - the agent's management protocol description

Enhacement description of agent management protocol.