A RPKI RTR Client C Lib (RTRlib) - Implementation Update

Fabian Holler, Thomas C. Schmidt, and Matthias Wählisch

holler_f@informatik.haw-hamburg.de

{t.schmidt, waehlisch}@ieee.org

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Freie Universität Berlin

Background

General objective:

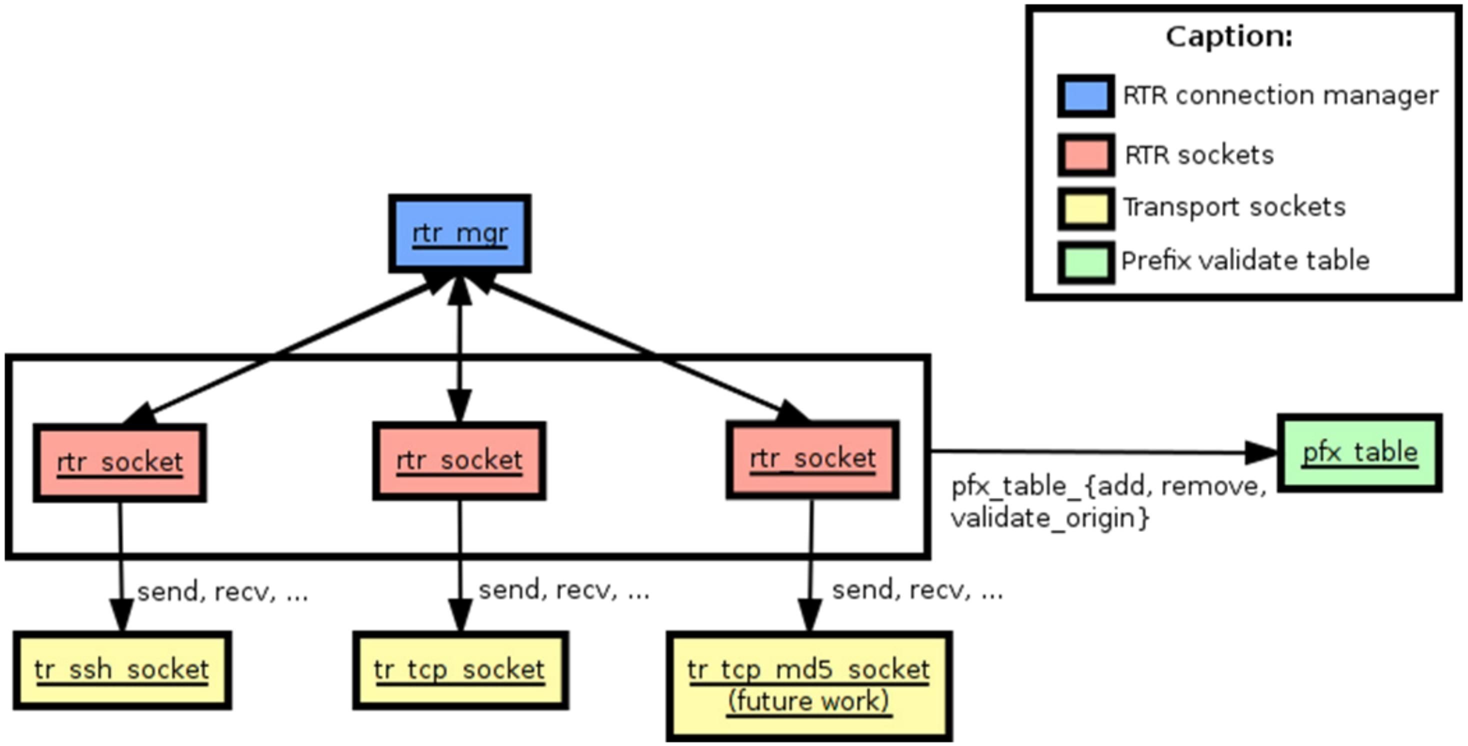
- Implement RPKI-RTR client protocol in C
  - Integrate RTRlib into BIRD
    to allow for prefix origin validation

Timeline so far:

- First idea announced @ IETF80
- We still working on the beta version …

Architectural Design

- Layered architecture to support flexibility



Transport Socket

- Implements a specific transport protocol
- Allows for RTR connections to abstract from transport
  - See TCP-AO vs. MD5 vs. SSH discussion
- Current implementations: Plain TCP, SSH

Functions:

- open, close, recv(_all), send(_all), …

RTR Socket

- Implements the RTR protocol
- Refers to a single transport socket
- Provides functions to fetch and store validation records
- State change invokes a configurable function pointer

Functions:

- init, start, restart, stop, …

PFX Table

- Abstract data structure to store validated prefix origin data received from cache server
- Accessible via a common interface
  - Allows for easy exchange of specific implementation
- Current implementation: Longest Prefix First Trees (separate instances for IPv4 and IPv6 prefixes)

Functions:

- add, remove, remove_from_origin, validate_origin..

RTR Connection Manager

- Maintains connections to multiple RTR servers and stores received data
- Implements preference & failover mechanisms
- Main interface for users

Functions:

- init, start, stop, validate, …

Example – Establish Transport

```
tr_socket* ssh_socket;                //create a SSH connection
tr_ssh_config config = {
   "123.321.123.321",                 //IP
   22,                                //Port
   "rpki_user",                       //SSH User
   "/etc/rpki-rtr/hostkey",           //Server hostkey
   "/etc/rpki-rtr/server.priv",       //Authentication private key
   "/etc/rpki-rtr/server.pub"         //Authentication public key
};
tr_ssh_init(&config, &ssh_socket);

tr_socket* tcp_socket;                //create unprotected TCP conn.
tr_tcp_config tcp_config = {
   "123.321.123.321",                 //IP
   "1234"                             //Port
};
tr_tcp_init(&tcp_config, &tcp_socket);
```

Example – Create Server Pool

```
//create 2 rtr_sockets for both transport sockets
rtr_socket rtr_ssh, rtr_tcp;
rtr_ssh.tr_socket = tr_ssh;
rtr_tcp.tr_socket = tr_tcp;

//create a rtr_server_pool
rtr_server_pool p1{
   3,                         //preference value
   &rtr_tcp,                  //rtr_socket
   NULL                       //next pointer
};
rtr_server_pool p0{
   5,                         //preference value
   &rtr_ssh,                  //rtr_socket
   &p1                        //next pointer
};
```

Example – Create Connection Manager and Perform Origin Validation

```
//init all rtr_sockets with the same settings
//srv. pool,polling_period,cache_timeout,update_fp,conn_fp
rtr_mgr_init(&p0, 60, 120, NULL, 0, NULL, 0);

//create and start the connection manager
rtr_mgr_socket mgr_sock;
rtr_mgr_start(&mgr_sock, &p0);

//validate the BGP origin ASN 12345 for 10.10.0.0/24
ip_addr prefix;
prefix.ver = IPV4;
prefix.u.addr4.addr = 0x0A0A0000;

pfxv_state result;
rtr_mgr_validate(mgr_sock, 12345, &prefix, 24, &result);
```

Conclusion & Outlook

- RTRlib provides a flexible, layered architecture
- Required 3rd party libs: libssh

- Release date for prototype: 1st September'11
- Project website: http://rpki.realmv6.org/
  - Mailing list available
  - Documentation of current API: **Please, comment!**