Beacon Client for BGP testbed

Ravikumar Jeyaraman

May 05, 2014

Introduction:

Transit Portal (TP) is a system that enables controlled access for researchers and educators to the Internet routing system. To experiment with novel routing ideas or to understand aspects of the current routing ecosystem, researchers need the ability to actively participate in this ecosystem by emulating an autonomous system (AS). The Transit Portal testbed solves this problem for the researchers. The testbed can multiplex multiple simultaneous research experiments, each of which independently makes routing decisions and sends and receives traffic. The Transit Portal (TP) allows researchers to announce IP prefix from predefined pool of allowed prefixes. The testbed essentially functions as a full-fledged participant in interdomain Internet routing.

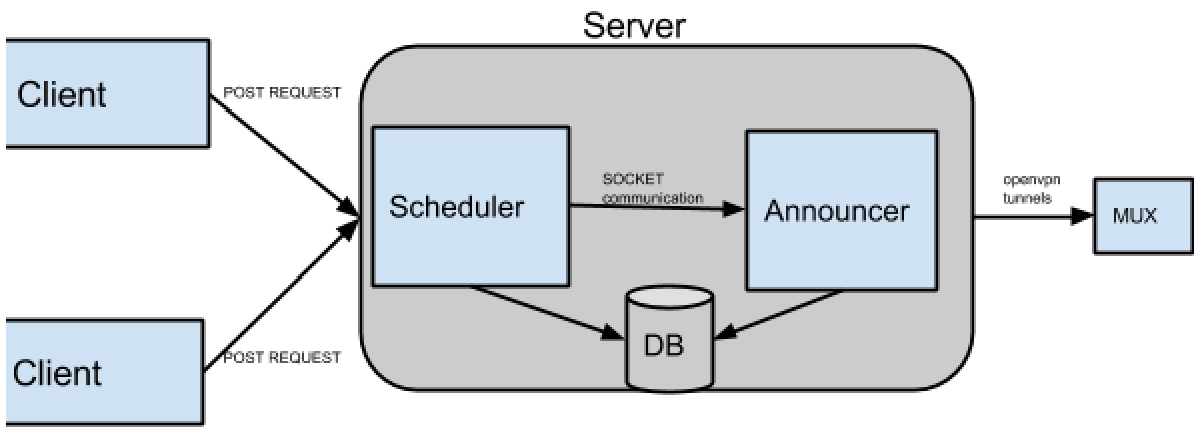
At present to run simple experiment Transit Portal requires significant manual configuration which includes having openvpn tunnel connection to the MUXES(border routers) from where we intend to announce our prefixes.There are currently 7 of these Points of Presence for the TP system, 6 spread out in the US and one in Amsterdam.We propose to have Beacon which will be pre configured with openvpn tunnels to all the available MUXES and have Beacon client which will allow researchers to make announcement through the Beacon Server.This improves openness of the Transit portal which is currently used only be select group of people.

Background:

Border Gateway Protocol (BGP) is a exterior gateway protocol designed to exchange routing and reachability information between autonomous systems (AS) on the Internet. It is a path vector protocol.BGP neighbors, called peers, are established by manual configuration between routers to create a TCP session.Border routers learns the route advertisement which are announced and propagated through its neighbors. Usually it takes a few minutes for a route advertisement to get propagated.Route propagation depends on several parameters some of which are under control of operators.Beacon provides systematic way to conduct Internet routing experiments with the Transit portal which in turn allows researchers to study how a route gets propagated in the Internet.

Architecture:

Beacon Server is composed of two entities. A scheduler and a announcer. Scheduler handles the request from clients and queues them in a priority queue. REST interface is used between client and the scheduler.One advantage of using REST interface is that it doesn’t bind our Beacon server to particular client scripts. Instead any client which is capable of sending a POST request with configuration message specified JSON format can communicate with the beacon.Announcer will refer a database which will be loaded with prefixes selects any of the available prefix and makes the configuration announcement through selected MUXES. MUXES are border routers which acts as interface through which TP makes announcement.



Beacon Design

Scheduler:

Scheduler will receive all the requests and queues them and send it to announcer at the end of every cycle. Cycle duration is configurable through a configuration file.
Scheduler is composed of two threads.

- One which deals with request from the client and queues them in a priority queue.
- Another thread which pulls the request from the queue and send it to the announcer.

If there is an available prefix for a new request, the scheduler will reserve that prefix, schedule it for the next cycle. If no prefixes are available, it will notify the client that they will be scheduled in the future cycles with a message on how long it will take to schedule the request.It should be noted that it is just an estimate and when a new request comes the current request might be pushed back in the queue if its priority is lower than the new request.

Priority queue will be used to schedule the requests.Requests can be classified into two classes based on user group. It can be either

- Research Request
- User Request

Ten levels of priorities are supported with 1 being the highest and 10 being the lowest.A research request can make request with any of the priorities from 1 -10 while a user request can make use of priority level from 6-10 only.It should be noted that priorities are just an integers but are restricted to only these 10 levels. If in future more levels are needed can be easily added with little modification in validation part.If no priority level is specified from the clients, default priority level based on user group will be used.For research request it will be level 5 and for user request default priority will be level 10.

Request will be pulled based on the priority at the end of every cycle and will be sent to announcer.If there are no requests to be announced default announcement is made at the end of every cycle.We call this default announcement as Beacon Request which serves as heartbeat announcement and helps us to monitor the beacon status.

Scheduler handles POST requests from the clients.These Post requests are expected to have JSON message as specified in below sections.This message will include username and password which helps us in authenticating the request. A requests gets queued in the priority queue only after it passes authentication.

When a request is received and unique ID is assigned to the request.This unique Id is notified to client in the response along with the estimate of schedule time.Client can use this unique id later point of time to check the status of the request.Currently we are using time the request received as unique Id.Even when multiple requests are made simultaneously it will be handled serially at the scheduler. So we believe having time as unique identifier will not create problems.

A request which is not of announcement type need not get into priority queue.Requests such as check schedule will get executed immediately and client will be notified on estimated time request will get announced.Priority queues are internally implemented as a heap and elements are not it sorted order unless pulled out. So to check the current position in the priority queue we are pulling all the elements from the queue which gives us a sorted list .We find the index of our interested request and then push back all the elements into the queue.We believe this is inefficient and should be improved in the future.

Announcer:

Announces the Prefix announcement through MUXES that are specified in the configuration message.MUXES are border routers which acts as interface through which TP makes announcement. As a requirement announcer should have openvpn tunnel connection with all the MUXES. Quagga routing software is used to create BGP announcements which is then sent through openvpn tunnels to the MUXES.

Announcer will choose a prefix from the list of available prefixes for making announcement.It will update the availability of a prefix in the DB once a particular prefix is used/withdrawn.Most of the complexity here is to configure openvpn tunnels and to bring up quagga daemon which helps in constructing BGP request.Once a announcement is made a email notification is sent to the user who made the announcement.To get the email Id of the user we use unique Id of the request which will help us in getting the username which is then used in usertable to get the email ID.

Client:

Since a REST interface is used at the scheduler, anyone who is able to POST a request with JSON message will be able to schedule a request and are not binded to our client scripts. Authentication information should be included in every request.

Configuration Message should be specified in JSON format with key value pairs.

```
{
      "username": "ravi",
      "password": "test",
      "configuration": [
      {
                  "data": "73,74",
                  "mux": "WISC"
      },
      {
                  "data": "withdraw",
                  "mux": "ISI"
      }
      ],
 "priority" : "default"
}
```

where

- username & password are authentication information.
- configuration is a json Array and can include any number of MUXES through which user wishes to make announcement.
    - mux includes MUX names through which user wishes to make announcements.
    - data includes AS numbers which uses wishes to poison.data can also include withdraw which enables users to announce from one MUX and withdraw the same prefix from different MUX. data can also be empty in which it won't poison any AS during announcement.
- priority : As specified in above sections it can include values from 1-10 and if not specified scheduler will use default values.

Usage of the client script:

```
python client_rpc.py -a -m Conf1 -m Conf2 -u username -p password -l priority
```

```
-a      - Annoucement
-c      - Check
-m      - Mux configuration multiple ASN separated by comma
-t      - transaction ID
-u      -username
-p      -password
-l      -priority level
```

For example if we wish to make priority level 9(lower)announcement through WISC MUX and poison AS 93 and 94 it can be specified as

```
python client_rpc.py -a -m WISC.93,94 -u ravi -p test -l 9
```

Multiple AS numbers to be prepended/ poisoned can be separated with comma.

To check announcement schedule with announcement ID 12345

```
python client_rpc.py -c -t 12345 -u ravi -p test
```

Database Schema:
Beacon has three tables to for it keep track on information of prefixes and users.
User table , Prefix table , Transaction Table

Each user is represented with a record in user table.The schema of user table is

*Username	Password	Email	Usergroup

where user_group can be either research or user. Accordingly priorities will be assigned in the scheduler. Once a queued request is announced ,user will be notified through email.All the entries in this table are mandatory.

Transaction table keeps track of mapping between Transaction ID and username.The schema of transaction table is

Username	*Transaction ID

This allows mapping many announcement request from single user.

Prefix table will have prefix details and its availability information.The schema is

*Prefix	Availability	Transaction ID	Beacon Avaialble

TransID will be update after each announcement. Availability indicates whether the prefix currently used for announcement. It is also updated after each announcement and at the end of the cycle. Beacon Available is a boolean flag to indicate whether the particular prefix can be used for default beacon request.

Logging:

Logs includes details on user who made the announcement and when a requests arrived and when it gets announced. Both scheduler and announcer will log the to the same file.Scheduler will log when a request is received and when a request is sent to announcer .Announcer will log once announcement is made.These log files are available at Beacon and needs administrative privilege for viewing or editing the file.

Configuration File:

Server reads configuration file at the start up. Configuration File includes

- information related to database
    - DB username
    - DB Password

    - DB name
- Cycle time
- MUXES configured

Recommended value for cycle time is 5400 seconds .This allows sufficient time for path convergence. Currently available MUXES are read from the config file and it has no ties with actual available MUXES. We plan to get the available MUX information from TP database at runtime and add it to MUX List.

Demonstration:

Here we are showing BGP route table entry of a router located at Amsterdam.
After announcement from ISI site:

Beacon client usage : python client_rpc.py -a -m ISI -u username -p password
show ip bgp routes detail 184.164.237.0/24

Status	Network	Next Hop	Metric	LocPrf	Weight	Path	Origin
BI	184.164.237.0/24	198.32.146.10	1529	100	0	226, 47065	IGP
I	184.164.237.0/24	198.32.146.10	1529	100	0	226, 47065	IGP
E	184.164.237.0/24	213.248.72.145	0	70	0	1299, 2914, 226, 47065	IGP

Table 1

After announcement from GATECH site:

Beacon client usage : python client_rpc.py -a -m GATECH -u username -p password
show ip bgp routes detail 184.164.237.0/24

Status	Network	Next Hop	Metric	LocPrf	Weight	Path	Origin
BI	184.164.237.0/24	198.32.132.11	1010	100	0	10490, 2637, 47065	IGP
I	184.164.237.0/24	198.32.132.11	1010	100	0	10490, 2637, 47065	IGP
E	184.164.237.0/24	213.248.72.145	0	70	0	1299, 174, 2637, 47065	IGP

Table 2

We can see route switching from above two experiments. Immediate hops from Home ASN changes after announcing from different site.

In the above announcement through GATECH site we observe 10490 in the AS -PATH for few entries. Suppose if we make announcement through GATECH site along with poisoning 10490 we see routing table entries as shown below.

show ip bgp routes detail 184.164.237.0/24

Status	Network	Next Hop	Metric	LocPrf	Weight	Path	Origin
BE	184.164.237.0/24	213.248.72.145	0	70	0	1299, 174, 2637, 47065, 10490, 47065	IGP

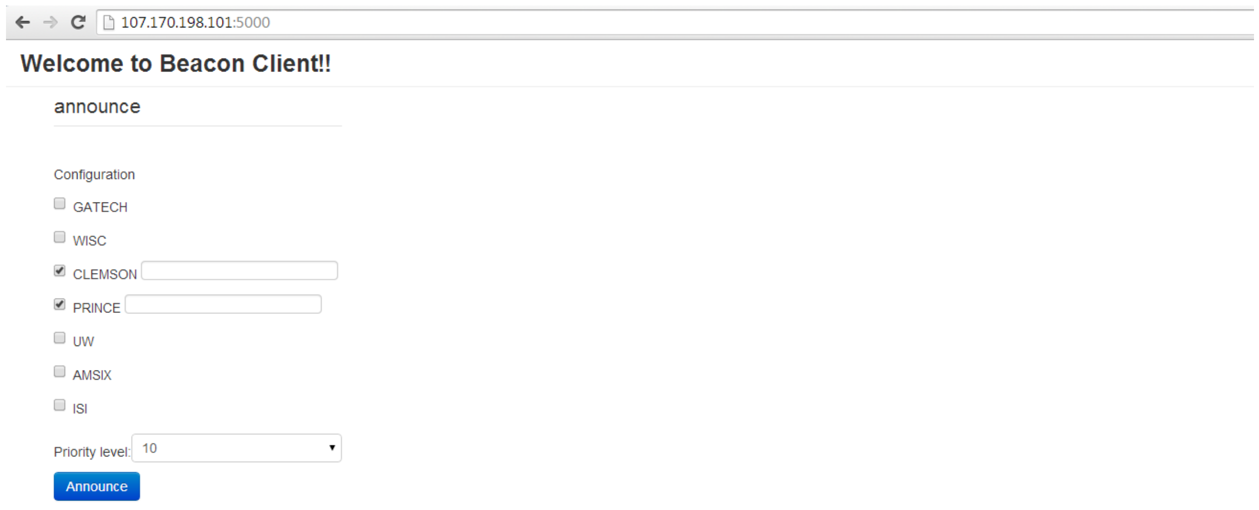
Table 3

Beacon client usage : python client_rpc.py -a -m GATECH.10490 -u username -p password

As seen in the table this announcements prepends 10490 in the AS-PATH. When AS 10490 sees this path it will drop the announcement silently.Hence the path seen in Table 2 through 10490 are not seen in Table 3

Web Interface:

Web Interface is provided to ease the one time announcement process. Interface looks like the image shown below


107.170.198.101:5000

Welcome to Beacon Client!!

announce

Configuration

- [ ] GATECH
- [ ] WISC
- [x] CLEMSON
- [x] PRINCE
- [ ] UW
- [ ] AMSIX
- [ ] ISI

Priority level: 10

Path Forward:

- Get available MUX from TP database at runtime
    - Currently Beacon reads available MUXES from a config file which has no ties with actual available MUXES.Server needs to modified such that it reads directly from TP database at runtime. Strategy for that would be check TP database every 30 minutes or so to get available Site information
- Capability to add prefix dynamically by the user and uses the prefix for experiment.
- configure an SMTP server for the email messages to be sent via instead of localhost SMTP service.

Summary:

Beacon improves the workflow of BGP experiments by queuing and scheduling the prefix announcement request.This creates a systematic way to conduct experiments instead randomly using same prefixes in many experiments.We are looking to increase the number of users of Transit Portal with this application which in turn will gives way to novel routing ideas in inter domain routing.

Code Location:
https://github.com/TransitPortal/beacon

Mentors:
Ethan Katz-Bassett
Kyriakos Zarifis
Ítalo Fernando Scotá Cunha
Brandon Schlinker