CS/CE 6390 – 002 Advanced Computer Networks
Project: Path Computation Element
Due December 1, 11:59 pm

Overview

In this project, you will implement a multi-domain network in which path computation elements (PCEs) are used to assist in the calculation of inter-domain explicit routes. Each domain will consist of a number of routers and one PCE. Domains are interconnected through their border routers. Within a domain, explicit routes are calculated by the PCE of that domain. The routers within the domain send link state advertisements to the PCE, and the PCE uses these advertisements to construct a view of the intra-domain topology. The PCE can then run a shortest-path algorithm to find an intra-domain path. Note that, in this project, we will only be implementing PCE-based routing. Therefore, LSAs need not be flooded throughout the entire domain, but just need to be sent to the PCE of that domain.

For inter-domain routing, the PCEs utilize a simplified cooperative approach. The PCE in the source domain will send a path request to the PCE in the next-hop domain. If this next domain is not the destination domain, a request will be sent to the PCE of the next domain. This continues until the request reaches the PCE of the destination domain. The PCE of the destination domain considers each ingress router that connects to the upstream domain and selects the one that has the least-cost path to the destination network or router. The PCE then sends the ID of this router to the upstream PCE. The upstream PCE likewise considers all ingress routers that connect to its upstream domain and selects the one with the least-cost path to the router indicated by the downstream PCE. The procedure continues until the message reaches the PCE of the source domain PCE. The PCE of the source domain then calculates the least-cost path from the source router to the downstream domain's ingress router.

For simplicity, we will assume that there are at most 10 domains, and each domain is assigned a unique AS number ranging from 0-9. Each domain will have at most 10 routers, and each router will be assigned a router ID from 0-9, which is unique only within a given domain. A router will have multiple interfaces, and each intra-domain interface will be associated with a globally unique two-digit network number from 00-99. If two routers in the same domain have interfaces with the same network number, then those two routers are connected by an intra-domain link. Each router may also be connected to at most one router in a different domain. An inter-domain interface of a router is specified by the AS number of the neighboring domain and the ID of the router in the neighboring domain.

Routers

A router will be implemented as a socket-based client program and will be initiated with the following command line:

```
router <AS> <routerID> <configfile> <neighborAS> <neighborrouterID> <net1>
<net2> …
```

The command-line input parameters are defined as follows:

- `AS`: The AS number of the domain in which the router resides.
- `routerID`: The router's ID within its domain.
- `configfile`: The filename of a file containing hostname and port numbers for PCEs.
- `neighborAS`: If the router is a border router, this parameter is a number ranging from 0-9 specifying the AS number of the router's neighboring domain. Otherwise, this parameter is set to 99 if the router is not a border router.
- `neighborrouterID`: If the router is a border router, this parameter is a number ranging from 0-9 specifying the ID of the router in the neighboring domain to which this router is connected. Otherwise, this parameter is set to 99 if the router is not a border router.
- `net1, net2`, etc.: These parameters are a sequence of numbers indicating the network numbers to which the router's intra-domain interfaces are connected. Each number takes on a value from 00-99, which is a globally unique identifier of a network or a link between two routers. Note that, while in practice, each intra-domain interface should also have a host ID associated with it, we will ignore the host ID in this project.

Once the router process is initiated, it begins by opening and reading a configuration file, configfile, that tells it the hostnames and port numbers of the machines on which the PCEs are running. An example of the configuration format is as follows:

```
0  net10  12340
1  net11  12341
2  net12  12342
```

Each line in the file indicates the AS number of a domain, the hostname of the machine on which the PCE for the domain is running, and the port number for the PCE process. Note that a router will only contact the PCE of its own domain; therefore, a router would only need to consider the line in the configuration file that corresponds to the AS number of its own domain. Also, when testing your programs, you may run multiple PCEs on the same machine as long as each instance has its own unique port number.

After reading the configuration file, the router establishes a TCP connection to its PCE and sends a link state advertisement to the PCE. Once the LSA is sent, the TCP connection is closed. The router will also send an updated LSA to its PCE whenever the status of one of its links changes. A link state advertisement to the PCE is formatted as follows:

LSA <routerID> <neighborAS> <neighborrouterID> <net1> <metric1> <net2> <metric2> <net3> <metric3> ...

The routerID is a number between 0-9 that gives the ID of this router in its domain. The neighborAS is the AS number of the neighboring domain, and the neighborrouterID is the ID of the router in the neighboring domain. Both neighborAS and neigborrouterID are set to 99 if the router is not a border router. The rest of the advertisement is a list of the networks to which the router's interfaces are connected, followed by a link metric for each interface. For now, we will assume that the link metric is set to 1 if the interface is active, and set to 99 if the corresponding interface is no longer active.

The router process prompts for user input and takes the following input commands from the user:

- `RT <net>`: this calculates a route to network <net> and returns the sequence of routers to the destination network.
- `DN <net>`: this brings down the interface <net> of the router if it is currently up.
- `UP <net>`: this brings up the interface <net> on the router if it is currently down.
- `LI`: this will print out all of the router's current interfaces along with their metrics.

When the router receives a routing request from the user, it will send a route request messages to its PCE through a TCP connection. The messages will be of the format:

RREQ <routerID> <dest_net>

The routerID is the ID of the router sending the request, and dest_net is the network number of the network to which the router is requesting a route. The PCE will respond with a route response message, which the router will print directly to the user's screen. The TCP connection remains established when the router sends the RREQ message its PCE, and the TCP connection is closed only after the router has received its response message from the PCE.
For simplicity, you may assume that the router blocks while waiting for the response message, i.e., it doesn't accept any user input until it has completed its existing request.

Path Computation Element

A PCE is responsible for collecting LSAs from the routers in its domain to construct the intra-domain topology, calculating intra-domain routes as requested by routers in the domain, and cooperating with PCEs from other domains to calculate inter-domain routes. We will also assume that the PCE performs simplified BGP-type functions to determine domain sequences for inter-domain route requests.

The PCE process is initiated with the linux command line:

```
PCE <AS> <configfile>
```

Where AS is a number from 0 to 9 indicating the AS number of the domain in which the PCE resides. This number also uniquely identifies the PCE, since there is exactly one PCE in each domain. The input parameter configfile specifies a text file listing the hostnames and port numbers for the PCEs. The format of the file is as described above.

The PCE will maintain information related to both intra-domain and inter-domain connectivity. It is up to you to decide on the data structures for maintaining this information. The intra-domain information will keep track of the networks in the domain, the routers attached to each network, and the metric associated with each network (link) connecting two routers. For its inter-domain neighbor information, a PCE will maintain a list of its neighboring domains as well as a list of border router pairs between the domains (one of its own border routers and the border router in the neighboring domain). Note that there may be more than one border router pair between two domains. For inter-domain reachability information, a PCE will maintain, for each network number, the next AS in the sequence of domains towards the network, and the number of AS hops to the network (e.g. if the network is in the neighboring domain, the number of AS hops is 1).

After the process for the PCE is started, it first reads in the information from the configuration file, storing the information in the appropriate data structures. The PCE then waits for incoming TCP connection requests from routers in its domain or from other PCEs.

When the PCE receives an LSA (described above) from a router in its domain, it updates its intra-domain topology information appropriately. If the LSA is from a border router, it also updates its information regarding its links to neighboring domains. The PCE will then send out BGP-type messages to the PCEs of its neighboring domains. The message has the following format:

BGP <AS> <AS_hops> <net1> <net2> <net3> ...

Where AS is the AS number of the sending PCE, AS_hops is the number of AS hops to reach the advertised networks through the domain of the PCE sending the advertisement, and net1, net2, … is the list of network numbers being advertised. For the case in which the PCE is advertising networks in its own domain, the AS_hops parameter is set to 0, and the net1, net2, … parameters list all of the networks in this domain, not just those in the received LSA message.

When a PCE receives a BGP advertisement from a PCE in a neighboring domain, it looks at the AS_hops field, and for each network in the advertisement, it check to see if the advertised AS_hops + 1 is less than its current AS hop distance to that network. If so, the PCE will update its inter-domain reachability information for that network by specifying the advertising PCE's domain as the next AS and by specifying the AS distance to the network as AS_hops + 1. For the list of networks that are updated, the PCE sends a BGP advertisement to each of its neighbors (except the neighbor from whom it received the original BGP message) with AS_hops incremented by 1. If AS_hops + 1 is greater than or equal to the PCE's current AS hop distance to a network, the advertisement for that network is ignored. Note that this approach for determining the sequence of domains for a route is not the approach used by standard BGP (i.e., no path vector is maintained in the advertisements).

When a PCE receives a route request message, it checks to see whether or not the destination network is in its domain. If so, it will calculate a route from the requesting router to the router attached to the destination network. The route calculation can be done using any standard routing algorithm, such as Dijkstra's algorithm. The PCE sends the requesting router a routing response message, which has the following format:

RRES <RouterID1> <RouterID2> …

The list of router IDs specifies the route from the source router to the destination router.

If the destination network is not in the domain, then the PCE checks to see what the next AS is for the network. The PCE sends an inter-domain routing request to the PCE in this next domain. The inter-domain routing request is formatted as follows:

IRRQ <AS> <dest_net>

Where AS is the AS number of the domain of the sending PCE, and dest_net is the destination network number. A PCE receiving an IRRQ message will likewise send a similar message to the next AS if the destination network is not in the AS of the PCE.

If the destination network is in the domain of a PCE, then the PCE will identify all ingress routers that connect to the upstream domain and will calculate the shortest path from each of these ingress routers to the destination router (the router connected to the destination network). The PCE will select the ingress router that results in the shortest path to the destination router and will send the corresponding path in an inter-domain routing reply message to the requesting PCE. The message is formatted as follows:

IRRS AS<AS> <RouterID1> <RouterID2> <RouterID3> …

Where AS is the responding PCE's AS number, and the RouterID parameters indicate the routers along the route, with RouterID1 being the ingress router to this domain. Note that the AS number is prefixed with the characters “AS” to distinguish the AS number from the router IDs.

The PCE of an intermediate domain receiving an IRRS message will likewise identify all of its ingress routers connected to the upstream domain and calculate a path from each of these routers to the egress router that connects to the ingress router specified by the downstream domain's PCE. The shortest path will be selected, and the IRRS message will be constructed by prepending the found path to the path provided by the downstream PCE. The IRRS message would then look something like:

IRRS AS<AS1> <RouterID11> <RouterID12> <RouterID13> … AS<AS2> <RouterID21> <RouterID22> <RouterID23> …

If at any point, the PCE of a domain cannot find a route, the IRRS message sent by that PCE to the upstream PCE will be formatted as: **IRRS AS<AS> BLK**. Any information regarding downstream routes are discarded. Upstream PCEs will then

simply prepend their own AS numbers to this message (**IRRS AS<AS1> AS<AS2> BLK**) and send the IRRS to the upstream PCE.

In the source domain, the shortest path will be calculated from the source router to the egress router that connects to the ingress router specified by the downstream domain's PCE. The PCE in the source domain will then send a RRES message to the requesting router. This RRES message will contain the entire path as a sequence of routers through each domain.

A TCP connection remains established whenever a PCE sends an IRRQ message to a downstream PCE, and the TCP connection is closed only after the PCE has received its IRRS message from the downstream PCE. For simplicity, you may assume that the PCE blocks while waiting for the IRRS message, i.e., it doesn't accept TCP connections for any other RREQ or IIRQ requests until it has completed its existing request.

Note that, in a practical situation, a PCE would most likely not inform upstream PCEs of the specific routers in the path through its domain, but would only indicate the ingress router through which the upstream node can establish a path. In this project, we maintain the entire path just to verify that the correct route is found.

Additional Notes

For this assignment, you may work in groups of up to two students. You may discuss general programming concepts (e.g., data structures, syntax, classes, library functions, compiling errors, etc.) with other students, but you should not discuss specific implementation details, nor should you allow other students to see or copy your code. On the Internet, you may refer to websites that provide general programming advice, but you should not seek specific code for implementing the assignment, nor should you seek external assistance on your project from individuals over the Internet. You are allowed to use any source code provided through the links on eLearning or provided in the textbook. If you do use existing code, you must explicitly indicate the source of the code. If you have any uncertainties regarding these policies, ask the course instructor.

All network programs should only be executed on the computers *net01.utdallas.edu* to *net45.utdallas.edu*. C and C++ socket programs must be compiled on *cs1.utdallas.edu* or *cs2.utdallas.edu*. You may remotely log on to these machines using your netid from any other machine or terminal within UTD. From another UNIX or Linux-based machine, you should use the *ssh* command (i.e., *ssh net01.utdallas.edu*). From a Windows-based machine, you can use *putty* (http://www.chiark.greenend.org.uk/~sgtatham/putty/). To transfer files from a Windows-based machine to a UNIX or Linux-based machine, you can use *WinSCP* (http://winscp.net/eng/index.php).

You are to submit your well-commented source code to eLearning along with a README file explaining how to compile and run your program. You should also include with your submission a document containing a brief description of your implementation. For C++, source code files should end with .cpp or .h extensions. For C, source code files should end with .c or .h extensions. For Java, source code files should end with a .java extension. Place all files in a single-level folder or directory named with your netid, e.g., xyz061000, and zip this directory into a single zip file, e.g., xyz061000.zip. Upload this zipped file to eLearning. You may upload a single submission per group. If both group members upload submissions, only one will be selected for grading.