UNIVERSITY OF ALBERTA

MINT'S CAPSTONE PROJECT REPORT

Dynamic Network Device Configurator: An Automatic Router Configuration Tool

A final report submitted in fulfillment of the requirements for the degree of Master of Science in Internetworking

December 2012

Author:

Nelson Young

Supervisors:

Dr. Mike MacGregor

Sharad Chitrakar

Table of Contents

ABSTRACT:

This paper describes Dynamic Network Device Configurator (DNDC), a graphical Java application that allows networks to be constructed easily, and linked by one or more dynamic routing protocols, such as ISIS, EIGRP, OSPF, RIPv2, and BGP. It is designed to help students learn the implementation of the routing protocols by generating the relevant basic router configurations underlying the protocols automatically.

1. INTRODUCTION:

Routing is an important part of networking, and dynamic routing protocols are one of the key components of routing. They are used to gather and distribute routing information dynamically so that the status and connectivity of the network are maintained at all times. Today, there are a number of popular, graphical network simulation software on the market, including GNS3, Cisco Packet Tracer, and Boson NetSim that provide support for routing and detailed simulations of complex networks. They are used primarily for complementary training as well as network lab experimentation and research.

GNS3 (GNS3, 2007-2012) is platform-independent, open source and free. It requires and mounts real IOS images for its emulation of Cisco routers and switches; thus, its use is limited by the availability of the Cisco proprietary IOS images. Another freeware tool that is similar to GNS3 but is less complex and only available on Windows platform is Cisco Packet Tracer (Cisco, 2012). It doesn't require Cisco IOS images because it is a simulator instead of an emulator; hence, it is less process intense and supports fewer

commands than GNS3. Finally, there is Boson NetSim (Boson, 2012), a commercial network simulator that rivals GNS3 in simulating Cisco networking components, and provides extensive built-in labs and exercises for educational purposes.

However, none of them support any level of automatic device configurations for dynamic routing protocols. Users build their network topologies, and manually set up the configurations for each device in the networks. This process is not only time-consuming and tedious but can also introduce errors and inconsistencies since the users still need to know how to configure the devices properly and correctly. This could be a very daunting and frustrating experience, especially for users who are trying to learn the routing protocols and troubleshoot the networks.

For this reason, I developed an auto-configuration tool called Dynamic Network Device Configurator (DNDC). This standalone application enables the automatic configurations of routers in a user-defined topology based on selected dynamic routing protocols. With an intuitive, embedded visual editor, DNDC allows network diagrams to be drawn quickly. Along with the networking devices, DNDC also supports the abstraction of various dynamic routing protocols. The protocols help to define the configurations for the routers in the network. Finally, DNDC has the capability to save the network diagrams and the router configurations easily for repeated experiments.

2. DYNAMIC NETWORK DEVICE CONFIGURATOR:

2.1 INITIAL INTERFACE

The initial user interface for DNDC was a simple web application that would guide users to design their network topologies through a sequence of steps. They would first specify the number and type of devices such as routers and switches in a network fragment. Then they would specify the type of connection between the devices such as Ethernet or serial. Next, they would specify the networks addresses for each device and its link. At the end, they would select the dynamic routing protocol to be used for that network. This would complete the requirements for that particular fragment and DNDC would automatically generate and return all the router configurations in that fragment. Note that complex networks could be built by connecting individual fragments.

This web-based approach was quick, easy to use and highly accessible via the Internet. However, it lacked a visual representation of the network topology, making it difficult for users to verify the correctness of the configurations. So, I attempted to automatically build the network topology too in the background using ImageMagick. DNDC would return both a schematic diagram of the network and the router configurations.

The task to automate the creation of an image for a network topology turned out to be more complex and challenging than I originally thought. It was very difficult to organize the devices and their connections along with their labels without any overlapping. In addition, the topology might not look exactly as what the users want.

Hence, I decided to develop a graphical user interface that would allow the users to draw their own networks instead.

2.2 DESIGN

The design of DNDC can be split into two components: one component is the graphic editor interface that holds the network topology and the other is the automation process to generate router configurations based on the visual model of the network. This model consists of routers, links and routing protocols. The process flow is shown in Figure 1.

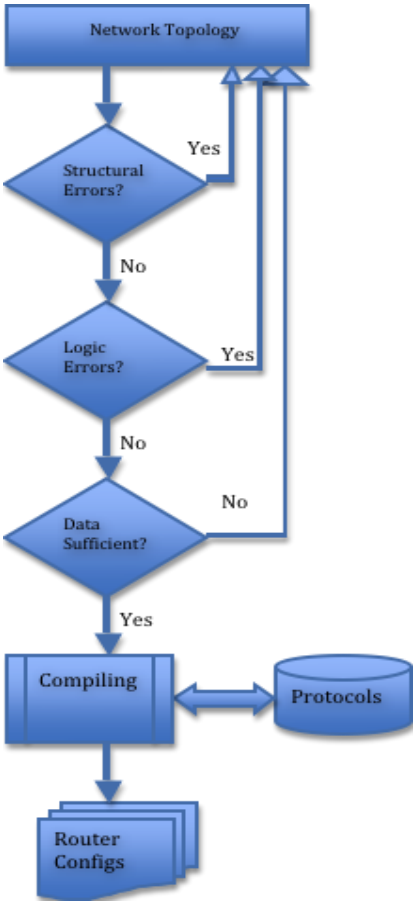

Figure 1: Auto-Configuration Process Overview

2.21 NETWORK SPECIFICATION

The first phase of the flow is network specification. The users begin by creating the layout of the network using network entity objects consisting of routers and routers' links (serial or Ethernet). Then they specify the IP addresses for the routers and their connections. Thirdly, the users decide what protocols to use in the network and where in the network to apply the protocols. This essentially sets up a high level abstraction of the logical structure of the network, as well as its allocated IP resources and entity-to-entity relationships.

When all the network entities are defined, the next phase is to validate the structural integrity of the network model. The following rules are enforced to ensure that the visual model is semantically and conceptually correct, or applicable to actual network setup:

1. autonomous systems cannot contain another autonomous system;
2. dynamic routing protocols cannot contain any autonomous systems or any other routing protocols;
3. dangling links and orphan routers are not allowed; a router must connect to at least one other router and routers must exist on both ends of a link;
4. overlapping of any protocols are not allowed; for example, a router's interface port can only be set up with one protocol;
5. a router can only belong to one autonomous system;
6. all links must be defined with a valid interface port and IP address on both ends.

In addition to validating the logical structure of the network, DNDC also validates the data logic of the network, particularly the IP resources and protocol-specific logic. The following rules are enforced to ensure that the network contains valid data only:

1. IP for the link and the routers are valid and within range for a specified subnet;
2. reuse of valid IP address block is not allowed;
3. reuse or overlapping of IP addresses is not allowed;
4. correct translation between dot-decimal notation to CIDR notation;
5. correct translation between wildmask (0.0.0.255) and netmask (255.255.255.0);
6. NETs are specified for IS-IS protocol;
7. autonomous system must have an AS number;
8. OSPF must have a backbone area;
9. stub areas cannot be connected to each other for OSPF;
10. transit area in OSPF must exist between the backbone area and another area that is not physically connected to the backbone area.

Finally, DNDC checks for data integrity. It checks for missing IP, interface port, name, and protocols' attributes. Once all the data is collected and validated, DNDC proceeds to the final phase of compiling and integrating the router configurations with the selected routing protocols.

DNDC has a library of predefined policies. Each policy handles a specific dynamic routing protocol by defining a set of basic commands that are required to configure in each router in order for the protocol to operate correctly in the network. For example, the

following configuration code snippet is required to be configured for all the routers in Figure 2 that are running EIGRP:

```
router  eigrp <ASN>
no auto-summary
network <network> <wildmask>
```

If certain loopback networks on some routers need to be advertised, then additional commands will need to be added to the routing protocol's policies dynamically and appended to the generated router configuration file. The same rule applies with other protocol-specific attributes that are altered at run-time by the users.

Some dynamic routing protocols, such as OSPF and BGP might have more attributes and commands than RIPv2 and EIGRP. For example, if an area in OSPF is changed from normal to stub area, then an "area <area_id> stub" command must be appended to the configuration.

Of course, there are several basic commands that are common to all protocols, such as setting up hostname, interface and IP. Instead of replicating the same codes to all the policies, these commands are stored in a separate, central policy called the Master Policy. At runtime, the Master Policy is merged with the appropriate protocol's policy to complete the configuration. The following configuration code snippet is consistent across all the routers in the network and is stored in the Master Policy:

```
hostname  <hostname>
interface <interface_number>
ip address <network> <netmask>
```

Note that the syntax for router configuration varies from equipment to equipment because each equipment has its own configuration language specific to its vendor's hardware. DNDC generates the configurations based on Cisco configuration language since the resulting configuration files are deployed to Cisco routers and GNS3 for testing. Also note that the router entity object has no defined number and type of interface ports. It is part of the design to have the router as generic as possible so that it is representative of any router. In this way, the router entity object can be logically mapped to any physical router in an actual network setup. It is up to the users to be aware of the device's physical hardware limitation when conceptually mapping it to an entity object. For example, if a router entity object is to be mapped to a Cisco 2800 router with two physical fastEthernet ports, then the object could only support up to two fastEthernet connections on the model network.

2.22 AUTO-CONFIGURATION PSEUDO CODE

The following pseudo code outlines the steps required to validate, compile and generate the router configurations:

1. Check for logical, structural errors
2. If structural errors exist, then prompt error message and return to Step 1
3. Check for network resource errors
4. If network resource errors exist, then prompt error message and return to Step 3
5. Check for data integrity
6. If data integrity violated, then prompt error message and return to Step 5
7. Sort all protocols in the order (IGP, eBGP)
8. Find router's protocol memberships (which protocol on which interface port?)

9. Loop through all protocols, <p>
10. Find all routers <r> in <p>
11. Apply Master policy to <r>
12. Apply <p>'s policy to <r>
13. Create configuration file

2.23 TESTING

To test that DNDC would properly auto generate the correct router configurations, sample networks employing some dynamic routing protocol were created in DNDC. Then, identical networks with similar configurations were either emulated using virtual routers in GNS3 or set up using real Cisco routers in the MINT lab. Next, the router configurations generated by DNDC were copied and pasted to the corresponding routers in the parallel network for verification. To verify that the network was configured as what we expected, 'show ip route' and 'ping' were used for testing. The command 'show ip route' displays the routing table of a router, while 'ping' tests the reachability of a targeted network. In most cases, if all the routers in the network were able to ping each other, then the network would be considered to be configured successfully. Of course, we would expect the routes to change according to the topology of the model network. As an illustration, if we add a local loopback network to a router and flag that network to be advertised, then we would expect an update of the router configurations to include another 'network' command, which would advertise that network to the protocol's domain. Changing the topology to trigger a corresponding update of the router configurations is one way to test the dynamics of DNDC.

There are several sample network topologies under the Help menu in DNDC that implement various dynamic routing protocols, including RIPv2, EIGRP, ISIS, OSPF and eBGP. Similar networks were set up in GNS3 and in the MINT lab for testing. For example, I used GNS3 to emulate one of the sample network topologies that employed EIGRP as its routing protocol. I created four c2600 virtual routers and connected them by fastEthernet similar to that in the sample network. After having executed the auto-generated configurations, the emulated network was fully connected and the traffic was properly routed as illustrated by the following routing tables of the virtual routers in GNS3:

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     34.0.0.0/24 is subnetted, 1 subnets
D       34.34.34.0 [90/33280] via 12.12.12.2, 01:53:45, FastEthernet0/0
     23.0.0.0/24 is subnetted, 1 subnets
D       23.23.23.0 [90/30720] via 12.12.12.2, 01:53:48, FastEthernet0/0
D    192.168.4.0/24 [90/161280] via 12.12.12.2, 01:53:45, FastEthernet0/0
     12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
C    192.168.1.0/24 is directly connected, Loopback0
D    192.168.2.0/24 [90/156160] via 12.12.12.2, 01:59:32, FastEthernet0/0
D    192.168.3.0/24 [90/158720] via 12.12.12.2, 01:53:46, FastEthernet0/0

R2#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
```

```
     i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
     ia - IS-IS inter area, * - candidate default, U - per-user static route
     o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

   34.0.0.0/24 is subnetted, 1 subnets
D     34.34.34.0 [90/30720] via 23.23.23.2, 01:54:38, FastEthernet0/1
   23.0.0.0/24 is subnetted, 1 subnets
C     23.23.23.0 is directly connected, FastEthernet0/1
D   192.168.4.0/24 [90/158720] via 23.23.23.2, 01:54:38, FastEthernet0/1
   12.0.0.0/24 is subnetted, 1 subnets
C     12.12.12.0 is directly connected, FastEthernet0/0
D   192.168.1.0/24 [90/156160] via 12.12.12.1, 02:00:23, FastEthernet0/0
C   192.168.2.0/24 is directly connected, Loopback0
D   192.168.3.0/24 [90/156160] via 23.23.23.2, 01:54:39, FastEthernet0/1

R3#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
     D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
     N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
     E1 - OSPF external type 1, E2 - OSPF external type 2
     i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
     ia - IS-IS inter area, * - candidate default, U - per-user static route
     o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

   34.0.0.0/24 is subnetted, 1 subnets
C     34.34.34.0 is directly connected, FastEthernet0/0
   23.0.0.0/24 is subnetted, 1 subnets
C     23.23.23.0 is directly connected, FastEthernet0/1
D   192.168.4.0/24 [90/156160] via 34.34.34.2, 00:02:17, FastEthernet0/0
   12.0.0.0/24 is subnetted, 1 subnets
D     12.12.12.0 [90/30720] via 23.23.23.1, 00:00:08, FastEthernet0/1
D   192.168.1.0/24 [90/158720] via 23.23.23.1, 00:00:08, FastEthernet0/1
D   192.168.2.0/24 [90/156160] via 23.23.23.1, 00:00:09, FastEthernet0/1
C   192.168.3.0/24 is directly connected, Loopback0

R4#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
     D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
     N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
     E1 - OSPF external type 1, E2 - OSPF external type 2
     i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
     ia - IS-IS inter area, * - candidate default, U - per-user static route
```

o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

```
    34.0.0.0/24 is subnetted, 1 subnets
C      34.34.34.0 is directly connected, FastEthernet0/0
    23.0.0.0/24 is subnetted, 1 subnets
D      23.23.23.0 [90/30720] via 34.34.34.1, 01:57:34, FastEthernet0/0
C    192.168.4.0/24 is directly connected, Loopback0
    12.0.0.0/24 is subnetted, 1 subnets
D      12.12.12.0 [90/33280] via 34.34.34.1, 01:55:23, FastEthernet0/0
D    192.168.1.0/24 [90/161280] via 34.34.34.1, 01:55:23, FastEthernet0/0
D    192.168.2.0/24 [90/158720] via 34.34.34.1, 01:55:24, FastEthernet0/0
D    192.168.3.0/24 [90/156160] via 34.34.34.1, 01:57:35, FastEthernet0/0
```

Note that a similar network was also set up in the MINT lab using four c2600 routers, connected by fastEthernet in the same way as shown in the sample network topology. The routers' routing tables displayed the same routes as those in GNS3.

In addition to EIGRP, other sample network topologies were set up and tested in a similar manner. Table 1 shows the different testing environment where the sample network topology was deployed to, the devices used and their connections as well as the expected and final results of the experimental network.

SAMPLE NETWORK TOPOLOGY	TEST ENV	ROUTER CONNECTIONS (HOSTNAME:PORT) ADJACENCIES	RESULTS
EIGRP	GNS3	C2600-R1:f0/0 <-> C2600-R2:f0/0 C2600-R2:f0/1 <-> C2600-R3:f0/1 C2600-R3:f0/0 <-> C2600-R4:f0/0 C2600-R1:Lo1= 192.168.1.1/24 C2600-R2:Lo1= 192.168.2.1/24 C2600-R3:Lo1= 192.168.3.1/24 C2600-R4:Lo1= 192.168.4.1/24	All the routers can ping each other, and can ping other router's loopback network. Results are as expected.
RIPv2	MINT LAB	C2600-R1:f0/0 <-> C2600-R2:f0/0 C2600-R2:f0/1 <-> C2600-R3:f0/1 C2600-R3:f0/0 <-> C2600-R4:f0/1 C2600-R1:Lo1= 192.168.1.1/24 C2600-R2:Lo1= 192.168.2.1/24 C2600-R3:Lo1= 192.168.3.1/24	All the routers can ping each other, and can ping other router's loopback network. Results are as expected.
ISIS	MINT LAB	C2600-R1:f0/0 <-> C2600-R2:f0/0 C2600-R2:f0/1 <-> C2600-R3:f0/1	All the routers can ping each other. R2 successfully advertised R3's and R1's

			networks to each other. Results are as expected.
eBGP	MINT LAB	C2600-Customer:f0/0 <-> C2600-ISP1:f0/0 C2600-Customer:f0/1 <-> C2600-ISP2:f0/1 C2600-ISP1:Lo1= 2.2.2.2/24 C2600-ISP2:Lo1= 3.3.3.3/24	All the routers can ping each other. Customer successfully advertised ISP1's and ISP2's networks to each other. Results are as expected.
OSPF-BGP	GNS3	C2600-R1:f0/0 <-> C2600-R2:f0/0 C2600-R2:f0/1 <-> C2600-R3:f0/1 C2600-R2:f1/0 <-> C2600-R4:f1/0 C2600-R4:f0/0 <-> C2600-R5:f0/0 C2600-R5:f0/1 <-> C2600-R6:f0/1	All the routers can ping each other. For example, R1 in Area1 can ping R4 in Area0 and R6 in AS200. Results are as expected.
OSPF-AREAS	GNS3	C2600-R1:f0/0 <-> C2600-R2:f0/0 C2600-R2:f0/1 <-> C2600-R3:f0/1	All the routers can ping each other and can ping other router's loopback network. R3 is an area border router because its f0/1 interface is in Area0 while its loopback network is in Area1 and is being advertised into Area0. Results are as expected.

Table 1: Router Configuration Deployment Results

2.3 DESCRIPTION

Users start by creating their network topology using the embedded graphical editor in DNDC as shown in Figure 2.

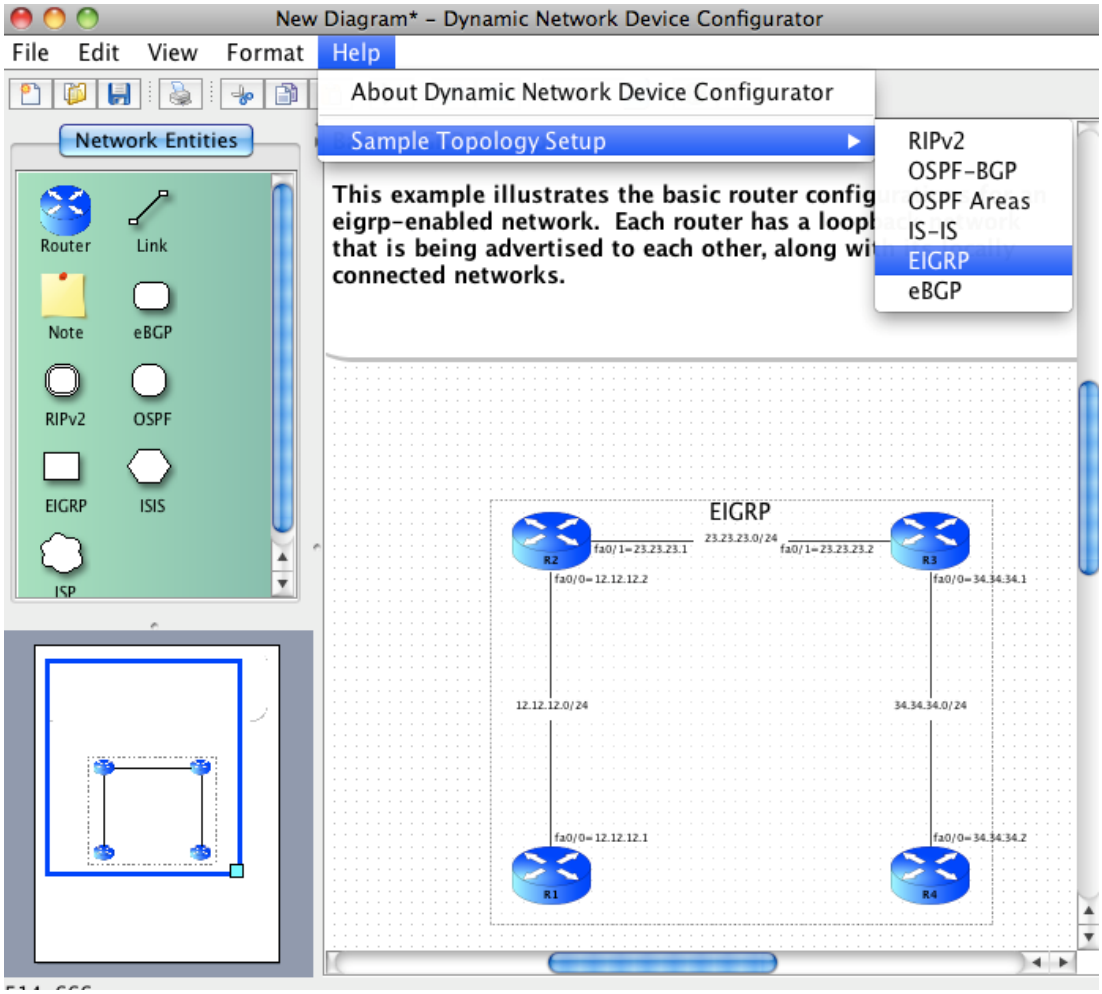

Figure 2: DNDC Editor Interface

The 'Network Entities' palette displays all the available types of devices, links and protocols. For example, DNDC supports Cisco routers for devices, serial and Ethernet for connection types, as well as ISIS, RIPv2, EIGRP, OSPF, and BGP for protocols. By default, the connection type is set to be Ethernet. Entities can be added to the diagram by drag-and-drop or by select-and-click from the palette to the diagram. Entities already drawn on the diagram can be selected, then copied and pasted as a group. Individual router can also be cloned too by select-and-drag. The cloned router would contain a copy of the parent's configurations and an Ethernet link between the routers

would automatically be created. This facilitates the fast creation of large, complex network with little efforts.

Below the 'Network Entities' palette is the page panel that shows the miniature view of the page. This helps the users identify the location on the page and navigate to the different areas of the page quickly. The blue square window on the panel can also be adjusted to set the zoom factor for the page or the users can select the zoom factor from the toolbar.

DNDC supports many graphical, editing functionalities. The sizes of the entities can be increased or decreased by drag-and-adjust. The grid sizes and lines can be toggled on and off. The labels on the entities can be selected and moved to avoid overlaps. The interface ports on the routers have the options to display just the port number, or port number + IP. The IP can either be displayed in either full or abbreviated CIDR notation. The font sizes, colors and types can also be changed based on the users' preferences. Further to the font formats, the editor supports 'Undo/Redo' functionality that tracks the history of changes on the diagram and allows the users to easily undo or redo their changes.

To view or set the properties for an entity, double-click it and a property dialog window would pop up as shown in Figure 3.

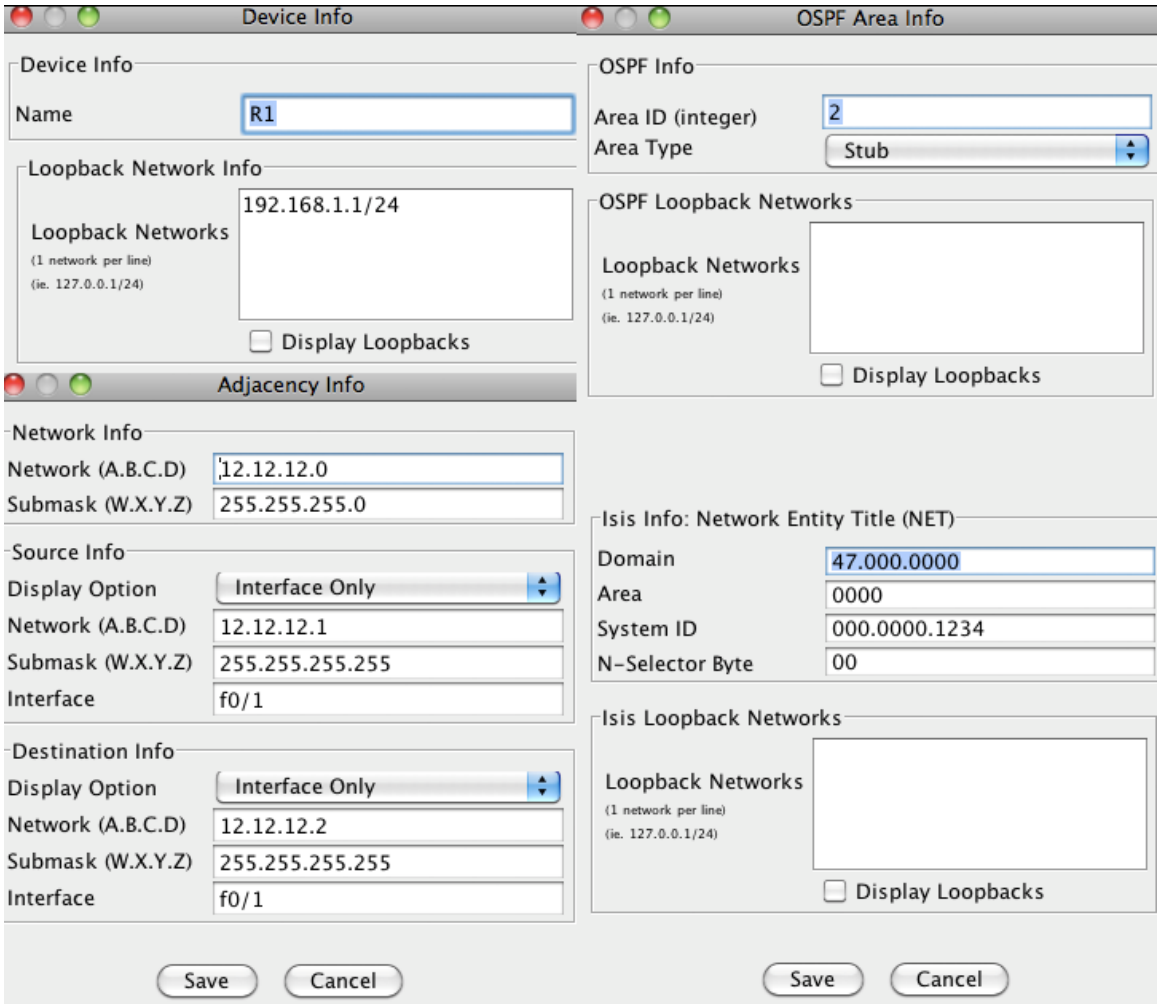

Figure 3: Property Windows for Different Entities

Each entity would have its own property window with the appropriate information. For example, a router's property window would contain the information about the router's name and its list of loopback network addresses; a link's property window would contain the IP information for the link segment, its target and source; a protocol's property window would contain some protocol-specific attributes, such as ASN for BGP, area name and type for OSPF, and the Network Entity Title for ISIS protocol.

Once a network fragment is created, the users can choose what dynamic routing protocol to apply in the network by dragging the appropriate protocol icons from the palette to the diagram. The protocol must contain or intersect the devices or the devices' connections for them to be configured for and be part of the protocol. An eBGP protocol can contain multiple network fragments and multiple interior protocols. In this setup, a network topology can have many autonomous systems with each autonomous system having its own interior routing protocols as shown in Figure 4.

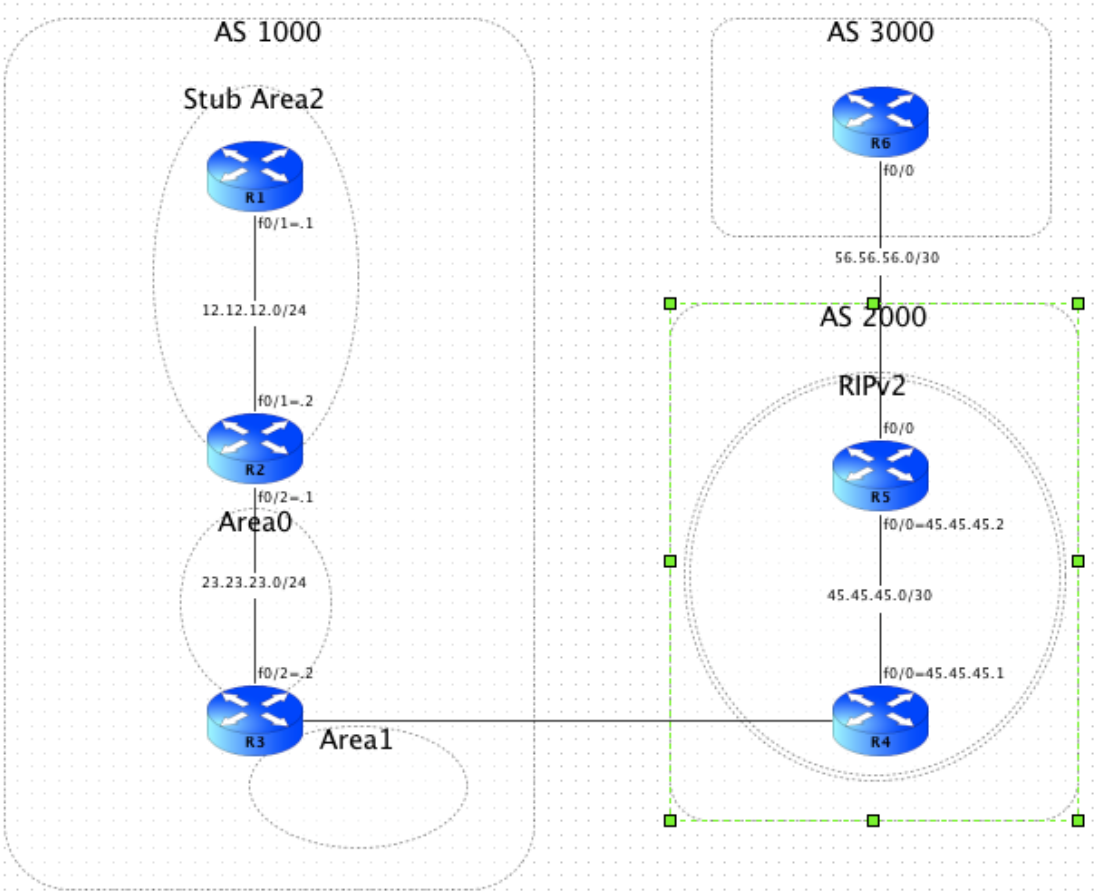

Figure 4: Small Network of Autonomous Systems

The eBGP protocol is selected by default to route information between the autonomous systems. The network fragments are assumed to be under the same

administrative control, or autonomous system if no other autonomous systems exist. Loopback networks on the routers are set to advertise using the underlying protocol, unless they are set not to under the router's property window.

After the network topology is completed, the users can click on the 'Run' button on the toolbar to have DNDC auto generate the configuration files for all the routers. If any error is encountered, the process is stopped and an appropriate error message is displayed. If successful, then the editor is immediately disabled from further editing (as indicated by a change of the background color to grey) and the router configurations can be viewed by clicking on the 'View All Run-Configs' button on the toolbar as shown in Figure 5.

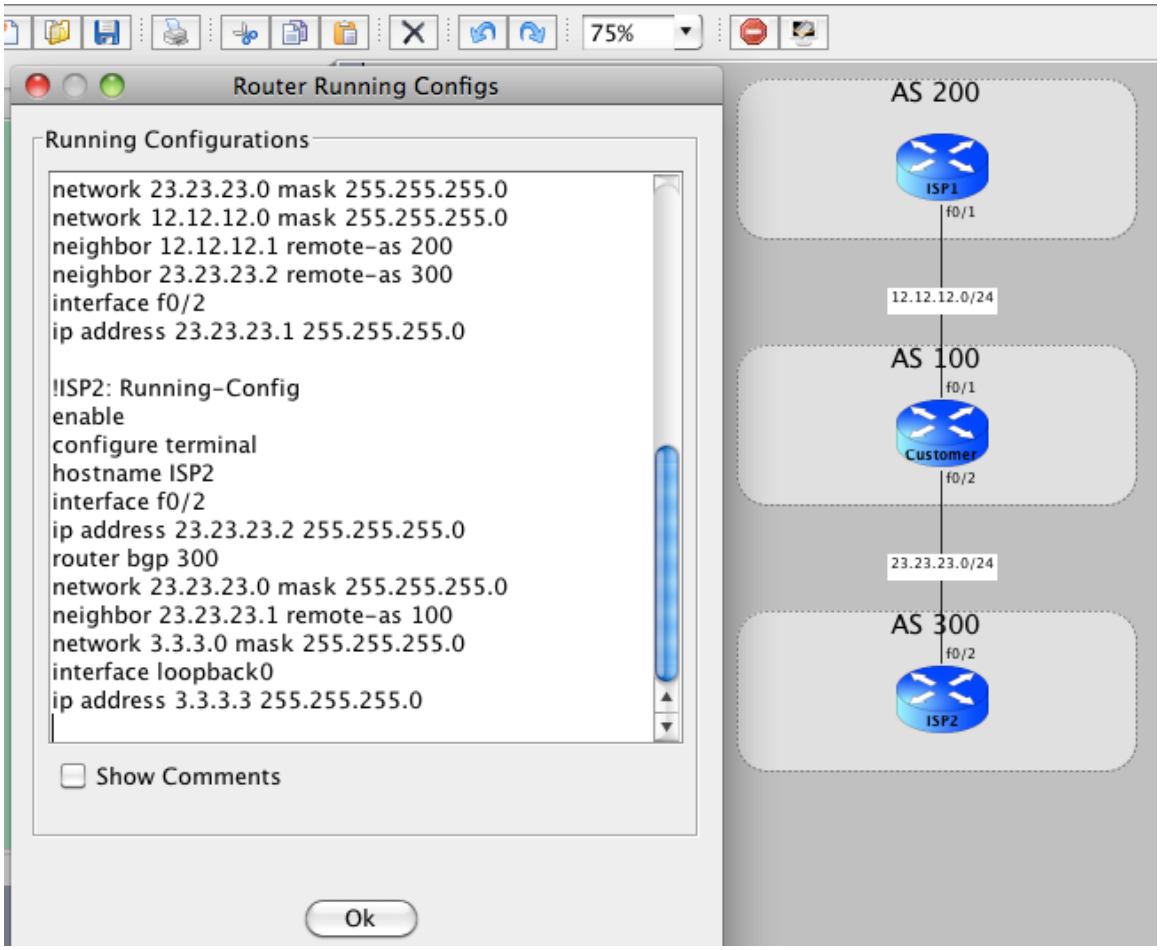

Figure 5: Auto Generated Router Configurations

To help users better understand each command line and its purpose, DNDC provides the 'Show Comments' option box at the bottom of the dialog that will show the appropriate comment above each command when checked.

2.4 IMPLEMENTATION

DNDC is written purely in Java because Java is an object-oriented programming language that I am familiar with. I can emulate the network topology by creating objects to represent the various network entities, such as routers, protocols and edges. Other advantages of using Java are that it is highly portable, reliable, robust and widely

supported. There are many graphic libraries, APIs and plugins freely available in Java. For example, I created the graphical user interface using Java swing components, including buttons, menus, layouts, windows, canvases, and textboxes.

To create the network topology, I extended the functionalities provided by mxGraph (mxGraph, 2006-2012), a freely available, open-source graph visualization tool with a Java API. I mapped the network topology to a graph structure with the routers being the nodes and the edges being the links. That way, I can utilize the inherent strength of the graphing package to generate very complex networks. mxGraph also provides several powerful graphing functionalities, such as loop detection, alternate path finding, and drill-down/step-up node searches.

Besides mxGraph, DNDC uses Apache Common SubnetUtils (The Apache Software Foundation, 2001-2012) network utility package to manage its network addresses. The package allows network addresses to be stored and represented in either dot-decimal notation (ie. 10.0.0.0 255.0.0.0) or CIDR notation (ie. 10.0.0.0/8). The utility package also calculates the valid IP range for a given subnet and creates all the IP addresses for a given network. DNDC uses SubnetUtils package to validate all the IP addresses entered by the users.

The data structure that is used to store the router configuration is Java component, JTree. Its model is similar to a hash table structure where each node is unique and hierarchical. This unique node constraint is useful because it ensures that there is no duplicate configuration code. For example, if command ‘hostname R1’ already exists in the tree as a node, it won’t be added again. It is also equally important that JTree

maintains its nodal, parent-child relationship, which allows commands to be nested and become subcommands of another command. This is required for some router commands. For example, as part of RIPv2 implementation, version2 and no auto-summary are subcommands that need to be configured under the router's rip process mode (ie. RouterA(config-router)#version 2).

2.5 TOOLS

SVN: Apache Subversion (SVN) was my version control system of choice. It was good programming practice to maintain and track changes in our source codes. SVN proved to be very useful in testing and debugging by allowing version rollbacks. I also used SVN because it would provide the opportunity for ease of code sharing and future collaboration. DNDC has over 10,000 lines of codes.

JDeveloper: JDeveloper was used as the IDE for the development of DNDC. It has a user-friendly editor for coding and a powerful debugging tool that simplifies development. JDeveloper assisted me in creating the graphical interface from dialog pop-ups to navigation menus to form layouts.

GNS3: GNS3 is used to test and verify the auto-generated router configuration files for the user-defined network.

3. SAMPLE NETWORK TOPOLOGIES

There are sample network topologies included in DNDC under the Help menu. Each topology shows how a network can be setup using different dynamic routing

protocols. This section describes a few topologies and explains their networks in more detail. Please refer to Figure 6 for references.

Figure 6a: In this example, the network is consisted of two autonomous systems, AS100 and AS200. AS100 is running OSPF with R2 being the area border router connecting Area1 to its Area0 (backbone). R5 is an autonomous system boundary router because the loopback networks on R6 from AS200 are set to redistribute to AS100's OSPF domain by R5. eBGP is used to exchange routing information between AS100 and AS200.

Figure 6b: In this example, RIPv2 is used as the routing protocol for the network. It is unnecessary to encapsulate the network with an autonomous system icon because there is no other autonomous system in the topology.

Figure 6c: In this example, OSPF is used as the routing protocol for the network. There are 5 area types: Normal, Stub, Totally Stub, Not-So-Stubby, NSSA Totally Stub, and Transit. In this example, Area2 is configured as a Stub Area and R2 is its area border router connecting to Area0. Note that R3 is also an area border router because some of its loopback networks are in Area1 (as indicated by the partial overlap of the Area1 protocol icon with R3).

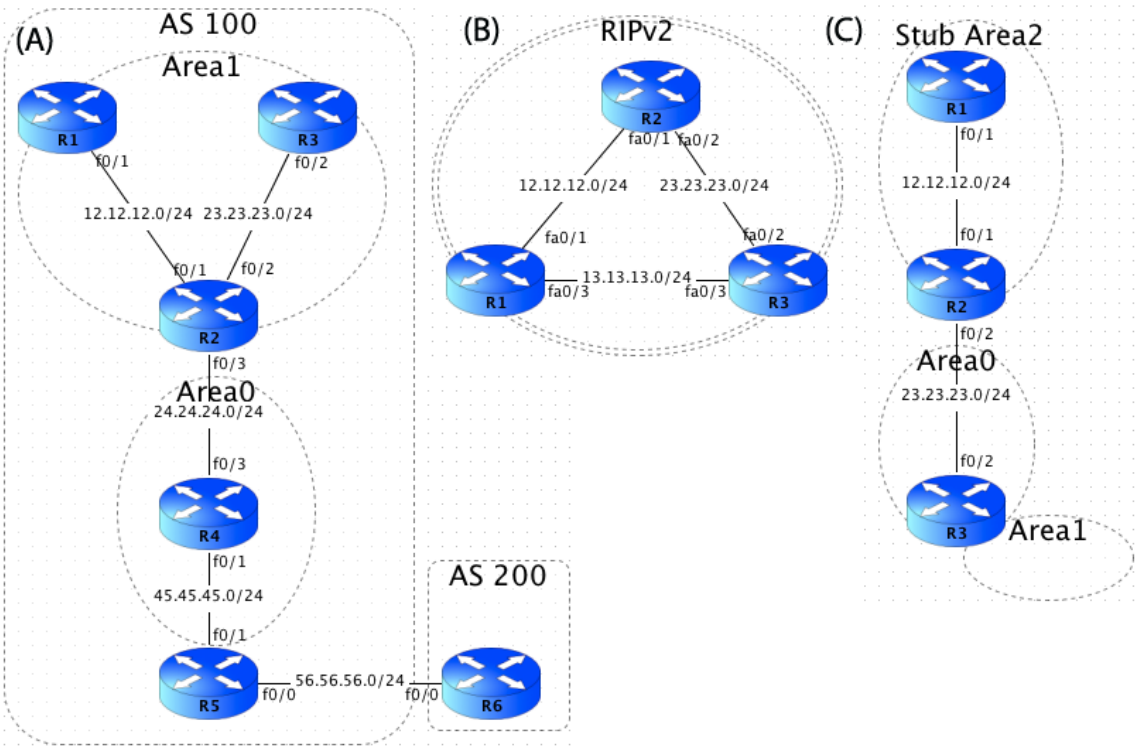

Figure 6: Sample Network Topologies

4. DISCUSSION

Similar to many network simulators on the market, DNDC achieves the goal of providing a user-friendly graphic editor for users to create their networks in a matter of steps. It is platform independent and simple to run, enabling users to easily design their networks. At the end, the users can export their network diagram along with all the router configurations.

Finally, DNDC provides the functionality that is not supported by other simulators; that is, the auto generation of router configurations for a variety of dynamic routing protocols. It not only helps students to learn the implementation of various routing protocols, but also dramatically reduces their efforts to configure the routers

manually themselves; thus, avoiding unnecessary work, errors and inconsistencies in the configuration phase.

Over the course of this project, I learned a plethora of design techniques and practices regarding user interface. Furthermore, I gained a more solid understanding of the principles and concepts of the different dynamic routing protocols. For example, I learned that in an OSPF-enabled autonomous system not all areas need to be physically connected to the backbone area (Area0); the areas can be connected via a virtual link to a transit area instead which would have full routing information; thus, the transit area cannot be a stub area at the same time.

I hope this application would benefit MINT students and that they would use it as a complementary tool for their learning. I look forward to introducing this application to MINT students next term so that they can provide me with feedbacks for future improvement of the application.

5. FUTURE WORK

Overall, DNDC has proven to be a success in easily generating networks and auto generating the routers' configuration files for different dynamic routing protocols used in the networks. The configuration files were successfully deployed and tested in real Cisco routers in the lab as well as on GNS3. Included in the Help menu are some sample network topologies that show how the network can be created and how the routing protocols can be selected to apply to the network.

There are still some additional features that I would like to implement in the future to improve the quality of the application. First, I plan to add layer-2 switch to the list of devices so that the network could be more complete and realistic to an actual lab setup. That way, the students can find my application more useful and applicable to their lab experiments. I would also like to add more support to the existing routing protocols. For example, additional protocol features include route filtering using Access Control Lists, BGP preferred path selection based on as-path pre-pending and MED.

In additional, I would like to add support for Alcatel-Lucent routers too because there are not as much resources available online compared to Cisco. The students can learn and compare the syntax of the configuration languages used by the different vendors' routers for implementing the same routing protocols. Finally, I would like to add support for auto router configurations for VOIP and MPLS so that the application can take the students to a different and more advanced level of internetworking than just dynamic routing.

DNDC is now a standalone Java application. Given the growing trend of web-based services and the ease of accessibility on the Internet, I would like to convert DNDC to an interactive web application. It would use Java applet as its front-end graphic editor, and Java servlet as its backend image-processing engine. Because codes used in an application can be ported to an applet, this change would be transparent to the users and all the features would still be supported. Making DNDC available online is also the easiest and most efficient method of pushing the latest software updates to end-users. I hope DNDC web application would gain greater popularity and usage online.

Bibliography

Boson. (2012). Retrieved 2012, from Boson NetSim: http://www.boson.com/netsim-cisco-network-simulator

Cisco. (2012). Retrieved 2012, from Cisco Packet Tracer: http://www.cisco.com/web/learning/netacad/course_catalog/PacketTracer.html

GNS3. (2007-2012). Retrieved 2012, from Graphical Network Simulator: http://www.gns3.net/

mxGraph. (2006-2012). Retrieved 2012, from mxGraph: http://www.jgraph.com/

The Apache Software Foundation. (2001-2012). Retrieved 2012, from SubnetUtils: http://commons.apache.org/net/api-3.2/org/apache/commons/net/util/SubnetUtils.html