James Madison University

Integrated Science & Technology (ISAT)

ISAT 461 Internetworking

Implementation of the Babel Protocol and Hybrid Network Architectures

Semester Project Lab Instructions

Author(s):
Troy Gamboa
Isaac Sumner
Joey Arbogast

Submitted to:
Dr. Emil Salib

May 3, 2017

Honor Pledge:I have neither given nor received help on his lab that violates the spirit of the JMU Honor Code.

Troy Gamboa	Isaac Sumner
Signature	*Signature*
Date	*Date*

Contents

1 Learning Objectives

-
-
-
-
-

2 Equipment & Software

Each team should have access to

- A desktop (9020 Dell) Linux 16.04 LTS (username: checkout, Password: hellocheckin)
- VMware Workstation 12 on Linux
- Virtual Machines Base (to be used for cloning and copying) are in /home/checkout/Base_VMs folder: Ubuntu Desktop 16.04 (x64), Ubuntu Server16.04 (x64)
- Access to iMac (OSX 10.11.x) with VMware Fusion 8.x (username admin3022, password: 3022$3022)
- Access to the Lab private network and to the public network (not simultaneously)
- Access to ShareLatex online
- Ethernet network connection
- C.O.R.E. VM
- Access to GNS3
- Acces to Asus NRT66 Wireless Access points (at least 2)

3 Best Practices (Trouble Shooting is How You Learn Networking!)

- To obtain the names of the available network interfaces use the following command: ls /sys/class/net.
- Get used to applying tcpdump for quick packet capturing: A good cheat sheet can be found in http://www.rationallyparanoid.com/articles/tcpdump.html
- Make sure that you have your SSD drive in every class. All your VMs should be on your SSD drive. Clone or copy the baseline VMs to your SSD drive. Never make use of the baseline VMs unless you are given a permission to do so.
- Lab Desktops are set up for weekly deep freeze, that is, every weekend, the desktops will be restored to their original image state. Whatever documents, VMs, etc. left will be wiped out with no way to be restored!
- Do not leave the lab without copying all materials (including VMs) that you care not losing!
- Make it a habit to look up protocol definition and specifications in the appropriate RFC and IEEE standards documents.
- We will be using ShareLatex in creating the Lab Reports. ShareLatex is available online for free.
- Take many screen-shots even if you decide later to drop some.
- Make it a habit to look up a new Linux command description using man ¡command¿.
- Check Firewall: Ubuntu #sudo ufw status, Fedora #sudo /etc/init.d/iptables status, Windows: Check firewall status.

- If you have problems connecting to the Internet, sometimes it's as simple as checking whether the Ethernet Network interface is registered with JMU registration server or whether the Ethernet Network interface adapter (real or virtual) is connected and/or configured correctly (manual versus DHCP).
- When you're working on remote desktop access, make sure that the PC you are trying to access remotely is configured to allow remote desktop access.
- Check reachability/connectivity between two PCs using the ping command.
- Power off or suspend all unused VMs to maintain a decent performance on the Host. The more VMs are powered on, the more demands on the Host's CPU and RAM.
- Check if the software application/program you need is installed.
- As a last resort, you could always reboot (power off/on).
- You should always check Edit ¿ Virtual Network Editor and make sure that vmnet0 is set up for bridged/Auto-bridging, vmnet1 is configured for Host-only and vmnet8 is configured for NAT. These vmnets configurations should not be changed unless instructed to do so.
- In the case that the Asus AP (Access Point) acts sluggish or becomes non responsive, make sure to reset the AP by unplugging the power, press the RED button. While still pressing the RED button, plug back the power and wait for 1 min before releasing the RED button. When asked to enter username and password enter admin, admin, and admin or root, root and root.
- If one of two connected devices has the automatic MDI/MDIX (Medium Dependent Interface/Meduim Dependent InterfaceCrossed) configuration feature there is no need for crossover cables. Introduced in 1998, this made the distinction between uplink and normal ports and manual selector switches on older hubs and switches obsolete.
- If you are working with one or more VMs on an exercise and you need to stop and continue at another time, you can always freeze the VMs in a state that would allow you to continue from where you left off. That is, instead of selecting Power Off you should select Suspend!
- If you get Error binding to port for 0.0.0.0 port. Check lsof i:1812. If you get a response, then you have hung up port. Kill port kill 9 pid. You may also check netstat unpl. However, the easiest way is to execute sudo pkill 9 radius.
- Check the following site for mysql commands: http://cse.unl.edu/~sscott/ShowFiles/SQL/CheatSheet/SQLCheatSheet.html
- tcpdump the easy tutorial http://openmaniak.com/tcpdump.php
- Note that // means COMMENTS
- Use clear!
- In a command line shell, you can scroll up using shift+pg up

4 Exercises

MAKE SURE TO PROVIDE NETWORK DIAGRAMs AND TABLEs CONTAINING INFORMATION ABOUT THE MACHINES INVOLVED (SUCH AS, IP ADDRESSES, MAC ADDRESSES, ETC.)

4.1 Exercise 1: Setting up Babel with a basic topology

4.1.1 Introduction

- Babel is a unique distance-vector routing protocol of which is designed in such a way that it is known as "Speedy RIP". Babel's main selling point is that it is primarily a loop-avoiding protocol that was originally designed for wireless ad-hoc networks. However, it is still very stable and usable in wired networks.
- Babel limits the frequency and duration of routing paths when a path is dropped. This being said, Babel is highly efficient at reconvergence in such a scenario, reconverging to an optimal configuration using `sequenced routes`. This is done based on the Bellman-Ford algorithm, using a technique called "Destination Sequenced Distance-Vector" (DSDV) routing.
- Babel is a "double-stack" routing protocol. This means that Babel supports routing for both IPv4 and IPv6.
- Babel can automatically detect wireless and wired interfaces and adjust accordingly.

4.1.2 Step 0: Preparation (see Appendix A)

- Launch GNS3. Create the network topology as seen in figure 1 below.

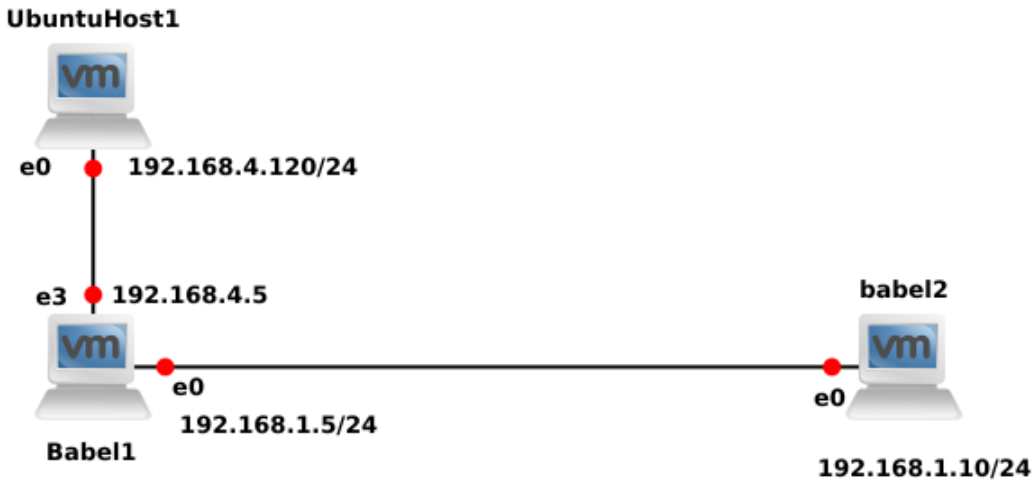

Figure 1: Preliminary network topology

- For this lab we are going to configure Ubuntu Server VMs as routers. To do this, create linked clones of the VMs. We only need 3 (2 Ubuntu Servers as routers and

1 Ubuntu Desktop host) for this exercise but later we will need more.
- In VMWare make sure to add VMnets for each interface. For now, we only need 4.

4.1.3 Step 1: Babel Configuration

- Once we have set up the VMs and given their interfaces IPs, we can start Babel.
- On babel1 and babel2, start the Babel routing daemon with the `sudo babeld -d1 eth0` command.
    - Q1 - What does the "-d1" option do?
- Next, start Wireshark on the link between babel1 and babel2. You should be able to use the "babel" capture filter and see lots of packets.
    - Q2 - What is the destination address shown for the Babel packets? What is its significance?
    - Q3 - What does the "ihu" shown in some of the packets stand for? What does "nh" stand for?

4.1.4 Step 2: Babel Redistribution

- On the Ubuntu Desktop Host, try pinging babel2. This will fail.
    - Q1 - Why can't the Ubuntu Desktop ping babel2? (Hint: Wireshark)
- On babel1, stop the Babel daemon and start it again with this command: `sudo babeld -d1 -C "redistribute metric 128" eth0`. This tells babel to redistribute all routes with a metric of 128.
- Try pinging babel2 from the host again. This time it should be successful.
    - Q2 - Check Wireshark and find out what's new in the Babel packets.

4.2 Exercise 2: Setting Up a Mesh Network & Convergence Times

4.2.1 Step 0: Preparation

- To begin this exercise create two more linked clones of the babel ubuntu Server VMs and 1 more linked clone of a client VM of your choosing(we used Ubuntu Desktop VMS).
- In the virtual network editor in VMWare we will need:
- 4 host-only vmnets w/o DHCP for VM1(Babel1).
- 3 host-only vmnets w/o DHCP for VM2(Babel2).
- 3 host-only vmnets w/o DHCP for VM3(Babel3).
- 4 host-only vmnets w/o DHCP for VM4(Babel4).
- You will also need 2 additional vmnets host-only w/o dhcp for the 2 linked Ubuntu Desktop VMs (or whatever you choose to be your clients).
- You will need a total of 16 vmnets
- On VM1 (Babel1) add 4 network adapters and select any of the custom vmnets so that they are different for each interface.
- Repeat the same procedure for VM4(Babel4).
- On VM2(Babel2) add 3 network adapters and assign different vmnets than the ones selected for VM1 and VM4.
- Repeat the Same procedures for VM3(Babel3).
- Connect all the links and assign static IPs to all of the interfaces as shown in Figure 2.
- Now babeld should've been installed on the VM you created your linked clones from, verify that it's installed using the command `babeld --version` on all of your babel VMs.
- Make sure your client VMs can pinged there respective gateways (which should have been added when assigning static IPs to them)

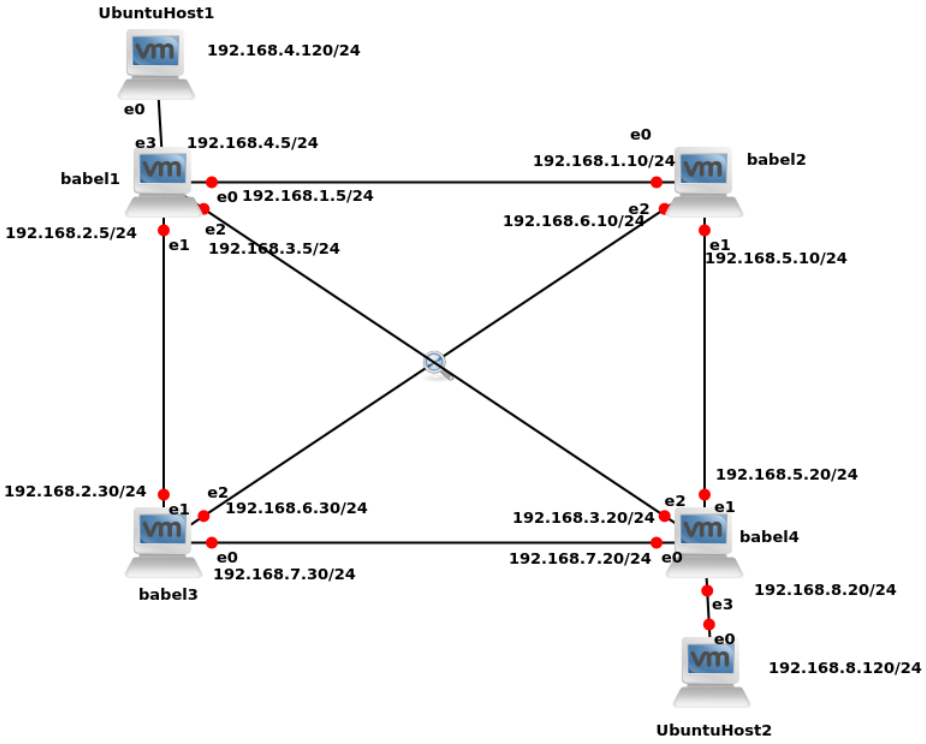

Figure 2: Network Diagram for Exercise 2

4.2.2 Step 1: Convergence Time in Comparison to RIP

- Next, we are going to start babel on the 4 VMs that are acting as routers.
- On Babel1, from the terminal launch the command `sudo babeld -d1 -C 'redistribute metric 128' eth0 eth1 eth2`. You should start to see some debug messages after a few seconds.
- On Babel2, start babel with the command `sudo babeld -d1 eth0 eth1 eth2`
- Repeat the same command as Babel1 on Babel4.
- Repeat the same command as Babel2 on Babel3.
- Next, we are going to test the convergence time of bringing a link down
- Start pinging UbuntuHost2 from UbuntuHost1.(should be successful at this point)
- Next, bring down interface eth2 on Babel4.
- Watch the ping on Host1 and record the time from when it pauses to when it begins to to sending request and receiving replys again.

◦ Q3 - How long does it take for the network to converge to the new route after brining eth2 down?

◦ Q4 - How does the convergence time compare to RIP from Lab 2?

- Next, launch wireshark on the link between Babel1 and Babel4 and the link

between Babel1 and Babel3 (or which ever link is currently capturing the ICMP packets)
- Bring interface eth2 back up on Babel2 and watch the Wireshark capture on the link between Babel1 and Babel2.

◦ Q5 - How long does it take for the icmp messages to return back to the original route? How does this time compare to that of Lab 2?

4.2.3 Step 2: Routing Tables Affecting Traceroutes

- Verify that traceroute is installed on Host1.
- Do a trace route from Host1 to Host2 and record the path.
- Launch wireshark on the Link between Babel1 and Babel4, Babel1 and Babel3, Babel1 and Babel2, Babel2 and Babel4.
- Bring interface eth2 down on Babel4 again.
- Run traceroute again (you may need to keep trying until the convergence is complete.

◦ Q6 - Describe what you observe and provide a screenshot. Anything interesting that you noticed. Hint: The route should be somewhat puzzling.

◦ Q7 - What do you observe in the Wireshark captures as the reason behind what you discovered in the traceroute? Provide evidence in the form of screenshots and Wireshark captures.

4.2.4 Step 3: Compare and Contrasting RIP and Babel

- Take a look back at Lab 2 particularly at how static routes were distributed in RIP.
    - ◦ Q8 - What commands did you use to redistribute static routes in RIP? How did we redistribute the static routes using Babel?
    - ◦ Q9 - Referring to your answer to the previous question about Babel, does this command only redistribute static routes?
- Another major difference between RIPv2 and Babel is their operating environment. RIP runs on Cisco IOS and we have Babel running on Ubuntu Servers.
- Babel works with both IPv4 and IPv6 by default. In RIP, you need to make separate configurations for IPv6 compatability.
    - ◦ Q10 - What protocol is used by Babel that is not used by RIPv2 by default? What protocol do they both use?

4.3 Exercise 3 - Setting up Babel Topology with physical routers

4.3.1 Step 0: Preparation (Installing Open-Wrt, preliminary physical topology)

- This exercise will involve the use of multiple access points flashed with the open-wrt firmware.
    - Q1 - What is the purpose of flashing the Access point with OpenWRT, rather than keeping DD-WRT?
- To prepare, flash your access point using the 30-30-30 method.
- Next, if the open-wrt frame work is not installed, download the firmware provided in the attachments. Make sure that the access point is connected via wire-line to the station that you have downloaded the firmware on.
- In the Administration tab under the DD-WRT settings, select the "Firmware Upgrade" subtab. Set the first field to "Don't reset", navigate to the firmware file you just downloaded, and click `Upgrade`.
- This step will take a while, so wait patiently until the entire process is done. Once completed, provide proof that you can access the open-wrt homepage.
- When asked to enter credentials, change the new credentials to username: "root", and password: "admin" without quotes. Be sure to enable ssh with these new credentials, as we there will be much configuration from the command line.

4.3.2 Step 1: Flashing and configuring Babel on Open-wrt

- Now that you have a router flashed with open-wrt, we will now begin to configure the openWRT router through the browser and later the terminal.
- In a browser, navigate to the default address of `192.168.1.1`, and navigate to the "Interfaces" tab under Network.
- Here, you will see an interface named "br-lan". Edit the interface so that the default address is `192.168.10.1`. Save and apply the changes. Note that you may have to exit out of the browser and navigate to the new address that you just assigned. This may take a few tries.
- Now, under the Network / wifi tab, you will see an interface known as "OpenWRT". Edit this interface, and change the SSID of the network to be `OpenWRTBabel`, the Mode to "802.11s", and the Channel to "7". Make sure to Save and Apply. When returned to the Wireless Overview tab, be sure to enable the interface.
    - Q10 - What is 802.11s and why is it used?
- Now, open a terminal, and ssh to the open-wrt router with the credentials you just configured.

  ssh root@192.168.10.1
- Once connected, install the babel daemon with the following command:

  opkg install babeld kmod-ipv6
- If this command does not work, be sure that the package manager is up to date by updating it with:

  opkg update
- Also make sure that the router is connected to the Internet in order to download the packages.

- Let us set up the babel daemon to start up on boot on the wireless interface.

  /etc/init.d/babeld enable
- Provide proof that you have ssh'd to the open-wrt router and have installed the babel daemon.
- Now that you have gone through the basic babel configurations, we will now configure the firewall on the router to allow babel traffic to flow through.
- In the `/etc/config/firewall` file on the router, add the following lines at the end of the file. You will have to use the vi text editor to do so.

```
config rule
        option name             Allow-Babel
        option src              wan
        option dest_port        6696
        option proto            udp
        option target           ACCEPT
```

Figure 3

- While OpenWrt supports the "802.11s" standard directly in the gui, it does not actually provide a place to specify the mesh id. To do so, edit the `/etc/config/wireless` file on the router, once again with vi, and add the following line at the end of the last interface of the file.

  option mesh_id "meshTest"
- Be sure to restart the firewall and network adapter to ensure that the changes take place.

  /etc/init.d/firewall reload
  /etc/init.d/network reload
- Back in the terminal ssh'd to the router, run the following command:

  babeld -d1 br-lan
- The previous command will allow you to test that the babel daemon is up and able to be ran on the br-lan interface. Stop this when you start to see messages.

4.3.3 Step 2: Configuring a Topology with multiple Babel enabled Physical routers.

- Using the same method in step 1 used to configure the 192.168.10.1 router, configure 3 more routers with addresses, 192.168.11.1, 192.168.12.1, and 192.168.13.1. This configuration will involve the installation of the babel daemon.
- We will now use these newly configured routers to create a mesh network architecture.
- Record all of the MAC addresses of the routers.
- Now that all of the MAC addresses had been recorded, for each one of the access points, change the type of connection under the wifi tab from "Access Point" to "802.11s", as done before. Be sure to edit each of the configurations of the routers to include the mesh id of "testMesh" in the /etc/config/wireless file, as done before.

- Before you go any further, ensure that you have a station for each of the routers so that they are available for ssh connectivity. Also, ensure that you have installed tcpdump on each of the routers, as the OpenWrt framework does not support wireshark directly.
- On a terminal for each of the stations, launch tcpdump on the br-lan interface.

  tcpdump -i br-lan -w babelTraffic.pcap
- Make sure that the file you're writing to is unique for each capture, not just "babelTraffic.pcap".
- Now that the mesh network is configured, begin to run the babel daemon on each of the routers, in a similar manner to the method in step 1.

  babeld -d1 br-lan
- Find each of the routes of each of the routers with:

  route -n
- Make sure that you see the debug messages in each of the babel daemon terminals. Now, open wireshark on each of the stations connected to the babel enabled routers. At first, there will only be debug messages that advertise the respective routers information. Soon, you will begin to see neighbor packets, along with an ipv6 and ipv4 address, next hop, and much more. Be sure to check the routing table again, and compare it with the routing tables before the babeld daemon was run.
- With wireshark still running, execute a ping from the 192.168.10.1 router to the others, and vice versa. You should now have full connectivity within a mesh network! Stop the captures on each of the tcpdumps, and export them to be viewable in wireshark.

4.4 Exercise 4: Babel/Rip enabled Hybrid Network Architecture

4.4.1 Step 0: Preparation

- Next, we are going to create a hybrid mesh network using the virtualization software Common Open Research Emulator (C.O.R.E) hosted by the U.S. Navy and created by Boeing. C.O.R.E is capable of emulating WIFI down to the physical layer and you can get creative with the 802.11 configuration, changing properties like jitter, transmission power, system noise and many others.
- Go to https://downloads.pf.itd.nrl.navy.mil/core/vmware-image/ and select the vcore-4.7.zip file to download the Vmware Image.
- After, unzipping, open the image in VMware, give the VM some more processors and around 4 GB of memory and start the VM.
- The C.O.R.E VM runs a version of Ubuntu called LUbuntu. We can use things such as aptitude for package installation.
- The routers in C.O.R.E are version of quagga docker and they have babel listed under services, but we were unable to configure and run Babel through the quagga vty shell. Instead we will download Babeld directly from the Github repository.
- Ensure git, gcc, wireshark, tcpdump are installed on the Core VM (root password is core).
- Install VMWare Tools.
- reboot the VM.
- from your home directory use the command `git clone https://github.com/jech/babeld` to clone the repository on the system.
- Next cd into the babeld directory. use the command `make` to compile the babel daemon executable.
- Copy the binary file babeld to /usr/sbin/.
- Verify that the program works by running the command `sudo babeld -d1 eth0`. You should see a debug message.
- In order to use Wireshark inside CORE, you need to edit the CORE launch configuration so you can use sudo to launch it.
- cd to the desktop in a terminal, and open CORE in leafpad, `sudo leafpad CORE`. Edit the line Exec=... by adding the sudo command after the equals sign.
- Launch core from the terminal `sudo ./CORE`
- You should now be able to launch Wireshark inside CORE emulator.

4.4.2 Step 1: Setting Up Wired Mesh Network with CORE

- After opening the CORE GUI you should see a toolbar on left side of the canvas.
- Select the Router icon (fourth button down), and select the first type of router.
- Click on the canvas to place the router. Create 4 total routers
- Click the link icon in the left hand toolbar to create links between the routers. Refer to Figure 4 for reference to the link toobar button.

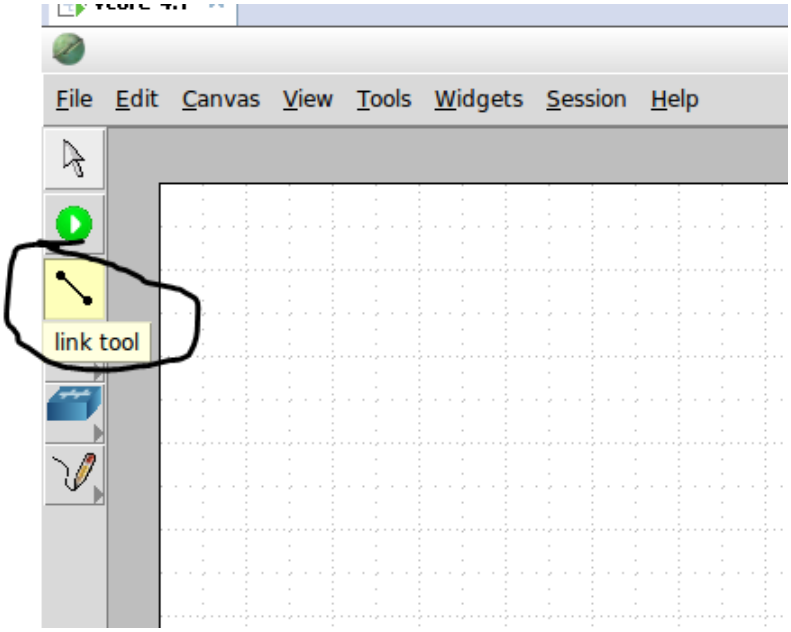

Figure 4: Shows the link tool bar button in CORE

- Create 2 ethernet switches connected to routers of your choosing, and 2 host connected to them (click the icon below router icon to create switches. Click the router icon and select host or PC to create a host machine).
- Right click on n1 router and select configure (note you can show interface labels, by click view>show>interface names)
- Here you can change the network address if you wish on the interfaces. At the top click services next to the router button.
- Deselect OSPFv2 and v3 and click babel. Then click the wrench next to the babel button.
- On the tabs at the top click Startup/Shutdown. We are going to create a startup command so babel automatically starts up.
- click the quagga sh command and then the trash can button. We don't need this.
- Under the startup commands, enter `babeld -d1 -D -L babel.log eth0 eth1 eth2`, and then click the notepad with the plus symbol to add it.
- Keep clicking apply until your back at the canvas.
- Next, on n2 (with the switch and host connected to it) right click and go to the services configuration and do the same thing as n1.
- At the Files tab of the babel configuration, enter the file name `babeld.conf` and press the notepad plus. In the text area box, enter `redistribute metric 128`.
- got to the Startup/shutdown tab and delete the quaaga command.
- Enter the startup commands box, `babeld -d1 -D -c babeld.conf -L babel.log eth0 eth1 eth2`, run babel on all the interfaces except the one connected to the switch. Click apply when finished and return to canvas.
- configure n3 in the same way as n2 (the other router with switch).
- configure n4 the same as n1.
- After all of the routers have been configure click the play button in the toolbar.
- Under the widgets drop down menu at the top, select observer widgets and IPv4

Routes. This allows you to hover over a router and see the routing table.

- Ping the host at 10.0.12.10/24 (or whatever your ip address is) from the other host, ours is at 10.0.5.10. This should be successful.
- Launch wireshark on interfaces from within CORE
- In CORE emulator, right click on one of the routers and click wireshark and select any interface (it does not matter right now).
- Click ok on the error message that pops up.
- Identifying the interface to capture on is tricky. I have determined that each interface begins with the prefix veth. These are the interfaces that corresponds to the virtual interfaces inside of CORE. The number that follows corresponds to the n<x> name in CORE. Then you should see .0.41, .1.41, etc...(or some other similar number The first digit after the veth1.x is the interface on that node, so vth2.1.41, would correspond to n2, interface eth1.

Refer to Figure 5 for the final network configuration for Step 1.

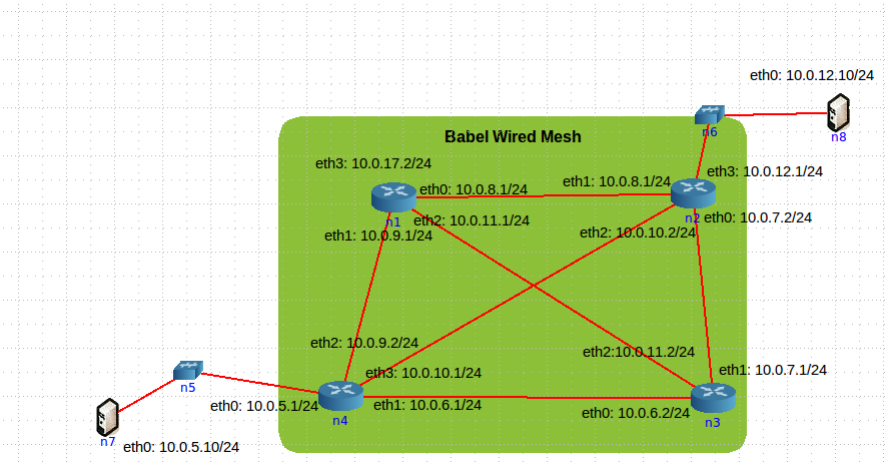

Figure 5: Configuration for Step 1

4.4.3 Step 2: Setting up a Wireless Mesh Network Using C.O.R.E.

- Stop the emulation and click file save.
- Add another router near n1. This is going to be our gateway into the wireless network.
- Next, lets add a wlan to the topology. Click the switch/hub button in the left toolbar and Select the cloud icon. Click to place it on the canvas.
- Right click the cloud and select configure.
- change the Ipv4 subnet to /24 and what ever network address you want.
- Next, click the EMANE tab under wireless, and select ieee802.11abg, under EMANE Models. Note you can configure more customization for the wlan under emane options and ieee80211abg options.
- Next, create 3 or 4 more of these wlan's in the same way and give them different networks and subnets.
- Create links between the gateway router and all of the wlan's and you should see little radio antennas come out of the router.

- Create 2 to 3 more wireless routers and connect them to all of the wlans. Create some host and connect them to a wlan as well.
- Configure all of the wireless routers for the babel service (just like we did on the other routers).
- Don't forget to change the startup command on n1 (you have an extra interface now going to the gateway).
- After all of the routers are configured start the emulation. Try an ping one of the wireless host from a wired host.
    - Q10 - Was the ping successful? If not, then why (Hint: What did we forget to do?)
- Stop the emulation and on the 2 wireless routers (not the gateway for the wired network), click configure, services, the wrench icon for babel.
- Create a babeld.conf like we did for the wired routers, and add the command `redistribute metric 128`. (don't forget to add the .conf file using the notepad icon.
- Next change the startup command to `babeld -d1 -D -c babeld.conf -L babel.log eth0 eth1 eth2 eth3`
- Try to ping from the wired host to the wireless now. Try from wireless host to wired.
- Lets experiment for a second with CORE's wireless emulation. Begin a ping from the wired network host.
- Now start dragging your wireless client icon away from the wireless routers.
- Now, ping again from the wired network and drag the wireless client as far away as you can get from the routers, but where it still has connectivity. Move the 2 wireless routers as far away from the client your pinging as your can.
    - Q11 - What happens? Why?

Refer to Figure 6 for the final network diagram you should have by the end of Step 2.

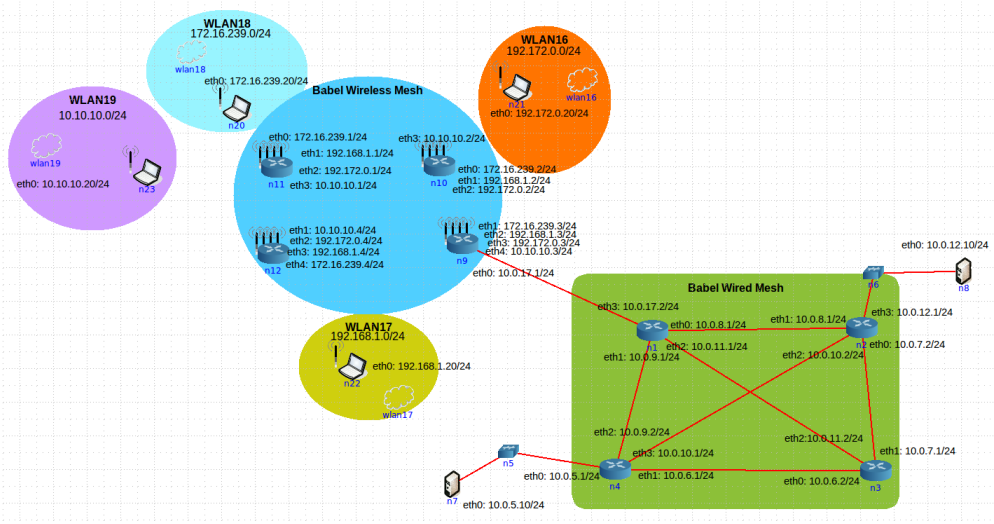

Figure 6: Network configuration for Step 2

4.4.4 Step 3: Adding A RIP network to the C.O.R.E Configuration

- Add 4 more routers with 3 with RIP enabled, the other with RIP and Babel enabled
- Add a switch and a host.
- Connect the host to the switch and the switch to one of the router with only RIP enabled.
- Create a wireline connection from the RIP only router to the other router with Babel and RIP enabled.
- Connect the RIP networks as shown in 7
- connect the Babel enabled router to all 4 WLAN's.
- On the wireless router with Babel and RIP, create a babeld.conf with redistribute metric like the other ones we did.
- Create the babel startup command `babeld -d1 -D -c babeld.conf -L babel.log eth1 eth2 eth3 eth4`
- Start the emulation
- Open a terminal on the Router with RIP and Babel
- Enter the command vtysh, to start the quagga shell
- Now enter conf t, to configure RIP. Enter no router rip first and end, to clear out the default config.
- Next, enter conf t again, the commands are as follows (this will be dependant on your networks address that were auto assigned:
- conf t
- router rip
- version 2
- network 10.0.10.0/24
- redistribute kernel
- redistribute connected
- end
- On the other 3 routers running rip only, configure them the same, but without the redistribute kernel. Also on the router connected to the host you need to do redistribute connected.
- The reason we are using redistribute kernel, is because we are running Babeld (daemon) so it's populating the kernel routing table. Configuring babel through quagga, we would have been able to use the command redistribute RIP, and in RIP configuration redistribute babel.
- The moment of truth, ping from the host on the RIP network to a host on the Wired Babel network.

Refer to Figure 7 for the final network configuration used in Exercise 4.

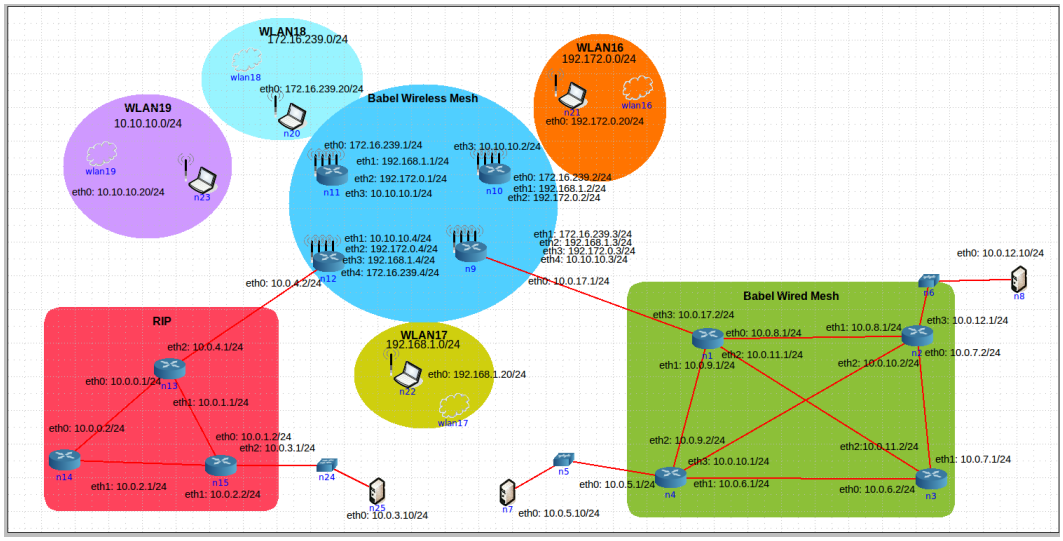

Figure 7: Final Network configuration for Exercise 4

4.4.4.1 Step 4: Wireless Mesh and Link Quality Routing

- In this step we are going to test the link quality routing that Babel and other wireless mesh routing protocols claim to do.
- First, up to this point we have not been using any of Babels optimizations for wired networks, link quality routing and split-horizon.
- Edit the babel config files on the wired network routers to look similar to Figure 8.

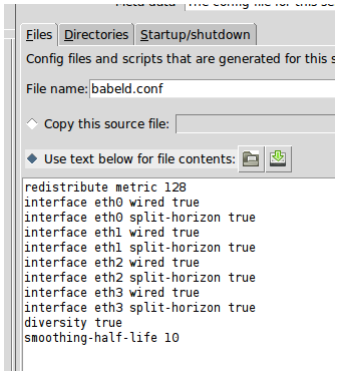

Figure 8: New configuration file for Babel Wired Network

- Next, change the babel config files on all of the wireless routers to look similar to Figure 9

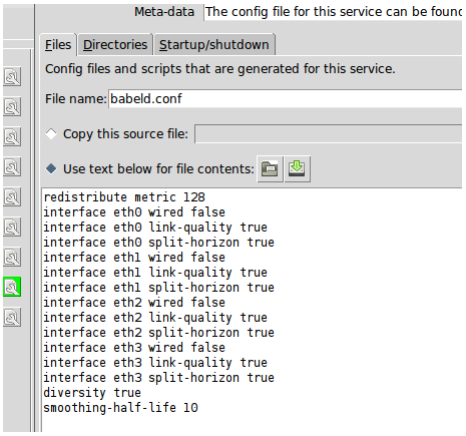

Figure 9: New configuration file for Babel Wireless Network

Note: You will need to configure your file based on interfaces your are running babel on. The nodes that have wireless and wired interfaces will need to have wired specified as true or false based on what that interface is

  - Q10 - Explain what the diversity command is for.
  - Q11 - Explain what the smoothing-half-life command is for.

- Next, start the emulation up (make sure you have the Widget Observer enabled for IPv4 Routing)
- Next, hover over the wireless AP that is running RIP and Babel. Watch network 10.0.5.0/24. (you may need to keep removing the cursor from the node to refresh the table and move it back over top of it)
- Take note of the routing table on the node that links the RIP network to the Babel wireless network. Select a network, we choose the network in the wired babel network where the host was at (10.0.5.0/24).

  - Q12 - Describe what you see. What do you notice? Why?

- Start a ping from the host in the RIP network to a host in the wired Babel network. **Note: you need to go back through the RIP configuration again on all of the routers. There is no way to save the configuration, so when the emulation is stopped the configuration is wiped**
- Start moving some of the nodes around and keep checking the routing table on the RIP/Babel node.
- Keep an eye on the ping running in the terminal you may temporarily loose connectivity while convergence is happening.
- Next, start a ping from the Wireless router running RIP and Babel to a host in the Babel wired network.
- Launch Wireshark on the router. Selecting the interface is a bit tricky as the labels for the wireless interfaces do not seem to have any method behind their naming convention like the wired interfaces. Select an interface labeled b.....something. There should be a bunch of packets on the interface from the ICMP we are sending.
- Continue capturing ICMP packets for a while. Stop the capture and save/export the pcap file.

- Q13 - Describe your observations from the ICMP messages on the Babel/RIP router. Provide Evidence (wireshark and screentshot)

4.5 Exercise 5: Security Issues with Babel Protocol

4.5.1 Step 0: Introduction

- While Babel is a lightweight and robust routing protocol that can prove to be highly effective in certain network configurations, it is not a protocol that is inherently secure. This is due to the large versatility and applicability of the protocol, being used in wired-wireless hybrid networks, lossy/unstable radio networks, etc.
- As stated in RFC 6126, any attacker can attract data traffic by advertising routers with a low metric. This being said, Babel also sends a lot of information to the entire routing domain when adveristing the relevant information. This could allow an attacker to determine the exact location of the node with decent accuracy / precision.
- Because of the insecure nature of the Babel protocol, there are many potential attacks that could passively, or actively be performed on a Babel Enabled Network. Some of these include:

(a) Active attacks:

Routing table poisoning

Lower metric attack

Higher seqno attack

Replay attack

Amplification through routing table poisoning

Amplification due to requests

Covert channel

(b) Passive Attacks:

Stable node identifiers

4.5.2 Step 1: Mitigations and Solutions

- To avoid some of the active attacks that are possible to implement in a babel enabled network, there is a cryptographic authentication mechanism that offers replay protection, defined in RFC 7298
  This RFC deals with "Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication.
- Mentioned earlier, as a double-stack procotol, Babel is capable of carrying traffic over both IPv4 and IPv6. This being said, when using IPv6, Babel packets are ignored unless they are sent via link-local IPv6 address. Because of this, reasonable protection is provided against altered or spoofed Babel packets from the global internet. This inherent protection is not the case in IPv4 however, more common.

5 Additional Questions

6 Deliverables

7 References

8 Appendices

8.1 Appendix A - Timeline/Milestones

1.

8.2 Appendix B - Weekly Status Updates

Status #1 TO Do List - 4/3/2017

(a) **Pre-class - Status of Last Week's Action Items and/or Milestone due this week**

i. We have Babel protocol running on quagga docker in GNS3.
ii. Basic configuration
iii. Captured some initial babel hello packets

(b) **Post-class - Action Items for Next week and/or Milestones due next week**

i. Dig deeper into configuring babel
ii. setup a more complex network and capture babel packets to learn more about the protocol
iii. Test babel's loop avoidance feature.
iv. Configure Babel to use IPv4 rather than IPv6
v. Explore possibilities of using BABEL in a wireless Mesh network architecture
vi. Limitations to Protocol?

Status #2 TO Do List - 4/10/2017

(a) **Pre-class - Status of Last Week's Action Items and/or Milestone due this week**

i. Enable babel routing on a network of quagga routers.
ii. Revamped topology, babeld now running on Ubuntu Server VMs (For updated version of babel)

(b) **Post-class - Action Items for Next week and/or Milestones due next week**

i. Create a hybrid topology that allows gns3 to talk to a wireless client via babel
ii. inject ospf or rip into babel (redistribution)
iii. Break links and see how babel recreate the links. (Ex2)
iv. create a physical topology using routers flashed with open-wrt (Ex3)
v. Creation of hybrid architecture? Connect GNS3 topology to physical world (Ex4)
vi. Implement BMX6 routing protocol in Physical world? (Ex5)

Dry-Run #1 TO Do List - 4/17/2017

(a) **Pre-class - Status of Last Week's Action Items and/or Milestone due this week**

i. Exercises 1, 2, and 3 Instructions are all partially completed
ii. Exercise 4 is defined (Hybrid Network Architecture)
iii. Investigated link re-convergence time when a link is broken and how loop avoidance and load balancing is maintainted. (Ex2)
iv. Have implemented and successfully configured babel on a single access point flashed with Open-wrt. (Ex3)
v.

(b) **Post-class - Action Items for Next week and/or Milestones due next week**

i. Finish Report for each of the exercises
ii. Implement Babel between 2 APs flashed with Open-WRT.
iii. Implement Babel in a Hybrid architecture between a Wireless AP and GNS3.
iv. Come up with Exercises 5 and 6.

Dry-Run #2 TO Do List - 4/24/2017

(a) **Pre-class - Status of Last Week's Action Items and/or Milestone due this week**

i. test

(b) **Post-class - Action Items for Next week and/or Milestones due next week**

i. test