DTN DevKit – Hands-On Portion of the ION Course



Purpose

If you have issues, please send questions to: kscott@mitre.org

- Part of the ION course involves hands-on labs:
  - Visualize how ION works and how data flows using ION
  - Get experience configuring ION nodes
  - Code a simple ION application

- The labs use a pre-built virtual machine that comes with
  - ION
  - An emulation mechanism with a GUI
  - A number of pre-built scenarios with applications that use ION
    - Separate scenarios will be distributed during the class for use / discussion
  - Tools (Wireshark, visualizations for the ION contact plan, etc.)

- These slides describe how to install the pre-built virtual machine to be ready to run the exercises

Overview

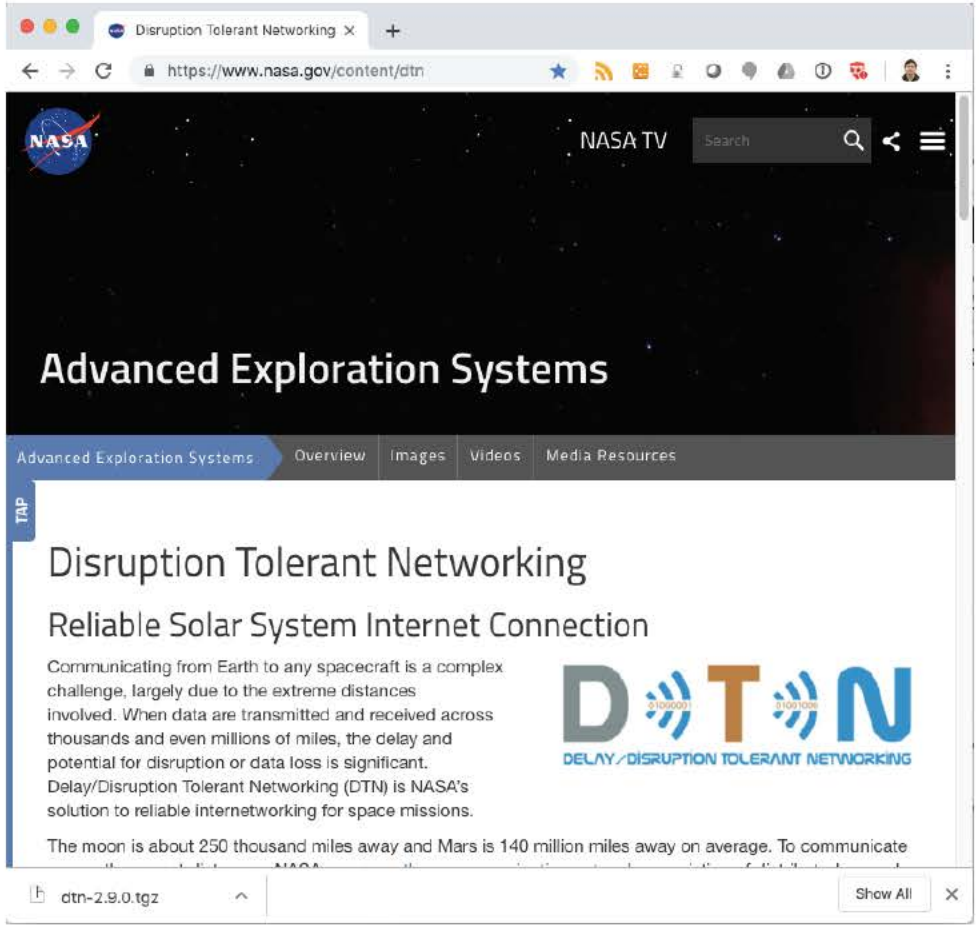
- Install Oracle VirtualBox
  - The pre-built VM is an Ubuntu machine that can be run under Windows or Mac
- Pull the DTNDevKit .iso image
  - https://www.nasa.gov/content/dtn
  - Scroll to the bottom
  - Use the link for the DTN Development/Deployment Kit
    - Note: the username and password are both 'cvm' (no quotes)
- Create a virtual machine in VirtualBox that uses the DevKit VM
  - Creates a VM that boots the DTNDevKit .iso image
- (Optional) Create a mutable copy of the VM
  - The .iso image is fixed – changes won't be saved between reboots
  - 'Installing' the .iso onto your own VM allows changes to persist

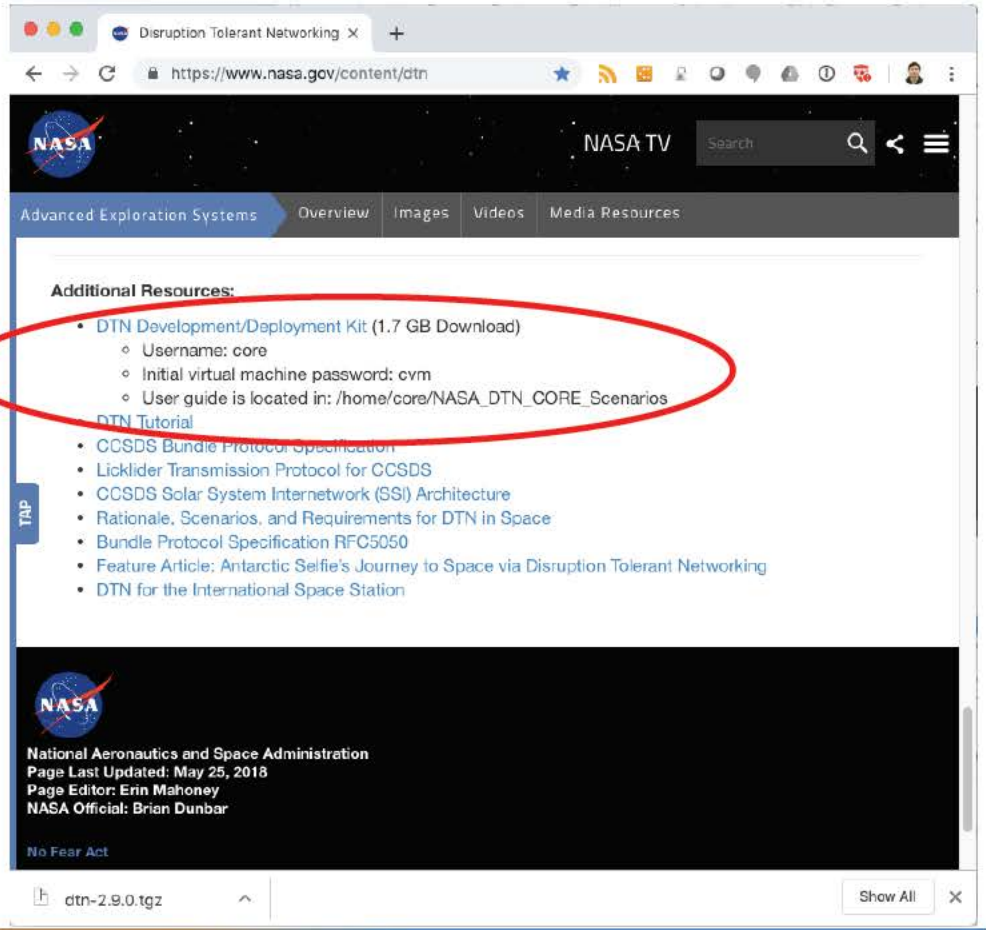
Install Oracle VirtualBox from https://www.virtualbox.org/wiki/Downloads



Pull the DTNDevKit .iso image

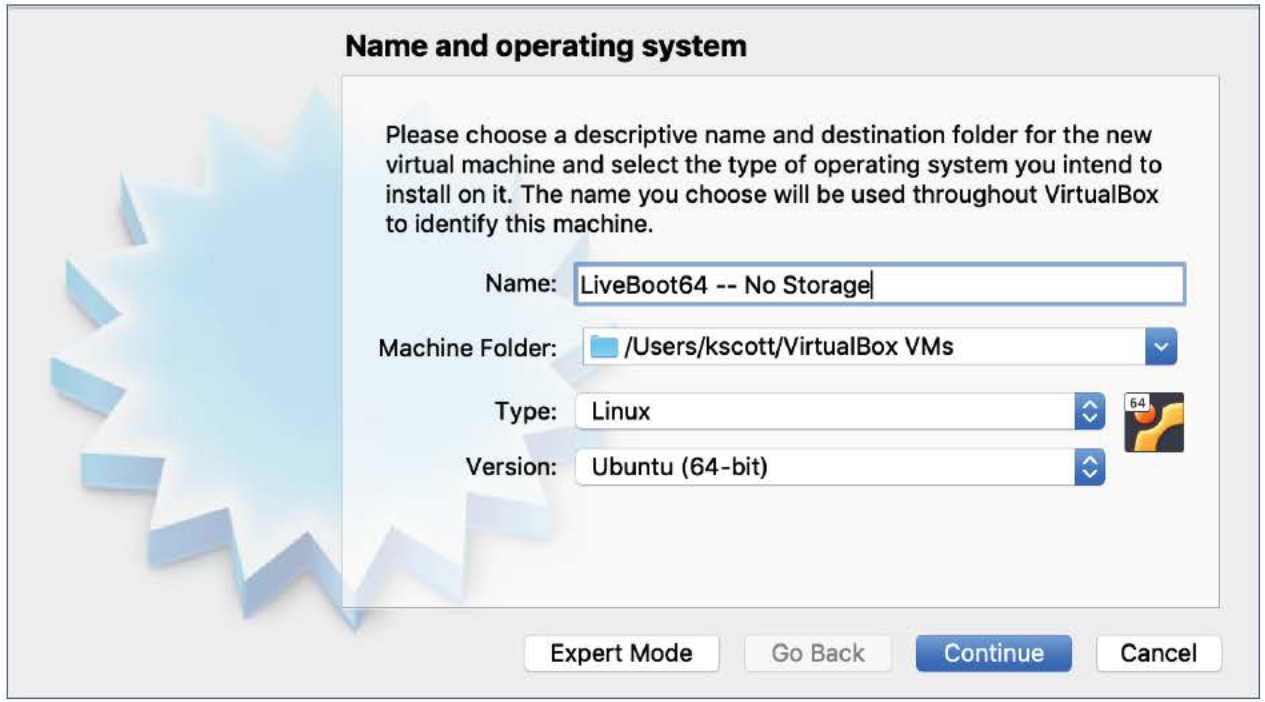
https://www.nasa.gov/content/dtn





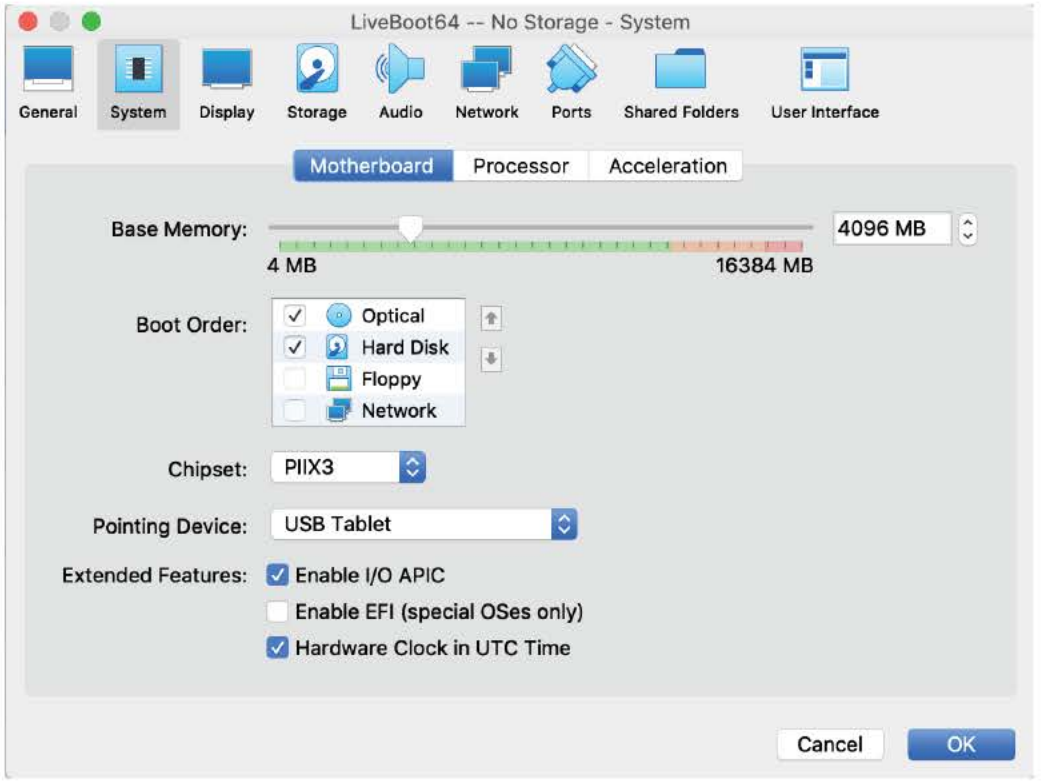
Create a virtual machine in VirtualBox that uses the DevKit VM

- Launch VirtualBox and use the ‘New’ button at the top to create a new VM
  - Give it a name
  - Leave the ‘Machine Folder’ alone
  - Set the type to ‘Linux’
  - Set the version to ‘Ubuntu (64-bit)



Optional – Set VM Parameters

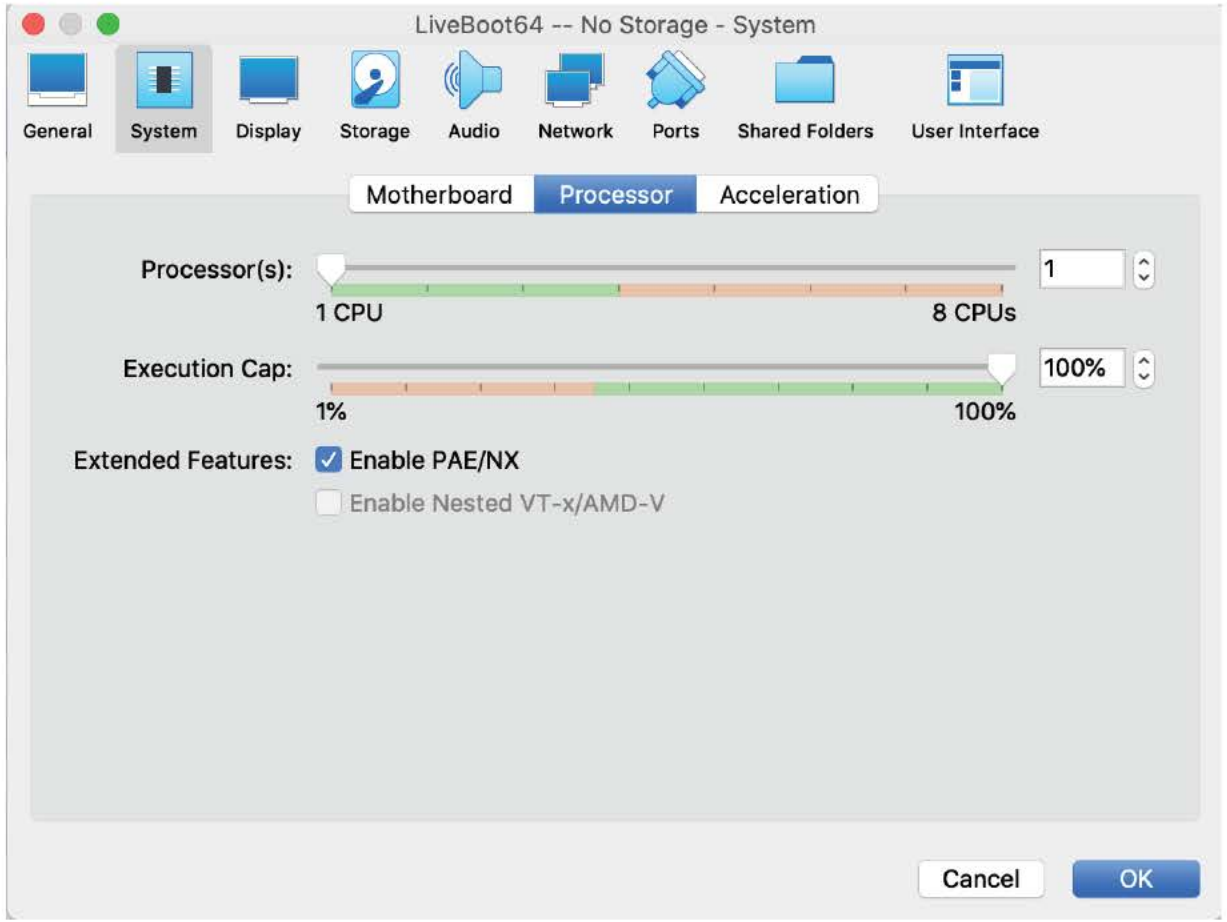
- In the Settings dialog you may want to modify the System parameters for better performance:
  - 'Motherboard' tab:
    - 4GB base memory



Note: these may be different on Windows / Mac. You may need to create the VM and save it before you can modify the settings, for example.

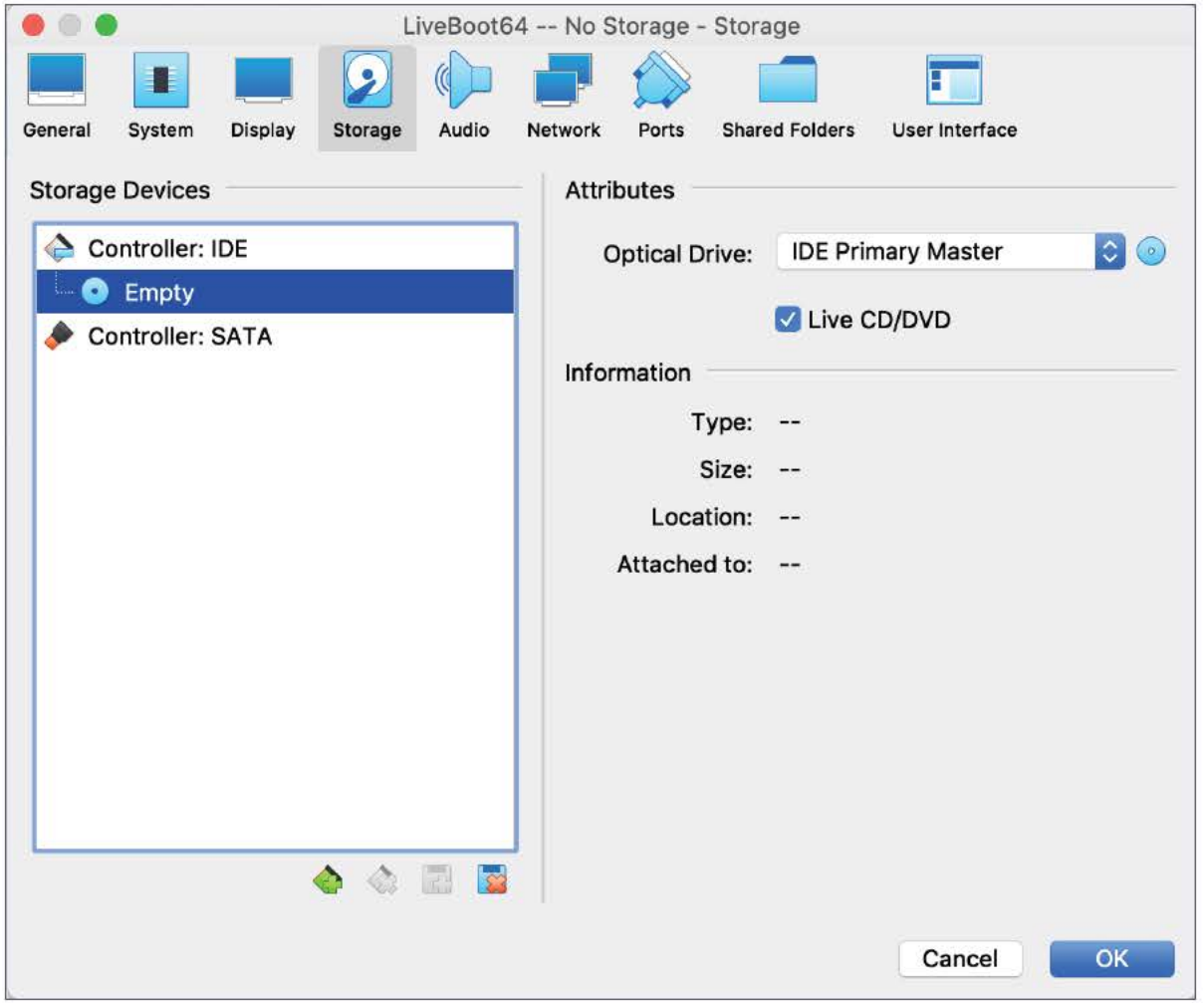
Optional – Set VM Parameters

- ‘Processor’ Tab:
  - 1 CPU should be fine, 2 will make it more responsive
  - Leave the Execution Cap at 100%



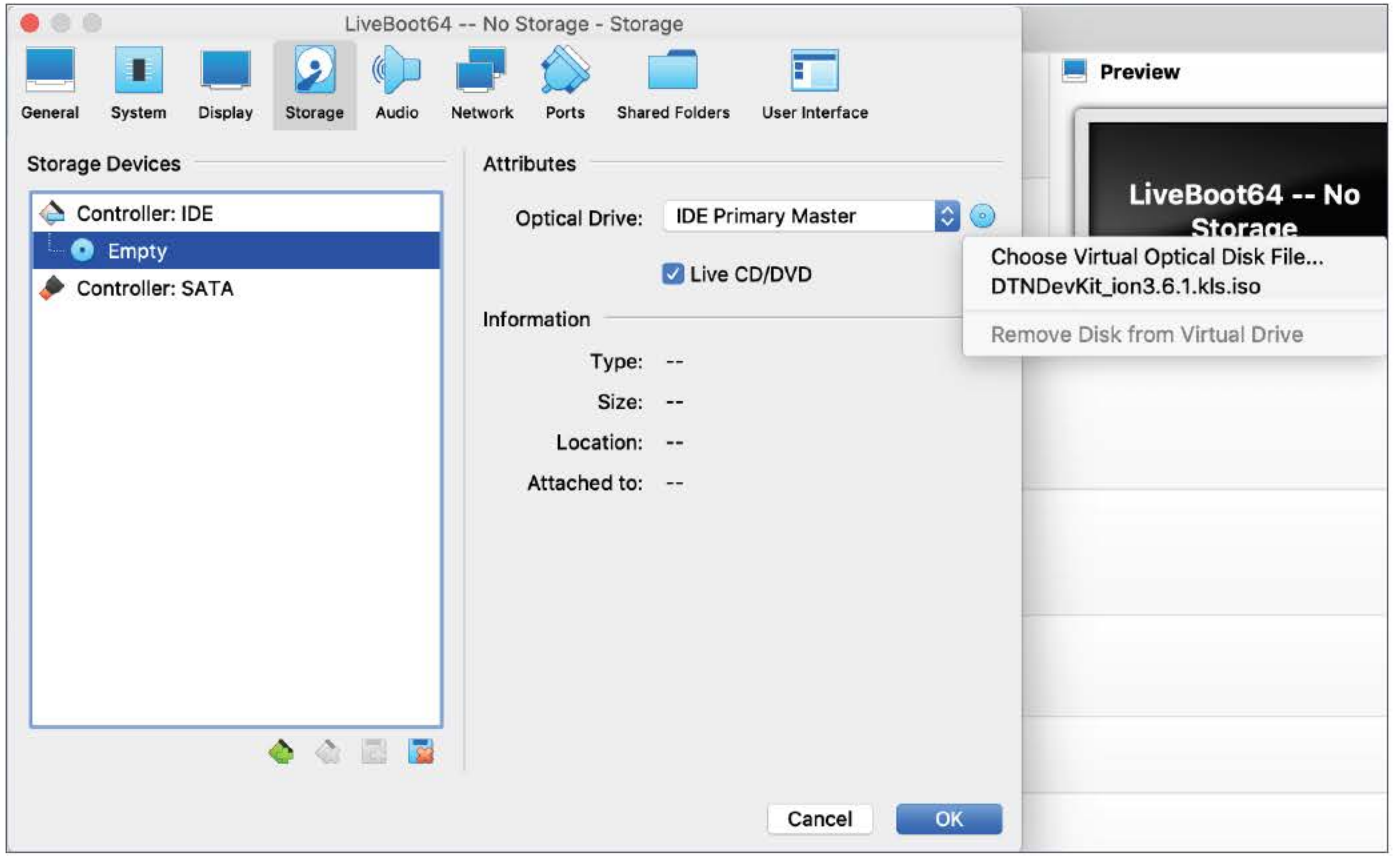
(Not Optional) Set up the VM to boot off the DTN DevKit .iso image

- Using the ‘Storage’ icon in the Settings Dialog, select the virtual CD-ROM drive (the ‘Empty’ disk under the IDE controller) and:
  - Click the Live CD/DVD box to the right
  - Click on the blue disk to the right of where it says ‘IDE Primary Master’ and select the DTN DevKit .iso image



Selecting the DTNDevKit .iso image

- Your image may be named differently than in the picture to the right
- Click ‘OK’



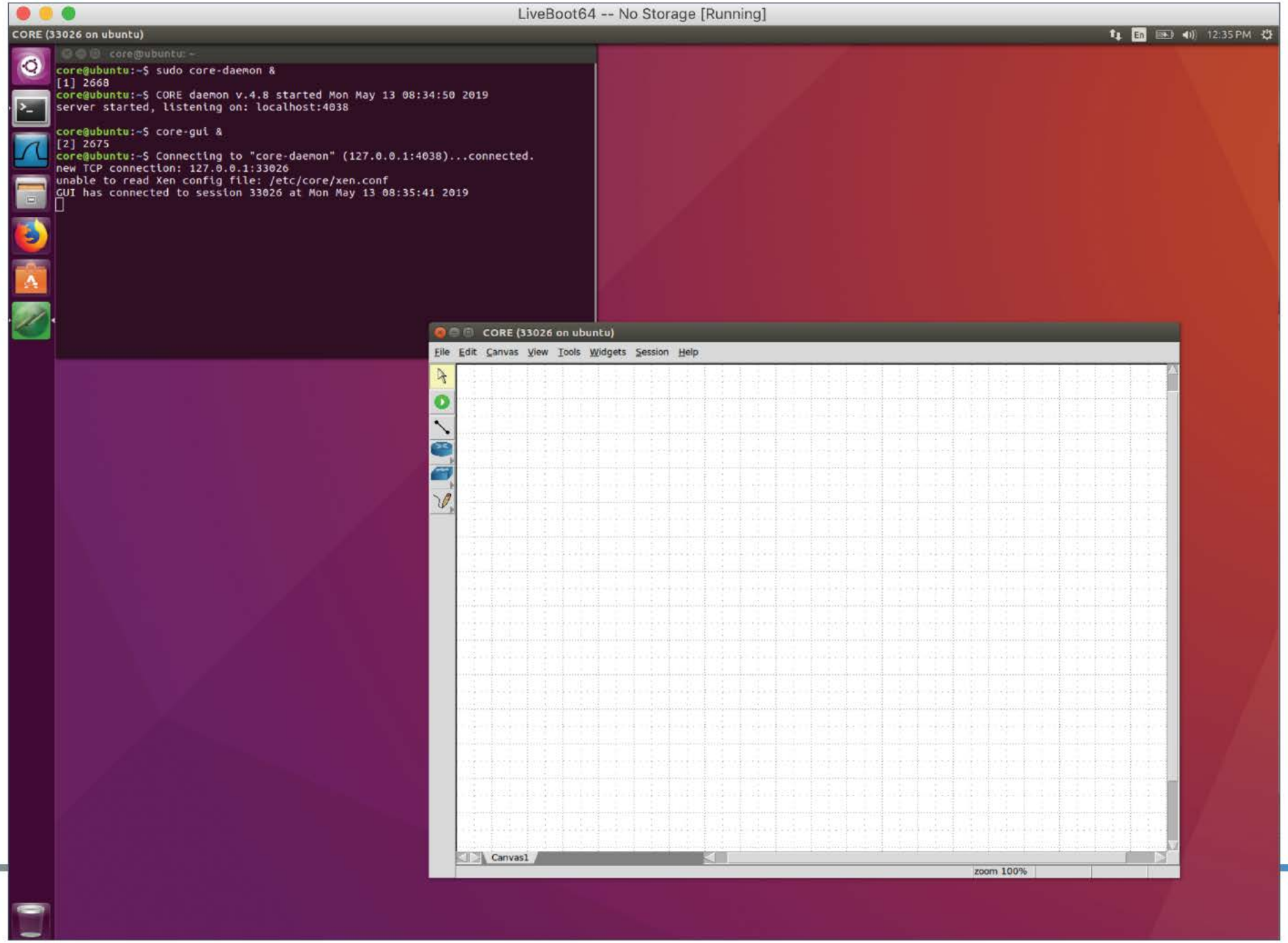
Boot the DevKit

- Select ‘Start’ from the top of the VirtualBox Window
- You can use either ‘Scaled’ mode or ‘Windowed’ mode as you like
  - Scaled mode may allow you to read text easier
- When the VM boots, the ‘core’ user should be automatically logged in (you’ll see a desktop)

Start the CORE Emulation Tool

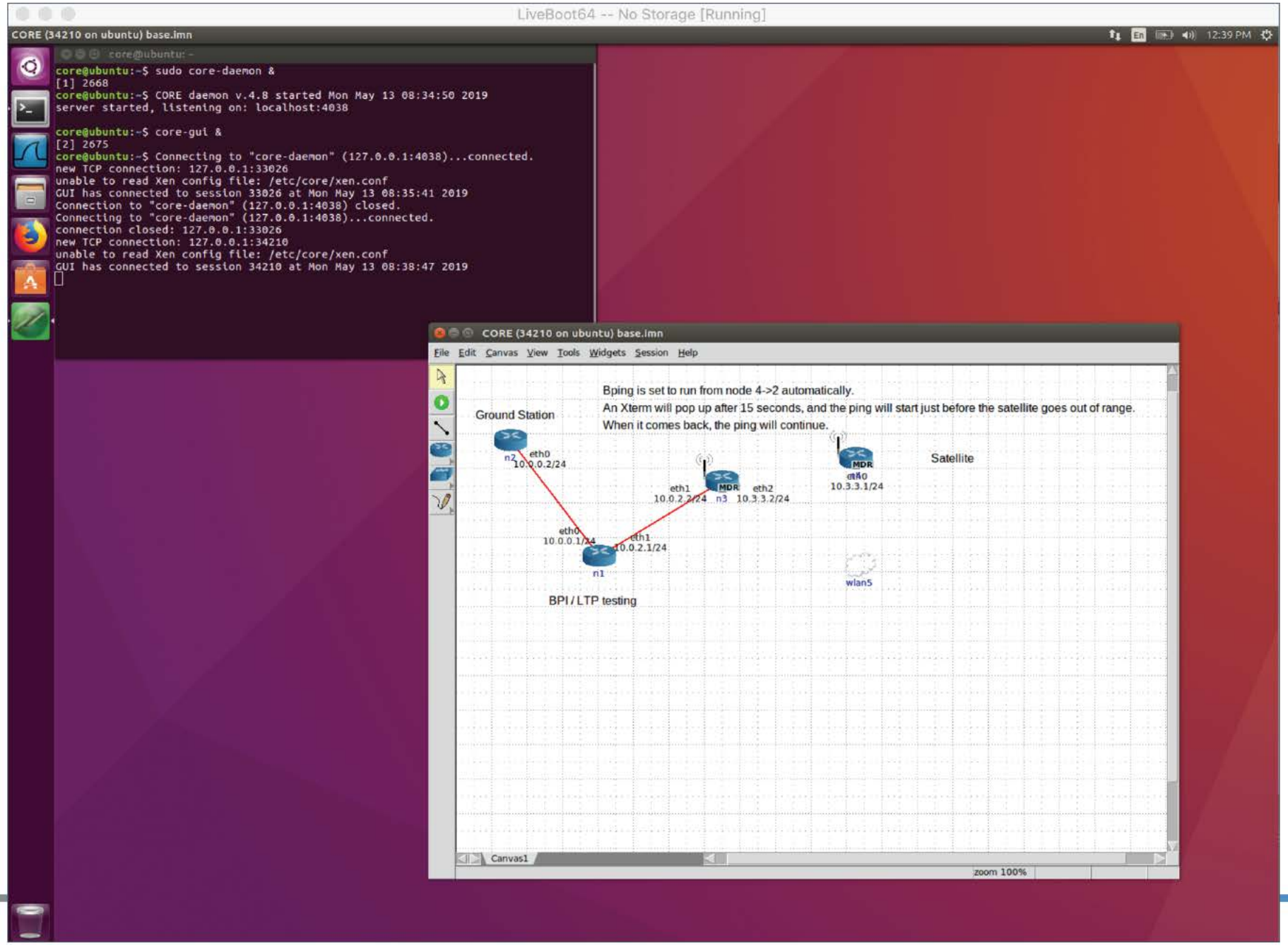
- Click on the terminal icon on the upper left to get a shell (black square with '>_' in it)
- Type 'core-gui &' (no quotes)
  - Starts the GUI front-end for the CORE emulator

- Note: the next screenshot shows starting the core-daemon process; that's now done automatically (you shouldn't have to do it).


LiveBoot64 -- No Storage [Running]
CORE (33026 on ubuntu)
12:35 PM
core@ubuntu: ~
core@ubuntu:~$ sudo core-daemon &
[1] 2668
core@ubuntu:~$ CORE daemon v.4.8 started Mon May 13 08:34:50 2019
server started, listening on: localhost:4038
core@ubuntu:~$ core-gui &
[2] 2675
core@ubuntu:~$ Connecting to "core-daemon" (127.0.0.1:4038)...connected.
new TCP connection: 127.0.0.1:33026
unable to read Xen config file: /etc/core/xen.conf
GUI has connected to session 33026 at Mon May 13 08:35:41 2019
CORE (33026 on ubuntu)
File Edit Canvas View Tools Widgets Session Help
Canvas1
zoom 100%

Open a Scenario

- The ION course has its own set of scenarios that are available separately; these slides describe how to run a scenario from the ‘base’ DevKit
- From the File Menu in the CORE window select ‘Open…’
- Double-click on the ‘NASADTNDevKit’ folder
- Double-click on the ‘base’ folder
- Double-click on the ‘base.imn’ file


LiveBoot64 -- No Storage [Running]
CORE (34210 on ubuntu) base.imn
12:39 PM
core@ubuntu: ~
core@ubuntu:~$ sudo core-daemon &
[1] 2668
core@ubuntu:~$ CORE daemon v.4.8 started Mon May 13 08:34:50 2019
server started, listening on: localhost:4038
core@ubuntu:~$ core-gui &
[2] 2675
core@ubuntu:~$ Connecting to "core-daemon" (127.0.0.1:4038)...connected.
new TCP connection: 127.0.0.1:33026
unable to read Xen config file: /etc/core/xen.conf
GUI has connected to session 33026 at Mon May 13 08:35:41 2019
Connection to "core-daemon" (127.0.0.1:4038) closed.
Connecting to "core-daemon" (127.0.0.1:4038)...connected.
connection closed: 127.0.0.1:33026
new TCP connection: 127.0.0.1:34210
unable to read Xen config file: /etc/core/xen.conf
GUI has connected to session 34210 at Mon May 13 08:38:47 2019
CORE (34210 on ubuntu) base.imn
File Edit Canvas View Tools Widgets Session Help
Bping is set to run from node 4->2 automatically.
An Xterm will pop up after 15 seconds, and the ping will start just before the satellite goes out of range.
When it comes back, the ping will continue.
Ground Station
n2
eth0
10.0.0.2/24
eth1
10.0.2.2/24
MDR
n3
eth2
10.3.3.2/24
MDR
eth0
10.3.3.1/24
Satellite
eth0
10.0.0.1/24
eth1
10.0.2.1/24
n1
wlan5
BPI / LTP testing
Canvas1
zoom 100%

Start the Scenario

- To Start the scenario, click the Green Ball with the white triangle ('Play' icon) on the left
  - Some boxes will show up around the router icons in the scenario and then should disappear
  - You'll see a 'wlan5 mobility script' window show up
  - Node n4 should start moving
  - After a few seconds, a new window titled 'n2 – bping ipn:2.2 ipn:4.1' should show up (and be blank)
    - In this window, there is a bping (like IP ping, but using Bundle Protocol) process pinging from the node on the far left (n2) to the 'Satellite' node on the far right
    - Bundles will queue up when they can't move forward, and pings will eventually work
    - NOTE: the ION contact plan is not *exactly* aligned with the emulation's notion of connectivity; don't expect bundles *immediately* when the satellite is connected to n3



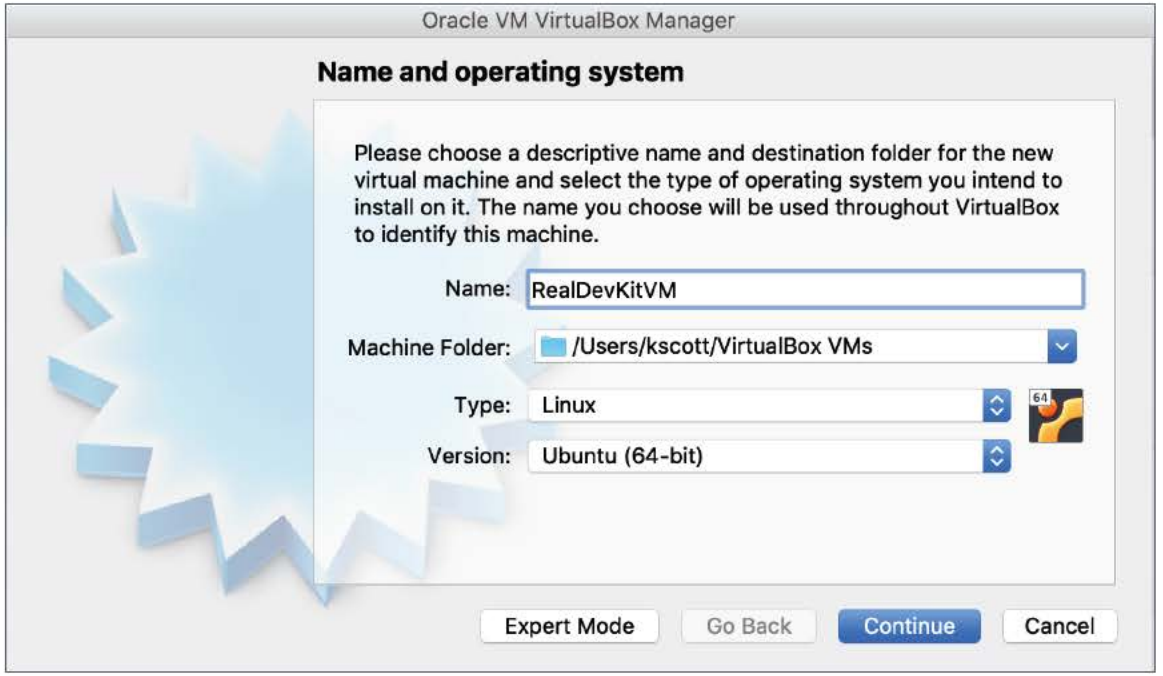
Optional: Make a Mutable Copy of the DevKit

Installing the DevKit onto a VirtualMachine with a Virtual Hard Drive

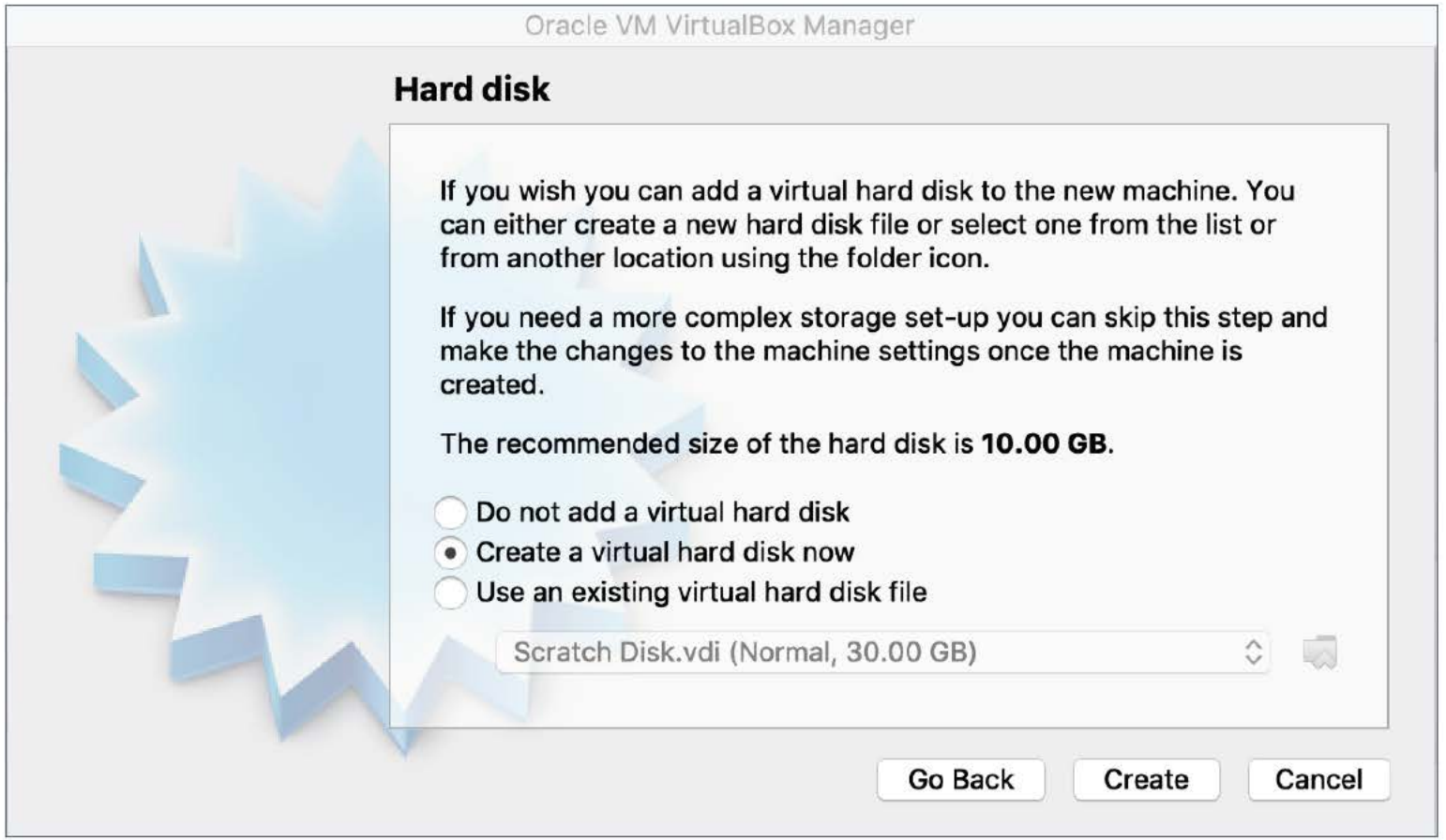
- The .iso image is immutable – every time you boot it you get exactly the same VM – changes you make during a session are NOT persisted
- If you want changes to be persisted, you need to 'install' the DevKit onto a new virtual machine that has a virtual disk drive
  - Create a new VM with a (blank) virtual hard disk
  - Boot the DevKit on that (new) virtual machine
  - Install the DevKit onto the virtual machine

Create a New VM With a (Blank) Virtual Hard Disk

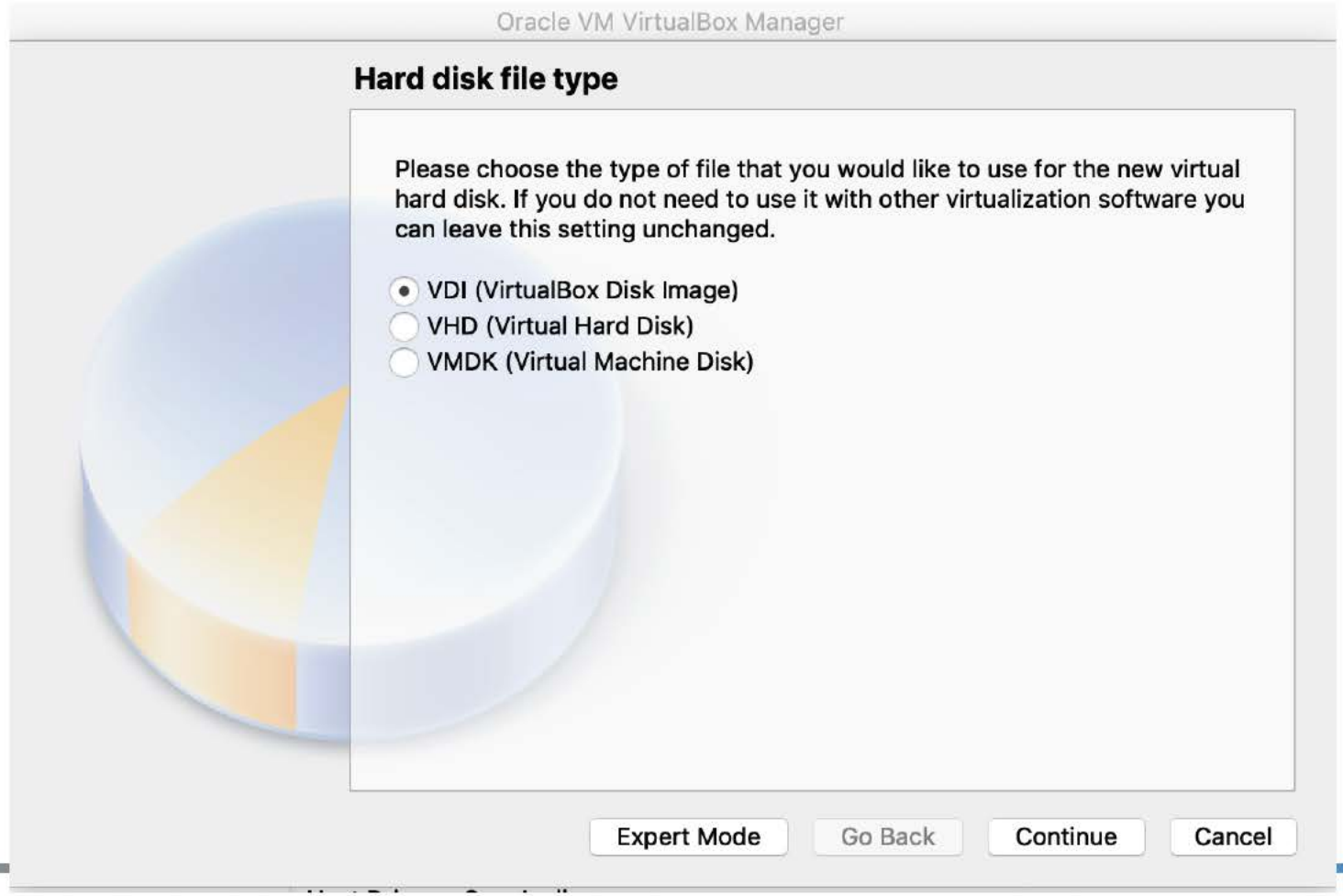
- As you did before, from the main VirtualBox window, create a new VM that is of Type Linux and Version Ubuntu (64-bit)
  - Suggest 4MB of memory



Create the VM with a Virtual Hard Disk



Choose VDI for the Hard Disk File Type

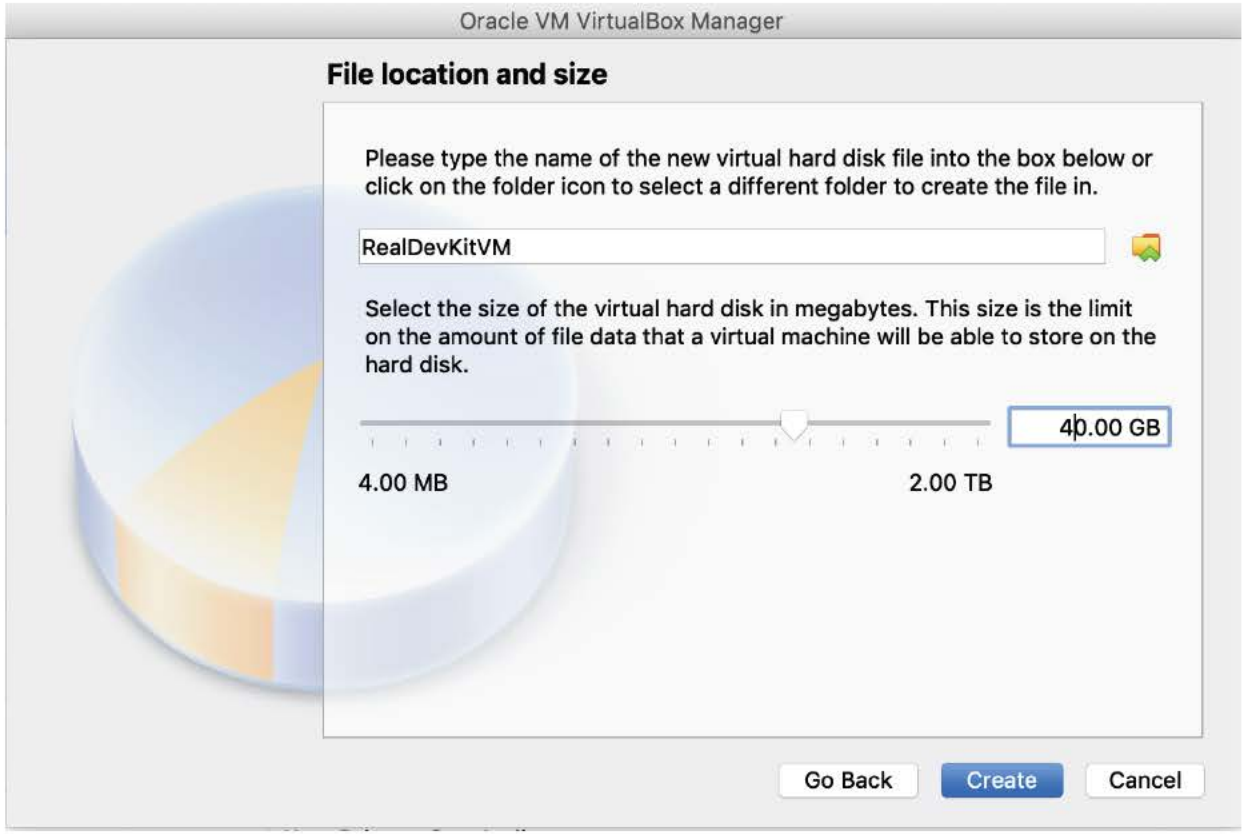


Choose ‘Dynamically Allocated’



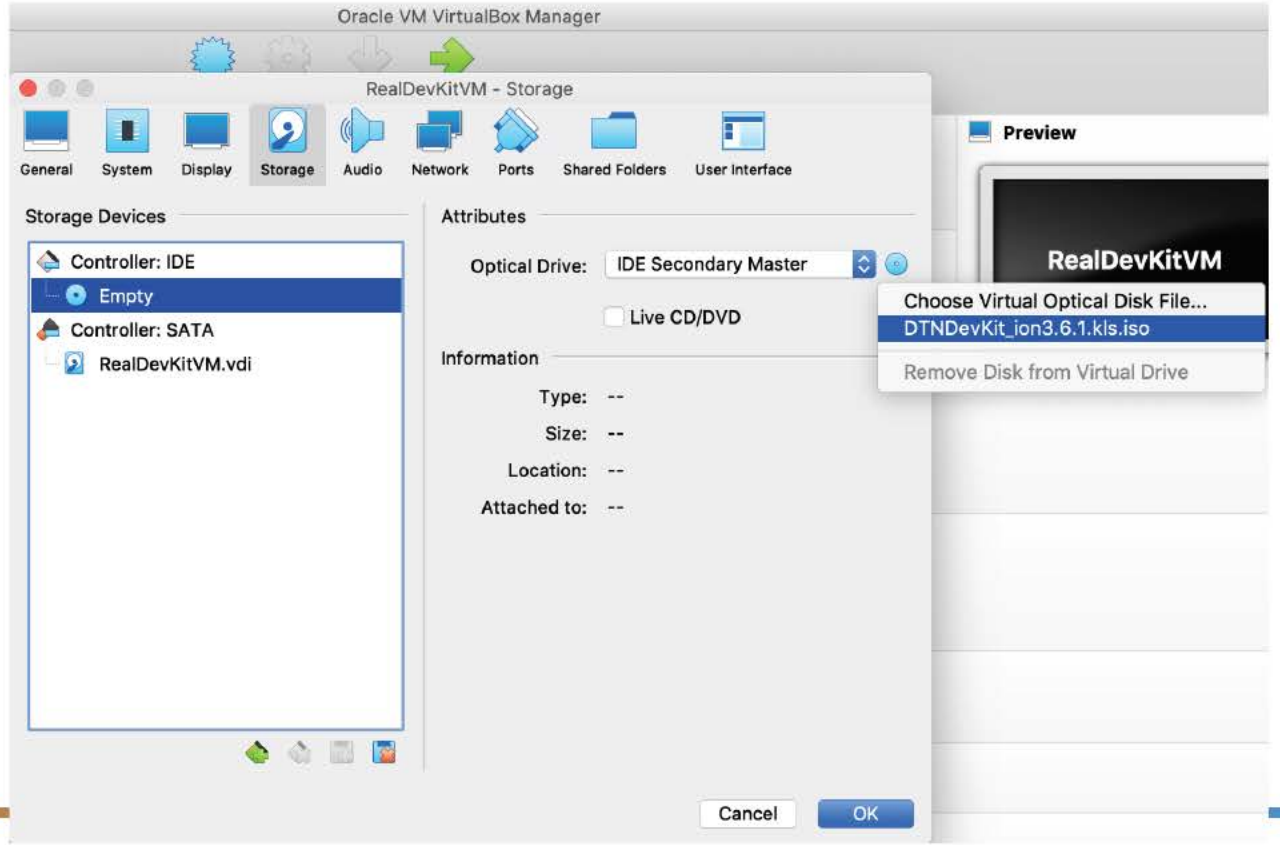
Set The Location and Size of the Virtual Hard Drive File

- Suggest setting the size at 40GB
  - Since it's dynamically allocated, it will only use what it needs



Boot the DevKit .iso

- As before, put the DevKit .iso into the virtual CD-ROM drive and boot the machine



Log In

- Remember, ‘cvm’ is the password

Patch to the Instructions to Install the ION DevKit onto a Virtual Machine

- The ubiquity installer tries to write over the active swapfile during the install process which, of course, does not work.
  - Workaround: patch the install script to skip trying to write to the swapfile
- Using ‘sudo -E bash’ before running the ubiquity installer does not cause the installer’s root permissions to ‘stick’ throughout the entire install process
  - Workaround: use ‘sudo su –’ instead of ‘sudo –E bash’ to gain root before running the installer.

Patching Instructions

- The patch on the next slide is a patch against /usr/lib/ubiquity/ubiquity/install_misc.py which causes install_misc.py to skip trying to copy a swapfile over the active one. Once that patch has been applied, the following slight modification to the install instructions should work. The



- Install the patch
  - Get it onto the vm; probably the easiest way would be to start the vm from the iso image and pull the patch from the web location above)
  - cd to the /usr/lib/ubiquity/ubiquity directory
  - Apply the patch. If the patch is in /downloads/install_misc.patch, then the command (once in the /usr/lib/ubiquity/ubiquity directory) to install the patch would be:

```
patch -p1 < /downaloads/install_misc.patch
```

The Patch File to Install The DevKit Onto a Local Machine

```
--- a/install_misc.py       2019-05-16 19:04:02.000474662 -0400
+++ b/install_misc.py_new   2019-05-16 19:03:52.219586663 -0400
@@ -751,6 +751,9 @@


 def copy_file(db, sourcepath, targetpath, md5_check):
+    if targetpath == '/target/swapfile':
+        return
+
     while 1:
         if md5_check:
             sourcehash = hashlib.md5()
```

Patch the file: /usr/lib/ubiquity/ubiquity/install_misc.py

```
--- a/install_misc.py       2019-05-16 19:04:02.000474662 -0400
+++ b/install_misc.py_new   2019-05-16 19:03:52.219586663 -0400
@@ -751,6 +751,9 @@


 def copy_file(db, sourcepath, targetpath, md5_check):
+    if targetpath == '/target/swapfile':
+        return
+
     while 1:
         if md5_check:
             sourcehash = hashlib.md5()
```

Install the DevKit onto the (Blank) Virtual Hard Disk

- Click the Terminal icon on the left to get a shell
- Give the root user a password
  - Type ‘sudo passwd root’ (no quotes) and assign a root password
- Type ‘su -’ (no quotes) to get a root shell
  - Sudo here is NOT enough. If anyone can explain WHY I’d be interested.
- Type ‘ubiquity gtk_ui’
  - Select ‘English’ (or your preferred language) from the language dialog
    - Press ‘Continue’
  - Don’t make any changes to the ‘Preparing to install Ubuntu’ screen
    - Press ‘Continue’

Install the DevKit onto the (Blank) Virtual Hard Disk

- For ‘Installation Type’ select ‘Erase disk and install Ubuntu’
  - Yeah, this is scary, but it’s going to erase the virtual hard drive file you made when provisioning the VM, NOT the host hard drive
  - Click ‘Install Now’
- When it ask if you want to write the changes to disk, hit ‘Continue’
  - Let it know what time zone you’re in
- Select the Keyboard Layout you want
    - Note: the whole dialog may not fit on the screen – just hit ‘Return’
- Now might be a good time for coffee…
- When it’s done, restart
  - Hit Return to remove the installation media (the DevKit .iso) from the CD-ROM drive

Now You Have a VM With a Hard Disk

- Again, the password is ‘cvm’
- But THIS VM has its own hard disk so you can:
  - Make changes to the DevKit Scenarios (or make your own) and save them
  - Install new software
- And all the changes will persist across reboots



Thank you!

Installing the ION DevKit onto a Local VM



HANDS-ON

Patch to the Instructions to Install the ION DevKit onto a Virtual Machine

- There are two issues with the installation instructions:
  - The ubiquity installer tries to write over the active swapfile during the install process which, of course, does not work.
    - Workaround: patch the install script to skip trying to write to the swapfile
  - Using ‘sudo -E bash’ before running the ubiquity installer does not cause the installer’s root permissions to ‘stick’ throughout the entire install process
    - Workaround: use ‘sudo su –’ instead of ‘sudo –E bash’ to gain root before running the installer.

Patching Instructions

- The patch on the next slide is a patch against /usr/lib/ubiquity/ubiquity/install_misc.py which causes install_misc.py to skip trying to copy a swapfile over the active one. Once that patch has been applied, the foll[redacted]k. The patch is also available from [redacted]
- Install the patch
  - Get it onto the vm; probably the easiest way would be to start the vm from the iso image and pull the patch from the web location above)
  - cd to the /usr/lib/ubiquity/ubiquity directory
  - Apply the patch. If the patch is in /downloads/install_misc.patch, then the command (once in the /usr/lib/ubiquity/ubiquity directory) to install the patch would be:

```
patch -p1 < /downaloads/install_misc.patch
```

- THEN, instead of doing 'sudo -E bash' to gain root and then running the installer with 'ubiquity gtk_ui' you need to gain root by executing 'sudo su -' (sudo su and a single dash). For some reason, the 'sudo su -' gives a more persistent 'root-y-ness' than sudo.

Patch the file: /usr/lib/ubiquity/ubiquity/install_misc.py

```
--- a/install_misc.py       2019-05-16 19:04:02.000474662 -0400
+++ b/install_misc.py_new   2019-05-16 19:03:52.219586663 -0400
@@ -751,6 +751,9 @@


 def copy_file(db, sourcepath, targetpath, md5_check):
+    if targetpath == '/target/swapfile':
+        return
+
     while 1:
         if md5_check:
             sourcehash = hashlib.md5()
```

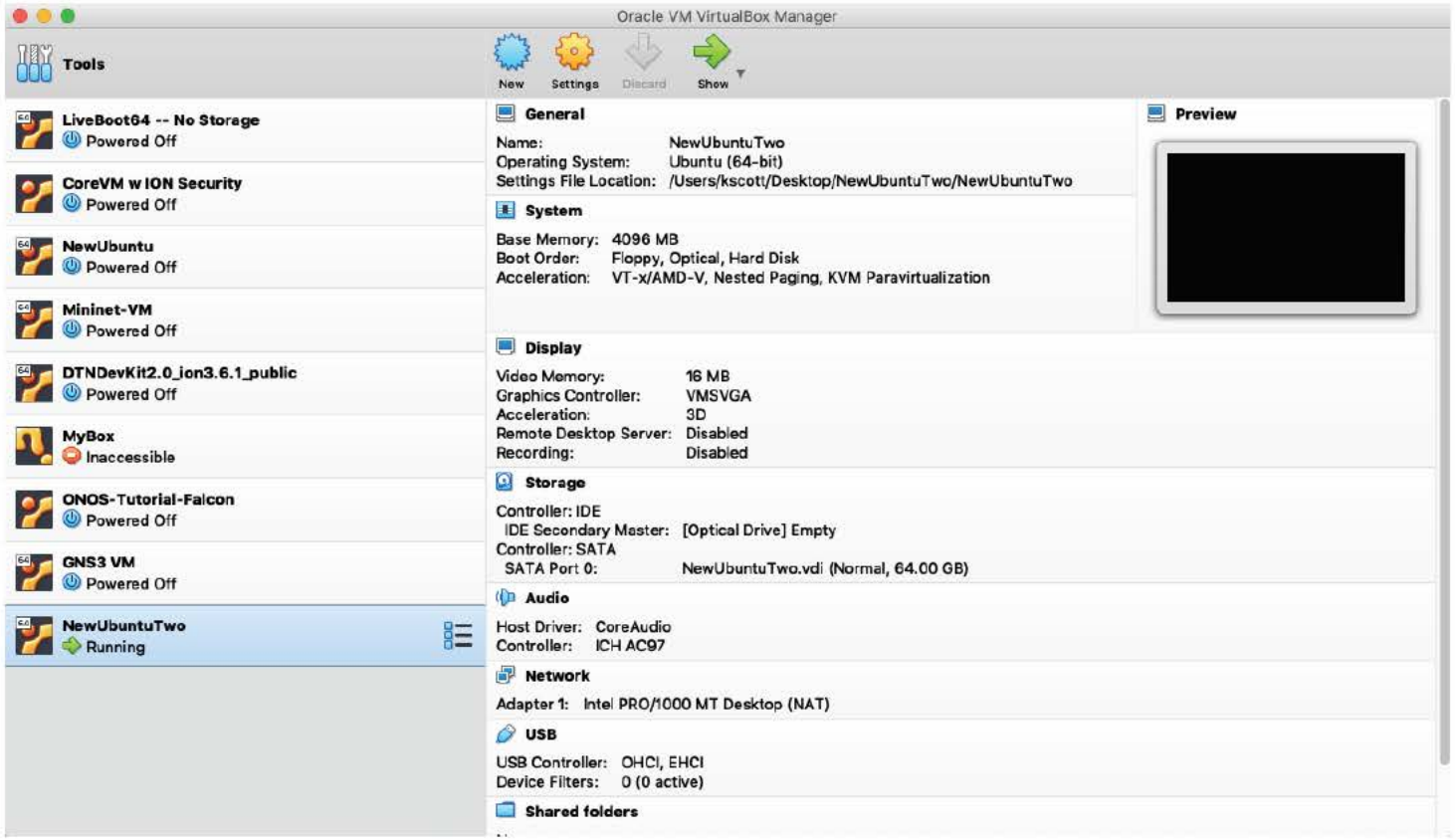


DTN DevKit – KickStart

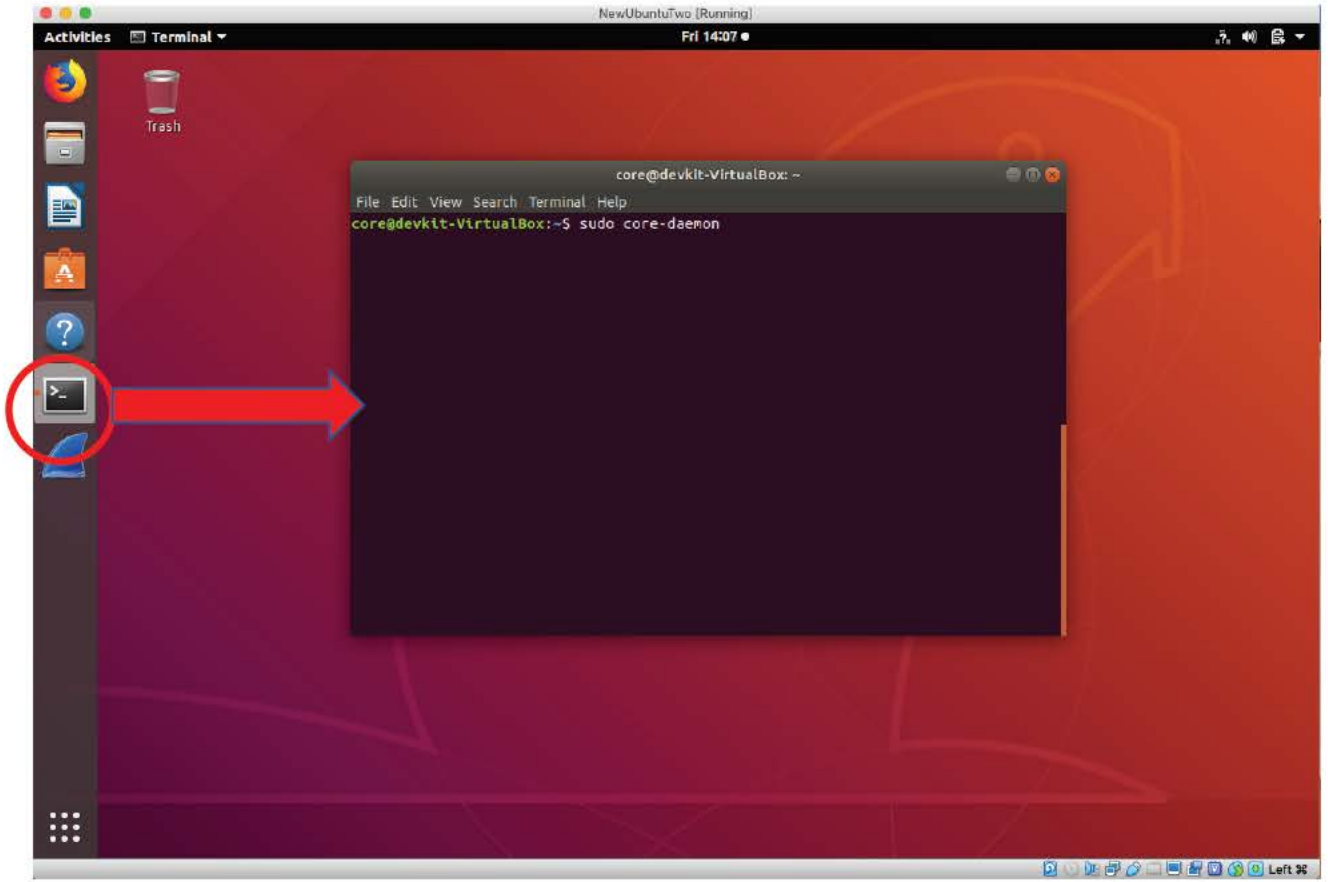
Agenda

- Start the ION DevKit VM
- Start the Common Open Research Emulator (CORE) [daemon and gui]
  - Need to start daemon by hand, not as an Ubuntu service
- Load and run the 'base' scenario
  - Look, ping works!
  - Graphic display of contact plan
  - Bundle counts
- Executing commands on emulated nodes
  - ping
  - tcpdump on the satellite to watch for incoming bundles
  - tshark / Wireshark on n2
  - bpsource / bpsink
- Start a different scenario to control connectivity by hand
  - Experiment w/ bping and/or bpsource/bpsink

Start VirtualBox and Run the DevKit Virtual Machine



Get a Terminal and Start the Core Daemon

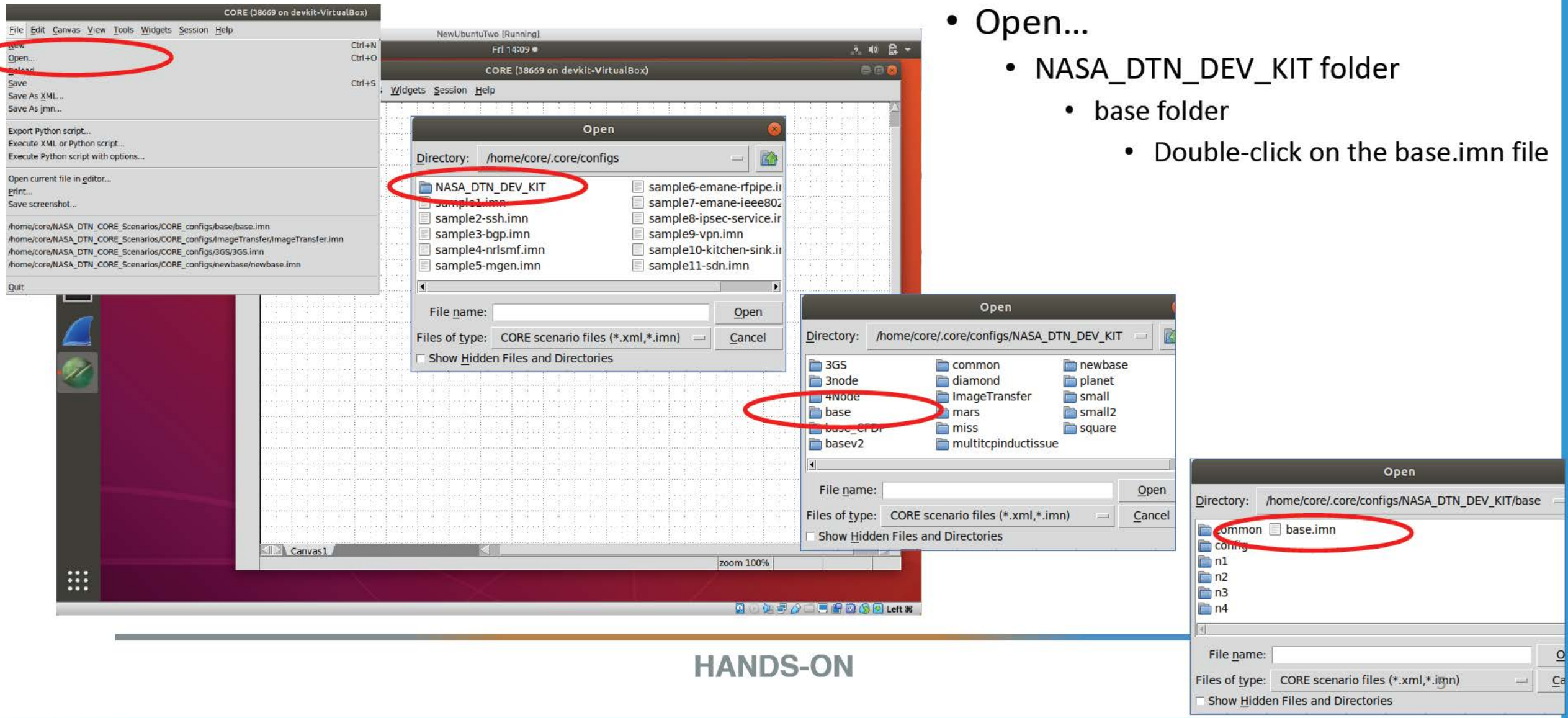


- core-gui &
  - Starts the CORE gui in the background

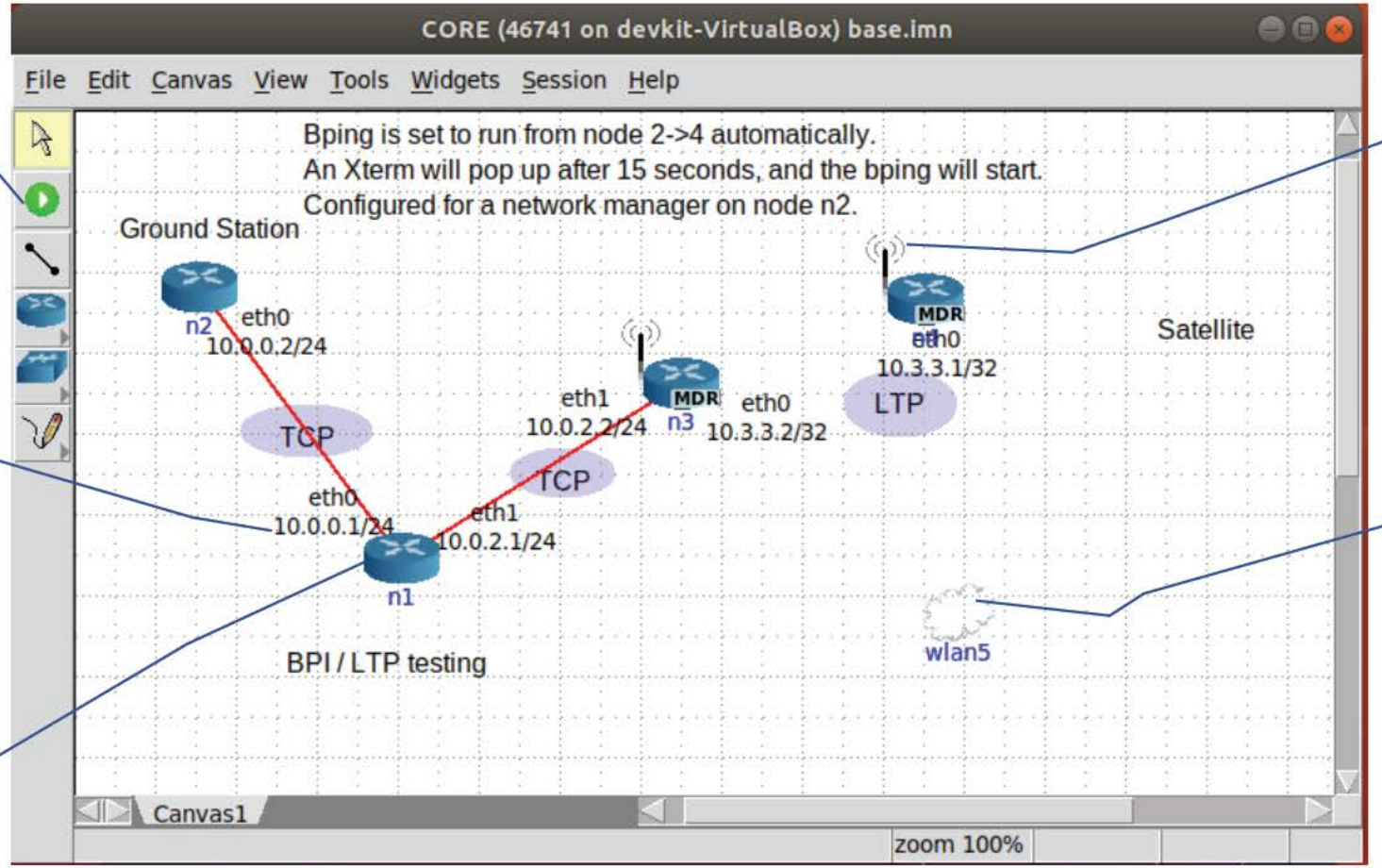
- Note: the core-daemon should already be running as a service

Load the ‘base’ Scenario

- File Menu
  - Open…
    - NASA_DTN_DEV_KIT folder
      - base folder
        - Double-click on the base.imn file



The base scenario



Start Button

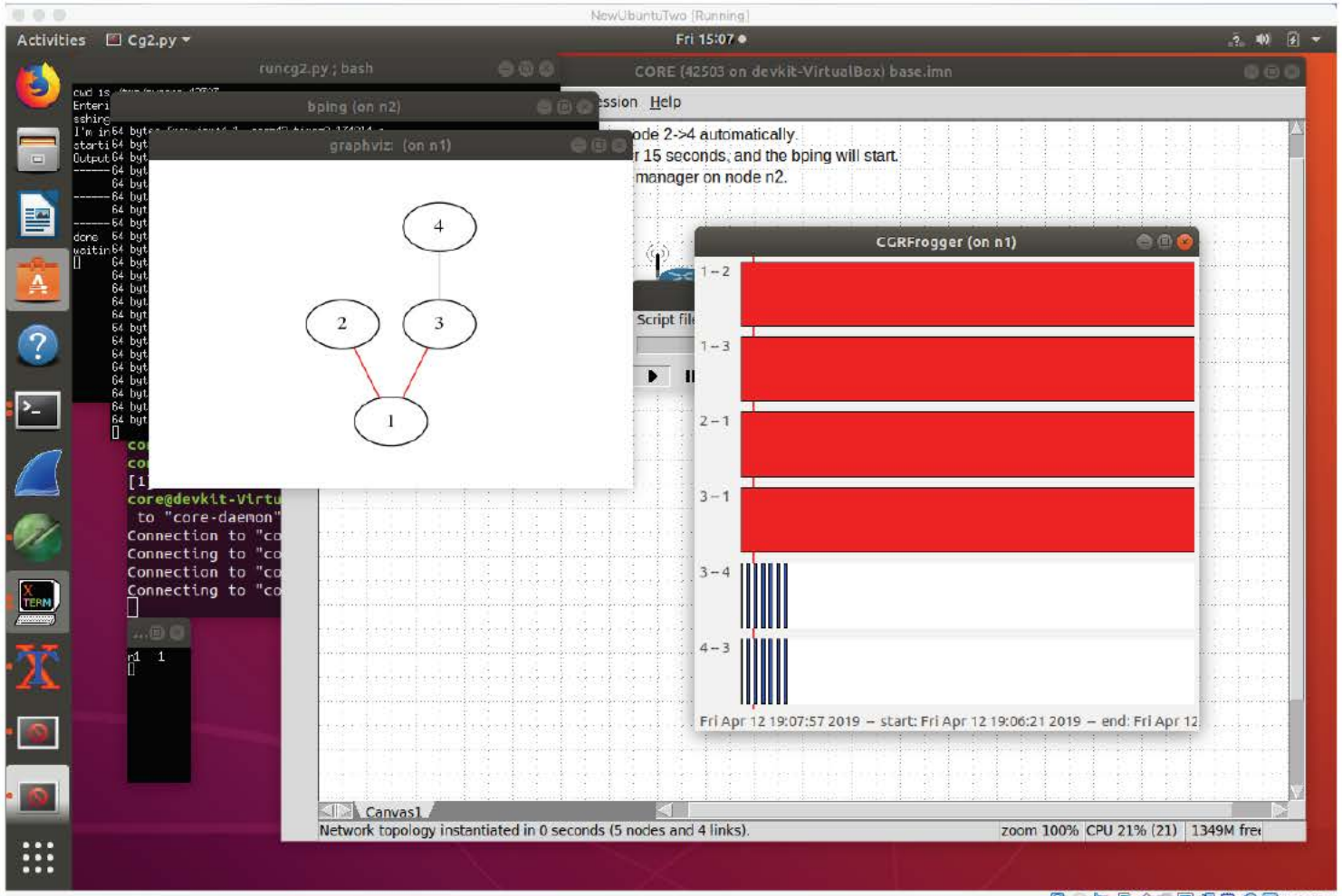
IP address of interface

Virtual node running ION

Wireless interface

Wireless LAN (configuration)

Press the Start button and wait ~ 10s…

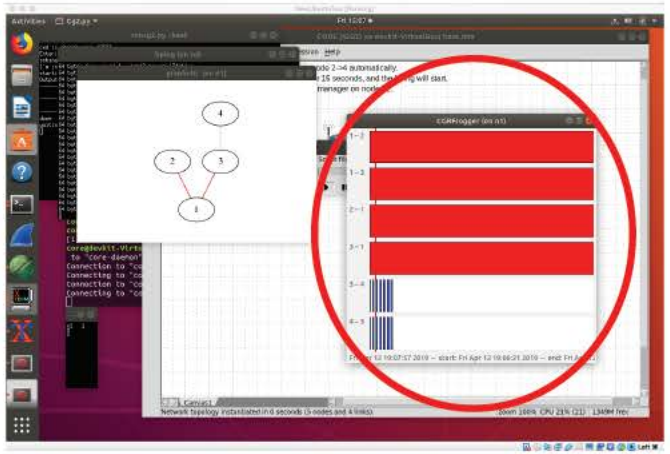


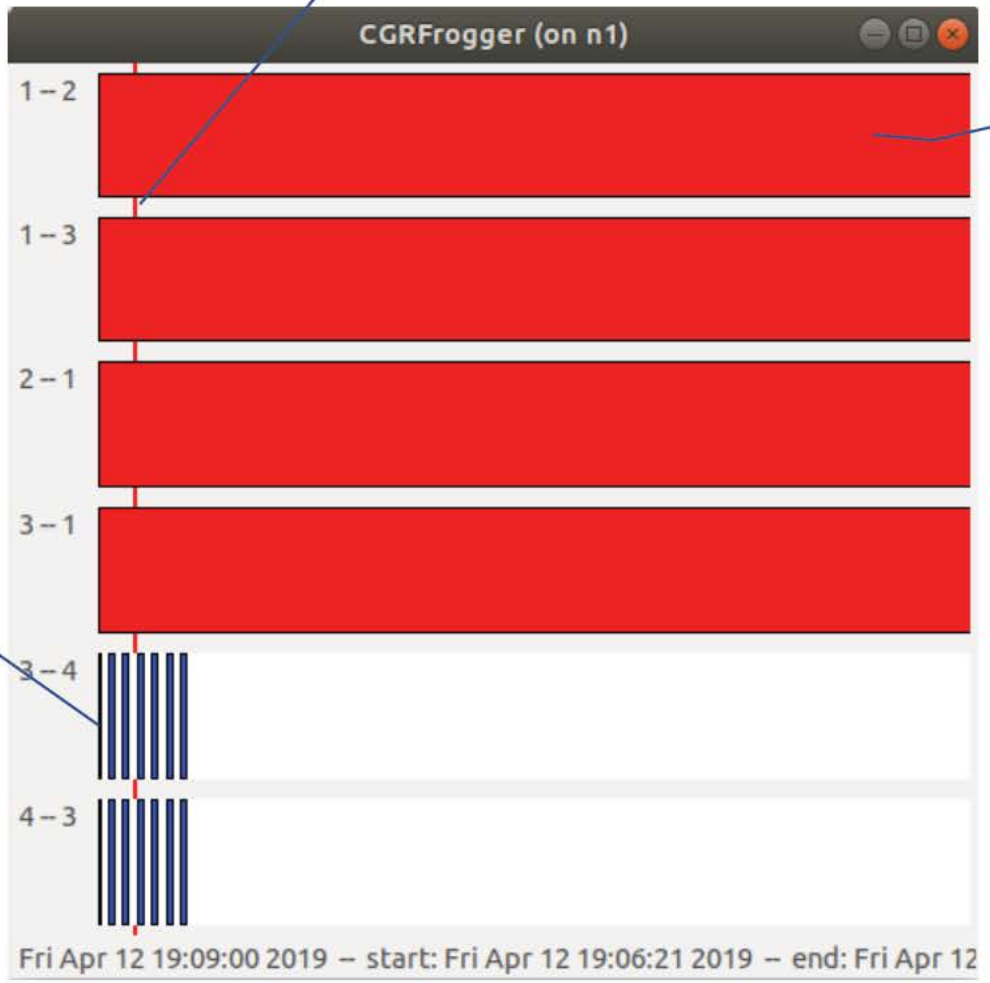
HANDS-ON

Graphic Display of Contact Plan

The vertical red line is the current time

Red means *currently* connected; here node 1 is connected to node 2 all the time

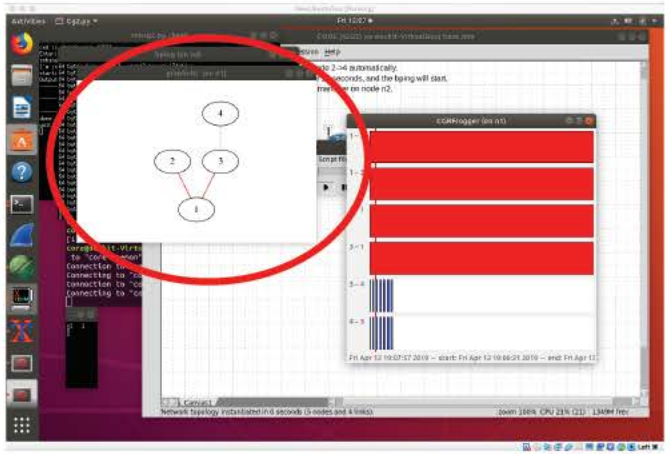


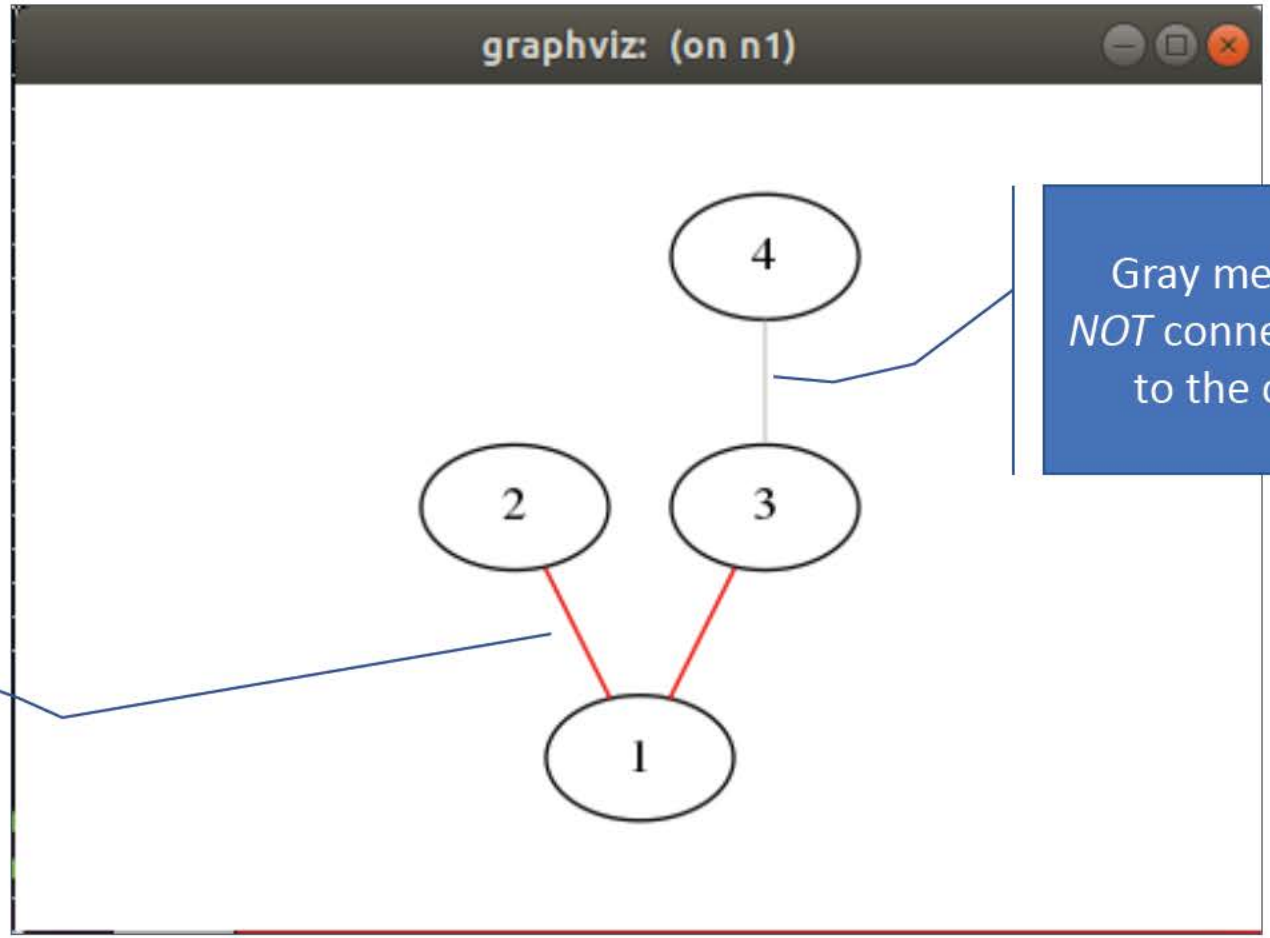


Blue means *not currently* connected

Generated from the contact plan of node 1.

Current Connectivity Graph

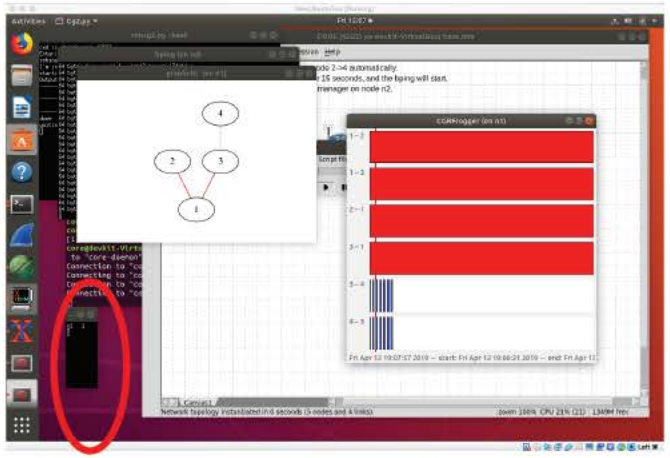


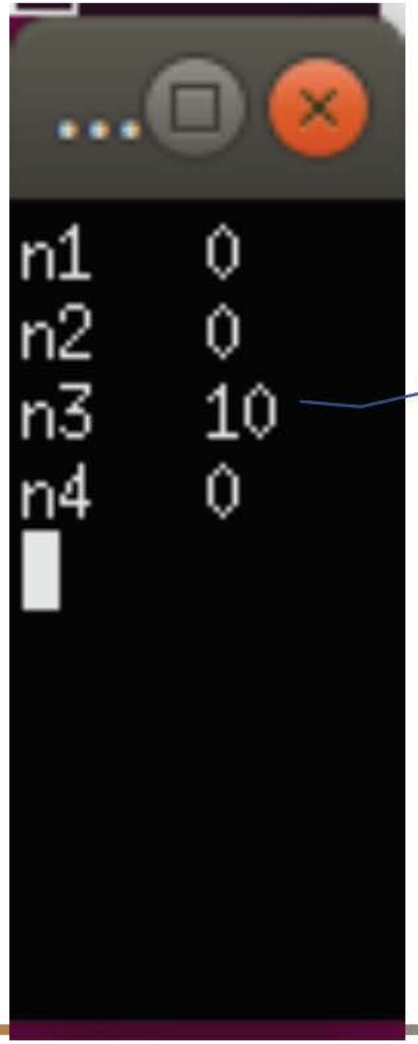


Gray means *currently NOT* connected according to the contact plan

Red means *currently* connected according to the contact plan.

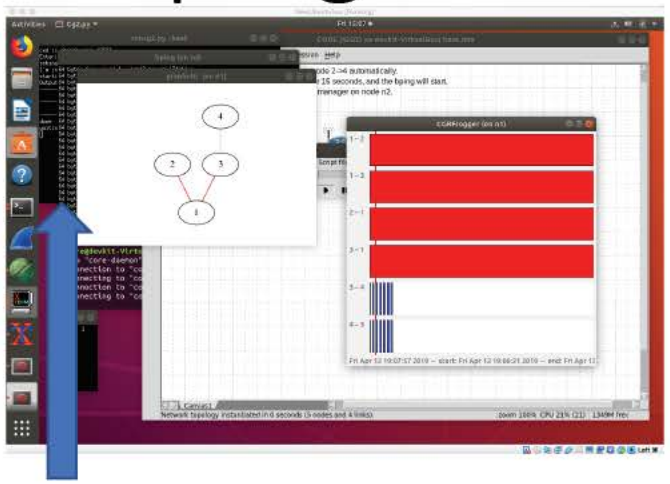
Bundle Counts





Shows the number of bundles currently resident at each node.

Bping from node2 to node4



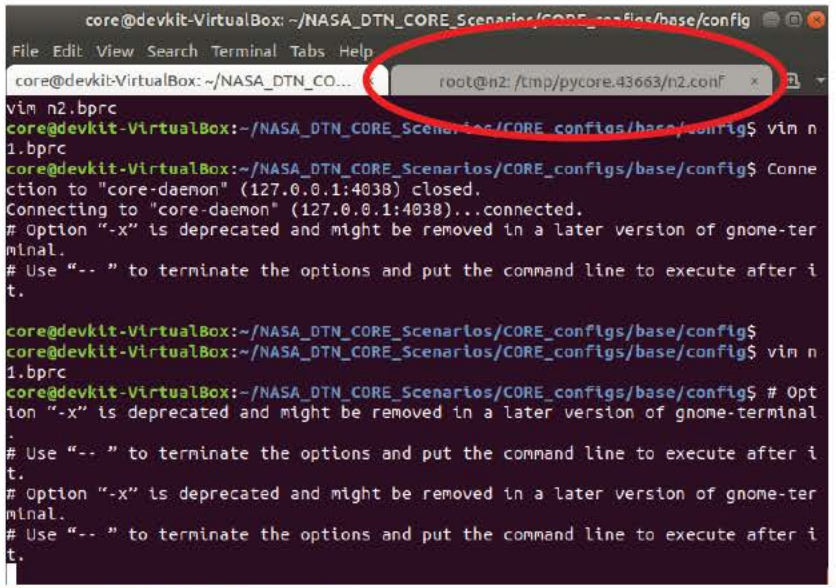
It's hiding behind the current connectivity graph

```
bping (on n2)

64 bytes from ipn:4.1  seq=0 time=0.181020 s
64 bytes from ipn:4.1  seq=1 time=0.136345 s
64 bytes from ipn:4.1  seq=2 time=0.140763 s
64 bytes from ipn:4.1  seq=3 time=0.130342 s
64 bytes from ipn:4.1  seq=4 time=0.133920 s
64 bytes from ipn:4.1  seq=5 time=0.130605 s
64 bytes from ipn:4.1  seq=6 time=0.129476 s
64 bytes from ipn:4.1  seq=7 time=0.133536 s
64 bytes from ipn:4.1  seq=8 time=0.134701 s
64 bytes from ipn:4.1  seq=9 time=0.132426 s
64 bytes from ipn:4.1  seq=10 time=0.139685 s
64 bytes from ipn:4.1  seq=37 time=0.193255 s
64 bytes from ipn:4.1  seq=38 time=0.128100 s
64 bytes from ipn:4.1  seq=39 time=0.128688 s
64 bytes from ipn:4.1  seq=15 time=24.541401 s
64 bytes from ipn:4.1  seq=16 time=23.541419 s
64 bytes from ipn:4.1  seq=17 time=22.542809 s
64 bytes from ipn:4.1  seq=18 time=21.544732 s
64 bytes from ipn:4.1  seq=19 time=20.545981 s
64 bytes from ipn:4.1  seq=20 time=19.549481 s
64 bytes from ipn:4.1  seq=21 time=18.551505 s
64 bytes from ipn:4.1  seq=22 time=17.552549 s
64 bytes from ipn:4.1  seq=23 time=16.553820 s
64 bytes from ipn:4.1  seq=24 time=15.554882 s
```

Getting a Shell on a Virtual Node

- To get a shell running on one of the virtual nodes, just double-click on it (start with n2, some examples below depend on your choosing it)
- The shell will probably show up as a tab in an existing terminal that you used to launch the gui
- Selecting that tab will show you as root in /tmp/pycore.XXXXX/nY.conf where *Y* is the node # and *XXXXX* is the process ID of the core process
- Each virtual node has its own interfaces, *shared memory*, etc.



Try Running Some Commands on the Virtual Node, e.g.

- ls (show files)
- pwd (print the current working directory)
- ifconfig (show networking interfaces)

Wait a bit before doing the commands below (wait for OSPF to tell n2 about the n1-n3 network)

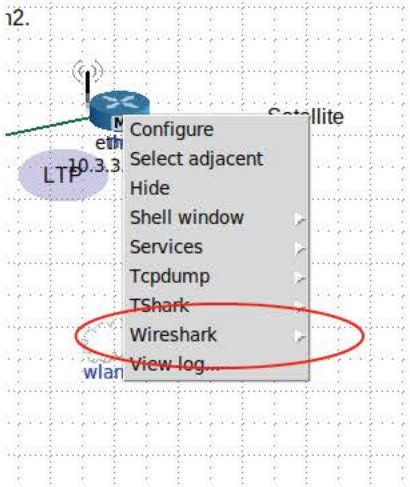
- netstat -rn (Show IP routing table)
- ping 10.0.0.1 (IP ping of n1's near interface from n2's perspective)
- Ping 10.0.2.1 (IP ping of n1's far interface from n2's perspective)
- Ping 10.0.2.2 (n3's wired interface)

tcpdump on Satellite to Watch for Bundles

- Double-click on the satellite icon (n4)
- Find the shell (again, probably shows up as a tab in an existing terminal window)
- Execute the command:
  - tcpdump -nn -l –i eth0 port 1113
    - (tcpdump –nn -<the letter L, lowercase> -i eth0 port 1113
  - You should see UDP packets showing up (tcpdump doesn't understand LTP or BP)

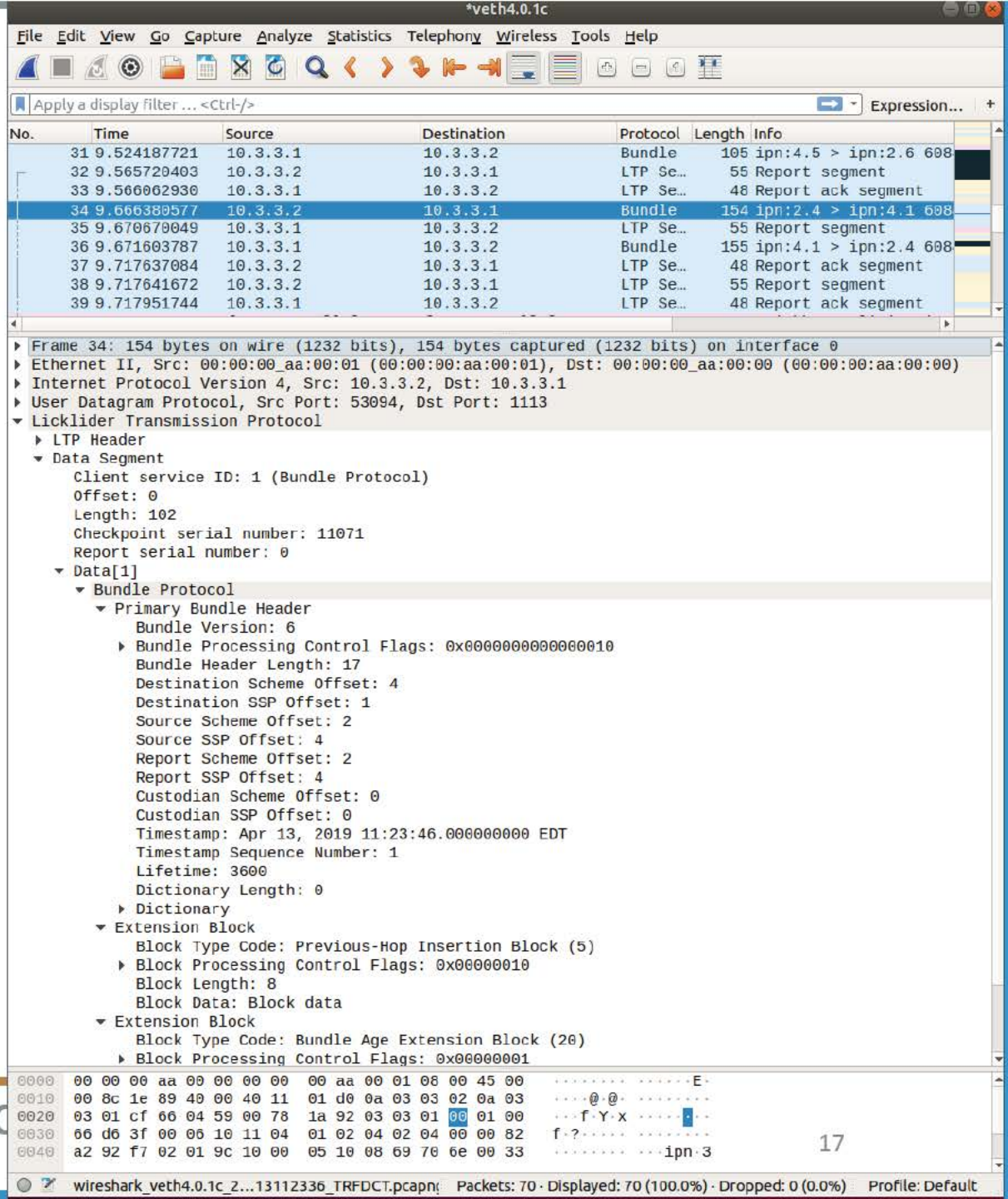
Wireshark on Satellite to Watch for Bundles

- Right-Click on the satellite node (n4) and select ‘Wireshark’ from the context menu (and ‘eth0’ from the submenu that shows up)



Wireshark Display

- Wireshark will decode LTP and BP



Start some ION Client Applications By Hand

- Stop the automated motion by clicking the ‘stop’ icon
- Double-click to get a shell on n2
- Double-click to get a shell on n3
- You may want to find the tabs for n2 and n3 shells and right-click on them and select ‘Detach Terminal’ so they get their own windows

Start some ION Client Applications By Hand

- From the n3 shell window, start the bpsink application on n3

```
bpsink ipn:3.3
```

- From the n2 shell window, send a bundle to the bpsink instance on n3

```
bpsource ipn:2.3 "test"
```

- In the n3 tab/window you should see the following:

```
ION event: Payload delivered.
   payload length is 4.
   'test'
```

Manual Connectivity

Manual Control over Connectivity

- Close the 'base' scenario and open and start the 'Exercise2a_constant' scenario
- Double-click on each of the nodes to get shells (again, you can detach these if you want to see both at the same time)
- From the Node1 shell, type:

```
bping ipn:1.3 ipn:2.1
```

- You should see something like:

```
root@n1:/tmp/pycore.41692/n1.conf# bping ipn:1.3 ipn:2.1
64 bytes from ipn:2.1  seq=0 time=1.219852 s
64 bytes from ipn:2.1  seq=1 time=1.229681 s
64 bytes from ipn:2.1  seq=2 time=1.219740 s
…
```

Let's try Bundle Ping (bping)

- (The scenario automatically starts the bping responder application, bpecho, on service ID #1 on all of the nodes)
- From the Node1 shell, type:

```
bping ipn:1.3 ipn:2.1
```

bping takes a source EID (on which it will listen for responses)

and a destination EID (the EID to ping).

- You should see something like:

```
root@n1:/tmp/pycore.41692/n1.conf# bping ipn:1.3 ipn:2.1
64 bytes from ipn:2.1  seq=0 time=1.219852 s
64 bytes from ipn:2.1  seq=1 time=1.229681 s
64 bytes from ipn:2.1  seq=2 time=1.219740 s
…
```

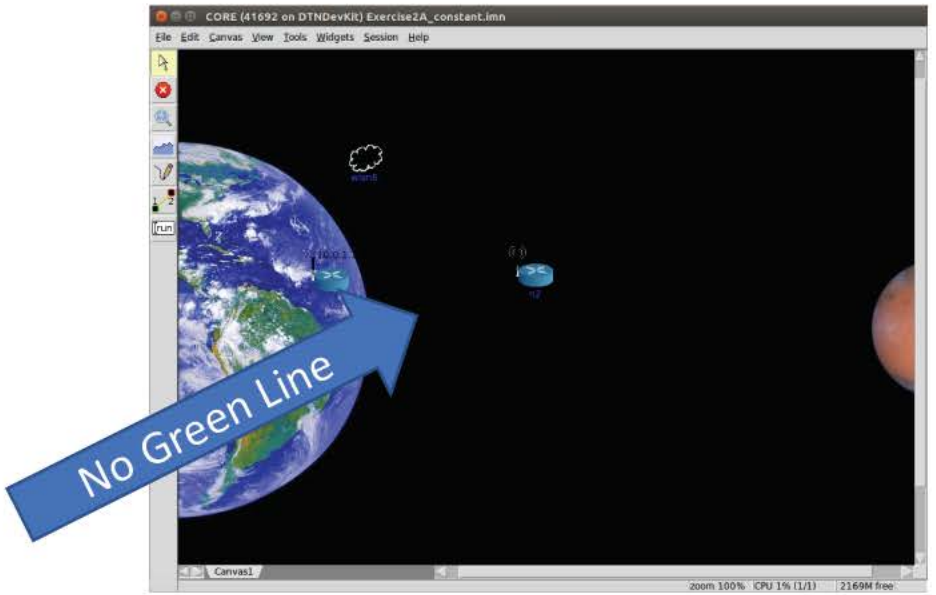
Now Let's Disconnect the Nodes

- Start bping from Node1 to Node2 (if you didn't just leave it running) with:

```
bping ipn:1.3 ipn:2.1
```

- You should see responses coming back

- Now click and drag Node2 to the right until the green line connecting it to Node1 disappears



What Just Happened

- These scenarios in CORE uses a very simple distance-based connectivity model.
  - If the distance between nodes is greater than a threshold, the nodes become disconnected
  - CORE can leverage EMANE (the Enhanced Mobile Ad-Hoc Network Emulator) for more realistic channel models – beyond the scope here
- More to the point, the ping responses stopped

Reconnect the Nodes

- Move Node2 back in range so that it can communicate with Node1 (the green line will come back)
- Wait a few moments…
- And you should see all the queued-up pings come in

What Just Happened

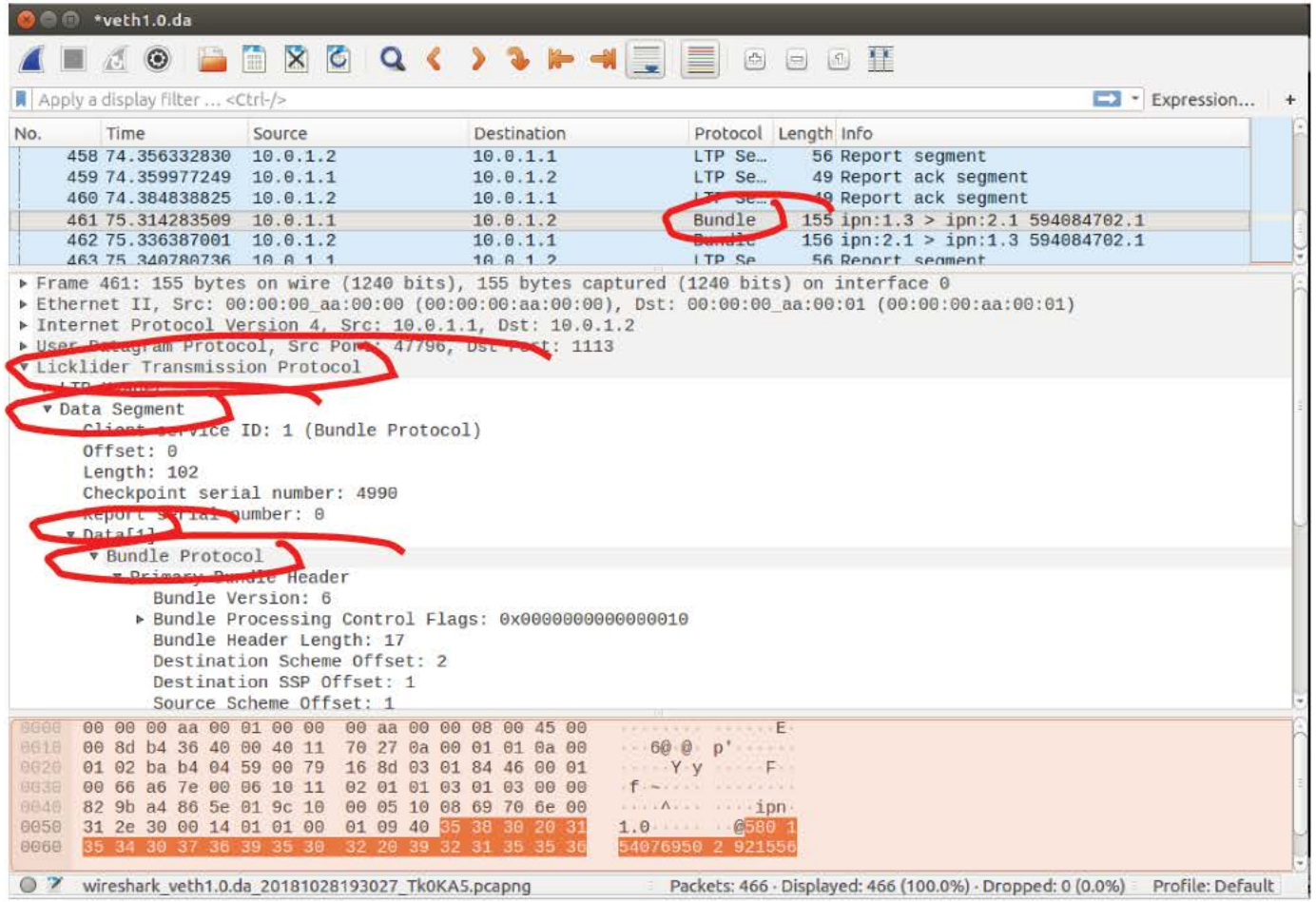
- This is a very simple case of disruption tolerance
- The ION instances on Node1 and Node2 are both configured to believe that there is constant connectivity between the nodes
- When connectivity is lost, ION (LTP) just keeps trying to retransmit until connectivity is restored

- We'll see a more complex example where the ION instances are configured with a communication schedule in Exercise2A

Backups

Look at a Bundle

- Find a packet with ‘Bundle’ in the ‘Protocol’ column and click on it.
- Drill down through Licklider Transmission Protocol; Data Segment; Data[1] to see the Bundle
- We’re not going to go into the details here, but that’s the protocol dissection of a ping bundle



DTN DevKit – Base Scenario Configuration



ION Configuration Files Overview

File Extension	Contents
.ionrc	• Specifies the ipn node # this node will use • Identifies the ionconfig file used to configure ion parameters (e.g. memory) • Contacts (connectivity among nodes)
.ionconfig	• Specifies the amount of memory ION will allocate at startup (working memory and heap)
.bprc	• What forwarding scheme(s) will be used • Which convergence layer protocols the node uses • Which endpoints the node is a member of • What inducts and outducts the node has
.ipnrc	• Specifies mechanisms to reach immediate neighbors
.ltprc	• Configure LTP parameters such as aggregation size/time • Identify the LSO LTP will use to transmit segments
.ionsecrc	• Used to configure security (not used here)
.acsrc	• Used to configure aggregate custody signaling
.cfdprc	• Configures CCSDS File Delivery Protocol engine parameters

From a Terminal Window on the Host
(We'll get back to the emulator for a short exercise at the end)

- Get into the directory with the config files for the 'base' scenario
  - cd ~/.core/configs/NASA_DTN_DEV_KIT/base/config
- Open the n1.ionrc file
  - With vi: vi n1.ionrc
  - With gedit: gedit n1.ionrc

n1.ionrc

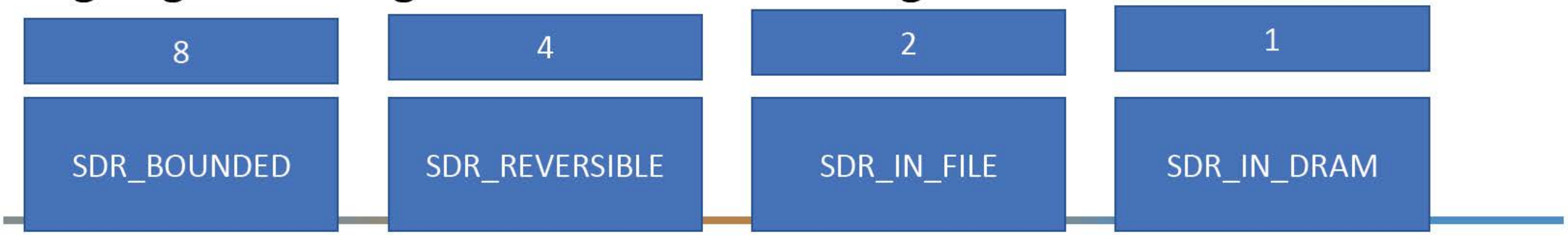
# Comments # # INITIALIZE # Ion node number: 1 # Ion configuration file name: n1.ionconfig 1 1 n1.ionconfig # # START # Program: rfxclock s	INITIALIZE NODE_NUMBER IONCONFIG_FILE



n1.ionconfig

wmKey	0	
sdrName	ion	
wmSize	5000000	Size of the working memory (memory allocated by ION at startup)
configFlags	1	
heapWords	5000000	# of words (32- or b4-bit) to be used for nominally non-volatile storage
pathName	/var/ion	

- configFlags is the logical OR of the following:



n1.bprc (1/x)

`# Initialization command (command 1).`
`1`
 	
`# Add an EID scheme.`
`a scheme ipn 'ipnfw' 'ipnadminep'`
 	
`# Add endpoints.`
`a endpoint ipn:1.1 x`
`a endpoint ipn:1.2 x`
`o`
`o`
`o`
	Endpoint commands specify the endpoints this ION node will be listening on.
	
	The 'x' means to discard bundles if no application is a member of that endpoint; 'q' means to queue bundles. More on that later in these slides.

As an example, if you want to issue the bping command to ping n2 from n1, the format is `bping <n1_EID> <n2_EID>`
For this to work, you will need to have specified (before doing the bping) that n1 is a member of endpoint <n1_EID> and n2 is a member of <n2_EID>

n1.bprc (2/x)

```
#------------------------------------------------------------------
# Add a protocol for external nodes.
#------------------------------------------------------------------
# Estimate transmission capacity assuming 1400 bytes of each frame
# for payload, and 100 bytes for overhead.
a protocol tcp 1400 100
a protocol udp 1400 100
a protocol ltp 1400 100
```

n1.bprc (3/x)

`#----------------------------------------------`
`# Add a inducts. (listen)`
`#----------------------------------------------`
`a induct tcp 0.0.0.0:4556 tcpcli`
`a induct udp 0.0.0.0:4556 udpcli`
`a induct ltp 1 ltpcli`
 	
`#----------------------------------------------`
`# Add outducts.`
`#----------------------------------------------`
`a outduct udp 127.0.0.1 udpclo`
`a outduct udp 10.0.0.2:4556 udpclo`
`a outduct tcp 10.0.2.2:4556 “”`
	tcpclo is deprecated but you still need a ‘program’ to run, this says “” (nothing)

n1.bprc (4/x)

`#----------------------------------------`
`# Select level of BP watch activities - 0 = None; 1 = All`
`w 0`
 	
`#  RUN`
`#  Program:  ipnadmin`
`#  Configuration file name: n1.ipnrc`
`r 'ipnadmin n1.ipnrc'`
 	
`# Start all declared schemes and protocols on the local node`
`s`
	Set the watch (diagnostic) characters to be printed
	
	Run the ipnadmin program with n1.ipnrc

n1.ipnrc

`#----------------------------------`
`# Add an egress plan. (to neighboring nodes/hosts)`
`#----------------------------------`
`#`
 	
`a plan 1 udp/127.0.0.1`
`a plan 2 udp/10.0.0.2:4556`
`a plan 3 tcp/10.0.2.2:4556`
	n1 can send to itself over loopback using udp
	n1 can send to n2 using UDP to 10.0.0.2
	n1 can send to n3 using TCP at 10.0.2.2

n3.ltprc

```
#Initialization command (command 1).
1 100

#------------------------------------------------
# LTP Spans
a span 4 100 100 64000 100 1 'udplso 10.3.3.1:1113
40000000'

#------------------------------------------------
# Listener on 0.0.0.0
s 'udplsi 0.0.0.0:1113'

w 0
```

No watch characters

n1.ionsecrc

`# Initialization command (command 1).`
`1`
`# Select level of "echo control" activities`
`# 0 = None; 1 = print to both log and stdout`
`e 1`
	All we do is start ionsec and tell it to log activities.
	
	You need this, otherwise ion will complain to the ionlog file which makes troubleshooting more difficult.

n1.acsrc

```
# Aggregate Custody Signal configuration
# -- DZ 11/28/2014

# Initialization command (command 1).
1 7 262144
```

GENERAL COMMANDS

1 <logLevel> [<heapWords>]

The initialize command. Until this command is executed, Aggregate Custody Signals are not in operation on the local ION node and most acsadmin commands will fail.

The logLevel argument specifies at which log level the ACS appending and transmitting implementation should record its activity to the ION log file. This argument is the bitwise "OR" of the following log levels:

0x01 ERROR
Errors in ACS programming are logged.

0x02 WARN
Warnings like "out of memory" that don't cause ACS to fail but may change behavior are logged.

0x04 INFO
Informative information like "this custody signal is a duplicate" is logged.

0x08 DEBUG
Verbose information like the state of the pending ACS tree is logged.

The optional heapWords argument informs ACS to allocate that many heap words in its own DRAM SDR for constructing pending ACS. If not supplied, the default ACS_SDR_DEFAULT_HEAPWORDS is used. Once all ACS SDR is allocated, any incoming custodial bundles that would trigger an ACS will trigger a normal, non-aggregate custody signal instead, until ACS SDR is freed. If your nodr intermittently emits non-aggregate custody signals when it should emit ACS, you should increase heapWords.

Since ACS uses SDR only for emitting Aggregate Custody Signals, ION can still receive ACS even if this command is not executed, or all ACS SDR memory is allocated.

Order of Execution for the DevKit Scenarios

ionrc (at least the initialization command; note the for the scenarios the contacts are genarlly split off to a separate file)

ionsecrc

ltprc

bprc

ipnrc (run from bprc file)

cfdprc



Exercise: In the Base Scenario, Change the UDP Link between n1 and n2 to Use TCP in the n1->n2 Direction

1. Edit n2's bprc file to have a TCP induct (for now, listen on INADDR_ANY (0.0.0.0))
2. Edit n1's bprc file to set the outduct to n2 to use tcp
3. Edit n1's .ipnrc to set its PLAN to communicate with n2 to use the tcpcl
4. Start / Restart the scenario and tcpdump n2's eth0 interface to confirm it's using TCP and not UDP

Note: You can run the various ION administrative programs interactively – so you could make the above changes to a running ION node. There's an exercise coming up to modify a running ION node.

Testing and Verifying

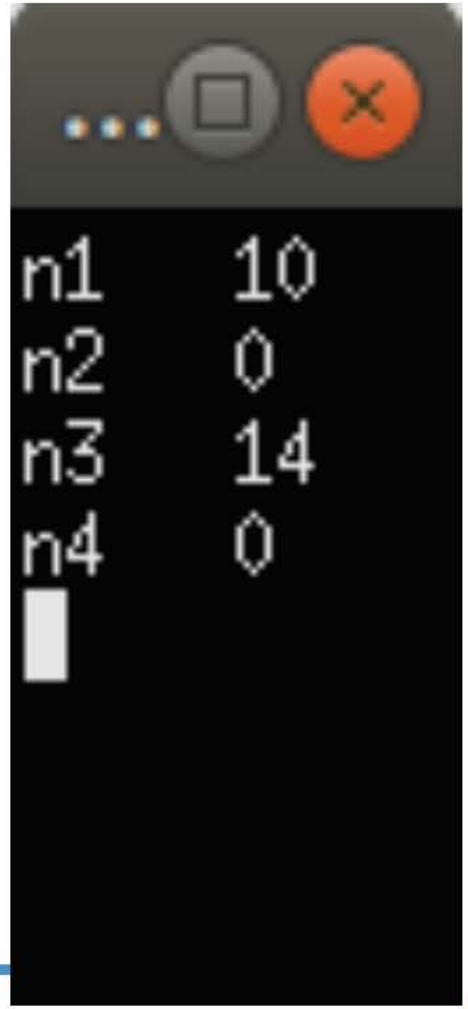
- Start bping from n1 to n2
  - Get a shell on n1
  - `bping ipn:1.3 ipn:2.1`
  - Start wireshark on n1's eth0 interface (may want to set the display filter to 'udp.port==4556 or tcp.port==4556'
  - Should see UDP and TCP on port 4556 (bundles flowing back and forth between n1 and n2)

- And yet we only see bundles from ipn:2.4 > ipn:4.1, so where's the bug…?

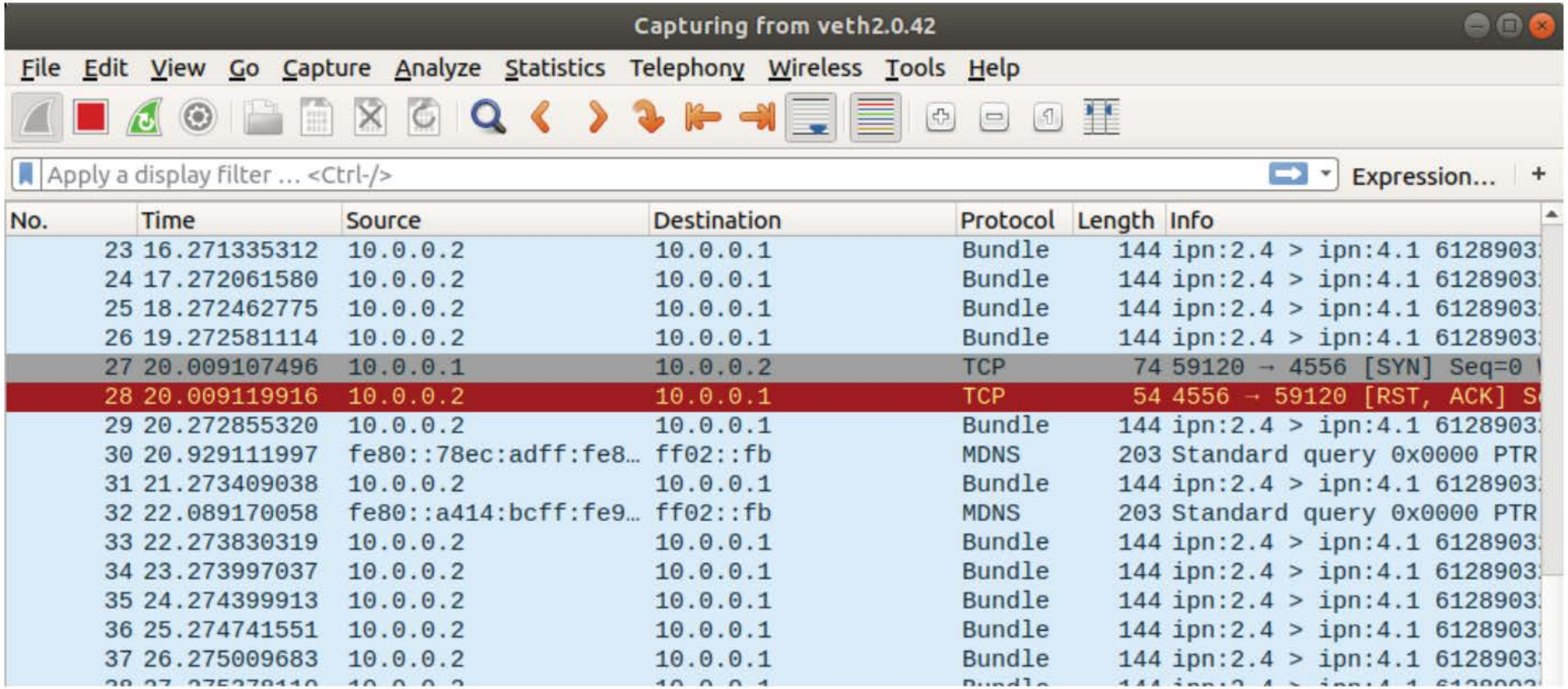
Into to Troubleshooting

- The instructions above deliberately left out a step so that we can debug the issue
- General framework:
  - Do we have IP connectivity on the link(s) we're trying to use?
  - Does the sending ION Node think it is transmitting?
    - Are bundles actually being emitted?
  - Does the receiving ION Node think it is receiving?

```
root@n1:/tmp/pycore.37331/n1.conf# cat ion.log
[2019/06/03-11:24:38] [i] rfxclock is running.
[2019/06/03-11:24:38] [i] No congestion collapse predicted.
[2019/06/03-11:24:38] [i] ionwarn finished.
[2019/06/03-11:24:39] [i] No congestion collapse predicted.
[2019/06/03-11:24:39] [i] ionwarn finished.
[2019/06/03-11:24:39] Stopping ionsecadmin.
[2019/06/03-11:24:39] [i] Total max export sessions does not exceed estimate.
[2019/06/03-11:24:39] [i] ltpclock is running.
[2019/06/03-11:24:39] [i] ltpdeliv is running.
[2019/06/03-11:24:39] [i] udplsi is running, spec=[0.0.0.0:1113].
[2019/06/03-11:24:40] [i] Bundle security is enabled.
[2019/06/03-11:24:40] [i] bpclm is running: ipn:3.0
[2019/06/03-11:24:40] [i] bpclm is running: ipn:2.0
[2019/06/03-11:24:40] [i] bpclm is running: ipn:1.0
[2019/06/03-11:24:41] [i] ltpcli is running.
[2019/06/03-11:24:41] [i] udpclo is running.
[2019/06/03-11:24:41] [i] udpcli is running, spec=[0.0.0.0:4556].
[2019/06/03-11:24:41] [i] ipnadminep is running.
[2019/06/03-11:24:41] [i] bpclock is running.
[2019/06/03-11:24:41] [i] tcpcli is running [0.0.0.0:4556].
[2019/06/03-11:24:41] [i] bptransit is running.
[2019/06/03-11:24:41] [i] ipnfw is running.
[2019/06/03-11:24:41] at line 3430 of ici/library/platform.c, Can't connect to TCP socket: Connection refused (10.0.0.2:4556)
[2019/06/03-11:24:41] at line 3430 of ici/library/platform.c, Can't connect to TCP socket: Connection refused (10.0.2.2:4556)
[2019/06/03-11:24:42] [x] src from 1969/12/31-19:00:00 to 2019/06/03-11:24:42: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:24:42] [x] fwd from 1969/12/31-19:00:00 to 2019/06/03-11:24:42: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:24:42] [x] xmt from 1969/12/31-19:00:00 to 2019/06/03-11:24:42: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:24:42] [x] rcv from 1969/12/31-19:00:00 to 2019/06/03-11:24:42: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:24:42] [x] dlv from 1969/12/31-19:00:00 to 2019/06/03-11:24:42: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:24:42] [x] ctr from 1969/12/31-19:00:00 to 2019/06/03-11:24:42: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:24:42] [x] rfw from 1969/12/31-19:00:00 to 2019/06/03-11:24:42: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:24:42] [x] exp from 1969/12/31-19:00:00 to 2019/06/03-11:24:42: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:24:42] at line 3430 of ici/library/platform.c, Can't connect to TCP socket: Connection refused (10.0.0.2:4556)
[2019/06/03-11:24:42] [i] Connected to TCP socket: 10.0.2.2:4556
[2019/06/03-11:24:42] [i] tcpcli admin thread has started: ipn:3.0
[2019/06/03-11:24:42] [i] tcpcli sender thread has started: ipn:3.0
[2019/06/03-11:24:43] [?] Can't close llcv, already closed.
[2019/06/03-11:24:43] [?] Can't signal llcv, already closed.
[2019/06/03-11:24:43] [?] Can't close llcv, already closed.
[2019/06/03-11:24:44] [i] tcpcli admin thread has started: ipn:3.0
```



```
root@n1:/tmp/pycore.37331/n1.conf# bpstats
root@n1:/tmp/pycore.37331/n1.conf# tail ion.log
[2019/06/03-11:07:40] [x] fwd from 1969/12/31-19:00:00 to 2019/06/03-11:07:40: (0) 0 0 (1) 0 0 (2) 0 0 (+) 138 8676
[2019/06/03-11:07:40] [x] xmt from 1969/12/31-19:00:00 to 2019/06/03-11:07:40: (0) 71 4544 (1) 0 0 (2) 0 0 (+) 71 4544
[2019/06/03-11:07:40] [x] rcv from 1969/12/31-19:00:00 to 2019/06/03-11:07:40: (0) 71 4544 (1) 66 4120 (2) 0 0 (+) 137 8664
[2019/06/03-11:07:40] [x] dlv from 1969/12/31-19:00:00 to 2019/06/03-11:07:40: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:07:40] [x] ctr from 1969/12/31-19:00:00 to 2019/06/03-11:07:40: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:07:40] [x] rfw from 1969/12/31-19:00:00 to 2019/06/03-11:07:40: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:07:40] [x] exp from 1969/12/31-19:00:00 to 2019/06/03-11:07:40: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/06/03-11:07:40] [i] ...end of statistics snapshot.
```


Capturing from veth2.0.42
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expression…
No. Time Source Destination Protocol Length Info
23 16.271335312 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
24 17.272061580 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
25 18.272462775 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
26 19.272581114 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
27 20.009107496 10.0.0.1 10.0.0.2 TCP 74 59120 → 4556 [SYN] Seq=0
28 20.009119916 10.0.0.2 10.0.0.1 TCP 54 4556 → 59120 [RST, ACK]
29 20.272855320 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
30 20.929111997 fe80::78ec:adff:fe8… ff02::fb MDNS 203 Standard query 0x0000 PTR
31 21.273409038 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
32 22.089170058 fe80::a414:bcff:fe9… ff02::fb MDNS 203 Standard query 0x0000 PTR
33 22.273830319 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
34 23.273997037 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
35 24.274399913 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
36 25.274741551 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903
37 26.275009683 10.0.0.2 10.0.0.1 Bundle 144 ipn:2.4 > ipn:4.1 6128903

N2 Doesn't use the TCP Protocol

- Add a protocol line for TCP to n2's bprc file
  - `a protocol tcp 1400 100`
- And try again…

Running ION Admin Commands Interactively Endpoints: 'x' vs. 'q'

- The 'x' or 'q' parameter to the bpadmin 'add endpoint' command determines what happens to bundles that are received when no application is bound to the endpoint
- Let's play with adding a new endpoint to a running ION node that queues bundles for which there is not currently a bound application
- On n1 run bpadmin to add a new 'q' endpoint
  - Shell on n1
  - `bpadmin`
  - `h` # help
  - `l endpoint` # List current endpoints, PIDS, rules, scripts
  - `a endpoint ipn:1.100 q` # Add a new endpoint
  - `l endpoint` # See that it showed up
- Run bpsource to send some text from n2 to n1
  - `bpsource ipn:2.3 ipn:1.100`
- THEN run bpsink on n1
  - `bpsink ipn:1.100`
- You sent a bundle to a defined endpoint that didn't have an application associated with it at the time of receipt. When the application showed up, it received the bundle.

Man Pages

Man Page
ionrc :: initialize

1 node_number [{ ion_config_filename | '.' | '' }]

The initialize command. Until this command is executed, the local ION node does not exist and most ionadmin commands will fail.

The command configures the local node to be identified by node_number, a CBHE node number which uniquely identifies the node in the delay-tolerant network. It also configures ION's data space (SDR) and shared working-memory region. For this purpose it uses a set of default settings if no argument follows node_number or if the argument following node_number is ''; otherwise it uses the configuration settings found in a configuration file. If configuration file name '.' is provided, then the configuration file's name is implicitly "hostname.ionconfig"; otherwise, ion_config_filename is taken to be the explicit configuration file name. Please see ionconfig(5) for details of the configuration settings.

For example:

1 19 ''

would initialize ION on the local computer, assigning the local ION node the node number 19 and using default values to configure the data space and shared working-memory region.

Man Page ionconfig

sdrWmSize

This is the **size of the block of dynamic memory** that will be reserved as private working memory for the SDR system itself. A block of system memory of this size will be allocated (e.g., by malloc()) at the time the SDR system is initialized on the host computer. The default value is 1000000 (1 million bytes).

configFlags

This is the bitwise "OR" (i.e., the sum) of the flag values that characterize the SDR database to use for this ION node. The default value is 13 (that is, SDR_IN_DRAM | SDR_REVERSIBLE | SDR_BOUNDED). The SDR configuration flags are documented in detail in sdr(3). To recap:

SDR_IN_DRAM (1)
The SDR is implemented in a region of shared memory. [Possibly with write-through to a file, for fault tolerance.]

SDR_IN_FILE (2)
The SDR is implemented as a file. [Possibly cached in a region of shared memory, for faster data retrieval.]

SDR_REVERSIBLE (4)
Transactions in the SDR are written ahead to a log, making them reversible.

SDR_BOUNDED (8)
SDR heap updates are not allowed to cross object boundaries.

heapWords

This is the **number of words** (of 32 bits each on a 32-bit machine, 64 bits each on a 64-bit machine) **of nominally non-volatile storage to use for ION's SDR database**. If the SDR is to be implemented in shared memory and no heapKey is specified, a block of shared memory of this size will be allocated (e.g., by malloc()) at the time the node is created. If the SDR is to be implemented in a file and no file named ion.sdr exists in the directory identified by pathName, then a file of this name and size will be created in this directory and initialized to all binary zeroes. The default value is 250000 words (1 million bytes on a 32-bit computer).

Man Page
ipnrc



DESCRIPTION

IPN scheme configuration commands are passed to ipnadmin either in a file of text lines or interactively at ipnadmin's command prompt (:). Commands are interpreted line-by line, with exactly one command per line.

IPN scheme configuration commands (a) establish egress plans for direct transmission to neighboring nodes that are members of endpoints identified in the "ipn" URI scheme and (b) establish static default routing rules for forwarding bundles to specified destination nodes.

The egress plan established for a given node associates a duct expression with that node. Each duct expression is a string of the form "protocol_name/outduct_name" signifying that the bundle isto be queued for transmission via the indicated convergence layer protocol outduct.

Note that egress plans must be established for all neighboring nodes, regardless of whether or not contact graph routing is used for computing dynamic routes to distant nodes. This is by definition: if there isn't an egress plan to a node, it can't be considered a neighbor.

Static default routes are declared as exits in the ipn-scheme routing database. An exit is a range of node numbers identifying a set of nodes for which defined default routing behavior is established. Whenever a bundle is to be forwarded to a node whose number is in the exit's node number range and it has not been possible to compute a dynamic route to that node from the contact schedules that have been provided to the local node and that node is not a neighbor to which the bundle can be directly transmitted, BP will forward the bundle to the gateway node associated with this exit. The gateway node for any exit is identified by an endpoint ID, which might or might not be an ipn-scheme EID; regardless, directing a bundle to the gateway for an exit causes the bundle to be re-forwarded to that intermediate destination endpoint. Multiple exits may encompass the same node number, in which case the gateway associated with the most restrictive exit (the one with the smallest range) is always selected.

Note that "exits" were termed "groups" in earlier versions of ION. The term "exit" has been adopted instead, to minimize any possible confusion with multicast groups. To protect backward compatibility, the keyword "group" continues to be accepted by ipnadmin as an alias for the new keyword "exit", but the older terminology is deprecated.

The formats and effects of the IPN scheme configuration commands are described below.

GENERAL COMMANDS

a plan node_nbr duct_expression [nominal_data_rate]

The add plan command. This command establishes an egress plan for the bundles that must be transmitted to the neighboring node identified by node_nbr. The nominal_data_rate is the assumed rate of transmission to this node in the absence of contact plan information. A nominal_data_rate of zero (the default) in the absence of contact plan information completely disables rate control.

Note that the plan commands consumed by ipnadmin are a simplified shortcut for submitting plan commands as consumed by bpadmin (see bprc(5)). The syntax of these commands is DIFFERENT from that of the more general and more powerful bpadmin commands.

EXIT COMMANDS

a exit first_node_nbr last_node_nbr gateway_endpoint_ID

The add exit command. This command establishes an "exit" for static default routing as described above.

Man Page
bprc :: scheme and endpoint

SCHEME COMMANDS

a scheme scheme_name 'forwarder_command' 'admin_app_command'

The add scheme command. This command declares an endpoint naming "scheme" for use in endpoint IDs, which are structured as URIs: scheme_name:scheme-specific_part. forwarder_command will be executed when the scheme is started on this node, to initiate operation of a forwarding daemon for this scheme. admin_app_command will also be executed when the scheme is started on this node, to initiate operation of a daemon that opens a custodian endpoint identified within this scheme so that it can receive and process custody signals and bundle status reports.

ENDPOINT COMMANDS

a endpoint endpoint_ID { q | x } ['recv_script']

The add endpoint command. This command establishes a DTN endpoint named endpoint_ID on the local node. The remaining parameters indicate what is to be done when bundles destined for this endpoint arrive at a time when no application has got the endpoint open for bundle reception. If 'x', then such bundles are to be discarded silently and immediately. If 'q', then such bundles are to be enqueued for later delivery and, if recv_script is provided, recv_script is to be executed.

Man Page
bprc :: protocol

PROTOCOL COMMANDS

a protocol protocol_name payload_bytes_per_frame overhead_bytes_per_frame [protocol_class]

The add protocol command. This command establishes access to the named convergence layer protocol at the local node. The payload_bytes_per_frame and overhead_bytes_per_frame arguments are used in calculating the estimated transmission capacity consumption of each bundle, to aid in route computation and congestion forecasting.

The optional protocol_class argument indicates the reliability of the protocol. The value 1 indicates that the protocol natively supports bundle streaming; currently the only protocol in class 1 is BSSP. The value 2 indicates that the protocol performs no retransmission; an example is UDP. The value 8 (which is the default) indicates that the protocol detects data loss and automatically retransmits data accordingly; an example is TCP. Protocol class need not be specified when protocol_name is bssp, udp, tcp, stcp, brss, brsc, or ltp, as the protocol classes for these well-known protocols are hard-coded in ION.

Man Page
bprc :: induct and outduct

INDUCT COMMANDS

a induct protocol_name duct_name 'CLI_command'

The add induct command. This command establishes a "duct" for reception of bundles via the indicated CL protocol. The duct's data acquisition structure is used and populated by the "induct" task whose operation is initiated by CLI_command at the time the duct is started.

OUTDUCT COMMANDS

a outduct protocol_name duct_name 'CLO_command' [max_payload_length]

The add outduct command. This command establishes a "duct" for transmission of bundles via the indicated CL protocol. The duct's data transmission structure is serviced by the "outduct" task whose operation is initiated by CLO_command at the time the duct is started. A value of zero for max_payload_length indicates that bundles of any size can be accommodated; this is the default.

Man Page
bprc watch characters

w { 0 | 1 | activity_spec }

The BP watch command. This command enables and disables production of a continuous stream of user-selected Bundle Protocol activity indication characters. A watch parameter of "1" selects all BP activity indication characters; "0" de-selects all BP activity indication characters; any other activity_spec such as "acz~" selects all activity indication characters in the string, de-selecting all others. BP will print each selected activity indication character to stdout every time a processing event of the associated type occurs:

a	new bundle is queued for forwarding
b	bundle is queued for transmission
c	bundle is popped from its transmission queue
m	custody acceptance signal is received
w	custody of bundle is accepted
x	custody of bundle is refused
y	bundle is accepted upon arrival
z	bundle is queued for delivery to an application
~	bundle is abandoned (discarded) on attempt to forward it
!	bundle is destroyed due to TTL expiration
&	custody refusal signal is received
#	bundle is queued for re-forwarding due to CL protocol failure
j	bundle is placed in "limbo" for possible future re-forwarding
k	bundle is removed from "limbo" and queued for re-forwarding
$	bundle's custodial retransmission timeout interval expired

Man Page
ltprc :: initialize

1 est_max_export_sessions

The initialize command. Until this command is executed, LTP is not in operation on the local ION node and most ltpadmin commands will fail.

The command uses est_max_export_sessions to configure the hashtable it will use to manage access to export transmission sessions that are currently in progress. For optimum performance, est_max_export_sessions should normally equal or exceed the summation of max_export_sessions over all spans as discussed below.

Appropriate values for the parameters configuring each "span" of potential LTP data exchange between the local LTP and neighboring engines are non-trivial to determine. See the ION LTP configuration spreadsheet and accompanying documentation for details.

Man Page
ltprc :: span

a span peer_engine_nbr max_export_sessions max_import_sessions max_segment_size aggregation_size_limit aggregation_time_limit
'LSO_command' [queuing_latency]

The add span command. This command declares that a span of potential LTP data interchange exists between the local LTP engine and the indicated (neighboring) LTP engine.

The max_segment_size and aggregation_size_limit are expressed as numbers of bytes of data. max_segment_size limits the size of each of the segments into which each outbound data block will be divided; typically this limit will be the maximum number of bytes that can be encapsulated within a single transmission frame of the underlying link service.

aggregation_size_limit limits the number of LTP service data units (e.g., bundles) that can be aggregated into a single block: when the sum of the sizes of all service data units aggregated into a block exceeds this limit, aggregation into this block must cease and the block must be segmented and transmitted.

aggregation_time_limit alternatively limits the number of seconds that any single export session block for this span will await aggregation before it is segmented and transmitted regardless of size. The aggregation time limit prevents undue delay before the transmission of data during periods of low activity.

max_export_sessions constitutes, in effect, the local LTP engine's retransmission "window" for this span. The retransmission windows of the spans impose flow control on LTP transmission, reducing the chance of allocation of all available space in the ION node's data store to LTP transmission sessions.

max_import_sessions is simply the neighoring engine's own value for the corresponding export session parameter; it is the neighboring engine's retransmission window size for this span. It reduces the chance of allocation of all available space in the ION node's data store to LTP reception sessions.

LSO_command is script text that will be executed when LTP is started on this node, to initiate operation of a link service output task for this span. Note that " peer_engine_nbr" will automatically be appended to LSO_command by ltpadmin before the command is executed, so only the link-service-specific portion of the command should be provided in the LSO_command string itself.

queuing_latency is the estimated number of seconds that we expect to lapse between reception of a segment at this node and transmission of an acknowledging segment, due to processing delay in the node. (See the 'm ownqtime' command below.) The default value is 1.

If queuing latency a negative number, the absolute value of this number is used as the actual queuing latency and session purging is enabled; otherwise session purging is disabled. If session purging is enabled for a span then at the end of any period of transmission over this span all of the span's export sessions that are currently in progress are automatically canceled. Notionally this forces re-forwarding of the DTN bundles in each session's block, to avoid having to wait for the restart of transmission on this span before those bundles can be successfully transmitted.

DTN DevKit – Diagnostics



Agenda

- ion.log file
- watch characters
- bpstats
- sdrwatch

ion.log file

```
[2019/04/14-15:47:03] [i] rfxclock is running.
[2019/04/14-15:47:03] [i] No congestion collapse predicted.
[2019/04/14-15:47:03] [i] ionwarn finished.
[2019/04/14-15:47:04] [i] No congestion collapse predicted.
[2019/04/14-15:47:04] [i] ionwarn finished.
[2019/04/14-15:47:05] Stopping ionsecadmin.
[2019/04/14-15:47:05] [i] Span to engine 2 (max BER 0.000100, max xmit segment size 64000, max recv segment size 1): xmit segment loss rate 0.990000, recv segment loss rate
0.000800, max timeouts 1375.
[2019/04/14-15:47:05] [i] Total max export sessions does not exceed estimate.
[2019/04/14-15:47:05] [i] ltpdeliv is running.
[2019/04/14-15:47:05] [i] ltpclock is running.
[2019/04/14-15:47:05] [i] udplso is running, spec=[10.0.1.2:1113], txbps=1000000 (0=unlimited), rengine=2.
[2019/04/14-15:47:05] [i] udplsi is running, spec=[10.0.1.1:1113].
[2019/04/14-15:47:05] [i] ltpmeter is running.
[2019/04/14-15:47:06] [i] Bundle security is enabled.
[2019/04/14-15:47:06] [i] bpclm is running: ipn:2.0
[2019/04/14-15:47:07] [i] bpclock is running.
[2019/04/14-15:47:07] [i] bptransit is running.
[2019/04/14-15:47:07] [i] ltpcli is running.
[2019/04/14-15:47:07] [i] ipnadminep is running.
[2019/04/14-15:47:07] [i] ltpclo is running.
[2019/04/14-15:47:07] [i] ipnfw is running.
[2019/04/14-15:47:07] [i] ltpclo is running.
[2019/04/14-15:47:08] [x] src from 1969/12/31-19:00:00 to 2019/04/14-15:47:08: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:47:08] [x] fwd from 1969/12/31-19:00:00 to 2019/04/14-15:47:08: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:47:08] [x] xmt from 1969/12/31-19:00:00 to 2019/04/14-15:47:08: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:47:08] [x] rcv from 1969/12/31-19:00:00 to 2019/04/14-15:47:08: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:47:08] [x] dlv from 1969/12/31-19:00:00 to 2019/04/14-15:47:08: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:47:08] [x] ctr from 1969/12/31-19:00:00 to 2019/04/14-15:47:08: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:47:08] [x] rfw from 1969/12/31-19:00:00 to 2019/04/14-15:47:08: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:47:08] [x] exp from 1969/12/31-19:00:00 to 2019/04/14-15:47:08: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:47:08] [?] Duplicate endpoint: ipn:1.5
```

bpstats

```
[2019/04/14-15:50:15] [x] src from 1969/12/31-19:00:00 to 2019/04/14-15:50:15: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:15] [x] fwd from 1969/12/31-19:00:00 to 2019/04/14-15:50:15: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:15] [x] xmt from 1969/12/31-19:00:00 to 2019/04/14-15:50:15: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:15] [x] rcv from 1969/12/31-19:00:00 to 2019/04/14-15:50:15: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:15] [x] dlv from 1969/12/31-19:00:00 to 2019/04/14-15:50:15: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:15] [x] ctr from 1969/12/31-19:00:00 to 2019/04/14-15:50:15: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:15] [x] rfw from 1969/12/31-19:00:00 to 2019/04/14-15:50:15: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:15] [x] exp from 1969/12/31-19:00:00 to 2019/04/14-15:50:15: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:15] [?] Duplicate endpoint: ipn:1.5
[2019/04/14-15:50:18] [i] Span to engine 2 (max BER 0.000100, max xmit segment size 64000, max recv segment size 107):
xmit segment loss rate 0.990000, recv segment loss rate 0.082056, max timeouts 1375.
[2019/04/14-15:50:36] [i] Start of statistics snapshot...
[2019/04/14-15:50:36] [x] src from 1969/12/31-19:00:00 to 2019/04/14-15:50:36: (0) 13 832 (1) 1 12 (2) 0 0 (+) 14 844
[2019/04/14-15:50:36] [x] fwd from 1969/12/31-19:00:00 to 2019/04/14-15:50:36: (0) 0 0 (1) 0 0 (2) 0 0 (+) 14 844
[2019/04/14-15:50:36] [x] xmt from 1969/12/31-19:00:00 to 2019/04/14-15:50:36: (0) 13 832 (1) 1 12 (2) 0 0 (+) 14 844
[2019/04/14-15:50:36] [x] rcv from 1969/12/31-19:00:00 to 2019/04/14-15:50:36: (0) 0 0 (1) 15 856 (2) 0 0 (+) 15 856
[2019/04/14-15:50:36] [x] dlv from 1969/12/31-19:00:00 to 2019/04/14-15:50:36: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:36] [x] ctr from 1969/12/31-19:00:00 to 2019/04/14-15:50:36: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:36] [x] rfw from 1969/12/31-19:00:00 to 2019/04/14-15:50:36: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:36] [x] exp from 1969/12/31-19:00:00 to 2019/04/14-15:50:36: (0) 0 0 (1) 0 0 (2) 0 0 (+) 0 0
[2019/04/14-15:50:36] [i] ...end of statistics snapshot.
```

Watch Characters

- Go to the terminal that started the admin program that caused the watch characters to get generated (or to the one that started ionadmin)
- In any case, for the CORE scenarios, watch characters end up in CORE_ionconfig.out

```
root@n1:/tmp/pycore.36429/n1.conf# tail -f CORE_IONConfig.out
pathName:       '/tmp'
Stopping ionadmin.
Stopping ionadmin.
Stopping ionsecadmin.
Stopping ltpadmin.
Stopping ipnadmin.
Stopping bpadmin.
Stopping cfdpadmin.
chmod: cannot access '/var/tmp/ion/ion.sdrlog': No such file or
directory
```

**bcdefgshgsgstyzgtyzssbcdefgshgsgtyzsbcdefgshgsgtyzsbcdefgshgsgtyzsb
cdefgshgsgtyzsbcdefgshgsgtyzs**

Watch Character Decoding

BP Watch Characters:

- a new bundle is queued for forwarding
- b bundle is queued for transmission
- c bundle is popped from its transmission queue
- m custody acceptance signal is received
- w custody of bundle is accepted
- x custody of bundle is refused
- y bundle is accepted upon arrival
- z bundle is queued for delivery to an application
- ~ bundle is abandoned (discarded) on attempt to forward it
- ! bundle is destroyed due to TTL expiration
- & custody refusal signal is received
- # bundle is queued for re-forwarding due to CL protocol failure
- j bundle is placed in "limbo" for possible future re-forwarding
- k bundle is removed from "limbo" and queued for re-forwarding
- $ bundle's custodial retransmission timeout interval expired

LTP Watch Characters

- d bundle appended to block for next session
- e segment of block is queued for transmission
- f block has been fully segmented for transmission
- g segment popped from transmission queue
- h positive ACK received for block, session ended
- s segment received
- t block has been fully received
- @ negative ACK received for block, segments retransmitted
- = unacknowledged checkpoint was retransmitted
- + unacknowledged report segment was retransmitted
- { export session canceled locally (by sender)
- } import session canceled by remote sender
- [import session canceled locally (by receiver)
-] export session canceled by remote receiver

So the first few characters of that mean…

```
b   (BP) bundle is queued for transmission
c   (BP) bundle is popped from its transmission queue
d   (LTP) bundle appended to block for next session
e   (LTP) segment of block is queued for transmission
f   (LTP) block has been fully segmented for transmission
g   (LTP) segment popped from transmission queue
s   (LTP) segment received
h   (LTP) positive ACK received for block, session ended
g   (LTP) segment popped from transmission queue
s   (LTP) segment received
g   (LTP) segment popped from transmission queue
s   (LTP) segment received
t   (LTP) block has been fully received
y   (BP)  bundle is accepted upon arrival
z   (BP)  bundle is queued for delivery to an application
g   (LTP) segment popped from transmission queue
```

sdrwatch

SDRWATCH(1) ICI executables SDRWATCH(1)

NAME

sdrwatch - SDR non-volatile data store activity monitor

SYNOPSIS

sdrwatch sdr_name [-t | -s | -r | -z] [interval [count [verbose]]]

DESCRIPTION

For count interations (defaulting to 1), sdrwatch sleeps interval seconds and then performs the SDR operation indicated by the specified mode: 's' to print statistics, 'r' to reset statistics, 'z' to print ZCO space utilization, 't' (the default) to call the sdr_print_trace() function (see sdr(3)) to report on SDR data storage management activity in the SDR data store identified by sdr_name during that interval. If the optional verbose parameter is specified, the printed SDR activity trace will be verbose as described in sdr(3).

If interval is zero, sdrwatch just performs the indicated operation once (for 't', it merely prints a current usage summary for the indicated data store) and terminates.

sdrwatch is helpful for detecting and diagnosing storage space leaks. For debugging the ION protocol stack, sdr_name is normally "ion" but might be overridden by the value of sdrName in the .ionconfig file used to configure the node under study.

sdrwatch ion (default behavior)

```
root@n3:/tmp/pycore.46853/n3.conf# sdrwatch ion
-- sdr 'ion' usage report --
small pool free blocks:
             385 of size           32
       total avbl:        12320
     total unavbl:        49000
       total size:        61320
large pool free blocks:
              11 of order           16
              21 of order           32
               1 of order           64
               2 of order          128
               1 of order          256
               3 of order          512
               7 of order         1024
               8 of order         2048
               1 of order         4096
               1 of order        32768
       total avbl:        72784
     total unavbl:        63632
       total size:       136416
total heap size:       40000000
total unused:          39802264
max total used:          197736
total now in use:        112632
max xn log len:               0
```

sdrwatch ion –s (statistics)

```
root@n3:/tmp/pycore.46853/n3.conf# sdrwatch ion -s
-- sdr 'ion' statistics report --
          transaction depth:              0
       transaction log size:              0
 max transaction log length:              0
     transaction log length:              0

                  sdr size:       40002544
              sdr heap size:       40000000
at line 1128 of ici/sdr/sdrmgt.c, Assertion failed. (sdrFetchSafe(sdrv))
[i] Current stack trace:
[i] : /usr/local/lib/libici.so.0(printStackTrace+0x30) [0x7f3f646a65b0]
[i] : /usr/local/lib/libici.so.0(_iEnd+0x21) [0x7f3f646a6b71]
[i] : /usr/local/lib/libici.so.0(sdr_usage+0xb0) [0x7f3f646cfc90]
[i] : /usr/local/lib/libici.so.0(sdr_stats+0x18b) [0x7f3f646d030b]
[i] : sdrwatch(+0x11c2) [0x55666e5c21c2]
[i] : /lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xe7) [0x7f3f642c9b97]
[i] : sdrwatch(+0xe4a) [0x55666e5c1e4a]
            small pool size:          61320
            small pool free:          12992
            large pool size:         136416
            large pool free:          75648
            unassigned free:       39802264

            sdr heap in use:         109096
        max sdr heap in use:         197736
root@n3:/tmp/pycore.46853/n3.conf#
```

sdrwatch ion –z (zco space utilization)

```
root@n3:/tmp/pycore.46853/n3.conf# sdrwatch ion -z
-- sdr 'ion' statistics report --
          transaction depth:              0
       transaction log size:              0
 max transaction log length:              0
     transaction log length:              0

                  sdr size:       40002544
             sdr heap size:       40000000
at line 1128 of ici/sdr/sdrmgt.c, Assertion failed. (sdrFetchSafe(sdrv))
[i] Current stack trace:
[i] : /usr/local/lib/libici.so.0(printStackTrace+0x30) [0x7f61d3ddd5b0]
[i] : /usr/local/lib/libici.so.0(_iEnd+0x21) [0x7f61d3dddb71]
[i] : /usr/local/lib/libici.so.0(sdr_usage+0xb0) [0x7f61d3e06c90]
[i] : /usr/local/lib/libici.so.0(sdr_stats+0x18b) [0x7f61d3e0730b]
[i] : sdrwatch(+0xfcb) [0x55701c3c3fcb]
[i] : /lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xe7) [0x7f61d3a00b97]
[i] : sdrwatch(+0xe4a) [0x55701c3c3e4a]
            small pool size:          61320
            small pool free:          13120
            large pool size:         136416
            large pool free:          76096
            unassigned free:       39802264

            sdr heap in use:         108520
        max sdr heap in use:         197736

[i] inbound file  max: 4294967295  current: 0
[i] inbound bulk  max: 4294967295  current: 0
[i] inbound heap  max: 8000000  current: 0
[i] outbound file  max: 4294967295  current: 0
[i] outbound bulk  max: 4294967295  current: 0
[i] outbound heap  max: 8000000  current: 0
```