Open resources for practicing computer networks

Fraida Fund ffund@nyu.edu

This document includes (1) resources for lab assignments, and (2) resources for problem sets, both of which can be used to give students more practice opportunities.

Lab exercises for computer networks

There are a variety of "lab platforms" through which students can get hands-on experience with computer networks. The lab exercises shared here run on GENI, an open testbed for networking research and education. Briefly, some benefits of this platform for education are:

- minimal hardware/software requirements (from you and from your students)
- supports open-ended exploration - students can go beyond the assignment and experiment on their own, develop open-ended projects directly from lab assignments,, etc.
- students work with Linux and Linux-based networking utilities directly (including all the good and all the bad elements of this!) rather than a platform-specific interface.

We have developed GENI-based labs on an assortment of networking topics, which you can find at:

- https://witestlab.poly.edu/blog/tag/education/
- https://ffund.github.io/tcp-ip-essentials/ (this is an example of a bottom-up networking course with weekly lab assignments)

Try it: If you have used GENI before, you can go ahead and try any lab exercise from the links above on GENI! Feel free to contact *ffund@nyu.edu* if you have questions or concerns, or if you want to try one of these without going through the process to be "verified" as a PI in GENI.

Question bank for networking problems

The second resource is a question bank of networking problems for PrairieLearn (a learning system for homeworks and tests developed at UIUC). These questions are randomized, so that students can practice with new "variants" of a homework question as often as they want. (Randomization also discourages unauthorized collaboration.)

When a student answers a question on PrairieLearn, they can see immediate feedback and explanation that is *specific* to the question variant and/or to their own submission. Here's an example of a question with student submission (left) and the feedback that a student would see after submission (right), which is specific to their own randomized variant of the question:

The routing table at a router includes the following rule:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.64.0	0.0.0.0	255.255.192.0	U	600	0	0	eth2

Which of the following addresses match this rule? Select all that apply.

- [ ] (a) 192.168.120.16
- [x] (b) 192.168.82.106
- [ ] (c) 192.168.63.252
- [ ] (d) 192.168.63.248
- [ ] (e) 192.168.128.10

Select all possible options that apply.

50%

Correct answer

Which of the following addresses match this rule? Select all that apply.

(a) 192.168.120.16
(b) 192.168.82.106

Comment This rule matches addresses from **192.168.64.0** to **192.168.127.255**.

The bottom end of this range is the network address, which is given in the routing table rule: 192.168.64.0.

To find the top end of this range, compute the network address OR the *inverse* of the subnet mask:

```
11000000 10101000 01000000 00000000
00000000 00000000 00111111 11111111
-----------------------------------
11000000 10101000 01111111 11111111
```

to get 192.168.127.255.

The question bank includes a variety of question types - in addition to the standard string/numeric input, multiple choice/multi-select, matching, and dropdown questions, PrairieLearn also supports question types like "drag these packets from one panel and put them in order in the next panel" or "make selections on a series of dropdown menus or checkboxes overlaid on a network diagram":

Describe the sequence of packets you will see on the network (note that not all of these blocks will be used!):

Drag from here:

```
00:16:3e:af:4f:43 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42:
    Request who-has 192.168.82.57 tell 192.168.82.216, length 28
```

```
00:15:5d:ef:e3:d1 > 00:16:3e:af:4f:43, ethertype ARP (0x0806), length 42:
    Reply 192.168.82.57 is-at 00:15:5d:ef:e3:d1, length 28
```

```
00:16:3e:af:4f:43 > 00:15:5d:ef:e3:d1, ethertype IPv4 (0x0800), length 98:
    192.168.82.216 > 192.168.82.57: ICMP echo reply, id 507, seq 1, length 64
```

```
00:16:3e:af:4f:43 > 00:15:5d:ef:e3:d1, ethertype ARP (0x0806), length 42:
    Reply 192.168.82.216 is-at 00:16:3e:af:4f:43, length 28
```

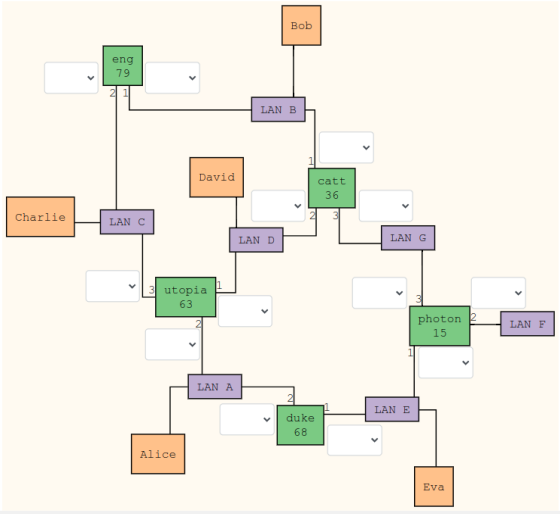
```
00:15:5d:ef:e3:d1 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42:
    Request who-has 192.168.82.216 tell 192.168.82.57, length 28
```

```
00:15:5d:ef:e3:d1 > 00:16:3e:af:4f:43, ethertype IPv4 (0x0800), length 98:
    192.168.82.57 > 192.168.82.216: ICMP echo request, id 507, seq 1, length 64
```

Construct your solution here:

On the following network diagram, each bridge (green) is labeled with a two-digit bridge ID beneath its name. Hosts are shown in orange.

Next to each bridge port, indicate its final state (designated port, root port, or blocking) after the spanning tree protocol has converged.



Try it: I have set up a "reference" course so you can test a few practice questions and see what the experience is like as a student. Try it at: https://www.prairielearn.org/pl/course_instance/129160 (you can sign in with any Google account).

If you are an instructor who is interested in using these questions, contact *ffund@nyu.edu* with your Github username to gain access to the question repository.