1 Implementing flooding

Begin with the code from Project 2a. Use your previous RandomNetworkLayer.java as a starting point for a new code file, FloodNetworkLayer.java in which you’ll implement flooding.

In this new routing implementation, when a packet arrives, the FloodNetworkLayer should determine whether the packet is destined for this host, and if so, deliver the data; if the packet is destined for another host, it should resend the packet on all links.

The set of methods within FloodNetworkLayer that need your attention are:

- **createPacket()**: The packet header may need to be modified to carry data needed for flooding purposes.

- **processPacket()**: Likewise, processing a received packet must handle any changes to the header layout and contents. Here is where packets not destined for this host must be forwarded through all links.

- **route()**: **This method should never be called.** An error should occur if this method is invoked, because there’s no routing choice to be made.

- **send()**: This method need to be overridden. The standard send() relies on route() to choose a single link, but flooding requires a different behavior, so send() must be specialized.

A few key considerations:

- Resending of the packet should not become a runaway process; the resending needs to be limited to avoid overwhelming the network.

- By default, duplicate packets may arrive at the destination. These multiple arrivals are acceptable.

- As an **extra challenge**, you may devise and implement a method that avoid the delivery of duplicates at the destination; the receiving host will identify later arrivals and discard them.³

³This extra challenge is indeed extra: you will not be graded on it, but it would be good practice with the concepts!
2 How to submit your work

Go to GradeScope for our course, where you can submit your work. Notice that you should only submit FloodNetworkLayer.java. Don’t submit the other source code or class files.

This assignment is due on Sunday, Dec-03, 11:59 pm.