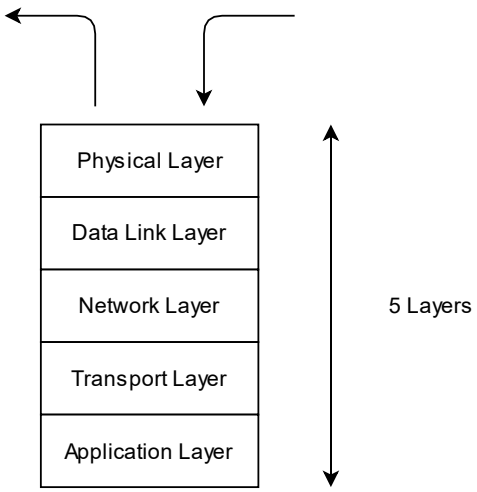
Use of Queues and Stacks

C++, like many other programming languages, has built-in implementations of a queue and a stack data structure, these are respectively called std::queue and std::stack. Besides these, there is also a double-ended queue called std::deque, and a priority queue called std::priority_queue.

Protocol Headers

Through some of your other classes, you are undoubtably familiar with the 5-layer network stack (Physical, Datalink, Network, Transport and Application). In this lab, we will build a simulation of a router in software, using stack and queue data structures.

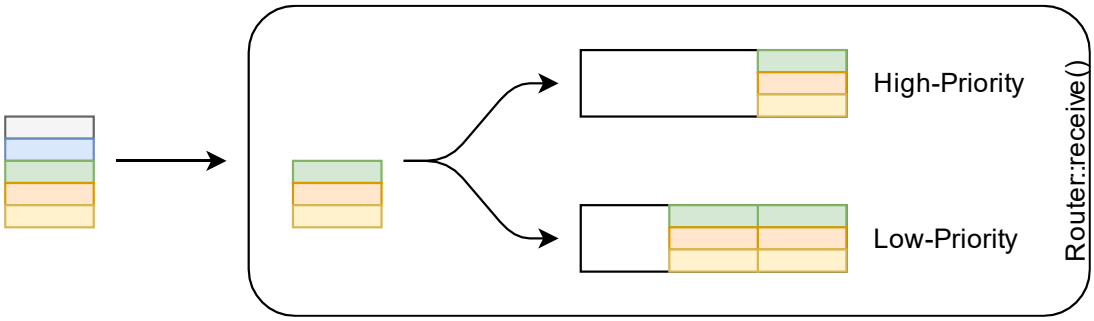


Code

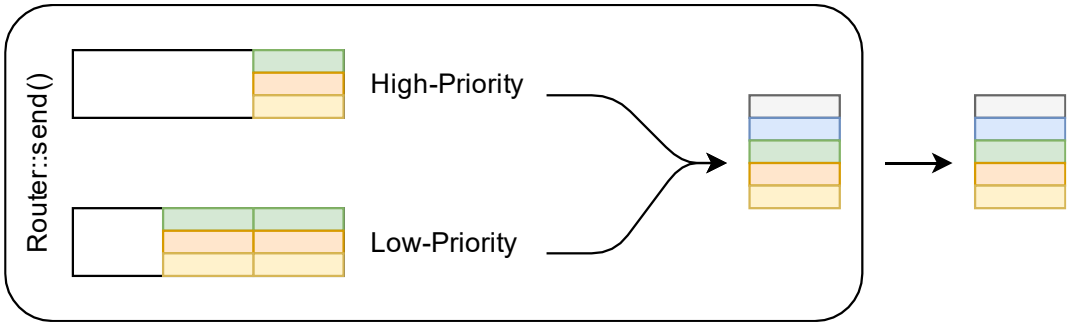
1. The first thing we'll need is a data structure to represent our network packets. For this, we will use a simple stack. Each element in the stack represents one layer of the 5-layer network stack. The application layer will be at the bottom of the stack, and the physical layer is at the top of the stack. This might sound counter-intuitive at first, but this way, we can pop-off headers in the correct order, without ever having to touch any of the other layers. To create this data structure, we will use a simple type alias. Create a type called "Packet" and alias it to a stack. Each "header" in our network stack will be represented by a single integer, to keep things simple. While the physical layer usually doesn't have headers in the real world, we will use our physical layer "header" to represent the type of physical layer (Optical, Copper, Wireless, ...).
2. Now that we have our "Packet" data structure, we need a function to easily generate packets. Create a function called "create_packet" that takes 5 integers as arguments, 1 for each layer of the network stack. This function should then create a "Packet" object and push the integers onto the stack in the correct order.
   You should also write some unit tests for "create_packet", you should check that the output stacks always has 5 elements, and that all elements are presentin the correct order.
3. Next, we will create a class called "Router". This class will represent a router in our network. A Router should have the following fields:
   - A physical layer type (an integer)
   - A MAC address (an integer)
   - A low-priority queue (std::queue<Packet>)
   - A high-priority queue (std::queue<Packet>)

   You should be able to set the MAC address and physical layer type in the constructor. Both queues should be initialized as empty in the constructor.

4. After creating the router class, we will add a method to the router for receiving a packet. This method should be called “receive”, and as an argument, it should take a “Packet&&”, the method doesn’t return anything. When the router receives a packet, the physical and data link layer headers should be popped off and the router should look at the network layer header. Specifically, it should look at the most significant bit in the network layer header. If this bit is 1, the packet should be placed in the high-priority queue. Otherwise, the packet should be placed in the low-priority queue.



5. Now that our router can receive packets, we should also make it capable of sending packets. Add a “send” method to your “Router” class. This method takes no arguments, and returns a std::optional<Packet>. When this method is called, the router should first check if there is anything in the high-priority queue. If there is, we send that packet. If the high-priority queue is empty, we take a packet from the low-priority queue. If both queues are empty, we return std::nullopt. After taking a packet from either queue, we need to add a new data link and physical header before sending the packet out again. For these headers, you can take the value of the Router’s MAC address and physical layer type, which you set in the constructor.



Report

1. In our “Router” class, can we replace both queues with a simple priority queue? Why, or why not?
2. In the “receive” method, you used std::optional to return “a value, or nothing”. The standard library also has a class for returning “one of many types” and “many types” in 1 object. What are they?
3. std::stack, std::queue and std::priority_queue are container adaptors, while std::vector, std::list and std::map are containers. What is the difference between a container adaptor and a container?

Extra

1. Make a change to the “receive” method. Your queues should now have a limited capacity. If the correct queue is full, the incoming packet should be dropped. You can change the return type of the method to “bool”, for instance, to indicate to the user that the packet was dropped.
2. Build another class called “Switch”, this class should do the same things the router class does, but only at the data link level, rather than the network level.