Task 0: Identifying Flit and Packets

For this task, we will try a debugging method in BookSim by printing the activity of a packet. This is useful when we want to know why a packet arrives much slower than the other packets, why a packet is trapped in a deadlock, etc.

To do this, we need to make some modification in `trafficmanager.cpp`. In the `_GeneratePacket` function, add the code below.

```
        if(gTrace){
            cout<<"New Flit "<<f->src<<endl;
        }
        f->type = packet_type;

        if ( i == 0 ) { // Head flit
            f->head = true;
            //packets are only generated to nodes smaller or equal to limit
            f->dest = packet_destination;
        } else {
            f->head = false;
            f->dest = -1;
        }
        switch( _pri_type ) {
        case class_based:
            f->pri = _class_priority[cl];
            assert(f->pri >= 0);
            break;
        case age_based:
            f->pri = numeric_limits<int>::max() - time;
            assert(f->pri >= 0);
            break;
        case sequence_based:
            f->pri = numeric_limits<int>::max() - _packet_seq_no[source];
            assert(f->pri >= 0);
            break;
        default:
            f->pri = 0;
        }
        if ( i == ( size - 1 ) ) { // Tail flit
            f->tail = true;
        } else {
            f->tail = false;
        }

        f->vc  = -1;

        if (f->id == 1)
                f->watch = true;

        if ( f->watch ) {
            *gWatchOut << GetSimTime() << " | "
                       << "node" << source << " | "
                       << "Enqueuing flit " << f->id
                       << " (packet " << f->pid
                       << ") at time " << time
                       << "." << endl;
        }

        _partial_packets[source][cl].push_back( f );
    }
}
```

Add the following line to the configuration file (`examples/flatflyconfig`). This tells BookSim to dump the debugging message to the terminal window (`cout`).

```
watch_out = -;
```

Don't forget to run `make` after making any changes to the code (but not the configuration file).

The activity of all flits whose `watch` variable is set to true will be printed out, as partly shown below.

```
0 | node119 | Enqueuing flit 1 (packet 1) at time 0.
0 | traffic_manager | Finding output VC for flit 1:
0 | traffic_manager |   Selected output VC 0.
0 | node119 | Injecting flit 1 into subnet 0 at time 0 with priority 0.
1 | network_0/network_0_fchan_ingress119 | Beginning channel traversal for flit 1 with delay 1.
1 | network_0/network_0_fchan_ingress119 | Completed channel traversal for flit 1.
2 | network_0/router_7 | Received flit 1 from channel at input 7.
2 | network_0/router_7 | Adding flit 1 to VC 0 at input 7 (state: idle, empty).
2 | network_0/router_7/buf_7/vc_0 | Changing state from idle to routing.
2 | network_0/router_7 | Beginning routing for VC 0 at input 7 (front: 1).
2 | network_0/router_7 | Completed routing for VC 0 at input 7 (front: 1).
2 | network_0/router_7/buf_7/vc_0 | Changing state from routing to vc_alloc.
3 | network_0/router_7 | Beginning VC allocation for VC 0 at input 7 (front: 1).
```

If you want to see which BookSim file that prints the message, you can the `grep` command, as shown below. In the example below, we know that the message is printed by `routers/iq_router.cpp`.

```
[hanskasan@intel03 src]$ grep -r "Beginning VC allocation" .
./routers/iq_router.cpp:                    << "Beginning VC allocation for VC " << vc
Binary file ./routers/iq_router.o matches
Binary file ./booksim matches
```

Questions

1. What is the `f->id` variable used for in BookSim? How is it different from `f->pid`? (*Hint: look into the difference between flit and packet. See Figure 12.2 in Dally's textbook)*
2. When `packet_size` is set to 1, each flit will have the same `f->id` and `f->pid`. Why? (*Hint: the packet size can be configured from the configuration file)*
3. When `packet_size` is set to a value larger than 1, each flit will have different `f->id` and `f->pid`. Why? (*Hint: try to print the `f->id` and `f->pid` values when the flit is generated at the traffic manager.)*

Task 1: Looking into Zero-load Latency

Run BookSim simulation on 1D Flattened Butterfly topology with uniform random traffic and very low load, as shown in the example below. Set the `watch` variable of any flit to `true`, as shown in Task 0.

```
./booksim examples/flatflyconfig injection_rate=0.025 traffic=uniform
```

Questions

Please refer Equation 3.11 in Section 3.3.2 of Dally's textbook. Fill in the value for the variables below.

Variable	Definition	Value?
T_r	Latency of a single router	
T_w	Length of a single channel	
N_r	Total number of traversed routers	
N_w	Total number of traversed channels	

Task 2: Zero-load Latency on Other Topologies

In this task, we will try to understand that a topology sets the minimal bound for the packet latency, e.g. the zero-load latency.

To do so, we will run BookSim on a ring (1D-torus, refer to Figure 5.1a in Dally's textbook). To do so, we will use `examples/torus88` as the configuration file and change the value of `n` (number of dimension) as shown below. It's recommended that you copy the `examples/torus88` to make another configuration file (for example `examples/torus18`)

```
n = 1;
```

Add the following line to configuration file.

```
// Router architecture
vc_allocator = islip;
sw_allocator = islip;
alloc_iters  = 3;

credit_delay   = 0;
routing_delay  = 1;
vc_alloc_delay = 1;
sw_alloc_delay = 1;
st_final_delay = 1;

input_speedup     = 1;
output_speedup    = 1;
internal_speedup  = 1.6;

// Latency
use_noc_latency = 0;

// Debugging
watch_out = -;
```

Run with uniform random traffic at low load.

```
./booksim examples/torus18 injection_rate=0.025 traffic=uniform
```

Questions

1. Using the values obtained from table 1, count (by hand) your expected zero-load latency for a 8-node 1D torus.
2. Compare your expected zero-load latency with the BookSim result. Are they the same?