Project: Network Simulator

Goal

Your goal is to gain an appreciation of the various aspects of a wide area network, albeit highly abstract, such as:

- basics: network topology, routing, congestion control, queueing (FIFO);
- performance metrics: throughput, backlog, packet delay, packet loss;
- advance concepts: fairness, distributed computing, feedback control, subtle interaction of different (heterogeneous) congestion control algorithms over the same network (open research).

Equally important is to learn about collaboration and management of a software project.

Here we provide the basic project descriptions. More details and/or updates might be added as the project progresses.

Simulator specification

The project is to develop a simple simulator for a general multi-link multi-flow network. Your simulator should take as input:

- network topology specified by:
  - network graph,
  - capacity of each link in bits per second (bps),
  - propagation delay in ms across each link,
  - buffer size in kilo-bytes (KB) at each link;
- flows specified by:
  - pairs of source-destination nodes,
  - the total number of bits each flow wishes to send, or probabilistic models for flow size,
  - the start time of each flow,
  - congestion control algorithm for each flow.
- a set of links where buffer occupancies, packet losses, and link flow rates will be measured;
- a set of flows where the sending and receiving rates (throughputs), and packet delays will be measured;

and output:

- time traces of buffer occupancies, packet losses, and link flow rates of the specified links;
- sending and receiving rates, and the packet delays of the specified flows;
- aggregate statistics such as average throughput of each flow, average and variance of buffer occupancy (backlog), average packet delay and loss.

Each link (buffer) implements FIFO and a packet will be dropped when it arrives at a full buffer. Each packet is assumed to be 1KB, including header and tailer. The network also implements a shortest-path routing algorithm that routes each flow from its source to its destination. Each source can use one of a selection (probably two) of congestion control algorithms. The routing and congestion control algorithms, flow size models, and other simulation details will be

discussed and fine tuned *in lectures throughout the quarter* (see "Format and schedule" below). For the first three weeks, your task is to learn the basic knowledge for the project, both in class and self-learning. At the same time you should think about the simulator architecture and module specifications. You are required to update this part in class during the fourth week. Afterwards, you should commit to steadily working on the project every week. We have set some milestones (see "Format and schedule" below).

You are free to choose the programming language and operation system, as long as you can clearly demonstrate your project results during and at the end of the project.

In the 2nd week after the teams are organized, one simulation spec(s) (network topology, flows, desired measurements) will be provided to each project team to better understand the project requirement. In the 6th week, the final simulation spec(s) will be provided and the simulation output from each team will be used to assess the correctness and quality of their simulator.

Format and schedule

The project is an integral part of the course and we will discuss the details of the simulation throughout the course. Students are highly recommended to attend the TA office hour to discuss project progress. For Week 4, Week 6, and Week 8, the first part of Fri lecture will be devoted to project update. Each project team will have 10 min to summarize the work they have done in the past week and the milestones achieved. There are 5 major milestones according to the following tentative schedule:

Week	Date	Details
1	9/29	Project description uploaded
2	10/6	M0: team organization, you need to send TA email about your team member and organization, e.g. project manager.
3	10/13	
4	10/20	M1: Simulator architecture, module spec and owners.
5	10/27	
6	11/3	M2: Version 1.0 with basic functions.
7	11/10	
8	11/17	M3: Version 2.0 with full functions.
9	11/24	
10	12/1	M4: Project presentation, demo, and submit code with Readme.

Grading

Project total: 40%

- Update 10%
- Simulator correctness and quality 20%
- Presentation and documentation 10%

Project management

Each project team should have 4-5 students. One student in each team will also serve as the project manager in addition to being an individual contributor as everyone else. The technical work, architecting the simulator, prioritizing tasks, developing the modules, integration and testing should be divided roughly equally among all team members, *including* the project

manager. The role of the project manager is *not* technical; it is to maintain project schedule, find out roadblocks of each team member, coordinate and manage meetings to address them, and facilitate communication and collaboration within the team. This is critical for the success of the whole team and must be taken very seriously. The role of project management is in addition to the technical contribution s/he is accountable for like every other team member.

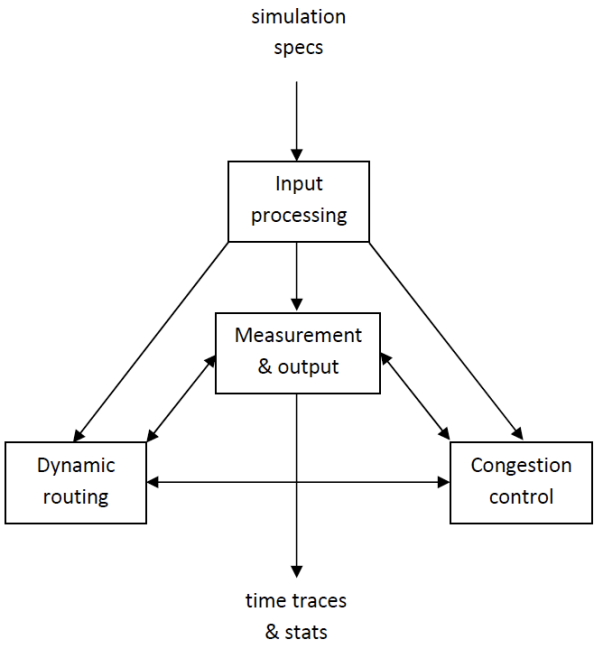
It is important that the architecture specifies in detail each simulator modules and their integration. Each module should be assigned an owner who will be responsible for weekly updates on that module.

More details

Here are some minimum requirements for your simulator.

- The flow size can be specified by a probability distribution, e.g., Zipf distribution (power law).
- Routing: it should implement either a link‐state protocol (Dijkstra) or a distance‐vector protocol (Bellman‐Ford). The link cost is a weighted sum of static and dynamic component. For instance, you can take the static component to be the propagation delay of the link and the dynamic component to be the (average) queueing delay of the link, averaged over the routing update period. It should implement message passing among the nodes to compute the routing asynchronously.
- Congestion control: It should implement at least two congestion control algorithms. One can be AIMD (Reno) and the other one can be FAST TCP. Each source and destination should implements window control. The source will transmit packets, which go through the set of links determined by the routing algorithm, and collect (implicitly) congestion information (loss or delay it experiences itself) and implicitly feedback to the source. The source then adjusts the window based on the end-to-end congestion information.

There are at least four high‐level modules, shown below, each of which needs to be expanded in much greater detail:



You also need to coordinate the design of each module and decide on API's early on. This will make your implementation and debugging much easier. You can take a look at the following references to get a sense of real implementation of TCP/IP. Even though you do not have to implement the real protocol, it will help you decide what to include in your simulator.

- Richard Stevens, TCP/IP Illustrated, Addison-Wesley
- Douglas Comer, Internetworking with TCP/IP, Prentice Hall