Quality-of-service routing

João Luís Sobrinho

September 19, 2022

1 Introduction

In Quality-of-Service (QoS) routing, every link and every path in a network is characterized by a width and a length with paths selected for routing as some function of their widths and lengths. The width of a path is the minimum of the widths of its constituent links; the length of a path is the sum of the lengths of its constituent links. Therefore, the extension of pair width-length (w, l) of some path P with pair width-length (x, m) of some other path Q yields pair width-length $(\min\{w, x\}, l + m)$ of the concatenation PQ of the two paths.

There are two rather opposite criteria for selecting width-length pairs and the corresponding paths. In the widest-shortest order, pairs of shorter length are preferred and, among pairs of common length, those of wider width are preferred; a widest-shortest path from a source to a destination in a network is a widest of the shortest paths from source to destination. Contrarily, in the shortest-widest order, pairs of wider width are preferred and, among pairs of common width, pairs of shorter length are preferred; a shortest-widest path from a source to a destination in a network is a shortest of the widest paths from source to destination.

In the figure, each link is annotated with a pair width-length. For example, link uv has width 10 and length 2, while link wx has width 20 an length 1. There are three paths from u to x: uwx with width-length $(5, 2)$; $uvwx$ with width-length $(10, 7)$; and uvx with width-length $(10, 4)$. The widest-shortest path from u to x is path uwx, because it has the shortest length of all three paths; the shortest-widest path from u to x is path uvx, since it has wider width than path uwx and the same width but shorter length than path $uvwx$.

The broad goals of this project are to compute the two types of QoS paths referred to above on input networks with a routing protocol and with sequential algorithms, and compare the results across the two methods and across the two orders for pairs width-length. More specifically, you shall:

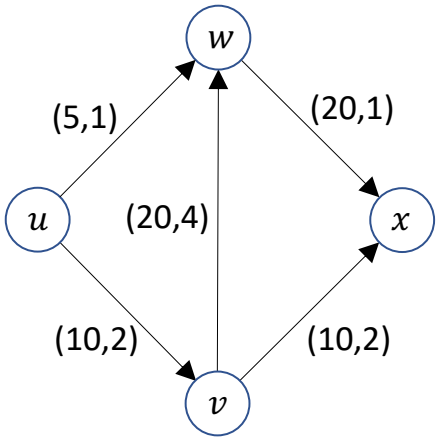

Figure 1: A network where links are annotated with pairs width-length.

- Build a simulator of a QoS vectoring protocol for the widest-shortest path order and the shortest-widest path order, and present stable state statistics of lengths and widths on input networks.
- Implement an efficient algorithm that computes the stable state of a QoS vectoring protocol for the widest-shortest path order and the shortest-widest path order, and check that it produces the same stable state results on input networks as the simulator, but much faster.
- Design and implement an efficient algorithm to compute shortest-widest paths from a source to a destination in a network, present the statistics of widths and lengths, and compare them with those of the algorithm that computes the stable state of the QoS vectoring protocol for the shortest-widest order.
- Investigate what happens when a QoS link-state protocol, rather than a QoS vectoring protocol, is used.

Each input network is presented as a file where each line represents a link containing, in order, the identity of the tail node, the identity of the head node, the width of the link, and the length of the link, separated by spaces. All networks have node identifiers in the range from 0 to 4095.

2 Simulation of a QoS vectoring protocol

A QoS vectoring protocol generalizes a distance vector protocol by substituting the less-than-or-equal order on lengths for the widest-shortest order or the shortest-widest order on pairs width-length, as the case may be, and addition by the extension operation on pairs width-length. Such a protocol executes a separate routing process per destination. For any given destination, every

node extends the width-length of its links to the out-neighbors with the respective width-lengths advertised by them, elects the most preferred width-length among those learned from the out-neighbors, and advertises it to all its in-neighbors. At any moment in time, the node forwards data-packets to the out-neighbors from which the elected width-length was learned. Eventually, the protocol terminates and data-packets reach the destination, hop by hop, along loop-free paths. A destination initializes a routing computation process by advertising itself to its in-neighbors.

The stabilization time of a node is the time it takes for the elected width-length at the node to remain unchanged ever since. Suppose that a QoS vectoring protocol for the shortest-widest order is run on the network of the figure and that x initiates the computation by advertising itself to v and w. Node w will elect $(20, 1)$ learned directly from x. Node v learns $(10, 2)$ directly from x and $(20, 5)$ from w, resulting from the extension of $(20, 4)$ of link vw and $(20, 1)$ of link wx; it elects $(20, 5)$ because of its wider width. Node u learns $(5, 2)$ from w and $(10, 7)$ from v; it elects $(10, 7)$. Data-packets arriving at u with destination at x are forwarded to v, there they are forwarded to w, and finally delivered to x, overall traversing path $uvwx$.

You will build a discrete-event simulator for QoS vectoring protocols. The underlying assumptions of the simulator are: (1) routing messages are delivered across each link in First-In First-Out (FIFO) order; (2) subject to FIFO order, each routing message incurs a delay of one unit of time plus a random value taken from a uniform distribution between zero and two. Note that routing messages travel in the opposite direction of links, from head to tail. The operation of a discrete-event simulator revolves around a calendar list containing all events that are currently scheduled ordered by their time of occurrence. At each step of the simulation, the next scheduled event is removed from the calendar and processed according to the specific rules of the protocol. This processing may schedule subsequent events to be inserted in the calendar, in due order, for future processing.

From the output of the simulator you should present the following statistics:

- The *Complementary Cumulative Distribution Function (CCDF)* of the stabilization times of the protocol, over all source-destination pairs;
- The CCDF of the width with which a node reaches a destination, over all source-destination pairs;
- The CCDF of the length with which a node reaches a destination, over all source-destination pairs.

For confidence on the results, consider running various replicas of the protocol on a common network with different random values for the delays across links.

The simulator should also have an interactive testing mode. In such a mode, the user inputs a network, a source, and a destination and the simulator outputs the width and length of the most preferred path from source to destination according to the two orders.

3 Algorithm to compute stable states

Running simulations is computationally expensive. If all that is needed are the widths and lengths of routing paths in stable state, then we may resort to an efficient sequential algorithm to compute those quantities. You are asked to implement an efficient stable-state algorithm that computes the widths and lengths of a QoS vectoring protocol in stable state, for both the widest-shortest path and the shortest-widest path orders. With that algorithm, compute:

- The CCDF of the width with which a node reaches a destination, over all source-destination pairs;
- The CCDF of the length with which a node reaches a destination, over all source-destination pairs.

The simulator and the algorithm can be validated against each other: for the same network, the stable state statistics obtained from simulation and from the algorithm should be exactly the same. These statistics do not depend on the delay of routing messages across the links of the network. The algorithm should run much faster than the simulator and, thus, be able to produce results for very large networks. However, the transient behavior of routing protocols can only be assessed through simulation.

The stable-state algorithm should also have an interactive testing mode, just like the simulator.

4 Algorithm for shortest-widest paths

You may notice that the QoS vectoring protocol operating according to the widest-shortest order does indeed compute widest-shortest paths. Counter-intuitively, the QoS vectoring protocol operating according to the shortest-widest order does not compute shortest-widest paths. For example, as discussed above, node u will elect width-length $(10, 7)$ corresponding to path $uvwx$. However, the shortest-widest from u to x is uvx with width-length $(10, 4)$.

You are now asked to design an efficient algorithm that computes shortest-widest paths from a source to a destination in a network. Compare the CCDF of widths and that of lengths produced by the shortest-widest path algorithm with the widths and lengths, respectively, obtained at the stable state of the corresponding QoS vectoring protocol.

5 What about link-state protocols?

This project started with QoS vectoring protocols, which generalize distance-vector protocols to the widest-shortest order and the shortest-widest orders. We now want to know what would happen with QoS link-state protocols. In a standard shortest paths link-state protocol, each node floods information about the lengths of its links to its out-neighbors all over the network. Eventually, all nodes have a complete map of the network. With its own map of the network, even when not complete, a node computes shortest paths from itself to all other nodes using Dijkstra's algorithm. For each destination, a node forwards data-packets to the out-neighbor along the computed shortest path to the destination.

In a generalized link-state protocol, Dijkstra's algorithm is generalized by substituting the less-than-or-equal order by the widest-shortest order or the shortest-widest order, as the case may be, and addition by the extension operation on pairs width-length. Does a link-state protocol operating according to the widest-shortest order route data-packets along widest-shortest paths? And what happens to a link-state protocol operating according to the shortest-widest order?

6 Report and evaluation

Students work on the project in groups of two. Each group will write a report with no more than five pages summarizing its findings. The report must include the following:

- A description of the implementation of the simulator, highlighting design or implementation decisions that are not directly derived from this text;
- A description of the stable-state algorithm with a convincing argument for its correctness, its complexity, and implementation decisions that the group considers important;
- The statistics obtained from the simulator and computed with the stable-state algorithm for two files that will be given later on;
- The statistics obtained with the shortest-widest path algorithm and a critical comparison with the obtained from the QoS vectoring protocol operating according to the shortest-widest order;
- Your overall conclusions about the subject of this project.

In order to evaluate your project, I will consider the following aspects:

- I will start by reading your report, which is the means for you to communicate your ideas with others. Organize your report in sections; present high-level, but precise descriptions of

your algorithms, highlighting their most subtle steps, if any; draw concise, but unambiguous conclusions.

- If the report is readable, I will run some tests on your programs.
- I will have a discussion with each group, at which time we test your programs and debate your choices. The discussion occurs at the end of the semestre.
- A mark with be given individually to each member of the group.

I will keep the option of asking for some small, extra requirements on the project on the first weeks — thus, please stay in tune. You are welcome to share ideas with your colleagues, but the code of the simulator, the design of the algorithm, and the conclusions of the report must absolutely be your own. Start early, on the first week really, and share your difficulties and insights with me during the course of the period.

The code and the report should be sent in a .zip file to my email address with subject para.<group number>.zip where <group number> is your group number. The deadline is Sunday, October 23, at 23:59. I plan to have the discussions on the week starting on October 31 .