Software-Defined Networking Traffic Engineering

Team cmos: Kristi Zhang, Benjamin Hu, Grace Han, Victor Cai
Repo: https://github.com/benbananas/NetworksFinal

1 Introduction

Network traffic engineering is the process of routing traffic between different nodes in a network in order to achieve some performance metric (throughput, fairness, latency) while respecting the network's resource limits. In doing so, traffic engineering develops routing mechanisms that help improve utilization of the network. Compared to traditional networking, software-defined networking (SDN) incorporates software in the network control plane and separates it from the data-forwarding plane (hardware). In SDN, one administrator has an abstracted centralized view of the distributed network state and is responsible for all the routing decisions. In this report, we explore the use of linear programming to define a throughput optimization formulation and a fairness optimization formulation for Sprint's network in a SDN fashion.

2 The Network Setup

Sprint's network topology (Figure 1) includes 11 nodes and undirected links/edges, each consisting of a capacity of 1,000,000,000 Gbps.

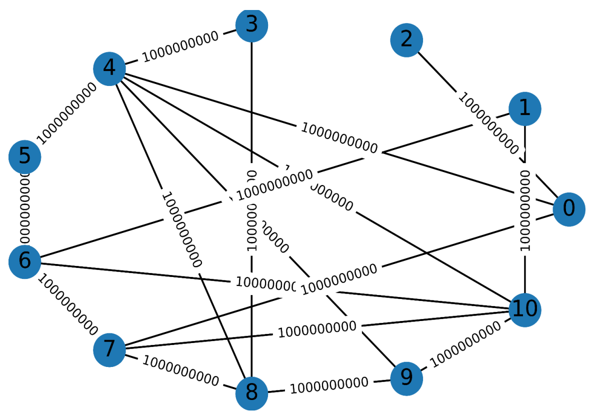

Figure 1. The visualization of Sprint's network topology

2.1 Network Demands

The demand from node A to node B represents the capacity that A wants along the tunnel(s) or path(s) to B. Given dynamic demands over a period of time from each node to every other node, we chose to represent the demands statically through the maximum demand over that entire period for each pair of nodes A and B, where A demands some capacity to B (note: the demand from A to B does not necessarily equal the demand from B to A). The formulation must be able to capture the internet's bursty nature so using the max across each pair of nodes gives a reasonable upper bound to the burst in demand.

3 SDN Formulation

The formulation of a linear program consists of decision variables, constraints, and an objective. These requirements are able to capture the complexity behind the SDN traffic engineering problem because the objective represents the performance metric to be enhanced and the constraints represent the network's limited resources. Then, using the Gurobi solver, we were able to solve the problem. Both formulations used the same decision variables and one shared constraint. The difference between the formulations is the use of the demand constraint.

Decision Variables: Each decision variable represents flow along a specific path/tunnel between two nodes. Since the network topology only consists of 11 nodes, we chose to represent the complete set of possible paths T, but it is important to note that the amount of decision variables will scale exponentially. For larger topologies, taking a subset of *top* paths can help reduce the exponential increase in variables.

Capacity Constraint: Each edge has an assigned capacity, which cannot be overloaded. This constraint is particularly important to ensure that packets do not get dropped due to insufficient capacity. Therefore, the sum of the flows of those paths that use an edge must be less than or equal to the capacity of that edge. Let $f(t)$ denote the flow for a path t and c_e denote the capacity along edge e, then the capacity constraint is:

$$\sum_{t \in T} 1[e \in t] f(t) \leq c_e \quad \forall edges\ e$$

where $1[e \in t]$ is an indicator function that yields 1 when $e \in t$, 0 otherwise.

3.1 Throughput Formulation

The throughput of a network is the total amount of data moved per time unit. Intuitively, any efficient network should want to move as much data as possible.

Objective: The flow along each decision variable shows that amount of data moved per time unit on a specific path. Thus, the sum of the decision variables is the throughput of the network.

$$\max \sum_{t \in T} f(t)$$

Demand Constraint: The demand between two nodes is an upper bound for the sum of flows on the paths of that demand. This upper bound is important because without it, the objective would force each path to be as saturated as possible. Having this constraint ensures efficient use of the network's resources. Let $T_{i \to j}$ denote the set of paths that go from i to j.

$$\sum_{t \in T_{i \to j}} f(t) \leq demand_{i \to j}$$

3.2.1 Fairness Formulation 1

The goal of the fairness formulation is to ensure load balancing between edges. Maximum link utilization (MLU) is the maximum ratio among edges of total flow on that edge over the capacity of that edge. By minimizing the MLU, we can achieve some fairness.

Objective: The objective is to minimize the maximize link utilization and in doing so, we ensure that loads will be balanced. Let $LU(e) = \frac{\sum_{t \in T} 1[e \in t] f(t)}{c_e}$ represent the link utilization of edge e.

$$\min \ [\ \max \{LU(e) \ \ \forall edges\ e\}\ ]$$

Demand Constraint: The demand between two nodes is instead a lower bound for the sum of flows on the paths of that demand. This constraint ensures that all demand is met (note: this is only possible if the network has sufficient resources (capacity) or else, there is no optimal solution; the capacities in the Sprint topology are significantly bigger than the demands). Let $T_{i \to j}$ denote the set of paths that go from i to j.

$$\sum_{t \in T_{i \to j}} f(t) \geq demand_{i \to j}$$

3.2.2 Fairness Formulation 2

Fairness Formulation 1 works for networks with sufficient resources. For insufficiently supplied networks it might fail, so instead, we can take advantage of the objective to ensure progress.

Objective: Essentially, there are two objectives: (1) minimize the MLU and (2) send as much demand as possible. (1) is a minimization problem whereas (2) is a maximization problem so we turn (2) to minimization by adding a negative, but to ensure that they are on a similar scale, we divide the demand by the total demand of the network.

$$\min \ [\ \max \{LU(e) \ \ \forall edges\ e\} \ - \frac{\sum_{t \in T} f(t)}{\sum_{(i,j) \in \{N \times N\}} demand_{i \to j}}]$$

Note: A weight factor ϵ can be used and tuned between the two terms in the objective.

Demand Constraint: The demand between two nodes is an upper bound for the sum of flows on the paths of that demand. Having this constraint ensures efficient use of the network's resources because it means that the network will never over allocate resources.

$$\sum_{t \in T_{i \to j}} f(t) \leq demand_{i \to j}$$

4 Results and Comparison of Throughput and Fairness Formulations

4.1 Flow Allocations

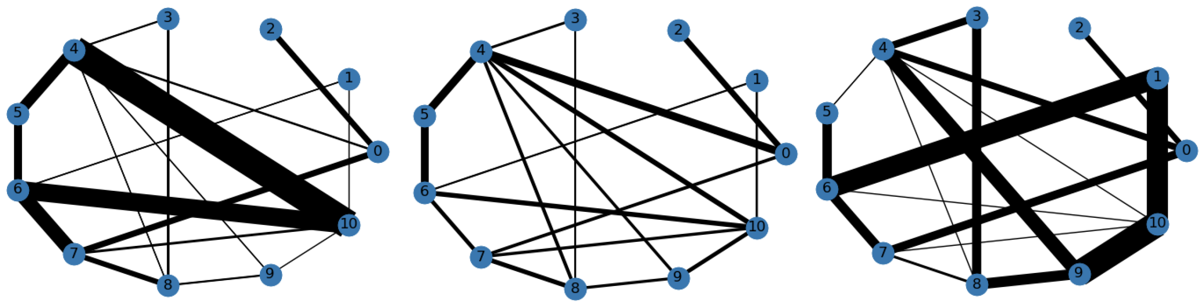

Figure 2. From Left: Throughput Formulation, Fairness Formulation 1, Fairness Formulation 2

In the above figure, the thickness of the edges are proportional to each link's utilization, with thicker edges representing higher link utilization.

4.2 Comparing Formulations

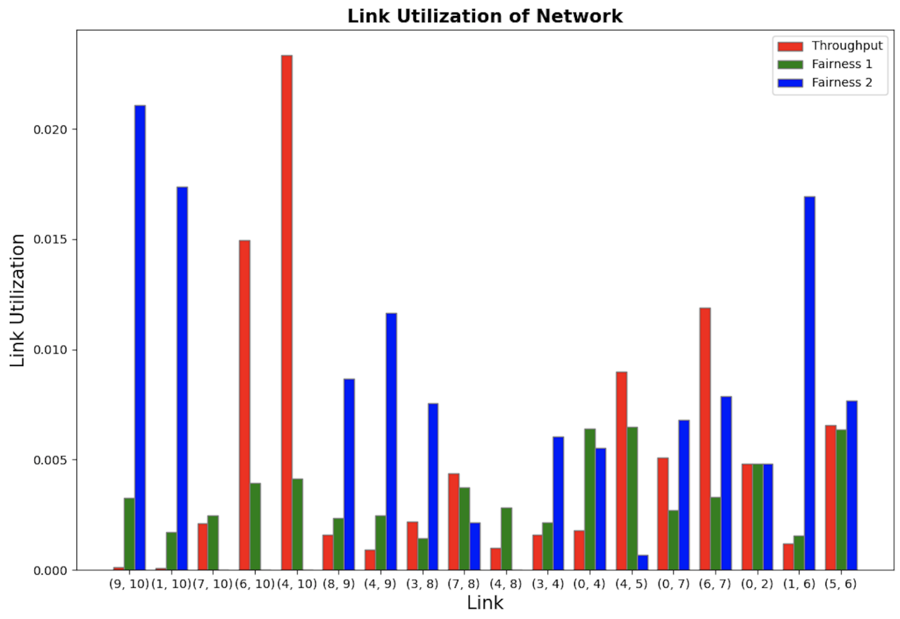

Figure 3. Bar plot comparing link utilization of the 3 formulations

Figure 3 shows the link utilizations of each edge of the network for the 3 formulations. We see that the Throughput Formulation had the highest MLU, and Fairness Formulation 1 had the lowest MLU, approximately 3.5 times lower than the Throughput Formulation's MLU. Fairness Formulation 2 had a higher MLU than Fairness Formulation 1, but lower MLU than Throughput Formulation.

For **Throughput Formulation**, link (4, 10) had the highest link utilization at 0.0233.
For **Fairness Formulation 1**, link (4, 5) had the highest link utilization at 0.0065.
For **Fairness Formulation 2**, link (9, 10) had the highest link utilization at 0.0211.
All of the formulations maintained a total throughput of 2.54e+07 Gbps.

Figure 4 shows the distribution of link utilization for each of the 3 formulations. We see that the Throughput Formulation's load distribution is highly variable, whereas Fairness Formulation 1's distribution is more balanced. As a result, the Fairness Formulations, particularly Fairness Formulation 1, are more resilient to link failures, as less traffic has to be redistributed with any one link failure. For example, if link (4, 10) in the Throughput Formulation fails, 2.33+07 Gbps of traffic needs to be redistributed, whereas any link failure for Fairness Formulation 1 results in at most 6.49+06 Gbps of traffic needing to be redistributed across the network. Additionally, in the event of a link failure and traffic redistribution, an already-heavily-loaded link in the Throughput Formulation (ie. link (4, 10) in our example above) would not have as much excess capacity to take on the additional traffic as any link in Fairness Formulation would.

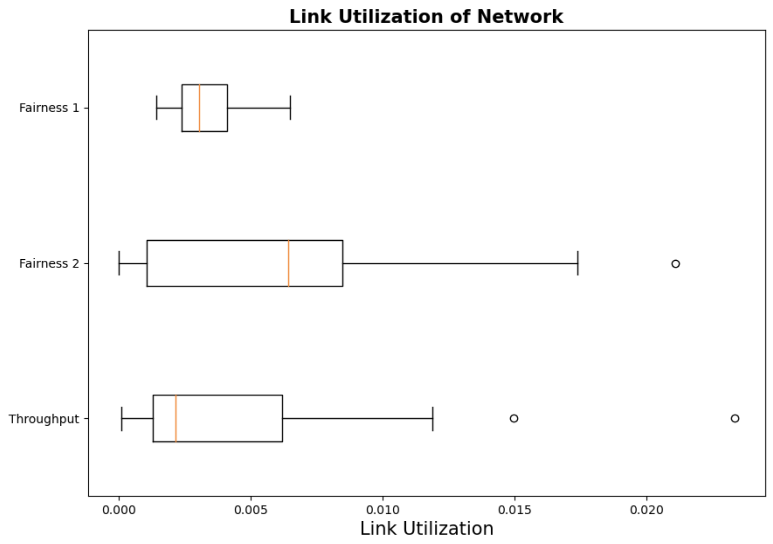

Figure 4. Box plot comparing link utilization of the 3 formulations.

Overall, the algorithms focused on minimizing the maximum link usage distribute the flow more evenly may be favorable in situations where the consequences of lost packets are high. Maximizing throughput provides the highest efficiency of the network, but link failures on high traffic edges can have very significant effects on the remaining network.

5 Scalability in Topology Size

The Gurobi runtime represents the time taken to find an optimal solution to the problem with predefined variables and constraints. A single, randomized graph topology was generated for a total of 11 different sizes (one for 5, 10, 25, 50, 100, 200, 300, 400, 500, 750, and 1000 nodes)

and the 5 shortest paths were taken for each pair of nodes. This sampling of paths was done since constructing all simple paths in a graph takes factorial time whereas getting the k shortest simple paths takes polynomial time with respect to k and the number of nodes. All three algorithms were then run on the graph to analyze how well each scaled.
As shown in Figure 5, the formulation for maximizing throughput has the highest solving time at 1232.079 seconds for a graph of 1000 nodes and 2991 edges, compared to the 100.062 seconds taken for the formulation minimizing the MLU with a different objective and 418.546 seconds for the formulation with a new constraint. One possible explanation for the relatively longer solving time for maximizing throughput may be due to caching. We solved the maximizing throughput formulation first, but since the formulations are all rather similar, each subsequent formulation could have benefited from the previous formulation. We noticed that the MLU with new constraint formulation took longer than the MLU with different objective formulation. Although this can also be explained by caching (MLU with different objective is very similar to maximizing throughput), we theorize that the MLU with a new constraint formulation is a harder problem because it has to meet all demand and ensure balance between all edges whereas the MLU with different object formulation only has to optimize the tradeoff between meeting demand and fairness. All three formulations look to be polynomial in both the number of nodes and the number of edges.

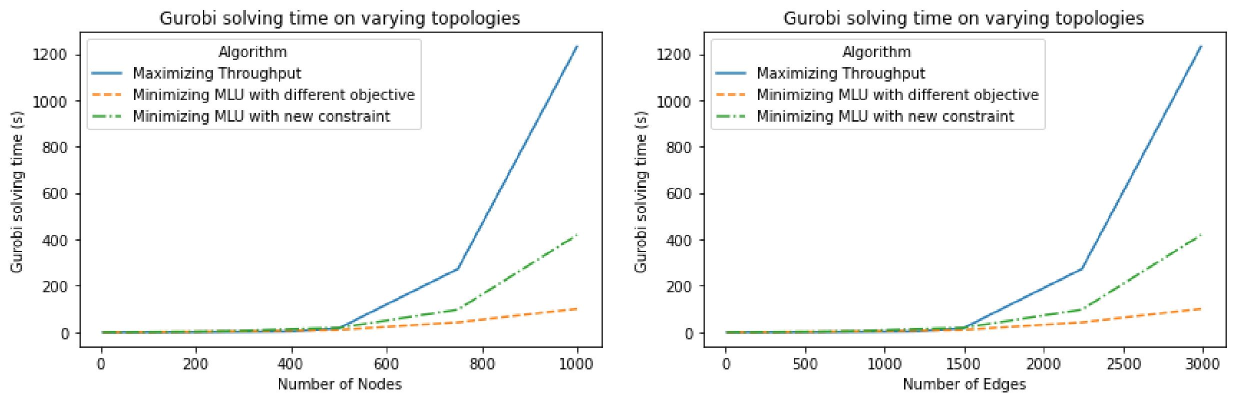

Figure 5. Relationship between the number of nodes in graph topology and solving time

6 Acknowledgements

Rachee Singh, Nikolaj Bjørner, and Umesh Krishnaswamy. 2022. Traffic Engineering: From ISP to Cloud Wide Area Networks. In The ACM SIGCOMM Symposium on SDN Research (SOSR ’22), October 19–20, 2022, Virtual Event, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3563647.3563652

Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, and C. Yang, “Traffic engineering in software-defined networking: Measurement and management,” IEEE Access, vol. 4, pp. 3246–3256, 2016.