Department of Electronic and Telecommunication Engineering
University of Moratuwa



EN 3370

Traffic Engineering

Project

Routing Path Optimization

Group Members:

Kasthuriarachchi S.N. - 160290R
Sameera K.A.D.G. - 160549V

Supervisors:

Dr. A.A. Pasqual
Dr. P.C. Weeraddana

This is submitted as a partial fulfillment for the module EN 3370: Traffic Engineering

February 28, 2021

Abstract

We present a real-time routing path modeling system that can be used in an SDN setting to gain optimum link utilization across the network while maintaining optimum Quality of Service(QoS) for different applications. Existing path routing methods mostly uses MPLS, which is static and not intelligent in terms of the overall system's traffic point of view. We solve this issue by routing the paths dynamically using an optimization algorithm to gain the maximum Return on Investment(RoI) to the network operator. We also explore QoS and ROI trade-off where we can choose a flexible operating point and still have the paths routed optimally.

Contents

1 Introduction **3**
1.1 Overview . . . 3
1.2 Problem Statement . . . 3

2 Single Variable Path Optimization Assuming Packet based Routing **5**
2.1 Method . . . 5
2.2 Problem 1: Optimizing the Return on Investment . . . 6
2.2.1 Part 1: Maximizing the Total Link Utilization . . . 6
2.2.2 Part 2: Maximizing the Minimum Link Utilization . . . 7
2.3 Problem 2: Optimizing the Quality of Service . . . 8
2.3.1 Part 1: Minimizing the Total Link Latency . . . 8
2.3.2 Part 2: Minimizing the Maximum Link Latency . . . 9
2.4 Problem 3: Optimizing the RoI and QoS . . . 10
2.4.1 Part 1: Maximizing the Total Link Utilization with Optimum Maximum Link Latency . . . 10
2.4.2 Part 2: Maximizing the Minimum Link Utilization with Optimum Maximum Link Latency . . . 11
2.4.3 Part 3: Minimizing Total Link Latency with Optimum Minimum Link Utilization . . . 12
2.4.4 Part 4: Minimizing Maximum Link Latency with Optimum Minimum Link Utilization . . . 13
2.4.5 Part 5: Multi-Objective Optimization for Trade-off between Maximum Utilization and Minimum Link Latency . . . 14
2.5 Conclusion . . . 16

3 Greedy Adaptive Floor Routing **17**
3.1 Method . . . 17
3.1.1 Finding All Paths between the Source and Destination Nodes . . . 17
3.1.2 The Greedy Adaptive Flow Routing Algorithm to Minimize Delay . . . 18
3.1.3 The Greedy Adaptive Flow Routing Algorithm to Maximize Utilization . . . 19
3.1.4 The Greedy Adaptive Flow Routing Algorithm to Minimize Delay and Maximize Utilization . . . 19

4 References **20**

5 Appendix **21**

1 Introduction

1.1 Overview

Efficiency and the Reliability of a network is crucial for a communication network. To design such a network, we should model it carefully considering all the related conditions. For that, we need to have a proper understanding about the characteristics of the network traffic. Traffic flow is the interesting quantity we model in this design. After modeling we can ensure certain optimum values for certain parameters up to some level of accuracy.

One of the primary concerns of any network operator is Return on Investment (ROI). No operator would invest on any expansion or any physical asset unless there is some link to ROI. Every operator invests on redundant links to ensure higher availability of networks as required by many Service Level Agreements (SLAs) with their key customers. While redundant links ensure high availability, it requires a significant investment just to make sure SLA requirements are met. A better way to improve the ROI is to explore the possibility of improving utilization across all links but maintain the required QoS for different applications. As of now, this is taken care of by MPLS (Multi-Path Label Switching) but it is not as dynamic as operators like it to be. The transition to Software Defined Networks opens up numerous opportunities for the operators to improve the overall efficiencies of the network. While SDN poses significant challenges to the operators from one hand, it also gives immense flexibility and power for the operator to control the network. Having a centralized view point of the network (birds eyeview) provides lot of insight into the real status of the network at any given time and one of the direct beneficiaries of this is much more efficient and dynamic traffic engineering.

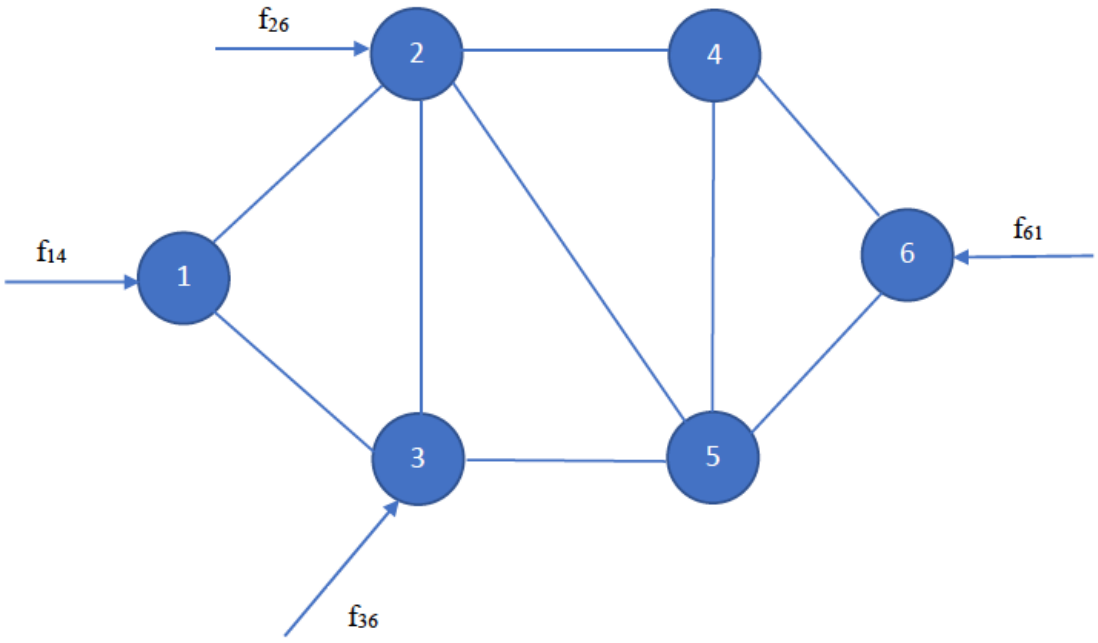
1.2 Problem Statement

This project aims to provide insights into how this path optimization can be achieved while maintaining minimum delay of links at the lowest and the minimum link utilization at the highest. In addition to that the channel bandwidth is taken as the parameter c_l and the maximum latency in each link is throttled with a parameter δ. At the end we experiment how we can trade off between these two parameters in a optimum way.

For the purpose of this project we assume a minimum of 6 node routing network (as given below) in order to provide sufficient traffic injection points to the network. Following notations will be used throughout this document.

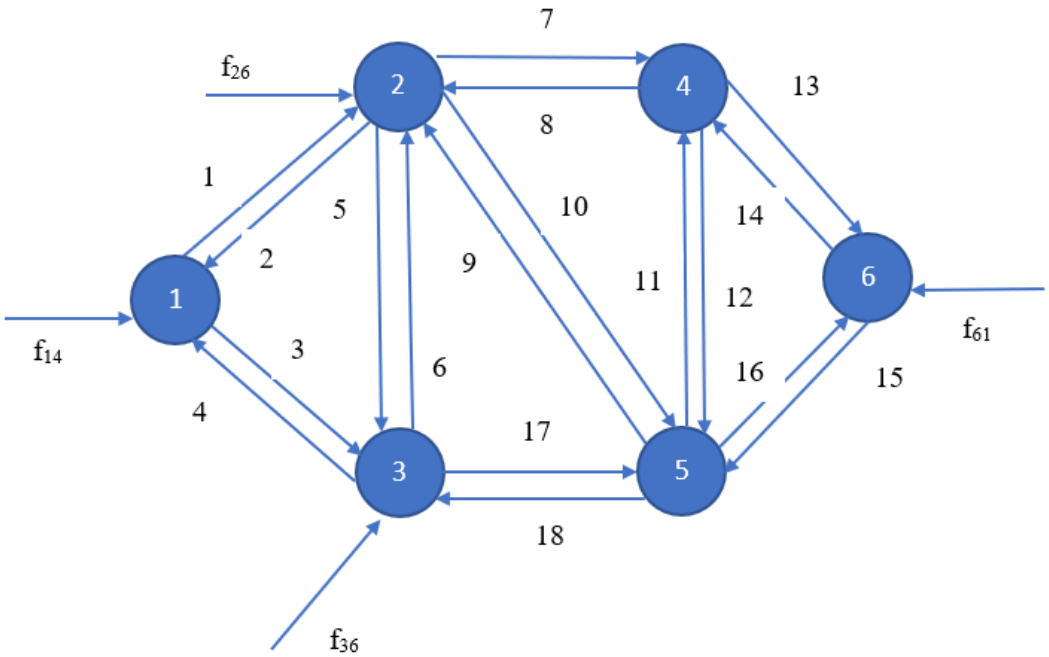
- N_i - refers to a routing node
- $R_{i,j}$ - is the bandwidth (full duplex) between nodes N_i and N_j. It is fixed by the interface speed. E.g. 10Gbps
- u_{ij} is the actual bandwidth between nodes N_i and N_j measured at time t_k–a sampling point in time.

- U_{ik} - is a traffic source connected to the node N_i
- F_{ijk} – is the network flow injected by the traffic source U_{ik} at the source node N_i and going to the destination node N_j.
- Each F_{ijk} has an associated fixed bandwidth and needs to have an end to end latency not more than δ.
- α_{ij} is the latency between the nodes N_i and N_j where the nodes i and j adjacent to each other.
- $\beta_{ij} = \Sigma\alpha_{ij}$ taken over some path – end to end latency for the flow F_{ijk}
- For the initial stage it can be assumed that each flow F_{ijk} will last for infinitely long duration.
- The flow F_{ijk} needs to be routed to satisfy the following two conditions:
  - End to end latency does not exceed δ for any flow F_{ijk}
  - Overall path utilization across all links needs to be maximized



2 Single Variable Path Optimization Assuming Packet based Routing

2.1 Method



As shown in the above figure, the network has 6 nodes labelled from 1 to 6 and 18 links (9 double-directed links) labelled from 1 to 18. The network topology of above network can be represented by a *node-link incidence matrix* A$\in \mathrm{R}^{N\times L}$, whose entry A_{nl} is associated with node n and link l via

$$A_{nl} = \begin{cases} 1, & \text{if } n \text{ is the start node of link } l \\ -1, & \text{if } n \text{ is the end node of link } l \\ 0, & \text{otherwise.} \end{cases}$$

Also, the flows with the same destination are considered as one single commodity, regardless of their sources. For each destination d we define a *source-sink vector* $s^{(d)} \in \mathrm{R}^N$, whose $n^{\text{th}}(n \neq d)$ entry $s_n^{(d)}$ denotes the non-negative amount of flow (data rate in bits/s) injected into the network at node n (the source) and destined for node d (the sink). By introducing the flow conservation law to this topology, we can write

$$s_d^{(d)} = -\sum_{n,n\neq d} s_n^{(d)} \quad , d \in \{1, 4, 6\}$$

We also define a *flow vector* $x^{(d)} \in \mathrm{R}^L$ for each destination d where, $x_l^{(d)} \geq 0$ represents the amount of flow destined for node d on each link l. The flow conservation law across the whole network can be compactly written as,

$$Ax^{(d)} = s^{(d)}, \quad d = 1, 4, 6$$

Then we can impose the capacity constraint on each link. We require the total amount of traffic on link l to be less than or equal to the capacity of the same link.

The constraints can be summarized as follows.

$$\begin{aligned} Ax^{(d)} &= s^{(d)}, \quad d = 1, 4, 6 \\ x^{(d)} &\succeq 0, \quad s^{(d)} \succeq_d 0, \quad d = 1, 4, 6 \\ \sum_d x_l^{(d)} &\leq c_l, \quad l = 1, 2, ..., 18 \end{aligned}$$

where $\succeq$ means component-wise inequality, and $\succeq_d$ means component-wise inequality except for the dth element. The total delay function can be obtained as below.

$$f_{\text{delay}}(t) = \frac{1}{\gamma} \sum_l \frac{t_l}{c_l - t_l}$$

where,

$$\gamma = \sum_{(i,j)} f_{ij}$$

The problem described in the problem statement can be divided into three main parts as discussed below. The following solutions were obtained when $c_l = 10$ Gbps $(l = 1, 2, ..., 18)$, $f_{14} = 2$ Gbps, $f_{26} = 5$ Gbps, $f_{36} = 7$ Gbps and, $f_{61} = 6$ Gbps

2.2 Problem 1: Optimizing the Return on Investment

2.2.1 Part 1: Maximizing the Total Link Utilization

The problem can be described mathematically as follows.

$$\begin{aligned} \text{maximize} \quad & \sum_{l=1}^{18} \frac{t_l}{c_l} \\ \text{subject to} \quad & Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1, 4, 6\} \\ & x^{(d)} \succeq 0, \quad d = \{1, 4, 6\} \\ & \textstyle\sum_d x_l^{(d)} \leq c_l, \quad t_l = \sum_d x_l^{(d)} \quad l = 1, 2, ..., 18 \end{aligned}$$

According to the solution, the maximum link utilization of 100% has been achieved by total of 11 links. The minimum utilization is 54.9%

Link	X1	X4	X6	Total Traffic
1	1.74	3.67	3.8	9.2
2	4.81	2.5	2.69	10
3	1.34	3.22	2.25	6.8
4	4.26	2.38	3.35	10
5	3.8	3.44	2.75	10
6	2.86	3.22	3.91	10
7	1.62	3.15	5.22	10
8	3.78	2.09	0.64	6.51
9	4.16	3.26	1.86	9.29
10	2.31	3.14	4.54	10
11	2.84	3.39	3.77	10
12	3.83	3.28	2.89	10
13	1.11	2.11	6.78	10
14	4.26	2.94	1.31	8.51
15	3.22	1.24	1.03	5.49
16	0.36	2.07	7.57	10
17	1.62	3.04	5.34	10
18	3.6	1.99	0.6	6.2

2.2.2 Part 2: Maximizing the Minimum Link Utilization

The problem can be described mathematically as follows.

$$
\begin{aligned}
\text{maximize} \quad & \min \frac{t_l}{c_l}, \quad l = 1, 2, .., 18 \\
\text{subject to} \quad & Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1, 4, 6\} \\
& x^{(d)} \succeq 0, \quad d = \{1, 4, 6\} \\
& \textstyle\sum_d x_l^{(d)} \leq c_l, \quad t_l = \sum_d x_l^{(d)} \quad l = 1, 2, ..., 18
\end{aligned}
$$

According to the solution, the maximum link utilization of 100% has been achieved by total of 2 links. The minimum utilization is 70%. Therefore, we can say that,

$$\text{Optimum minimum link utilization } = 70\%$$

Link	X1	X4	X6	Total Traffic
1	2.02	3.27	3.15	8.44
2	5.12	2.34	2.46	9.91
3	1.63	3.34	2.46	7.43
4	4.53	2.27	3.15	9.96
5	3.25	2.77	2.01	8.02
6	2.64	3.13	4.04	9.81
7	0.84	2.95	6.21	9.99
8	3.32	1.9	2.15	7.36
9	3.21	2.48	1.61	7.3
10	1.99	2.72	5.28	9.99
11	2.66	3.35	3.96	9.98
12	3.09	2.4	2.11	7.61
13	1.89	0.71	7.4	10
14	4.8	0.71	1.49	7
15	4.14	1.35	1.51	7
16	1.05	1.35	7.6	10
17	0.94	2.74	6.32	9.99
18	3.24	2.03	2.05	7.31

2.3 Problem 2: Optimizing the Quality of Service

2.3.1 Part 1: Minimizing the Total Link Latency

The problem can be described mathematically as follows.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{l=1}^{18} \frac{t_l}{c_l - t_l} \\
\text{subject to} \quad & Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1, 4, 6\} \\
& x^{(d)} \succeq 0, \quad d = \{1, 4, 6\} \\
& \textstyle\sum_d x_l^{(d)} \leq c_l, \quad t_l = \textstyle\sum_d x_l^{(d)} \quad l = 1, 2, ..., 18
\end{aligned}
$$

According to the solution, the minimum link latency is 0.01 ns bit^{-1} and, the maximum link latency is 0.08 ns bit^{-1}.

Link	X1	X4	X6	Total Traffic	Latency
1	0	2	0	2	0.01
2	3.48	0	0	3.48	0.03
3	0	0	0	0	0
4	2.52	0	0	2.52	0.02
5	0	0	0	0	0
6	0	0	1.94	1.94	0.01
7	0	1.16	4.04	5.2	0.05
8	2.52	0	0	2.52	0.02
9	0.96	0	0	0.96	0.01
10	0	0.84	2.9	3.74	0.03
11	0	0.84	1.67	2.52	0.02
12	0	0	0	0	0
13	0	0	5.72	5.72	0.07
14	2.52	0	0	2.52	0.02
15	3.48	0	0	3.48	0.03
16	0	0	6.28	6.28	0.08
17	0	0	5.06	5.06	0.05
18	2.52	0	0	2.52	0.02

2.3.2 Part 2: Minimizing the Maximum Link Latency

The problem can be described mathematically as follows.

$$
\begin{aligned}
\text{minimize} \quad & \max \frac{t_l}{c_l - t_l}, \quad l = 1, 2, ..., 18 \\
\text{subject to} \quad & Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1, 4, 6\} \\
& x^{(d)} \succeq 0, \quad d = \{1, 4, 6\} \\
& \textstyle\sum_d x_l^{(d)} \leq c_l, \quad t_l = \sum_d x_l^{(d)} \quad l = 1, 2, ..., 18
\end{aligned}
$$

According to the solution, the minimum link latency is 0.01 ns bit^{-1} and, the maximum link latency is 0.08 ns bit^{-1}. Therefore, we can say that,

$$\text{Optimum maximum link latency } = 0.08 \text{ ns bit}^{-1}$$

Link	X1	X4	X6	Total Traffic	Latency
1	0.01	1.87	0.02	1.9	0.01
2	3.84	0.01	0.05	3.91	0.03
3	0.08	0.17	0.05	0.3	0
4	2.25	0.03	0.01	2.29	0.01
5	1.61	0.01	0.03	1.66	0.01
6	0.05	0.08	2.55	2.68	0.02
7	0	0.24	3.42	3.67	0.03
8	1.55	0	0	1.55	0.01
9	3.84	0	0	3.84	0.03
10	0	1.68	4.07	5.75	0.07
11	0.01	1.77	2.6	4.38	0.04
12	2.62	0.02	0.01	2.65	0.02
13	0	0	6	6	0.08
14	4.16	0	0	4.16	0.04
15	1.84	0	0	1.84	0.01
16	0	0	6	6	0.07
17	0	0.07	4.51	4.59	0.04
18	0.61	0	0	0.61	0

2.4 Problem 3: Optimizing the RoI and QoS

2.4.1 Part 1: Maximizing the Total Link Utilization with Optimum Maximum Link Latency

The problem can be described mathematically as follows.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{l=1}^{18} \frac{t_l}{c_l} \\
\text{subject to} \quad & Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1, 4, 6\} \\
& x^{(d)} \succeq 0, \quad d = \{1, 4, 6\} \\
& \textstyle\sum_d x_l^{(d)} \leq c_l, \quad l = 1, 2, ..., 18 \\
& \frac{t_l}{c_l - t_l} \leq 1.86, \quad t_l = \sum_d x_l^{(d)} \quad l = 1, 2, ..., 18
\end{aligned}
$$

According to the solution,

$$
\begin{aligned}
\text{minimum link utilization} &= 17.1\% \\
\text{maximum link utilization} &= 60\% \\
\text{minimum link latency} &= 0.01 \text{ ns bit}^{-1} \\
\text{maximum link latency} &= 0.08 \text{ ns bit}^{-1}
\end{aligned}
$$

Link	X1	X4	X6	Total Traffic	Latency
1	1.02	2.27	2.22	5.51	0.06
2	4.13	0.85	1.02	6	0.08
3	0.1	1.62	0.77	2.49	0.02
4	2.98	1.04	1.98	6	0.07
5	3.21	1.86	0.93	6	0.07
6	0.79	2.14	3.07	6	0.07
7	0.31	1.06	4.62	6	0.07
8	1.26	0.21	0.24	1.71	0.01
9	4.78	0.39	0.61	5.78	0.07
10	0.19	1.23	4.58	6	0.08
11	1.2	2.28	2.52	6	0.07
12	3.97	1.13	0.9	6	0.07
13	0	0	6	6	0.08
14	3.71	0	0	3.71	0.03
15	2.29	0	0	2.29	0.01
16	0	0	6	6	0.07
17	0.24	0.82	4.94	6	0.07
18	0.7	0.52	1.29	2.51	0.02

2.4.2 Part 2: Maximizing the Minimum Link Utilization with Optimum Maximum Link Latency

The problem can be described mathematically as follows.

$$
\begin{aligned}
\text{maximize} \quad & \min \frac{t_l}{c_l}, \quad l = 1, 2, .., 18 \\
\text{subject to} \quad & Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1, 4, 6\} \\
& x^{(d)} \succeq 0, \quad d = \{1, 4, 6\} \\
& \textstyle\sum_d x_l^{(d)} \leq c_l, \quad l = 1, 2, ..., 18 \\
& \frac{t_l}{c_l - t_l} \leq 1.86, \quad t_l = \sum_d x_l^{(d)} \quad l = 1, 2, ..., 18
\end{aligned}
$$

According to the solution,

$$
\begin{aligned}
\text{minimum link utilization} &= 30\% \\
\text{maximum link utilization} &= 60\% \\
\text{minimum link latency} &= 0.02 \text{ ns bit}^{-1} \\
\text{maximum link latency} &= 0.08 \text{ ns bit}^{-1}
\end{aligned}
$$

Link	X1	X4	X6	Total Traffic	Latency
1	0.43	1.32	1.32	3.07	0.02
2	3.71	0.59	0.58	4.88	0.05
3	0.34	1.95	0.87	3.16	0.02
4	3.07	0.68	1.61	5.35	0.06
5	1.63	0.96	0.5	3.1	0.02
6	1.03	1.65	2.64	5.32	0.06
7	0.11	1.31	4.43	5.86	0.07
8	2.56	0.31	0.26	3.13	0.02
9	1.63	1.1	0.39	3.12	0.02
10	0.2	1.52	4.08	5.8	0.07
11	1.02	1.91	2.46	5.4	0.06
12	1.57	0.92	0.64	3.13	0.02
13	0	0	6	6	0.08
14	3	0	0	3	0.02
15	3	0	0	3	0.02
16	0	0	6	6	0.08
17	0.03	1.08	4.64	5.76	0.07
18	2.15	0.5	0.51	3.16	0.02

2.4.3 Part 3: Minimizing Total Link Latency with Optimum Minimum Link Utilization

The problem can be described mathematically as follows.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{l=1}^{18} \frac{t_l}{c_l - t_l} \\
\text{subject to} \quad & Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1, 4, 6\} \\
& x^{(d)} \succeq 0, \quad d = \{1, 4, 6\} \\
& \textstyle\sum_d x_l^{(d)} \leq c_l, \quad l = 1, 2, ..., 18 \\
& \frac{t_l}{c_l} \geq 0.7, \quad t_l = \sum_d x_l^{(d)} \quad l = 1, 2, ..., 18
\end{aligned}
$$

According to the solution,

$$
\begin{aligned}
\text{minimum link utilization} &= 69.7\% \\
\text{maximum link utilization} &= 99.9\% \\
\text{minimum link latency} &= 0.11 \text{ ns bit}^{-1} \\
\text{maximum link latency} &= 42.77 \text{ ns bit}^{-1}
\end{aligned}
$$

Link	X1	X4	X6	Total Traffic	Latency
1	2.2	2.15	2.61	6.97	0.11
2	5.26	1.91	1.45	8.62	0.31
3	1.23	3.57	2.22	7.01	0.12
4	4.15	1.82	3.16	9.13	0.53
5	3.63	2.61	0.8	7.04	0.12
6	1.8	2.82	4.63	9.25	0.61
7	0.02	0.04	9.7	9.77	2.12
8	0.06	0.04	6.97	7.07	0.12
9	4.94	2.04	0.08	7.06	0.12
10	0.09	2.61	7.28	9.99	42.77
11	1.63	0.15	7.62	9.4	0.78
12	1.61	4.99	0.4	6.99	0.12
13	0	0.01	9.9	9.9	4.72
14	0.07	6.87	0.05	6.98	0.12
15	6.22	0.72	0.05	6.98	0.12
16	0.11	7.52	2.26	9.89	4.36
17	0.01	1.62	8.1	9.73	1.77
18	1.16	0.07	5.89	7.11	0.12

2.4.4 Part 4: Minimizing Maximum Link Latency with Optimum Minimum Link Utilization

The problem can be described mathematically as follows.

$$
\begin{aligned}
\text{minimize} \quad & \max \frac{t_l}{c_l - t_l}, \quad l = 1, 2, ..., 18 \\
\text{subject to} \quad & Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1, 4, 6\} \\
& x^{(d)} \succeq 0, \quad d = \{1, 4, 6\} \\
& \textstyle\sum_d x_l^{(d)} \leq c_l, \quad l = 1, 2, ..., 18 \\
& \frac{t_l}{c_l} \geq 0.7, \quad t_l = \sum_d x_l^{(d)} \quad l = 1, 2, ..., 18
\end{aligned}
$$

According to the solution,

$$
\begin{aligned}
\text{minimum link utilization} &= 68.8\% \\
\text{maximum link utilization} &= 99.8\% \\
\text{minimum link latency} &= 0.11 \text{ ns bit}^{-1} \\
\text{maximum link latency} &= 33.06 \text{ ns bit}^{-1}
\end{aligned}
$$

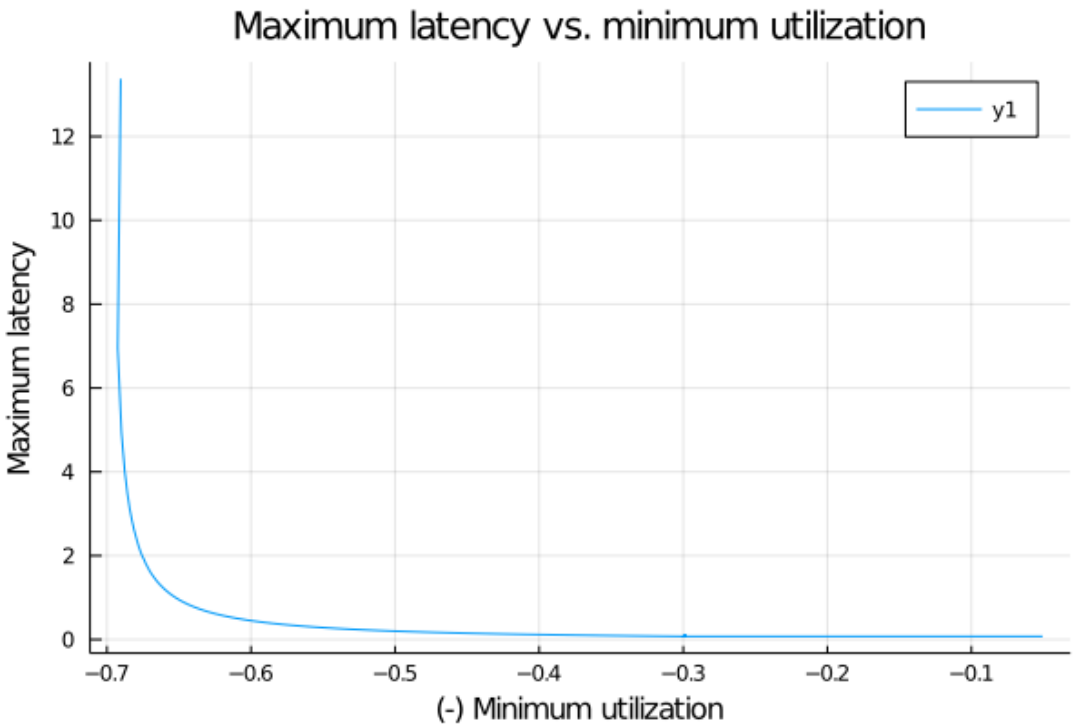
Link	X1	X4	X6	Total Traffic	Latency
1	3.21	4.23	0.85	8.29	0.24
2	5.82	3.32	0.71	9.85	3.3
3	0.67	3.01	3.46	7.14	0.13
4	4.13	2	3.68	9.81	2.56
5	2.21	3.03	2.26	7.5	0.15
6	1.52	3.08	4.84	9.45	0.85
7	1.26	2.8	5.83	9.89	4.46
8	4.44	2.01	0.82	7.27	0.13
9	3.98	2.45	0.75	7.18	0.13
10	3.82	2.6	3.45	9.88	4
11	2.08	3.9	3.59	9.57	1.12
12	1.7	2.92	2.7	7.32	0.14
13	0.17	2.51	7.31	9.98	33.06
14	2.96	2.72	1.38	7.05	0.12
15	4.37	1.12	1.39	6.88	0.11
16	1.15	1.35	7.45	9.96	11.72
17	2.38	2.48	5.03	9.89	4.38
18	5.1	1.44	0.77	7.31	0.14

2.4.5 Part 5: Multi-Objective Optimization for Trade-off between Maximum Utilization and Minimum Link Latency

The problem can be described mathematically as follows.

$$
\begin{aligned}
&\text{minimize} && \lambda f_{delay}(x,t) + (1-\lambda)\, g_u(x,t) \\
&\text{subject to} && Ax^{(d)} = s^{(d)}, \quad \text{d} \in \{1,4,6\} \\
& && x^{(d)} \succeq 0, \quad d = \{1,4,6\} \\
& && t_l = \textstyle\sum_d x_l^{(d)} \leq c_l, \quad l = 1,2,...,18 \\
&\text{where} && f_{delay}(x,t) = \max \frac{t_l}{c_l - t_l}, \quad l = 1,2,...,18 \\
& && g_u(x,t) = -\min\left(\tfrac{t_l}{c_l}\right), \quad l = 1,2,...,18
\end{aligned}
$$

The following graph depicts the solution curve for the above optimization problem. We have plotted the graph by iterating from $\lambda = 0$ to $\lambda = 1$ with a step size of 0.001 .


Maximum latency vs. minimum utilization
y1
Maximum latency
0
2
4
6
8
10
12
−0.7
−0.6
−0.5
−0.4
−0.3
−0.2
−0.1
(-) Minimum utilization

2.5 Conclusion

As mentioned in the problem statement, we need to maximize the overall link utilization while keeping the end-to-end link latency below some value. We achieve this by decomposing the above problem into two separate problems and, calculating the optimum values for each of them. In order to maximize the link utilization, we solve the maximin optimization problem by maximizing the minimum link utilization.

$$\therefore \text{Optimum minimum link utilization } = 70\%$$

This makes sure that link utilization will not go under 70% for any link. Similarly, we solve the minimax optimization problem by minimizing the maximum link latency to identify an upper bound for the end-to-end link latency.

$$\therefore \text{Optimum maximum link latency } = 0.08 \text{ ns bit}^{-1}$$

This ensures that the end-to-end link latency for any link will not go above 0.08 ns bit^{-1}. In third problem we have tried to come to a balance between the RoI and the QoS. At there we have solved the following optimization problems,

1. Optimizing the RoI with the optimum maximum latency.
2. Optimizing the QoS with the optimum minimum utilization.

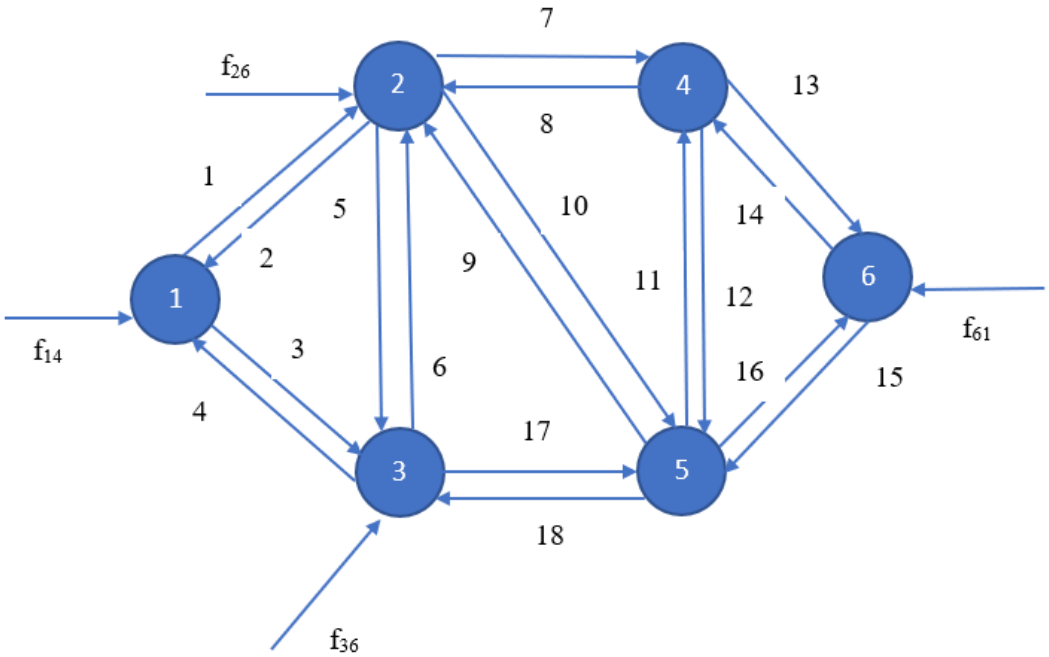
In the first case, we were able to achieve a lower bound of 30% and an upper bound of 60% for the link utilization while keeping the upper bound for the link latency at 0.08 ns bit^{-1} but, in the second case even though the minimum link utilization has been increased to around 70%, the end-to-end link latency can go as high as 43 ns bit^{-1}. Therefore, it is clear that there is a trade-off between optimizing the link utilization and the end-to-end latency. This happens because,

$$\begin{aligned} \text{Link Utilization}(U) &= \frac{t_l}{c_l} \\ \text{Link Latency} &= \frac{1}{\gamma}\frac{t_l}{c_l - t_l} \\ &= \frac{1}{\gamma}\frac{1}{\left(\frac{1}{U} - 1\right)} \end{aligned}$$

Therefore, when the link utilization(U) increases the denominator decreases resulting in increased latency. The same concept has been visualized by the graph at the end of section 2.4.5.

3 Greedy Adaptive Floor Routing

3.1 Method



3.1.1 Finding All Paths between the Source and Destination Nodes

Algorithm 1: Finding all paths between the source and destination nodes in the controller (Findpaths)

Input: ID_{src}, ID_{dst}, Connectivity Matrix
Output: all existing paths between ID_{src} and ID_{dst}

Define path as a list of nodes which make a path between the source and destination nodes;
Define paths as a list of paths which contains all existing paths between source and destination;
Select the row ID_{src} from connectivity to create a list of nodes which are connected to the source node;
Put 0 at the column of source ID in connectivity to prevent from the infinite loop;
for *i the number of directly connected nodes* **do**
    Add ID_{src} to the path; **if** ID_i *is equal to* ID_{dst} **then**
        Add ID_i to the path and capsulate path; Add the path to the list of paths;
    else
        Add ID_i to the path; Go to the Findpaths function with ID_{src}, ID_{dst}, Connectivity Matrix;
    end
end

After finding the paths between two nodes requested by the flow, there is a request to select the optimum path based on performance metrics. The greedy algorithm tries to select the best path in every step. Hence, it sorts F as biggest D^f towards the smallest one as stated in Equation (1). Then it tries to allocate the paths to f. The paths sorting is shown in Equation (2), where k is the number of paths.

$$F = \{f_0, f_1 ... f_{m-1}\},\ D^f_{m-1} < D^f_{m-2} < ... < D^f_1 < D^f_0 \text{ and m is the number of flows} \quad (1)$$

$$Paths = \{p_0, p_1 ... p_{k-1}\},\ D^{m-1} < Delay_{path(k-1)} < ... < Delay_{path(1)} < Delay_{path(0)} (2)$$

The paths are sorted by descending from the biggest to largest one from the point of delay. This sorting covers more paths to be checked for the flow and increase the BP. Algorithm 2 shows the pseudocode of the proposed algorithm.

3.1.2 The Greedy Adaptive Flow Routing Algorithm to Minimize Delay

Algorithm 2: The greedy adaptive flow routing algorithm for Delay

Input: F as the requested flows set
Output: a path or null for each flow in F

```
Sort F based on D^f descending;
for each flow in F do
    Find all existing paths between current flow source and destination with
     Findpaths function;
    Sort the paths of the current flow descending according to delay(or hop
     count); for each path in paths do
        if find the path which support the D^f_{src, dst} requirement and for all links in
         path flow rate < bandwidth then
            Allocate the current flow to the current path and return the current
             path Goto the next flow if exists
        else
        end
        Goto another path if exists, otherwise return the null value for the current
         flow
    end
end
```

3.1.3 The Greedy Adaptive Flow Routing Algorithm to Maximize Utilization

Algorithm 3: The Greedy Adaptive Flow Routing Algorithm to Maximize Utilization

Input: F as the requested flows set
Output: a path or null for each flow in F

```
Sort F based on U^f ascending;
for each flow in F do
    Find all existing paths between current flow source and destination with
     Findpaths function;
    Sort the paths of the current flow ascending from smallest utilization to
     largest utilization;
    Allocate the current flow to the lowest utilized path and return the current
     path Goto the next flow if exists;
end
```

3.1.4 The Greedy Adaptive Flow Routing Algorithm to Minimize Delay and Maximize Utilization

$$DPP = 1 - UPP \tag{3}$$

where,
DPP : Delay Priority Percentage
UPP : Utilization Priority Percentage

Algorithm 4: The Greedy Adaptive Flow Routing Algorithm to Minimize Delay and Maximize Utilization

Input: F as the requested flows set, DPP set for each flow
Output: a path or null for each flow in F

```
Sort F based on D^f descending;
for each flow in F do
    Find all existing paths between current flow source and destination with
     Findpaths function;
    Sort the paths of the current flow ascending according to the Utilization of
     the paths;
    Sort the paths of the current flow descending according to the delay of the
     paths;
    Filter the list of paths which satisfies the D^f_{src,dst};
    Allocate the current flow to the DPP^th percentile path and return the current
     path Goto the next flow if exists;
end
```

4 References

1. Xiao, L., Johansson, M., Boyd, S. P. (2004). Simultaneous Routing and Resource Allocation Via Dual Decomposition. IEEE Transactions on Communications, 52(7), 1136–1144. https://doi.org/10.1109/tcomm.2004.831346

2. Bertsekas, D. P., G. (1992). Data Networks (2nd Edition) (2nd ed.). Pearson.

3. `https://convexjl.readthedocs.io/en/latest/`

4. Shirmarz, A., Ghaffari, A. (2019). An adaptive greedy flow routing algorithm for performance improvement in software-defined network. International Journal Of Numerical Modelling: Electronic Networks, Devices And Fields, 33(1). doi: 10.1002/jnm.2676

5 Appendix

```
using Convex,LinearAlgebra,Pkg, Plots
using SCS
## Defining constants

A=[1 -1 1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
-1 1 0 0 1 -1 1 -1 -1 1 0 0 0 0 0 0 0 0;
0 0 -1 1 -1 1 0 0 0 0 0 0 0 0 0 0 1 -1;
0 0 0 0 0 0 -1 1 0 0 -1 1 1 -1 0 0 0 0;
0 0 0 0 0 0 0 0 1 -1 1 -1 0 0 -1 1 -1 1;
0 0 0 0 0 0 0 0 0 0 0 0 -1 1 1 -1 0 0]

f14 = 2;
f26 = 5;
f36 = 7;
f61 = 6;

gamma = f14+f26+f36+f61;

s1 = [-f61;0;0;0;0;f61];
s4 = [f14;0;0;-f14;0;0];
s6 = [0;f26;f36;0;0;-f26-f36]

solver = () -> SCS.Optimizer(verbose=0);
## Problem 1

####part 1
####maximizing the total link utilization

x1 = Variable(18,1);
x4 = Variable(18,1);
x6 = Variable(18,1);

t = x1 + x4 + x6;

problem1 = maximize(sum(t/10))
problem1.constraints += [
    A * x1 == s1,
    A * x4 == s4,
    A * x6 == s6,
    x1 >= 0,
    x4 >= 0,
    x6 >= 0,
    t <= 10,
```

```
    t>=0
    ];

solve!(problem1, solver)

println("")
println("Maximizing the total link utilization")
println(problem1.optval)
println(round.(evaluate(x1),digits=2))
println(round.(evaluate(x4),digits=2))
println(round.(evaluate(x6),digits=2))
println(round.(evaluate(t),digits=2))

####part 2
####maximizing the minimum link utilization

x1 = Variable(18,1);
x4 = Variable(18,1);
x6 = Variable(18,1);

t = x1 + x4 + x6;
problem2 = maximize(minimum(t/10))

problem2.constraints += [
    A * x1 == s1,
    A * x4 == s4,
    A * x6 == s6,
    x1 >= 0,
    x4 >= 0,
    x6 >= 0,
    t <= 10,
    t>=0
    ];

solve!(problem2, solver)

println("")
println("Maximizing the minimum link utilization")
println(problem2.optval)
println(round.(evaluate(x1),digits=2))
println(round.(evaluate(x4),digits=2))
println(round.(evaluate(x6),digits=2))
println(round.(evaluate(t),digits=2))

optimum_utilization = problem2.optval
```

```
## Problem 2

####part 1
####minimize the total delay

x1 = Variable(18,1);
x4 = Variable(18,1);
x6 = Variable(18,1);

t = x1 + x4 + x6;

problem3 = minimize(sum(dot(/)(10/gamma,10-t))-18/gamma)
problem3.constraints += [
    A * x1 == s1,
    A * x4 == s4,
    A * x6 == s6,
    x1 >= 0,
    x4 >= 0,
    x6 >= 0,
    t <= 10,
    t >=0
    ];

solve!(problem3, solver)

println("")
println("Minimize the total delay")
println(problem3.optval)
println(round.(evaluate(x1),digits=2))
println(round.(evaluate(x4),digits=2))
println(round.(evaluate(x6),digits=2))
println(round.(evaluate(t),digits=2))
println("Link Delays =")
println(round.(dot(/)(evaluate(t)/gamma,10 .-evaluate(t)), digits=2))

####part 2
####minimizing the maximum delay

x1 = Variable(18,1);
x4 = Variable(18,1);
x6 = Variable(18,1);

t = x1 + x4 + x6;
```

```
problem4 = minimize(maximum(dot(/)(10/gamma,10-t)-1/gamma))
problem4.constraints += [
    A * x1 == s1,
    A * x4 == s4,
    A * x6 == s6,
    x1 >= 0,
    x4 >= 0,
    x6 >= 0,
    t <= 10,
    t >=0
    ];

solve!(problem4, solver)

println("")
println("Minimizing the maximum delay")
println(problem4.optval)
println(round.(evaluate(x1),digits=2))
println(round.(evaluate(x4),digits=2))
println(round.(evaluate(x6),digits=2))
println(round.(evaluate(t),digits=2))
println("Link Delays =")
println(round.(dot(/)(evaluate(t)/gamma,10 .-evaluate(t)), digits=2))

optimum_delay = problem4.optval

## Problem 3

####part 1
####Maximizing the total utilization with the optimum latency

x1 = Variable(18,1);
x4 = Variable(18,1);
x6 = Variable(18,1);

t = x1 + x4 + x6;

problem5 = maximize(sum(t/10))
problem5.constraints += [
    A * x1 == s1,
    A * x4 == s4,
    A * x6 == s6,
    x1 >= 0,
    x4 >= 0,
    x6 >= 0,
```

```
    t>=0,
    t <= 10,
    dot(/)(10/gamma,10-t)-1/gamma <=optimum_delay
    ];

solve!(problem5, solver)

println("")
println("Maximizing the total utilization with the optimum latency")
println(problem5.optval)
println(round.(evaluate(x1),digits=2))
println(round.(evaluate(x4),digits=2))
println(round.(evaluate(x6),digits=2))
println(round.(evaluate(t),digits=2))
println("Link Delays =")
println(round.(dot(/)(evaluate(t)/gamma,10 .-evaluate(t)), digits=2))

####part 2
####Maximizing the minimum utilization with the optimum latency

x1 = Variable(18,1);
x4 = Variable(18,1);
x6 = Variable(18,1);

t = x1 + x4 + x6;

problem6 = maximize(minimum(t/10))
problem6.constraints += [
    A * x1 == s1,
    A * x4 == s4,
    A * x6 == s6,
    x1 >= 0,
    x4 >= 0,
    x6 >= 0,
    t>=0,
    t <= 10,
    dot(/)(10/gamma,10-t)-1/gamma <=optimum_delay
    ];

solve!(problem6, solver)

println("")
println("Maximizing the minimum utilization with the optimum latency")
println(problem6.optval)
println(round.(evaluate(x1),digits=2))
```

```
println(round.(evaluate(x4),digits=2))
println(round.(evaluate(x6),digits=2))
println(round.(evaluate(t),digits=2))
println("Link Delays =")
println(round.(dot(/)(evaluate(t)/gamma,10 .-evaluate(t)), digits=2))

####part 3
####Minimizing the total delay with the optimum utilization

x1 = Variable(18,1);
x4 = Variable(18,1);
x6 = Variable(18,1);

t = x1 + x4 + x6;

problem7 = minimize(sum(dot(/)(10/gamma,10-t))-18/gamma)
problem7.constraints += [
    A * x1 == s1,
    A * x4 == s4,
    A * x6 == s6,
    x1 >= 0,
    x4 >= 0,
    x6 >= 0,
    t>=0,
    t>=10*optimum_utilization,
    t<= 10
    ]

solve!(problem7, solver)

println("")
println("Minimizing the total delay with the optimum utilization")
println(problem7.optval)
println(round.(evaluate(x1),digits=2))
println(round.(evaluate(x4),digits=2))
println(round.(evaluate(x6),digits=2))
println(round.(evaluate(t),digits=2))
println("Link Delays =")
println(round.(dot(/)(evaluate(t)/gamma,10 .-evaluate(t)), digits=2))

####part 4
####Minimizing the maximum delay with the optimum utilization

x1 = Variable(18,1);
x4 = Variable(18,1);
```

```
x6 = Variable(18,1);

t = x1 + x4 + x6;

problem8 = minimize(maximum(dot(/)(10/gamma,10-t)-1/gamma))
problem8.constraints += [
    A * x1 == s1,
    A * x4 == s4,
    A * x6 == s6,
    x1 >= 0,
    x4 >= 0,
    x6 >= 0,
    t >= 10*optimum_utilization,
    t<= 10
];

solve!(problem8, solver)

println("")
println("Minimizing the maximum delay with the optimum utilization")
println(problem8.optval)
println(round.(evaluate(x1),digits=2))
println(round.(evaluate(x4),digits=2))
println(round.(evaluate(x6),digits=2))
println(round.(evaluate(t),digits=2))
println("Link Delays =")
println(round.(dot(/)(evaluate(t)/gamma,10 .-evaluate(t)), digits=2))

## Graph

f=[]
g=[]

# increse step for a smoother graph
for lambda = 0:0.001:1

    y1 = Variable(18,1);
    y4 = Variable(18,1);
    y6 = Variable(18,1);

    u = y1 + y4 + y6;

    obj_func = minimize(lambda*(maximum(dot(/)(10/gamma,10-u)-1/gamma))
     + (1-lambda)*(-minimum(u/10)))
    obj_func.constraints += [
```

```
        A*y1 == s1,
        A*y4 == s4,
        A*y6 == s6,
        y1 >= 0,
        y4 >= 0,
        y6 >= 0,
        u<= 10
        ];

    solve!(obj_func, solver)
    push!(f,evaluate(maximum(dot(/)(10/gamma,10-u)-1/gamma)))
    push!(g,evaluate(-minimum(u/10)))
end

plot(g,f, title = "Maximum latency vs. minimum utilization")
xlabel!("(-) Minimum utilization")
ylabel!("Maximum latency")
```