Bottleneck Services with ALTO

Kai Gao
ALTO WG

Outline

- Basic concept of bottleneck
- Use cases for bottleneck services
- G2 optimization framework
- Bottleneck services with ALTO

Bottleneck Information as a Service

Many networks use **dynamic resource allocation to improve network utility**

- TCP congestion control
- Infrastructure resource management (e.g., Google BwE)

Bottleneck information is important for applications to **predict network performance and get traffic optimization guidance** in such networks

- Throughput prediction
- Time-bounded flow optimization
- Network Planning

Bottleneck service can be an important motivating use case of ALTO

- G2 system by Reservoir Labs
- Potential deployment: networks for super computing, SD-WAN, cloud/edge platforms

Bottlenecks in Networks with Dynamic Resource Allocation

Many dynamic bandwidth allocation is based on optimization problems

Bottleneck for Max-Min Fairness: A link l is a bottleneck of a flow f if and only if $f \in F_l$, $\sum_{f \in F_l} \tilde{x}_f = c_l$ and $\tilde{x}_f \geq \tilde{x}_j, \forall j \in F_l$.

Bottleneck: A link l is a bottleneck of a flow f if and only if $f \in F_l$, $\sum_{f \in F_l} \tilde{x}_f = c_l$ and $\frac{\partial \tilde{x}_f}{\partial c_l}^- > 0$, i.e., the rate of flow f will decrease if the capacity of l decreases.

Example: max-min fairness

$$\{\tilde{x}_f\} = \operatorname{argmax}\{\min\{\tilde{x}_f\}\}$$

subject to:

$$\forall l \in L, \sum_{f \in F_l} x_f \leq c_l$$

Example: unconstrained NUM problem

$$\{\tilde{x}_f\} = \operatorname{argmax} \sum_{f \in F} U(x_f)$$

subject to:

$$\forall l \in L, \sum_{f \in F_l} x_f \leq c_l$$

Example of Bottleneck

Resource allocation: max-min fairness

2 links:

l_1: $A \to B$, capacity: 3

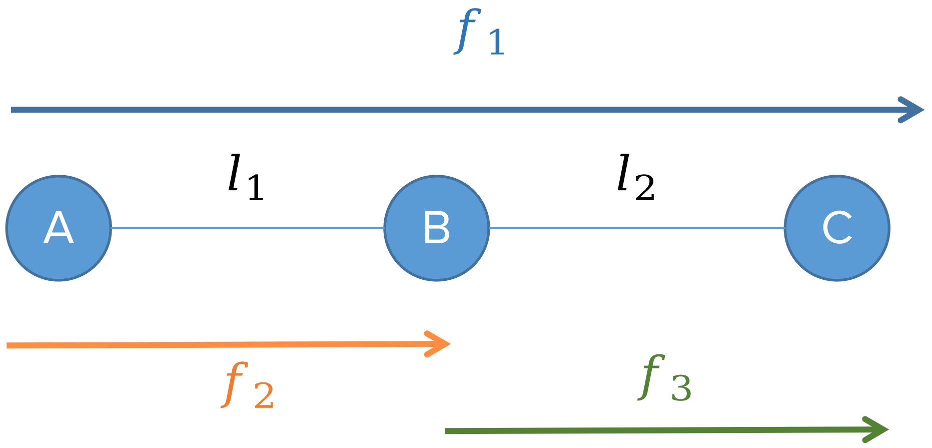
l_2: $B \to C$, capacity: 2

3 flows:

f_1: $A \to C$, rate: 1, $\frac{\partial \tilde{x}_1}{\partial c_1} = 0$, $\frac{\partial \tilde{x}_1}{\partial c_2} = 0.5$

f_2: $A \to B$, rate: 2, $\frac{\partial \tilde{x}_2}{\partial c_1} = 1$

f_3: $B \to C$, rate: 1, $\frac{\partial \tilde{x}_3}{\partial c_2} = 0.5$



Bottleneck of f_1: l_2

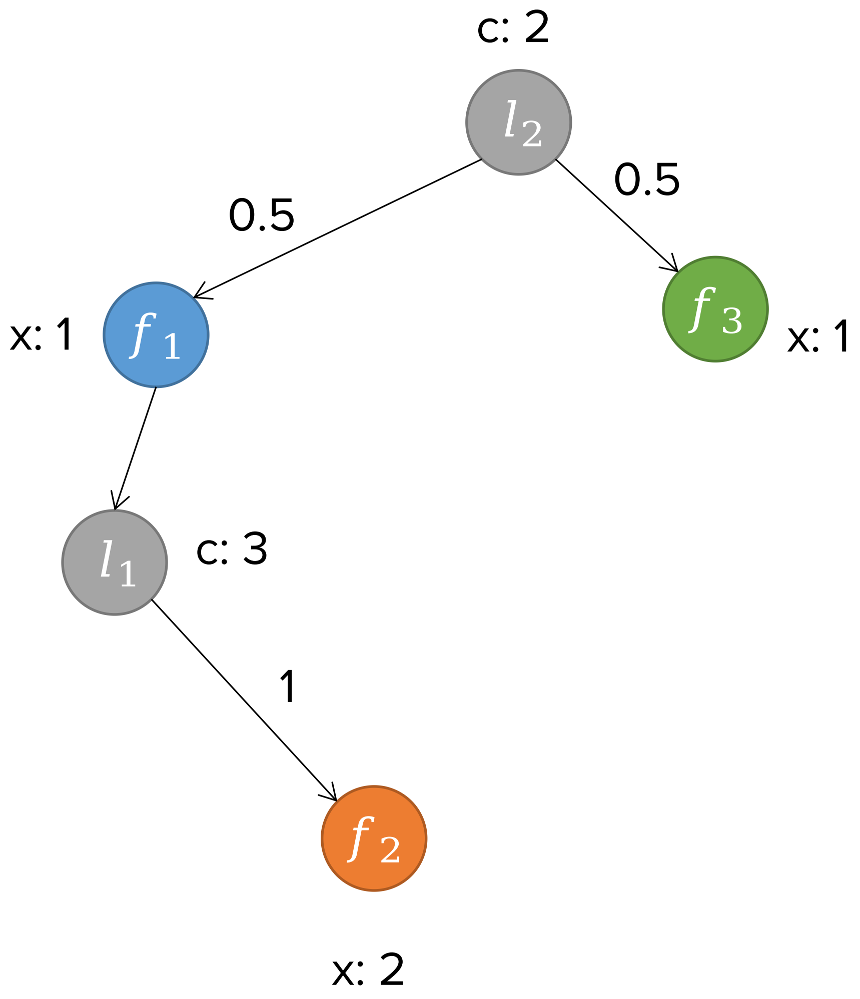
Bottleneck of f_2: l_1

Bottleneck of f_3: l_2

Bottleneck Structure

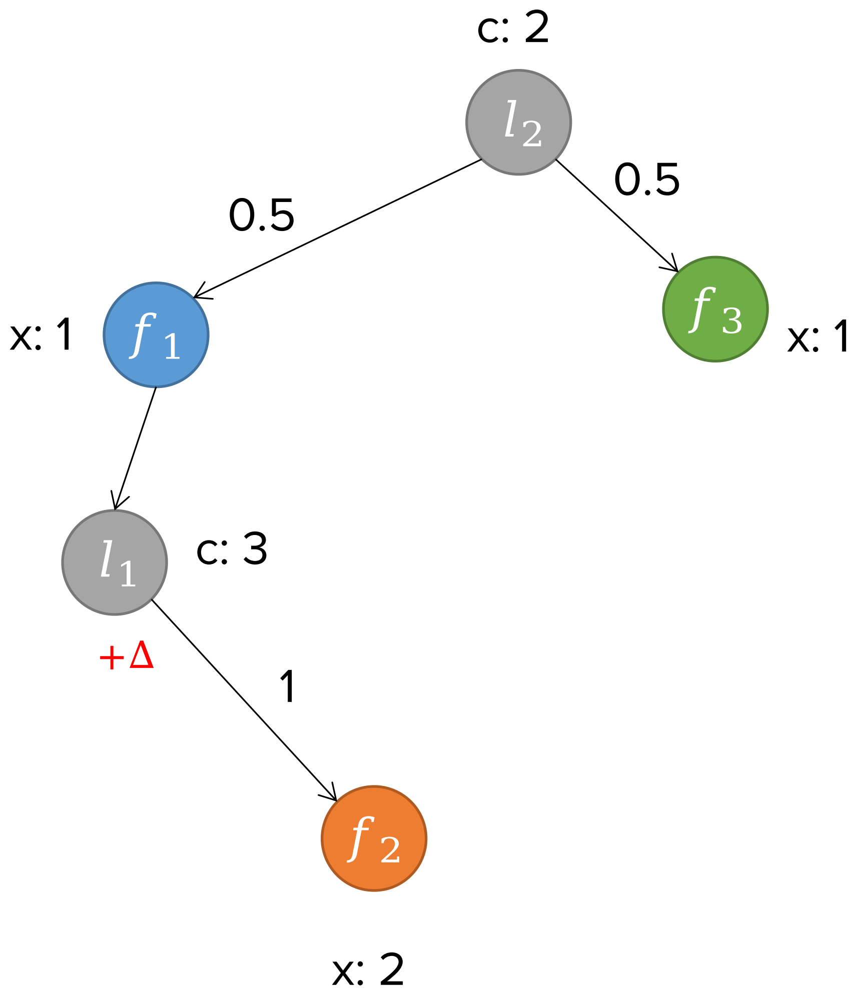
In networks with dynamic resource allocation, non-traversed links may also have an impact on the flow rate

Bottleneck structure enables quantative analysis of bottleneck links or flows over other flows



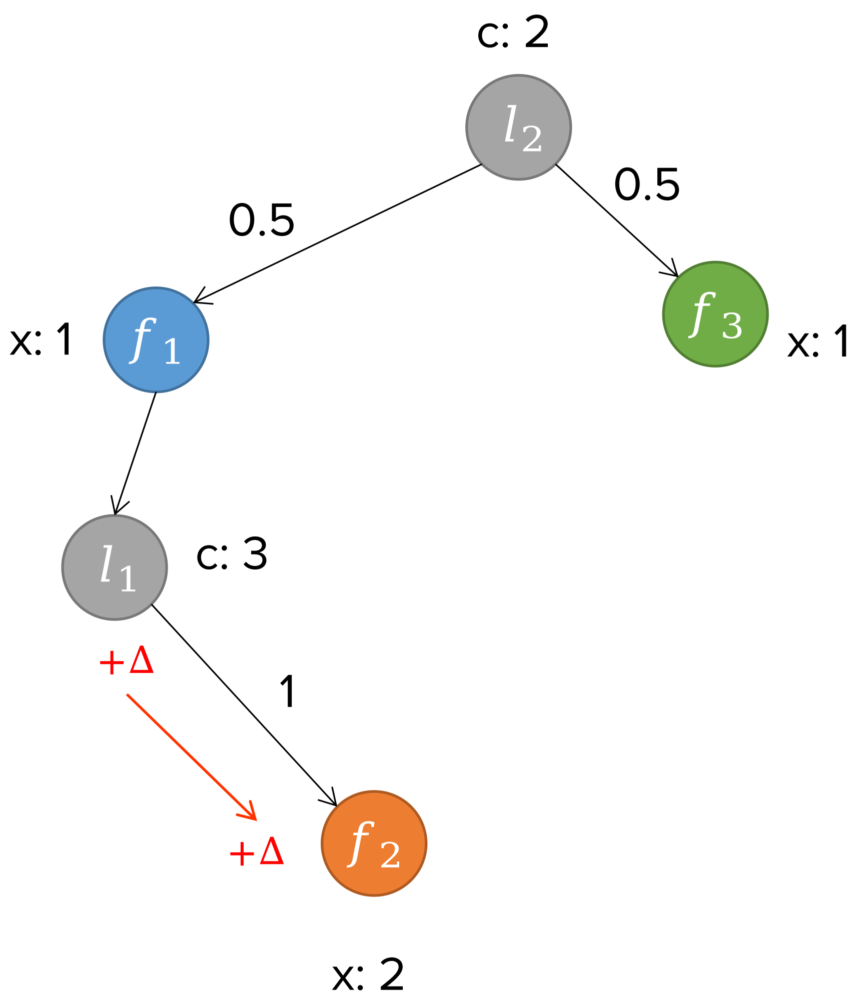
Bottleneck Structure

What happens if the capacity of l_1 is increased by Δ?



Bottleneck Structure

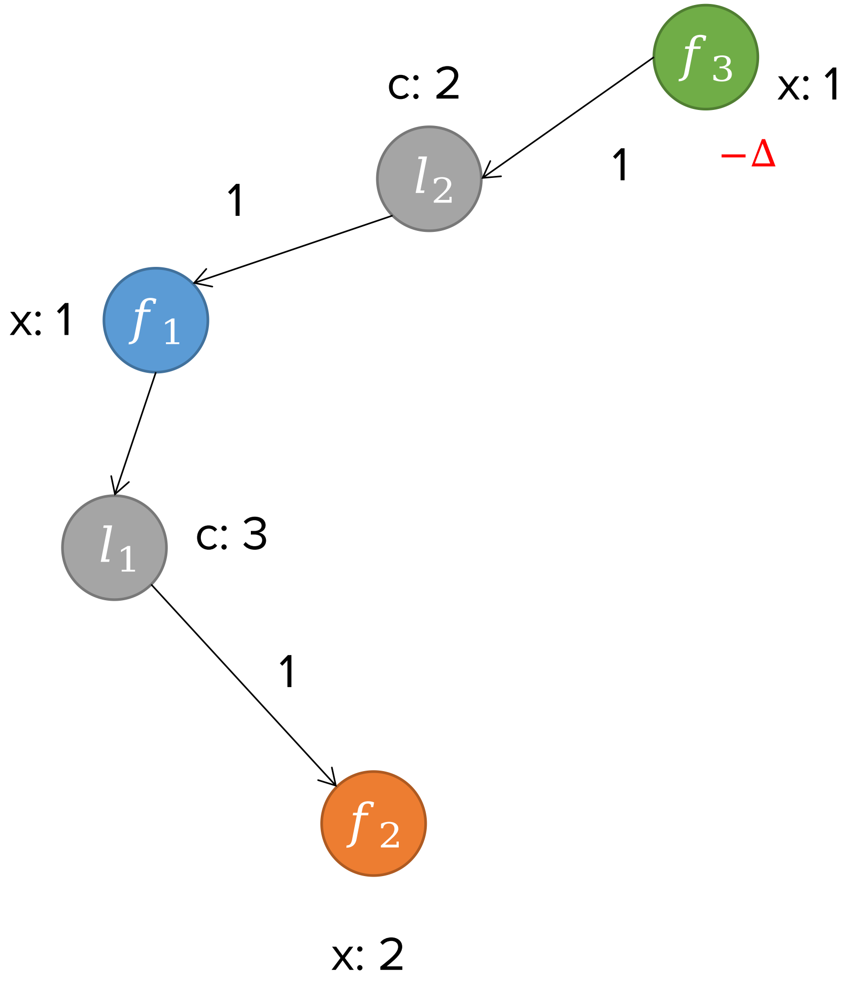
What happens if the capacity of l_1 is increased by Δ?



Bottleneck Structure

What happens if the capacity of l_1 is increased by Δ?

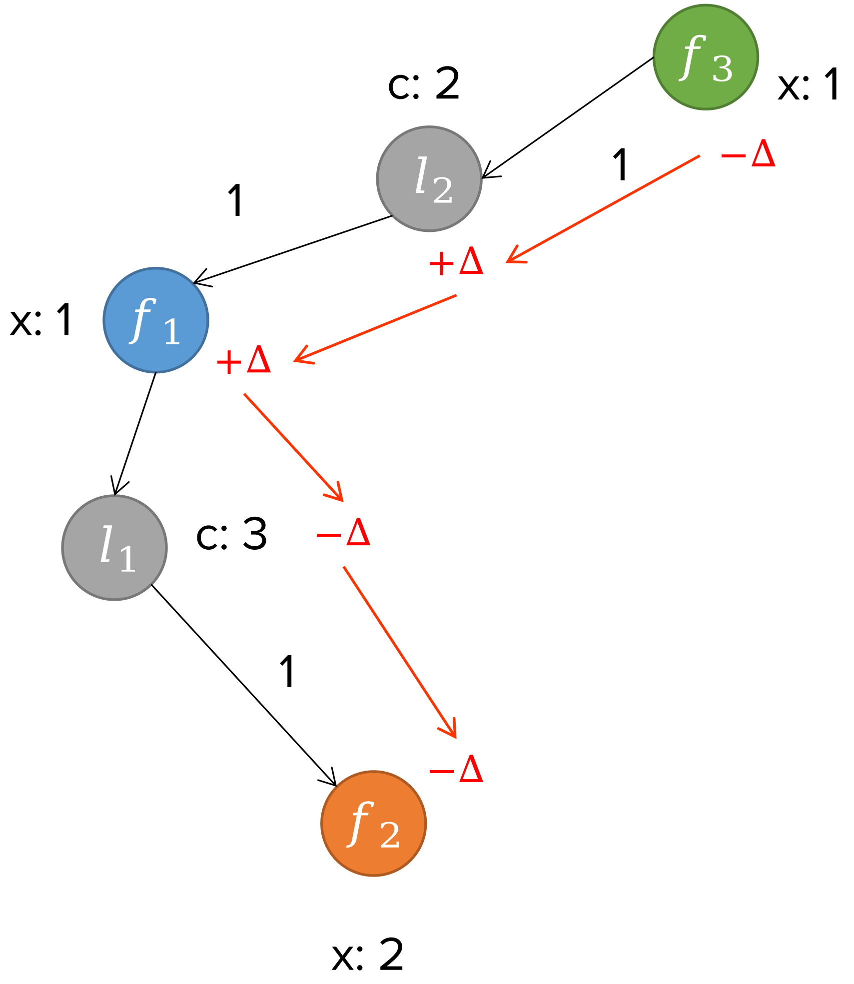
What happens if the rate of f_3 is decreased by Δ?



Bottleneck Structure

What happens if the capacity of l_1 is increased by Δ?

What happens if the rate of f_3 is decreased by Δ?



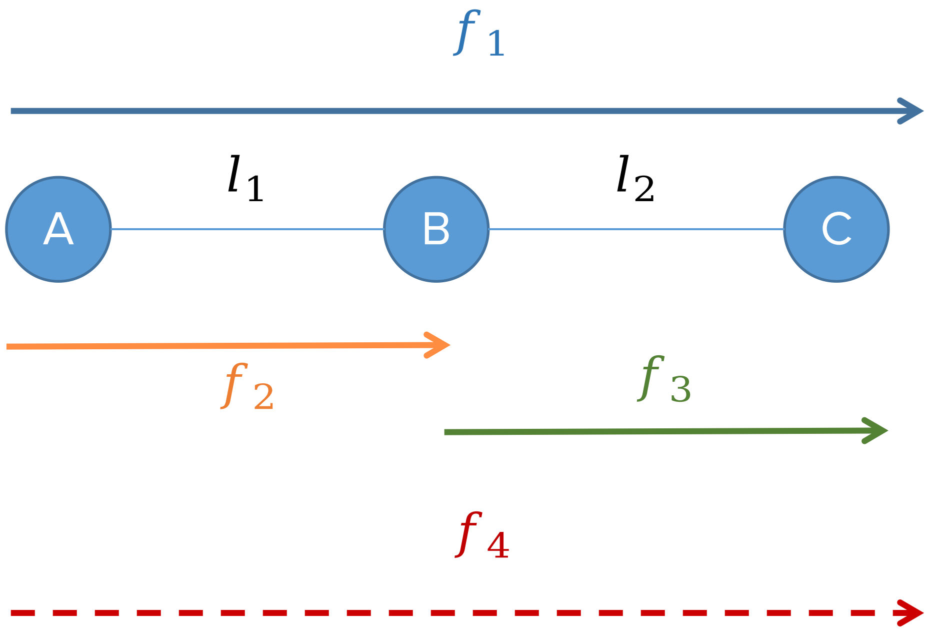
Use Case: Throughput Prediction Service

In a network with dynamic resouce allocation and a set of background flows F_B, predict the rate for a set of unestablished new flows F_N, e.g., to determine the best server/peer selection

Throughput prediction with the bottleneck service:

- Application specify the set of flows F_N
- Server adds the flows to the flow set $F = F_B \cup F_N$ and computes the bottleneck structure
- Server returns the flow nodes that represent the unestablished flows with the rate attribute

Example



$$F_N = \{f_4\},\ F_B = \{f_1, f_2, f_3\}$$

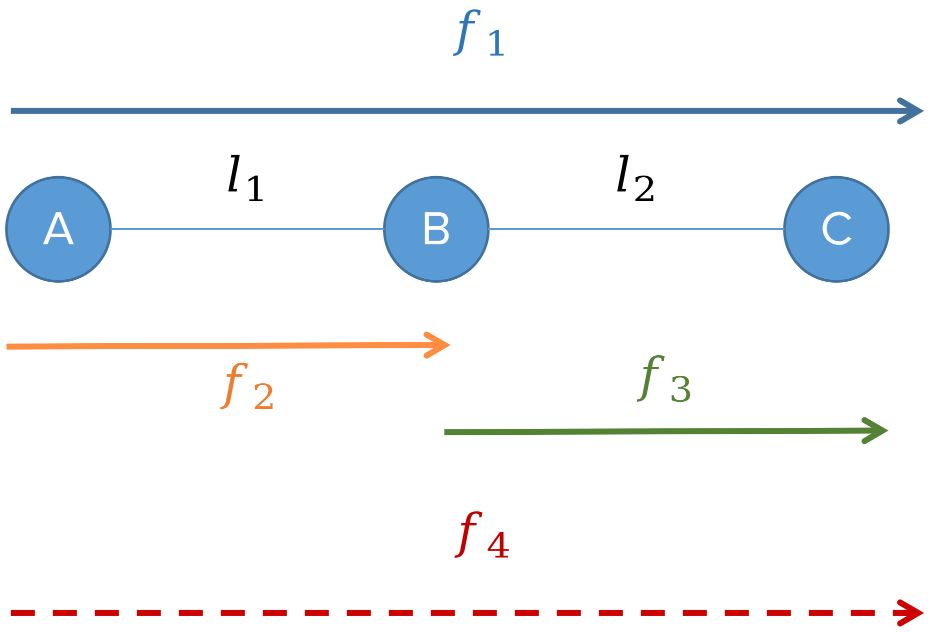
Assume a client wants to query the rate of f_4 before establish the connection

The ALTO server monitors the network and there are 3 background flows*

* The background flows are giant flows that will reach the equilibrium rate. Bandwidth consumed by small flows will be subtracted from the link capacity

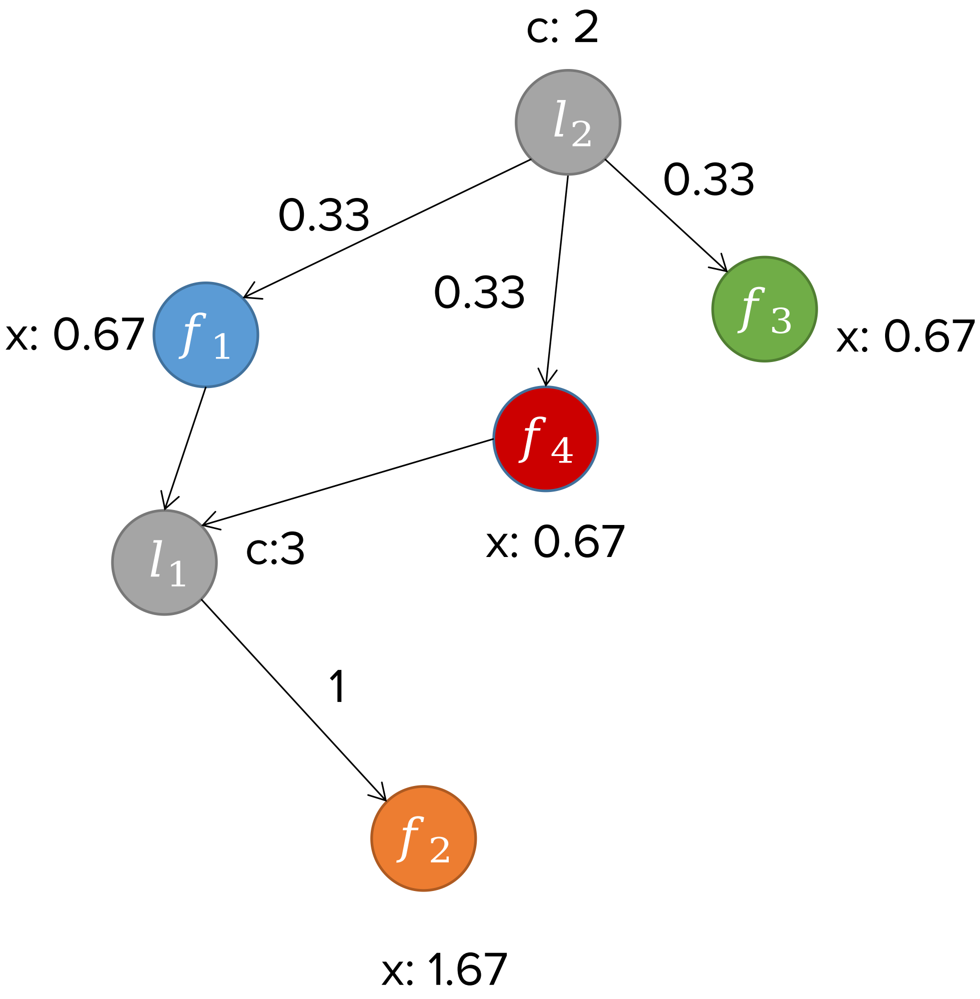
Example

With the bottleneck service, the server will be able predict the rate of f_4: 0.67 Mbps



$F_N = \{f_4\}, F_B = \{f_1, f_2, f_3\}$

return: rate (f_4)=0.67 Mbps



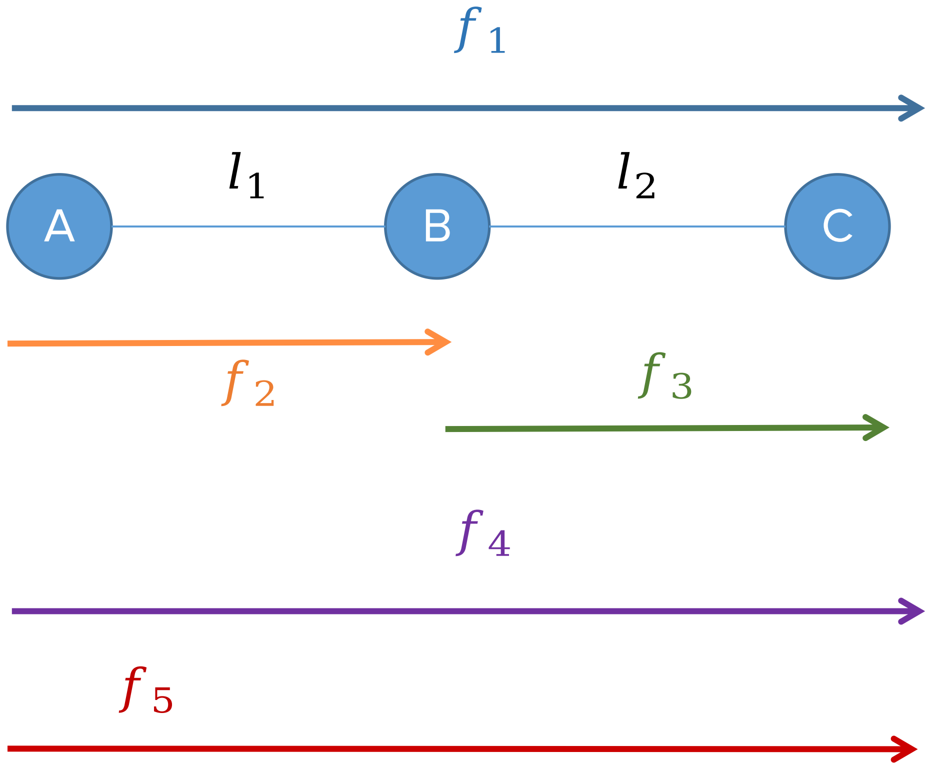
Use Case: Application TE for Time-bounded Data Transfers

In a network with dynamic resource allocation, a client wants to speed up a flow by rate limiting flows with lower priorities

Realize with bottleneck service:

- Application submits the target flow f_t and the flows F_R to be rate limited (including established or unestablished flows)
- Server returns the bottleneck structure
- Application/Server calculates the partial derivative of $\mathrm{rate}(f_t)$ over the rate of flows in F_R and chooses the one with the smallest negative value

Example



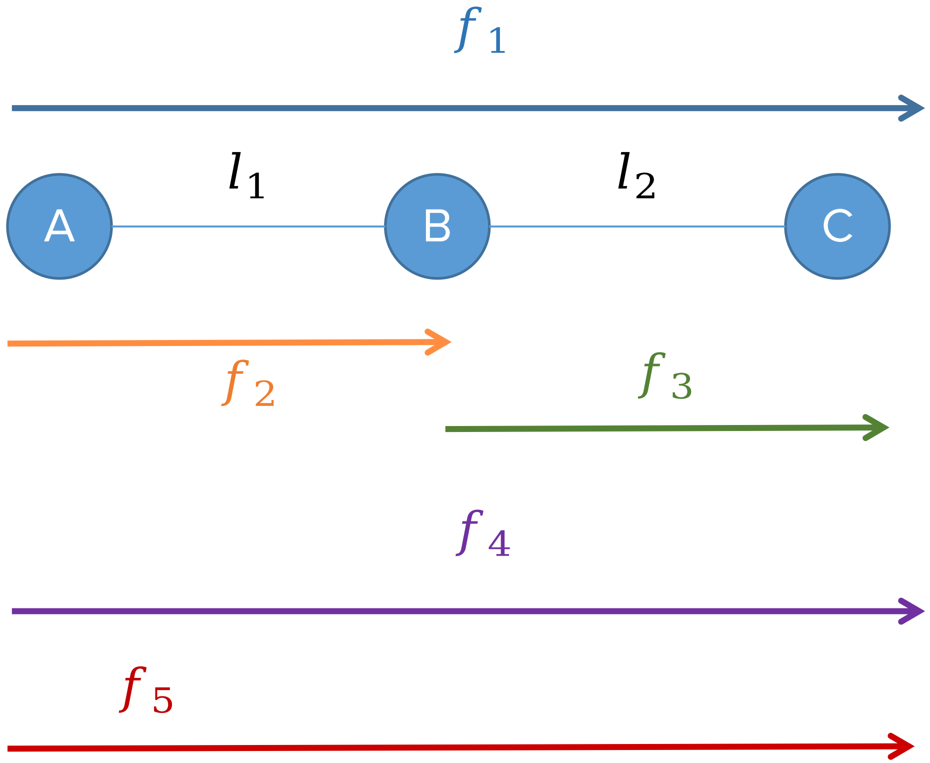
$f_t = f_5$, $F_R = \{f_2, f_3, f_4\}$

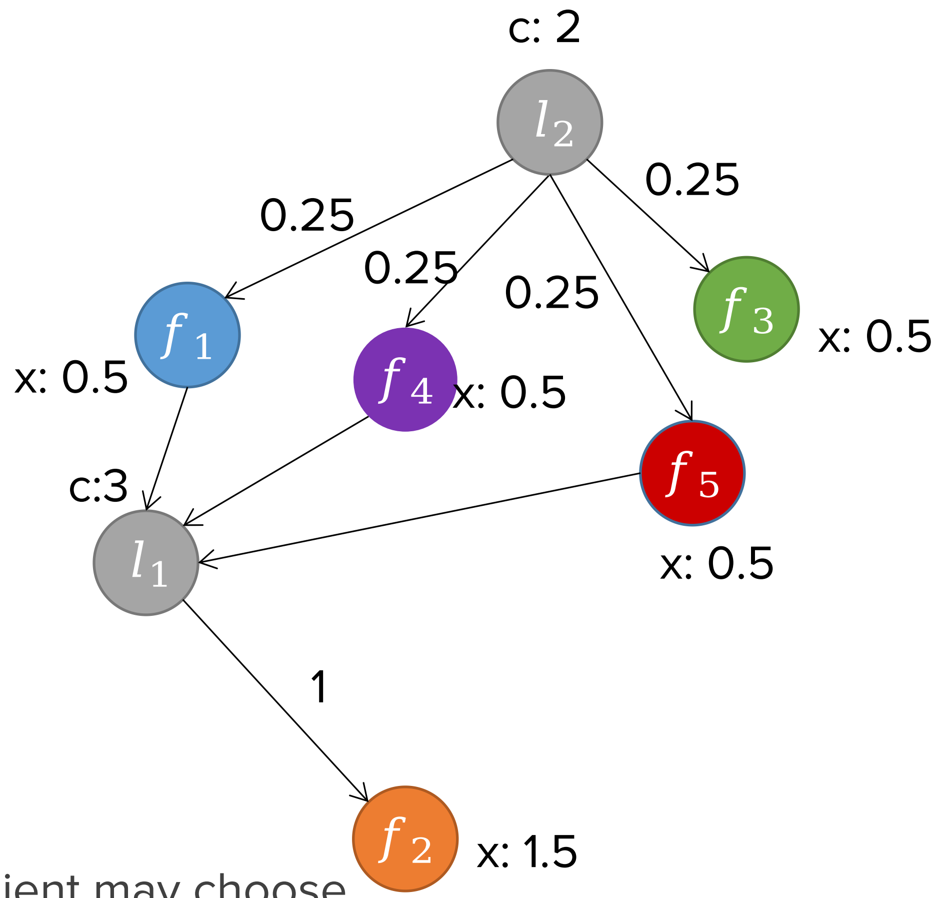
Target rate is 0.8

Assume a client wants to increase the rate of f_5 to be 0.5 Mbps and can only rate limit flows f_2, f_3, f_4

Example

$$\frac{\partial r_5}{\partial r_2} = 0,\ \frac{\partial r_5}{\partial r_3} = -0.5, \frac{\partial r_5}{\partial r_4} = -0.5$$



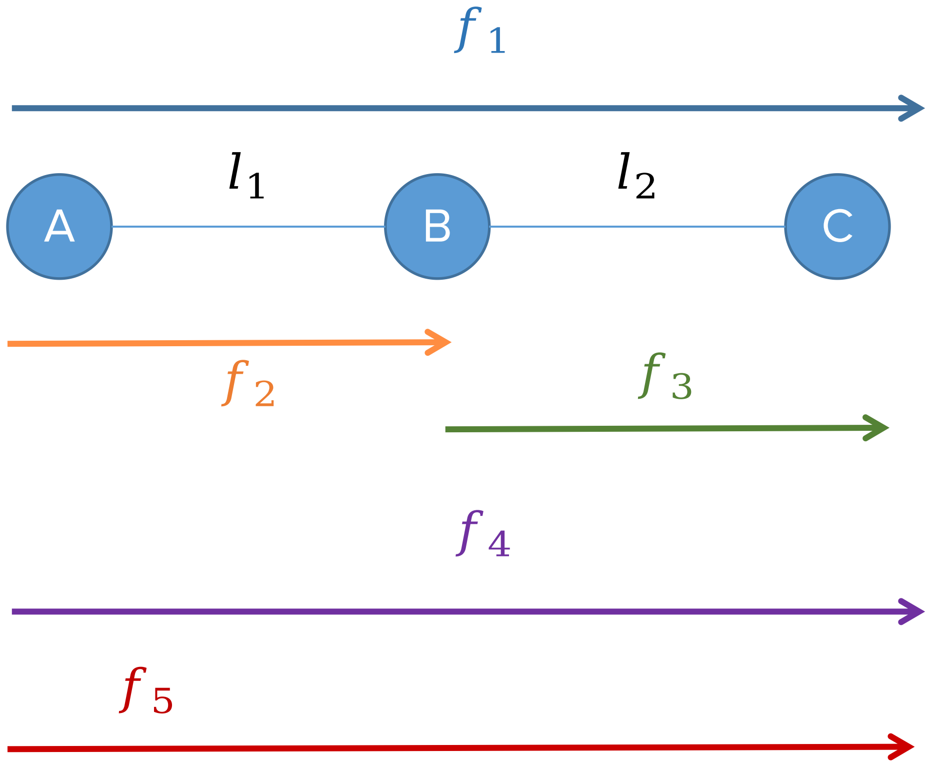


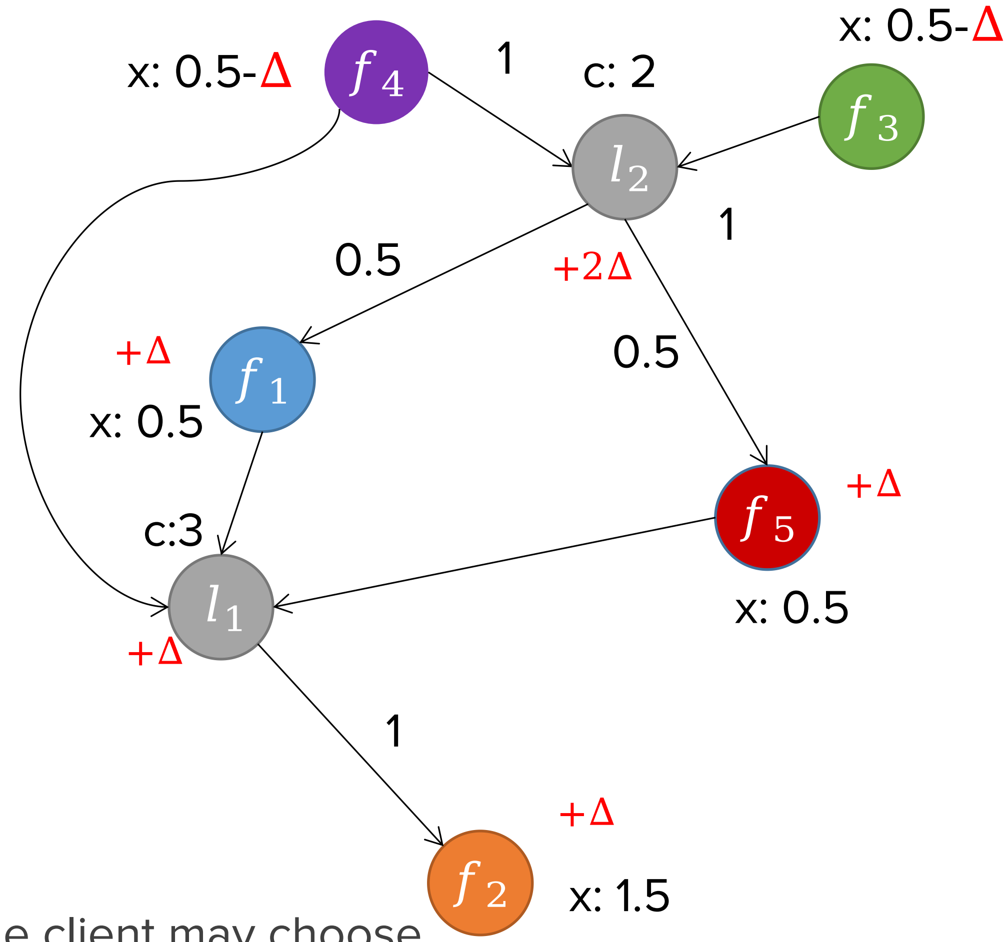
$f_t = f_5$, $F_R = \{f_2, f_3, f_4\}$

Target rate is 0.8

With the flow gradient, the client may choose the one with the smallest negative gradient to rate limit the traffic: r_3 and r_4

Example





$f_t = f_5$, $F_R = \{f_2, f_3, f_4\}$

Target rate is 0.8

With the flow gradient, the client may choose the one with the smallest negative gradient to rate limit the traffic: r_3 and r_4, $\Delta = 0.3$

G2

G2 is an optimization framework based on the quantative theory of bottleneck strcuture

It provides efficient algoirthms to compute bottleneck structures and conduct application optimizations for networks with max-min fairness

It is getting a lot of attention in both academia and industry

On-going deployment effort

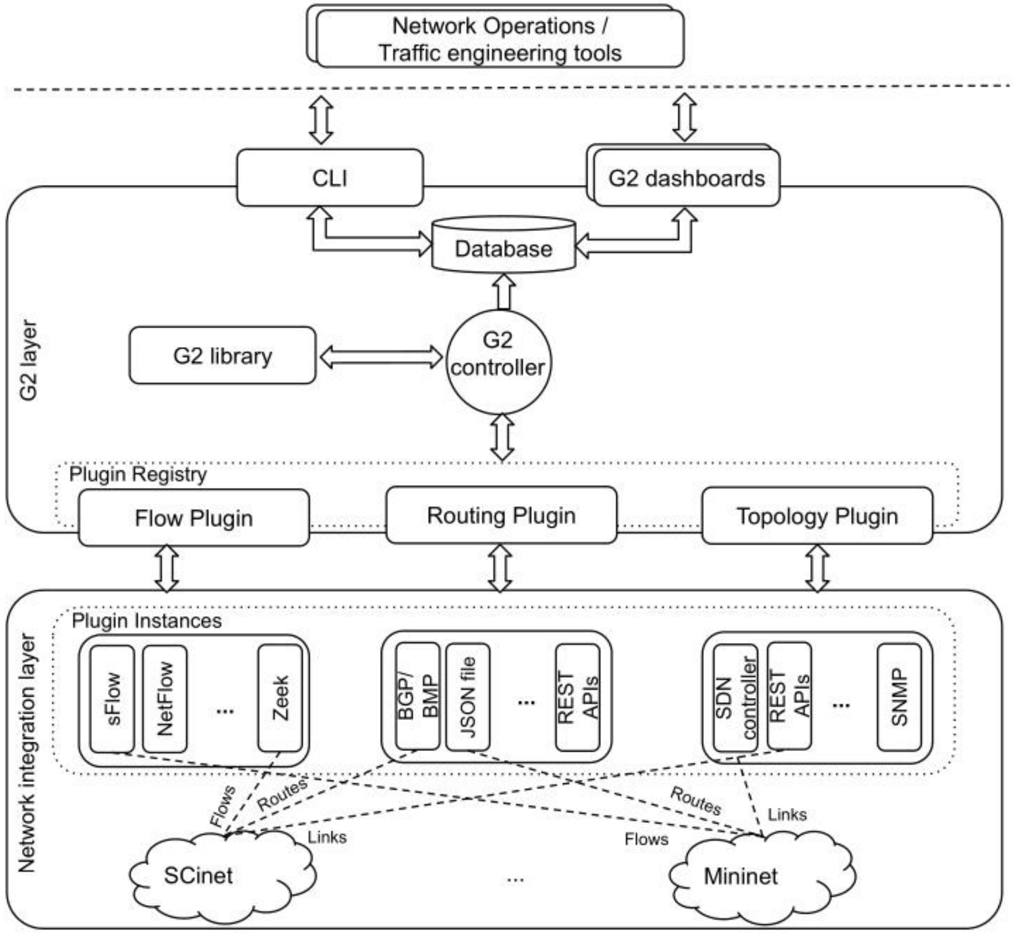
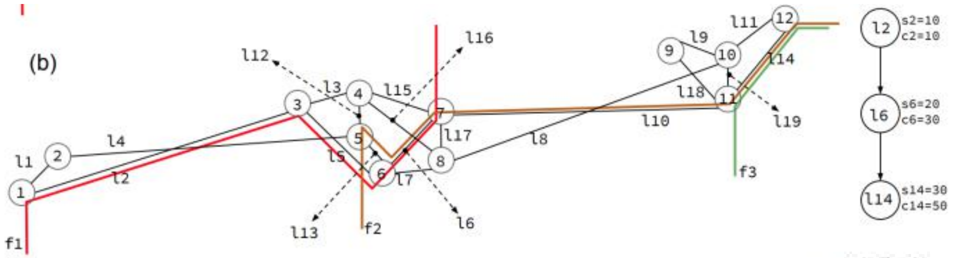


Fig. 5: G2 framework architecture.

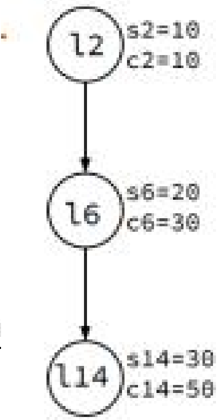
G2

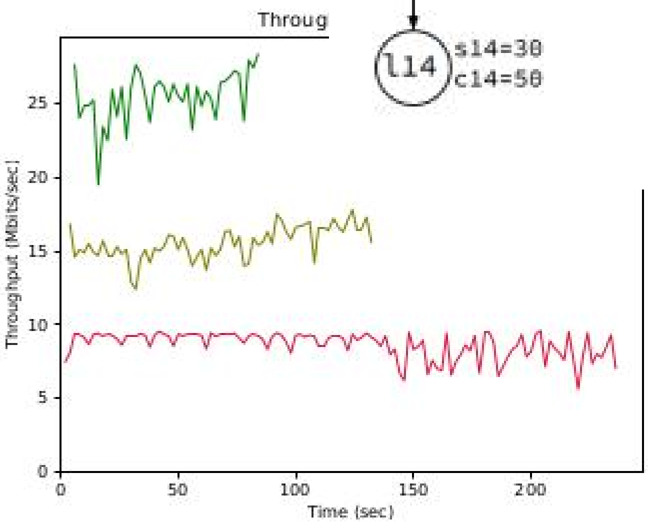
It has shown potential in predicting throughput for TCP traffic

It has demonstrated use cases such as **optimal rate path, network planning** and **TE for time-bounded traffic** can be effectively solved using bottleneck structure

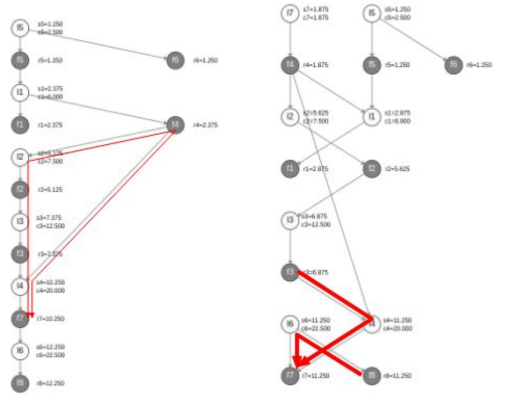


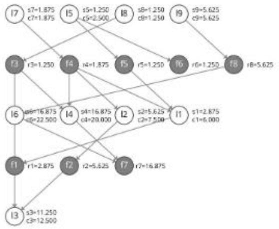
Jordi Ros-Giralt, Sruthi Yellamraju, Atul Bohara, Richard Lethin, Josie Li, Ying Lin, Yuanlong Tan, Malathi Veeraraghavan, Yuang Jiang, and Leandros Tassiulas. 2019. *G2: A Network Optimization Framework for High-Precision Analysis of Bottleneck and Flow Performance*. INDIS'19





(b) 3-level / 3 BBR flows.





Ros-Giralt, J., Amsel, N., Yellamraju, S., Ezick, J., Lethin, R., Jiang, Y., Feng, A., Tassiulas, L., Wu, Z., Teh, M.Y. and Bergman, K., 2021. *A Quantitative Theory of Bottleneck Structures for Data Networks*. IEEE Transactions on Networking (under review)

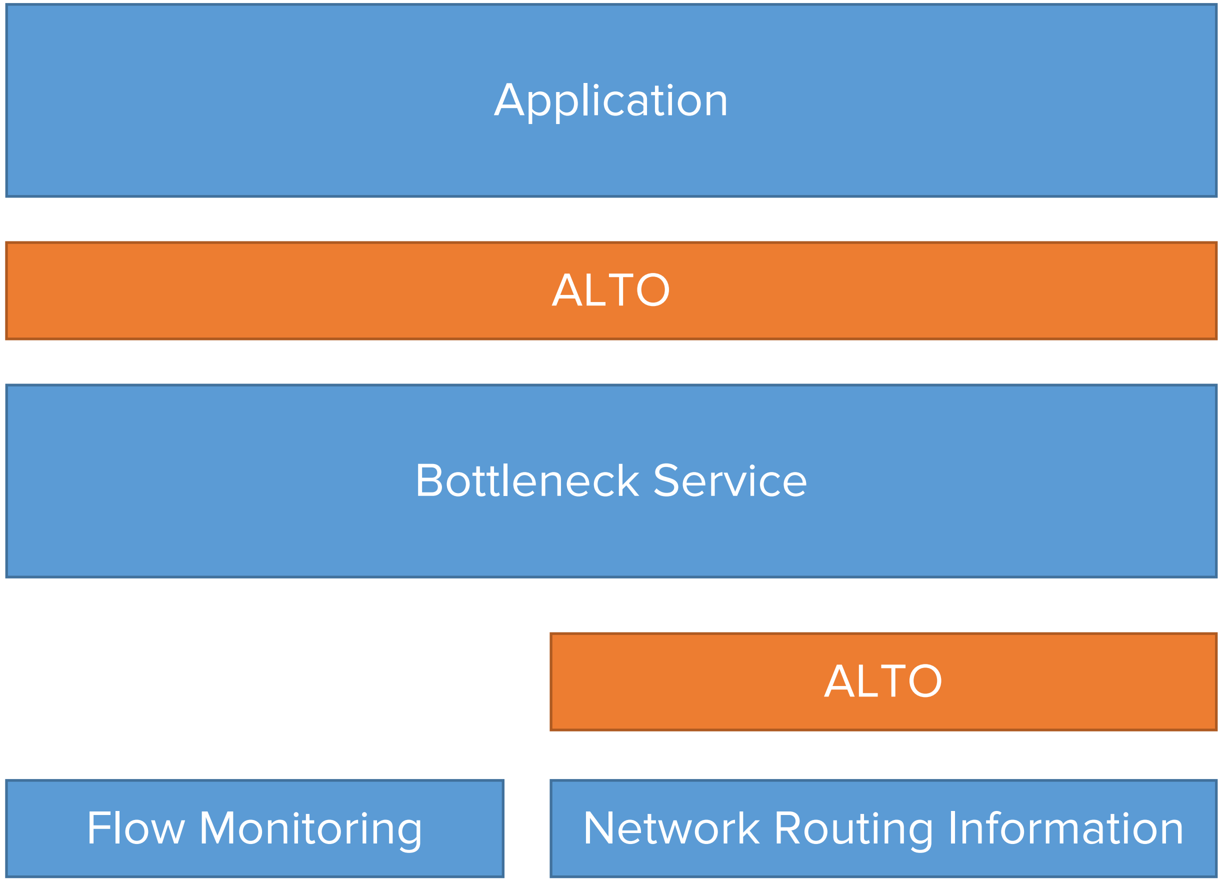
Flow	Experiment 1	Experiment 2	Experiment 3
f_1	2.44 / 2.37	2.57 / 2.87	2.65 / 2.87
f_2	4.78 / 5.12	5.16 / 5.62	5.33 / 5.62
f_3	6.99 / 7.37	6.57 / 6.87	1.18 / 1.25
f_4	2.72 / 2.37	1.74 / 1.87	1.73 1.87
f_5	1.18 / 1.25	1.33 / 1.25	1.29 / 1.25
f_6	1.42 / 1.25	1.19 / 1.25	1.19 / 1.25
f_7	9.51 / 10.25	9.81 / 11.25	15.34 / 16.87
f_8	11.48 / 12.25	11.06 / 11.25	5.27 / 5.62

(a) Without any traffic shaping. (b) Traffic shaping f_4. (c) Traffic shaping f_3, f_4 and f_8.

Bottleneck Services with ALTO

ALTO can both be the **southbound** or the **northbound** of the bottleneck service

We focus on providing bottleneck service with ALTO as the northbound



Requirement from Use Cases

Specify flows and their status (unestablished or established)

Specify customized (linear) constraints

Get throughput prediction

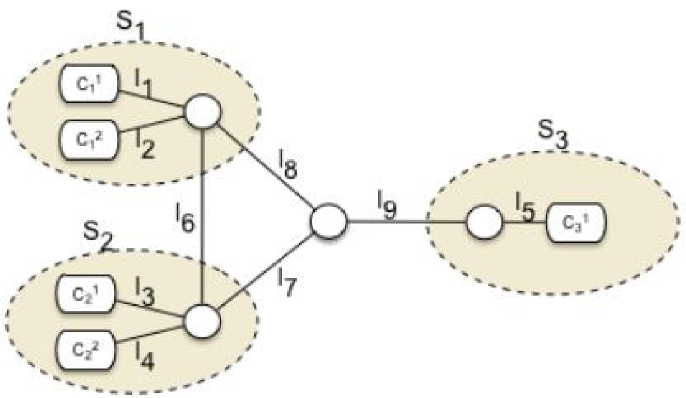
Get link gradient

Get flow gradient

UC1: throughput prediction

UC2: TE for time-bounded flow

Throughput Prediction (Site-level Flow)



Input:

```
{
  "pid-flows": {
    "f1": { "type": "unestablished",
            "src": "c11", "dst": "c21" },
    "f2": { ... }
  },
  "cost-type": { "cost-metric": "tput",
                 "cost-mode": "numerical" }
}
```

Output:

```
{
  "meta": ...,
  "flow-cost-map": {
    "f1": 500000,
    "f2": ...
  }
}
```

Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. 2015. *BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing*. SIGCOMM'15

A new filter type to enable flow-based query and response
At most one established/unestablished flow between a source and destination pair
Use the "tput" metric to specify the throughput value

Throughput Prediction (TCP-level Flow)

Input:

```
{
  “flows”: {
    “f1”: { “type”: “unestablished”, “src”: ..., “dst”: ... },
    “f2”: { “type”: “established”,
            “src”: ..., “dst”: ...,
            “srcport”: ..., “dstport”: ... }
  },
  “cost-type”: { “cost-metric”: “tput”,
                 “cost-mode”: “numerical” }
}
```

Output:

```
{
  “meta”: ...,
  “flow-cost-map”: {
    “f1”: 66666,
    “f2”: 133333
  }
}
```

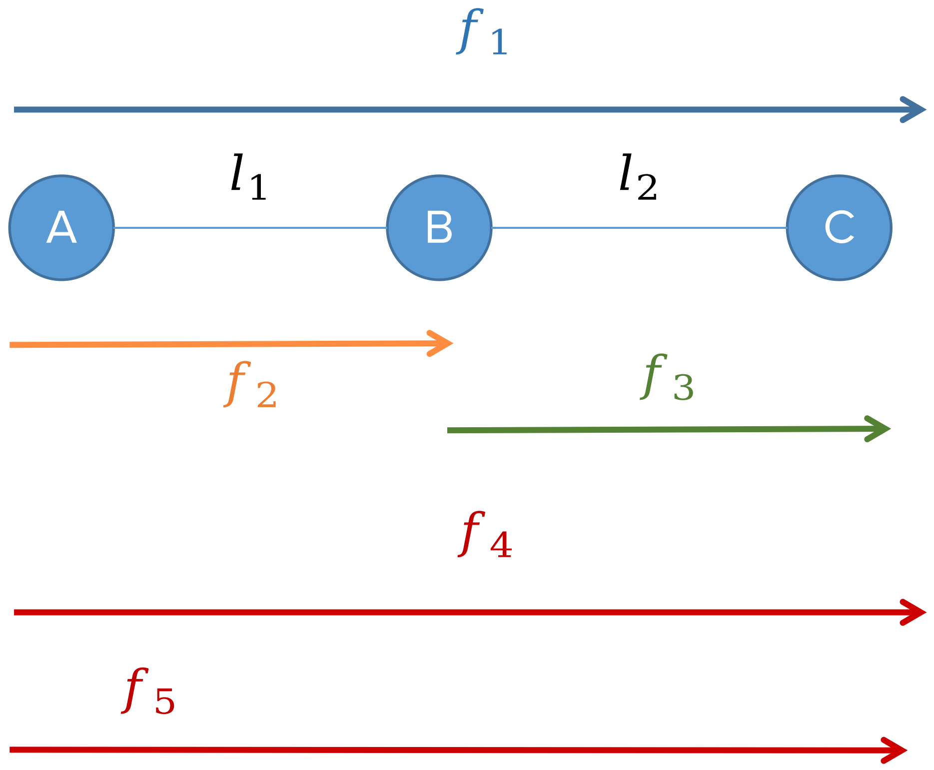
A new filter type to enable flow-based query
Established flows and unestablished flows have different attributes
Use the “tput” metric to specify the throughput value

Use Case: Application TE for Time-bounded Data Transfers

In a network with dynamic resource allocation, a client wants to speed up a flow by rate limiting flows with lower priorities

Realize with bottleneck structure:

- Application submits the target flow f_t and the flows F_R to be rate limited (including established or unestablished flows)
- Server returns the bottleneck structure
- Application/Server calculates the partial derivative of $\mathrm{rate}(f_t)$ over the rate of flows in F_R and chooses the one with the smallest negative value



Traffic Engineering for Time-bounded Data Transfers

Target: f1 = 80000 bps Iteration 1:

Input:


```
{
  “flows”: {
    “f5”: { “type”: “unestablished”, “src”: ..., “dst”: ... },
    “f2”: { “type”: “established”,
            “src”: ..., “dst”: ...,
            “srcport”: ..., “dstport”: ... },
    “f3”: {...}, “f4”: {...}
  },
  “multi-cost-types”: [
    { “cost-metric”: “tput”, “cost-mode”: “numerical” },
    { “cost-metric”: “flow-gradient”, “cost-mode”: “numerical” }
  ]
}
```


Output


```
{
  “meta”: ...,
  “flow-cost-map”: {
    “f5”: [ 50000, { ... }],
    “f2”: [ 150000, { “f5”: 0, ...}],
    “f3”: [ 50000, { “f5”: -0.5, ...}],
    “f4”: [ 50000, { “f5”: -0.5, ...}]
  }
}
```


Query the predicted throughput and flow gradient

Traffic Engineering for Time-bounded Data Transfers

Target: f1 = 80000 Iteration 2:

Input:

```
{
  “flows”: {
    “f5”: { “type”: “unestablished”, “src”: ..., “dst”: ... },
    “f2”: { “type”: “established”,
            “src”: ..., “dst”: ...,
            “srcport”: ..., “dstport”: ... },
    “f3”: {...}, “f4”: {...}
  },
  “flow-constraints”: [ “f3.tput <= 20000”, “f4.tput <= 20000” ],
  “multi-cost-types”: [
    { “cost-metric”: “flow-gradient”, “cost-mode”: “numerical” },
    { “cost-metric”: “tput”, “cost-mode”: “numerical” }
  ]
}
```

Output

```
{
  “meta”: ...,
  “flow-cost-map”: {
    “f5”: [ 80000, { ... }],
    “f2”: [ 180000, { “f5”: 0 }],
    “f3”: [ 20000, { “f5”: -0.5, ...}],
    “f4”: [ 20000, { “f5”: -0.5, ...}]
  }
}
```

Choose the flow with negative gradient and set up rate limit
Update the query with the rate limit constraint

Summary

- Bottleneck service is useful in many networks
- Solid work has been established (G2 work) for max-min fairness and is moving towards real deployment
- Bottleneck service can potentially be integrated as part of the ALTO framework
  - Some scenarios (e.g., site-level throughput prediction) ready with current protocol standards
  - Some scenarios may need extensions (flow-level query, encoding of bottleneck structure, gradient information)

Thanks!

Q & A