Optimization-Based Network Flow Deadline Scheduling

Shih-Hao Tseng, (pronounced as "She-How Zen")
joint work with Andrey Gushchin and Kevin Tang

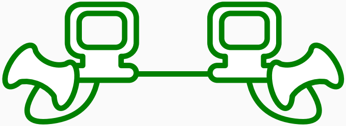
November 11, 2016

School of Electrical and Computer Engineering, Cornell University

Introduction

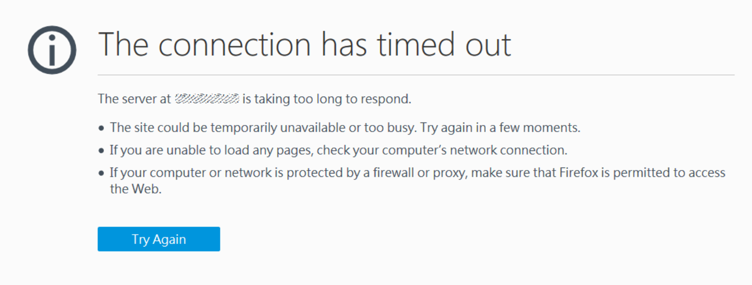
- Real-time network applications in modern network require the services to be provided in a timely fashion.

(a) Videoconference

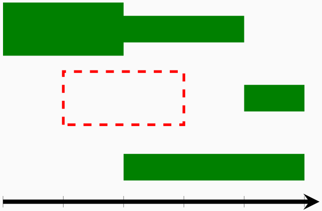


(b) Online gaming

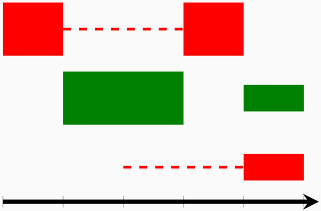
- Traditional traffic rate control approaches, such as TCP, generally do not take flow deadlines into account.



- Some new heuristics have been proposed to increase the number of satisfied flow deadlines comparing to the deadline-oblivious protocols.



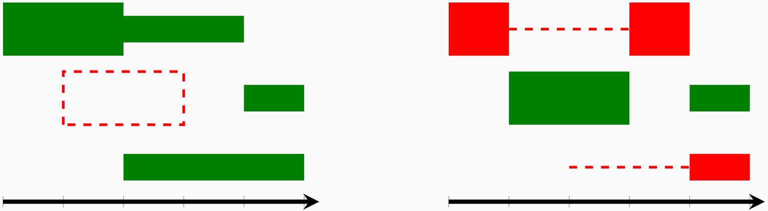
(a) FCFS Greedy:
D^3 (Wilson et al., 2011)



(b) Prioritizing:
PDQ (Hong et al., 2012),
pFabric (Alizadeh et al., 2013)

Introduction

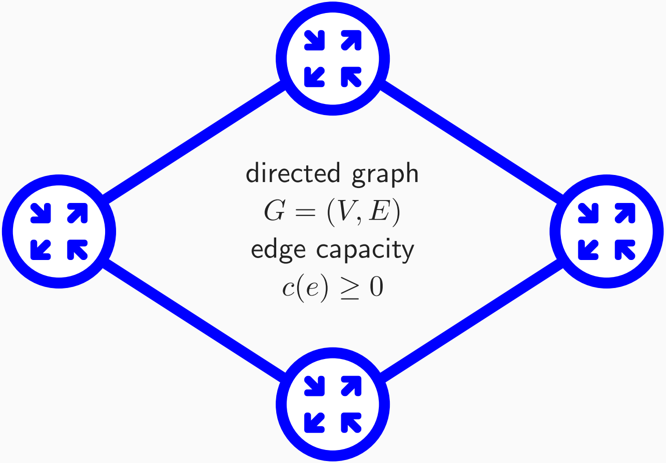
- Some new heuristics have been proposed to increase the number of satisfied flow deadlines comparing to the deadline-oblivious protocols.
- What is the best possible scheduling algorithm?



(a) FCFS Greedy:
D^3 (Wilson et al., 2011)

(b) Prioritizing:
PDQ (Hong et al., 2012),
pFabric (Alizadeh et al., 2013)

Formulation



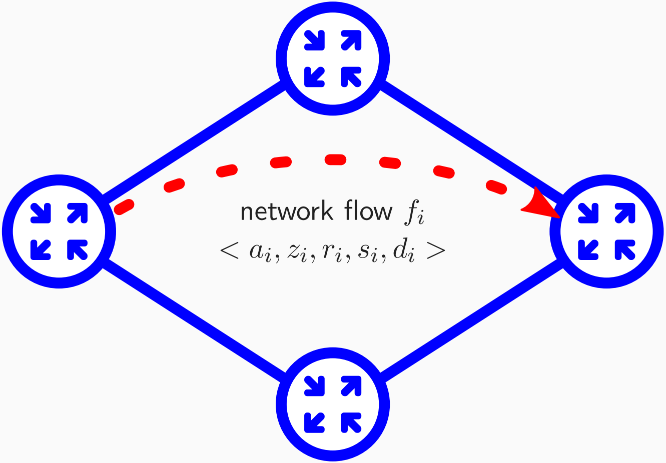
directed graph

$G = (V, E)$

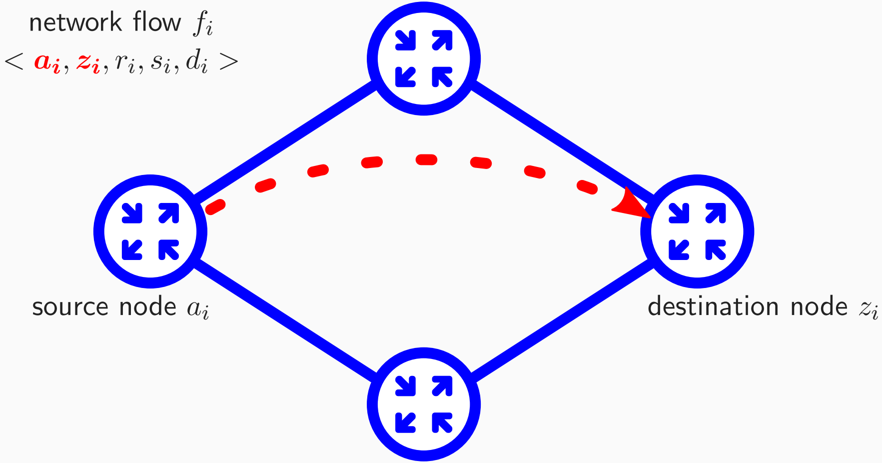
edge capacity

$c(e) \geq 0$

Formulation



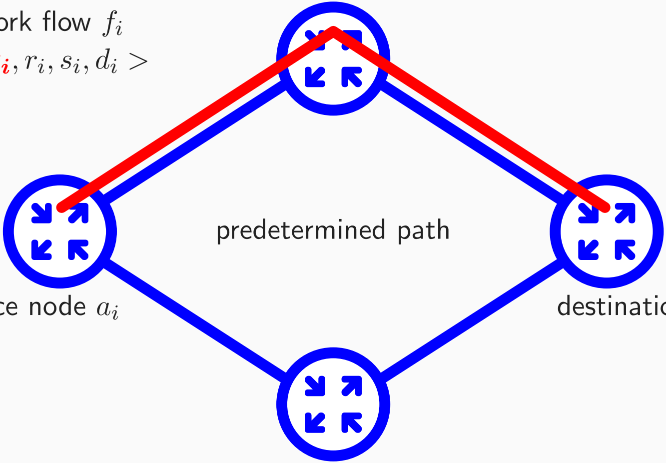
Formulation



Formulation

network flow f_i

$< \boldsymbol{a_i}, \boldsymbol{z_i}, r_i, s_i, d_i >$

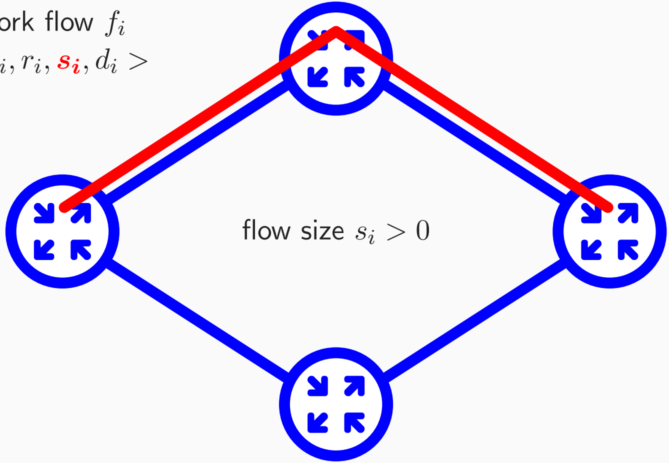


source node a_i

destination node z_i

Formulation

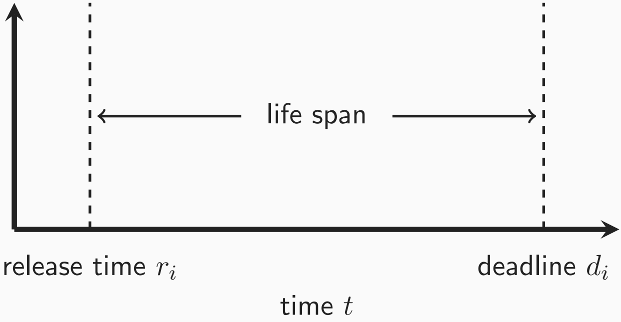
network flow f_i
$< a_i, z_i, r_i, \boldsymbol{s_i}, d_i >$



Formulation

network flow f_i

$< a_i, z_i, \boldsymbol{r_i}, s_i, \boldsymbol{d_i} >$



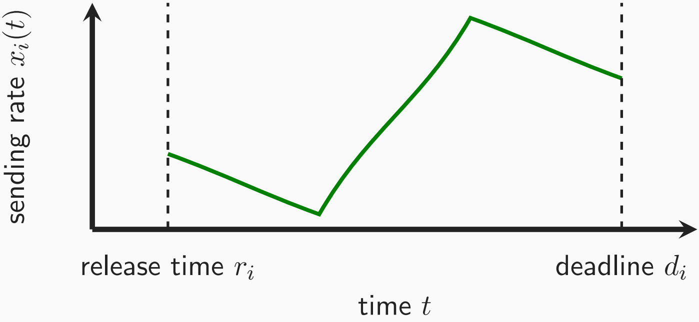
Formulation

network flow f_i

$< a_i, z_i, r_i, s_i, d_i >$

state variable: the residual of the flow f_i

$$S_i(t) = s_i - \int_0^t x_i(\hat{t})d\hat{t}$$



Formulation

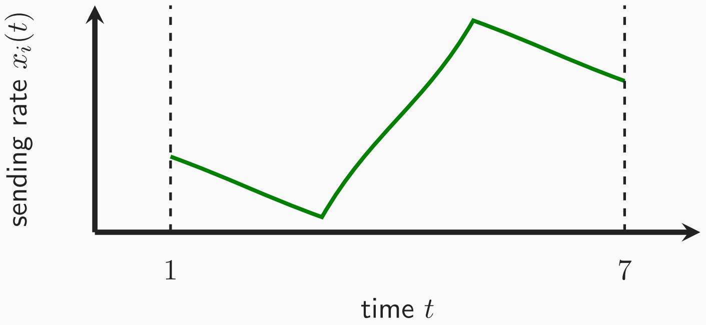
network flow f_i

$< a_i, z_i, 1, 9, 7 >$

state variable: the residual of the flow f_i

$$S_i(t) = s_i - \int_0^t x_i(\hat{t})d\hat{t} = 9$$

for all $t < r_i = 1$



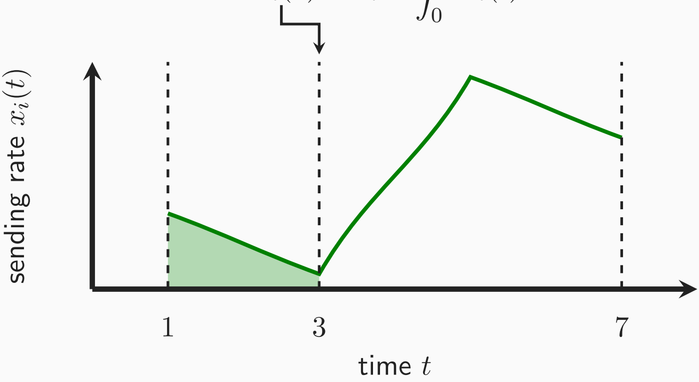
Formulation

network flow f_i

$< a_i, z_i, 1, 9, 7 >$

state variable: the residual of the flow f_i

$$S_i(3) = s_i - \int_0^3 x_i(\hat{t})d\hat{t} = 7.8$$



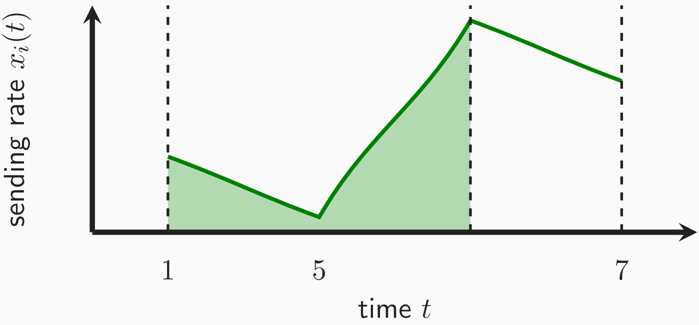
Formulation

network flow f_i

$< a_i, z_i, 1, 9, 7 >$

state variable: the residual of the flow f_i

$$S_i(5) = s_i - \int_0^5 x_i(\hat{t})d\hat{t} = 4.8$$



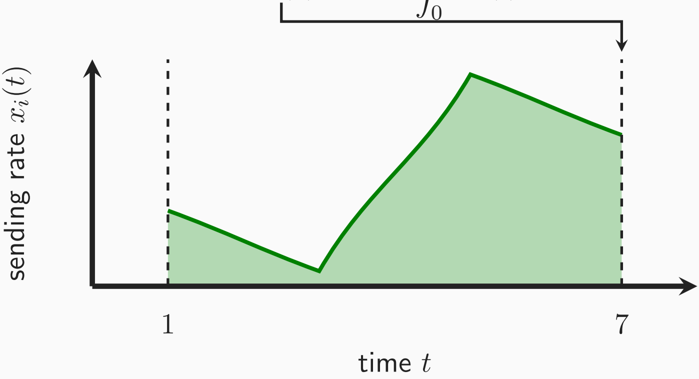
Formulation

network flow f_i

$< a_i, z_i, 1, 9, 7 >$

state variable: the residual of the flow f_i

$$S_i(7) = s_i - \int_0^7 x_i(\hat{t})d\hat{t} = 0 \Rightarrow \text{satisfied}$$

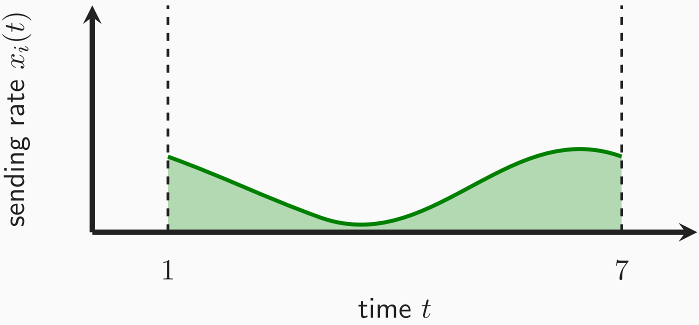


network flow f_i

$< a_i, z_i, 1, 9, 7 >$

state variable: the residual of the flow f_i

$$S_i(7) = s_i - \int_0^7 x_i(\hat{t})d\hat{t} = 5.4 \Rightarrow \text{unsatisfied}$$



The Continuous-Time Control Problem

maximize	number of satisfied flows
subject to	state variables
	capacity constraints
	sending rate constriants

Transformation to Optimization Problem

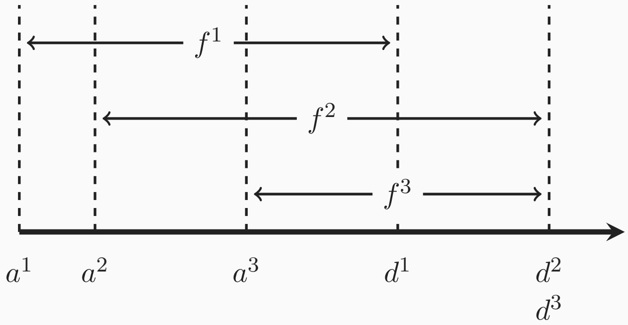
- Solving continuous-time control problem involves dealing with the Hamilton-Jacobi-Bellman (HJB) equation, which is in general hard to solve.

Transformation to Optimization Problem

- Solving continuous-time control problem involves dealing with the Hamilton-Jacobi-Bellman (HJB) equation, which is in general hard to solve.
- There exists a way to transform the continuous-time control problem into a finite variable optimization problem.

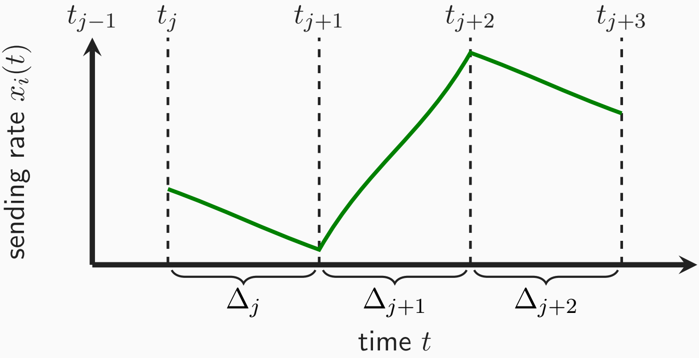
Transformation to Optimization Problem

We can consider all the release time and deadlines.



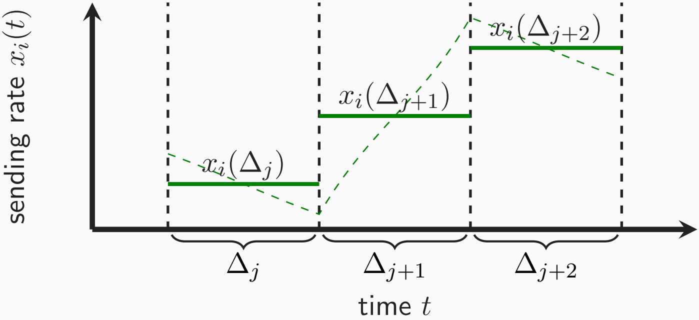
Transformation to Optimization Problem

$$t_j \in \bigcup_i \{a_i, d_i\} \qquad \Delta_j \in [r_i, d_i] \Leftrightarrow \Delta_j \in f_i$$



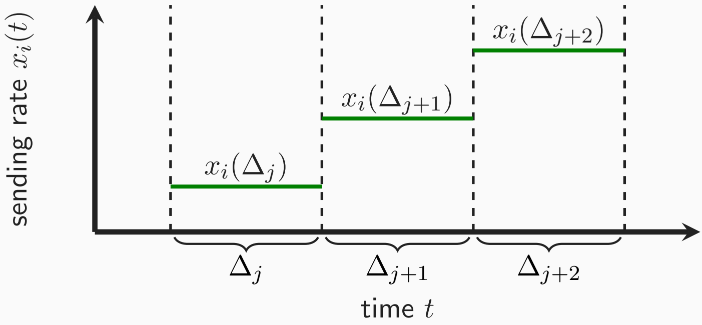
Transformation to Optimization Problem

The average of $x_i(t)$ over the intervals yields another feasible solution, which is piecewise constant.



Transformation to Optimization Problem

$$S_i(T) = s_i - \int_0^T x_i(\hat{t})d\hat{t} \quad \Rightarrow \quad S_i(T) = s_i - \sum_{\Delta_j \in f_i} x_i(\Delta_j)\,|\Delta_j|$$



The Finite Variable Optimization Problem

maximize	number of satisfied flows
subject to	weighted sums
	capacity constraints
	piecewise constant sending rates

NP-hardness

Proposition

The problem of finding an optimal piecewise constant rate control for the offline environment is NP-hard, and cannot be approximated to a constant factor in polynomial time (unless $P = NP$*).*

- It is also unlikely to solve the original continuous-time control problem in polynomial time.

NP-hardness

Proposition

The problem of finding an optimal piecewise constant rate control for the offline environment is NP-hard, and cannot be approximated to a constant factor in polynomial time (unless $P = NP$*).*

- It is also unlikely to solve the original continuous-time control problem in polynomial time.
- The proposition justifies the use of heuristics when approaching the problem.

The Optimization Problem - MILP

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} z_i \\
\text{subject to} \quad & S_i(T) = s_i - \sum_{\Delta_j \in f_i} x_i(\Delta_j)\,|\Delta_j| && \forall i \\
& z_i = \begin{cases} 1 & S_i(T) = 0 \\ 0 & S_i(T) > 0 \end{cases} && \forall i \\
& \text{capacity constraints} \\
& \text{piecewise constant sending rates}
\end{aligned}
$$

The Optimization Problem - MILP

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} z_i \\
\text{subject to} \quad & S_i(T) = s_i - \sum_{\Delta_j \in f_i} x_i(\Delta_j)\,|\Delta_j| && \forall i \\
& s_i z_i \le s_i - S_i(T) && \forall i \\
& z_i \in \{0, 1\} && \forall i \\
& \text{capacity constraints} \\
& \text{piecewise constant sending rates}
\end{aligned}
$$

Linear Programming Approximation (LPA)

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} z_i \\
\text{subject to} \quad & S_i(T) = s_i - \sum_{\Delta_j \in f_i} x_i(\Delta_j)\,|\Delta_j| && \forall i \\
& s_i z_i \leq s_i - S_i(T) && \forall i \\
& \underline{z_i \in \{0, 1\}} && \forall i \\
& \text{capacity constraints} \\
& \text{piecewise constant sending rates}
\end{aligned}
$$

Linear Programming Approximation (LPA)

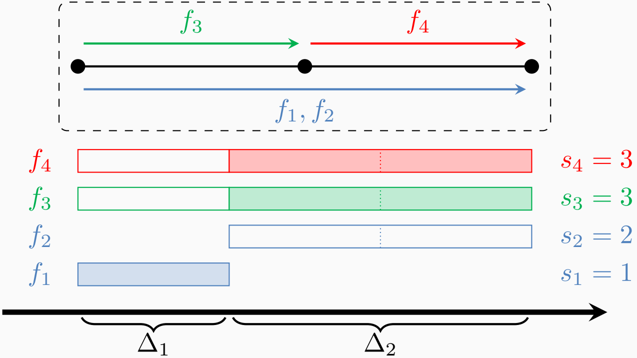
$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} z_i \\
\text{subject to} \quad & S_i(T) = s_i - \sum_{\Delta_j \in f_i} x_i(\Delta_j)\,|\Delta_j| && \forall i \\
& s_i z_i \leq s_i - S_i(T) && \forall i \\
& 0 \leq z_i \leq 1 && \forall i \\
& \text{capacity constraints} \\
& \\
& \text{piecewise constant sending rates}
\end{aligned}
$$

The Drawback of LPA

- LPA performs well, but it is still far from the optimal.

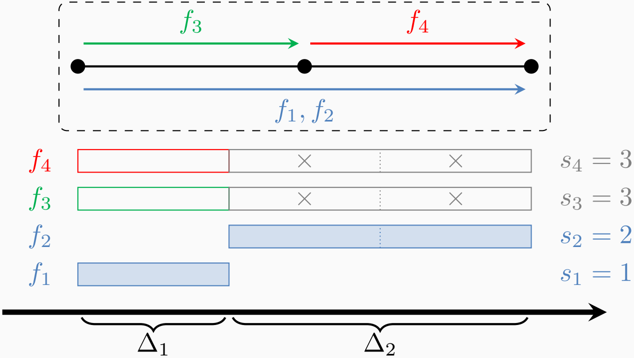
The Drawback of LPA

- LPA performs well, but it is still far from the optimal.
- One drawback of LPA is that it would allocate bandwidth for the flows whose deadlines cannot be satisfied anymore.



Iterative Removal of Unsatisfiable Flows

- To prevent the drawback, we can remove a flow whenever it is no longer possible to be satisfied.



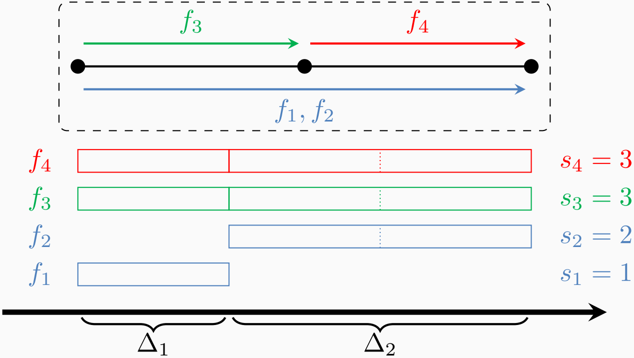
Iterative Linear Programming Approximation (ILPA)

Algorithm 1: Iterative Linear Programming Approximation (ILPA)

```
for Δ_j from earliest to the last do
    Remove the flows that cannot be satisfied anymore.
    Apply LPA to solve for x_i(Δ_j).
end for
```

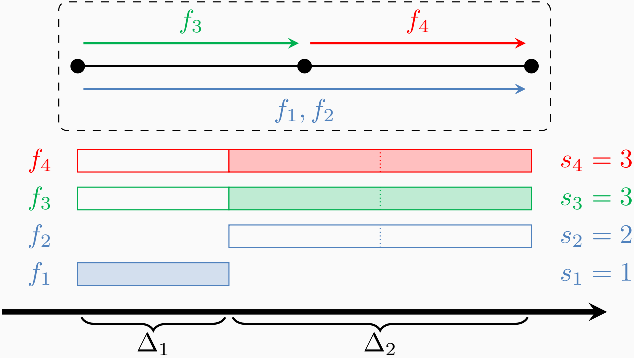
Iterative Linear Programming Approximation (ILPA)

- Consider only the satisfiable flows.



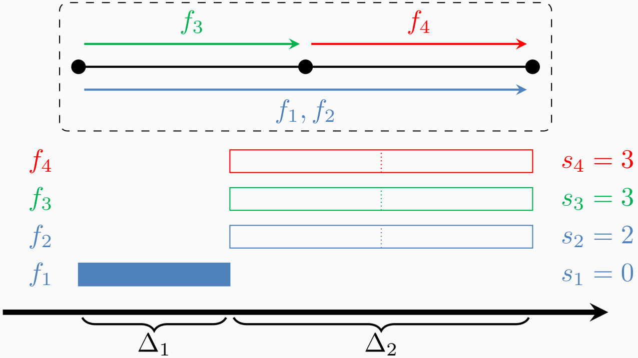
Iterative Linear Programming Approximation (ILPA)

- Use LPA to assign the sending rates.



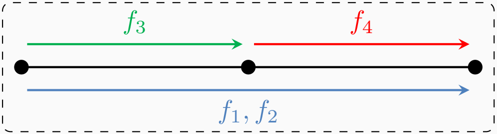
Iterative Linear Programming Approximation (ILPA)

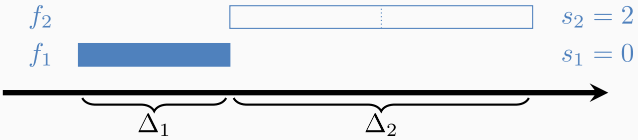
- Fix the sending rate $x_i(\Delta_1)$.



Iterative Linear Programming Approximation (ILPA)

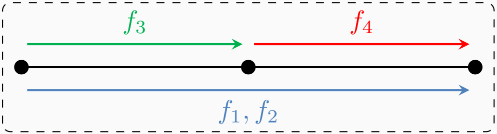
- Remove the unsatisfiable flows.

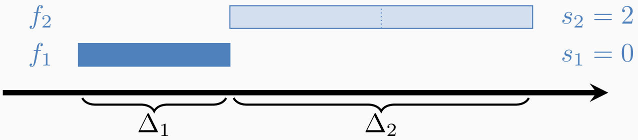




Iterative Linear Programming Approximation (ILPA)

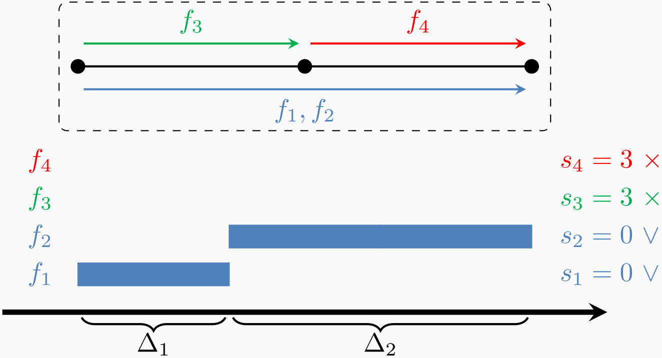
- Again, apply LPA to assign the sending rates.





Iterative Linear Programming Approximation (ILPA)

- Repeat the steps until walking through all Δ_j to get the resulted sending rates.



Online Linear Programming Approximation (OLPA)

- Offline: the information of all the flows is available at time 0.
- Online: no information about a flow f_i is available prior to its arrival time r_i.

We can design the online linear programming approximation (OLPA) similar to ILPA.

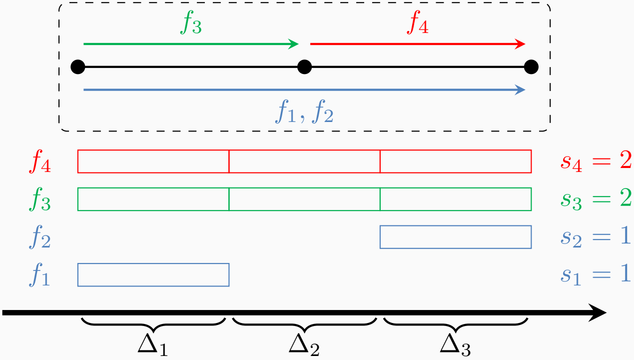
Online Linear Programming Approximation (OLPA)

Algorithm 2: Online Linear Programming Approximation (OLPA)

```
for whenever a new flow arrives do
    Compute the residual sizes of the flows.
    Apply ILPA to assign the sending rates.
end for
```

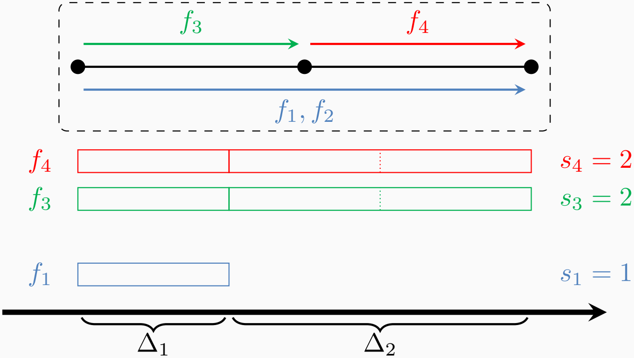
Online Linear Programming Approximation (OLPA)

- Suppose there are four flows, but f_2 arrives later.



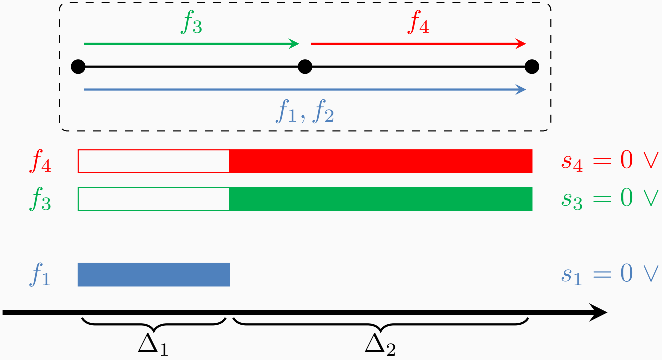
Online Linear Programming Approximation (OLPA)

- At the beginning of Δ_1, f_2 is not yet known.



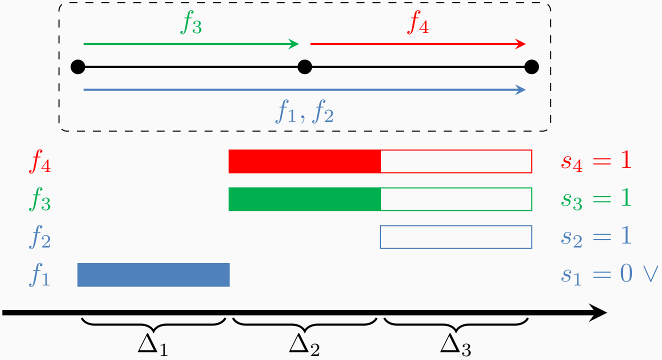
Online Linear Programming Approximation (OLPA)

- Apply ILPA to assign the sending rates.



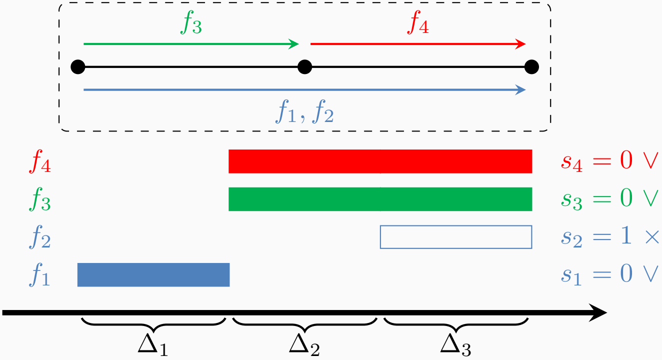
Online Linear Programming Approximation (OLPA)

- At the beginning of Δ_3, f_2 arrives and the residual sizes of the flows are updated.



Online Linear Programming Approximation (OLPA)

- Again, apply ILPA to assign the sending rates.



The Proposed Algorithms

- LPA is the linear relaxation.
- Iterative LPA (ILPA) maintains only the flows that can still be satisfied and solve LPA every time a new flow arrives or leaves.
- Online LPA (OLPA) maintains only the arrived flows that can still be satisfied and solve ILPA every time a new flow arrives.

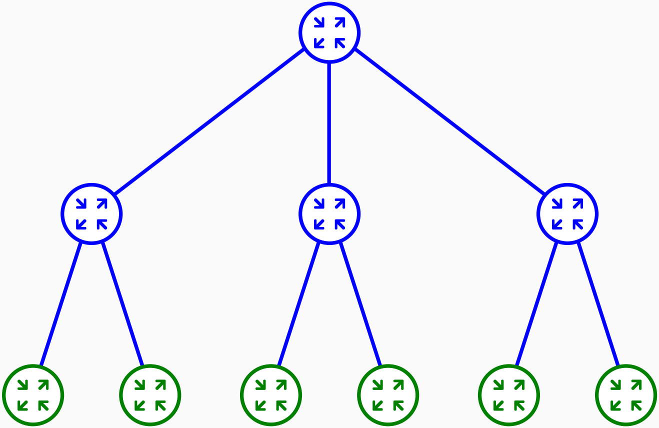
Simulations

- Flow level simulations consider only the theoretical performance.
- Packet level simulations take into account the real network features such as propagation delay and packet overhead.

Flow Level Simulation

- The network flows arrive according to Poisson processes between each source-destination pair with the arrival rate λ sampled uniformly from different intervals.
- The sizes of the network flows is distributed over $[0, \hat{s}]$.
- The algorithms LPA and ILPA are compared with best-effort solution to the MILP (running COIN-OR branch and bound MILP solver with 10 minute timeout).

Flow Level Simulation

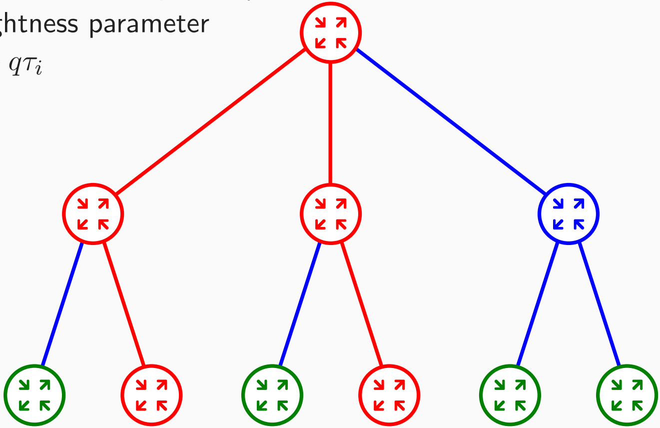


Flow Level Simulation

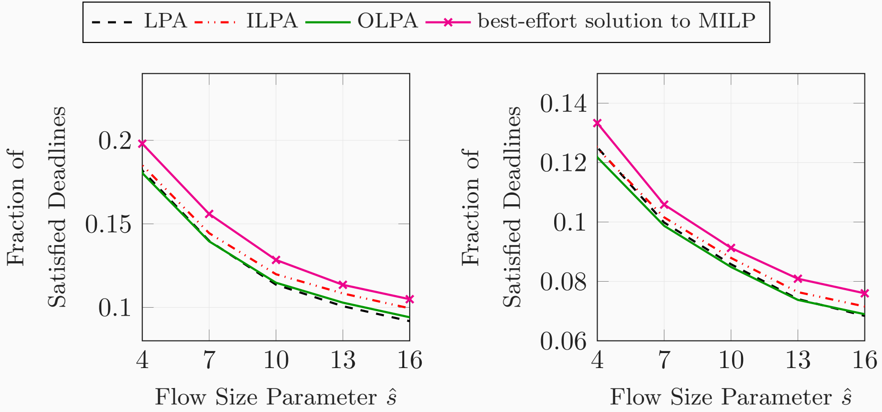
τ_i: the minimum life span of f_i

q: the tightness parameter

$d_i = r_i + q\tau_i$



Flow Level Simulation

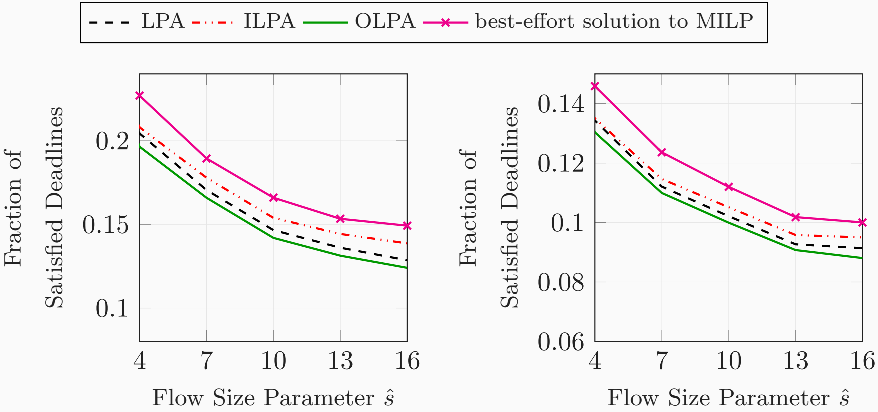


(a) Arrival rate $\lambda \in (0, 2)$.

(b) Arrival rate $\lambda \in (0, 4)$.

Figure 1: Tightness parameter $q = 1$.

Flow Level Simulation



(c) Arrival rate $\lambda \in (0, 2)$.

(d) Arrival rate $\lambda \in (0, 4)$.

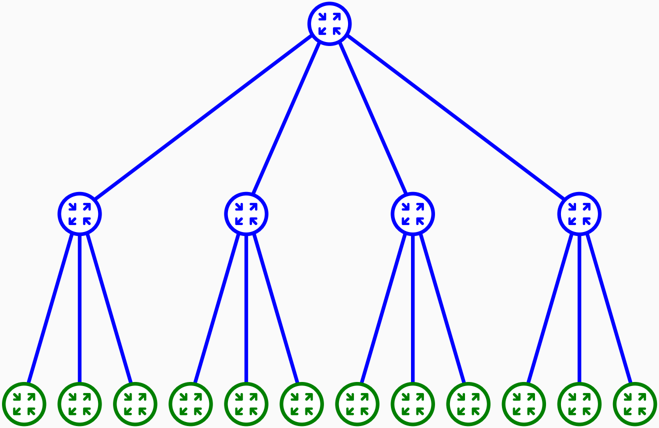
Figure 1: Tightness parameter $q = 2$.

Packet Level Simulation

We focus on two scenarios:

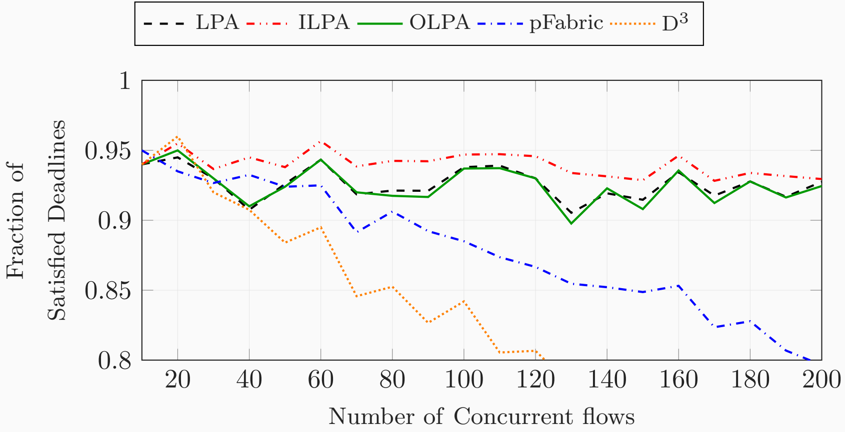
- Request burst scenario: when several source-destination pairs request to send traffic through the network at the same time (service surge).
- Stochastic demand scenario: when traffic enters the network as some random processes (normal network operation).

Packet Level Simulation



Packet Level Simulation: Request Burst

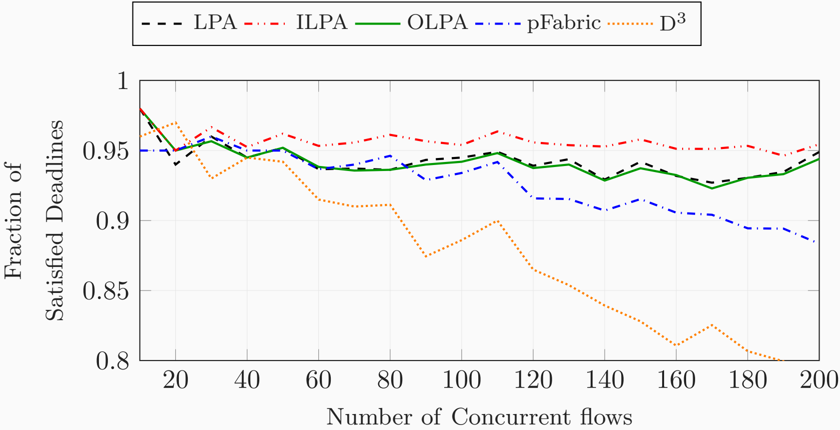
- At time 0, a number of concurrent flows request to send traffic with a random deadline, e.g., 100 flows all arrive at time 0 while their sizes and deadlines are randomly generated.
- We compare our methods with $\mathrm{pFabric}$ and D^3. $\mathrm{pFabric}$ prioritizes the flows based on their remaining time to their deadlines. D^3 applies the FCFS greedy strategy and quenches the expired flows.



(a) The mean of the generated deadlines is 2 ms.

Figure 2: The fraction of satisfied deadlines under request burst scenario.

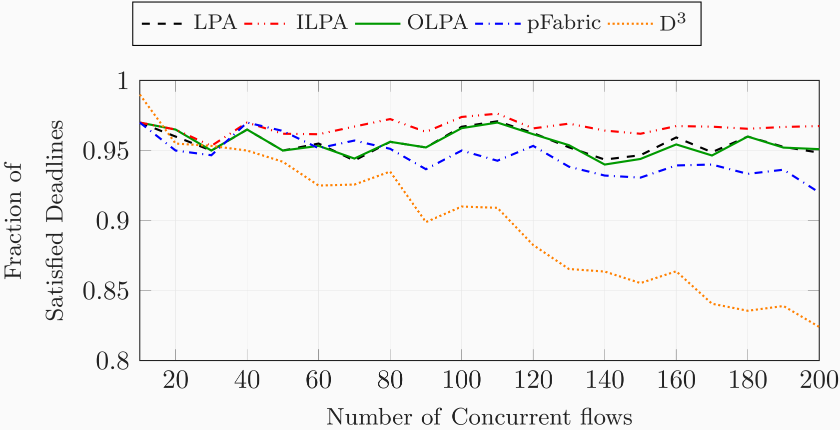
Packet Level Simulation: Request Burst



(b) The mean of the generated deadlines is 3 ms.

Figure 2: The fraction of satisfied deadlines under request burst scenario.

Packet Level Simulation: Request Burst



(c) The mean of the generated deadlines is 4 ms.

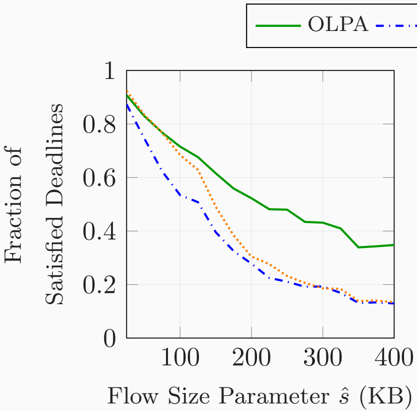
Figure 2: The fraction of satisfied deadlines under request burst scenario.

Packet Level Simulation: Stochastic Demand

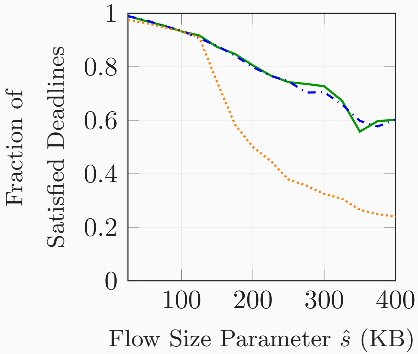
- Each source-destination pair of the hosts in the network generate flows according to a Poisson process with the interarrival time uniformly distributed over $(0, 50]$ ms.
- Each flow has a size uniformly distributed over $(0, \hat{s}]$ KB.
- The deadline is set according to the tightness parameter q.

As such, less flows can be satisfied as $\hat{s}$ increases (more traffic) or q decreases (tighter deadline).

Packet Level Simulation: Stochastic Demand



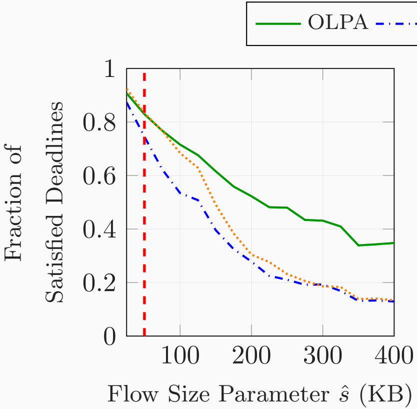
(a) Tightness parameter $q = 1$.



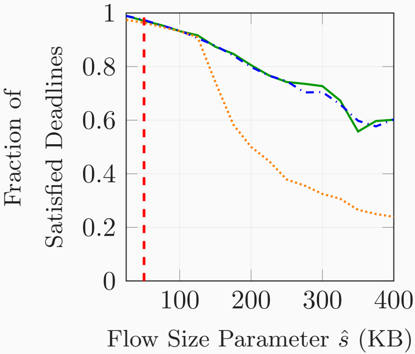
(b) Tightness parameter $q = 2$.

Figure 3: The median of the fraction of satisfied deadlines under stochastic demand scenario.

Packet Level Simulation: Stochastic Demand



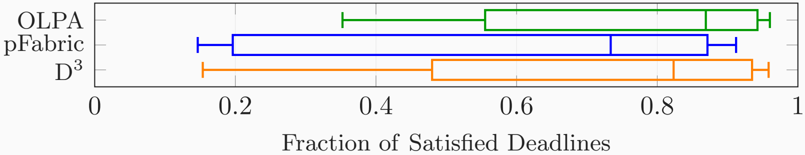
(a) Tightness parameter $q = 1$.



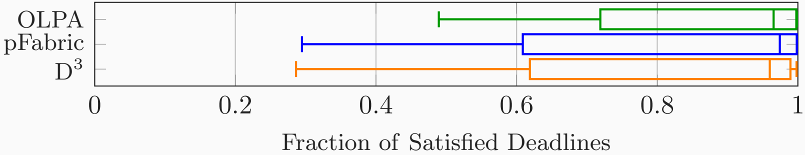
(b) Tightness parameter $q = 2$.

Figure 3: The median of the fraction of satisfied deadlines under stochastic demand scenario.

Packet Level Simulation: Stochastic Demand



(a) Tightness parameter $q = 1$.



(b) Tightness parameter $q = 2$.

Figure 4: The $1^{\text{st}} - 5^{\text{th}} - 50^{\text{th}} - 95^{\text{th}} - 99^{\text{th}}$ percentiles of the fraction of satisfied deadlines when $\hat{s} = 50$ (KB).

Conclusion

- The flow deadline scheduling problem is NP-hard. Moreover, it cannot be approximated within a constant factor in polynomial time (unless $P = NP$).
- We develop optimization-based offline and online algorithms.
- Simulation results show that the proposed algorithms are effective.

Questions & Answers

References

- C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," *ACM SIGCOMM CCR*, vol. 41, no. 4, pp. 50–61, 2011.
- C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM CCR*, vol. 42, no. 4, pp. 127–138, 2012.
- M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 435–446, 2013.

References

CBC (COIN-OR branch and cut). [Online]. Available: https://projects.coin-or.org/Cbc