Efficient Computation of Worst-Case Delay-Bounds for Time-Sensitive Networks

Author: Edin Guso

Advisors: Seyed Mohammadhossein Tabatabaee, Stéphan Plassart, Jean-Yves Le Boudec

Institute: Computer Communications and Applications Laboratory 2 (LCA2), École Polytechnique Fédérale de Lausane (EPFL)

EPFL

Outline

- Introduction
- Solution
- Achievements
- Skills
- Major Events
- Self-Assessment

Introduction

Time-Sensitive Networks

- IEEE Time-Sensitive Networking (TSN) and IETF Deterministic Networking (DetNet)
- Safety-critical applications and deterministic services

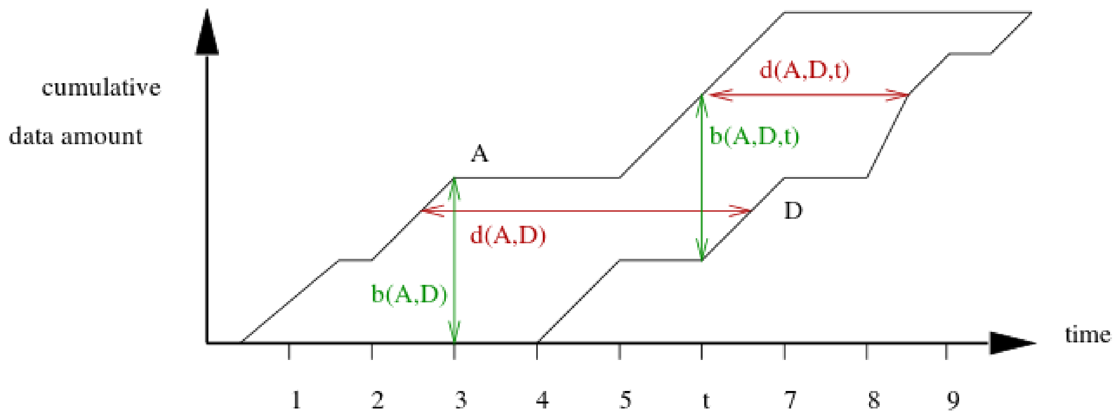
➢ Importance of worst-case delay bounds

Network Calculus

- Mathematical framework for performance analysis

➢Upper bounds on worst-case performance parameters

- End-to-end delay and backlog



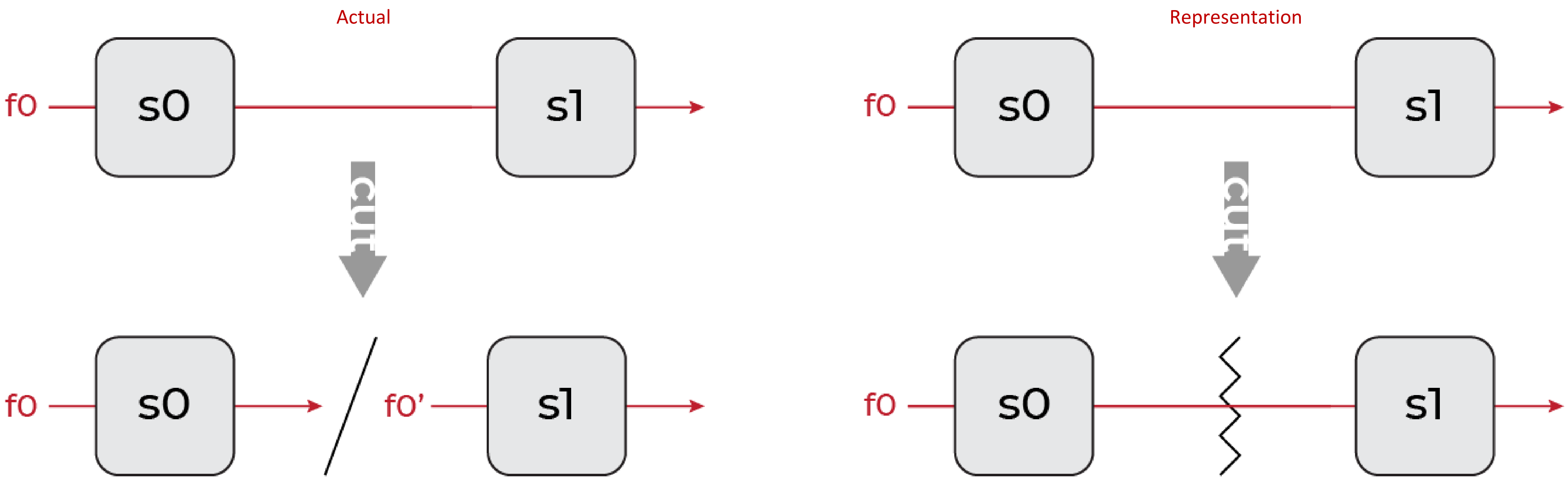
Computing Upper Bounds on Worst-Case Performance

- Complexity and computational intensity
- Therefore, existing approaches rely on heuristics
- One such example is the Polynomial-size Linear Programming (PLP)

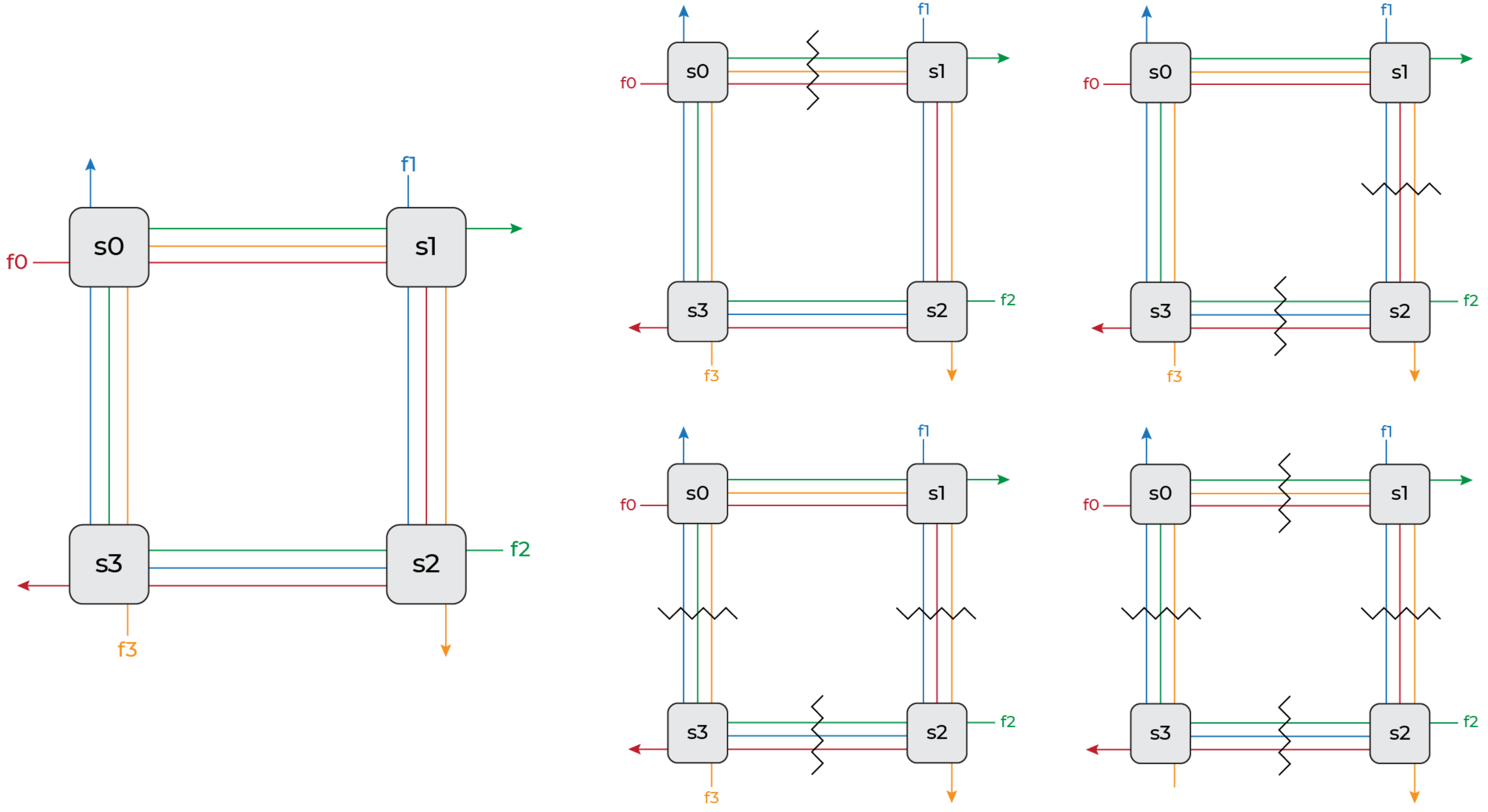
Polynomial-size Linear Programming

- Leveraging linear programming techniques
- Works by cutting the network to achieve a forest
- Importance of cut selection

Cutting a Network



Valid Cuts of a Network



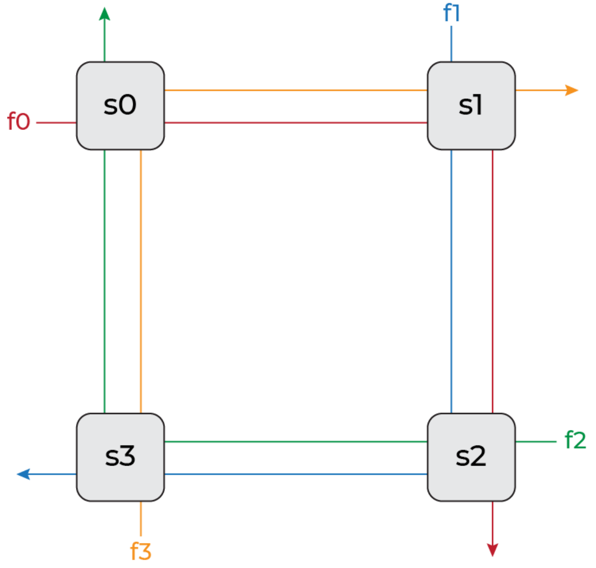
Problem Definition

- Problem: Simplistic cut selection in the original PLP algorithm resulting in sub-optimal worst-case delay bounds
- Process: Investigating the relationship between cut size, shape, and composition
- Goal: Developing efficient heuristics for selecting cuts that achieve accurate worst-case delay bounds

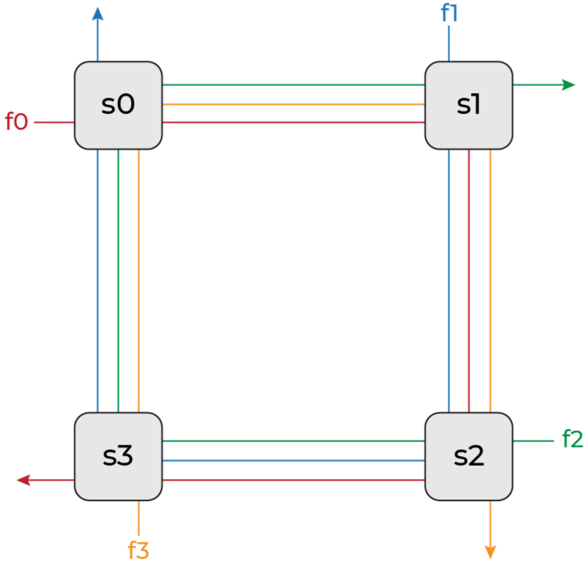
Solution

Performing Exhaustive Search on Small Networks to Obtain Insights

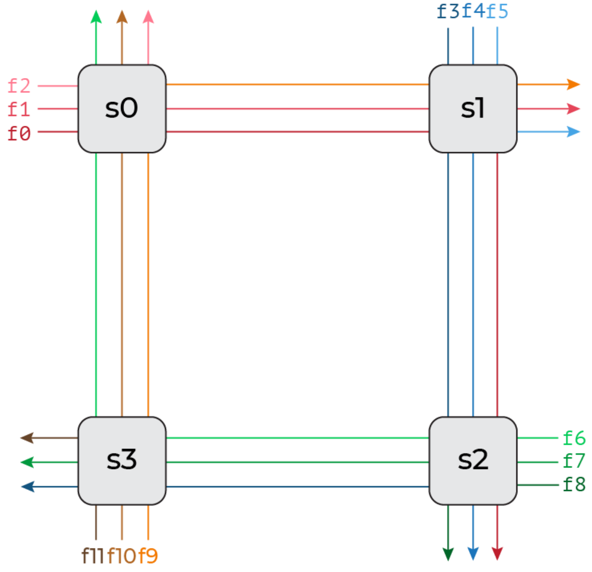
Semi Ring Network (N=12)



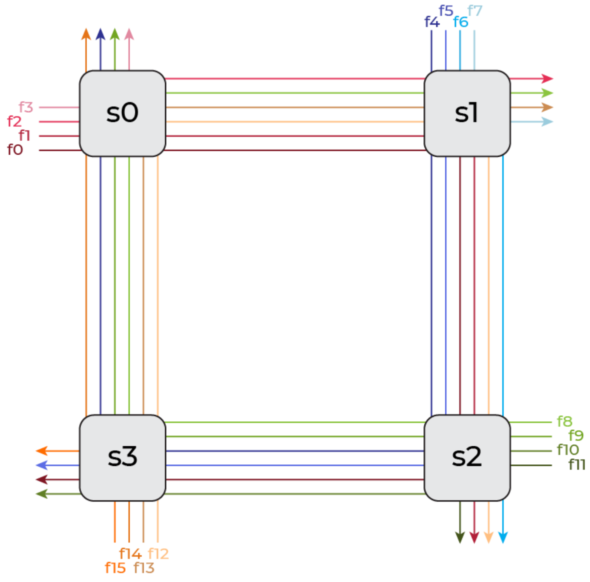
Full Ring Network (N=12)



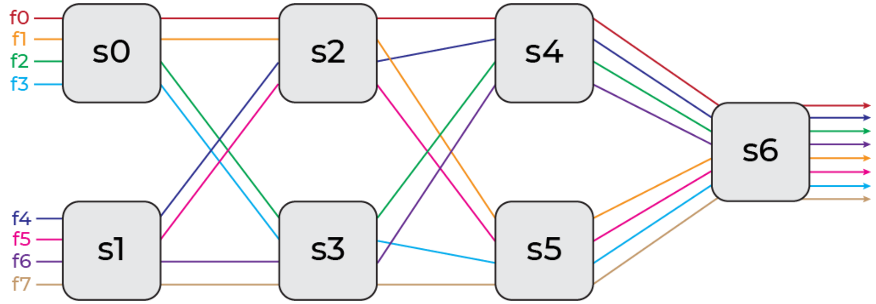
Complete Semi Ring Network (N=11)



Complete Full Ring Network (N=7)

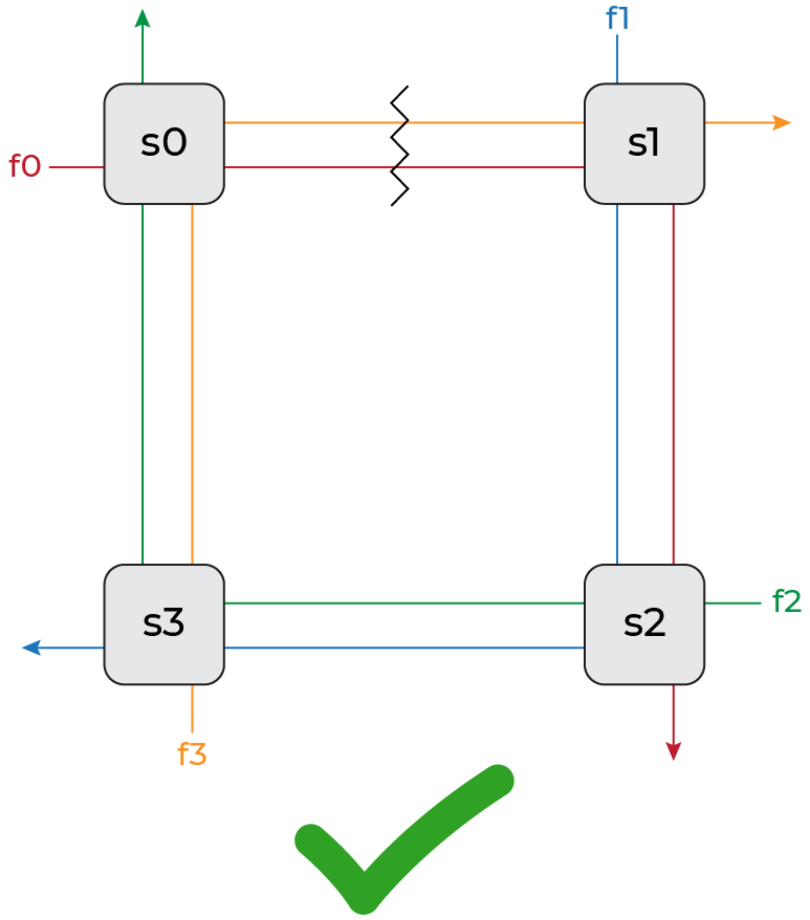


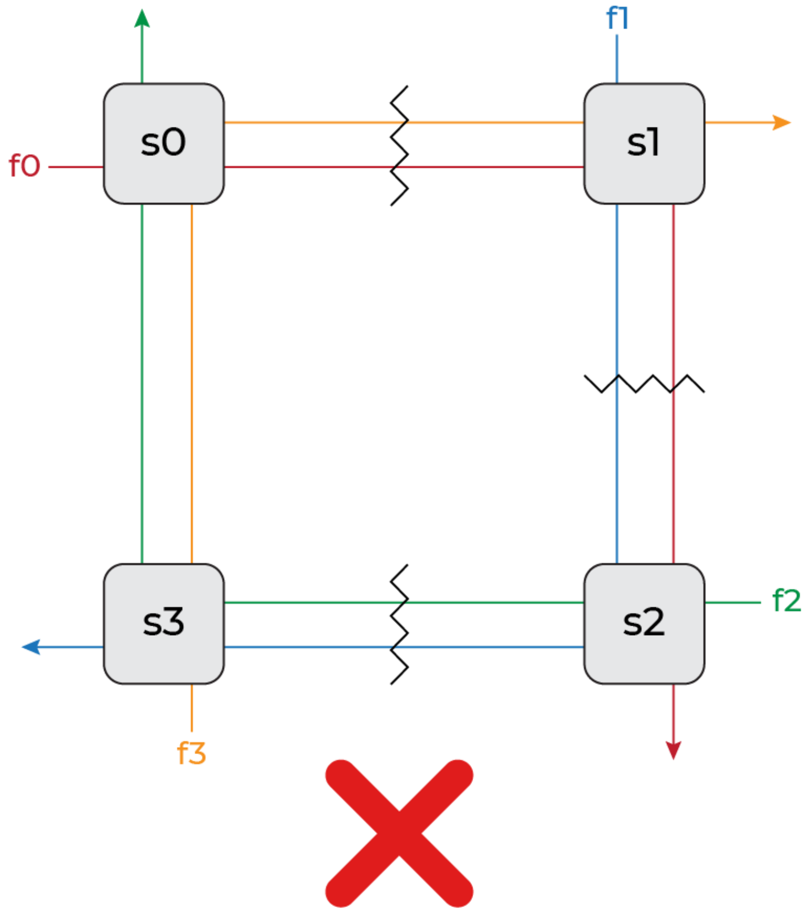
Mesh Network (N=9)



Performing Smaller Cuts Improves Worst-Case Delay Bounds

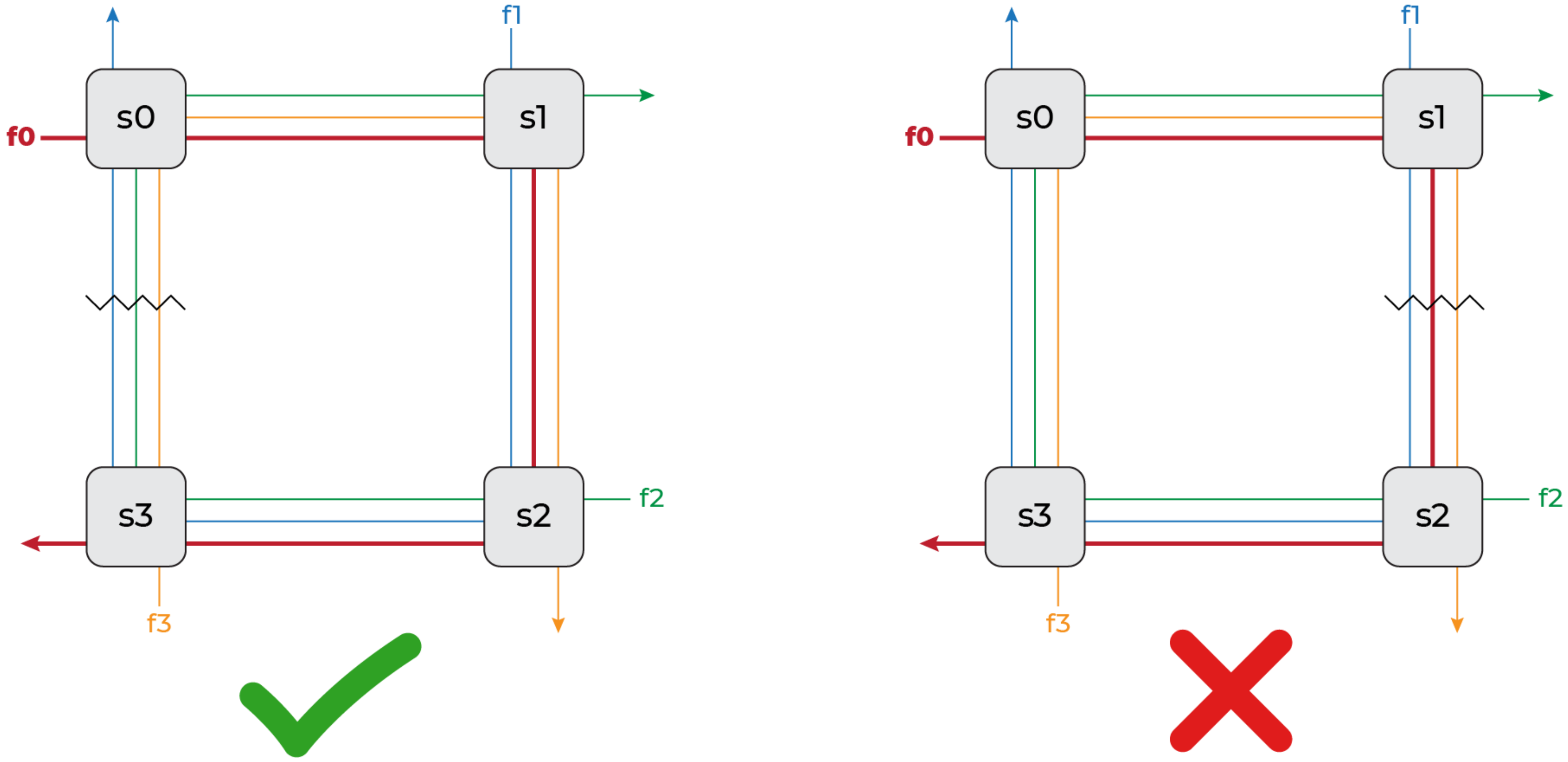
Emphasis on Selecting Cuts with Smaller Impact on the Network





Not Cutting the Flow of Interest Improves Worst-Case Delay Bounds

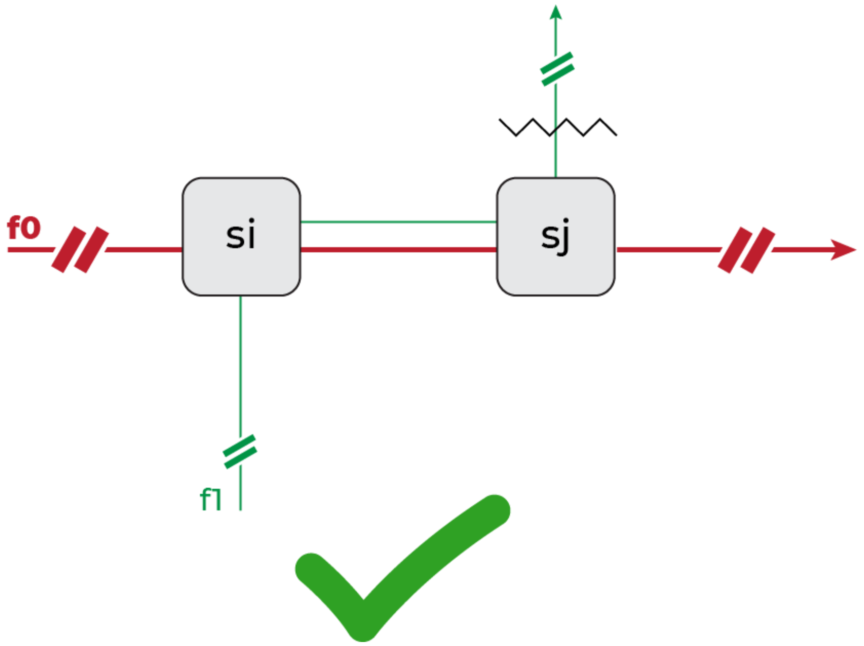
Prioritizing Cuts That Avoid Interrupting the Flow of Interest

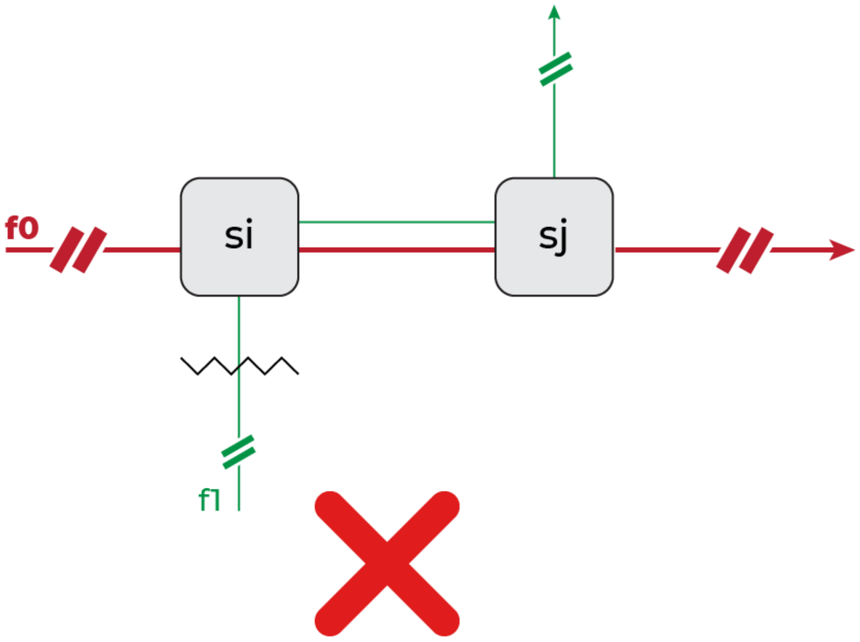


Supporting Observation

- Preserve flows that overlap with the flow of interest

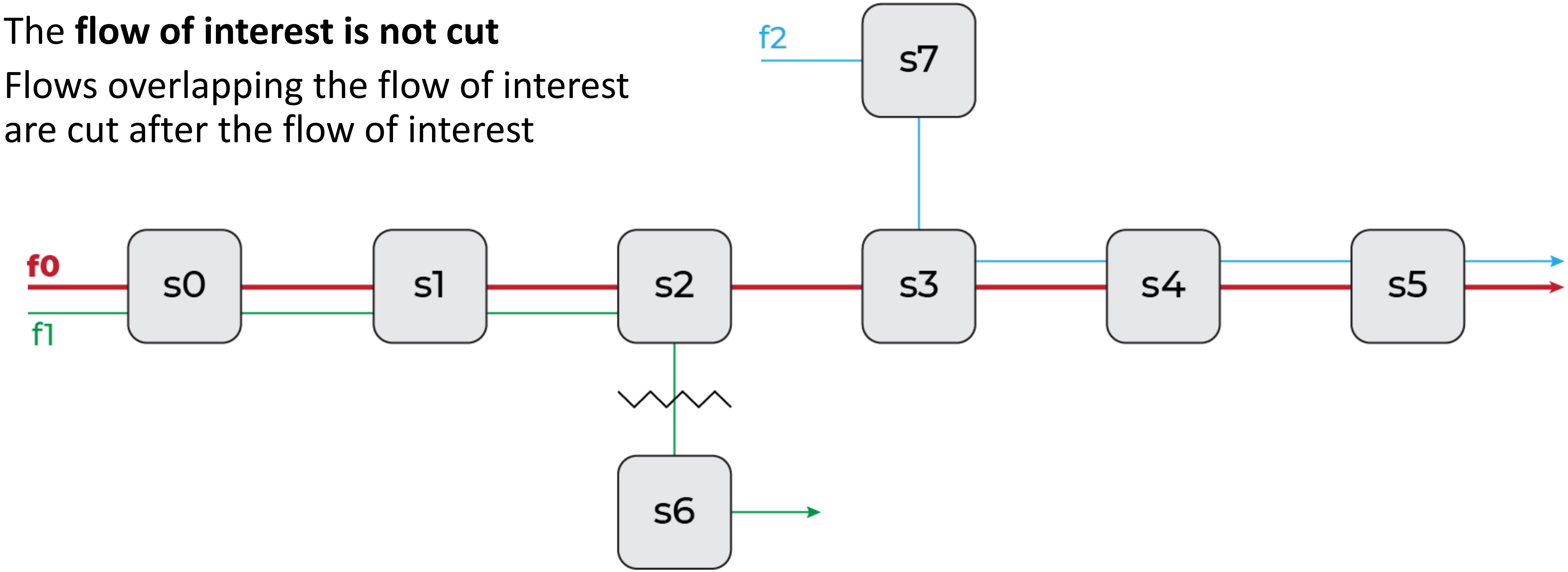
➢Positioning cuts after the overlap or towards the start of the flow



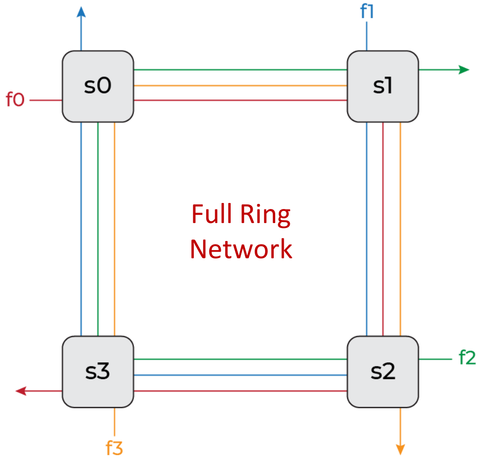


Min-Cut Forest (MCF) Algorithm

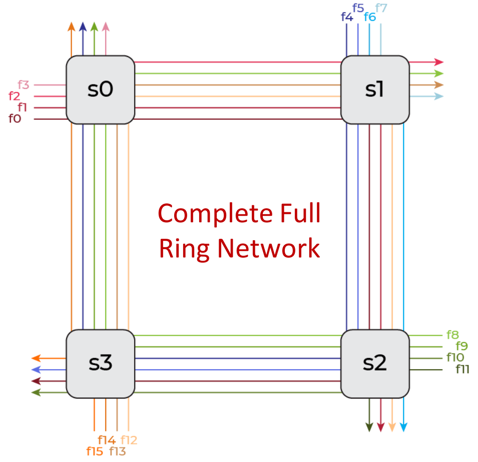
- Returns the **minimum cut** such that
  - The **flow of interest is not cut**
  - Flows overlapping the flow of interest are cut after the flow of interest



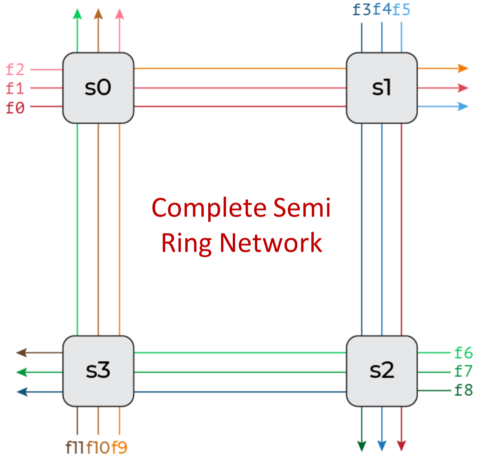
Min-Cut Forest Algorithm Achieves Optimal or Near-Optimal Results



[12 Servers, 12 Flows]
Exhaustive Search: 149.13µs
Min-Cut Forest: 149.13µs

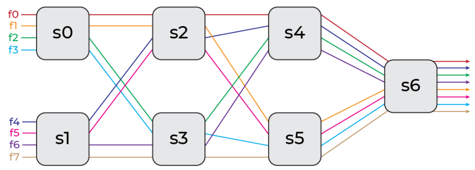


[7 Servers, 49 Flows]
Exhaustive Search: 139.18µs
Min-Cut Forest: 139.27µs



[11 Servers, 66 Flows]
Exhaustive Search: 109.17µs
Min-Cut Forest: 109.17µs

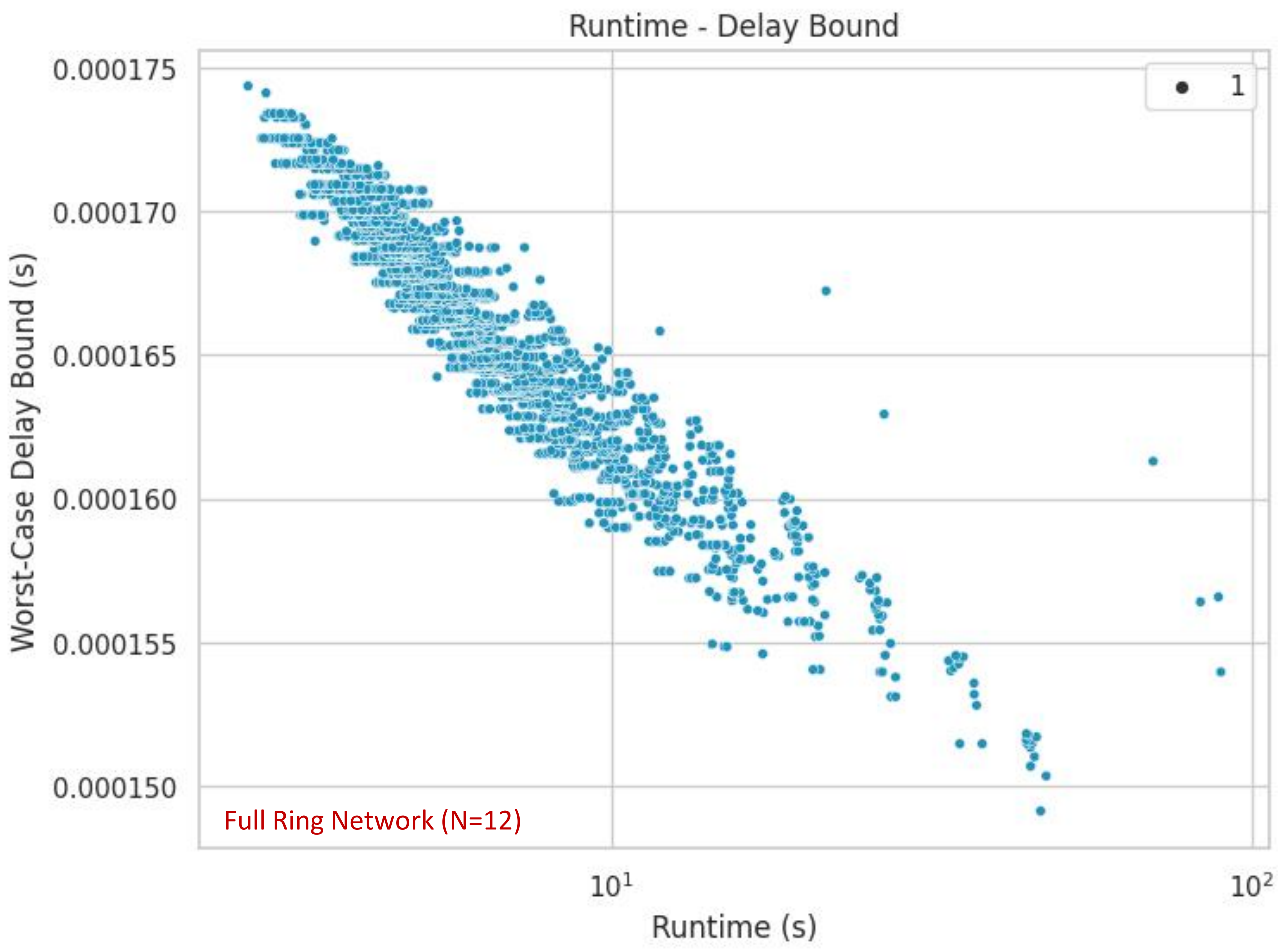
Mesh Network



[9 Servers, 16 Flows]
Exhaustive Search: 89.25µs
Min-Cut Forest: 98.39µs

Runtime Considerations of the PLP Algorithm

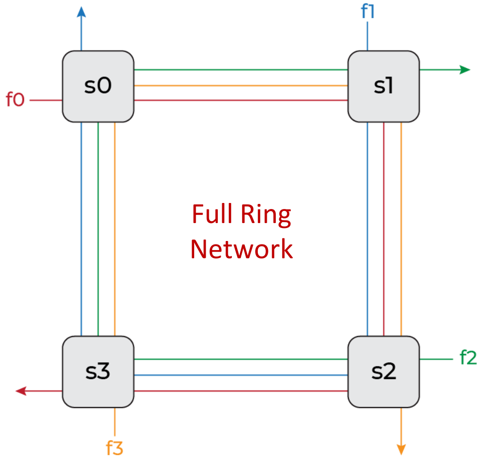
- Importance of runtime in practical applicability
- Relationship between size of the cut and runtime
- Trade-off between delay bounds and computational efficiency



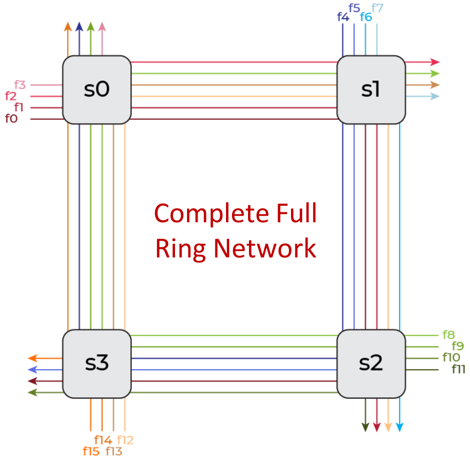
Introducing More Cuts

- PLP solves more, smaller linear programs
- Impact of cutting the network on runtime
- Considering the trade-off based on network size

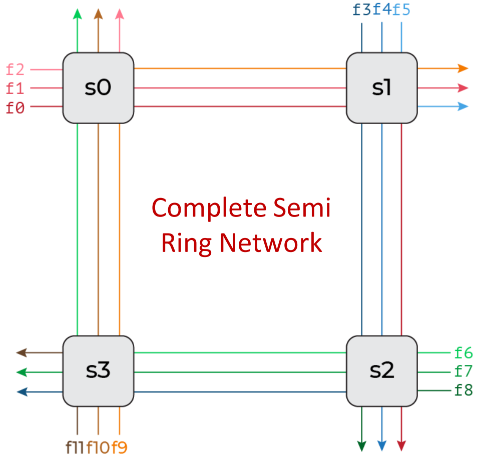
Significant Runtime Increases in Computing Delay Bounds Using the Min-Cut Forest



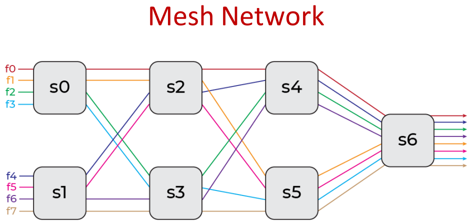
[24 Servers, 24 Flows]
Runtime: 2h 38m 19s



[8 Servers, 81 Flows]
Runtime: 35m 31s



[15 Servers, 120 Flows]
Runtime: 1h 39m 13s

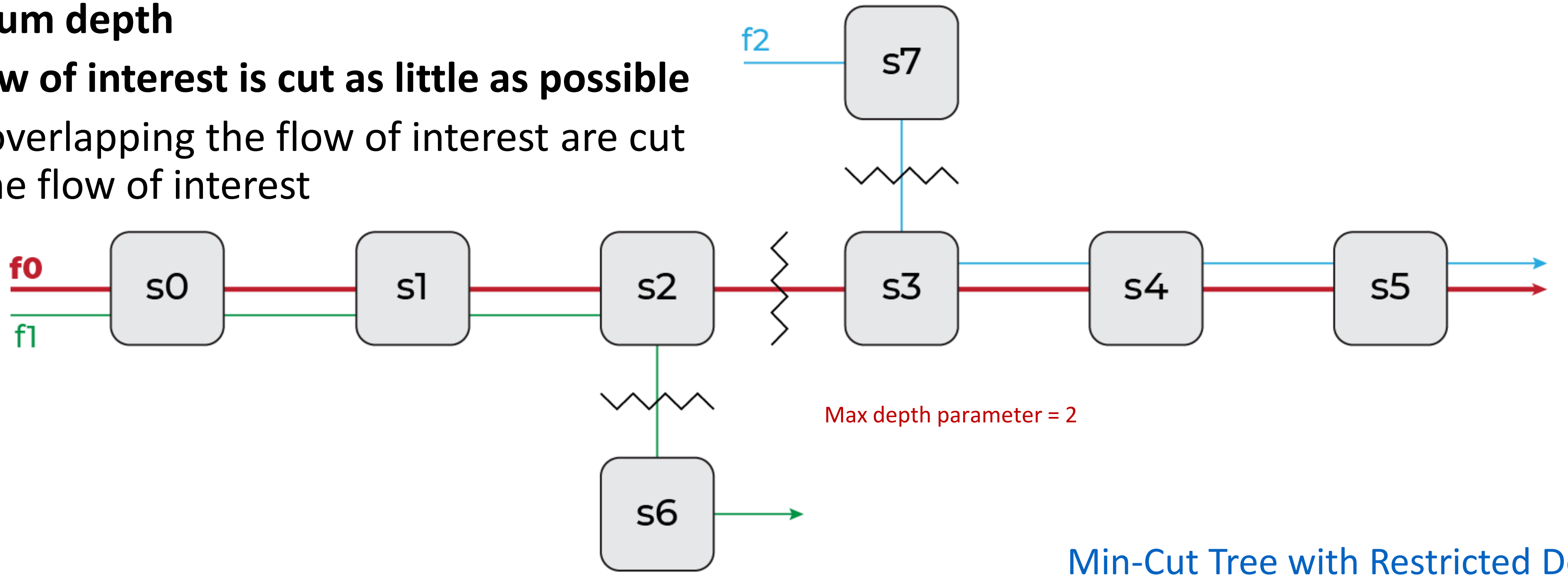


[17 Servers, 256 Flows]
Runtime 52m 6s

System Specs

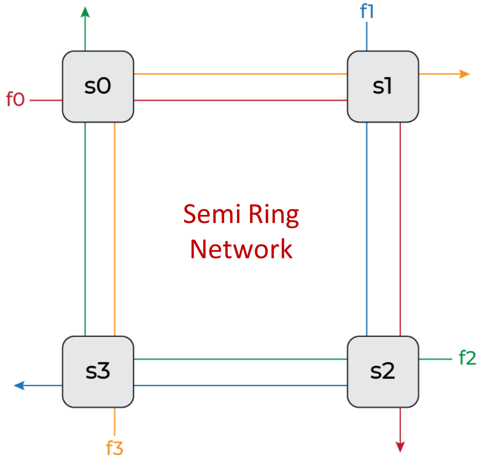
Min-Cut Forest with Restricted Depth (MCFr) Algorithm

- Returns the **minimum cut** such that
  - No server in the network **exceeds the maximum depth**
  - The **flow of interest is cut as little as possible**
  - Flows overlapping the flow of interest are cut after the flow of interest

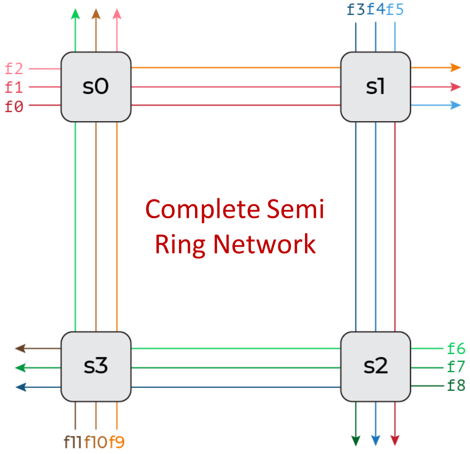


[Min-Cut Tree with Restricted Depth]

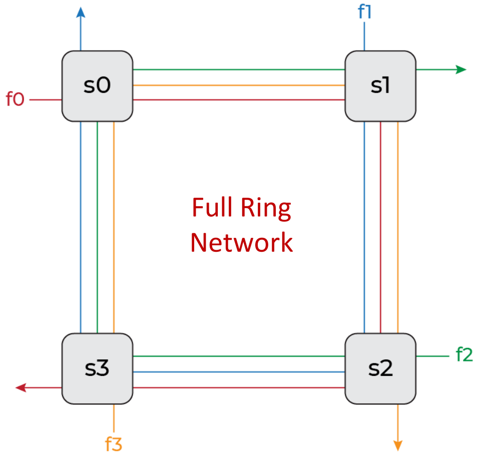
MCFr Algorithm Achieves Close Results to MCF Algorithm But in Considerably Less Time



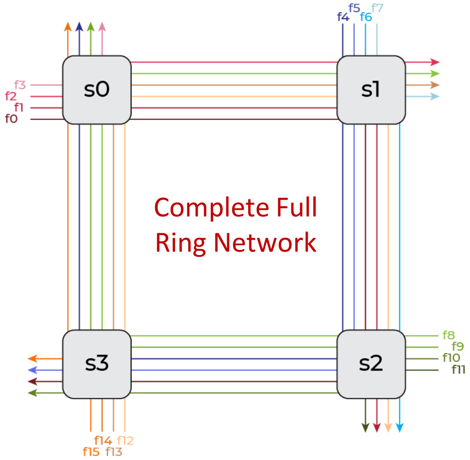
[24 Servers, 24 Flows]
Min-Cut Forest:
Delay: 158.98µs
Runtime: 19m 43s
MCFr (max depth = 12):
Delay: 161.91µs
Runtime: 4m 8s



[15 Servers, 120 Flows]
Min-Cut Forest:
Delay: 169.12µs
Runtime: 1h 39m 13s
MCFr (max depth = 8):
Delay: 170.19µs
Runtime: 28m 38s



[24 Servers, 24 Flows]
Min-Cut Forest:
Delay: 301.11µs
Runtime: 2h 38m 19s
MCFr (max depth = 12):
Delay: 312.40µs
Runtime: 40m 35s

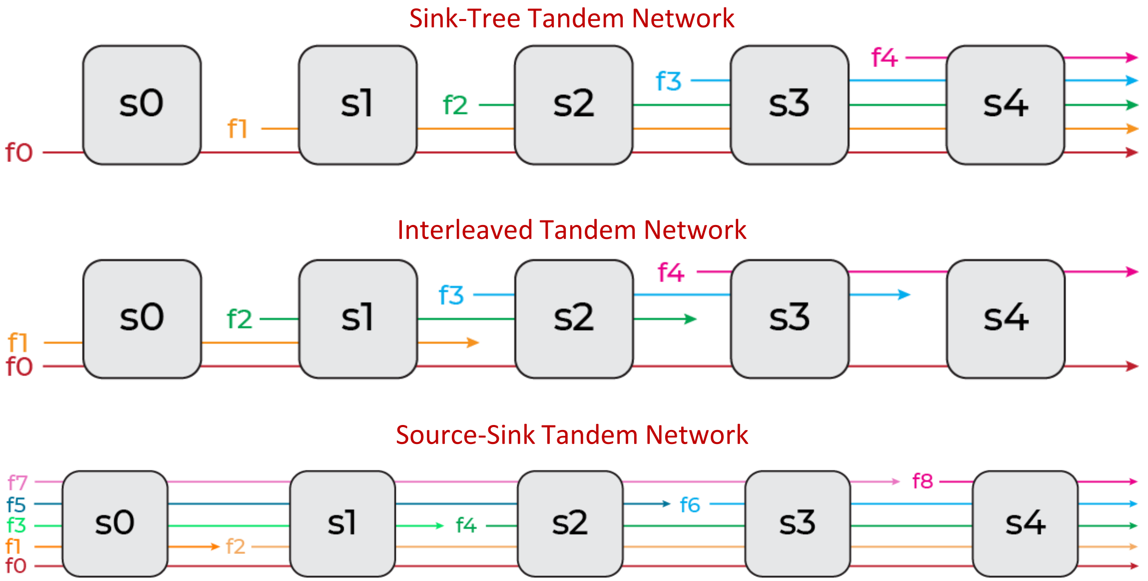


[9 Servers, 81 Flows]
Min-Cut Forest:
Delay: 205.86µs
Runtime: 35m 31sm
MCFr (max depth = 5):
Delay: 211.04µs
Runtime: 13m 2s

System Specs

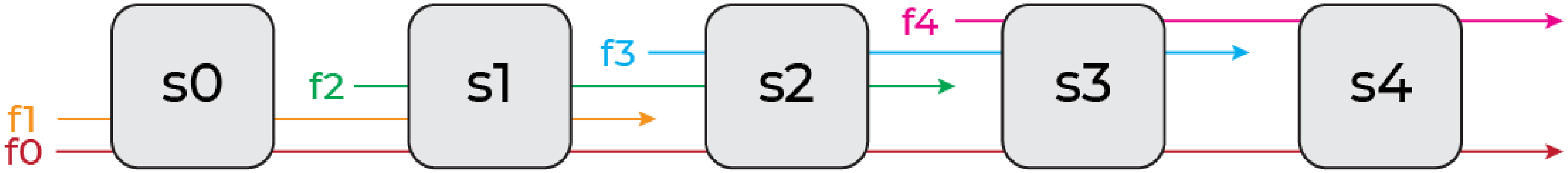
Importance of Cutting in Tree Networks

- Value of cutting networks beyond non-forest networks
- Applying cutting in tree networks like tandems
- Benefits of decreased runtime



MCFr Displays Significant Runtime Benefits in Tandem Networks Compared to MCF

Interleaved Tandem Network



[128 Servers, 128 Flows]

Min-Cut Forest:

Delay: 2025.24µs

Runtime: 34m 13s

Min-Cut Forest with Restricted Depth (max depth = 64):

Delay: 2117.89µs

Runtime: 3m 54s

System Specs

Further Results and Discussion

- Results and Discussion

Achievements

Major Achievements

Insights into Cut Selection

Development of a Heuristic Algorithm

Improved Worst-Case Delay Bounds

Runtime Considerations

Comprehensive Experimental Evaluation

Practical Implementation

Implementation in Python with modifications to existing codebase

ECOWCDB library and tools development

>2500 lines of code, ~500 lines high value code

GitHub Repository

Skills

Skills That I Exercised Throughout the Project

Research

Software Engineering

Graph Theory

Algorithm Design

Version Control

Skills That I Had to Acquire for the Project

Understanding the Basics of Network Calculus

Project Setup and Extension

Advanced Knowledge of Graph Theory, Networks, and Cuts

Documentation

Major Events

Major Events in Chronological Order

- Overcoming Initial Technical Challenges (Weeks 2-5)
- Integrating PLP Algorithm and Initiating Exhaustive Search (Weeks 6-8)
- Efficient Generation of Generic Network Topologies (Week 9)
- Partial Search (Week 14)
- Overcoming Challenges with lp_solve (Weeks 9-15)
- Finalizing the Heuristic Algorithm (Week 16)

Self-Assessment

Successes in the Project

- Showcasing problem-solving abilities
- Delivering high-quality solutions
- Strong algorithm design skills
- Well-structured repository
- Thorough documentation

Challenges Encountered and Lessons Learned

- Difficulties in project setup
  - Exploring alternative solutions
  - Adapting and trying different approaches
- Developing incorrect or unnecessary solutions
  - Importance of thorough research
  - Seeking clarification when needed
  - Enhancing theoretical understanding

References

- Boudec, J.-Y. L. and Thiran, P. (2001). Network calculus: A theory of deterministic queuing systems for the internet. Springer.
- Bouillard, A., Boyer, M., & Corronc, E. L. (2018). Deterministic Network Calculus: From Theory to Practical Implementation. John Wiley & Sons.
- Bouillard, A. (2022). Trade-off between accuracy and Tractability of network calculus in FIFO networks. Performance Evaluation, 153, 102250. https://doi.org/10.1016/j.peva.2021.102250
- https://github.com/Huawei-Paris-Research-Center/panco

Thank you