# Online and Global Network Optimization with SDN

Jérémie Leguay

**Traffic and Network Optimization Team**

Mathematical and Algorithmic Sciences Lab
Huawei Technologies, Paris

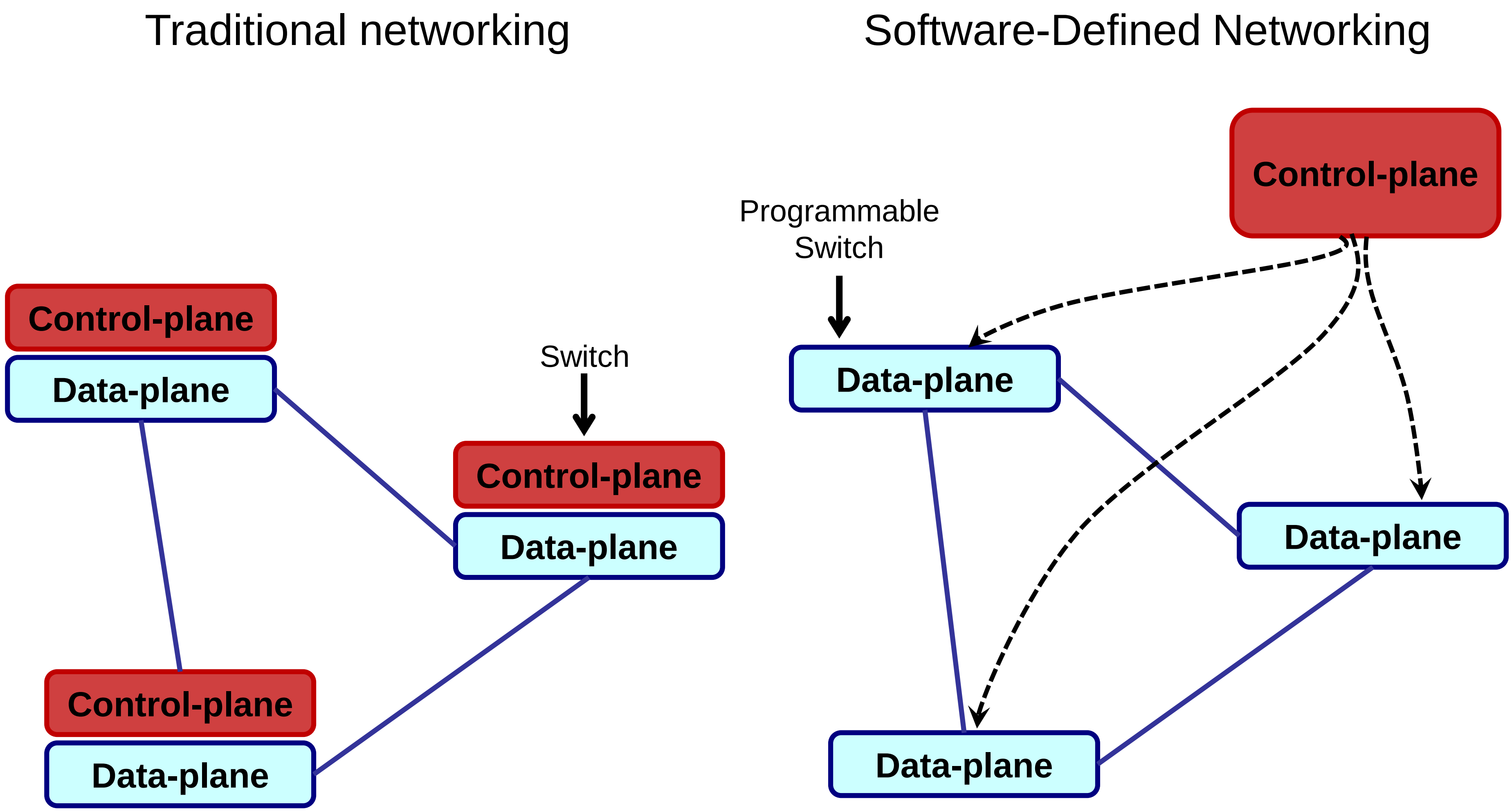


June 2017

# Outline

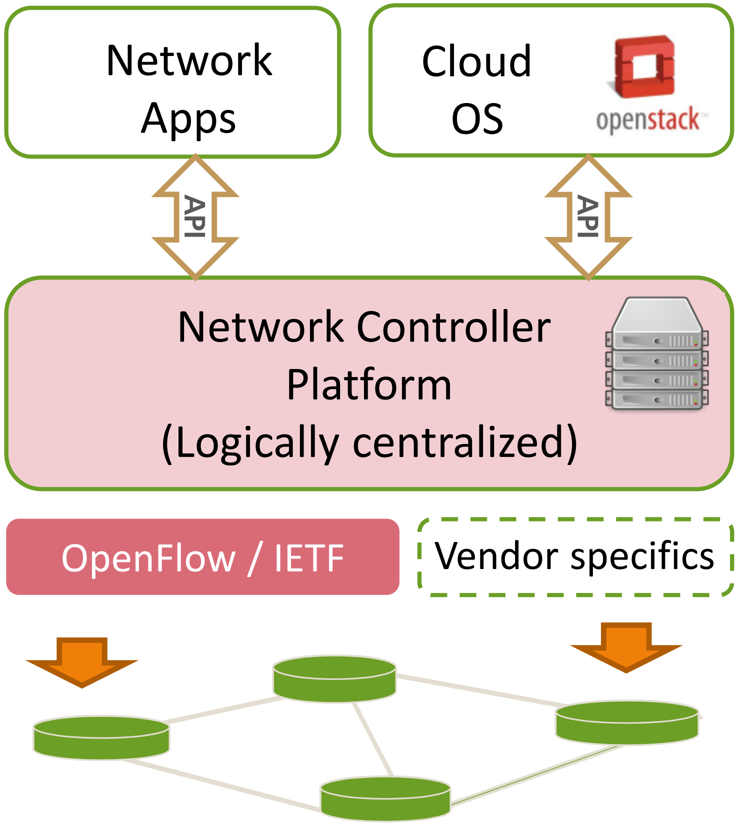
- **Introduction on SDN**
- **Path computation algorithms**
- **Network optimisation: offline algorithms**
- **Network optimisation: online algorithms**

# The (new) paradigm: SDN



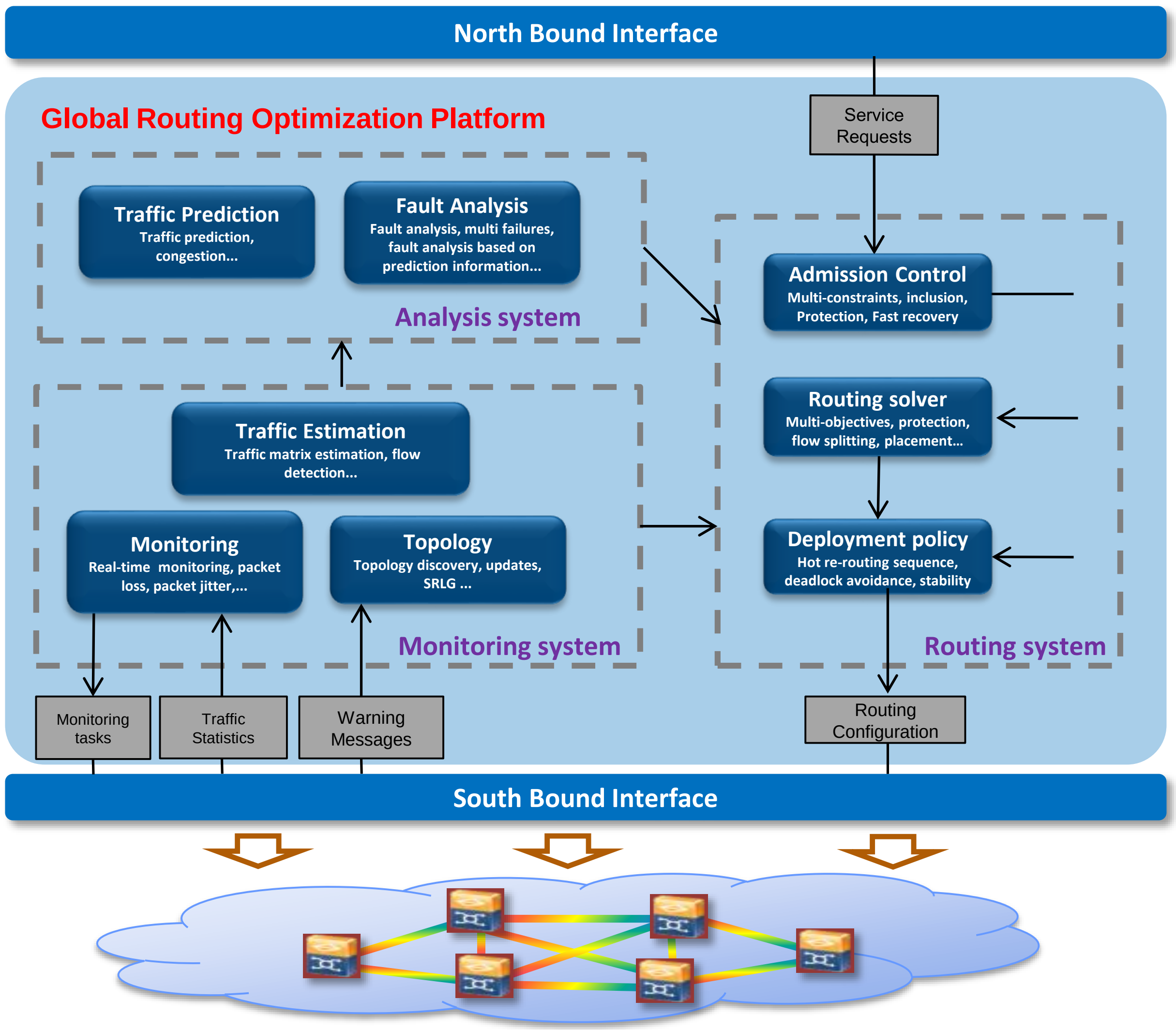
# Global and Online Network Optimization in SDN

- Main properties of SDN / PCE
  - **Offload** the control plane to (powerful) external x86 servers
  - Provide **network programmability** through abstractions
- Operational benefits
  - Advanced **automation** O&M **60%**
  - Global **optimization and control** Network efficiency **10 times**
- Huawei solutions
  - Agile Controller, T-SDN



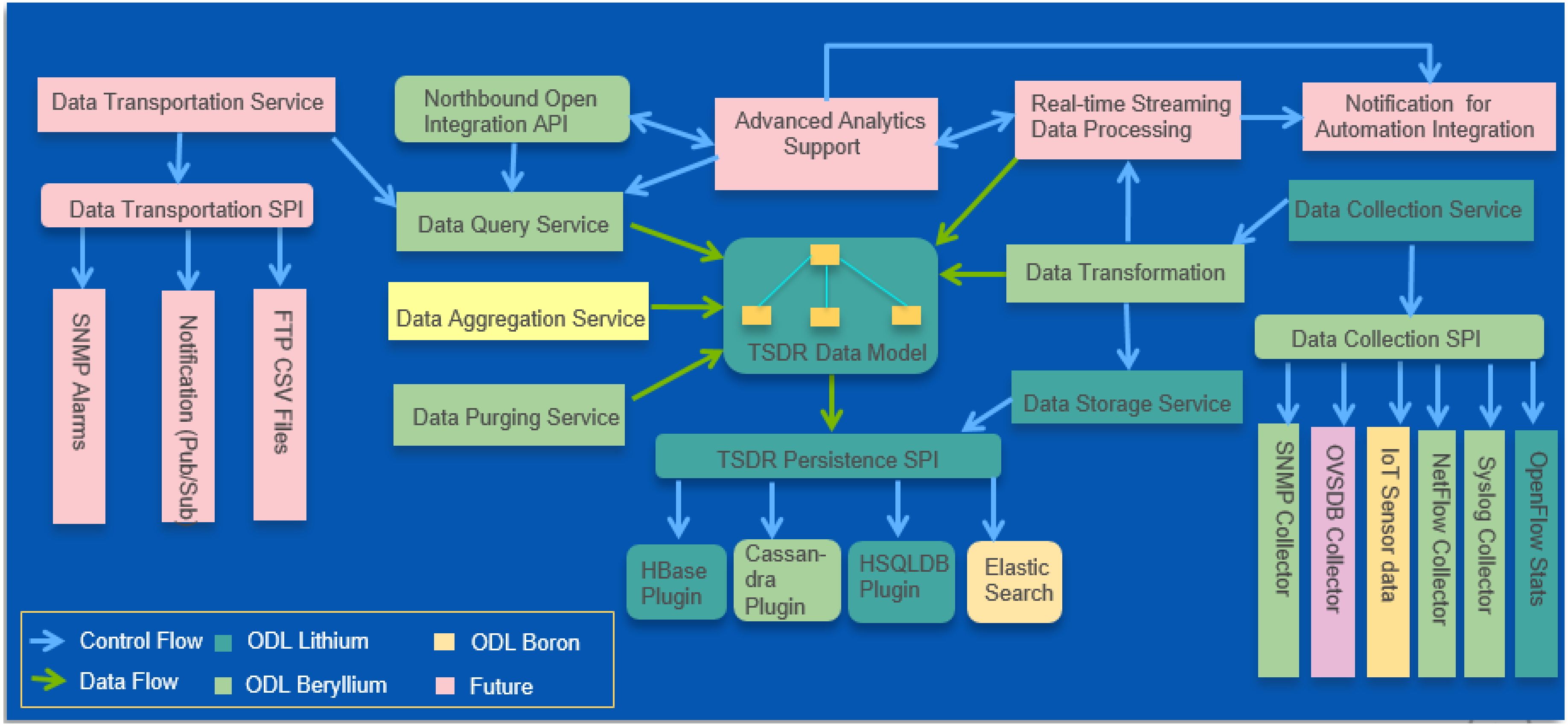
**SDN architectures**

# Routing systems in next generation controllers



# Built-in Machine Learning is coming

## Time Series Data Repository in ODL



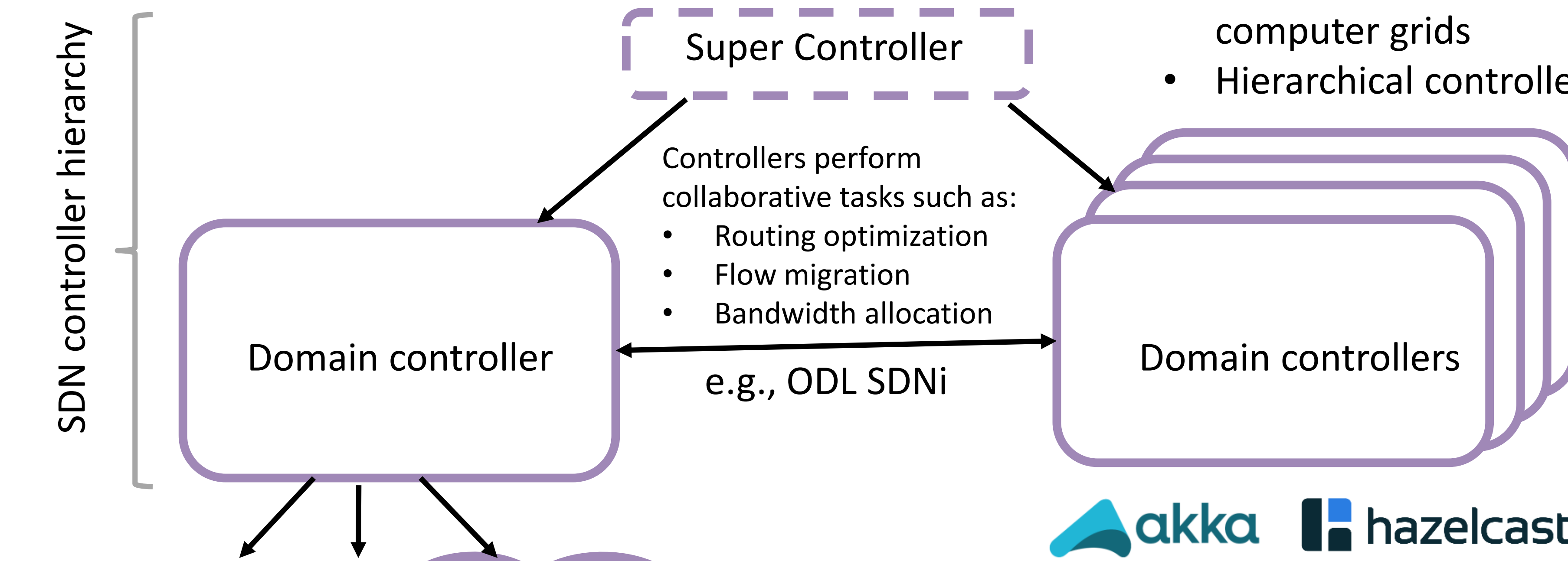

https://wiki.opendaylight.org/view/Project_Proposals:Time_Series_Data_Repository

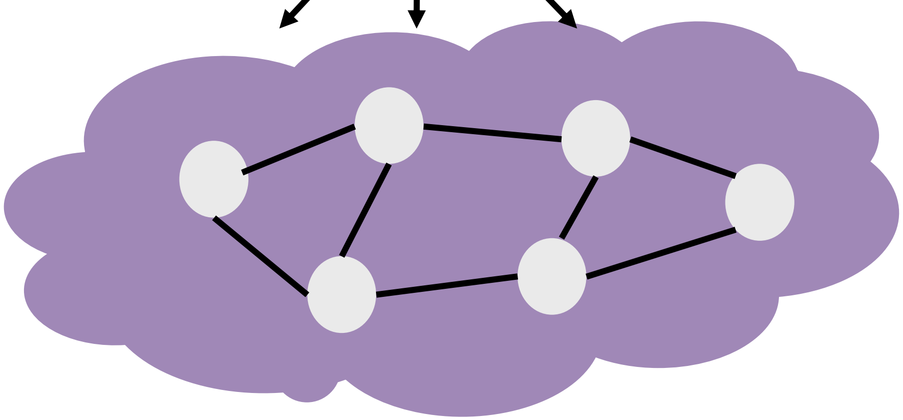
# Built-in distributed / parallel computing

## From routing protocols to distributed routing algorithms

**Architectural features:**

- Multi-domain networks
- Domain controllers are computer grids
- Hierarchical controllers



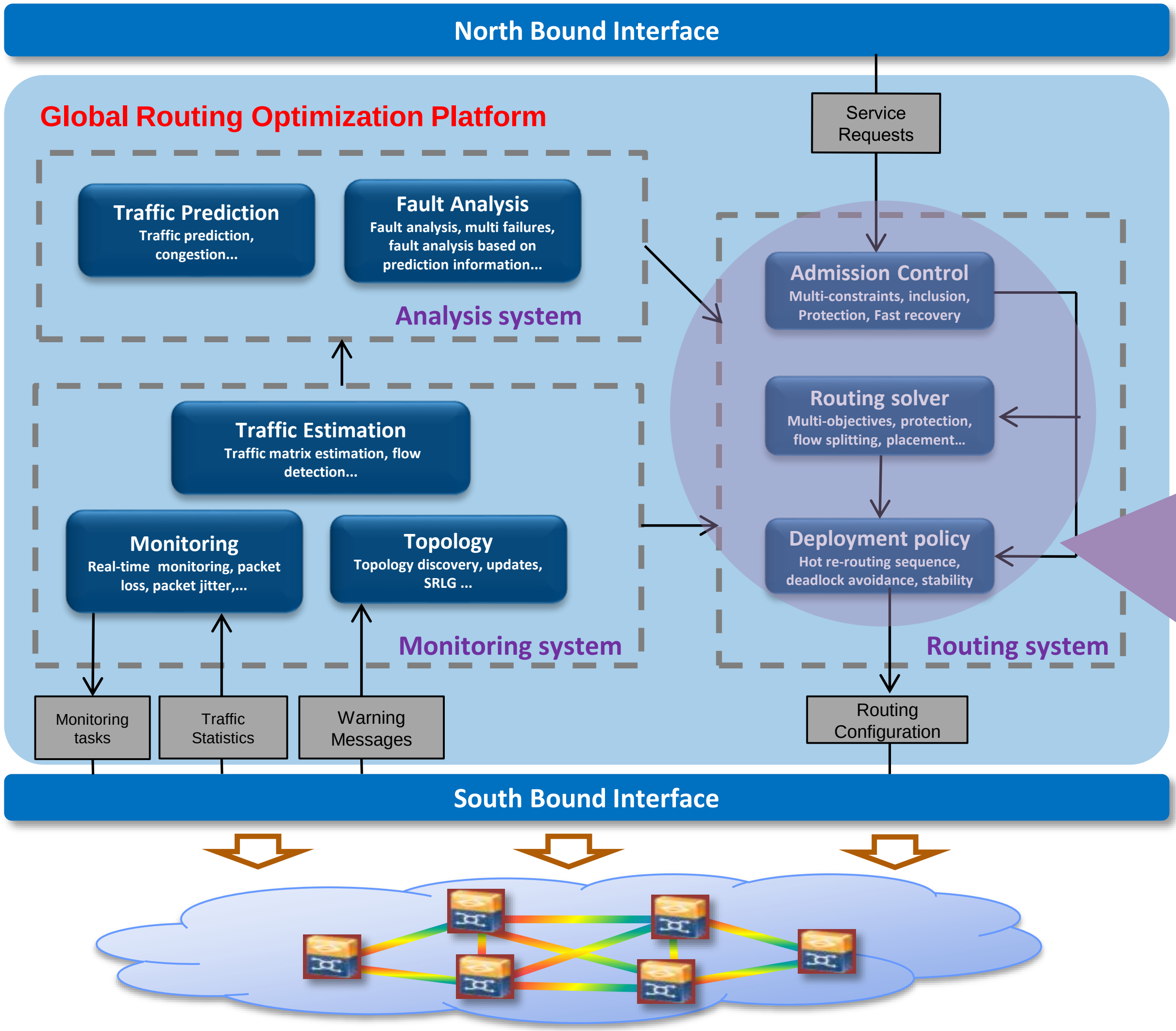



Main challenges:

- Break down optimization problems into many small ones which could be solved in parallel
- Take real-time decisions

https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:Architecture:Clustering

# Routing systems in next generation controllers

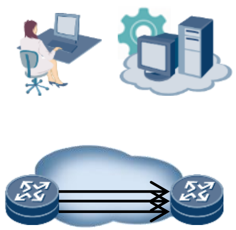



## Our focus so far

- Fast path computation
- Routing optimization
- Robust routing optimization
- Real-time and fair resource allocation
- Admission control

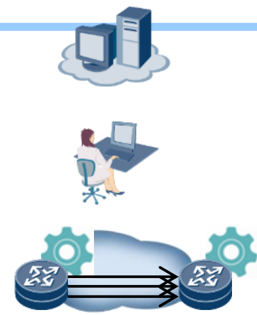
# Intelligence (ML) Defined Network

**Manually** Defined Network



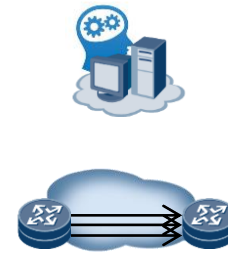
- **Basic approach**
  - Manually plan/ configure/ optimize/ diagnose/ debug network
- **Characteristics**
  - Fully depend on human's experience and knowledge
  - High cost, low efficiency
  - Resource waste
  - Configuration mistakes
  - Network throughput is empirical

**Software** Defined Network



- **Basic approach**
  - Derive/abstract forwarding table from application requirements, use Netconf/YANG or OpenFlow to configure the network forwarding table
- **Characteristics**
  - Semi-autonomic
  - Cost reduced
  - Network throughput is improved, but not yet optimal

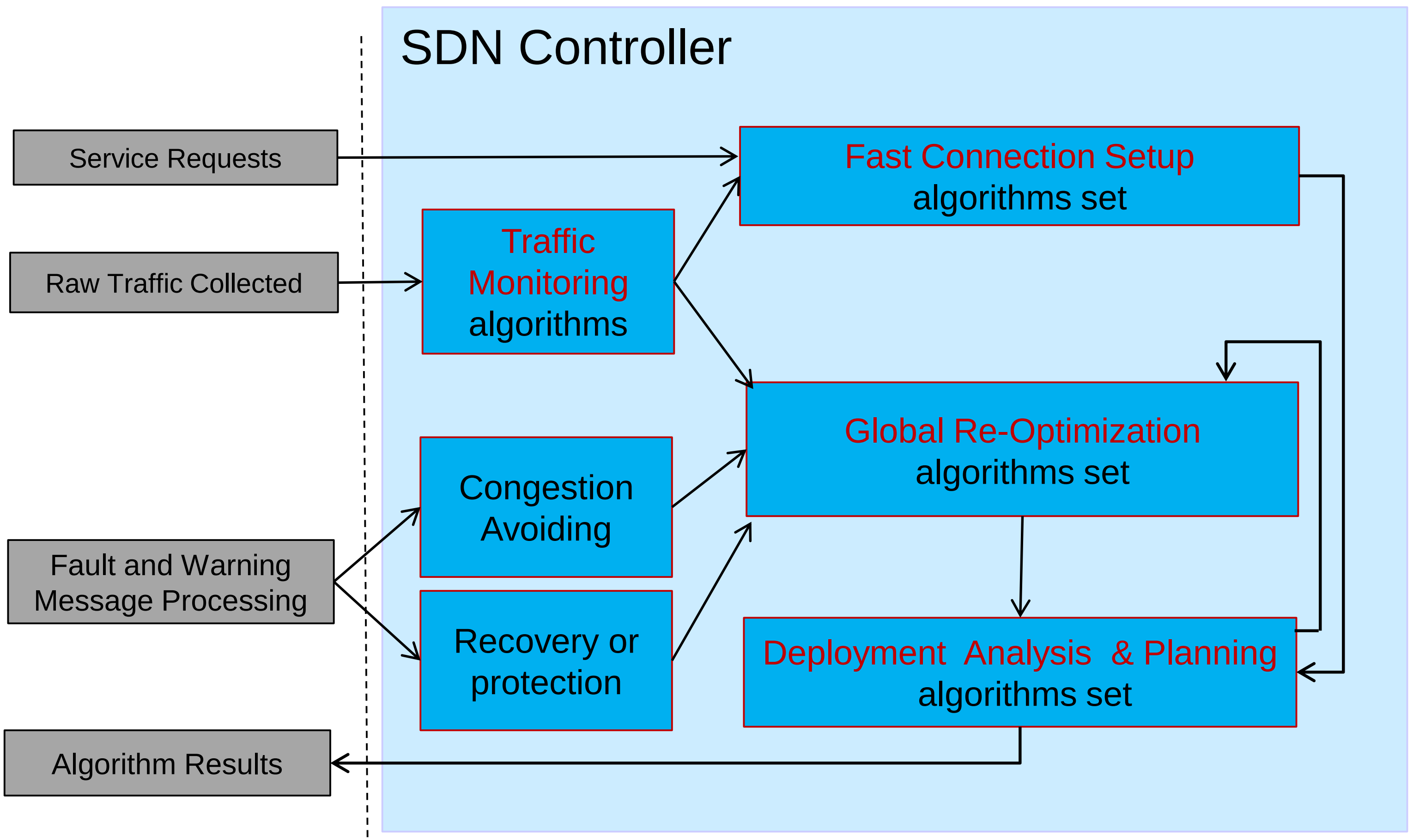
**Intelligence** Defined Network



- **Basic approach**
  - Extensive use of optimization tools
  - Machine learning: learn traffic and application patterns
  - Re-define and drive the network by machine learning technologies.
- **Characteristics**
  - Fully-autonomic
  - Minimum cost
  - Maximum throughput

https://www.ietf.org/mailman/listinfo/idnet

# Algorithmic framework in routing solvers

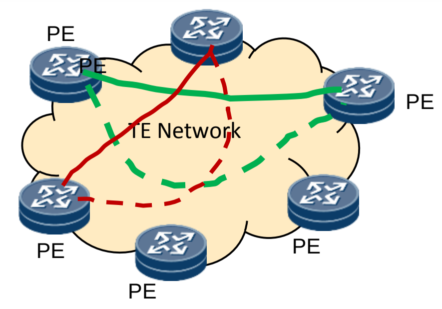



**At large scale and online**

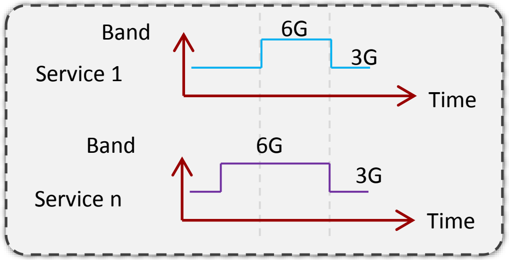
# Routing Problems for SDN "solvers"

- A **large set of requirements** to meet
  - Multi- constraints Path Computation
    - Single, Disjoint paths, SRLGs
  - Network optimization
    - Single layer, Multi-layer (IP + Optical)
  - Point to multi-point (Multicast)
  - Bandwidth calendaring
  - Multipath flow splitting
  - Service Chaining, VNE..
  - Reroute Sequence Planning
- **Leading to hard problems**
  - Path computation, resource allocation, scheduling, placement, etc…

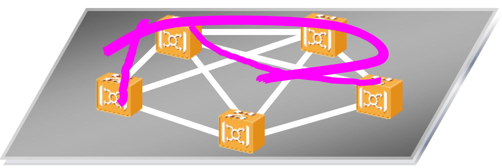
### Hot standby




### Bandwidth Calendaring

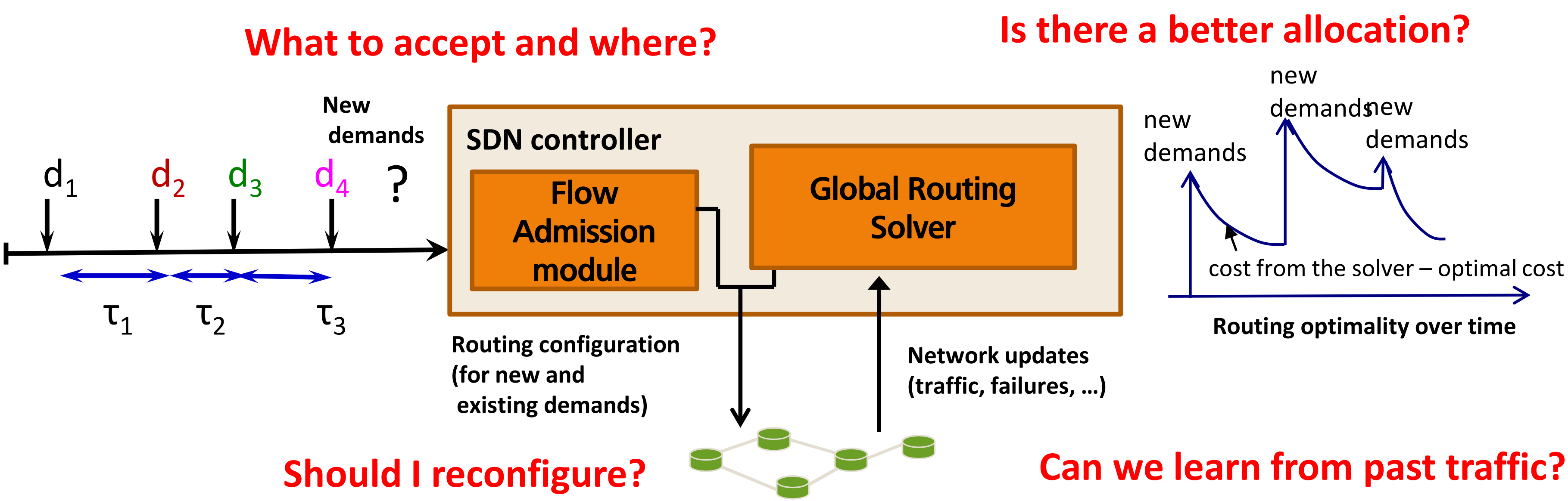



### Flow Splitting



# Online Routing Optimization Challenge

- **Solving an evolving instance of an optimization problem**
  - Demands arrive and depart, congestions and failures happen
  - Limited time to compute a feasible solution at each step ➔ possibly not enough time to converge to the optimal point.
  - Sequential discovery of demands ➔ compete with the offline optimal.



# Outline

- Introduction on SDN
- Path computation tool box
- Network optimisation algorithms
- Online algorithms

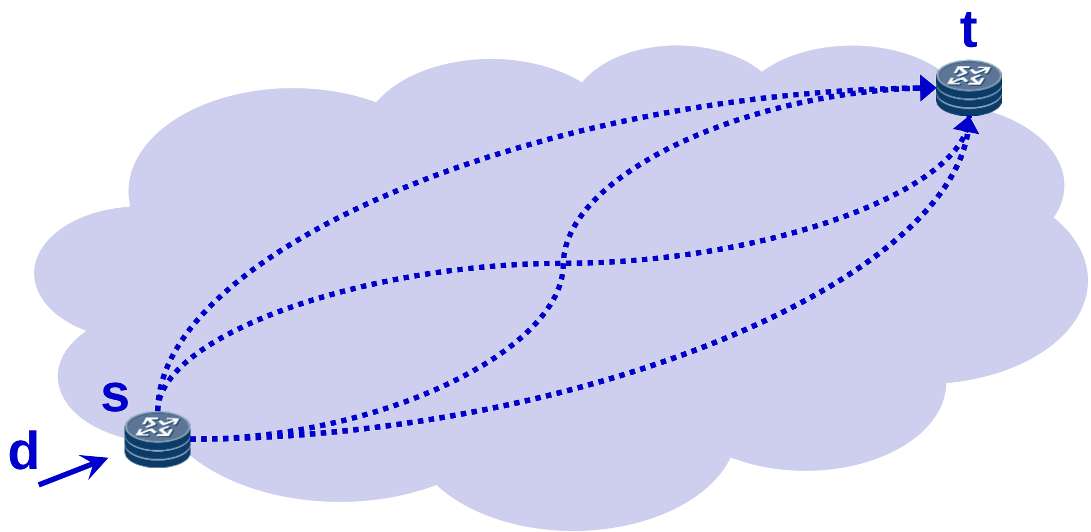
# CONSTRAINED PATH COMPUTATION

OVERLAY ROUTING FOR FAST VIDEO TRANSFERS IN CDN. IEEE IM 2017

P. MEDAGLIANI, S. PARIS; J. LEGUAY, L. MAGGI

IEEE IM 2017

# Multi Constraints Shortest Path




Single demand (commodity) d = (s, t)

- s = source
- t = destination (target)
- d(p) = **delay** of path p connecting s-t
- $\{w_1(p), w_2(p), \ldots, w_m(p)\}$ = m **additive weight** functions (jitter, pLoss, etc.)

**MCSP** (Multi Constraints Shortest Path) Problem: ⇨ **MCSP is NP-Complete**

$$\min \{ d(p) : p \in P_{st} \wedge w_1(p) \leq \Delta_1 \wedge w_2(p) \leq \Delta_2 \ldots \wedge w_m(p) \leq \Delta_m \}$$

# CSK(k) problem formulation

- **K+1 addictive weigths (e.g., jitter, packet loss)**
- **For each path p define**
  - Delay $d(p) = \sum_{(u,v)\in p} d_{uv}$
  - Other metrics $w_i(p) = \sum_{(u,v)\in p} w_{uv}^i \quad i = 1,2,\dots,k$

**GOAL: Find a minimum delay feasible s-t path**

$$
\begin{aligned}
\min \quad & \sum_p d(p) x_p \\
s.t \quad & \sum_p x_p = 1 \\
& \sum_p w_i(p) x_p \le r_i \quad && i = 1,2,\dots,k \\
& x_p \ge 0 && \forall p \in P_{st}
\end{aligned}
$$

# Relaxation of the original problem

- **Define the Lagrangian function as**

$$L(\Lambda) = \min_{p \in P_{st}} \left\{ d(p) + \sum_{i=1}^{k} \lambda_i (w_i(p) - r_i) \right\}$$

$$\Downarrow$$

**Maximizing multiplier is defined as**

$$\Lambda^* = \arg\max_{\Lambda > 0} L(\Lambda)$$

# GEN-LARAC algorithm

**Step 1**: $\Lambda^0 \leftarrow (0, 0\ldots, 0)$; $t \leftarrow 0$; $flag \leftarrow$ true; $B \leftarrow 0$
**Step 2**: **(Coordinate Ascent Steps)**

```
while (flag)
   flag ← false
   for i = 1 to k
      γ ← arg max_{ξ≥0} L(λ^t_1,…, λ^t_{i−1}, ξ , λ^t_{i+1},…, λ^t_k).
      if (γ ≠ λ^t_i) then
         flag ← true
```

$$\lambda_j^{t+1} = \begin{cases} \gamma & j = i, \\ \lambda_j^t & j \neq i. \end{cases}, j = 1, 2\ldots, k$$

```
         t ← t + 1
      end if
   end for
end while
```

**Step 3:** If $\Lambda^t$ is optimal then return $\Lambda^t$.
**Step 4:** $B \leftarrow B + 1$ and go to Step 5 if $B < B_{max}$ ($B_{max}$ is the maximum number of iteration allowed); Otherwise, stop.
**Step 5:** Compute a new vector $\Lambda^+$ such that $L(\Lambda^+) > L(\Lambda^t)$.
**Step 6:** $t \leftarrow t + 1$, $\Lambda^t \leftarrow \Lambda^+$, and go to Step 2.

Ying Xiao, Krishnaiyan Thulasiraman, Guoliang Xue, "GEN-LARAC: A Generalized Approach to the Constrained Shortest Path Problem Under Multiple Additive Constraints", 2005.

# Verification of optimality

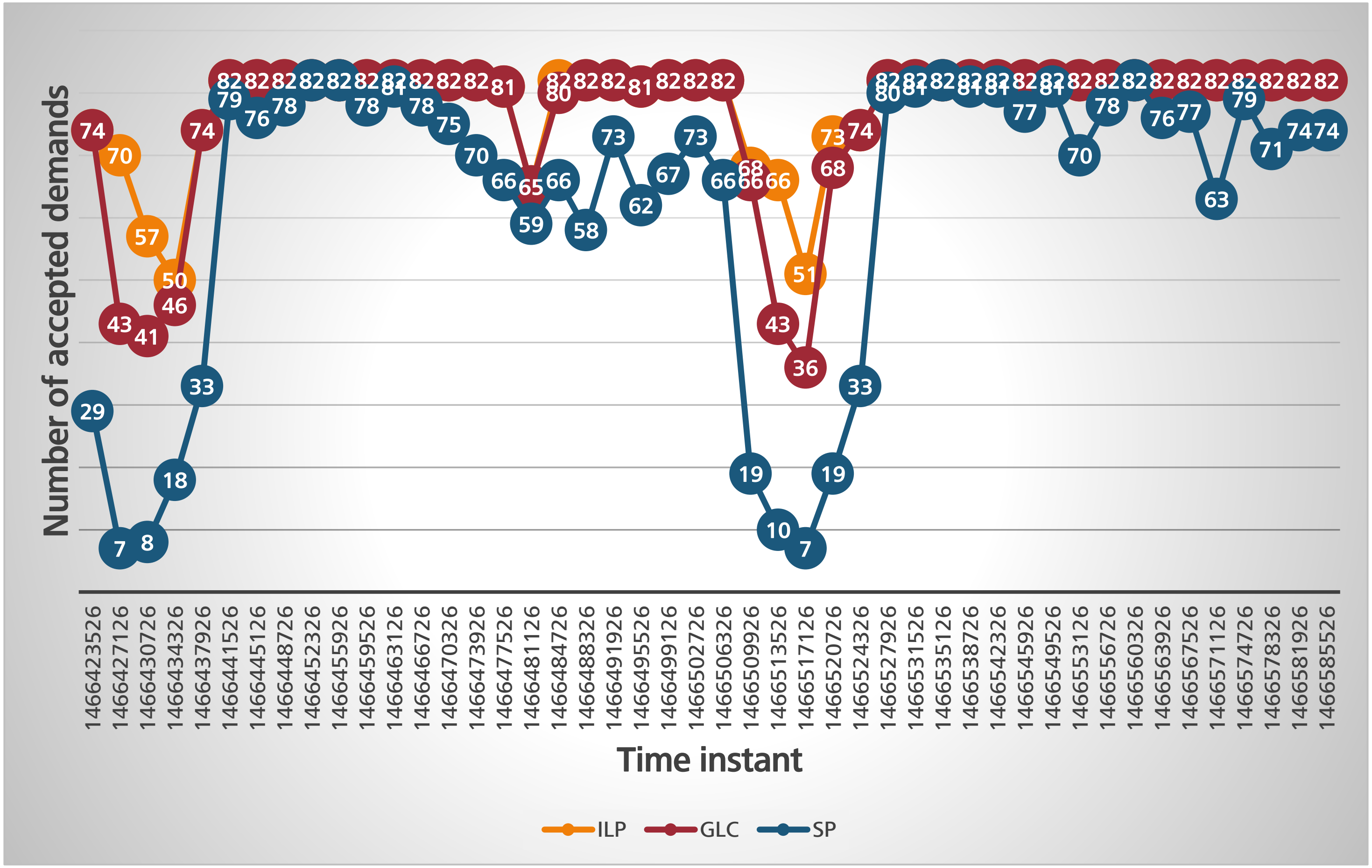
- **Step 3 requires verification of optimality**
- **This can be accomplished by solving the following linear problem where $P_\lambda=\{P_1,P_2,\ldots,P_k\}$ is the set of Λ-minimal paths.**
- **If this problem is feasible, than Λ is a maximizing multiplier**

## Complementary slackness

$$
\begin{aligned}
\max \quad & 0 \\
s.t \quad & \sum_{p_j \in P_\Lambda} u_j w_i(p_j) = r_i \qquad \forall i, \lambda_i > 0 \\
& \sum_{p_j \in P_\Lambda} u_j w_i(p_j) \le r_i \qquad \forall i, \lambda_i = 0 \\
& \sum_{p_j \in P_\Lambda} u_j = 1 \\
& u_j \ge 0 \qquad\qquad\qquad \forall j, p_j \in P_\Lambda
\end{aligned}
$$

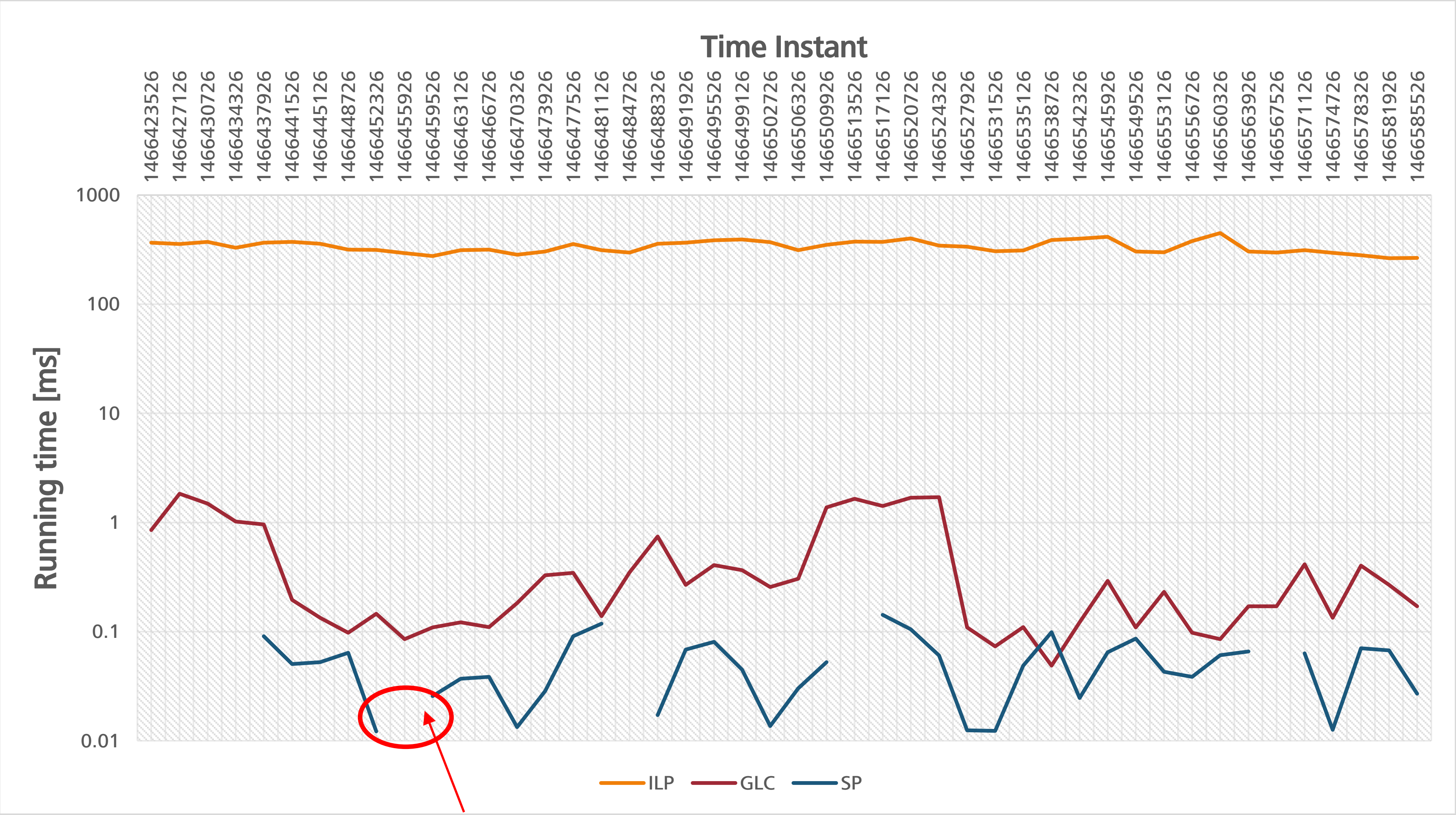
# Performance – Number of accepted demands

## Evaluation in a CDN overlay network



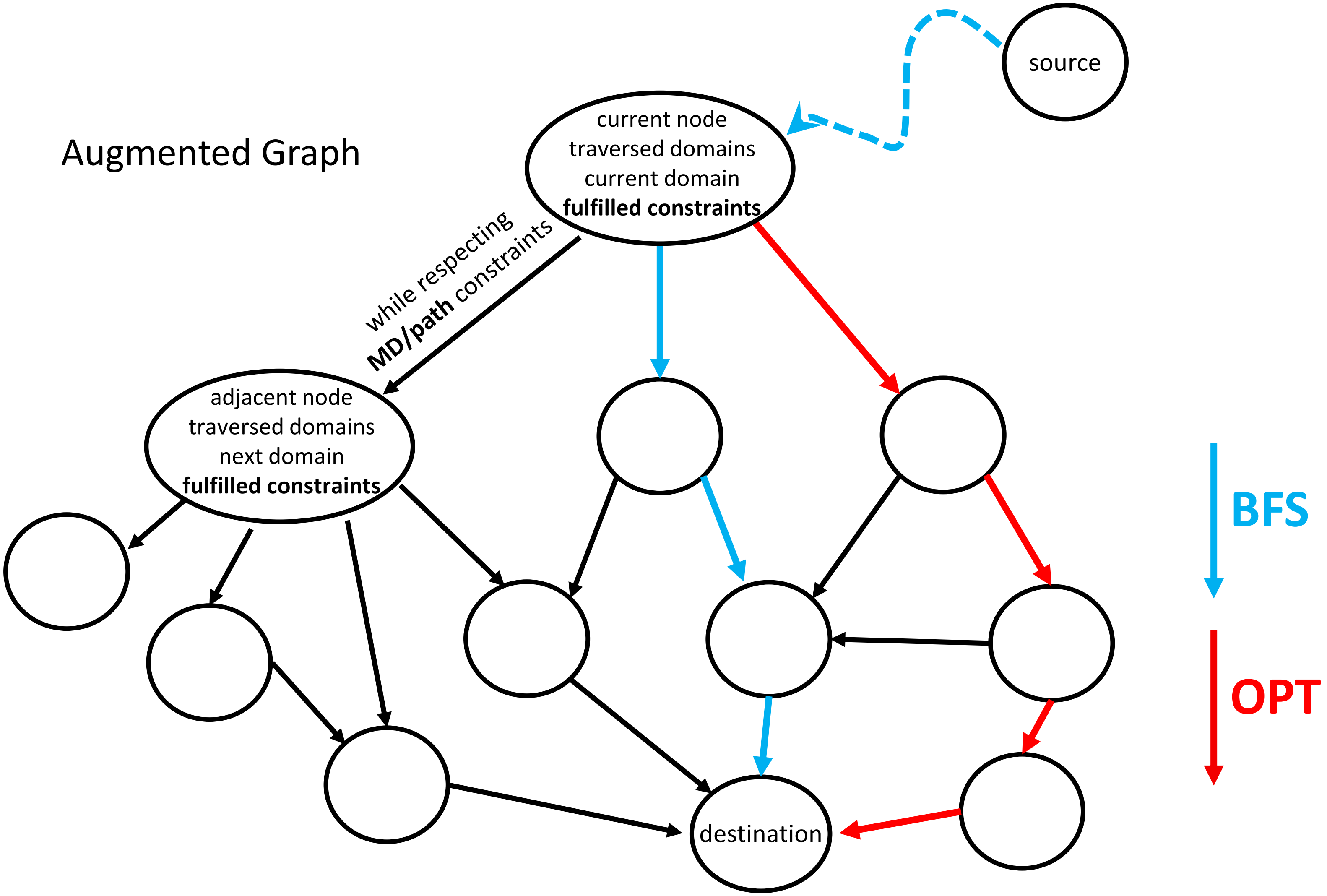
# Performance – Running time

## Evaluation in a CDN overlay network




**Below min clock resolution**

# Other techniques: Dynamic programming



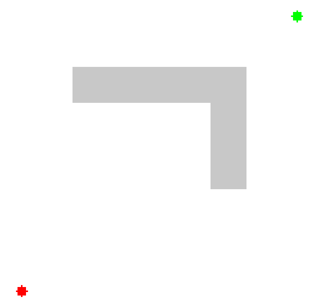

1. Shortest Path on Augmented graph (BFS opt) (optimal, slow)
2. Breadth First Search (BFS) (always gives a feasible, suboptimal solution)

# Other techniques: Pre-computation – A* Algorithm

- **Informed search algorithm**
  - Also called best first search algorithm
- **Selects the path that minimizes**

$$f(n) = g(n) + h(n)$$

  - *n* is the last node on the path
  - $g(n)$ is the cost of the path from the start node to *n*
  - $h(n)$ is a **heuristic** that estimates the cost of the cheapest path from *n* to the goal.



# Outline

- Introduction on SDN
- Path computation algorithms
- Network optimisation algorithms
- Online algorithms

# BANDWIDTH CALENDARING

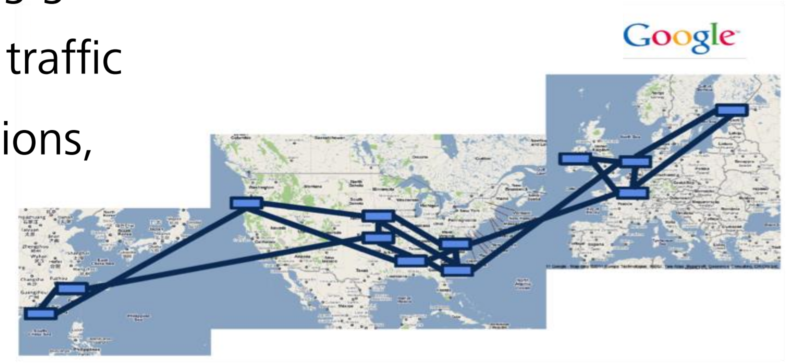
## BANDWIDTH CALENDARING: DYNAMIC SERVICES SCHEDULING OVER SOFTWARE DEFINED NETWORKS

LAZAROS GKATZIKIS, STEFANO PARIS, IOANNIS STEIAKOGIANNAKIS, SYMEON CHOUVARDAS

IEEE ICC 2016

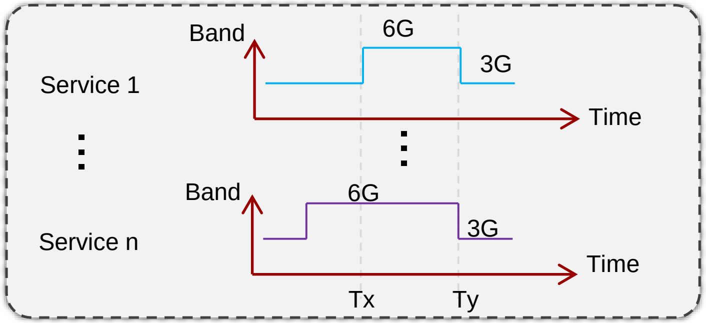
# Bandwidth Calendaring

- **Context: Inter-datacenter networks**
  - Deployed by cloud companies operating geo-distributed datacenters
  - Need to support bulky and predictable traffic across datacenters (map reduce operations, database synchronization, etc.)



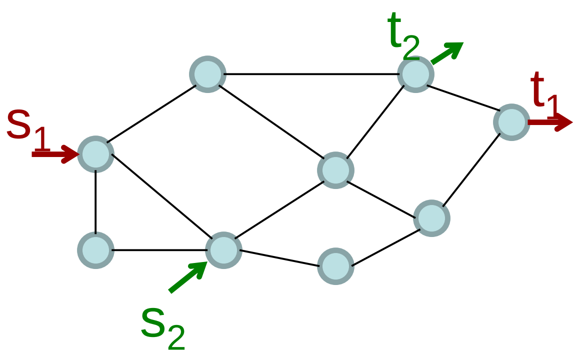

- **Main problem**
  - Find feasible transfers in time and space
  - At scale (high demands, large networks)

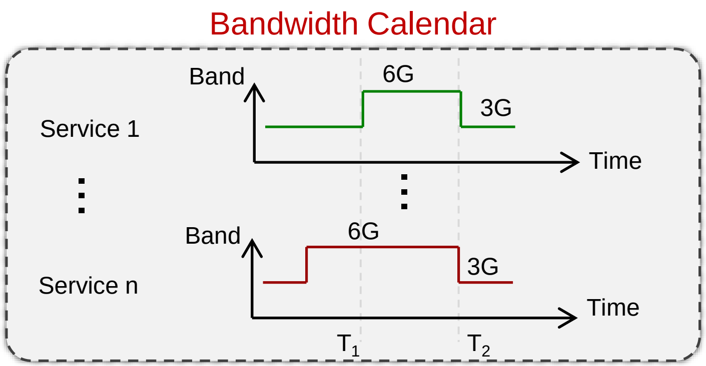
Bandwidth Calendar



# Bandwidth Calendaring

- **Problem:** Optimal scheduling and routing of future bandwidth reservations
- **Input parameters:**
  - Network topology
  - Current network state at $T_0$ (paths and bandwidth allocated to existing demands)
  - Future arrivals along with their time-varying requirements (bandwidth demand changes on certain time points)




- **Control:** allocation of paths and schedule to each demand
- **Objective:** min rejection ratio
- **Secondary objective:** min routing cost

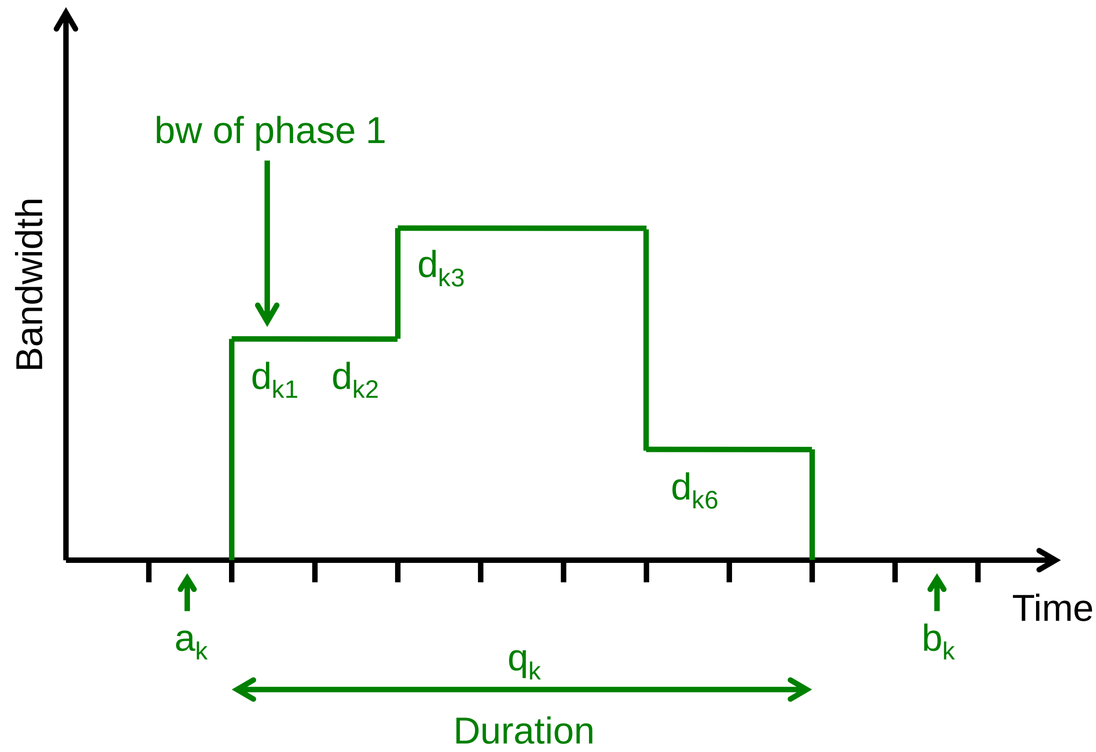
# Problem Formulation – Demands

- **Demand k**

  ($s_{k,}$ $\tau_k$, $a_k$, $b_k$, $q_k$, $\mathbf{d}_k$)

  - $s_k$ = source
  - $\tau_k$ = destination
  - $a_k$ = start time
  - $b_k$ = end time
  - $q_k$ = duration
  - $\mathbf{d}_k$ = traffic profile

    [$d_{k1}$ … $d_{kf}$ … $d_{1|F|}$]




- Traffic profile is a vector where each element represents the bandwidth requested in the corresponding phase
  - Strict request ➔ $b_k - a_k + 1 = q_k$
  - Elastic request ➔ $b_k - a_k + 1 > q_k$

# Problem Formulation – Input & Variables

- **Input parameters:**

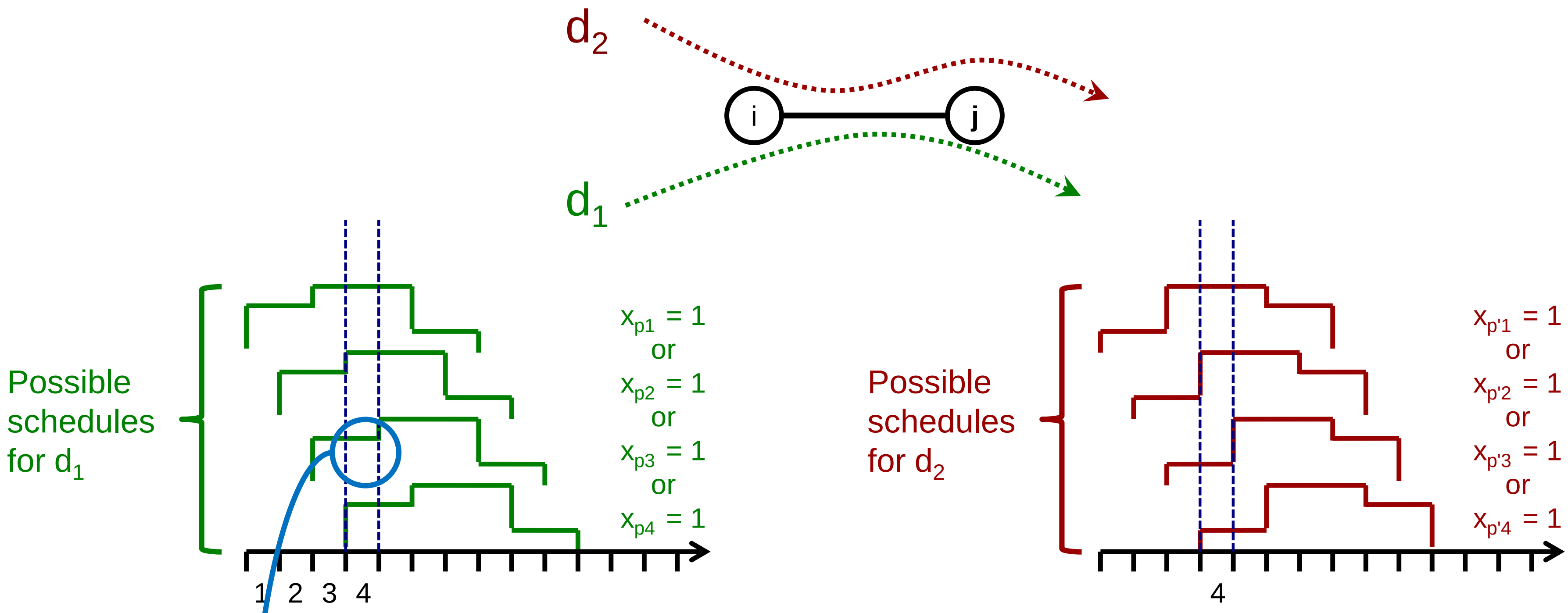
| Notation | Meaning |
|---|---|
| $c_e$, $b_e$ | Cost (delay) and capacity of link e |
| $d_{kf}$ | Bandwidth of demand k during time f |
| $a_k$, $b_k$ | Start and end time of demand k |
| $q_k$ | Duration of demand k ($q_k \leq b_k - a_k + 1$) |
| $P_e$ | Set of paths using link e |
| $P_k$ | Set of paths available for demand k |

- **Variables (binary):**

| Notation | Meaning |
|---|---|
| $x_{pt}$ | Starting time of utilization of path p |
| $y_{et}$ | Temporal utilization of link e |

- We have to use **disjoint sets of paths** for different **demands** (otherwise $x_{pt}$ represents the aggregated traffic)
  - Same physical path available to multiple demands → multiple virtual paths

# Problem Formulation – Used Capacity




Total demand flowing over link e at time t

$d_{k2}\, x_{p(4-2+1)}$

$$d_{et} = \sum_{p \in P_e} \sum_{k:p \in P_k} \sum_{\substack{f=1 \\ f \le t}}^{q_k} d_{kf}\, x_{p(t-f+1)}$$

# ILP Formulation

$$\min \sum_{t \in T} \sum_{e \in E} c_e y_{et}$$ ← Cost over time → **NP-**Hard

$$s.t \quad \sum_{p \in P_e} \sum_{k: p \in P_k} \sum_{\substack{f=1 \\ f \le t}}^{q_k} d_{kf} x_{p(t-f+1)} \le b_e y_{et} \qquad \forall e \in E, \forall t \in T$$ ← Capacity

$$\sum_{t=a_k}^{b_k - q_k + 1} \sum_{p \in P_k} x_{pt} = 1 \qquad \forall k \in K$$ ← Single path schedule

- single start time and path

$$x_{pt} \in \{0,1\} \qquad \forall p \in P, \forall t \in T$$

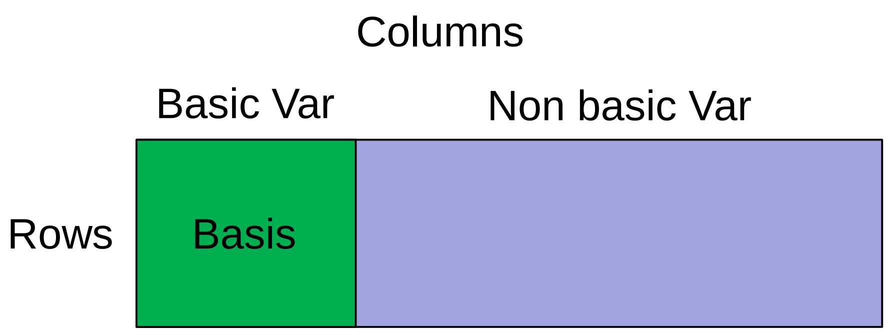
$$y_{et} \in [0;1] \qquad \forall e \in E, \forall t \in T$$

- Solving time increase sharply with the problem size.
- **Solution**: Solve the **LP formulation** and **round** the solution afterward.
  - If we are lucky, no rounding is needed (or just a small number of variables must be rounded up/down).

# Challenges – Complexity

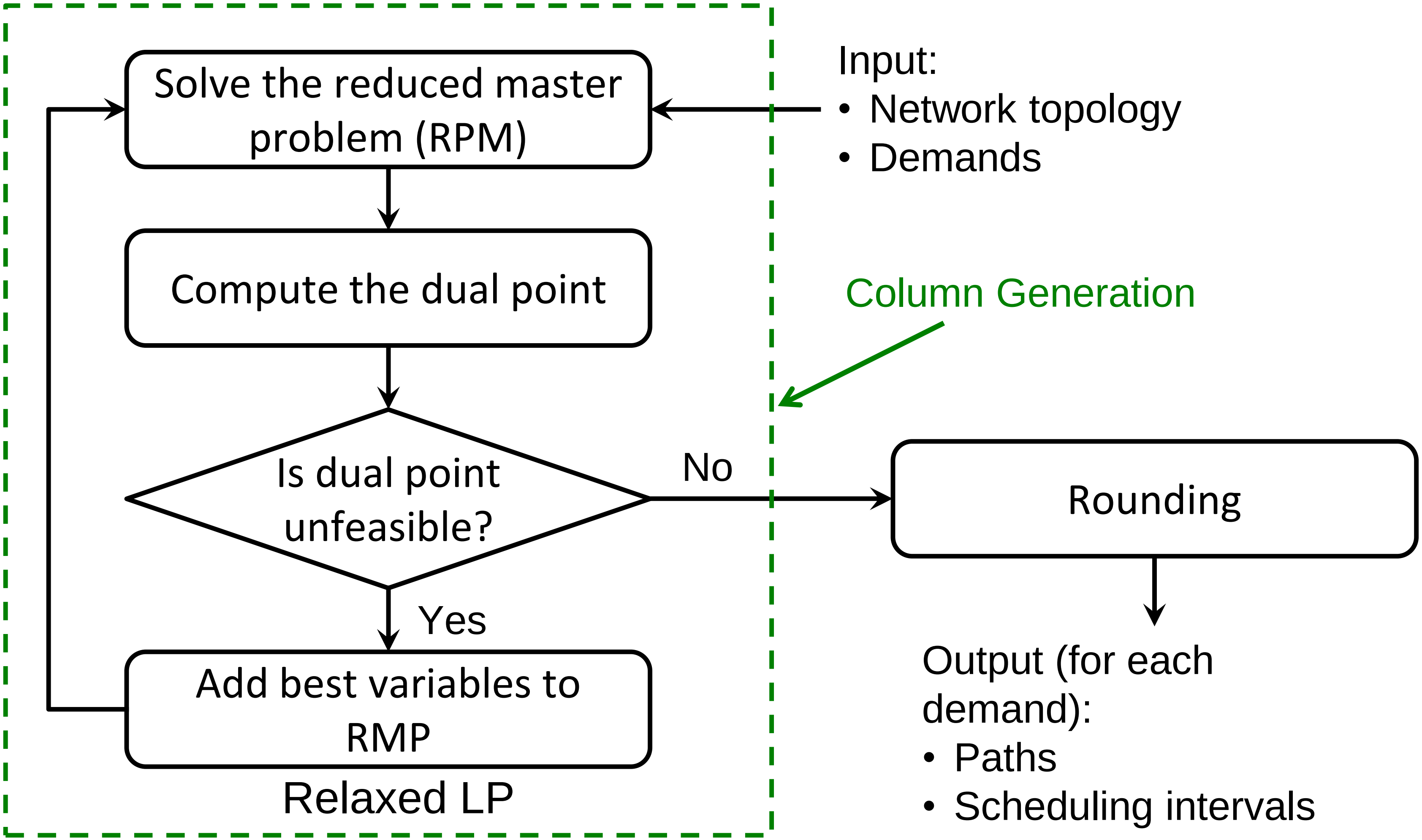
## Memory

- Assuming traffic profile with granularity of 15 min, 5k links, and 250k demands we have :
  - |E| |T| = 5k * 0.1k = 500 k capacity constr.
  - |K| = 250 k scheduling constr.
- 750k total constraints
  - Basis matrix = $(750k)^2$ * 4 byte = **2.25 Tbyte**…
  - OK, 250k demands are too many… But $(500k)^2$ * 4 byte = **1 Tbyte**…



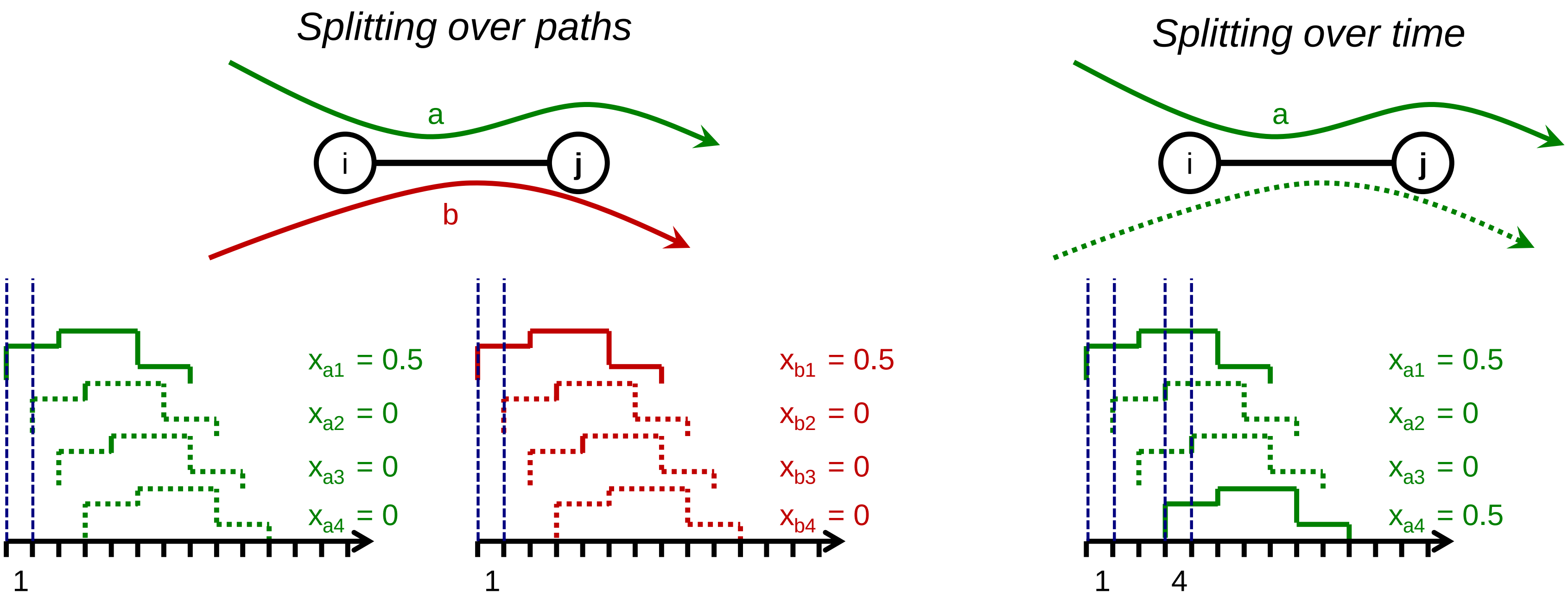

**Problem size reduction** needed

# Joint Scheduling & Routing



# Challenges – Rounding

- The linear relaxation may lead to **splitting over paths** and **over time**




- Need for new **rounding** mechanisms
  - **Post-processing**: exploration of possible schedules and selection of a single starting time

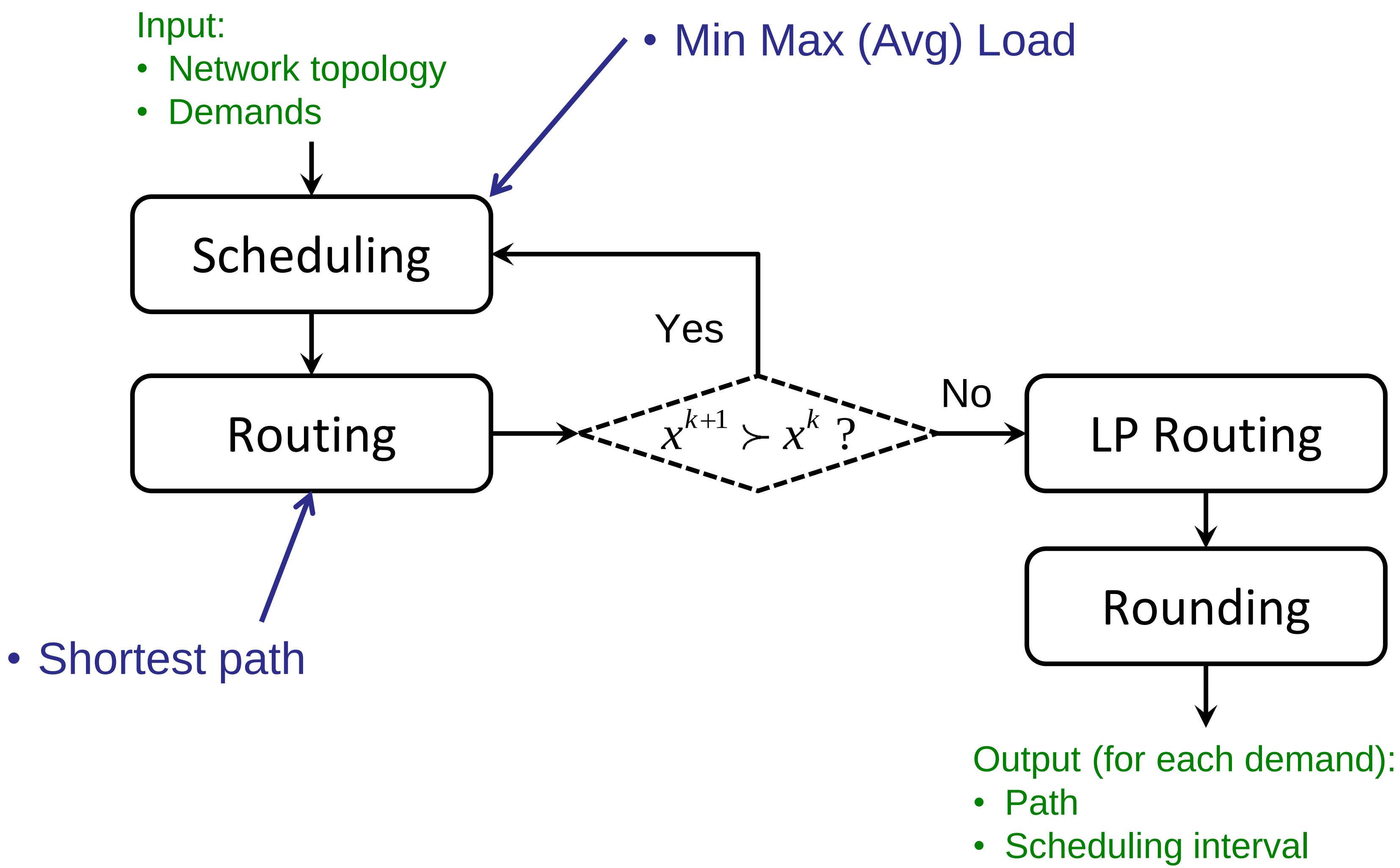
# Decoupling Scheduling and Routing

- **Problems**:
  - Problem size is too huge.
  - Splitting over time and paths (poor relaxation, rounding is difficult)

- **Solution**: Decouple the scheduling and routing
  - Schedule the demands to load balance the use of the network over time.
  - Solve the routing only with strict demands.

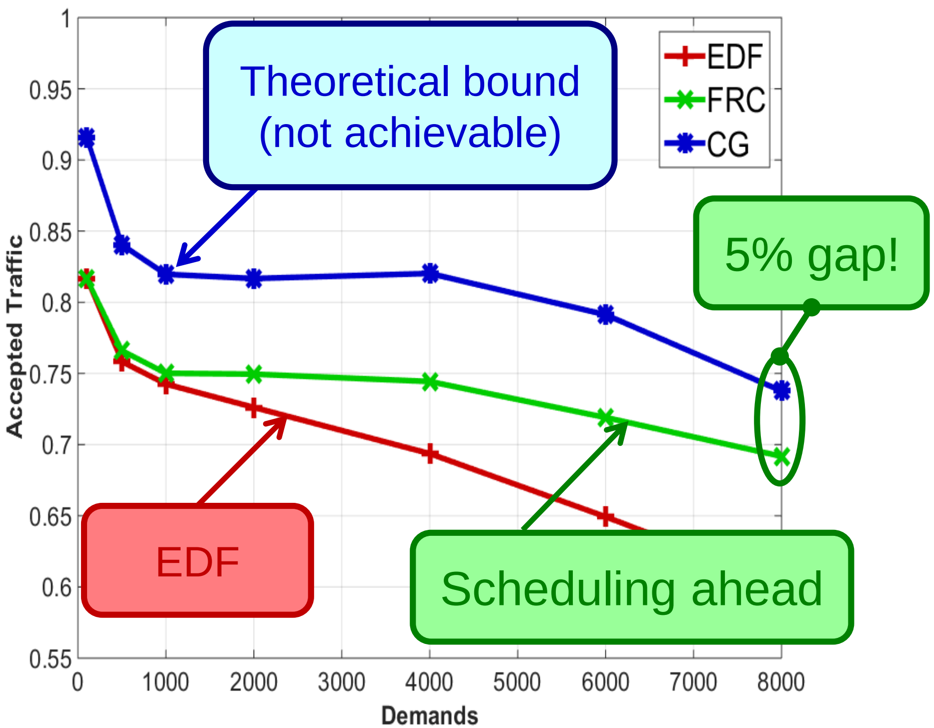
- **Operating only with strict demands makes the problem simpler**
  - For each demand we have to compute a path on a single graph.
  - We do not have to deal with the splitting over time.

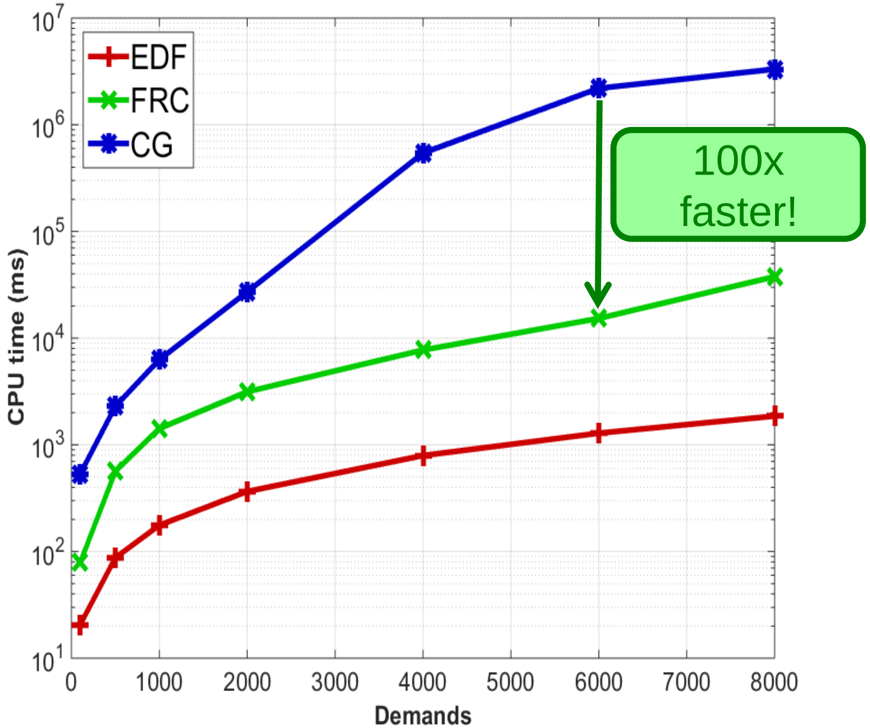
# Decoupled Scheduling & Routing



# Performance evaluation

**Scalability tests**
Network size (600 nodes, 6000 links)



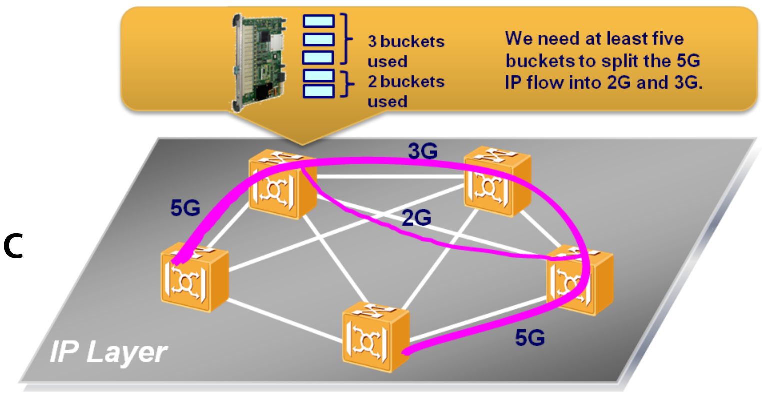
# FLOW SPLITTING / LOAD BALANCING

## GLOBAL OPTIMIZATION FOR HASH-BASED SPLITTING

PAOLO MEDAGLIANI, JÉRÉMIE LEGUAY, MOHAMMED ABDULLAH, MATHIEU LECONTE, STEFANO PARIS

IEEE GLOBECOM 2016

# Why flow splitting?




- **Flow splitting**
  - Helps to balance traffic or accept more traffic
  - Improves reliability in case of failures
- **Current solutions: equal splitting**
  - ECMP – Even split on equal cost paths
- **Hash-based splitting for unequal splitting**
  - Only a limited set of possible splits are possible
  - Forwarding rules are stored in precious “buckets”
- **Main problem**
  - Find a feasible solution that maximizes the throughput and min cost.
  - The problem is not linear and extremely hard even to approximate.

# Hash-based splitting

dest. address → **Forwarding table (node 0)**

| | Match | Action |
|---|---|---|
| demand $D_1$ | D1 | Apply Group 1 |
| demand $D_2$ | D2 | Forward to 56.78.91.**2** |
| | … | … |

Nb entries: $\tau_0^F$

group ID → **Group table (node 0)** → next-hop

| Group Id | Hash | Next-Hop |
|---|---|---|
| 1 | 0 | 56.78.91.**1** |
| 1 | 1 | 56.78.91.**2** |
| 1 | 2 | 56.78.91.**2** |
| 1 | 3 | 56.78.91.**3** |
| 1 | 4 | 56.78.91.**3** |
| 1 | 5 | 56.78.91.**3** |
| … | … | .. |

Nb entries: $\tau_0^G$

**1 bucket**

**1/6**

**1/3**

**1/2**

**split ratios for $D_1$**

**Extend ECMP by repeating entries**

Typical size of TCAM memory is 1M entries[1]

[1] K. Kannan, S. Banerjee, *“Compact TCAM: Flow entry compaction in TCAM for power aware SDN”*, Distributed Computing and Networking, 2013

# Hash-based splitting

dest. address → Forwarding table (node 0) → group ID → Group table (node 0) → next-hop

**Forwarding table (node 0)**

| | Match | Action |
|---|---|---|
| emand $D_1$ | D1 | Apply Group 1 |
| emand $D_2$ | D2 | Forward to 56.78.91.**2** |
| | ... | ... |

Nb entries: $\tau_0^F$

**Group table (node 0)**

| Group Id | Hash | Next-Hop | |
|---|---|---|---|
| 1 | 0 | 56.78.91.**1** | 1/6 |
| 1 | 1 | 56.78.91.**2** | 1/3 |
| 1 | 2 | 56.78.91.**2** | |
| 1 | 3 | 56.78.91.**3** | 1/2 |
| 1 | 4 | 56.78.91.**3** | |
| 1 | 5 | 56.78.91.**3** | |
| ... | ... | .. | |

1 bucket

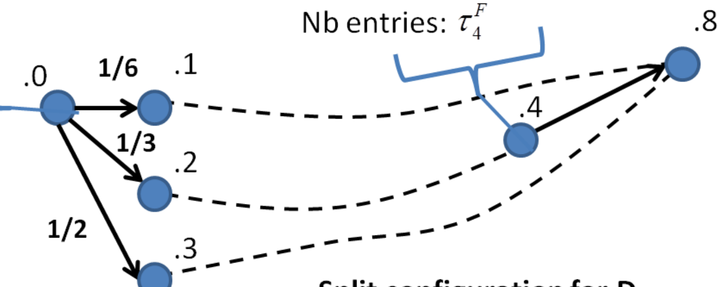
Nb entries: $\tau_0^G$

split ratios for $D_1$

**Forwarding table (node 4)**

| Match | Action |
|---|---|
| D1-sub path 2 | 56.78.91.**8** |
| ... | ... |

Nb entries: $\tau_4^F$

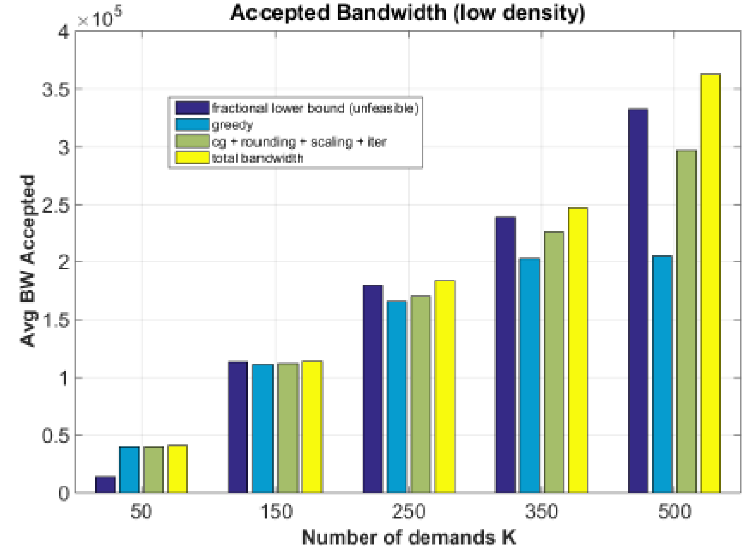



Split configuration for $D_1$

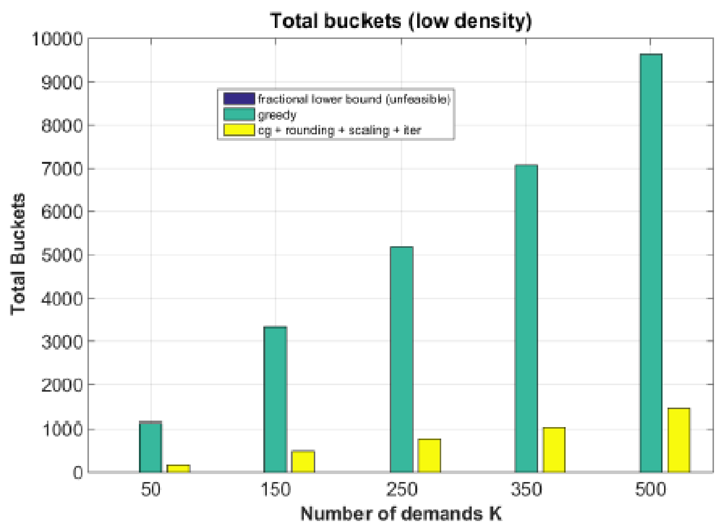
- **The larger the number of buckets  the better the flow distribution accurately models a fractional ideal**
- **The distribution of flow volume amongst the paths is constrained by the use of a limited number of TCAM entries**

# Iterative scaling approach

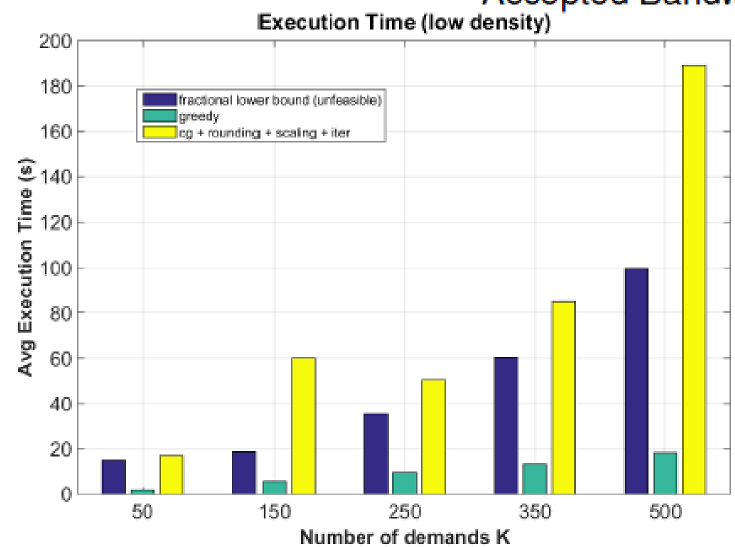
- Column Generation on the unconstrained prob.
- A rounding phase to find a feasible allocation
- Network capacity scaling by  different factors α
- Run in parallel several instances
- Iterate to allocate remaining demands




Accepted Bandwidth




Average number of buckets




Execution time

**Path allocations  close to the optimal solution**

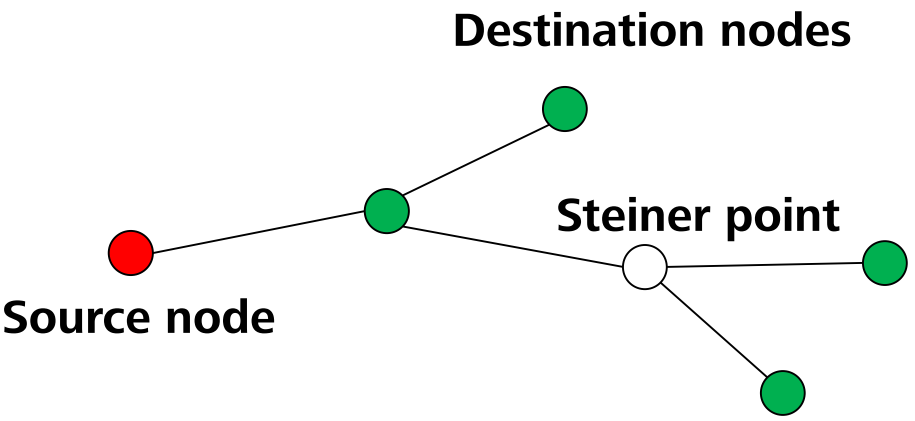
# MULTICAST ROUTING

# Multicast routing for live video streaming

- **For each multicast service, compute a tree from a source to a given set of destinations under some constraints**
    - » Inclusion/Exclusion, Available bandwidth, Delay/Hop limitation
- **Goal: maximize the admitted traffic (primary) and minimize the total trees cost (secondary), respecting the given constraints**

Multicast service :

- Source node
- A set of destination nodes
- Bandwidth
- Max delay
- Max # hops
- Optional Steiner points



# Lagrangian Relaxation

## ILP Program

$$\begin{aligned} \min \quad & cx \\ s.t. \quad & Ax \geq b \\ & Bx \geq d \\ & x \in \{0,1\} \end{aligned}$$

## Lagrangian Relaxation

$$\max_{\lambda \geq 0} \left\{ \begin{aligned} \min \quad & cx + \lambda(b - Ax) \\ s.t. \quad & Bx \geq d \\ & x \in \{0,1\} \end{aligned} \right\}$$

**Lagrangian Lower Bound Program (LLBP)**

- For any **λ ≥ 0 (Lagrangian multipliers)**, the program **LLBP** provides **a lower bound** to the original problem.
- The **best multipliers** are computed using the **subgradient method**.

# Multicast for Video

**Decouple routing from resource assignment** → **Simpler problems can be handled more efficiently**

**Lagrangian Relaxation**

### Tree Computation Module
- Compute multicast tree for each demand under various constraints

### Resource Pricing Module
- Compute optimal dual price for all link-time bandwidth

### Feasibility Solution Module
- A heuristic to find good feasible solution based on dual price info

# Sub-gradient based algorithm

1. Heuristic algorithm (initial feasible solution) — **Pre-processing**

2. Repeat
   a. Solve decomposed subproblems
   b. Aggregate subproblems
   c. Compute subgradient
   d. Update stepsize
   e. Update Lagrangian multipliers
   f. Normalize Lagrangian multiplers

   Until (no improvement) and (timeout is expired)

   — **Subgradient Method**

4. Feasibility step — **Post-processing**

**Slow and (sometimes difficult) convergence,**
**but highly parralelizable and small memory footprint**

# Outline

- Introduction on SDN
- Path computation algorithms
- Network optimisation algorithms
- **Online and anytime algorithms**

# ADMISSION CONTROL

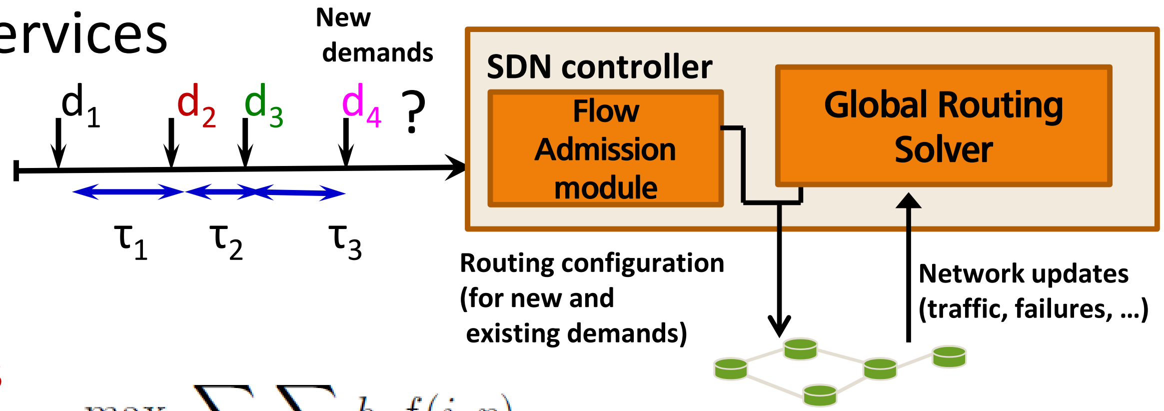
ADMISSION CONTROL WITH ONLINE ALGORITHMS IN SDN

J LEGUAY, L MAGGI, M DRAIEF, S PARIS, S CHOUVARDAS

IEEE NOMS 2016

# Admission Control Problem

- Context of guaranteed services
- Main Goal
  - Maximize throughput (i.e., profit) over time
  - Accept or reject demands when they arrive




$$\max_{f} \sum_{i \in K} \sum_{p \in \mathcal{P}_i} b_i \, f(i,p)$$

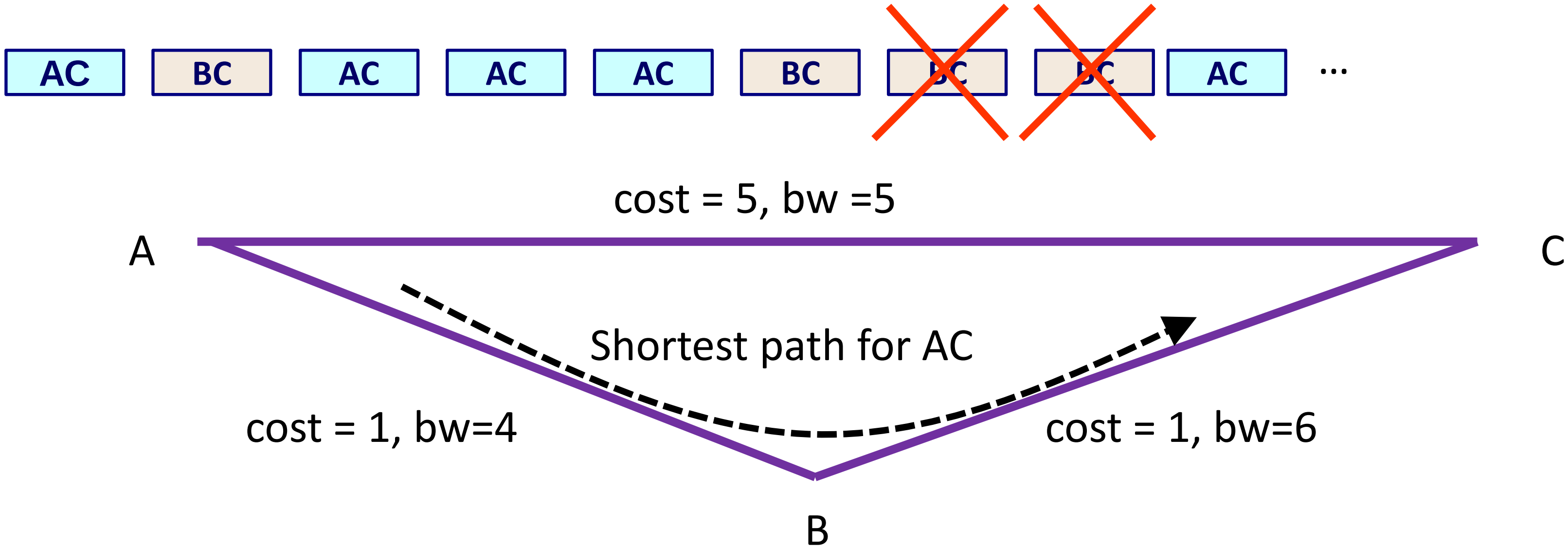
$$\text{s.t.} \sum_{i \in K} \sum_{p \in \mathcal{P}_i | e \in p} f(i,p)\, r_i(\tau) \leq u(e), \quad \forall e \in E, \; \tau \geq 0$$

$$\sum_{p \in \mathcal{P}_i} f(i,p) \leq 1, \qquad \forall i \in K$$

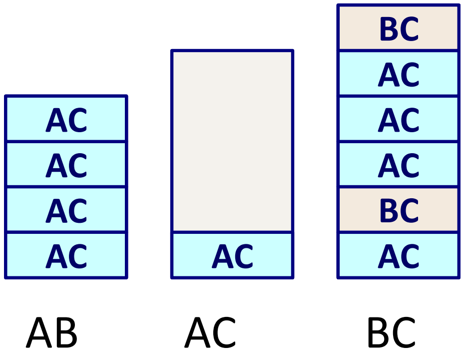
$$f(i,p) \in \{0,1\}.$$

- **State of the art**
  - Immediately accept if room (mainly for video or voice) ➔ here our objective is to maximize the overall throughput over time
  - Planning tools to use max-, min-, exclusive- and nonexclusive- limits on resource portions for different classes of flows. ➔ does not capture traffic dynamic

# Naive approach: greedy admission




Allocations to links




**Could we find a better online algorithm? (agnostic of future demands)**

# Online Algorithms for Covering / Packing Problems

- Problem objective
  - Consider an input sequence σ, OPT is the offline optimum
  - Find an algorithm A such that:

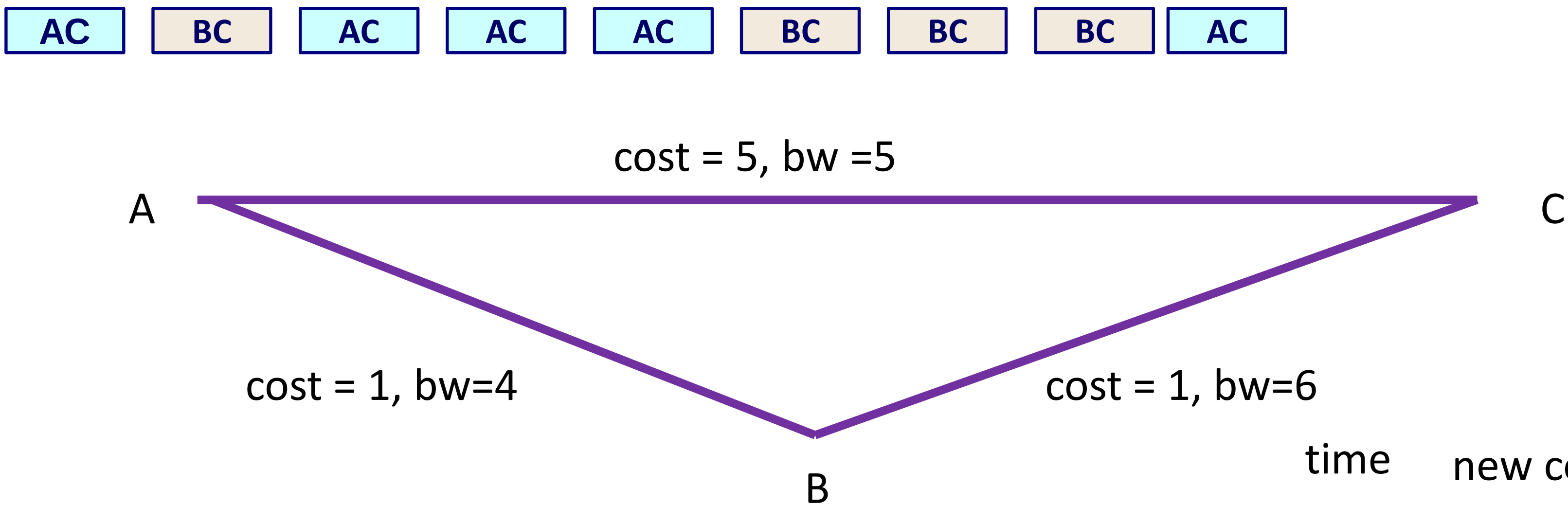
| Cost problems | Benefit problems |
|---|---|
| $\forall\sigma\ \ A(\sigma) \le C{*}OPT(\sigma)$ | $\forall\sigma\ \ A(\sigma) \ge \frac{1}{C}{*}OPT(\sigma)$ |

C is the competitive ratio

- Worst-case algorithms
  - [Awerbuch, Azar, Plotkins 93] – O(log V) competitiveness
  - Framework for covering and packing problems [Buchbinder05, Naor 06]
- Beyond worst case (stochastic) algorithms
  - [Kesselheim, STOC 14', Agrawal, SODA 15']

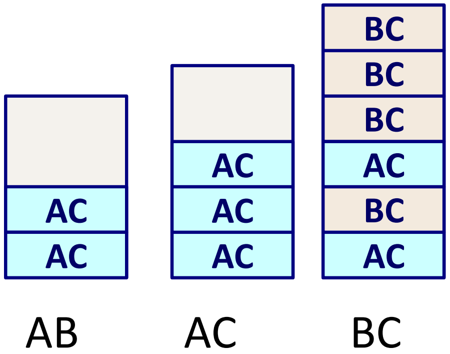
**Never applied** as they require a central execution **(Now possible in SDN!)**

# AAP's Oracle: exponential increase of edge costs




| time | new cost (BC) |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 4 |
| 5 | 5 |
| 6 | 5 |
| 7 | 5 |
| 8 | 5 |
| 9 | 10 |

Allocations to links




**Pushes away demands from bottlenecks**

# Primal-Dual AAP – log(n) competitive

## Using Primal / dual framework for online packing [Naor'06]

$$x_e(t) = \frac{1}{n}\exp\left(\frac{\ln(1+n)}{u_e}\sum_i \sum_{p\in\mathcal{P}_i|e\in p} f(i,p).r_i(t)\right) - \frac{1}{n}.$$

Can be transformed with multiplicative updates (Naor06):

---

**Algorithm 4** Primal-Dual AAP Algorithm [7], [12]

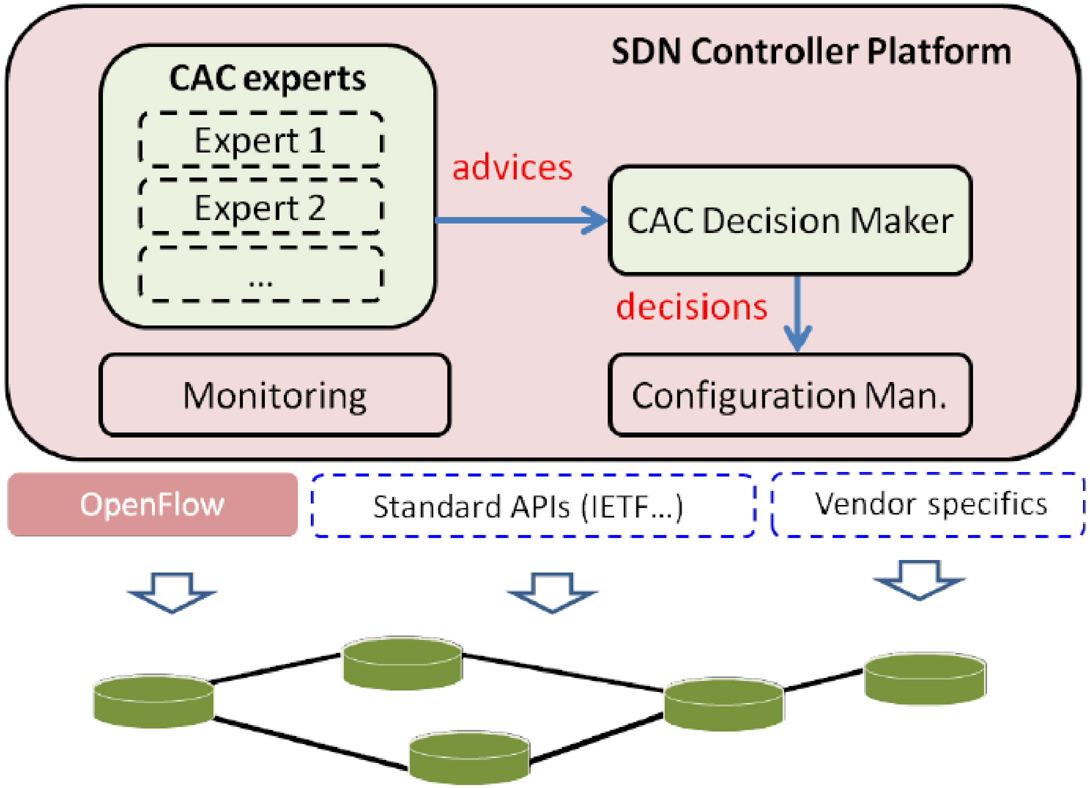
---

Initialize $x_e = 0$
**function** ROUTE(request $j$)
  **if** $\exists$ a path $P \in \mathcal{P}_j$ of cost $< 1$ in the graph weighted by $x_e$. **then**
    Route request $j$ on $P$
    **for** each edge $e \in P$ **do**
      $x_e = x_e \exp\frac{\ln(1+n)r_j}{u(e)} + \frac{1}{n}(\exp\frac{\ln(1+n)r_j}{u(e)} - 1)$
    **end for**
  **else**
    Reject request $j$
  **end if**
**end function**

---

# Expert Algorithms

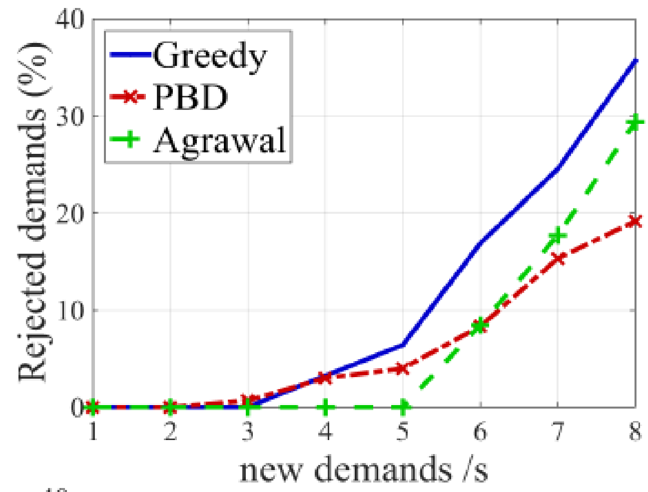
- **Opportunity**
  - SDN Controller Platforms have **tremendous computation power**
  - Boosting techniques from **Machine Learning** can be used to solve online optimization problems



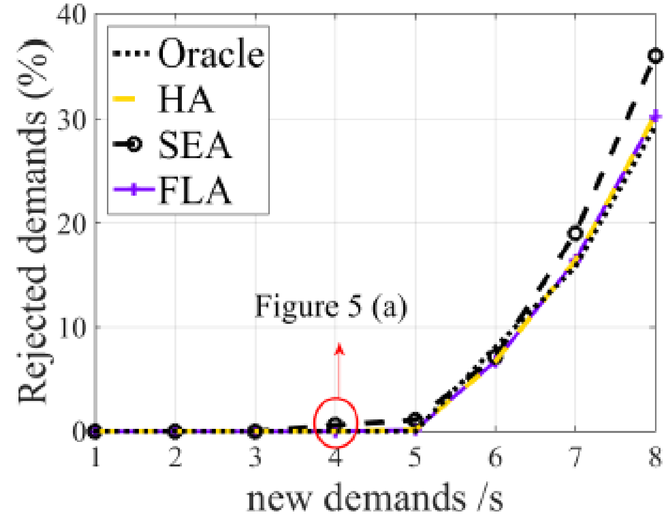

- **Main idea**
  - As no individual AC algorithms is good in every traffic conditions
  - **Use expert meta-algorithms** to keep track of the best CAC algorithms

# Online algorithms for Admission control in SDN

- **Performance evaluation**
  - Much better than Greedy (Accept all flows until there is no resources left)
  - No algorithms is best in all traffic conditions




- **Expert-algorithms to learn the best one**
  - SDN Controller Platforms have **tremendous computation power**
  - Used in Machine Learning, **Boosting techniques** are good candidates to solve optimization problems




**Reduction of 25% to 100%** of the rejected demands

# ROBUST TRAFFIC ENGINEERING

## ONGOING WORK WITH POLIMI (ANTONIO CAPONE)

# Robust Traffic Engineering

## 1) OPTIMIZE UNDER UNCERTANTY

- Robust optimization
- Stochastic optimization

**Find robust configuration regions.**

network configurations where congestion happens

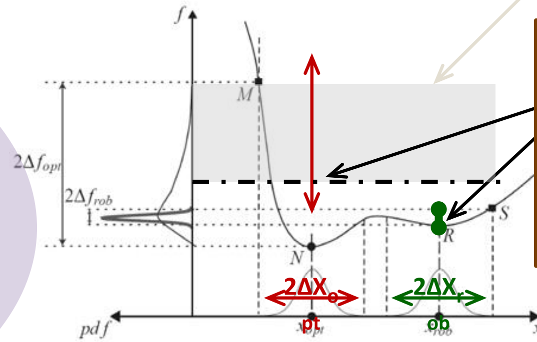
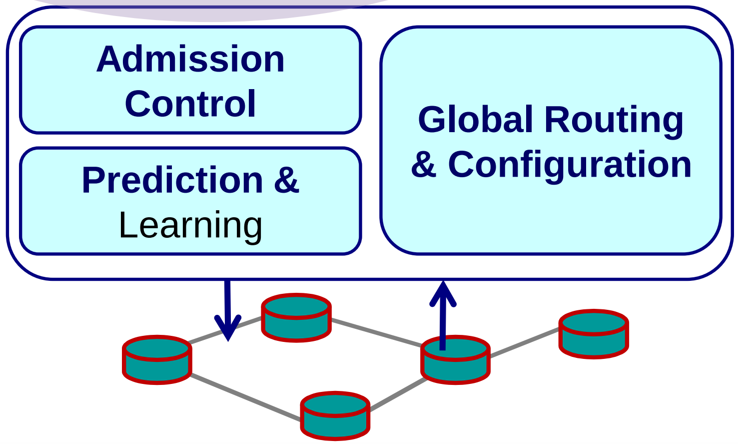


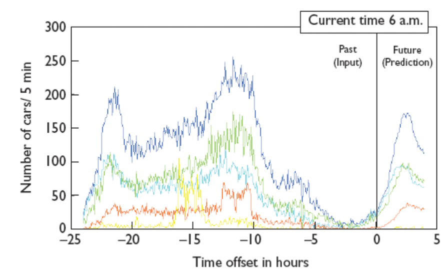

Figure 1 – A robust solution *vs.* an optimal solution.

Search for feasible points
- with the lowest variation
- with the variation bounded in an area
- …

➔ Improve ***optimality*** and ***stability***




$$\begin{pmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1i} \\ \gamma_{21} & \gamma_{22} & & \vdots \\ \vdots & & \ddots & \\ \gamma_{i1} & \cdots & & \gamma_{ij} \end{pmatrix}$$




## 2) OBSERVE & LEARN

- **Traffic matrices**
- **Evolution models**

**Track the system evolution in order to** anticipate the next state
- Predict the next state
- Apply the best configuration

➔ ***Adapt*** to changes

**Our goals:**
- **Robust routing with traffic predictions**
- **Tools: Robust optimization, stochastic optimization**

# Methods to deal with traffic uncertainty

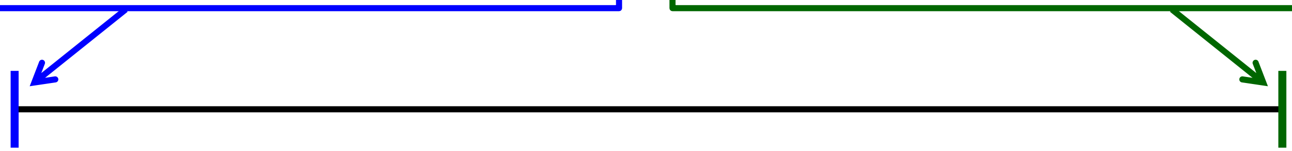
- Two extreme solutions to reconfigure the network after traffic changes:

**Dynamic TE**

- **Monitor** traffic over a **short period**
- Predict traffic for the next period
- Compute the optimal **TE policy for the next period**

**Stable TE**

- **Monitor** traffic over a **long period**
- Consider a larger set of TMs (envelope)
- Compute a **single TE policy** for this set of TMs

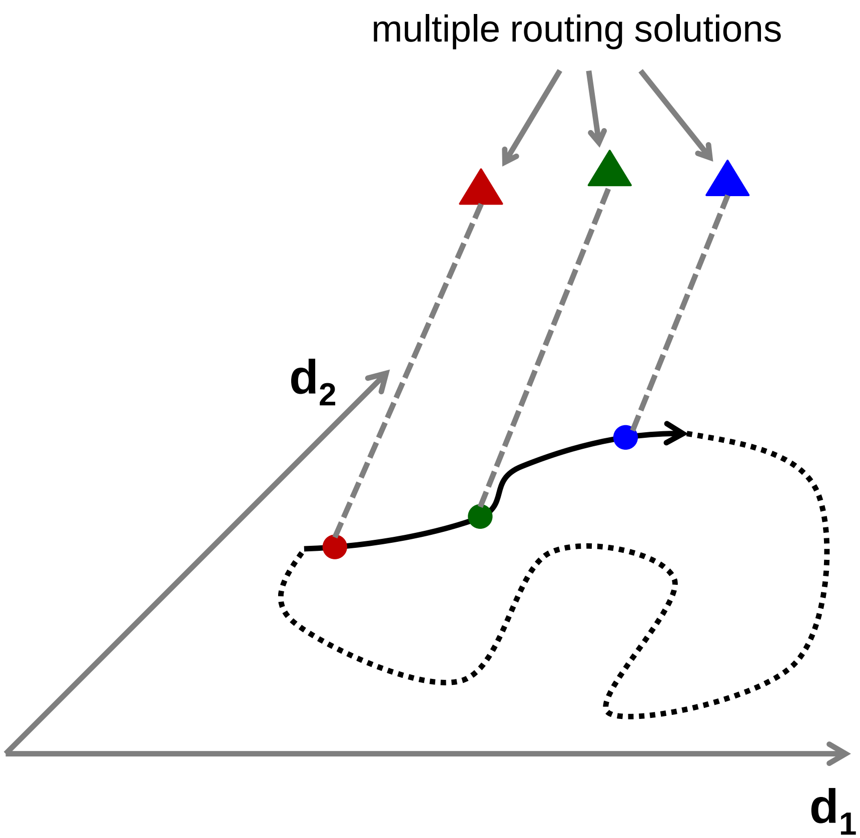


Always reconfigure

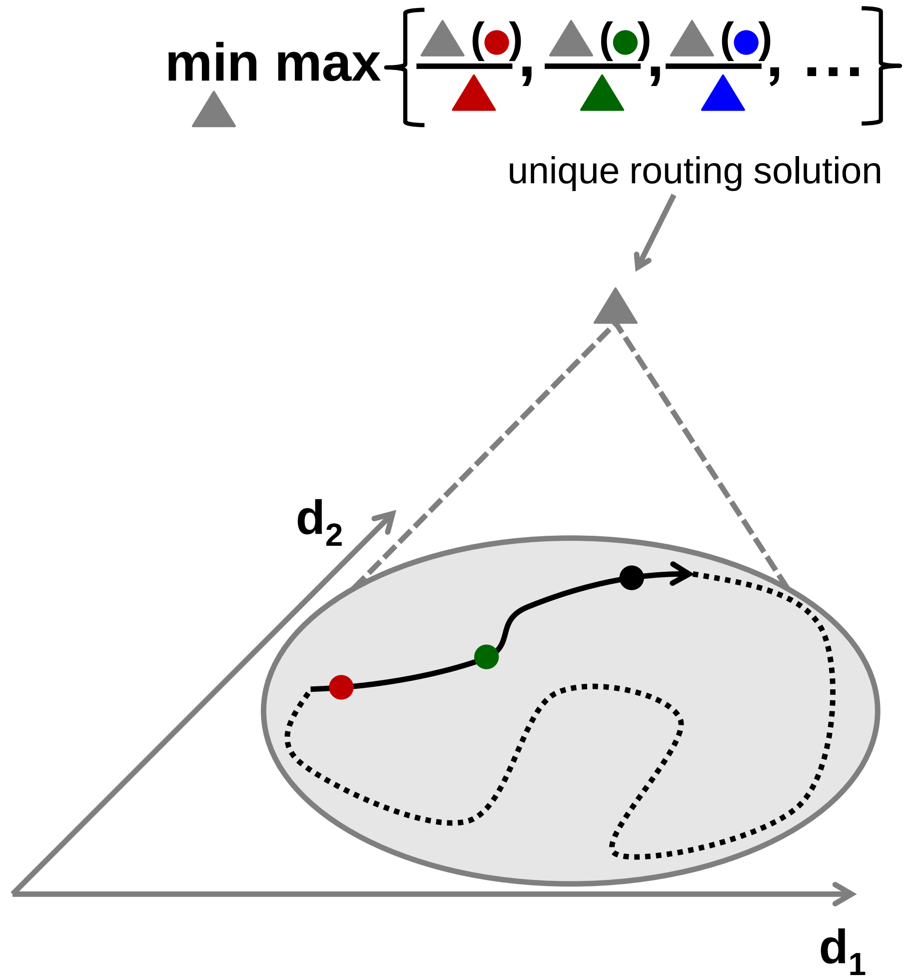
Never reconfigure

# Methods to deal with traffic uncertainty

**Dynamic TE**




**Stable TE**



# Robust Traffic Engineering

■ Two extreme solutions to reconfigure the network after traffic changes:

## Dynamic TE

- **Monitor** traffic over a **short period**
- Predict traffic for the next period
- Compute the optimal **TE policy for the next period**

## Stable TE

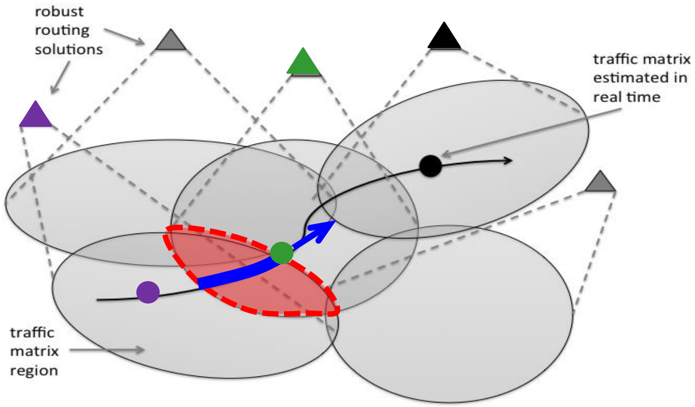
- **Monitor** traffic over a **long period**
- Consider a larger set of TMs (envelope)
- Compute a **single TE policy** for this set of TMs

Always reconfigure

Never reconfigure

## Robust TE

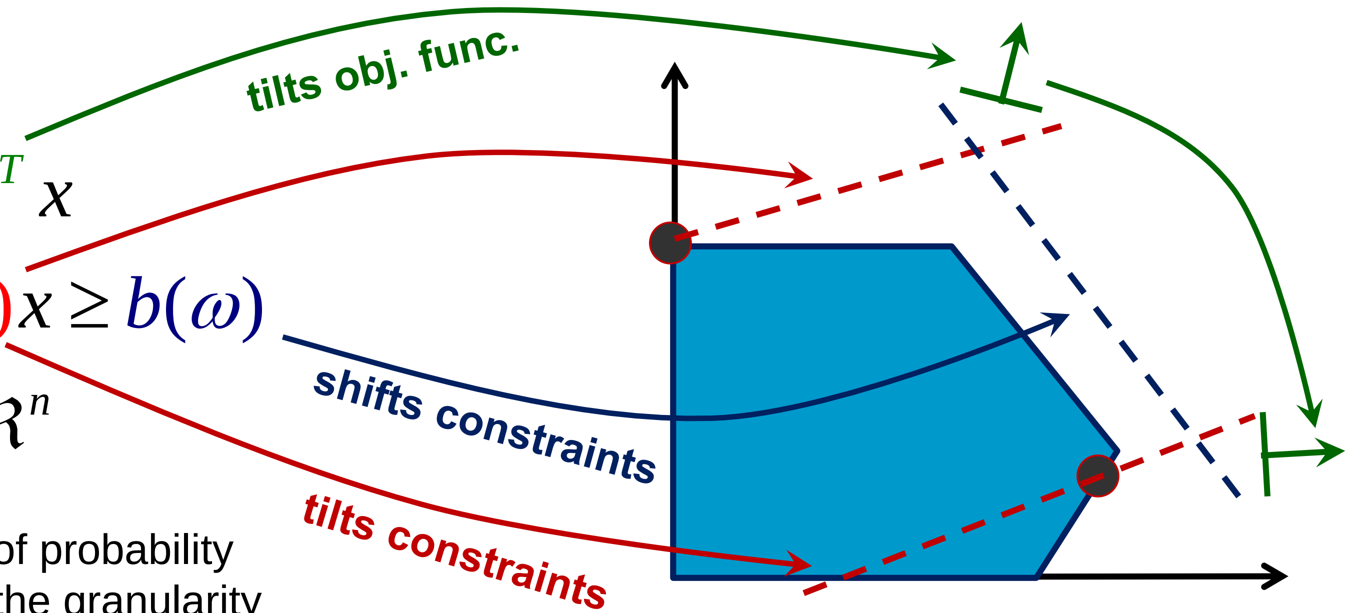
- Combines the strengths of both approaches
  - **Multiple configurations** to improve optimality
  - **Stability** of configurations (time overlap, routing similarity, prefetching)



# Traffic Engineering Under Uncertainty

- **Linear programming**:
  - Fast and optimal, but only with respect to the worst case.
  - Sensitive to variations of input (small variations can have huge effects).
- **Stochastic optimization**:
  - Robust, but huge problem to be solved.
  - Sensitive to the definition of the uncertainty (it needs exact historical data).

$$\begin{aligned} \min \quad & c(\omega)^T x \\ s.t. \quad & A(\omega)x \geq b(\omega) \\ & x \in \Re^n \end{aligned}$$




- Sensitive to definition of probability
- Complexity grow with the granularity

# Robust Traffic Optimization

M, N, R, and S are all feasible points:
- **N ➔ deterministic minimum**
- **R ➔ robust minimum**

network configurations where congestion happens

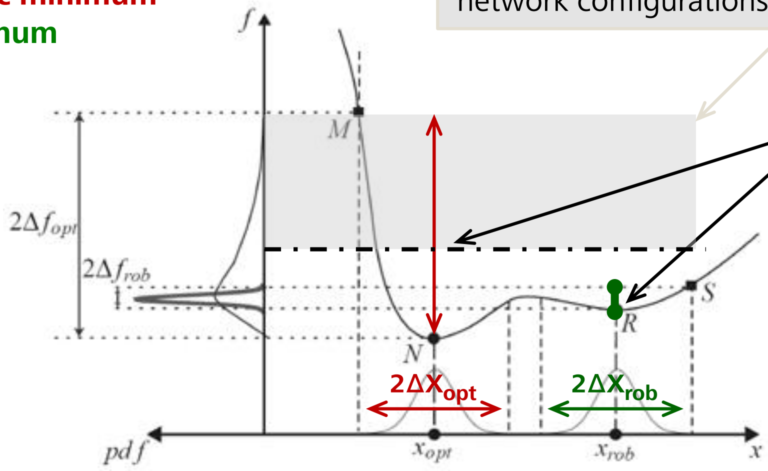



**Figure 1** – A robust solution *vs.* an optimal solution.

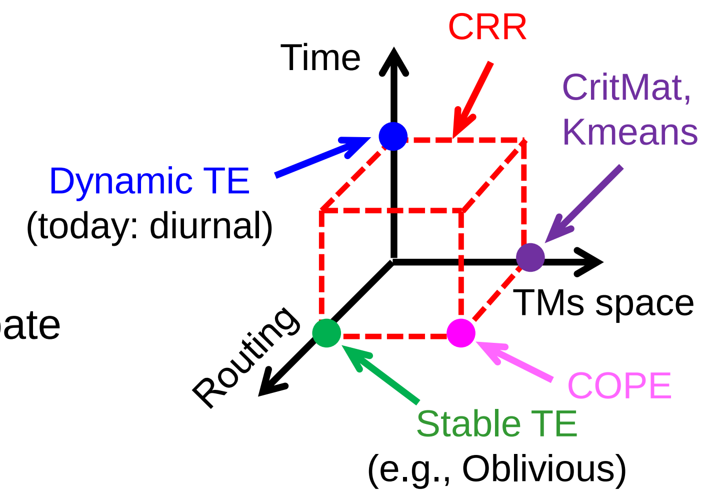
Same interval for decision variables (Δx)
Different variation of objective function (2Δf)

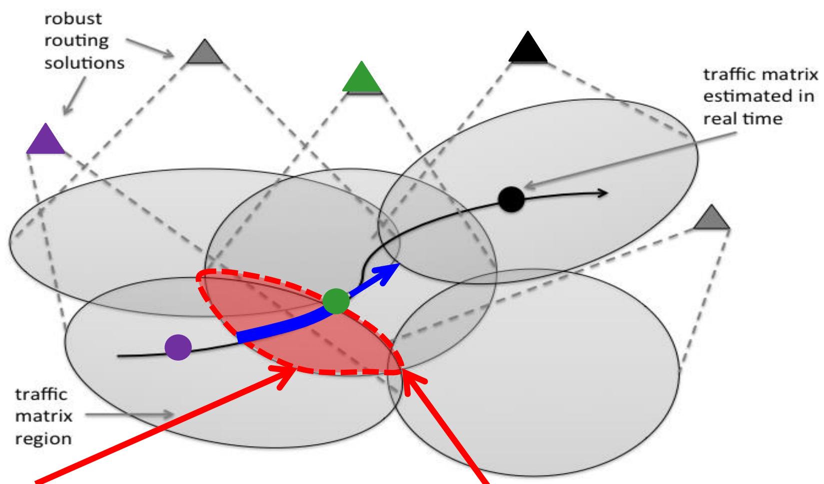
Search for feasible points
- with the lowest variation
- with the variation bounded in an area
- …

- **Pros:**
  - Robust Optimization (**RO**) **simplifies** modeling and optimization under uncertainty.
  - **RO** is **less sensitive to** low accuracy of **uncertainty** (noisy historical data/measurements).
  - **RO** permits to **reuse** fast **LP solvers** (like FlowEngine).
- **Challenges**:
  - How to **divide regions** for robust optimization ➔ Reconfigurations vs. optimality.

# Robust Traffic Optimization

- **Robust routing for different clusters** considers jointly:
  - **Temporal** continuum of TMs
  - **Space** continuum of TMs
  - Similarity of **routing** solutions applied to close TMs
- **Temporal overlap** among clusters leaves time to anticipate network reconfiguration




Using configuration for traffic matrix when its direction is will cause congestion

- Follow the direction of TM
- Anticipate the reconfiguration

This **region** leaves time to
1. **Observe the evolution** of the traffic
2. **Decide whether to reconfigure**
3. **Decide what to prefetch**

In this **region both TE configurations** and works with **a service/performance guarantee**.

# REAL-TIME FAIR RESOURCE ALLOCATION

REAL-TIME FAIR RESOURCE ALLOCATION IN DISTRIBUTED SOFTWARE DEFINED NETWORKS

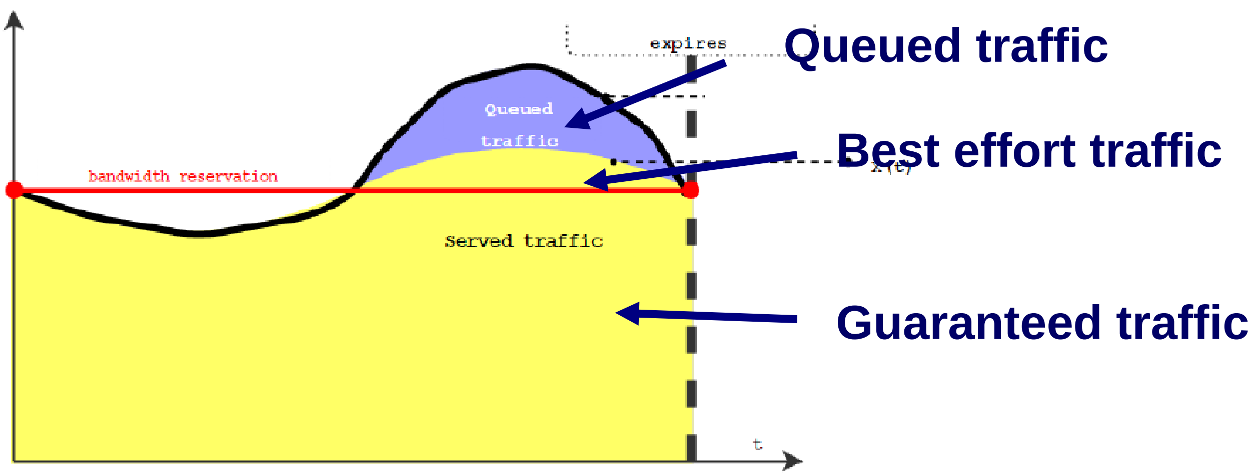
ZAID ALLYBOKUS, KONSTANTIN AVRACHENKOV, JÉRÉMIE LEGUAY, LORENZO MAGGI

ITC 2017

# Fair Resource Allocation Over Time

## MPLS for dynamic IP traffic

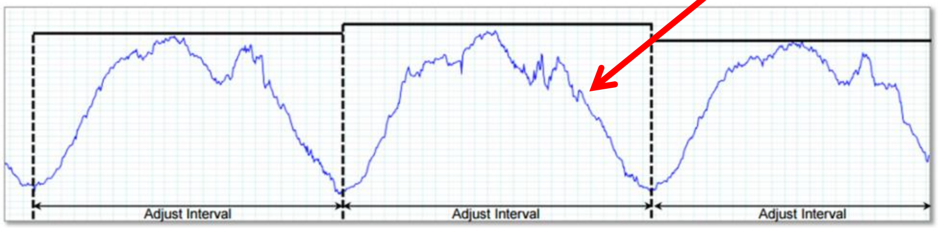
- Each LSP is configured by RSVP with a given amount of allocated bandwidth
- How do we configure the bandwidth on a particular LSP?
  - After all, IP networks are dynamic and packet switched.
  - Bandwidth usage can change and be unpredictable.

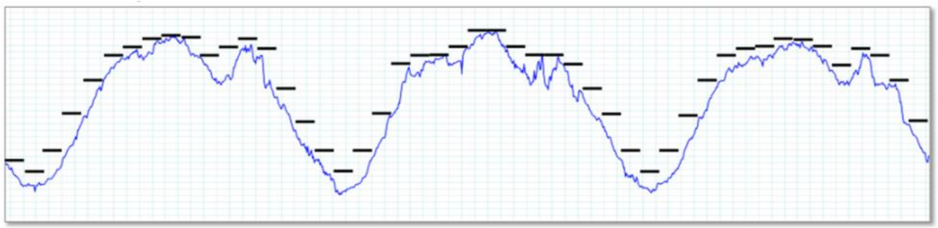



Reference:
https://www.nanog.org/sites/default/files/tues.general.steenbergen.autobandwidth.30.pdf

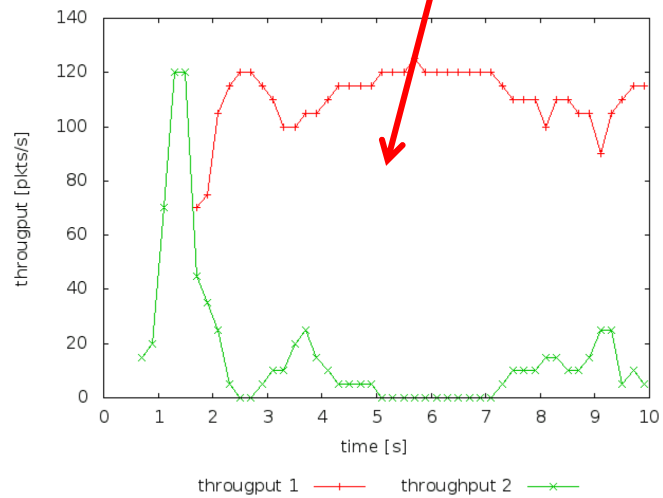
# Adjust bandwidth more efficiently

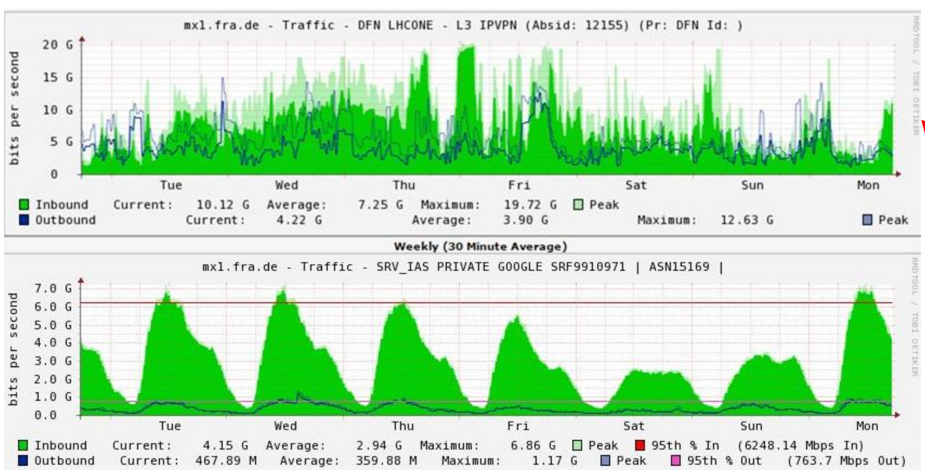
Avoid wasted resources



Avoid unfairness over time




Handle unpredictable traffic

GÉANT

# Our objectives in a nutshell

- Considering a set of flows carrying dynamic IP traffic
- The goal is to maximize utilities
  - Network utility (e.g., average amount of routed traffic)
    - Avoid needless bandwidth reservations
  - User utility (e.g., average traffic per user)
    - Avoid  congestions or packet losses
  - Ensure fairness
- Quickly react to sudden traffic changes
- Be ready for distributed SDN architectures

# Fair resource allocation

- Considering a set of flows *R* carrying dynamic IP traffic over a network of already established routes:
- The goal is to allocate resources to the set of flows while ensuring - fairness:

$$U_\alpha(x) = \sum_r w_r \frac{x_r^{1-\alpha}}{1-\alpha}, \alpha \neq 1, \quad U_1(x) = \sum_r w_r \log x_r, x \in \mathbf{R}_{++}^{|R|}$$

- A spectrum of fairness levels according to specific objectives: max-min (α=inf), proportional (α=1), max-throughput (α=0), min delay (α=2), …
- Weights $w_r$ may determine operational priorities of flows, accumulated traffic backlogs …

# Utility maximisation problem

Network of $N$ nodes and $J$ bi-directed edges (links). Each link $j$ has a capacity of $C_j \in \mathbf{R}_+$.
A set $R$ models:

- a set of requests $r \in R$ with a *weight* $w_r \in \mathbf{R}_+$ and *utility function* $U_r$
- a set of routes $r \in R$ s.t. $r = \{j_1, \ldots, j_t\} \subset J$

Equivalently:

$$\min \sum_{r \in R} g_r(x_r) \text{ s.t. } Ax \leq C$$

where $A = (a_{jr})_{jr}$ is the link-route incidence binary matrix:

$$a_{jr} = \begin{cases} 1 & \text{if } j \in r \\ 0 & \text{otherwise.} \end{cases}$$

**In our study:** $g_r = g_r^\alpha = -U_r^\alpha$.

$g_r$ is convex, non decreasing, proper and closed.

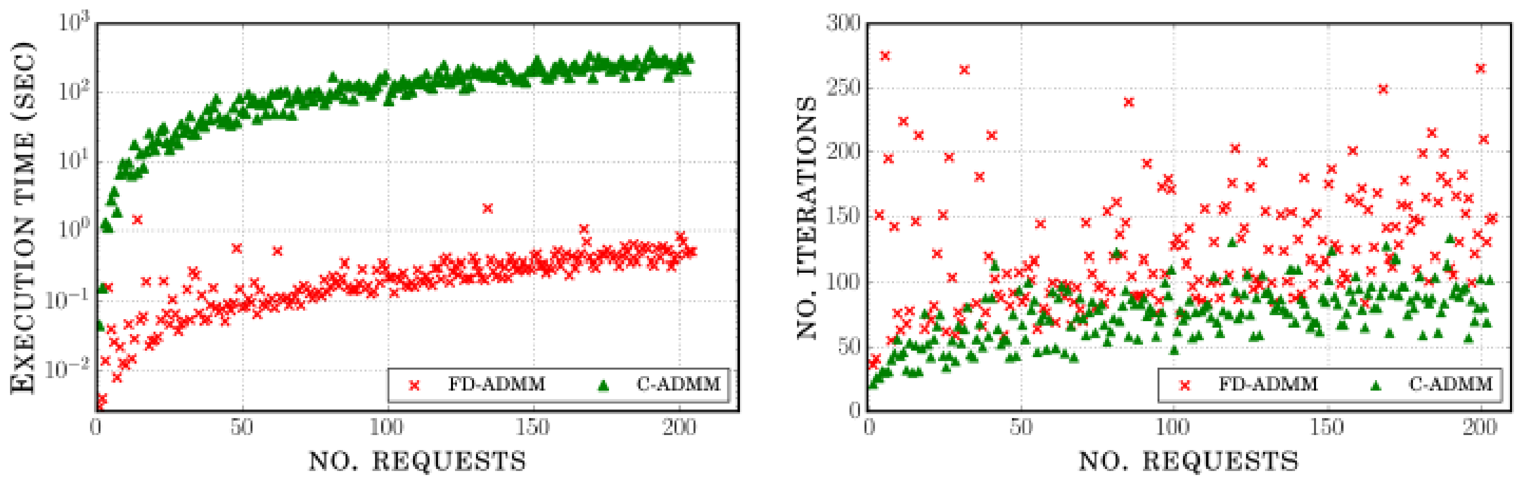
# Contribution vs SoTa

- State of the art: Lagrangian methods
  - Slow convergence rate $O(1/n^2)$
  - Violates feasibility
    - We could use projected sub-gradient but convergence is slower

- Our work: Distributed algorithm based on ADMM
  - Fast convergence rate: $O(1/n)$ in general and linear when the problem is strongly convex
  - Anytime algorithm: feasible solutions at all iterations
  - Well adapted to distributed SDN architectures
    - No need for compute intensive operations (i.e, global projection)

**Tools: convex optimization, utility maximisation, online optimization**

# Fast Distributed ADMM (FD-ADMM)




| ADMM Step | C-ADMM | FD-ADMM |
|---|---|---|
| Utility maximization step | Distributed | Distributed |
| Feasibility optimization step | Centralized | Distributed |
| Variables update step | Distributed | Distributed |

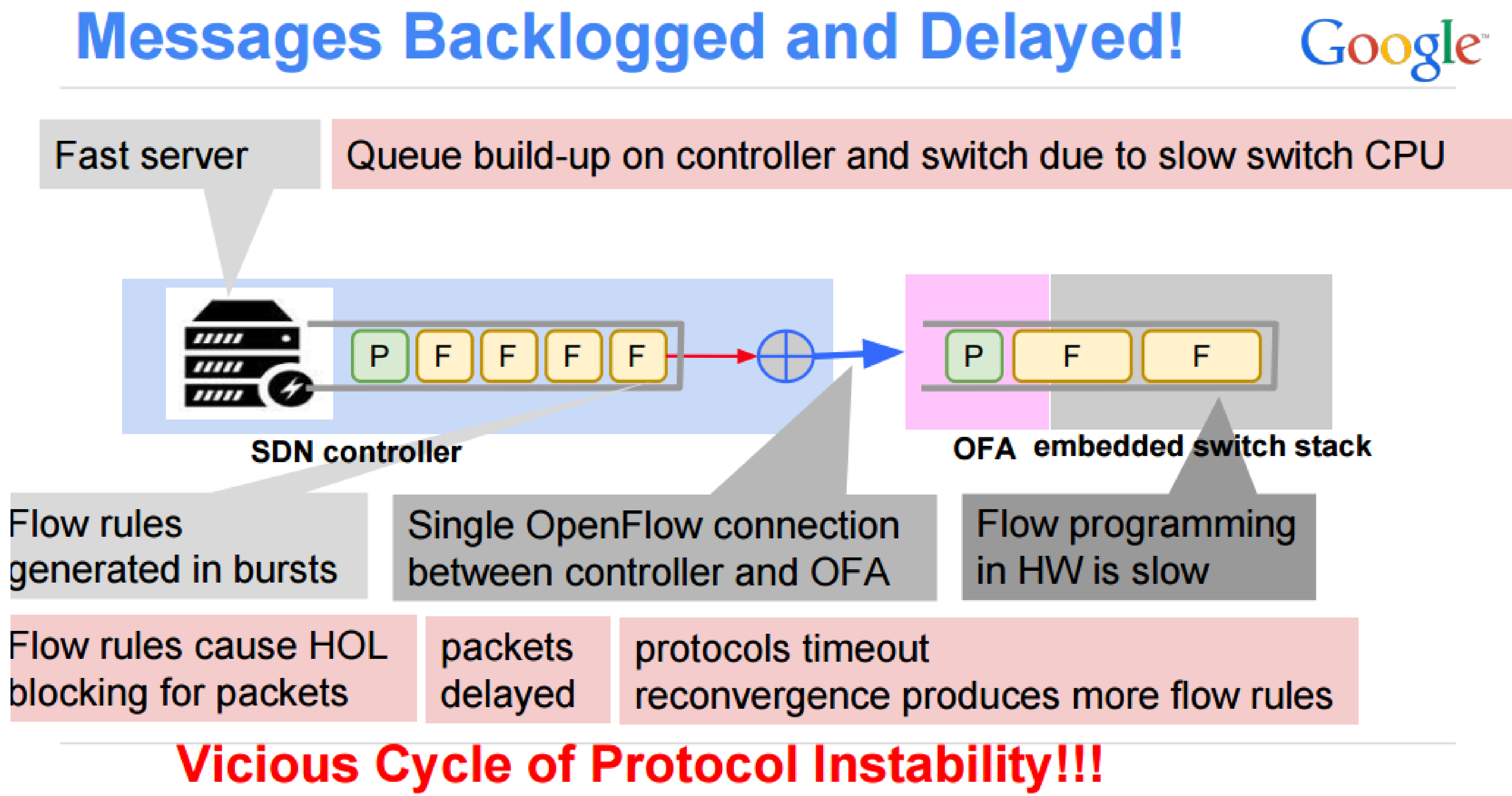
# CONTROLLING ROUTING RECONFIGURATION

CONTROLLING FLOW RECONFIGURATIONS IN SDN

S PARIS, A DESTOUNIS, L MAGGI, GS PASCHOS, J LEGUAY

IEEE INFOCOM 2016

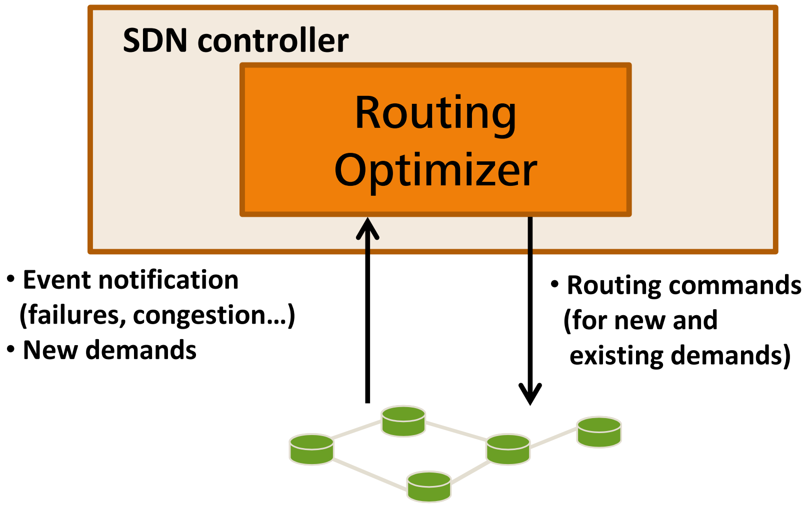
# Flow programming is slow (HW, control plane)

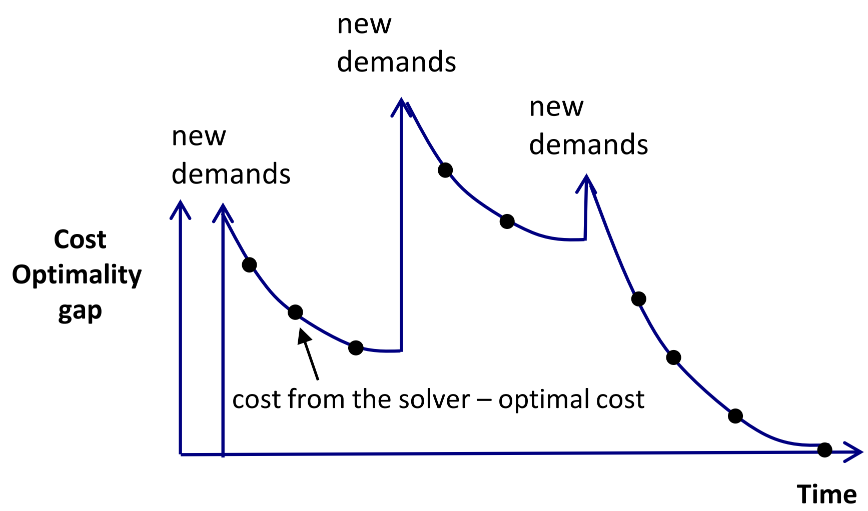



https://www.usenix.org/sites/default/files/conference/protected-files/atc15_slides_mandal.pdf

# Stability vs Optimality in Routing Systems

- System considerations
  - Flow programming in HW is slow
  - Control plane can only satisfy a limited reconfiguration rate
  - Routing solver issues a sequence of feasible solution

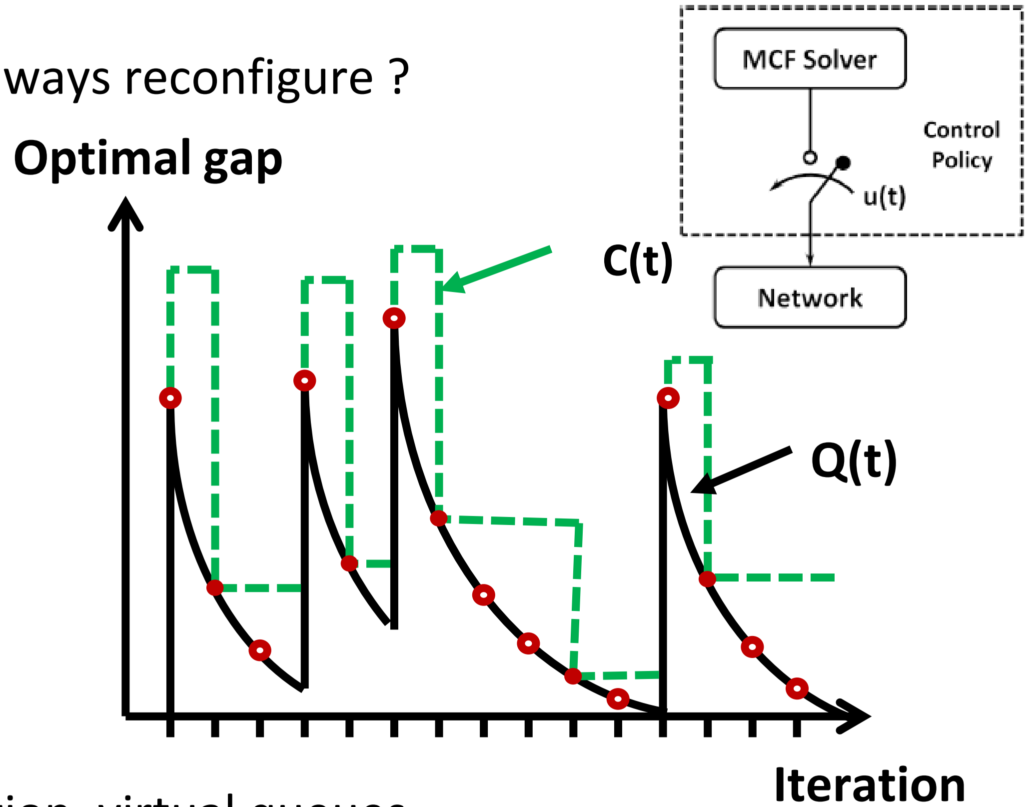



- Main problem
  - ➔ *Acheive a good trade off between optimality* and network stability

# Solution #2 Adaptive pre-filtering policy

- What if we do not want to always reconfigure ?
  - Compute iterative solutions
  - Use the following control
    - $u(t)=1$ (apply)
    - $u(t)=0$ (do not apply)




- Stochastic Optimization

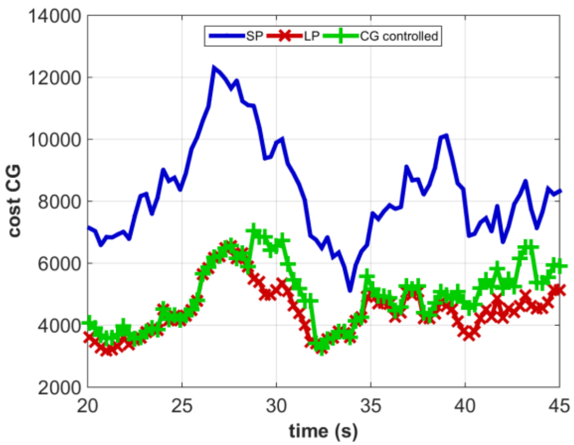
$$\min_{(u(t))_{t=0,1,\dots}} \quad \overline{C}$$

$$\text{s.t.} \quad \overline{u} \leq u_{\max}$$

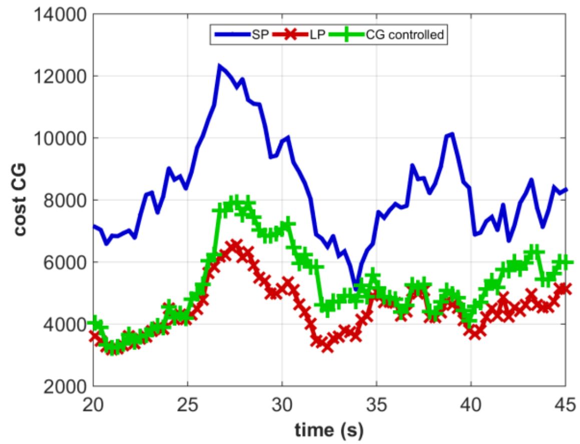
- Toolbox: Liapunov Optimization, virtual queues

We proposed a drift-plus-penalty strategy that minimizes routing cost while keeping the reconfiguration rate below a threshold

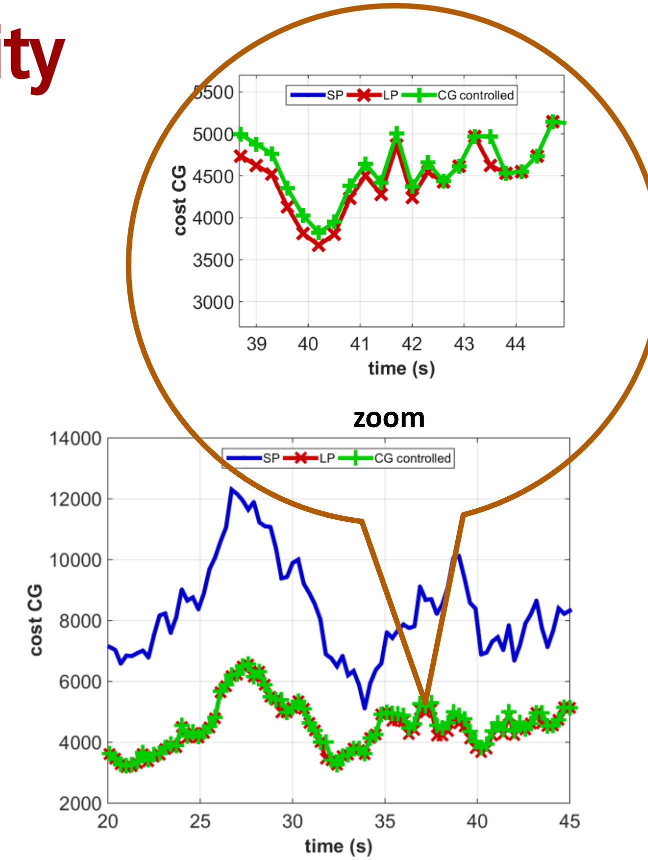
# Network simulation results: optimality




Random




Periodic




Drift-plus penalty

- Average routing cost: random (4975), periodic (5279), Optimal (4578)
- The optimal policy minimizes the cost while meeting the target reconfiguration rate

# Conclusion

- **SDN looks at flow problems under new perspectives**
  - Large problem instances
  - Under time constraints
  - Using commodity servers
    - Distributed and parallel computing
    - Machine Learning

- **Toolbox**
  - Combinatorial optimization
  - Online and expert algorithms
  - Convex optimization
  - Robust and stochastic optimization

# Selected publications from the team


- P. Medagliani, J. Leguay, M. Abdullah, M. Leconte, S. Paris.
  Global Optimization for Hash-based Splitting. IEEE Globecom 2016. Best paper award.
- "Domain Clustering for Inter-Domain Path Computation Speed-Up", by L. Maggi, J. Leguay, J. Cohen, P. Medagliani. Networks, Journal (submitted)
- "Virtual Function Placement for Service Chaining with Partial Orders and Anti-Affinity Rules". Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, E. Gourdin. Networks, Journal. 2017.
- "Minimum Cost SDN Routing with Reconfiguration Frequency Constraints", A. Destounis, S. Paris, L. Maggi, G. Paschos, J. Leguay. IEEE Infocom 2016
- "Online Bandwidth Calendaring: On-the-Fly Admission, Scheduling, and Path Computation". M. Dufour, S. Paris, J. Leguay, M. Draief. ICC 2017
- "Fair Distributed Resource Allocation in Software Defined Networks". Z. Allybokus, K. Avrachenkovy, J. Leguay, L. Maggi. ITC 2017.
- "Overlay Routing for Fast Video Transfers in CDN" by P. Medagliani, S. Paris, J. Leguay, L. Maggi, X. Chuangsong, H. Zhou. IEEE IM 2017.
- A Closed/Open-Loop cache update strategy by peeking into the future. L. Maggi and J. Leguay. Computer Communication Journal. 2017.
- Lorenzo Maggi, Lazaros Gkatzikis, Georgios Paschos, Jeremie Leguay. Adapting Caching to Audience Retention Rate: Which Video Chunk to Store?. Under submission.