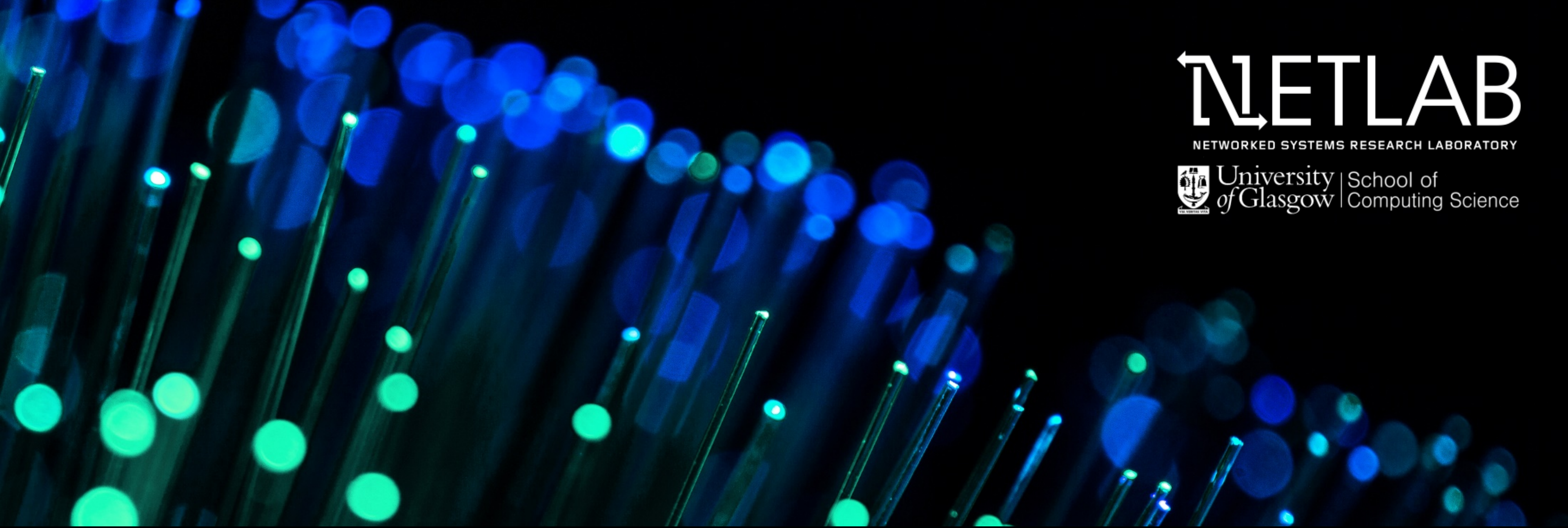
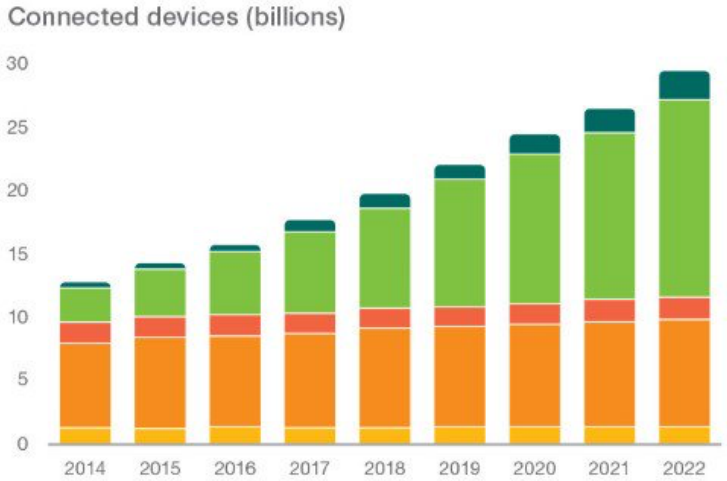


Dynamic, Latency-Optimal vNF Placement at the Network Edge

Richard Cziva, Christos Anagnostopoulos, Dimitrios P Pezaros | **University of Glasgow** | Richard.Cziva@glasgow.ac.uk | INFOCOM’18 | Honolulu, Hawaii

Number of connected devices



	2016	2022	CAGR
Wide-area IoT	0.4	2.1	30%
Short-range IoT	5.2	16	20%
PC/laptop/tablet	1.6	1.7	0%
Mobile phones	7.3	8.6	3%
Fixed phones	1.4	1.3	0%
	16 billion	29 billion	10%

Source: Ericsson IoT forecast
https://www.ericsson.com/en/mobility-report/internet-of-things-forecast

Richard Cziva, Christos Anagnostopoulos, Dimitrios P Pezaros | **University of Glasgow** | Richard.Cziva@glasgow.ac.uk | INFOCOM’18 | Honolulu, Hawaii

Increased expectations

- Future networks are expected to support
  - Context-aware
  - Ultra-reliable
  - User-specific network services
- Connected by
  - High-bandwidth and
  - **Low-latency** connections

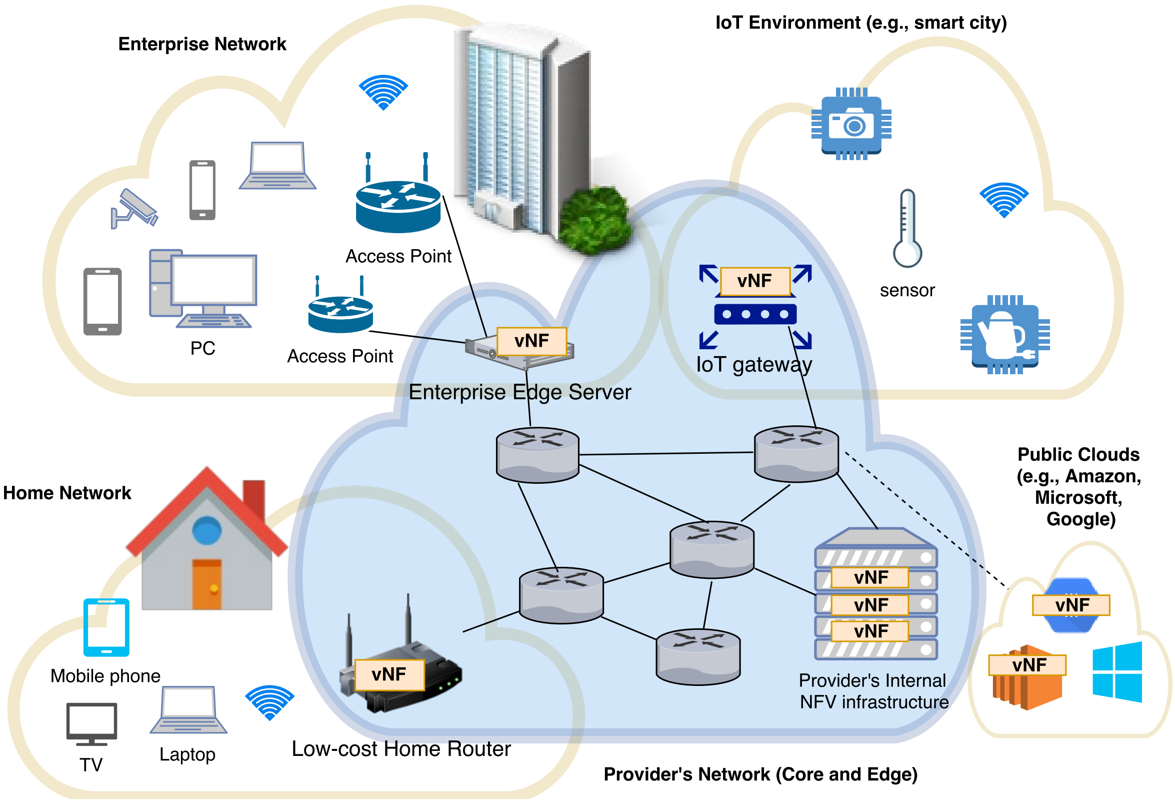
Example services: video content caches, user-specific firewalls, DDoS mitigation modules, etc.

Opportunities with Edge NFV

One way to solve these challenges is to
bring Network Function Virtualization to the Network Edge

- **Network Function Virtualization**
  - Decoupling network services from hardware and running them in software
  - Used in data centers, in the core of the network
  - *Lacks latency-optimal service orchestration*
- **Multi-Access Edge Computing**
  - Compute infrastructure at the edge of the network
  - Also known as “fog computing”
  - Close proximity to the user => low latency connectivity
  - Services at the edge save utilization for the core

Edge NFV Architecture

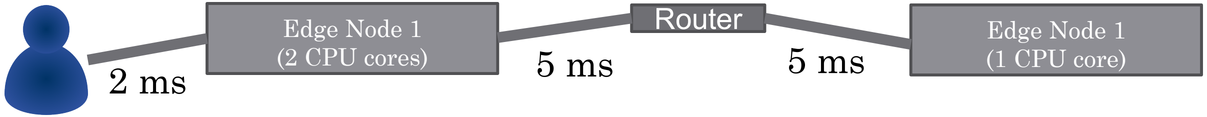


Richard Cziva, Christos Anagnostopoulos, Dimitrios P Pezaros | **University of Glasgow** | Richard.Cziva@glasgow.ac.uk | INFOCOM'18 | Honolulu, Hawaii

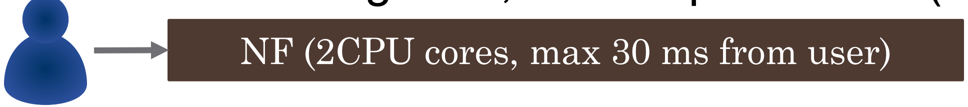
Latency-optimal vNF placement

- We focus on placing vNFs to **latency-optimal edge locations**
  - For each vNF association, we need to find a hosting device where a user-to-vNF end-to-end latency is minimal!

- **Given**: topology, hosting devices (with capabilities), latency on links, user's locations

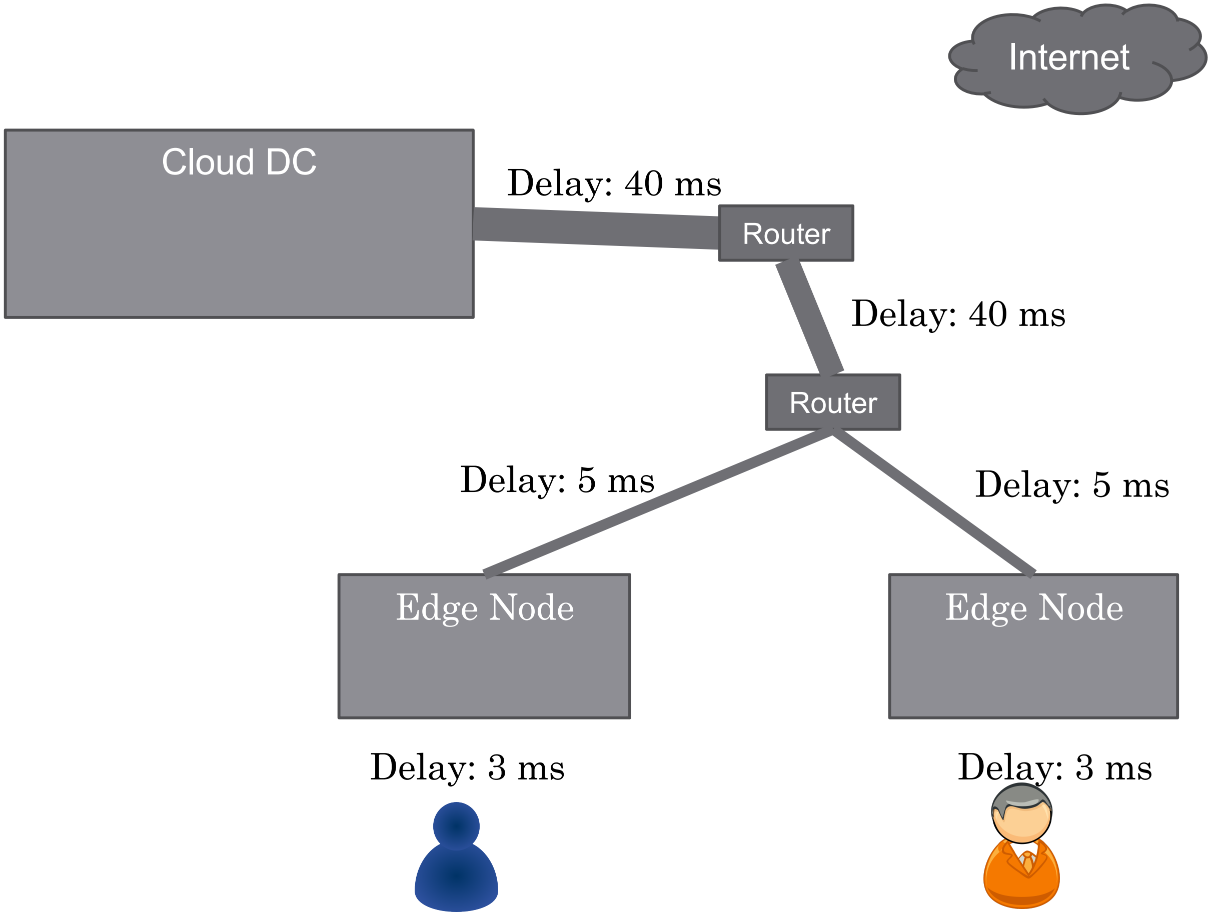


- **Problem input:** user to vNF assigment, vNF requirements (latency, compute)

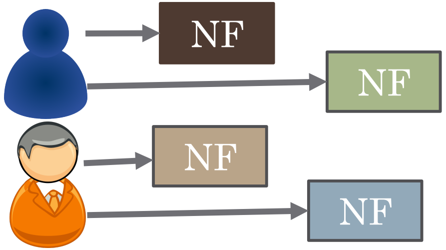


- **Output:** vNF to edge mapping

Richard Cziva, Christos Anagnostopoulos, Dimitrios P Pezaros | **University of Glasgow** | Richard.Cziva@glasgow.ac.uk | INFOCOM'18 | Honolulu, Hawaii



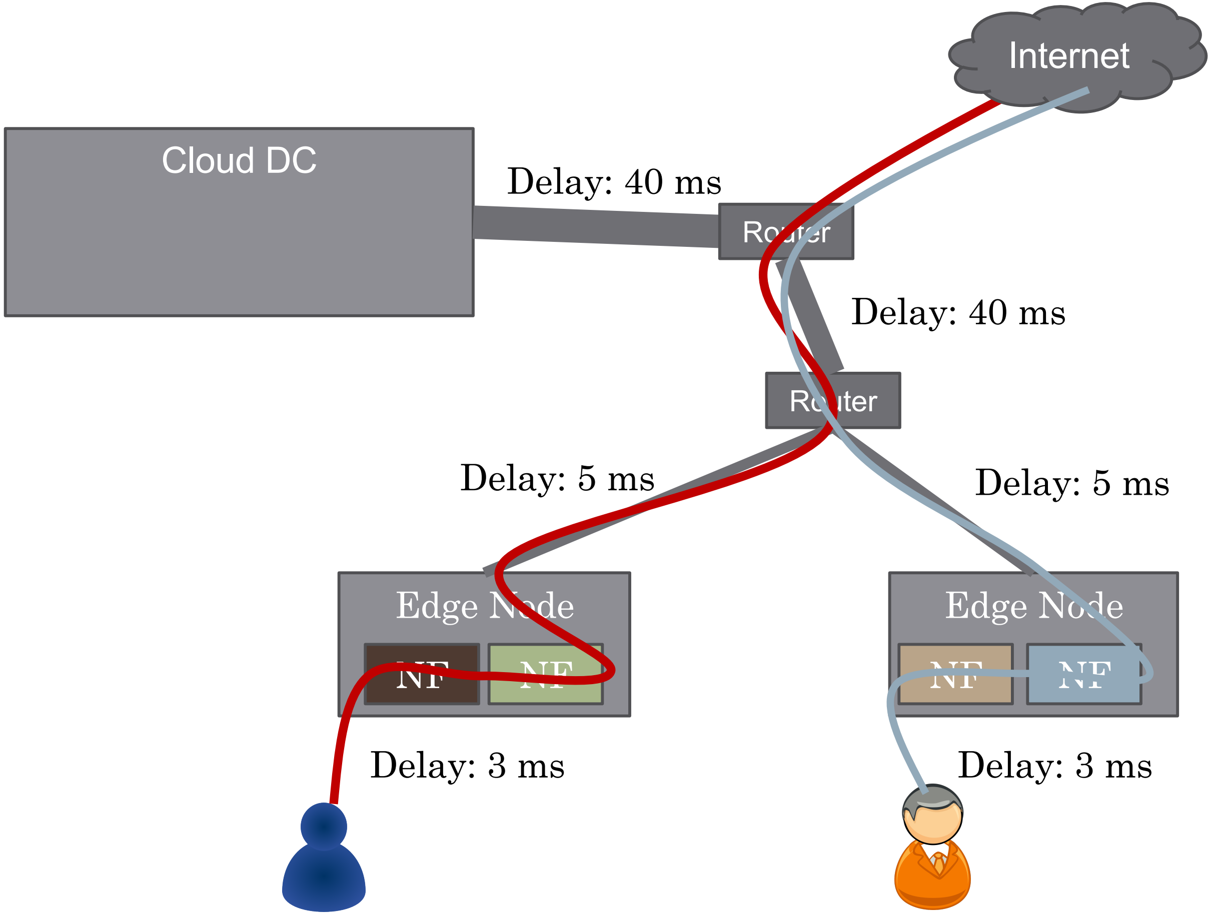
NF assignments



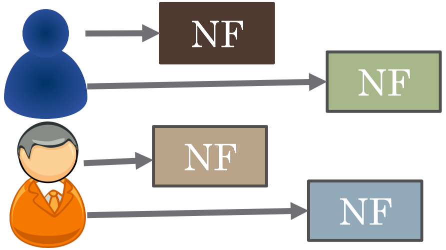
Edge properties:
- CPU / Memory available

Link properties:
- Bandwidth available (cost)
- Delay

Goal is to have a placement, where:
- All NFs are placed and traffic is router through them
- No overloading on links / edge devices



NF assignments



Edge properties:
- CPU / Memory available

Link properties:
- Bandwidth available (cost)
- Delay

Goal is to have a placement, where:
- All NFs are placed and traffic is router through them
- No overloading on links / edge devices

Network parameters	Description
$\mathbb{G} = (\mathbb{H}, \mathbb{E}, \mathbb{U})$	Graph of the physical network.
$\mathbb{H} = \{h_1, h_2, h_j, \dots, h_H\}$	Compute hosts (e.g., edge devices) within the network.
$\mathbb{E} = \{e_1, e_2, e_m, \dots, e_E\}$	All physical links in the network.
$\mathbb{U} = \{u_1, u_2, u_o, \dots, u_U\}$	All users associated with network functions.
$\mathbb{P} = \{p_1, p_2, p_k, \dots, p_P\}$	All paths in the network.
W_j	Hardware capacity $\{cpu, memory, io\}$ of the hosts $h_j \in H$.
C_m	Capacity of the link $e_m \in E$.
A_m	Latency on the link $e_m \in E$.
Z_k	Last host in path $p_k \in P$.
vNF parameters	**Description**
$\mathbb{N} = \{n_1^1, n_2^2, n_i^o, \dots, n_N^U\}$	Network functions to allocate, where the vNF $n_i^o \in \mathbb{N}$ is associated to user $u_o \in \mathbb{U}$.
R_i	vNF's host requirements $\{cpu, memory, io\}$ of vNF $n_i \in \mathbb{N}$.
θ_i	The maximum latency vNF $n_i \in N$ tolerates from its user.
Derived parameters	**Description**
b_{ijk}	Bandwidth required between the user and the vNF n_i in case it is hosted at h_j using the path p_k. Derived from the physical topology and the vNF requests.
l_{ijk}	Latency between the user of the vNF n_i in case it is hosted at h_j and uses the path p_k. Derived from the physical topology and the vNF requests.
Variables	**Description**
X_{ijk}	Binary decision variable denoting if n_i is hosted at h_j using the path p_k or not.

Edge vNF Placement ILP

Decision variable

$$X_{ijk} = \begin{cases} 1 & \text{if we allocate } n_i^o \text{ to } h_j \text{ using path } p_k \\ 0 & \text{otherwise} \end{cases}$$

Objective function

$$\min . \sum_{p_k \in \mathbb{P}} \sum_{n_i^o \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} X_{ijk} l_{ijk}$$

Constraints

$$\sum_{n_i^o \in \mathbb{N}} \sum_{p_k \in \mathbb{P}} X_{ijk} R_i < W_j, \forall h_j \in \mathbb{H} \quad (3)$$ *Hardware limitations*

$$\sum_{h_j \in \mathbb{H}} \sum_{p_k \in \mathbb{P}} X_{ijk} l_{ijk} < \theta_i, \forall n_i^o \in \mathbb{N} \quad (4)$$ *Maximum latency*

$$\sum_{h_j \in \mathbb{H}} X_{ijk} = 1, \forall n_i^o \in \mathbb{N}, \forall p_k \in \mathbb{P} \quad (5)$$ *Allocate a vNF to 1 host*

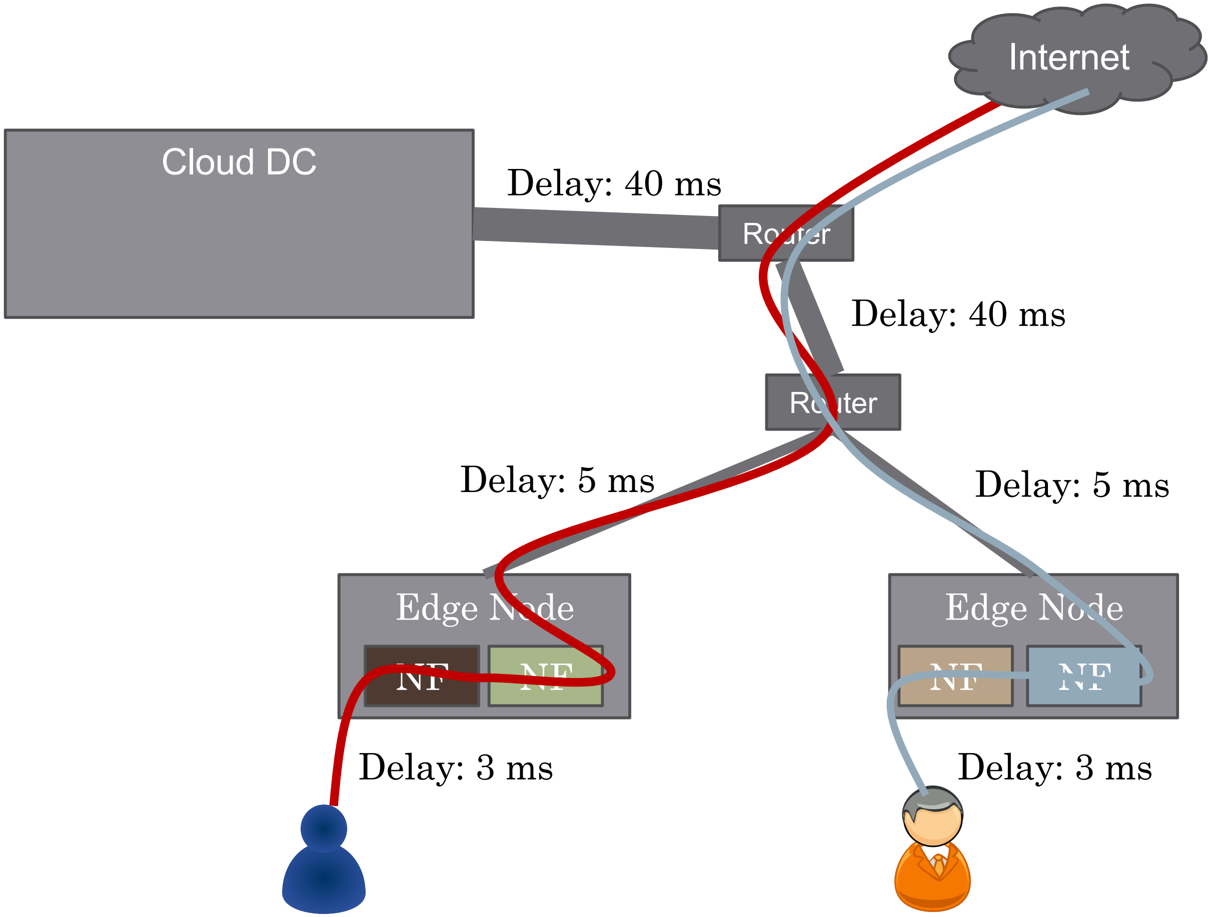
$$\sum_{h_j \in \mathbb{H}} X_{ijk} b_{ijk} < C_m, \forall e_m \in p_k, \forall p_k \in \mathbb{P} \quad (6)$$ *Bandwidth constraint*

$$X_{ijk} = 0, n_i^o \neq Z_k, \forall n_i^o \in \mathbb{N}, \forall p_k \in \mathbb{P}, \forall h_i \in \mathbb{H} \quad (7)$$ *Valid path constraint*

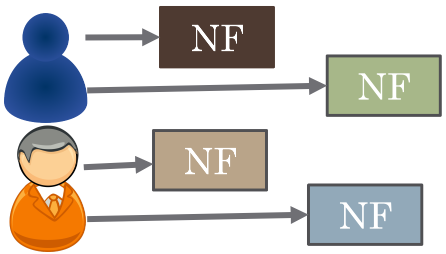
Are we done?

- The ILP allocates vNFs to latency-optimal location. However:
  - User's move between edge devices
  - Latencies change on links frequently
    - Other users impact traffic / congestion on the path
- These all impact the once optimal allocation!

We need dynamic re-allocation of edge vNFs to keep allocation latency-optimal!



NF assignments



Edge properties:
- CPU / Memory available

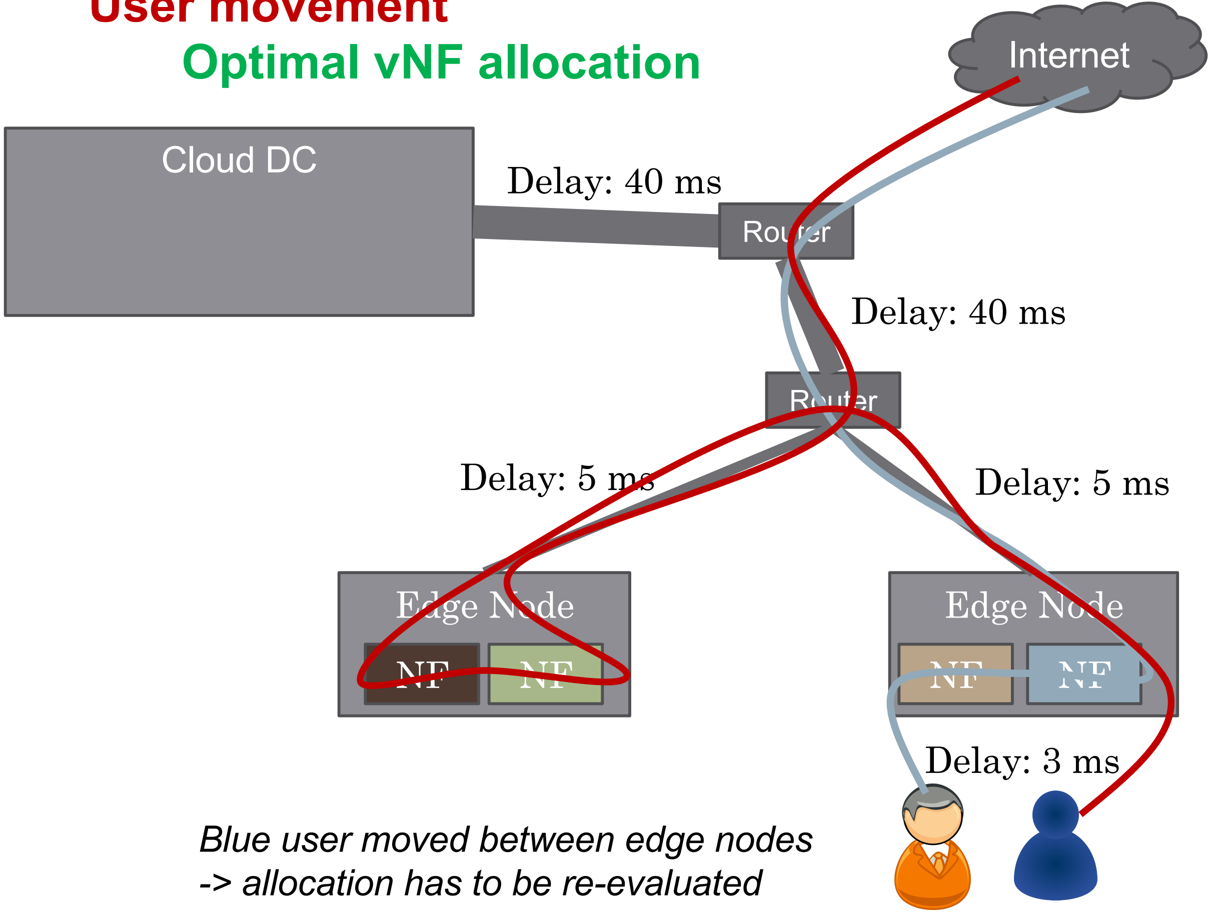
Link properties:
- Bandwidth available (cost)
- Delay

Goal is to have a placement, where:
- All NFs are placed and traffic is router through them
- No overloading on links / edge devices

Richard Cziva, Christos Anagnostopoulos, Dimitrios P Pezaros | **University of Glasgow** | Richard.Cziva@glasgow.ac.uk | INFOCOM'18 | Honolulu, Hawaii

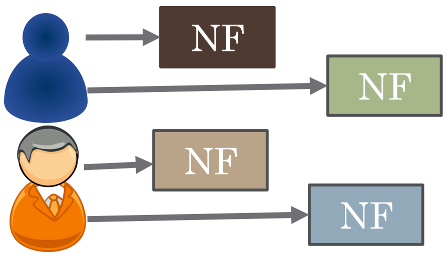
User movement

Optimal vNF allocation



Blue user moved between edge nodes
-> allocation has to be re-evaluated

NF assignments

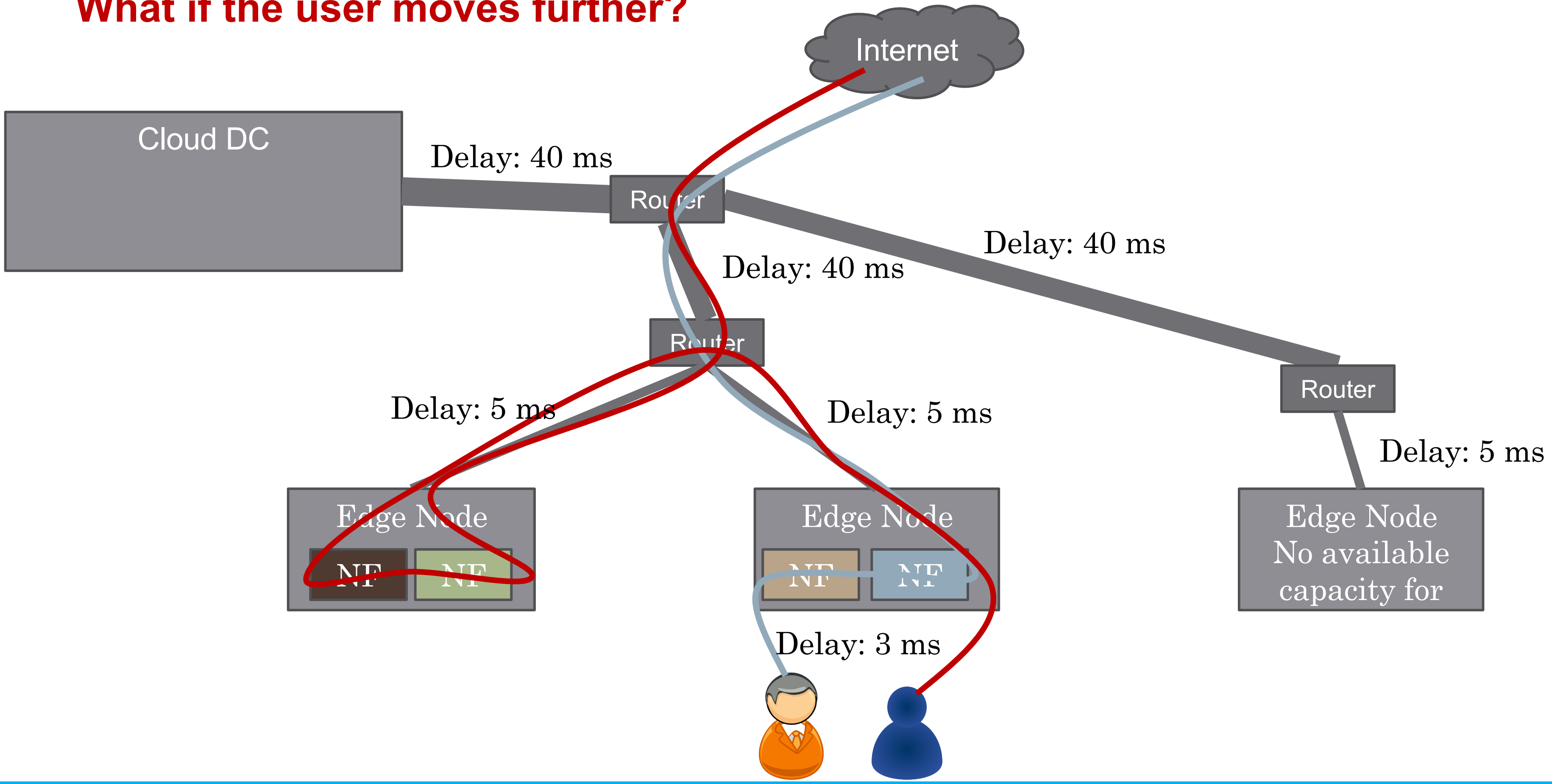


Edge properties:
- CPU / Memory available

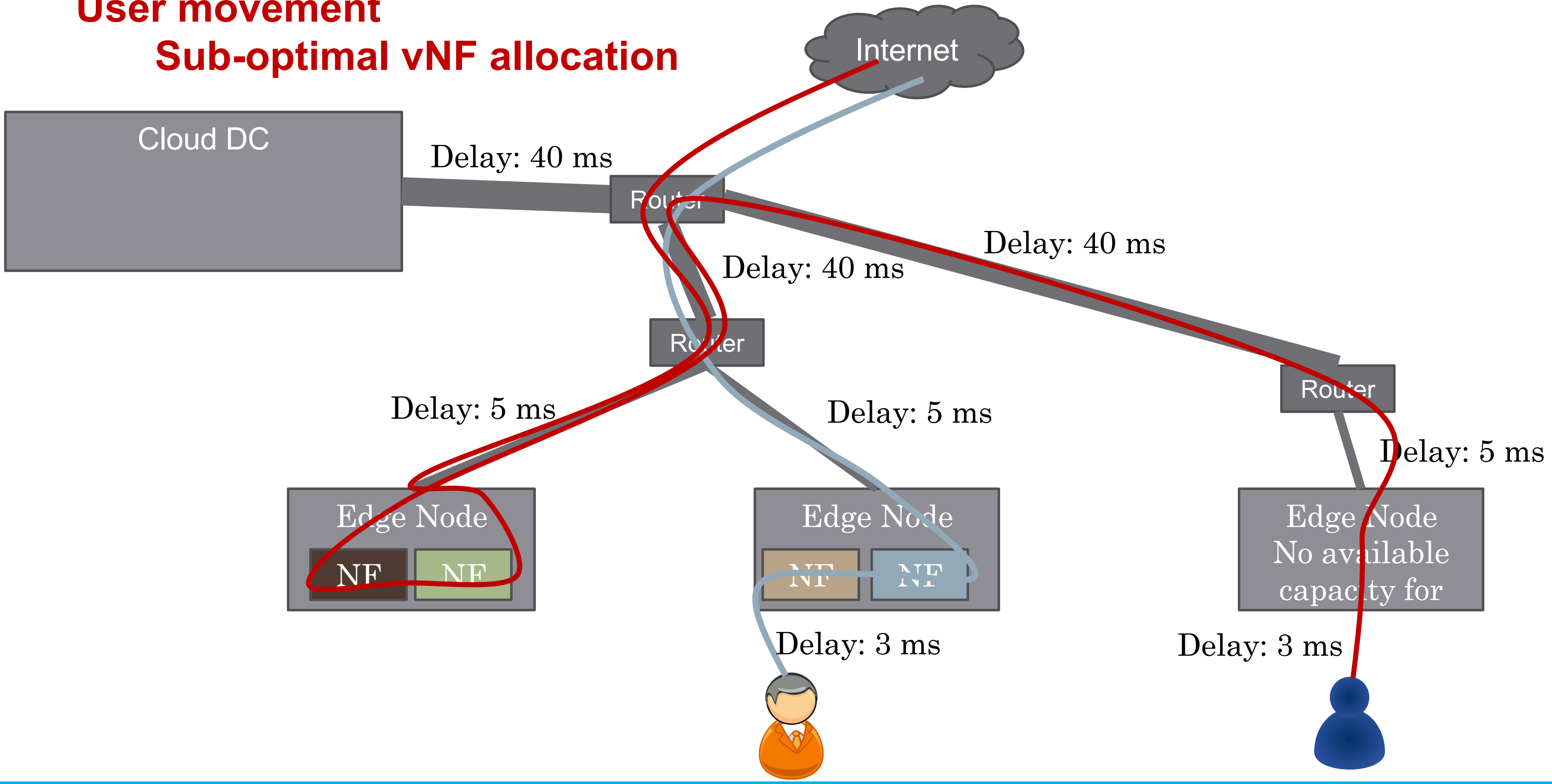
Link properties:
- Bandwidth available (cost)
- Delay

Goal is to have a placement, where:
- All NFs are placed and traffic is router through them
- No overloading on links / edge devices

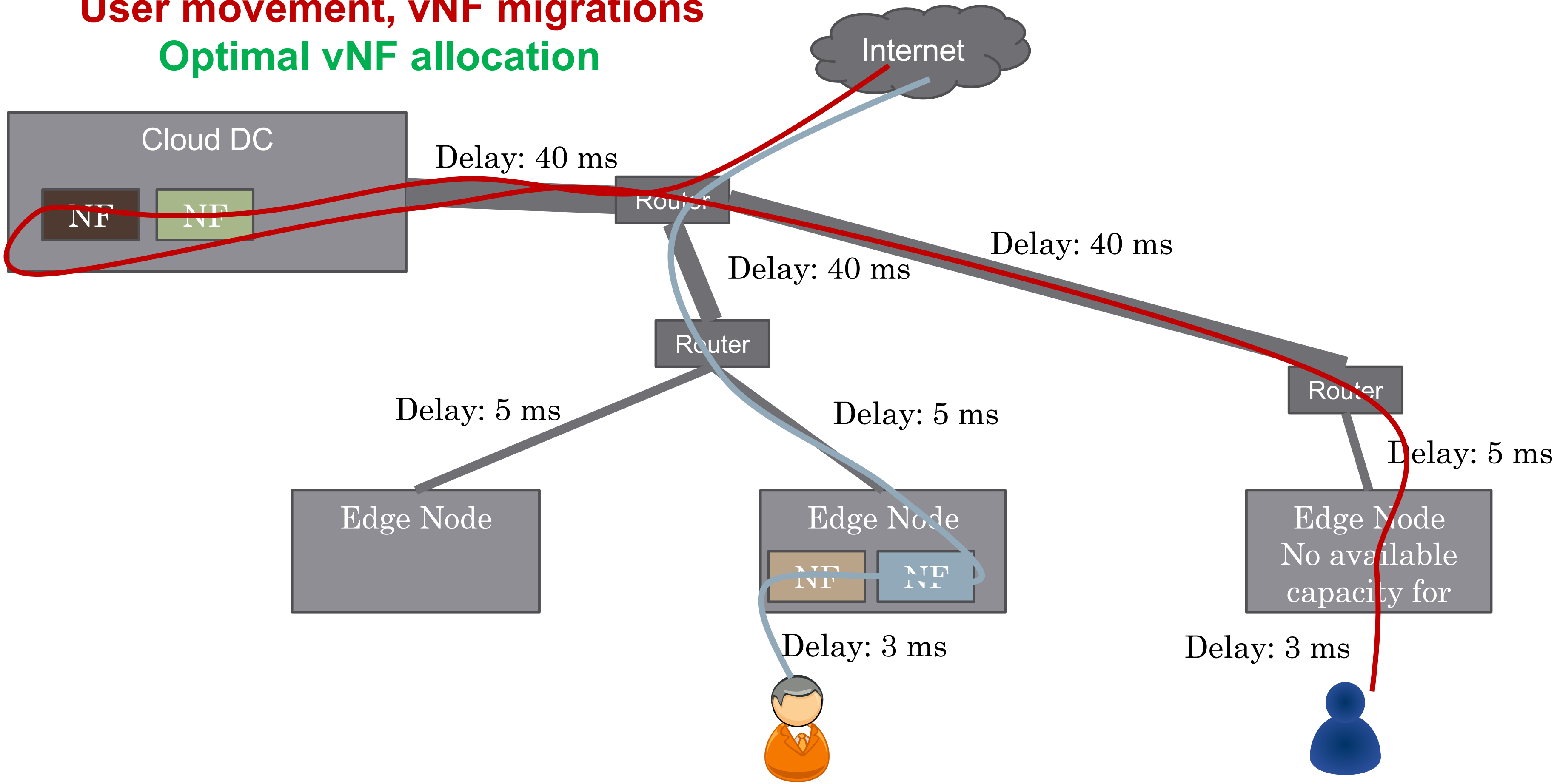
What if the user moves further?



User movement
Sub-optimal vNF allocation



User movement, vNF migrations
Optimal vNF allocation



Latency violations

- Assume that each vNF has a latency violation threshold that is a maximum latency the vNF should get from the user. This is θ_l
  - For instance a cache vNF can have 20 ms for this value, while a control plane vNF can have 150 ms
- Latency can not be guaranteed 100%, so the system will experience latency violations frequently
- Upcoming latency violations can be mitigated with a new latency-optimal vNF placement (but that costs migrations and placement calculation)

Goal: minimize latency violations, while keeping number of vNF migrations low

So, the new question is:

How often (when) do we rearrange vNFs?

Every time we can

- easy to implement, always latency-optimal allocation
- way too many migrations

Periodically

- easy to implement, easy to predict migrations
- can results in too many latency violations, if the period is too long

Optimal time

- low number of latency violations and low number of migrations

How do we get this "optimal time"?

- Counting latency violations experienced:

$$\mathcal{M}_{t\to\tau} = \sum_{ijk} I(x^t_{ijk}, x^\tau_{ijk}),$$

Migration cost between placements

$$L_t = \sum_i L^i_t \qquad L^i_t = \sum_j \sum_k L^t_{ijk}, \qquad L^t_{ijk} = \begin{cases} 1 & \text{if} \quad l^t_{ijk} > \theta_i \\ 0 & \text{otherwise} \end{cases}$$

$$f(Y_t) = \begin{cases} Y_t & \text{if} \quad Y_t \le \Theta, \\ \lambda \mathbb{E}[\mathcal{M}_0] & \text{if } Y_t > \Theta, \end{cases}$$

Reward function

$$Y_t = \sum_{k=0}^{t} L_k.$$

Cumulative sum of all violations at time t

- The challenge is to find the (optimal stopping) time instance t^* for deriving an optimal placement for the vNFs, such that Y_t be as close to the system's maximum tolerance Θ as possible

Problem 2. *Find the optimal stopping time t^* where the supremum in (14) is attained:*

$$\sup_{t\ge 0} \mathbb{E}[f(Y_t)]. \qquad (14)$$

How do we get this “optimal time”?

Theorem 2. *Given an initial optimal vNF placement $\mathcal{I}_0$ at time $t = 0$, we re-evaluate the optimal placement $\mathcal{I}_t$ at time instance t such that:*

$$\inf_{\tau \leq 0} \{\tau : \sum_{\ell=0}^{\Theta - Y_\tau} \ell P(L = \ell) \leq (Y_\tau - \lambda \mathbb{E}[\mathcal{M}_0])(1 - F_L(\Theta - Y_\tau))\} \quad (16)$$

where $F_L(\ell) = \sum_{l=0}^{\ell} P(L = l)$ and $P(L = \ell)$ is the cumulative distribution and mass function of L in (11), respectively.

For deriving the 1-sla, we have to stop at the first time instance t where $\mathbb{E}[f(Y_{t+1})|Y_t \leq \Theta] \leq Y_t$, that is, at that t:

$$\sum_{\ell=0}^{\Theta - Y_t} \ell P(L = \ell) + (Y_t - \lambda \mathbb{E}[\mathcal{M}_0]) F_L(\Theta - Y_t) + \lambda \mathbb{E}[\mathcal{M}_0] \leq Y_t,$$

Please find proof + solution fundamentals in the paper.

Note: we take only previous observations to make a decision.

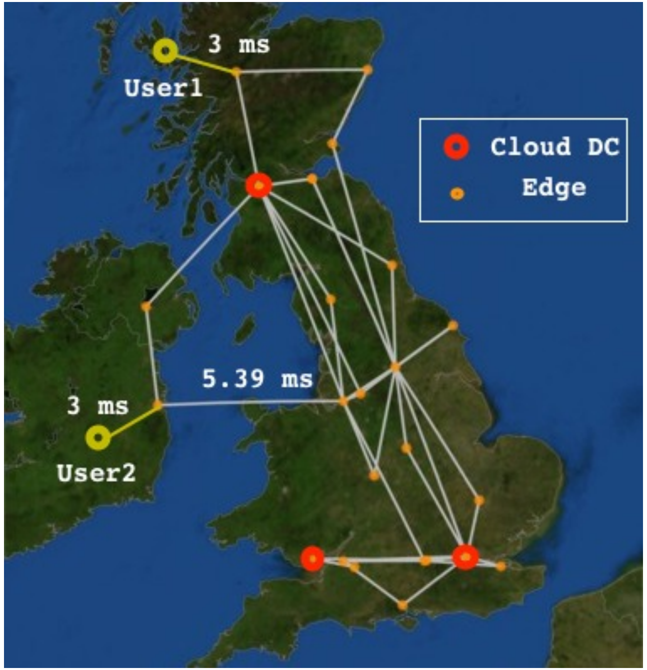
Evaluation

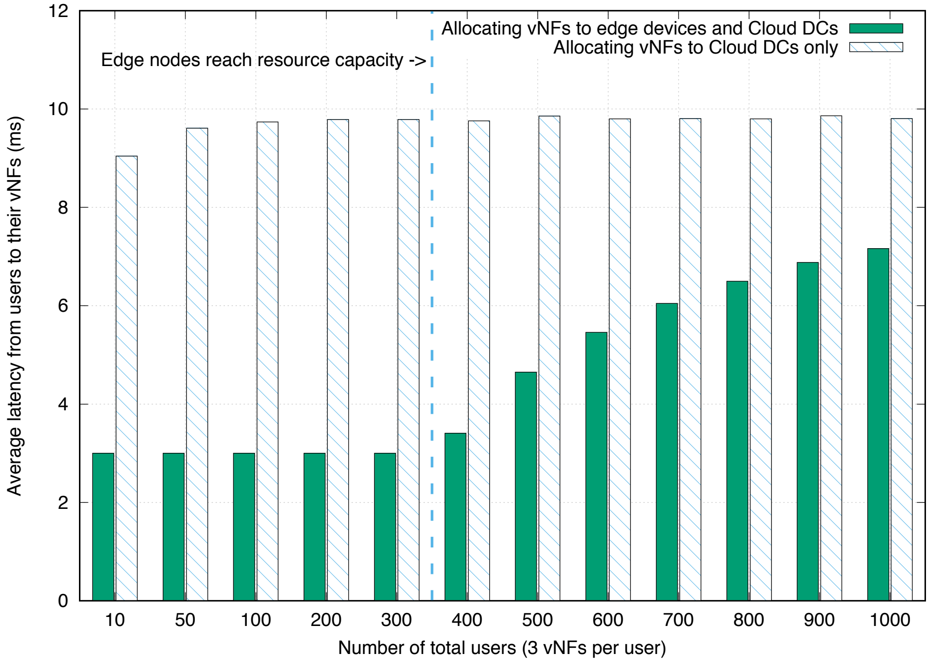
- We have divided the evaluation into two parts:
  - Latency-optimal allocation
  - Placement scheduling (dynamic extension)
- Simulation environment:
  - Gurobi solver used for ILP (with Python binding)
  - Python implementation for the optimal stopping time triggering the solver at the optimal stopping time

TABLE II: Latency tolerance of different vNF types

Type of network function	**Maximum delay**
Real-time (e.g., packet processing functions)	10 ms
Near real-time (e.g., control plane functions)	30 ms
Non real-time (e.g., management functions)	100 ms

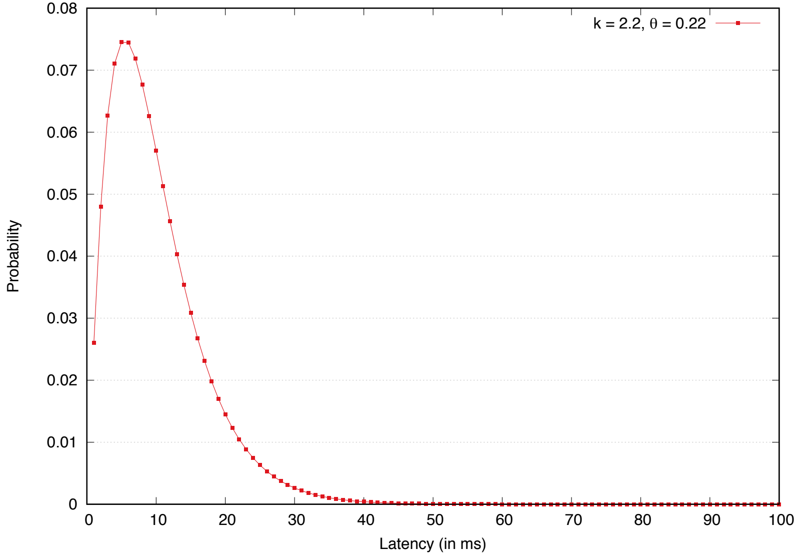
Edge vNF allocation

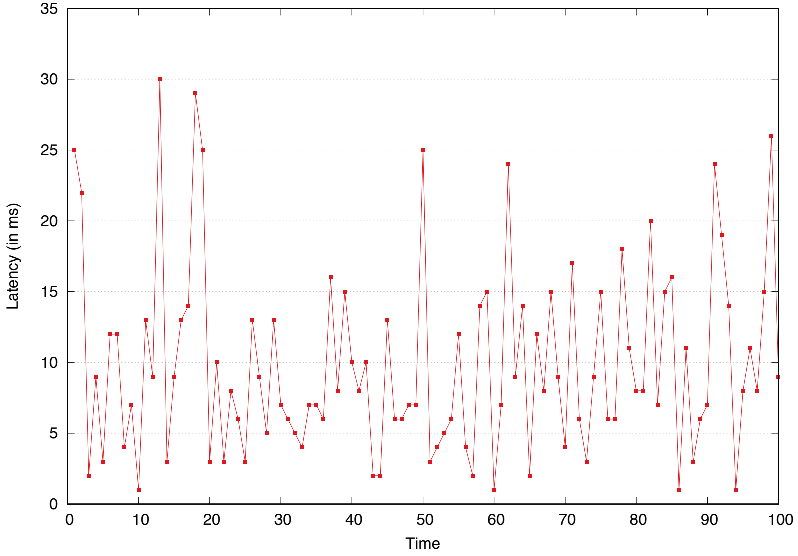




Richard Cziva, Christos Anagnostopoulos, Dimitrios P Pezaros | **University of Glasgow** | Richard.Cziva@glasgow.ac.uk | INFOCOM'18 | Honolulu, Hawaii

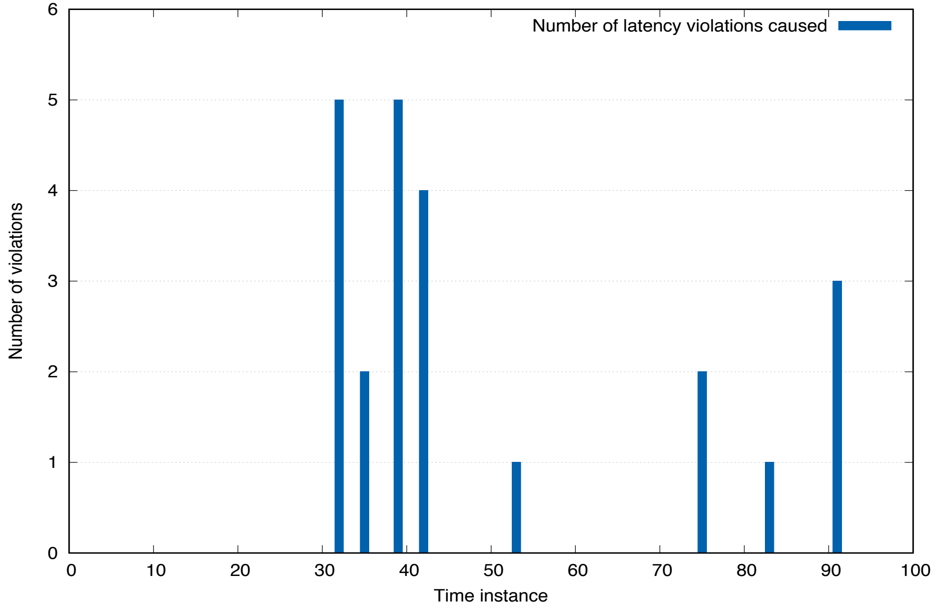
Latency fluctuations

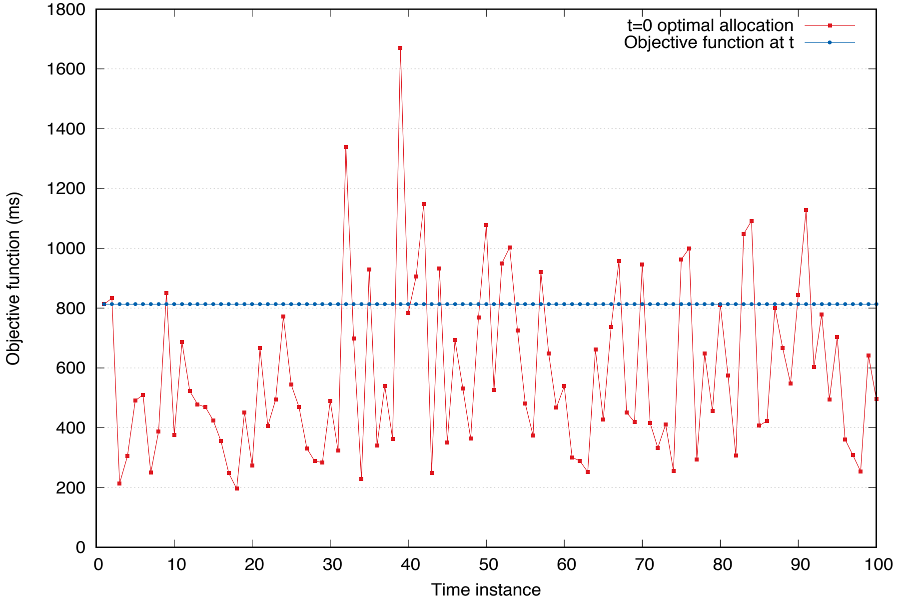




Based on empirical data collected with Ruru.

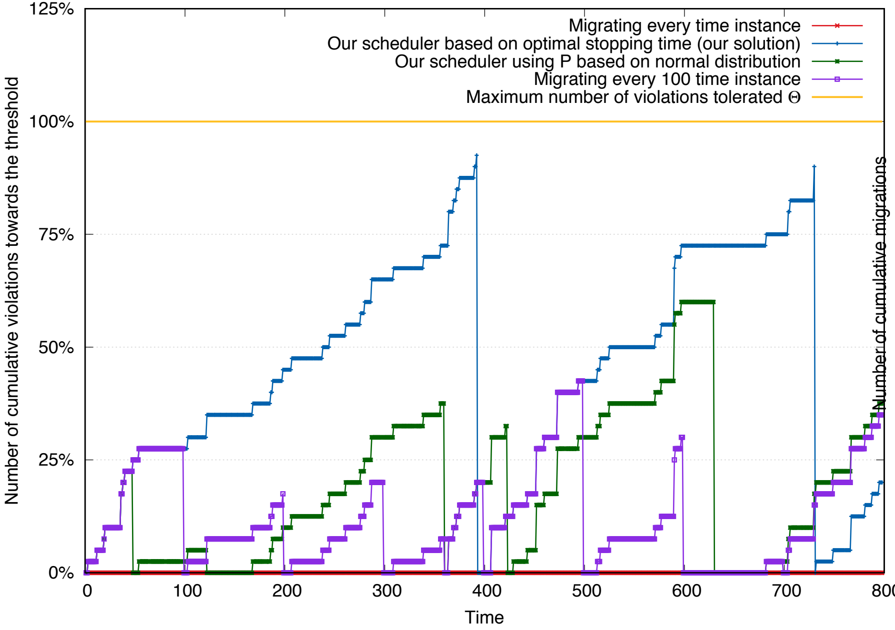
Deviation from optimal

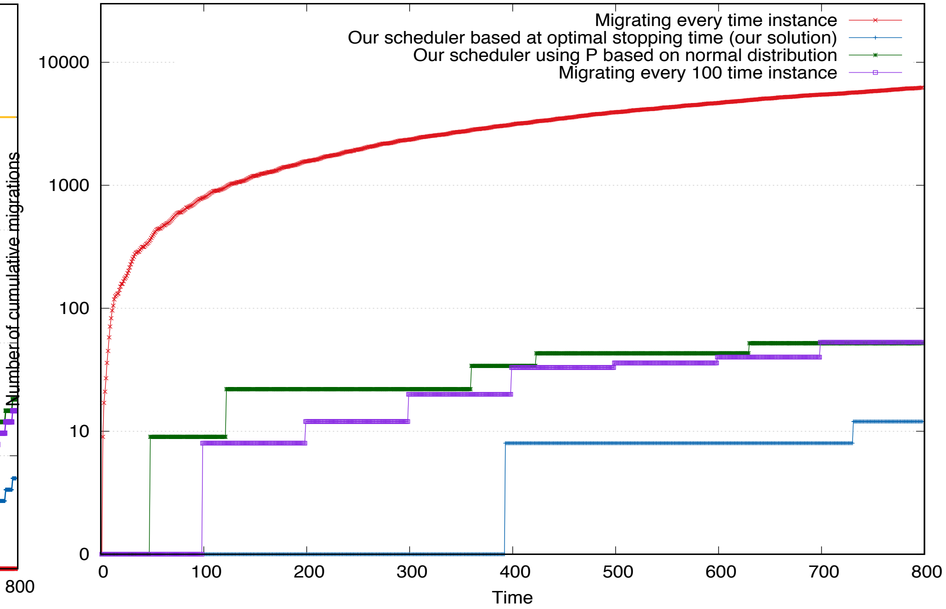




Richard Cziva, Christos Anagnostopoulos, Dimitrios P Pezaros | **University of Glasgow** | Richard.Cziva@glasgow.ac.uk | INFOCOM'18 | Honolulu, Hawaii

Placement scheduling





Our solution does not reach the latency violation threshold, and gives low number of migrations.

Summary

- Edge vNFs can support low-latency – if allocated to the right devices
- Our work proposed a dynamic, latency-optimal vNF allocation algorithm
  - Optimal allocation used Integer Linear Programming
  - Dynamic extension was built on top of Optimal Stopping Theory
- Evaluation was conducted using real-world latency characteristics and a nation-wide network topology
- Our solution reduces the number of migrations by 94.8% and 76.9% compared to a scheduler that runs every time instance and one that would periodically trigger vNF migrations to a new optimal placement, respectively.

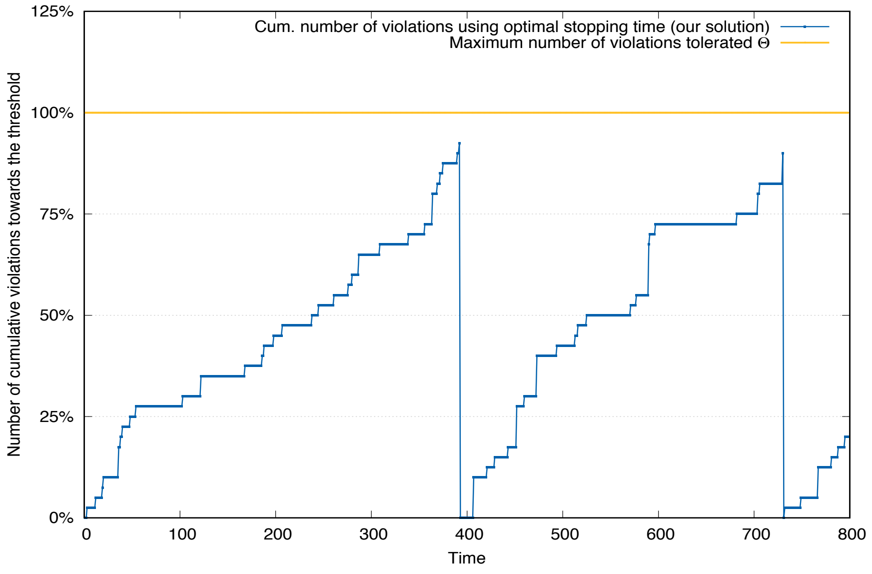
NETLAB
NETWORKED SYSTEMS RESEARCH LABORATORY
University of Glasgow | School of Computing Science

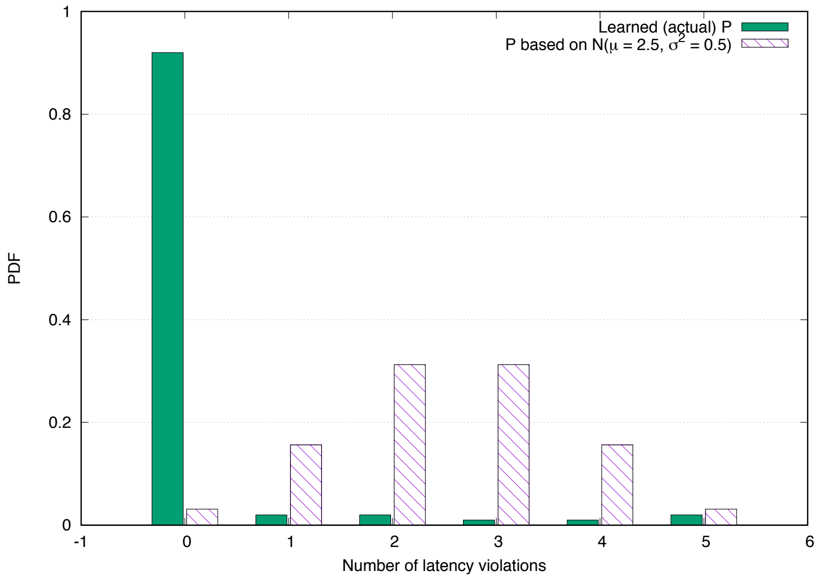
Thank you for your attention!

Download this presentation from http://netlab.dcs.gla.ac.uk

Richard Cziva, Christos Anagnostopoulos, Dimitrios P Pezaros | **University of Glasgow** | Richard.Cziva@glasgow.ac.uk | INFOCOM'18 | Honolulu, Hawaii

Extra: learning phase





Glasgow Network Functions