# Workshop 1: 5G Core Slicing

## Slice Modeling and Dynamic Resource Scaling

**Raouf Boutaba**
David R. Cheriton School of Computer Science
University of Waterloo

UNIVERSITY OF WATERLOO

# Table of Contents

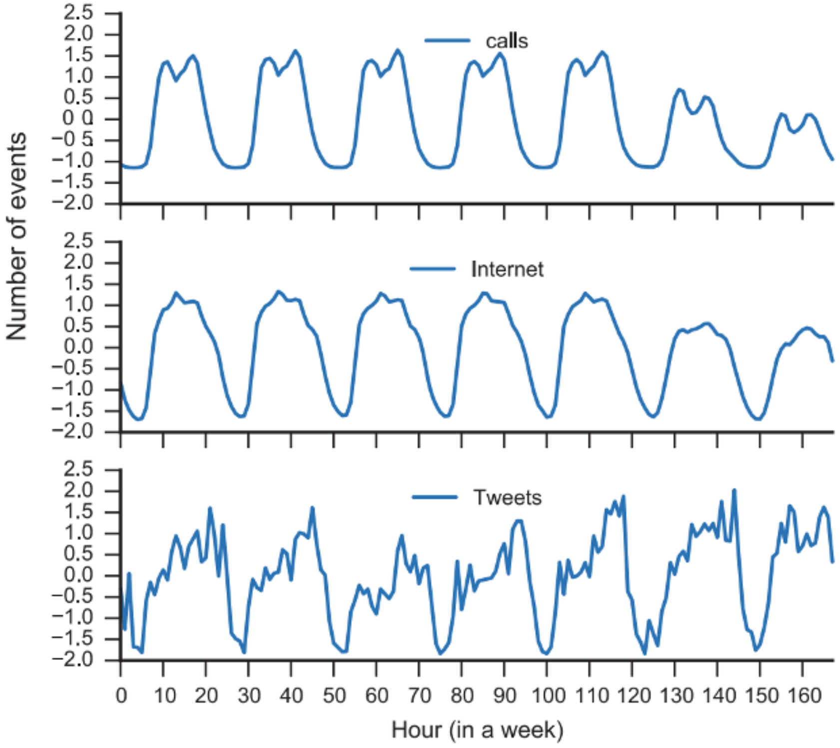

- Introduction
- Challenges
- Solutions
- Session#2 structure

# INTRODUCTION

# Introduction: Dynamic Resource Scaling

- **Slice Traffic**: Number of active slice users
  - Varies throughout the week
- **Resource allocation and QoS**
  - Peak allocation vs. average allocation

- **Dynamic resource scaling**:
  - Dynamically Scaling resources allocated to slices based on current or predicted traffic
  - Objective:
    - Minimize resource allocation
    - Satisfy QoS requirements




Scaled weekly behavior of Calls, Internet, Tweets in MILAN [1]

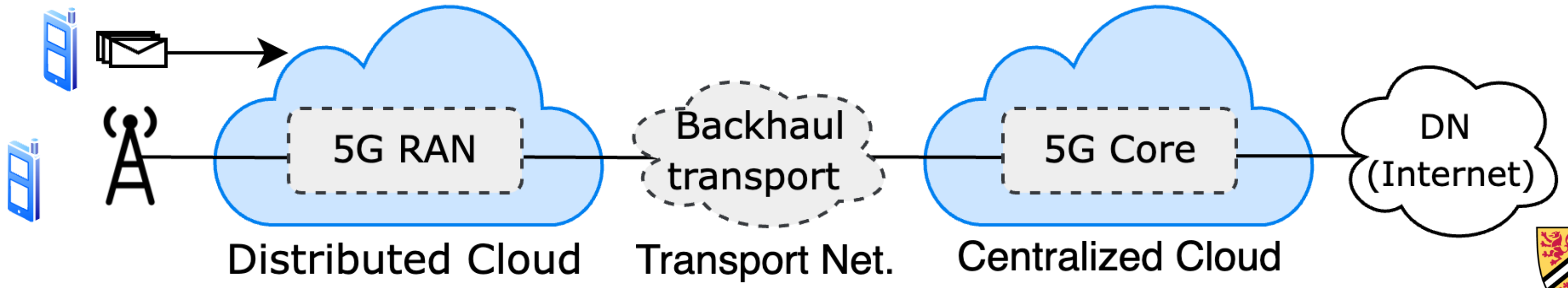

[1] G. Barlacchi et al., "A multi-source dataset of urban life in the city of Milan and the province of Trentino," Scientific Data, 2015.

# CHALLENGES

# Dynamic Resource Scaling: Challenges (1/2)

## Slice Modeling:

- Modeling relationship between resource allocation and QoS
- Slices span multiple network segments (RAN, transport, core) networks
- Heterogeneous QoS and resource requirements
  - Latency, packet loss, reliability, jitter
  - CPU, PRBs, bandwidth, memory
- Traditional modeling approaches are slow and lack real-time application feasibility




UNIVERSITY OF WATERLOO

# Dynamic Resource Scaling: Challenges (2/2)

## Constrained Optimization:

- Finding minimum resource allocation that satisfies QoS requirements
- Needs to integrate neural network-based slice model
- Must be fast and efficient

$$\min_{\mathbf{r}} \quad \frac{1}{|T|} \sum_{t \in T} \sum_{s \in \mathbf{S}} \eta^\top \mathbf{r}_t^s$$ → Minimize resource allocation to slices, subject to

$$\text{s.t.} \quad \mathbb{E}\left(\beta^s_{max(T)}\right) \leq \beta_{s,thresh}, \quad \forall s \in S$$ → QoS degradation threshold constraint

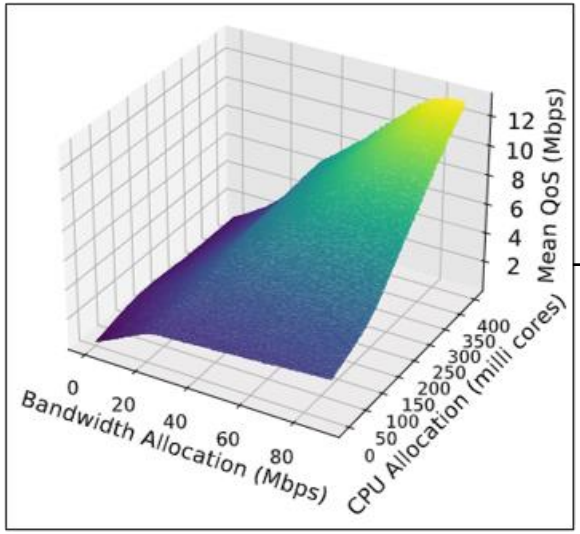
$$\sum_{s \in S} \mathbf{r}_t^s \leq \mathbf{R}, \quad \forall t \in T,$$ → Resource capacity constraint

UNIVERSITY OF WATERLOO

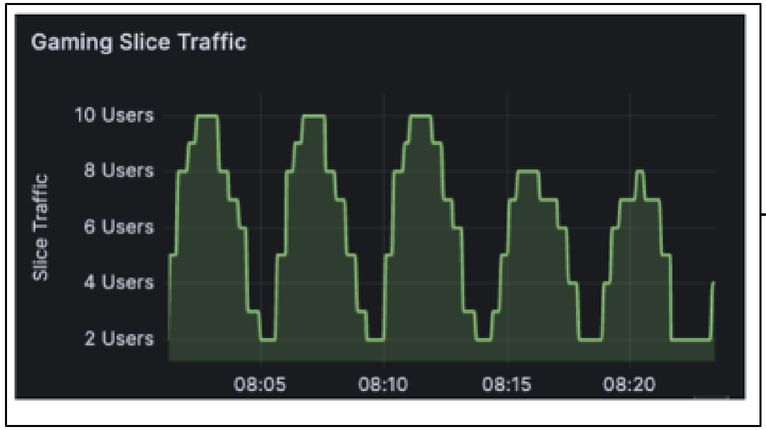
# SOLUTIONS

# Solution Overview

Network Modelling (**vNetRunner**)




Slice Traffic




Resource Scaling Algorithm (**MicroOpt**)

**Algorithm 1** Resource Allocation Algorithm

**Input:** Traffic $x^s_{\tau_i}$, Network Model $f^s_{QoS}(x^s_{\tau_i}, r^s_{\tau_i})$, QoS threshold $q^s_{thresh}$, QoS degradation threshold $\beta^s_{thresh}$, $\tau_{1,max}, \tau_{2,max}, \alpha_1, \alpha_2, \alpha_3, \epsilon_1, \epsilon_2$

**Output:** Optimal resource allocation vector $r^s_{\tau_i}$

Initialize $\boldsymbol{\lambda}, \boldsymbol{\mu}$, LB $= 0$, UB $= \infty$, $\tau_1 = 0$, $\tau_2 = 0$

**while** $\frac{\text{UB}-\text{LB}}{\text{UB}} > \epsilon_1$ **or** $\tau_1 < \tau_{1,max}$ **do**

  $\mathbf{r} \leftarrow$ `Gridsearch`$(x^s_{\tau_i}, f_{QoS}(x^s_{\tau_i}, \mathbf{r}))$

  **while** $|\nabla_r \hat{\mathcal{L}}| > \epsilon_2$ **or** $\tau_2 < \tau_{2,max}$ **do**

    $\mathbf{r} \leftarrow [\mathbf{r} - \alpha_1 \nabla_r \hat{\mathcal{L}}]^+$

    $\tau_2 \leftarrow \tau_2 + 1$

  **end while**

  $\lambda_s \leftarrow [\lambda_s + \alpha_2(\beta^s - \beta^s_{thresh})]^+, \forall s$

  $\mu_k \leftarrow [\mu_k + \alpha_3(\sum_{s \in S} r^{s,k} - R^k)]^+, \forall k$

  LB $= \max(\text{LB}, \mathcal{L}(\mathbf{r}, \boldsymbol{\mu}, \boldsymbol{\lambda}))$

  UB $= \min(\text{UB}, \sum_{s \in S} \eta^\top \mathbf{r}^s)$

  $\tau_1 \leftarrow \tau_1 + 1$

**end while**

**return** $\mathbf{r}$

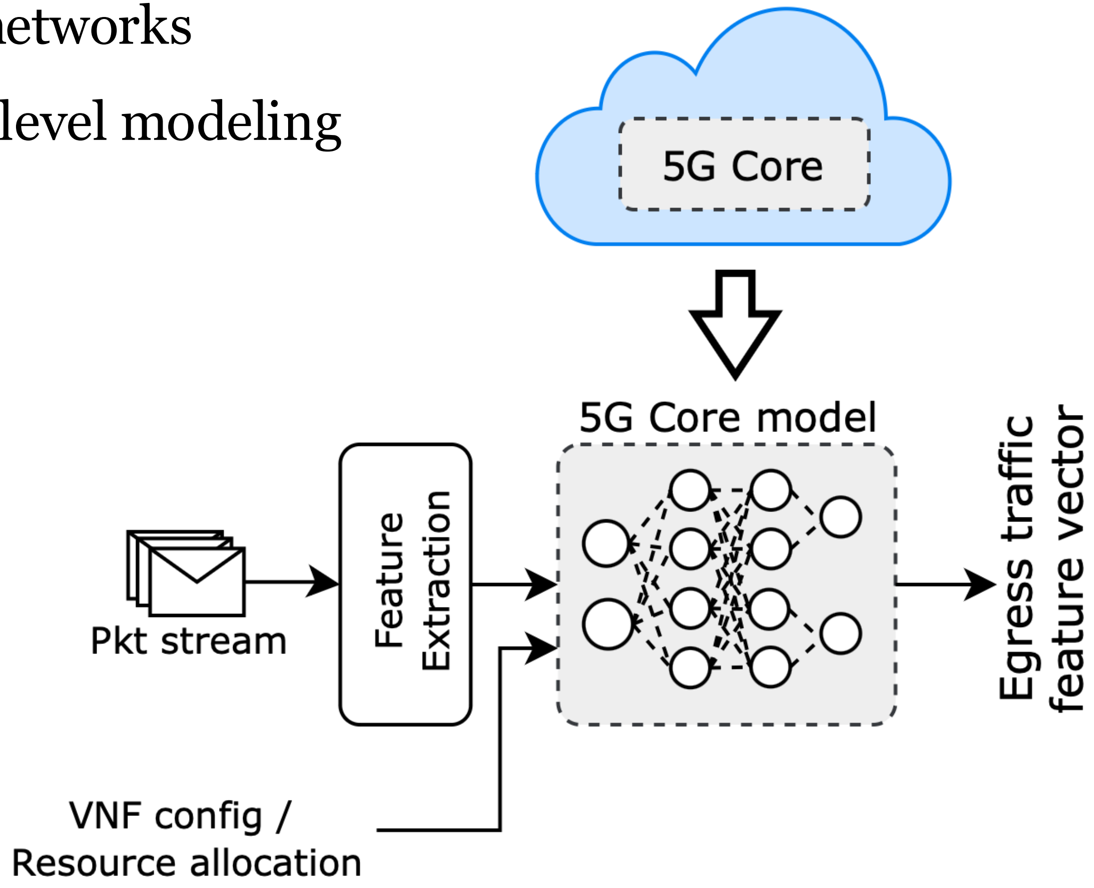
Resource Allocation



UNIVERSITY OF WATERLOO

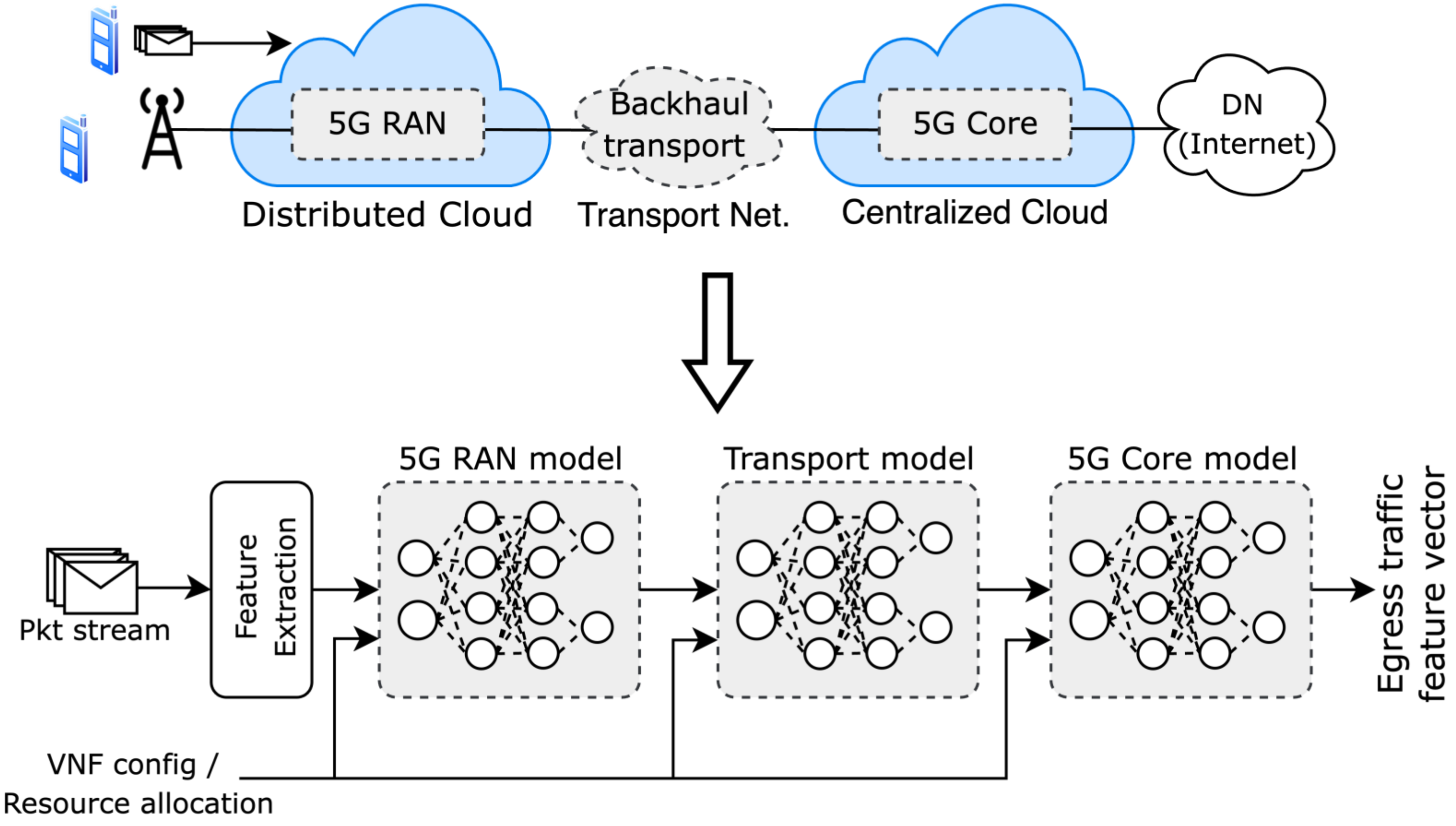
# Network Modeling: vNetRunner (1/2)

- Slice Modeling using neural networks
- Two steps slice modeling: VNF modeling, Slice modeling using VNF models
- Step 1: Individual VNF slice modeling using neural networks
  - Reduces dataset requirement compared to slice-level modeling
  - Allows for composable VNF models
  - Fast inference (milliseconds)



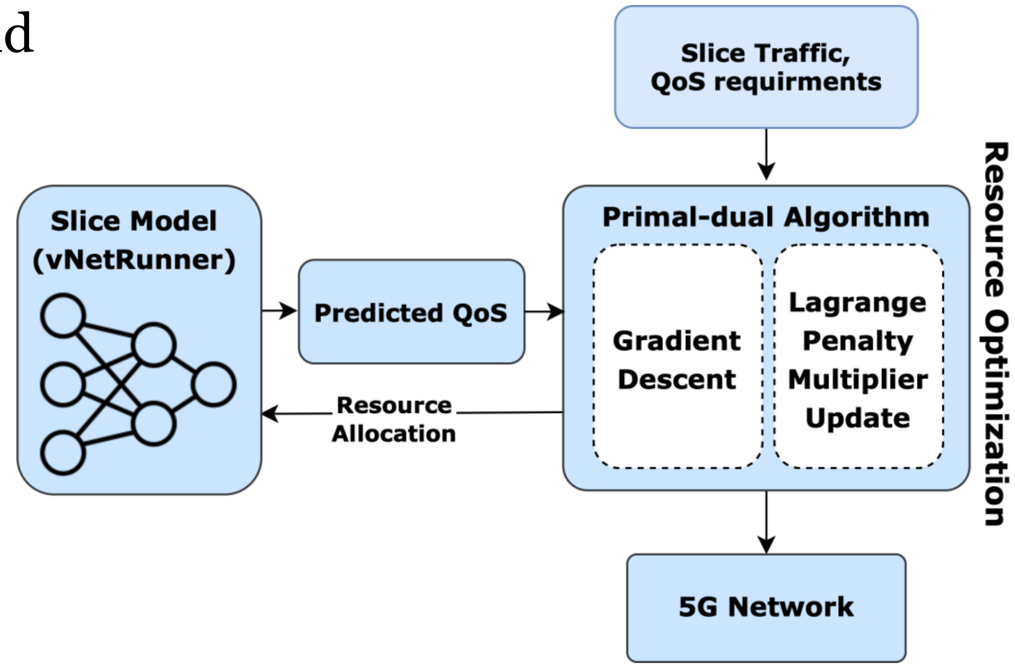
# Network Modeling: vNetRunner (2/2)

- Step 2: Composing slice model from VNF models



# Constrained Optimization: MicroOpt

- Primal-dual optimization and gradient descent for fast and efficient resource optimization
- User vNetRunner for QoS estimation
- Gradient Descent:
  - Adjusts resource allocation to minimize the overall resource usage while paying QoS violation penalty
- Lagrange Multiplier Update:
  - Ensures that QoS constraints are met by adjusting QoS violation penalties in each iteration.



# WORKSHOP SESSION#2 STRUCTURE

# Workshop Session#2 Structure

- **Part1: Data exploration and visualization**
  - Explore and visualize the resource allocation dataset gathered from in-lab 5G testbed.

- **Part2: vNetRunner**
  - Train and visualize VNF models using machine learning.
  - Use trained VNF models to compose end-to-end slice model.

- **Part3: MicroOpt**
  - Implement dynamic resource scaling with the MicroOpt framework.