Fair Computing-Aware Traffic Steering for Smart-City Services across the Edge–Cloud Continuum

Francesco Taverna, Laura Lemmi, Carlo Puliafito, Enzo Mingozzi, Antonio Virdis
Department of Information Engineering, University of Pisa, Pisa, Italy
f.taverna1@studenti.unipi.it, laura.lemmi@phd.unipi.it, carlo.puliafito@unipi.it, enzo.mingozzi@unipi.it, antonio.virdis@unipi.it

I. Introduction

Applications such as smart mobility, urban sensing, e-health, and digital-twin-based city services require low latency, context awareness, and elastic access to computational resources. These requirements can be addressed by replicating the same logical service across multiple sites of the edge–cloud continuum, including far-edge nodes, regional edge facilities, and cloud data centers.

In this setting, selecting where each request should be processed is a non-trivial task. Traditional routing mechanisms mainly rely on network-level information and may therefore steer traffic toward the topologically closest site even when that site is overloaded or has insufficient computational capacity. Conversely, decisions based only on computing availability may select remote sites and degrade latency-sensitive services. Computing-Aware Traffic Steering (CATS) addresses this limitation by jointly considering network and computing information when steering traffic toward replicated service instances [1].

This problem is especially relevant for smart-city infrastructures. Urban services are often characterized by spatially and temporally variable demand: traffic-monitoring services may experience peaks during commuting hours, public-safety analytics may require sudden additional processing capacity during events or emergencies, and digital-twin applications may generate bursts of data depending on sensing activity. A static or purely local traffic-steering policy may be unable to react efficiently to these dynamics.

This extended abstract revisits the centralized CATS framework introduced in [2] from a smart-city perspective, relating its fairness-based architecture to spatially and temporally variable urban workloads. The framework coordinates ingress routers through a logically centralized orchestrator that collects demand and capacity information, computes fair traffic-splitting decisions, and enforces them in the network through weighted forwarding rules. The goal is not only to select a reachable service instance, but to jointly account for network proximity and computing availability when distributing requests, so as to avoid overload and achieve balanced utilization of the shared edge–cloud infrastructure.

II. Computing-Aware Steering Architecture

The considered system consists of a set of Ingress Forwarders (IFs), which receive client traffic, and a set of Service Sites (SSs), which host replicated instances of one or more services. Each SS is reached through an Egress Forwarder (EF), namely the network node acting as the output point toward the SS. Without losing any generality, for each IF we group the SSs into three levels of proximity, respectively (i) the far edge, which includes the closest servers to the IF, (ii) the near edge, which comprises regional DCs in between the far edge and the cloud, and, finally, (iii) the centralized cloud DC. From the perspective of a smart-city application, a service may correspond, for example, to an urban analytics function, a mobility-control backend, a public-safety processing component, or a digital-twin processing module. Clients access the service through an anycast IPv6 address, while the network is responsible for steering each flow toward one of the available service instances.

The architecture builds on a Segment Routing over IPv6 (SRv6) - based MultiProtocol Border Gateway Protocol (MP-BGP) network. SRv6 provides the forwarding overlay used to steer packets across the network, while MP-BGP is used to advertise routes. A different Virtual Routing and Forwarding (VRF) table is configured per proximity level on each IF. On EF side, one VRF table is configured per SS on each EF. This design ensures clients satisfaction while preserving service transparency. The IFs perform traffic steering by using Weighted Equal-Cost Multi-Path (WECMP), which allows traffic to be split among multiple service sites while preserving flow affinity.

The key architectural element is a logically centralized orchestrator, which maintains a global view of the system state. To this end, it collects information from three sources. First, IF agents monitor the local traffic demand associated with each service and proximity level. Second, EF agents report the computational capacity available at the SSs. Third, the orchestrator observes control-plane events, such as service announcements or failures, through the BGP Monitoring Protocol (BMP).

Fig. 1 highlights the main functional blocks of the proposed architecture. The Route Reflector provides the orchestrator with visibility on service reachability changes, while IF and

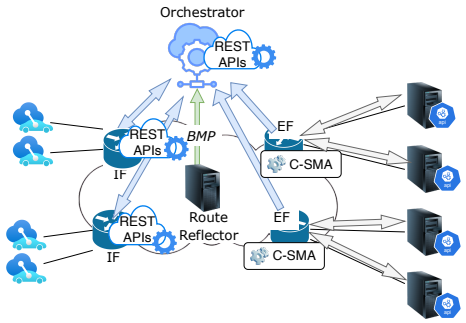

Fig. 1. High-level logical view of the centralized CATS architecture.

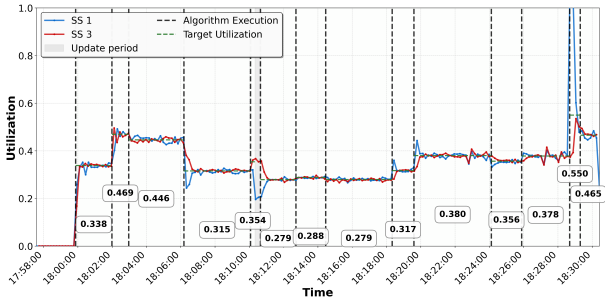

Fig. 2. Utilization over time.

EF agents supply runtime telemetry on demand and available computational capacity through REST APIs exposed by the orchestrator. Based on this information, the orchestrator executes the centralized decision logic and translates the resulting allocations into weighted forwarding rules to be installed at the ingress routers. Centralized decision-making enables globally coordinated and fair use of the available service sites, while keeping packet forwarding lightweight and compatible with standard SRv6 and WECMP mechanisms.

Upon changes in demand, capacity, or service reachability, the orchestrator recomputes the allocations and installs the corresponding WECMP weights at the IFs through REST-based interfaces.

The resulting architecture combines two complementary properties. On the one hand, forwarding remains lightweight and compatible with standard routing technologies: packets are handled by SRv6 and WECMP mechanisms, without requiring per-packet decisions by the orchestrator. On the other hand, traffic splitting is computed with global visibility, allowing the system to account for the aggregate demand generated by multiple ingress routers and for the heterogeneous capacity available at different SS.

III. Fair Allocation and Performance Insights

The centralized orchestrator formulates traffic steering as a fair allocation problem. Let $D_{i,l}$ denote the demand observed at IF i for proximity level l, and let C_s denote the computational capacity of SS s. The allocation variable $R_{(i,l),s}$ represents the amount of traffic sent from IF/proximity pair (i,l) to SS s. Feasible allocations must satisfy three basic conditions: allocations must be non-negative, the demand generated by each IF/proximity pair must be fully assigned to reachable SSs, and the total traffic assigned to each SS must not exceed its available capacity.

The utilization of SS s is defined as

$$U_s = \frac{\sum_{(i,l)} R_{(i,l),s}}{C_s}. \tag{1}$$

The objective is to compute a min–max fair allocation, i.e., an allocation that lexicographically minimizes the maximum utilization among service sites. In practice, the orchestrator solves the allocation problem through an iterative min–max fairness procedure inspired by [3]. Since at least one bottleneck SS is fixed at each iteration, the procedure terminates after at most $|S|$ iterations and returns the leximin-optimal allocation. The computed traffic splits are then converted into WECMP weights, so that the forwarding plane implements the desired allocation.

The evaluation was performed on a containerized testbed based on FRRouting and Containerlab. The experiments considered dynamic traffic demand, dynamic computational capacity, joint random variations of both demand and capacity, and service failure/recovery events. Among these scenarios, the case with joint random variations is especially representative of realistic smart-city conditions, because it combines fluctuating workloads and changing resource availability. The results show that centralized orchestration can rebalance traffic and keep the utilization of SSs close to the fair target.

Fig. 2 reports a representative long-run experiment in which both traffic demand and computational capacity vary dynamically over time. The plot shows that the orchestrator repeatedly reacts to these variations, recomputing the steering weights so that the measured utilizations of the SSs remain close to the corresponding fair target values.

A complementary reconfiguration-time analysis showed that optimization required 0.15–0.5 s, while rule actuation accounted for 60–80% of the reconfiguration time.

This result conveys the main performance insight of the proposed framework: centralized CATS orchestration tracks the evolution of the operating conditions, avoids persistent overload of individual service sites, and keeps the infrastructure close to a balanced operating point. These properties are relevant for smart-city services because they support elastic and resilient use of edge–cloud resources under bursty, heterogeneous, and variable urban workloads.

References

[1] C. Li, Z. Du, M. Boucadair, L. M. Contreras, and J. Drake, “A Framework for Computing-Aware Traffic Steering (CATS),” IETF Internet-Draft, 2025.

[2] F. Taverna, L. Lemmi, C. Puliafito, E. Mingozzi, and A. Virdis, “A Centralized Framework for Fair and Efficient Computing-Aware Traffic Steering in Edge-Cloud Environments,” in *Proc. IEEE WoWMoM*, Bologna, Italy, 2026, to appear.

[3] B. Radunović and J.-Y. Le Boudec, “A Unified Framework for Max-Min and Min-Max Fairness with Applications,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1073–1083, 2007.