Adaptive Graph-based SDN Load Balancing with Virtual and Volunteer Nodes

Zahra Javadi*, Luca D'Agati*, Francesco Longo*‡, Giovanni Merlino*‡
* Department of Engineering, University of Messina, Messina, Italy
‡ CINI: National Interuniversity Consortium for Informatics, Rome, Italy
{zjavadi, ldagati, flongo, gmerlino}@unime.it

***Abstract*—The Fourth Industrial Revolution promotes Software-Defined Cities, where interconnected systems operate seamlessly. On the other hand, softwarized networks, enabled by Software-Defined Networking and Network Function Virtualization, have revolutionized network management by providing programmability and flexibility. However, dynamic load balancing in such environments remains a challenge due to the complexity of real-time traffic patterns and network conditions. This paper proposes a novel four-layer architecture leveraging graph theory and algorithms to achieve adaptive load balancing in softwarized feedback control systems, including volunteer and virtual nodes alongside physical devices. By modeling the network as a dynamic graph, we employ graph algorithms such as shortest path, max-flow/min-cut, and graph partitioning to optimize traffic distribution. A closed-loop feedback control system continuously monitors network metrics and adjusts load balancing decisions dynamically.**

***Index Terms*—Software-Defined City (SDC), Software-Defined Networking (SDN), Network Function Virtualization (NFV), softwarized feedback control systems, adaptive load balancing, graph-based optimization.**

I. Introduction

A smart city emerges when devices collaborate autonomously with minimal human intervention through software-defined infrastructures. In smart cities, people can connect with each other, interact with smart devices, and enable devices to communicate seamlessly among themselves. They all work together in real time using digital technology to make the city smarter [1]. The advent of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) has transformed traditional network architectures by decoupling the control plane from the data plane and enabling softwarization of network functions [2]. This paradigm shift offers flexibility, scalability, and programmability, making it ideal for new technologies such as cloud computing, IoT, and 5G networks [3]. However, one of the critical challenges in softwarized networks is dynamic load balancing, which ensures efficient resource utilization and minimizes latency in the face of fluctuating traffic patterns and network conditions [4]. Traditional load balancing, based on static or heuristic methods, fails in dynamic environments [5]. They are unable to adapt in real time, causing inefficiencies and congestion. [6]. To address these limitations, this paper proposes a graph theory-based approach for adaptive load balancing in softwarized feedback control systems. By modeling the network as a dynamic graph and employing graph algorithms, we aim to achieve optimal traffic distribution and real-time adaptability. The proposed architecture is based on a two-layer SDN controller framework, where the Global Controller oversees network-wide optimization with dynamic topology, and Local Controllers handle real-time adjustments for switches and IoT gateways. This hierarchical design ensures scalability and responsiveness, making it suitable for large-scale networks. The integration of a closed-loop feedback control system further enhances adaptability by continuously monitoring network metrics and adjusting load balancing decisions dynamically. The architecture also includes: (i) a Volunteer Layer of user-contributed devices, and (ii) a Virtual IoT Layer with digital twins of sensors and actuators.

II. Proposed Methodology

In dynamic graph-based load balancing, using SDC as a case study, SDN acts as the control plane managing graph evolution.

A. From Traditional Methods to Four-Layer SDN Architecture

Traditional networks often rely on static routing (e.g., OSPF, BGP) and manual reconfiguration, which limits adaptability in dynamic contexts. Unlike static routing (e.g., OSPF, BGP), SDN with dynamic graphs enables adaptive, centralized, and algorithmic traffic management.

To overcome such limitations, we propose a four-layer SDN architecture shown in Fig. 1(a) that enables adaptive load balancing through graph-based control and feedback mechanisms. The architecture includes:

– *Global controller layer*: handles high-level decision making using a dynamic graph and graph path algorithms.

– *Local controller layer*: manages real-time flow adjustments and forwards local metrics upstream.

– *Volunteer layer*: includes user-contributed devices (e.g., PCs, smartphones) that offer auxiliary resources.

– *Virtual IoT layer*: composed of software-defined twins of physical IoT nodes for flexible simulation and actuation.

This layered model addresses key challenges in scalability, responsiveness, and consistency. The global controller avoids bottlenecks through hierarchical designs, while local controllers enable low-latency reactions. Distributed consensus ensures rule consistency, and incremental algorithms improve responsiveness to dynamic changes.

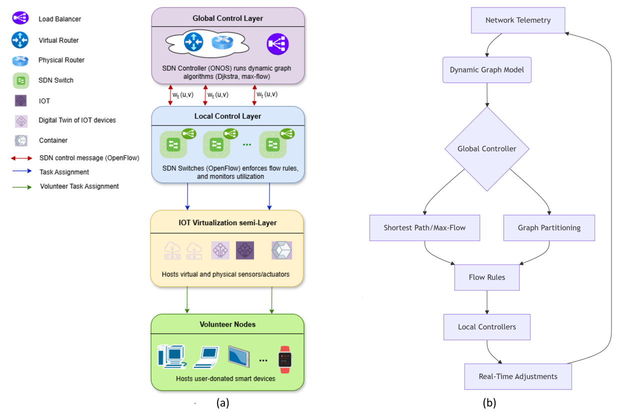
Load Balancer
Virtual Router
Physical Router
SDN Switch
IOT
Digital Twin of IOT devices
Container
SDN control message (OpenFlow)
Task Assignment
Volunteer Task Assignment
Global Control Layer
SDN Controller (ONOS) runs dynamic graph algorithms (Djkstra, max-flow)
w_t (u,v)
w_t (u,v)
w_t (u,v)
Local Control Layer
...
SDN Switches (OpenFlow) enforces flow rules, and monitors utilization
IOT Virtualization semi-Layer
Hosts virtual and physical sensors/actuators
Volunteer Nodes
...
Hosts user-donated smart devices
(a)
Network Telemetry
Dynamic Graph Model
Global Controller
Shortest Path/Max-Flow
Graph Partitioning
Flow Rules
Local Controllers
Real-Time Adjustments
(b)

Fig. 1: Proposed SDN-based system: (a) Four-layer architecture with virtualization and volunteer nodes; (b) Control loop using dynamic graph algorithms.

B. *Graph-Based Network Modeling*

Dynamic graphs, which evolve over time, are ideal for representing real-time systems, with nodes and edges being added, removed, or modified in response to real-world changes. Dynamic graphs capture temporal dependencies, making them ideal for modeling real-time systems like communication and transportation networks. Therefore, we propose to model the network with the following graph-based method:

$$G_t = (V \cup V_{\text{vol}} \cup V_{\text{iot\_virt}},\ E \cup E_{\text{virt}},\ W_t) \tag{1}$$

The network is represented as a dynamic weighted graph. Nodes represent devices, edges represent links weighted by latency, bandwidth, or congestion. Multi-layer graph is used to model different network layers (e.g., physical, virtual).

- $V_{\text{iot\_virt}}$: virtual IoT nodes that are software twins of physical IoT devices like virtual temperature sensors.
- V_{vol}: Volunteer nodes: user-donated PCs, laptops, or smartphones.
- E_{virt}: virtual edges which include two types of links,
  - E_{vol}: links to volunteer's nodes which are volatile and they have high latency.
  - $E_{\text{iot\_virt}}$: links to virtualized IoT nodes which are stable and also have low latency.
- weights: in our proposed model it will be calculated based on the following equation:

$$w_t(u,v) = \begin{cases} \alpha \cdot \text{latency} + \beta \cdot \text{trust\_score} & \text{if } v \in V_{\text{vol}} \\ \alpha \cdot \text{latency} + \delta \cdot \text{virtualization\_overhead} & \text{if } v \in V_{\text{iot\_virt}} \\ \alpha \cdot \text{latency} + \gamma \cdot \text{utilization} & \text{otherwise} \end{cases} \tag{2}$$

Utilization $[v]$ quantifies how busy a network node (e.g., switch, router) or link is at time t. It is a normalized metric ranging from 0 (idle) to 1 (fully saturated). These metrics influence dynamic weight updates for load balancing. For example, if a vertex v has high utilization, the SDN controller avoids routing traffic through v to reduce congestion risks.

C. *Feedback Control System*

Considering a real SDC scenario, a closed-loop control mechanism is necessary to ensure adaptive load balancing. Our proposed system (Fig. 1b) monitors key network metrics (e.g., latency, packet loss, throughput), uses graph algorithms to compute optimal decisions, and dynamically updates routing paths or resource allocation. The feedback loop consists of four phases:

1) **Monitoring phase** –The SDN controller gathers port stats (e.g., byte count, link latency)
2) **Graph update phase** – Edge weights are recalculated, and nodes are added or removed as needed.
3) **Re-optimization phase** – Algorithms (e.g., Dijkstra) identify new optimal paths.
4) **Action phase** – The controller installs updated flow rules across the network.

While this paper focuses on architectural and algorithmic design, future work will involve implementing and evaluating the framework using OMNeT++ and the INET framework to simulate SDN-based dynamic topologies under varying traffic conditions.

D. *Graph Algorithms for Adaptive Load Balancing*

Our architecture employs classical graph algorithms, adapted for real-time SDN. Key requirements include sub-millisecond execution, incremental updates, and scalability to large topologies. The global controller uses max-flow and graph partitioning for coarse-grained optimization, distributing traffic across subgraphs and avoiding overloads. For fine-grained decisions, incremental Dijkstra recalculates only affected paths after topology changes, reducing overhead. Temporal community detection tracks evolving traffic clusters and enables proactive path adjustments. For instance, when a volunteer node fails, Dijkstra updates local paths without global disruption; community detection flags emerging hotspots for early mitigation. This hybrid approach combines local responsiveness with global efficiency. Unlike ML or control-theory techniques, it offers lightweight, explainable behavior and scales to tens of thousands of nodes.

Acknowledgment

This work was partially supported by the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU through Project "SERICS – Security and Rights in CyberSpace" under Grant PE00000014.

References

[1] V. S. Rao, "Software defined smart cities," https://india.theiet.org/media/1251/vsr-mini_book-software_defined_smart_cities.pdf, Aug. 2019, online.

[2] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, and F. Saad, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, 2015.

[3] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for the internet," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3456–3469, Dec. 2016.

[4] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an openflow architecture," in *Proceedings of the 23rd International Teletraffic Congress (ITC)*, 2011, pp. 1–7.

[5] M. Al-Fares, A. Loukissas, and A. Vahdat, "Hedera: Dynamic flow scheduling for clouds," in *ACM SIGCOMM*, 2015.

[6] P. Wang, H. Xu, L. Huang, J. He, and Z. Meng, "Control link load balancing and low delay route deployment for software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2446–2456, Nov. 2017.