Adaptive Policy-Driven Network Intelligence for Edge-to-Cloud Continuum

Ioannis Pastellas
UBITECH Ltd
26 Nikou and Despinas Pattichi
Limassol 3071, Cyprus
ipastellas@ubitech.eu

Sophia Karagiorgou
UBITECH Ltd
26 Nikou and Despinas Pattichi
Limassol 3071, Cyprus
skaragiorgou@ubitech.eu

Mariza Konidi
Dept. of Electronic Engineering
Hellenic Mediterranean University
Crete, Greece
ddk184@edu.hmu.gr

***Abstract*—Edge-to-Cloud (E2C) is a rapidly emerging technology that aims to reduce overall traffic to the cloud by enabling Internet of Things (IoT) data processing as close to the data sources as possible, either on near- or far-edge devices. This paper introduces an adaptive, policy-driven framework for network intelligence designed to optimize E2C applications by intelligently enforcing network rules. By leveraging decentralized decision-making and Artificial Intelligence (AI) driven application profiling, the framework enables dynamic and adaptive network resource allocation, significantly enhancing network capacity, improving the capabilities between user applications and core network resources, and better correlating Quality of Service (QoS). The deployment of novel AI models, which leverage real-world monitoring data from E2C environments, demonstrates the framework's ability to enforce adaptive network policies and make informed decisions, contributing to more intelligent and resilient networks.**

***Index Terms*—network resource optimization, dynamic policies, runtime adaptations, extreme heterogeneity**

I. Introduction

Modern communication networks rank among the most extensive and intricate man-made systems. Their decentralized structure contributes to the complexities of managing and predicting outcomes. Therefore, basic robust design principles such as continuous monitoring and redundancy are insufficient to guarantee the ultra-reliable performance required for future critical applications, including adaptive and dynamic network intelligence.

To realise adaptive, policy-driven network intelligence in Edge-to-Cloud (E2C) contexts, it is necessary to:

- Perform real time monitoring to detect changes and thus promptly respond in an adaptive manner. An automated agent capable of detecting alterations and learning about network behaviour in real time can provide early warnings or enforce policies in unexpected situations. When linked to specific features, this agent can also activate automatic and adaptive rules to mitigate such conditions.
- Detect and accurately classify events. In network management, additional resources can be dispatched to locations where unforeseen events occur. This is especially pertinent in E2C contexts, where intelligent event management mechanisms should be implemented and enforced at the remote points where problems arise.
- Simplify the core network and tackle edge diversity by leveraging common protocols used by E2C applications. This setup enables end users and administrators to interact within a cloud environment based on service-oriented architecture principles, providing also access to a common shared platform for data exchange.

In today's E2C continuum, the traditional methods for optimizing network resources face limitations and insurmountable complexities, especially in highly competitive situations [1]. Many valuable insights derived from network data depend on Artificial Intelligence (AI) technologies such as neural networks and deep learning. These techniques can be applied across different environments to make decisions and determine optimal runtime adaptations. Therefore, integrating AI with network policies is essential to support adaptive mechanisms that improve network services at runtime. A unified state machine is needed to act as an orchestrator, coordinating network resources, interacting with AI model results, and enforcing adaptive policies to dynamically allocate compute power, memory, bandwidth, cache capacity, and channel resources based on application workload and demand.

The merits of this paper are as follows:

- Harvests real time monitoring data collected from diverse and simulated application contexts over a real-world E2C testbed. This data is further used to assess network resource usage and plan for optimal resource allocation, runtime adaptations, and optimisation.
- Applies real time data analytics derived by predictive and prescriptive learning tasks, which are crucial for E2C applications that require immediate responses while considering energy limitations, dynamic bandwidth allocation, and dimensioned compute resources.
- Enforces adaptive and policy-driven mechanisms to support network intelligence, enabling the creation of multiple virtual overlay networks with different characteristics, thereby serving multiple applications efficiently.

The rest of the paper is structured as follows: Section II provides a review of the current literature on innovative methods for adaptive data-driven mechanisms targeting the E2C continuum. Section III offers a detailed overview of the architecture design and deployed frameworks. Section IV

discusses our experimental approach and scenarios applied to evaluate the adaptive, policy-driven network intelligence mechanisms, while Section V presents the experimental results. Finally, Section VI concludes the paper and outlines directions for future research.

II. Literature Review

Developing data-driven applications requires developers and service providers to orchestrate data-to-discovery pipelines across distributed data sources and computing units. Realizing such pipelines poses two major challenges: programming analytics that can react to unforeseen events at runtime and adapting resources and computing paths between the edge and the cloud. Balouek-Thomert et al. [2] propose a system stack for the adaptation of distributed analytics across the computing continuum. They evaluate the system's ability to continuously balance the cost of computation or data movement with the value of operations relative to the application objectives. This work differentiates to the proposed system as it leverages AI model predictions to proactively derive optimal thresholds for monitored metrics and workload classes such as requested bandwidth classes.

The concept of the E2C continuum aims to significantly reduce overall traffic to the cloud by enabling Internet of Things (IoT) data processing as close to the data sources as possible, whether on near- or far-edge devices. In this highly dynamic environment, where IoT devices and edge nodes are constantly changing their state and location, services running on edge nodes must be scheduled, deployed, and managed to ensure high service availability with appropriate Quality of Service (QoS) parameters. Čilić and Žarko [3] propose a general architecture for adaptive data-driven routing in the edge-to-cloud continuum, implementating this architecture using a content-based publish / subscribe approach. They evaluate their implementation against a real-world use case scenario for federated learning in an edge-to-cloud environment hosting digital twins. This work distinguishes itself from prior research in several key aspects. First, we introduce a comprehensive, general architecture specifically designed for adaptive, policy-driven network intelligence across the E2C continuum. Second, we present a novel implementation of this architecture that leverages application behavioural patterns and AI Models to enable accurate resource allocation and proactive network workload management. Finally, we rigorously evaluate our framework using a real-world testbed simulating different application contexts.

Wang et al. [4] present ShutPub, a publisher-side middleware that performs message filtering before forwarding messages to the broker service. ShutPub limits the publisher's message dissemination based on subscriptions and their filters while remaining transparent to the publisher. By shifting content-based filtering from the broker to the publisher, only messages with an intended receiver are transmitted. Their prototype evaluation demonstrates that ShutPub can reduce system strain on both the network and the broker without simply shifting the burden to the publisher, who benefits from sending fewer messages. Unlike ShutPub, which reduces network and broker strain, our approach achieves similar benefits without increasing complexity. Instead, we employ a network intelligence mechanism that learns from application behavioural patterns to adaptively enforce policies when events occur.

Pfandzelter and Bermbach [5] introduce an approach to a fog-native data processing benchmark that integrates workload and infrastructure specifications. While their approach combines these specifications, our work goes beyond simple offloading by introducing adaptive, policy-driven automation through agents that support intelligent and efficient resource utilization in the edge-to-cloud continuum.

Sekigawa et al. [6] analyse discrepancies and clarify telecom network requirements for Kubernetes. They also perform qualitative and quantitative evaluations on several network extensions developed as open-source software. While their work provides valuable insights into telecom network requirements for Kubernetes and offers a comparative analysis of network extensions, our work complements this research by focusing on the dynamic aspects of network management. Specifically, we propose and implement a novel architecture for an adaptive, data-driven policy-enforcement mechanism, enabling real time optimization of data flows based on application demands and network conditions.

III. Architecture

Network Functions Virtualisation (NFV) and Software Defined Networking (SDN) have transformed the framework for deploying E2C services. On one hand, vast amounts of data are produced every second by interconnected software and hardware services. On the other hand, data collection and data quality are often secondary concerns. Feature engineering algorithms are currently applied in modern SDN infrastructures. High-quality data ensure that SDN controllers can make precise adjustments to network configurations, optimise performance, and proactively respond to evolving conditions. Moreover, a well-crafted set of features not only enhances the accuracy of predictive models but also facilitates intelligent decision-making within a dynamic network environment.

SDN and AI are two distinct but increasingly interconnected technologies that play a crucial role in modern network management and optimisation. AI can play a pivotal role in automating network policy management and monitoring by providing intelligent and adaptive solutions to emerging challenges in E2C applications. The applicability of automated decisions via network policies is made by analysing network traffic patterns. The real time monitoring of network metrics allows the derivation of behavioural patterns for diverse applications, fine-tuning their thresholds, optimising the network, identifying anomalies, performing predictive maintenance, and predicting potential security threats. As a result, the interplay between network management mechanisms, AI, and network policies enhances the availability, security, stability, and scalability of the edge-to-cloud ecosystem. This proactive approach enables the development of dynamic network policies that

can autonomously adjust to evolving issues (e.g., load balancing in response to traffic bursts), thereby improving the overall resilience of the network infrastructure. Additionally, AI Models contribute to the creation of more sophisticated control mechanisms that fine-tune their network policies based on network / application / pod behaviour, infrastructure's environmental characteristics, and contextual information.

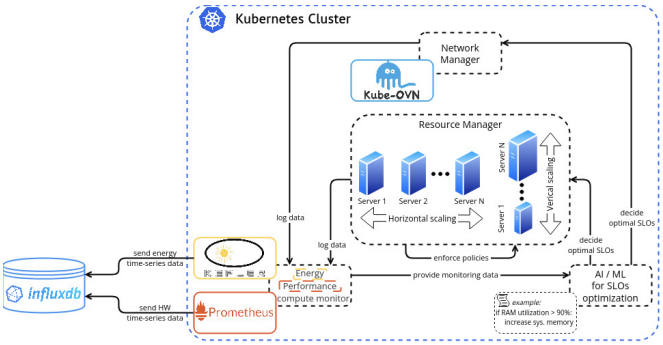


Fig. 1. Network Intelligence Architecture. The architecture is built on top of a Kube-OVN Kubernetes-based cluster of nodes and pods. Network Intelligence is applied by defining and enforcing resource and network policies by utilizing AI/ML predictions. AI/ML models are trained on logged data of different nodes and pods running in the cluster that are monitored by Prometheus, and Kepler and persistently stored in an InfluxDB database.

The overall architecture of this paper is described in the following paragraphs. The architecture follows the principles of Next Generation SDN (NG-SDN) and distributed intelligence to secure onboarding and operation of a consistent overlay network between edge and cloud nodes, as well as the selection of a cluster head to manage the physical deployment of applications.

The architecture consists of a Kubernetes [7] cluster of nodes built on top of Kube-Ovn [8] CNI, allowing for both the resource management capabilities of Kubernetes and full network virtualization. This setup utilizes the most of the networking functionalities, including Quality-of-Service (QoS) management. The entire infrastructure (nodes, pods, and services) is continuously monitored by utilizing different metrics agents such as Prometheus [9], [10] to measure different behaviours of deployed applications and resources. This logging data is further stored in an InfluxDB [11] database for long-term persistence.

This process is responsible for generating and populating network and other metrics to feed AI models. The results derived from the AI model training serve to meet specific QoS requirements (e.g., high availability, high bandwidth, etc.) and Service Level Objectives (SLOs) (e.g., latency below 5ms). The high-level conceptual architecture of the network intelligence component is depicted in Figure 1. The core functionality of this component is to manage virtual overlay networks between edge and cloud. The input consists of Network Rules that dynamically update routing or replace a pod within the cluster at runtime through policy-enforcement, as detailed in the following sections. The component continuously feeds itself with new measurements to optimize the deployments based on the target QoS and SLOs. The architecture is further enhanced by deploying real time monitoring of nodes and pods.

A. *Kube-OVN*

Kube-OVN serves as the core virtual networking framework of the data plane that will be extended in the context of SDN and network intelligence. By default, Kube-OVN employs the CNI chaining mode to augment its existing set of features. As an open-source, cloud-native solution, it integrates OVN-based Network Virtualization with Kubernetes, providing a powerful networking solution that excels in both containerized and VM-based environments and scenarios, while offering robust multi-tenant networking capabilities. The high-level Kube-OVN architecture is depicted in Figure 2. Kube-OVN enhances Kubernetes networking capabilities with multiple advanced features, though we will focus on the ones showcased in this paper. It supports VLAN and underlay networks, enabling better performance and direct connectivity to physical networks alongside traditional overlay networking. Virtual Private Cloud (VPC) support allows multi-tenant networking with isolated address spaces, including Elastic IPs (EIPs), NAT gateways, security groups, and load balancers. Workloads can be assigned either static or dynamically allocated random IP addresses, while multi-cluster networking facilitates L3 connectivity across Kubernetes and OpenStack clusters. With DualStack IP support, Kube-OVN enables IPv4-only, IPv6-only, or dual-stack pod networking. Pod NAT and EIP management ensure external traffic control similar to traditional VMs. Additionally, IPAM for multi-NIC extends subnet and static IP allocation functions to other CNI plug-ins like macvlan and host-device. Dynamic QoS enables real time configuration of traffic rate, priority, packet loss, and latency, optimizing network performance within Kubernetes environments. Furthermore, Kube-OVN also provides additional network metrics that are logged and monitored using Prometheus.

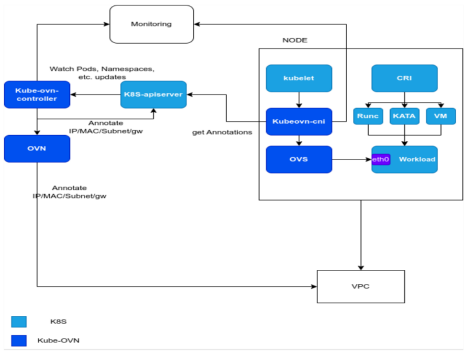


Fig. 2. Kube-ovn High-level Architecture. The main components of Kube-OVN (in dark blue) are illustrated and how they communicate (as denoted by the arrows) with K8S components (light blue) and other components.

The core components of Kube-OVN that have been deployed and configured for inter-pod communication and mon-

itoring are as follows:

1) Ovn-central:

- **ovn-nb**: Manages the storage of virtual network configurations and offers an API for virtual network administration. The primary interaction of kube-ovn-controller involves configuring the virtual network through ovn-nb.
- **ovn-sb**: Stores the logical flow table created from the logical network in ovn-nb, along with the current physical network state of each node.
- **ovn-northd**: Converts the virtual network information from ovn-nb into a logical flow table in ovn-sb.

2) Ovn-ovs:

- **ovs-ovn**: Functions as a DaemonSet deployed on every node, featuring Open vSwitch, ovsdb, and ovn-controller operating within the pod. These elements serve as agents for ovn-central, facilitating the translation of logical flow tables into tangible network configurations.

3) Kube-ovn-cni: This component, operating as a DaemonSet on each node, serves as a CNI interface implementation and manages the local Open vSwitch (OVS) for network configuration. The DaemonSet deploys the kube-ovn binary to each machine, facilitating interaction between kubelet and kube-ovn-cni. This binary, located in the default `/opt/cni/bin` directory, communicates CNI requests to kube-ovn-cni.

kube-ovn-cni is responsible for configuring the specific network to handle various traffic operations. Its key tasks include:

- Configuring `ovn-controller` and `vswitchd`.
- Managing CNI Add/Del requests:
  - Creating or deleting veth pairs and binding or unbinding them to OVS ports.
  - Configuring OVS ports.
  - Updating host iptables, ipset, and route rules.
  - Dynamically updating network QoS.

The primary resources monitored include Pods, Services, Endpoints, Nodes, NetworkPolicies, VPCs, Subnets, VLANs, and ProviderNetworks.

4) Kube-ovn-controller: This component serves as the control plane for the entire Kube-OVN system by translating all Kubernetes resources into OVN resources. The kube-ovn-controller monitors events related to network functionality across various resources and updates the logical network within OVN based on changes in these resources. The primary resources monitored include:

B. Real-time Monitoring

The monitoring lifecycle is activated through a set of daemon services, as depicted in Figure 1. We highlight that all the metrics exposed by the node exporter of Prometheus have been successfully integrated into InfluxDB, a time-series database that ensures long-term data persistence. Key monitoring components include:

1) **Node Exporter**: A Prometheus agent that exposes a wide variety of hardware- and kernel-related metrics, collecting essential Linux host metrics [12].
2) **Kube-state Metrics**: Kube-state-Metrics (KSM) [13] is a straightforward lightweight service that monitors the Kubernetes API server and produces metrics related to the status of objects (refer to examples in the Metrics section).
3) **Kepler**: Responsible for exposing metrics related to energy consumption [2]. Inspired from the work of Cañete et. al. [14], this enables energy-aware E2C deployments.
4) **Resource Optimization**: Ensures efficient network resource management, preventing over-provisioning while maintaining sufficient capacity for peak demand periods.

IV. Experimental Scenarios

In modern communication networks, the proactive allocation of network resources is essential to meet the ever-increasing demand for high-bandwidth, low-latency services. In addition to that, proactive network resource allocation helps maintain network and computing usage on track. This approach involves anticipating future network conditions and user requirements to allocate resources efficiently and effectively ahead of time, in contrast to reactive methods, which respond to real time demands and often result in suboptimal performance due to latency and resource contention. The proactive allocation of network resources is based on several key principles:

1) **Predictive Analytics**: Harnessing the power of historical data in tandem with cutting-edge AI models to accurately predict future network traffic dynamics and intricate application behaviour.
2) **Dynamic Runtime Adaptation**: Continuously fine-tuning resource distribution through real time monitoring and forward-looking insights to optimize network performance.
3) **Quality of Service Management**: Guaranteeing that the network not only meets but succeeds in fulfilling stringent QoS requirements for a diverse array of applications and services, thereby ensuring optimal performance and improved user experiences.
4) **Resource Optimization**: Orchestrating network resources to prevent the pitfalls of over-provisioning, while simultaneously ensuring ample capacity to accommodate peak demand periods effortlessly and seamlessly.

A. Scenario

The scenario is described as follows. The figure represents a network architecture with three edge devices labelled ”edge device 1,” ”edge device 2,” and ”edge device 3,” all depicted as oval shapes on the left side. Each device transmits data to a central Gateway API, also represented as an oval shape on the right side, via arrows indicating the traffic flow. The label ”dynamic traffic” highlights that the Gateway API is receiving a substantial and bursty amount of data from the edge devices, emphasizing the significant load that must be handled. The traffic produced not only increases the network congestion but also the CPU usage on the pod, resulting in the need of optimized network and resource management, as

well as a self-healing mechanism. Using monitoring data on the Gateway API pod (e.g., CPU / RAM usage, and metrics for the network traffic), a set of policies can be applied as part of network intelligence and self-healing.

B. Data Collection

The data were collected using Prometheus for logging metrics and queried from the InluxDB database, where they were stored for a long period. More specifically, as shown in Figure 3, we setup four (4) pods. Three (3) pods to simulate edge devices with diverse network requirements, serving different application types, and one (1) server pod periodically receiving traffic and acting as a Gateway API. All relevant metrics (CPU usage, memory consumption, bandwidth, etc.) were monitored and stored over multiple days. The collected data were then queried from InfluxDB for further analysis.

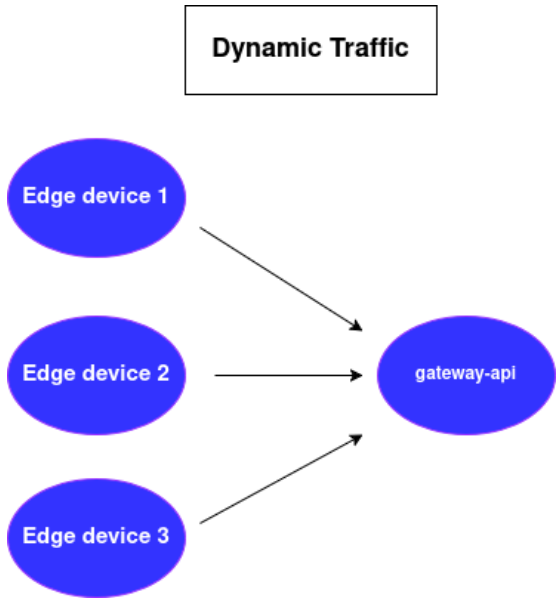


Fig. 3. The Illustration of the scenario. Multiple edge devices send dynamic traffic to a gateway-api. Various metrics of Gateway API (bandwidth, CPU/RAM usage, etc.) are measured.

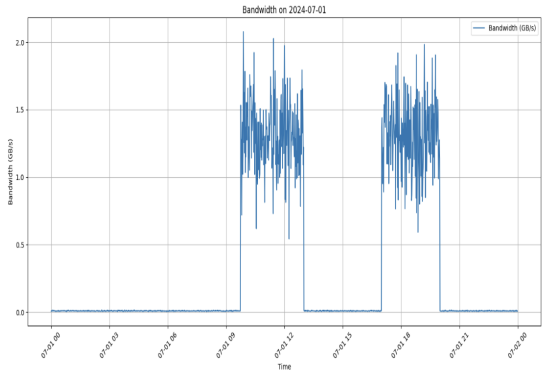


Fig. 4. A data sample of the logged data for one day of the bandwidth metric (GB/s) of the Gateway API. There are some times in the day when bandwidth demand is high while the rest of the demand is relatively low.

C. Policy Definition

In Kubernetes (K8S) and networking environments, managing Quality of Service for network resources is critical to ensure fair allocation and optimal performance across pods. Kube-OVN, a comprehensive network plug-in for Kubernetes, enables and facilitates the enforcement of policies that set maximum bandwidth limits for pods. This section outlines the methodology for dynamically adapting QoS using Kube-OVN, highlighting the steps involved in defining, applying, and adjusting bandwidth policies.

The key policies include:

1) **Vertical Scaling**: Expanding the vertical capacities of a pod, by augmenting its computational power through increased CPU cores and/or RAM, is pivotal when these pods suddenly demand resources. For instance, an upgrade from a modest 0.5 CPU core to a robust 2 CPU cores can substantially elevate pods' performance.
2) **Quality of Service Management**: Ensuring that the network not only meets but consistently exceeds predefined QoS requirements for a diverse array of applications and services, thereby guaranteeing improved performance and reliability.
3) **AI-driven Policy Enforcement**: Using historical data insights alongside cutting-edge AI models to forecast future network traffic patterns and intricately anticipate pod behaviour. This advanced approach empowers us to enforce the aforementioned policies with foresight and precision, seamlessly translating into dynamic runtime adaptations that elevate network efficiency and availability.

D. Policy Engine

To start assigning policies (e.g. QoS) we needed to setup a Policy engine that will take care of applying this policies in real-time. For e.g. to enforce QoS policy while using K8s-OVN you must define network policies to specific the maximum bandwidth for each pod. Again, with Kubernetes Kube-OVN annotations to specify bandwidth limits for a given pod the policies are created.

Our Policy engine is built in Python using the k8s client, to communicate with Kubernetes and perform the appropriate policies when needed. An example of the QoS policy being applied by the policy engine can been seen below in figure 5.

For the enforcement of the AI-driven policy, the procedure is similar as that described above . The only difference is that the predicted value can be inferred some time before, and thus the policy can use the result to proactively enforce the policy.

Below, is an illustrative code, that gets the prediction of the AI model, and applies the policy in the same manner as above(annotate_qos() function).

E. AI Modelling

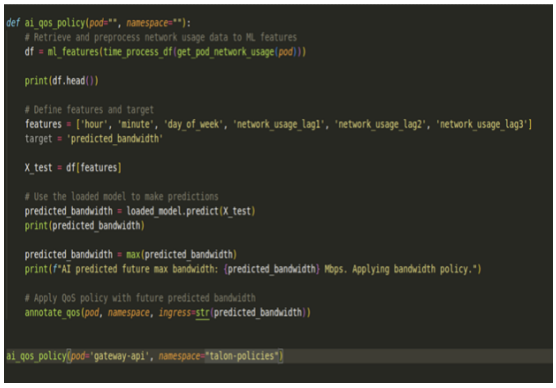
Traditional methods of resource management and policy enforcement in K8S and network environments often rely on reactive measures, which can lead to suboptimal performance

```
def annotate_pod(namespace, pod, annotation):
    v1.patch_namespaced_pod(pod, namespace, body={
  "metadata":{"annotations":annotation}
})


def annotate_qos(pod,namespace, ingress='3', engress=None):
    if ingress is not None:
        annotate_pod(pod, namespace, {"ovn.kubernetes.io/ingress_rate": ingress} )
    if engress is not None:
        annotate_pod(pod, namespace, {"ovn.kubernetes.io/engress_rate": engress} )


# Apply QoS policy on pod
annotate_qos("demos", "gateway-api",ingress='14000')
```

Fig. 5. Python code of how the policy engine applies the QoS policy programmatically



```
def ai_qos_policy(pod="", namespace=""):
    # Retrieve and preprocess network usage data to ML features
    df = ml_features(time_process_df(get_pod_network_usage(pod)))

    print(df.head())

    # Define features and target
    features = ['hour', 'minute', 'day_of_week', 'network_usage_lag1', 'network_usage_lag2', 'network_usage_lag3']
    target = 'predicted_bandwidth'

    X_test = df[features]

    # Use the loaded model to make predictions
    predicted_bandwidth = loaded_model.predict(X_test)
    print(predicted_bandwidth)

    predicted_bandwidth = max(predicted_bandwidth)
    print(f"AI predicted future max bandwidth: {predicted_bandwidth} Mbps. Applying bandwidth policy.")

    # Apply QoS policy with future predicted bandwidth
    annotate_qos(pod, namespace, ingress=str(predicted_bandwidth))


ai_qos_policy(pod='gateway-api', namespace="talon-policies")
```

Fig. 6. Python code of how the policy engine applies the AI-driven QoS policy programmatically

and resource allocation, especially for high-demand applications. Instead, leveraging AI models trained on historical data enables a proactive approach by predicting future bandwidth demands and enforcing network policies dynamically using Kube-OVN.

Thus, building on the previous section, this section describes the enhanced QoS policy and vertical scaling policy. In this way, we mean to proactively enforce the QoS / vertical scaling policy ahead of its needed enforcement with the assistance of an AI model that is fitted to the historical and periodical data of a pod.

The AI-fuelled approach utilizes machine learning models trained on historical network or resource usage data from pods to forecast future bandwidth or CPU / memory needs. This predictive capability enables the Kubernetes cluster to apply different policies in advance, thereby maintaining optimal network and application performance and resource utilization. The key steps in this process are as follows:

1) **Data Collection**: A dummy application was deployed to simulate periodic bursts of heavy traffic directed at a specific pod acting as the Gateway API for multiple edge devices that produced the traffic. The dataset includes daily traffic patterns, showing periods of peak bandwidth utilization, nearly 2GB/s between 09:10-12:30 and 15:45-20:00. In total, 86000 rows/samples were collected, with each sample corresponding to bandwidth measurement recorded at one-minute intervals.
2) **Feature Engineering and Modelling**: Using the historical data collected, as described above, features were extracted from bandwidth and / or CPU measurements over time, to train a model to proactively predict high bandwidth or high CPU load usage, thus to proactively apply the QoS or vertical scaling policy.
   Since the objective was to predict future and periodical patterns of the bandwidth value of the pod, the extracted features are aligned with this goal. More specifically the features extracted are the following:

TABLE I
THE FEATURES INPUT TO THE AI MODEL

Features	
Feature	**Description**
hour	The hour of the day (0-23) when the measurement was recorded.
minute	The minute within the hour (0-59) when the measurement was recorded.
day	The day of the week (0-6), where 0 represents Sunday and 6 represents Saturday.
bandwidth / CPU usage lag 1	The bandwidth measurement from the previous time step (lag 1).
bandwidth / CPU usage lag 2	The bandwidth measurement from two time steps before the current one (lag 2).
bandwidth / CPU usage lag 3	The bandwidth measurement from three time steps before the current one (lag 3).

3) **Predictive Model**: The target variable is the high bandwidth binary value. which essentially holds values of 1, when bandwidth measured is above a specific predetermined threshold value, otherwise is 0. The threshold in our case is set to 0.5 GB/s.
   For the training, a Random Forest Classifier is used for the prediction if there is high bandwidth usage or not. We only used Random Forest, since from its results, a perfect score was achieved, as can be seen in the results table.
4) **AI-driven Policy Enhancement**: Utilizing historical data and AI Models to predict future network traffic patterns and pod behaviour, so the above policies can be proactively enforced resulting in runtime adaptations.

V. EXPERIMENTAL RESULTS

This section summarizes the results from both the evaluation of the AI models and the real time application of the network policies.

Tables III and IV summarize the performance of four distinct classification algorithms, i.e. Random Forest, Logistic Regression, Decision Tree, and a 5-Nearest Neighbors approach, for predicting the high-bandwidth class in the Kubernetes-based testbed. As shown, all methods achieve near-perfect results in terms of accuracy, precision, and recall (i.e. 1.0 across the board, except for a slightly lower recall of 0.97 and 0.99 by the logistic model for CPU and bandwidth

prediction, respectively). These exemplary metrics indicate that the trained models are highly effective at distinguishing between high-bandwidth / high-CPU load and other classes, demonstrating robust learning and generalization capabilities and showcasing the potential of AI in enhancing network intelligence.

Figures 7 and 8 illustrate the effect of applying AI-driven policies to dynamically adjust the network configuration and bandwidth allocation for the Gateway API pod. In Figure 7 (prior to policy enforcement), the pod sustains an incoming data rate of around 25.6 MB/s. After the policy is applied (Figure 8), the pod is capable of reaching or surpassing 2 GB/s, peaking at approximately 2.16 GB/s, thereby accommodating significantly higher traffic loads. This increase in throughput highlights the system's ability to scale its resources in real time. Such a policy-based approach ensures that mission-critical Kubernetes services can dynamically adapt to traffic surges, helping to maintain optimal performance and minimize downtime.

Figure 9 illustrates the CPU usage metric of the Gateway API pod over time. It is visible that before 9:59, CPU usage exceeds the allocated CPU resource request and limit, necessitating vertical scaling. At 9:59, the system performs vertical scaling, increasing the pod's allocated resources (request and limit).

Figure 10 illustrates the effect of applying an AI-driven policy to proactively increase CPU core allocation for the Gateway API pod. Here, the AI model notifies for a rising CPU load and the orchestrator performs the vertical scaling policy at 9:46 (indicated by the change in the red and orange lines), several minutes before the pod experiences increasing demand around 9:49. This proactive scaling eliminates the need for a potentially costly reactive approach, ensuring efficient resource allocation.

TABLE II
RESULTS

	Accuracy	Precision	mAP50
YOLOv8 small	0.925	1.000	0.460
YOLOv8 large	0.950	1.000	0.470

TABLE III
EVALUATION METRICS AND PERFORMANCE OF DIFFERENT AI MODELS FOR THE CPU HIGH DEMAND CLASSIFICATION

	Accuracy	Precision	Recall
Random Forest	1.000	1.000	1.000
Logistic	0.99	0.99	0.97
Decision Tree	1.000	1.000	1.000
Nearest Neighbors (5)	1.000	1.000	1.000

VI. CONCLUSION

In a nutshell, the findings of this work validate the benefits of combining AI-driven traffic classification with agent-based and automated network policy enforcement in Kubernetes. By accurately forecasting high-bandwidth flows in real time, the E2C network orchestrator can make informed decisions and proactively allocate resources, enhancing scalability, QoS, and the resilience of the underlying network infrastructure.

TABLE IV
EVALUATION METRICS AND PERFORMANCE OF DIFFERENT AI MODELS FOR THE BANDWIDTH HIGH DEMAND CLASSIFICATION

	Accuracy	Precision	Recall
Random Forest	1.000	1.000	1.000
Logistic	1.000	1.000	0.99
Decision Tree	1.000	1.000	1.000
Nearest Neighbors (5)	1.000	1.000	1.000

Fig. 7. The bandwidth of the Gateway API pod before the applied policy.

In conclusion, the developed adaptive, policy-driven framework for network intelligence not only optimizes E2C applications with remarkable effectiveness but also does so by meticulously enforcing network rules derived from learned application patterns. This is achieved through decentralized decision-making coupled with cutting-edge AI-driven application profiling. By seamlessly integrating advanced Artificial Intelligence for predictive dimensioning and sophisticated pattern mining, the framework empowers dynamic and adaptable network resource allocation, significantly amplifying network capacity and enhancing quantitative QoS correlations. The innovative deployment of cutting-edge AI models, which utilize real-world monitoring data from E2C environments, further demonstrates the framework's ability to enforce adaptive network policies and execute well-informed decisions. These capabilities collectively create more intelligent and resilient networks, ensuring unparalleled performance and unwavering reliability for E2C applications.

Future efforts will focus on scaling the adaptive, policy-driven framework for network intelligence to support larger, more complex networks and refining AI models for better predictive accuracy and policy enforcement, enabling wider applicability and enhanced performance.

VII. MATCH & CONTRIBUTION

This contribution aligns closely with the theme of the ICE IEEE 2025 conference on "AI-driven Industrial Transformation: Digital Leadership in Technology, Engineering, Innovation & Entrepreneurship." The paper presents an adaptive, AI-driven framework for network intelligence in the Edge-to-Cloud (E2C) continuum, addressing the growing need for dynamic, resilient, and intelligent networks in highly distributed industrial and technological environments.

Fig. 8. The bandwidth of the Gateway API pod after the applied policy. The capacity of bandwidth is increased to support the increased demand where the received bandwidth goes from 30MB/s to 2GB/s.

Fig. 9. Graphical representation of network policy adaptation showing CPU usage in cores for the Gateway API pod (yellow line), requested CPU (red line), and CPU limits (orange line) over time. The graph illustrates a restart period with no CPU usage, followed by increased resource utilization after the application of a new policy that allowed more CPU resources.

By leveraging real-world monitoring data and deploying advanced AI models, the framework enables proactive resource allocation, real-time network optimization, and dynamic policy enforcement. These capabilities support industrial transformation by facilitating ultra-reliable communication, optimal resource usage, and service innovation across heterogeneous and large-scale infrastructures.

The integration of AI-driven decision-making and dynamic policy mechanisms fosters digital leadership in networked systems, aligning with the conference's focus on utilizing cutting-edge AI technologies to drive engineering advancements, technological innovation, and entrepreneurial progress. This work significantly contributes to the future of industrial networking, demonstrating practical pathways for embedding intelligence at the core of next-generation Edge-to-Cloud applications.

ACKNOWLEDGMENT

This work has received funding from the Research and Innovation Foundation under Restart Research 2016-2020 Programme and the DUAL USE/0922/0024 agreement of CYGNUS project, and the European Union's Horizon Europe TALON project with GA No 101070181.

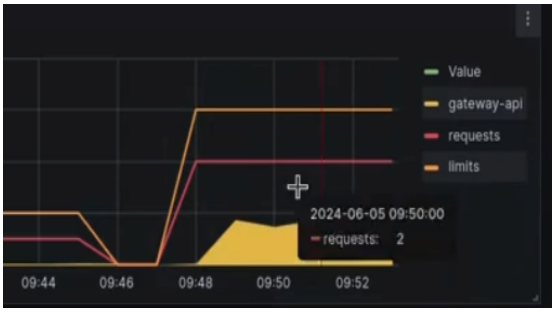


Fig. 10. Graphical representation of CPU resource allocation in cores showing the proactive application of a new policy informed by AI-driven CPU load prediction. The policy is enforced just before resource starvation, as indicated by the seamless increase in CPU limits (orange line) and requests (red line), preventing disruption and ensuring optimal resource utilization (yellow area).

REFERENCES

[1] A. Zappone, M. Di Renzo, M. Debbah, T. T. Lam, and X. Qian, "Model-aided wireless artificial intelligence: Embedding expert knowledge in deep neural networks for wireless system optimization," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 60–69, 2019.

[2] D. Balouek-Thomert, I. Rodero, and M. Parashar, "Evaluating policy-driven adaptation on the edge-to-cloud continuum," in *2021 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC)*. IEEE, 2021, pp. 11–20.

[3] I. Čilić and I. P. Žarko, "Adaptive data-driven routing for edge-to-cloud continuum: A content-based publish/subscribe approach," in *Global IoT Summit*. Springer, 2022, pp. 29–42.

[4] M. Wang, T. Schirmer, T. Pfandzelter, and D. Bermbach, "Shutpub: Publisher-side filtering for content-based pub/sub on the edge," in *Proceedings of the 7th International Workshop on Edge Systems, Analytics and Networking*, 2024, pp. 13–18.

[5] T. Pfandzelter and D. Bermbach, "Towards a benchmark for fog data processing," in *2023 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2023, pp. 92–98.

[6] S. Sekigawa, C. Sasaki, and A. Tagami, "Toward a cloud-native telecom infrastructure: Analysis and evaluations of kubernetes networking," in *2022 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2022, pp. 838–843.

[7] The Kubernetes Authors, "Kubernetes: Production-grade container orchestration," 2024, accessed: 2024-01-30. [Online]. Available: https://kubernetes.io/

[8] The Kube-OVN Developers, "Kube-ovn: A kubernetes-native sdn based on ovn," 2024, accessed: 2024-01-30. [Online]. Available: https://kube-ovn.io/

[9] The Prometheus Authors, "Prometheus: Monitoring system & time series database," 2024, accessed: 2024-01-30. [Online]. Available: https://prometheus.io/

[10] Kepler Developers, "Kepler: Kubernetes-based efficient power level exporter," 2024, accessed: 2024-01-30. [Online]. Available: https://kepler.sustainable-computing.io/

[11] InfluxData, "Influxdb: Open source time series database," 2024, accessed: 2024-01-30. [Online]. Available: https://www.influxdata.com/

[12] The Prometheus Authors, "Prometheus node exporter: Collecting system metrics," 2024, accessed: 2024-01-30. [Online]. Available: https://github.com/prometheus/node_exporter

[13] The Kubernetes Authors, "Kube-state-metrics: Add-on agent to generate cluster state metrics," 2024, accessed: 2024-01-30. [Online]. Available: https://github.com/kubernetes/kube-state-metrics

[14] A. Cañete, A. Rodríguez, M. Amor, and L. Fuentes, "Energy-aware placement of network functions in edge-based infrastructures with open source mano and kubernetes," in *International Conference on Service-Oriented Computing*. Springer, 2022, pp. 183–195.