2023 International Conference on Computing, Networking and Communications (ICNC) | 978-1-6654-5719-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICNC57223.2023.10074183

Network Services Management using Programmable Data Planes for Visual Cloud Computing

Alicia Esquivel Morel, Prasad Calyam, Chengyi Qu
Durbek Gafurov, *University of Missouri - Columbia*
{ace6qv, calyamp, cqy78, dgvkh}@umsystem.edu

Cong Wang, Komal Thareja, Anirban Mandal
RENCI, University of North Carolina at Chapel Hill
{cwang, kthare10, anirban}@renci.org

Eric Lyons, Michael Zink
University of Massachusetts Amherst
{elyons, mzink}@umass.edu

George Papadimitriou, Ewa Deelman
University of Southern California
{georgpap, deelman}@isi.edu

***Abstract*—Visual Cloud Computing (VCC) applications provide highly efficient solutions in video data processing pipelines on edge/cloud infrastructures. These applications and their infrastructures demand end-to-end monitoring and fine-grained application traffic control to meet user quality of experience requirements. In this paper, we propose a novel network services management methodology for VCC applications by leveraging the advantages of programmable data planes enabled by Protocol-independent Packet Processors (P4). Specifically, we define a custom fixed-length application header and use it to improve the performance of a video streaming application through congestion avoidance using Multi-Hop Route Inspection (MRI), a variant of In-band Network Telemetry (INT), and switch port forwarding (tunneling) capabilities. For evaluation experiments, we use P4 per-packet telemetry metadata for routing paths, ingress/egress timestamps, queue occupancy in a given node, and egress port link utilization in a VCC testbed on the NSF-supported FABRIC infrastructure. Our experiment results demonstrate performance improvement obtained with our methodology in terms of both packet loss and throughput metrics.**

***Index Terms*—Programmable Data Planes, In-band Network Telemetry, Video Delivery, Traffic Engineering**

I. Introduction

With today's boost of 5G technologies, the highly distributed, edge-to-cloud infrastructures are rapidly growing to support the latency-critical applications in e.g., public safety, transportation [1]. Corresponding edge-to-cloud computing systems require precise and flexible monitoring mechanisms that are open, and able to capture short-lived events i.e., micro-burst traffic events that can occur in milliseconds. Moreover, new monitoring methods are needed to detect service issues quickly, track rapid network state changes, and provide an end-to-end state visibility for pertinent network control actions using programmable technologies. As shown in Figure 1, edge-to-cloud computing systems benefit from new monitoring methodologies that offer real-time network status feedback for network service management in the presence of faults. Monitoring methodologies for *Visual Cloud Computing (VCC)* applications has to be fine-grained, customized, and highly scalable to meet the requirements of visibility and control, at both the data and control planes to enable online traffic engineering and enhanced application quality of experience.

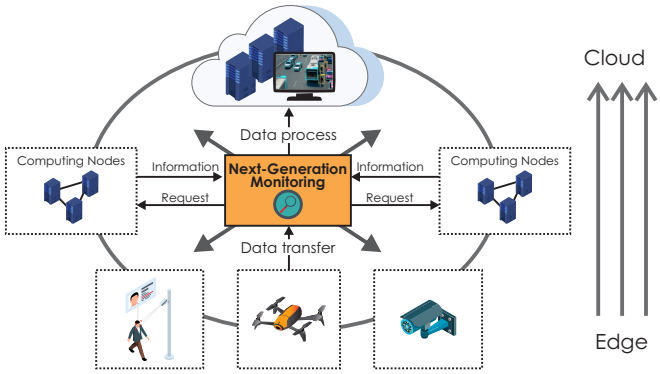

Figure 1: Applications with fine-grained, customizable, and scalable monitoring to adapt data and control planes.

VCC applications involving e.g., face recognition, or drone video analytics [2] require high traffic forwarding capabilities along with high quality video/image data transmission. The existing Software Defined Networking (SDN) [3] approach enables separation of data and control plane using controller algorithms to dynamically update network state on routers and switches to process packets as per application requirements. Consequently, the new monitoring methodologies along with SDN can be adapted for network customization of control plane components and protocols via command line and web interfaces, or Application Programming Interfaces (APIs) in 5G or data center networks.

Recently, new monitoring methodologies have emerged to provide better network-awareness for programmable network services deployment and their management. For example, the telemetry work in [4] allows network nodes to stream performance parameters such as Bit Error Rate (BER) every few seconds, at a more increased rate than conventional solutions founded on the Simple Network Management Protocol (SNMP). More promisingly, In-band Network Telemetry (INT) based on Programming Protocol-independent Packet Processors (P4) [5] (a widely used abstraction, programming language that is compatible with SDN) has been shown to effectively enable monitoring of edge-to-cloud computing systems [6]. INT relies on additional packet headers used

to carry metadata information retrieved from the forwarding plane. For instance, INT can be used to gather the time spent in a queue by every packet in each traveled node within an edge-to-cloud computing path of a specific 5G service [7].

In this paper, we propose a novel network services management approach for VCC applications by leveraging the advantages of programmable data planes enabled with P4/INT to improve the quality of a video streaming application. We implement a monitoring and debugging system, based on Multi-Hop Route Inspection (MRI), a variant of INT and switch port forwarding (tunneling) capabilities. Our solution delivers per-hop granular data in the forwarding plane, allowing the observation of changes in the transmission of packets. With INT, the exact metrics on network performance when dealing with advanced and automatic forwarding capabilities through high bandwidth video/image data are introduced. Using custom fixed-length application headers, switches are aware of congestion, information is extracted on where the path was conceived, and which packets are associated with it. This allows for "top-down" data flow management to dynamically route packets to a different path when congestion happens through port forwarding (tunneling). As a result, this improves the performance of a video streaming application in an edge-cloud computing system.

We implement and evaluate our methodology on FABRIC [8], which is a state-of-the-art network research infrastructure supported by the National Science Foundation (NSF). Using a custom FABRIC testbed, we evaluate the performance using P4 per-packet telemetry metadata for routing paths, ingress/egress timestamps, queue occupancy in a given node, and egress port link utilization. Our experiments employ packet traces with timestamps of packet arrivals at an input interface of a P4 switch, and the timestamps when they are forwarded by the output interface. Experiment results demonstrate the benefits of our methodology implemented using P4 (MRI + port forwarding) for delivering packet processing time (data transmission speeds) required in VCC applications. Specifically, the experiment results show how our custom P4 program executes well under network congestion in terms of both packet loss and throughput metrics in video streaming.

The remainder of this paper is organized as follows: Section II presents the related work. Section II-C details the current limitations impacting VCC application performance. In Section III, we detail our proposed network services management methodology. Section IV presents the performance evaluation results. Lastly, Section V concludes the paper.

II. Related Work

A. Monitoring Strategies on Non-Traditional Data Planes

Network monitoring is of crucial significance for adequate network control, allowing to continuously observe a network's behavior, and to trigger strategies in case of failures and anomalies [9]. Monitoring methods may differ, but more recent techniques use sophisticated methods involving programmability of both control and data planes. Authors in [10] proposed a monitoring instrument using programmable data planes. Their approach concentrates on data plane programmability to perform tasks that are usually accomplished solely by the control plane (e.g., traffic monitoring and information gathering). They demonstrated methods to create constant, snapshot-free analysis of network traffic. Similarly, authors in [11] presented a flow monitoring tool implemented on programmable data planes and INT. Their technique focuses on cloud network providers, allowing a system to monitor a set of flows according to the application's needs. Particularly, their method utilizes an additive weighting procedure to find the most appropriate network devices to inform the flow telemetry in a P4 network. Authors in [12] observe flows with application-level granularity in the data plane, and supply the control plane with information on the existence of organized sub-flows, and their statistics.

The above works support our claim that programmable data planes can allow highly precise and flexible monitoring of network functionality. In addition, our literature review clearly shows that there is a lack of investigations addressing P4/INT monitoring strategies for end-to-end network awareness, specifically, network services management for VCC applications. The novelty of our work is also in the design of algorithms for end-to-end monitoring based on P4/INT strategies considering realistic and scaled deployments of VCC applications.

B. Visual Cloud Computing Application Performance

We consider exemplar VCC application scenario (e.g., aerial video surveillance data collection), and their demand for end-to-end monitoring and fine-grained application traffic control to meet user experience requirements in the presence of e.g., congestion bottlenecks. Authors in [13] studied VCC application deployments with SDN by specifying a collection, computation, and consumption (3C) architecture over edge-to-cloud computing resources. Benefits of SDN are presented for on-demand computation offloading with congestion-avoiding traffic steering to improve remote user quality of experience.

The work in [14] studied a predictive cyber-foraging use case for VCC in large-scale IoT systems. This paper focuses on the problem of growth and capability for IoT applications to operate complex video processing tasks, which demands scalability of those services while handling device heterogeneity or biased communication. Similarly, authors in [15] propose an edge-computing video analytics using real-time traffic monitoring in a smart city context. They run video analytics directly on the device, and only the results of the processing are sent to a cloud platform. The authors highlighted the need for accurate analysis when the video is generated in real-time, and how this lowers the network bandwidth requirement, enhancing the effectiveness of video analysis. Our work benefits these VCC application scenarios that involve remote consumption of video content and services to provide visual situational awareness, centered on the efficient delivery of both live and file-based video streams.

C. VCC Applications and Their Challenges

In this section, we discuss aspects of video processing using edge-to-cloud computing resources, and network traffic management using programmable data planes. In addition, we introduce our study challenges for VCC applications.

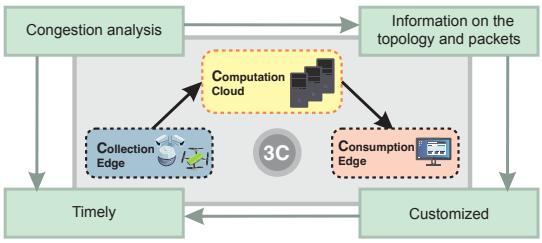

Figure 2: VCC pipeline to integrate the network edge with core cloud computing resources using monitoring systems that enable congestion analysis using timely and customized information on the network topology and individual packets.

1) Video Processing using Edge-to-Cloud Computing: We consider exemplar VCC application scenarios because they demand highly efficient solutions in video data processing pipelines on edge-to-cloud computing infrastructures. Figure 2 shows the collection, computation, and consumption (3C) architecture for an incident response scenario involving video surveillance. The collection edge collects data in the incident site and transfers the data to core cloud services using SDN-enabled networks. This aims to leverage the high-capacity cloud infrastructure for visual analytics involving machine learning algorithms that perform e.g., object detection and tracking to provide visual situational awareness for decision makers at the consumption edge [16]. The VCC application data collection and analysis process between the edge devices and core cloud resources thus requires end-to-end monitoring and fine-grained application traffic management.

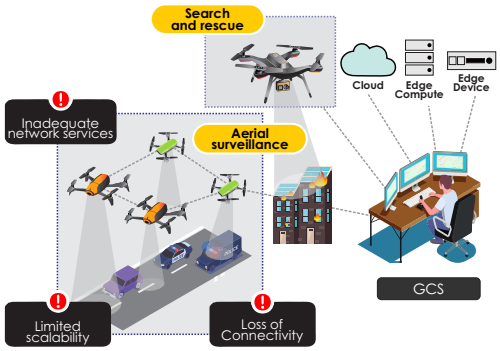

Figure 3: Challenges for network services management that need to be overcome to guarantee satisfactory performance of network-edge based applications.

In our study, the network services management for VCC applications targets the delivery of video streaming by ensuring stability in the performance of the network paths, even in the presence of congestion bottlenecks. We assume Unmanned Aerial Vehicle (UAV) or drones-sourced video streaming as the video source. Figure 3 illustrates how drones transmit videos and images at high data rates for critical purposes, i.e., search and rescue or aerial surveillance. A Ground Control Station (GCS) manages the command control, processes the data, and provides various solutions, making decisions to satisfy the application and data rate requirements (e.g., to stream a high or a low definition video) [17]. Data collection in VCC is made more challenging with heterogeneous technology or IoT devices (e.g., cameras, sensors) deployed in austere networks with limited connectivity or performance reliability, and inadequate/inflexible network services.

2) Network Services Management for VCC Applications: Several factors can impact a given VCC application's performance. For instance, in terms of network control, SDN control plane is a candidate solution to sustain automated network configuration by means of an open API (e.g., OpenFlow), isolating and abstracting the data plane from the orchestration at the control layer. However, SDN solutions need to support heterogeneous softwarized networks, such as next-generation edge-to-cloud computing nodes with IoT gateways that operate massive sensors, cameras and other smart devices. Specifically, they need to support: refined and automated forwarding capabilities, per-packet treatment and manipulation processes, high-precision traffic monitoring/telemetry, congestion detection, and services with intense conditions (e.g., low jitter, bounded low latency).

Another challenge for network services management is in the ability to suitably leverage P4/INT network monitoring capabilities to meet VCC application demands. Most existing monitoring infrastructures are restricted to specific network segments and it is challenging to achieve the synchronization between latency and location information. Network monitoring infrastructures for VCC applications need to be efficient and cost-effective, and should capture short-lived events, providing visibility for flexible network control. Our use-case scenarios demand end-to-end monitoring and fine-grained traffic control in order to meet user experience requirements in the presence of congestion bottlenecks. Ideally, related solutions should support and refine automation in terms of processes, high-precision traffic monitoring, or congestion detection.

III. INT-based Methodology for Network Services Management

A. Programmable Data Plane and INT strategies

For our study purpose, we adapt user-defined software that describes the behavior of how packets are processed, conceived, tested, and deployed. In particular, we apply network traffic control for enhancing application performance for data processing involving video streaming. Taking a different approach from standardized networking, it is critical to highlight that traditional network vendors set a fixed protocol design, and this significantly reduces the flexibility in how the network can operate. This can also lead to sub-optimal performance when dealing with the transmission of video streams that cannot be handled by the parameters set in place. Our approach employs networking techniques that greatly

reduce the constraints that network vendors put in place when trying to configure a network for a specific goal. Introducing programmable data plane technologies, not only gives users the freedom to operate the network flow but also greatly improves monitoring of networking metrics when dealing with cutting-edge and automatic forwarding capabilities for handling high throughput video/image data streams.

B. INT for fine-grained application traffic control

In latency-critical services, application performance and Quality of Service (QoS) can be used on a per-flow basis with high scheduling priority. Also, these high-class packets compete for the same queuing resources, and in general, unequal treatment may be experienced in addition to latency delta variations over time that impact jitter in video streams. To deal with this problem, P4 language, exploiting INT, can deliver per-packet QoS with desired performance [18]. INT information can be added as a header, and it can be modified in all high-class packets per crossed node. This can help to report the consumed time in the outgoing queues across the network. The INT data can be examined, including cumulative delay, minimizing jitter, and the highest experienced end-to-end latency [19]. Finally, another interesting feature is the unprecedented amount of monitoring, telemetry, and logging data that may be input to machine learning based adaptive designs and strategies, with the main goal to boost the overall network performance [7].

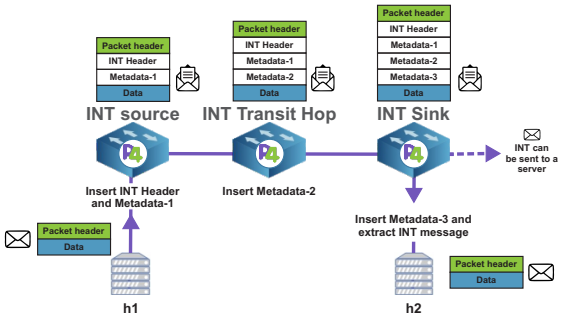

Figure 4: INT process deployed in a custom data packet sent between two hosts $h1$ to $h2$ using three programmable devices; INT source, transit hop and sink add headers to report the time spent in the outgoing queues across the network path.

Figure 4 depicts our INT implementation for a particular VCC application case in which host $h1$ sends a data packet to $h2$. Host $h1$ and $h2$ are connected to each other through three programmable P4 switches. Our queuing rule for the INT process exhibit properties of the FIFO (First-in-First-out) scheduling algorithm. The procedure starts in host $h1$ which sends a data packet with a header. This INT header with Metadata-1 is sent through three P4 programmable switches. These switches add headers and metadata through INT source, transit hop, and sink. Finally, this packet with the INT header and the metadata that is transferred through the switches can be sent over to a server for further analysis, and reach the destination host $h2$, which will subsequently extract the INT message with the inserted metadata.

C. Network control to adapt application performance

We address the need of a network monitoring framework that delivers per-hop granular data in the forwarding plane, allowing for observation differences in e.g., packet transmission, and latency per node. For this, we introduce a new layer of softwarization and use a fully-programmable environment, rather than the partial-programmable environment carried by SDN/Openflow tools employed as solutions for service-centric models. Particularly, we rely on P4 as a domain-specific language for network devices [5], to determine how data plane devices (switches, routers, filters) process packets. OpenFlow supports partial programmability, whereas P4 improves the extent of flexibility and creates the possibility of determining customized behavior per packet processor.

D. MRI and port forwarding implementation

Our solution is possible through the programmability and flexibility of the P4 switches, decreasing delay, and improving the video transmission quality. P4 switches do not have any information about congestion in the topology unless this information is included in the packet. The approach to include this information into the packet headers is through integration of INT. **Algorithm 1** describes the ingress pipeline as a function and how the port forwarding function is applied. Inputs for this algorithm represent programmable blocks that are platform-independent: *ingr_intr_metadata*, *ingr_prsr_metadata*, *ingr_dprsr_metadata*. The Function *Ingress()* takes these input values to carry the INT ingress pipeline, getting the outputs for *header, local_metadata, ingr_dpr_metadata*. In line 9, INT source table is applied if the ingress port matches the local metadata source as it is stated in line 10, otherwise, if the INT is valid then, the switch properties are saved for the egress pipeline, as specified in line 24.

Algorithm 2 represents the INT Egress pipeline. Inputs for this algorithm represent programmable blocks that are platform-independent: *header*, *local_metadata*, *egr_intr_metadata*, *egr_prsr_metadata*, respectively. The function *Egress()*'s main purpose is to restore the packet to the actual header and remove the INT information, getting the outputs for *header, local_metadata, egr_dprsr_metadata, and egr_port_metadata*, respectively. If the node is in a transit hop, this will add the INT metadata, resulting in a packet telemetry report. Multi-Hop Route Inspection (MRI), which is a variant of INT allows to track the path and the length of the queues that every packet traverses. The switch appends an ID and queues the size to the header stack of every packet. After that, the destination, and the series of switch IDs corresponding to the path are tracked by the queue size of the port at the switch. Note that our P4 program describes a packet-processing pipeline, but the rules controlling packet processing are inserted into the pipeline by the control plane.

When a rule corresponds to a packet, its action is gathered with parameters provided by the control plane as part of the rule. The rule we propose involves port forwarding (tunneling)

Algorithm 1: INT Ingress pipeline

```
   Input: header, local_metadata, ingr_intr_metadata, ingr_prsr_metadata,
          ingr_dprsr_metadata
   Output: header, local_metadata, ingr_dprsr_metadata
1  Function Ingress():
2      Begin
3      Basic Port Forward Function
4      if IPv4 destination address matches with Longest Prefix Match then
5          Set the Egress Port;
6      end
7      End
8      Begin
9      Apply INT Source Table
10     if Ingress Port Matches Exactly then
11         Set local_metadata.source = True;
12     end
13     Apply INT Sink Table
14     if Egress Port Matches Exactly then
15         Set local_metadata.sink = True;
16     end
17     End if local_metadata.source = True then
18         Insert INT Shim Header and INT Header to the Packet;
19     end
20     if local_metadata.sink = True and Valid INT Header then
21         Ingress to Egress Mirror, Sink Node;
22     end
23     if Valid INT Header then
24         Save switch properties for Egress;
25     end
26 return
```

Algorithm 2: INT Egress pipeline

```
   Input: header, local_metadata, egr_intr_metadata, egr_prsr_metadata
   Output: header, local_metadata, egr_dprsr_metadata, egr_port_metadata
1  Function Egress():
2      if Packet has a Valid INT Header then
3          Determine if Node is a Sink ;
4      end
5      if Node is a Sink then
6          Restore Packet Original Header and Remove INT information;
7          else
8              Node is a Transit Hop ; Add INT metadata
9          end
10     end
11     if Packet is Mirrored then
12         Packet is a Telemetry Report ;
13     end
14 return
```

to re-route the packet flow that is moving via congested links, avoiding delays, and improving the quality and performance of the video transferred. For instance, if the *Packet has a valid INT Heather*, then this will determine if the node is a sink, as it is observed in line 3. In addition, in line 5, *if Node is a Sink*, then the packet will be restored to the original header and the INT information will be removed. Other aspects, including lines 8 and 12, report what happens when the node is a *Node is a Transit Hop*, or if the *Packet is Mirrored*.

In Figure 5, we present an aerial video surveillance data collection scenario. In the figure we can observe a congestion bottleneck between $h11$, $s1$, $s2$, and $h22$. This scenario is deployed with regular switches i.e., without our implementation, with congestion being observed in the link with the shortest path. A shortest path algorithm would route all the packets through the shortest link, and consequently generate congestion. In our solution, the port forwarding algorithm is implemented in the hosts that guides each switch in the network to send the packet to a specific port. Basically, the

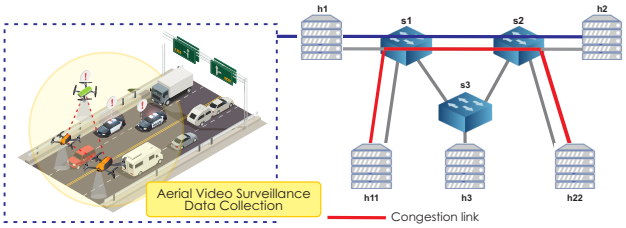

Figure 5: Aerial video surveillance data collection scenario experiencing congestion bottlenecks without P4 devices.

host puts a stack of output ports in the packet. To achieve that, the routing scheme needs to know all possible paths and needs to have a feedback loop from the system. This will provide information about the topology, paths, and the existence of congestion. We use the *Iperf* tool to send low-rate traffic from $h1$ to $h2$ and high rate cross traffic from $h11$ to $h22$. The link between $s1$ and $s2$ is common between the flows and causes a bottleneck by reducing end-to-end bandwidth to 200 Mbps for experimental purposes. Capturing packets at $h2$ shows build up of a high queue size for the congested link. Once the destination receives the information about the congestion, this destination host needs to send this information to the source host, and the source will plan the most suitable (non-shortest) path for the packet.

IV. Performance Evaluation

A. Testbed Setup and Evaluation Metrics

To evaluate the efficacy of our solution, we define criteria for congestion control metrics to set performance expectations, and to provide a baseline for comparison of normal network performance (without the integration of P4 programmable switches). Our solution evaluation goal is to observe reduction of packet loss in video data transmission, and also delivery with increased throughput (bytes/s, packets per second (PPS)).

The FABRIC infrastructure [8] (Adaptive ProgrammaBle Research Infrastructure for Computer Science and Science Application), a very recently deployed infrastructure (funded by National Science Foundation) enables cutting-edge networking experimentation and research at-scale. For our experiments, we leverage the FABRIC infrastructure with distributed resources relating to latest programmable network technologies such as P4 and dedicated optical links. P4 switches are emulated and installed on FABRIC nodes we dedicated to switching. The current revision of the P4 language $P4_{16}$ is instantiated, and the BMV_2 *(Behavioral model version 2)* was installed in the switches. We benefit from the P4Runtime tool to programmatically install rules into each switch. For our solution implementation, we allocated eight nodes i.e., five hosts (server and end-users), and three basic NICs (Virtual NICs implemented as Single Root I/O Virtualization, on top of ConnectX-6 physical cards). Disk of the switches are 100 GB each, whereas each of the hosts have 10 GB disk.

Our experiments are based on a real-world use-case in which a UAV surveillance system a.k.a. swarm of drones,

composed of six drones (connected via an ad-hoc network) is responsible for collecting video data through a VCC application. For the video transmission, we installed VLC, an open source multimedia engine with the capability to stream, convert, encode and manipulate streams into numerous formats. We consider a critical scenario where high throughput and low latency are required to sustain the application, enabling reliable and timely process, and transmission of video streams.

B. MRI+Port Forwarding Results

For our experiments, we relied on a topology (see Figure 6) comprised of: five hosts which act as servers of the drone swarm video streams, end-users of the video streams [$h1$, $h2$, $h3$, $h11$, $h22$] corresponding to decision makers, and three P4 programmable switches [$s1$, $s2$, $s3$] of the video content delivery network. Our solution, P4 program and algorithms are implemented within the switches [$s1$, $s2$, $s3$]. Specifically, we combined concepts of MRI and port forwarding algorithms, creating a control system that knows the entire topology, being aware of any congestion that might occur, and taking the appropriate steps to overcome it. Our system is also able to diagnose congestion issues at ingress level, thus, solving the issue before it reaches the egress level.

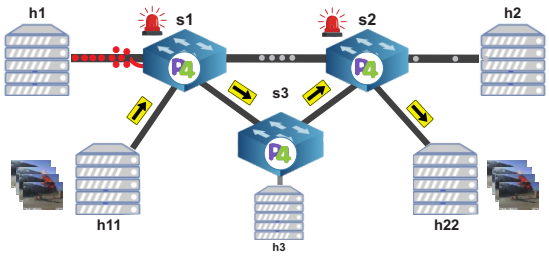

Figure 6: Experiment testbed for solution implementation for a VCC application deployed in a FABRIC testbed with MRI+Port forwarding (tunneling) to reroute the packets through a link with no congestion.

In our initial settings, the bandwidth of the links between all the nodes was 200 Mbps. We start by creating congestion between $h1$, $s1$, $s2$, and $h2$ using the *Iperf* tool. In our experiments, we collect data from traffic measurements to compare the benefits of using programmable P4 switches versus regular switches. We set up two cross traffic congestion cases of 400 Mbps and 800 Mbps, respectively. Correspondingly, we collect metrics for average PPS, Max. Delta (ms), percentage of packet loss, and RTP packets as shown in Tables I and II.

Table I: Traffic measurements to compare benefits of using P4 switches versus regular switches in presence of 400 Mbps cross traffic congestion on path between $s1$ and $s2$ with capacity of 200 Mbps.

Metric	W/o P4	With P4	Difference%
Average PPS	17.392	32.225	+85.29%
Max. Delta (ms)	1066.61	136.591	-87.19%
Packet Loss %	59.818	0	-100.00%
RTP Packets	1548.333	3855.9	+149.04%

When packets were sent from $h1$ to $h2$, we observe a large percentage of packet loss and delay. In a regular setting, without our P4 implementation, the packets would be transmitted through switches $s1$ and $s2$. With our P4/INT solution based on MRI+port forwarding capabilities, we were able to automatically route those packets through an alternative route, $s1$-$s3$-$s2$, avoiding the network congestion between $s1$-$s2$. In our video transmission experiment pipeline, we start transmitting from $h1$ to $h2$, then at the same time, from $h11$ to $h22$. This transmission will take an alternative route, through switches $s1$, $s2$, $s3$, and every packet will include the switch trace and other information. The video transmission from $h1$ to $h2$, does not experience considerable delay. Similarly, as we started streaming video from $h11$ to $h22$, the number of packets received per second increased, with no considerable delays. When streaming video from $h1$ to $h2$ and $h11$ to $h22$ simultaneously, notable information such as the the queue depths are at zero, which is an indicator that no congestion is present when streaming from all known hosts. This information consists of the programmable switches traces for alternative routes for packet forwarding, and the MRI congestion information.

Table II: Traffic measurements to compare benefits of using P4 switches versus regular switches in presence of 800 Mbps cross traffic congestion on path between $s1$ and $s2$ with capacity of 200 Mbps.

Metric	W/o P4	With P4	Difference%
Average PPS	14.475	32.225	+122.63%
Max. Delta (ms)	2535.2	136.591	-94.61%
Packet Loss %	66.5	0	-100.00%
RTP Packets	1291.6	3855.9	198.54%

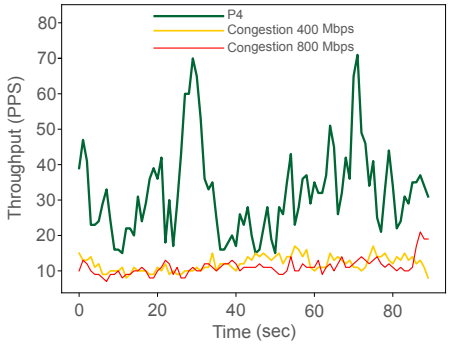

Figure 7: Throughput measurements in PPS to compare benefits of using P4 switches versus regular switches in presence of 400 Mbps and 800 Mbps cross traffic congestion cases on path between $s1$ and $s2$ with capacity of 200 Mbps.

Our P4/INT solution shows significant benefits for service deployment methodology for VCC applications. Results in Figure 7 show increased Average PPS in the presence of 400 Mbps and 800 Mbps cross traffic congestion on paths between *s1* and *s2* by 85.29% and 122.63% respectively, while reducing maximum delta (ms) by 87.19%, and 94.61% as shown in Figure 8. Results in Figure 7 show higher network performance without delay in the packet delivery using our P4/INT solution, as seen in the number of packets per second (PPS) i.e., the throughput for the programmable P4 switches versus using regular switches in the presence of 400 Mbps and 800 Mbps congestion cases. Our real-time video streaming

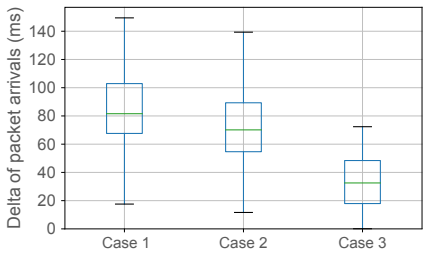

Figure 8: Mean Delta measurements to show impact of increased congestion in inter-packet arrival times on the path between $s1$ and $s2$ with capacity of 200 Mbps for the cases: w/o P4 and Congestion of 800 Mbps (Case 1), w/o P4 and Congestion of 400 Mbps (Case 2), and with P4 (Case 3).

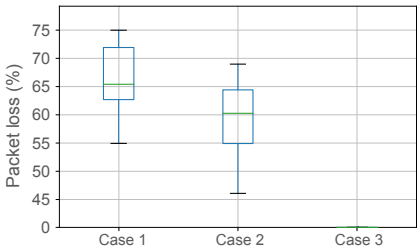

Figure 9: Packet Loss measurements to show impact of increased congestion on the path between $s1$ and $s2$ with capacity of 200 Mbps for the cases: w/o P4 and Congestion of 800 Mbps (Case 1), w/o P4 and Congestion of 400 Mbps (Case 2), and with P4 (Case 3).

application use case and the MRI+port forwarding solution gives us important information that includes the queue depths, an indicator of the presence of congestion. We can also see in both 400 Mbps and 800 Mbps cross traffic congestion cases, a packet loss of 0% is experience with P4 as shown in Figure 9, which further demonstrates the benefits of our P4/INT solution.

V. Conclusion

In this paper, we proposed a novel network services management methodology to support Visual Cloud Computing (VCC) application scenarios that feature video streaming in e.g., search and rescue, aerial surveillance. Our experiment results in a realistic network testbed in FABRIC infrastructure with regular and P4 switches show that our solution achieves increased throughput in terms of number of packets received per second. In addition to observing no considerable delays, the streaming video application was delivered using our solution with no packet loss due to the programmable switches redirection of traffic in alternative (non-shortest) routes using the MRI congestion information. Future work can identify different congestion scenarios and investigate relevant packet routing strategies.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Award Numbers: CNS-1950873, CNS-1647182 and OAC-2018074. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. Al-Ansi, A. M. Al-Ansi, A. Muthanna, I. A. Elgendy, and A. Koucheryavy, "Survey on intelligence edge computing in 6g: Characteristics, challenges, potential use cases, and market drivers," *Future Internet*, vol. 13, no. 5, p. 118, 2021.

[2] E. Vattapparamban, İ. Güvenç, A. İ. Yurekli, K. Akkaya, and S. Uluağaç, "Drones for smart cities: Issues in cybersecurity, privacy, and public safety," in *IEEE Int. Wireless Commun. and Mobile Comput. Conf.*, 2016, pp. 216–221.

[3] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, "A survey on data plane programming with p4: Fundamentals, advances, and applied research," *arXiv preprint arXiv:2101.10632*, 2021.

[4] L. Tan, W. Su, W. Zhang, J. Lv, Z. Zhang, J. Miao, X. Liu, and N. Li, "In-band network telemetry: A survey," *Computer Networks*, vol. 186, p. 107763, 2021.

[5] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[6] D. Scano, F. Paolucci, K. Kondepu, A. Sgambelluri, L. Valcarenghi, and F. Cugini, "Extending p4 in-band telemetry to user equipment for latency-and localization-aware," *Journal of Optical Communications and Networking*, vol. 13, no. 9, pp. D103–D114, 2021.

[7] F. Paolucci, F. Cugini, P. Castoldi, and T. Osiński, "Enhancing 5g sdn/nfv edge with p4 data plane programmability," *IEEE Network*, vol. 35, no. 3, pp. 154–160, 2021.

[8] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.

[9] D. Ding, M. Savi, F. Pederzolli, and D. Siracusa, "Design and development of network monitoring strategies in p4-enabled programmable switches," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–6.

[10] L. Castanheira, R. Parizotto, and A. E. Schaeffer-Filho, "Flowstalker: Comprehensive traffic flow monitoring on the data plane using p4," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[11] H. Mostafaei and S. Afridi, "P4flow: Monitoring traffic flows with programmable networks," *IEEE Communications Letters*, vol. 25, no. 11, pp. 3546–3550, 2021.

[12] R. Hark, M. Ghanmi, R. Kundel, P. Lieser, and R. Steinmetz, "Monitoring flows with per-application granularity using programmable data planes," in *2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2021, pp. 1–6.

[13] R. Gargees, B. Morago, R. Pelapur, D. Chemodanov, P. Calyam, Z. Oraibi, Y. Duan, G. Seetharaman, and K. Palaniappan, "Incident-supporting visual cloud computing utilizing software-defined networking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 1, pp. 182–197, 2016.

[14] J. Patman, D. Chemodanov, P. Calyam, K. Palaniappan, C. Sterle, and M. Boccia, "Predictive cyber foraging for visual cloud computing in large-scale iot systems," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2380–2395, 2020.

[15] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, p. 2048, 2019.

[16] J. Patman, D. Chemodanov, P. Calyam, K. Palaniappan, C. Sterle, and M. Boccia, "Predictive cyber foraging for visual cloud computing in large-scale iot systems," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2380–2395, 2020.

[17] G. A. Q. Marrogy, "Enhancing video streaming transmission in 5 ghz fanet drones parameters," *Telecommunications and Radio Engineering*, vol. 79, no. 11, 2020.

[18] F. Cugini, P. Gunning, F. Paolucci, P. Castoldi, and A. Lord, "P4 in-band telemetry (int) for latency-aware vnf in metro networks," in *Optical Fiber Communication Conference*. Optica Publishing Group, 2019, pp. M3Z–6.

[19] J. Hyun, N. Van Tu, and J. W.-K. Hong, "Towards knowledge-defined networking using in-band network telemetry," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–7.