INTERNET 2025

The Seventeenth International Conference on Evolving Internet

ISBN: 978-1-68558-234-0

March 9th –13th, 2025

Lisbon, Portugal

INTERNET 2025 Editors

Dirceu Cavendish, Kyushu Institute of Technology, Japan

INTERNET 2025

Forward

The Seventeenth International Conference on Evolving Internet (INTERNET 2025), held between March 9th, 2025, and March 13th, 2025, in Lisbon, Portugal, continued a series of international events addressing challenges raised by the evolving Internet, making use of the progress in different advanced mechanisms and theoretical foundations. The gap analysis aims at mechanisms and features concerning the Internet itself, as well as special applications for software defined radio networks, wireless networks, sensor networks, or Internet data streaming and mining.

Originally designed in the spirit of interchange between scientists, the Internet reached a status where large-scale technical limitations impose rethinking its fundamentals. This refers to design aspects (flexibility, scalability, etc.), technical aspects (networking, routing, traffic, address limitation, etc.), as well as economics (new business models, cost sharing, ownership, etc.). The evolving Internet poses architectural, design, and deployment challenges in terms of performance prediction, monitoring and control, admission control, extendibility, stability, resilience, delay-tolerance, and interworking with the existing infrastructures or with specialized networks.

We take the opportunity to warmly thank all the members of the INTERNET 2025 technical program committee, as well as all the reviewers. The creation of such a high-quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to INTERNET 2025. We truly believe that, thanks to all these efforts, the final conference program consisted of top-quality contributions. We also thank the members of the INTERNET 2025 organizing committee for their help in handling the logistics of this event.

We hope that INTERNET 2025 was a successful international forum for the exchange of ideas and results between academia and industry for the promotion of progress in the field of the evolving Internet.

INTERNET 2025 Chairs

INTERNET 2025 Steering Committee
Renwei (Richard) Li, Southeast University, China
Eugen Borcoci, National University of Science and Technology POLITEHNICA Bucharest, Romania
Terje Jensen, Telenor, Norway
Przemyslaw (Przemek) Pochec, University of New Brunswick, Canada
Parimala Thulasiraman, University of Manitoba – Winnipeg, Canada
Dirceu Cavendish, Kyushu Institute of Technology, Japan
Sonia Ben Rejeb, ISI/University El Manar, Tunisia

INTERNET 2025 Publicity Chairs
Francisco Javier Díaz Blasco, Universitat Politècnica de València, Spain
Ali Ahmad, Universitat Politècnica de València, Spain
José Miguel Jiménez, Universitat Politècnica de València, Spain
Sandra Viciano Tudela, Universitat Politècnica de València, Spain

INTERNET 2025
Committee

INTERNET 2025 Steering Committee

Renwei (Richard) Li, Southeast University, China
Eugen Borcoci, National University of Science and Technology POLITEHNICA Bucharest, Romania
Terje Jensen, Telenor, Norway
Przemyslaw (Przemek) Pochec, University of New Brunswick, Canada
Parimala Thulasiraman, University of Manitoba – Winnipeg, Canada
Dirceu Cavendish, Kyushu Institute of Technology, Japan
Sonia Ben Rejeb, ISI/University El Manar, Tunisia

INTERNET 2025 Publicity Chairs

Francisco Javier Díaz Blasco, Universitat Politècnica de València, Spain
Ali Ahmad, Universitat Politècnica de València, Spain
José Miguel Jiménez, Universitat Politècnica de València, Spain
Sandra Viciano Tudela, Universitat Politècnica de València, Spain

INTERNET 2025 Technical Program Committee

Majed Alowaidi, Majmaah University, Saudi Arabia
Mohammad Alsulami, University of Connecticut, USA
Mário Antunes, Polytechnic of Leiria & INESC-TEC, Portugal
Andrés Arcia-Moret, Xilinx, Cambridge, UK
Marcin Bajer, ABB Corporate Research Center Krakow, Poland
Michail J. Beliatis, Research Centre for Digital Business Development | Aarhus University, Denmark
Laura Belli, University of Parma, Italy
Sonia Ben Rejeb, ISI/University El Manar, Tunisia
Driss Benhaddou, University of Houston, USA
Nik Bessis, Edge Hill University, UK
Maumita Bhattacharya, Charles Sturt University, Australia
Filippo Bianchini, Studio Legale Bianchini, Perugia, Italy
Eugen Borcoci, National University of Science and Technology POLITEHNICA Bucharest, Romania
Fernando Boronat Seguí, Universitat Politecnica de Valencia-Campus De Gandia, Spain
Christos Bouras, University of Patras, Greece
Lianjie Cao**,** Hewlett Packard Labs, USA
Dirceu Cavendish, Kyushu Institute of Technology, Japan
Hao Che, University of Texas at Arlington, USA
Albert M. K. Cheng, University of Houston, USA
Hongmei Chi, Florida A&M University, USA
Andrzej Chydzinski, Silesian University of Technology, Poland
Franco Cicirelli, ICAR-CNR, Italy
Victor Cionca, Munster Technical University, Ireland
Fábio M. Costa, Institute of Informatics (INF) | Federal University of Goiás (UFG), Brazil
Vittorio Curri, Politecnico di Torino, Italy

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Design Concepts to Satisfy User Data Accessabily of IoT Devices Postulated by the EU Data Act

Felix Fischer, Paul Seidel, Dirk Labudde
Faculty Applied Computer Sciences and Biosciences
Hochschule Mittweida University of Applied Sciences
Mittweida, Germany
e-mail: {fische11 | seidel6 | labudde}@hs-mittweida.de

***Abstract*—The Data Act of the European Union (EU-2023/2854) came into effect on the 11th of January, 2024. By September 2025, the 20-month grace period for adoption will end, after which full compliance with the regulation will be necessary. Implementation depends on multiple factors, including the infrastructure in place currently. Accordingly, a typical operation of a smart home device is chosen as the basis for this paper. Sharing the data with the user is of particular interest to the user in this scenario. We describe different implementations designed to satisfy the requirements established by the EU Data Act. In our view, there exist four distinct design solutions to address those requirements. Access can be granted via the data-generating device, a user-hosted server, a smart home hub or the existing cloud. Design proposals are discussed to explain the benefits and disadvantages of each solution. We arrive at the conclusion that expanding existing cloud Application Program Interfaces (APIs) would be the preferred method for most Internet of Things (IoT) devices. However, sharing data via a local controller could also be an effective solution.**

Keywords-eu data act; data availability; iot; smart home; design.

I. Introduction

Under the General Data Protection Regulation (GDPR) framework, users can currently request access to their data, usually provided within a three-month time frame following the request [1]. However, the upcoming European Union (EU) Data Act will impose stricter requirements on data accessibility [2]. Specifically, the EU Data Act mandates that manufacturers ensure that users have direct access to product and related service data. Manufacturers have to fulfill the requirements before the 12th September, 2025. These data must be provided "by default, easily, securely, free of charge, in a comprehensive, structured, commonly used, machine-readable format" [2].

Wolfgang Kerber already looked at the text of the law from a legislative perspective. He came to the conclusion that the EU Data Act does not achieve its goals. "(a) empowering the users of IoT devices (especially the consumers), (b) unlocking large amounts of IoT data for innovation (for IoT-related services and across sectors), and (c) contributing to a fair sharing of the value from the generated IoT data" are mentioned as objectives. All are missed in his view [3].

In this paper, the technical implementation design is mainly considered. Consequently, the scenario of consideration of this paper will be defined in Section II. To fulfill these new requirements, various technical approaches are being considered. These approaches will be judged by selected criteria, which are discussed in Section III. Three primary methods for granting users continuous access to their data have been identified. The first approach is pulling data from the data-generating device, allowing users to extract data directly from the origin. The technical advantages and limitations of this approach are discussed in Section IV. The second method involves enabling users to set up a dedicated server for data collection, allowing the device to transmit data both to a cloud-storage service and to the user-managed server. The feasibility, advantages, and drawbacks of this approach are analyzed in Section V. Extending the previous method, a local server could be provided by other existing smart home devices which then become a hub for all connected devices. This approach is presented in Section VI. The last option that comes to mind provides user access to data via the existing cloud infrastructure, potentially through an Application Programming Interface (API). This method is detailed in Section VII. Finally, this paper concludes with a discussion on the most suitable design choice in Section VIII, followed by recommendations for future research in Section IX.

II. Scenario under consideration

The term Internet of Things (IoT) device covers a wide range of devices. The considerations in this paper are limited to the classification of IoT devices in the home sector. The term smart home is often used for those. Despite this focus on a small sub-area, it should be possible to cover many other areas affected by the EU Data Act. Therefore, the term IoT device continues to be used here.

For this paper, a typical network setup of an IoT device is assumed. This is shown in Figure 1. The end device connects to a multifunctional device (WiFi Access Point + Switch + Router + Modem), which is colloquially referred to as a "WiFi Router". This multifunctional device forwards the recorded data to a server of the respective manufacturer. There, data are collected and prepared for the user, but also for the manufacturer itself. The data flow is unidirectional towards the infrastructure of the manufacturer, shown with blue arrows in Figure 1. The user can then typically access a visual representation of the data of their own devices via a web interface or a smartphone app. An access of the raw values collected by the IoT device is not mediatory at the moment.

Depending on the area of application however, the user does not receive a complete data set. Such visual representations

 ISBN: 978-1-68558-234-0

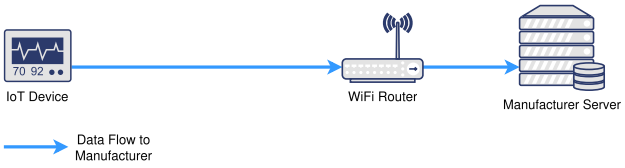


Figure 1. Typical data flow of IoT-devices at home.

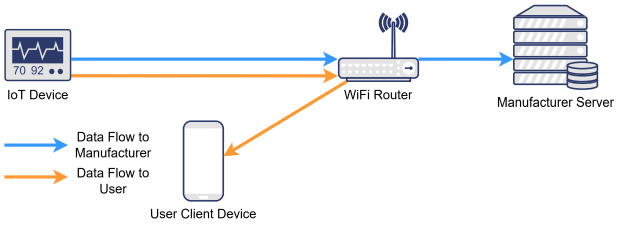


Figure 2. User pulls data from device directly.

of data are rarely machine-readable without some form of preprocessing, a fact the EU Data Act aims to change. Furthermore, the manufacturer can generate broad knowledge by combining data from different users. In Figure 1 and the following figures, only data streams that are necessary for data provision under the EU Data Act are illustrated.

III. Implementation Considerations

The solutions presented here offer various advantages as well as disadvantages. In order to be able to evaluate the different approaches, the most important factors to be considered are included.

First, a change in the way of working can require a change on the IoT device itself. Consequently, the IoT device must be able to perform a firmware update. Especially for small devices such as wireless contact sensors for doors and windows or motion sensors, the capability to perform a firmware update is not provided and, therefore, impossible. Thus, a replacement of the hardware would be necessary. This could potentially result in a massive one-time payment.

However, there may also be additional costs due to other factors. This includes, among other things, the implementation costs for changing the software. These costs for programming new functionality come into play both when changing firmware and when adapting running software in the cloud. Especially because the data must be provided free of charge, the cost factor can not be neglected. Additional costs reduce profits. Special attention should, therefore, be given to minimizing operating costs. This also includes costs for the provision of network traffic. In addition to computing power, the data stream itself has to be paid for. In the case of one-time paid devices, the running costs for data provision could otherwise wipe out any gained profit.

Finally, the provision of new interfaces increases the attack surface. By choosing a smart design, the attack surface can be reduced. Therefore, possible hacking attacks can be mitigated in advance.

In the analysis carried out here, the respective factors are presented from the point of view of a manufacturer of that IoT device.

IV. Access via the Data Generating Device

The intuitive solution to provide the data according to the requirements listed above is to provide the user with an interface on the data-collecting device itself. As can be seen in Figure 2, the user can collect the data at their own will using this new interface (orange arrows). The current data flow to the manufacturer's server remains untouched (blue arrows). Consequently, the server software does not have to be modified. This means that there are no additional costs for the manufacturer due to the operation.

However, the device needs to be updated with a new firmware, which not all IoT devices support. This means that this option cannot be technically implemented for these devices.

Another positive aspect is that the user can be provided with an arbitrarily dense temporal resolution of the data. In addition, it can be positively emphasized that the data are immediately available to the user. However, most IoT devices lack sufficient storage capacity for a continued retention of the collected data. This means that the user has to query their data at an increased frequency.

IoT devices usually work according to the push principle. This means that they send data and are not actively waiting for a connection. This allows them to switch to a so-called deep sleep mode after sending the data, saving power. In this mode, the Central Processing Unit (CPU) switches to a very economical co-processor, which only waits for an expiring timer. As soon as the time elapses, the CPU is woken up again. This technique can massively reduce electricity consumption and, therefore, the operating cost of a device. However, if the IoT device will await for a connection from the user, it cannot switch to deep sleep mode. As a result, electricity consumption increases. Thus, this will render small battery-powered appliances unusable within a few days or even hours.

In the event that electricity consumption does not have a restrictive effect, this method can be used to provide the data cost-effectively. This is especially the case if a so-called smart home controller is used. Such a controller is mainly present when a transmission technology that differs from WiFi is implemented and the device cannot communicate directly with the WiFi router. Thus, the controller acts as an access point for the alternative protocol. Common alternative protocols include ZigBee, Digital Enhanced Cordless

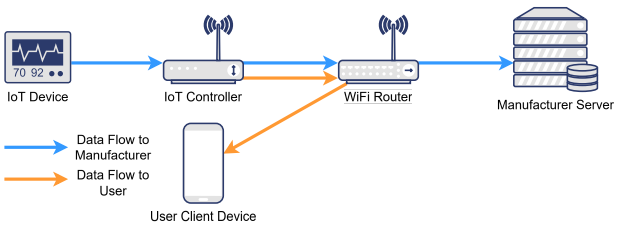


Figure 3. User pulls data from controller.

Telecommunications (DECT), Bluetooth Low Energy (BLE), and Matter. In such a case, the data could be cached on the controller. Figure 3 illustrates, how data can be accessed from there, according to the solution described above in this section. The orange arrows show the data collection using the controller cache. The blue arrows illustrate the unchanged data delivery to the manufacturer server via the WiFi Router. For this implementation, an update of the smart home controller is required.

V. Access via User Hosted Cloud

One way to change the method from Section IV to a push method is to have the IoT device send the data twice: First, as before, to the manufacturer's server (blue arrows) and second, to a local server in the user's network (orange arrows). Strictly speaking, the server operated by the user can also be connected to the Internet. From this second server, the user can access the data at any time and without any other restrictions (green arrows). Figure 4 illustrates this setup.

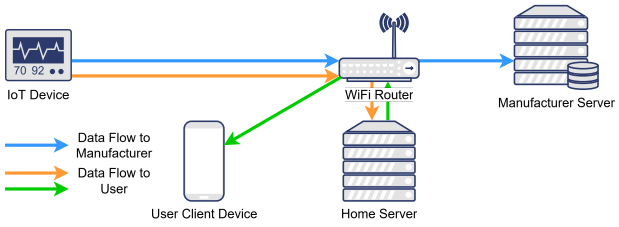


Figure 4. IoT device sends data twice and user accesses data via his local server.

By changing the access procedure from pull to push, the IoT device can switch back to deep sleep mode between transmission phases. Therefore, the energy consumption will have increased somewhat, but not drastically changed. Because transmission technologies, that are already in use are implemented, no hardware replacement is necessary. However, the firmware must be adapted in the sense that the IoT device sends the data twice and the user can enter to which server the data should be sent the second time. Thus, an update of the firmware will be necessary.

After the data have been transferred, the user has full control, but also full responsibility for their data. Especially from a data protection perspective, this fact is relieving for the manufacturer. Likewise, there are no further costs for the manufacturer's cloud operation. The management of the data, for its part, remains untouched.

It can be assumed that users expect that server software must be provided by the manufacturer for such a scenario. This must be additionally programmed and maintained, thus causing additional costs. However, these are likely to be quite limited, as the software is based on the already existing server software of the manufacturer's server.

The user has to operate their own server for this approach. It is unclear how this will be assessed as an obstacle to compliance with the EU Data Act by a judge. Making the data accessible in this way could be seen as a contradiction to making it available free of charge. Thus, there is a possibility that this approach will not be classified as meeting the requirements of the EU Data Act.

VI. Access via Smart Home Hub

Building on the approach in the previous Section V, existing hardware can be used as a local server. Some examples include Network Attached Storage (NAS) systems or smart home control solutions such as Alexa that are already in households [4]. The previous approach could be integrated into these systems. However, this results in the same advantages and disadvantages, except that no new hardware has to be employed.

VII. Access via Existing Cloud

The last approach builds on currently existing infrastructure. In this way, the data are already transferred to the manufacturer's server. Accordingly, it makes sense to set up direct access to the data for the user. As a result, an additional API will be provided or the existing one will be expanded. Figure 5 visualizes this implementation variant. The IoT device transmits data in the way currently used, illustrated by blue arrows. Afterwards, the user accesses the data from the manufacturers server, shown with orange arrows.

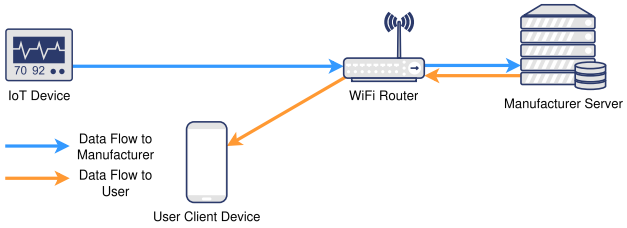


Figure 5. User pulls data from server of manufacturer.

If the manufacturer currently does not store its data on its own server, this approach offers a clear disadvantage. The data is stored at the manufacturer's expense. However, the data are of no use to him. The manufacturer's server would, therefore, act as the user's free cloud storage. However, the authors are not aware of any smart home device where this described case is observed. Manufacturers always have access to the data.

Expanding existing cloud access offers the clear advantage that no changes or updates of the Smart Home Network or the IoT devices therein are necessary. No firmware update or hardware replacement is required. Since there is no change to the existing hardware, there is no additional energy consumption.

In addition, an extension of server access to user data can be implemented comparatively quickly. Supplementary changes can be continuously integrated in the future. However, this method of implementation generates additional costs in computing power and network traffic on the server.

Access to the data for third parties could be realized via the same interface. For example, the user could grant third parties access to his data by releasing an API token. On the other hand, the manufacturer could also share the collected data with third parties via the same API.

Finally, it can be positively emphasized that this form of transition to the fulfillment of the EU Data Act is not perceptible to the user.

VIII. Discussion

Since every approach except the one in Section VII requires an update of the firmware, it should first be clarified whether a firmware update is technically feasible. Without the possibility of adapting the software on the IoT devices or the associated controller, the data provision can only be realized via an API on the manufacturer's server. In particular, if the additional cost that can be expected from making the data available via an API from the cloud is negligible, this implementation should be chosen. It represents the smallest change compared to the status quo.

If the positive factors from Section VII outweigh the negative factors and it is technically feasible, we recommend the implementation from Section IV. This is especially true if a controller is already in use, as shown in Figure 3. Here, it should be checked whether all user-related data are already provided. If this is the case, no changes are necessary to comply with the EU Data Act. Otherwise, the EU Data Act can be implemented cost-effectively by updating and caching the data on the controller with this procedure.

There are valid reasons for the other implementation options. However, it should be considered whether the two options mentioned above are not more suitable for one's own starting position. In particular, due to the uncertain legal situation described in Section V.

IX. Conclusion and Future Work

In summary, it can be said that there are different approaches to making data available to the user according to the EU Data Act. These different approaches allow a solution to be cleverly adapted to the current situation.

This work can be followed by demonstrations of implementation proposals of the designs shown here. For example, the approach from Section VII alone could be implemented in many ways. Depending on which framework is used on the server, the corresponding implementation changes.

For the design of various solutions in this paper, the scenario described in Section II was limited. Thus, the designs presented here do not basically cover all scenarios. In particular, we consider connected cars and cloud services to be worthwhile, advanced fields of research for a submission regarding the EU Data Act.

In this study, only implementations for private devices specifically in smart homes were discussed. In the business-to-business sector, there are different starting points and different requirements are placed on the finished product. This allows for other perspectives on possible solutions.

X. Funding

This paper was funded by the European Union and the Free State of Saxony (Germany).

Kofinanziert von der Europäischen Union

Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des vom Sächsischen Landtag beschlossenen Haushaltes.

References

[1] Council of European Union, *Council regulation (EU) no 2016/679*, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679, last access: 01.07.2025, 2016.

[2] Council of European Union, *Council regulation (EU) no 2023/2854*, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L_202302854, Article 3, last access: 03.05.2025, 2023.

[3] W. Kerber, "Governance of IoT Data: Why the EU Data Act Will not Fulfill Its Objectives", *GRUR International*, vol. 72, no. 2, pp. 120–135, Oct. 2022, ISSN: 2632-8550. DOI: 10.1093/grurint/ikac107. eprint: https://academic.oup.com/grurint/article-pdf/72/2/120/49174428/ikac107.pdf.

[4] S. Jain, "Amazon alexa enabled smart wi-fi switch", *Journal of Multi Disciplinary Engineering Technologies*, vol. 13, no. 1, Jul. 2019. eprint: http://jmdet.com/wp-content/uploads/2019/09/JMDET_13_1_15-NEW.pdf.

Beyond Best Effort: Routing with Requirements

Bradley R. Smith
Department of Computer Science & Engineering
University of California Santa Cruz
Santa Cruz, California, USA
e-mail: brad@soe.ucsc.edu

Paul S. Tatarsky
Tatarsky.com
Washington, D.C., USA
e-mail: paul@tatarsky.com

***Abstract*—Originally designed for the exchange of best effort traffic (email, web, etc.), the Internet had the modest requirements of best-effort service and global reachability. The resulting architecture provides robust and scalable networking, however it is insecure, does not support the performance and policy requirements of modern applications, and makes inefficient use of network resources. While mechanisms have been developed that attempt to address these limitations (firewalls, Policy-Routing, Traffic Engineering with Multi-Protocol Label Switching, Segment Routing, etc.), they are expensive (requiring additional devices and expensive expertise), complicated to configure, and fragile in the context of a changing network. We have developed a new routing architecture based on flow requirements that enhances the Internet to forward traffic based on the requirements of each network flow. We accomplish this by computing a *best set of paths* that provides the full range of performance and policy available in a network, and forwarding flows over the least congested of the subset of these paths that satisfies their requirements. The resulting architecture ensures traffic is forwarded over paths that provide the performance, security, and resource control required by applications, users, and network administrators for each flow, while optimizing use of network resources. We have developed a prototype and submitted it to an independent testing lab that has verified the functionality and quantified the increase in performance in their testbed network (6x capacity increase).**

Keywords-Network Routing; Quality-of-Service; Traffic Engineering; Routing Requirements.

I. INTRODUCTION

The Internet is based on a best-effort communication model where "the network makes no specific commitments about transfer characteristics, such as speed, delays, jitter, or loss. It is assumed that end-system software, both transport layer protocols and applications, would (and must) take this unpredictability into account" [1]. Combined with reliable delivery provided by the Transmission Control Protocol (TCP) transport protocol, best-effort services provide flow-rate fairness, which is defined by the goal of equal flow rates for different flows over the same path. Flow rate fairness is an appropriate goal for best effort traffic (file transfer, email, web, etc.) [2]. As a result, the best-effort service model was a good match for the best effort traffic that the Internet was originally designed to carry. "The best-effort paradigm was very powerful - it meant that a wide range of communication technologies could be incorporated into the Internet, technologies with a wide range of basic characteristics. One factor that made the Internet protocols a success was that they could work over 'anything'" [1].

In addition, the Internet adopted a model of universal connectivity. "The original design of the Internet has been described as *transparent*: what goes in comes out. The net does not observe, filter, or transform the data it carries; it is oblivious to the content of packets. This transparency may have been the single most important factor in the success of the Internet, because transparency makes it possible to deploy a new application without having to change the core of the network. On the other hand, transparency also facilitates the delivery of security attacks, viruses, and other unwelcome data." [1]. For the original environment where the network was small and there was a high degree of trust and shared context among the users, the power of universal connectivity outweighed its risk.

In the Internet, packet forwarding is implemented on a hop-by-hop basis where forwarding tables are computed independently at each router, and the forwarding decision is done on a per-packet basis. Paraphrasing [3], packets in a flow traverse a set of interconnected networks (an internet) by, at each hop, forwarding the packet to the next hop router on the path to the packet's destination, where the next hop router *is derived from the packet's destination*. This derivation of the next hop router was initially based on the single best path in terms of a distance metric, and Internet forwarding state was composed of a single entry for each destination in the Internet giving the next-hop router on the best path to the destination. As a result, only one path is supported to any given destination, and that path is computed to optimize a single metric.

The use of single-path routing significantly compromises the ability of a network to meet the ordered Quality-of-Service (QoS) and categorical Traffic Engineering (TE) requirements of diverse applications. Single-path routing has a similarly detrimental effect on the utilization of network resources. As the load in a network increases, sending all traffic between a given source and destination over a single path tends to result in links on that path becoming congested.

The hop-by-hop style of packet forwarding used in the Internet exacerbates this problem. With destination-based forwarding each router forwards packets by matching each packet's destination address with a single entry in the router's forwarding table. This leads to the constraint that all traffic forwarded through an intermediate router to a destination must follow the same path used by traffic sent from that router to the destination. This aggressive tendency to concentrate traffic on a subset of a network's topology causes traffic to

experience congestion while usable network resources are left idle, resulting in poor utilization of network resources.

This shortest-path model has been expanded to support Equal-Cost Multi-Path (ECMP) forwarding state composed of the *set of paths* with the same (shortest) distance metric. However, ECMP is not widely utilized, and the result is still limited to the single best path *cost* to a destination. ECMP does not address the QoS or TE requirements of a flow, and only partially addresses the poor utilization of network resources.

However, as the Internet has transitioned to the role of global communication infrastructure, with paying users of more diverse and demanding applications managing increasingly sensitive information, there is a growing need to provide QoS, trust and TE control of network resources as a basic part of the architecture.

These new requirements come from the growth of two new traffic classes called real time and policy-constrained (our term). *Real time* traffic has ordered, time-based constraints for its delivery (delay, jitter, etc.). Examples include voice, video, telemetry (e.g., computer gaming) and real time trading. *Policy-constrained* traffic has categorical constraints for its delivery. Examples include disclosure requirements (sensitive traffic must be carried on eavesdrop-resistant network infrastructure [4]), jurisdictional constraints (restrict genomics data to networks operated in a specific jurisdiction), multi-tenant networks (a network environment shared by multiple customers), and zero-trust environments where the network is considered untrustworthy, traffic is encrypted and strict access control enforced on what traffic can be shared between which endpoints.

In the early 2000's, the Defense Advanced Research Projects Agency (DARPA) funded the "Future Generation Internet Architecture" project (aka NewArch) to answer the question "if we could now design the Internet from scratch, knowing what we know today, how would we make the basic design decisions?" [1]. The project addressed many issues with the Internet architecture and made many intriguing recommendations. Of particular interest to this paper were two recommendations; one to transition from the Internet's traditional best-effort delivery model to a model they called *trust-modulated transparency*, and another to adopt a generalized version of the routing concept of *regions* as a first-class object in the architecture.

Trust-modulated transparency generalizes the best-effort concept to empower the network to "offer a range of behavior when two (or more) nodes communicate, based on the declared wishes of those nodes. If all the endpoints request, the flow of data among them should be as transparent and unconstrained as the Internet of today. But either end should be able to require that the packets being received be checked, filtered, or constrained in ways that limit the risk of damage and limit the range of unexpected behavior."

Our solution combines the two concepts into a unified mechanism for resource allocation in the form of a routing architecture based on computing paths subject to requirements defined by users, applications, and network administrators. Combining trust-modulated transparency with regions makes it possible to address the problems of scaling and heterogeneity in a wide range of domains including trust, and the articulation, administration and enforcement of resource allocation policies involving QoS and other policy-constraints. The ultimate goal being to tame the challenges of scale and heterogeneity to maximize trust, user empowerment, and the effective use of network resources.

The spirit of this trust-enhanced region abstraction is not to replace the best-effort model, but to augment it. The resulting Internet will still "work over anything," however it will also allow applications to exploit special functionality when it is available on some paths, thereby ensuring the best experience, in terms of trust, QoS, and policy compliance that is possible in a network.

Recent work [5], [6] has explored the related issue of routing with partial orders. Both explore distributed routing protocols for what we have called here routing over the *best set of paths*. These two works have focused on implementing this approach in Bellman-Ford routing protocols (where paths are computed from destination back to source; see solution to "Problem B" in [7]).

This paper is organized as follows. Section II reviews current solutions that have been developed and deployed in an attempt to address the problems discussed above. Section III provides a concise overview of our requirements-based routing approach and presents a series of scenarios that illustrate the approach. Scenarios encompass Quality of Service (QoS) management, traffic engineering for multitenant networks, zero-trust networking, the utilization of Boolean variables to reflect network state evaluated at runtime, and finally, the programmatic control of Boolean variables by external systems. Section IV outlines the challenges and opportunities we have identified for this architecture. Section V presents the outcomes obtained from an independent testing laboratory evaluation of a prototype of this model that we implemented. Section VI concludes by summarizing the results and drawing conclusions.

II. Current Solutions

As described in the Introduction, the Internet's best effort communications model is limited in its ability to satisfy the QoS and TE requirements of modern network applications. A number of solutions have been developed to address these limitations under the rubric of *Traffic Engineering*.

Fundamentally, TE is the ability to route traffic over paths that differ from the lowest cost paths used by best-effort routing [4]. TE mechanisms were originally developed primarily to manage network bandwidth with the goal of minimizing congestion [8]. Since their introduction, these mechanisms have been generalized to address a broader set of requirements, such as meeting QoS requirements (specifically bandwidth and delay), restricting specific classes of traffic to topological regions of a network (i.e. multi-tenant capabilities), enforcing flow priorities (in the sense of preemption), and meeting

administrative goals (e.g., restricting sensitive traffic to paths composed of eavesdrop-resistant media such as fiber).

This section reviews the two generations of TE technology developed to date: MPLS-TE and Segment Routing.

A. MPLS TE

The first comprehensive solution for these issues was called *MPLS TE* (Traffic Engineering with Multi-Protocol Label Switching). On its own, MPLS provides the capability to forward traffic over multiple paths, including paths that are different from the lowest cost paths used by the default best-effort routing, as required for traffic engineering. Using MPLS TE, real-time and policy-constrained traffic can be forwarded over paths that better meet their requirements and, sometimes as a specific goal and sometimes as a side-effect, distribute traffic more broadly over a network, resulting in a reduction in congestion and more efficient use of network resource.

MPLS TE accomplishes this by including additional link attributes in the routing computation, using an enhanced routing algorithm called *Constrained Shortest Path First* (CSPF) [9], and using MPLS forwarding state to forward traffic over diverse paths. In addition to the cost used in best-effort routing, MPLS TE includes additional link information such as a TE metric (distinct from the standard link cost), bandwidth, and administrative "*color*" attributes [4], [10].

For QoS requirements, CSPF computes a single path that minimizes a specified, additive metric (the traditional cost metric and an additional *TE metric*, which enables "engineering" the routing computation). For policy requirements, CSPF assigns "colors" to links and interfaces in the network. The set of colors is represented by a 32 bit color bitmap. Each color represents some attribute of a link; e.g., encryption, jurisdiction, maintenance status, link media (optical, copper, wireless), service-level agreement (Gold, Silver, Bronze), etc. Given a set of constraints (expressed in terms of link colors to be included and excluded), a traditional SPF routing algorithm is run on the subset of the topology that satisfies the constraints using the specified QoS metric.

CSPF is limited in a number of ways. Limiting QoS support to one *least cost* path is painfully restrictive. For example, the requirements for video streaming (high bandwidth and high delay) and network-based telephony (low bandwidth and low delay) are almost in conflict (a high bandwidth, low delay path would satisfy both, but at a premium price when their individual needs are not that demanding).

Similarly, the color-based abstraction for TE requirements of a network flow is limiting. The number of attributes used for defining a policy is limited to the 32 bits in the color bitmap. The attributes available for defining policies are all related to properties of links and interfaces on a path. Policies are statically defined as a part of the network configuration.

MPLS-TE implements point-to-point (P2P) forwarding state specific to each flow, resulting in very poor utilization of label-swap resources and poor scalability. Lastly, MPLS-TE implements on-demand route computation and path signaling, adding significant overhead to the forwarding process.

These limitations led to the development of the improved Segment Routing architecture.

B. Segment Routing

A more recent solution for the original Internet architecture's limitations involves a combination of network technologies based on Segment Routing (SR) [11]. SR computes and builds paths similar to MPLS-TE that better meet the QoS and TE needs of network applications. When TE is not required, SR is able to implement ECMP paths.

SR improves on MPLS-TE in a number of ways. SR integrates the label distribution, TE path signaling, and routing functions that are implemented separately in MPLS-TE into a single protocol. SR builds any-to-one, "multi-point to point" (MP2P) label-swap forwarding state. SR implements a forwarding model that still includes an on-demand routing computation, but makes use of pre-computed forwarding state. The resulting solution is dramatically simpler to configure and operate than MPLS-TE, much more efficient in its use of label-swap resources, and improves on the MPLS-TE forwarding process.

While SR improves on MPLS-TE in the ways listed above, it inherits some of MPLS-TE's limitations including only supporting least-cost paths, its use of the limited abstraction of colors for TE requirements, and it still requires a routing computation for each new flow.

III. Beyond Best Effort

As described in the Introduction, our requirements-based routing architecture implements the *trust-modulated transparency* and routing *region* capabilities identified by the DARPA NewArch project as needed to address the requirements of modern network applications. Specifically, requirements-based routing computes and forwards traffic over paths that satisfy requirements articulated by users, applications and network administrators for each flow carried in a network. As a result traffic carried in a given routing domain ("region") complies with the QoS and TE requirements defined for that domain. The result is an augmented best-effort architecture where the Internet protocols are still able to work over "anything," but now are able to exploit special functionality in the network when it is available, ensuring the best experience in terms of trust, QoS, and policy-compliance that is possible in a given region.

The rest of this section illustrates the mechanics and power of this approach with a number of scenarios. Each scenario is defined by a set of requirements for how traffic in a given class of flows is to be handled. As described in the Introduction, there are two types of requirements: QoS and TE.

QoS requirements of a network application address the *ordered*, performance requirements needed for an application to perform well, typically expressed in terms of bandwidth, latency, jitter (variation in latency), reliability, etc. *TE* requirements specify the *categorical*, non-performance related characteristics of network links such as security (e.g., encryption), jurisdictional issues (for example restricting private

TABLE I. VOICE/VIDEO QOS REQUIREMENTS

Flow Type	Perf Rqmts	
VoIP	$\leq$ 40ms	$\geq$ 100Kbps
Video Streaming	$\leq$ 10sec	$\geq$ 3Mbps

TABLE II. MULTI TENANT TE REQUIREMENTS

Flow Type	Boolean Variable	Path Expressions
Tenant A	`TA`	`TA`
Tenant B	`TB`	`TB`
		`(TA` **or** `TB)`
		`(TA` **and** `TB)`
		`True`
		`False`

health information to networks within the jurisdiction of a given country), network maintenance status, etc.

A. Quality of Service

As an example, consider a network being used by both an interactive voice application implementing an Internet-based telephony service (commonly called Voice over IP, or VoIP), and a video streaming service such as Netflix.

Interactive voice communication has relatively modest bandwidth requirements (100Kbps provides a high quality voice encoding) but fairly stringent delay requirements (interactive communications is awkward with delays much above 50ms). So, VoIP service requires low delay and can can live with relatively low bandwidth. In contrast, video streaming has very modest delay requirements, but relatively high bandwidth requirements (i.e. even many seconds delay in starting a video is tolerable as long as once it starts there is adequate bandwidth for it to smoothly run to completion). So a video streaming service require requires high bandwidth and can live with high delay. Table I shows these requirements.

Given these performance requirements defined in terms of delay and bandwidth, the routing computation collects topology information that includes QoS metrics for each link. It then runs a modified shortest-path first routing routing algorithm that computes the set of paths in the network that are not comparable to each other, and forwards traffic over one of these paths that satisfies the flow's performance requirements; in the event there are more than one it uses the least congested.

Using the video and voice example from above, the low and high bandwidth and delay paths can be see as *incomparable*. Specifically, low delay is better than high delay however high bandwidth is better than low. This incomparability can be restated as *it depends on the needs of the flow*, resulting in the opportunity to compute a *best set of routes* as those paths in the network where some application might prefer one path over the others. Further, with potentially a choice of satisfying paths, it is possible to distribute traffic more widely over a network, thereby reducing congestion and increasing utilization.

B. Multitenant

Multi-tenancy is when several network customers are sharing a set of network resources, such as when several different small business are using the same network resources to communicate within their offices in a building an to reach the Internet. Despite the fact that they share resources, these network customers are not aware of each other, and their data is typically kept separate.

To implement such a set of requirements we define a set of Boolean variables that reflect policy-relevant attributes of network traffic, the network itself, or of the network's environment. TE requirements are articulated as Boolean expressions composed of these variables, and are used in the routing computation to compute policy-compliant paths for the flow to use.

A subset of these expressions can be used to label links in the network to express the TE constraints each link imposes on traffic that traverses the link. Path expressions are constructed as a part of the routing computation (by *and*'ing together the link expressions), to express the constraints imposed on traffic that traverses the path. Expressions that are not assigned to links define what we will call *end-to-end* requirements that are used to define requirements of traffic in terms of its content, source, and destination. We will see examples of all of these in the following.

Table II shows the Boolean variables that could be defined to support two tenants, and some likely path expressions that would be used to control traffic on a multi-tenant network. The Boolean expressions extracted from a flow are used to determine if a flow can use a path by determining if the conjunction (and'ing) of the flow expression with the path's expression is *satisfiable* (meaning there is a truth assignment to the variables that results in a $True$ value for the combined expression).

The `True` and `False` path expressions indicate any or no flows may use a link, respectively (these expressions can be used for any path expression and are not included in the remaining scenarios). `TA` or `TB` represent traffic sent or received by tenant A or B (perhaps set based on a flow's source or destination address). `(TA` **or** `TB)` allows tenants A and B to share a link, and `(TA` **and** `TB)` indicates a link only for use for flows between tenant A and B.

C. Zero Trust

This scenario illustrates support for Zero Trust security applied to the traditional three layer web application architecture using TE requirements. The general Zero Trust architecture, based on the assumption that networks cannot be trusted, adopts a least privilege strategy by encrypting all traffic and strictly enforcing access control expressed as an access matrix specifying what combination of users, applications, and security zones can access other security zones. Security zones are logical containers for physical interfaces, VLANs, and IP address ranges (i.e. a region of the network) [12].

In the three layer web application architecture, applications are organized into three logical tiers: web, application, and

TABLE III. ZERO TRUST

Flow Types	Zones	End-to-End Requirements
$\texttt{WEB}_F$	$\texttt{USER}_Z$	($\texttt{WEB}_F$ **and** $\texttt{USER}_Z$ **and** $\texttt{WEB}_Z$)
$\texttt{APP}_F$	$\texttt{WEB}_Z$	($\texttt{APP}_F$ **and** $\texttt{WEB}_Z$ **and** $\texttt{APP}_Z$)
$\texttt{DB}_F$	$\texttt{APP}_Z$	($\texttt{DB}_F$ **and** $\texttt{APP}_Z$ **and** $\texttt{DB}_Z$)
	$\texttt{DB}_Z$	

TABLE IV. CONTROL BACKUPS OVER CORE

Flow Types	Time Periods	Path Requirements
BKP	NT	(**not** BKP **or** (NT **and** BKP))

data. The web (or presentation) tier is the user interface to the application, responsible for collecting data from the user and displaying data from the application to the user. The application (or logic) tier is where data collected from the user is processed, sometimes using information from the data tier, and results are presented to the user or saved in the data tier. The database tier is where information produced by the application is stored and managed. The benefits of this architecture include faster development, and improved scalability, reliability, and security. For security purposes, firewalls are commonly deployed between tiers.

Table III illustrates a three tier architecture implemented on a single subnet using TE requirements. Boolean variables are defined for flow types ($\texttt{WEB}_F$, $\texttt{APP}_F$, $\texttt{DB}_F$) and network zones ($\texttt{USER}_Z$, $\texttt{WEB}_Z$, and $\texttt{DB}_Z$). The zone variables could be set based on the IP prefix of servers in each zone, and TCP ports or application detection technology could be used for setting the flow variables. In this scenario the links have no TE requirements, but end-to-end TE requirements limit traffic between zones to the appropriate classes of flows (e.g., $\texttt{WEB}_F$ traffic is only allowed between the $\texttt{USER}_Z$ and $\texttt{WEB}_Z$ zones, etc.). Note that, with this solution, the integrity of the three tier architecture does not depend on the location of servers. Servers from different tiers could be connected to the same layer 2 switch and the integrity of the tiers would still be maintained.

The two previous scenarios represent static TE requirements in the sense that how a Boolean variables is set is specified as part of configuring TE requirements for the network. So zones in the Zero Trust scenario could be defined by an IP prefix, etc. The remaining two scenarios illustrate an important capability of Boolean expression-based configurations to dynamically define the value of variables based on attributes of the network's state or environment.

D. Dynamic Variables

Table IV illustrates a simple scenario where backup traffic is only allowed to flow over a core portion of the network at night. The idea being that during the day the core portions of an organization's network are reserved for operational data and backups are only allowed to traverse peripheral networks, or be delayed to run at night.

TABLE V. BOOLEAN SATISFIABILITY AND ONEHOT()

DY	NT	BKP	Path Req	OH(DY,NT)	Result
$False$	$False$	$False$	$True$	$False$	$False$
$False$	$False$	$True$	$False$	$False$	$False$
$False$	$True$	$False$	$True$	$True$	$True$
$False$	$True$	$True$	$True$	$True$	$True$
$True$	$False$	$False$	$True$	$True$	$True$
$True$	$False$	$True$	$False$	$True$	$False$
$True$	$True$	$False$	$True$	$False$	$False$
$True$	$True$	$True$	$True$	$False$	$False$

Two Boolean variables are defined including BKP, which is set to true for flows that carry backup traffic, and NT, which is set to true when it is currently nighttime. The link expression (**not** BKP **or** (NT **and** BKP)) is defined for all core network links specifying that BKP traffic can only traverse core links at night.

The Boolean variable NT is a *dynamic* variable whose value is determined by the network at the time the flow is processed. While the time period to define as night would be configured statically as part of the network configuration, the value of the variable is determined dynamically. This capability introduces a bit of autonomic control into the network configuration, and leads to the more general solution presented next. The primary limitation to the dynamic nature of Boolean variables like NT is they only support state directly available to the network device implementing the routing function (a router, switch, or controller).

There are some subtleties to satisfiability that need explanation. We illustrate this by adding a variable DY that is $True$ for a flow occurring during they day (added for illustration since DY can be expressed as (**not** NT)). The first four columns of Table V show the truth table for the path expression (**not** BKP **or** (NT **and** BKP)) given these three variables. This shows that a flow sent during the day, with DY set to True and NT not set (i.e. in a "don't care" state), would be allowed because the path requirements would be satisfied in the last two rows, which is a mistake. This mistake comes from the fact that we have not expressed the requirement that a flow can only occur either *during the day or night but not both*, which is why the last two rows of the fourth column (where both DY and NT are $True$) show as $True$. To fix this we need a Boolean expression of the DY and NT variables that is $True$ only for truth assignments where only one variable is $True$. We represent such a function as $OH(variables...)$ (short for $OneHot(...)$) in the fifth column of Table V, and use it to complete the satisfiability test.

Applying this to our problem, the "Result" column shows the conjunction of the path requirements and OneHot(DY,NT) columns, where only rows three through six are valid, and show the desired truth table (the only blocked flows are backup flows not sent at night). So whenever we have a set of variables where only one can be $True$ for a given flow, we must include the $OneHot(...)$ function of those variables in the combined flow and path expression to avoid false positives. This is

TABLE VI. DEFCON WITH MULTILEVEL SECURITY

Flow Types	Threat Levels	Path Requirements
TS_f	D_1	((D_1 **and** (U_f **or** S_f **or** TS_f)) **or** (D_3 **and** (S_f **or** TS_f)))
S_f	D_3	((D_1 **and** (U_f **or** S_f)) **or** (D_3 **and** S_f))
U_f		(U_f)

assumed in the examples in the paper. Note, a similar set of constraints is needed for the Zero Trust scenario.

E. Programmatically-Controlled Variables

The final example illustrated in Table VI implements functionality that can demonstrate a fully dynamic Boolean variable. This scenario has two components, DEFCON threat levels and MultiLevel Security (MLS). MLS provides support for multi-tenant use of networks in the form of the traditional, military-style multilevel security using TE requirements. Traffic is classified at unsecured, secret, or top secret security levels and is routed over infrastructure certified at the traffic's level or above. The Boolean variables U_f, S_f, TS_f are defined for a flow's security level. An unspecified mechanism determines the security level for a new flow, and the flow is assigned to the least congested path that satisfies the MLS routing requirement (e.g., unclassified traffic can be forwarded over paths of any security level, but top secret traffic can only traverse strongly secured paths) as specified by the TE Boolean expressions assigned to each link.

DEFCON builds on MLS by adding Boolean variables (D_1 and D_3) reflecting the military *defense readiness condition* (DEFCON) levels used to characterize the current threat level. Higher threat levels are indicated by lower DEFCON numbers (DEFCON1 being the highest threat level). In this scenario the MLS link expressions have been modified to integrate D_1 and D_3 threat levels. In the modified expressions, D_3 enables TE requirements equivalent to the MLS scenario (flows at a given sensitivity level are allowed to traverse links at that same level or above), but D_1 enables TE requirements that drop unclassified (U_f) traffic from links rated at S_f and TS_f levels. The logic being that, in a time of heightened threat, secured network resources should be reserved for important traffic.

The dynamic nature of this scenario comes from the ability to implement programmatic control of the DEFCON variables. In our prototype, implemented as a Software-Defined Network (SDN) controller with a web user interface, we implemented programmatic control as a Representational State Transfer (REST) service for setting the values of Boolean variables, which support the remote invocation of functions on the Web server using HTTPS messages. Using such programmatic control mechanisms, Boolean variables can be defined to reflect any state in the network or its environment that has policy significance for the network's configuration. With such variables, the network's configuration can be changed immediately, without the need for reconfiguration of network devices or reprogramming of SDN-based systems.

This capability has profound implications for network management. Imagine a scenario where Boolean variables are defined to reflect workstation configuration acquired using network access control technology (e.g., operating system version and patch levels) combined with variables defined to represent information from threat feeds reflecting the severity of vulnerabilities discovered in operating system versions and patch levels. TE requirements could be defined that only allowed systems to access sensitive parts of a network if they are at patch levels with no known vulnerabilities and traffic from vulnerable systems can be routed to sites that facilitate upgrades of vulnerable systems), with new vulnerabilities being integrated into network behavior as soon as they are discovered.

IV. CHALLENGES AND OPPORTUNITIES

A fundamental challenge of requirements-based routing is the need to determine the *satisfiability* of Boolean expressions used to express categorical requirements [13]. Satisfiability, which is the test of whether there is a truth assignment of the variables in a Boolean expression that cause the expression to evaluate to *True*, is the prototypical NP-Complete problem [14]. The essential meaning of this is there is no known way to determine satisfiability "efficiently".

One possible approach to containing the cost of the satisfiability test is to restrict the syntax of these expressions to forms with efficient algorithms for satisfiability. Significant work has been done along this line, culminating in Schaefer's Dichotomy theorem [15]. Schaefer's theorem comprehensively defines the boundary between expressions for which satisfiability can be determined efficiently and those for which no efficient solutions are known. The theorem shows that efficient solutions exist for six classes of expressions, and any expressions not in these classes are NP-complete.

Unfortunately for the work here, Schaefer also showed that none of these classes support negation, which is required for routing with requirements. However, fortunately, driven by the needs of integrated circuit design testing, there has been dramatic progress in the optimization of satisfiability algorithms such that, in spite of the inherent challenges of the general problem (e.g., current algorithms can determine satisfiability of expressions with millions of variables and clauses in minutes [16]).

These results, and the likely size and characteristics of requirements-based routing problems, give hope that the cost of satisfiability will not be a problem. Experience with our (un-tuned and research-grade) prototype, where path selection based on Boolean requirements are made once per flow, is that the time required for these decisions is consistent with normal switching speeds (single-digit milliseconds). Additionally, we have not implemented the use of "assumptions" [17], which should significantly speed up determining satisfiability in the path selection process.

TCP TEST RESULTS			RSTP IAT (sec) 3	
DNSR IAT (sec)	DNSR Gbps	RSTP Gbps	Throughput Gain	Load Factor
0.25	3.25	1.70	-4.4%	12.0
0.5	3.78	2.32	**11.2%**	**6.0**
1	3.87	3.09	13.8%	3.0
1.25	3.76	3.15	10.6%	2.4
1.5	3.79	3.29	11.5%	2.0
2.5	3.79	3.39	11.5%	1.2
3	3.77	3.40	10.9%	1.0

Figure 1. TCP performance results

UDP TEST RESULTS			RSTP IAT (sec) 1.5	
DNSR IAT (sec)	DNSR loss rate	RSTP loss rate	Relative Loss	Load Factor
0.25	24.9%	42.3%	**1.03**	**6.0**
0.5	15.0%	39.1%	0.62	3.0
1	9.3%	31.8%	0.39	1.5
1.25	8.7%	27.9%	0.36	1.2
1.5	9.1%	24.1%	0.38	1.0
2.5	2.7%	14.8%		
3	1.9%	11.2%		
	DNSR Goodput (Gbps)	RSTP Goodput (Gbps)	Goodput Gain	Load Factor
0.25	2.40	1.84	**-0.8%**	**6.0**
0.5	2.71	1.95	12.0%	3.0
1	2.87	2.18	18.6%	1.5
1.25	2.91	2.30	20.2%	1.2
1.5	2.90	2.42	19.8%	1.0
2.5	3.10	2.71		
3	3.13	2.83		

Figure 2. UDP performance results

At a more engineering-level, there are a number of other challenges/opportunities that need to be addressed. Architectures for forwarding traffic over multiple paths to the same destination (currently include OpenFlow [18], P4 [19], and MPLS [4])) are in constant flux. Assessing the scalability and performance of solutions requires attention, and possibly impacts the architecture for a comprehensive solution.

Regarding opportunities, developing and assessing distributed implementations of this technology, along the lines of traditional routing protocols, needs to be evaluated as an approach to addressing scalability and performance issues. As mentioned earlier, recent work along these lines [5], [6] has explored related approaches to routing using distributed Bellman-Ford routing protocols.

V. Prototype

To validate this architecture we developed a prototype that implements policy-based (Layer 2) switching in a (SDN) environment using the OpenFlow protocol, the Ryu open-source controller, and Linux-based Open vSwitch software switches. The prototype includes a web interface that allows users to define the supported traffic classes for a network and the TE and QoS requirements for these classes.

Implementation in Layer 2 was done for both convenience and functionality. A centralized, controller-based implementation made configuration significantly easier by centralizing the definition and implementation of policy in one place. Additionally, implementation of the requirements-based routing model at Layer 2 provides fine-grained control of network traffic down to the switch port level, enabling the full power of this architecture to be displayed. However, with some loss of granularity (working at the subnet vs swithing level), this architecture can support a Layer 3 implementation equally well.

We engaged an independent, third-party test lab to evaluate the prototype in terms of functionality and performance. Focusing on the performance evaluation, they deployed the system as a 4x4 torus, with two hosts per switch, in a VMware-based virtual environment. Each test involved 10 traffic flows for each host between random nodes in the graph with restrictions on the distribution of hops traversed (2 flows traversed 1 hop, 3 flows 2 hops, 4 flows 3 hops, and 1 flow 4 hops). Tests were run for a range of flow Inter-Arrival Times (IATs) between hosts (0.25, 0.5, 1, 1.25, 1.5, 2.5, and 3 seconds). TCP performance was characterized by the cumulative throughput of all 320 flows, and UDP by the average loss rate and cumulative good-put of the flows. The relevant results are presented in Figures 1 and 2. For TCP, requirements-based routing at 0.5sec IAT provides 11.2% better throughput at six times the load of Rapid Spanning Tree Protocol) RSTP at 3sec IAT. For UDP, requirements-based routing at 0.25sec IAT provides roughly the same loss rate and good-put at six times the load of RSTP at 1.5sec IAT.

VI. Conclusion

We have given an overview of requirements-based routing and presented a number of scenarios that demonstrate the power of this paradigm. Explicitly stating QoS and TE requirements enhances network routing to compute a *best set of routes* that satisfy the full range of QoS and TE requirements supported by a given network environment.

Articulating and enforcing the QoS and TE requirements enhances the Internet's original default-allow security model to default-deny where only requirement-compliant flows are allowed. Security is further enhanced by a dramatic reduction in the network's attack surface as it is limited to network devices whose access is typically tightly controlled (compared to the attack surface of all connected devices).

Use of requirements-based routing optimizes the user's experience, ensuring that traffic is forwarded over paths customized to the application's QoS and TE requirements and is compliant with network administration's policies. By working with a *set* of candidate paths, traffic can be forward over the least congested requirement-compliant path, dramatically improving network utilization. Simulations predicted a ten-fold increase with a somewhat "meshy" (average node degree

of four) network topology [20]; these results have been verified by an independent testing lab using an un-tuned *prototype* implementation.

Network services can be safely reconfigured with programmatic control of TE Boolean variables as they do not require reconfiguration of network equipment or re-programming of software-defined networking functions. Many functions currently implemented by expensive devices external to the core network, such as firewalls, load balancers and zero-trust network equipment, can be replaced by a software upgrade. Furthermore, implementing these functions using requirements-based routing results in significantly more robust services as they are implemented in the network layer where they have knowledge of the network's topology as it evolves.

Most importantly for many environments, requirements-based routing provides a more intuitive, high-level network configuration paradigm based on specifying *what* the requirements of the network are, allowing the network to solve the problem of *how* to enforce the requirements rather than depending on highly trained network engineers. This enables support of significantly more sophisticated network services by available engineers.

REFERENCES

[1] D. Clark *et al.*, *New Arch: Future Generation Internet Architecture*, Dec. 2003.

[2] S. Floyd and M. Allman, *RFC 5290: Comments on the usefulness of simple best-effort traffic*, Request For Comments, 2008.

[3] V. Cerf and R. E. Kahn, "A Protocol for Packet Network Intercommunication," *Communications, IEEE Transactions on*, vol. 22, no. 5, pp. 637–648, Jan. 1974.

[4] A. Sanchez-Monge and K. G. Szarkowicz, *MPLS in the SDN Era*. Sebastopol, CA: O'Reilly Media, Dec. 2015.

[5] J. J. Garcia-Luna-Aceves, B. R. Smith, and J. T. Samson, "QoS routing using dominant-distance vectors," in *Proceeding IEEE/ACM International Symposium on Quality of Service (IWQoS 2022)*, Jun. 2022.

[6] J. L. Sobrinho and M. A. Ferreira, "From non-optimal routing protocols to routing on multiple optimality criteria," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 294–307, Feb. 2023.

[7] L. R. Ford, "Network flow theory," RAND, Tech. Rep. P-923, Aug. 1956.

[8] D. O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, *RFC 2702: Requirements for traffic engineering over mpls*, Request For Comments, Sep. 2008.

[9] I. Minei and J. Lucek, *MPLS-Enables Applications: Emerging Developments and New Technologies*. Wiley, Jun. 2010.

[10] D. Katz, D. M. Yeung, and K. Kompella, *Traffic Engineering (TE) Extensions to OSPF Version 2*, Request For Comments, Sep. 2003.

[11] S. F. Hassan, A. Orel, and K. Islam, *A Network Architect's Guide to 5G*, M. Taub, N. Davis, S. Schroeder, S. Schroeder, and B. Reed, Eds. Addison-Wesley Professional, Jun. 2022.

[12] J. Kindervag, *No more chewy centers: The zero trust model of information security*, Forrester Research Technical Report, Mar. 2016.

[13] B. R. Smith, "Efficient Policy-Based routing in the internet," Ph.D. dissertation, University of California, Santa Cruz, Aug. 2003.

[14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co., 1979.

[15] T. J. Schaefer, "The Complexity of Satisfiability Problems," in *10th ACM Symposium on the Theory of Computing*, 1978, pp. 216–226.

[16] S. Garfinkel, J. M. Abowd, and C. Martindale, "Understanding database reconstruction attacks on public data," en, *Commun. ACM*, vol. 62, no. 3, pp. 46–53, Feb. 2019.

[17] A. Nadel and V. Ryvchin, "Efficient SAT solving under assumptions," in *Theory and Applications of Satisfiability Testing – SAT 2012*, ser. Lecture notes in computer science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 242–255.

[18] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, Mar. 2008.

[19] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," en, *Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[20] B. R. Smith and L. Thurlow, "Practical multipath load balancing with QoS," in *Proceedings International Conference on Computing, Networking and Communications*, Dec. 2013, pp. 937–943.

NGAPI and NGMonitor: Bridging NovaGenesis and the Current Internet

William S. Mamede, Antônio M. Alberti, José M. C. Brito
Instituto Nacional de Telecomunicações
Santa Rita do Sapucaí - MG, Brazil
e-mails: william.silva@mtel.inatel.br, antonioalberti@gmail.com, brito@inatel.br

***Abstract*—As novel Internet structures have been proposed over the past decade, the task of monitoring data transmissions and providing interoperability among different architectures has become increasingly complex. Consequently, the development of backward compatible solutions which also facilitate network traffic analysis has become a pivotal element for enabling the visualization and interconnection of Future Internet (FI) data within the current WEB infrastructure. Moreover, Artificial Intelligence (AI) has emerged as an invaluable tool for enhancing data analysis and visualization, especially within complex environments. This paper introduces an innovative data visualization tool, specifically engineered to oversee data flows within the NovaGenesis (NG) architecture, by addressing the critical challenge of bridging the gap between the prospective capabilities of the FI and the existing Internet, which also enables the incorporation of generative AI to assess network-related content.**

***Keywords*-NovaGenesis, Future Internet, ICN, ID-based Networking, Browser, Interoperability, Overlay Networks, Artificial Intelligence, Data Analysis using AI.**

I. Introduction

The Internet, which is a fundamental pillar of contemporary society, is facing an unprecedented increase in the number of interconnected devices and data exchange. This exponential expansion, in addition to the user requirements and the emergence of novel technologies, has become a substantial challenge on the existing Internet architecture, developed on the host-centric Transmission Control Protocol/Internet Protocol (TCP/IP) paradigm. This architecture, developed several decades ago [1], contains significant limitations in effectively addressing modern challenges related to scalability, security, mobility, and flexibility [2], which presents an impediment to address the demands of modern applications and other Internet solutions.

In response to these constraints, the academic community is conducting studies aiming to demonstrate that the advances from Future Internet Architectures (FIAs) [3] provide a more resilient, secure and user-oriented digital environment. FIAs adopt content-centric paradigms, prioritizing the information itself over host addresses, and integrate cutting-edge technologies that exploit the capabilities of data-driven networking. However, the possibility of running existing applications in different Internet environments became a challenge, which, according to Zali *et al.* [4], could be mitigated by developing a backward compatible architecture, aiming to promote adherence to industry standards and protocols, ensuring compatibility and seamless communication between WEB applications. Such a solution could create a cross-platform infrastructure composed of multiple FIAs, and enhance content delivery, since the existence of an environment which allows interoperability among different networks will enable simultaneous use of various applications and protocols, resulting in smooth progression of the network. Also, as stated by Siddiqui and Mueller [5], this type of solution could expand the user base and services, as it would offer a flexible and adjustable approach to controlling network resources. Therefore, we concluded that interoperability provides the capability of having applications created with different architectures to be executed and distributed without relying on the Internet's structure.

NovaGenesis (NG) [6] represents a pioneering FIA, designed with the goal of transforming the Internet through the synthesis of principles of Information-Centric Networking (ICN), Software-Defined Networking (SDN), and Service-Oriented Design. Its architecture aims to provide a solution that provides features such as self-certifying identifiers, a publish/subscribe communication paradigm, and a distributed hash table mechanism, thereby enabling efficient content dissemination, service discovery, and autonomous network management.

With the goal of providing a tool to facilitate monitoring and analysis in a NG environment, we have designed and implemented NGMonitor, an innovative data visualization instrument that integrates the NG architecture with existing WEB technologies by furnishing a user-centric WEB-based dashboard for real-time visualization of network activity, and NGAPI, a module that provides a set of RESTful APIs that expose NG data and enables its interoperation with the current Internet. As a result, it facilitates the integration of pre-existing WEB solutions, including generative AI. Considering that, we developed a solution to consume Google Gemini API [7], aiming to perform network data analysis with the purpose of generating and present hierarchical visualization of network components, such as interrelations between domains, hosts, operating systems, processes, and transferred files, offering a comprehensive overview of network architecture and data exchange.

This paper presents the design, implementation and evaluation of NGAPI and NGMonitor, highlighting the practical applications of the introduced features and demonstrating how researchers and developers of NovaGenesis could benefit from this feature. This article is organized as follows. Section II reviews the related work as well as the methodologies adopted by other FIAs. Section III offers a summary of the architecture of the system that was developed. Section IV seeks to provide an in-depth overview of the NGAPI module, which facilitates IP communication within NovaGenesis. Section V presents NGMonitor, a WEB-based dashboard that presents valuable

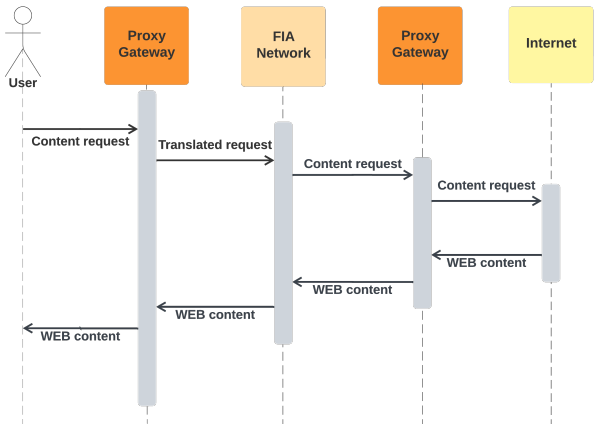


Figure 1. Tunneling approach for WEB interoperability used by FIA.

information from network operations. Section VI provides an analysis of the system's performance results from the experimental evaluation. Finally, Section VII concludes the article and suggests future research directions.

II. Related work

Numerous studies focus on proposing an innovative Internet architecture that serves as an alternative to the current TCP/IP network, originally developed several decades ago, in order to address present-day requirements, which, according to Conti *et al.* [8], include scalability, security, privacy, quality of service, interoperability and flexibility. Various solutions have been proposed to reach the desired goal, such as NovaGenesis [6], XIA [9], RINA [10], NDN [11], and MobilityFirst [12]. Notwithstanding, these approaches are well suited to address the limitations related to the network infrastructure and its design. However, there are other issues to be addressed, including the interoperability with already established WEB systems, which have been designed for the current Internet and often need to run over FIAs.

Our study has identified a potential approach to achieve interoperability, by adding an intermediate proxy between the existing Internet infrastructure and the novel network stack (post-IP) on the access side, as well as on the far-end side between the novel network and an IP network, which is presented in Figure 1. Imagine a legacy application that performs HTTP requests for a server through a TCP/IP socket. Following this tunneling approach, the application request and the server response will remain the same, but the FIA must be enhanced to accept the application request and then deliver the requested data from the responsible server. This could be reached by implementing a new layer for translating HTTP requests to FIA requests (subscriptions or other primitives) and vice versa. Thus, this kind of implementation can allow the network core to evolve and, in the same way, make it backward compatible with existing WEB-based applications. This concept was discussed in [5], where the authors proposed a socket to perform an API translation for the future Internet.

Following the approach presented before, a solution named Content-oriented interoperability framework for current and future Internet architectures (COIN) [13], for example, was implemented providing an interoperability tool, similar to that presented previously, in which a proxy was implemented responsible for translating HTTP requests for NDN [11] at the network entry point and vice versa. Therefore, this solution does not imply any change at the application and server level. A similar solution can also be found in Performance Enhancing Proxy for Deploying Network Architectures (PEP-DNA) [14], in which a proxy was used to translate messages from TCP to RINA [10], as well as reverse translation. The main difference between their approaches is that the proposed proxy has been implemented as part of the Linux operating system kernel [15].

Taking the previous information into account, we developed an intermediate layer (NGAPI) which aims to allow the interoperability of NovaGenesis with the current Internet, going beyond the approaches described before, since it enables the consumption of existing WEB applications deployed in a real-world scenario. Furthermore, our solution includes the addition of a WEB application (NGMonitor), which aims to monitor the traffic in the network, thereby introducing a unique characteristic that distinguishes it from other FIAs. In the subsequent sections, we will detail the architecture of the solution that has been implemented, along with use case examples that aim to demonstrate the practicality and effectiveness of the new feature introduced.

III. System Architecture: Modular and Extensible Design

NGAPI and NGMonitor were developed following the microservices pattern, to build independent components that, when deployed together, provide a complete end-to-end solution. This modular architecture offers several advantages, including:

- Scalability: The system can be easily scaled by adding or removing modules as needed, ensuring that in the future it is capable of increasing data volumes in complex network environments.
- Flexibility: Each module can be developed and maintained independently, which makes it simple to perform updates and enhancements without impacting other system parts.
- Generality: The microservices design allows for potential adaptation and extension to support other features, including solutions developed for other Internet architectures.

The architecture of the NGMonitor is composed of three main services (as presented in Figure 2):

1) **NG Network Core**: This is the main component of the solution, which contains all the business logic related to NovaGenesis [16], [17]. It operates following the publish/subscribe behavior purpose instead of consuming from the conventional TCP/IP protocol stack. The essential services that make the system operate are the Proxy/Gateway Controller Service (PGCS) and the Name Resolution and Network Caching System (NRNCS), which establish the infrastructure for data exchange. These services enable some of the NG functionalities keys:

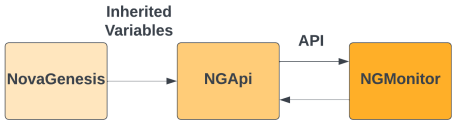


Figure 2. System architecture overview.

- **Name Resolution**: NRNCS offers a mechanism capable of retrieving previously published bindings, ensuring their direct delivery to authorized subscribers.
- **Content Distribution**: The combined operation of PGCS and NRNCS enables efficient publication and subscription of content, ensuring data distribution on the network.
- **Service Discovery**: PGCS is responsible for performing an orchestration in order to discovery of the Publish/Subscribe Service (PSS), Generic Indirection Resolution Service(GIRS), and Hash Table Servic (HTS) during the initialization phase of the system. In addition, it provides a proxy service that allows the representation of other NG services within an operating system.

2) **NGAPI: NovaGenesis Application Programming Interface**: This module aims to provide a set of APIs to enable communication between the NG network and applications developed under the current Internet. This is achieved by implementing a set of endpoints using restCppSdk [18]. Therefore, NGAPI makes it possible to interoperate NovaGenesis with the current Internet, enabling its communication using the HTTP protocol. This interoperability is imperative for NGMonitor, facilitating the consumption and expose of data from external APIs.
3) **NGMonitor: A WEB-Based Dashboard**: This module is an HTTP-based application, developed using the Angular framework [19], aiming to provide a dynamic and intuitive interface to enable visualization of NG data. This module consumes the data provided by NGAPI, processes, and then provides refined and comprehensive data to the end user, including the consumption of Gemini AI [7], which is provided by NGAPI, with the aim of analyzing a set of NG data and generating insightful reports. These reports, presented to the user within the dashboard, provide an understanding of network relationships and data flow patterns, providing insights into the NG network structure, which enhances the analytical capabilities of NGMonitor, empowering researchers and developers to gain deeper knowledge and insights from what the network is executing.

In order to understand how NGMonitor works, it is needed to make clear how the communication happens between the network elements. Firstly, NGMonitor periodically consumes the interfaces provided by NGAPI, obtaining consequently the most updated data related to services that are offered, files that are transmitted and received, which aims to ensure that dashboard presents a real-time overview of the network activities. In addition to that, when information related to the connection between network elements and its processes is requested, it consumes the NGAPI endpoint that enables

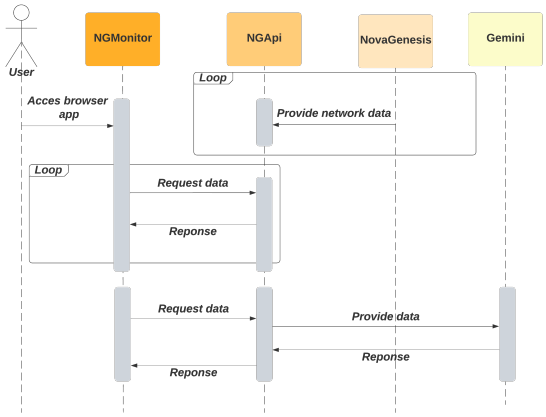


Figure 3. System diagram illustrating the integration between components.

communication with Gemini API, making it possible to perform an evaluation of network data and producing insightful reports for visualization on the dashboard. This synergistic integration of components is presented in Figure 3.

Therefore, NGAPI and NGMonitor emerge as a promising solution that aims to mitigate the lack of an interoperable data visualization and monitoring tool implemented over a FIA. Considering that, we can consider that the proposed solution has the following strengths:

- **Interoperability**
- **Real-Time Insights**
- **Insights Enhanced using AI**
- **User-friendly interface**

In the following sections, we will describe the introduced modules, NGAPI and NGMonitor, by providing a comprehensive analysis of the technologies used at this development, a detailed explanation regarding the implemented functionalities, and provide details about how the modules interact with each other.

IV. NGAPI: Interoperating and Exposing NovaGenesis Data through a RESTful Interface

This section presents how NGAPI was developed, which is the module responsible for providing NovaGenesis data to an application developed under an IP based network, by serving as an intermediary layer between the core of the NG network and the current Internet, enabling seamless communication and interoperability via the HTTP protocol. This enables the integration of a wide number of existing WEB applications with NovaGenesis and introduces the possibility of providing information from FIA usage for other architectures. Thus, the solutions we propose are designed to promote seamless data exchange, thereby enabling the creation of innovative applications that can fully utilize the potentials offered by the advanced architecture of next-generation networks. This is accomplished without necessitating bespoke integration efforts for each specific application intended to be targeted.

1) NGAPI Architecture and Implementation: NGAPI was engineered using the C++ programming language [20] and

incorporates the *cpprestsdk* library, which provides a set of tools that enables the development and consumption of HTTP-based endpoints [18]. This choice was guided by the native implementation of NovaGenesis, which is mostly developed in C++, making then it easy to access network variables and data in real time, which avoids the need of creation of additional layers.

2) Exposing Data through RESTful Endpoints: NGAPI, as presented at a high level in Figure 4, provides a set of RESTful endpoints designed to provide NovaGenesis data to external applications. Then, during the operation of the network, it is stored information related to its operation and the content that was transmitted, with the objective of delivering such data as a JSON response from an API request. This feature aims to facilitate the understanding of activities carried out in NovaGenesis. The endpoints provided include the following:

- serviceOffers: This endpoint retrieves and provides information about all services offered in a given NG domain, allowing researchers and developers to understand the services that are exposed and explore potential interactions or collaborations.
- transmittedMessages: This endpoint returns a list of all transmitted data files, providing details such as file names, timestamps, and the source application that sent the data. This endpoint enables monitoring of data flow patterns.
- receivedMessages: Similarly to the 'transmittedMessages' endpoint, this endpoint retrieves information about data files received by hosts, providing insight into the flow of data from different sources.
- transmittedImage: This endpoint allows retrieval of specific transferred images.
- getDataAnalysedByAI: This endpoint is responsible for collecting NG bindings [17] and offers these resources for analysis by Gemini AI. Bindings are key-value pairs elements stored in the HTS in a categorized way, which describe the relationships between several elements, including connections between physical and virtual entities, associations between services and processes, and links between physical devices. This data is sent to Gemini AI, which processes this relational information and generates insights into domains, hosts, operating systems, processes, and transferred files, detailing their complex connections and dependencies. The resulting analytical insights are then used by NGMonitor to build a hierarchical visualization of the network topology.

3) Interoperating with current Internet: Furthermore, NGAPI has the ability to act as an intermediate layer, allowing data to be exchanged within the network in a format compatible with standard Internet protocols, particularly HTTP. Therefore, in an NG environment, in which a host has access to the current Internet, it can act as a bridge, which allows to:

- Leverage Existing Applications: Make it possible to consume existing WEB applications, provided through the HTTP protocol in NovaGenesis. For instance, a network user might utilize any data supplied by an API (accessed through HTTP), within an NG terminal, without TCP/IP Internet connection, if any host along the network infrastructure has the NGAPI module connected to the Internet. Therefore, this solution is useful in a range of applications, including the Internet of Things (IoT). As detailed in the findings by Alberti *et al.* [16], there is a noticeable improvement in performance associated with Internet of Things (IoT) use cases, particularly when evaluating parameters such as data transfer, memory utilization, and CPU resource consumption. Therefore, the use of NGPAPI provides a significant advantage in terms of backward compatibility, facilitating effective communication among devices that operate within varied network architectures. This interoperability presents a critical benefit, as it allows the smooth integration of current IoT devices and applications into the NovaGenesis network, while maintaining uninterrupted access to previously established Internet-based services.
- Integrate with Cloud Services: Enable the consumption of popular cloud services and platforms in NovaGenesis, extending its capabilities, and providing access to a wider range of tools and resources.
- Share Data Across the Internet: Facilitate communication between hosts connected in different networks.

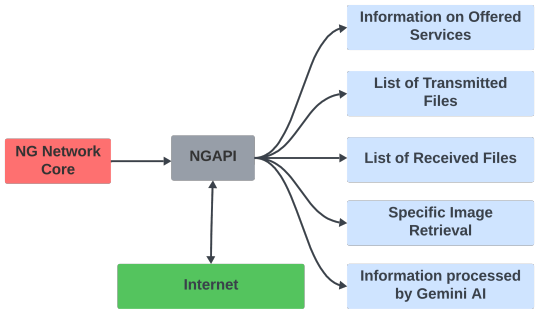


Figure 4. A simplified representation of the NGAPI module.

Figure 5 illustrates the steps taken when attempting to consume content present on the current Internet from a NovaGenesis node, also described as follows:

1) *Content request:* If a user would like to access some data provided via a HTTP API in the current Internet, such as the consumption of information provided by an IoT device or some WEB page, it is possible to consume that by performing a request for desired content through NGAPI, without requiring a direct Internet connection on their device, by providing the path of interest at the end of the request URL (e.g. http://ngapi/wantedContentPath).
2) *Process request:* After receiving the request, the desired path is converted using a hash algorithm and then checked if the information is already available on the network. This hash value represents a unique identifier for the data within the NovaGenesis network. The NGAPI then checks if the hash value exists in the network:
   - *Content Available:* If the resource exists, the NGAPI retrieves the content from the NovaGenesis network and returns it to the user.
   - *Content Not Available:* If the resource does not exist, the

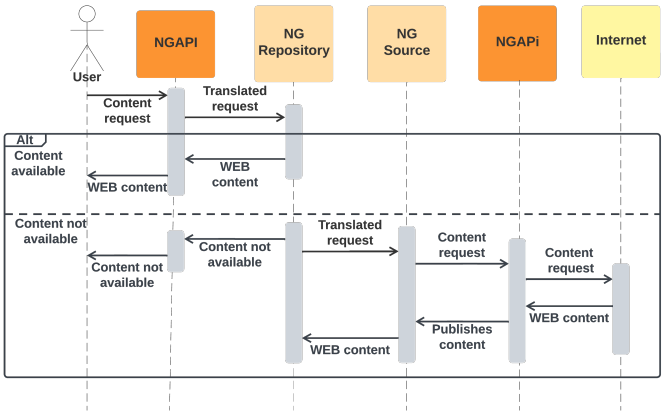


Figure 5. NGAPI sequence diagram.

NGAPI proceeds to the next step, initiating a request to the source of the content.

3) *Request to Source:* The node which received the request publishes a content request to the source node in the NG network, containing the desired URL.
4) *Source Fetching:* The source node, upon receiving the request, fetches the content from the Internet through the NGAPI external API.
5) *Source Publishing:* After receiving the content from the Internet, the source publishes the received content on NG networks, making it available to all consumers who seek to access it.

Hence, NGAPI's capability to expose and consume HTTP API facilitates the interaction between existing Internet applications and NovaGenesis, enhancing its accessibility and real-world solutions, allowing users and developers to take the advantages and befits provided by a content-oriented architecture in applications developed under the current Internet. Therefore, this solution elucidates the critical role of interoperability in the advancement of FIAs and enables a smooth evolution of the network, which subsequently allows the coexistence of IP and ICN networks, as outlined by Conti *et al.* [8]. Its ability to seamlessly integrate with existing Internet infrastructure, utilizing the prevailing technologies and tools, ensures a fluid transition toward a more interconnected and versatile digital ecosystem. By providing a comprehensible and accessible interface for external applications to interact with NovaGenesis, NGAPI provides researchers, developers, and network administrators with the tools necessary to fully investigate the potentialities inherent in FIAs.

V. NGMonitor - An Angular Dashboard

This section presents NGMonitor, the WEB-based dashboard developed with the aim of providing a tool that enables visualization and analysis of data extracted from NovaGenesis. This solution was developed using an Angular framework [19], and provides a robust and interactive user interface, allowing researchers and developers to receive insights related to the operation of the network. This solution, in contrast to existing FIAs, presents an innovative tool designed to provide user-friendly insights related to network behavior. Therefore, NovaGenesis network administrators and researchers will benefit from our solution, as it provides a simplified overview of the network, consequently leading to an improved resource management and an improved process of decision making.

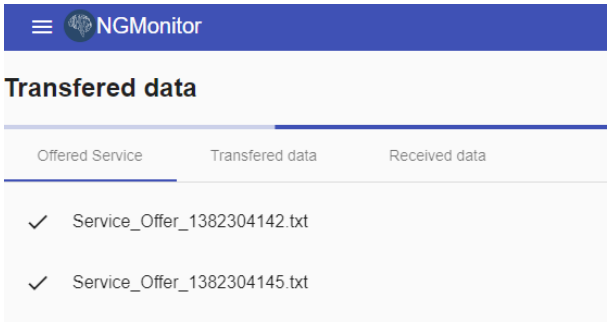


Figure 6. Assigned offered services.

1) Capabilities of NGMonitor: Visualization and analytical evaluation: NGMonitor contains a set of functionalities, designed and developed to enable users to perform monitoring and in-depth analysis of NovaGenesis, by providing:

- Real-time visualization: NGMonitor presents dynamic and interactive visualizations of NG data, updated in real time in order to provide an updated representation of the network activity. This enables users to observe and analyze the progression of network events and data transfers as they happen, thereby facilitating the comprehension of network behavior. This feature incorporates the following aspects:
  - Visualization of the Service Offer: The dashboard provides a list of available services, as presented in Figure 6, based on data retrieved from the 'serviceOffers' endpoint. This enables researchers and developers to efficiently identify the set of services available in the network.
  - Data Flow Monitoring: It enables the user to monitor the transmission of files, providing file names, timestamps, and the respective source applications, as demonstrated in Figure 7. This information is obtained by the endpoints 'transmittedMessages' and 'receivedMessages'.
  - Image Visualization: NGMonitor provides an image viewer for 'transmittedImage' data. After user interaction with an image entry within the 'transmittedMessages' or 'receivedMessages', the dashboard is able to retrieve and show transmitted images in an emergent pop-up, as presented in Figure 8.
- Hierarchical Visualization powered by AI: It provides integration with Google Gemini AI [7], a state-of-the-art AI model. This integration enables the analysis of NG data and consequently the generation of a hierarchical visualization of network components. This visualization presents the interrelationships between domains, hosts, operating systems (OSIDs), processes (PIDs), and files transferred within the NG network, as presented in Figure 9. This report can

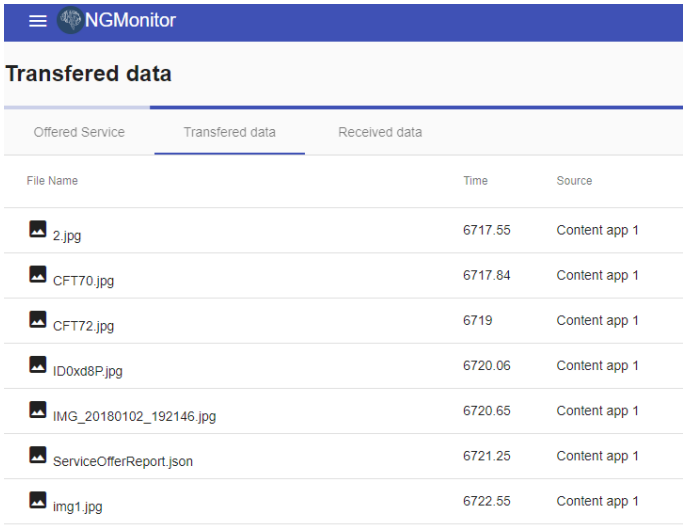


Figure 7. Transferred files from a NG source content distribution application.

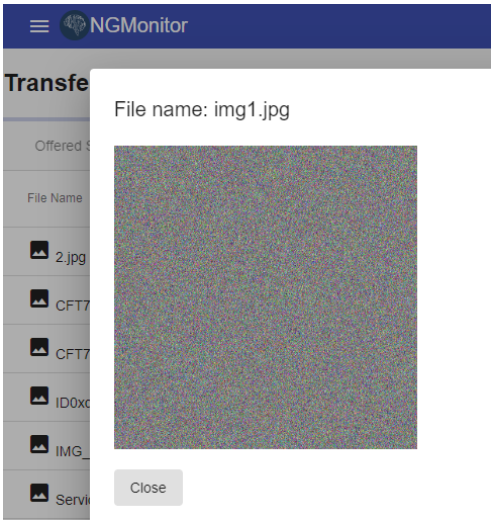


Figure 8. Visualization of a transmitted image.

provide comprehensive information related to the network's architecture, enabling users to identify potential bottlenecks or issues.

- User-friendly: NGMonitor boasts an intuitive user interface that prioritizes the user experience. It has been constructed using the Angular Material library [21], a decision that enhances its visual appeal and interactive features, making it more engaging and accessible for users.

VI. NGAPI Performance Metrics

This section presents an analysis of the NGAPI performance, which constitutes the cornerstone of the introduced monitoring enhancement, allowing seamless communication between NovaGenesis, NGMonitor and the current Internet. In order to analyze the responsiveness of the API, we perform extensive tests using JMeter [22], which is a widely recognized open source tool for load testing and performance evaluation. To do

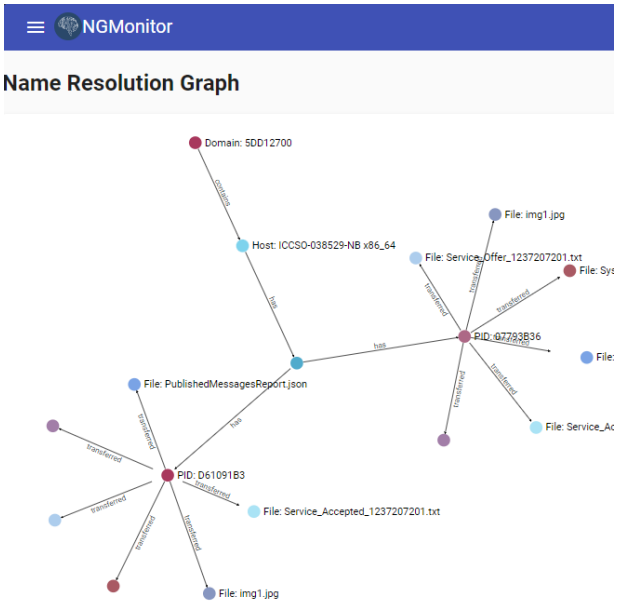


Figure 9. AI-Driven Visualization of Data Flow Patterns in NovaGenesis.

so, we develop a test case that tries to simulate a scenario in which multiple users generate a high number of requests in a certain time slot. This scenario is designed in order to stress the implemented APIs and the interoperability feature, then producing critical insights related to system performance.

To perform these analyses, we define the following essential metrics: Mean response time, which provides the typical duration required for the API to process a request and subsequently provide the requisite information to the front-end application; Minimum response time, representing the swiftest duration recorded during API interactions; Maximum response time, conversely, this metric indicates the most prolonged duration encountered, thereby reflecting the worst-case scenario.

The results of our analysis for the internal interfaces, which provides the network information data directly, are encapsulated in the graph presented in Figure 10, which shows the response times distribution for a variety of requests. In this representation, the x-axis denotes individual APIs, while the y-axis quantifies the average, minimum, and maximum response times, measured in milliseconds, corresponding to each API.

Moreover, we perform an experiment designed to assess the interoperability of APIs between NovaGenesis and the existing Internet. In order to achieve this, our experimental setup involves executing multiple requests via an external interface accessible over the current Internet, together with the execution of an identical request using NGAPI, which is aligned with the process illustrated in Figure 5. Therefore, every initial call to NGAPI will initiate the process to have the requested content available on the NG network, which subsequently triggers a request to the original API, followed by the release and availability of the data to the user. Consequently, a delay is anticipated before the content becomes accessible. Observations from our experimental execution indicate that the duration

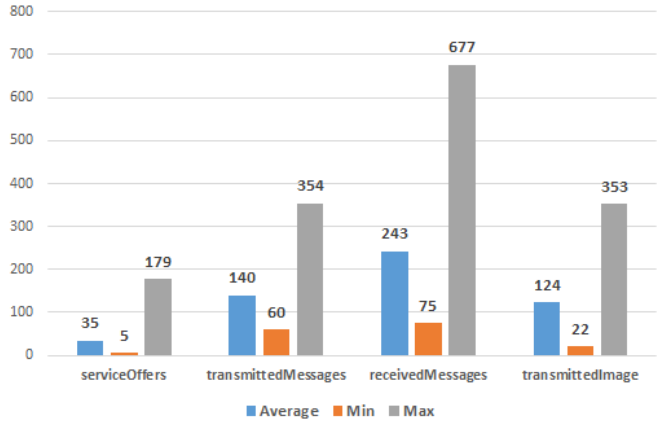


Figure 10. NGAPI Response Time Metrics.

necessary for the content to become accessible initially reached a maximum of 6 seconds. However, subsequent requests for identical content demonstrated markedly reduced response times, attributable to the NovaGenesis routing mechanism. This experiment underscored an important performance aspect: the first retrieval of information, necessitating publication, presents a slower pace compared to later requests, which make use of the content that is already available within the NovaGenesis network. The results presented in Table I show a significant difference in efficiency between accessing data from a standard Internet API and NGAPI, when comparing its response time in milliseconds.

TABLE I
RESPONSE COMPARISON IN MILLISECONDS BETWEEN INTERNET API AND NOVAGENESIS NGAPI

Requested from	Average	Median	Min	Max
Internet	686	645	637	1630
NovaGenesis	10	6	4	464

It is important to mention that the performance evaluation was not executed for interactions with the Google Gemini API. Since this data processing occurs outside NovaGenesis, the response time and performance related to this interface are defined by external factors, such as the availability of computational resources and the prevailing network conditions. Due to that, it was decided to omit this endpoint in the performance tests, since our analysis is based on the evaluation of the efficiency of NGAPI and obtain information within NovaGenesis.

VII. CONCLUSION

This paper introduces a solution composed of NGAPI and NGMonitor, which are pioneering tools designed for interoperation, visualization, and analysis of data within the NovaGenesis future Internet architecture. The proposed method offers an efficient approach to tackle the problem of incorporating FIA advances with existing Internet features and infrastructure, providing an intuitive platform for researchers, developers, and network administrators who seek to systematically monitor and interpret intricate data flows.

To accomplish this objective, a solution was proposed that takes advantage of a synthesis of key fundamental attributes, specifically enumerated as interoperability, real-time visualization, and AI-driven analysis. Therefore, the user benefits from our solution through NGAPI by enabling the communication for TCP/IP designed solutions over NovaGenesis, which means that any existing application could benefit from NovaGenesis strengths, also allowing a smooth evolution of the network, since it became possible to have backward compatible applications. Furthermore, NGMonitor provides a novel feature compared to other FIAs, in which a network administrator could monitor, analyze and receive insights from an external generative AI platform, allowing a proactive identification of network issues during data transfer and the generation of customized reports with the aim of facilitating a decision on network operations and resource management.

In summary, this paper effectively elucidates the potential of interoperability and AI-driven analysis to advance the research, development, and practical applications of FIAs. By providing an intuitive platform endowed with robust visualization and analytical capabilities, the combination of NGAPI and NGMonitor substantiates its significant contribution to the progression of FIA technologies. As the Internet continues to advance and merge with a wider range of devices and services, it became crucial to allow coexistence of IP-based applications with FIAs, therefore, instruments similar to the solution implemented by this study will become indispensable for the surveillance, comprehension, and governance of the intricacies inherent in the future Internet.

ACKNOWLEDGMENT

This work was funded by RNP, with resources from MCTIC, Grant No. 01245.020548/2021-07, under the Brazil 6G project of the *Centro de Referência em Radiocomunicações* (CRR) of the *Instituto Nacional de Telecomunicações* (Inatel), Brazil. The work has also been partially funded by *Fundação de Amparo à Pesquisa do Estado de Minas Gerais* (FAPEMIG RED-00194-23). This work was carried out with support from CNPq, National Council for Scientific and Technological Development - Brazil (303382/2021-0). It was also supported by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES) - Finance Code 001. The authors also thank CNPq, and Vivavox Telecom.

REFERENCES

[1] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks", *IEEE Communications Magazine*, vol. 50, no. 12, pp. 44–53, 2012. DOI: 10.1109/MCOM.2012.6384450.

[2] G. Xylomenos *et al.*, "A survey of information-centric networking research", *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014. DOI: 10.1109/SURV.2013.070813.00063.

[3] P. Mueller, "Towards the future internet: A review", in *From Multimedia Communications to the Future Internet: Essays Dedicated to Ralf Steinmetz on the Occasion of His Retirement*, S. Schulte and B. Koldehofe, Eds. Cham: Springer Nature Switzerland, 2024, pp. 74–89, ISBN: 978-3-031-71874-8. DOI: 10.1007/978-3-031-71874-8_6.

[4] Z. Zali, E. Aslanian, M. H. Manshaei, M. R. Hashemi, and T. Turletti, "Peer-assisted information-centric network (picn): A backward compatible solution", *IEEE Access*, vol. 5, pp. 25 005–25 020, 2017. DOI: 10.1109/ACCESS.2017.2762697.

[5] A. A. Siddiqui and P. Mueller, "A requirement-based socket api for a transition to future internet architectures", in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2012, pp. 340–345. DOI: 10.1109/IMIS.2012.117.

[6] A. M. Alberti and D. Singh, "Developing a novagenesis architecture model for service oriented future internet and iot: An advanced transportation system scenario", in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 359–364. DOI: 10.1109/WF-IoT.2014.6803188.

[7] *Gemini API Docs and Reference*, Online. Available: https://ai.google.dev/gemini-api/docs [retrieved September 2024].

[8] M. Conti, A. Gangwal, M. Hassan, C. Lal, and E. Losiouk, "The road ahead for networking: A survey on icn-ip coexistence solutions", *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2104–2129, 2020. DOI: 10.1109/COMST.2020.2994526.

[9] D. Naylor *et al.*, "Xia: Architecting a more trustworthy and evolvable internet", *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 50–57, Jul. 2014, ISSN: 0146-4833. DOI: 10.1145/2656877.2656885.

[10] J. D. Day, *Patterns in network architecture: A return to fundamentals*. Prentice Hall, 2010.

[11] L. Zhang *et al.*, "Named data networking", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[12] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future internet", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, pp. 2–13, Dec. 2012. DOI: 10.1145/2412096.2412098.

[13] M. Jahanian, J. Chen, and K. K. Ramakrishnan, "Managing the evolution to future internet architectures and seamless interoperation", in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020, pp. 1–11. DOI: 10.1109/ICCCN49398.2020.9209599.

[14] K. Ciko, M. Welzl, and P. Teymoori, "Pep-dna: A performance enhancing proxy for deploying network architectures", in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, 2021, pp. 1–6. DOI: 10.1109/ICNP52444.2021.9651953.

[15] *What is the Linux kernel?*, Online. Available: https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel [retrieved September 2024].

[16] A. M. Alberti *et al.*, "Advancing novagenesis architecture towards future internet of things", *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 215–229, 2019. DOI: 10.1109/JIOT.2017.2723953.

[17] A. M. Alberti, M. A. F. Casaroli, D. Singh, and R. da Rosa Righi, "Naming and name resolution in the future internet: Introducing the novagenesis approach", *Future Generation Computer Systems*, vol. 67, pp. 163–179, 2017, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2016.07.015.

[18] Microsoft Corporation, *C++ rest sdk*, Online. Available: https://microsoft.github.io/cpprestsdk/index.html [retrieved September 2024].

[19] N. Jain, A. Bhansali, and D. Mehta, "Angularjs: A modern mvc framework in javascript", *Journal of Global Research in Computer Science*, vol. 5, no. 12, pp. 17–23, 2014.

[20] B. Stroustrup, *The C++ Programming Language*, 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2013, ISBN: 978-0-321-56384-2.

[21] V. K. Kotaru, "Introduction to angular material", in *Material Design implementation with AngularJS: UI Component Framework*. Berkeley, CA: Apress, 2016, pp. 1–5, ISBN: 978-1-4842-2190-7. DOI: 10.1007/978-1-4842-2190-7_1.

[22] *Apache jmeter*, Online. Available: https://jmeter.apache.org/ [retrieved October 2024].

On the FullMesh Path Selection of Multipath TCP Video Streaming

Yosuke Komatsu*, Dirceu Cavendish**, Daiki Nobayashi**, Takeshi Ikenaga**
*Graduate School of Engineering, **Faculty of Engineering
Kyushu Institute of Technology
Fukuoka, Japan
e-mail: komatsu.yosuke620@mail.kyutech.jp, {nova@ecs, ike@ecs, cavendish@net.ecs}.kyutech.ac.jp

***Abstract*—Video streaming makes most of Internet traffic nowadays, with most video applications transported over Hypertext Transfer Protocol/Transmission Control Protocol (HTTP/TCP). Being the predominant transport protocol, TCP stack performance in transporting video streams plays an important role, especially with regard to MultiPath Transport Control Protocol (MPTCP) and multiple client device interfaces currently available. One overlooked aspect of multipath transport is the management of all possible paths between sender and receiver endpoints. In this paper, we study the usage of all possible paths created by MPTCP vis a vis video streaming performance on wired networking environments. We show that a fullmesh path usage may result in degraded video streaming performance due to common bottlenecks between paths under a simple (default) path scheduler. We then propose a bottleneck aware path scheduler, and show its superior performance on various multipath scenarios. Our results cover both Bottleneck Bandwidth and Round-trip (BBR) propagation time TCP variant, as well as CUBIC variant in transporting video streams over wired networks. We use network performance level, as well as video quality level metrics to characterize quality of video streaming over TCP variants.**

Keywords*—Video streaming; TCP congestion control; Multipath TCP; TCP BBR.

I. Introduction

Video streaming nowadays accounts for the majority of Internet traffic. Regarding streaming applications, video stream quality is related to two factors: the amount of data discarded at the client end point due to excessive transport delay/jitter and data rendering stalls due to lack of timely playout data. Transport delays and data starvation depend heavily on how Transport Control Protocol (TCP) handles retransmissions upon packet losses during flow and congestion control. Moreover, in multipath transport scenarios, it is important to manage head-of-line blocking across various networking paths, potentially with diverse loss and delay characteristics. Head-of-line blocking occurs when data already delivered at the receiver has to wait for additional packets that are blocked at another path, potentially causing incomplete or late frames to be discarded at the receiver, as well as stream rendering stalls. In addition, the various paths used in transporting the data may interfere with each other at times. In this paper, we study interference of multiple transport paths in a full mesh configuration, where all available networking paths are used for video transport. As transport delays and data starvation depend heavily on how TCP handles retransmissions upon packet losses during flow and congestion control, we analyze two TCP variants currently widely deployed: CUBIC [1] and BBR [2].

The paper is organized as follows. Related work is included in Section II. Section III describes video streaming transport over TCP, with focus to BBR and CUBIC TCP variants. Section IV describes the default path scheduler used in Linux environments, and introduces our new bottleneck aware path scheduler. Section V introduces these variants. Section VI characterizes video streaming performance over wired paths via network emulation. We compare the application and network performance of BBR against CUBIC, using the default (estimated shortest transmission time), as well as bottleneck aware path schedulers. Section VII summarizes our studies and addresses future directions to this work.

II. Related Work

Since MultiPath Transport Control Protocol (MPTCP) has become available, several multipath transport studies have appeared in the literature, mostly focusing on throughput performance of data transfers over mobile networks (see [3] and related work). [8] introduces path selection techniques, such as stickness (staying on a same path for as long as possible), in order to reduce head of line blocking in Video Streaming applications. Although the study shows improvements on wireless cellular/WiFi topology scenarios, it did not address interference among available paths.

[9] studies the performance of Adaptive Video Streaming (Dynamic Adaptive Streaming over HTTP (DASH)) on top of MPTCP transport. The adaptive bit rate nature of DASH causes large bit encoding fluctuations when paths of different throughput characteristics are present, causing bad video experience. The authors advocate blocking low throughput paths in order to stabilize adaptive bit rate and, improving Quality of Experience. The authors of [4] evaluate throughput of multipath video streaming over Digital Subscriber Line (DSL) multipath scenarios, without providing video level performance measures. Although they also propose a cost optimized scheduler, the lack of video quality performance measures limits conclusions about the impact of such a scheduler on video quality. Along the same lines, Imaduddin et al. [5] provide a performance evaluation of Multipath TCP (MPTCP) using CUBIC and Vegas TCP variants, as well as minimum Round Trip Time (RTT), round-robin and coupled Balia schedulers. Finally, Xing et al. [6] propose a new MPTCP scheduler which they show via network experiments to lower the number of out-of-order packets. The scheduler estimates receiver arrival times, and sends redundant packets to cope with estimation errors. Video streaming is simulated

 ISBN: 978-1-68558-234-0

via iperf3, and no application layer performance measures are used.

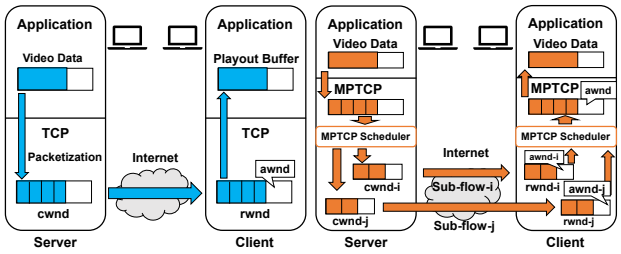
Regarding full mesh path selection, [10] studies the interference between paths sharing bottleneck resources, from a goodput performance perspective. They characterize inter-path positive and negative interference, and propose a "least interpath contention" path management strategy, where they limit the multipath transport to use only disjoint paths. They then use emulated single hop client/server testbed topology to evaluate full-mesh vs path disjoint goodput results vis a vis various couple and uncoupled congestion control schemes. In contrast, our work focuses on the intelligent path management taking into account common resources along fullmesh paths. Our path selection then strives to select paths in a timely manner so as not to cause interference even though paths may intersect.

III. Video Streaming Over MPTCP

A video application over Hypertext Transfer Protocol/Transmission Control Protocol (HTTP/TCP) starts at an HTTP server storing video content. At the transport layer, a TCP variant provides reliable transport of video data over IP packets between server and client end points (Figure 1 (a)). Upon HTTP video request, a TCP sender is instantiated to transmit packetized data to the client machine, connected to the application via a TCP socket. At the TCP transport layer, a congestion window is used at the sender to control the amount of data injected into the network. The size of the congestion window ($cwnd$) is adjusted dynamically, according to the level of congestion experienced on the network path, as well as space available for data storage ($awnd$) at the TCP client receiver buffer. Congestion window space at the sender is freed only when ACK packets acknowledging data packets are received. Lost packets are retransmitted by the TCP layer to ensure reliable data delivery. At the client, in addition to acknowledging arriving packets, the TCP receiver informs the TCP sender about its current receiver available space, so that $cwnd \leq awnd$ condition is enforced by the sender at all times to prevent receiver buffer overflow. At the client application layer, a video player extracts data from a playout buffer, which draws packets delivered by the TCP receiver from the receiver TCP socket buffer. The playout buffer hence serves to smooth out variable network throughput and delay. Multiple path transport brings communication reliability enhancements, as well as bandwidth increase. The challenge is video rendering degradation due to increase frame discards and buffer underflows originated from head of line blocking.

A. MPTCP

MPTCP is an Internet Engineering Task Force (IETF) extension of TCP transport layer protocol to support data transport over multiple concurrent TCP sessions when multiple interfaces are available [7]. The network multipath transmission of the transport session is hidden from the application layer by a legacy TCP socket exposed per application session. At the transport layer, however, MPTCP coordinates concurrent TCP



(a) TCP (b) MPTCP

Figure 1. Video Streaming over TCP/MPTCP.

sessions on various subflows (paths), each of which is itself unaware of the multipath nature of the application session. In order to accomplish multipath transport, a path scheduler connects the application socket with transport subflows, extracting packets from the application facing the MPTCP socket, selecting a subflow for transmission, and injecting packets into the selected subflow. The MPTCP transport architecture is depicted in Figure 1 (b).

The first and most used path scheduler, called default scheduler, selects the path with shortest RTT among paths with currently available congestion window space for new packets. Other path schedulers have appeared recently. These path schedulers can operate in two different modes: uncoupled, and coupled. In uncoupled mode, each subflow congestion window $cwnd$ is adjusted independently of other subflows. On the other hand, in coupled mode, the MPTCP scheduler couples the congestion control of the subflows, by adjusting the congestion window $cwnd_k$ of a subflow k according to the current state and parameters of all available subflows. Although many coupling mechanisms exist, we focus on the performance study of BBR [2] TCP variant over uncoupled schedulers in this work.

Regardless of the path scheduler used, IETF MPTCP protocol supports the advertisement of multiple IP interfaces available between two endpoints via specific TCP option signalling. IP interfaces may be of diverse nature (e.g., Wi-Fi, Long term evolution (LTE)). In addition, multipath transport requires an MPTCP stack at both endpoints for the establishment and usage of multiple paths. MPTCP signalling allows for a full mesh of connecting paths between two transport endpoints. That is, if there are N interfaces at the client and M interfaces at the server available, each N x M combination of interfaces constitutes a viable path.

IV. Path Schedulers

In this section, we use MPTCP default scheduler as a springboard to propose a novel bottleneck aware path scheduler.

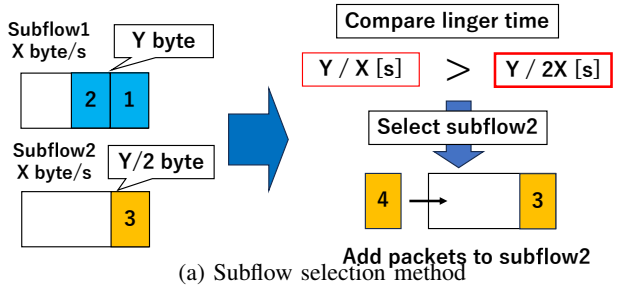
A. Default scheduler

An overview of the default scheduler algorithm is shown in Figure 2. The default scheduler (Linux kernel Version 6.1) selects the subflow that takes the least amount of time ($linger\_time$) to transmit all packets in the subflow buffer (Figure 3(a)). This time is calculated as:

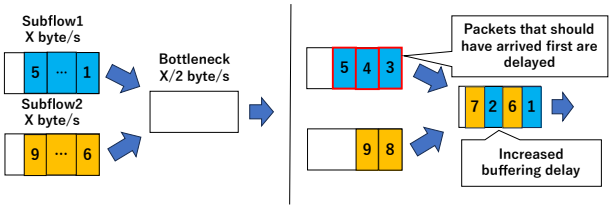
$$linger\_time = \frac{wmem}{pace} \quad (1)$$

Data: Set of subflows S
Result: Selected subflow S_{best}
Initialization:
  Set $best\_linger\_time \leftarrow 2^{32} - 1$, $S_{best} \leftarrow \emptyset$;
if *Last used subflow* S_{last} *is available* **then**
  return S_{last} ;
end
foreach *subflow* $S_n \in S$ **do**
  $linger\_time \leftarrow \frac{wmem}{pace}$;
  if $linger\_time < best\_linger\_time$ **then**
    $S_{best} \leftarrow S_n$;
    $best\_linger\_time \leftarrow linger\_time$;
  end
end
return S_{best} ;

Figure 2. Default scheduler Algorithm.



(a) Subflow selection method



(b) Common Bottleneck

Figure 3. Default scheduler behavior and common bottleneck.

where Pace rate ($pace$) is the available transfer volume per second, and write memory data ($wmem$) is the value of the queued data volume in the subflow send buffer. Because $pace$ is dynamically adjusted with Round Trip Time (RTT) and congestion control, it can prevent head-of-line blocking while maintaining high throughput at all subflows. However, the default scheduler has one deficiency in full-mesh connection. If there are multiple subflows sending packets to the same destination and there is a common bottleneck link on the route, packets can cause buffering delays at the bottleneck link (Figure 3(b)). TCP congestion control will then back off from injecting new traffic, causing subflows to the same destination to interfere with each other, resulting in a long time before a new packet is injected on interfering flows. This problem is most likely to occur in TCP variants' congestion control where many packets are continuously sent, such as CUBIC.

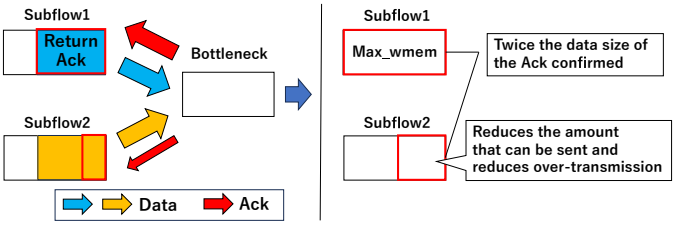


Figure 4. Buffer limitation method.

B. Bottleneck aware path scheduler

To solve the aforementioned problem, there is a need to limit the amount of packets injected on each subflow before too much queueing builds up at bottlenecks. A Bottleneck aware path scheduler is a scheduler with a limit on the amount of buffer used by each subflow, in addition to the short RTT mechanism of the default scheduler. The upper limit of the buffer used (max_wmem) is initially set to the amount of data that can be transmitted during 500 ms at the post-connection pace rate (ini_pace) (Eq:(2)). The reason for setting the initial maximum buffer amount to 500 ms of the pace rate is to ensure that all subflows do not slow down too much immediately after the start of the connection, as it is not known whether there are shared bottleneck links before transmission yet. Subsequently, if the smooth RTT ($sRTT$) is less than twice the minimum RTT (min_RTT), then twice the amount of data (snd_data) returned in ACK out of the data size transmitted between the previous and current subflow selection and the previous max_wmem are compared and the larger one is adopted (Eq:(3)). If $sRTT$ exceeds twice min_RTT and then returns to less than twice that value again, the smaller amount of data that can be transmitted per second with ini_pace or the previous max_wmem is adopted (Eq:(4)). By keeping max_wmem when $sRTT$ is more than twice min_RTT and updating it when $sRTT$ becomes less than twice again, the buffer size can be optimized while maintaining the throughput of each subflow.

- Initialization

$$max\_wmem = ini\_pace * 0.5\,[\text{byte}] \tag{2}$$

- If $sRTT < 2 * min\_RTT$

$$max\_wmem = max(max\_wmem, 2 * snd\_data)\,[\text{byte}] \tag{3}$$

- If $sRTT >= 2 * min\_RTT \rightarrow sRTT < 2 * min\_RTT$

$$max\_wmem = min(max\_wmem, ini\_pace * 1)\,[\text{byte}] \tag{4}$$

Iterating equations (3) and (4) during the transport session prevents excessive transmission, leading to a situation where too many packets accumulate in the subflow's buffer. This allows subflows with the same destination address to use paths that do not harm each other while satisfying the bandwidth of the bottleneck link (Figure 4).

V. CUBIC AND BBR TCP VARIANTS

The TCP protocol has evolved into different variants, implementing different congestion window adjustment schemes.

TCP protocol variants can be classified into delay and loss based congestion control schemes. Loss based TCP variants use packet loss as primary congestion indication signal, typically performing congestion window regulation as $cwnd_k = f(cwnd_{k-1})$, which is ACK reception paced. Most f functions follow an Additive Increase Multiplicative Decrease (AIMD) window adjustment scheme, with various increase and decrease parameters. In contrast, delay based TCP variants use queue delay information as the congestion indication signal, increasing/decreasing the window if the delay is small/large, respectively. Delay based congestion control does not suffer from packet loss undue window reduction due to random packet losses, as experienced in wireless links.

CUBIC TCP Congestion Avoidance: CUBIC TCP is a Loss-based TCP that has achieved widespread usage as the default TCP of the Linux operating system. During congestion avoidance, its congestion window is adjusted as follows (5):

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= C(t-K)^3 + Wmax \\ K &= (Wmax\frac{\beta}{C})^{1/3} \\ PktLoss: \quad cwnd_{k+1} &= \beta cwnd_k \\ Wmax &= cwnd_k \end{aligned} \tag{5}$$

where C is a scaling factor, Wmax is the cwnd value at time of packet loss detection, and t is the elapsed time since the last packet loss detection. The K parameter drives the CUBIC increase away from Wmax, whereas β tunes how quickly cwnd is reduced on packet loss. This adjustment strategy ensures that its $cwnd$ quickly recovers after a loss event.

BBR TCP Congestion Avoidance: BBR is a bandwidth delay product based TCP that has achieved widespread usage as one of available TCP variants in the Linux operating system. BBR uses measurements of a connection delivery rate and RTT to build a model that controls how fast data may be sent and the maximum amount of unacknowledged data in the pipe. Delivery rate is measured by keeping track of the number of acknowledged packets within a defined time frame. In addition, BBR uses a probing mechanism to determine the maximum delivery rate within multiple intervals. More specifically, BBR regulates the number of inflight packets to match the bandwidth delay product of the connection, or $BDP = BtlBw \times RTprop$, where $BtlBw$ is the bottleneck bandwidth of the connection, and $RTprop$ its propagation time, estimated as half of the connection RTT. These quantities are tracked during the lifetime of the connection, as per equations below (6):

$$\begin{aligned} rtt_t &= RTprop_t + \eta_t \\ \hat{RTprop} &= RTprop + min(\eta_t) \\ &= min(rtt_t) \forall t \in [T - W_R, T] \\ \hat{BtlBw} &= max(deliveryRate_t) \forall t \in [t - W_B, T] \end{aligned} \tag{6}$$

where η_t represents the noise of the queues along the path, W_R a running time window, of tens of seconds, and W_B a larger time window, of tens of RTTs. This adjustment strategy seeks to tune its $cwnd$ to a number of packets equivalent to the connection bandwidth delay product.

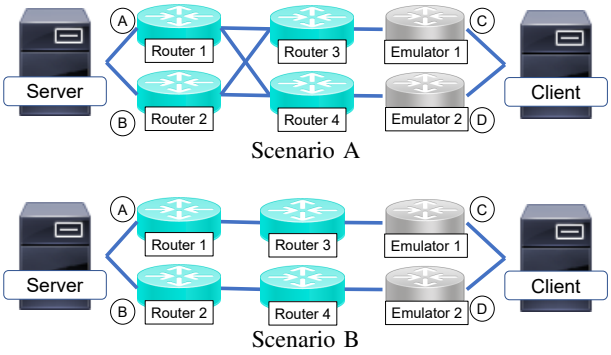


Figure 5. Experimental environment scenarios.

TABLE I. EXPERIMENTAL NETWORK SETTINGS

Element	Value
Video size	225 MBytes
Video rate	5.24 Mb/s
Playout time	6 mins
Video Codec	H264 MPEG-4 AVC
MPTCP variants	BBR, CUBIC
MPTCP schedulers	1. Default (Estimated shortest transmission time) 2. Bottleneck aware

TABLE II. EXPERIMENTAL NETWORK SCENARIOS

Scenario	Emulator (BW, Packets Loss, Delay)
A: Fullmesh	A-1...BW: 3Mb/s, Loss: 0.1%, Delay: 60ms A-2...BW: 3Mb/s, Loss: 0.1%, Delay: 120ms
B: Parallel	BW: 3Mbps, Loss: 0.1%, Delay: 60ms

VI. Video Streaming Performance

Figure 5 describes the network testbed used for emulating network paths with wired links. An HTTP Nginx video server is connected to two L3 switches. In order to support multiple network scenarios, the L3 switches can be directly connected to another router, at which a client is connected. In this paper, the emulator boxes are used to vary each path RTT. We use two topology scenarios: a cross path scenario, where routers' cross-connections produce paths with common bottlenecks; and a parallel path scenario, where paths do not share common bottlenecks. These simple topologies and isolated traffic allow us to better understand the impact of differential delays on TCP variant's performance vis-a-vis path selection properties of path schedulers.

Application and network scenarios are described in Tables I and II, respectively. Video settings are typical of a video stream, with video playout rate of 5.24 Mb/s, and content size short enough to run multiple streaming trials within a short period of time. Four network scenarios are used (Figure 5). i) Two scenarios emulate fullmesh four path subflows, with short (60msec) and long (120msec) propagation delays; The four subflows share bottlenecks at routers close to the video server; ii) Two parallel path scenarios, for short and long propagation delays on two-path subflows, not sharing any bottlenecks. The fullmesh scenarios are used to expose the default scheduler's inadequacy as compared with our proposed bottleneck aware scheduler, whereas the parallel scenarios are meant to show "no harm" of a bottleneck aware scheduler when bottlenecks

are absent. Emulator boxes are tuned to generate multiple path network latency conditions. Path latency directly impacts the default path scheduler, as it gives preference to paths with shorter RTTs. Performance measures are:

- **Picture discards:** number of frames discarded by the video decoder.
- **Buffer underflow:** number of buffer underflow events at video client buffer.
- **Out-of-order packets:** Total number of out-of-order packets during each video streaming session.
- **subflow throughput:** TCP throughput of each subflow.

A. Fullmesh Scenarios

Scenarios A force the sharing of bottlenecks between the four paths available to MPTCP streaming. Each path has a maximum 3Mbit bandwidth, 0.1% packet loss rate, and 60ms or 120ms RTT delays.

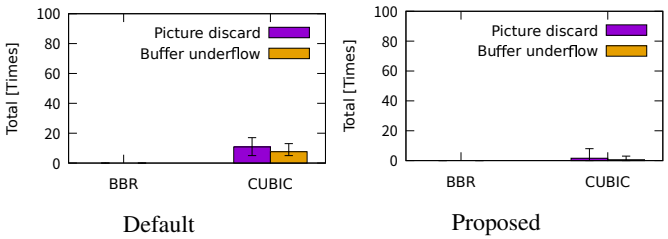
Figures 6 (a) and (b) show five average video streaming frame discard / buffer underflows, and the number of out-of-order packets, respectively for the short 60msec delay scenario. Notice the significant performance improvement of the bottleneck aware scheduler on frame discard and buffer underflow application level performance for the CUBIC TCP variant as compared to the MPTCP default scheduler. Interestingly enough, the average number of out-of-order packets does not seem to change significantly across schedulers. This seems to suggest that more out-of-order packets are concentrated on specific paths, reducing their impact on application level performance.

Figures 6 (c) and (d) show a single streaming trial of BBR and CUBIC, respectively, for the short 60msec delay scenario. The BBR TCP variant shows little throughput dynamic changes between schedulers. However, the CUBIC throughput seems to have become much more stable, showing an even throughput across all paths available. Throughput stability positively impacts video quality, as each path data reception and frame reassembly becomes more predictable.

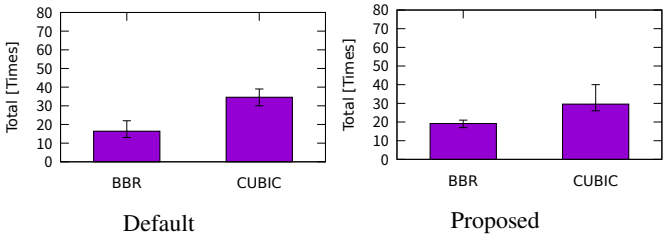
Figures 7 (a) and (b) show five average video streaming frame discard / buffer underflow, and the number of out-of-order packets, respectively, for the long 120msec delay scenario. Although frame discard and buffer underflow have become worse than the short delay scenario when the default scheduler is used, these application performance measures show the same qualitative benefits of the short delay scenario when using our proposed bottleneck aware scheduler. Figures 7 (c) and (d) show the same throughput stability benefits of our scheduler for CUBIC TCP variant.

B. No-cross-link Scenarios

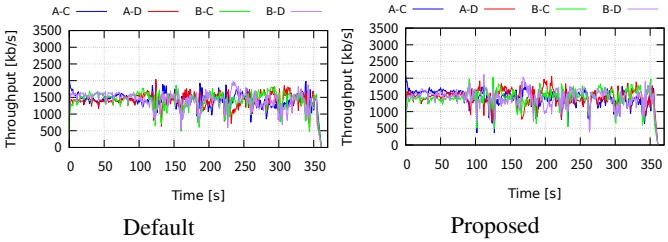
Figures 8 (a) and (b) show five average video streaming frame discard / buffer underflows, and the number of out-of-order packets, respectively, for the short 60msec delay scenario. Notice similar performance between the default and proposed the bottleneck aware schedulers, with the later still improving performance when CUBIC TCP variant is used.



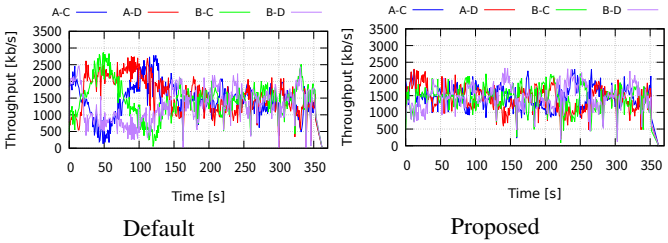
(a) Picture discard and Buffer underflow



(b) Out-of-order



(c) BBR throughput



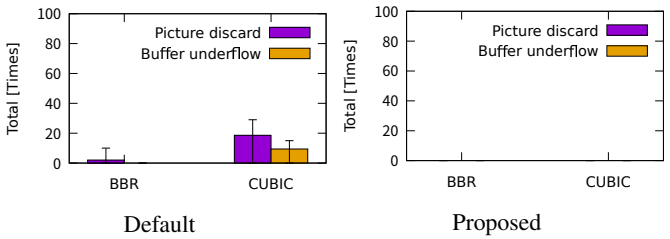
(d) CUBIC throughput

Figure 6. A-1 - Fullmesh Video Performance (delay 60ms.)

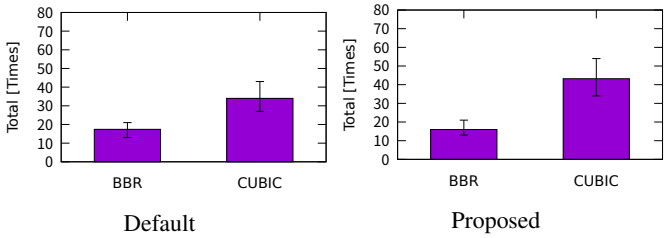
Figures 8 (c) and (d) show a single streaming trial of BBR and CUBIC, respectively, for the short 60msec delay scenario. Throughput dynamics are stable and similar for both schedulers. Hence, in the absence of shared bottlenecks, the proposed scheduler does not disturb throughput stability. Similar qualitative results are obtained for the long 120msec delay scenario, omitted for space's sake. These results verify that the proposed scheduler does not perform any worse than the default scheduler in the absence of shared path bottlenecks.

VII. Conclusion and Future Work

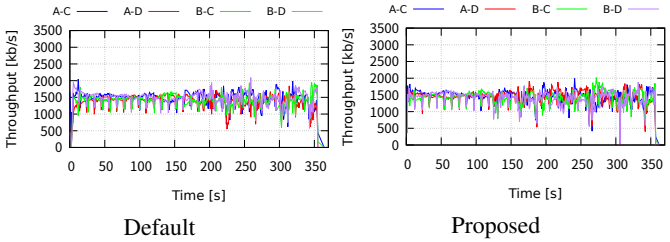
In this paper, we have studied multipath video streaming over fullmesh path scenarios. We have characterized the problem of path interference within a same video stream, demonstrating video application degradation when the default path scheduler is used. We have also proposed a bottleneck aware scheduler, where usage of paths containing a common bottleneck is controlled so as not to interfere negatively with each other. We have shown that the proposed scheduler not



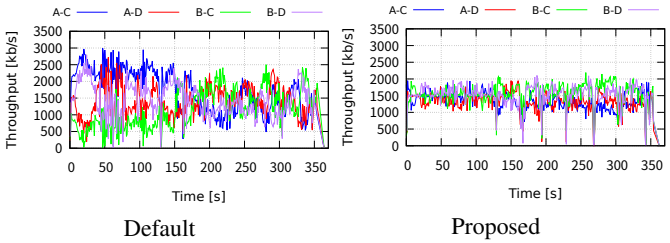
(a) Picture discard and Buffer underflow



(b) Out-of-order

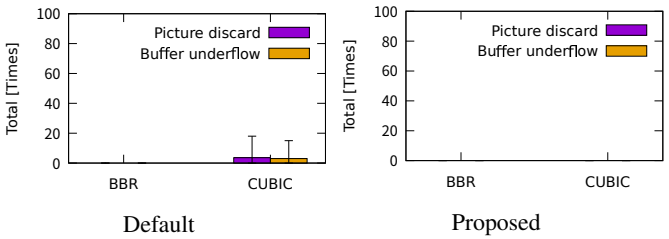


(c) BBR throughput

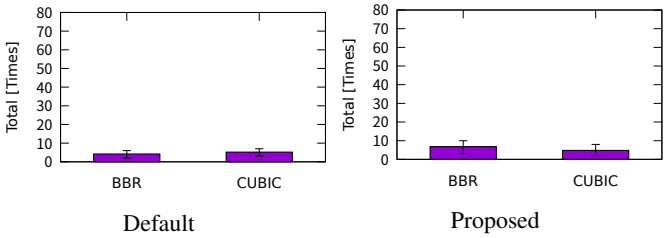


(d) CUBIC throughput

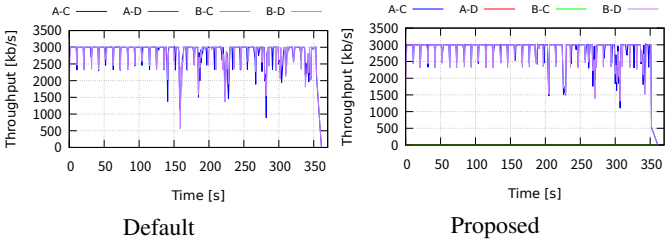
Figure 7. A-2 - Fullmesh Video Performance (delay 120ms).



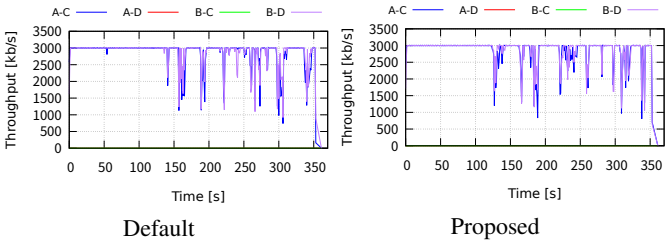
(a) Picture discard and Buffer underflow



(b) Out-of-order



(c) BBR throughput



(d) CUBIC throughput

Figure 8. B - Parallel Video Performance (delay 60ms).

only delivers better video application performance, but also helps ”stabilize” network throughput performance at each path. We have also shown that the proposed path scheduler does not cause performance degradation when paths do not share a bottleneck, when compared to the MPTCP default scheduler.

We are currently investigating the performance of our proposed scheduler on satellite access links.

ACKNOWLEDGMENTS

Work supported by JSPS KAKENHI Grant #24K03045.

REFERENCES

[1] I. Rhee, L. Xu, and S. Ha, “CUBIC for Fast Long-Distance Networks,” Internet Draft, draft-rhee-tcpm-ctcp-02, August 2008.

[2] N. Cardwell, Y. Cheng, I. Swett, and V. Jacobson, “BBR Congestion Control,” *IETF draft-cardwell-iccrg-bbr-congestion-control-01*, November 2021.

[3] M. R. Palash and K. Chen, “MPWiFi: Synergizing MPTCP Based Simultaneous Multipath Access and WiFi Network Performance,” IEEE Transactions on Mobile Computing, vol. 19, no. 1, pp. 142-158, Jan. 2020

[4] M. Amend, V. Rakocevic, and J. Habermann, “Cost optimized multipath scheduling in 5G for Video-on-Demand traffic,” In Proc. of IEEE Wireless Communications and Networking Conference - WCNC 21, pp. 1-6, March 2021.

[5] M. F. Imaduddin, A. G. Putrada, and S. A. Karimah, “Multipath TCP Scheduling Performance Analysis and Congestion Control on Video Streaming on the MPTCP Network,” In Proc. of Intern. Conference on Software Engineering & Computer Systems and 4th Intern. Conference on Computational Science and Information Management - ICSECS-ICOCSIM, pp. 562-567, August 2021.

[6] Y. Xing et al., “A Low-Latency MPTCP Scheduler for Live Video Streaming in Mobile Networks,” IEEE Transactions on Wireless Communications, vol. 20, no. 11, pp. 7230-7242, Nov. 2021.

[7] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, “Architectural Guidelines for Multipath TCP Development,” IETF RFC 6182, 2011.

[8] S. Nagayama, D. Cavendish, D. Nobayashi, and T. Ikenaga, “Path Switching Schedulers for MPTCP Streaming Video,” IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), August 2019, pp. 1-6.

[9] J. Zhao et al., “MPTCP+: Enhancing Adaptive HTTP Video Streaming over Multipath,” *Proceedings of IEEE/ACM International Symposium on Quality of Service (IWQoS)*, pp. 1-6, June 2020.

[10] B. Kimura et al., “Interpath Contention in MultiPath TCP Disjoint Paths,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1387-1400, August 2019.