Semantically Aware, Mission-Oriented (SAMO) Networks: Fine-Grained, Network-Layer Services for Intelligent, Next Generation Networks

Timothy J. Salo
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, U.S.A.
salo@saloits.com

Zhi-Li Zhang
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, U.S.A.
zhzhang@cs.umn.edu

2023 IEEE Future Networks World Forum (FNWF) | 979-8-3503-2458-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/FNWF58287.2023.10520533

***Abstract*—We introduce the Semantically Aware, Mission-Oriented (SAMO) framework, which enables fine-grained, host-application-to-network signaling. This signaling employs SAMO metadata, carried in an existing network-layer header, to inform the network of the application's desires. These metadata can invoke SAMO network-layer virtual network functions (VNFs) to provide sophisticated services for the packet. The framework could enable, for instance, a secure, application- and application-protocol independent, network-layer, publish/subscribe or situational awareness service.**

The SAMO framework is particularly beneficial in mobile edge or wireless networks, where the state of the network may change rapidly, and where quickly adapting to limited, and often variable, network resources is more important than, for example, maximizing router throughput. The framework is equally applicable in enterprise networks, industrial verticals, or other private networks, where the organization deploying or using the network needs the network to adapt to the specific semantics of the data being carried.

SAMO signaling creates a disciplined, cross-layer interface, which can promote improved application/network integration or support other in-network computing architectures. Furthermore, the SAMO framework avoids embedding application knowledge in network devices and it functions even when user data are encrypted. Moreover, the framework permits new, sophisticated, network-layer extensions and services to be easily tested or deployed in exiting IPv4/IPv6 or 5G/NextG networks.

SAMO VNFs are ideally positioned to employ artificial intelligence and machine learning (AI/ML) technologies to enable networks to modify their behaviors in response to the semantics of the data streams. The SAMO framework ensures that AI/ML-enhanced SAMO VNFs have available semantic information about the data, via the application-generated SAMO metadata!

A proof-of-concept (PoC) SAMO-enabled host application and a simple SAMO VNF were implemented and used to evaluate the efficacy and performance of the framework.

***Index Terms*—Semantically Aware Networks, Semantically Aware Mission-Oriented Networks, SAMO Networks, Network-Layer Applications, 5G Networks, 6G Networks, NextG Networks, Virtual Network Functions, Network Programming**

I. Introduction

The Semantically Aware, Mission-Oriented (SAMO) network framework enables an application to signal a network, potentially per-packet, to request enhanced network-layer processing for an individual packet. An application signals the network by attaching semantic information, "SAMO metadata", to certain packets. These metadata may describe the semantics, the contents, the meaning or other attributes of the application data. In other cases, these metadata may explicitly invoke a specific, potentially complex, network-layer service, (provided by a "SAMO Virtual Network Function", or "SAMO VNF"), to process the packet. Because these metadata are carried in an IP header, they are accessible to network-layer software, including SAMO VNFs. These network-layer VNFs operate principally on the packet's SAMO metadata, other information available to the network layer, and potentially stored state maintained by the VNF itself. Most importantly, the VNFs do *not* access and do *not* modify the application-layer data and therefore have no awareness of applications, application data formats, application-layer protocols, or application data. SAMO VNFs are implemented as replaceable, configurable, software components, which can be tailored to the specific needs of particular host applications, classes of host applications, or operational environments.

When might a host application benefit from requesting specialized, network-layer processing for individual packets? An *application's* performance, or more specifically the performance of the services that the application delivers to its human or inanimate users, is often the most important measure of a networked system (e.g., a system composed of hosts, applications, and networks). After all, *applications* are the *raison d'être* for most computer networks, (well, that and network research, of course). Application-specific *Quality of Experience* (QoE) measures are often used to assess how well an *application* meets the needs of its users, in contrast to traditional network-focused metrics such as Quality of Service (QoS), (which typically includes such measures as delay, packet loss rate, bandwidth, latency and numerous others). That is, QoE characterizes the services *provided by the application, as perceived by its users*, while QoS characterizes the services *provided by the network to applications*. The framework focuses on enhancing the Quality-of-Experience

Identify applicable funding agency here. If none, delete this.

delivered by an application.

A host application is likely to benefit from explicitly requesting special processing for individual packets when doing so improves the QoE delivered by the application to its users, particularly when compared to traditional, flow-based QoS assurances. Additionally, per-packet metadata could enable more efficient provision of existing network services, or even enable new network-layer services that are difficult to provide efficiently using traditional architectures.

Consider a very simple example where a host application uses per-packet requests for enhanced network-layer services to improve the QoE delivered by the application, namely a video streaming application. (This simple example illustrates how the framework operates, and could be effected using existing facilities, such as the differentiated services code point, DSCP; later examples build on this example to demonstrate the framework's power and flexibility. Note that the proof-of-concept implementation described below includes over 30 bits of information in the metadata, much more than fits in the six-bit DSCP field.) This example video streaming application considers some packets more important to its QoE than others, such as those that contain lower-resolution video information or audio. The loss of these more-important packets adversely affects the application's QoE more than, for example, the loss of less-important packets containing high-resolution video information. Given this, the application's QoE is likely be better if the network makes a greater effort to successfully deliver the packets that the *application* considers more important. How should the network determine which packets the application considers most important? While the network could try to guess which packets are the most important to the application, perhaps using deep packet inspection to examine the application-layer protocol and data, the framework offers a mechanism that allows the application to explicitly *tell* the network which packets it considers more important. The application in this example explicitly informs the network of the importance of each packet by attaching to each packet SAMO metadata that contain the application's priority for that packet. The network uses this information to prefer to discard packets that the application has signaled are less important. This selective discard, enabled by the framework, helps the application deliver an enhanced QoE.

In this example, the network layer can make decisions that provide better service to this application (based on information signaled by the application), but the network layer has *no* understanding of the video streaming application, its protocols, or its data formats. Rather, the network layer merely operates on metadata inserted in the network-layer packet header to prioritize packets. This signaling mechanism, SAMO metadata, isolates the network layer from the application layer by providing a well-structured method of enabling the application to signal the network. In short, the framework enables the network to deliver services that are better matched to the needs of the application, without embedding application knowledge within the network layer.

Because this simple example avoids embedding any application awareness or dependencies in the network layer, other applications could easily use this new network-layer service, simply by attaching the appropriate SAMO metadata to packets. As we show in the remainder of this paper, this per-packet signaling mechanism, coupled with sophisticated network-layer services, can be used to improve applications' QoE, without adversely affecting the flexibility, generality or scalability of the network layer. It enables the network to better match the services it provides to the needs of the application by responding to explicit application requests.

The remainder of the paper is organized as follows: Section II details how the SAMO framework can be used to provide enhanced network-layer services in IPv4 and IPv6 networks, while Section III summarizes implementation experience using the framework. Finally, Section IV contains thoughts on how the SAMO framework might be applied in and benefit 5G/NextG networks.

II. The SAMO Framework

The SAMO framework permits SAMO-enabled host applications to invoke customizable, potentially sophisticated, network-layer functionality ("SAMO virtual network functions", "SAMO VNFs", or simply "VNFs") to influence how a network device processes individual packets. The framework can provide fine-grained, content-aware, network-layer functionality and services that are powerful, flexible, configurable, and scalable. This section describes the operation of the SAMO framework in IPv4 and IPv6 networks.

The framework includes two principal components:

- A SAMO-enabled host application may signal a network, by attaching SAMO metadata to a packet, of the application's desire that the packet receive special processing by the network. SAMO metadata may describe the semantics, the contents, meaning, priority, or other attributes of the application data contained in the packet, or may explicitly request the special processing desired for the packet. SAMO metadata are transported in an IP header. With IPv4 packets, SAMO metadata are carried as an IP option, while with IPv6 packets, SAMO metadata are carried in a hop-by-hop extension. The precise interpretation of the SAMO metadata is an agreement between the SAMO-enabled application and SAMO VNFs executing on network devices. As a result, networks operating in different environments (e.g., different industrial verticals or enterprises) may interpret metadata values differently.
- A SAMO-enabled network device may, upon detecting SAMO metadata in an IP packet header, perform special processing on the packet, perhaps invoking a SAMO VNF to do the actual work. The network device generally considers the contents of the SAMO metadata, as well as other information accessible to the network layer, when processing the packet.

Two examples are used here for expository purposes. These examples will assist in explaining how the SAMO framework works, but don't, in and of themselves, demonstrate its full power and utility. Note that these potential applications are

illustrative of the potential of the framework, but are not part of the framework itself.

- An application, perhaps running on a small IoT device, could benefit from the network providing an efficient, intelligent, configurable, network-layer, publish/subscribe dissemination service. This represents a broad class of solutions enabled by the framework, namely solutions in which the network provides potentially sophisticated, per-packet services and maintains internal state, in effect moving responsibilities, intelligence, and state from hosts to the network.
- A network might provide a secure, resilient, situational awareness system. This system could securely collect and deliver situational awareness updates to participating hosts. Because the framework only examines SAMO metadata (and other network-layer information), the user data could be encrypted. And, because the framework doesn't require applications to be aware of the network configuration, an infrastructure-less situational awareness system could be deployed in demanding environments where connectivity to the global Internet isn't assured or where the network might become partitioned.

A. The SAMO Framework

First a bit of terminology. For the purposes of these discussions, "SAMO network-layer service", or simply "SAMO service", denotes the unique processing or service that a SAMO network device performs on packets that contain SAMO metadata. Likewise, "SAMO virtual network function", or simply "VNF", identifies network-layer software that operates on packets that contain SAMO metadata.

1) Extensible Metadata: The interpretation of SAMO metadata is an agreement between applications and a VNF. As a result, different VNFs may interpret metadata values differently, Therefore, networks must administratively configure the mapping between metadata values and VNFs. Different networks may employ different VNFs, depending on the operational environment and application needs. Metadata values are limited only by the size of the container within which they are transported (e.g., IPv4 options, up to 38 bytes, or IPv6 hop-by-hop extension header, up to 2310 bytes), although practical considerations may dictate lower limits. In particular, the SAMOnet designer must carefully weigh the additional bandwidth consumed by SAMO metadata against the potential benefit of employing the metadata.

The framework does not specify the meaning of the SAMO metadata, how a SAMO VNF uses that metadata, or how a VNF processes a packet. Rather, these decisions are left to the application and network designers (or, perhaps application/network co-designers). These VNFs are created as part of the development and deployment of the SAMOnet and SAMO-enabled host applications, rather than as part of the development of the network device or by the SAMO framework.

2) Network-Layer Operation and Services: The SAMO framework operates at the network layer. SAMO metadata are transported in the IPv4 header (as an IP option) or in an IPv6 hop-by-hop extension header, and so are generally accessible by software that operates at the network layer, including SAMO VNFs. Because the framework operates at the network layer, and uses information accessible to the network layer, decisions made by a SAMO VNF can easily reflect current local knowledge about the state of the network, including the network topology, bandwidth, traffic, congestion, delay, wireless channel state, and other conditions, to an extent that is difficult with centralized, host-based, or application-layer solutions.

3) Per-Packet Services: SAMO VNFs operate on individual packets, rather than on flows: the appropriate VNF is invoked when a network device recognizes SAMO metadata when processing a packet. Even though VNFs operate on individual packets, many of them will maintain internal state. As such, *some* VNF may be aware of flows, and will treat the packet being processed as part of a flow.

Because SAMO VNFs operate on individual packets, rather than on flows, they are generally better suited for use use with datagram transport protocols, such as UDP, SCTP, or datagram QUIC, rather than stream transport protocols such as TCP. Clearly, treating packets within a TCP session differently is likely to result in less-than-desirable behaviors.

Inasmuch as SAMO metadata are in-band, they are easily associated with specific packets. This contrasts with out-of-band signaling protocols, such as RSVP, where the association with specific packets is less clear.

4) Application Layer Independence: SAMO VNFs examine SAMO metadata provided by host applications, but do *not* access the application data itself: as such, VNFs have *no* understanding of application-layer protocols or data formats. Likewise, SAMO VNFs do not perform deep packet inspection (DPI) or otherwise access higher level protocol headers or data in any manner. This application-protocol-independence means that application layer data may be encrypted, without affecting, or even knowledge of, SAMO VNFs.

A SAMO service could be used by any application that attaches the appropriate SAMO metadata to IP headers. Again, VNFs are unaware of application-layer formats and protocols, and so are unaware of which application is invoking a VNF. Similarly, application-layer protocols may evolve, without affecting or requiring modification of a SAMO service or VNF.

5) Mission-Oriented, Enterprise, and Industrial Vertical Networks: The SAMO framework is motivated by mission-oriented networks: networks that are deployed for a specific purpose, that are usually administered by a single organization, and that are generally edge, rather than transit, networks. Because SAMO-enabled network devices are highly configurable, SAMO VNFs can provide services that are tailored to the network's specific mission. Examples of these networks include large-scale wireless sensor networks, mobile tactical networks, and mobile wireless networks, as well as enterprise or industrial vertical networks that could benefit from semantically aware, network-layer intelligence. The framework is particularly beneficial in bandwidth-challenged networks,

those in which bandwidth, rather than network device processing capacity, is the scarce resource that should be conserved. SAMO-enabled network devices can employ network-layer processing in order to conserve network bandwidth or to adapt to the currently available bandwidth. RFC 8799 [Carpenter and Liu(2020)] describes these environments as "limited domain" networks.

6) Operation with Security Protocols: Because the SAMO framework operates at the network layer, it can be deployed seamlessly with transport-layer security protocols, such as the Datagram Transport Layer Security (DTLS) protocol [Rescorla et al.(2022)]. The framework can also be deployed with network-layer security protocols, such as the IPsec protocols.

7) Incremental Deployment: SAMO VNFs can be deployed incrementally: not all network devices in a network need to support the SAMO framework. Rather, non-SAMO network devices merely need to forward packets containing SAMO metadata without examining or modifying the metadata. This simplifies the deployment of these new services.

The framework also does not require changes to application-layer protocols, such as extending the application-layer protocol to include new control messages. Rather, the host application needs only to be modified to attach SAMO metadata to the appropriate packets transmitted by the application.

8) A Framework, Not a Collection of Applications: The SAMO framework offers an approach to closer collaboration between applications and networks. But, the framework does *not* specify what applications can or ought to take advantage of this capability. The example applications included here are not part of the framework; they are just a few of the countless potential solutions that the framework enables.

9) A Framework, Not an Implementation Strategy: The framework does *not* specify how, for example, SAMO VNFs ought to be implemented. The proof-of-concept VNF described in Section III was implemented as a user-space process. Similar functionality could have been implemented, for example, in the kernel, using a P4 switch (perhaps extended with microservices or external functions), or employing a kernel-bypass architecture such as DPDK.

10) Enhanced Application/Network Isolation: Applications using the framework treat a network as an opaque service that transports and processes packets. This contrasts with some recent proposals, such as Application-Layer Traffic Optimization (alto) [Kiesel et al.(2014)] or Segment Routing [Filsfils et al.(2018)], which enable hosts to learn about state of the network and change their behaviors accordingly.

11) Infrastructure-less Design and Operation.: The framework does not rely on centralized servers, either to manage SAMO VNFs or to respond to host or application queries. As such, the framework is better suited to, for example, mobile wireless networks, where the network topology and state could change rapidly.

B. Example: Network-Layer Publish/Subscribe Dissemination Service

The SAMO framework could enable a network-layer publish/subscribe dissemination service that would be useful for constrained IoT devices. The SAMO metadata associated with packets being published might include: a type parameter that indicates that the pub/sub VNF should process this packet, a pub/sub topic, a request (e.g., publish or subscribe), and optionally attributes that further describe the contents of the pub/sub message. In this example, the *network*, (specifically, a SAMO VNF, rather than application-layer software), maintains a list of subscribers and forwards a message to the hosts that have subscribed to a particular topic. Hosts wishing to receive messages would subscribe to a topic, perhaps indicating attributes on which messages should be filtered. The SAMO publish/subscribe service may use this information to optionally filter pub/sub messages, based on the attributes of a published message and the attributes of the messages that a subscriber wishes to receive.

An obvious question is: *Why might it be beneficial to implement a publish/subscribe dissemination service at the* network *layer?* Most readers will note that this example is a stark departure from the traditional smart host / dumb network model historically employed by the Internet and IP networks. Beyond serving as an example of flexibility and expressiveness of the framework, this network-layer publish/subscribe service would:

- Enable messages to be routed efficiently, because the network layer would have a better understanding of the current topology of the network, compared to application-layer solutions, (e.g., by avoiding the path expansion that often results from the use of application-level overlay networks).
- Simplify the implementation of IoT publishers. Many IoT devices are severely constrained, with limited memory and computational power. Often, these devices are battery-powered, and so their lifetime is determined by the device's total expenditure of electrical energy, particularly the total number of bits they transmit cumulatively. Being able to push much of the complexity of a pub/sub system off onto, for example the network, would save these devices' limited memory, CPU cycles and battery energy. To publish messages, the IoT device would need to do nothing more than attach SAMO metadata to a single copy of the message message and transmit it. Everything else is handled, in this example, by the network, rather than by applications, perhaps hosted by a geographically distant cloud service.
- Support multiple application-layer pub/sub applications and protocols, since the pub/sub VNF would not examine or be cognizant of application-layer formats and protocols. For example, a SAMO pub/sub VNF could conceivably support both MQTT and CORE publish/subscribe protocols with a common dissemination service.
- Offer Internet service providers (ISPs) or carriers op-

portunities to offer new, general, useful, and potentially revenue-generating [network-layer] services.
- Demonstrate the potential of a mechanism that permits existing networks to provide new, novel services, without modifying the basic Internet protocols.

C. Example: Secure, Robust, Situational-Awareness System

This network-layer publish/subscribe service could easily be extended to create a secure, robust situational awareness system.

- *Secure Communications* Pub/sub messages could be encrypted; again, this encryption would be invisible to the SAMO VNFs. Some publishers might use multiple security associations, to control the information that particular subscribers are permitted to access.
- *Authenticated Messages or Hosts* Pub/sub messages could be cryptographicaly signed, enabling receivers to authenticate messages or hosts.
- *Duplicate Message Suppression* The publish/subscribe dissemination service could filter duplicate messages (e.g., messages that contain no new information, perhaps beyond a timestamp) based on hashes of the packet's user data and timestamps that are inserted in the metadata.
- *Digital Twin* A SAMO VNF could create a digital twin by retaining copies of the most recent messages published by an IoT device. The VNF would consider messages with similar SAMO metadata values to be equivalent and retain only the most recent one. Of course, the VNF wouldn't examine, much less understand, the contents of the messages, which might even be encrypted.

D. Semantically Aware, Network-Layer Intelligence

SAMO VNFs provide an ideal platform for employing artificial intelligence or machine learning AI/ML techniques to leverage semantic information. Application-provided semantic information (contained in SAMO metadata) is available to SAMO VNFs, and so could be easily accessed by AI/ML functions. For example, AM/ML processing could generate an enhanced understanding of the instantaneous state of the physical wireless channels, and use that understanding, combined with semantic information to, for example, select the most appropriate wireless channel for a packet.

E. Related Work

The SAMO framework is consistent with a long tradition of embedding greater intelligence in the network layer.

1) Future Internet Architectures: McCauley, et al. [McCauley et al.(2019)] suggest a future Internet architecture in which new Internet functionality is implemented in a new "layer 3.5" protocol that is inserted between between the IPv6 header and the higher-level protocols. This layer 3.5 protocol is carried as a newly defined protocol header. Our work suggests that reusing the existing IPv6 hop-by-hop options header ([Hinden and Deering(1998)], [Carpenter and Jiang(2013)]), rather than creating a new protocol header, may simplify the deployment of this and similar future Internet architectures, in part by using existing host API code to create this header [Jinmei et al.(2003)]. This use of the hop-by-hop options header appears to be consistent with its somewhat vaguely defined purpose, although the use of this header is frowned upon, by at least some, in the Internet core [Carpenter and Jiang(2013)], [Gont et al.(2016)], [Hinden and Fairhurst(2023)], [Peng et al.(2023)].

2) Software-Defined Networks: The framework shares some similarities with software-defined networking (SDN), in that they both assume that network devices are highly programmable. Beyond that, the SAMO framework differs from software-defined networks (SDNs) in several key aspects. First, a SAMO network is fully distributed: it does not rely on a central controller, although a central controller might be useful in initially configuring the network. This decentralized approach may be beneficial in environments such as wireless tactical networks or first responder networks, where communications with a central controller might be costly or unreliable. Second, the SAMO framework operates only on network-layer packets: it does not need to understand or access higher-layer protocols. In fact, a SAMO might not even be able to access to these data or protocols, if they are encrypted. This contrasts with SDN's potential to examine higher-layer protocols. Third, the framework operates on individual packets (VNFs are invoked when SAMO metadata are encountered), while SDNs operate on flows (e.g., by matching packets with flow table entries). Finally, SDNs invoke pre-defined functions to process packets, while the framework assumes that new VNFs will be designed to meet the needs of new applications or operational environments.

3) Application-Layer Traffic Optimization: The Application-Layer Traffic Optimization (ALTO) system uses a server-based approach to improve coherence between the operations of applications and the network. The ALTO system uses a series of servers to "enable P2P applications to obtain information regarding network topology" [Seedorf et al.(2009)]. The SAMO framework offers an alternate approach, namely SAMO VNFs (which can directly access information about the network that is maintained by the underlying network device) and the SAMO host applications with which they cooperate. Unlike with the ALTO framework, SAMO-enabled applications signal the network about their desires, rather than try to discern and respond to the current state of the network.

4) Application-Aware Networking: The Application-aware Networking (APN) Framework is being developed to help networks better meet the needs of applications [Li et al.(2022)]. The APN framework signals an application's network performance requirements, rather than the more general signaling mechanism used by the SAMO framework.

A new IPv6 header has been defined for APN [Quinn et al.(2018)]. Again, our work has shown the benefits of reusing the loosely defined IPv6 hop-by-hop options header.

5) Segment Routing: Segment Routing in an application of the source routing model [Filsfils et al.(2018)]. While Segment Routing can invoke processing along the path, the principal

difference with the SAMO framework is that the framework is largely driven by the end applications. Additionally, segment routing requires the application to be aware of the configuration of the network, while for SAMO applications, the network is largely opaque.

6) Service Function Chaining: The Service Function Chaining (SFC) architecture is designed "for the creation and ongoing maintenance of Service Function Chains (SFCs) in a network" [Halpern and Pignataro(2015)]. Service Functions are roughly analogous to SAMO VNFs. This work was driven by a desire to "steer" traffic through several Service Functions, such as firewalls, load balancers, or functions that manipulate HTTP headers [Quinn and Nadeau(2015)]. There is nothing inherent in the SAMO framework that motivates a similar mechanism to "steer" packets through a collection of VNFs, although conceivably VNFs could be created that might benefit from such a scheme.

7) Network/Application Integration: The framework offers one approach to network/application integration. A very early version of these ideas were presented as a two-page poster at an ACM SIGCOMM Workshop on Network Application Integration/CoDesign [Salo and Zhang(2020)].

F. Contributions

The framework contributes to the evolving views on how and how much intelligence ought to be implemented in the network, rather than in the end systems. Of course, the framework is intended to address special use cases (e.g., edge networks, rather than transit networks). Nonetheless, the authors argue that the framework and some of its potential applications demonstrate that there are cases where network systems *can* perform better, when applications, and networks are designed together to meet the needs of particular applications and environments. The framework offers:

- A disciplined approach to adding intelligence to networks through the use of cross-layer signaling to enable applications to request advanced network services
- An effective approach to network/application integration by permitting the services networks provide to be tailored to the specific requirements of the limited domains within which they operate.

III. Implementation Experience

A proof-of-concept (PoC) SAMO system was developed to evaluate the technical feasibility, performance, and efficacy of the SAMO framework. This PoC system was intended to better understand:

- The feasibility of creating SAMO-enabled applications, specifically whether user-space applications hosted on modern Linux systems could set SAMO metadata without requiring modifications to the kernel or other parts of the operating system.
- The performance implications of creating SAMO metadata, specifically the cost of having the kernel insert metadata in the IPv4 header or creating an IPv6 hop-by-hop extension header.

Experiment	Mode	IPv4	IPv6
Builtin Ethernet	transmit	33,177 pps 30 μsec/pkt	34,329 pps 29μsec/pkt
	receive	25,350 pps 39 μsec/pkt	23,802 pps 42 μsec/pkt
	loss	10,179 pps	8,986 pps
Builtin No Metadata	transmit	46,499 pps 22 μsec/pkt	34,632 pps 29 μsec/pkt
	receive	26,904 pps 37 μsec/pkt	21,955 pps 46 μsec/pkt
USB	transmit	29,305 pps 34 μsec/pkt	29,305 pps 34 μsec/pkt
	receive	19,165 pps 52 μsec/pkt	19,165 pps 52 μsec/pkt
	loss	10,037 pps	10,037 pps

TABLE I
samoapp IPv4 and IPv6 Performance.

Experiment	Mode	IPv4	IPv6
Kernel Forwarding	transmit	30,503 pps	35,081 pps
	receive	18,915 pps	27,670 pps
	loss	11,787 pps	7,314 pps
sampvnf No Metadata	transmit	51,651 pps	29,421 pps
	receive	32,728 pps	18,022 pps
	loss	11,787 pps	7,314 pps

TABLE II
samoapp Performance (Packets per Second).

- The feasibility of implementing a SAMO VNF, executing in user space and using a modern Linux kernel, with no modifications to the kernel.
- The performance implications of implementing a SAMO VNF as a user-space application.

A. Proof-of-Concept Software

A PoC system that employed the SAMO framework was implemented and evaluated. This system included a priority-drop VNF, similar to that described in the first example above. Two SAMO applications were implemented:

- *samoapp*, a SAMO-enabled, user-space, host application that generates and receives a simulated layer-coded data stream
- *samovnf*, a user-space SAMO VNF that, when congestion is experienced, discards packets that are identified by SAMO metadata as lower-priority.

B. Testbed Configurations

The testbed on which these experiments were conducted included Raspberry Pi 4 Model Bs connected either through a gigabit Ethernet switch or through direct Ethernet cables. The hosts run Ubuntu 22.04. The following results were generated with a three-node configuration, shown in Figure 1 in which a router running the *samovnf* software selectively forwarded packets between two nodes running *samoapp*.

C. Experimental Results

Experiments were conducted to explore the feasibility, efficacy, and performance of the SAMO applications and framework.

1) samoapp *performance:* The performance of the *samoapp* software was characterized.

The objective of these experiments is to understand the transmission and reception performance, and bottlenecks, of the *samoapp* application. Two hosts, each running *samoapp*, were connected directly (i.e., with no intervening network-layer device). The transmitting application transmitted packets continuously, limited only by the resources available on the transmitting host. Likewise, the receiving node attempted to receive packets continuously. In these experiments, the length of the data was reduced by 100 bytes every 10 seconds, starting at 1400 bytes. These experiments affirm that *sampapp* performs no per-byte processing. Each of these experiments were conducted using both IPv4 an IPv6. These experiments used three configurations:

- Direct connection through an Ethernet cable using the higher-performance builtin Ethernet interface. These tests illustrated the maximum network performance of the Raspberry Pis.
- A similar experiment, but with no metadata being attached to the packets. This experiment shows the cost of attaching metadata to packets.
- Direction connection using gigiabit Ethernet dongles and the higher-performance USB ports. This experiment provides a baseline for evaluating the performance of later experiments using a SAMO-enabled network device (router).

Table I summarizes these results. In these experiments, the transmit and receive threads consumed nearly 100 per cent of the CPU on which they were running. Several results are worth noting. First, receiving packets required more CPU resources than transmitting packets (approximately 30 per cent more for IPv4 and 45 per cent more for IPv6). The cost of adding metadata to the transmitted packets was small (27 percent of the time required to transmit a packet for IPv4 and zero percent for IPv6).

2) samovnf *performance.:* The three-node testbed configuration was used to simulate nodes connected to wireless links, the bandwidth of which varied over time. The link from the host running *samoapp* was throttled to avoid overrunning the network devices. The bandwidth of the simulated, variable-bandwidth wireless link was reduced every ten seconds. The results of three configurations are shown in Table II: 1) kernel forwarding, where packets were forwarded in the kernel, with no processing of SAMO metadata or by the application-level *samovnf* software, 2) forwarding through the *samovnf* software, but with not SAMO metadata processing, and 3) forwarding through the *samovnf* software with SAMO metadata examined.

Figures 2, 3, and 4 show that the *samovnf* software behaved as expected. Figure 2 shows that the sending node transmitted roughly equal amounts of data for each layer. When *samovnf* ignored the SAMO metadata when dropping packets (i.e., performed tail drop), each layer experienced similar packet loss. On the other hand, when *samovnf* considered the SAMO metadata when dropping packets (i.e., dropped the lowest priority packets), the highest-priority packets were received when congestion was experienced.

D. Discussion

These experiments using a proof-of-concept SAMO app and SAMO VNF demonstrate that the systems using the SAMO framework can effectively be implemented on modern Linux system. Moreover, the framework did not appear to introduce any significant performance penalties.

IV. Future Work

The framework was originally developed for IPv4 and IPv6 networks; future work will extend it for use in 5G and beyond networks. 5G networks are, in several important aspects, fundamentally different than traditional IP networks. Most notably, 5G networks are aware of the identity of most of their users (”subscribers”) and maintain information about those users, in sharp contrast to IP networks, where users, hosts, and applications are anonymous to the network. Because users generally authenticate with a 5G network before using it, these networks could, for example, permit specific SAMO VNFs to be available to a particular user.

The framework could simplify the provision of new or experimental 5G (or 6G) network-layer services in an architecturally clean fashion. SAMO VNFs could potentially reside in several locations within 5G networks. They could operate on ingress to the User Plane Function (UPF), (e.g., on entry from a Data Network (DN), such as the Internet), on egress from the UPF (e.g., prior to be handed to the Radio Access Network), or somewhere along the UPF (e.g., in an edge computing configuration). Likewise, SAMO VNFs could be hosted on the gNodeB (5G base station) or the UE (5G user equipment or end device). The SAMO metadata will be carried as an extension to the encapsulating protocol, rather that expecting the various 5G services to extract the SAMO metadata from the encapsulated IP packet.

Acknowledgment

This research was supported in part by the NSF under Grants CNS-1814322, CNS-1836772, CNS-1901103, CNS-2106771, CNS-2128489, and Seed Grants from University of Minnesota MnRI, CTS and CSE and an unrestricted gift from InterDigital.

References

[Carpenter and Jiang(2013)] Brian E. Carpenter and Sheng Jiang. 2013. Transmission and Processing of IPv6 Extension Headers. RFC 7045. https://doi.org/10.17487/RFC7045

[Carpenter and Liu(2020)] Brian E. Carpenter and Bing Liu. 2020. Limited Domains and Internet Protocols. RFC 8799. https://doi.org/10.17487/RFC8799

[Filsfils et al.(2018)] Clarence Filsfils, Stefano Previdi, Les Ginsberg, Bruno Decraene, Stephane Litkowski, and Rob Shakir. 2018. Segment Routing Architecture. RFC 8402. https://doi.org/10.17487/RFC8402

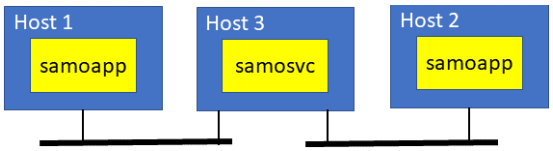

Fig. 1. SAMO VNF Testbed Configuration.

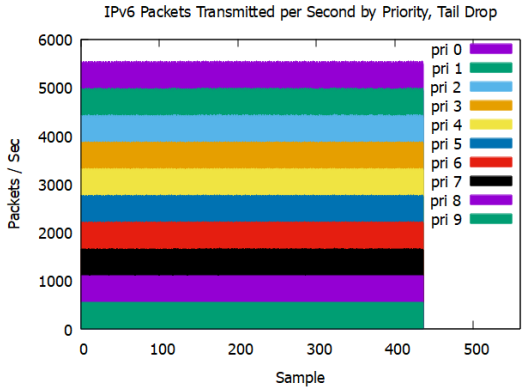

Fig. 2. Transmitted Data.

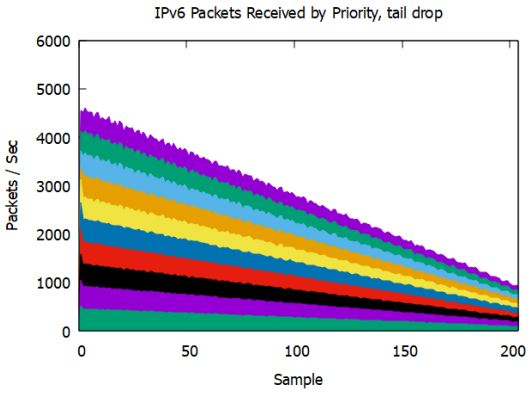

Fig. 3. Received Data - tail drop.

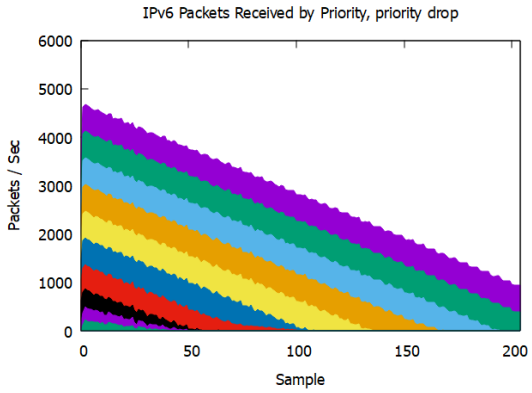

Fig. 4. Received Data - Priority drop.

[Gont et al.(2016)] Fernando Gont, Jen Linkova, Tim Chown, and Will (Shucheng) LIU. 2016. Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World. RFC 7872. https://doi.org/10.17487/RFC7872

[Halpern and Pignataro(2015)] Joel M. Halpern and Carlos Pignataro. 2015. Service Function Chaining (SFC) Architecture. RFC 7665. https://doi.org/10.17487/RFC7665

[Hinden and Deering(1998)] Bob Hinden and Dr. Steve E. Deering. 1998. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460. https://doi.org/10.17487/RFC2460

[Hinden and Fairhurst(2023)] Bob Hinden and Gorry Fairhurst. 2023. *IPv6 Hop-by-Hop Options Processing Procedures*. Internet-Draft draft-ietf-6man-hbh-processing-08. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-6man-hbh-processing/08/ Work in Progress.

[Jinmei et al.(2003)] Tatsuya Jinmei, W. Richard Stevens, and Erik Nordmark. 2003. Advanced Sockets Application Program Interface (API) for IPv6. RFC 3542. https://doi.org/10.17487/RFC3542

[Kiesel et al.(2014)] Sebastian Kiesel, Wendy Roome, Richard Woundy, Stefano Previdi, Stanislav Shalunov, Richard Alimi, Reinaldo Penno, and Y. Richard Yang. 2014. Application-Layer Traffic Optimization (ALTO) Protocol. RFC 7285. https://doi.org/10.17487/RFC7285

[Li et al.(2022)] Zhenbin Li, Shuping Peng, Daniel Voyer, Cong Li, Peng Liu, Chang Cao, and Gyan Mishra. 2022. *Application-aware Networking (APN) Framework*. Internet-Draft draft-li-apn-framework-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-li-apn-framework/06/ Work in Progress.

[McCauley et al.(2019)] James McCauley, Yotam Harchol, Aurojit Panda, Barath Raghavan, and Scott Shenker. 2019. Enabling a Permanent Revolution in Internet Architecture. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) *(SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3341302.3342075

[Peng et al.(2023)] Shuping Peng, Zhenbin Li, Chongfeng Xie, Zhuangzhuang Qin, and Gyan Mishra. 2023. *Operational Issues with Processing of the Hop-by-Hop Options Header*. Internet-Draft draft-ietf-v6ops-hbh-03. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-v6ops-hbh/03/ Work in Progress.

[Quinn et al.(2018)] Paul Quinn, Uri Elzur, and Carlos Pignataro. 2018. Network Service Header (NSH). RFC 8300. https://doi.org/10.17487/RFC8300

[Quinn and Nadeau(2015)] Paul Quinn and Thomas Nadeau. 2015. Problem Statement for Service Function Chaining. RFC 7498. https://doi.org/10.17487/RFC7498

[Rescorla et al.(2022)] Eric Rescorla, Hannes Tschofenig, and Nagendra Modadugu. 2022. The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. RFC 9147. https://doi.org/10.17487/RFC9147

[Salo and Zhang(2020)] Timothy J. Salo and Zhi-Li Zhang. 2020. Semantically Aware, Mission-Oriented (SAMO) Networks: A Framework for Application/Network Integration. In *Proceedings of the Workshop on Network Application Integration/CoDesign* (Virtual Event, USA) *(NAI '20)*. Association for Computing Machinery, New York, NY, USA, 41–42. https://doi.org/10.1145/3405672.3409490

[Seedorf et al.(2009)] J. Seedorf, S. Kiesel, and M. Stiemerling. 2009. Traffic localization for P2P-applications: The ALTO approach. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*. IEEE, Piscataway, NJ, 171–177.