Benchmarking of Emulated Wireless Edge Cloud Connectivity for Maritime Environments

Antti Kolehmainen*, Miika Komu†, Sepehr Javid†, Jimmy Kjällman†,
Tero Kauppinen†, Fayezeh Ghavimi†, Bilhanan Silverajan*
*Tampere University
Finland
firstname.lastname@tuni.fi
†Ericsson Research
Finland
firstname.lastname@ericsson.com

***Abstract*—Remote ship pilotage, smart fairway navigation, autonomous maritime transport and smart maritime logistics present new challenges to the maritime industry. One of the challenges is increasing dependency on reliable network connectivity that can range from low priority entertainment communications to high priority safety critical communications. Traffic prioritization is needed at the open sea, where ships typically resort to expensive satellite connectivity that incurs long latency and narrow bandwidth. Near the coastline, ships can switch to 5G-based communications with low latency and high bandwidth, and possibly even utilize low-latency edge computing capabilities. Our contribution in this paper is a cloud-based networking testbed that supports network traffic shaping to emulate satellite and 5G-based communications between an edge cloud located onboard a ship and another edge cloud located onboard another ship or at the shore under traffic loss scenarios. Potential use-cases for our work include emulation of peer-to-peer communications (i.e., ship-to-ship or ship-to-shore) and mesh networking. The system can also be used as a basis for real-time ship networking, for instance, as building block for traffic prioritization in Kubernetes-based edge clouds.**

***Index Terms*—maritime; networking; 5G; satellite; Kubernetes; edge cloud**

I. Introduction

Maritime industry is slowly being digitalized due to new emerging use cases. For example, remote ship pilotage could be common in near future, especially when a ship arrives at or leaves the harbor. In such a case, the harbor pilot does not have to physically board the ship but could rather operate the ship remotely. Of course, a natural extension to this are completely autonomous ships, for which, the biggest obstacles are still perhaps related to maritime laws that do not allow yet unmanned ships. As another use case, smart fairway navigation could be used, for instance, for increasing ship safety to avoid collisions using sensor fusion but also to optimize the speed of ships to avoid congestion at harbors. Finally, transport logistics chains can be optimized according to the ship arrival at harbor, and to minimize the ship turnaround time.

One critical factor in all of the maritime digitalization use cases is network connectivity. At the open sea, satellite connectivity is the main option but it is typically avoided because it can be rather expensive. Ships usually utilize terrestial cellular connnectivity at the proximity of a harbor. The two different connectivity types have very different characteristics. For example, 5G connectivity offers high bandwidth and low latency as shown in table I. In contrast, satellite connectivity typically is quite the opposite as shown in table II, and thus may even require traffic classification and prioritization. Similarly to satellite networking, ship mesh networking may also incur long latency, and is not yet commonplace.

To digitalize ships and harbors for automated or remote pilotage, some IT infrastructure needs to be placed both onboard the ships and at harbors. We envision that such infrastructure consists of edge cloud infrastructure that can be seamlessly connected together at the infrastructure level rather than using some narrow APIs for remote monitoring and control. This way, a ship can utilize directly the computational resources of the harbor or remote control center, for example, to enable AI-assisted or completely AI-operated docking sequence of the ship.

In this paper, we emulate ship-to-ship or ship-to-shore edge cloud networking to understand how it behaves under wireless conditions. Contrary to datacenter internal or even datacenter-to-datacenter networking scenarios, we also consider packet loss scenarios occurring due to wireless connectivity. We mostly focus on emulating of different network profiles with cloud external tools but we also provide some insight to cloud internal tools. As Kubernetes is the industrial de facto standard, we are also utilizing it to provide the cloud infrastructure in our experiments. Our benchmarks are limited throughput measurements because latency is not usually the main issue with slowly moving ships.

City	Operator	Download (Mbps)	Upload (Mbps)	Latency (ms)
Cardiff	Vodafone	181.96	17.74	20
Cardiff	EE	168.47	18.27	32
Edinburgh	EE	168.97	11.47	42
London	O2	127.98	8.00	26

Table I
5G Performance in UK Capital Cities (Adapted from [1])

Provider	Download (Mbps)	Upload (Mbps)	Latency (ms)
Starlink	97.23	13.89	45
HughesNet	19.73	2.43	724
Viasat	18.13	3.38	630

Table II
SATELLITE INTERNET PERFORMANCE IN THE US (ADAPTED FROM [2])

II. RELATED WORK

We utilize three alternative network solutions for Kubernetes in this paper. We have earlier benchmarked and compared two of them, Network Service Mesh (NSM) and Calico, in high speed networking scenarios in multi-cluster environment [3]. The mentioned publication describes multi-cloud networking in NSM in more detail, and will not be covered in this paper because we utilize NSM based connectivity here as a simple bitpipe. Others [4] have compared Calico and Cilium in multi-cluster environments. Compared to the prior art, our contribution involves benchmarking Kubernetes networking in packet loss and traffic shaping scenarios.

Federated Kubernetes (KubeFed) allows joining multiple Kubernetes clusters into a single logical cluster. KubeFed supports only starting and terminating of Linux containers in different Kubernetes clusters but it does not support network connectivity between the containers across cluster boundaries. For this, cross-cluster network connectivity solutions, such as NSM or Cilium, are required for Kubernetes. We have qualitatively analyzed some of these cross-cluster solutions in our earlier work [3]. KubeFed is not analyzed further in this paper because the focus will be on cross-cluster networking.

Kidston et al [5] model and simulate ship networking for traffic engineering purposes in a naval scenario where ships communicate either directly with each other over radio or over a satellite link, relayed by a command center. The traffic engineering consists of monitoring health of the maritime network, traffic prioritization into different quality of service classes using Differential Services, and adaptive routing to switch into a more suitable communication link. Our work differs from their work because we are emulating wireless network with wired but the rest of the infrastructure in our work is "real", whereas their work is based on simulations. We also provide insight into Kubernetes networking and 5G traffic emulation. Our work complements their original and solid work, and we believe that combining their traffic classification ideas with the modern traffic shapers we have experimented with would be a good match.

Dummynet is a link emulator found natively in FreeBSD and macOS [6]. It allows shaping of network traffic by feeding packets through ”pipes” for which bandwidth parameters can be set. Parameters such as bandwidth, delay and Packet Loss Rate (PLR) together can be used to emulate different types of link conditions combined with packet scheduling and queue management. Packets can be driven through multiple pipes with dynamic rulesets based on, e.g., source, destination and random probabilities. Alternative link emulation and traffic shaping tools for Linux are Traffic Control (TC) and Netem [7] and the Extended Berkeley Packet Filter (eBPF) [8].

There also has been some research on TCP performance over satellite links that discusses some potential issues relating to e.g. TCP’s slow start mechanism, link asymmetries and congestion avoidance [9]. Henderson et. al also discovered that Geostationary satellite (GEO) -based connections are not as good performing as Low Earth Orbit (LEO)-based ones due the high Round Trip Time (RTT). Our measurements should also confirm this even without emulating the whole satellite system.

III. TEST ENVIRONMENT

The test environment consists of three different Kubernetes (version 1.21.8) clusters running on Ubuntu 20.04.3 LTS. Two of the clusters are virtualized on top of SmartOS (Release 20220421T000508Z) hypervisors and one is running on "bare metal" on a set of four physical Intel Atom-based servers. One additional server is located between the physical cluster and the virtualized ones, and the server acts as a link emulator. NSM based connectivity is set up automatically prior to measurements, so NSM v0.2 acts as a VXLAN based bitpipe from the viewpoint of measurements. On both Kubernetes and host side, Maximum Transmission Unit (MTU) is lowered from 1500 to 1440 due to the extra overhead in VXLAN tunnel headers.

A. Virtual Machines

The virtual machines are running inside an illumos zone, which is a jail-like [1]lightweight virtualization technology inherited from SUN Solaris[10], on top of type-2 hypervisor called ”bhyve”[11] on the SmartOS servers. The Kubernetes clusters are deployed on two separate physical SmartOS servers and consist of four virtual machine nodes each. The servers are not dedicated only to this project and thus also include unrelated virtual machines that consume the total resources available. The version of Ubuntu in use is the 20.04.3 LTS that was first installed from the official Joyent provided *ubuntu-certified-18.04* image as a bhyve VM and then upgraded to the 20.04.3 LTS version. We opt to use bhyve instead of KVM as bhyve is the better supported Hardware Virtual Machine (HVM) system on the illumos platform.

B. Physical Machines

The physical machines are fully dedicated to run the Kubernetes clusters and are installed on of Ubuntu 20.04.3 LTS as well. Furthermore, each of the bare metal Atom servers are identical in hardware configuration running on Intel Atom C3558 CPU with 64 Gigabytes of RAM.

C. Network Setup

The network between the SmartOS hypervisors is constructed via 10 Gigabit trunks with 10 Gigabit routing on them 1. The networking inside the hypervisors uses the illumos Crossbow network virtualization, another technology inherited

[1]https://www.usenix.org/system/files/login/articles/1085-mckusick.pdf

from SUN Solaris[12], that provides fully virtualized network stacks to each of the VMs. The uplink connection between the SmartOS hypervisors and the link emulator is a one gigabit connection. Furthermore, the downlink connection from the link emulator to the Atom server is one gigabit. The final piece of the network setup is the use of NSM as the Kubernetes network plugin through which all our emulated traffic flows.

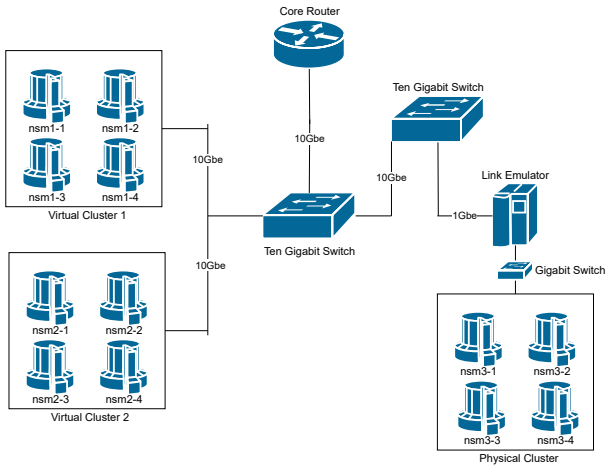

Figure 1. Network Setup

D. Link emulator

The link Emulator is a physical desktop computer running FreeBSD 13.0-RELEASE-p11. The server itself has Intel Core i7-6700 CPU with 32 Gigabytes of memory. The network connection to the Atom cluster and to upstream is established via quad port Intel PRO/1000 Gigabit Network Interface Card (NIC). Network emulation is performed by FreeBSD's integrated ipfw with dummynet[6] set up as a bridge on the quad port NIC. The dummynet has also been set up with two different pipes for tuning the downlink and uplink parameters independently.

E. Kubernetes Internal Traffic Shaper

Our experimentation with Kubernetes internal tools are tested in a different environment than in the previous sections. While the previous test cases utilize NSM framework, it does not yet support for traffic shaping. However, two other popular network plugins for Kubernetes, Calico and Cilium, do have bandwidth shaping capabilities. We benchmark Calico 3.23.2 and Cilium 1.11.6. Contrary to the other benchmarks, we utilize only a single Kubernetes cluster, due to the inter-cluster communication issues with Calico. The cluster consists of one master node and two worker nodes, each running in their separate VMs, based on KVM virtualization and operated in an OpenStack environment (Pike release).

IV. Throughput Measurements

For the measurements we opt to emulate two different scenarios. In the first scenario, the ship is at or near a port with fast 5G connection. In the second scenario, the ship is underway and must utilize satellite connectivity. In addition to link bandwidth we also emulate scenarios where the link quality is deteriorated. The deterioration of the link is achieved by varying dummynet PLR value between 0 and 0.1 in 0.01 increments. The PLR is a floating-point value between 0 and 1 where 0 is 0% and 1 is 100% packet loss. In addition, the network setup is also designed to some extent emulate a ship-to-shore (or ship-to-ship) communications where the physical cluster acts a ship connected to the cluster running on the VMs acting as the datacenter at the shore.

The measurements are conducted with iperf3 as the packet source, connecting from NSM3-1 client to destination server running on the NSM1-1 VM. The iperf measurement is taken with 20 second interval and total measurement time of 60 seconds per service data rate (or bandwidth in iperf terms) for each of the PLR values. The command line parameters used in the measurement are shown in listing 1.

Listing 1. benchmarking script

```
for bitrate in 10K 100K 500K 1000K 1500K 2000K
do
    iperf3 -i 20 -t 60 -b \${bitrate} -c \${
        serverip} -f k
    sleep 5
done
```

The measurement is conducted utilizing NSM based connectivity both on the Ubuntu host machine as well as on the Kubernetes node. The host measurement is used as a baseline to evaluate the impact of Kubernetes networking on the throughput performance.

A. Service Data Rates

Maritime Radio Communications Plan (MRCP) developed by International Association of Marine Aids to Navigation and Lighthouse Authorities (IALA) classifies maritime services into three different categories [13]:

- **Safety Services** (e.g. Radar, GMDSS, LiDAR, SAR)
- **Operational Services** (e.g. Weather, Chart Updates, Ship reporting)
- **Commercial Services** (e.g. Cargo Telemetry, VoIP, Infotainment)

Out of these data services the ones we have selected to emulate are demonstrated in table III.

Service	Data Rate
GMDSS	10 Kbps
Radar/AIS	100 Kbps
Infotainment	1500 Kbps
LIDAR	2000 Kbps

Table III
Bandwidth used by Data Service

In addition to listed service data rates we have conducted measurements for 1000 Kbps and 500 Kbps to investigate more precisely the breaking point where bandwidth limitations have adverse effects on the link quality.

For the link emulation two 5G links were selected based on values measured by others [14], [15]. The values are rounded

to an even number and round trip latencies are divided by two to provide the same value for both uplink and downlink.

As for maritime satellite providers, the testbed environment is configured to emulate a GEO satellite network. The bandwidth for the first link and a typical delay for such network are taken from other sources [16], [17] respectively. For the second satellite link, Ookla measurements are utilized as the reference value (Table II). According to the mentioned references, Table IV summarizes the parameters selected for the dummynet.

Service	Uplink (Mbps)	Downlink (Mbps)	Latency up/down (ms)
Satellite #1	5	30	125/125
Satellite #2	3	18	315/315
5G #1	88	535	6/6
5G #2	29	592	13/13

Table IV
DUMMYNET LINK PARAMETERS

B. 5G Link

From the measurements, negligible decays are observed regarding 10 Kbps, 100 Kbps and 500 Kbps data rates even at the highest packet loss value related to the selected dummynet link parameters. Due to this observation we omit them from the final figures.

Comparing baselines of the two 5G links to their respective NSM measurements also demonstrates that there is not much added overhead from Kubernetes and NSM as shown in, e.g., 5G #1 baseline (Figure 2) and NSM measurement (Figure 3). As 5G #1 is lower latency link we notice fairly consistent throughput until around 0.07 PLR when the link quality starts to decay.

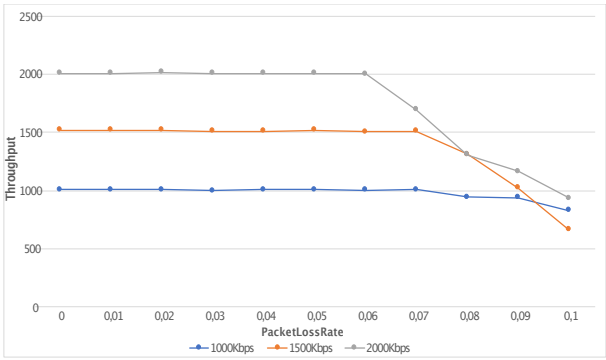

Figure 2. Baseline Measurement for 5G #1

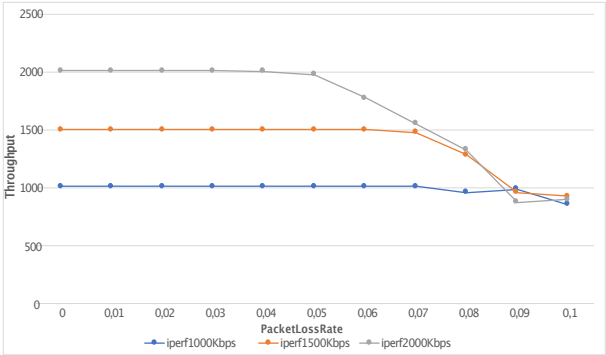

Figure 3. Throughput Decay on NSM 5G #1

The effect of higher latency can be noticeably observed on both the 5G #2 baseline (Figure 4) and NSM (Figure 5) results. The throughput decay already begins at PLR 0.03 and, as with 5G #1, it is the highest service data rate that decays the most. It is also noticeable that from PLR 0.06 onward the decay on all of the service data rates seems to be fairly similar, especially on the NSM side.

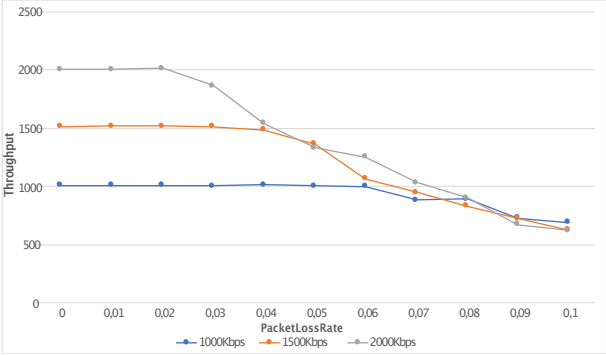

Figure 4. Baseline Measurement for 5G #2

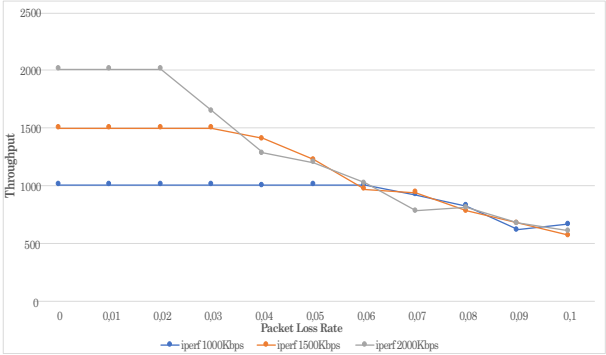

Figure 5. Throughput Decay on NSM 5G #2

C. Satellite Link

Different from the 5G measurement results, the satellite links also demonstrate decay at the 500 Kbps service data rate (e.g. Figures 6 and 8), thus this data rate is also taken into consideration. Similar to the 5G results, the 10 Kbps and

100 Kbps results remain quite even up to the highest measured packet loss value, thus they are omitted from the figures.

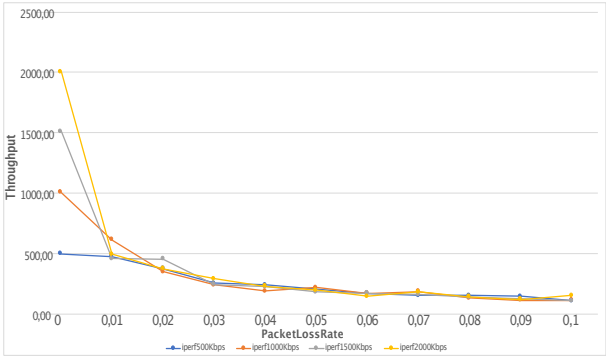

Figure 6. Baseline Measurement for Satellite #1

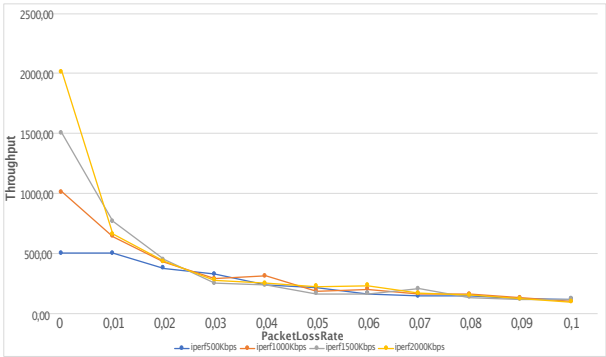

Figure 7. Throughput decay on NSM Satellite #1

Comparing the results of the satellite measurements we once again notice no significant difference between the baseline host measurement and NSM on Kubernetes (i.e., in Figures 8 and 9). However, comparing the result to both 5G ones there is already significant decay in link quality already starting from PLR value 0.01. Furthermore, it can be observed that the links converge a lot earlier, around PLR 0.03/0.04 unlike with 5G measurements where the throughput was kept up for a longer time.

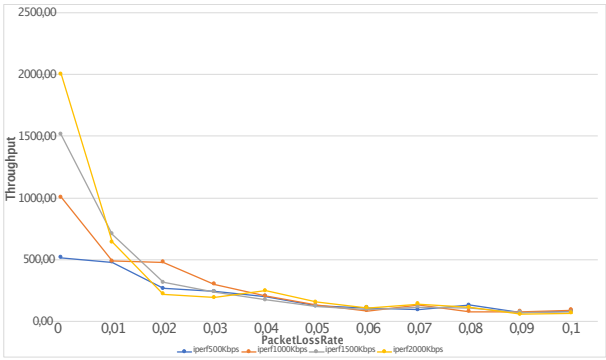

Figure 8. Baseline Measurement for Satellite #2

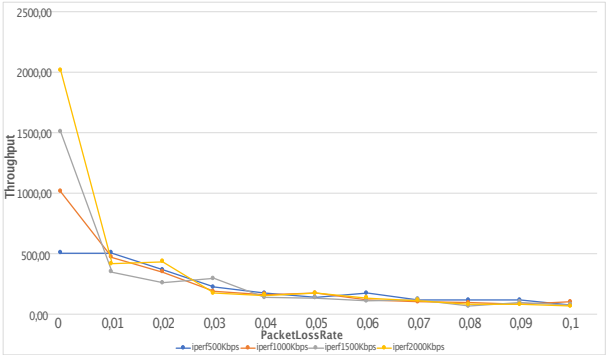

Figure 9. Throughput decay on NSM Satellite #2

As we already noticed between the 5G measurements (Figures 3 and 5), the effect of higher latency causes fairly dramatic decay on the throughput. This is even more evident on the emulated satellites (Figure 7 and 9) as we specifically emulated GEO satellite links that come with very high RTTs, 250ms and 630ms, in our case, because GEO satellites still remain dominant in maritime scenarios.

In addition, it can be said that the effect of the available upload and download bandwidth is not as critical in our scenarios as our current traffic profiles are only using up to 2Mbit/s throughput.

D. Kubernetes Internal Traffic Shaping Measurements

In addition to the Kubernetes external traffic shaping mechanism described in previous sections, we have also briefly experimented with two existing Kubernetes internal traffic shapers. In particular, these traffic shapers could be utilized for adapting traffic from Kubernetes-based cloud and edge environments to available link characteristics, for example, to prioritize certain traffic when link capacity is limited or expensive. We discovered two alternatives for Kubernetes, Calico and Cilium network plugins, that can support traffic shaping.

Calico supports bandwidth management using the Kubernetes experimental bandwidth plugin [2] based on Linux traffic control (TC), whereas Cilium [3] is based on Linux eBPF. Calico allows setting bandwidth limits to both inbound and outbound traffic for a pod, whereas Cilium supports only outbound direction. Of the these two, only Cilium supports inter-cluster connectivity out-of-the-box.

In the experiments, we ran iperf3 between two pods located on different worker nodes, and we capped the maximum bandwidth of the traffic between the pods to 10 Mbit/s because we observed that Calico did not function reliably with lower values than this. The advantage of Calico was the ability to control both inbound and outbound bandwidth cap for a single pod. A disadvantage was that Calico ignored both inbound and outbound bandwidth limits for the first 5-7 TCP connections, during which Calico stabilized slowly and then

[2]https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/

[3]https://docs.cilium.io/en/v1.9/gettingstarted/bandwidth-manager/

started to respect the bandwidth cap (with standard deviation in the range of 0.8 - 1.4 for the outbound direction). On the other hand, Cilium only allowed to set the bandwidth cap for outbound direction for a single pod. For outbound traffic, Cilium exceeded the bandwidth cap by 2.6 Mbit/s only for the first TCP connection, and the standard deviation was ranging between 0.5 - 3.1 Mbit/s. The results are shown in Figure 10.

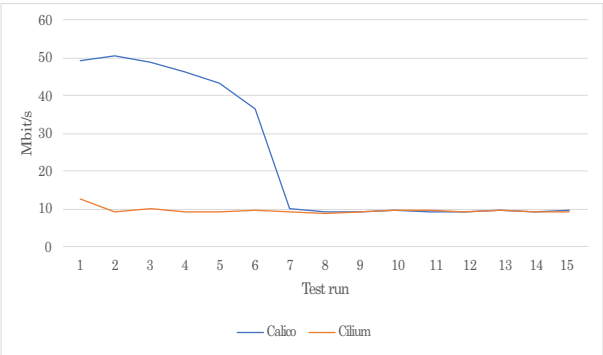

Figure 10. Egress bandwidth shaping in Calico and Cilium

V. Conclusion and Future Work

In this paper we presented a cloud based networking testbed that supports network traffic shaping to emulate satellite and 5G-based connectivity under traffic loss scenarios between a Kubernetes based edge cloud located onboard a ship and another edge cloud located onboard another ship or at the shore. Based on our experiments, we discovered that the utilized Kubernetes network plugin, Network Service Mesh, does not add significant overhead to the links when compared to non-Kubernetes traffic with the ship traffic profiles that we experimented with. It appears that increase in latency correlates negatively with throughput in the case of TCP; only the lowest TCP based traffic classes of 10Kbps and 100Kpbs were resilient enough to persist the whole spectrum of packet loss rates with only minor decay.

As future work, we should stress test the testbed and record CPU usage, and conduct more experiments with different traffic patterns, such as with bursty traffic. To support real edge networks in ships, traffic originating from many different, parallel data services should also be considered which leads into traffic prioritization questions. Hence, we would also like to extend our link emulator to support traffic engineering at the level of individual streams. However, traffic characterization at a session level might be an issue as many modern applications encrypt their traffic nowadays. As a solution, the traffic could either be characterized by the application itself running in the edge cluster, for example, using Differentiated Services, or by the underlying Kubernetes platform since it can be made aware what types of applications it is running. For the latter case, we discovered some problems and limitations in the experimented Kubernetes network plugins. More work is required to allow for changing the bandwidth limits more dynamically, and to automatically profile and prioritize ship networking traffic. Finally, we also believe our testbed and the benchmarking results could be used to emulate network connectivity of digital twins of ships.

VI. Acknowledgements

This project has partially been funded by DIMECC Sea4Value FFN project.

References

[1] Ookla, "The State of Mobile 5G in the United Kingdom," [Accessed 8-June-2022]. [Online]. Available: https://www.ookla.com/articles/5g-united-kingdom-2019

[2] ——, "How Starlink's Satellite Internet Stacks Up Against HughesNet and Viasat around the Globe," [Accessed 8-June-2022]. [Online]. Available: https://www.ookla.com/articles/starlink-hughesnet-viasat-performance-q2-2021

[3] L. Osmani, T. Kauppinen, M. Komu, and S. Tarkoma, "Multi-cloud connectivity for kubernetes in 5g networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 42–47, 2021.

[4] G. Budigiri, C. Baumann, J. T. Mühlberg, E. Truyen, and W. Joosen, "Network policies in kubernetes: Performance evaluation and security analysis," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2021, pp. 407–412.

[5] D. Kidston and T. Kunz, "Using simulation to evaluate traffic engineering management services in maritime networks," in *Proceedings of the 2008 Spring Simulation Multiconference*, ser. SpringSim '08. San Diego, CA, USA: Society for Computer Simulation International, 2008, pp. 699–706.

[6] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 12–20, 2010.

[7] L. Nussbaum and O. Richard, "A comparative study of network link emulators," in *Communications and Networking Simulation Symposium (CNS'09)*, 2009.

[8] D. Scholz, D. Raumer, P. Emmerich, A. Kurtz, K. Lesiak, and G. Carle, "Performance implications of packet filtering with linux ebpf," in *2018 30th International Teletraffic Congress (ITC 30)*, vol. 1. IEEE, 2018, pp. 209–217.

[9] T. R. Henderson and R. H. Katz, "Tcp performance over satellite channels," in *Internetworking and Computing Over Satellite Networks*. Springer, 2003, pp. 131–157.

[10] D. Price and A. Tucker, "Solaris zones: Operating system support for consolidating commercial workloads." in *USENIX LISA*, vol. 4, 2004, pp. 241–254.

[11] "Reintroducing Bhyve," https://tritondatacenter.com/blog/reintroducing-bhyve, [Accessed 11-July-2022]".

[12] N. Droux, S. Tripathi, K. Belgaied, and S. Khare, "Crossbow virtual wire: Network in a box," in *USENIX LISA*, 2009.

[13] S.-W. Jo and W.-S. Shim, "Lte-maritime: High-speed maritime wireless communication based on lte technology," *IEEE Access*, vol. 7, pp. 53 172–53 181, 2019.

[14] O. D. S. Al-Gburi, "General overview of 4g and 5g with field measurements and performance comparison," *Master's Thesis*, 2021, [Accessed 11-June-2022]. [Online]. Available: https://trepo.tuni.fi/handle/10024/125111

[15] A. Jürgens, "5g-nr measurements campaign and connectivity analysis," *Master's Thesis*, 2021, [Accessed 5-July-2022]. [Online]. Available: https://digikogu.taltech.ee/et/Download/a3e91679-9968-4c76-884a-420c1c184aae

[16] J. Deutschmann, T. Heyn, C. Rohde, K.-S. Hielscher, and R. German, "Broadband internet access via satellite: State-of-the-art and future directions," in *Broadband Coverage in Germany; 15th ITG-Symposium*, 2021, pp. 1–7.

[17] H. Hu, H. Li, and Y. Liu, "Bandwidth and round-trip time detection based congestion control for multipath tcp over highly lossy satellite networks," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2018, pp. 1072–1075.