Software Defined Naval Network for Satellite Communications (SDN-SAT)

Sobhan Nazari[*‡], Pengyuan Du[*], Mario Gerla[*], Ceilidh Hoffmann[†], Jae H. Kim[†], Antonio Capone[‡]
[*] *Department of Computer Science, University of California, Los Angeles, USA*
[†] *Boeing Research & Technology, Seattle, WA 98124-2207*
[‡] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy*
{sobhannazari, pengyuandu}@ucla.edu, gerla@cs.ucla.edu,
{ceilidh.hoffmann2, jae.h.kim}@boeing.com, antonio.capone@polimi.it

***Abstract*—We propose Software Defined Networking (SDN) framework to a fleet of naval ships that rely on multiple satellite communication systems for onboard communication. Our solution addresses practical issues in current shipboard naval networks such as sharing and load balancing of multiple satellite communication links as well as overcoming limited bandwidth constraints. To ameliorate link intermittence and outage, we propose Multipath Transmission Control Protocol (MPTCP), which improves end-to-end data delivery by creating several subflows under one TCP session. In our SDN framework, each ship is an SDN switch with multiple SATCOM connections. The management and classification of MPTCP subflows is handled by a remote SDN controller. The cooperation between MPTCP and SDN controller leads to an agile, bandwidth efficient, robust naval network. System analysis and numerical evaluation validate the feasibility and efficacy of our SDN-based solution for such a network.**

***Index Terms*— Naval Ship Network, Multipath TCP, Software Defined Networking (SDN), Satellite Communication (SATCOM)**

I. INTRODUCTION

The naval surface fleet (aircraft carriers, destroyers, cruisers etc. [19]) relies on a combination of government and commercial satellite communication systems (SATCOM) for ship-to-ship, ship-to-shore and ship-to-aircraft communication. On board each ship, there are multiple SATCOM terminals and enclaves of end-user local area networks (LAN) interconnected via a common service router. In the current service model, reliability issues such as network outages and high packet loss rates due to capacity saturation and/or non-optimal flow balancing are ameliorated by leasing additional commercial SATCOM capacity or by investing in larger capacity satellite terminals for certain individual vessels.

On the SATCOM side, each military or commercial satellite system has its own communication service options, operating principles and management assets. Navy's combined SATCOM network is mostly composed of geosynchronous satellites with the inclusion of low earth orbit (LEO) satellite links on some vessels.

In the current shipboard architecture [1], the clients inside the ship are served by a specific satellite system based on the type of communication service requested (e.g. circuit-switched voice versus IP data) or mission thread type. In other words, different satellite terminals provide segregated connections to different user classes on the ship. We can thus view a shipboard network as a collection of several independent sub-networks, each feeding to and from a different satellite terminal. Consequently, a user can experience congestion due to overloading of traffic in a subnet it is assigned to while the capacity in adjacent satellite channels is under-utilized.

Current navy's shipboard networks rely on enterprise-grade IP service hardware routers that maintain tight vertical integration between control and data plane functions. Furthermore, both hardware and software operations are intertwined via a vendor's proprietary operating system. Therefore, dynamic re-configuration of networking rules is limited, hindering enhancements and adoption of new innovative solutions as these networks evolve after their initial deployment.

To tackle above-mentioned constraints, we propose Software Defined Networking framework for a network of naval ships that relies on SATCOM as its primary communication service. Such a network is termed *Software-Defined Naval Network using SATCOM services* (*SDN-SAT*). This approach brings forward SDN-based optimization solutions such as efficient bandwidth allocation, multiple survivable paths and differentiated quality of service (QoS) measures.

Technologies such as Multipath TCP (MPTCP) [2] and OpenFlow [3, 4] have been shown to provide improved data transport mechanisms over a Wide Area Network. In MPTCP an application's TCP connection is split into multiple TCP subflows based on the configured IP interfaces of the connection-initiating host and the remote partner. Within the operating system kernel, MPTCP manages the flows on the sender's side and is also responsible for the recombination at the destination. The MPTCP concept is transparent to both the network below and the applications above [5].

OpenFlow is a control plane protocol used between a network device such as a switch and a remote controller where the execution code runs on an adjacent server. When the switch receives a packet from a new flow, it forwards the packet to the controller. The controller

inspects the packet, decides what the appropriate action is and instructs the switch to execute this action for all packets of this particular flow [6].

MPTCP and OpenFlow are complementary and naturally suited to work in tandem since MPTCP is implemented by the end hosts while each of the multiple paths relies on OpenFlow for forwarding and routing. In this paper we first address the question of how these two approaches coexist and are synergized in an operational production network. Second, we elaborate on the benefits of such an approach when implemented over a naval ship network that connects to multiple SATCOM systems.

The outline of our paper is as follows: In Section II, a brief review and background information on Software Defined Networking and Multipath TCP are presented. In Section III, we review the current naval network architecture which will serve as the baseline network model prior to SDN-based modification. Section IV is dedicated to the proposed Software Defined architecture for navy fleet network (SDN-SAT). Section V covers performance analysis and numerical evaluation of our proposed solutions. In Section VI we conclude with a summary of our main results and an outline of ongoing and future work in this area.

II. BACKGROUND

A. *Software Defined Networking (SDN)*

Software defined networking (SDN) is an emerging concept that promises to overcome the bundling between control plane (that decides how to handle network traffic) and data plane (that forwards traffic according to the decisions made by the control plane) in traditional ossified [7] IP networks by breaking the so-called vertical integration and removing the network control functions from the underlying routers and switches, thereby promoting the logical centralization of network control and introducing the network programmability [8] [9].

The Open Networking Foundation (ONF) [10], a non-profit user driven organization, is dedicated to the promotion and the adoption of SDN through open standards development (e.g., OpenFlow [11]).

OpenFlow is an open interface for remotely controlling the forwarding tables in network switches, routers, and access points. Upon this low-level primitive, researchers can build networks with new high-level properties. For example, OpenFlow enables more secure default-off networks, wireless networks with smooth handoffs, scalable data center networks, host mobility, more energy-efficient networks and new wide-area networks – to name a few.

Centralized control, network programmability and the decoupling between control and data plane are the main principles that simplify network management, allowing administrators to dynamically manage network devices through abstraction layers, enabling innovations and changing the limitations of current infrastructures.

By separating the control and data planes, network routers and switches become simple forwarding devices. The control logic is implemented in a logically centralized controller, simplifying policy enforcement and network (re)configuration, evolution and scalability [12]. Forwarding decisions made by the data plane are flow-based, instead of destination-based. A flow is defined as a set of packet field values acting as a match (filter) criterion and a set of actions (instructions). The flow abstraction allows the classification of network traffic based on predefined matching rules, thus unifying the behavior of different types of network devices. Flow programming enables unprecedented flexibility, limited only to the capabilities of the implemented flow tables inside the forwarding devices [7].

When a flow is generated from an entry that was previously installed in the switch, no control traffic is generated. The packets immediately pass through the switch and a much lower delivery time is experienced for these packets of the flow. When the switch receives a packet from an unknown flow, it forwards the packet to the controller. The controller inspects the packet, decides what the appropriate action is and instructs the switch to execute this action for all packets of this particular flow. Fig. 1 shows a simplified view of the SDN architecture.

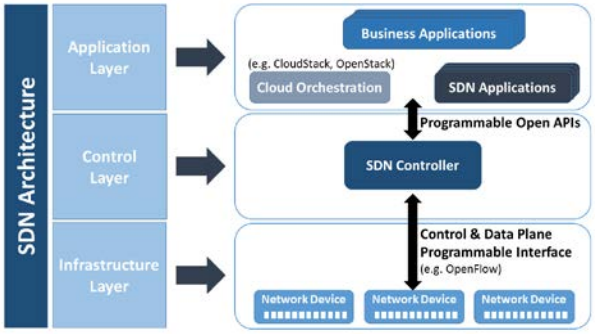

Figure 1: Overview of the SDN architecture

B. *Multipath Transmission Control Protocol (MPTCP)*

In this section we present the prerequisite details of MPTCP pertinent to the design of our SDN controller. More information about MPTCP can be found in [13]. Multipath TCP is an ongoing effort by the Internet Engineering Task Force's (IETF) Multipath TCP working group, aimed at allowing a TCP connection to use multiple paths to maximize resource usage and increase redundancy. MPTCP provides the ability to split a flow into multiple paths, thus providing better performance and resilience to failures.

Each MPTCP has a unique identifier, called *token*, which is used for the association/authentication of new subflows. A subflow is established like a regular TCP connection except that the handshake contains MPTCP-specific options such as MP_CAPABLE and MP_JOIN.

MP_CAPABLE is used to verify whether the remote host is MPTCP-enabled or not.

Fig. 2 illustrates the subflow establishment of an MPTCP connection. For the initial subflow (Fig. 2(a)), the hosts perform a handshake with the MP_CAPABLE option: it contains randomly generated keys which are used for the calculation of the token. For the establishment of additional subflows (Fig. 2(b)), the hosts perform a handshake with the MP_JOIN option as follows: The sender sends a SYN packet with the token and a random nonce. (The MP_CAPABLE and MP_JOIN SYN packets are the subflow setup packets). Upon reception of the packet, the receiver responds with a SYN+ACK packet containing its own nonce and a calculated HMAC code of the sender's nonce. Finally, the sender sends an ACK packet that contains the calculated HMAC of the receiver's nonce. The HMACs are calculated using the keys exchanged during the MP_CAPABLE handshake and the nuances are exchanged in the MP_JOIN handshakes [14].

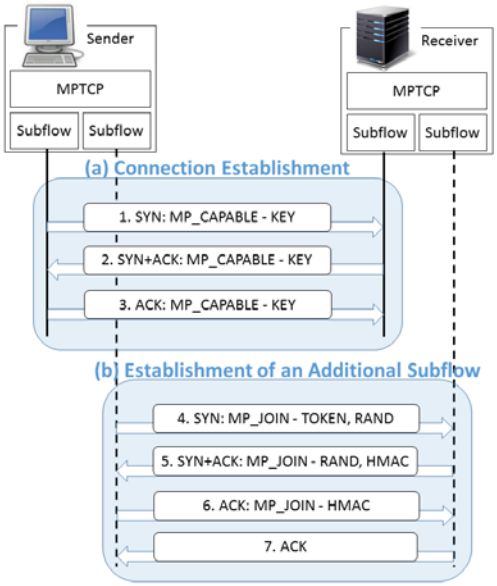

Figure 2: Subflow establishment in MPTCP

III. Baseline Naval SATCOM Network

The current naval SATCOM network, which we defined as the "baseline" is composed of two parts: 1) the naval ship fleet as "users" and 2) the SATCOM service provider network which includes all satellites within view of the ships and their associated earth stations. The current US naval fleet consists of less than 300 ships [19] of which we assume at most 100 are under the earth coverage area of a single geostationary satellite. For system analysis and modeling purposes, we assume there are three different satellites in view. The baseline Wide-Area Network architecture on board a naval ship is shown in Fig. 3. There are three main subsystems: local area network (LAN) enclaves on the left side, multiple satellite terminals on the right side and a common router in the middle that interconnects them. Traffic flow is always between a LAN to a satellite terminal and vice versa. No traffic flows from one LAN to another; likewise, none from a terminal to another. For example, an IP datagram generated by a user in Low Priority LAN is forwarded to the router which then selects an appropriate SATCOM terminal port based on a pre-defined routing policy based on user application type, LAN ID, QoS value, security classification or some other indicator. Over-the-air transmission over the satellite link is in terms of data frames—in which a fraction or multiple IP datagrams are encapsulated—that conform to the link and physical layer protocols adopted by the SATCOM system. The reverse process is similar where the IP datagram is extracted from a SATCOM data frame and routed to a LAN user based on its IP address. To carry out SDN implementation to this baseline network, the following assumptions must hold: 1) A succession of IP datagrams as part of an application session can traverse across any available SATCOM link as long as it can be re-assembled at the other end, 2) There exists a control interface between the router and each SATCOM terminal such that the router has capacity statistics of each SATCOM link.

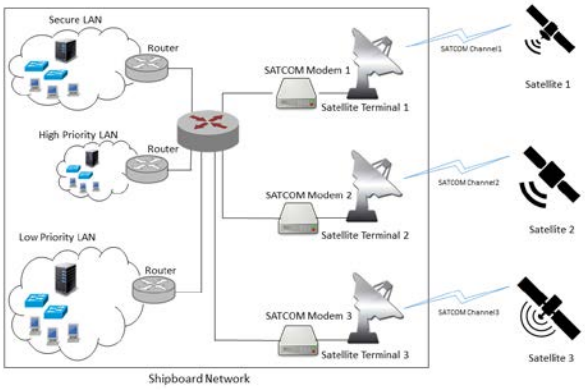

Figure 3: Architecture of Baseline Shipboard Network

IV. SDN Naval SATCOM Network (SDN-SAT)

Our proposed SDN-based naval SATCOM network adds two additional component to the baseline: shipboard SDN switch and the SDN controller network. Each ship is equipped with a SDN-enabled switch such as the Open vSwitch [15] as shown in Fig. 4. Compared to the baseline network of Fig. 3, the only differences are the replacement of the service router with an SDN switch and the requirement for a logical control channel (control API) between the shipboard switch and its designated SDN controller. At the data plane a SATCOM link is used to exchange control plane information between an SDN switch and its controller. As assumed in the previous section, every SATCOM terminal—in particular the link layer modem subsystem—will maintain control signal exchange with the SDN-enabled

switch. In particular, it has two main functional responsibilities: First, it interacts with the SATCOM service provider to broker and request SATCOM bandwidth based on the utilization statistics provided by its network switch. Second, it communicates with its network switch to relay SATCOM bandwidth statistics that are achievable in the current or near-future time frames.

It is recommended that control traffic between SDN switches and their controller is transported via SATCOM links that are reliable and of high capacity such that they meet the Southbound Application Programming Interface (API) protocol requirements as defined in OpenFlow [2].

The SDN controller can reside on the ground or on a ship as long as it has connectivity to the same SATCOM links. Another design choice is the total number of SDN controllers for the entire system. At a minimum a single controller can communicate with all shipboard SDN switches. At the maximum an SDN controller is used for each SATCOM service, thereby creating a more reliable and redundant network of SDN controllers at different geographic locations. In this second option, each SATCOM-specific controller only needs to maintain a single SATCOM link that it is servicing, hence enabling the possibility of co-locating the SDN controller with the ground earth station of the SATCOM service provider. It is assumed that in the case of multiple SDN ground controllers, they are all inter-connected via reliable terrestrial links. To run global optimization algorithms across all SATCOM systems, a global controller must be designated, either via an election process amongst them or as a stand-alone unit with interconnection to other controllers.

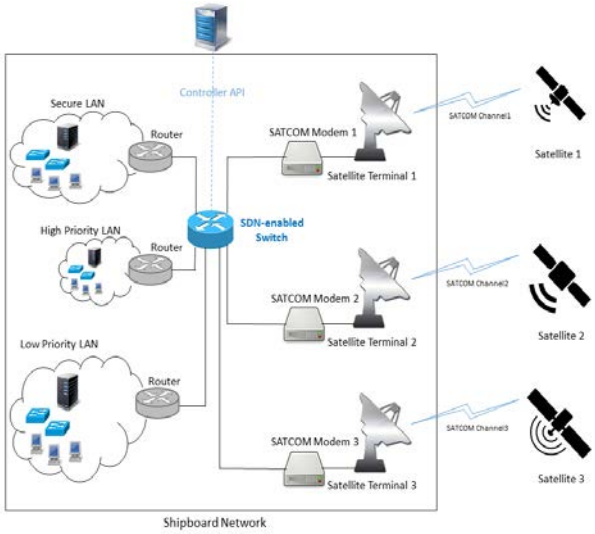

Figure 4: Architecture of SDN-based Shipboard Network

A. SDN Control Data Exchange

There are two control APIs: SDN switch-controller and SDN controller-controller. The former is known as Southbound interface while the latter is East-West interface. If all controllers are ground terminals, then East-West control overhead cost is negligible since data is transported via high speed terrestrial links. If a controller is co-located at the SATCOM provider's ground earth station, then control message propagation delay between SDN switch and controller is equal to twice the ground-to-space delay (~540 msec). This delay is doubled if the controller at a remote site.

SDN switch-controller data exchange uses over-the-air SATCOM bandwidth. In general, bandwidth overhead is proportional to the number of switches (i.e. ships) in the network—currently at 100, maximum—and the frequency of control exchange. Due to one-way propagation delay of at least 540 msec, switch-controller control exchange frequency is expected to be every second or longer. Each SDN switch reports router ingress and egress traffic statistics as well as SATCOM link capacities. Based on our estimates, SATCOM bandwidth overhead per ship is roughly 50 bytes/sec with 1-sec reporting rate. Likewise, the controller sends back load-balancing instructions as well as SATCOM capacity requests, which together is roughly 50 bytes/sec. It is assumed that control data messages can be multiplexed (in-band) with regular user data frames.

B. Multipath TCP for SDN-SAT

Multipath TCP allows each shipboard LAN user the option of maintaining several subflows within one TCP session. The number of subflows between each IP addresses pair can be set inside the MPTCP kernel of each LAN user host. The number of subflows for each individual user must be at least two. However, we claim that the optimum number of subflows for a LAN user on SDN-SAT with N satellite terminals to be N. Further explanation is provided in Section V.

C. SDN-SAT Controller Operation

Operational details of an SDN-SAT controller is as follows: First, it inspects incoming packets to a port of the SDN switch and assigns each MPTCP subflow of a MPTCP session to a different satellite link. This mechanism provides robustness for LAN clients since the connection is maintained via different SATCOM links. Even if the vessel loses one of its satellite assets, other links will remain alive without any interruption. Moreover, according to the congestion status of the satellite assets of a vessel, LAN clients can achieve the maximum available throughput which is equivalent to the sum of all the satellite terminals throughput.

The core of SDN-SAT controller (from now on being referred to simply as the "controller") is based on Floodlight [16], a well-known OpenFlow framework. Floodlight was chosen due to its modularity. The controller fulfills two main actions: First, it needs to recognize the subflows belong to an MPTCP instance.

Second, it needs to assign appropriate disjoint paths to such subflows. The process illustrated in Figure 2 is used as a basis for flow detection and grouping in a controller. Since all MPTCP information is encoded as TCP option (not headers), it is not possible to simply use OpenFlow's normal matching methods to identify MPTCP subflows. Therefore, a method to identify the subflows from the OpenFlow controller's perspective is necessary. In order to do so, special information beyond IP addresses and TCP port numbers are required. Fortunately, all MPTCP exchanges need information during the initial MPTCP subflow establishment (MP_CAPABLE) and subsequent subflows (MP_JOIN), as described in section II.B. A controller uses keys and tokens, which are unique for each connection and host, to find which subflows belong to which MPTCP connection. As the process of exchanging keys between hosts uses different IP address and TCP port pairs, an OpenFlow packet will be sent from an OpenFlow switch to the controller. The controller observes the MP_CAPABLE and MP_JOIN options to correctly identify MPTCP subflows.

The mechanisms of finding optimal path sets and deciding which path an MPTCP subflow should use are implemented with Path Finder and Path Allocation modules in the controller. The Path Finder component calculates and provides sets of paths between host interfaces to the Path Allocation module. The Path Allocation component is responsible for the selection of paths and the installation of the appropriate OpenFlow rules to the switches. These rules consist of header values to match packets and an associated action applied to packet matching. In this specific case, the matching header values are source-destination IP addresses and port numbers, which together identify packets belonging to the same subflow. The action is the selection of the port in the switch to forward the matching packets.

The Path Finder component has an up-to-date global view of the topology based on the exchanged messages between the controller and its switches (details of LinkDiscoveryManager module in [16]). The Path Finder component is queried by the Path Allocation whenever a set of paths between source-destination interfaces (IP addresses) is needed. The Path Finder uses the Depth First Search (DFS) graph traversal algorithm and finds all available paths. Subsequently, the Path Finder filters the obtained set of paths to extract a subset of edge-disjoint paths, which is returned to the Path Allocation component.

The Path Allocation assigns each subflow to a different path, which belongs to the obtained set of paths for the source-destination IP addresses of the MPTCP instance. The goal is to avoid assigning more than one subflow of the same connection to a single path. If the number of subflows is larger than the number of available paths, the subflows are deterministically assigned to paths in a uniform manner.

To route flows, the controller maintains a list of pending and connected sessions. When a new connection handshake message is received, it calculates a new path set and adds both the information of the connection initiation and the path set to the pending list. When a response from the pending connection is received, it calculates another path set for the reverse direction and moves everything to the connected sessions list. Any subsequent connections would only require a lookup in the connected session list to find an appropriate path. Path Allocation component keeps track of all available paths it can choose from to allocate to a new subflow by caching both the paths sets for each already requested source-destination IP addresses and current assigned paths of the set to subflows of an MPTCP instance. This information expires after 5 seconds as the cached information is useful only until the establishment of all the connection's subflows, which occurs at the beginning of the connection [17]. Switches also delete a rule from their table if they do not receive a packet matching rule after a specified time interval [16].

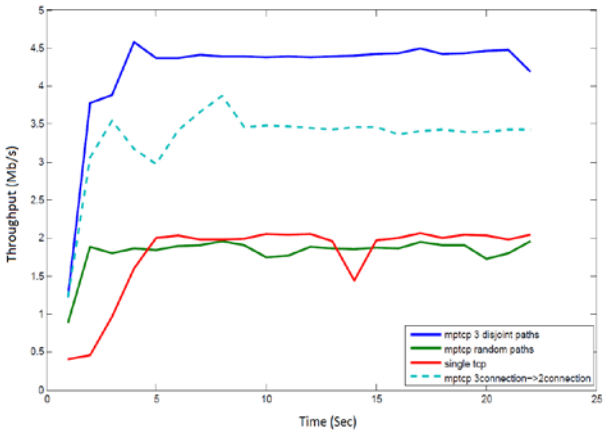

Figure 5: Comparison of SD-NSN throughput vs. other methodologies such as random forwarding, single TCP

V. SDN-SAT Performance Evaluation

For our tests, we run Mininet emulator [18] on a commodity laptop. The SDN-SAT controller protocol and the MPTCP Linux kernel v0.90 [17] are run on the same machine. For the emulation of the switches we use Open vSwitch [15].

As the first experiment to validate the capability of our proposed SDN-SAT architecture in boosting the clients' throughput, several ship nodes with the following SATCOM assets are created: Ship 1 and 2 have three satellite terminals with the following characteristics: SAT1 with 1.544 Mbps channel bandwidth, SAT2 with 1.544 Mbps, and SAT3 with 2.048 Mbps. Ship 3 is equipped with two satellite terminals with the following characteristics: SAT1 with 1.544 Mbps bandwidth, SAT2 with 1.544 Mbps. Since all the satellites terminals of the vessels in the experiments are geostationary satellites, the round trip SATCOM channel delays are considered

greater than 540 msec. In order to simplify the modeling of onboard LAN traffic flow, we consider one LAN user per ship for the entire network. Our results are still applicable when generalized to multiple LAN users per ship. In this case, SATCOM resources are distributed fairly (equal or proportional to demand) among them.

Fig. 5 shows TCP session throughputs between two users on two different ships. Each user runs a MPTCP kernel which creates three subflows for each MPTCP session, which is the maximum number of different satellite terminals for a ship in our experiment set up. The red line in Fig. 5 illustrates the throughput of a TCP connection between users in ship 1 and ship 2 when they are using conventional TCP. Since the maximum capacity among the SATCOM interfaces in ship 1 and ship 2 is offered by the SAT3 terminal with 2.048 Mbps typical bandwidth, the controller allocates SAT3 interface for this connection. Thus, the TCP connection throughput between these two users eventually reaches about 2 Mbps at steady state.

In another experiment, clients are using MPTCP with three subflows. Recall that the intelligence of the SDN_SAT controller is acquired from the SDN switches on the vessels' shipboard network. Even though on the client side, several subflows are generated for each MPTCP connection, the capability of deep inspection provided by the controller is ignored in this case. As a result, all the subflows belonging to a MPTCP session will be sent through the same satellite terminal and same channel. The green line in Fig. 5 shows the TCP connection throughput of the explained scenario.

In the next two experiments the controller architecture is applied and corresponding results are shown as blue and dashed lines in Fig. 5. Both users are using MPTCP to create different subflows for each MPTCP session. The designed Floodlight controller wisely inspects the first packet of the incoming flows to detect whether the flows belong to the same MPTCP session. Finally, based on packet inspection the controller assigns different satellite terminal capacity to each subflow of a MPTCP session. The blue line demonstrates the throughput of a connection between two navy users inside ship 1 and 2. Both ships 1 and 2 are equipped with three satellite terminals. To distribute internet traffic among different satellite systems, achieve load balance and provide disjoint paths for the subflows of a MPTCP session, the controller assigns each MPTCP subflow to a distinct satellite terminal. Theoretically, by this mechanism the throughput of this connection should reach the sum of the capacities of three satellite terminals in ship 1 or 2, which is 5.136 Mbps (=1.544+1.544+2.048), under no frame loss condition. As shown in Fig. 5, the connection throughput between these users are indeed close to the theoretical value.

The dashed line belongs to the throughput of users inside ships 2 and 3. As previously mentioned, ship 3 only has two satellite terminals with respective bandwidths of 1.544 and 2.048 Mbps. Although users run MPTCP to create 3 subflows per each MPTCP connection, the controller indeed cannot find three disjoint paths between these users. However, the SDN-SAT architecture always keeps the throughput of users to the sum of the available SATCOM physical resources, which theoretically in our experiments is 3,592 (=1.544+2.048), which closely matches with the obtained throughput from the experiment.

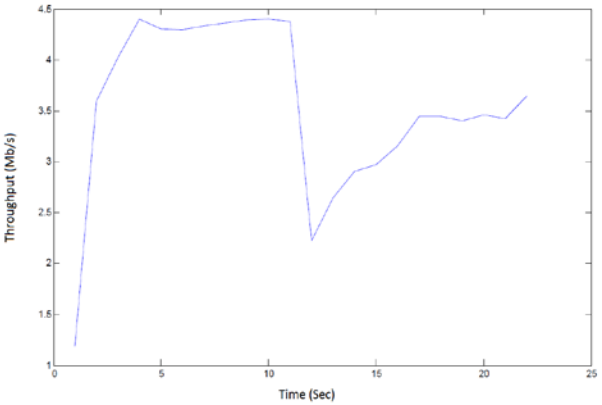

Figure 6: SD-NSN capability to handle losing a satellite terminal

In the next experiment we are measuring the performance of the SDN-SAT architecture in terms of robustness. It is possible that under certain scenarios (link outage, physical damage, network congestion etc.) a ship loses one of its satellite terminal operability. The two users, respectively in ship 1 and 2, are connected through a MPTCP session, and as described previously they have a throughput around 4.5 Mbps in steady state status. As shown in Fig. 6, we bring down the SAT1 terminal (with 1.544 Mbps bandwidth) at 11 second. Therefore, the user inside the ship 1 have access to only two operational satellite terminals. Thus, there are just two disjoints paths between the hosts in ship 1 and 2. The controller manages the situation in short order and users eventually resumes a connection with the sum of two other remaining operable satellite terminals.

VI. CONCLUSION

In this paper, we first discuss the benefits of combining SDN and MPTCP. System-level or operational constraints in existing naval shipboard network architecture may lead to under-utilization or non-optimal consumption of SATCOM services. Specifically, policy based routing solutions may preclude shipboard LAN users from sharing all available SATCOM links. We propose SDN-SAT, which is a Software Defined architecture for resource management among all naval vessels sharing the same set of SATCOM services. SDN-SAT offers all the advantages of a SDN-based network. Moreover, its capability of enabling deep packet

inspection provides an intelligent traffic engineering mechanism for MPTCP traffic generated by shipboard LAN user hosts. Our preliminary experiments show that 1) SDN-SAT can boost LAN users' connection throughput to the total capacity provided by the SATCOM terminals on each vessel and 2) the provided connection between users will be maintained steady and robust by SDN-SAT architecture even in cases when communication disruptions occur on a subset of available satellite terminals on a ship.

As final remarks, we highlight several topics not addressed in this paper but will be elaborated in future works. First, we have not addressed SDN and MPTCP performance when the underlying network consists of a large number of ship nodes. This problem requires detailed description of the resource allocation policies of all SATCOM service providers since we must account for the distribution of total capacity among all nodes involved. Thus it is outside the scope of current work. Second, in Section IV A. we give a rough estimate of control data overhead (~ 50 bytes per second) between an SDN switch and its controller. In future work, we plan to provide precise byte count and technical details of router and modem information that must be exchanged over the Southbound control API. Third, we have assumed the controller maintains reliable connectivity with its SDN switches even in scenarios where a subset of SATCOM links are disrupted. In future work we will describe a distributed SDN controller network architecture that can meet such reliability requirements through the use of multiple remote SDN controllers with integrated redundancy features.

ACKNOWLEDGMENT

This work was supported by the Office of Naval Research (ONR) Contract N00014-15-C-5038 Software Defined Naval Battlefield Network (SD-NBN). The authors would like to thank Dr. Santanu Das (ONR Program Manager) for his support and guidance. We also thank anonymous reviewers for their insightful and constructive comments.

REFERENCES

[1] K. Fiscko, "Shipboard Networks," AFCEA Publication, PEO C4I, US Navy, SPAWAR, 2008.

[2] T. J. Hacker, B. D. Athey, and B. Noble, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," in *Parallel and Distributed Processing Symposium., Proceedings Intl., IPDPS 2002,* 2001, pp. 10.

[3] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-defined networking (sdn): Layers and architecture terminology," 2070-1721, 2015.

[4] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: a survey," *ACM Computing Surveys (CSUR),* vol. 47, p. 27, 2015.

[5] S. Barré, O. Bonaventure, C. Raiciu, and M. Handley, "Experimenting with multipath TCP," *ACM SIGCOMM Computer Communication Review,* vol. 41, pp. 443-444, 2011.

[6] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *Communications Surveys & Tutorials, IEEE,* vol. 16, pp. 493-512, 2014.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford*, et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review,* vol. 38, pp. 69-74, 2008.

[8] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE,* vol. 103, pp. 14-76, 2015.

[9] B. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys & Tutorials, IEEE,* vol. 16, pp. 1617-1634, 2014.

[10] Open Networking Foundation [Online]. Available: https://www.opennetworking.org

[11] *OpenFlow*. Available: https://www.opennetworking.org/sdn-resources/openfow

[12] H. Kim and N. Feamster, "Improving network management with software defined networking," *Communications Magazine, IEEE,* vol. 51, pp. 114-119, 2013.

[13] M. Handley, O. Bonaventure, C. Raiciu, and A. Ford, "TCP extensions for multipath operation with multiple addresses," IETF RFC 6824, 2013.

[14] F. Gont, O. Bonaventure, C. Raiciu, and M. Bagnulo MP TCP, IETF RFC 7430 UC3M Category: Informational C. Paasch," *Analysis,* 2015.

[15] *Open vSwitch*. Available: http://openvswitch.org/

[16] *Floodlight Controller Documentation*. Available: http://floodlight.atlassian.net/wiki/display/floodlightcontroller

[17] *MPTCP Linux Kernell Implementation*. Available: http://multipath-tcp.org/

[18] *Mininet*. Available: http://mininet.org/blog/2015/04/16/announcing-mininet-2-2-1/

[19] US Navy Fleet Size, http://www.nvr.navy.mil /nvrships/fleet.htm