Improving Robustness and Reducing Control Overhead via Dynamic Clustering in Tactical SDN

Philipp Zißner*, Paulo H. L. Rettore*, Bruno P. Santos†,
Roberto Rigolin F. Lopes‡, Johannes F. Loevenich‡, and Peter Sevenich*
*Dept. of Communication Systems, Fraunhofer FKIE, Bonn, Germany
†Dept. of Computer Science, Federal University of Bahia, Salvador, Brazil
‡Secure Communications & Information (SIX), Thales Deutschland, Ditzingen, Germany
Emails: {philipp.zissner, paulo.lopes.rettore, peter.sevenich}@fkie.fraunhofer.de, bruno.ps@ufba.br
{roberto.rigolin, johannes.loevenich}@thalesgroup.com

***Abstract*—This study presents a dynamic clustering approach to address the challenges of managing modern and dynamic tactical communication using diverse radio links in Software-Defined Tactical Networks (SDTNs). State-of-the-art conventional Software-Defined Networking (SDN) protocols, which rely on centralized controllers, are unsuitable for tactical scenarios due to their inherent demands for high-throughput, low-latency control planes, and static nodes. To enable SDTN, we propose a clustering solution designed to distribute network nodes across multiple remote controllers, thereby reducing the control message overhead created by OpenFlow. Moreover, we propose a flexible controller assignment strategy to dynamically balance the network load during runtime and improve network robustness using backup controllers. Our comparative analysis suggests that our clustering solution significantly reduces control overhead across heterogeneous topologies with VHF and UHF links. Furthermore, we replicate a real-world scenario by emulating a convoy of vehicles demonstrating the effective distribution of low control overhead among the controllers and maintaining robust network connectivity, even in the event of a controller failure.**

***Index Terms*—Software-defined Tactical Network, Control Overhead, Clustering.**

I. Introduction

Software-defined Networking (SDN) holds great potential in effectively managing Tactical Networks (TNs), essential for mission planning and control. Recent literature [1] indicates that SDN can facilitate the development and deployment of resilient and adaptive tactical systems. These systems are reconfigurable through software to effectively address the diverse requirements of military operations.

In highly dynamic environments such as battlefields, tactical radios might encounter performance limitations in fulfilling the minimum Quality-of-Service (QoS) requirements of mission-critical services. Thus, managing radio links becomes essential for the dynamic enforcement of QoS policies [2], encompassing tasks such as message prioritization, reliability, security, and load balancing [3], [4]. TNs operate using diverse communication technologies in areas lacking robust communication infrastructure. Various factors like mobility, radio capabilities, terrain characteristics, security threats, and signal interference contribute to fluctuations in network connectivity. Thus, tactical edge systems must demonstrate robustness against connection failures, frequent changes in network topology, cyber vulnerabilities, and flexibility in dynamically managing network policies.

The use of SDN to enhance TNs, known as Software-defined Tactical Networks (SDTNs), presents various challenges, including network vulnerabilities, Single Point of Failure (SPoF), Controller Placement Problem (CPP), dynamic topology changes, and increased overhead for network monitoring and management. To address these challenges and enable the effective deployment of SDTNs, managing the control overhead generated by SDN protocols like OpenFlow is essential as an initial step. While SDN is widely applied in overprovisioned networks, such as enterprise networks, it remains to be adapted for TNs.

Our previous investigation [5] identified the control overhead of OpenFlow in SDTNs in both static and mobile (using Anglova) network scenarios. The findings highlighted a necessity for further research into control overhead reduction, including exploring network clustering and optimizing SDN protocols for SDTNs. Therefore, this paper aims to extend the previous study by investigating network clustering in a multi-controller setup. We propose a flexible controller assignment strategy, which enhances robustness against controller failures and enables efficient load balancing. This paper makes the following contributions:

- Introducing a methodology to compute control overhead, split the network into clusters based on radio links and a specified control plane threshold, and propose a flexible controller assignment strategy to rebalance network load and ensure connectivity in the event of controller failure.
- Experimental results quantifying the control overhead of a SDTN using OpenFlow varying the numbers of vehicles (4-32) and controllers (1-12) in emulated networks with Very High Frequency (VHF) and Ultra High Frequency (UHF) links for different numbers of clusters.
- Emulation of a vehicular convoy scenario, demonstrating the distribution of overhead and network robustness in case of controller failures and new vehicles joining the network.

The paper's organization is as follows: Section II introduces the motivation for clustering to address SDN control overhead in SDTNs, along with the problem statement and related work. Section III presents the proposed methodology to cluster the network and to improve the robustness of SDTN-controllers against control overhead and topology changes. Section IV provides the evaluation of the proposed methodology. Finally, Section V concludes the paper and discusses future work.

II. Background

A. Motivation

The results of our previous investigation [5] revealed that SDN control messages in network topologies encompassing more than eight switches exceed the VHF link capacity. This indicates ineffective scalability of the OpenFlow protocol in TNs, based on the high control message overhead exceeding the nominal link capacity of low-bandwidth networks as the number of devices increases. Accordingly, a higher number of control messages will directly affect the data plane and the installation of QoS policies, which can compromise the objectives of the SDTN. Therefore, the data rate available for the control plane in a SDTN can impact the topology discovery and the network's reliability. Besides, it was demonstrated that a centralized controller is a bottleneck in SDTNs demanding decentralized solutions and optimized protocols.

Furthermore, we identified three key aspects to minimize the overhead of OpenFlow's control messages in SDTNs: i) reduce the frequency of control messages; ii) eliminate non-essential packet types (e.g., error reports or basic status updates) and iii) slice the network into clusters to the optimal number of controllers within a specific network topology.

This investigation explores the clustering approach quantitatively and proposes a dynamic clustering solution, through a flexible controller assignment strategy, for optimizing the control overhead, load balancing, and improving network robustness in case of controller failures. We defined the research problem for this investigation as follows:

Problem: *How does clustering of a SDTN affect the control message overhead in different link technologies? Moreover, how can a dynamic balancing of nodes among the clusters further improve robustness against controller failures?*

B. Related Work

Traditional TNs employ Mobile ad-hoc Network (MANET) protocols to provide reliable communication over unreliable radio links, commonly used in tactical communication systems. Recent advancements in TNs have embraced the utilization of SDN, which can enhance management capabilities in dense and heterogeneous networks with dynamically changing topologies [6]. However, the significant control overhead generated by standardized SDN protocols poses challenges for developing solutions in the tactical domain. This section presents a set of approaches discussed in the literature to reduce SDN control message overhead, including optimizing the OpenFlow protocol, clustering, and addressing challenges with multiple SDN-controllers (e.g., controller placement, domain partitioning, and load balancing).

OpenFlow has been extensively studied to reduce its control overhead [7]. However, the actual performance of the protocol can vary depending on factors such as the version, vendor, and whether it is running on fabric or virtual switches [6]. In TNs, OpenFlow offers potential flexibility and programmability, crucial for fast network reconfiguration to meet a wide range of QoS requirements in Command and Control (C2) applications.

A single centralized controller is used in common SDN topologies, posing scalability and reliability challenges in TNs. Clustering the network with multiple controllers and implementing load-balancing mechanisms can reduce scalability and reliability issues [3]. Furthermore, it mitigates control overhead, but the applicability to TNs is uncertain, as most literature discusses solutions for large-scale networks.

The usage of multiple controllers is widely discussed in the literature, offering a solution to the increasing demand for flow processing in large-scale networks and solving problems regarding scalability and reliability. There are two main architectures to design solutions for multi-controller SDN topologies: i) a flat architecture containing multiple SDN domains, each linked to a controller that implements techniques to ensure communication between clusters and ii) a hierarchical architecture utilizing a global network view through an extra root controller managing the controllers. Besides offering solutions to the problems of a single centralized controller, multi-controllers' usage adds new challenges, mainly related to optimal controller placement and domain partition [3].

In [8], a k-self-adaptive SDN-controller placement approach is proposed for Wide Area Networks (WANs), sharing similarities with TNs in terms of long propagation delay and limited bandwidth. The method divides the network into multiple SDN domains, each with its own controller, using spectral clustering to optimize controller placement and minimize propagation delay for improved reliability. Load balancing among multiple controllers has received significant research attention. Yu et al. [4] propose a load balancing mechanism for distributed floodlight SDN-controllers, utilizing a load information strategy. They enable local balancing decisions through periodic load information sharing.

III. Methodology

In this investigation, we emulate SDTNs using VHF and UHF links with capacities up to 9.6 kbps and 240 kbps, respectively. To avoid link capacity overflow due to the extra control messages sent by the SDN-controller, strategies to reduce overhead are required. Therefore, we introduce a multi-controller network topology reducing control overhead by distributing nodes across different clusters. The solution described in this section addresses the following problems that emerge from using multiple controllers in SDTNs: i) the optimal number of clusters for a given amount of nodes and ii) a flexible controller assignment to nodes, balancing the network and improving the robustness by countering controller failures.

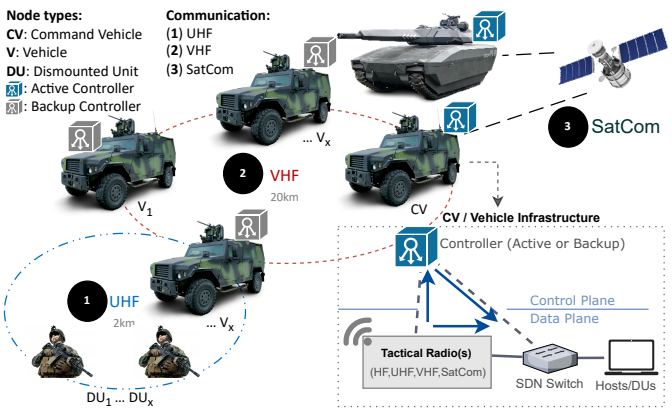

Fig. 1: Cluster setup in a tactical network scenario.

A. SDTN Clustering

1) Network Setup: Fig. 1 illustrates a network with multiple SDN-controllers deployed in a clusterized SDTN. Each network cluster may host multiple vehicles (V), ensuring connectivity among themselves and the Dismounted Units (DUs), e.g., soldiers or drones, connected to them and communicating with each other. In the scenarios discussed here (Section IV), a cluster contains one vehicle with an active SDN-controller, called Command Vehicle (CV), controlling the connectivity among all vehicles and DUs. The remaining vehicles provide backup controllers. Thus, our solution assumes that all vehicles are equipped with a remote SDN-controller (active or backup) connected to an Open vSwitch (OvS) and an Access Point (AP) in the form of a tactical radio as shown in Fig. 1. The radios are arranged in a wireless mesh topology, providing a connection to each vehicle within range and ensuring connectivity in the complete network. The OvS of each vehicle maintains the required flow tables to establish network routes and ensure data plane communication. All network topologies were emulated using *Mininet-WiFi* and containerized SDN-controllers (using Ryu controller). *Mininet-WiFi* is designed to emulate WiFi links with high capacity, so Linux Traffic Control (TC) was used to limit the link capacity to VHF and UHF radio links (9.6 and 240 kbps, respectively).

2) Network Monitoring: OpenFlow transmits a set of packet types via the control plane, such as *HELLO*, *PACKET_IN* and *PACKET_OUT*, to control the data flow and the neighbor discovery. SDN overhead denotes the number of packets that are additionally transmitted and received by the controllers. Considering the limited radio link capacities in the tactical domain, this overhead plays a crucial role in assessing the feasibility of deploying SDTNs. To measure it, all OpenFlow packets are captured on each controller, and the sum of packet lengths is calculated and aggregated to an average value based on a predefined time window. The reliability and robustness of SDTNs is quantified using the packet loss, defined as the amount of data packets that were not transmitted between the nodes (e.g., DUs or vehicles) through the data plane. Therefore, a modified *ping* function to measure the packet loss within each cluster is applied. First, a list of radio interfaces that are connected to the respective CV (active controller within the cluster) is obtained using *ovs-vsctl*, a utility for querying and configuring an OvS. This information lets us obtain the vehicles connected to each CV. Utilizing the *iw* tool, we output all DUs connected to the respective radio interfaces within each cluster and parse the output. Combining these methods from *ovs-vsctl*, a list of all DUs inside the different clusters is obtained, allowing us to measure the respective packet loss. Therefore, we take a subset of DUs connected to distinct radios in the same cluster and transmit ping requests between them. Using this approach, we obtain a small set of nodes representing the communication within each cluster, reducing the number of packets to be transmitted to a minimum if compared to ping between all nodes. The functions to measure these statistics are executed for each cluster during the network runtime.

B. Flexible Controller Assignment

1) Network Balancing: The proposed solution for dynamic load balancing of a SDTN during runtime consists of two main steps: i) detecting the network imbalance and ii) applying clustering or controller balancing.

The imbalance detection mechanism relies on the available information about the network. Two types of control architectures can be used: hierarchical or flat controller setup [3]. In the hierarchical controller setup, all overhead measurements are shared with the master (root) controller, allowing a comparison of overhead between clusters to detect an unequal distribution of load. However, the additional root controller increases the control overhead. By contrast, in the flat architecture, the controllers maintain a local view, just having access to the network information within each cluster. In this case, the overhead is compared to a certain threshold, such as 50% of link capacity. If this threshold is exceeded, a control message can be transmitted to all other clusters to share the current network state. Therefore, to reduce the control overhead, we detect the network imbalance considering a flat controller setup.

Following the detection, there are two applicable methods to reduce the overhead: i) further clusterization of the cluster that exceeds the control plane capacity by activating one (or more) available backup controllers, reassigning the vehicles to the new CV, and increasing the total number of network clusters or ii) balancing all network nodes evenly to the set of CVs already defined. Both methods require access to the controllers of each cluster to rearrange all vehicles of the network. In a real-world scenario, such methods could be executed when all troops gather at a meeting point and can be arranged between CVs.

On detection of the network imbalance, the balancing process calculates the number of nodes (vehicles and DUs) per controller (CV) and distributes them across all clusters. The number of nodes per controller is obtained by dividing the amount of vehicles by the number of command vehicles. All network nodes are iterated and assigned to a specific

Algorithm 1: Cluster Robustness

Input: sw, $radios$ (local view)
Result: Assigns cluster nodes to backup controller

```
1  i ← 0
   /* PL values are measured for each cluster
      during runtime                                  */
2  PL ← packetLoss() /* array with PL values         */
3  Npl ← |PL|
   /* check for 100% PL on two consecutive
      measures                                        */
4  if (Npl >= 2) and
     (PL[Npl − 1] == PL[Npl − 2] == 100%) then
       /* select one of the backup controllers      */
5      BackupCon ← selectBackupController()
       /* reassign nodes to backup controller       */
6      while i < |sw| do
7          setController(sw[i], BackupCon)
8          setController(radios[i], BackupCon)
9          i ← i + 1
10     end
11 end
12 return True
```

controller using the *set-controller* function from *ovs-vsctl*. This results in a balanced network, distributing the control messages equally across all SDN-controllers that are active in the current network state and therefore reduce the control overhead inside each cluster.

2) Network Robustness: Besides achieving a balanced distribution of control overhead, SDTNs also demand reliability and robustness in the control plane. A way to achieve such requirements in a clusterized SDTN is by using multiple controllers to reduce the probability of outages and failures due to technical problems or destruction. Each vehicle inside a cluster holds the complete communication infrastructure (controller, switch, and radio), allowing a connection to the other vehicles, DUs, and clusters through a radio mesh. Using one active (CV) and multiple backup controllers (vehicles) allows the creation of a reliable and robust network topology by implementing a function executed in each network cluster that can dynamically activate a backup controller, if required.

The packet loss measurements between DUs in a cluster are utilized to detect a controller failure and initialize the re-establishment of connectivity. A controller failure is detected by multiple consecutive pings between DUs transmitting no packets within the same cluster. When a failure happens, the selection of a backup controller is triggered, followed by the assignment of switches and radios within the local cluster to the new controller, as described in Algorithm 1. First, the algorithm initializes an index i (line 1) and accesses the array PL containing the packet loss measurements from the respective cluster (line 2). The values are obtained by the previously introduced packet loss function that is executed within each cluster. Starting with the second measurement, the algorithm tests for two consecutive measurements, failing to transmit packets and reaching a packet loss of 100%, indicating a connectivity problem within the cluster (line 4).

	CON	4 Vs	8 Vs	16 Vs	24 Vs	32 Vs
VHF Link	1	5.24	23.19	42.45	164.74	179.49
	2	3.08	10.34	19.35	60.67	79.48
	4	1.58	5.23	9.01	29.86	35.56
Cap. = 9.6 kbps	8		2.32	3.47	14.36	12.13
50% Thr. = 4.8 kbps	10			3.50	11.24	13.39
	12				9.21	11.10
UHF Link	1	8.43	38.33	55.60	156.24	168.68
	2	4.40	19.31	24.53	57.25	76.79
	4	2.28	9.37	11.26	27.82	25.92
Cap. = 240 kbps	8		4.75	4.08	14.18	16.61
10% Thr. = 24 kbps	10			3.64	10.64	13.05
	12				9.14	10.53

Above Link Cap. ; Above Thr. ; Below Thr.

TABLE I: SDN Overhead on various network topologies

If this statement is fulfilled, one backup controller inside the cluster is selected (line 5). Currently, the selection is arbitrary, but it could be based on other network attributes, such as signal strength and latency. Finally, all switches and radios inside the cluster are reassigned to the backup controller, using the *setController* function (lines 6-10). The presented algorithm runs in a thread for each cluster and it is executed regularly to detect connectivity failures and activate a backup controller.

IV. Evaluation

A. SDTN Clustering Overhead

We evaluated the control overhead reduction in clusterized SDTNs through a set of different experiments. The experiment duration was set to 10 minutes in each scenario varying the number of vehicles and controllers in the network. All test scenarios contained two DUs (e.g., wireless stations) connected to each tactical radio (e.g., AP), equally distributed among the controllers. Table I shows the average overhead of each experiment, measured every 30 seconds, in a VHF and UHF network with the maximum nominal data rate of 9.6 kbps and 240 kbps, respectively. These networks were emulated using Linux Traffic Control (TC) and Network Emulator (NetEm). Due to the shared link capacity between the control and data planes, we define a threshold value of 50% for the VHF network and 10% for the UHF network that should not be exceeded by the control messages. Using three different colors, the table indicates measurements that lie below these thresholds (green), above the threshold but below the link capacity (orange), and above the link capacity (red).

All experiments utilizing a single controller on VHF links exceed the maximum capacity, except a topology with four vehicles, which still remains above the defined threshold. Applying clustering, we achieve a reduction of overhead in all experiments and even undercut the capacity threshold in smaller network topologies up to 16 vehicles, by adding a sufficient amount of controllers. In a topology with 24 vehicles, applying 12 clusters shows a strong reduction of overhead, lying below the maximum link capacity. Finally, for 32 vehicles, applying 12 controllers shows a strong reduction of overhead, from 179.49 kbps on a single controller setup to 11.10 kbps, but does not undercut the link capacity. In conclusion, clustering can be applied to a VHF network. However, in

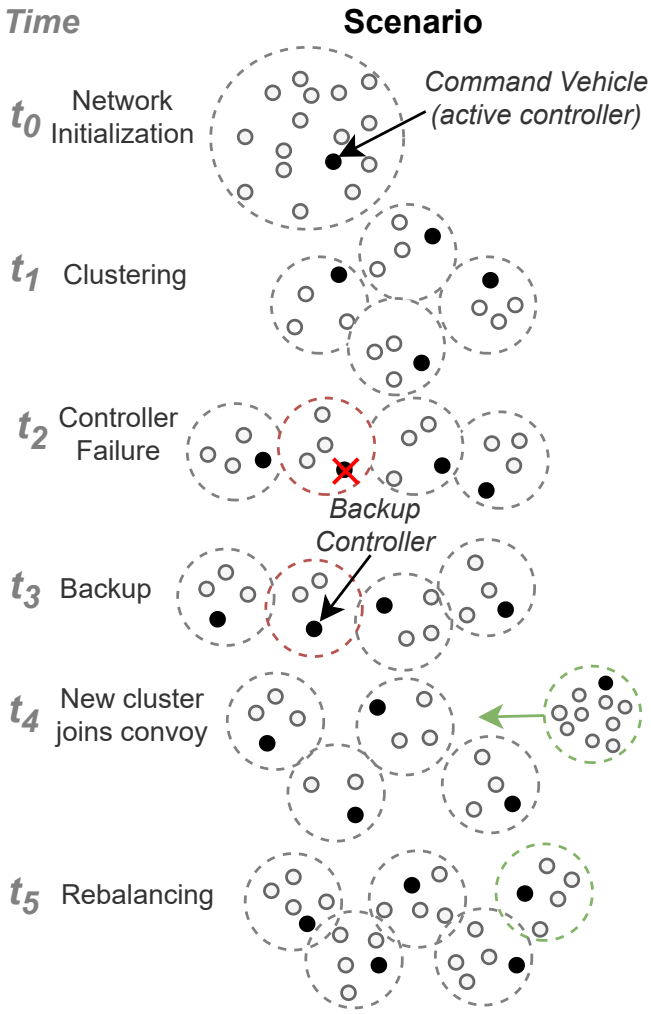

Fig. 2: Scenario to test the robustness of SDTN controllers.

larger topologies, the link capacity is not sufficient, requiring a big slice of the available bandwidth for the control plane. Although in small networks, OpenFlow can be used, it is still limited due to the reduced number of devices that can be managed by the SDN controller.

The second set of experiments emulates the same network topologies on a UHF link. Noticeably, the majority of the topologies are below the 10% threshold when applying more than one controller to the network, showing the benefits of clustering and encouraging our desired goal. For up to 16 vehicles in the network, the threshold is undercut by creating a second cluster in the network. A topology with 24 or 32 vehicles requires eight controllers to not exceed the threshold.

Our results provide a reference to the optimum number of clusters in various network topologies with different communication technologies and control plane thresholds. Based on these numbers, our flexible clustering approach creates the network clusters during the initialization phase. Moreover, the results show that clustering significantly reduces the overhead on each controller in the network, benefiting the applicability

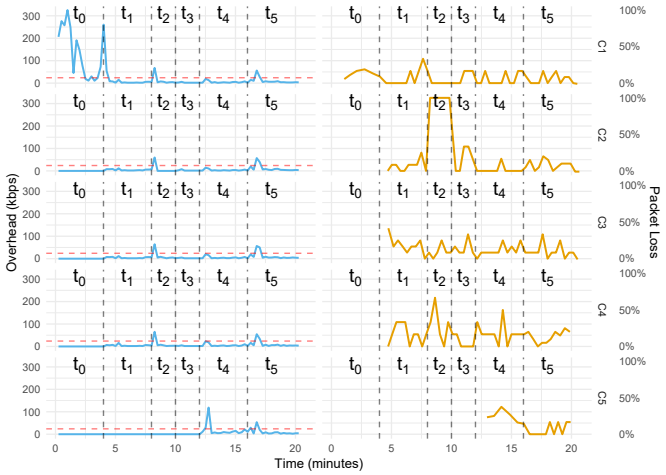

Fig. 3: Emulation measurements of one experimental run.

of SDTNs. In some cases, mainly on VHF networks, the additional overhead remains to take up a substantial amount of the available link capacity. Reduction of control packets or optimized protocols could be applied in addition to network clustering to further reduce the control overhead. For the UHF link, the proposed clustering solution can reduce the overhead below the 10% threshold in all experiments.

B. Flexible Controller Assignment

To evaluate the flexible controller assignment regarding robustness and balancing, we constructed a scenario that reflects a possible real-world convoy moving on a battlefield (inspired by the Russia-Ukraine war). Fig. 2 illustrates the scenario, including various events at time intervals $t_0, ..., t_5$ occurring during the network emulation. Furthermore, Fig. 3 plots the control overhead (kbps) and packet loss (%) within each cluster ($C1 - C5$), with an additional indication of the time intervals and a red horizontal line showing the 10% link capacity threshold of UHF, at 24 kbps. The network contains 16 vehicles arranged in an UHF mesh network topology.

During the network initialization phase (t_0), a significant control overhead is generated because at this time all vehicles are linked to a single controller, the CV. This is expected due to OpenFlow transmitting many control messages between the network switches that are linked to the tactical radios, initializing the flow tables to ensure communication between all connected vehicles and dismounted units. The overhead is reduced during this phase, still lying above the 10% threshold of the UHF data rate most of the time. After four minutes, the network is split into four clusters (t_1), based on the prior results listed in Table I for a UHF network with 16 vehicles, equally distributed across the controllers. Following a peak in C1, each of the four active controllers (CVs) shows a significant reduction in control overhead compared to t_0, undercutting the UHF link capacity, and a packet loss fluctuating around 15%, showing peak values of up to 33%. This fluctuation is expected due to wireless communication and the loss probabilities.

Time	OV (kbps)	PL	CON	Event
t0	148.13	15.97%	1	Initialization
t1	5.99	9.74%	4	Clusterization
t2	10.29	30.92%	3	C2 Failure
t3	2.54	9.06%	4	C2 Backup
t4	7.56	9.77%	5	New Cluster
t5	11.41	10.46%	5	Balancing

TABLE II: Emulation Results - Average

To test the robustness, at t_2 the scenario emulates a failure of the active controller in one of the four clusters, C2, removing the CV from the network topology. The failure is immediately visible through the packet loss of 100% in C2 during this phase and the general overhead slightly increasing, while trying to re-establish a connection to C2. After this, the controller overhead in C2 has a value of zero, with no more control packets being transmitted. Due to the presence of multiple vehicles in each cluster, another controller is activated to restore the network connection by randomly choosing one vehicle within the cluster. All affected nodes inside C2 are assigned to the backup controller. This immediately re-establishes the communication within the cluster, reflected by the measurements during interval t_3 in Fig. 3. Following this event, C2 only maintains three remaining nodes as illustrated in Fig. 2 t_3.

Finally, during t_4 and t_5, another event happens requiring a flexible rebalancing of the network clusters. In t_4, a new cluster with ten additional vehicles joins the convoy, increasing the network traffic. The new cluster C5 is transmitting nearly three times the amount of control messages compared to the other clusters due to the new controller being assigned to a significantly higher number of nodes. To solve this problem, the clusters are dynamically rebalanced, resulting in a well-distributed network topology. This is visualized in Fig. 3, showing that the previous increase of overhead during t_4 is not further given but instead equally distributed among all clusters, lying below the 10% threshold. Subsequent to the network balancing on t_5 each cluster contains an equal amount of five vehicles, with one of them taking the role of a CV with an active controller.

The scenario was emulated ten times, with Table II tabulating the average overhead (OV), average packet loss (PL), and the number of active controllers (CON) during each time interval together with the corresponding event. This confirms the measurements of the scenario emulation discussed previously, stating a high overhead during the network initialization phase using a single controller for all nodes (t_0). During the cluster creation (t_1), the overhead decreases, reaching a value below the predefined 10% link capacity threshold. Throughout the C2 Failure event, the average packet loss increases to 30.92% due to the total loss of communication within the C2 cluster (t_2). Following the reconnection using the backup controller in t_3, the overhead is significantly reduced, and the packet loss normalizes to a low level at 9.06%. When the additional cluster joins the network, the number of controllers expands to five, reflected by a slight increase in overhead in each cluster (t_4). During the controller rebalancing event (t_5), a minor increase in overhead and packet loss due to the additional reassignment process is observable.

V. Conclusion

This paper introduced a flexible controller assignment strategy to rebalance network load and improve connectivity in the event of controller failure in SDTNs. The proposed solution has an initial phase exploring network topologies over VHF and UHF links to determine the optimum number of clusters for reduced overhead. The experimental evaluation quantified OpenFlow's control overhead on a clusterized network hosting a set of events, including a controller failure and new nodes joining the network. These experiments suggest that our solution can significantly reduce the control overhead in network topologies utilizing VHF and UHF links. However, larger topologies using VHF required additional overhead reduction measures, such as modified SDN protocols or OpenFlow optimizations. After the initialization phase, our flexible assignment strategy achieved minimal overhead and maintained a robust connection during the runtime of a network with multiple controllers.

Future work entails improving controller selection in the balancing algorithm to detect failures faster and adapting the OpenFlow protocol to support network balancing requirements. Additionally, we aim to investigate strategies to reduce control packets, creating a lightweight OpenFlow protocol that can be combined with clustering to further minimize control overhead, particularly for VHF links.

Acknowledgement

The authors would like to thank CAPES, CNPq, PROPCI-PROPG/UFBA 007/2022 - JOVEMPESQ for funding support.

References

[1] M. von Rechenberg, P. H. L. Rettore, R. R. F. Lopes, and P. Sevenich, "Software-Defined Networking Applied in Tactical Networks: Problems, Solutions and Open Issues," in *2021 International Conference on Military Communication and Information Systems (ICMCIS)*, 2021.

[2] S. M. Eswarappa, P. H. L. Rettore, J. Loevenich, P. Sevenich, and R. R. F. Lopes, "Towards Adaptive QoS in SDN-enabled Heterogeneous Tactical Networks," in *2021 International Conference on Military Communication and Information Systems (ICMCIS)*, 2021.

[3] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, "Multi-controller Based Software-Defined Networking: A Survey," *IEEE Access*, 2018.

[4] J. Yu, Y. Wang, K. Pei, S. Zhang, and J. Li, "A load balancing mechanism for multiple SDN controllers based on load informing strategy," in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2016.

[5] P. H. L. Rettore, M. Djurica, R. R. F. Lopes, V. F. S. Mota, E. Cramer, F. Drijver, and J. F. Loevenich, "Towards Software-Defined Tactical Networks: Experiments and Challenges for Control Overhead," in *MILCOM 2022 - IEEE Military Communications Conference (MILCOM)*, 2022.

[6] L. C. Costa, A. B. Vieira, E. d. B. e Silva, D. F. Macedo, L. F. Vieira, M. A. Vieira, M. d. R. M. Junior, G. F. Batista, A. H. Polizer, A. V. G. S. Goncalves *et al.*, "OpenFlow data planes performance evaluation," *Performance Evaluation*, 2021.

[7] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, "Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey," *IEEE Access*, 2019.

[8] P. Xiao, Z.-y. Li, S. Guo, H. Qi, W.-y. Qu, and H.-s. Yu, "AK self-adaptive SDN controller placement for wide area networks," *Frontiers of Information Technology & Electronic Engineering*, 2016.