Performance Evaluation of Kubernetes Distributions (K8s, K3s, KubeEdge) in an Adaptive and Federated Cloud Infrastructure for Disadvantaged Tactical Networks

M. Fogli, T. Kudla, B. Musters, G. Pingen, C. van den Broek, H. Bastiaansen, N. Suri and S. Webb

2021 International Conference on Military Communication and Information Systems (ICMCIS), 2021

Cite this as

M. Fogli, T. Kudla, B. Musters, G. Pingen, C. van den Broek, H. Bastiaansen, N. Suri and S. Webb, "Performance Evaluation of Kubernetes Distributions (K8s, K3s, KubeEdge) in an Adaptive and Federated Cloud Infrastructure for Disadvantaged Tactical Networks," *2021 International Conference on Military Communication and Information Systems (ICMCIS)*, The Hague, Netherlands, 2021, pp. 1-7, doi: 10.1109/ICMCIS52405.2021.9486396.

Performance Evaluation of Kubernetes Distributions (K8s, K3s, KubeEdge) in an Adaptive and Federated Cloud Infrastructure for Disadvantaged Tactical Networks

Mattia Fogli
University of Ferrara
Ferrara, Italy
mattia.fogli@unife.it

Niranjan Suri
US Army Research Laboratory
Adelphi, MD USA
niranjan.suri.civ@mail.mil
Florida Institute for Human & Machine Cognition (IHMC)
Pensacola, FL USA
nsuri@ihmc.org

Thomas Kudla
Fraunhofer Institute for Communication, Information Processing and Ergonomics
Wachtberg, Germany
thomas.kudla@fkie.fraunhofer.de

Sean Webb
Maritime Decision Support
Defence Research and Development Canada
Dartmouth, Canada
sean.webb@drdc-rddc.gc.ca

Bram Musters,
Geert Pingen,
Casper van den Broek and
Harrie Bastiaansen
Monitoring and Control Services
The Netherlands Organisation for applied scientific research (TNO)
The Hague, The Netherlands
bram.musters@tno.nl,
geert.pingen@tno.nl,
casper.vandenbroek@tno.nl
harrie.bastiaansen@tno.nl

***Abstract*—The tactical edge domain, primarily consisting of dismounted soldiers and vehicles on the move, are typically interconnected via wireless tactical networks that are limited in terms of bandwidth, reachability, reliability, and latency. Hence, nodes in the tactical network cannot simply rely on assured access to enterprise cloud computing. Instead, they must explore other alternative models to leverage resources that are in situ, by means of a federated cloud architecture that spans the three tiers of dismounted soldiers, vehicles on the move, and operations centers. The NATO IST-168 RTG has been exploring approaches to best exploit available resources in such a federated architecture while living within the constraints of the tactical networks. The first approach has been to evaluate Kubernetes technologies to see if they are able to be deployed over tactical networks and provide the capabilities to dynamically distribute data and computing tasks over a federated cloud infrastructure composed of multiple partner nation networks. This paper provides initial performance results for various Kubernetes distributions (K8s, K3s, KubeEdge) in federated and adaptive tactical networks, leading to recommendations for further development and experimentation.**

***Keywords*—Tactical Clouds, Federated Clouds, Kubernetes, K8s, K3s, KubeEdge, Disadvantaged Tactical Networks, Network Adaptation**

I. Introduction

Military vehicles and personnel have ever more data sensing, processing, storage, and communication devices available. This provides major opportunities for improving information availability in mission contexts. However, degraded and disadvantaged networks are typical in battlefield situations. This makes sharing relevant data and information a challenge. It may prevent (potential mission-critical) information from being available in a timely manner.

In previous work [1], the NATO IST-168 RTG (Research Task Group) on 'Adaptive information processing and distribution to support Command and Control' proposed an adaptive and federated approach to address this challenge. It builds upon the concept of federation in the sense that mission partners mutually expose their computing and storage resources to other mission partners through well-defined APIs. Jointly, they provide adaptivity to users as to where to execute information processing tasks. As stated in [1], potential military benefits are increased availability of information to all echelons including resource-constrained nodes (e.g., mobile or dismounted soldiers) and communication-constrained nodes (e.g., underwater platforms) [2], reduction in data overload [3], and increased safety of military personnel due to a more cyber resilient military information infrastructure. Technical benefits include more balanced and efficient utilization of scarce resources, lowered demands on the limited, disadvantaged, battlefield communications network (e.g. by performing data preprocessing, fusion, and filtering at or closer to the source of the data), improved security through local processing of sensitive data within a confined security domain, prioritization of processing tasks over scarce network and computing resources in a pro-active mode, and improved resilience of cloud deployment [4].

This paper is based on NATO IST Panel RTG IST-168, 'Adaptive Information Processing and Distribution to Support Command and Control'

The adaptive and federated cloud infrastructure approach is based on using Kubernetes technology within each individual partner's domain. Developed by Google for enterprise environments, Kubernetes (K8s) [5] is a de facto industry standard open-source platform that acts as an orchestrator by providing mechanisms for deploying, maintaining, and scaling containerized applications in a cluster. Designed for enterprise networks, it assumes a robust and high-capacity data plane and relies on TCP for its network communications – assumptions that do not hold for military Tactical Networks (TNs) that are resource constrained. Nowadays, alternative Kubernetes distributions have become available, some of them specifically designed for resource-constrained environments, such as Lightweight Kubernetes (K3s) [6], and Kubernetes Native Edge Computing Framework (KubeEdge) [7], which may be more suitable to TNs.

This paper presents ongoing research within the IST-168 RTG to assess the performance of the various Kubernetes distributions for federated and adaptive clouds in TNs in a set of representative deployment models. The TNs are characterized by disadvantaged network conditions with limited and variable bandwidth, variable latency, unreliability, and intermittent connectivity [8]. A rigorous study of the performance of various Kubernetes distributions under such conditions enables a detailed study of their behavior, an understanding of the extent to which military personnel can take advantage of their capabilities [1], identify options for technically improving the (military deployment of the) various Kubernetes distributions, and help mission partners in defining their cloud strategy.

The paper has the following structure: Section II describes the representative deployment scenarios in TNs to assess the performance of the various Kubernetes distributions (K8s, K3s, KubeEdge). Subsequently, section III addresses the experimentation testbed. The experiment set-up is presented in section IV, after which section V provides the results of the performance assessment. Finally, section VI provides the overarching conclusions and future work.

II. Deployment Scenario in Tactical Networks (TNs)

A fundamental assessment of the performance of the various Kubernetes distributions for federated and adaptive clouds in disadvantaged TNs requires a testing environment that represents a realistic TN with disadvantaged network conditions. The deployment context for assessing the performance of the various Kubernetes distributions is based on the Anglova scenario, vignette 3 [9], detailing the neutralization of insurgents in a city environment.

Various activities related to the adaptive cloud infrastructure take place in this land-based scenario [1]: Video feeds from vehicles of advancing platoons of various mission partners are locally analyzed, indexed and stored, e. g. in a vehicle at the edge of the TN), resulting in a federated and distributed video footage library. Subsequently, an object recognition service can be executed on the distributed video footage library to identify relevant video footage available in the various mission partner clouds through analysis on specific aspects. By means of a well-defined set of cloud service APIs the availability of video analysis services, availability of processing and networking resources in the various federated and adaptive partner clouds can be discovered. On its outcome, a plan for the distribution of how and where to move the data, the code, and the results between the various mission partner clouds is constructed and its deployment is orchestrated. The performance of the various Kubernetes distributions is assessed mimicking these types of activities over the federated and adaptive cloud infrastructures in realistic military scenarios, such as vignette 3 of the Anglova scenario [9].

In the federated and adaptive cloud infrastructure based on Kubernetes, each participating nation will setup its own cloud environment(s), with its own specific security policies and resource constraints. A nation can either run one single cloud environment, i.e., a single Kubernetes cluster, or run several clusters. A cluster consists of an arbitrary number of master and worker nodes. This design decision determines the demarcation of the individual clouds. It is referred to as the 'deployment model' and should provide a balance between the available computing and storage resources with adequate "intra cloud" bandwidth (e.g., a cloud spanning the resources and network within a vehicle) on one hand and the restricted availability of "inter cloud" bandwidth in the disadvantaged TN on the other hand [1].

Initially, Kubernetes (K8s) had been designed to run in a single data center [5] with an abundance of processing and networking resources. Given that these assumptions do not hold for our target environment with TNs, the question arises as to whether any of the Kubernetes distributions would work as intended, and if not, whether there are alternate deployment models that could be used to adapt Kubernetes for TNs. Therefore, the IST-168 RTG conducted experiments with running different distributions over a variety of different TNs with disadvantaged network conditions and compared the performance with an enterprise networking environment.

For the performance assessment of the various Kubernetes distributions in TNs, the following deployment models are taken into account:

- **One Cloud per Mission Context**
  A single Kubernetes cluster is deployed and orchestrates all the available resources for that mission, i.e., deployed datacenters and deployed (edge) nodes on vehicles and mobile devices. Assuming it is a feasible solution in TNs, the main drawback would be that the Kubernetes' scheduling algorithm does not optimize bandwidth usage.
- **Several Clouds per Mission Context**
  Depending on the mission setup there are several Kubernetes clusters deployed and each instance only manages a subset of the available resources within the mission. A setup could be that each platoon in that mission (up to 36 soldiers and related vehicles) is a cluster. Therefore, a scheduling algorithm to deploy containerized applications across different Kubernetes clusters should be developed.

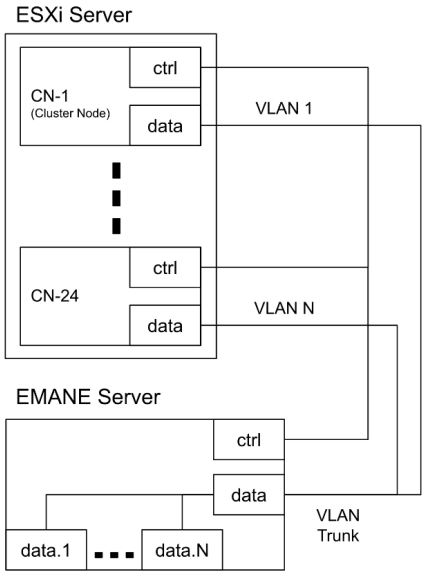

Fig. 1. Testbed configuration.

These deployment models raise questions regarding which is suited best for different mission contexts and what are the limits for the size of a cluster with regards to the worker nodes. The experiments described in this paper give an overview of the possibility to run Kubernetes clusters with different sizes and with different Kubernetes distributions on TNs.

III. Experimentation Testbed

The testbed is composed of a single physical server Dell PowerEdge R630 equipped with two Intel(R) Xeon(R) E5-2680 v3 CPUs with 192 GB of RAM and running VMWare ESXi 6.7 as a bare-metal hypervisor. Twenty-four VMs based on Ubuntu 18.04 LTS work as cluster nodes with each of them equipped with two CPUs, four GB of RAM, and two network interfaces. There is one network interface for application data (data interface - which is used by Kubernetes for all of its network traffic) and one for experiment control traffic (control interface – used by us to start and stop experiments and collect results). A separate Dell PowerEdge R620 server with Ubuntu 18.04 LTS runs the Extendable Mobile Ad-hoc Emulator (EMANE) [10]. EMANE is an open-source framework for emulating real-world wireless network conditions. The testbed configuration is shown in Fig. 1.

The data network is configured as a virtual switch within ESXi. Each cluster node connects its data interface to that virtual switch. A different VLAN identifier (VID) is assigned to each connection. The EMANE server is connected to that switch through a VLAN trunk, which uses the 802.11Q protocol. Within the EMANE server, the VLANs are broken down into multiple virtual interfaces - one for each VLAN that is in place. Any traffic passing through the cluster nodes' data interfaces is transferred to the corresponding virtual interfaces inside the EMANE server. At runtime, EMANE reads from those interfaces, applies the necessary communication effects, and writes the data out to the interfaces that should receive the data. The control network is used to log into the VMs, start/stop the experiments, and collect the data.

TABLE I
Network scenarios used in the experiments.

	Bandwidth	**Reliability**	**Latency**
LAN	1 Gbit/s – no restrictions	No Added Loss	No Added Latency
best	4 Mbit/s	95%	200 ms
avg	2 Mbit/s	90%	400 ms
worst	650 kbit/s	85%	3000 ms

IV. Experiments for Performance Assessment

This section describes the experiments conducted to investigate how K8s (v1.20), K3s (v1.19), and KubeEdge (v1.5) perform in TNs. Such experiments explore the network overhead introduced by the aforementioned Kubernetes distributions while managing clusters in disadvantaged network conditions. Assuming the enabling software is pre-downloaded on cluster nodes, the network overhead represents the network resources consumed by cluster nodes for accomplishing Kubernetes-related tasks. Because this experimentation is limited to K8s, K3s, and KubeEdge, we acknowledge that the effectiveness of other Kubernetes distributions in TNs should be investigated as well. The choice to focus this work on these three distributions is led by a twofold objective: narrowing the number of possible experiments and comparing K8s to well-known solutions explicitly designed for resource-constrained scenarios.

The experiments vary along four dimensions. First, the cluster size, i.e., the number of cluster nodes. We consider clusters involving six, twelve, eighteen, and twenty-four nodes, with only one node acting as a master in all cases. Therefore, this experimentation does not consider high-availability Kubernetes clusters in which there are multiple masters. Second, the quality of communication links. As shown in Table I, we propose four network scenarios in which bandwidth, reliability, and latency are incrementally degraded to emulate the tactical environment. More specifically, the bandwidth is shared (fractions of the whole bandwidth are equally allocated to cluster nodes), reliability refers to the percentage of packets being lost on each side of a communication link, and latency describes the delay on a communication link. The same network scenario is applied to all the communication links involved in an experiment. Third, the cluster workload, i.e., the number of running containerized applications per node. We consider application deployments of 10 and 20 replicas per node. All the replicas are based on the same Docker image, which contains a simple "Hello world" application. Fourth, the Kubernetes distribution itself (K8s, K3s, KubeEdge).

Through the experiments, we seek to evaluate two fundamental aspects in the management of containerized applications across multiple hosts, i.e., cluster initialization and application deployment. At the beginning of the cluster initialization, one of the cluster nodes is initiated as the master. Then, the remaining nodes (acting as workers) try to join

the master simultaneously. We analyze how K8s, K3s, and KubeEdge manage the cluster initialization phase by varying cluster size and network scenario. The application deployment phase regards the deployment of containerized applications through the Kubernetes API. Assuming both the cluster is initialized and the Docker image is pre-pulled on the workers, the master tries to deploy the same number of replicas on each worker simultaneously. In the application deployment experiments, we vary cluster size, network scenario, and also cluster workload. Altogether, there were 144 experiments, with each of them executed three times over a period of ten minutes.

To automate the execution of the experiments, we developed a test framework based on Ansible [11]. The test framework uses the control network to command EMANE and the cluster nodes. Ansible is a configuration management tool and it was selected mainly due to its simplicity. Configurations can be easily defined in YAML syntax, and it does not require agents running on managed nodes. Configurations are pushed from a control node (where Ansible is installed) to managed nodes over SSH. In Ansible, a playbook is a list of tasks that automatically execute against a set of hosts defined in an inventory. The test framework is constituted of inventories, playbooks, and shell scripts. Inventories target subsets of the cluster nodes accordingly to the cluster sizes proposed in the experiments. Playbooks describe the necessary tasks to perform the experiments. Such tasks range from the deployment of a Kubernetes distribution to the setup of communication effects using EMANE to the capturing of network traffic using TCPDUMP.

V. Results

The analysis of the collected data of the experiments was done in an automated and reproducible manner. During the course of each experiment, the incoming and outgoing traffic from each node is captured and stored on that node. Therefore, for each experiment, up to 24 capture files are generated, depending on the number of nodes involved. The file names for the experiments follow a specific pattern. On the one hand, this helps to distinguish between different experiment configurations, and on the other hand it helps to group the same configuration during later analysis. Once an experiment finished, the data was uploaded to a fileserver from where it was later downloaded for analysis. The analysis utilizes Docker to create a common environment with all necessary software and scripts installed. Preprocessing of the captured files, analysis, and generation of meaningful graphs is done once the Docker is started with all the captured data files mapped into the container. For the analysis and graph generation the data science tool R was used. The following depicts the analysis steps:

- Dividing the raw data set into succeeded and failed experiments
  - Cluster initialization experiments are marked as having succeeded, if all nodes that were configured for that experiment were able to join the cluster within a 5-minute timeout.
  - Application deployment experiments are marked as having succeeded, if all pod instances that were supposed to be deployed for that experiment were successfully deployed within a 5-minute timeout.

  All other experiments are marked as having failed.
- Splitting traffic into bootstrapping and steady state phase
  For each experiment, there are two phases, bootstrapping and steady state. Before an experiment run is started, a specific packet is sent using the Nping tool with a specific payload to indicate the start and the end of the bootstrapping phase. After that phase, the experiment runs until the timeout of ten minutes is reached. By using those specific packets, the captured traffic is split into datasets for each phase:
  - Bootstrapping phase
    For the cluster initialization experiments, this is the time needed from issuing the join command on the nodes to join the master up to moment when all nodes successfully joined the master. In the application deployment experiments, the cluster is already fully initialized. Here, the time measured includes the time taken from issuing the application deployment command on the master node up to the moment when each node has successfully deployed all application replicas and communicated back that state to the master node.
  - Steady state phase
    This phase starts when bootstrapping has finished. The state of the cluster does not change in this phase, so all measured network traffic within the cluster consists of messages, (e.g., liveness probes), initiated by Kubernetes to keep track of the cluster state. One entire experiment takes 10 minutes, so the length of the steady state phase is defined as 10 minutes minus the duration of the bootstrapping phase.
- Statistical analysis and graph generation
  R is used to analyze the data and to generate graphs showing different aspects of the behavior and performance of Kubernetes in TNs. Given the number of experiments that were conducted, we generated many different results. In the following subsections, we report on a subset of the results that were the most relevant to answering our initial questions regarding Kubernetes on TNs.

A. Duration of bootstrapping phase for cluster initilization and application deployment

Fig. 2 shows the average time needed to initialize a cluster for all successful experiments. The horizontal grouping shows the number of nodes involved (6, 12, 18 or 24). Within the grouping, the network conditions are plotted on the horizontal axis.

In the LAN network scenario (considered as the baseline), K8s performs best for all cluster sizes. In general, we see that cluster initialization time increases when either the network quality degrades (within a horizontal group) or the cluster

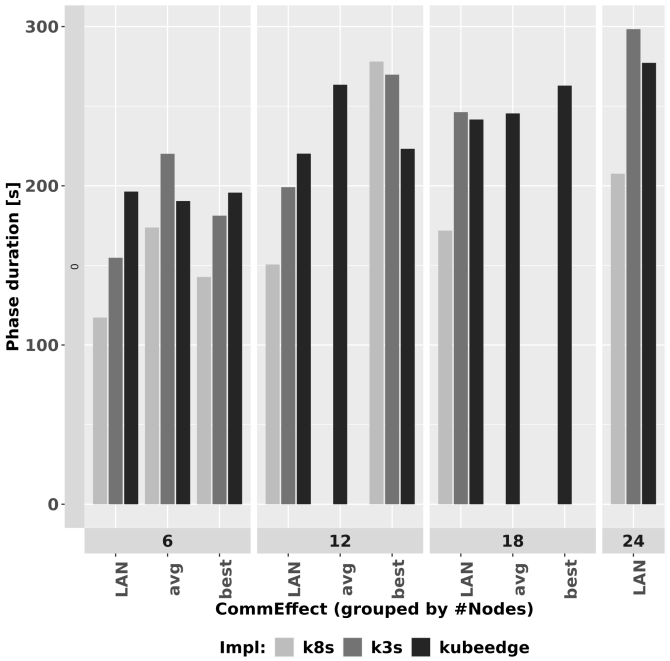

Fig. 2. Bootstrapping phase in the cluster initialization experiments.

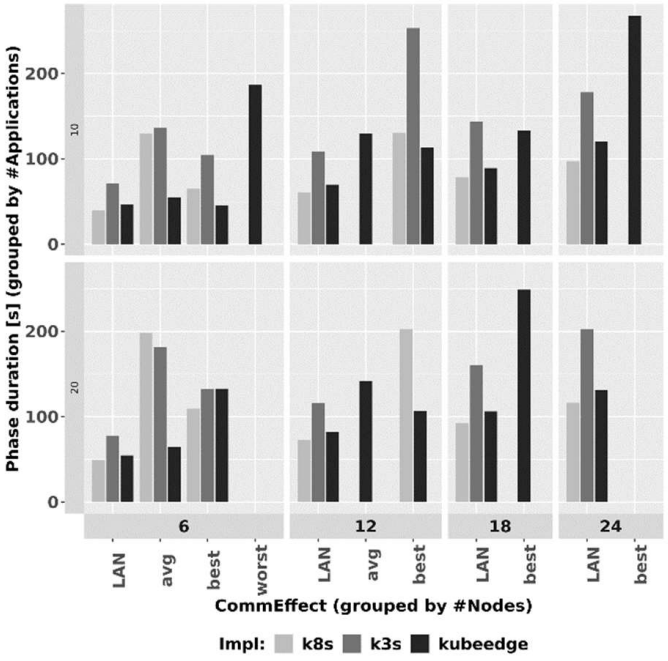

Fig. 3. Bootstrapping phase in the application deployment experiments.

gets bigger (over the horizontal groups). However, we also observe that the decline in performance is the smallest for all KubeEdge experiments. For example, in the experiments with 6 nodes and best network conditions we observe that K8s is fastest, followed by K3s and KubeEdge. When the cluster size increases to 12 nodes, we observe that KubeEdge is faster than both K8s and K3s. It shows that with increasing number of nodes or declining network quality KubeEdge is still able to initialize a cluster. In worst network scenario, like the experiments involving 12 nodes and avg network conditions, we observe that KubeEdge is the only distribution still able to initialize a cluster within the timeout.

Fig. 3 shows the bootstrapping time needed to deploy 10 or 20 application replicas on each node on an already initialized cluster.

In the LAN network scenario all three distributions behave roughly similar. However, in degrading network conditions KubeEdge holds up best, as can be seen in the experiments consisting of 6 nodes and avg network conditions. Also, KubeEdge manages to succeed in more experiment setups. In the following ones, 12 nodes avg, 18 nodes best, and 24 nodes best, KubeEdge is the only distribution that succeeded. We argue that this is justified by its enhanced cloud/edge communication architecture [12] and the underlying QUIC [13] protocol used for communication among cluster nodes, instead of TCP which is used by K8s and K3s. The enhanced cloud/edge communication states that QUIC provides:

"Dramatically reduced connection establishment time, improved congestion control, multiplexing without head of line blocking, forward error correction, and connection migration." [12]

B. Average data rate of bootstrapping phase for cluster initilization and application deployment

Fig. 4 and 5 show the average data rate for incoming and outgoing traffic for the bootstrapping phase for the master node. During the experiments there was no traffic between the worker nodes, therefore all analysis focus on the master node.

Based on the average data rate as shown in Fig. 4 and 5, KubeEdge's communication is not impacted as much compared to K8s or K3s by declining network quality. The graphs show that, in the LAN network scenario, the traffic increase for cluster initialization from 6 to 12 nodes is about 2.2 times larger for K8s, 2.3 times larger for K3s and 2.4 times larger for KubeEdge. For the best network scenario, the traffic is 4.4 times lager for K8s, 6 times larger for K3s and only 2.3 times larger for KubeEdge. Due to failed experiments for avg and worst network scenarios for K8s and K3s and for experiments with more than 12 nodes, a comparison is not possible in those cases.

As K8s and K3s use TCP as transport protocol we assume that the amount of transmitted data increases due to TCP and TLS handshakes and new handshakes once the connection breaks. In fact, as mentioned in [12], "in edge scenarios, network connectivity could be unstable. With TCP + TLS, it becomes an overhead to establish / re-establish connections frequently due to intermittent networks. In such scenarios, QUIC with its zero RTT can help reduce this overhead and re-establish broken connections faster."[12]

By combing the result of average duration and average data rate for cluster initialization and application deployment experiments, we argue that K8s and K3s can be used in

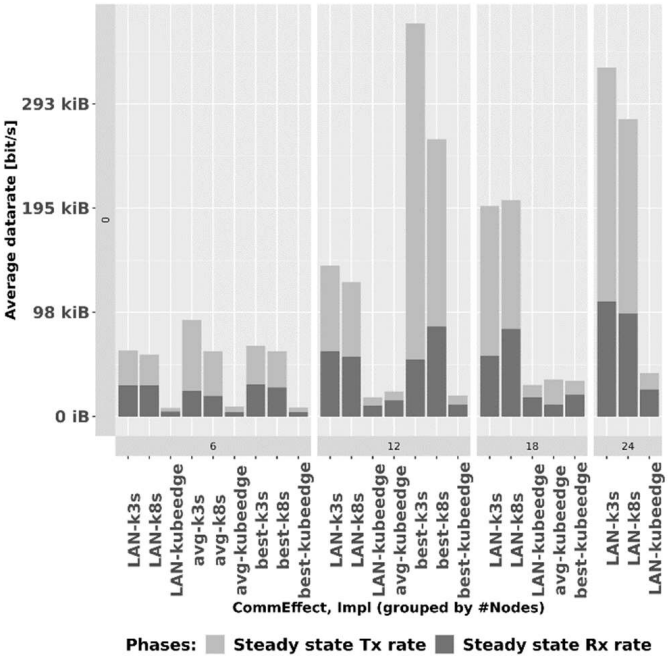

Fig. 4. Average data rate for bootstrapping phase in the cluster initialization experiments.

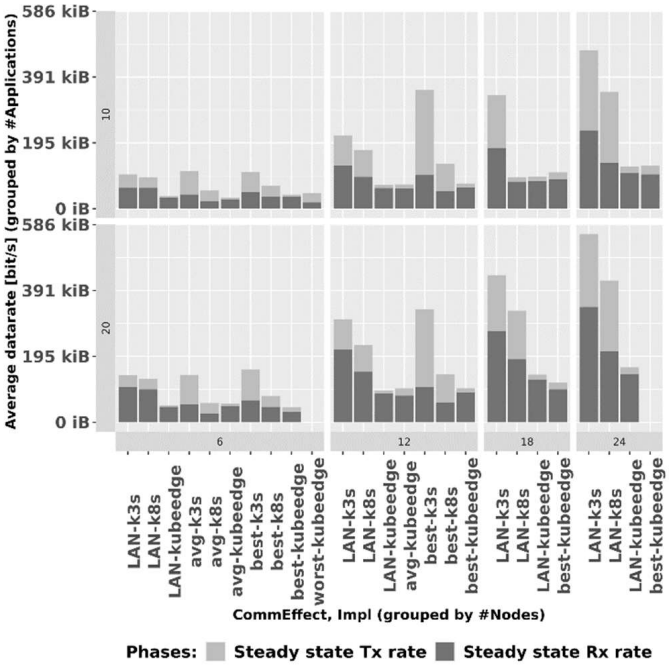

Fig. 5. Average data rate for bootstrapping phase in the application deployment experiments.

environments where up to 12 nodes participate in the cluster and the network quality is at least best (4Mbit, 95%, 200ms) by our definition. For KubeEdge it is possible to either increase the number of nodes up to 18 by still adhering to the network quality in the best network scenario, or limit the number of nodes to 12 and decline the network quality to the avg network

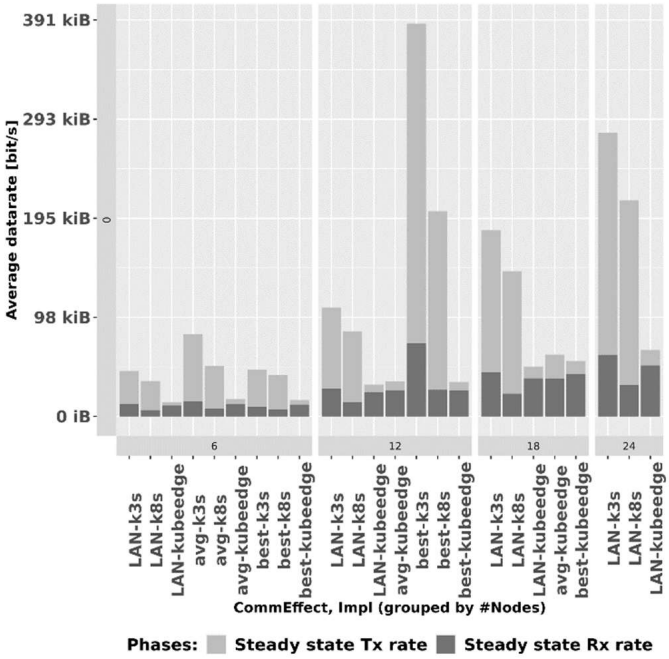

Fig. 6. Average data rate for steady state phase in the cluster initialization experiments.

scenario (2Mbit, 90%, 300ms).

C. Average data rate in the steady state for cluster initialization and application deployment

As depicted in Fig. 6 and 7, the data shows a similar pattern to the bootstrapping phases. KubeEdge uses lower bandwidth and is able to do cluster maintenance more efficiently. In some mission contexts, cluster initialization and initial deployment of applications can be done at headquarters prior to the mission starting. During a mission, sharing of mission relevant data in a timely manner depends amongst other things on the available bandwidth. As bandwidth is limited, maintaining a cluster by an orchestrator should be as bandwidth efficient as possible without sacrificing too many features.

The data shows that KubeEdge is highly efficient in reducing its necessary bandwidth in contrast to the other two distributions. Under the best network scenario, KubeEdge was successful with a mission setup with up to 12 vehicles being nodes and each of them running up to 20 applications over a TN. Although the best network scenario (4Mbit, 95%, 200ms) is quite optimistic for the current state of technology, it is still useful for comparison purposes as those experiments succeeded. KubeEdge never exceeds more than 74kBit/s for maintaining the cluster after application deployment. Whereas, K8s uses up to 102kBit/s and K3s up to 309kBit/s for the same task. In that setup, only 7.7% of the available bandwidth is used for cluster maintenance and the remaining amount should be enough for applications to transmit mission relevant data.

Additionally, the data shows that none of the experiments that succeeded overloaded the network bandwidth. Therefore, we assume that the success of experiments is more impaired

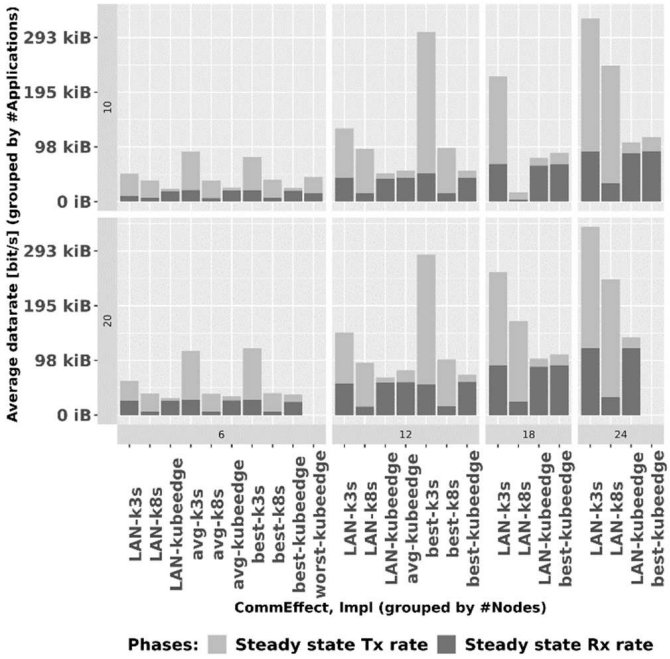

Fig. 7. Average data rate for steady state phase in the application deployment experiments.

by reduced reliability and latency then by reduced bandwidth. Taking that into account and by projecting the LAN network scenario data rate onto the avg and worst network scenario, i.e. only bandwidth is limited, then K8s would take up to 12.4%, K3s up to 17.2% and KubeEdge up to 7.1% of the available 2Mbit/s in the avg network scenario. In the worst network scenario K8s would consume up to 38%, K3s up to 52.9% and KubeEdge only 22.1% of the available bandwidth. This amount of cluster maintenance traffic for up 24 nodes and up to 20 applications has to be taken into account when planning current or future TN that uses Kubernetes as an orchestrator.

D. Experiment summary

The outcome of the experiments provides valuable insights regarding the performance of the three different Kubernetes distributions over varying TN network conditions. Examining the failures provides an indication about the limits on scalability as the network conditions degrade. Note that since each experiment was run three times, situations where the systems fail one or two times are defined as edge cases. Although edge cases should be investigated further, they certainly represent cases in which Kubernetes could not run smoothly.

To summarize the most notable results, we provide some considerations worth mentioning that are common to both cluster initialization and application deployment experiments. Under the worst network scenario, K8s, K3s and KubeEdge failed consistently. Therefore, Kubernetes cannot be deployed on TNs affected by such disadvantaged network conditions. Regarding the avg network scenario, the examined Kubernetes distributions succeeded while managing clusters of six nodes (the smallest cluster size considered in the experiments). As the cluster size grew to twelve nodes, experiments involving K8s and K3s failed, whereas those using KubeEdge succeeded. A similar pattern occurs in all the experiments, where K8s and K3s nearly mirror themselves in terms of succeeded and failed experiments. Interestingly, neither K8s nor K3s experiments succeeded when clusters were composed of more than twelve nodes and some communication effects were in place.

Concerning the application deployment experiments, KubeEdge substantially raised the bar by successfully managing clusters up to eighteen nodes in the best network scenario. Although still falling into an edge case, KubeEdge also distinguishes itself in terms of cluster initialization experiments. In fact, in both the best and avg network scenario, KubeEdge completed the experiments partially successfully (two successes out of three runs).

VI. Conclusion and Future Work

One of the primary objectives of this paper has been to evaluate the usability of different Kubernetes distributions in TNs with different network conditions. We showed that it is possible to run Kubernetes in TNs and that there are distributions that may be better suited to such environments, e.g. KubeEdge. Although KubeEdge has been released only recently in 2018 and is not as mature as K8s (released in 2014), its unique architectural approach and edge/cloud design principles makes it more performant and stable in TNs. Also, we produced evidence in favor of the so-called "Several Clouds per Mission Context" deployment model. In fact, a single Kubernetes instance would not be a feasible option for all the available mission context's resources. Both cluster initialization and application deployment experiments proved that as the network conditions worsen the cluster size gets limited. Therefore, several small clusters are preferable to a single large one. Lastly, we elaborated on the average bandwidth consumption for the Kubernetes distributions with different cluster setups and network scenarios. This can help with planning current and future mission networks.

Future work will use the results stated in this paper to further explore the usability of Kubernetes in TNs. This includes:

- Running experiments in a realistic scenario – So far experiments have been conducted in an artificial environment, i.e. no changes to the network by means of quality parameters over time and the deployed applications did not generated any traffic. Future experiments will deal with the performance and behavior of selected Kubernetes distributions by performing similar tests using the Anglova scenario [9].
- Exploration of edge cases in the experiments – Current experiments change all parameters of a network scenario at once. Additionally, the number of nodes and applications increases in steps of 6 nodes and 10 applications. Therefore, conducting similar experiments by only changing a single parameter at a time and in finer steps could give more detailed insight into the requirements in terms of bandwidth, reliability, and latency.

- Measuring performance under adaptation – One of the objectives of the IST-168 RTG is the dynamic adaptation of the workflows to accommodate changing mission requirements, resource availability, and network state. Future experiments will examine the ability of these Kubernetes distributions to handle adaptation and the associated network bandwidth.

VII. Acknowledgment

The work as presented in this paper is being done as part of the NATO IST-168 RTG on 'Adaptive information processing and distribution to support Command and Control'. We would like to thank the NATO IST Panel for providing us the opportunity to do this highly relevant and interesting research and the individual participating partners (as represented by this paper's authors) for providing valuable inputs within a stimulating and cooperative setting.

Finally, the raw captured data from all experiments as well as the necessary tooling, bash and R scripts, to reproduce our results can be made available by contacting the authors.

VIII. References

[1] H. Bastiaansen *et al.*, "Federated control of distributed multi-partner cloud resources for adaptive C2 in disadvantaged networks," *IEEE Communications Magazine*, vol. 58, no. 8, pp. 21–27, 2020, doi:10.1109/MCOM.001.2000246.

[2] A. Gibb *et al.*, "Data replication in low bandwidth military environments – state of the art review," in *Proc. 2000 Command and Control Research and Technology Symposium*, Naval Postgraduate School, Monterey, CA, 2000.

[3] J. G. Hollands *et al.*, "Cognitive load and situation awareness for soldiers: Effects of message presentation rate and sensory modality," *Human Factors*, vol. 61, no. 5, pp. 763–773.

[4] F. T. Johnsen *et al.*, "SMART II: Android apps, cloud computing and mobile device management as enablers for efficient operations," in *24th ICCRTS Symposium*, Laurel, MD, USA, Oct. 2019.

[5] Kubernetes, "Kubernetes concepts: Federation," https://kubernetes.io/docs/concepts/cluster-administration/federation/, 2019, accessed: Jan. 2020.

[6] K3S, "Light weight kubernetes – K3S distribution," https://k3s.io/, accessed: Jan. 2020.

[7] KubeEdge, "An open platform to enable edge computing," https://kubeedge.io/en/, accessed: Feb. 2020.

[8] N. Suri and A. C. Scott, "A perspective on defining the collective adaptive systems problem," 2014, doi:10.1109/SASOW.2014.12.

[9] N. Suri *et al.*, "The Angloval tactical military scenario and experimentation environment," 2018, doi:10.1109/ICMCIS.2018.8398729.

[10] EMANE, "Extendable mobile ad-hoc network emulator," https://github.com/adjacentlink/emane, accessed: Feb. 2020.

[11] Ansible, "Ansible," https://github.com/ansible/ansible, accessed: Jan. 2020.

[12] KubeEdge, "QUIC design," https://github.com/kubeedge/kubeedge/blob/master/docs/proposals/quic-design.md, accessed: Feb. 2020.

[13] IETF Internet Draft, "QUIC: A UDP-based multiplexed and secure transport," https://tools.ietf.org/html/draft-ietf-quic-transport-22#section-1, accessed: Feb. 2020.