Resilient Hybrid SatCom and Terrestrial Networking for Unmanned Aerial Vehicles

Paresh Saxena, Thomas Dreibholz, Harald Skinnemoen, Özgü Alay,
M. A. Vazquez-Castro, Simone Ferlin, and Guray Acar

***Abstract*—Today, Unmanned Aerial Vehicles (UAVs) are widely used in many different scenarios including search, monitoring, inspection, and surveillance. To be able to transmit the sensor data from the UAVs to the destination reliably within tangible response times to the relevant content is crucial, especially for tactical use cases. In this paper, we propose network coded torrents (NECTOR) to leverage multiple network interfaces for resilient hybrid satellite communications (SatCom) and terrestrial networking for UAVs. NECTOR is significantly different from the state-of-the-art multipath protocols such as multipath TCP (MPTCP) as it does not require any additional packet scheduler, rate-adaptation or forward error correction. We present the design and implementation of NECTOR, and evaluate its performance compared to MPTCP. Our experimental results show that NECTOR provides goodput (up to 70%) higher than MPTCP with 5.49 times less signaling overhead.**

***Index Terms*—Network coding, torrents, UAVs, drones, SatCom, cellular networks, multipath transmission, MPTCP.**

I. Introduction

The civilian Unmanned Aerial Vehicle (UAV), also known as "drones", market is taking off due to significant improvements in robotics and technology, which also lead to numerous new applications including search, monitoring, inspection and surveillance. Drones may often replace manned flights, by taking the risk of losing lives out of the equation. Many applications, such as pipeline monitoring, have high accident risks due to flights in difficult conditions, e.g. at low altitude with potential high wind gusts. Unmanned flights can be also conducted at night and for longer duration compared to manned flights.

Drones are equipped with different sensors, and users need best possible sensor data within tangible response times to the relevant content, especially for tactical use cases. For example, cameras are the most common sensors that are used either as the payload itself, for monitoring other operations, or for providing a remote pilot a First Person View (FPV) live video. Streaming video requires a good communication channel, and considering the drones often fly in very remote areas, the only reliable form of communications is via satellites. Satellite communications (SatCom) can be both, expensive and limited in capacity. Occasional cellular coverage via terrestrial networks may be available, but, in general, these are designed for ground and not aerial coverage. Therefore, there exists many coverage holes. Using satellite together with cellular when available will still provide gains even if the extent of the cellular availability is in general unknown. Thus, in this paper, we focus on the drone use case, and we study the benefits of using the combination of one reliable low-bandwidth SatCom with several potential high capacity cellular networks with less availability and reliability, in order to enable resilient and high capacity communication for the drones.

Simultaneous use of multiple networks can lead to significant performance improvement in terms of reliability, throughput, traffic offloading, improved quality of service, etc. [1]–[4]. However, combining multiple networks with heterogeneous characteristics is challenging. This challenge can partially be addressed with Multi-Path TCP (MPTCP) [5], which is a well-known protocol that leverages multiple networks simultaneously to provide reliability and bandwidth aggregation. However, MPTCP's widely adoption is hindered due several factors [6], and the protocol inherits several of the well-known challenges with TCP such as head-of-line blocking. Furthermore, strenuous efforts should be made to utilise the aggregated capacity when network conditions change rapidly, and for each network individually, since the scheduling decisions need a robust estimation of the capacity of each network, which, in a UAV scenario, may be rapidly changing.

In this paper, we present the design and implementation of Network Coded Torrents (NECTOR), an UDP-based application-level multipath solution. Our primary use case is a source node located remotely, e.g., an UAV, connected with a set of networks e.g. satellite, 3G/4G, radio, as shown in Fig. 1, sending data for tactical Intelligence, Surveillance and Reconnaissance (ISR) operational video. Although NECTOR is a generic protocol that can be used in other scenarios, in this paper we focus on addressing the challenge of providing resilient communications for UAVs that are equipped with multiple networks with varying coverage and capacity. Our network-coded torrent-based approach jointly tackle the above challenges using two key technologies: torrents [7] and Network Coding (NC) [8]. The torrent-based strategy requires no additional rate adaptation per network, since it is driven by

P. Saxena is with BITS Pilani, Hyderabad, India (e-mail: psaxena@hyderabad.bits-pilani.ac.in)

T. Dreibholz is with the Simula Metropolitan Center for Digital Engineering, Oslo, Norway (e-mail: dreibh@simula.no).

H. Skinnemoen is with AnsuR Technologies AS, Oslo, Norway (email: harald@ansur.no)

M. A. Vazquez-Castro is with Autonomous University of Barcelona, Barcelona, Spain (e-mail: angeles.vazquez@uab.es.)

Ö. Alay is with University of Oslo and Simula Metropolitan Center for Digital Engineering, Oslo, Norway (e-mail: ozgua@ifi.uio.no).

S. Ferlin is with Ericsson AB, Stockholm, Sweden (e-mail: simone.ferlin@ericsson.com).

G. Acar is with the European Space Agency, Noordwijk, The Netherlands (e-mail: Guray.Acar@esa.int).

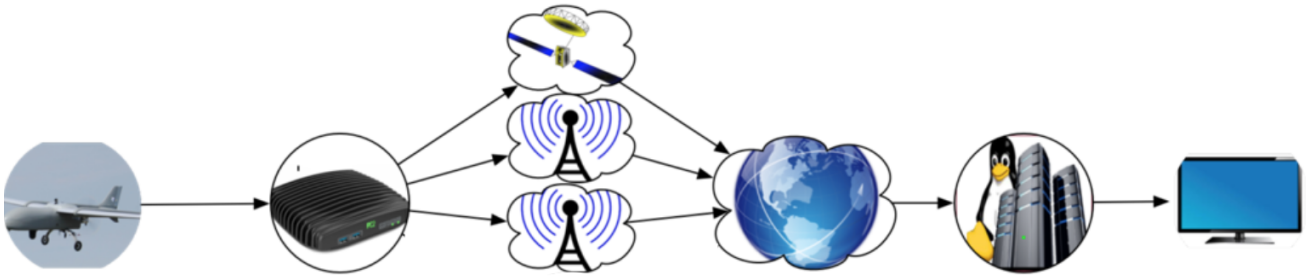


Fig. 1: Use case: UAV is sending data to a remote location over different networks including satellite and terrestrial networks.

the packet requests from the receivers. Additionally, there is no need for designing an efficient scheduler to distribute and track packets when NC is used, since linear combinations of the same packets are sent instead of uncoded packets.

The main contributions of this paper can be summarized as:

- We design and implemented an UDP-based application-level multipath data transfer solution NECTOR atop of two key technologies: torrents and NC. NECTOR is a comprehensive multipath networking solution without the need of scheduling, rate-adaption or forward error correction algorithms.
- We conduct extensive evaluations with NECTOR and compare its performance against MPTCP when multiple networks are available. We focus on the combination of low bandwidth and high reliable SatCom with cellular networks with lower availability and average reliability. Our results show that NECTOR provides up to 70% higher average goodput when the networks vary slowly but are unavailable for longer duration.

II. Related Work

There are several proposals leveraging the benefits of NC together with the benefits of multipath transport using MPTCP [9]–[14]. However, they are mainly focusing on the integration of NC on the TCP level to provide a sub-flow selection control policy for network-coded packets, i.e. NC is applied on each TCP connection individually. These solutions inherit several TCP shortcomings, e.g. poor performance over networks with high losses and high delays, and they are also unable to fully profit from the benefits of multipath transport. In [13] Pseudo-Random Network Coding (PRNC) is also proposed for MPTCP, with the goal of reducing the NC overhead and increase the overall throughput. However, PRNC has shown to not scale with re-encoding at the intermediate nodes [14]. One of the reasons is that while it is easy to select a seed for the encoding vector at the source, it is not straightforward to do the same for the re-encoding vector at intermediate nodes. To make this possible, an extension of PRNC requires the synchronisation of all nodes and potential additional complexity with large lookup tables.

For proposals integrating network coding with torrents, in [17], the authors investigate the performance of network coding and torrents for peer-to-peer content distribution networks followed by [18] and [19] for vehicular ad-hoc networks and Bluetooth, respectively. Furthermore, [20] and [21] verify the feasibility of a hybrid cellular and vehicular-to-vehicular (V2V) collaborative content distribution network. Finally, [22] studies the performance of chunked NC in wireless cooperative downloads. Our work departs from these, where we investigate the benefits of NC and torrents for content distribution over multiple network interfaces. NECTOR uses network coding to provide both resilience against network outages and to counter for packet losses. Following this area of work, we have compared our proposal with state-of-the-art MPTCP protocol.

III. NECTOR Design

In this section, we first describe the system model we considered and then we provide an overview of the NECTOR protocol and describe the torrents' operation. We then present how NC is incorporated into the system and then finally discuss the implementation aspects.

A. System Model

Let us consider that a source node, e.g., UAV is connected to a destination node, e.g. control center, via f networks as shown in Fig. 2. Each network is characterized by bandwidth, delay and packet loss. Let us denote b_i (in Kbps), d_i (in ms) and p_i (in %) as the bandwidth, delay and packet loss of the i-th network, respectively.

At the source node, the data is segmented into small files, referred as Datagrams (DGRAM). These datagrams are transmitted using torrent-based technology [7]. Each datagram is further segmented into several small chunks. Each chunk is segmented into several slices. Fig. 3 illustrates the integration of the NECTOR in the end-to-end protocol architecture.

B. Overview of NECTOR Protocol

An overview of the NECTOR protocol and its data units is presented in Fig. 3. For transmission, a DGRAM is encoded into chunks; each chunk is further encoded into slices. We assume the size of each datagram D is n_d (in bytes). We denote the j-th chunk as C_j, and its size as n_c (in bytes) where $j = 1, 2, \ldots, n$ with $n = \left\lceil \frac{n_d}{n_c} \right\rceil$ as the number of chunks per datagram. We denote the k-th slice as S_k and its size as n_s (in bytes) where $k = 1, 2, \ldots, m$ with $m = \left\lceil \frac{n_c}{n_s} \right\rceil$ as the number of slices per chunk. Then, we have $D = \begin{bmatrix} C_1 & C_2 & . & . & C_n \end{bmatrix}$ and $C_j = \begin{bmatrix} S_1 & S_2 & . & . & S_m \end{bmatrix}$.

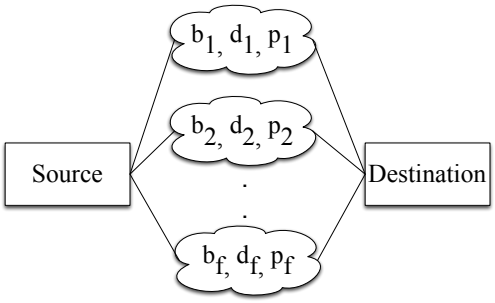


Fig. 2: System model illustrating the multipath communication between a source node and destination.

C. Torrents Operation

The operations of the NECTOR protocol with torrents can be summarised as follows: First, the torrent file that describes the hash table, which contains key values of chunks, is sent to the receiver. This assures the verification of integrity and authenticity. Torrents have a receiver-driven method, where it pseudo-randomly asks particular chunks or a set of chunks from the sender. Once these are received, new chunks are requested. In this way there is load balancing on all the networks and a slow network would send fewer packets [7]. The receiver-driven packet request mechanism removes most of the scheduling complexity at the sender side and the need to estimate the network capacity per network. Although the torrents can use any underlying protocol, we choose to use UDP. Compared to TCP, NECTOR ensures reliability at the application layer through NC.

D. Network Coding

Sending chunks via torrents over a network with possible outages and losses may reduce the performance. Therefore, to assure reliability, NC is performed and coded packets are generated. As illustrated in Fig. 3, NC is implemented at both, chunk and slice level. The chunk level NC assures the delivery of network-coded chunks via different paths without the need of any specific scheduling, while the slice level NC assures the reliability against packet losses. Each slice is then transported as UDP datagrams over the different networks to the receiver.

1) NC at Chunk Level: We call "chunk NC generation" a set of n chunks, denoted as X_c 2 $\mathbb{F}_q^{n_c}$ ^{n} with elements chosen from a finite field $\mathbb{F}_q$. We use rateless NC at chunk level. The encoder can generate a fountain of (coded) chunks and stop transmission only when it receives the signal from the receiver. This is the best for torrent-type and larger blocks, where there is time to stop or signal to continue sending more chunks.

2) NC at Slice Level: We call "slice NC generation" a set of K_s slices, denoted as X_s 2 $\mathbb{F}_q^{n_s}$ ^{K_s}. We assume per-slice NC generation block coding with block length N_s. We denote $\rho = \frac{K_s}{N_s}$ as slice-level coding rate.

In both cases, we assume each coded chunk is generated by a linear combination of the chunk NC generation with coding coefficients from either deterministically or randomly chosen columns of Pascal matrices P_q of size q q, see [23].

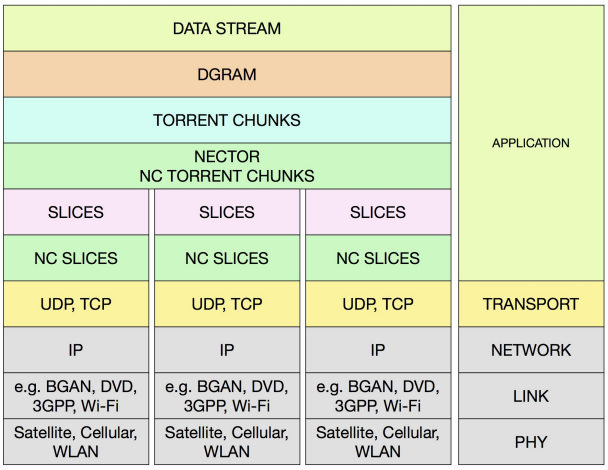


Fig. 3: NECTOR protocol stack and data units overview.

E. Implementation

NECTOR is implemented in C++ using libraries that make it independent of the underlying operating system. The sender side accepts DGRAMs from the application. The datagrams are passed by shared memory. Once there is a new datagram, the sender starts producing the chunks, creating the torrent file. The torrent file is provided to a tracker (fixed at known addresses), which signals the availability of the new file to the receiver. Then, the receiver starts to initiate chunk transfers by requesting a chunk over each path. The sender transmits a chunk over a path as bursts of its configured number of slices. Since each path transports only *one* chunk at a time, there is no need for congestion control. Note, that the paths are independent and – in case of UAV operations – each path is exclusively usable. Once a chunk is decodable (i.e. a sufficient number of NC-coded slices has been received to successfully compute the chunk), or no further slice has been received within a configured timeout (i.e. the chunk transport has failed), a new chunk is requested by the receiver on the path. The receiver signals the sender to stop the entire transmission once it has a sufficient number of NC-coded chunks received to decode the full datagram. If there is a new torrent file available while the transmission is still in progress, the current transmission is aborted (since the datagram became obsolete, and continuation would be useless) and the transfer of the new datagram is initiated instead. The datagram is then passed by shared memory to the application.

IV. Measurement Setup and Performance Metrics

In order to better understand the impact of different network settings on NECTOR, we describe our constructed testbed together with the experimental setting in Subsection IV-A and the performance metrics in Subsection IV-B.

A. The Testbed Setup

We built a small scale testbed to evaluate the performance of NECTOR over off-the-shelf products that can be used in UAVs. Figure 4 shows our testbed consisting of two fanless mini-computers IPC3 (Intel Core i7@2.7 GHz, 4 GiB

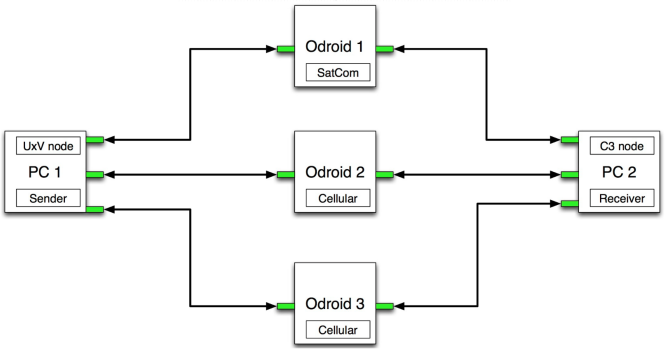


Fig. 4: Testbed setup

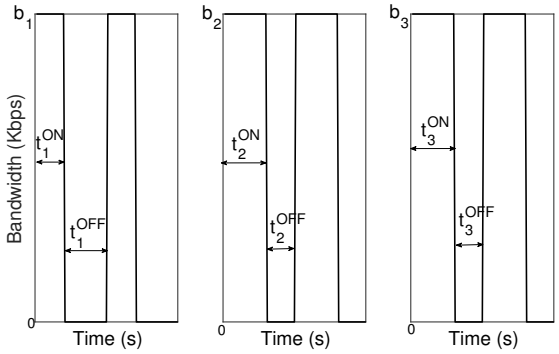


Fig. 5: Two-state on-off model for bandwidth variations

SDRAM) and three single board computers Odroid-XU4. The mini-computers are used as sender and receiver, while Odroids are used for network emulation. Two IPCs are connected via three Odroids via Ethernet. All Odroids run Linux Traffic Control (tc) with Network Emulation (netem) and the Token Bucket Filter (tbf) queuing discipline to limit bandwidth, vary delay and add network impairments such as packet loss. For most of the measurements, we focus on the scenario in Fig. 4 with three network paths and sender and receiver with three interfaces. All machines run Ubuntu Linux 16.04, and the IPC3s run the Linux kernel v4.19 with Linux MPTCP v0.95.

We consider four scenarios as shown in Table I. In each scenario, network 1 is configured to behave as a satellite link while networks 2 and 3 are configured to behave as low bandwidth cellular links, mimicking 2G/3G coverage in rural areas. The bandwidth and delay characteristics of network 1, 2, and 3 are illustrated in Table II. The experiments' bandwidth settings follow the variations shown in Fig. 5 with a two-state on-off model: The i^{th} network is available during t_i^{ON} and unavailable during t_i^{OFF}. In other words, the bandwidth is b_i during t_i^{ON} and 0 during t_i^{OFF}.

Furthermore, we have considered that network 1 (satellite link) is always available while networks 2 and 3 availability (cellular links) vary over time based on Fig. 5. Such a configuration is common in cellular networks as 2G/3G networks can have coverage holes, especially in high altitudes and rural areas, and as the UAVs travel, it will get in and out of coverage. The on/off time duration is configured such that network 2 is mostly available (66.67% of total time) and network 3 is hardly available (33.33% of total time) in scenarios 1, 2 and 3. In scenario 1, we are emulating fast bandwidth variations and in scenario 3 we are emulating slow bandwidth variations. Note that in case of scenario 1, 2 and 3, we considered that all three networks are available at the start of the experiments. Finally, we have also considered the scenario where on/off time duration varies randomly between 1 and 10 seconds. In this case, $t_i^{ON} = rand(1, 10)$ and $t_i^{OFF} = rand(1, 10)$, i.e. they take random values between 1 and 10 seconds.

B. Performance Metrics

The total time taken to transfer a datagram from the sender to the receiver is given by T (in s). The application layer performance is defined as goodput (γ) (in Kbps) as the number of useful information (n_v) delivered at the receiver per unit of time:

$$\gamma = \frac{8 \quad n_v}{T}. \tag{1}$$

The utilization ratio (θ) is given as the goodput divided by the sum of the total bandwidth of all f networks:

$$\theta = \frac{\gamma}{\sum_i b_i(1 \quad p_i)}. \tag{2}$$

The receiver has f interfaces corresponding to f networks, where n_i^r as is the number of bytes received, and n_i^s as is the number of bytes sent by the i^{th} interface of the sender. The total number of bytes at the receiver are: $n_{all}^r = \sum_i n_i^r$, where overhead (η) is defined as additional bytes $n_{all}^r \quad n_d$ divided by the datagram size n_d:

$$\eta = \frac{n_{all}^r \quad n_d}{n_d}. \tag{3}$$

Finally, we also define μ_i as the percentage of network traffic load shared by the i^{th} network:

$$\mu_i = \frac{n_i^r}{n_{all}} \quad 100. \tag{4}$$

V. Results

In this section, we will first discuss the optimal selection of n and then present the performance of NECTOR as compared to MPTCP for the different scenarios discussed in Section IV. We further discuss NECTOR's performance under lossy path characteristics and discuss the limitations of the protocol.

Optimal selection of n: We first study the impact of n on the performance of NECTOR. In order to find the optimal n, we run some preliminary experiments. We chose n 2 f8, 16, 32, 48, 64, 80, 96, 112, 128g and observe that the goodput is comparatively higher from NECTOR when $n = 32$ and $n = 48$. In general, we observe that the larger the number of chunks, the smaller is the goodput. The reason is that the encoding and decoding complexity of the network coding is proportional to the number of chunks [24]. When the number of chunks is higher, the overall encoding and decoding time is higher which increases the overall time duration. Hence, the goodput decreases with the increase in the number of

Scenarios	*Network 1 (SatCom)*	*Network 2 (Cellular)*	*Network 3 (Cellular)*
Scenario 1	$t_1^{ON} = \infty, t_1^{OFF} = 0$	$t_2^{ON} = 10, t_2^{OFF} = 5$	$t_3^{ON} = 5, t_3^{OFF} = 10$
Scenario 2	$t_1^{ON} = \infty, t_1^{OFF} = 0$	$t_2^{ON} = 20, t_2^{OFF} = 10$	$t_3^{ON} = 10, t_3^{OFF} = 20$
Scenario 3	$t_1^{ON} = \infty, t_1^{OFF} = 0$	$t_2^{ON} = 30, t_2^{OFF} = 15$	$t_3^{ON} = 15, t_3^{OFF} = 30$
Scenario 4	$t_1^{ON} = \infty, t_1^{OFF} = 0$	$t_2^{ON} = rand(1, 10), t_2^{OFF} = rand(1, 10)$	$t_3^{ON} = rand(1, 10), t_3^{OFF} = rand(1, 10)$

TABLE I: Four scenarios with different on/off time duration.

Network 1 (SatCom)	*Network 2 (Cellular)*	*Network 3 (Cellular)*
$b_1 = 150\,Kbit/s$	$b_2 = 1000\,Kbit/s$	$b_3 = 1000\,Kbit/s$
$d_1 = 250\,ms$	$d_2 = 50\,ms$	$d_3 = 50\,ms$

TABLE II: Network Parameters for Satcom and Cellular links

chunks. However, the values that we observe are specific to the computational power (hardware) used in the experiments. These values may change if machines with higher/smaller processing capabilities are used but we expect a general trend results where the goodput will decrease with the increase in the number of chunks. Due to space constraints, for the remainder of the paper, we present the results and analysis only for $n = 32$ and $n = 48$.

Performance evaluation for different scenarios with on/off model: We now focus on NECTOR's performance evaluation compared to MPTCP for the scenarios in Table I with network settings in Table II. We measure goodput for both NECTOR and MPTCP, and the goodput percentage gain and the network utilization ratio for NECTOR in Table III. The datagram size is configured as $n_v = 10\,Mbyte$, i.e. video encoded at 500 Kbit/s for 160 seconds and we have limited ourselves to 50 iterations for each experiment. In our experiments, NECTOR achieves higher utilization ratio, with 66.95% compared to only 46.69% of MPTCP in scenario 3. As a consequence of better utilization, NECTOR also consistently achieves a higher goodput than MPTCP in all scenarios. We observe that, NECTOR achieves up to 70% and 43.38% higher goodput compared to MPTCP in scenario 2 and scenario 3 respectively, when the networks vary slowly and then become unavailable for longer duration. We observed in the experiments that with slow variations, MPTCP seems to take time for ramping up the transfer speed, while NECTOR utilizes most of the capacity when the networks are available.

In Figure 6 we present the following metrics measured at the receiver for all scenarios: (i) total number of bytes received by each interface, i.e., n_i^r (ii) total number of bytes sent by each interface, i.e., n_i^s and (iii) percentage of load shared by each interface, i.e., μ_i. Our results indicate that the total number of bytes at the receiver, i.e., n_r^{all}, is smaller for NECTOR compared to MPTCP. In other words, NECTOR requires less data to recover the datagram of the same size. In scenario 3, the total number of bytes received with MPTCP is $n_{all}^r(MPTCP) = 11.9$ Mbyte, whereas the total number of bytes received with NECTOR is $n_{all}^r(NECTOR) = 10.7$ Mbyte. Thus, MPTCP requires 11.21% more data. We also show that the number of bytes sent by the receiver, i.e. signalling data, is much smaller with NECTOR compared to MPTCP. In scenario 3, the total number of bytes sent with MPTCP is $n_{all}^s(MPTCP) = 0.39$ Mbyte whereas the total number of bytes received with NECTOR is $n_{all}^s(NECTOR) = 0.071$ Mbyte. Thus, the signaling overhead with MPTCP is almost 5.49 times larger.

Finally, we note that even with smaller capacity, the satellite link carries a substantial amount of traffic, since it is always available in all scenarios. For scenarios 1, 2 and 3, the satellite link carries between 15% and 30% of the traffic. In scenario 4 with random network variations, the satellite link carries up to 40% of traffic, due to more abrupt and frequent disruptions of the other network paths (cellular networks).

Performance under Lossy Networks: NECTOR is shown to provide gains in other scenarios as well. For example, we have considered a different scenario, where instead of using on/off bandwidth variations, we considered networks with constant availability but have lossy characteristics. In this scenario, as expected, the benefits of NECTOR evident. NECTOR can provide up to 88.76% higher average goodput as compared to MPTCP when 10% losses are configured in each network. Even when losses are configured between 0% to 1%, NECTOR seems to provide up to 5%-6% higher average goodput than MPTCP. Due to space limitation, these results have not been presented in this paper.

Limitations: We have also encountered limitations of NECTOR. We observe that for higher bandwidth values, the transmission time is smaller and therefore the network coding/decoding time of NECTOR shares a larger percentage of the overall time duration. The encoding/decoding complexity of network coding results in smaller utilization of the aggregated bandwidth. Since these results depend on the hardware used for testing, a better configuration of hardware can yield to better performance of NECTOR as compared to MPTCP.

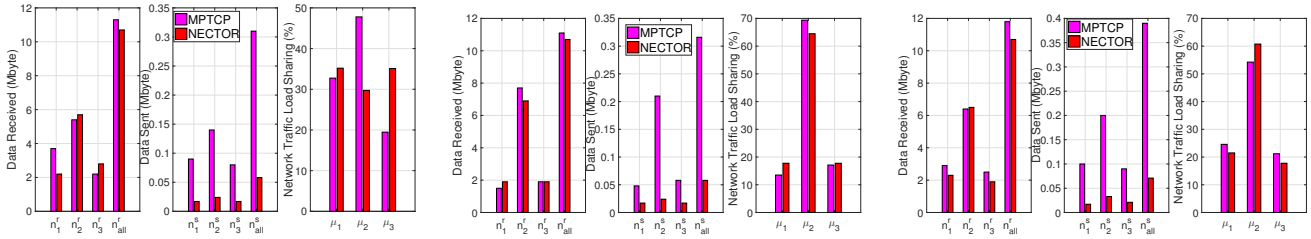
VI. Conclusion

In this paper, we have proposed NECTOR, which is a powerful network coding solution that utilizes the aggregated capacity of available multiple networks for transmission. We have presented our experimental results that show the benefits of NECTOR as compared to state-of-the-art MPTCP. In particular, our results show the performance gain of NECTOR in different scenarios representing hybrid SatCom and terrestrial networking for UAVs.

Future work includes the investigation of NECTOR on more complex use cases including the transmission of sensor data from UAV to the convoy (one-to-many connections).

TABLE III: Goodput from NECTOR (with $n = 32$ and $n = 48$) and MPTCP for different scenarios

Scenarios	γ (NECTOR) n = 32	γ (NECTOR) n = 48	γ (MPTCP)	Gain with NECTOR n = 32	Gain with NECTOR n = 48	θ (NECTOR) n = 32	θ (NECTOR) n = 48	θ (MPTCP)
Scenario 1	450	508	399	12.78%	27.31%	39.13%	44.17%	34.69%
Scenario 2	734	690	431	70.30%	60.09%	63.82%	60%	37.47%
Scenario 3	770	731	537	43.38%	36.12%	66.95%	63.56%	46.69%
Scenario 4	309	342	250	23.36%	36.80%	26.86%	29.73%	21.73%



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 6: Total number of bytes received, sent and network traffic load sharing from NECTOR ($n = 32$) and MPTCP.

Furthermore, we plan to study the impact of several other factors including the limitations of network codes and bursty network traffic on the performance of NECTOR.

ACKNOWLEDGMENT

This work has been supported by the European Space Agency, under the contract number 4000118143/16/NL/EM (HENCSAT), European Union's Horizon 2020 research and innovation programme under grant agreement No 815178 (5GENESIS) and SERB, DST, Government of India's start-up research grant agreement SRG/2019/002027 (MUT-DROCO).

REFERENCES

[1] A. Nikravesh, Y. Guo, F. Qian, Z. M. Mao, and S. Sen. An In-depth Understanding of Multipath TCP on Mobile Devices: Measurement and System Design. In Proc. of ACM MobiCom, 2016.

[2] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen. Accelerating Multipath Transport Through Balanced Subflow Completion. In Proc. of ACM MobiCom, 2017.

[3] Sana Habib, Junaid Qadir, Anwaar Ali, Durdana Habib, Ming Li, and Arjuna Sathiaseelan. The past, present, and future of transport-layer multipath. J. Netw. Comput. Appl. 75, C (November 2016), 236-258.

[4] K. V. Yedugundla, S. Ferlin, T. Dreibholz, Ozgu Alay, N. Kuhn, P. Hurtig, and A. Brunstrom, Is Multi-Path Transport Suitable for Latency Sensitive Traffic? Computer Networks, vol. 105, pp. 1-21, Aug. 2016.

[5] Ford A., Raiciu C., Handley M., Bonaventure O. TCP Extensions for Multipath Operation with Multiple Addresses. RFC; 2013.

[6] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How hard can it be? Designing and implementing a deployable multipath TCP. In Proc. USENIX Conference on Networked Systems Design and Implementation (NSDI), 2012.

[7] B.Cohen, "Bit Torrent Protocol1.0," BitTorrent.org, Tech.Rep.

[8] D. S. Lun, M. Medard, R. Koetter, M. Effros, "On coding for reliable communication over packet networks", Phys. Commun., vol. 1, no. 1, pp. 3-20, Mar. 2008.

[9] S. Gheorghiu, A. L. Toledo, and P. Rodriguez, "Multipath TCP with network coding for wireless mesh networks," in Proc. IEEE International Conference on Communications (ICC), 2010.

[10] X. Zhuoqun, C. Zhigang, Y. Hui, and Z. Ming, "An improved MPTCP in coded wireless mesh networks," in Proc. IEEE International Conference on Broadband Network & Multimedia Technology (IC-BNMT), 2009.

[11] Z. qun Xia, Z. gang Chen, Z. Ming, and J. qi Liu, "A multipath TCP based on network coding in wireless mesh networks," in Proc. IEEE International Conference on Information Science and Engineering (ICISE), 2009.

[12] A. Kulkarni, M. Heindlmaier, D. Traskov, M.-J. Montpetit, and M. Médard, "An implementation of network coding with association policies in heterogeneous networks," in Proc. IFIP TC International Conference on Networking (NETWORKING'11), 2011.

[13] J. Cloud, F. du Pin Calmon, W. Zeng, G. Pau, L. M. Zeger and M. Medard, "Multi-Path TCP with Network Coding for Mobile Devices in Heterogeneous Networks," IEEE Vehicular Technology Conference (VTC Fall), Las Vegas, NV, 2013.

[14] G. Giambene et al., "Network coding applications to high bit-rate satellite networks," WiSATS 2015, LNICST 154, July 2015.

[15] S. Ferlin, S. Kucera, H. Claussen and O. Alay, "MPTCP meets FEC: Supporting Latency-Sensitive Applications over Heterogeneous Networks", IEEE/ACM Transactions on Networking, Volume 26 Issue 5, October 2018.

[16] H. Skinnemoen, "Visual Situational Awareness: Revolutionizing UAV Communication Via Satellite", AUVSI Xpoential, 2018.

[17] C. Gkantsidis and P. Rodriguez, Network Coding for Large Scale Content Distribution, in Proc. IEEE INFOCOM, 2005.

[18] U. Lee, J.-S. Park, J. Yeh, G. Pau, M. Gerla, "Code Torrent: Content Distribution Using Network Coding in VANET", in "Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking", 2006.

[19] S. Jung, U. Lee, A. Chang, D.-K. Cho, M. Gerla, "BlueTorrent: Cooperative content sharing for Bluetooth users", in "Pervasive and Mobile Computing (PerCom)", no. 6, pp. 609–634, 2007.

[20] R. Zhang, B. Yu, H. Krishnan, "Simulation Study on Collaborative Content Distribution in Delay Tolerant Vehicular Networks", in "EEE 88th Vehicular Technology Conference (VTC-Fall)", 2018.

[21] D. Recharte, A. Aguiar, H. Cabral, "Cooperative Content Dissemination on Vehicular Networks", in "IEEE Vehicular Networking Conference (VNC)", 2018.

[22] X.-x. Wen, H.-q. Wang, J.-y. Lin, G.-s. Feng, H.-w. Lv, J.-z. Han, "Performance analysis and optimization for chunked network coding based wireless cooperative downloading systems", in "Frontiers of Information Technology Electronic Engineering", 2017.

[23] M. A. Vázquez-Castro, P. Saxena, T. Do-Duy, T. Vamstad and H. Skinnemoen, "SatNetCode: Functional Design and Experimental Validation of Network Coding over Satellite," 2018 International Symposium on Networks, Computers and Communications (ISNCC), 2018.

[24] P. Saxena and M. A. Vázquez-Castro, "DARE: DoF-Aided Random Encoding for Network Coding Over Lossy Line Networks," in IEEE Communications Letters, vol. 19, no. 8, pp. 1374-1377, Aug. 2015.