Aerial Video Stream over Multi-hop using Adaptive TDMA Slots

Luis Ramos Pinto*†, Luis Almeida*, Hassan Alizadeh* and Anthony Rowe†
* Instituto de Telecomunicações, Faculdade de Engenharia, Universidade do Porto, Portugal
† Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

***Abstract*—Unmanned Aerial Vehicles (UAVs) are rapidly becoming an important tool for applications like surveillance, target tracking and facility monitoring. In many of these contexts, one or more UAVs need to reach an area of interest (AOI) while streaming live video to a ground station (GS) where one or more operators inspect the AOI and carry out fine control of UAVs position. In remote areas, intermediate UAVs can act as relays and form a line network to extend range. Interactive control requires a live video stream where both throughput and delay are important. In this paper, we show that routing packets over CSMA/CA (native medium access protocol of WiFi, the most common wireless technology among UAVs) behaves poorly in this context due to link asymmetries. We propose a novel distributed, adaptive and self-synchronized TDMA protocol (DVSP) that both enhances delay and packet delivery while operating on commodity hardware and leveraging a standard UDP/IP protocol stack. We prove that DVSP converges to a global solution that minimizes delay using local information, only, thus in a fully distributed manner. Real world experiments with multiple UAVs show gains in delay up to 75%, and packet delivery up to 50%, without sacrificing goodput.**

***Index Terms*—delay; IEEE 802.11; multi-hop; relay network; TDMA; throughput; UAV; WiFi; wireless networks**

I. Introduction

Inspection of large-scale structures like bridges and towers, as well as search-and-rescue in areas affected by disasters can benefit tremendously from remotely operated drones. In these scenarios an operator located at a ground station often needs to fine-tune the position of drones and sensors in order to improve sensing resolution in certain areas of interest. Ad-hoc communication between drones [1] offers a viable alternative compared to infrastructure networks, e.g., cellular, in terms of availability, reliability and/or cost.

Unmanned Aerial Vehicles (UAVs) are increasingly becoming a commodity to support these types of inspection tasks given their ad-hoc WiFi (IEEE 802.11) native support, and inclusion of cameras and other sensors in their hardware. Unfortunately, video streaming on standard WiFi performs poorly, particularly with relays, resulting in long delays and lost frames. Typically, the losses are compensated using complementary IP-based technologies, such as TCP/IP, at the cost of a severe and unpredictable impact on delays.

In this paper, we design and analyze a new data-link protocol optimized for multi-hop online video streaming applications. Our system provides soft real-time guarantees in terms of delay such that operators can interactively pilot UAV fleets while maximizing reliability to provide reasonable video Quality-of-Service (QoS). Figure 1 shows an example scenario where two UAVs (sensors) are being manually controlled to track desired features in the area of interest using video streams. Meanwhile, other UAVs are relaying the streams across the network to the Ground-Station. We first show that a naive solution that uses commodity WiFi hardware on commercial multirotor UAVs struggles in terms of both reliability and timeliness. We then propose an overlay Time Division Multiple Access (TDMA) protocol for use over WiFi that self-synchronizes transmitters (based on RA-TDMA [2]) and supports multi-hop video routing. Our protocol adapts the length of the TDMA slots in a distributed fashion to minimize in-network queuing, which is the primary cause of network delay. By controlling the length of the TDMA slots according to the status of their associated transmitters, we can mitigate throughput asymmetries among network links and achieve end-to-end throughput equalization. We analytically and experimentally show the advantage of our TDMA protocol on a four-hop network of quadrotor UAVs streaming video, when compared to a naive approach based on using WiFi directly. Thus, our contributions are:

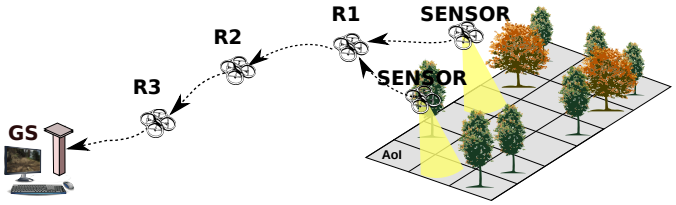

Fig. 1. Multi-source aerial stream to a ground sink, resorting to multiple UAV relays to transmit data.

- DVSP - a new TDMA framework that adapts its slots using a model of actual link bandwidth,
- A proof of DVSP convergence under distributed operation,
- Experimental validation with real UAVs.

The paper is organized as follows. Section II discusses related work. Section III states the problem we are addressing, followed by the solution we propose in Section IV, namely DVSP. Section V presents the architecture of our implementation, and Section VI shows the experimental results that confirm the expected improvements. Section VII concludes the paper.

II. Related work

Several authors have developed numerous UAV test-beds for commercial, military and research purposes. Many of these have explored using UAVs as flying wireless sensor networks, especially comprising cameras. In many works, such as [3], UAVs are used to collect aerial imagery for mapping and localization. This can be performed locally if UAVs have enough computing power or remotely using a cloud infrastructure if a connection is available. In tasks such as monitoring or target tracking, a human operator is often the end user that takes final decisions. This means that the UAVs should be able to form a network and stream video from a remote location. In [4], authors describe multiple ways of organizing UAVs to form a sensor network, whether the sensors can be disconnected from the base station for sometime, or whether all sensors need to remain directly connected to the base station, or even whether relays can be used to keep connectivity while increasing the communication range. Multi-hop UAV networks are therefore a known tool for remote sensing but a thorough analysis on network metrics is missing. The work in [5] shows a UAV wireless multi-source video stream use case. Despite using a network topology different from a line and the sink being another UAV, not a ground station, the authors show that transmitters should adapt their PHY rate to improve throughput, depending on the network load and link conditions. In contrast to our work, none of these approaches explore the impact on delay of buffered packets at intermediate relays with a loaded network.

The authors in [6] analyze network throughput using one relay to transmit pictures to a base station. With only a single relay added, the buffering problem was not clearly identified. We see identical performance in our two-hop network experimental results, which degrades significantly for three hops and beyond, with the introduction of hidden terminals.

Using TDMA to guarantee timeliness has been studied extensively as it is a technique that grants all nodes a guaranteed periodic transmission window, called a slot, thus preventing mutual interference and associated phenomena like starvation. Changing slot size to improve network metrics has been studied before, as in [7], but not applied to a multi-hop aerial line network, where data is generated at one tip of the network, only, and relayed through the other nodes, over links that present variable throughput. Most wireless sensor networks using TDMA, focus on guaranteeing that all nodes can communicate their own data. The fact that, in our work, middle nodes are solely relaying data, but their links can present variable throughput, makes the system prone to inefficiencies when using traditional TDMA approaches. Relays in our network only require a time slot long enough to relay incoming data while minimizing in-network queuing; hence slots should adapt to overall network throughput.

Other works such as [8] and [9] clearly identified distance as the main factor of packet delivery ratio (PDR), in fixed sensor networks. However, in most such works, there is no on-line stream of sensor data; packets are sent scarcely, not generating queuing issues. When streaming data intensely, as in our case, buffer overflow becomes a strong problem and a potential cause of PDR degradation, requiring adequate traffic management to avoid stalling the network.

Another domain where related research works can be found is that of robotic networks. For example, the work in [10] investigates how robots motion can be controlled in order to maintain high throughput for streaming data to a base-station using a multi-hop network. They conclude that, instead of transmitting from every point directly to a gateway, it is better to concentrate transmissions in areas where/when the channel is good, slowing the robot, and then moving faster in areas with poor channel characteristics. The focus of this work is on the robot's mobility control and not on the network configuration. The paper does motivate the variability and asymmetry of wireless links, which we also consider. In [11], the authors analyze mobile robotic networks performance as a function of distance from a base station and required data-rate/delay requested by users. They also consider the implications of using relay nodes. However, when robots move far from the base, the authors propose swapping to a data mule model that leverages delay tolerant networking, giving away the live connection to the base. Although other researchers have also explored different UAV network operation modes, most tolerate breaking base connectivity, which is incompatible with the live streaming scenarios we consider. Moreover, such works do not provide any experimental data on delay and PDR over single or multi-hop links, which in our case is paramount.

The works in [12] and [13] address a similar purpose as ours, in the sense of aiming at establishing a line topology of relays to support a live multimedia connection. However, they focus on the specific characteristics of tunnels and pipelines which, under certain circumstances, behave like wave-guides. This makes the results of these works inapplicable to our scenario of operation in semi or wide open areas, but corroborates the need for remote live streaming. The work in [14] also shows relationship to ours since it analyses the behavior of multi-hop networks under TDMA versus CSMA/CA. This work addresses networks in general, focusing on a small scale case, and the authors conclude that, depending on payload size and slot length, both medium access control techniques can dominate one another in terms of worst-case network delay.

Our work is the first to propose an adaptive overlay TDMA framework on-top of CSMA/CA links in a mobile line relay network, that keeps TDMA cycles constant, but adjusts slots dynamically in a distributed fashion in order to minimize end-to-end delay.

III. Problem Statement

Consider a multi-hop wireless network architecture with n UAV nodes and a sink, as illustrated in Figure 2 where the aim is to deliver real-time message streams, such as live videos, produced by one or more sources to a unique sink, i.e., a ground station, through a line of $n-1$ relays. We have studied before [1] the delay-range trade-off implied by using UAV relays to connect a live video source to a ground station. Each

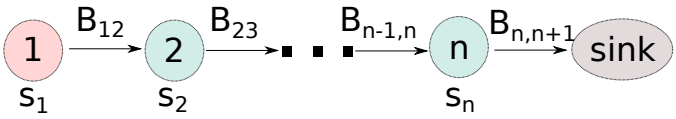

Fig. 2. Multi-hop Line Network model. Bandwidth of each link is represented by $B_{i,j}$. s_i represents the units of time (time slot) available to each node to transmit periodically every T.

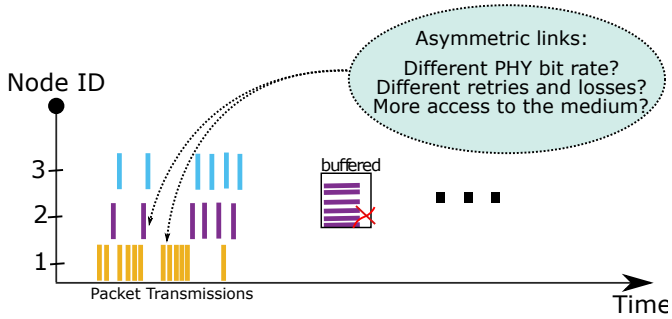

Fig. 3. Inefficiencies such as buffered packets and wasted bandwidth are created when all nodes are allowed to transmit concurrently and asynchronously, and links have different characteristics such a PHY rate or packet loss. The source node 1 is transmitting faster than node 2 can cope.

relay provides additional range but it must use a buffer to hold received packets, which are forwarded later to the following node, downstream in the line topology, thus adding delay. However, transient reductions of the outgoing packet rate with respect to the incoming rate require further increasing the buffer depth that, in turn, increases the end-to-end network delay. This delay must be below a certain deadline so that an operator in the base station can still interact with the sensor(s) UAV(s) through the live video stream(s) effectively. This corresponds to upper bounding the buffer depth.

For comparison purposes, we establish a baseline case in which the source generates as much data as it can transmit to its immediate neighbor, i.e., the first relay. In turn, relays forward immediately every received packet to the next hop, subsequently closer to the sink. In this case, transmissions are carried out using the native distributed and asynchronous CSMA/CA arbitration of WiFi, without any further control.

As expected, this approach quickly degrades under high load. Transient bandwidth asymmetries between the links of each relay lead to packet buffering, longer delay and eventually to overflow and packet losses. Increasing buffer size is not a solution, as it will increase end-to-end delay. Figure 3 illustrates this situation with asymmetric links resulting from either different PHY rates, asymmetric antennas, localized interference generating asymmetric packet loss that leads to different retries at the MAC level, or simply because some node accesses the medium more often under the CSMA/CA random arbitration. Furthermore, the line topology with concurrent asynchronous network access is prone to hidden nodes, which can contribute to degrade the network performance even more. Both buffer overflow and hidden nodes decrease link PDR and lead to high end-to-end delays. As a consequence

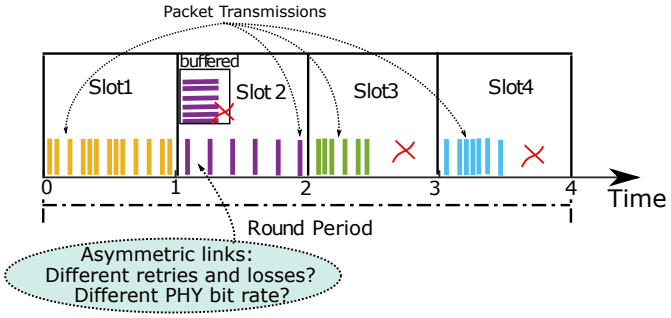

Fig. 4. Inefficiencies such as buffered packets and wasted transmission time are also created under TDMA when all time slots are of equal length and links are asymmetric.

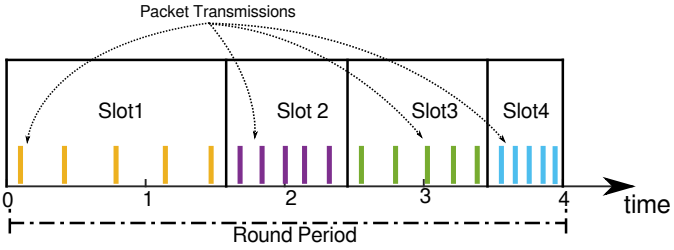

Fig. 5. Using DVSP, each slot has different length, to guarantee that every node has enough time to transmit all received data from its upstream neighbor. All nodes are sending the same amount of data.

the video stream at the sink will be both chopped and lagged.

This is the problem we are tackling in this paper, i.e., how to manage the traffic in a line multi-hop network, adjusting to variations in instantaneous bandwidth of individual links so to minimize in-network queuing and reduce end-to-end delays.

IV. A Variable Slot-length TDMA Solution

To maximize end-to-end throughput at minimal delay, we propose a TDMA-based solution that operates on top of the standard WiFi protocol where each node i is scheduled to transmit in a predefined time window called *Time Slot*, with duration s_i, that arrives periodically with period T, called *Round Period*.

TDMA schemes typically provide an exclusive (collision-free) slot to every transmitter in the network, granting a fixed length to the slots of all nodes ($s_1 = s_2 = \cdots = s_N = T/N$) as Figure 4 illustrates. Due to bandwidth irregularities across links, we propose a dynamic slot length assignment where each node has an exclusive time slot as Figure 5 exemplifies[1].

Time slot length is dynamically set according to the current bandwidth status of the network to mitigate buffer queuing. We define bandwidth $B_{i,j}$ as the average capacity in bytes per second available to transmitter node i to send data to receiver node j (cf. Figure 2). Knowing both bandwidth estimates of all links in the line network and (fixed) round period T, we can compute the optimal slot length (s_i) of every node that

[1]Channel reuse could eventually improve bandwidth, but not decrease delay which is our major concern.

guarantees no buffered data, and therefore minimum delay. Under our TDMA assumptions, in average a node i receives $\frac{s_{i-1}}{T}B_{i-1,i}$ bytes per second from its up stream node $(i-1)$, and sends out $\frac{s_i}{T}.B_{i,i+1}$ bytes of data per second to its down stream node $(i+1)$. To enforce long term stability of the network with limited buffering, we need to ensure these rates coincide and round period stays constant, thus Equation 1.

$$\begin{cases} s_1 B_{1,2} = s_2 B_{2,3} = \cdots = s_n B_{n,n+1} \\ s_1 + s_2 + \cdots + s_n = T \end{cases} \quad (1)$$

Solving the system in Equation 1 for s_i yields the slot-length solution Equation 2.

$$s_i = \frac{(B_{i,i+1})^{-1} T}{\sum_{j=1}^{n} (B_{j,j+1})^{-1}} \quad (2)$$

In order to implement this system, a central node (GS for instance) would need to collect every link bandwidth estimation and then disseminate the corresponding slot length to every node. Assuming that slot order is fixed and chosen to minimize delay from source to sink[2], collection of link bandwidth estimates would take one round period, and dissemination would take n rounds to complete, where n is the number of nodes excluding the GS (same as the number of slots) [15]. There are two main problems with this approach: (1) if the GS misses some of the bandwidth estimations or fails to distribute the new slot length to every node, we can get inconsistencies leading to different round periods and potential slots overlapping; and (2) buffers can fill up before the dissemination of slots is completed.

Alternatively, we chose to design a distributed approach where each node sets the best slot length for itself and the node up stream. Since by design, stream data is coming exclusively from neighbor node(s), one node can do flow control and instruct its neighbor to decrease its transmission slot, thus reducing buffering needs. Based on this idea, we created a new protocol for live streaming in multi-hop lines that we call Distributed Variable Slot-length Protocol (DVSP).

A. Distributed Variable Slot-length Protocol (DVSP)

We propose a distributed variable time slot allocation protocol where the task of time slot adjustment is locally performed by every pair of neighbor nodes. In this approach, initial time slot length values are iteratively redefined until convergence to their optimal values as defined in Equation 2. Each node adapts its own slot s_i and up stream node slot s_{i-1} simultaneously such that the sum of the slot lengths is kept constant, thus keeping the round period unchanged. By changing up stream slot time, we can quickly solve buffer problems at the local source, and propagate this effect to the initial data source node. Using an equation system similar to Equation 1, and assuming that 1) node i has an accurate estimation of current bandwidth available in the previous and next links, respectively $B_{i-1,i}$ and $B_{i,i+1}$, and 2) bandwidths are constant for a round period, yields Equation 3, for iteration k.

[2]This means the slot order in the TDMA round matches the physical order in the link topology from source to sink, to favor propagation of source data.

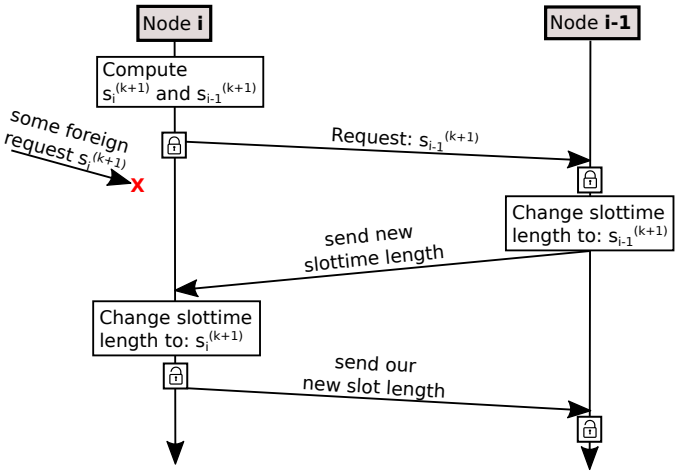

Fig. 6. Handshake diagram of Distributed Variable Slot-length Protocol (DVSP)

$$\begin{cases} s_i^{(k+1)} + s_{i-1}^{(k+1)} = s_i^{(k)} + s_{i-1}^{(k)} \\ s_{i-1}^{(k+1)}.B_{i-1,i} = s_i^{(k+1)}.B_{i,i+1} \end{cases} \quad (3)$$

This system of equations result in two recursive functions (Equation 4).

$$\begin{aligned} s_i^{(k+1)} &= \zeta_{i-1,i} \left(s_i^{(k)} + s_{i-1}^{(k)} \right) \\ s_{i-1}^{(k+1)} &= \overline{\zeta}_{i-1,i} \left(s_i^{(k)} + s_{i-1}^{(k)} \right) \end{aligned} \quad (4)$$

where

$$\begin{aligned} \zeta_{i-1,i} &= \frac{B_{i-1,i}}{B_{i-1,i} + B_{i,i+1}} \\ \overline{\zeta}_{i-1,i} &= \frac{B_{i,i+1}}{B_{i-1,i} + B_{i,i+1}} = 1 - \zeta_{i-1,i} \\ \overline{\zeta}_{i-1,i}, \zeta_{i-1,i} &\in]0,1[\quad , \quad i \in \{2, \cdots, n\} \end{aligned}$$

We design an handshake protocol to make these changes consistent and robust to failure. As Figure 6 depicts, node i initiates the process computing Equation 4. It then sends a request of a new slot length ($s_{i-1}^{(k+1)}$) to node $i-1$, and locks it self to incoming requests from any other node. This guarantees that until the handshake is completed no other handshakes are initiated that could corrupt the round period value (sum of all slot times). In the case this request packet is missed, it is repeated once per round until the handshake is finalized. Upon reception, node $i-1$ sets up its slot length to the new value present in the request ($s_{i-1}^{(k+1)}$). Then, at the beginning of node $i-1$'s slot time, this new length is already used. Every packet contains information about the current transmitter time slot length in its TDMA header to allow slot synchronization. Therefore, as node i receives packets from node $i-1$, the former will know that the new and expected time slot length is being used. Node i changes its own slot time length to new value $s_i^{(k+1)}$. This terminates the handshake.

By design, nodes under handshake lock to incoming requests from other nodes. In a line network, where slots are ascendantly ordered, this means that all nodes with even slot IDs (or odd), can perform handshakes with all upstream odd neighbors (or even), simultaneously. In a following moment, nodes with odd slot IDs (or even) can initiate requests.

Convergence

To prove the convergence of this method assume a line network with $n+1$ nodes, n of which are transmitters and so n time slots (the sink does not transmit). We name the correspondent time slots length s_1 (source), s_2 (first relay), etc.. We assume every node starts (iteration $k=0$) with the same time slot length $s_1^{(0)} = \ldots = s_n^{(0)}$. According to our protocol, in the subsequent iteration ($k=1$) all nodes with even ids ($i = 2, 4, \ldots$) initiate handshakes with upstream nodes ($i = 1, 3, \ldots$, respectively). For odd number of transmitters, node n remains unchanged. Thus:

$$\begin{aligned}
s_1^{(1)} &= \zeta_{1,2}\left(s_1^{(0)} + s_2^{(0)}\right)\\
s_2^{(1)} &= \overline{\zeta}_{1,2}\left(s_1^{(0)} + s_2^{(0)}\right)\\
s_i^{(1)} &= \ldots\\
s_{i+1}^{(1)} &= \ldots\\
s_n^{(1)} &= s_n^{(0)}
\end{aligned}$$

where $i \in \{3, 5, \ldots\}$.

At the next iteration ($k=2$), odd nodes ($i = 3, 5, \ldots$) initiate their handshakes with up stream even nodes ($i = 2, 4, \ldots$), and node 1 is unchanged. If n is even, node n is also unchanged. Thus:

$$\begin{aligned}
s_1^{(2)} &= s_1^{(1)}\\
s_2^{(2)} &= \zeta_{2,3}\left(s_2^{(1)} + s_3^{(1)}\right)\\
s_3^{(2)} &= \overline{\zeta}_{2,3}\left(s_2^{(1)} + s_3^{(1)}\right)\\
s_i^{(2)} &= \ldots\\
s_{i+1}^{(2)} &= \ldots\\
s_n^{(2)} &= s_n^{(1)}
\end{aligned}$$

where $i \in \{4, 6, \ldots\}$.

We can convert the system into a more compact form using matrix notation:

$$\mathbf{s}^{(k+1)} = A\mathbf{s}^{(k)} \quad \wedge \quad \mathbf{s}^{(k+2)} = B\mathbf{s}^{(k+1)}$$

where A and B are $(n \times n)$ matrices, as shown next, and $\mathbf{s}$ the $(n \times 1)$ time slot vector; combining two iterations at a time, yields:

$$\mathbf{s}^{(2k)} = (BA)^k \mathbf{s}^{(0)} = C^k \mathbf{s}^{(0)}$$

$$A_n = \begin{bmatrix}
\zeta_{1,2} & \zeta_{1,2} & 0 & \cdots & \cdots & 0 & 0\\
\overline{\zeta}_{1,2} & \overline{\zeta}_{1,2} & 0 & \cdots & \cdots & 0 & 0\\
0 & 0 & \zeta_{3,4} & \zeta_{3,4} & \cdots & 0 & 0\\
\vdots & \vdots & \overline{\zeta}_{3,4} & \overline{\zeta}_{3,4} & \cdots & 0 & 0\\
 & & \cdots & & \zeta_{ij} & \zeta_{ij} & 0\\
 & & \cdots & & \overline{\zeta}_{ij} & \overline{\zeta}_{ij} & 0\\
(0 & 0 & \cdots & \cdots & \cdots & 0 & 1)
\end{bmatrix}$$

$$B_n = \begin{bmatrix}
1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0\\
0 & \zeta_{2,3} & \zeta_{2,3} & 0 & \cdots & \cdots & \cdots & 0\\
0 & \overline{\zeta}_{2,3} & \overline{\zeta}_{2,3} & 0 & \cdots & \cdots & \cdots & 0\\
0 & 0 & 0 & \zeta_{4,5} & \zeta_{4,5} & \cdots & \cdots & 0\\
\vdots & \vdots & 0 & \overline{\zeta}_{4,5} & \overline{\zeta}_{4,5} & \cdots & \cdots & 0\\
 & & \cdots & 0 & 0 & \zeta_{ij} & \zeta_{ij} & 0\\
 & & \cdots & 0 & 0 & \overline{\zeta}_{ij} & \overline{\zeta}_{ij} & 0\\
(0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1)
\end{bmatrix}$$

If n is even, last row and column of A_n do not exist. If n is odd, last row and column of B_n do not exist. To prove this system converges, we show that in the limit:

$$\lim_{k\to\infty}(\mathbf{s}^{(2k+2)} - \mathbf{s}^{(2k)}) = \mathbf{0}_n \Leftrightarrow \lim_{k\to\infty}\left(C^{k+1} - C^k\right) = \mathbf{0}_n$$

Performing an eigenvalue decomposition on C, results in $C = VDV^{-1}$, where D is a diagonal matrix with C's eigenvalues. We know that $C^k = (VDV^{-1})\cdots(VDV^{-1}) = VD^kV^{-1}$, therefore:

$$\begin{aligned}
\lim_{k\to\infty}\left(C^{k+1} - C^k\right) = \mathbf{0}_n \Leftrightarrow\\
\lim_{k\to\infty}\left(VD^{k+1}V^{-1} - VD^kV^{-1}\right) = \mathbf{0}_n \Leftrightarrow\\
\lim_{k\to\infty}\left(VD^k(D - I_n)V^{-1}\right) = \mathbf{0}_n
\end{aligned}$$

If in the limit $D^k(D-I_n)$ is zero, then the equation holds and the system converges. Exploring the structure of D, yields:

$$D^k(D - I_n) = \begin{bmatrix}
\lambda_1^k(\lambda_1 - 1) & 0 & 0\\
0 & \ddots & 0\\
0 & 0 & \lambda_n^k(\lambda_n - 1)
\end{bmatrix}$$

From the expression above, we see if the modulo of the eigenvalues is less than or equal to 1, then $\lim_{k\to\infty}\lambda_i^k(\lambda_i - 1) = 0$, and the system converges. For the general case, we know that the maximum modulo of the eigenvalues of C, also known as spectral radius $\rho(C)$, is indeed not greater than 1. This comes from Gelfand's formula corollary that states that the spectral radius of the product of two matrices is less or equal to the product of spectral radius of both matrices: $\rho(C) = \rho(BA) \leq \rho(B)\rho(A)$. We can prove that matrices A and B have spectral radius 1. Note that A has n rows, and by design $\lfloor n/2 \rfloor$ pairs of rows are linear dependent, so there are $\lfloor n/2 \rfloor$ zero-valued eigenvalues.

$$\lambda_1 = \ldots = \lambda_{\lfloor n/2 \rfloor} = 0$$

All other $\lceil n/2 \rceil$ eigenvalues are in fact 1. Knowing that the eigenvalues of A^T are the same of A for any matrix, we can

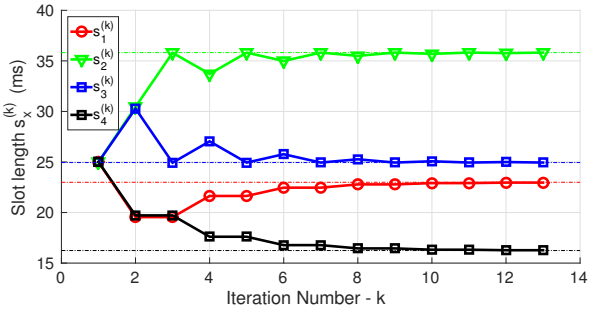

Fig. 7. Example of slot length convergence in a network with four slots. For any bandwidth values, and for any number of nodes it is proven that the distributed system converges to the global solution.

trivially find the remaining $\lceil n/2 \rceil$ eigenvectors v_i that make $A^T v_i = 1 v_i$ equation true, because all rows add up to 1. An example is shown below.

$$A^T v_1 = \begin{bmatrix} \zeta_{1,2} & \overline{\zeta}_{1,2} & 0 & \cdots & \cdots & 0 & 0 \\ \zeta_{1,2} & \overline{\zeta}_{1,2} & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & \zeta_{3,4} & \zeta_{3,4} & \cdots & 0 & 0 \\ \vdots & \vdots & \zeta_{3,4} & \overline{\zeta}_{3,4} & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & & \zeta_{ij} & \overline{\zeta}_{ij} & 0 \\ \vdots & \vdots & \cdots & & \zeta_{ij} & \overline{\zeta}_{ij} & 0 \\ (0 & 0 & \cdots & \cdots & \cdots & 0 & 1) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ (0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ (0) \end{bmatrix}$$

It is clear from the example above that all vectors v_i with exactly $n-2$ zeros and two consecutive ones in a odd and even index are eigenvectors and have the corresponding eigenvalue 1, such that:

$$v_i = \begin{bmatrix} 0 & \cdots & 1_{2i-1} & 1_{2i} & \cdots & 0 \end{bmatrix}^T$$

When n is odd, there is an extra eigenvector $v_{\lceil n/2 \rceil}$ that also has an associated eigenvalue 1, namely:

$$v_{\lceil n/2 \rceil} = \begin{bmatrix} 0 & \cdots & \cdots & 1 \end{bmatrix}^T$$

The rationale used for matrix A can also be used for matrix B, and we have now proved that both matrices have a spectral radius equal to 1 ($\rho_A = \rho_B = 1$), and therefore the whole distributed system converges.

Figure 7 shows a mock example of the distributed algorithm running on four nodes - 1 is the source, 2, 3 and 4 are relays ordered from source to sink. Bandwidths were randomly selected and the round period set to $T = 100$ms. We can see the slot length of each node (bold lines) changing over thirteen iterations, and converging within 5% of the final value in 6 iterations. That limit is the global solution given by Equation 2 (dashed lines). Note how nodes at the tip of the network (1 and 4), only change every other iteration. All other nodes change at every iteration.

Bandwidth changes as UAVs move, due to antennas orientation and path properties, but inspection uses slow velocity and frequent hovering, and bandwidth statistical variations take several seconds. Conversely, our protocol can converge in less than 1s assuming an iteration every round period and a realistic configuration of T=100ms and 3 relays.

B. Worst Case Delay

In a line network with n hops, packet end-to-end delay d_{wc} is the sum of the time taken by the packet while being transmitted over the air at all links (d_{tx}), plus the time a packet spends within buffers, waiting to be sent (d_w), yielding:

$$d_{wc} = d_{tx} + d_w \tag{5}$$

Time d_{tx} can be considered an affine function of hop count n, since payload size P is fixed and PHY bit-rate R, too. So $d_{tx} = nP/R$.

In Figure 5, one can see that every node transmits the same amount of information during its own slot. In this situation, buffer usage does not increase, and in the worst case we can consider that buffers are consistently full. We consider all nodes have the same buffer size U. This way, a new packet entering the system at the source waits till all buffered packets are sent, at each node. At each round T, all links are able to send the same amount of data $k^{(1)} = B_j s_j, \forall j \in [1, n]$, and since there are no concurrent transmissions there is no back-off time to consider. Therefore, the worst case delay is:

$$d_w^{(1)} = n \left\lceil \frac{U}{k^{(1)}} \right\rceil T \tag{6}$$

where, using the global solution Equation 2,

$$k^{(1)} = \frac{T}{\sum_i^n B_i^{-1}}$$

The best case is when all buffers are empty ($U = 1$) and as expected, it takes one round to deliver the packet.

Using the same time slot at every node as depicted in Figure 4 (rigid TDMA case), each node is able to send a different amount of data during its own time slot, and therefore the worst case is when the downstream node has a lower bandwidth than its upstream neighbor, and as such, buffers keep filling or keep full at all times. For such, when a packet reaches its next hop, its buffer is full and has to wait for U packets to be delivered before it proceeds to the next hop. Furthermore, all slots have the same length, but different bandwidths. So, each node can send at most $B_i.s_i$ amount of data per round; they actually send $k^{(2)}$, the minimum of all hops since all sent data comes from the upstream nodes, st.:

$$k^{(2)} = \min_i (B_i.s_i) \quad , \forall i \in [1, n] \tag{7}$$

we have $d_w^{(2)}$ st:

$$d_w^{(2)} = n \left\lceil \frac{U}{k^{(2)}} \right\rceil T \tag{8}$$

Under rigid TDMA, delay $d_w^{(2)}$ is guaranteed to be never better than DVSP delay $d_w^{(1)}$, since $k^{(2)}$ is never greater than

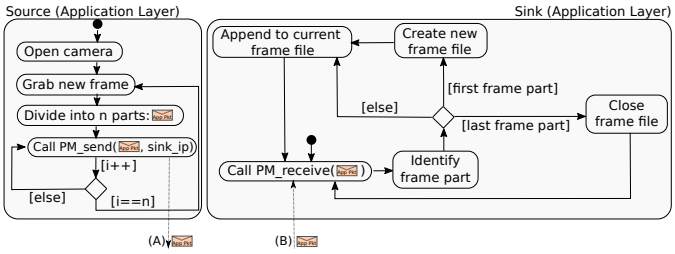

Fig. 8. Behavior diagrams for the Application Layer running at the Source and Sink. In the Source, the AL is responsible for grabbing camera frames, fragmenting them and sending fragments to the PM layer. In the Sink, the AL does the frame re-assembly. The AL communicates with the PM layer using `PM_send()` and `PM_receive()` functions.

$k^{(1)}$. We prove this by showing that since under rigid TDMA all slots have the same size T/n, yields:

$$k^{(1)} = \frac{T}{\sum_i^n B_i^{-1}} \geq \min_i(B_i).\frac{T}{n} = k^{(2)} \quad (9)$$

$$\Leftrightarrow \sum_i^n B_i^{-1} \leq n\frac{1}{\min_i(B_i)} \quad (10)$$

Furthermore, buffer usage U will be lower in DVSP by design than with rigid TDMA slots.

When using the native WiFi CSMA/CA, the worst case delay for a packet to be delivered is not exactly determined since it depends on the link load and the time taken by the back-off mechanism. Nevertheless, under CSMA there is no guarantees of balanced throughput, leading to strong queuing delays as in rigid TDMA case.

V. Architecture

Our architecture is a three-tiered design with an Application layer (AL) on top, a Packet Manager (PM) and our TDMA layer at the bottom.

A. Application Layer (AL) - Video capture and collection

The AL is only accessed at the source and sink devices since these are the only nodes that are responsible for generating or consuming data. The AL in the source runs the camera application, dedicated to grabbing video frames directly from the camera device. The frames are fragmented to fit WiFi packets that are then sent to the PM layer. Each fragment takes attached a corresponding header for proper identification and later re-assembly of frames at the sink. The left side of Figure 8 shows the activity diagram of the AL on the source node. The AL on the sink side just collects frame packets, re-assembles the frames into image files and hands them to the operator application, for display and/or further processing. The respective behavior is outlined in the right side of Figure 8.

B. Packet Manager (PM) Layer - Routing

The PM layer (Figure 9) is in charge of finding the next hop of an incoming packet, whether it comes from the upper AL layer or from other node through the lower TDMA layer.

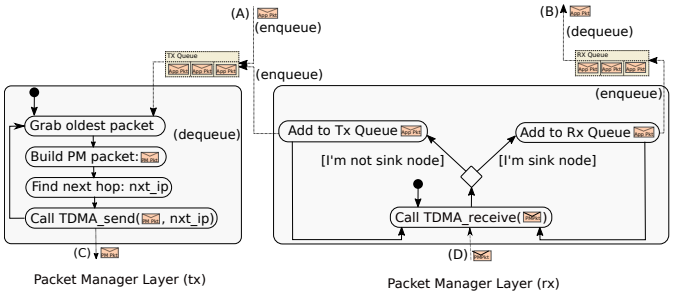

Fig. 9. Packet Manager (PM) layer, routes packets to the next hop, or holds them for the AL if packets have reached their final destination (Sink). Communicates both with AL (above) and TDMA layers (below).

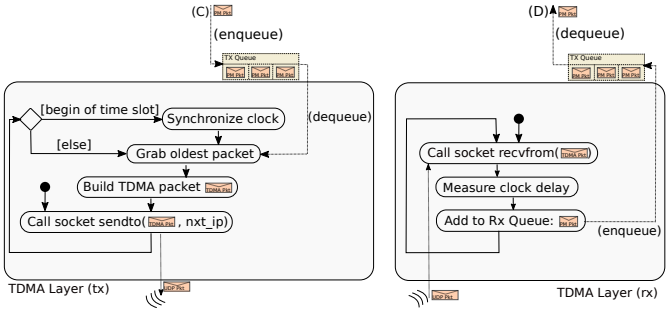

Fig. 10. TDMA layer is responsible for measuring network delays affecting the incoming packets. The receiver node adjusts the phase of its own TDMA slot according to any such delays, to keep slots sequential and reduce overlap. Queued packets are only transmitted over the air during the TDMA time slot.

Then, the PM either forwards packets to the TDMA layer to proceed their way along the network or, for packets meeting their final destination (the Sink in this case), the PM saves them to an internal queue, for later delivery to the AL.

C. TDMA Layer - Transmission Shaper

The TDMA layer does the shaping of the outgoing communications to the respective node TDMA slot and its behavior is explained in detail in Figure 10. Basically, whenever the TDMA slot of a node comes, and during its duration, that node enqueues pending packets in the wireless card for transmission over the air. Note, however, that a new packet is only sent to the wireless card when it informs the TDMA module that the previous packet has been sent.

The TDMA layer is also in charge of synchronizing the node's slot with the global TDMA framework. We use a sliding phase adjustment based on the delay affecting the incoming packets following the same principle as proposed in [2]. At the time of transmission, senders tag packets with their offset relative to the start of the respective slot. Thus, receivers get these offsets, average per sender, and estimate the start of the sender slot. If incoming packets are delayed, nodes will estimate a late sender slot start time, thus delaying their own slots to reduce overlap, which corresponds to shifting the phase of the TDMA round (Figure 11). Note that slots are not strictly isolated as typical in TDMA implementations. Residual overlaps can occur and are sorted out with the native WiFi

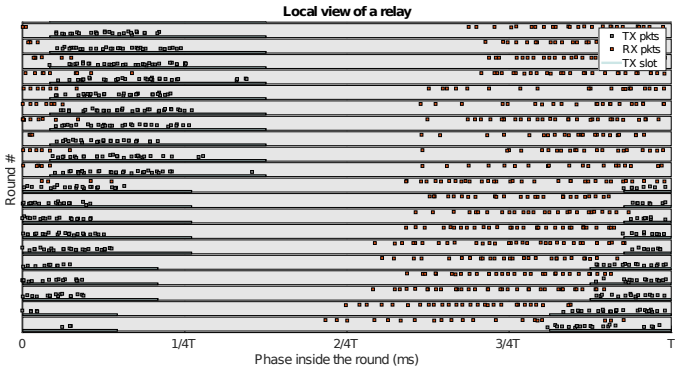

Fig. 11. Example of packet reception and transmission (relaying) occurring in a relay. The transmission window (sender time slot) is delayed to initiate after the previous transmissions - TDMA self-synchronization.

CSMA/CA arbitration. This feature allows our protocol to operate in open networks. Interference caused by other traffic and hidden nodes will appear as delays affecting packets, thus delaying the following slots. When triggered frequently, the slot shifting increases the actual TDMA round period, causing a reduction of the effective channel bandwidth available. The protocol transparently adapts to these cases.

Figure 11 clarifies this process showing packet reception and transmission at a node in the network. Each line contains events occurred within one round period. Each red square represents a received packet and its x-axis position the time in the round when it was received. The black squares represent the packets transmitted (relayed) by the node. In light blue we see the time slot of this node, i.e., the period of time in which the node can transmit data. The y-axis has an ever-increasing round counter. Note how the time slot is occasionally being shifted to the right, to accommodate delays in the incoming packets due to delays in the nodes and in the medium.

Finally, the TDMA layer is also assessing the bandwidth of both upstream and downstream links and running the DVSP. Its requests and replies are periodically sent as TDMA packets and filtered at this layer, being transparent for the layers above.

D. Encapsulation

Our protocol stack, from the AL to the TDMA layer, is shown in Figure 12, which highlights the logical packets (protocol data units - PDUs) of each layer with their specific header information. The header of each layer is only relevant for that layer and accessed and modified at that layer, only, following good layering practices. Thus, each layer can be independently modified, or swapped among different implementation options, without any impact on the remainder of the stack, i.e., in a transparent way. Overall, the three layers impose an extra 23 byte overhead, which we consider negligible compared to the payload of 1KB that we are using.

Finally, the TDMA packets are encapsulated into UDP/IP packets. We opted for UDP instead of TCP to keep better control of the timings of packet transmissions over the network. We also decided not to do cross-layer optimizations, despite some potential performance improvement that they could bring. The reason was also to strictly enforce layering, facilitating debugging and future extensions.

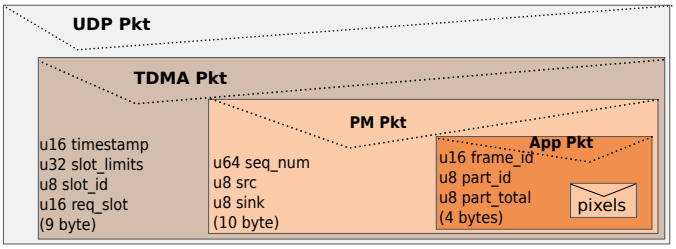

Fig. 12. Our protocol stack, fully respecting the layering principle, with an independent protocol at each layer using own protocol data units, and final encapsulation in a UDP/IP packet for transport.

VI. Evaluation

In this section, we evaluate delay and packet delivery ratio under immediate routing (traditional *CSMA/CA*) and under TDMA with adaptive time slots (*DVSP*). Despite being less important, end-to-end goodput or in other words video frame rate is also measured to guarantee that we compare both methods fairly.

A. Setup

The experiments were taken inside a laboratory with ArDrone 2.0 [16] UAVs, in fixed locations as Figure 13 shows[3]. One UAV is simultaneously source, using its frontal camera, and sink, to facilitate delay measurements. The video stream is sent through other UAVs in a circular route, returning back to the sink. The number of hops is varied from two to four ($n = 2 \ldots 4$) skipping some relays in the path (Figure 13). With DVSP, we chose a round period of $T = 100$ms and the number of slots is updated to match the number of hops. Queues in the relays are limited and, when overflowing, the system will drop oldest packets first.

All nodes were set to operate in the same IEEE 802.11g ad-hoc network at a bit rate of 24Mbps. This bit rate guarantees that we can produce data faster than the channel capacity can support, and therefore we are not limited by the processing power of our platform/camera.

Each video stream experiment lasted for roughly 180 consecutive seconds, limited by local log file storage, during which one of the two methods was used. Each frame has 57600 bytes divided in 50 packets that are sent to the Packet Manager layer. Each packet corresponds to one horizontal line of pixels in one image frame. The source is programmed to capture a new frame only if the Packet Manager queue has room for at least 50 packets.

To cause notable asymmetry in the links, the transmitter of the last hop (#4) was set to transmit at 10dBm where all other nodes were set at 15dBm. During a typical video stream scenario, links have different lengths or/and are affected by different attenuation factors due to obstacles or anisotropic antennas, for instance.

[3]Outdoor experiments with actual flights were reported in [1] with rigid TDMA slots for channel characterization.

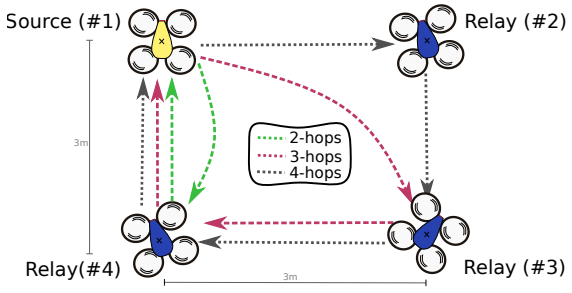

Fig. 13. Three possible different topologies used during the experiments. Depending on the desired number of hops, traffic is routed through different paths. Under DVSP the number of slots in use is also updated.

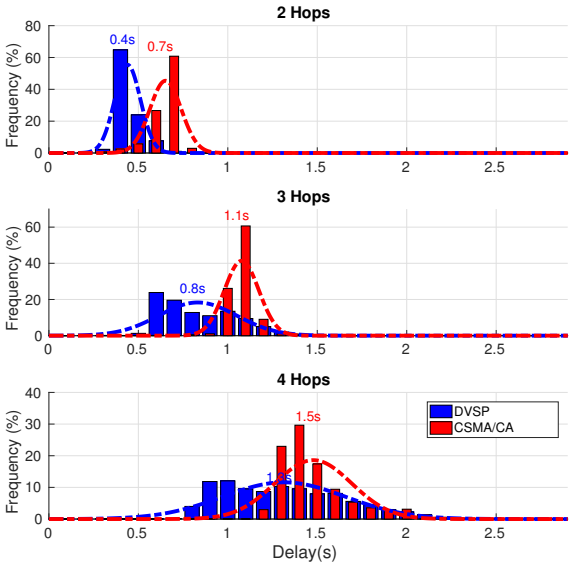

Fig. 14. Histogram of measured end-to-end delay of each packet with CSMA/CA (red thiner bars) and with DVSP (blue thicker bars), using different number of network hops. It is clear that the average delay increases with hop count, but it is always better when DVSP is in use.

B. Delay

Figure 14 shows the histograms of end-to-end delay measurements of successful packets using different number of hops. Red thiner bars show the case of CSMA/CA while blue thicker bars are used for DVSP. We consider end-to-end delay to be the elapsed time between a packet being successfully sent to the Packet Manager at the source and being received by the Application layer at the sink. Naturally, the average delay and its variance increase with hop count, but DVSP is consistently better than CSMA/CA. With two-hops, delay increases approximately up to 75% when CSMA/CA is in use. Overlaid dashed lines are Gaussian curves with the same average and variance as the histograms, to merely provide a visual notion of these metrics.

C. Packet Delivery

Due to buffer overflow, packets are expected to be dropped under heavy load. This translates to a low packet delivery ratio or PDR. Figure 15 shows that phenomena when CSMA/CA

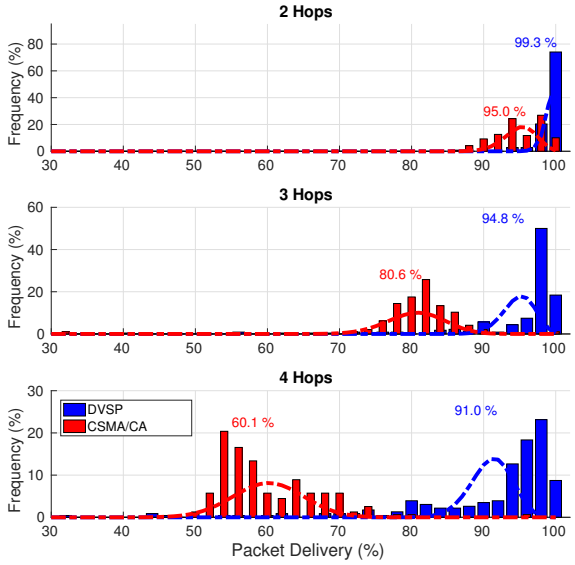

Fig. 15. Histogram of measured end-to-end packet delivery ratio. There is a noticeable improvement with DVSP, specially when the number of hops increases. Packet loss occurs essentially due to buffer overflow.

Fig. 16. Snapshot of the video stream at the sink. With CSMA/CA (on the right), relays cannot handle every received packet, dropping some that appear as black lines in the image. Under DVSP (on the left), video streaming is visibly improved and frames are generally complete.

is in use. The average PDR is far from 100%, specially when more and more relays are added. With CSMA/CA, relays receive packets at a higher rate than they can retransmit. This means that the source is actually sending more data than the network can handle. With DVSP, the source has a periodic slot to transmit data and DVSP shortens its duration whenever the packets are not going through, down in the link, effectively doing a kind of flow control and avoiding network overload. Thus, PDR is close to 100%. With four hops, we obtain gains of 50% in PDR. Figure 16 shows a snapshot of the video stream at the sink. With CSMA/CA (right), the low PDR manifests as black lines on the image (missing data). When using DVSP (left), video streaming is visibly improved and frames are generally complete.

D. Goodput

The last analyzed metric is goodput, which measures application payload data received at the sink per second. As we can see in Figure 17, more hops imply less goodput since there are more transmitters sharing the medium, thus end-to-end bandwidth is divided accordingly. Unlike delay and PDR, there is no difference on average goodput between

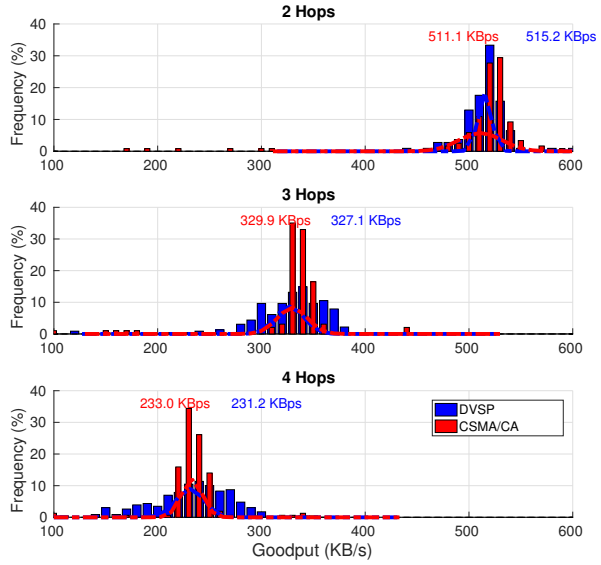

Fig. 17. Histogram of measured end-to-end goodput, i.e., actual application payload data received at the sink per second. More hops imply less goodput since there are more transmitters sharing the channel. There is no difference, in average, in this metric between both methods.

both transmission control methods. This is actually expected since, under heavy load, nodes always have some packets in their buffer (DVSP case) or their buffers are always full (CSMA case), and therefore the medium ends up being used at maximum capacity either way. Therefore, the end-to-end goodput is determined by the slowest link in the network in both methods. The fact that variance is higher with DVSP is explained by the constant adaptation of the time slots, which causes data to arrive at the sink in bursts, unlike the CSMA/CA scenario.

VII. Conclusion

In this paper, we addressed the problem of supporting live video streaming from an area of interest to a ground station through an ad-hoc relay network of UAVs. Data is generated by the first UAV, only, and relayed through the remaining ones, overcoming the distance limitation of the communication range of a single UAV. We then showed that due to asymmetry among the network radio links, simply relaying packets immediately over the typical CSMA/CA medium access control leads to large queuing delays or high packet drops. To remedy this problem, we propose a novel distributed adaptive TDMA overlay protocol called DVSP, for Distributed Variable Slot-length Protocol, that balances the amount of data each node transmits every round in order to minimize data buffering. We proved that this distributed approach converges to the optimal global solution using only local information. Experimental results show that without loss in goodput, our DVSP protocol outperforms immediate relaying in both network delay and packet delivery ratio. Despite not showing in the paper, we also carried out some experiments with typical TDMA implementation using fixed equal slots. The results were similar to those of immediate relaying. We are currently studying the connection of DVSP with the control of the relays placement for further improving the end-to-end link.

Acknowledgment

This work is a result of the project NanoSTIMA (NORTE-01-0145-FEDER-000016), supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF). The work was also partially supported by UID/EEA/50008/2013 and CMU|Portugal (SFRH/BD/51630/2011).

References

[1] L. Pinto, A. Moreira, L. Almeida, and A. Rowe, "Aerial Multi-hop Network Characterisation using COTS Multi-rotors," in *12th IEEE World Conference on Factory Communication Systems (WFCS)*, 2016.

[2] L. Oliveira, L. Almeida, and P. Lima, "Multi-hop routing within tdma slots for teams of cooperating robots," in *2015 IEEE World Conference on Factory Communication Systems (WFCS)*, May 2015, pp. 1–8.

[3] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular slam with multiple micro aerial vehicles," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 3962–3970.

[4] N. Goddemeier, K. Daniel, and C. Wietfeld, "Role-based connectivity management with realistic air-to-ground channels for cooperative uavs," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 951–963, June 2012.

[5] R. Muzaffar, V. Vukadinovic, and A. Cavallaro, "Rate-adaptive multicast video streaming from teams of micro aerial vehicles," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1194–1201.

[6] M. Asadpour, B. V. den Bergh, D. Giustiniano, K. A. Hummel, S. Pollin, and B. Plattner, "Micro aerial vehicle networks: an experimental analysis of challenges and opportunities," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 141–149, July 2014.

[7] E. Wandeler and L. Thiele, "Optimal tdma time slot and cycle length allocation for hard real-time systems," in *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '06. Piscataway, NJ, USA: IEEE Press, 2006, pp. 479–484.

[8] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proc. First Int. Conf. Embed. Networked Sens. Syst. (SenSys)*. Los Angeles, CA, USA: ACM Press, Nov 2003, p. 1.

[9] F. Jia, Q. Shi, G.-m. Zhou, and L.-f. Mo, "Packet Delivery Performance in Dense Wireless Sensor Networks," in *Proc. Int. Conf. Multimed. Technol. (ICMT)*. Ningbo, China: IEEE, Oct. 2010, pp. 1–4.

[10] M. Lindhe and K. Johansson, "Using robot mobility to exploit multipath fading," *IEEE Wirel. Commun.*, vol. 16, no. 1, pp. 30–37, Feb. 2009.

[11] D. Henkel and T. X. Brown, "Delay-tolerant communication using mobile robotic helper nodes," in *Proc. Int. Symp. Model. Optim. Mobile, Ad Hoc, Wirel. Networks Work. (WiOPT)*. Berlin, Germany: IEEE, Apr. 2008, pp. 657–666.

[12] D. Sicignano, D. Tardioli, S. Cabrero, and J. L. Villarroel, "Real-time wireless multi-hop protocol in underground voice communication," *Ad Hoc Networks*, vol. 11, no. 4, pp. 1484–1496, June 2013, special Issue on Wireless Communications and Networking in Challenged Environments.

[13] C. Rizzo, D. Tardioli, D. Sicignano, L. Riazuelo, J. L. Villarroel, and L. Montano, "Signal-based deployment planning for robot teams in tunnel-like fading environments," *Int. J. Rob. Res.*, vol. 32, no. 12, pp. 1381–1397, Oct. 2013.

[14] Q. Wang, K. Jaffrès-Runser, Y. Xu, J. L. Scharbarg, Z. An, and C. Fraboul, "Tdma versus csma/ca for wireless multihop communications: A stochastic worst-case delay analysis," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 877–887, April 2017.

[15] T. Facchinetti, L. Almeida, G. C. Buttazzo, and C. Marchini, "Real-time resource reservation protocol for wireless mobile ad hoc networks," in *25th IEEE International Real-Time Systems Symposium*, Dec 2004, pp. 382–391.

[16] "Parrot AR. Drone 2," https://www.parrot.com/us/drones, accessed: 21-04-2016.