Real-Time Optimization of Video Transmission in a Network of AAVs

Ahmed Abdel-Hadi, Jonas Michel, Andreas Gerstlauer and Sriram Vishwanath
Electrical and Computer Engineering Department, The University of Texas at Austin
{aabdelhadi,jonasrmichel,gerstl,theory}@mail.utexas.edu

***Abstract*—Mobile cyberphysical systems have received considerable attention over the last decade, as communication, computing and control come together on a common platform. Understanding the complex interactions that govern the behavior of large complex cyberphysical systems is not an easy task. The goal of this paper is to address this challenge in the particular context of multimedia delivery over an autonomous aerial vehicle (AAV) network. Bandwidth requirements and stringent delay constraints of real-time video streaming, paired with limitations on computational complexity and power consumptions imposed by the underlying implementation platform, make cross-layer and cross-domain co-design approaches a necessity. In this paper, we propose a novel, low-complexity rate-distortion optimized (RDO) protocol specifically targeted at video streaming over mobile embedded networks. We test the performance of our RDO algorithm on simulation models developed for aerial mobility of multiple wirelessly communicating AAVs. Results show that our optimized streaming leads to 47% and 39% less video distortion with very little computational overhead compared to regular ACKed and non-ACKed transmission, respectively.**

***Index Terms*—Multimedia Networks, Rate-Distortion Optimization, Cyber-Physical Systems**

I. Introduction

An explosion of interest in cyberphysical systems has resulted in new research challenges emerging from the need to integrate communication, control and computing. There are a multitude of applications that need high bandwidth communication coupled together with reliable control and distributed, often high-performance and real-time computing and decision making. Such systems often have to operate in tightly constrained environments that severly limit the available computational performance or the amount of power that can be consumed. Given a vast array of possible cyberphysical system implementation options and parameters, analyzing and designing protocols and application algorithms for them is a considerably daunting task that invariably requires integrated, cross-layer and cross-domain co-design approaches.

In this paper, we target the analysis and co-design of a particular cyberphysical system consisting of a network of autonomous aerial vehicles (AAVs). Such systems are of considerable interest across multiple civilian and military applications, including search and rescue, perimeter monitoring and object tracking. A network of AAVs poses a vast array of challenges - mobility, tracking and collision avoidance are essential for the physical operation of each AAV, while coordination and teaming critical for the network to carry out the task at hand. Central to all of these challenges is the ability to exchange high bandwidth delay sensitive data between the nodes in the network as reliably and efficiently as possible.

This paper specifically focuses on developing efficient protocols for delivery of rich-content multimedia (such as video) across a wireless mobile network. Such media is delay sensitive while being both computationally and bandwidth intensive, and it poses considerable challenges in guaranteeing its reliable delivery. Multimedia delivery is also essential for ensuring that the other cyberphysical functions such as distributed control and decision making can be executed effectively. The main features that make real-time packetized media delivery particularly challenging are [4]:

1) *High data rate:* media (especially video) requires high data rate even when the physical, wireless link between nodes is rapidly changing over time.
2) *Real-time constraints:* there is a time-to-live (TTL) associated with each packet, so a packet received after its TTL expires is lost.
3) *Dependencies between frames:* H.264 and other compressed video formats are characterized by interframe dependency . This has two effects: first, if some frames are not successfully received this will lead to dropping of other successfully received frames because of their interdependency. Second, the number of interdependent frames determines the extent of compression in the video.

Our ultimate goal is to develop algorithms that exploit the structure of multimedia to deliver them efficiently and reliably over an AAV network, and then test them in a real-world setting using a testbed. We have developed our own low-complexity rate-distortion optimized (RDO) streaming algorithm, and show that it outperforms other mechanisms in the context of Horus, a custom built AAV testbed [1]. This paper represents the first of many steps in making Horus a fully operational testbed. We have developed software simulations for the mobility and channel models between AAVs, and we tested both existing and our proposed RDO video streaming techniques using these simulation models. Results show that optimized streaming can result in much more reliable and efficient video delivery than traditional protocols, in variants both with or without feedback.

A. Related Work

There is considerable existing literature on developing protocols for efficient delivery over networks. A majority of this literature tends to focus on sensor networks designed for static

settings, where nodes sense physical quantities that undergo a gradual change over time, e.g. temperature. For these applications, only a low data rate is required. Increasingly, sensor networks research incorporates dynamic network topologies as well. In [7], [8], [9], the authors conduct an experimental analysis on a dynamic sensor network where nodes move in a large area gathering data and then sending the collected data when near an access point. In our paper, the nodes move in a prespecified pattern and gather media, e.g. video signals, and communicate them through the wireless network to an access point in another network. Note that our network is dynamically changing rapidly and intended to support a much higher data rate than conventional sensor networks.

Simultaneously, there is a growing body of work on media compression and streaming, both over wired and wireless networks. One of these research efforts is presented in [5], where the authors address the problem of streaming packetized media over a lossy packet network in a rate-distortion optimized way. In [5], simulation results show that systems based on rate-distortion optimization (RDO) algorithms have steady-state gains of more than 2-6 dB compared to systems that are not rate-distortion optimized. In this work, a simplified simulation model approximating real-world conditions is assumed, which is only a first step in measuring the performance and expected improvement an algorithm has over existing implementations. For wide-spread system deployment and evaluation of achievable gains of any algorithm, experiments and validations must be carried out in a realistic setting. Towards this goal, we model and deploy RDO implementations in the context of an actual AAV testbed, where realistic simulations are a first step towards running physical experiments in the field. Furthermore, the original RDO algorithm presented in [5] is based on ideal assumptions, e.g. in terms of its implementability. We instead propose a modified RDO version that can be efficiently realized with little to no overhead as part of standard network stacks on restricted embedded platforms.

B. Our Contributions

The main contributions of this paper are summarized as:

- We introduce a low-complexity RDO transmission algorithm that requires a smaller memory and less computation power when compared to the rate distortion optimization as presented in [5].
- We present a simulation model for the Horus testbed, a network of AAVs where the nodes move in fixed circular pattern while communicating with one another.
- We show that low-complexity RDO transmission algorithm outperforms ACKed transmission by 47% and not ACKed transmission by 39% with respect to net distortion on received video as measured by simulations.

The rest of the paper is organized as follows. Section II discuss the rate-distortion optimization problem, our algorithm and the distortion measure used. Section III describes Horus setting and its mapping into network simulator. Section IV includes comparative simulation results and a discussion of their implications. The paper concludes with Section V.

II. Rate-Distortion Optimization Problem

The RDO problem aims to optimize the amount of distortion in a network against the rate. Distortion is defined as the degradation in media quality as packets are dropped. The rate represents the amount of data, i.e. the number of packets transmitted per unit time. The data packets that comprise a stream vary in their importance in contributing to output quality and, conversely, distortion. As such, RDO is concerned with deciding which packets to drop based on media quality metrics, measuring both the deviation from the source material and the bit cost for each possible decision outcome. In other words, the problem aims to solve the question of, *which* packets to select for transmission, *when* to transmit them, and *how* to transmit them (e.g., how many times), such that the expected distortion is minimized, subject to constraints on the expected rate.

In [5], the authors presented an algorithm that minimize the distortion D for a given rate R. This is done by minimizing the Lagrangian $D + \lambda R$ for some Lagrange multiplier λ. This algorithm is based on off-line transmission policy computation and on-line transmission policy truncation. This is not efficient for embedded applications in a real-time setting as the transmission policy computation is time consuming and needs to be performed off-line. We present a new algorithm with two important characteristics: (1) real-time compatibility, where the transmission policy is computed on-line, and (2) lower complexity in terms of computational processing requirements.

In video encoding, frames are compressed with different ratios and dependencies, giving each frame a different priority. Frames are divided into packets, where the amount of data and hence the number of packets increases with the frame priority. On the receiver side, these packets are recombined to form a frame that is then decoded. Therefore, losing a frame with high priority will lead to more deterioration in the quality of decoded video.

A. Rate-Distortion Optimization Algorithm

In line with existing video standards, we assume that the compressed video is composed of three types of frames: frames with first priority f_1 (also known as *i-frames*), frames with second priority f_2 (*p-frames*), and frames with third priority f_3 (*b-frames*). For compression purposes, the video frames are divided into Groups of Frames (GOF). Each GOF contains one f_1 frame and a fixed number of f_2 and f_3 frames. The low-complexity algorithm that we use to optimize the video transmission is implemented in application layer and is shown in Algorithm 1.

The algorithm consists of two main sections, *timestamp check* and *packet selection*. In the timestamp check, the algorithm starts by comparing the current time with the timestamp of current GOF $t_{GOF\_timestamp}$. If the GOF timestamp is not yet reached, packet selection is performed. Otherwise, the algorithm aborts the current GOF and advances to the next. In packet selection (i.e. within the same GOF), the transmitter first sends f_1 frame packets. Each f_1 packet is (re-)transmitted until it is successfully sent and the algorithm can switch to the

Algorithm 1 Low-Complexity RDO

```
loop
  if t < t_{GOF_timestamp} then
    if ID_{pkt} = 1 then
      transmit f_1 packets
      if f_1 transmit_ success then
        ID_{pkt}++ {switch to next packet}
      end if
    else
      transmit f_2 and f_3 packets
      ID_{pkt}++ {switch to next packet}
    end if
  else
    t_{GOF_timestamp} += Δt_{GOF}
    ID_{GOF}++ {start new GOF interval}
    ID_{pkt} ← 1
  end if
end loop
```

next one. After finishing the transmission of all f_1 packets, f_2 and f_3 packets are sent. These lower priority packets are transmitted without waiting for feedback. This avoid wasting time in packet retransmissions, acknowledgements and reception times for these less important packets. At the end of packet selection, the packet ID is incremented to send the next packet until all packets in the current GOF are transmitted or a timeout is reached.

The proposed RDO algorithm will work on top of standard protocol stacks. As an additional optimization we can, however, modify the 802.11 MAC to further improve overall real-time performance. Specifically, when the MAC layer receives a packet with a f_1 flag, it will use its default behavior to retransmit the packet up to 3 times until an ACK is received. If no ACK is received after 3 tries, the MAC will report a drop to the application layer. By contrast, in case of f_2 or f_3 packets, the RDO algorithm does not require feedback about transmission success and a modified MAC layer can transmit the packets only once without waiting for any ACK. This further reduces overall overhead and latencies.

B. Distortion Measure

We investigate the RDO algorithm using a multiplicative distortion measure. The multiplicative distortion D_m is initially set to the maximum distortion level D_0. When frames are successfully received, the distortion decreases by the number of frames of type i, N_{f_i}, multiplied by all the frames of higher priority successfully received within a GOF. This gives zero weight if frames of higher priority have not been successfully received. D_m is evaluated every GOF as

$$D_m = D_0 - N_{f_1}\Big(1 + N_{f_2}(1 + N_{f_3})\Big);\ 0 \le D_m \le D_0.$$

We also define the sum of multiplicative distortion D_M as the sum of D_m for all transmitted GOFs:

$$D_M = \sum_{GOF} D_m.$$

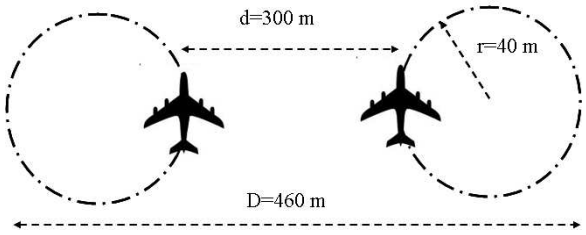

Fig. 1. Simulated network topology.

D_M is the distortion measure we use to compare different transmission protocols.

III. Experimental Setup

We demonstrate real-world performance of our optimized RDO algorithm in the context of Horus, a testbed composed of a network of AAVs communicating wirelessly in an ad-hoc fashion. For our experimental analysis, we assume that AAV nodes are equiped with video cameras and the capability to stream packetized media between them. We have setup a simulation of the Horus network in the OMNeT++ network simulator framework [3] using the MiXiM package [2]. To accurately mimic and evaluate RDO behavior in the Horus setup, we model both the time-varying nature of the network topology as well as the modified protocol stack including our RDO layer on top of the OMNET++ component library.

A. Network Topology

For our experiments, we assume a topology in which two or three UAVs (hosts) move in fixed circular patterns with a radius of 40 meters and a distance of 380 meters between the circles centers, as shown in Figure 1. Nodes send data packets in a one-hop fashion over a MAC 802.11 wireless connection, where the source node transmits packets directly to the destination node. Next to the transmission under test, we include a third node that simultaneously transmits other packets not related to the main video stream. This setting allows us to analyze RDO transmission in the presences of high interference and consequently when experiencing a large packet loss.

We simulated this setup using OMNET++'s circular motion module and a simple path loss channel model as the one most closely resembling AAV-to-AAV conditions with little to no fading and no shadowing effects. The continuous motion of the nodes in circular path leads to time-varying channel effects and a network packet drop rate that depends on the relative position of the UAV. We use different path loss exponents α to model and experiment with normal and worst case channel conditions. For worst-case analysis, we assume an exponent α of 3.7. Together with interference from a third node as described above, we observe an overall packet drop rate of 40%, which allows for comparison of various transmitters under realistic conditions.

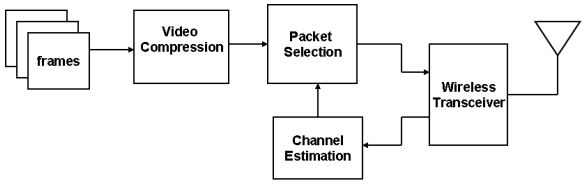

Fig. 2. Conceptual block diagram of the RDO implementation.

B. Rate-Distortion Optimization

Each AAV node is described using a standard OMNET++/MiXiM network stack consisting of a physical layer, a MAC layer and an application layer running the RDO optimized video streaming. We realize the RDO algorithm as part of the application layer of our network. A conceptual block diagram for general RDO implementation is shown in Figure 2. The video streams received from the camera are compressed into frames with different priorities. The *channel estimator* estimates the channel condition based on the received ACKs from previously sent packets. The *packet selector* is responsible for selecting the suitable packet for transmission based on (1) the information received from the channel estimator and (2) the packet timestamp associated with each packet. In our case, the decision for retransmitting a packet or sending a new packet is made by the packet selector according to the algorithm described in Section II.

For testing our RDO algorithm, we compare it against conventional transmission algorithms. Overall, we define three types of transmitters:

1) *Transmitter without ACK:* transmits every packet without waiting for an ACK from the receiver.
2) *Transmitter with ACK:* retransmits every packet until it receives an ACK for each packet.
3) *RDO Transmitter:* implements the RDO algorithm described previously.

IV. Experimental Results

To compare different transmitters, we run the network simulator for each transmitter under the exact network conditions mentioned in the previous subsection. We specify 3500 packets to be transmitted from the source to destination node. For simplicity, we fix the number of packets per frame for a given priority frame. The payload parameters are shown in Table I.

Our performance investigation for this problem includes both a network measure (e.g. number of packets drops) and an optimization measure (e.g. quality of the received media). We illustrate both using an easy-to-visualize plotting variable, which is a counter at the receiver. This counter counts the number of successfully received packets at the receiver side and increments the counter until it reaches the end of the frame and then resets the counter to zero. It then starts counting in the same manner for the next frame, and so on, see Figure 3. The counter can determine if the received packet is in the current frame or not by checking the packet ID number associated with it. This way, plotting the counter values over time visualizes the performance of the receiver with respect to the video frames. These values are later used to measure the distortion.

TABLE I
Payload Parameters

Total number of packets	3500
Number of priority f_1 frames per GOF	1
Number of priority f_2 frames per GOF	2
Number of priority f_3 frames per GOF	6
Number of packets per frame f_1	50
Number of packets per frame f_2	20
Number of packets per frame f_3	10

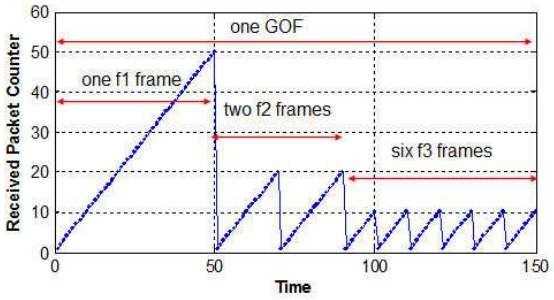

Fig. 3. Received packet counter for a complete GOF successfully received. f_1 frame counts to 50, f_2 frame counts to 20 and f_3 frame counts to 10. Any packet reception less than that count is not complete and the frame is considered dropped in the distortion measure.

The No-ACK Transmitter sends packets continuously without receiving any ACKs from the receiver. This leads to loss of packets with equal probability for different priority frames, which results in a 40% loss of the first priority packets essential to decode the other packets sent within a GOF. Due to this behavior (Figure 4), about 40% of frames are not completely received leading to high distortion, as shown in Figure 7.

The ACK Transmitter sends new packets only after receiving an ACK for the previous packet, and it otherwise continues to retransmit the same packet. Therefore, all the frame packets within the current GOF timestamp are received successfully regardless of their priority. The frames received after their GOF timestamp are dropped, but at the same time cause more delay to build up with time. This accumulative delay is caused by retransmission and ACKing of packets of lower priority. Due to this delay, the number of frames that are lost increases as the transmission continues, causing a large degradation in the video quality over time. This leads to a significant increase in the distortion at the end of simulation. As shown in Figure 5, the number of frames lost per GOF increase as the transmission continues, leading to high distortion D_M (Figure 7). This transmitter experiences the highest distortion when the simulation is allowed to run for a sufficiently long time.

The RDO transmitter is designed to minimize the distortion measure and give better performance than conventional transmitters. It retransmits until an ACK is received only for the first priority packets. Second and third priority packets are sent without waiting for an ACK from the destination. This guarantees that a first priority f_1 will be received at the

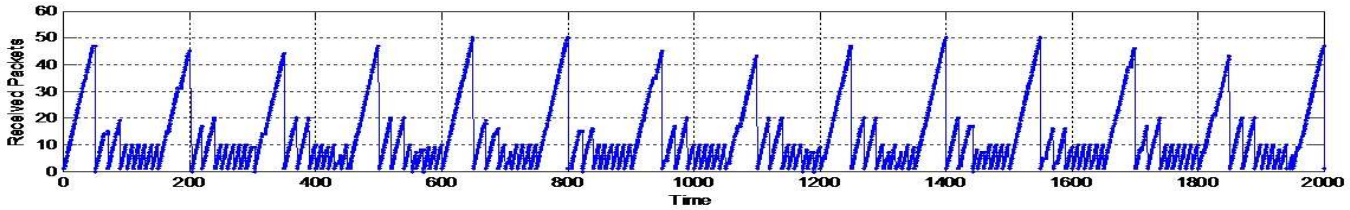

Fig. 4. Received packet counter for No-ACK transmission.

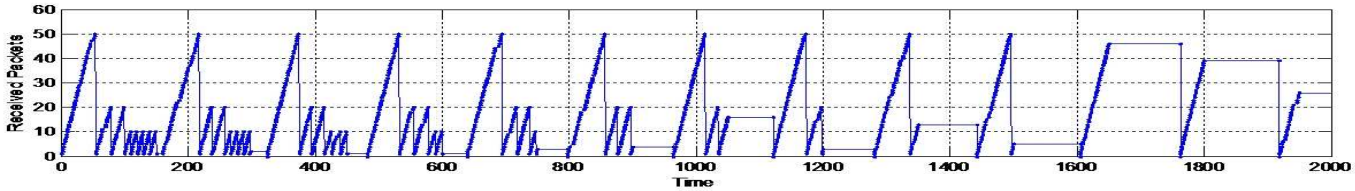

Fig. 5. Received packet counter for ACK transmission.

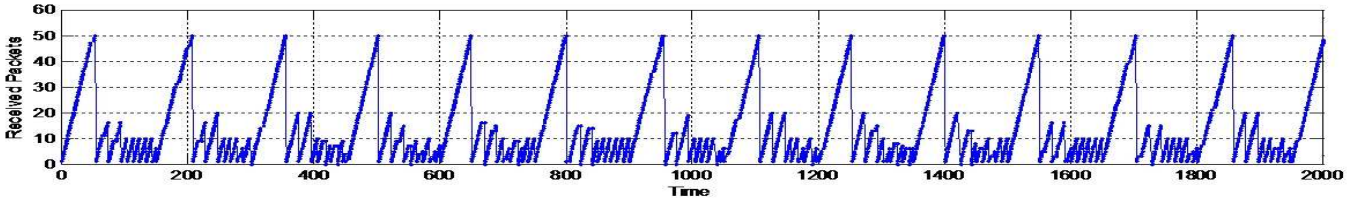

Fig. 6. Received packet counter for RDO transmission.

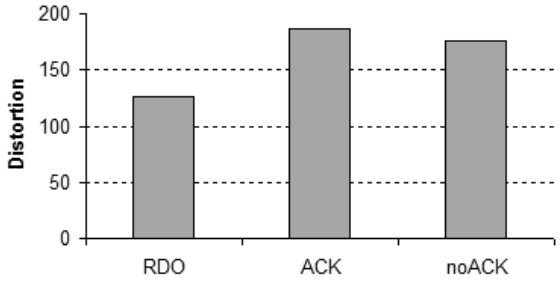

Fig. 7. Distortion measure D_M for different transmission protocols.

receiver even under bad channel conditions, which is the case in our simulation. The second and third priority frames, f_2 and f_3 respectively, are dropped with about 40% probability, as shown in Figure 7. Overall, the RDO transmitter guarantees a minimum video quality at the receiver side and gives a better distortion than other transmitters.

In all simulated cases, the average packet drop rate is 40%. In the RDO case, it is guaranteed that most of these drops are lower priority packets, which affects transmission quality less and makes RDO more robust. In the No-ACK and ACK transmitters, these dropped packets can be any type of packet priority. Therefore, in these two cases, the 40% drop rate significantly affects transmission quality.

V. Conclusions

Rate distortion optimization has its origins in information theory [6]. Rate distortion represents the minimum rate required to compress information at a distortion level of D. The rate distortion function with and without state is well known [6], and can be computed for most cases algorithmically and in some cases in closed form. Our rate distortion algorithm thus aims at bridging theory and practice, by building an algorithmic framework for real-time rate distortion optimization that is low complexity and thus practically viable over an AAV testbed.

We successfully simulated our RDO implementation and the AAV testbed in the OMNeT++ simulation framework. Our setup models the real-world network topology and its RDO transmitter. Results show that optimized RDO transmission outperforms standard protocols by a significant margin under poor wireless channel conditions.

References

[1] Horus: A Wireless Network of AAVs, http://theseus.ece.utexas.edu/horus/.
[2] MiXiM OMNeT++ Framework Project, http://mixim.sourceforge.net/.
[3] OMNeT++ Network Simulation Framework, http://www.omnetpp.org/.
[4] D. Agrawal, T. Bheemarjuna Reddy, C. Siva, and Ram Murthy. Robust Demand-Driven Video Multicast over Ad hoc Wireless Networks.
[5] P. A. Chou and Z. Miao. Rate-Distortion Optimized Streaming of Packetized Media. *IEEE Transaction on Multimedia*, May 2005.
[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.
[7] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li shiuan Peh, and Daniel Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet, 2002.
[8] Ting Liu, Christopher M. Sadler, Pei Zhang, and Margaret Martonosi. Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet. In *In MobiSYS 04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 256–269. ACM Press, 2004.
[9] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware Design Experiences in ZebraNet, 2004.