# TOWARD ZERO DELAY VIDEO STREAMING

By

Hothaifa Tariq Al-Qassab

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering -Doctor of Philosophy

2018

**ABSTRACT**

TOWARD ZERO DELAY VIDEO STREAMING

By

Hothaifa Tariq Al-Qassab

Video streaming has been growing rapidly since the beginning of this century and it is expected to continue growing. Meanwhile, transmission delay is a well-known problem in video streaming and it has been addressed by many prior works that demonstrated the feasibility of reducing packet delays over the Internet by employing a variety of end-to-end techniques.

This thesis consists of two parts that introduce new video streaming frameworks over the Internet and over connected-vehicle networks, respectively. Our objective in the first part of this thesis is to improve video streaming over the Internet. The emerging of new technology such as the HTTP-based Adaptive Streaming (HAS) approach has emerged as the dominant framework for video streaming mainly due to its simplicity, firewall friendliness, and ease of deployment. However, recent studies have shown that HAS solutions suffer from major shortcomings, including unfairness, significant bitrate oscillation under different conditions and significant delay. On the other hand, Quality-of-Service (QoS) based mechanisms, most notably multi-priority queue mechanisms such as DiffServ, can provide optimal video experience but at a major cost in complexity within the network. Our objective in this thesis is to design an efficient, low complexity and low delay video streaming framework.

We call our proposed Internet streaming framework Erasable Packets within Internet Queues (EPIQ). Our proposed solution is based on a novel packetization of the video content in a way that exploits the inherent multi-priority nature of video. An important notion of our proposed framework is Partially Erasable Packet (PEP). Furthermore, to evaluate our framework performance, we developed an analytical model for EPIQ that shows significant improvements when compared to the conventional and multi-priority queue video transmission models. Under congestion, a best-effort AQM router can simply erase an arbitrary portion of a PEP packet starting from its tail where we denote this process as Partial Erasing (PE). To complement partial erasing in the AQM, a rate control protocol similar to TFRC is proposed to ensure fairness for video and non-video traffic. Our results show that EPIQ provides improvements in video quality in terms of PSNR by at least 3dB over traditional video streaming formworks. In addition, packet loss ratio and delay jitter performance are comparable to the optimal video streaming mechanism that is offered by multi-priority systems such as DiffServ.

The main objective of the second part of the thesis is to develop a vehicle active safety framework that utilizes video streaming and vehicle-to-vehicle (V2V) communication for driver warning. Most prior efforts for V2V safety applications have been limited to sharing vehicle status data between connected vehicles.

We propose a Cooperative Advanced Driver Assistance System (C-ADAS) where vehicles share visual information and fuse it with local visuals to improve the performance of driver assistance systems. In our proposed system, vehicles share detected objects (e.g., pedestrians, vehicles, cyclists, etc.) and important camera data using the DSRC technology. The vehicle receiving the data from an adjacent vehicle can then fuse the received visual data with its own camera views to create a much richer visual scene. The sharing of data is motivated by the fact that some critical visual views captured by one vehicle are not visible or captured by many other vehicles in the same environment. The experimental results showed that our proposed system performed as intended and was able to warn drivers ahead of time, and consequently, it could mitigate major accidents and safe lives.

*To my family…*

## ACKNOWLEDGMENTS

First of all, I wish to express my great gratitude to my advisor, Dr. Hayder Radha, for his guidance and support throughout all stages of this research. His advice and encouragement have provided me with the skills needed to develop and refine this research.

Special Thanks for Dr. Subir Biswas, Dr. Jian Ren and Dr. Guoliang Xing for agreeing to serve on my committee. I would like to acknowledge and thank Michigan State University (MSU) for allowing me to conduct my study and research on campus. I also wish to thank all my friends and colleagues in the Electrical and Computer Engineering departments at MSU. Many thank to all my friends who have assisted and supported me in so many ways during my study.

Lastly, I would like to express my great gratitude to my mother, father, brothers and sisters for all their support and encouragement for me.

## TABLE OF CONTENTS

**LIST OF TABLES**

## LIST OF FIGURES

# Chapter 1 Introduction

## 1.1 Video Streaming

Video streaming has been growing rapidly since the beginning of this century and it is expected to continue growing. In 2015, the video traffic occupied 70% of total internet traffic and it is expected to reach 82% by the end of 2020. With rapid growth of internet traffic led by video traffic, the Internet busy hours will double before the end of this decade [1].

One of the most popular video streaming solutions today is based on the HTTP Adaptive Streaming (HAS) [2]. HAS client uses HTTP signals to request video chunk with certain bit-rate from the server. The user estimates the TCP throughput, which is used to carry video stream data, to select the corresponding video chunk from the server. This method is called Pull-Based video streaming. It is more suitable for Over-The-Top (OTT) streaming services that do not need high QoS.

HTTP-based Adaptive Streaming (HAS) has emerged as the dominant framework for video streaming mainly due to its simplicity, firewall friendliness, and ease of deployment [3]. HAS solutions rely on popular protocols, such as HTTP and TCP, and on standard network services supported currently by the best-effort Internet. An important example of HAS is the well-known and popular MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard that is supported by many major industry players [2]. Consequently, HAS has been receiving a great deal of attention, and many efforts have been proposed toward improving its performance and addressing many of its technical challenges. Some of the primary challenges that have been highlighted and studied by recent efforts include HAS's inefficiency in using network resources, and its inherent reliance on TCP, which leads to key issues such as delays and a highly undesirable oscillatory behavior that changes the streamed video bitrate and its quality profile rather frequently [4][5][6]. Recent studies have been proposed to improve the performance of HAS technology by improving the rate adaption algorithm and its approach for estimating available bandwidth (e.g. [7][8][9]). Meanwhile, the popularity of multimedia applications and the unprecedented growth of video traffic over the best-effort Internet have arguably aggravated DASH's challenges and led to a noticeable increase in the frequency and severity of network congestion events [7][9][11]. All of these challenges and issues have resulted in well-documented problems in degrading the end users' Quality-of-Experience (QoE). More importantly, and as highlighted recently by key marketing studies[12][13], continued degradation in consumer's QoE could lead to a negative impact on the overall HAS and Over-The-Top (OTT) video market in a very significant way.

Another type of a video streaming method is called Push-Based streaming. In this streaming method the server keeps streaming video contents until the client interrupts or stops the server. The server collects information from the client and determines the sending rate. This method usually relies on multimedia transport protocol (like Real-Time Streaming Transport Protocol (RSTP)) rather than TCP. RSTP provides an efficient and low latency medium to multimedia contents. Since this method provides low latency, it is often used for real time streaming applications such as video conference and video broadcasting. One important advantage of Push-based method is Multicast support which is essential in broadcasting. However, the Push-based method has a high overhead on the servers and requires special multimedia servers which make it not commercially profitable as the HTTP/TCP Pull-based method [3]. Additionally, because it uses especial multimedia transport protocol, the Push-based method is not network firewall friendly. Nowadays, real-time transport protocol use is limited to video conference applications like Skype, Google Hangout and Apple Facetime.

In addition to smooth video streaming, Multimedia Transport protocol also provides a congestion control mechanism that is essential for fair network resources sharing. For a multimedia transport protocol to achieve the high quality of experience it has to offer the following: a) low network latency and jitter b) high data rate and low queueing delay c) fast response to any change in network bandwidth d) can handle fluctuation in the send video bit rate due to variable rate video coding e) the congestion control mechanism has to be TCP friendly, and f) react to both explicit and implicit congestion indications from the network [14].

## 1.2 Video Streaming Related Work

HAS is currently the most used technology for video streaming. There have been several works that study the performance of commercial HAS solutions. In [7] the authors summarize the drawbacks of the conventional HAS systems. These drawbacks include: (a) Unfairness among video and non-video flows; (b) Oscillation in the requested video bit rate; and (c) Inefficient video bit-rate selection; and a forth drawback, (d) large delays, especially for live streaming, was studied in [15].

To improve the performance of HAS, several approaches were proposed. The authors of [9] proposed a four-step model (estimate, smooth, quantize and schedule) to design a rate adaptation algorithm. Several studies proposed using control theoretic approaches to govern the rate adaptation process. For example, in [8] proportional integral derivative (PID) controller is used. A stochastic Markov Decision Process (MDP) approach was proposed in [16] to maximize video quality. In [17] the author suggests using an online algorithm to adapt the video bitrate to achieve a stable video quality experience. Scalable Video Coding (SVC) was proposed by [18] and [19] to minimize the required storage for video files and to increase the supported video rates. In [20], the authors proposed a scheduling algorithm to deliver different SVC priority packets over HTTP. The authors in [15] suggested using new a HTTP 2 to reduce the HTTP delay.

Regarding Push based method, One of the first real time protocol is the TCP Friendly Rate Control protocol (TFRC)[21]  which is a rate-based protocol and the main part of IETF Real Time protocol (RTP). Rate-based protocols offer a smooth video transmission by using TCP sending rate equation. The early Multimedia protocol is focused on maintaining TCP friendliness by using packet loss as the sole indication for congestion in the network, similar to the loss based TCP[22]. However, loss based algorithms tend to fill bottleneck queues and cause buffer bloating [32] . Delay based congestion control algorithms represent the best alternative to loss based congestion control algorithm since they react to increase buffer size [22]. However, it is well known that delay based congestion control algorithms don't share resources fairly with loss based congestion control algorithms [14]. New real time streaming protocols like Network Assisted Dynamic Adaptation (NADA) and Google Congestion Control (GCC) use delay in addition to packet loss in computing the sending rate to achieve smooth video streaming[14] [22]. IETF Real Time Communication for Web (RTCWeb) proposes to uses GCC algorithm for real time video communication. GCC uses a loss based rate calculation algorithm at the sender, but a delay based rate calculation at the receiver. The algorithm uses both rates to compute the appropriate sending rate [23]. However, the author in [22] found that GCC does not share bottleneck bandwidth fairly with other flows. Another recent algorithm is Network Assisted Dynamic Adaptation (NADA). The algorithm combines packet loss, one-way delay and packet marking into a composite signal that is sent periodically from the receiver to the sender. The sender uses the composite signal to compute a reference sending signal and a rate shaping buffer to achieve a smooth sending rate. Results show that NADA[14] achieves good results in multiple video stream scenarios while it achieves mixed results when competing with TCP flows.

## 1.3 Multimedia Priority Dropping

Quality of service (QoS) is one of the most interesting topics for in computer networking nowadays. A network with QoS provides better performance than best-effort to end flows. Applications such as multimedia requires high quality QoS characteristics (low delay and packet loss) for end to end data flow that the current best effort internet does not offer. Significant research efforts have been made to improve the current internet [24] .

It is well known that delivery of optimal video over variable bandwidth networks, such as the Internet, can be achieved by exploiting the multi-priority nature of video content. A primary example of a video-streaming framework that can achieve consistent high-quality video delivery is based on multi-priority queuing of the video packets. In particular, DiffServ [25] has emerged as the leading multi-priority video queueing solution.

Under DiffServ, the video stream is divided into sub streams that are marked with different dropping precedence. As packets travel in the DiffServ network, low priority packets are dropped during congestion. To reduce queue management overhead and overall network complexity, the number of DiffServ priority queues is limited to only a few. Another related framework is based on a single network node that utilizes Active Queue Management (AQM) in conjunction with Priority Dropping, or AQM-PD. The single node may use single or multiple queues. Using a single queue with a priority dropping mechanism causes significant overhead when compared with DiffServ. When multi queue is used it is limited to few to reduce complexity. Consequently, despite the broad support for DiffServ within network routers, it has not been utilized in a significant way, and certainly not at the level that HTTP Adaptive Streaming (HAS) based solutions have been employed and used.

## 1.4 Multimedia Priority Dropping Related Work

Priority-based video transmission is a vast area that emerged from the early days of video streaming. Here, we briefly outline a very short list of key solutions that are relevant to the proposed EPIQ framework. In particular, complete system solutions that include end-to-end solutions with full network support and are not limited to network edge solutions like PET (Priority Encoding Transmission) and Unequal Error Protection (UEP) that are widely used in lossy wireless interface (i.e. network edge)

One of the first successful efforts to introduce QoS guarantees was DiffServ. Its scalable and distributed mechanism made it the most widely deployed QoS supported technology nowadays[25] . One early example for exploiting unequal priority of video packets using DiffServ was proposed in[26][27]. To take advantage of DiffServ, the packets of a video stream are divided and marked using a rather sophisticated mechanism to adhere to certain ISPs' policies. The authors of [28] proposed a DiffServ, UMTS and DVB cross-technology system solution that provides priority dropping to video streams across all systems. Because DiffServ packets marking polices are defined by the ISPs not by the end users, any network condition changes inside the DiffServ network would force video packets remarking to maintain adherence to ISPs' policies[29][30]. However, to overcome DiffServ packet marking polices, authors of[31] proposed a simple differentiated service by supporting multi service profiles (e.g. low delay) for video streams.

Other mechanisms were proposed to improve priority-based video transmission using AQM. In general, an AQM algorithm operates on a single network device queue to improve channel utilization and minimize network delay and congestion (See[27]  for a comprehensive survey about AQM.). In recent years, AQM has also been proposed to solve buffer bloating within Internet queues[32].  AQM with priority dropping (AQM-PD) provides the desired video stream priority packet dropping. In REDN3[33] , a multi queue AQM algorithm was proposed where each queue use Random Early Detection (RED) [34] algorithm with different packet dropping probability to provide priority dropping. Similarly, the authors in [35] proposed PID-PD, a single queue algorithm that employs proportional-integral-derivative (PID) controller to perform AQM and a sorting algorithm that sorts packets according to their dropping precedence. An analytical model of best effort video transmission is proposed in[36]. The model shows the importance of priority dropping over random dropping in layered video streams. In addition, the author proposed a multi queues mechanism that has different dropping precedence and congestion control algorithm to mark the video packets according to network condition. However, all of the proposed solutions that use multi queues to support video priority dropping have a limited number of queues. If the number of the video priority levels is more than the number of queues then the usefulness (or utility) of the packets delivered to the video decoder cannot be guaranteed.

## 1.5 Vehicle to Vehicle Safety Systems

Vehicle-accident related fatalities, especially those caused by human errors exceed one million every year worldwide [37]. In response to such statistics, a variety of safety measures have been proposed. In particular, in the United States, the US Department of Transportation (USDOT) in collaboration with state-level DOTs and experts nationwide have pursued the development of the Dedicated Short-Range Communications (DSRC) technology and related standards, which are designed for significantly improving safety measures through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications [38]. The USDOT pilot test program concluded that DSRC can reduce vehicle related accidents significantly. The USDOT also issued a recommendation that the DSRC technology should be mandated for all new light vehicles in the near future [39].

One important category of vehicle-related accidents involves pedestrian-vehicle collision. In the US in 2015, the number of pedestrian fatalities caused by vehicle accidents was 5,376, a 23% increase from 2009. Pedestrians' fatality is one of the few categories that experienced an increase in the past few years. Furthermore, most of the pedestrian accidents happen in urban areas [40].

One of the many accident scenarios that involve pedestrians is when a stopping vehicle occludes a crossing pedestrian from being viewed by other vehicles. A second passing vehicle's driver only notices the presence of a crossing pedestrian after the pedestrian is within a very close proximity to the second vehicle. In such scenario, the passing vehicle driver may fail to stop the vehicle in a timely manner, due to the close proximity to the pedestrian, and this leads to a potential injury or even fatality for the pedestrian.

## 1.6 Vehicle to Vehicle Safety Systems Related Work

DSRC is designed to support a variety of Vehicle to Infrastructure (V2I) and Vehicle to Vehicle (V2V) safety applications. In academic literature, extensive research has been done on vehicle communication applications. One of the important applications that rely on DSRC for active safety is C-ADAS. However, many of these safety applications, for example cooperative adaptive cruise control (CACC)[41][42] Cooperative Active Blind-Spot Assistance (CABSA) [43] and Cooperative Frontal Collision Avoidance  (CFCA)[44], use DSRC to share only vehicle status information to the surrounding vehicles ( i.e. no radar, lidar or visual information is shared) .See [45] for a comprehensive survey about C-ADAS . In this section we will focus on the work that is related to video streaming over DSRC.

Video streaming for safety applications is challenging because such applications require low end-to-end delay and good visual quality. In[46][46] the authors suggest using radar to determine distances between vehicles for efficient power control and to enhance the video quality. In [47]the authors proposed using adaptive MAC that adjusts the retransmission limit based on channel condition. In [48] and [49] the authors proposed using hybrid DSRC/LTE system. The authors suggested using QoS-aware and handover algorithm to switch between the two communication interfaces[50].  From a safety application prospective, in [51][52] the authors proposed using video transmission in overtaking maneuver assistance system (see-through system (STS)). The authors of[53], further improved the system by using adaptive channel coding and heuristic-based power control algorithm. To the best of our knowledge, STS is the only safety application that has been proposed to improve road safety. STS only provides visual information without any additional information. Meanwhile, our proposed system processes and shares both visual and computational information to further improve road safety.

A variety of new vehicle models typically include an Advanced Driver Assistance System (ADAS) that helps prevent pedestrian and other forms of accidents. The success of such system usually depends on the distance between the moving vehicle and pedestrian and on the vehicle speed [54]. On the other hand, standard DSRC's Basic Service Messaging (BSM) application only shares information about the vehicle conditions and status [55].

## 1.7 Summary of Contributions

In this thesis, we introduce novel solutions for video streaming and vehicle safety systems. Our contribution can be summarize to two topics as follows;

a) Video streaming: We proposed a novel video streaming framework that is capable of delivering optimal and consistent quality-of-experience similar to the quality promised by DiffServ while requiring minimal low-complexity network support similar to the support provided by standard and popular AQM algorithms over the best-effort Internet. We refer to our framework as Erasable Packets within Internet Queues (EPIQ). There are two important aspects of the proposed EPIQ framework. First, EPIQ is based on a novel packetization of the video content in a way that exploits the inherent multi-priority nature of video. In short, each EPIQ packet consists of multiple segments, and where each segment consists of video data of a different priority (see Fig. 2.1). The highest priority segment is placed next to the packet header whereas the lowest priority segment is placed at the tail of the packet. For example, an EPIQ packet can include Intra-coded frame (I-frame) bytes next to the header, followed by Prediction frame (P-frame) bytes, and then Bi-directional (B-frame) bytes at the tail of the packet. This novel packetization enables the second major aspect of the proposed EPIQ framework. Under congestion, a simple AQM mechanism can be realized where a network router could drop (or *erase*) only small portions located toward the tail of EPIQ packets instead of dropping complete packets as traditionally done by current routers. Hence, EPIQ enables a novel *partial erasing* paradigm within the network. Thus, we refer to an EPIQ packet as Partially Erasable Packet (PEP). Throughout this dissertation, we will use the two terms “EPIQ packets” and “PEP packets” interchangeably. It is important to note that EPIQ does not require multi-priority queues or highly complex marking algorithms. There is no need to maintain state information of any set of separate flows or multiple queues. EPIQ can be realized using a single queue, and the only added complexity is that the router needs to distinguish between PEP packets and traditional packets.

b) Vehicle Safety: We propose a novel Cooperative Advanced Driver Assistance System (C-ADAS) and we call it Cooperative Pedestrian Collision Avoidance System (CPCAS). The system could improve vehicle safety significantly, and consequently, it could save thousands of lives and prevent many injuries annually. Our system's aim is to improve road safety by enabling vehicles to share sensors information to improve their on-board active safety systems and driver assistance system. Our novel system design is flexible and comprehensive because it can share any type of sensor information with other vehicles. Here, we mainly focus on visual information sharing in reducing pedestrian accidents. In particular, the system shares vehicle's visual and data information about detected objects that are occluded from other surrounding vehicles. Our proposed system includes a novel selective sharing algorithm. Specifically, the system provides low latency information sharing that enables the driver and active safety systems to make a real-time emergency decision. The low latency is attributed to DSRC equipment and, more importantly, to our selective sharing algorithm. We present results that are based on actual road tests. We conducted our experiments using DSRC compatible devices and off the shelf sensors to validate our system performance and functionality.

## 1.8 Thesis Organization

The remainder of the Thesis is organized as follows.

- Chapter 2 presents the EPIQ framework and its analytical model.

- Chapter 3 describes the detailed aspects of EPIQ including the AQM (EPIQ-RED) and rate control mechanisms (EPIQ-TFRC).
- Chapter 4 presents our extensive ns-2 based simulation results.
- Chapter 5 presents our vehicle to vehicle safety system design and implementation
- Chapter 6 provides a brief summary of the dissertation

# Chapter 2 EPIQ System Design and Analysis

## 2.1 EPIQ System Overview

Traditional random packet dropping by the network could cause severe degradation in the quality of user experience of streamed video applications. This quality-of-experience degradation is usually manifested by frequent video-freeze events [36]. A root cause of this phenomenon is the well-known dependency among video packets and the inherent multi-priority nature of compressed video content. The proposed EPIQ framework mitigates these issues through a novel packet design that we call a Partially Erasable Packet (PEP) structure. A PEP structure is conceptually simple, and it is illustrated in Fig. 2.1. For any set of compressed video frames, we can identify different levels of priority for the content of such set. These levels could correspond to different picture types (i.e., I, P, and B frames) of a non-scalable video stream; or, they could correspond to different layers of a scalable video frame (i.e., a base-layer and multiple enhancement layers); or, a combination of both. Whatever the case, a video stream can be divided into different priority levels as shown in Fig. 2.1. Under EPIQ, we partition the video content of each priority-level and distribute that content into multiple PEP packets. More importantly, we place the highest priority content next to the header while placing the lowest priority content at the tail end of the packet. All other priority levels are placed progressively between the highest and lowest priority level content between the header and the tail of the packet as shown in Fig. 2.1.

For the sake of clarity in the presentation and analysis, we present a few definitions associated with the proposed EPIQ framework and PEP structure. ***EPIQ Video Set*** (EVS) represents a set of prioritized video content. In general, EVS could represent any set of video frames that can be partitioned into multiple priorities. For example, an EVS can correspond to a GoP for non-scalable video.

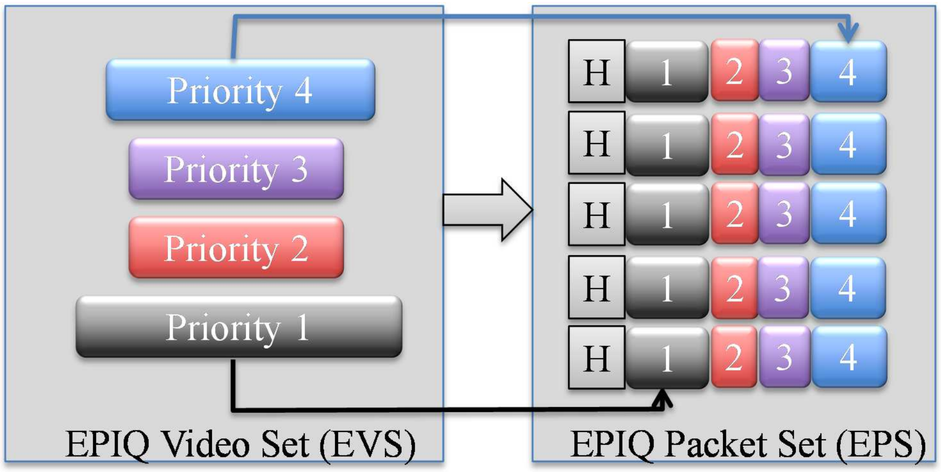


Figure 2.1: The mapping of multi-priority video content from an EPIQ Video Set (left) to Partially Erasable Packets of an EPIQ Packet Set (right)

EVS could also correspond to a single scalable frame that includes multiple layers of a scalable video stream. Hence, an EVS represents a video content that can be partitioned into multiple priority levels (the left side of Fig. 2.1). ***EPIQ Packet Set*** (EPS) represents the set of PEP packets that carry the content of one EVS. Thus, we divide each EVS priority level into different segments that we distribute among the packets of an EPS. Although not necessarily the case, in Fig. 2.1, each EVS priority level is divided into equal-size segments, which are uniformly distributed among the EPS packets.

The novelty and power of the proposed EPIQ packet structure is that it enables a simple and best-effort-like queue management mechanism while mimicking and achieving the benefits of multi-priority queue management approaches. During congestion, and instead of dropping a complete packet, a router can now simply erase an arbitrary portion of a PEP packet starting from its tail. It is important to note that the router does not have to be aware of the number of priority segments or their sizes within a PEP packet. Based on a RED-like AQM algorithm, the router can decide the size of the portion to be dropped from any randomly selected set of PEP packets. Hence, this partial erasing paradigm leads to dropping low-priority content and preserving high-priority content. Consequently, an EPIQ streaming solution can achieve the performance of complex multi-queue systems while maintaining the low-complexity of best-effort solutions without the need to manage a large number of queues or maintain the state information of a large number of flows. The only added complexity for an EPIQ-based solution is that the router needs to simply identify if the packet has a traditional structure or if it is a PEP packet. This requirement can be simply achieved, for example, by using the IPv4 header's differentiated services code point (DSCP) and flag fields. Also, to increase the speed of the packets selection process, the router could maintain a list that points to the starting locations of the PEP packets in the queue. Finally, it is worth noting the following. For a given EPIQ packet set, the EPIQ sender usually generates *equal-priority packets* since each PEP packet contains the same priority levels. This notion reinforces the need for single queue AQM management since all PEP packets are, in general, equally important.

## 2.2 Systems Analytical Model

In this section, we develop an analytical model for the performance of the proposed system. This analytical model is intended to provide an insight into how our system performs in comparison with other traditional streaming systems.

### 2.2.1 EPIQ

Our analysis is applied to an EPIQ packet set (EPS) that carries the content of multi-priority video. An EPIQ packet set can represent the content associated with a single scalable video frame that is mapped onto this packet set. Or, an EPS set can carry the content of a complete GoP of scalable or non-scalable frames. In general, successful decoding of low-priority data requires successful delivery and decoding of higher-priority data; otherwise, lower-priority data become useless even if they are delivered. Hence, regardless of the form of an EPIQ packet set, it is crucial to evaluate the utility of the data delivered. Thus, our objective here is to quantify the performance of our system by measuring the *utility* of the delivered data to the decoder. This is crucial for achieving optimal video delivery since reducing congestion by dropping video content should have minimal detrimental effect on video quality.

Let $p$ be the probability of dropping a packet within the network under traditional best-effort streaming (e.g., using HAS or RTP). If the average size of a packet is $S$ bytes, then $p$ is also the probability of dropping $S$ bytes. Under the proposed EPIQ framework, $p$ becomes the probability of erasing the same number ($S$) bytes from PEP packets in the network queue. Hence, in either case, $p$ represents the average loss rate.

For our analysis, we use a set of parameters that are outlined and described in Table 2.1. Further, Fig. 2.2 shows an example for mapping a traditional scalable video frame with $N_P$ packets into an EPIQ packet set using the same number $N_P$ of Partially Erasable Packets (PEPs).

Let $B_T$ be the size of one EPIQ packet set in bytes. Since $S$ is the packet size in bytes, for both the traditional and EPIQ packetization we have the following:

$$B_T = N_P \cdot S \tag{2.1}$$

Table 2.1: Stochastic model parameters

| **Parameter** | **Description** |
|---|---|
| $N_P$ | Number of packets in an EPIQ packet set (EPS). For consistency, this is the same number of packets used to packetize the corresponding video data according to a traditional approach. |
| $S$ | Size of each packet in bytes. This can be the average size of a packet when the packets have variable sizes. Or, it can be a constant if the same size used for all transmitted packets. |
| $N_E$ | Number of packets (from an EPIQ packet set) that are subjected to partial erasing. |
| $\delta$ | Number of bytes that are erased (or "deleted") from each of the $N_E$ Partially Erasable Packets (PEPs) of an EPIQ packet set. Note that $\delta$ varies from one EPS to another. |
| $\bar{\delta}$ | The expected value of $\delta$. |
| $\Delta$ | Expected loss in one layer |
| $B_T$ | Total number of bytes in an EPIQ packet set: $$B_T = N_P \cdot S$$ |
| $B_{lost}$ | The total number of bytes that are erased (lost) by a router due to a congestion event. Under EPIQ, this erasing takes place over $N_E$ Partially Erasable Packets (PEPs). $$B_{lost} = \delta \cdot N_E$$ For consistency, $B_{lost}$ also represents the total number of bytes lost due to packet dropping under a traditional approach. |
| $B_{un}$ | The number of bytes of an EPS that become useless (or unused by the video decoder) due to a loss/erasing event. |
| $B_U$ | Useful data, measured by number of bytes from an EPIQ packet set, that are used by the video decoder: $$B_U = B_T - (B_{lost} + B_{un})$$ |

Note that Eq. (2.1) is valid for both our proposed solution and a conventional system. One of the important aspects of our proposed analysis is that it maintains consistency in terms of the amount of lost data in comparison with a conventional system. To capture this consistency, let $B_{lost}$ be the number of bytes that would be lost in a conventional system due to packet dropping. Now, if the probability of dropping a packet is $p$, then the expected value of the number of bytes to be lost within an EPIQ packet set can be written as:

$$E[B_{lost}] = p \cdot B_T \tag{2.2}$$

Similarly, we use $B_{lost}$ to represent the total number of bytes lost from an EPIQ packet set due to partial erasing. Note that in a conventional system, $B_{lost}$ corresponds to the number of bytes lost due to dropping complete packets; while in our proposed solution $B_{lost}$ represents the number of bytes lost due to erasing portions of multiple packets. Either case, we use the same value for $B_{lost}$ for the sake of consistency and fairness when comparing the two solutions (traditional and EPIQ).

To capture our partial erasing (PE) process for distributing the loss $B_{lost}$ bytes over multiple packets, we define two parameters $N_E$ and $\overline{\delta}$. Fig. 2.2 shows an illustrative example, where $N_E =$ 3 packets that have been randomly selected by the network for partial erasing. In general, each selected packet that is subjected to partial erasing encounters a different level of erasing. Note that the Fig 2.2 shows that the $N_E = 3$ selected packets are contiguous. This is not necessarily the case; contiguous packets are shown in the Fig 2.2 mainly for ease of presentation. Also, the dropped portions are arbitrary, and they do not have to coincide with the border of video slices within the selected PEP packets. While there are many PEP packets in the queue, the EPIQ AQM algorithm can select any number of PEP packets. As our analysis and simulations will demonstrate, optimum video quality can be achieved by selecting the maximum possible number of packets in the queue for partial erasing. This process, however, leads to a variable number ($N_E$) of packets (from a given EPIQ packet set) that encounter partial erasing. Consequently, when congestion occurs at the network, then at the receiver, we may observe anywhere from a single packet to a maximum number of $N_P$ packets that have suffered from partial erasing for a given EPIQ packet set. Hence, under congestion, $N_E$ is a uniform random variable with a probability mass function $\frac{1}{N_P}$. It is important to note that we are assuming that the erasing happens at the slice boundary. We call this EPIQ Fine Grain Scalability (FGS) model.

The second parameter in this analysis, $\bar{\delta}$, represents the average number of bytes that are erased from the $N_E$ packets selected for partial erasing. Consequently, for an EPIQ based loss event, on average, we have the following:

$$\bar{\delta} = E\left[\frac{B_{lost}}{N_E}\right] \tag{2.3}$$

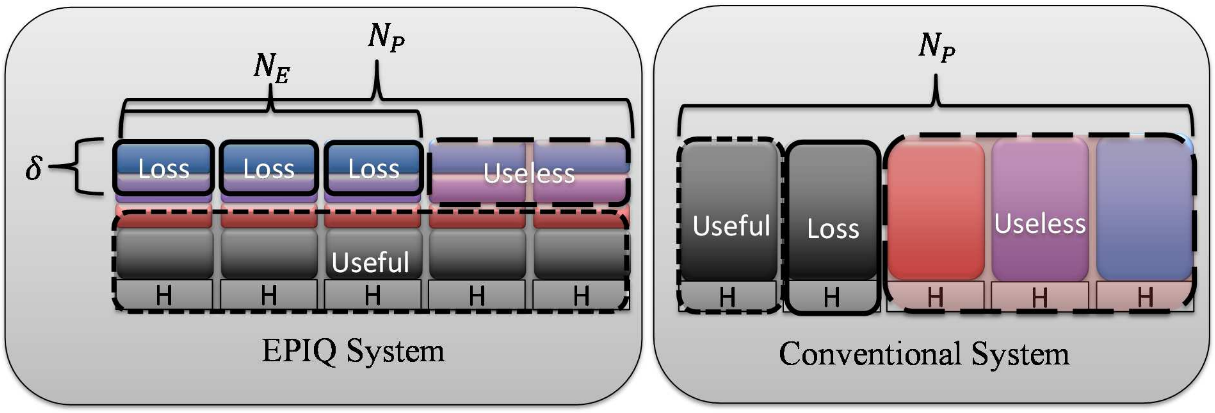


Figure 2.2: EPIQ vs. conventional packetization

Now, we would like to compute the amount of useful data received by the decoder. Let $B_U$ be the useful data (measured in bytes) recovered from one EPIQ packet set (EPS), which consists of

$N_P$ PEP packets. As illustrated in Fig. 2.2, the useful portion is the lower part of PEP packets that belong to the same EPIQ packet set. Hence, under this model, *on average*, erasing a set of $\bar{\delta}$ bytes from the $N_E$ packets that have been subjected to partial erasing will, on average, cause $\bar{\delta}$ bytes from all other $(N_P - N_E)$ packets of an EPIQ packet set to become useless. Consequently, we have the following:

$$E[B_U]_{FGS} = B_T - \bar{\delta} \cdot N_P = B_T - E\left[\frac{B_{lost}}{N_E}\right] \cdot N_P \tag{2.4}$$

$$E\left[\frac{B_{lost}}{N_E}\right] \cong \frac{E[B_{lost}]}{E[N_E]} - \frac{cov(B_{lost}, N_E)}{E^2[N_E]} + \frac{var(N_E)E[B_{lost}]}{E^3[N_E]} \tag{2.5}$$

Here, we assume that: (a) the variation in $N_E$ value is rather small relative to its mean; and (b) the correlation between $B_{lost}$ and $N_E$ is also small relative to the expected value of $N_E$. Our simulation results confirm that these assumptions are quite reasonable. This leads to the first order approximation of Eq. (2.5).

$$E\left[\frac{B_{lost}}{N_E}\right] \cong \frac{E[B_{lost}]}{E[N_E]} \tag{2.6}$$

Hence,

$$E[B_U]_{FGS} = B_T - \frac{E[B_{lost}]}{E[N_E]} \cdot N_P \tag{2.7}$$

Or,

$$E[B_U]_{FGS} = B_T \left(1 - \frac{p. N_P}{E[N_E]}\right) = B_T \left(1 - \frac{2. p. N_P}{1 + N_P}\right) \tag{2.8}$$

In the FGS model we assume that the erasing happens at the EPIQ slice boundary. We would consider the FGS an ideal model for our system performance when we have Multi Grain

Scalability (MGS) coding like Scalable Video Coding (SVC). In practice, the lower priority frames tend to be small, thus it is more likely to experience drop at the boundary rather than experiencing partial erasing. When a layer experiences partial erasing the whole layer become useless. Fig. 2.3 shows the difference between the two models.

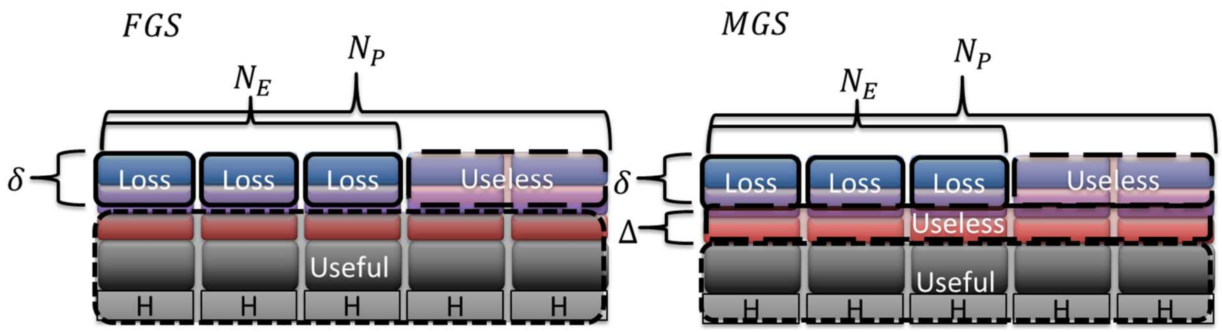


Figure 2.3: EPIQ FGS vs. EPIQ MGS packetization

As we stated earlier, the EPIQ Packet Set (EPS) may carry one frame and its enhancement layers or a GoP frames. Let's assume that each PEP consists of $k$ slices ( $L_1 \ldots L_k$) and a slice size of $m_i$ bytes. Low priority layers are small which results in small slices size that reduces our useless data. However, for simplicity, let's assume all layers that are carried by one EPS have a similar size of $M$ such that

$$B_T = N_P \left( \sum_1^k m_i \right) = N_P \; (k * M) \tag{2.9}$$

Even losing the smallest portion of a layer will result in a useless layer. Since we are assuming equally sized slices, then we can find the useless portion of the layer $\Delta$ in one PEP using $[B_U]$ in Eq. (2.8) as

$$\Delta = \left( \frac{[B_U]}{M * N_P} - \left\lfloor \frac{[B_U]}{M * N_P} \right\rfloor \right) . \; M \tag{2.10}$$

The expected usefulness for our MGS model $[B_U]_{MGS}$ can be rewritten as follows

$$E[B_U]_{MGS} = B_T - (\bar{\delta} + \Delta) \cdot N_P \tag{2.11}$$

To further normalize the models, we define a utility function $U$:

$$U = \frac{E[B_U]}{E[B_U]_{Capacity}} \tag{2.12}$$

Where $E[B_U]_{Capacity}$ is the channel capacity which is:

$$E[B_U]_{Capacity} = N_p(1-p) \tag{2.13}$$

$$U_{EPIQ\_FGS} = \frac{B_T - \bar{\delta} \cdot N_P}{1-p} \tag{2.14}$$

$$U_{EPIQ\_MGS} = \frac{B_T - (\bar{\delta} + \Delta) \cdot N_P}{1-p} \tag{2.15}$$

A comparison between the two models is shown in Fig. 2.4. It is clear that the ideal model (FGS) performs better than the MGS model. However, we can notice that increasing the number of layers in each packet improves the performance. Hence, we can improve the performance by increasing the size of EPIQ Packet Set (EPS)

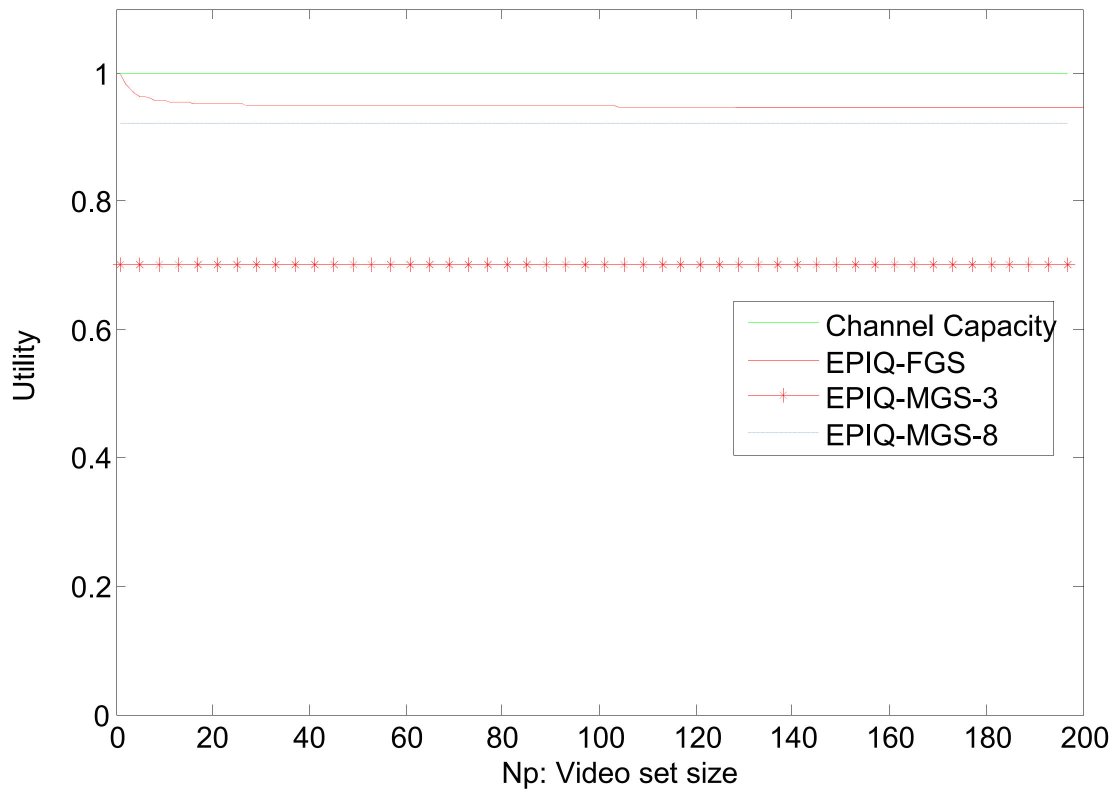


Figure 2.4: EPIQ FGS vs. EPIQ MGS utilization

### 2.2.2 Best Effort

In Best Effort (BE) video transmission, all video packets are treated equally regardless of the packets contents priority. Similar to our proposed system, EPIQ, we measure the performance of the BE system by computing the *usefulness* of the received packets. An analytical model for Fine Granularity Scalability (FGS) stream is developed in[36] for Best Effort streaming. The model can be used to measure the performance of a packet with sequential dependency. Using the same terminology for EPIQ, let $p$ be the probability of packets dropping and $N_P$ be the number of packets in a video set. The video set can be GoP or SVC frame. Now, let us assume that the distance between two packets drop locations is a geometric random variable. If packet $i$

experiences packet loss, then $i-1$ packets are useful. The expected usefulness of BE can be written as:

$$E[B_U]_{BE_{FGS}} = \sum_{i=1}^{N_P}(i-1)(1-p)^{i-1}p + \sum_{i=N_P+1}^{\infty} N_P(1-p)^{i-1}p \tag{2.16}$$

$$E[B_U]_{BE_{FGS}} = p\sum_{i=1}^{N_P}(i-1)(1-p)^{i-1} + N_P(\,1-p\sum_{i=1}^{N_P}(1-p)^{i-1}\,) \tag{2.17}$$

$$E[B_U]_{BE_{FGS}} = p\sum_{i=0}^{N_P-1} l(1-p)^{l} + N_P(\,1-p\sum_{i=0}^{N_P-1}(1-p)^{l}\,) \tag{2.18}$$

The two summation~~s~~ terms can be reduced to :

$$E[B_U]_{BE_{FGS}} = \tfrac{1-p}{p}(1-(1-p)^{N_P}) \tag{2.19}$$

Similar to the EPIQ system , we developed an MGS model for the Best effort system. Fig 3.5 shows the difference between the FGS and the MGS models of BE system.

The expected usefulness for the MGS model of the Best Effort model can be written as

$$E[B_U]_{BE_{MGS}} = \left(\sum_{i=1}^{k} L_i\right)\left[\prod_{i=1}^{k}(1-p)^{L_i}\right]\,(1-(1-p)^{L_{n+1}}) \tag{2.20}$$

Similar to EPIQ, we compare the utilization of the MGS and the FGS models of the BE system. Fig. 2.6 shows a comparison between the two Best Effort models. It is clear that the FGS model performs better than the MGS model.

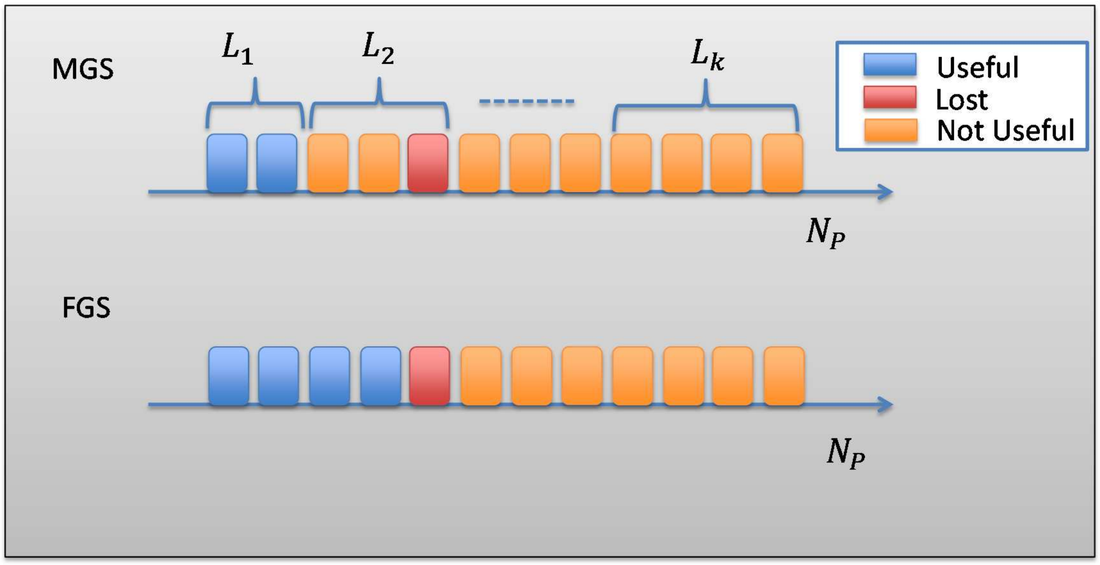


Figure 2.5: BE FGS vs. BE MGS packetization

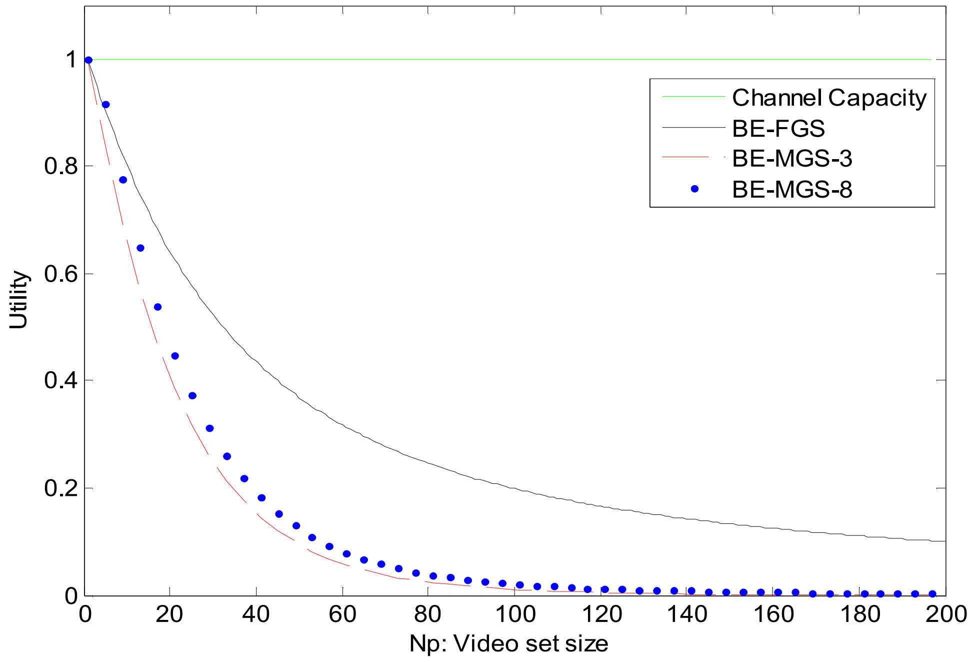


Figure 2.6: BE FGS vs. BE MGS utilization

### 2.2.3 Multi Priority Queue (MPQ)

In MPQ, the system manages a number of queues where each queue has a different packet dropping precedence. Hence, $N_P$ packets are enqueued into MPQ system according to packet importance. In an ideal scenario, only packets that belong to the lowest priority queue are dropped during congestion. In that case, we can consider Eq.(2.12) to represent the expected useful data of the last queue (least important) of the MPQ system while higher priority queues do not experience any packet dropping. Assuming that the MPQ system has $m$ queues and each queue can accommodate $k_i$ packets, then, the useful data of the MPQ system $E[B_U]_{MPQ}$ can be expressed in terms of the variable $E[B_U]_{MPQ}^m$ , which is the expected usefulness of the $m_{th}$ queue. Thus, Eq. (2.11) can be rewritten as:

$$E[B_U]_{MPQ_{FGS}} = \sum_{i=1}^{m-1} k_i + E[B_U]_{MPQ}^m \tag{2.21}$$

Where $E[B_U]_{MPQ}^m$ is the expected usefulness of the $m_{th}$ queue. Eq. (2.13) can be rewritten as:

$$E[B_U]_{MPQ_{FGS}} = (N_p - k_m) + \frac{1-p_m}{p_m}(1-(1-p_m))^{k_m} \tag{2.22}$$

It is important to note that $p_m$ is not the same as the $p$ that is found in Eq.(2.12); however, it is related to it. $p_m$ can be calculated using the following relationship :

$$p_m = \frac{p(N_p - k_m)}{k_m} + p = p(1 + \frac{N_p - k_m}{k_m}) \tag{2.23}$$

To develop our MGS model for the MPQ system, we use the same concept that we used in developing the FGS model. Using the BE MGS model as our reference, the MGS model for the MPQ system can be written as follows:

$$E[B_U]_{MPQ_{MGS}} = \sum_{i}^{L_m} L_i + \left(\sum_{i=L_m}^{k} L_i\right)\left[\prod_{i=L_m}^{k} (1-p_m)^{L_i}\right](1-(1-p_m)^{L_{n+1}}) \tag{2.24}$$

The FGS and MGS performances of the MPQ system are shown in Fig. 2.7. It is evidently that the FGS performs better performance. Furthermore, we can measure the performance of the MPQ system by comparing the number of priority queues with the number of packets carried. The best performance can be achieved when both the number of priority queues and the number of video layers are the same.

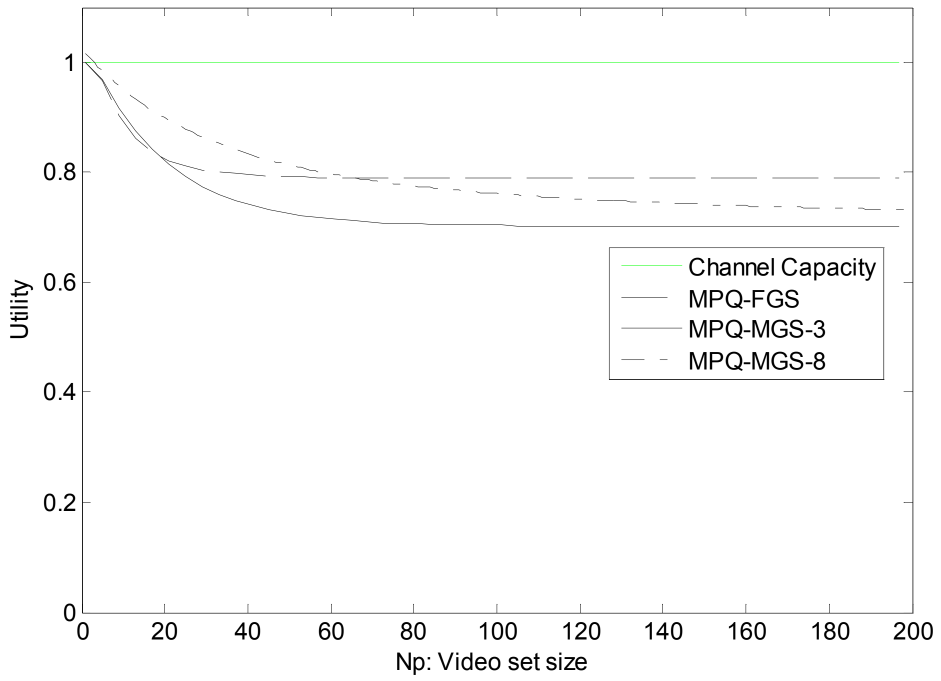


Figure 2.7: MPQ FGS vs. MPQ MGS utilization

The utility function represents the ratio of the useful information received relative to channel capacity. Fig. 2.8 shows a comparison of the utility as a function of the number of packets in an EPIQ packet set ($N_P$) for EPIQ, MPQ and BE FGS models. It is obvious that our system delivers significantly higher levels of useful data to the receiver when compared with BE. In addition, it also outperforms MPQ with 3 priority queues usefulness. For example, EPIQ at $N_P = 20$ achieves 90% utility while MPQ-3 and BE achieve 85.5% and 40% utility respectively. To Further illustrate the difference, when $N_P = 100$, EPIQ achieves 89% while MPQ-3 and BE could only maintain 76% and 10% respectively. Our extensive simulations will further validate the results of this model.

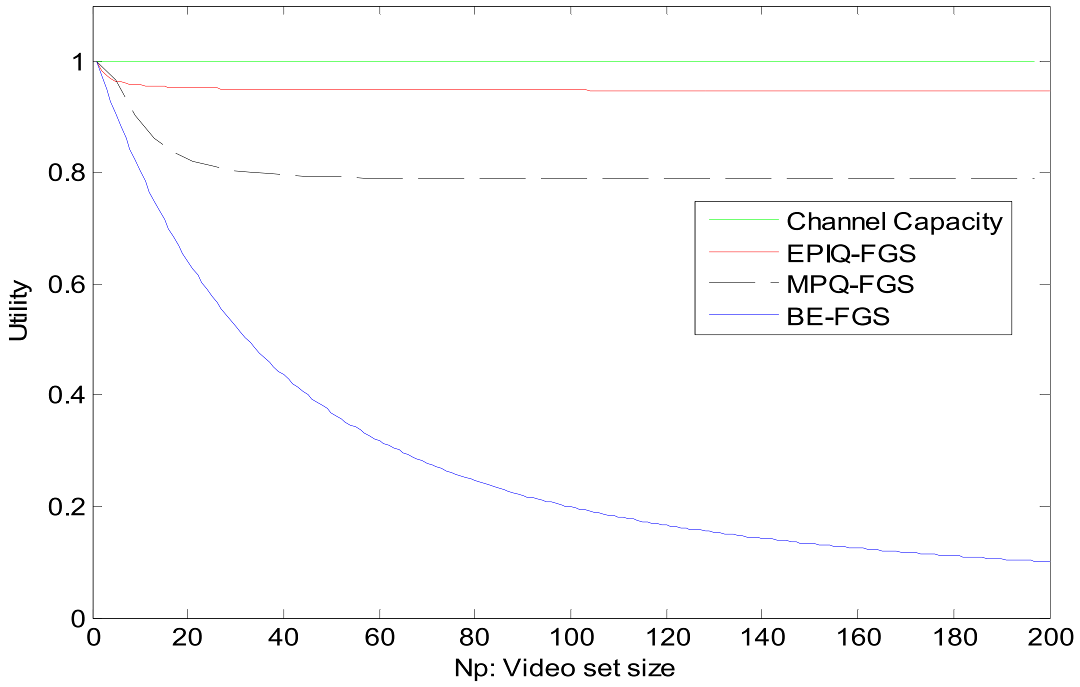


Figure 2.8: FGS utility for EPIQ, BE and MPQ

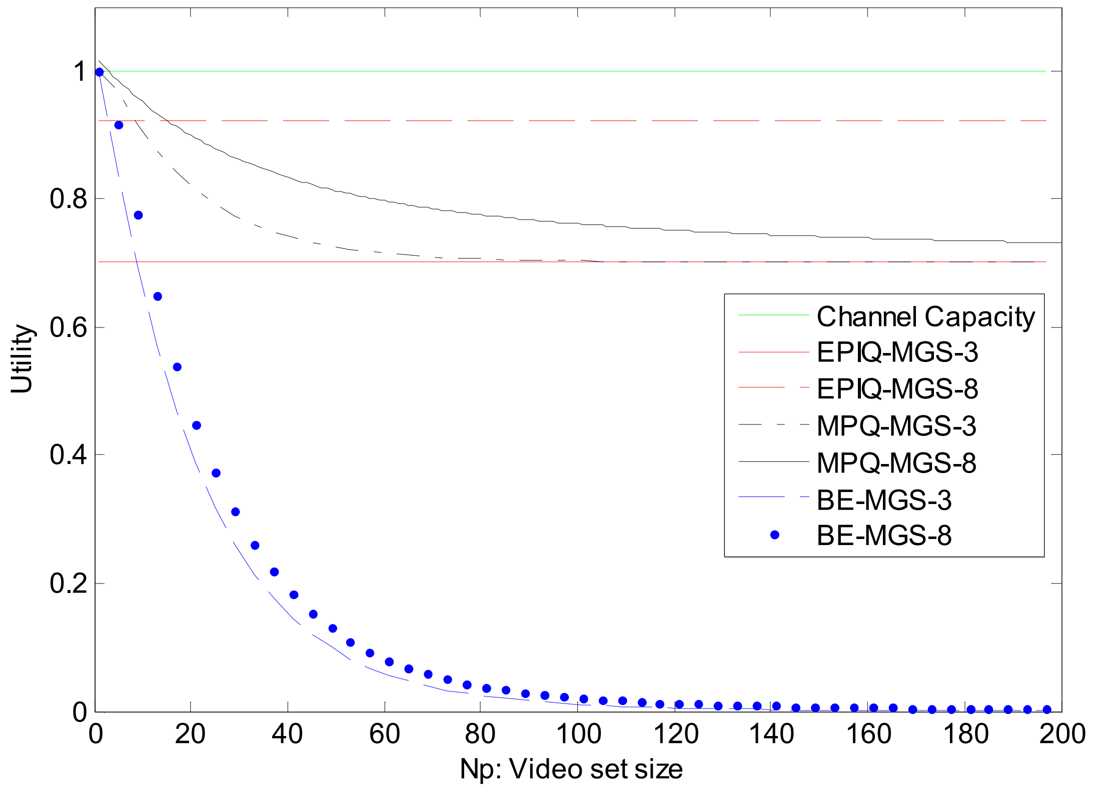


Figure 2.9: MGS utility as a function of video-set size for EPIQ, BE and MPQ

Fig. 2.9 shows a comparison of the utility as a function of the number of packets in an EPIQ packet set ($N_P$) for EPIQ, MPQ and BE MGS models for a frame with 3 and 8 layers. For the case of MPQ, we used 3 priority queues. For the 3 layers video, only the lowest layer experienced packet loss while for the 8 layers video, the last two layers experienced packet loss. It is obvious that for the 8 layer frame, our system delivers a higher level of useful data to the receiver when compared with the BE and MPQ models. It is also clear that MPQ achieves optimum performance when only the lowest priority experiences lost at the lowest priority queue. In other words, the MPQ model achieves the optimum performance when the number of queues equals the number of layers. It is important to note that the BE model with 8 layers has higher utility than BE with 3 layers. Hence, utility increases as the number of layers increases. Our extensive simulations will further validate the results of this model.

## 2.3 Experimental Evaluation

In this section we discuss the experiment that is conducted to verify our model

### 2.3.1 Experimental Setup

We used the ns-2 simulator to implement the EPIQ framework. The simulation topology consisted of a sender, receiver and a middle box that represents the congested network. The video content was generated and evaluated using JVT reference software[56], SAEF framework [57] and Open SVC Decoder[58]. We simulate both AVC and SVC [59] video stream of Park Run video sequence encoded at 50 FPS and GoP of size 8. The SVC stream consisted of one base layer and two enhancement layers and GOP of size 8. Both streams have Peak Signal to Noise Ratio (PSNR) quality of 36.6 dB. We evaluated the performance of our proposed method and multi-priority systems at packet loss probability $p$ of 1%, 5%, 10%, 20% and 30%. Best effort video transmission results were not included in our experiment because the results were always showing the same result of 17dB which is the same as random noise data. Therefore, we only compared our model with Multi-Priority Queue system.

**a) EPIQ**

To configure EPIQ, we need to determine the content of our EPS and the number of packets to be erased during congestion. For the AVC video stream, we specified that each EPIQ video set (EPS) carry the content of one GoP, which consists, in our simulations, of 8 frames. In the SVC case, we have chosen two different configurations; in the first configuration; the EPS carried the content of one SVC frame (3 layers), while in the second configuration, the EPS carried the content of one SVC GoP frames. As stated earlier in the analysis section, any number of EPS packets can be selected during congestion.

**b) Multi Priority Queue (MPQ)**

In the multi-priority system case, one queue is assigned for each priority in the congested node so that during congestion the lowest priority queue drops its packets. For AVC stream, we decided to use the most common number of priority queues which is three based on frame types (I, P and B) classification. In a second scenario, we decided to use the maximum number of possible priorities that the video supports to capture the frame's dependency. In such ideal case, the number of priorities is the GoP size, which is eight in our configuration. For SVC video, first we decided to use three priority queues and assign the highest priority to the base layer while the second and third (lowest priority) are assigned to the first and second enhancement layers respectively. In another SVC scenario, similar to AVC 2nd scenario, we set the number of priorities to the maximum number of priorities that can be supported by the SVC stream. In such case, the number of priorities is set to the GoP size multiplied by the number of video layers (i.e., the total number of priorities is 24 in the ideal SVC case).

### 2.3.2 Experimental Results

Fig..2. shows the average PSNR of the received video of both EPIQ and the multi-priority solution for the AVC stream case. Our proposed method (EPIQ-GoP), on average, is both 6dB better than the system supporting three-priority queues (Pri-3) within the network, and only 0.8 dB less than the significantly more complex (ideal) eight-priority system (Pri-8). The reason behind the poor performance of the three-priority system is that although it gives highest priority to I and P frames, it assigns the same lowest priority to all B frames without taking into consideration the dependability among B frames. Meanwhile, the eight-priority system takes into consideration all frame dependencies with added overhead of managing eight queues rather than three. On the other hand, our system employs a single queue without complex management or significant overhead, and its performance is independent of the number of priority levels that are carried by the video packets.

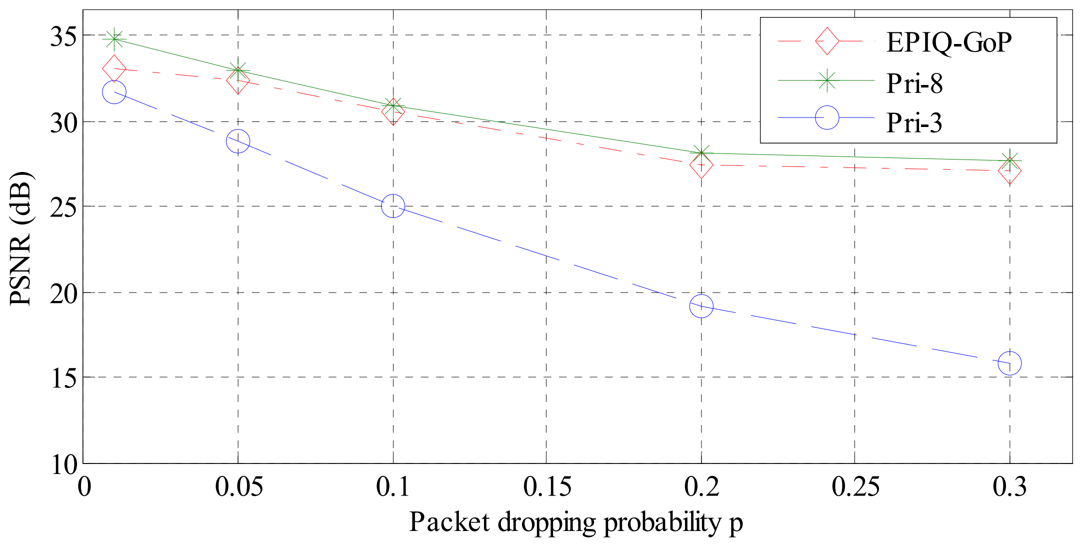


Figure: 2.10: PSNR of EPIQ and MPQ for AVC stream

Fig. 2.5 shows the results of SVC video streaming for both EPIQ and the multi-priority system. For the three-priority queues (Pri-3), it is noticeable that the PSNR values do not change even when the number of dropped packets increases. The reason for such behavior is that the decoder is only decoding the first enhancement layer. Similar to the AVC case, not all of layer two (enhancement layer) packets have the same priority. Our proposed system performance with EPS carrying only one frame content (EPIQ-Frame) is 3 dB better than the three-priority systems (Pri-3). It is important to note that EPIQ with EPS carrying GoP (EPIQ-GoP) achieves, on average, virtually the same performance of the ideal high-complexity 24 -priority system (Pri-24); the difference in performance is only 0.3dB.

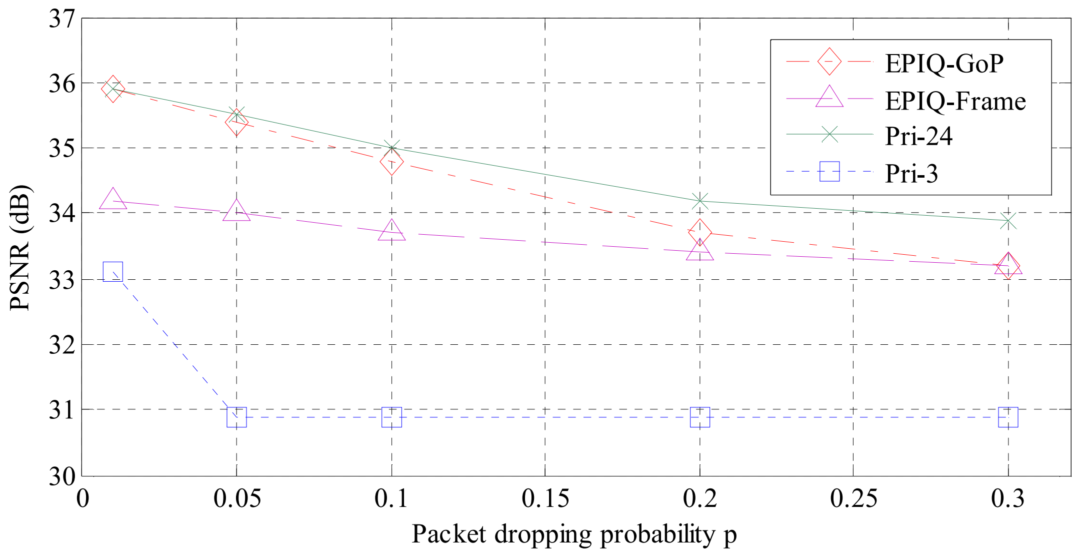


Figure 2.11: PSNR of EPIQ and MPQ for SVC stream

# Chapter 3 EPIQ System Implementation

To implement the concept that we discussed in previous chapter, we use Active Queue Management (AQM) that exists in the network nodes. So in this chapter, we first discuss our proposed AQM algorithm. Also, to complement our AQM role, we implement rate control algorithm which is discussed in the second part of this chapter.

## 3.1 Active Queue Management

In this section we propose a mechanism as a part of our framework. The goal of the algorithm is to maximize the performance of Partial Erasing process in addition to its role as AQM algorithm.

### 3.1.1 EPIQ-Random Early Detection algorithm (EPIQ-RED)

Despite the many AQM approaches that have been proposed, only a few have been incorporated in Internet routers and are being used in practice[27]. Hence, we are interested in focusing on employing existing AQM algorithms and concepts that have been used in practice in conjunction with the proposed EPIQ framework. In particular, our AQM algorithm is built on the popular Random Early Detection (RED) algorithm[60]. In light of efforts that studied, analyzed, and improved RED [60][61], we designed our algorithm by taking these approaches into consideration; and we call our approach EPIQ Random Early Detection (EPIQ-RED). It is important to note that we use Byte version (Byte mode) and packet marking (Explicit Congestion Notification (ECN)) of the RED algorithm.

Under RED, the AQM algorithm drops a packet entering the queue with a certain probability when the queue size operates within certain thresholds. Here, $p_a$ represents the AQM algorithm packet dropping probability, and $q_{avg}$ is the queue average size in bytes. The $min_{th}$ and $max_{th}$

parameters are the RED maximum and minimum thresholds, respectively. To calculate the dropping probability $p_a$, we modified the original RED algorithm by fixing the $max_p$ parameter of RED to 1. Thus leading to:

$$p_a = \frac{(q_{avg} - min_{th})}{max_{th} - min_{th}} \tag{3.0}$$

The advantage of fixing the parameter $max_p$ is to have a continuous linear dropping probability. Discontinuity in the dropping probability has a negative effect on the performance [61][21]. Many studies have proposed a modified RED that enables automatic parameters configuration; however, in many cases such configuration is dynamic in nature. In our case, we introduce simple modifications to the original RED, where we take into consideration the proposed EPIQ system requirements in terms of delivering optimal video performance while maintaining RED automation. One key EPIQ-RED parameter is $N$, which is related to (but not the same as) the parameter $N_E$ described in Chapter 2. Here, $N$ represents the number of PEP packets that are selected randomly from the queue for partial erasing. These $N$ packets that are selected for partial erasing might belong to multiple EPIQ packet sets that belong to multiple users. On the other hand, $N_E$, which was employed by our analysis in Chapter 2, is the number of packets within a single EPIQ packet set that suffer from partial erasing (as observed by a given single receiver). Hence, under EPIQ-RED, $N$ PEP packets, which may belong to one or more EPIQ packet sets, are randomly selected from the queue for partial erasing. As explained earlier, for an optimal EPIQ performance, we need to have the value of $N$ as large as possible. Consequently, it is possible that all the PEP packets that belong to the same EPS can be selected. Assuming that we have different EPIQ packet sets with different sizes in the queue, under EPIQ-RED, we set $N$ to some maximum value $N_{max}$. Here, $N_{max}$ can be thought of as the maximum number of packets of an EPIQ packet set. However, in cases when $N_{max}$ is larger than the current number of PEP packets in the queue, $N$ is set to the current number of EPIQ packets in the queue $q_{EQ}$. Hence,

$$N = min(q_{EQ}, N_{max}) \tag{3.1}$$

Recall that PEP packets have a size of $S$ bytes. Although it is possible to erase more than one packet size $S$ of bytes at a time, we use the RED standard queue management procedure where one packet is dropped (or erased) at a time. This is also important to achieve fairness with different non-EPIQ flows. The amount of data that are erased from each of the $N$ selected packets is represented by $\delta$:

$$\delta = \frac{S}{N} \tag{3.2}$$

It is important to note that in our design of EPIQ-RED, we take into consideration that the network queues might have a mix of EPIQ and traditional packets. Hence, to accelerate the process of packet erasing, EPIQ-RED has to maintain the location and the number of EPIQ packets in the queue. A virtual list can be used to point to the location of the EPIQ packets in the queue. The minimum number of packets for RED threshold $min_{th}$ is usually set to a number of bytes equivalent to five packets as recommended by [60][62]. However, and since we are planning to spread the erasing process over as many EPIQ packets as possible, a small queue will affect EPIQ transmitted video quality as shown in Eq. (3.1)-(3.2). Thus, we set the minimum threshold $min_{th}$ as the maximum between the RED recommended threshold and the average value of the EPIQ packet set size $\overline{N}_P$. On the other hand, a large queue will cause long delays. As discussed in [32], limiting the number of packets in the queue reduces packet latency and prevents buffer-bloating. Hence, we set $max_{th}$ as the minimum between: (a) The number of packets in the queue $N_Q(\tau)$ that is equivalent to a $\tau = 0.2$ second packet delay; and (b) Twice the maximum size $N_{max}$ of an EPIQ packet set. The motivation for choosing twice the size of $N_{max}$, is because EPIQ video content will most likely occupy more than half of the queue. Setting $max_{th}$ to $2N_{max}$ allows $N_{max}$ or a higher number of PEP packets in the queue. Hence, in terms of bytes, we employ the following expressions for $min_{th}$ and $max_{th}$:

$$min_{th} = S \cdot max(5, 0.5\overline{N}_P). \tag{3.3}$$

$$max_{th} = S \cdot min(\, N_Q(\tau), 2 \cdot N_{max}) \tag{3.4}$$

To have a more stable queue we let $min_{th}$ and $max_{th}$ be increasing functions and updated every 5 minutes. In Section 3.3, we show the stability of our proposed AQM algorithm.

The algorithm works by evaluating $p_a$ for every incoming packet. If the packet that is marked to be dropped is a Partially Erased Packet (PEP), the packet is enqueued and $N$ PEP packets are randomly selected from the queue by using the virtual list and are partially erased; and only one packet is marked as lost. If it is not a PEP packet, the packet is dropped. Thus, we guarantee fairness between the different data types of traffic in the queue.

### 3.1.2 Stability analysis of EPIQ-Random Early Detection algorithm (EPIQ-RED)

To prove the functionality of our proposed algorithm, we used the linearized TCP model that was developed in [61]. The proposed model paved the way for the control theoretic AQM algorithm. The following is a brief review. Starting with the fluid model of a network that consists of a TCP sender, AQM and receiver.

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)W\big(t - R(T)\big)}{2R\big(t - R(t)\big)} p(t - R(t)) \tag{3.5}$$

$$\dot{q}(t) = \frac{W(t)}{R(t)} N(t) - C \tag{3.6}$$

Where $W, q, R, C, N \; and \; p$ are expected TCP windows size in packets, expected queue length in packets, round trip time, link capacity, number of TCP session and probability of packet marking/dropping, respectively. Since the TCP model is a nonlinear system, the model must be linearized to be able to do classical linear system analysis. The linearization process starts with defining an equilibrium point $(W_0, q_0, p_0)$ when $\dot{W}(t) = 0 \; and \; \dot{q}(t) = 0$ , So that

$$\dot{W} = 0 \;\Rightarrow\; W_0^2 p_0 = 2 \tag{3.7}$$

$$\dot{q} = 0 \Rightarrow W_0 = \frac{R_0 C}{N} \tag{3.8}$$

Where

$$\dot{W} = 0 \;\Rightarrow\; W_0^2 p_0 = 2 \tag{3.9}$$

Where $T_p$ is propagation delay. The TCP model is linearized about the equilibrium points. Using Taylor series linearization method, the linearized model after simplification is

$$\delta\dot{W}(t) = \frac{2N}{R_0^2 C}\delta W(t) - \frac{R_0 C^2}{2N^2}\delta p(t - R_0) \tag{3.10}$$

$$\delta\dot{q}(t) = \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta q(t) \tag{3.11}$$

Where

$$\delta W = W - W_0 \tag{3.12}$$

$$\delta q = q - q_0 \tag{3.13}$$

$$\delta p = p - p_0 \tag{3.14}$$

The control system for the linearized system is depicted in Fig 3.1.

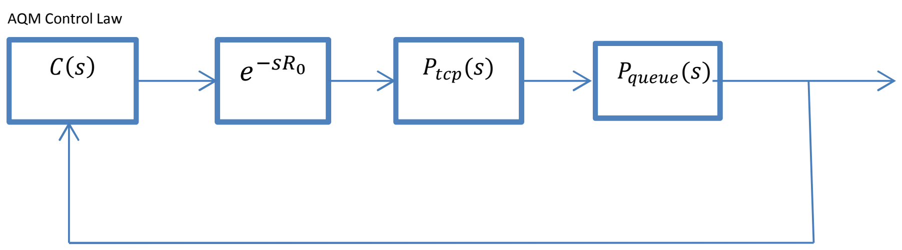


Figure 3.1: Linearized control system

$$P_{tcp} = \frac{\frac{R_0 C^2}{2N^2}}{s + \frac{2N^2}{R_0^2 C}} \tag{3.15}$$

$$P_{queue} = \frac{\frac{N}{R_0}}{S + \frac{1}{R_0}} \tag{3.16}$$

From the above system, since both poles are negative the system is stable. Let assume that W $C(S)$ is the AQM control function. Since RED is a low pass filter then the transfer function of the system can be represented as [ low pass filter analysis]

$$C(s) = C_{red}(s) = \frac{L_{red}}{\frac{s}{K} + 1} \tag{3.17}$$

Where

$$L_{red} = \frac{P_{max}}{max_{th} - min_{th}} \tag{3.18}$$

$$K = \frac{log_e(1 - w_q)}{\delta} \tag{3.19}$$

Hence, the system transfer function can be written as:

$$C(s).P(s) = \frac{\left(\frac{C^2}{2N}\right) e^{-sR_0}\ \ L_{red}}{\left(s + \frac{2N}{R_0^2 C}\right)\left(s + \frac{1}{R_0}\right)\left(\frac{s}{K} + 1\right)} = G(s)e^{-sR_0} \tag{3.20}$$

***Lemma*** For all $N \in\ [N^-, \infty)$ and $R^0 \in\ (0, R^+]$ , then if $w_g$ is chosen based (3.21)

$$w_g = \propto min\left\{\frac{2N^-}{(R^+)^2 C}, \frac{1}{R^+}\right\}, \qquad 0 < \propto < 0.5$$

Then $L_{red}$ and $K$ satisfy the following condition

$$\frac{L_{red}(R^+C)^3}{(2N^-)^2} \leq \sqrt{\frac{{w_g}^2}{K^2} + 1} \tag{3.22}$$

And the closed-loop system Eq. 3.20 will be stable $\forall N \in [N^-, \infty)$ and $R^0 \in\ (0, R^+]$

Proof:

let $s = jw$ then we can rewrite Eq. 3.20 as follows:

$$G(jw)e^{-jwR_0} = \frac{L_{red}\left(\frac{C^2}{2N}\right)}{\left(\frac{jw}{K} + 1\right)\left(jw + \frac{2N}{R_0^2 C}\right)\left(jw + \frac{1}{R_0}\right)} \tag{3.23}$$

The magnitude of $(jw)e^{-jwR_0}$ :-

$$\left| G(jw)e^{-jwR_0}\right| = |\ G(jw)| \tag{3.24}$$

The transfer function $G(jw)$ can be rewritten as:

$$G(jw) = \frac{L_{red}\frac{(R_0C)^3}{(2N)^2}}{\left(\frac{jw}{K} + 1\right)\left(\frac{jw}{\frac{2N}{R_0^2 C}} + 1\right)\left(\frac{jw}{\frac{1}{R_0}} + 1\right)} \tag{3.25}$$

Suppose $N \in\ [N^-, \infty)$ and $R_0 \in\ (0, R^+]$, in this case if $N^- \to \infty$ and $R^+ \to 0$

$$G(jw) \rightarrow \hat{G}(jw) = \frac{L_{red}\frac{(R^+C)^3}{(2N^-)^2}}{\left(\frac{jw}{K}+1\right)}, \forall w \in [0, w_g] \tag{3.26}$$

Where $w_g$ : gain cross-over freq.

For stability requirements, it is required that

$$|\hat{G}(jw_g)| \leq 1 \tag{3.27}$$

This implies that

$$\frac{L_{red}\frac{(R^+C)^3}{(2N^-)^2}}{\sqrt{1+\frac{{w_g}^2}{K^2}}} \leq 1 \tag{3.28}$$

$$\frac{L_{red}(R^+C)^3}{(2N^-)^2} \leq \sqrt{\frac{{w_g}^2}{K^2}+1} \tag{3.29}$$

The phase of $G(jw)e^{-jwR_0}$ :-

$$\angle G(jw)e^{-jwR_0} = -wR_0 - \tan^{-1}\left(\frac{w}{K}\right) - \tan^{-1}\left(\frac{w}{\frac{2N}{R_0^2C}}\right) - \tan^{-1}\left(\frac{w}{\frac{1}{R_0}}\right) \tag{3.30}$$

Again if $N \in [N^-, \infty)$ and $R_0 \in (0, R^+]$, and for stability requirement (Nyquist stability criterion), it is needed to have

$$\angle G(jw_g)e^{-jw_gR_0} > -180 \tag{3.31}$$

$$-w_gR^+ - \tan^{-1}\left(\frac{w_g}{K}\right) - \tan^{-1}\left(\frac{w_g}{\frac{2N^-}{(R^+)^2C}}\right) - \tan^{-1}\left(\frac{w_g}{\frac{1}{R^+}}\right) + \pi > 0 \tag{3.32}$$

Let

$$P_{red} = min\left\{\frac{2N^-}{(R^+)^2C}, \frac{1}{R^+}\right\} \tag{3.33}$$

The idea to achieve condition Eq. 3.32 above, we choose $K$ such that the $C(s)$ will dominate the closed-loop system, to do that it is needed to choose $\frac{w_g}{K} \leq P_{red}$

If

$$w_g \leq P_{red} \rightarrow \frac{w_g}{K} \leq P_{red} \tag{3.34}$$

This implies that the term

$$-w_g R^+ - \tan^{-1}\left(\frac{w_g}{K}\right) - \tan^{-1}\left(\frac{w_g}{\frac{2N^-}{(R^+)^2C}}\right) - \tan^{-1}\left(\frac{w_g}{\frac{1}{R^+}}\right) \leq -\frac{\pi}{2} \tag{3.35}$$

If the inequality Eq. 3.35 satisfied, then it implies that inequality Eq. 3.32 will be satisfied. Therefore, let

$$w_g = 0.1P_{red} \tag{3.36}$$

In this analysis, we evaluated the RED stability boundaries. We will use Eq. (3.22) of analysis to make sure that our proposed system is stable.

## 3.2 Rate Control

In this section, we propose a mechanism as a part of our framework. The goal of the algorithm is to maximize data rate and share network resources fairly with other network flows .

### 3.2.1 EPIQ- TCP Friendly Control Protocol (EPIQ-TFRC)

The channel between the sender and the receiver should be monitored to ensure high-quality video delivery. Because of the changing condition of the network, rate control is necessary to maintain quality video streaming while ensuring fairness to other competing real-time and non-real-time sessions. Based on the reported available bandwidth, the sender adjusts the sending rate of the video stream. In general, the rate control mechanism that is used in video transmission either relies on the traditional Transmission Control Protocol (TCP) or multimedia oriented TCP friendly protocols. Each has its own advantages and disadvantages. The most popular video streaming mechanism, DASH, employs HTTP/TCP to transport video over the network. The video rate is adjusted according to reported available bandwidth by TCP. Other methods include Real-time Transport Protocol (RTP). Several congestion control mechanisms were proposed to be used in conjunction with RTP[3] like TFRC[21] GCC[22] and Kelly[62]. Equation based congestion control methods are the most widely used with RTP. Protocols like Datagram Congestion Control Protocol (DCCP)[63] and TCP-friendly Rate Control Protocol (TFRC) are well known RTP solutions. The congestion algorithm can be implemented as an application that uses UDP as a transport layer. Equation based control protocols use TCP response function to estimate the sending rate:

$$T = \frac{S}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)} \tag{3.37}$$

where $T$ is the sending rate in bytes/sec. while $S, R, p$ and $t_{RTO}$ are the packet size, round trip time, study state loss event rate and TCP retransmission timeout, respectively. The loss event is defined as a packet loss within a single Round Trip Time (RTT). Since our proposed solution does not drop whole packets but partially erases them, using a multimedia oriented rate control protocol is easier to implement. It is possible to employ TCP in conjunction with the EPIQ framework. However, mapping partially erased packets into equivalent conventional packet loss is a complex task. Our objective is to employ a rate control algorithm that achieves fairness with other flows that share the network resources yet is compatible with our novel solution. Hence, we propose a modified TCP-Friendly Rate Control protocol (TFRC) and refer to it as EPIQ-TFRC which is implemented as an application and uses UPD as a transport protocol. It is also important to note that we use Explicit Congestion Notification (ECN) mode of the TFRC algorithm. The TCP response function (Eq.(3.37) ) is used by TFRC to achieve TCP-friendly sending rate that maintains fairness with other TCP flows in the network. In our proposed solution, evaluating the round trip time $R$ and TCP time-out $t_{RTO}$ is similar to the proposed methods that are used in the TFRC standard[21].

In a conventional network, packet dropping is considered an indication of congestion. Meanwhile, in our solution we drop only portion of the packets and we only mark one of the partially erased packets using ECN bit in the header. The receiver detects only marked packets as a congestion indicator.  Using ECN mode of the TFRC standard, packet loss is calculated such that we achieve fairness with other TCP flows in the network.

After computing the packet loss, we use Exponential Weighted Moving Average (EWMA) to compute the packet event loss $p$ as shown in Eq. (3.31).

$$p_{event_{new}} = \beta * p_{cur} + (1-\beta) p_{event_{old}} \tag{3.38}$$

Where $p_{cur}$ is the current loss probability reported by TFRC while $p_{event}$ and $\beta$ are the event probability that is used to calculate current sending rate as Eq.(3.30) and history parameters respectively. The standard TFRC/RFC 5348 protocol uses Average Loss Interval (ALI) method to compute the event loss rate $p$. ALI response to network changes is better than EWMA. On the other hand, under EPIQ, we observed that EWMA provides smoother video transmission than the ALI averaging method.

Since the packet size $S$ in the TFRC standard is fixed, our solution has to be adaptable to such scenario. One possible resolution to this constraint is to pad the tail of a packet with zeros in case the packet is less than the desired packet size $S$. Because the zero padding is at the tail of the packet, the padding is first dropped during congestion. This method can be enforced on all EPIQ packets to provide extra protection to video slices during heavy congestion. In Section (3.3) we discuss the fairness of our rate control protocol.

Another modification to standard TFRC that we included is a video streaming profile control algorithm. EPIQ-TFRC selects the appropriate video transmission rate (video quality profile) by calculating the running average of sending bitrate using:

$$\hat{R}_{bps} = \begin{cases} \alpha\hat{R}(i-1) + (1-\alpha)R(i) & i > 1 \\ R\,(1) & i = 1 \end{cases} \tag{3.39}$$

We used $\alpha = 0.8$ in our simulation. $\hat{R}(i)$ represents the average sending rate at event $i$ where, here, the event duration is 10 seconds. Also, reducing the quality profile of the video is allowed only if the playback buffer size is less than 5 seconds. Hence, based on the running average and the playback buffer size, the closest video quality profile is selected.

## 3.3 Experimental Evaluation

In this section, we validate our proposed EPIQ-RED and EPIQ-TFRC by simulation.

### 3.3.1 EPIQ-RED

The main role of AQM algorithm is to improve channel utilization, minimize delay and prevent network collapse during congestion. Our work is based on the original Random Early Detection (RED) algorithm. In Fig. 3.2 we show an example of unstable RED performance while Fig 3.3 shows a stable RED performance of the same network configuration. The instantaneous queues size is changing rapidly which causes poor performance. The unstable AQM causes low channel utilization. Fig. 3.4 depicts a comparison of low and high channel utilization of Fig 3.2 and Fig 3.3 respectively. Due to unstable AQM, only 15% of the bottleneck bandwidth was utilized. The average network delay can affect network performance negatively. However, delay jitter has far worse implication than average delay on applications like multimedia and broadcasting. Furthermore, Table 3.1 shows the different parameters results for both stable and unstable AQM.

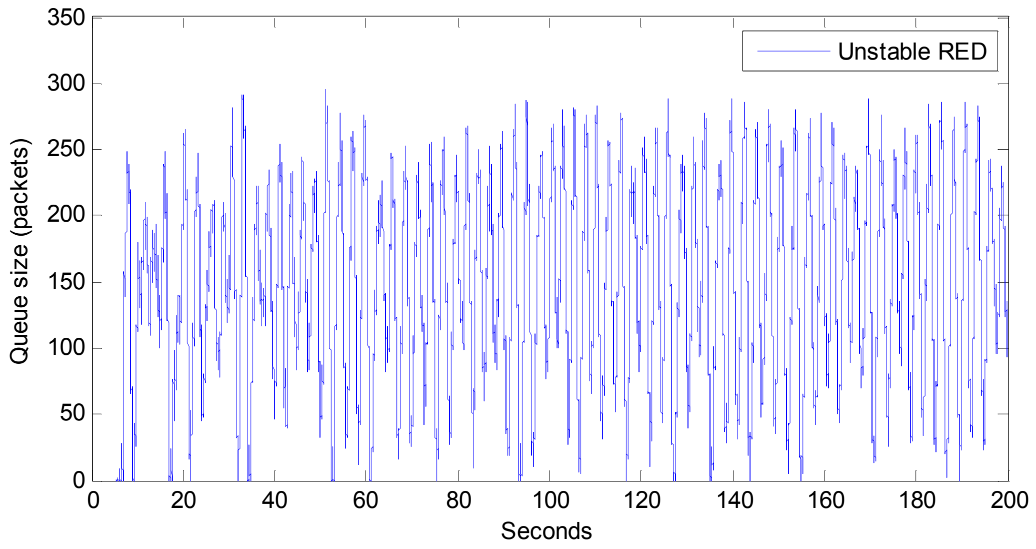


Figure 3.2: Unstable RED instantaneous queue size

Table 3.1: Comparison between stable and unstable AQM performance

| | **Avg. Throughput** | **Ch. Utilization** | **Delay (ms)** | **Delay Jitter (ms)** |
|---|---|---|---|---|
| **Unstable AQM** | 2.8 Mbps | 15 | 220 | 42 |
| **Stable AQM** | 13.4 Mbps | 97 | 240 | 16 |

It is clear from Table 3.1 that a stable AQM performance is necessary to have optimum network performance.

We tested our proposed EPIQ-RED algorithm with different network configuration to verify its performance. Fig. 3.5 shows instantaneous queue size of 75Mbps channel. The $min_{th}$ and $max_{th}$ are set to 75 and 575 respectively. Also, Fig 3.6 shows the network utilization of the bottleneck channel. Our AQM converges to 100% channel utilization and has minimal delay jitter.

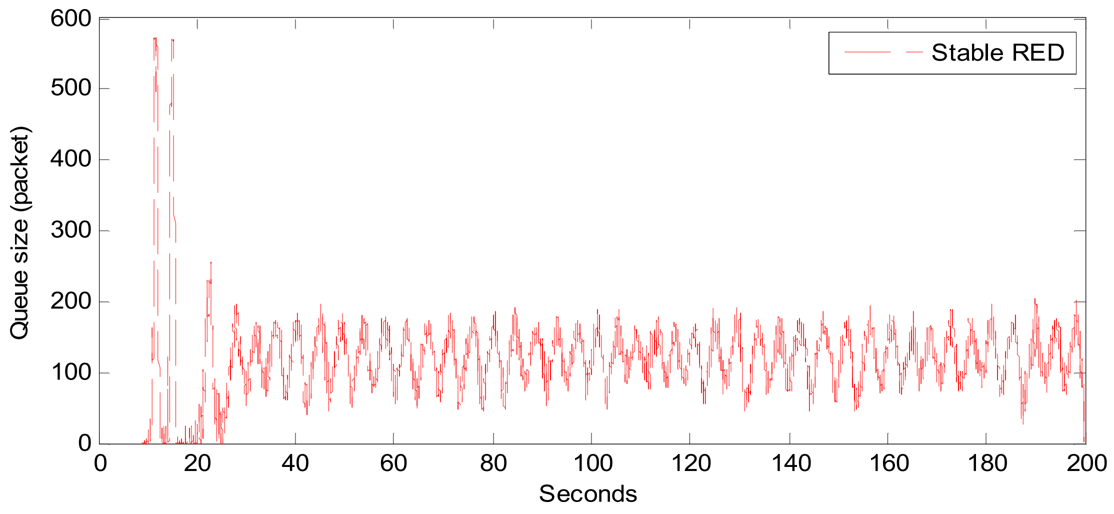


Figure 3.3: Stable RED instantaneous queue size

It is clear that our proposed algorithm has a stable network configuration. In another network configuration, the channel is set to 15 Mbps and $min_{th}$ and $max_{th}$ are set to 35 and 125 respectively. Fig. 3.7 and 3.8 depict the instantons queue size and channel utilization. Similar to the previous network configuration, the performance of our proposed queue is optimal.

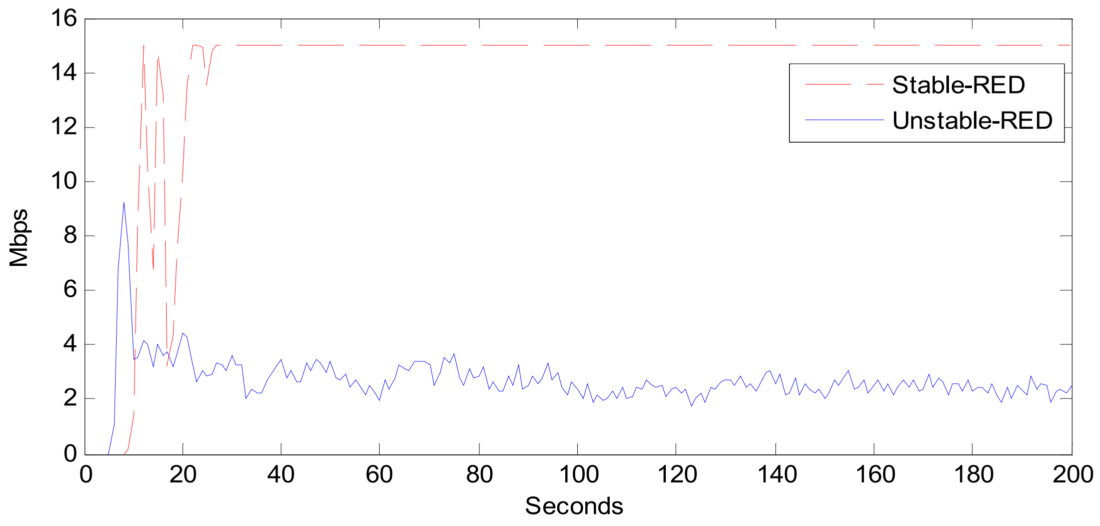


Figure 3.4: Comparison between stable and unstable AQM throughput

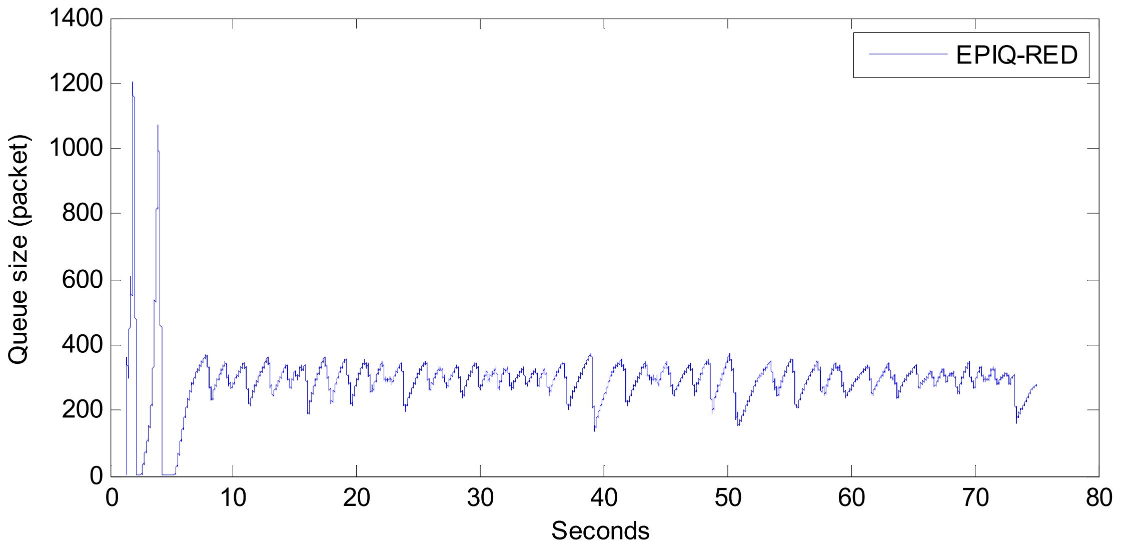


Figure 3.5: Instantaneous queue size of example 1

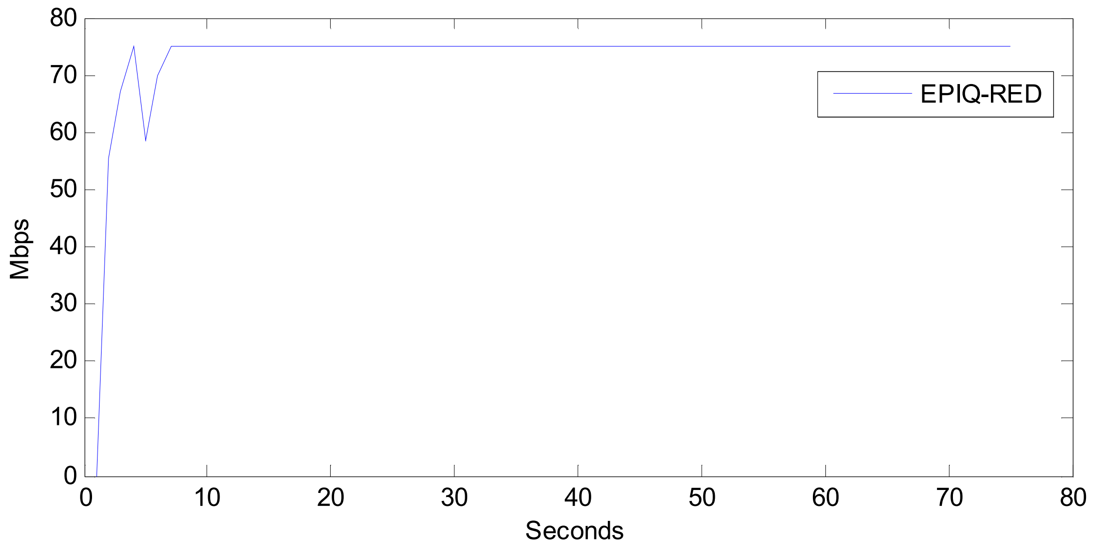


Figure 3.6: Channel utilization of example 1

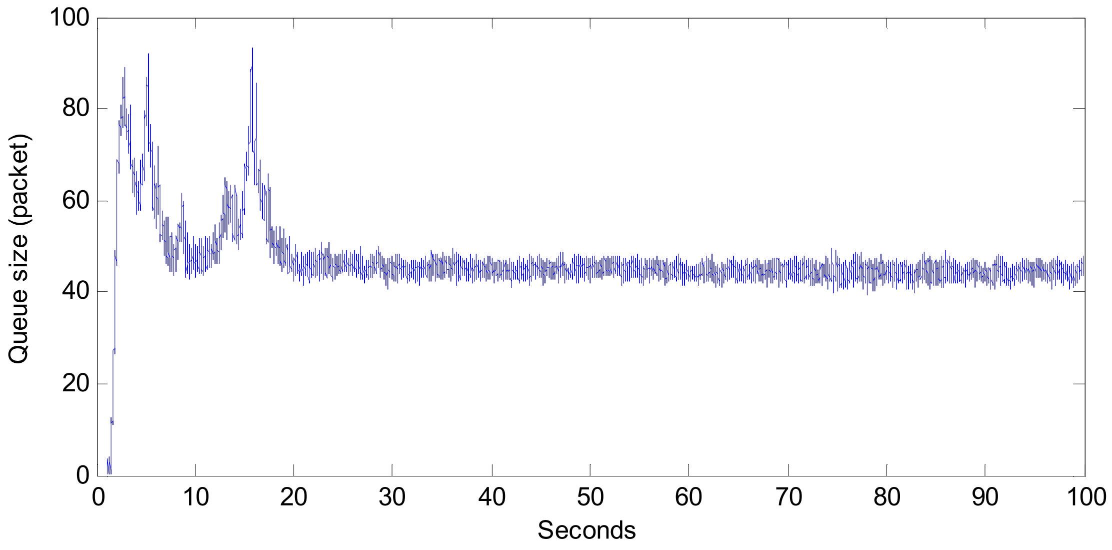


Figure 3.7: Instantaneous queue  size of example 2

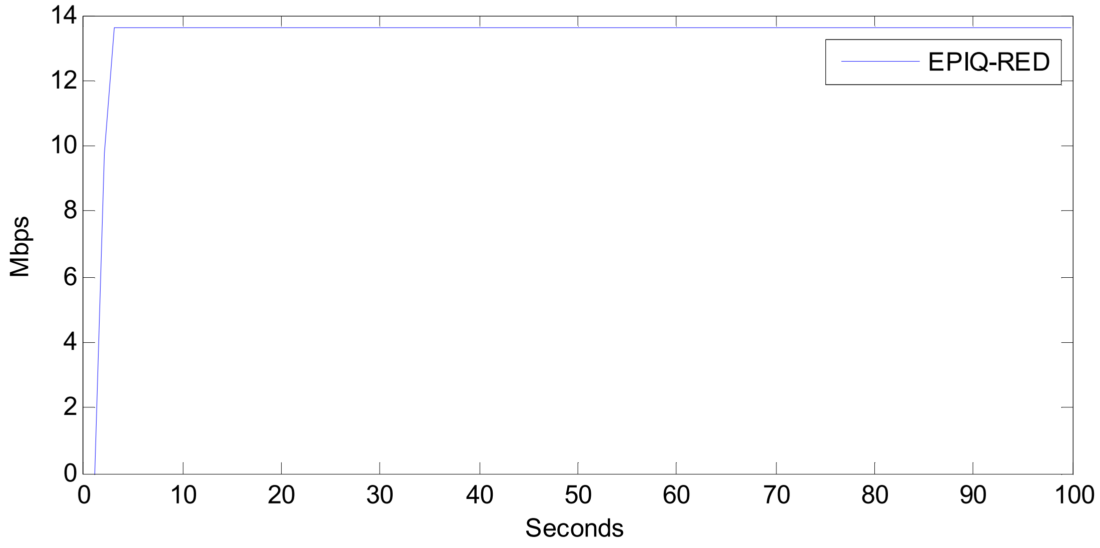


Figure 3.8: Channel utilization of example 2

### 3.3.2 EPIQ-TFRC

The main role of the rate control algorithm is to provide maximized data rate while simultaneously sharing network resources fairly with other flows. In this section, we will test our proposed EPIQ-TFRC and compare our proposed EWMA with ALI algorithm. Fairness is an important attribute of the rate control algorithm. To test the fairness of EPIQ-TFRC, the test network consists of three video streaming users, three TCP users and a bottleneck link of 15 Mbps. We tested EPIQ-TFRC with both EWMA and ALI. We tested EWMA with three history values of 0.5, 0.7 and 0.9. Table 3.2 shows the comparison of EWMA to ALI.

The algorithms were tested on fixed (F) and varied (V) bottleneck link bandwidth. The standard deviation of video flows represents the fairness between video flows, while the video share ratio represents the fairness between video flows and TCP flows. It is clear that ALI has the best performance in a fixed network. However, on a network with varying bandwidth, ALI has the worst fairness. On the other hand, we can see that EWMA 0.7 has optimum performance in a fixed and varied bottleneck network. Fig. 3.9 shows the performance of EWMA and ALI on a varied bandwidth bottleneck. It is clear that EWMA 0.9 has the best performance. However, it is less fair to other flows in the network. From the Table 3.2, we consider using EWMA 0.7 as the optimum method to compute packet loss average $p$.

Table 3.2: EPIQ-TFRC event loss calculation methods comparison

| **Loss Calculation Method** | **Intra-fairness STD (%)-F** | **Intra-fairness STD (%)-V** | **EIPQ share ratio (%)- F** | **EIPQ share ratio (%)- V** |
|---|---|---|---|---|
| **EWMA 0.5** | 6.3 | 12 | 49.9 | 47.7 |
| **EWMA 0.7** | 3.5 | 4.6 | 49.8 | 49.6 |
| **EWMA 0.9** | 9.2 | 1.15 | 52.7 | 49 |
| **ALI** | 1.15 | 3.7 | 49.6 | 44.1 |

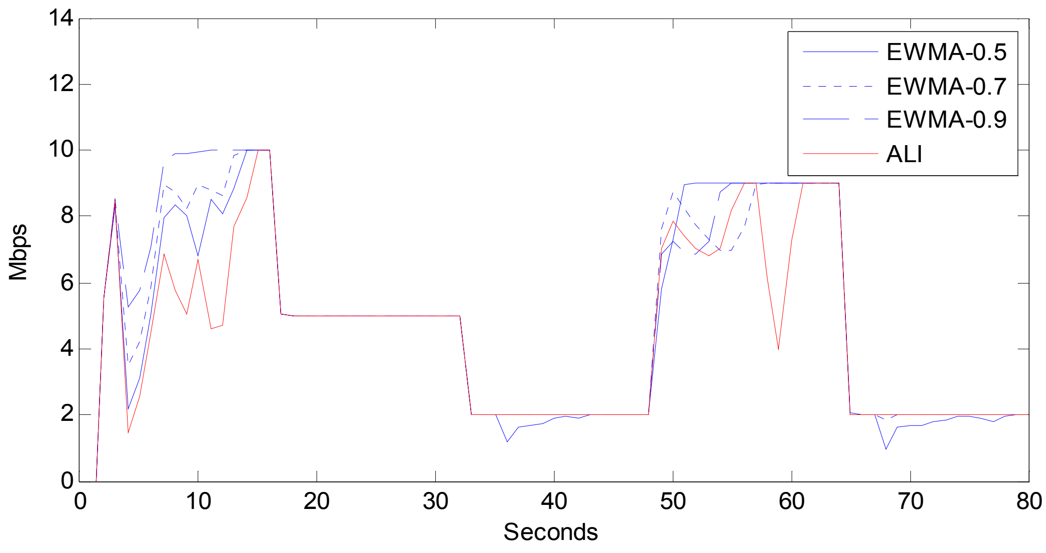


Figure 3.9: Comparison between ALI and EWMA throughput

# Chapter 4 EPIQ Applications

In this chapter we discuss the application of EPIQ system. EPIQ framework can applied on both Video On Demand (VOD) and delay sensitive live streaming.

## 4.1 Video On Demand (VOD) Experimental Evaluation

Most of today's video traffic is generated by VOD websites like Netflix and YouTube. Also, most VOD providers use DASH technology as their streaming framework because of its simplicity and economic feasibility. In this section we will compare our proposed framework, EPIQ, with DiffServ and DASH.

### 4.1.1 Experimental Setup

We used the ns-2 simulator to implement the EPIQ framework. As shown in Fig. 4.1, we employed the widely used bar-bell topology with multiple sources connected to a single bottleneck link with 800 packet queue size. In our experiments we used two bottleneck link bandwidth scenarios. In the first scenario, the capacity of the bottleneck link was set to 200 Mbps, which we refer to as Fixed (F) bottleneck, while in the second scenario, the bottleneck link bandwidth varied between 200Mbps and 60Mbps, thus we denote this configuration as Varied (V) bottleneck. Meanwhile the rest of the links were fixed to 350 Mbps.

In our simulations, the traffic is generated by two equal member application groups where each group has 10 sources (users). The first group is setup as FTP/TCP traffic generator while video streaming applications are setup as the second traffic group. For the video content, we used JSVM [56] and OpenSVCDecoder [58] to both generate trace files for our simulations and to evaluate the received video quality. Eight High Definition (HD) video sequences[64] , with total video duration of 80 seconds, were encoded as a single video streaming sequence in our simulation. The video was encoded as AVC stream at 30 FPS.  Four versions of the video were generated to simulate different video quality profiles. The video supports the following quality profiles: 2, 3.5, 5.1, and 10 Mbps. To evaluate the performance of our system, we compare it with commonly used video streaming methods. In the following subsection, we discuss the configuration related to each system.

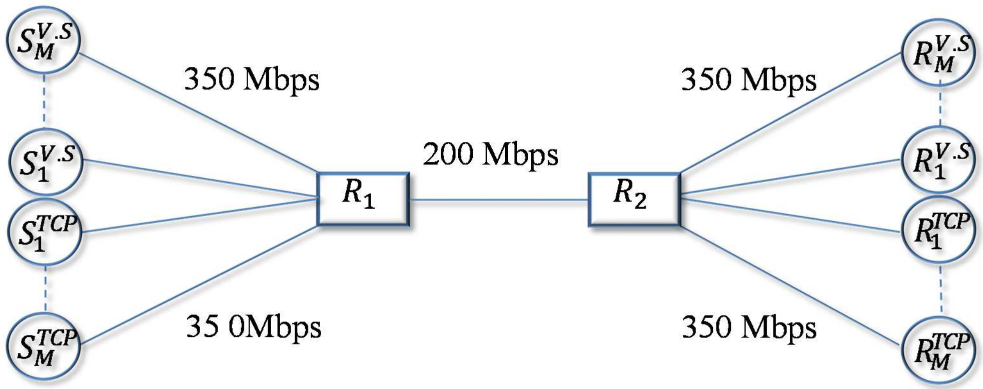


Figure 4.1: Simulation topology

A) EPIQ

Our proposed EPIQ video streaming system was configured according to Chapter 3. In our simulations, each AVC GoP represents one EPIQ video set (EVS), and it was carried by one EPIQ packet set (EPS). EPIQ-RED and EPIQ-TFRC were preconfigured to the ns-2 default parameters.

B) Multi Priority Queue (MPQ)

Multi-based video streaming is the prominent traditional solution to benefit from the priority structure of video. Our multi-queue comparison model is based on DiffServ. We implemented a simple multi-queue similar to the one found in[26]. The AQM for DiffServ employs two sets of queues. The first set of queues is for streaming data and the second set, which consists of a single queue, is used for FTP data. The multimedia queue set is divided into three virtual queues (Green, Yellow and Red) with each having different dropping precedence. The most important video traffic is queued in the Green queue while the least important is queued in the RED queue. At the sender, the AVC video stream was divided into three different priority streams. We used a simple packet marking method where the packets are marked based on frame type (i.e. I, P and B frames). In the mentioned method, I frames packets have the highest priority and B frames packets have the lowest priority. Also, to control the video sending rate we used standard TCP Friendly Congestion Control (TFRC) protocol[21]. Similar to the EPIQ system, the AVC quality profile was selected in a similar manner. The system used the default ns-2 TFRC and DiffServ parameters.

C) HTTP Steaming

Nowadays, one of the most widely used video streaming algorithm is Dynamic Adaptive Streaming over HTTP (DASH). DASH video traffic is streamed over best effort Internet. Hence, in our simulations, Drop Tail was used as the AQM algorithm in the bottleneck network. At the sender, The H.264 AVC video Quality profiles were used as DASH supported bitrates similar to EPIQ and DiffServ systems. In standard DASH, data are transported using TCP, which provides bandwidth estimate to the application layer to adjust the sending rate accordingly. We used the DASH standard method to control the selected video bitrate.

### 4.1.2 Experimental Results

In this section we present the results of our experiment.

A) AQM Stability

Our EPIQ-RED algorithm shares the core principles of RED. According to Eq. (3.3)-(3.4), $min_{th}$ and $max_{th}$ are set automatically to 79 and 412 packets, respectively, while the average queue size is 120 packets. We found that our algorithm is relatively stable in comparison to what is found in [34] and consistent with RED performance that we discussed in previous section. Table 4.1 shows AQM performance of the three systems.

Table 4.1: AQM performance of EPIQ, DiffServ and DASH

| | **Avg. Thro. (Mbps)** | **Ch. Utilization** | **Delay (ms)** | **Delay Jitter (ms)** |
|---|---|---|---|---|
| **EPIQ-F** | 196.1 | 98 | 18 | 2.8 |
| **DiffServ-F** | 195.6 | 97.5 | 17 | 6.3 |
| **DASH-F** | 186.8 | 93.4 | 18 | 4.5 |
| **EPIQ-V** | 117.2 | 99.3 | 20.5 | 4.5 |
| **DiffServ-V** | 116.8 | 98.5 | 21.6 | 3.4 |
| **DASH-V** | 113.9 | 96.5 | 43 | 8.3 |

It is clear from Table 4.1 that EPIQ and DiffServ performances are close in all aspects. However, DASH didn’t achieve the full channel utilization as well as it had the highest end to end delay compared to EPIQ and DiffServ. Drop-tail buffers are widely known for not achieving full channel utilization.

B) Congestion Control

Congestion control is necessary to achieve smooth video streaming while maintaining fairness with other flows in the network. For a completed 80 seconds simulation, the average share ratios are show in table 4.2. It is clear that DiffServ-TFRC and EPIQ-TFRC connections provide a higher level of fairness toward FTP/TCP than DASH and DiffServ. It is important to note that despite the fact that we introduce changes to the original TFRC work, we find our system operates within original TFRC boundaries.

Table 4.2: Network resources sharing fairness of EPIQ, DiffServ and DASH

| | **Video Fairness STD (%)-F** | **Video Fairness STD(%)-V** | **Video share ratio (%)- F** | **Video share ratio (%)- V** |
|---|---|---|---|---|
| **EPIQ** | 3.7 | 4.6 | 100.6 | 99.2 |
| **DiffServ** | 32 | 36 | 104.2 | 102.4 |
| **DASH** | 44 | 17 | 80.4 | 82.5 |

It is clear from Table 4.2 that DiffServ has the highest share ratio where 100% mean FTP and Video sessions share bottleneck bandwidth equally. However, DiffServ video sessions have high STD network share values compared to EPIQ, indicating that the difference in data rate between DiffServ video sessions flows are high. The difference in video means some users experience very high quality video streaming while other users experience low quality. On the other hand, DASH suffers from low network resources sharing ratio in addition to high video share STD similar to DiffServ.

C) Video Quality

A key goal of our work is to provide high quality video transmission. The performance metrics used to evaluate the quality of the received video traffic include the average Peak Signal to Noise Ratio (PSNR) of the luminance (Y) component, packet retransmission, delay, average download, and video quality profile changes. Also, we compared the performance of two startup buffer sizes of 5 seconds and 15 seconds. In our simulations, retransmissions were allowed only for I frame packets of the AVC video for both EPIQ and DiffServ. Meanwhile, the retransmission function was allowed for all DASH packets, which include both control packets and video streaming packets because of the necessity of all packets to DASH operation.

Table 4.3: Transmission and video quality of EPIQ, DiffServ and DASH

| | App. Delay (msec) | Packet Retransmit (%) | Avg. Download (Mbps) | Quality Change | | PSNR (dB) | |
|---|---|---|---|---|---|---|---|
| | | | | **5 s.** | **15 s.** | **5 s.** | **15 s.** |
| **EPIQ-F** | 19.2 | 0 | 9.88 | 2.2 | 1.8 | 37.5 | 38.0 |
| **DiffServ-F** | 17.5 | 0.02 | 10.08 | 1.8 | 1.6 | 34.4 | 34.8 |
| **DASH-F** | 393 | 0.163 | 8.75 | 3.2 | 2.5 | 33.6 | 34.0 |
| **EPIQ-V** | 24.2 | 0 | 5.8 | 3.5 | 3.4 | 35.5 | 36.2 |
| **DiffServ-V** | 22.6 | 0.17 | 5.92 | 3.1 | 2.56 | 32.5 | 33.4 |
| **DASH-V** | 403 | 0.48 | 5.0 | 4.5 | 4.8 | 32.5 | 34.5 |

We tested our proposed solution and computed the average results for the performance metrics that we stated earlier. All results represent the average per user over 80 seconds, where this average is evaluated over 10 video sessions corresponding to the 10 users in our simulations.

Important performance parameters include application to application data delay. Table 4.3 shows the average delay parameters in both fixed and varied bottleneck bandwidth of the three mechanisms as seen by user application layer. EPIQ delay is slightly higher than DiffServ because the EPIQ receiver forwards the received data to the application after receiving all EPS's packets. DASH on the other hand, has very high delay because it relies on TCP to deliver data. Both the packet retransmission process and the TCP buffer contribute to the delay. Fig. 4.2 shows a comparison between EPIQ, DiffServ and DASH application to application Delay.

Regarding retransmission, EPIQ did not retransmit any packet while DiffServ retransmitted 0.02% and 0.17% of the total number of packets transmitted in fixed and varied bottleneck bandwidth respectively. Because DASH uses HTTP/TCP, it retransmitted 0.16% and 0.48% of the total packets in both fixed and varied bottleneck respectively. Also, as shown in Table 4.3, DiffServ has the highest download rate compared to EPIQ and DASH, because it has the highest network share ratio as shown in Table 4.2.

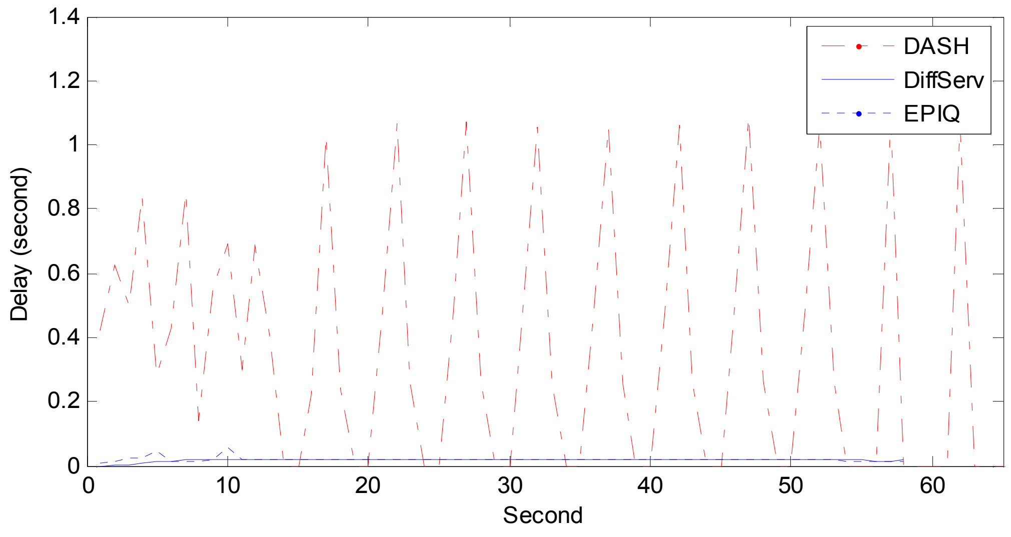


Figure 4.2: Application to application delay of EPIQ, DiffServ and DASH

Also shown in Table 4.3, the performance of the different video transmission mechanisms from the user prospective with startup buffering of 5 seconds and 15 seconds. It is crucial to highlight that EPIQ achieves on average 4.0 dB and 2.3 dB in PSNR improvements over DASH in fixed and vary bottleneck respectively. Furthermore, EPIQ outperforms DiffServ by 3.15 dB and 2.3 dB in fixed and varied bottleneck respectively. Although the average DiffServ video session had the highest average download rate compared to both EPIQ and DASH video users' sessions, DiffServ had the lowest average PSNR. The main reason is that DiffServ users had high video share STD as we explained earlier which causes some sessions to have poor PSNR. Regarding video quality-profile changes (which correspond to bitrate changes), DASH suffers from around 30% more quality changes when compared to EPIQ and DiffServ while DiffServ and EPIQ have close profile changes. Note that neither EPIQ, DiffServ nor DASH suffer from any stalling. It should be clear from these results the importance of quality/bitrate adaptation, such bitrate changes are necessary to avoid video stalling. (During the stalling time, the video playback stops for buffering.)

# Chapter 5 Vehicle to Vehicle Safety System

## 5.1 Cooperative Advanced Driver Assistance System (C-ADAS) Overview

The development of connected-vehicle technology, which includes vehicle-vehicle and vehicle-infrastructure communications, opens the door for unprecedented active safety and driver-enhanced systems. In addition to exchanging basic traffic messages among vehicles for safety applications, a significantly higher level of safety can be achieved when vehicles and designated infrastructure-locations share their sensor data. For example, while cameras installed in one vehicle can provide visual information for mitigating many avoidable accidents, we envision a whole new level of improved safety paradigm where visual data captured by multiple vehicles are shared and fused for significantly more optimized active safety and driver assistance systems. Under this example, we envision a new system where cameras installed on multiple vehicles and infrastructure-locations share and fuse their visual data and detected objects in real-time. The transmission of camera data and/or detected objects (e.g., pedestrians, vehicles, cyclists, etc.) can be accomplished by many communication methods. In particular, such communications can be accomplished using the emerging Dedicated Short-Range Communications (DSRC) technology. The vehicle receiving the visual data from an adjacent vehicle, for example, can fuse the received visual data with its own camera views to create a much richer visual scene. The sharing of visual data is motivated by the fact that some critical visual views captured by one vehicle or by an infrastructure-location are not visible or captured by many other vehicles in the same environment. Sharing such data in real-time provides an invaluable new level of awareness that can significantly enhance a driver-assistance, connected vehicle, and/or autonomous vehicle's safety-system. An example of such scenario is illustrated in the Fig. 5.1, where the view captured by the “U-Haul” van is fused by an adjacent-vehicle camera view. This enables the adjacent vehicle’s active-safety system and/or driver to clearly recognize that there is a pedestrian crossing; without such fusion the vehicle’s system/driver would be completely oblivious to the presence of the pedestrian. It is worth noting that the proposed new framework can be used by new driver-assistance systems, and it can be employed by future connected/autonomous vehicles.

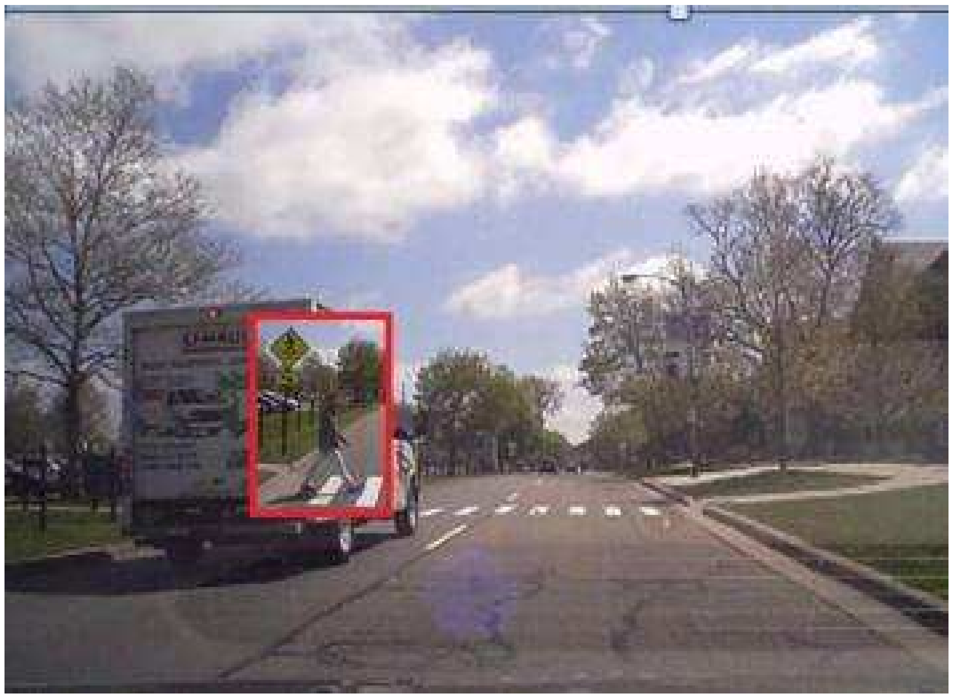


Figure 5.1: Pedestrian collision scenario

## 5.2 C-ADAS Architecture

Fig. 5.2 illustrates the architecture of our concept which is applicable to both autonomous and human driven vehicles. The system can be generalized to include any form of the following three sensor types:

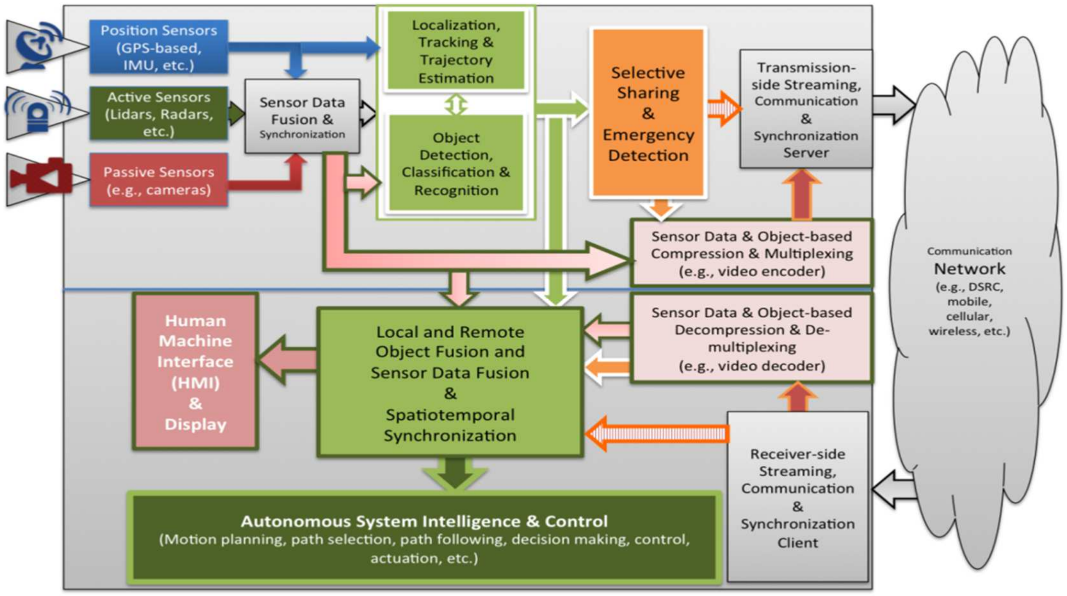
1. Position Sensors: This includes any GPS-based and IMU sensors.
2. Active Sensors: This includes radars, lidars, and ultrasonic sensors.
3. Passive Sensors: This includes vision cameras and multi-spectral imaging sensors.

An integrated two subsystems architecture is shown in Figure Z. The architecture includes the following two subsystems:

A) A Capture-and-Transmit (CAT) subsystem that includes all sensors devices , sensor fusion, object detection, classification, tracking, trajectory estimation, ego motion estimation and other objects' absolute and relative position estimation, object selection for sharing with remote vehicles or the infrastructure, and compression, multiplexing, communication and transportation functions and modules.

B) A Receive-and-Fuse (RAF) subsystem that receives objects' position and description data captured by other remote entities (e.g., other vehicles and/or infrastructure entities) and fuse this data with locally detected objects to create a complete view of the 3D environment surrounding the vehicle.

The final fused data and objects can be used by an autonomous system for path planning, selection. The fused data and objects can also be used by an HMI system and can be displayed for human use (e.g., a driver in a manually-driven vehicle or for occupants in an autonomous vehicle).


Position Sensors (GPS-based, IMU, etc.)
Active Sensors (Lidars, Radars, etc.)
Passive Sensors (e.g., cameras)
Sensor Data Fusion & Synchronization
Localization, Tracking & Trajectory Estimation
Object Detection, Classification & Recognition
Selective Sharing & Emergency Detection
Transmission-side Streaming, Communication & Synchronization Server
Sensor Data & Object-based Compression & Multiplexing (e.g., video encoder)
Communication Network (e.g., DSRC, mobile, cellular, wireless, etc.)
Human Machine Interface (HMI) & Display
Local and Remote Object Fusion and Sensor Data Fusion & Spatiotemporal Synchronization
Sensor Data & Object-based Decompression & De-multiplexing (e.g., video decoder)
Receiver-side Streaming, Communication & Synchronization Client
Autonomous System Intelligence & Control
(Motion planning, path selection, path following, decision making, control, actuation, etc.)
Figure 5.2: An integrated system for multi-modal sensor data sharing and fusion

In Fig. 5.3, we describe the baseline architecture for the proposed system using a camera sensor as an example. However, the system can employ any sensor modality including lidars, radars, ultrasonic sensors, etc. A more powerful system can be realized by the fusion of the multimodal-sensor system. For example, one possible realization is a system that employs any combination of camera, lidar, radar, and/or ultrasonic sensors.

Within each vehicle, the proposed system will include a video capturing unit (i.e., camera(s) and lenses) and object detection/recognition processor (e.g., for pedestrians, cyclists, other vehicles etc.) in addition to a complete chain of compression/decompression, streaming, and fusion units. Fig. 5.3 illustrates a simplified architecture that includes: (a) the transmission section within the vehicle transmitting the video and detected objects; and (b) the receiver section within the vehicle that is receiving the video and objects. (Both vehicles will also include the other receiver/transmitter sections, which are not shown for simplicity.) The video cameras and object detection/recognition can be based on any state-of-the-art cameras or visual systems that are capable of object detection, classification, and/or recognition.

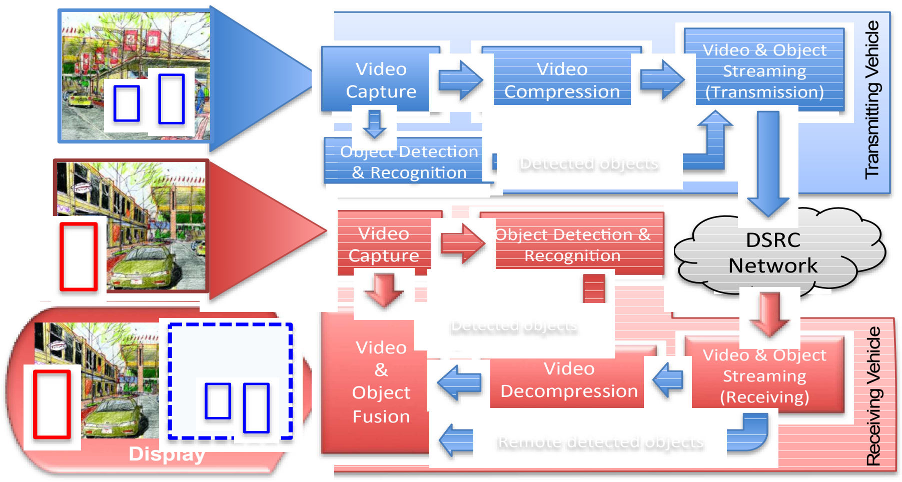


Figure 5.3: Example of baseline system architecture

In cases of sensor modalities that generate a large amount of data, the need for data compression could become necessary. Hence, in the case of using visual sensors, video compression/decompression will be critical for achieving efficient communication among the vehicles and/or infrastructure. Any state-of-the-art video coding standards or technologies that are either standalone or built-in within popular cameras can be used. The video-streaming framework can be based on advanced solutions or popular standards such as the Real-Time Streaming Protocol (RTSP). The wireless network can be based on underlying DSRC transceivers that adhere to the Intelligent Transportation System of America (ITSA) and 802.11p WAVE standards, and which are certified by the US DOT. Other communication mechanisms can also be used, including traditional wireless and cellular services such as LTE, 5G, or other mechanisms.

The system is applicable for single-lane, double-lane, and any multi-lane roadways and highways in urban or rural environments. Furthermore, the system includes intelligence that utilizes localization and mapping algorithms in conjunction with real-time messages to enable a vehicle or an infrastructure-location to determine if its video and/or detected objects can be utilized by other nearby vehicles. For example, in a three-lane road, vehicles driving in the middle-lane can broadcast their video; this will enable vehicles driving on both the right and left lanes to receive the middle-lane vehicle's videos and fuse them with their own visuals.

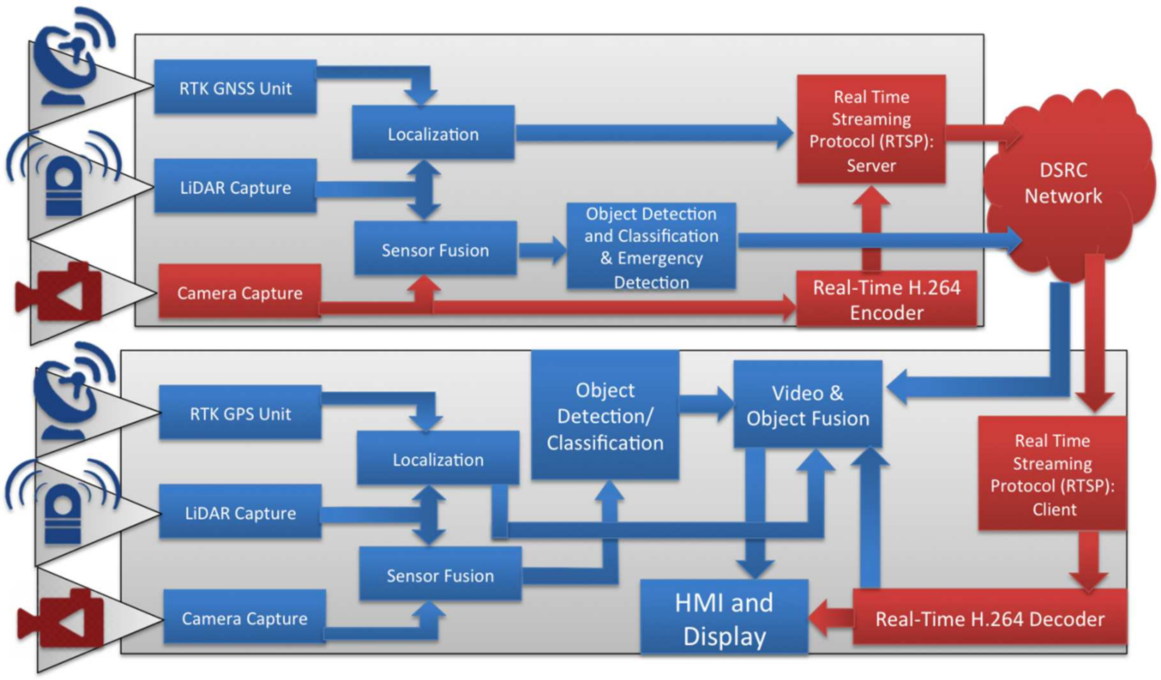


Figure 5.4: System architecture with object localization, positioning, and trajectory estimation

Fig. 5.4 also highlights the integration of key modules of the baseline system (red) with other modules (blue). Examples of particular solutions for the baseline system models are shown in the Fig 5.3 as well. For instance, video compression is shown based on the popular H.264 standard. However, any other video compression (or even uncompressed video stream over high-bandwidth communication links) can be used. The overall system includes the following modules and functions:

1 Object detection and classification: There are many options under this area. Here, we list three examples. First, deep learning approaches that use visuals (only) or fused sensor data (e.g., visuals-plus-lidar signals) can be employed. Due to the high-complexity of

deep learning, a second option is to employ traditional computer vision object detection and classification methods that can operate fast and reliably in real-time. A third option would be to employ a sensor platform (most notably camera) that supports built-in object detection such as a Mobileye vision system.

2 Selective object sharing and emergency detection: The system has to be selective towards which objects to share with other vehicle(s). In principle, there is no point in sharing objects that can be observed and detected by other vehicles. This decision can be determined through intelligent 2D/3D localization and mapping algorithms of the 2D/3D scene surrounding the vehicles. The system could identify emergency situations when the vehicle detects objects that can't be observed by other vehicles, and hence it declares an emergency that triggers the rapid transmission of the detected occluded objects.

3 Integrated fusion and localization: As mentioned above, the system has to perform highly accurate fusion and localization of multiple objects, both captured remotely and locally. In addition to cameras, lidar and GPS-based localization can play a key role in this integrated fusion-localization framework.

   a. RTK GNSS**:** RTK GNSS is a high accuracy system that provides up to 1 cm localization. It allows a very accurate estimation of object distances and visual fusion. High end GNSS fuses IMU (Inertial Measurement Unit) measurements with GNSS signal to improve location accuracy.

   b. Lidar and Radar: A 360 degrees Lidar can measure the distance between the car and various objects to build a 3D model of the vehicle surroundings. This technology can achieve around 10 cm accuracy and is very useful when the GNSS service is not available. Radar provides robust sensing under a wide range of weather conditions. It is also an excellent modality for estimation of range and object motion trajectory.

c. Fusion and Localization: GNSS service might not always be available or reliable. lidar-based localization can also suffer from degradation under many scenarios. By analyzing the surrounding environment and positioning accuracy, an *Integrated Fusion-Localization* paradigm fuses GNSS and lidar (and possibly other sensors) data and computes the most accurate positions. Fusing IMU measurements with both lidar and GNSS data also contributes to overall system reliability and stability. Lidar can also play a role in a fused camera-lidar learning based on object detection and classification.

4 Human-Machine Interface (HMI) and cockpit integration: For vehicles operated by human drivers, the final presentation of the fused visual data is another crucial component. This can be achieved by using visual and audible warnings, and a variety of display devices such as Head-Up Display (HUD) or dashboard display..

## 5.3 C-ADAS Design and Operation

In order to share information between vehicles, a connection has to be setup between the connected vehicles. By default, DSRC equipment periodically sends Basic Safety Messages (BSM) [55].

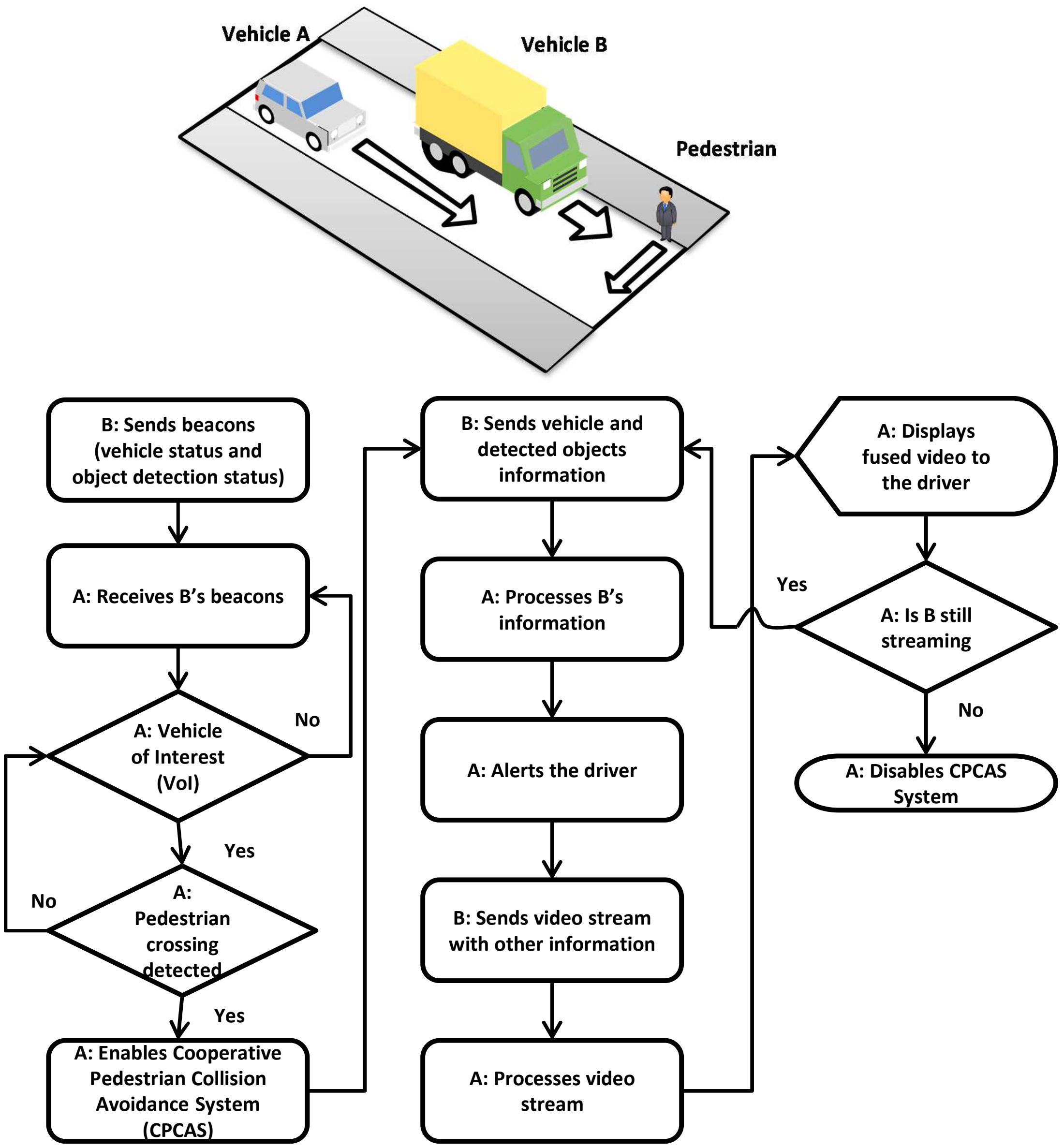

Vehicle A
Vehicle B
Pedestrian
B: Sends beacons (vehicle status and object detection status)
A: Receives B's beacons
A: Vehicle of Interest (VoI)
No
Yes
A: Pedestrian crossing detected
No
Yes
A: Enables Cooperative Pedestrian Collision Avoidance System (CPCAS)
B: Sends vehicle and detected objects information
A: Processes B's information
A: Alerts the driver
B: Sends video stream with other information
A: Processes video stream
A: Displays fused video to the driver
Yes
A: Is B still streaming
No
A: Disables CPCAS System

Figure 5.5: Proposed system (Cooperative Pedestrian Collision Avoidance System (CPCAS) ) operation flowchart

The messages contain vehicle status and applications information. In our system, we are assuming that all vehicles have Pedestrian Collision Avoidance System (PCAS) also known as Pedestrian Automatic Emergency Braking (PAEB) system, which is available in many new vehicles [65] nowadays. The PCAS system can detect crossing pedestrians and apply breaks when it detects imminent collision. We call our system Cooperative Pedestrian Collision Avoidance System (CPCAS).

Table 5.1: Variables used in our proposed system calculations

| Variable | Description |
|---|---|
| $A$ | Vehicle A |
| $B$ | Vehicle B |
| $C$ | Pedestrian |
| $D$ | Expected collision point |
| $w$ | Vehicle width |
| $d$ | Vertical distance between vehicle A and B (similar to $\Delta Z$ ) |
| $l$ | Distance between vehicle B and pedestrian |
| $e$ | Horizontal distance between vehicle A and B (similar to $\Delta X$ ) |
| $r$ | Horizontal distance between pedestrian and vehicle B |
| $\alpha$ | Angle between vehicle A and pedestrian |
| $\beta$ | Angle between vehicle B and pedestrian |
| $n$ | Euclidian distance between vehicle A and pedestrian |
| $k$ | Euclidian distance between vehicle B and pedestrian |
| $\Delta Y$ | Difference between camera A and camera B altitude |

Fig. 5.5 shows our system operation flowchart and a corresponding sketch illustrating a typical scenario involving two vehicles (Vehicle A and Vehicle B) and a pedestrian. Fig. 5.6 shows a top view of Fig. 5.5 with the variables that we use in our calculations. In addition, Table

5.1 defines the variables that are used in our system parameter calculations. The reported locations could be measured in any distance units. For example, they could be in meters as used in the Universal Transverse Mercator (UTM) coordinate format. Also, we consider the camera location as our vehicle reference location. If more than one pedestrian is detected, we perform the same calculations for each pedestrian. Meanwhile, it is possible to combine two pedestrians, who are adjacent or in close proximity, as one pedestrian. Here, and for illustrative purposes only, we focus on a single pedestrian crossing. Each vehicle has a Vehicle of Interest (VoI) list that includes all vehicles that may share useful information to the ego-vehicle. For our proposed system, the criteria includes: (a) the other vehicle transmitting the visual information should be heading in the same direction, and it is already ahead of the receiving ego-vehicle (b) the vehicle with clear view of the pedestrian should have a distance $l$ that is within a certain threshold from the pedestrian; for example, in our experiments the distance between the transmitting vehicle and the pedestrian has to be less than 50 meters (i.e., $l < 50$) (c) no more than two lane horizontal distance $e$ . When vehicle **B** detects a pedestrian crossing, it estimates the pedestrian distance $l$ as follows;

$$l = f_y \frac{R_h}{I_h} \tag{5.1}$$

Where $f_y$ is the focal length and $R_h$ and $I_h$ are the real pedestrian height in meter and height in image pixels, respectively. When one of vehicle **A**'s Vehicle-of-Interest (VoI) lists sends a BSM that contains information regarding a possible pedestrian detection, the egovehicle A starts the driver alert processing and requests more information about the detected pedestrian. Once the requested information is shared, vehicle A estimates both Distance to Collision (DTC) and

Time to Collision (TTC) regarding point D, which represents the expected position of the pedestrian where a potential collision may occur, as shown in Fig. 5.6.

$$DTC = l + d \tag{5.2}$$

$$TTC = \frac{DTC}{S_A} \tag{5.3}$$

Where $S_A$ is the speed of vehicle **A** (e.g., in meters per second). Our goal is to warn the driver five seconds before collision. After vehicle **B** receives a request for video streaming, vehicle **B** shares only the Region of the image containing the detected pedestrian, also called Region of Interest (RoI). Before sending the RoI to vehicle **A,** the RoI is compressed into a video stream. When the vehicle receives the first image of the video stream, it has to determine if it is within the local camera Horizontal Field Of Viewing (HFOV). Hence, we calculate angle $\angle\alpha$ as follows,

$$\angle\alpha = \arctan\left(\frac{r + e}{d + l}\right) \tag{5.4}$$

Where

$$r = \tan(\beta) * l \tag{5.5}$$

Note that $r$ might be negative if $\angle\beta$ is negative. The $\angle\beta$ is estimated by vehicle **B**. A simple way to estimate an object's horizontal angle is by measuring the average horizontal object pixels' locations to the camera Horizontal Field of View (HFOV) as follow;

$$\angle\beta = \frac{HFOV}{2} - \left(\frac{u}{u_{max}} * HFOV\right) \tag{5.6}$$

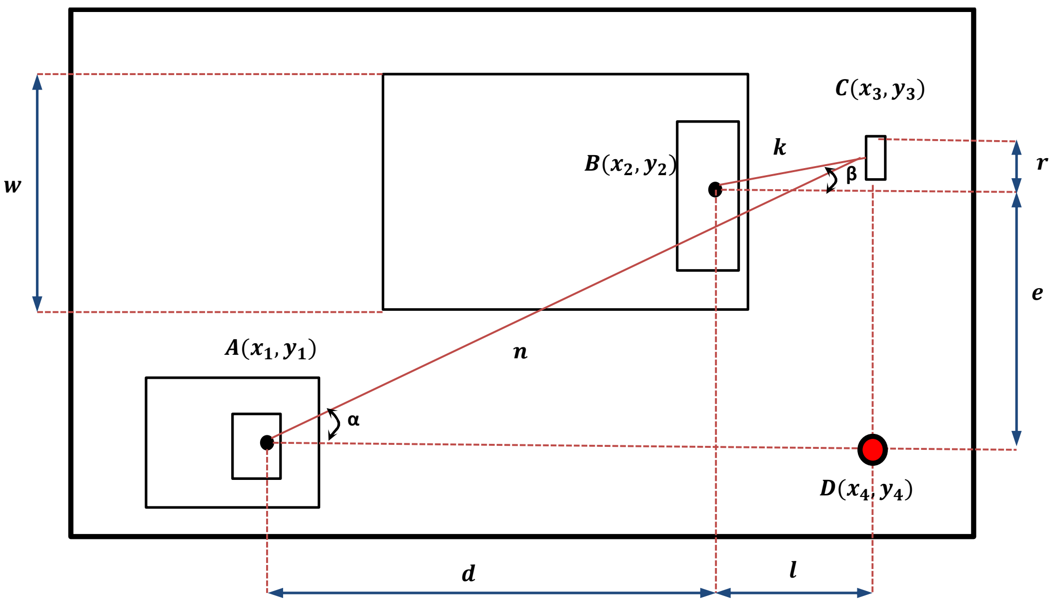


Figure 5.6: Top-view schematic of collision scenario with variables

When $\angle\beta$ is positive, the object is on the left side of the camera and vise versa. Now if $\angle\alpha$ is larger than the HOFV of vehicle **A**, only audible warning is made to the driver. Otherwise, the pedestrian image is transposed on the local video stream image. As shown in Fig. 5.7, using the camera pinhole model, we transfer the object from camera **B** image plane to camera **A** image plane as follows,

$$u_1 = \frac{f_x(X + \Delta X)}{Z + \Delta Z} \tag{5.7}$$

$$v_1 = \frac{f_y(Y + \Delta Y)}{Z + \Delta Z}$$

$\Delta X, \Delta Y\ and\ \Delta Z$ are the differences in coordinate between the two cameras' locations which are similar to variables shown in Fig.5.5 . Both variables $X\ and\ Y$ are estimated from camera **B** using:

$$X = \frac{Z * u_2}{f_x} \qquad (5.8)$$
$$Y = \frac{Z * v_2}{f_y}$$

After imposing the detected object on camera **A** image, the fused image is presented to the driver. The process is repeated until vehicle **B** stops sharing detected object information. To avoid sharing unnecessary information, vehicle **B** stops sharing detected object information when the object is no longer in front of the vehicle and visible to other vehicles (i.e. $r > \frac{w}{2}$). It is important to note that shared sensors information might be updated at a different rate. As a result, time (clock) synchronization between the two vehicles is necessary.

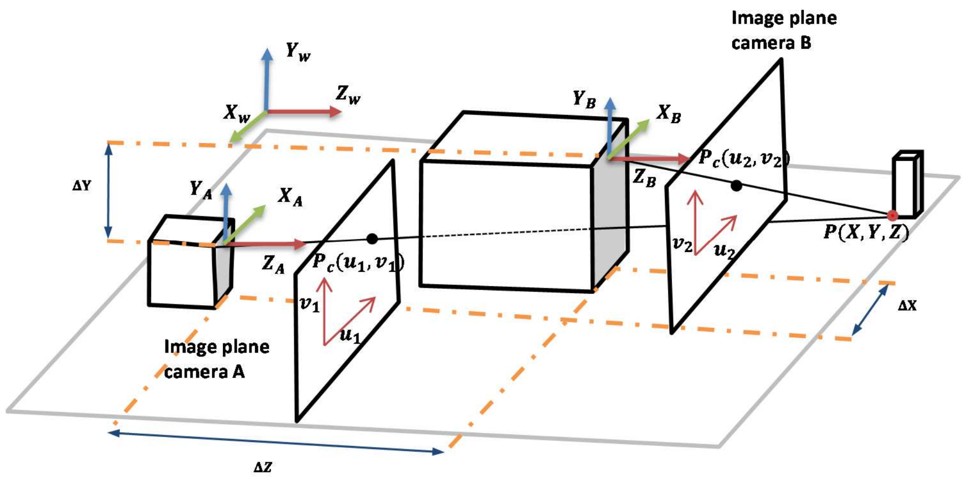


Figure 5.7: Pin hole model and image transpose calculations

## 5.4 Experimental Setup

In this section, we discuss our experiment setup. We divided our experiment into two parts. In the first part, we collected on-road data and in the second part we evaluated the performance of our system using the collected data.

### A) Data collection

Our experimental setup consists of two vehicles (SUV and Sedan). We installed a Cohda MK5 DSRC transceiver ( shown in Fig. 5.8), Global Navigation Satellite System (GNSS) and a dashboard camera (DashCam) on each vehicle. Although DSRC transceivers are equipped with GNSS, we opted to use our own separate Real-Time Kinematic (RTK) GNSS because RTK-GNSS offers a high-accuracy location estimates when compared to standalone GNSS that is used in DSRC transvers. In our experiment, we used Emlid Reach RTK GNSS receiver (shown in Fig 5.9), which is a low-cost off-the-shelf device [66]. To store the collected data, all sensors on each vehicle are connected to a laptop that has Robotic Operation System (ROS) installed on it. We connected the two vehicles' laptops via DSRC transceivers during the data collection to synchronize laptop clocks. In addition, we conducted a bandwidth test experiment between two vehicles to verify the available bandwidth and to emulate channel performance when conducting the experiment in the lab.

Figure 5.8: Cohda MK5 transceiver

The RTK-GNSS output was set to the maximum limit of 5Hz and the camera to 24 Frame Per second (FPS). The DSRC data rate channel was set to 6 Mbps. We performed our experiment on the Michigan State University campus and surrounding areas (as shown in Fig. 5.14) with wide ranging speed limits up to 55 kilometer-per-hour (kph). All of our experiments were conducted during daytime. In the first part, we collected channel bandwidth data while driving at a speed ranging between 0 and 55 kph; and the distance between the two vehicles' DSRC transceivers ranged from 5 to 100 meters. In the second part, we simulated a pedestrian pre-collision scenario which was coordinated by our test team.

**B) Lab experiment**

In our lab setup, we conducted two experiments. In the first experiment, we compared the performance of two video coding and streaming techniques. The techniques are H.264 encoding and Motion JPEG. The goal of the experiment was to determine which method is more suitable for our work. We used two ROS supported PCs that are connected to each other directly via Ethernet cable. We used wired connection instead of DSRC wireless connection to minimize the network queuing delay effect. Our goal was to measure the bandwidth and the delay of each encoding technique. We configured the H.264 encoder to use the zero latency profile to minimize the delay. However, this configuration reduced the compression efficiency and increased the streaming bandwidth. For Motion JPEG, we used the standard JPEG compression. We tried to match the PNSR of both video streaming techniques as close as possible. H.264 encoder uses dynamic compression techniques that change the image compression quality. Our most important objective was to transport the visual information with minimal delay.

Figure 5.9: Emlid GNNS receiver

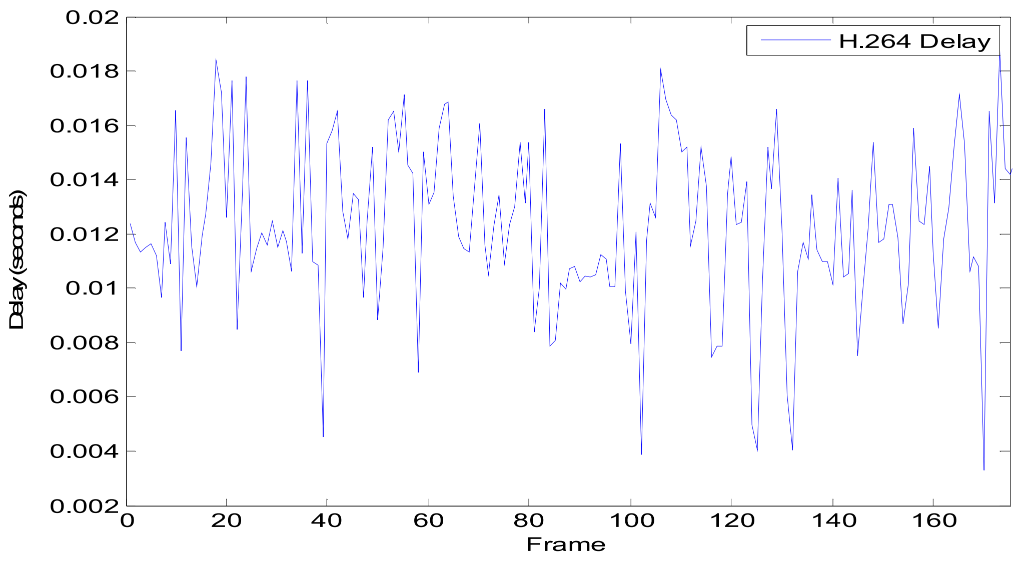


Figure 5.10: H.264 frame delay

Fig. 5.10 and Fig. 5.11 shows the average frame end to end delay for H264 and MJPEG, respectively. The H264 encoder had almost four times the delay of MJPEG. In fact, the average start up delay for H264 was 15 ms, meanwhile MJPEG had only a 4ms delay. It is worth noting that a 15 ms delay is considered acceptable for streaming. However, H264 suffers from a 560 ms startup delay which is very high compared to only 6 ms for MJPEG.

Furthermore, the biggest advantage of H.264 is bandwidth saving, hence, we compared the bandwidth used by the two techniques. A comparison between the MJPEG and H.264 bandwidth for 3 seconds of the test video is shown in Fig. 5.12.

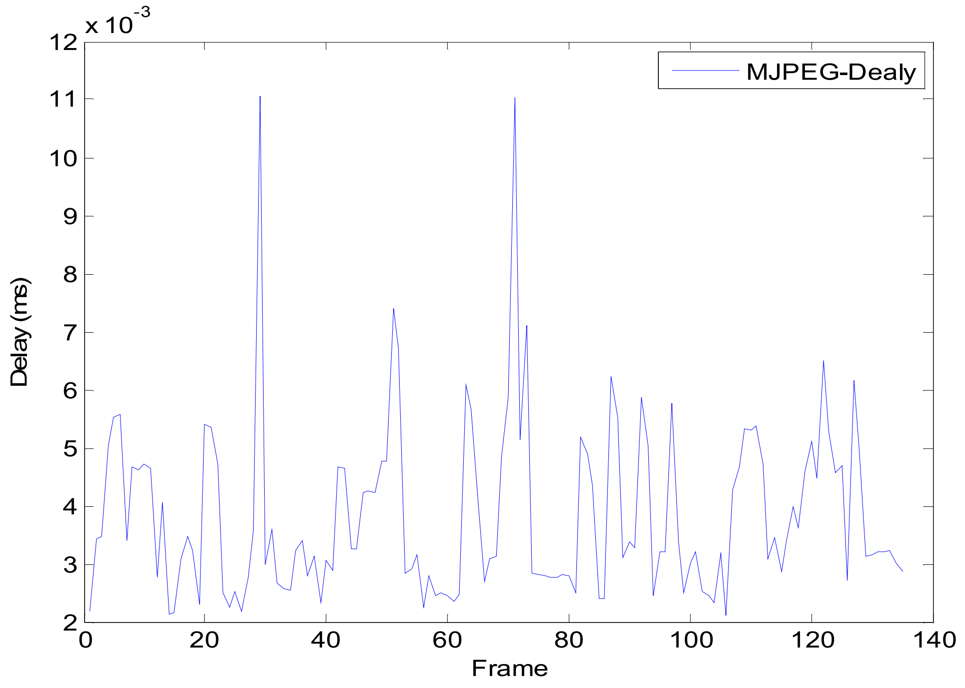


Figure 5.11: MJPEG frame delay

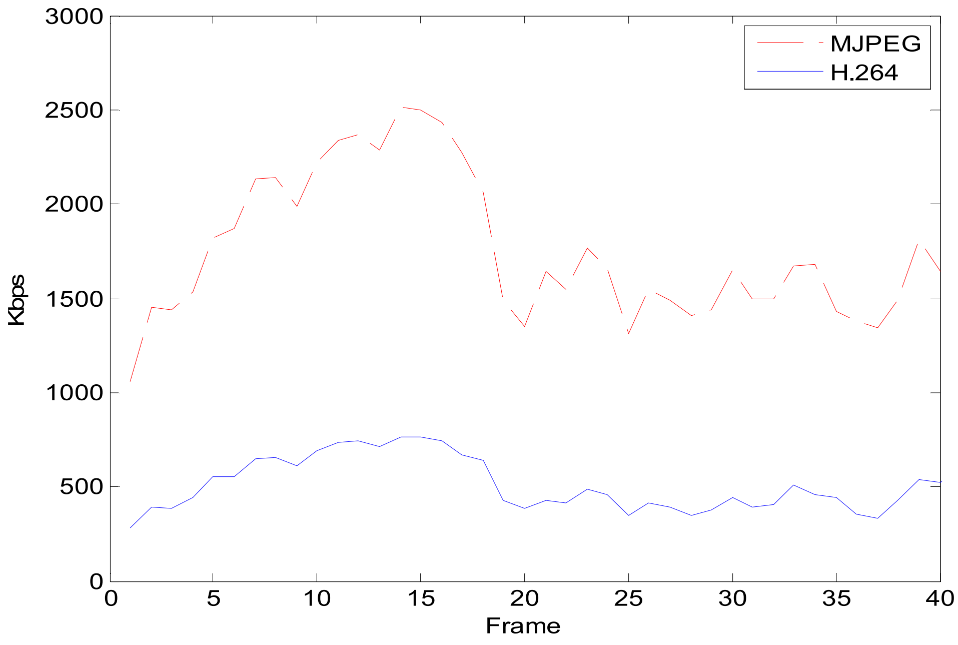


Figure 5.12: H.264 vs. MJPEG bandwidth usage

Our algorithm is made flexible by only sharing regions of interest (ROI), in other words, only portions of the image is shared. To show the effectiveness of our selective sharing in reducing the required bandwidth~~,~~ we simulated selective sharing by resizing the original frame resolution and then compared its transmission bandwidth with the original frame size~~.~~ An MJPEG bandwidth compression of a 480x640 image from 100% to 10% of the original size is shown in Fig. 5.13. At the 100% scale, the bandwidth H.264 and MJPEG are 550Kbps and 1750Kbs respectively. The MJPEG bandwidth is three times that of H.264 at 100% . However, the needed bandwidth is reduced to less than half when the image size is decreased to 10% of the original. Moreover, we reduced the file size online without shutting down the streaming process. Such online resolution change is only possible using MJPEG and not with H.264. If a change in resolution occurs, then the encoder and decoder have to reset, causing a 550 ms delay. In our application, the image resolution is always changing from frame to another. By combining selective sharing with MJPEG, we could achieve minimum delay and bandwidth with frame changing size flexibility.

In our second experiment, we used two ROS supported desktop PCs that were connected with stationary DSRC transceivers. The distance between the two transceivers was fixed to 5 meters. To emulate the moving vehicle, based on our road test findings, we added a random delay of 5 to 15 milliseconds to the channel and set the maximum channel bandwidth to 2.8Mbps. Both PCs have core I7 processors and one PC has NVIDIA GTX 1080ti GPU. The GPU capable PC represents vehicle B, while the other PC represents vehicle A. We implemented our proposed system components as ROS nodes.

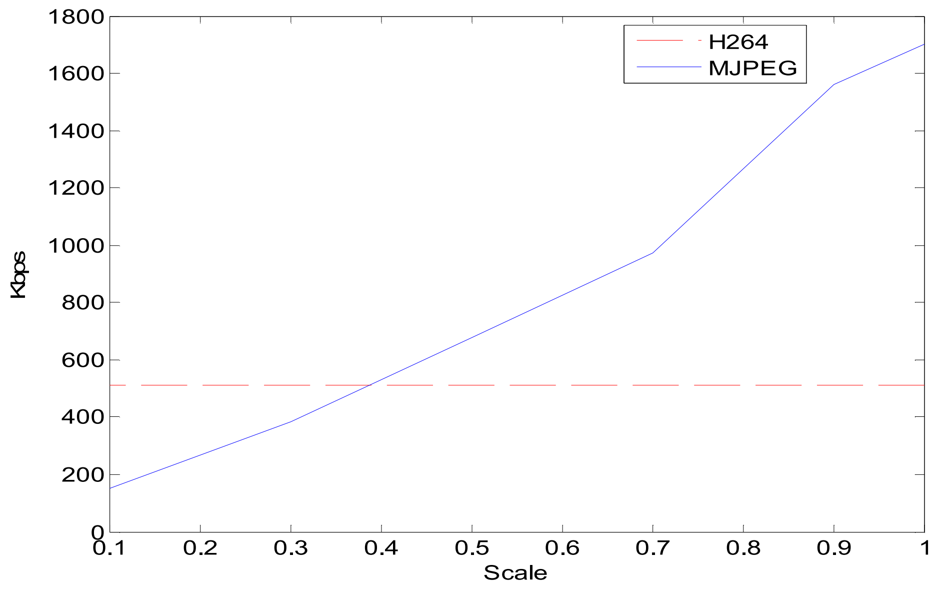


Figure 5.13: H.264 vs. MJPEG bandwidth with scaled image

We used the You Only Look Once (YOLO) object detection algorithm in our lab experiment [67]. We trained the algorithm for pedestrian detection using Visual Object Classes (VOC) dataset [68]. Also, we used Motion JPEG (MJPEG) as our video/image encoding/decoding technique. Using ROS replay feature, we replayed our pedestrian pre-collision scenario data and evaluated our system performance.

## 5.5 Experimental Results

Fig. 5.15 and Fig 5.16 show a sample of DSRC bandwidth and packet delay test results, respectively. During this sample results, the distance between the two vehicles was between 90 to 120 meters and at a speed of 55 kph. The average bandwidth and delay were 2.85 Mbps and 34.5 ms respectively. We found that DSRC equipment can carry high quality video stream with minimal delay. Similar findings are found in [52].

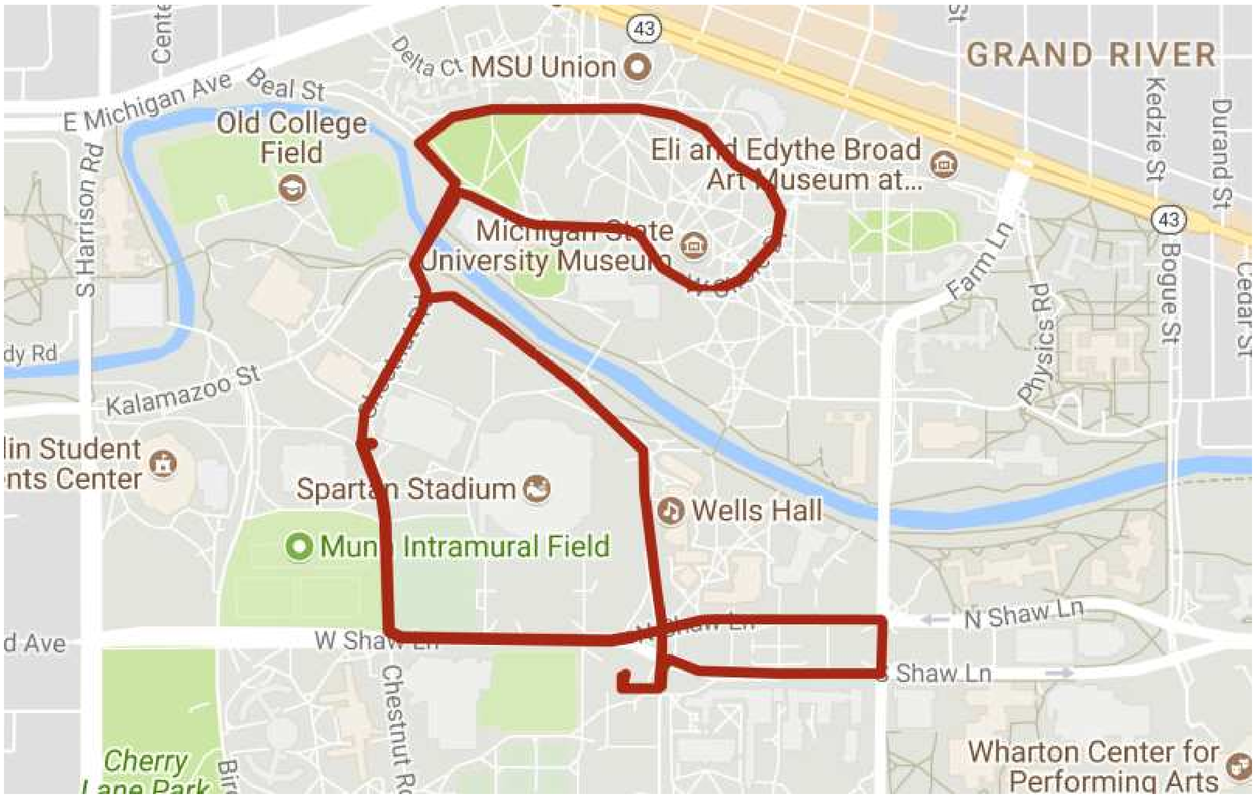


Figure 5.14: DSRCs test route

Object detection algorithm YOLO was able to process 8-10 FPS which is considered acceptable. However, it is possible to achieve higher processing using automotive oriented hardware. As we discussed in Section 5.3, after a pedestrian is detected, the pedestrian distance and angle is estimated. The Region of Interest (ROI) is extracted from the original image and sent to the video/image encoder. Our MJPEG encoder compresses each image individually as JPEG image.

This compression method saves a significant amount of time compared to other advance video compression techniques as we shown in our experiment. The average compressed image size is 3.5KB which is much smaller than sharing the full image at high quality setting. However, we limit the video streaming rate to 5 Hz similar to GNSS update rate to achieve best accuracy. Pedestrian distance $l$ and $\angle\beta$ are sent at the detection rate which is 8 to 10 Hz.

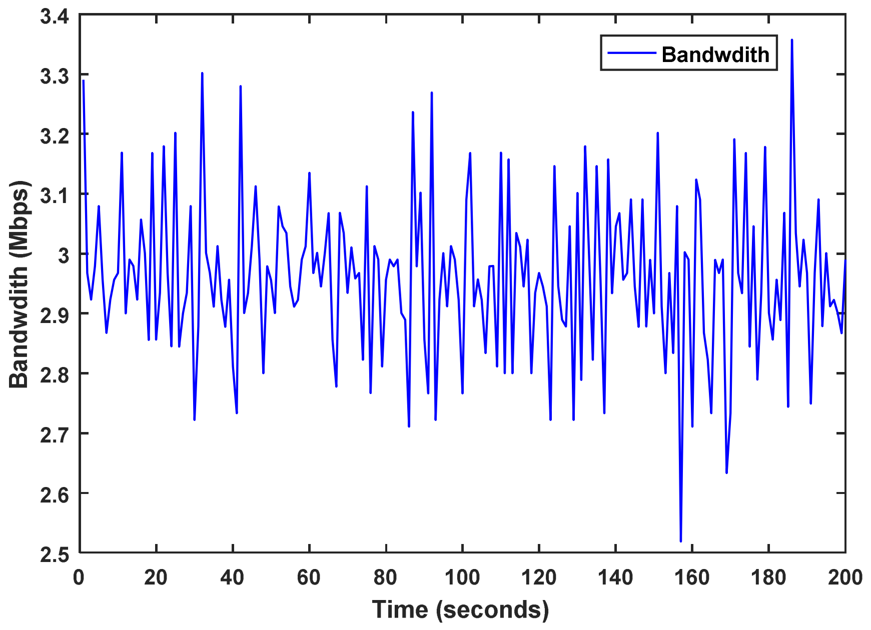


Figure 5.15: Bandwidth between two DSRC units

Fig.5.17 depicts the delay at every step of operation, where overall delay is between two consecutive images fusions including the display of the final fused image. The average overall delay is 200 ms which is similar to the video sharing rate of 5 Hz, mainly due to the fact that the

GNSS update is limited to 5 Hz. The fusion processes delay average is 33 ms and includes the delay caused by calculation, fusion and synchronization between remote and local data. Meanwhile the average channel object detection delays are 10ms and 122ms respectively. It is clear that the sum of the fusion, channel and object detection is less than overall delay, suggesting the 200ms delay is not processing delay but due to the 5Hz update rate of GNSS. It is possible to increase the information sharing rate by a) improving object detection processing rate without decreasing detection accuracy b) increasing GNSS rate.

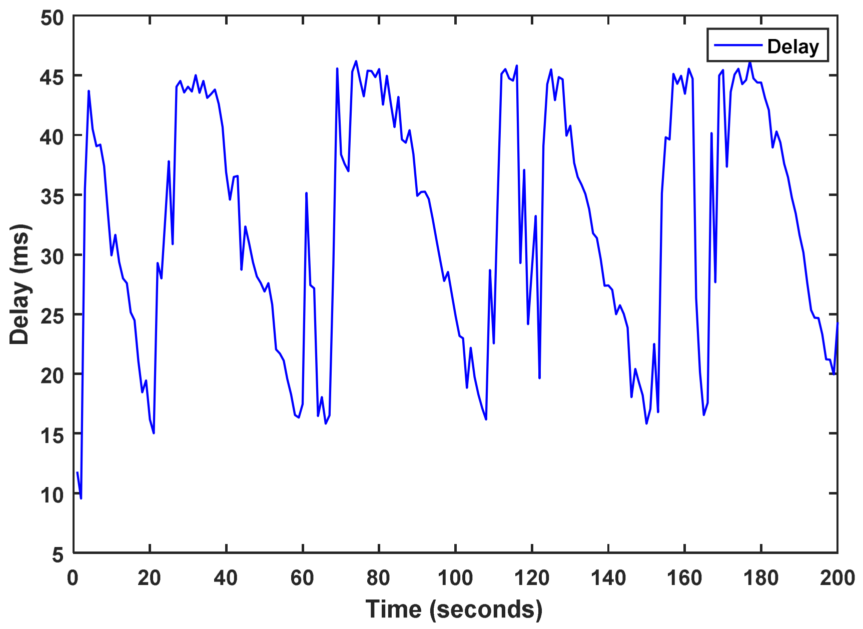


Figure 5.16: Packet delay between two DSRC units

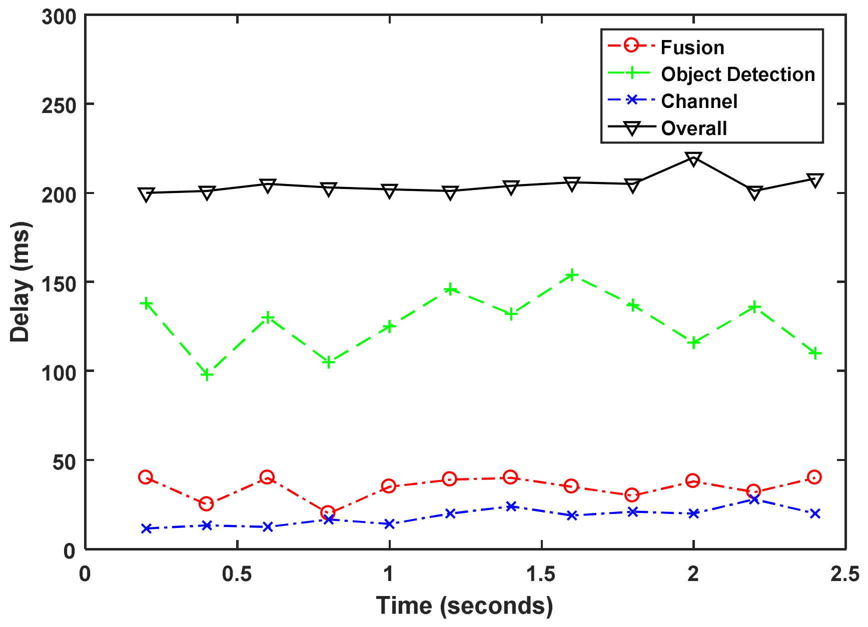


Figure 5.17: The delay of proposed system (CPCAS).

Table (5.2) shows the calculations that are conducted during our pre-collision interaction which lasted 2.4 seconds. During that interaction, the driver is warned about pedestrian crossing. A sample of the fused images is shown in Fig. 5.18.

Table 5.2: Proposed system (CPCAS) calculations results

| **Time (seconds)** | **Speed (m/s)** | **DTC (m)** | **TTC (seconds)** |
|---|---|---|---|
| 0 | 8.98 | 20.1 | 2.25 |
| 0.2 | 8.94 | 19.1 | 2.18 |
| 0.4 | 8.65 | 17.99 | 2 |
| 0.6 | 8.64 | 16.5 | 1.9 |
| 0.8 | 8.49 | 15.62 | 1.8 |
| 1 | 8.31 | 14.4 | 1.61 |
| 1.2 | 7.77 | 12.79 | 1.53 |
| 1.4 | 7.64 | 11.5 | 1.47 |
| 1.6 | 7.64 | 10.8 | 1.42 |
| 1.8 | 7.10 | 10.1 | 1.41 |
| 2 | 6.52 | 9.4 | 1.43 |
| 2.2 | 6.13 | 9.1 | 1.4 |

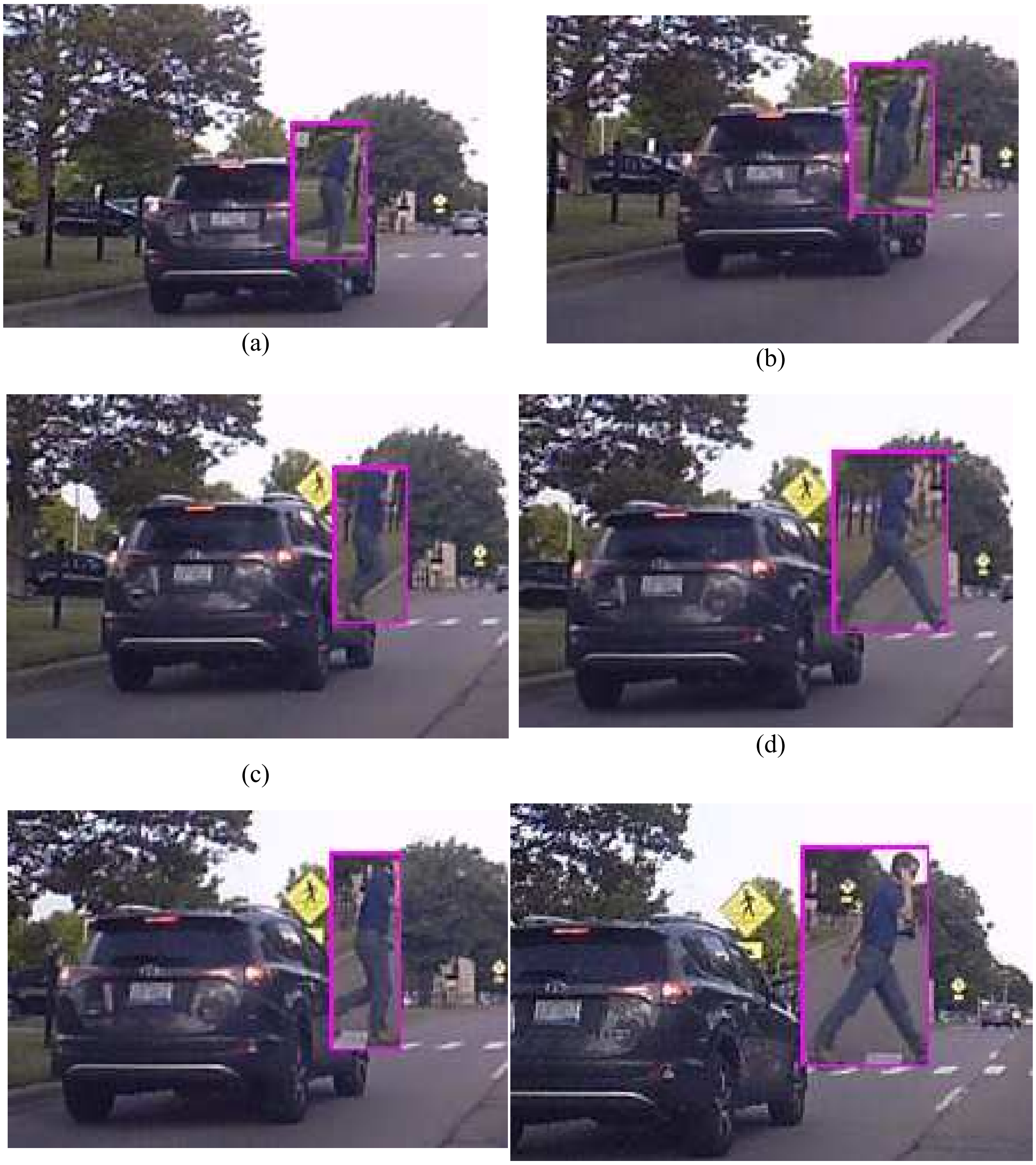

(a) (b) (c) (d) (e) (f)

Figure 5.18: A sample of CPCAS system image fusion results

# Chapter 6 Conclusions

In the first three sections of this thesis, we developed a novel approach, Erasable Packets within Internet Queues (EPIQ), for video streaming. EPIQ provides high quality video transmission with minimal network complexity. Under congestion, EPIQ works by erasing portions of multiple video packets within a network queue rather than the conventional packet dropping mechanism. We developed analytical models for our proposed solution for ideal and non- ideal operation then we verified them using simulation. We further introduced EPIQ-RED AQM, based on the well know RED algorithm, to implement our partial packet erasing. Furthermore, to maintain fairness and to control the rate of competing video streams, we presented EPIQ-TFRC. Our simulation results show that our system can achieve better performance than both multi-queue and HTTP-based video streaming. In summary, the proposed EPIQ solution introduces a new paradigm in video streaming by partially erasing packets to control congestion in the network while maintaining the highest video transmission quality.

In the last section of this thesis, we proposed a new Cooperative Advance Driver Assistance system (C-ADAS) that will potentially reduce the number of pedestrian accidents significantly. Our proposed system tries to eliminate obstruction of view problem that vehicle drivers face every day. Our system achieves low end-to-end delay .We introduced a selective sharing algorithm that decided what to share and with whom, thus minimizing the amount needed bandwidth for visual information sharing. Our proposed system can be utilized with various types of sensors to improve its performance. Our experiment shows that our system can achieve the desired results with minimum bandwidth usage and low delay.

# BIBLIOGRAPHY

# BIBLIOGRAPHY

[1] "White paper: Cisco VNI Forecast and Methodology, 2015-2020," Cisco. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. [Accessed: 20-Nov-2016].

[2] MPEG-DASH. ISO/IEC 23009, Retrieved November 16, 2015 from http://mpeg.chiariglione.org/standards/mpeg-dash

[3] A. Begen, T. Akgul, and M. Baugher, "Watching Video over the Web: Part 1: Streaming Protocols," IEEE Internet Computing, vol. 15, no. 2, pp. 54–63, Mar. 2011.

[4] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?," in Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video, 2012, pp. 9–14.

[5] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," Signal Processing: Image Communication, vol. 27, no. 4, pp. 271–287, Apr. 2012.

[6] R. Kuschnig, I. Kofler, and H. Hellwagner, "An Evaluation of TCP-based Rate-control Algorithms for Adaptive Internet Streaming of H.264/SVC," in Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems, New York, NY, USA, 2010, pp. 157–168.

[7] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive," IEEE/ACM Trans. Netw., vol. 22, no. 1, pp. 326–340, Feb. 2014.

[8] G. Tian and Y. Liu, "Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming," in Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, New York, NY, USA, 2012, pp. 109–120.

[9] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," IEEE Journal on Selected Areas in Communications, vol. 32, no. 4, pp. 719–733, Apr. 2014.

[10] "Inside The Netflix/Comcast Deal and What The Media Is Getting Very Wrong," Dan Rayburn - StreamingMediaBlog.com, 24-Feb-2014. [Online]. Available: http://blog.streamingmedia.com/2014/02/media-botching-coverage-netflix-comcast-deal-getting-basics-wrong.html. [Accessed: 22-Jul-2016].

[11] M. Nagy, V. Singh, J. Ott, and L. Eggert, “Congestion Control Using FEC for Conversational Multimedia Communication,” in Proceedings of the 5th ACM Multimedia Systems Conference, New York, NY, USA, 2014, pp. 191–202.

[12] “Viewer Experience Report 2015,” Conviva. [Online]. Available: http://www.conviva.com/conviva-viewer-experience-report/vxr-2015/

[13] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, “Understanding the Impact of Video Quality on User Engagement,” in Proceedings of the ACM SIGCOMM 2011 Conference, New York, NY, USA, 2011, pp. 362–373.

[14] [X. Zhu and R. Pan, “NADA: A Unified Congestion Control Scheme for Low-Latency Interactive Video,” in 2013 20th International Packet Video Workshop, 2013, pp. 1–8.

[15] R. Huysegems, J. van der Hooft, T. Bostoen, P. Rondao Alface, S. Petrangeli, T. Wauters, and F. De Turck, “HTTP/2-Based Methods to Improve the Live Experience of Adaptive Streaming,” in Pro. of the 23rd ACM Intern. Conf. on Multimedia, NY, USA, 2015, pp. 541–550

[16] D. Jarnikov and T. Ozcelebi, “Client intelligence for adaptive streaming solutions,” in 2010 IEEE International Conference on Multimedia and Expo (ICME), 2010, pp. 1499–1504

[17] Z. Li, A. C. Begen, J. Gahm, Y. Shan, B. Osler, and D. Oran, “Streaming Video over HTTP with Consistent Quality,” in Proceedings of the 5th ACM Multimedia Systems Conference, New York, NY, USA, 2014, pp. 248–258

[18] M. Grafl, C. Timmerer, H. Hellwagner, W. Cherif, and A. Ksentini, “Evaluation of hybrid Scalable Video Coding for HTTP-based adaptive media streaming with high-definition content,” in World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a, 2013, pp. 1–7

[19] Y. Sánchez de la Fuente, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Le Louédec, “iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding,” in Proceedings of the second annual ACM conference on Multimedia systems, 2011, pp. 257–264

[20] R. Kuschnig, I. Kofler, and H. Hellwagner, “An Evaluation of TCP-based Rate-control Algorithms for Adaptive Internet Streaming of H.264/SVC,” in Proceedings of ACM SIGMM Conference on Multimedia Systems, New York, NY, USA, 2010, pp. 157–168

[21] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation-based Congestion Control for Unicast Applications,” in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, New York, NY, USA, 2000, pp. 43–56

[22] L. De Cicco, G. Carlucci, and S. Mascolo, “Experimental Investigation of the Google Congestion Control for Real-time Flows,” in Proceedings of the 2013 ACM SIGCOMM

Workshop on Future Human-centric Multimedia Networking, New York, NY, USA, 2013, pp. 21–26

[23] S. Loreto and S. Pietro Romano, “Real-Time Communications in the Web: Issues, Achievements, and Ongoing Standardization Efforts,” IEEE Internet Computing, vol. 16, no. 5, pp. 68–73, Sep. 2012

[24] J. W. Evans and C. Filsfils, Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice. Morgan Kaufmann, 2010.

[25] D. Clark and W. Fang, “Explicit Allocation of Best-effort Packet Delivery Service,” IEEE/ACM Trans. Netw., vol. 6, no. 4, pp. 362–373, Aug. 1998.

[26] Juha Heinanen and Roch Guerin. A Two Rate Three Color Marker. IETF RFC 2698, 1999.

[27] R. Adams, “Active Queue Management: A Survey,” IEEE Communications Surveys & Tutorials, vol. 15, no. 3, pp. 1425–1476, 2013.

[28] T. Pliakas, G. Kormentzas, and C. Skianis, “End-to-end QoS issues of MPEG-4 FGS video streaming traffic delivery in an IP/DVB/UMTS network,” European Journal of Operational Research, vol. 191, no. 3, pp. 1089–1100, Dec. 2008

[29] J. Shin, J.-G. Kim, J. Kim, and C.-C. Jay Kuo, “Dynamic QoS mapping control for streaming video in relative service differentiation networks,” Eur. Trans. Telecomm., vol. 12, no. 3, pp. 217–229, May 2001

[30] V. Firoiu and X. Zhang, “Best Effort Differentiated Services: Trade-off Service Differentiation for Elastic App.,” in in Proc. IEEE ICT 2000.

[31] P. Hurley, J.-Y. Le Boudec, P. Thiran, and M. Kara, “ABE: providing a low-delay service within best effort,” IEEE Network, vol. 15, no. 3, pp. 60–69, May 2001.

[32] J. Gettys and K. Nichols, “Bufferbloat: Dark Buffers in the Internet,” Commun. ACM, vol. 55, no. 1, pp. 57–65, Jan. 2012

[33] Y. Li, X. Gong, W. Wang, X. Que, and J. Ma, “An Autonomic Active Queue Management Mechanism to Improve Multimedia Flow Delivery Quality,” in 2010 International Conference on Communications and Mobile Computing (CMC), 2010, vol. 1, pp. 493–497.

[34] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” IEEE/ACM Trans. Netw., vol. 1, no. 4, pp. 397–413, Aug. 1993

[35] Y. Xiaogang, L. JiQiang, and L. Ning, “Congestion Control Based on Priority Drop for H.264/SVC,” in International Conference on Multimedia and Ubiquitous Engineering, 2007. MUE ’07, 2007, pp. 585–589

[36] S. R. Kang, Y. Zhang, M. Dai, and D. Loguinov, "Multi-layer active queue management and congestion control for scalable video streaming," in 24th International Conference on Distributed Computing Systems, 2004. Proceedings, 2004, pp. 768–777

[37] "Intelligent Transportation Systems - Connected Vehicle Pilot Deployment Program." n.d. Accessed October 17, 2017. https://www.its.dot.gov/pilots/pilots_nycdot.htm.

[38] Matthew Lynberg. 2016. "U.S. DOT Advances Deployment of Connected Vehicle Technology to Prevent Hundreds of Thousands of Crashes." Text. NHTSA. December 15, 2016. https://www.nhtsa.gov/press-releases/us-dot-advances-deployment-connected-vehicle-technology-prevent-hundreds-thousands.

[39] 2016. "Pedestrian Safety." Text. NHTSA. September 9, 2016. https://www.nhtsa.gov/road-safety/pedestrian-safety.

[40] Geronimo, D., A. M. Lopez, A. D. Sappa, and T. Graf. 2010. "Survey of Pedestrian Detection for Advanced Driver Assistance Systems." IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (7):1239–58. https://doi.org/10.1109/TPAMI.2009.122.

[41] Lidstrom, K., K. Sjoberg, U. Holmberg, J. Andersson, F. Bergh, M. Bjade, and S. Mak. 2012. "A Modular CACC System Integration and Design." IEEE Transactions on Intelligent Transportation Systems 13 (3):1050–61. https://doi.org/10.1109/TITS.2012.2204877.

[42] Sawade, O., B. Schäufele, J. Buttgereit, and I. Radusch. 2014. "A Cooperative Active Blind Spot Assistant as Example for Next-Gen Cooperative Driver Assistance Systems (CoDAS)." In 2014 IEEE Intelligent Vehicles Symposium Proceedings, 76–81. https://doi.org/10.1109/IVS.2014.6856570.

[43] Xiong, G., H. Li, Y. Jin, J. Gong, and H. Chen. 2017. "Collision Avoidance System with Cooperative Adaptive Cruise Control in Highway Entrance Ramp Environment." In 2017 18th International Conference on Advanced Robotics (ICAR), 549–53. https://doi.org/10.1109/ICAR.2017.8023664.

[44] Sawade, O., and I. Radusch. 2015. "Survey and Classification of Cooperative Automated Driver Assistance Systems." In 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), 1–5. https://doi.org/10.1109/VTCFall.2015.7391161.

[45] Belyaev, E., P. Molchanov, A. Vinel, and Y. Koucheryavy. 2013. "The Use of Automotive Radars in Video-Based Overtaking Assistance Applications." IEEE Transactions on Intelligent Transportation Systems 14 (3):1035–42. https://doi.org/10.1109/TITS.2013.2248731.

[46] Asefi, M., J. W. Mark, and X. S. Shen. 2012. "A Mobility-Aware and Quality-Driven Retransmission Limit Adaptation Scheme for Video Streaming over VANETs." IEEE

Transactions on Wireless Communications 11 (5): 1817–27. https://doi.org/10.1109/TWC.2012.030812.111064.

[47] Mir, Z. H., and F. Filali. 2014. "On the Performance Comparison between IEEE 802.11p and LTE-Based Vehicular Networks." In 2014 IEEE 79th Vehicular Technology Conference (VTC Spring), 1–5. https://doi.org/10.1109/VTCSpring.2014.7023017.

[48] Dreyer, N., A. Moller, Z. H. Mir, F. Filali, and T. Kurner. 2016. "A Data Traffic Steering Algorithm for IEEE 802.11p/LTE Hybrid Vehicular Networks." In 2016 IEEE 84th Vehicular Technology Conference (VTC-Fall), 1–6 https://doi.org/10.1109/VTCFall.2016.7880850.

[49] Brahim, M. Ben, Z. Hameed Mir, W. Znaidi Global Status Report on Road Safety, 2015. [Online]. Available: http://www.who.int/violenceinjuryprevention/roadsafetystatus/2015/en/

[50] F. Filali, and N. Hamdi. 2017. "QoS-Aware Video Transmission Over Hybrid Wireless Network for Connected Vehicles." IEEE Access 5:8313–23. https://doi.org/10.1109/ACCESS.2017.2682278.

[51] Olaverri-Monreal, C., P. Gomes, R. Fernandes, F. Vieira, and M. Ferreira. 2010. "The See-Through System: A VANET-Enabled Assistant for Overtaking Maneuvers." In 2010 IEEE Intelligent Vehicles Symposium, 123–28. https://doi.org/10.1109/IVS.2010.5548020.

[52] Gomes, P., C. Olaverri-Monreal, and M. Ferreira. 2012. "Making Vehicles Transparent Through V2V Video Streaming." IEEE Transactions on Intelligent Transportation Systems 13 (2):930–38. https://doi.org/10.1109/TITS.2012.2188289.

[53] Belyaev, E., A. Vinel, K. Egiazarian, and Y. Koucheryavy. 2013. "Power Control in See-Through Overtaking Assistance System." IEEE Communications Letters 17 (3):612–15. https://doi.org/10.1109/LCOMM.2013.012213.122362.

[54] Kenney, J. B. 2011. "Dedicated Short-Range Communications (DSRC) Standards in the United States." Proceedings of the IEEE 99 (7):1162–82. https://doi.org/10.1109/JPROC.2011.2132790.

[55] Naus, G. J. L., R. P. A. Vugts, J. Ploeg, M. J. G. van de Molengraft, and M. Steinbuch. 2010. "String-Stable CACC Design and Experimental Validation: A Frequency-Domain Approach." IEEE Transactions on Vehicular Technology 59 (9):4268–79. https://doi.org/10.1109/TVT.2010.2076320.

[56] Joint Video Team, "Reference software," http://iphome.hhi.de/suehring/tml/

[57] A. Detti, G. Bianchi, C. Pisa, F. S. Proto, P. Loreti, W. Kellerer, S. Thakolsri, and J. Widmer, "SVEF: an open-source experimental evaluation framework for H.264 scalable video streaming," in IEEE Symposium on Computers and Communications, 2009. ISCC 2009, 2009, pp. 36–41

[58] M. Blestel and M. Raulet, "Open SVC decoder: a flexible SVC library," in Proceedings of the international conference on Multimedia, 2010, pp. 1463–1466

[59] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103–1120, Sep. 2007

[60] V. Jacobson, K. Nichols, K. Poduri, and C. Systems, "RED in a Different Light," 1999

[61] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, " A control theoretic analysis of RED," in *IEEE INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, 2001, vol. 3, pp. 1510–1519

[62] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate Control for Comm.Netw.: Shadow Prices, Proportional Fairness and Stability," *The Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998

[63] E. Kohler, M. Handley, S. Floyd*, Datagram Congestion Control Protocol (DCCP)*, IETF RFC4340, March 2006

[64] Derf's short video collection." http://media.xiph.org".

[65] andrew.currin.ctr@dot.gov. 2016. "Safety Technologies." Text. NHTSA. September 8, 2016. https://www.nhtsa.gov/equipment/safety-technologies.

[66] "Reach." . Emlid (blog). Accessed October 17, 2017. https://emlid.com/reach/.

[67] Redmon, Joseph, and Ali Farhadi. 2016. "YOLO9000: Better, Faster, Stronger." ArXiv:1612.08242 [Cs], December. http://arxiv.org/abs/1612.08242.

[68] "The PASCAL Visual Object Classes Homepage." Accessed October 17, 2017. http://host.robots.ox.ac.uk/pascal/VOC/.