Improving Resilience of Communication in Information Dissemination for Time-Critical Applications

Rajvardhan Somraj Deshmukh
University of Massachusetts Amherst

IMPROVING RESILIENCE OF COMMUNICATION IN INFORMATION DISSEMINATION FOR TIME-CRITICAL APPLICATIONS

A Thesis Presented

by

RAJVARDHAN.S.DESHMUKH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

May 2019

Electrical and Computer Engineering

IMPROVING RESILIENCE OF COMMUNICATION IN INFORMATION DISSEMINATION FOR TIME-CRITICAL APPLICATIONS

A Thesis Presented

by

RAJVARDHAN.S.DESHMUKH

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my advisor, Professor Michael Zink, who has inspired and encouraged me to go beyond and grow as a student as well as a person. His valuable ideas, constructive feedback and right push has enabled me to perform well in academics, research and extra-curricular as well. I definitely look up to him as a life long mentor and I feel grateful to work with him and have him as my advisor. I am heartily thankful to my committee members Professor Lixin Gao and Professor David Irwin for their constructive advice and invaluable help in my research and future career.

I also would like to acknowledge my labmates for the knowledge and experience they have shared with me. In particular, I want to thank Thiago Teixeira and Divyashri Bhat, who worked together with me on thoughtful projects and helped me put together the resources used in this thesis. A special thank you also goes out to everyone in the Data Analytics Lab for the knowledge and experience they have shared with me. Especially, Babak for his motivation, help and belief in me.

Finally, I appreciate all of the sincere support from my family and friends who have always been there for me and continue to inspire and encourage me.

ABSTRACT

IMPROVING RESILIENCE OF COMMUNICATION IN INFORMATION DISSEMINATION FOR TIME-CRITICAL APPLICATIONS

MAY 2019

RAJVARDHAN.S.DESHMUKH

B.Tech., VELLORE INSTITUTE OF TECHNOLOGY, INDIA

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Michael Zink

Severe weather impacts life and in this dire condition, people rely on communication, to organize relief and stay in touch with their loved ones. In such situations, cellular network infrastructure[1] might be affected due to power outage, link failures, etc. This urges us to look at Ad-hoc mode of communication, to offload major traffic partially or fully from the infrastructure, depending on the status of it.

We look into threefold approach, ranging from the case where the infrastructure is completely unavailable, to where it has been replaced by make shift low capacity mobile cellular base station, such as in [58] and [71].

First, we look into communication without infrastructure and timely, dissemination of weather alerts specific to geographical areas. We look into the specific case of

[1]We refer to cellular network infrastructure as infrastructure for the entirety of this document

floods as they affect significant number of people [2]. Due to the nature of the problem we can utilize the properties of Information Centric Networking (ICN) in this context, namely: i) Flexibility and high failure resistance: Any node in the network that has the information can satisfy the query ii) Robust: Only sensor and car need to communicate iii) Fine grained geo-location specific information dissemination. We analyze how message forwarding using ICN on top of Ad hoc network, approach compares to the one based on infrastructure, that is less resilient in the case of disaster. In addition, we compare the performance of different message forwarding strategies in VANETs (Vehicular Adhoc Networks) using ICN. Our results show that ICN strategy outperforms the infrastructure-based approach as it is 100 times faster for 63% of total messages delivered.

Then we look into the case where we have the cellular network infrastructure, but it is being pressured due to rapid increase in volume of network traffic (as seen during a major event) or it has been replaced by low capacity mobile tower. In this case we look at offloading as much traffic as possible from the infrastructure to device-to-device communication. However, the host-oriented model of the TCP/IP-based Internet poses challenges to this communication pattern. A scheme that uses an ICN model to fetch content from nearby peers, increases the resiliency of the network in cases of outages and disasters. We collected content popularity statistics from social media to create a content request pattern and evaluate our approach through the simulation of realistic urban scenarios. Additionally, we analyze the scenario of large crowds in sports venues. Our simulation results show that we can offload traffic from the backhaul network by up to 51.7%, suggesting an advantageous path to support the surge in traffic while keeping complexity and cost for the network operator at manageable levels.

[2]According to NWS statistics (http://www.nws.noaa.gov/om/hazstats.shtml) flood related death are the highest amongst all weather fatalities

Finally, we look at adaptive bit-rate streaming (ABR) streaming, which has contributed significantly to the reduction of video playout stalling, mainly in highly variable bandwidth conditions. ABR clients continue to suffer from the variation of bit rate qualities over the duration of a streaming session. Similar to stalling, these variations in bit rate quality have a negative impact on the users' Quality of Experience (QoE). We use a trace from a large-scale CDN to show that such quality changes occur in a significant amount of streaming sessions and investigate an ABR video segment retransmission approach to reduce the number of such quality changes. As the new HTTP/2 standard is becoming increasingly popular, we also see an increase in the usage of HTTP/2 as an alternative protocol for the transmission of web traffic including video streaming. Using various network conditions, we conduct a systematic comparison of existing transport layer approaches for HTTP/2 that is best suited for ABR segment retransmissions. Since it is well known that both protocols provide a series of improvements over HTTP/1.1, we perform experiments both in controlled environments and over transcontinental links in the Internet and find that these benefits also "trickle up" into the application layer when it comes to ABR video streaming where HTTP/2 retransmissions can significantly improve the average quality bitrate while simultaneously minimizing bit rate variations over the duration of a streaming session. Taking inspiration from the first two approaches, we take into account the resiliency of a multi-path approach and further look at a multi-path and multi-stream approach to ABR streaming and demonstrate that losses on one path have very little impact on the other from the same multi-path connection and this increases throughput and resiliency of communication.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Motivation

Reliable communication is important in general and it is even more crucial during disasters. Be it sending alerts, trying to communicate with people or organizing relief. As shown in [7], it is important for government and the telephone operators to work together during recovery. Due to wide availability of mobile phones, [7] they are an important, supplementary source of information dissemination, as other infrastructure is vulnerable and widely affected in the aftermath of a disaster. As infrastructure is vulnerable during these scenarios, it motivates us to look at ad-hoc mode of communication, to be used independently or along with the infrastructure, as a multipath approach.

1.2 Contributions and Outlines

This thesis includes six chapters. In Chapters 1-2, we provide general information of this dissertation. Chapter 1 gives an overall introduction and motivation of this thesis. Chapter 2 introduces the background information and provides explanations of key concepts that will be commonly used in later parts of this thesis.

In Chapter 3, we compare the feasibility of using purely ad-hoc mode of communication with ICN on top of it as compared to the standard infrastructure based current approach. We test it with an application that disseminates flood related alerts. ICN's prefix naming and caching properties remove the dependency on end-to-end paths as compared to an ip based approach. Our timer based message forwarding strategy which considers the geographical location and velocity of the devices reduce packet collusions.

In Chapter 4, we propose a scheme that utilizes Device-to-Device (D2D) communication and ICN to offload increasingly congested base stations, facilitating the communication between peers when the infrastructure is impaired. Additionally, our scheme considers energy saving measures and reduces content flooding in the MANET by suppressing the propagation of requests when the energy on the node falls below a certain threshold. Our solution successfully offloads traffic from the base station, consequently reducing cost and complexity for the cellular network.

In Chapter 5, we begin by analyzing 5 million video streaming sessions that show switches in quality representations which result in gaps almost 36% of all streaming sessions. In the case of mobile clients this number increases to 50%. The overall QoE of these sessions could benefit from retransmitting ABR video segments in a higher quality. We make a systematic comparison of the multiplexing feature of HTTP/2 for this case. Our evaluation results show that HTTP/2 retransmissions can *significantly* improve the average quality bitrate while simultaneously *minimizing* bit rate variations over the duration of a streaming session. We further look at using multipath, as it increases the throughput and resilience.

In the last chapter, we conclude this thesis, and provide possible directions for future research works.

CHAPTER 2
BACKGROUND AND RELATED WORK

2.1 Information Centric Networking

In ICN, data is immutable and decoupled from its location, enabling a node to fetch content from any other node in the network that has a cached copy of the requested data. In-network caching is supported as part of the architecture. Furthermore, ICN supports multipath communication and prevents loops. These characteristics make ICN more tolerant to delay and disruptions than host-centric architectures. The work presented in Chapters 3 and 4 is based on Named-Data Networking (NDN) [89], one of the many flavors of ICN. NDN is a consumer driven architecture, i.e., the consumer application initiates the communication by sending *Interest* packets upstream to retrieve *Data* packets. Interest packets are generally small packets (approximately 40 bytes) that are used by the consumer to express interest in retrieving certain data. It does so by using a hierarchical naming scheme, also called namespace (a set of named contents that begins with a certain prefix, e.g., */edu/umass/*home/version). Data packets contain the actual payload, with all packets being signed by the originator. NDN also has a Negative-Acknowledgment (NACK) packet type, used by the network nodes to express that a certain content is not present or that it received a duplicate Interest request. Similar to IP networks, packets are forwarded by intermediate nodes towards the destination. In NDN however, these nodes are augmented by the following data structures. A *Content Store (CS)* that caches incoming Data packets. Upon receiving an Interest request, the node verifies whether the requested data is cached, returning it if positive. Besides reducing the amount of traffic that is sent upstream, this feature increases the tolerance to disruption (in the case of a disconnection, the content will be stored closer to the client). A *Pending Interest Table (PIT)* keeps a record of Interest requests, incoming and outgoing interfaces, aggregating similar requests, resulting in a stateful data plane. Data packets follow the reverse Interest path. The PIT prevents loops in the network, which is an important feature in ad-hoc networks. The *Forwarding Information Base (FIB)* holds the information on which interface to forward a specific Interest packet. Another important part of the NDN architecture is the strategy layer, that determines the behavior of a node when multiple paths exist. Moreover, a node can apply different strategies to different namespaces.

2.2 D2D Communication and VANETs

In [46], the authors base their VANETs approach on a traditional TCP/IP architecture. They analyze different types of routing protocols in the case of VANETs and conclude that geographical (position based) routing is the best fit. They used the following parameters: position of the current node, its distance from the destination (assuming it knows this) and link layer quality (SINR and MAC frame rate) to assign weights to select the next hop nodes. They also implement the carry-and-forward mechanism (use cache) for less dense networks. We are not concerned about the link quality in particular, as we do not select a single next hop node and use the NDN architecture's implicit properties of Interest aggregation to tackle the problem of broadcast storms. In addition, we make us of in-network caching such that intermediate nodes that have cached the data can satisfy the Interest.

In [85], the authors present a VANETs approach for rapid traffic information dissemination that uses NDN. They identify collisions of packets in the network as a major problem and implement a set of timers to minimize this. These timers are designed to accommodate the idea that data has more value further away from the point of origin. A source agnostic approach is used, i.e., the distance from the current node to the source is not known, which is unlike to the approach presented in [46]. This increases the scalability of the approach. The topology in our case is such that all the mobile nodes are data consumers and the producers are static. This is different to the scenario described in [85], where single consumer nodes are placed at various positions (e.g., start, middle, or end of the traffic queue). In our scenario, traffic is multi-directional (see Fig. 3.2) and we also include the velocity of a vehicle as a factor in the forwarding strategy. In addition, we use a naming convention that includes location information which better supports Interest dissemination.

In [17], the authors have designed (Ad Hoc Dynamic Unicast) ADU, which adaptively switches between unicast and broadcast in case of link failures. Such link failures are detected by the MAC layer. This approach employs the use of tables that store the MAC address of the next hop in case of uni-cast. Location information as used in our approach is not considered in this work, thus forwarding timers based on distance from the source can not be realized. In addition, setting up a path during provider discovery and switching to broadcast in case of link failure introduces too much overhead in our scenario, where low volume flooding alert information has to be transmitted reliably and with very low latency to cars in a certain region.

There have been a number of studies focused on using D2D communication to offload cellular base stations in the TCP/IP domain. In fact, the authors in [69] survey the literature to classify existing offloading approaches into two main categories: AP-based (where traffic is offloaded via IEEE 802.11 networks) and Terminal-to-Terminal (or simply D2D communication). In the latter, we can further segment the techniques into timer-based (where nodes delay forwarding to reduce collision) [55, 16], geographical-based (where farther nodes re-broadcast first) [86, 35], randomized broadcasting (again to reduce collisions by making nodes re-broadcast at random times) [16], contact-based (based on the number of neighbors (degree of a node) or number of visited nodes) [19], and a combination of two or more approaches [47]. On the contrary, there have been few studies to use *ICN over LTE* to take advantage of in-network caching to decrease traffic in the backhaul network.

In [18], Lopes et al. developed an application whereby messages are stored in a *Content Manager* module, much like a cache. In the case that the destination is not in the vicinity of the sending node, the *Routing* module consults the *Social Proximity* of the neighbors to decide which node to forward the packet to. This decision is based on the frequency that each node meets the destination node (thus, social proximity). Therefore, the sending node creates a socket-like connection via WiFi-Direct and transfers the message to the node that will most likely meet the destination node in the future. Every node can also carry other nodes' messages (data muling). Although this scheme implements concepts of ICN, it uses MAC addresses to route messages instead of content names. Moreover, it does not support multicast, as only the node that has seen the destination more often carries the request.

The authors in [34] expand an LTE-based architecture to include NDN routers, which are co-located with eNBs to implement caches at the backhaul network, then address the problem of content allocation optimization. In particular, the problem of content allocation optimization is addressed by determining where, when, and how content should be migrated. Their approach enhances the network response to user mobility via a set of parameters derived from the LTE network. Among the results, Gomes et al. found that latency can be reduced by using NDN default caching strategies (e.g. LRU). In addition, considering the amount of free space at the destination cache when placing content yields the most benefits. The authors, however, do not consider D2D communication as part of their design. In our work, we leave content migration to the LTE handover process (which was not considered in [34]).

In the vehicular network domain, Navigo [35] proposes a location-based packet forwarding mechanism to reduce disruption and path changes. The main idea is to forward requests to geographical regions where the content might be located. The location of contents is computed using a broadcast strategy, called exploration phase. Navigo is purely ad hoc as it does not consider base stations in their topology. Our approach is comprised of an ad hoc and infrastructure components. In [81], Vigneri et al. propose an augmented architecture where public service vehicles serve as relay points to offload traffic from the backhaul network. In this architecture, MNOs can place content on these vehicles that can later be retrieved by end-users. This approach showed improvements of up to 50% in the traffic offloaded from the backhaul network. In contrast, our approach differs from previous works in the sense that we modify the NDN protocol stack to develop a custom forwarding strategy to better take advantage of D2D communication that improves the reliability of cellular networks, making it more resilient to failures.

2.3 Transport layer for HTTP2

With the advent of QUIC there are now two options for the transport of HTTP/2 sessions between a web server and a browser. First, the original approach specifies the use of HTTP/2 over TCP. The second approach involves QUIC as an additional application layer protocol, which results in a HTTP/2 over QUIC over UDP solution [6].

TCP requires a 3-way handshake resulting in a 1.5 RTT before any data request is received at the server. In the case of QUIC, the data request arrives after 0.5 RTT at the server.

The default congestion control algorithm implemented in QUIC is similar to that of TCP Cubic [37] with some important differences. In order to notify the sender of the train of packets received, existing TCP mechanisms (including CUBIC) make use of Selective Acknowledgements (SACK) that include a maximum of the 3 most recent sequential packets that arrived successfully. The sender then retransmits the lost packets with sequence numbers that lie within the range of the 3 SACKs received. It is obvious that this approach imposes a heavy constraint on the number of TCP retransmissions that can take place without response from the receiver. QUIC aims to resolve this by including the use of NACKs and allows the receiver to send up to 256 NACKs without waiting for a response from the sender. The use of NACKs allows much faster loss recovery and can lead to asignificant reduction in rebuffering at DASH clients.

A recent paper by Google [50] provides a detailed analysis of an Internet-scale deployment of QUIC. They specifically look at latency and rebuffer rate in order to understand the performance implications of QUIC for video streaming over YouTube. Timmerer et al. [78] evaluate ABR streaming over QUIC for varying network latencies and show that there is no significant benefit to QoE streaming with the use of QUIC. In [77], a demonstration by Szabó et al. provides a new congestion control mechanism for QUIC that aggressively varies download rate according to a buffer-based priority level assigned by the ABR streaming client. Carlucci et al. [24] present results that compare TCP and QUIC under varying network conditions and buffer size. In [45], Kakhki et al. perform a detailed analysis of QUIC under varying network conditions to investigate the benefits of using QUIC for applications such as web browsing and video streaming over YouTube. The authors of [21] also compare the performance of several rate adaptive DASH players including QUIC and conclude that QUIC is more aggressive compared to TCP. The authors of [39] devise and deploy an SDN approach to to improve the QoE of ABR streaming by monitoring MPTCP retransmissions where their system dynamically switches between network paths and protocols to mitigate re-ordering effects. While we similarly compare the performance of TCP (using HTTP/1.1 and HTTP/2) with QUIC, our work is more focussed on the potential benefits that QUIC can provide for video streaming especially with respect to retransmitting video segments in higher qualities. Similar experiments are performed by the authors of [41], where they use the multiplexing feature of HTTP/2 to simultaneously request multiple qualities of a segment. While retransmissions can be regarded as an additional burden on the available bandwidth we note that recent works such as [82] suggest different types of redundant transmission to provide higher QoS. In contrast to [41] and [82], we only invoke retransmissions in a systematic way, thereby guaranteeing an improvement in QoE while also minimizing the consumption of additional bandwidth. Moreover, in order to analyze the implications of specific network conditions that affect ABR video streaming, we design, develop and prototype such a system in a nearly isolated, controlled testbed environment.

Legacy protocols that perform adaptive bitrate video streaming over UDP include systems such as Real-time Transport Protocol (RTP) [36] and Stream Control Transport Protocol (SCTP) [76]. Similar to QUIC, SCTP also allows multiplexing of multiple chunks into one packet and avoids HOL blocking, thus, allowing unordered delivery to the application layer. Unlike QUIC, SCTP implements congestion control according to the TCP *NewReno* specification which uses Selective Acknowledgement (SACK) for loss recovery. Another example of an ABR protocol over UDP is the Video Transport Protocol (VTP) which was designed and evaluated by Balk et al. [22]. In this work, the authors employ a form of congestion avoidance where the sending rate at the server is increased by a single packet for every RTT measurement. This design is different from the AIMD congestion control employed by TCP and QUIC since it eliminates the effect of slow start and attempts to provide an accurate estimate of the available bandwidth in the network. Some drawbacks of this approach are the requirement of two UDP sockets for every connection and the use of Berkeley Packet Filters to collect timestamps at the server and client for every video stream, thus, reducing both performance and scalability of the system. Although there are a

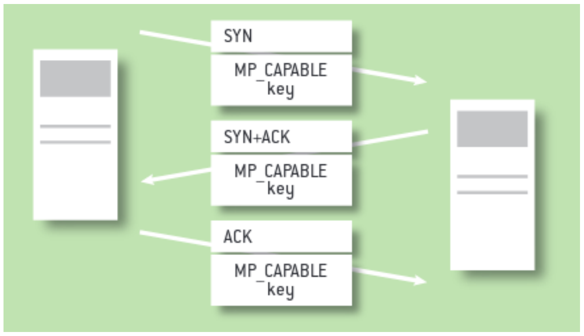


Figure 2.1: MPTCP connection establishment [1]

number of server push approaches such as [41] and [87] that have been proposed for HTTP/2, adapting such systems for retransmissions would not scale since the computation and storage overhead incurred on the server per individual client connection would render such an approach infeasible.

2.4 Multipath TCP

Current networks have become more friendly to multipath and the availability of multiple radio interfaces gives us the opportunity to harness multiple paths so as to ensure reliability, as we can detect link failures immediately and switch all the traffic to the other path. Furthermore, sending data simultaneously across more than one path can balance load and pool resources.

Multipath TCP (MPTCP) [31] is a major modification to TCP that allows multiple paths to be used simultaneously by a single transport connection. It distributes data from one connection across both the paths and they are called subflows.

According to [67], MPTCP is negotiated via a new TCP options in the SYN packets and to later add new paths (subflows) to an existing connection, identifiers are exchanged between end-points, as show in Figure 2.1. The 3-way handshake establishes the first TCP subflow over one interface. Figure 2.2 shows that, while adding a subflow to an existing MPTCP connection the corresponding MPTCP connection is

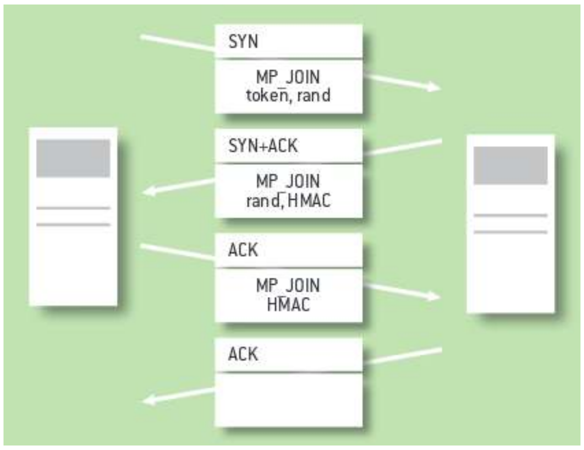


Figure 2.2: Establishment of Additional Subflow [1]

uniquely identified on each end host, for this MPTCP assigns a locally unique token to each connection. When a new subflow is added to existing connection, the token of the associated connection is sent in the MP_JOIN option of the SYN segment. Subflows resemble TCP flows on the wire otherwise the middle-boxes can cause some issues. These subflows share a single send and receive buffer at the end-points. To detect losses and perform retransmissions, it uses per subflow sequence number and to allow re-ordering at receiver, it uses connection level sequence number. Connection level ACK's are used to implement proper flow control.

As seen by [61], MPTCP can quickly recover from a WiFi loss in presence of a 3G interface with only a small impact on the application delay and goodput. Nonetheless, one needs to be careful while configuring MPTCP parameters, as [28] show that the performance depends on application specific MPTCP configurations. [66] Further explore load balancing capabilities, and they use very short timescale distributed load balancing, so as to make effective use of parallel paths, which results in higher performance and better resilience to failure, by combining several high speed interfaces.

2.5 ABR Streaming

Many content providers have switched to ABR streaming as it ensures higher user experience. Recent ABR streaming technologies are built on top of the Hyper Text Transfer Protocol (HTTP) application layer.

As the worse cases in the 'wild' internet consist of low network capacity or high bandwidth fluctuation, clients can be exposed to frequent packet losses which leads to stalled/interrupted playback experience. To tackle this issue, AVC based ABR streaming approach transcodes single layer, high quality videos into multiple copies with different bitrates, then for each bitrate quality, the video is encoded into smaller segments with short duration, typically between 2 to 10 seconds, as per the implementation. This facilitates different client devices, varying from mobile phones to TV's, to request video segments of suitable bitrates from the same video server according to their screen resolution, processor capabilities and network conditions.

Before the client begin downloading video content, they have to retrive the Media Presentation Description (MPD) file containing a list of all video and audio bitrates and segment available on the server. Depending on the client capabilities explained above, it chooses a video segment of particular bitrate and requests it by sending HTTP GET request to the appropriate server or cache, and, when it gets the video segment in response, it's stored in the client buffer. While downloading, the ABR streaming algorithm continuously monitors the available network bandwidth, and makes decision on the next bitrate request accordingly. In the case of network congestion, instead of stalling the playback, the ABR streaming mechanism requests a lower quality bitrate segment, and therefore can maintain a smooth playback experience for users.

MPEG's Dynamic Streaming over HTTP (DASH) is a popular implementation of ABR streaming, as *i)* DASH-format videos can be streamed from any kind of HTTP server *ii)* The adaptation logic which decides the quality of the next segment to be

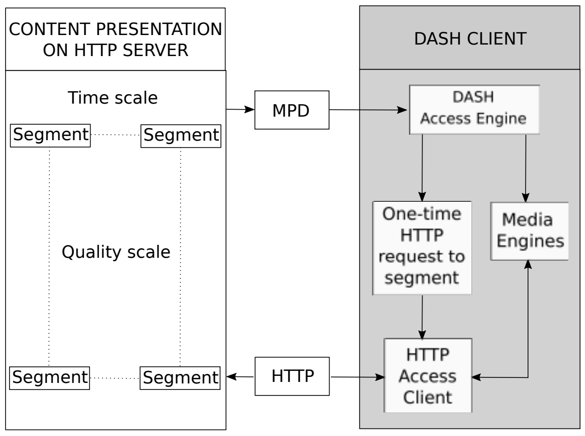


Figure 2.3: DASH flow diagram

downloaded, resides in the client *iii)* It is an open standard. Fig. 2.3 depicts the DASH streaming flow process. As mentioned in ABR, DASH gets the MPD and depending on the available bandwidth and local buffer conditions, requests a suitable quality video segment. Therefore, with the benefits of lightweight HTTP server, and the flexibility of bitrate adaptation, DASH is able to provide high resilience to network variations and smooth playback experience.

2.5.1 QoE Metrics

For the evaluation of the performance of ABR streaming applications we make use of the following metrics, which are widely used in related work:

2.5.1.1 Average Quality Bitrate (AQB)

One of the objectives of quality adaptation algorithms is to maximize the average quality bitrate of the streamed video. For a comprehensive QoE representation, we need to combine this metric with the *Number of Quality Switches* which is explained below.

2.5.1.2 Number of Quality Switches ($\#QS$)

This metric is used together with AQB to draw quantitative conclusions about the perceived quality (QoE). For example, for two streaming sessions having the same AQB, the session with the lower $\#QS$ will be perceived better by the viewer [91].

2.5.1.3 Spectrum (H) [91]

The spectrum of a streamed video is a centralized measure for the variation of the video quality bitrate around the AQB. A lower H indicates a better QoE.

2.5.1.4 Rebuffering Ratio (RB)

The average rebuffering ratio is given by the following equation:

$$RB = \mathsf{E}\left[\frac{t_a - t_e}{t_e}\right], \tag{2.1}$$

Where t_a is the actual playback time and t_e is the video length in seconds, respectively.

It is well known that low rebuffering and high average quality bitrate are highly desirable for an optimal QoE. In our previous work [84], we show that reducing quality gaps through ABR segment retransmissions can contribute significantly to a higher AQB. In the following, we present an analysis of actual quality gaps that occur in a real-world trace.

CHAPTER 3

INFORMATION CENTRIC NETWORKING SYSTEM FOR DISSEMINATING ALERTS IN VANETS

3.1 Introduction

Flooding events are amongst the most devastating disasters world-wide. Such events cause a high number of casualties[1] and significant damage of property. One particular problem with (even smaller, localized) flooding events in urbanized areas in the US is the fact that vehicles are driven into hazardous flood waters. Drivers often underestimate the depth of the water and the strength of the current of the flood waters, which is also shown in Fig. 3.1 [2]. To prevent such incidents, more and more underpasses (and other flood prone road sections) are outfitted with gauges that can measure water depth and flow. First generation systems were often coupled to barrier and warning systems with the goal of preventing drivers to enter the underpass in the case of flooding. Current, second generation systems, use the public Internet to disseminate warning information directly to smart phones through apps like Flood Alert [26]. While these second generation systems are a major improvement, since they can reach out to a large audience and warn drivers significantly ahead of the threat, they heavily rely on communication infrastructure. Especially in severe weather events, which are one of the major causes for such flooding events, infrastructure is highly susceptible to failure and an alert system might not be able

[1]According to NWS statistics (`http://www.nws.noaa.gov/om/hazstats.shtml`) flood related death are the highest amongst all weather fatalities.

[2]Survey by Brenda Phillips, CASA

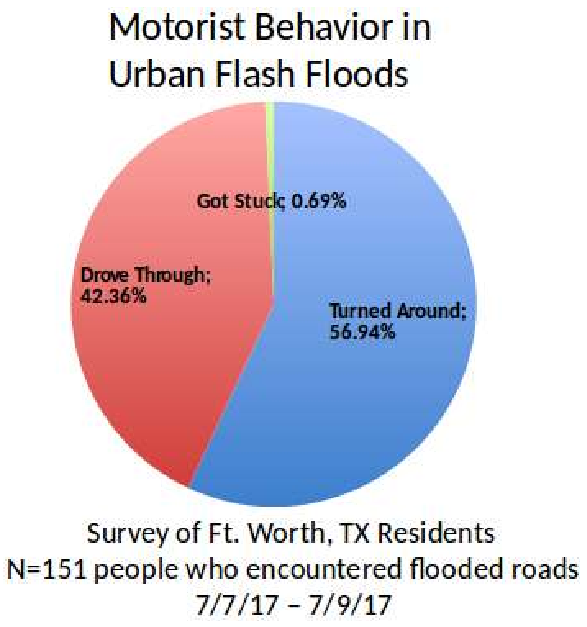


Figure 3.1: People driving through floods

to warn its users. In addition, users have to register with such systems and are often only notified if an event in the region they specified as the "home" location occurs. Thus a driver who is traveling, e.g., in a rental car during a trip many miles away from his/her home might not receive an alert.

In this chapter, we investigate an approach that is based on the new principle of Information Centric Networking (ICN), which does not rely on the end-to-end data delivery principle [90] that exists in traditional IP-based networks. In ICN, information can be transmitted from any node in the network that currently stores it, which is particular interesting in a scenario where cars (or smartphones in cars) are part of the communication infrastructure. Our approach makes use of this characteristic and enables direct communication between the flooding sensor and vehicles in its vicinity. In addition, information can also be exchanged in between cars. In comparison to the second generation warning systems described above, this approach has the advantage that a significant amount of the communication infrastructure can fail, while warnings can still be transmitted to cars. We claim that only the sensor and the cars in a certain vicinity of the sensor have to be able to communicate on an ad-hoc basis to allow the reliable and prompt dissemination of warnings.

It is our main goal to evaluate the performance of an ICN-based road assistance approach that alerts drivers in the case of flooding related road hazards. While our work focuses on the hazard of flooded underpasses, we believe that this approach can be easily applied to other road assistance systems as the ones that warn drivers of icy roads. The initial evaluation we present in this chapter is based on simulations and makes use of the Named Data Networking (NDN) [89] architecture which is an instantiation of ICN. While we evaluate the basic NDN-based approach and compare it with an approach that mimics a second generation system, we also study the performance of different forwarding strategies within NDN. Instead of simple flooding as strategy for the forwarding of data, we investigate approaches that take into account information such as distance between sensor and cars or the velocity and direction of a car.

Our simulation-based evaluation results, in which we investigate a typical traffic scenario involving an underpass, show that the NDN-based approaches outperform a traditional, infrastructure-based approach if node densities stay below a certain level. Our results show that the timer based approaches outperforms the pure flooding technique in at least 70% of the cases. In the case of an infrastructure based approach with LTE, the NDN approach outperforms it in low-density cases while performing equally in high-density cases.

3.2 Design

NDN [89] has the goal to build a future, ICN-based Internet architecture. In NDN, when a consumer requests data it send and Interest packet to the network that includes a name, which specifies the requested content. Once the Interest arrives at an NDN router, the router checks its Content Store (CS, built-in cache) for matching

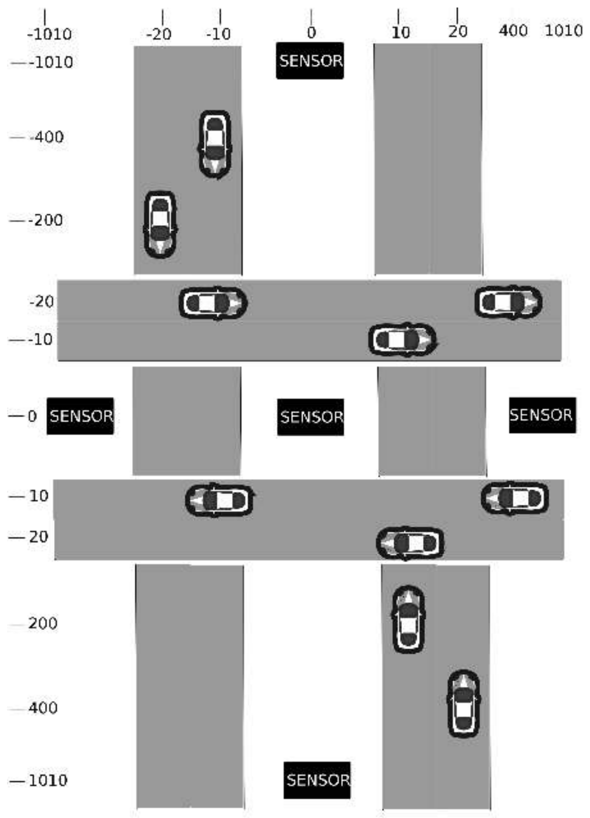


Figure 3.2: VANETs topology

data. If the data are found in the CS the router returns a DATA packet to the consumer. Otherwise, an entry in the Pending Interest Table (PIT) is created and the Interest is forwarded towards a producer according to information in the Forwarding Information Base (FIB). Should Interests for the same name arrive at the router, only one is forwarded upstream towards the producer.

In our scenario (see Fig. 3.2), the sensor is the producer, while the cars take on the role of both consumers and NDN routers. The cars send Interest packets to obtain information about potential road hazards (e.g., a flooded underpass). These Interests are forwarded according to the strategies described in Sect. 3.2.2. Since the cars also act as NDN routers Data packets can be cached in the CS of each mobile node. Thus, it is not necessary to forward an Interest all the way to the sender but it can also be answered by a car that is on the path between the requesting consumer and the sensor. With the use of NDN, no additional infrastructure than the sensor itself and connected cars[3] is required.

Our decision to use NDN as the basis for our approach, in comparison to alternative ICN approaches, is based on the fact that the project offers simulation tools [15] and an actual implementation of the architecture that can be used by the research community.

We also introduce an alternative, infrastructure-based architecture with which we compare the NDN approach in our evaluation (see Sect. 5.1.5). In this architecture, the mobile nodes (cars) and the sensor use LTE wireless technology to connect to the public Internet. Here, the sensor periodically transmits sensed data to a web server that is located somewhere in the Internet. Instead of requesting data directly from the sensor, the mobile nodes send HTTP GET request to this server to obtain flooding information.

3.2.1 Prefix Naming

Since our approach relies on forwarding based on location, direction, and velocity of the consumer, we introduce the naming scheme used in our approach in this section.

First of all, we use named prefix [64] to maintain a hierarchical structure, for example "/ndn/umass/WaterSensor", and further use a uni-dimensional structure in adding encoded geographical coordinates, for example "/ndn/umass/WaterSensor/9/8/0/2" (we assume that the mobile nodes have a device that provides geographic location information like GPS). This geographical information can be used in the forwarding algorithm. The resulting naming scheme is used such that the data source irrespective of whether it is the original producer or an intermediate node that has cached the content, can drop the Interest based on the location information in the name. In addition, by adjusting the resolution of the encoded geographical coordinates, we

[3]For this initial work, we assume that cars are connected through the drivers' smartphones

can ensure that nodes only reply to or forward Interests of customers in a certain geographical region.

NDN names, although hierarchical, are uni-dimensional, therefore we use Cantor pairing [64] to encode location information in the name. Let x and y be the two coordinates that identify the location that we want to embed in the name. The pairing function π is a primitive recursive bijection π : N x N $\rightarrow$ N. As the cantor function only works for non-negative numbers, we shift the topology by a constant $CANTOR = 5000$ (i.e., x and y have non negative values).

$$z = \pi(x, y) = \frac{(x+y)(x+y+1)}{2} + y \tag{3.1}$$

Then, we place this value in the prefix generated by each mobile customer. The location information of a node is updated in one second intervals and represented as, e.g., "/ndn/umass/WaterSensor/z".

Each digit in z is separated by "/ ". For example, for $z = 9802$, the corresponding prefix is : "/ndn/umass/WaterSensor/9/8/0/2". This geographical location is used to make forwarding decisions. We inverse z into x and y values according to [64] by:

$$w = \frac{\sqrt{8z+1}}{2} \tag{3.2}$$

$$t = \frac{w^2 + w}{2} \tag{3.3}$$

$$y = z - t \tag{3.4}$$

$$x = w - y \tag{3.5}$$

3.2.2 Forwarding Strategy

The area in which Interests propagate is limited by the routing nodes. That is, a node forwards an incoming Interest packet only if the prefix (generated depending

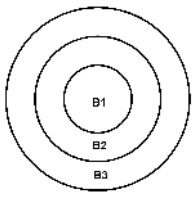


Figure 3.3: Interest propagation domains

on the customer node's location) is within the same domain. Domains are structured as shown in Fig. 3.3. As specified in Sect. 3.2.1, x and y have non negative values.

$$\frac{x_{prefix} + y_{prefix}}{r} == \frac{x_{current} + y_{current}}{r} \tag{3.6}$$

As shown in Fig. 3.3 $B1$ covers an area with radius r, $B2$ covers an area from radius r to $2 * r$, $B3$ covers an area from radius $2 * r$ to $3 * r$ and so on. In the evaluation of our approach (see Sect. 5.1.5) we use $r = 200m, 2000m$.

In our evaluation we use the following forwarding strategies:

1. **Flooding Strategy:** The flooding strategy broadcasts Interest to all nodes within its communication range. The strategy does not check the area in which the customer node is present and ignores the Interest propagation limit as shown in Fig. 3.3. It also forwards an Interest that is already registered in the PIT.

2. **WaitDist Strategy:** In our scenario, data have more value further away from their point of origin. At the same time we do not want to eliminate the possibility of a nearby mobile node to re-broadcast and to make at least some progress in data forwarding [46]. We modify the flooding strategy to include the Interest propagation limit as shown in Fig. 3.3 and induce a delay T_{gap} depending on how far the node that generated the Interest is from the current node that has received the Interest.

   $D_{\text{max}} = 150m$: approximate maximum range of radio transceiver

   $T_{\text{dist}} = 10\mu s$: as per SIFS [25]

$D_{\text{transmitter}}$: is the distance of the current node from the node that generated the Interest

$$T_{\text{gap}} = T_{\text{dist}}.\frac{D_{\max} - \min(D_{\max}, D_{\text{transmitter}})}{D_{\max}} \tag{3.7}$$

3. **WaitDistDrop Strategy:** In this strategy, we first check if the incoming Interest is already present in the PIT and if so, it is not forwarded but dropped. Otherwise, the Interest is forwarded as described in the WaitDist forwarding strategy.

4. **WaitVel Strategy:** This strategy looks at another parameter that could affect/help in disseminating Interests quicker and further away from the point of origin. This other parameter is the velocity of the mobile node. We reason that the node that takes into account the WaitDist strategy and also has greater velocity is much better suited to forward an Interest. (We assume that a node with higher velocity that is already further away from the sensor will contribute to quicker information dissemination.). Here the induced delay is T_{total}.

   $V_{\max} = 20m/s$: speed of our fastest car

   $T_{\text{vel}} = T_{\text{dist}} = 10\mu\text{s}$: as per SIFS [25]

   $V_{\text{transmitter}}$: is the velocity of the current car/node

$$T'_{\text{gap}} = T_{\text{vel}}.\frac{V_{\max} - \min(V_{\max}, V_{\text{transmitter}})}{V_{\max}} \tag{3.8}$$

$$T_{\text{total}} = T_{\text{gap}} + T'_{\text{gap}} \tag{3.9}$$

5. **WaitVelDrop Strategy:** In this case, we first check if the incoming Interest is already present in the PIT and if so, it is not forwarded but dropped. Otherwise, forwarding according to the WaitVel strategy is performed.

3.2.3 Security Concerns

We look into 3 main security concerns:

1. Attacking the nodes for their details: *i)* To obtain a particular interest a potential attacker has to be present in that region (in our case either 200 or 2000 meters) to receive that Interest packet. *ii)* To obtain the geographical coordinates the inverse function needs to be known. *iii)* Even if the geographical coordinates are obtained, the customer cannot be identified.
2. False interest injection: As described in Sect. 3.2.2, we restrict the domain in which an Interest packet with a certain prefix circulates. This requires that a potential attacker be present in that region (in our case either 200 or 2000 meters).
3. False data injection and unauthorized data retrieval: Transmitting forged Data in NDN can be detected by the customer through NDN's inherent signature mechanism and valid PKI keys are needed to decrypt the valid data.

3.3 Simulation Setup

In this section, we describe the simulation setup that we employ to evaluate our approach and to compare it with a traditional infrastructure-based IP approach. Our approach that is based on NDN is simulated using ndnSIM [52] which is based on ns3 [4]. The traditional IP approach is simulated using ns3 only.

3.3.1 Path Loss Model

To achieve a more realistic simulation of the wireless channels between the sensor and the mobile nodes the Nakagami Propagation Loss model [2] is employed. This model is a multi-path fading model and matches some empirical data better than other models. This model is a multi-path fading model and well matches empirical data.

Table 3.1: Simulation environment for NDN approach

Network Simulator	NS3/ndnSIM
Simulation Time	100s
Simulation Area	2000 m x 2000 m
Number of Mobile Nodes	10, 40, 70, 100, 130, 160, 190
Number of Sensors	1, 5
Speed of Mobile Nodes	-20, -10, 10, 20 (m/s)
Mobile Node Traffic Type	Multi-lane Bi-directional
MAC Protocol	802.11p (DSRC)
Wireless Traffic	Constant Rate 24Mbps OFDM
Loss Model	Nakagami Propogation Loss Model
Content Store Policy	Least Recently Used

Table 3.2: Simulation environment for IP approach

Network Simulator	NS3
Simulation Time	100s
Simulation Area	2000 m x 2000 m
Number of Mobile Nodes	10, 40, 70, 100, 130, 160, 190
Number of Sensors	1, 5
Number of eNB's	2
Number of PGW	1
Number of Server	1
Speed of Mobile Nodes	-20, -10, 10, 20 (m/s)
Mobile Node Traffic Type	Multi-lane Bi-directional
MAC Protocol	LTE and RS232
Loss Model	Nakagami Propagation Loss Model
Transport layer	TCP

3.3.2 NDN approach

In our simulation setup for the NDN-based approaches the mobile nodes are initially equally distributed into ten groups and in each group the nodes are separated by a distance of 20m or 40m (as per the 2 second rule[4]). These groups mimic the lane and traffic regulations as stated in the US driving rule book. As shown in Fig. 3.2, there are roads in east-west and north-south direction. For each of the four directions

[4]https://dmv.ny.gov/about-dmv/chapter-8-defensive-driving

there are two lanes. Nodes in the fast lane travel with a speed of 20m/s, while nodes travel with a speed of 10m/s in the slow lane. The intersection is modeled as an underpass and cars travel through that area without slowing down. The groups of cars are arranged in a manner that allows us to observe three different communication stages:

1. When nodes are in range of the sensor.
2. When nodes enter the range of the sensor node that can dissipate information.
3. When nodes are leaving the range of the sensor node that can dissipate information.

We evaluate different node densities (10, 100, and 190 mobile nodes) in our simulations. For better illustration, each block in Fig. 3.2 represents a group of cars. For example, in the case of 100 nodes, one block contains $\frac{100}{10} = 10$ nodes.

We further evaluate a multiple sensors scenario as shown in Fig. 3.2. In this case, we place 4 more sensors in 4 directions at a distance of 1010m as the cars move in all 4 directions and the slowest car covers 1000m in 100s.

3.3.3 IP based Approach

Currently the fastest and most popular method to disseminate environmental/traffic data is by using cloud servers [88], to which the data is pushed from the sensor and various clients pull the required data from one or more servers. In this scenario a quite complex topology is employed. Both, the sensor and the mobile nodes have to use wireless, cellular technology to communicate with base stations. In our scenario we assume LTE [65] as the cellular wireless technology. We also assume that the eNB's are femtocells, since they provide better connectivity (e.g., the UE needs less power to connect to the base station [33]). The eNB is connected to the Gateway (PGW) through RS232 (point-to-point) links, which connect it to the Internet. Such

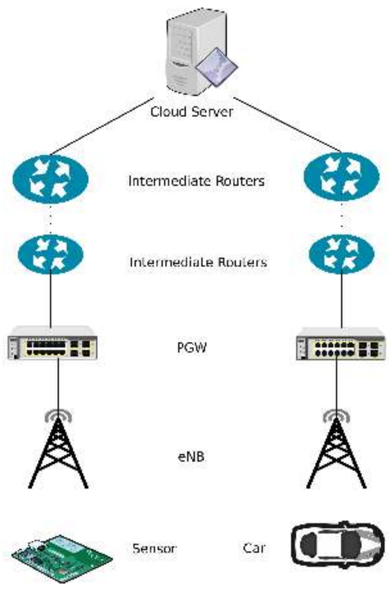


Figure 3.4: Real LTE TCP/IP based Architecture scenario

a setup usually results in 6-7 hops between PGW and the Cloud server. The overall topology for that scenario is shown in Fig. 3.4. Compared to our NDN approach this solution introduces significant delay and the fact that LTE requires a femtocell in range of the end nodes poses a scalability and deployment issue.

For simplicity we have implemented the topology shown in Fig. 3.4 with a direct link between the PGW and the Cloud server, since the exact number of routers between the two is unknown. In our simulations, the delay on that link is set to 10ms. Therefore, the setup we have chosen for the LTE scenario represents the best possible scenario. Thus, a data packet travels from the sensor via the LTE eNB to the PGW. From there over the wired Internet to the cloud server. From the cloud server it travels via the wired Internet to the PGW that serves the mobile node, and from there over the LTE eNB to its final destination.

The topology at the end eNB which connects the multiple mobile clients, is similar to the one we used in our NDN approach and, the eNB's position can be considered analogous to the one of the sensor, i.e. the nodes in the radio range of eNB can establish a TCP connection to the Cloud server.

We make use of the TCP on-off helper (tcp connection on for 10 seconds then off for 1 second) [52] to generate the traffic from the sensor to the cloud server and form cloud server to the mobile clients.

3.4 Evaluation

The main goal of our evaluation is to investigate how quick the approaches presented in Sect. 3.2 disseminate warning information from a roadside sensor to a group of vehicles in the vicinity of that sensor. We compare the time required for the Interest to be satisfied (delay) for the strategies mentioned in Sect. 3.2 for node densities of 10, 100, and 190.

In the case of LTE we are taking into consideration the best case scenario possible, i.e., delay between packet transfer from sensor to cloud server and from cloud server to mobile node, but ignore the delay introduced by intermediate routers and processing at the web server. Whereas in the NDN case we have taken into account all possible delays (accessing the cache, PIT, and FIB and the actual communication delays) as provided by ndnSIM [52].

Figure 3.5 shows the case of 10 nodes with a CS size of 100 entries and a 2000m domain, where just one node occupies each of the four lanes shown in Figure 3.2. In this scenario, all of the timer-based forwarding approaches perform similar. Due to the low mobile node density, Interests arriving at the nodes are not already logged in the PIT and drop-based and non-drop-based strategies behave similar in this scenario. Figure 3.5 shows that 63% of the vehicles have received the warning message after $\sim 5000\mu s$ in the case of *WaitVelDrop*. In the infrastructure-based LTE scenario it takes $\sim$ 100 times longer to transmit the warning message to the same fraction of nodes. For example, in the LTE cases 9.09% of the nodes have received the warning message after a delay of .2045 seconds, whereas 33.75%, 65.1%, and 66% of the the nodes have received the message in the case of flooding-, distance-, and velocity-based

NDN strategies. Note that the LTE case is much more vulnerable to failures in the infrastructure (which is often a side effect in disaster scenarios), but such effects are not regarded in our simulation study.

Comparing these results to scenarios with higher node densities (100 nodes shown in Fig. 3.6 and 190 nodes shown in Fig. 3.7) shows a slightly better performance of the drop-based strategies. As shown in Fig. 3.6, 1.98% of the nodes have received the warning message in the LTE case after a delay of .177 seconds. In the case of the NDN-based strategies, 26%, 38.5%, 40.8%, 43.6% and 45.5% of the nodes have received the warning message in the same time span. For the case of 190 nodes (Fig. 3.7), 5.23% of the nodes received the warning message in the LTE case after .195 seconds. In the NDN case, 24.26%, 42%, 38.5%, 45.5% and 44% of the Interests are satisfied for the strategies strategies presented in Sect. 3.2.2. This is caused by the fact that entries in a mobile node's PIT for arriving Interests might already exist. In general, higher node densities result in increased message delivery delay due to increased collisions and retransmission. Nevertheless, even in this high node density scenarios the timer-based strategies outperform flooding and LTE.

A comparison with a much smaller domain of 200m (see Fig.3.8) shows a significantly reduced message propagation delay. In this scenario, nodes are much closer to each other and an Interest send by one of the mobile nodes can be received by multiple other mobile nodes leading to a faster Interest and also Data propagation. Reducing the domain size has no significant impact on flooding and LTE. As shown in Fig. 3.8, 9.09% of LTE packets are received after a delay of .2045 seconds, whereas 33.8%, 89% and 91% Interests are satisfied for flooding-, distance- and velocity-based strategies in the same time interval.

Finally, we were interested to what extent the CS size of the mobile nodes impact the message propagation delay. For this evaluation we chose a scenario with 100 mobile nodes, a CS of 1000 (compared to 100 in the earlier simulations), and a

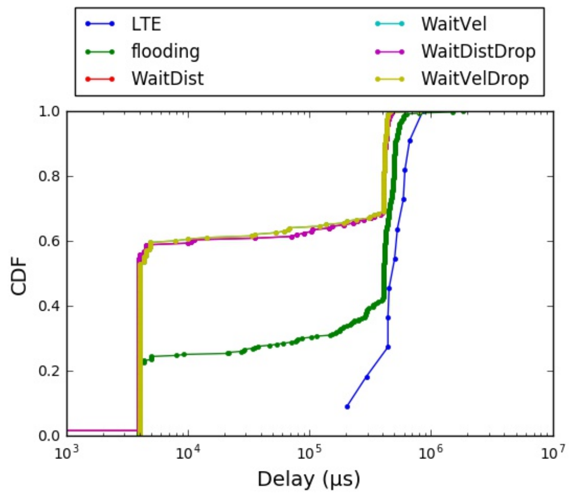


Figure 3.5: 10 nodes with 100 CS and 2000m domain

domain size of 2000m. Comparing the results shown in Fig. 3.9 with the ones in Fig. 3.5 reveals that increasing the CS size does not shorten the message propagation delay significantly. We conjecture that this is because only one type of message is transmitted and we do not simulate any competing traffic. In future work, we plan to perform simulations with cross-traffic to evaluate the impact of larger CSs. It can also be observed that all timer-based strategies behave almost similar.

A comparison between various node densities that use the pure flooding approach, as shown in Fig. 3.10, reveals that the delay proportionally increases with the node density until 60% of the Interests are satisfied. We can see a significant difference between the scenario which has 10 nodes (1 node per lane per direction) and the others. The other node density scenarios perform quite similar.

Whereas in the LTE case as shown in Fig. 3.11 the delay to send all packets (to achieve CDF 1) keeps increasing proportional to the density. The shape of the plots for all densities is similar and the plots are only slightly shifted in time. This property can be attributed to the fact that there is no in network caching and the cloud server need to transfer packets to all the nodes.

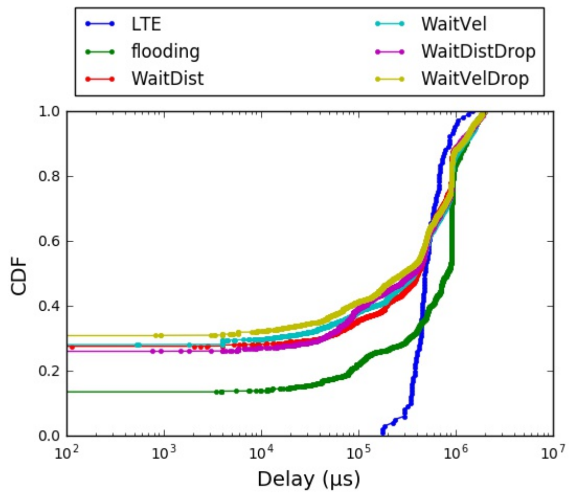


Figure 3.6: 100 nodes with 100 CS and 2000m domain

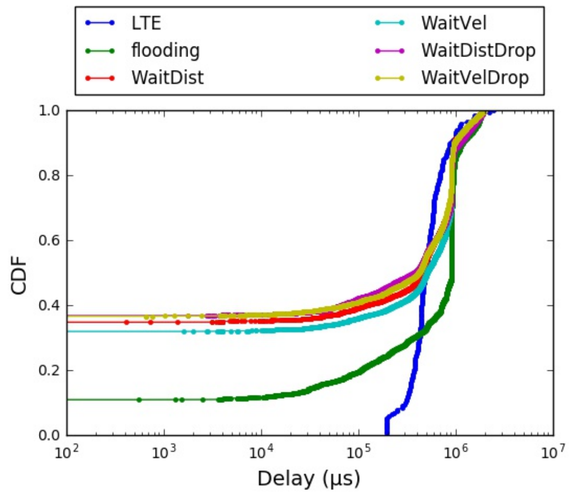


Figure 3.7: 190 nodes with 100 CS and 2000m domain

We further explore a scenario with multiple sensors as described in Sec. 3.3.2. For the LTE case, we calculate the delay by taking into account the TCP connections from all the sensors to the cars as we want the cars to get the information about the closest sensor (i.e., 5 in this scenario). Simulation results from this scenario are shown in Fig. 3.12 (100 vehicles) and Fig. 3.13 (10 nodes). Both results show that the LTE-based approach performs significantly worse than the ndn approach. By the time 90% of the vehicles have received the warning messages in the ndn case

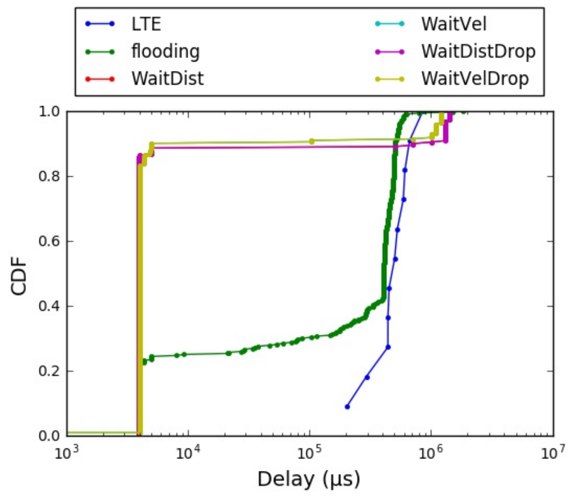


Figure 3.8: 10 nodes with 100 CS and 200m domain

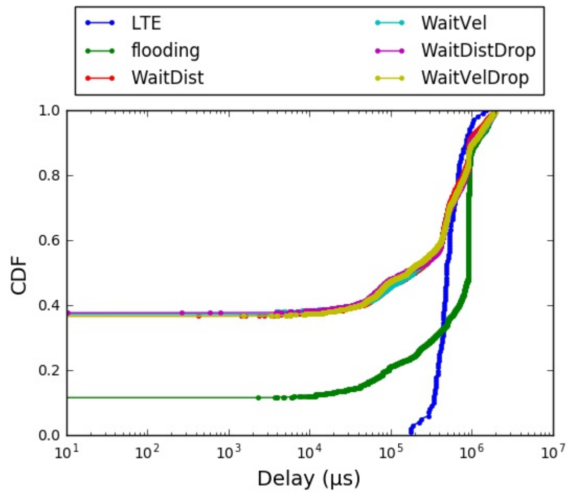


Figure 3.9: 100 nodes with 1000 CS and 2000m domain

only 9% have received the messages in the LTE case. The delay in the LTE case increases proportionally to the number of sensors. This demonstrates the scalability of ndn-based approach.

3.5 Discussion

One significant point that is not shown as part of our simulation is the increased resilience of our approach compared to an infrastructure-based approach. In our

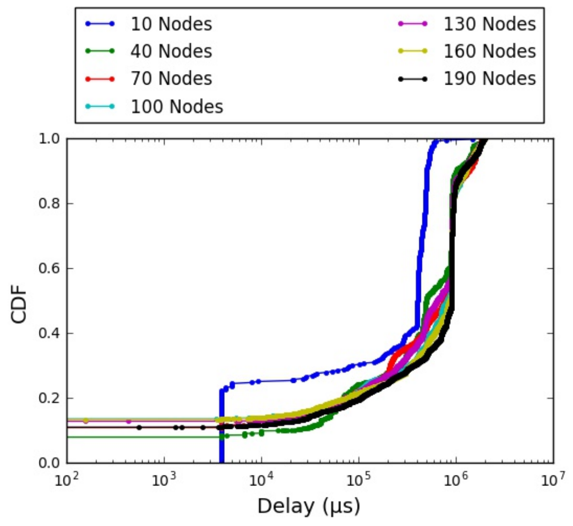


Figure 3.10: Case of flooding with 100 CS and no domain limit

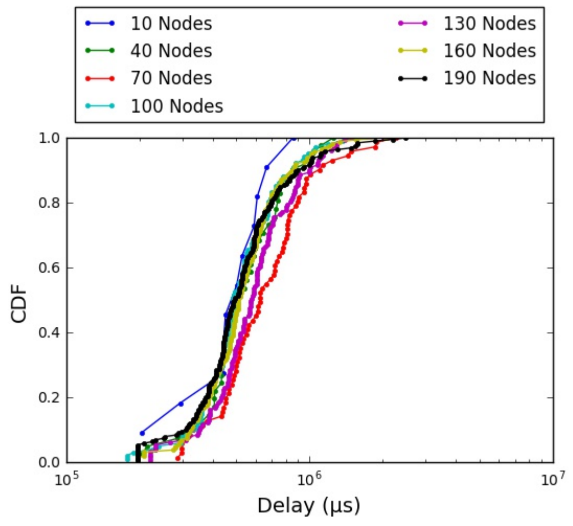


Figure 3.11: Case of LTE

NDN-based approach sensor node and mobile nodes can communicate without any additional devices. Thus, only the failure of the sensor node will prevent the dissemination of warning messages. The failure of one or more mobile nodes will lead to delayed message propagation but not the complete failure of warning message dissemination.

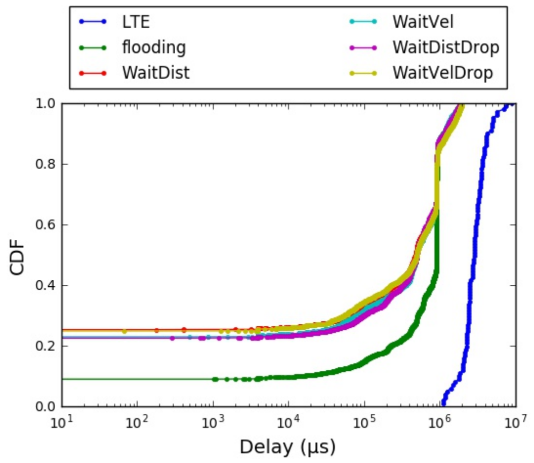


Figure 3.12: 5 sensors and 100 nodes with 100 CS and 2000m domain

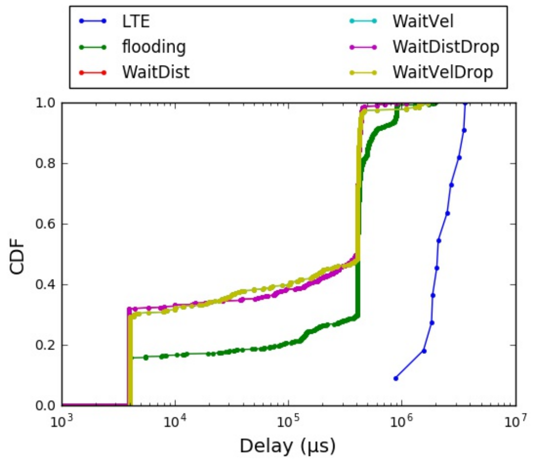


Figure 3.13: 5 sensors and 10 nodes with 100 CS and 2000m domain

This is quite different in scenarios that heavily rely on infrastructure like the one based on LTE we used for comparison in our evaluation. Works like the one presented by Schulman and Spring [72] have shown that disasters can significantly impact the communication infrastructure.

3.6 Conclusion

In this chapter, we present a ICN-based approach for the dissemination of sensor information in disaster scenarios to vehicles. We have chosen this approach since it minimally relies on infrastructure. We present different forwarding strategies for Interests and Data forwarding between sensor and cars and evaluate their performance. In addition, we compare this with an infrastructure-based LTE approach. Results we obtained through simulations show that our approach performs in most cases better than one that heavily relies on infrastructure. In addition, the timer based approaches outperform if not equal the flooding technique. In future work, we plan on creating a framework that will also be able to simulate infrastructure failures to further evaluate the performance of our approach.

CHAPTER 4

TRAFFIC OFFLOADING VIA INFORMATION-CENTRIC NETWORKING MOBILE CLOUD

4.1 Introduction

The ever increasing popularity of smartphones and other mobile devices allows users to access a variety of services such as video on demand, online banking, and social media virtually anywhere in the world. Smartphone technology and wireless networks also play a key role in emergency and disaster scenarios, where first responders need to communicate among themselves, command posts, and the public to perform emergency management tasks (e.g., to perform triage in the case of many significant injuries). With the current infrastructure reaching its capacity limits in dense urban scenarios, it is fundamental that communication in the case of an emergency can be performed in a timely and reliable manner.

The evolution of Radio Access Networks (RAN) is mostly focused on increasing capacity and reducing cost for network operators. For instance, the Third Generation Partnership Project (3GPP), the body responsible for cellular network standards, adopted carrier aggregation and multiple input multiple output (MIMO) technologies in the LTE standard [11] that specifies data rates on the order of 300 Mbps. This increase in capacity has the goal to cope with the rapid increase in demand from users. This demand is predicted to increase 7-fold by 2021 on top of an 18-fold increase from 2011 to 2016 [10]. Consequently, mobile network operators (MNO) have been deploying more base stations and femtocells to accommodate this increase in traffic, an approach that increases complexity, cost, and management for the MNO.

To alleviate the infrastructure, the 3GPP formalized in releases 13 and 14 the LTE Licensed Assisted Access (LTE-U/LAA/eLAA), that allows the coexistence of LTE in unlicensed 5 GHz ISM-bands to expand the capacity of current networks. While these solutions increase network capacity in response to the rising user demand, their increasing complexity makes them more vulnerable in the face of disasters. The recent events in the aftermath of Hurricane Maria on the island of Puerto Rico have demonstrated this in a shocking manner.

We believe that Device-to-Device (D2D) communication has the potential to address the challenge in providing extra capacity to the edge of the network, while reducing capacity requirements at the core. Moreover, it increases reliability in disaster scenarios, where low-capacity networks are usually deployed to maintain a minimum level of connectivity with emergency services. For instance, the authors in [32] analyzed network data during the 2014/2015 floods in Malaysia and Indonesia, finding that the signal quality from the available base stations deteriorates while users tend to use more WiFi networks when available. However, a reliable D2D communication is challenging as nodes can enter and exit the communication range at any time, breaking end-to-end paths and altering routing state. Additionally, in larger ad hoc networks, nodes are subject to hidden terminal problems.

We present a scheme that combines Information-Centric Networking (ICN) [48, 42] with D2D communication to offload traffic from cellular networks, allowing users to communicate even without the aid of wireless infrastructure. In ICN, data is immutable and decoupled from its location, enabling a node to fetch content from any other node in the network that has a cached copy of the requested data. In-network caching is supported as part of the architecture. Furthermore, ICN supports multipath communication and prevents loops. These characteristics make ICN more tolerant to delay and disruptions than host-centric architectures. The work presented

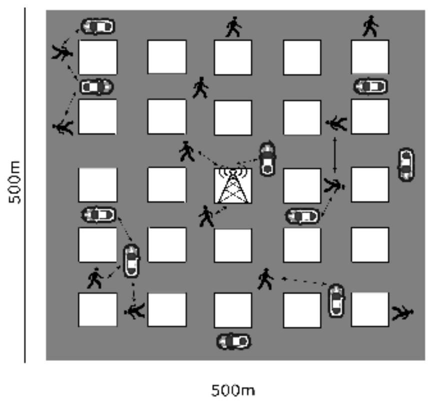


Figure 4.1: Manhattan-Grid mobility model. Nodes (cars and pedestrians) can only move along the gray roads.

in this chapter is based on Named-Data Networking (NDN)[89], one of the many flavors of ICN.

In this chapter, we assume that mobile nodes have two wireless interfaces (e.g. Wi-Fi and cellular) which is the case for modern smartphones. Therefore, when a node sends out a request to fetch data, it first queries nearby devices for a cached copy. If the request times out, the request is retransmitted to the cellular network. We evaluate our approach through simulations using NS-3 and ndnSIM [53].

4.2 Model and implementation

Our application scenario is focused on data dissemination in urban environments, including pedestrian and vehicular nodes. We assume that all nodes have at least two wireless interfaces, one WiFi and one LTE, and that they are willing to join the MANET, and share storage space for collaborative caching.

In our first scenario, nodes move along a Manhattan grid (shown in Figure 4.1) generated using BonnMotion [20], a widely used mobility generation tool. We implement two other scenarios: the vehicular cloud proposed in [81] and a pedestrian crowd (e.g., concerts or sports events). For evaluation, we use the ndnSIM simulator [53], a NS-3 based NDN simulator.

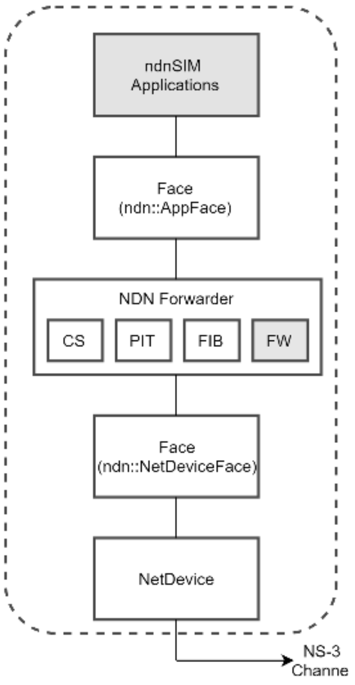


Figure 4.2: NDN-Node block diagram with the modified blocks in gray.

We customized the NDN forwarder to meet our application scenarios as follows (a complete view of the node structure and modified blocks is shown in Figure 4.2). First, when a node sends out an Interest request to the network, it attaches a retransmission tag (*Retx tag*) to the outgoing packet. The Retx tag informs the forwarder whether the packet is a retransmission or not. I.e., if the desired content was not found in the MANET within one timeout period, it sets the Retx tag. All retransmitted Interest packets are forwarded directly to the cellular network. We use the hop count tag (*HopCount*) to identify if a node is the originator or a forwarder. If a node is a forwarder, it may drop packets based on its energy level, which we will described next. This approach is motivated by the fact that energy might be scarce in disaster scenarios due to power outages. In such a case, users might not be willing to deplete there battery below a certain threshold.

Considering the energy consumption of the nodes, we create two thresholds where nodes change their forwarding behavior based on the current energy level. The first threshold, E_{th_1}, is at 35% battery level and the second threshold, E_{th_2}, is at 25%. These thresholds were chosen based on a common Li-Ion battery discharge curve [79]. At the beginning of each simulation, each node is randomly assigned an initial energy level ranging from 5% to 100%. When the energy at a certain node falls under E_{th_1}, it stops forwarding packets with hop count greater than three. In our initial tests, we found that more than 90% of the packets retrieved from the MANET come from a range of 3 hops or less. Therefore, this threshold aims at saving energy by reducing a packet's reachability while still being able to serve the majority of contents. Furthermore, when the energy level falls below E_{th_2}, the node leaves the MANET by forwarding all Interest packets to the cellular network. The described behavior is formalized in Algorithm 1.

Algorithm 1: Modified behavior of NDN forwarder

```
   Input  : Interest packet
   Output: Interface to forward
1  if $HopCount = 0$ then
2      if $E_n \leq E_{th_2}$ then
3          return LTE;
4      else
5          return WiFi;
6      end
7  else
8      if ( $E_n \leq E_{th_1}$ and $HopCount \leq 3$ ) or
9      ( $E_n > E_{th_1}$ and $HopCount \leq 6$ ) then
10         return WiFi;
11     else
12         return Drop;
13     end
14 end
```

For the consumer application, we gathered real data from Twitter's trending topics in the United States to create our content request pattern.[1] Through maximum likelihood estimation, we found that the popularity in our dataset follows a Zipfian

[1]https://developer.twitter.com/en/docs/trends/locations-with-trending-topics/api-reference

distribution with $\alpha = 0.7$ and $q = 0.7$, described by Equations 4.1 and 4.2. Other works have characterized Internet content to follow the Zipfian distribution as well, among them [14, 23]. The main assumption is that some tweets will reach more users than others, which is the case with popular accounts from public figures (usually millions of followers) and regular accounts (a couple dozen to hundreds of followers).

$$f(k; N, q, \alpha) = \frac{1/(k+q)^{\alpha}}{H_{N,q,\alpha}} \tag{4.1}$$

$$H_{N,q,\alpha} = \sum_{i=1}^{N} \frac{1}{(i+q)^{\alpha}} \tag{4.2}$$

Our content catalog (N) is composed of a total of 1,000 contents. Each node requests an Interest every 20 ms, and can cache contents varying from 0.1% to 10% of the content catalog. We simulate three scenarios where the infrastructure is being challenged by the rapid increase in traffic and it would benefit from D2D communication in the case of a crisis.

4.2.1 Urban Scenario

This scenario resembles an urban environment where nodes can only move along the grid (streets). Due to the large number of users, MNOs turned to small cell densification to cope with the increase in network traffic. For our simulations, half of the nodes are pedestrians (moving at 1 m/s) and half are vehicles (moving at 13 m/s). Node count and simulation area were calculated using the framework in [49]. There is one base station located at the center of the grid that is used only when end-users cannot fetch the requested content from neighboring nodes. Table 4.1 summarizes the simulation parameters.

Table 4.1: Summary of simulation parameters for scenario I

Parameter	Value
Node count	25, 50, 100, 150
Area	500 x 500 m^2
Access technology	IEEE 802.11g and LTE
Communication range	Wi-Fi: 100 m, LTE: To base station
Node cache (CS)	1, 5, 50, 100 kB
Cache Policy	LRU
Data payload	1 kB
Total contents	1,000

4.2.2 Vehicular cloud

Following the findings in [81], we implemented the concept of a vehicular cloud where utility or emergency vehicles serve as data mules to assist end-users in fetching content. In our implementation, nodes move according to the Manhattan-Grid model. First, the vehicular nodes randomly request contents from the backhaul network to fill up their caches (i.e., a node will sequentially request as many contents as it can store in its cache, with the first requested content being randomly selected), then consumer nodes request contents from the vehicular cloud. If that request times out, pedestrians retransmit to the cellular network. In this scenario, the only communication allowed is device-to-vehicle and device-to-infrastructure. We vary the number of vehicles from 25 to 100 nodes in increments of 25, while the number of pedestrians remains static at 25 nodes. Moreover, we evaluate the effects that different cache sizes on the vehicular node has on the network. The expectation is that as the number of vehicular nodes increase, pedestrians will be able to fetch more contents from them. Similarly, as vehicle cache size increases, the pool of contents available to pedestrians will be greater.

4.2.3 Large crowds

Another possible application for our model is large crowds, where often the influx of people exceeds the capacity of the infrastructure. In [56], the authors reported that terabytes of data were transfered via cellular networks (AT&T, Verizon, and Sprint) by in-stadium fans during the 2015 Superbowl. We develop a scenario where spectators at an arena are able to watch on-demand replays, reducing the load on the infrastructure. Our scenario comprises of 200 users in one section of a stadium. Users can communicate with the LTE base station (in this case a femtocell) as well as other users in the same stands to fetch contents. The replay videos have a total duration of 20 seconds each, segmented into 2-second chunks (following the MPEG-DASH standard [73]). The chunk size is 100 kB. We leave the question of fetching different quality levels for future work. In total, five replay videos are requested by all users, with the video segments being requested in order. However, each user will start requesting the videos at a random time within a short interval (10 seconds). The reason for this interval is two-fold: in reality, not all spectators request videos at the same time; second, by using slightly different times, nodes can take advantage of caching.

4.3 Simulation Results and Evaluation

We first analyze the Manhattan-Grid scenario (Section 4.2.1). Figure 4.3 shows the average percentage of traffic that was successfully offloaded from the cellular network with different node densities and different cache sizes (CS). The remaining requests that could not be satisfied from the MANET were served by the cellular network. The error bars are the standard deviation of ten runs. According to Figure 4.3, our model can offload up to 51.7% of traffic from the infrastructure. We attribute this result to the combination of inherent in-network caching in NDN (enabled by named-content) and the Zipfian request pattern described earlier. We can also observe a higher variation in lower density scenarios. This is due to the social proximity of the nodes, i.e., in low densities, nodes are more susceptible to the presence of neighbors, as for higher densities the pool of content from neighbor caches is greater. Additionally, we see an increase in the traffic being offloaded as the density increases. This can also be attributed to the higher social proximity in higher densities (more neighbors translate to a greater set of contents to choose from). Figure 4.3 confirms our expectation that the network performance increases as cache size increases.

In applications that are more tolerant to a higher latency (e.g., where the infrastructure is partly unavailable), the fraction of offloaded traffic can be improved by increasing the number of retries injected in the MANET. Figure 4.4 shows the percentage of traffic that is offloaded when the consumer application rebroadcasts the Interest request to the MANET, instead of sending it directly to the cellular network. Our experiments show improvements of up to 16.61% (68.31% offload in total) when resending up to four requests to neighboring nodes before sending it to the cellular network.

Figure 4.5 depicts the PDF of data packets' hop count in all node densities when the cache size per node is 0.5% of the total content catalog. We can see that in all cases, less than 5% of contents come from the nodes' own cache, while the majority of contents comes from nodes that are three hops away. It is important to note that as MANET grows, the PDF becomes wider. We do not limit the propagation of data packets (as we do with Interest packets); therefore, in some occasions, the data packets travel multiple hops back to the consumer node before the Interest timeout expires. Figure 4.6 shows the CDF for latency for different node densities. We compute latency as the elapsed time from the first Interest requested by a node until the data packet returns, either via the MANET or via the cellular network. We also distinguish between vehicular and pedestrian nodes to study the effects of mobility on latency; however, the graphs do not show a significant difference in latency between pedestrian and vehicular nodes with both curves overlapping in most cases. Figure 4.6 also shows that in approximately 80% of the cases the latency is below 50 ms. The retransmission timeout in NDN varies according to the number of satisfied (timeout decreases) and lost requests (timeout increases); thus, the tail in the CDF in the MANET cases. The LTE CDF shows the sum of the first timed-out request plus the

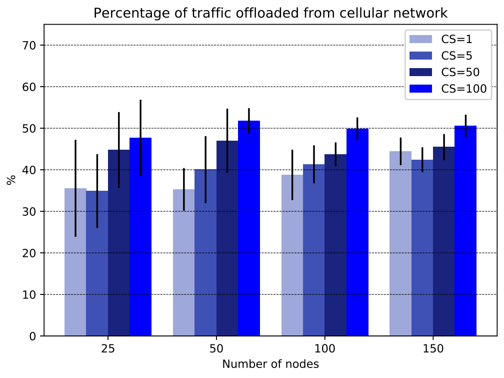


Figure 4.3: Percentage of traffic offloaded via the MANET, illustrating the effect of varying cache size and node density.

retransmitted request to the cellular network. Moreover, the shallower slope in the LTE case reflects the variation of the Interest timeout in the MANET.

We also evaluated the energy consumption on the nodes. Our simulation results showed that nodes below the first ($25\% < E_n \leqslant 35\%$) and second ($E_n \leqslant 25\%$) threshold consume 5.8% and 2.9% less energy than other nodes, respectively. Nodes that are in the low power range have a steeper drop in the energy levels due to the battery discharge curve. This energy saving approach provides a longer battery life for mobile user equipment, which is specially important when the infrastructure is impaired (e.g. power outages).

The vehicular cloud case (Section 4.2.2) is presented in Figure 4.7. The *x-axis* represent the number of utility vehicles, while the bars show the average percentage of traffic that was offloaded from the cellular network for different cache sizes used in the simulation. The error-bars represent the standard deviation of ten runs. Figure 4.7 shows the effect of increasing the number of vehicles, increasing the cache size, and a combination of both. As expected, increasing the number of vehicular nodes that cache content for the MNO has a positive impact on the amount of traffic that can be offloaded from the LTE network as more cache space becomes available. Similarly, increasing the cache size on the available nodes also has a positive impact on the

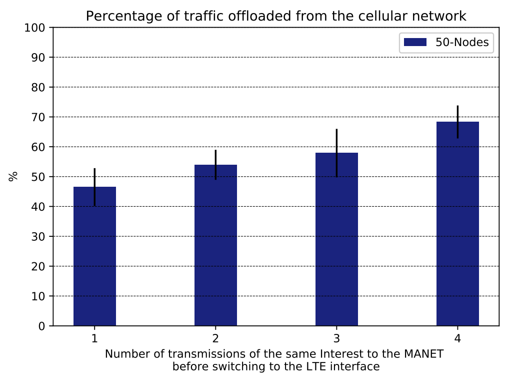


Figure 4.4: Percentage of traffic offloaded from the cellular network when the consumer application retries to send Interest requests to the MANET. The figure shows up to 4 transmissions of the same Interest to the MANET for the 50-Node case.

offloaded traffic. It is important to note that increasing the number of vehicles is more effective than increasing the cache size on the vehicles, suggesting that the network benefits from more nodes joining the MANET, but without the necessity of having large caches. This result can be explained because increasing the number of vehicles increases cache diversity in the network, since end-users benefit from the short contact time (when communication between vehicles and consumers happens) with the utility vehicles. Furthermore, our simulation results are aligned with the findings in [81], where the authors experienced an offload ratio of approximately 50% in their vehicular cloud.

The third scenario (Section 4.2.3) focuses on large crowds, where spectators at a sports event have the capability to watch instant replays at their mobile devices. Figure 4.8 (a) shows the percentage of traffic offloaded by the MANET, as well as the percentage of the video that was downloaded for different request rates. Figure 4.8 (b) shows the average and standard deviation of latency to fetch one segment. According to Figure 4.8 (a), 10.3% and 16.59% of traffic was offloaded by the MANET for request rates of 1 and 5 Interest/s, respectively, showing that we can alleviate traffic from the backhaul network. It is important to note that as we increase the request

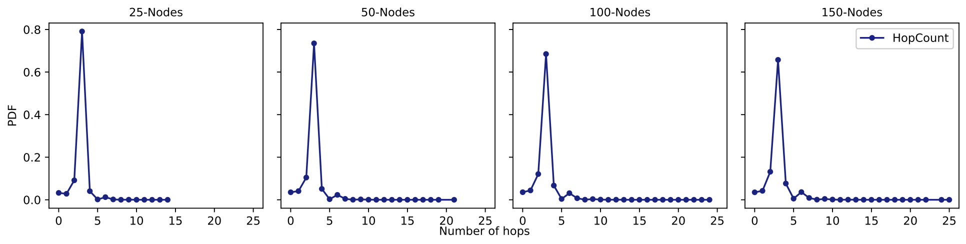


Figure 4.5: Number of hops from successfully retrieved data packets.

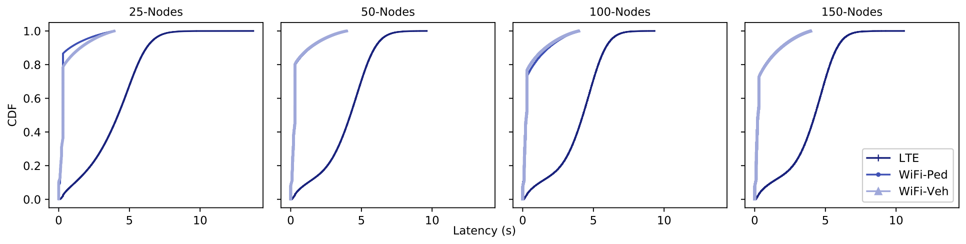


Figure 4.6: Cumulative distribution function of latency for different node densities.

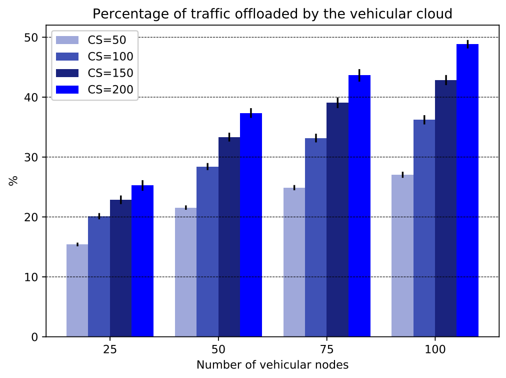


Figure 4.7: Percentage of traffic offloaded by the vehicular cloud for different node densities and cache sizes.

rate, the network saturates and we are no longer able to download the entire video, leading to re-buffering. Moreover, we see a lower offload traffic compared to the two urban scenarios, which reflects the sequential request pattern used in this case (as opposed to the Zipfian pattern used previously). Figure 4.8 (b) shows the average and standard deviation of latency for each 2-second video segment. In all cases the latency remained under 1.2 seconds and decreases as the request rate increases, suggesting a

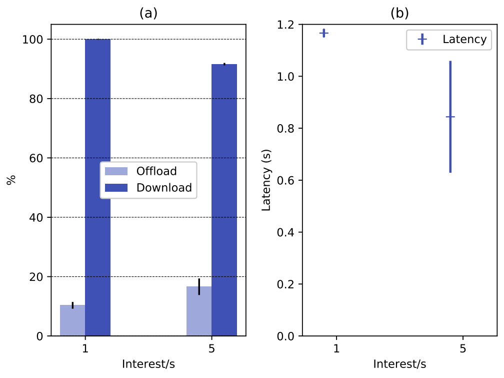


Figure 4.8: Percentage of traffic offloaded, video download completion (a), and latency (b) for 200 nodes in sports events.

smooth playback for lower request rates. This decrease in latency reflects the variation in the retransmission timeout (RTO), that reduces as more packets that are in flight are served by the producer or neighbor caches.

All of the above results are also available at the project website.[2]

4.4 Conclusion

This work proposes and evaluates a custom ICN design tailored to environments where the infrastructure is being pressured by the surge in traffic, while increasing reliability in disaster scenarios. Our forwarding strategy seeks to empower D2D communication so users can download contents directly from the MANET, easing the burden on the wireless access network as well as the core. We assessed our model in two urban scenarios using real data from social media to create our request pattern, and one indoor in-stadium scenario where users can view on-demand replays directly on their devices. Our simulation results showed an improvement of up to 51.7% in traffic being offloaded from the cellular network in the urban scenario. Moreover, our energy saving approach reduces the average consumption by up to 5.8% com-

[2]http://people.umass.edu/tteixeira/icn-adhoc.html

pared to the normal operation, despite the steeper decrease in the battery discharge curve towards the end of its cycle. In the in-stadium scenario, our approach successfully downloads 20-second replay videos where 10.3% of the contents are fetched from nearby devices. We believe that offloading traffic from next-generation cellular networks (5G and onwards) is a promising solution to accommodate the traffic generated from new applications without increasing the complexity and capacity for the MNO. Additionally, an ICN D2D communication has the potential to play a key role in disaster scenarios where the infrastructure is impaired.

CHAPTER 5

IMPROVING QOE AND RESILIENCE OF ABR STREAMING

5.1 Improving QoE of ABR Streaming Sessions through QUIC Retransmissions

5.1.1 Introduction

After two decades, the HyperText Transfer Protocol has undergone a significant makeover resulting in the introduction of the HTTP/2 standard [75] gaining notable popularity in the Internet, where it is currently used by 24.6% of all web sites [8]. HTTP/2 makes several improvements over its predecessor HTTP/1.1 [29]. These improvements include *a)* multiplexing, where streams for multiple requests can be sent over a single TCP session; *b)* header compression; and, *c)* an option where the web server can push content to the client proactively. HTTP/2 has been specified to use TCP as the underlying transport protocol. This combination of HTTP/2 and TCP has several performance issues, including a delay introduced by the 3-way handshake for each connection setup (this is even higher if Transport Layer Security (TLS) is used). In addition, the issue of head of line (HOL) blocking still exists. The Quick UDP Internet Connections protocol (QUIC) [50] is a new approach designed to combine the speed of UDP with the reliability of TCP and, thus overcome these issues. QUIC has been specifically designed to reduce latency of web page loads and mitigate rebuffers in video streaming clients.

Adaptive bitrate (ABR) streaming has become the de-facto streaming standard for video on demand platforms such as Netflix [3] and Youtube [9]. With more than 70% of the peak hour US Internet traffic [80], video streaming has become *the* killer-application of today's Internet. ABR video streaming solutions like Dynamic Adaptive Streaming over HTTP (DASH) [74] are, however, stuck in an HTTP/TCP setting that has been shown to possess substantial drawbacks with respect to Quality-of-Experience (QoE) [40, 83].

Recently, we proposed a DASH-based ABR approach (SQUAD) [83] that has the goal to improve QoE for viewers watching video streams over the Internet. One specific feature of SQUAD is the ability to retransmit segments[1] in a higher quality than they were originally transmitted in [84] to reduce frequent quality changes during a streaming session. The drawback of implementing this approach on top of HTTP/1.1 is the inability to efficiently schedule such retransmissions. In the case of one TCP session, retransmission requests[2] have to be interleaved with requests for new original segments, that have not been requested in the past. Parallel transmissions require the setup of a new connection, which comes with the drawback of additional delay due to the 3-way handshake. The use of HTTP/2 over TCP makes such retransmissions more efficient, since they can be scheduled within the same TCP connection. While HTTP/2 has the potential to improve the performance of SQUAD in the case of retransmission, the impact of losses and the resulting HOL blocking has not been studied. In addition, it has not been evaluated to what extent QUIC can further improve SQUAD with retransmissions, since it eliminates the HOL blocking issue.

5.1.2 Segment retransmission scheduling

Traditional ABR approaches stream the ABR video segments in the order provided by the MPD file. Looking closely at the segment qualities buffered at the client at

[1]In the remainder of this chapter, we use the word segments to denote ABR video segments unless specified otherwise.

[2]In the remainder of this chapter, we use the word retransmissions to denote retransmissions of a received video segment in a higher quality unless specified otherwise.

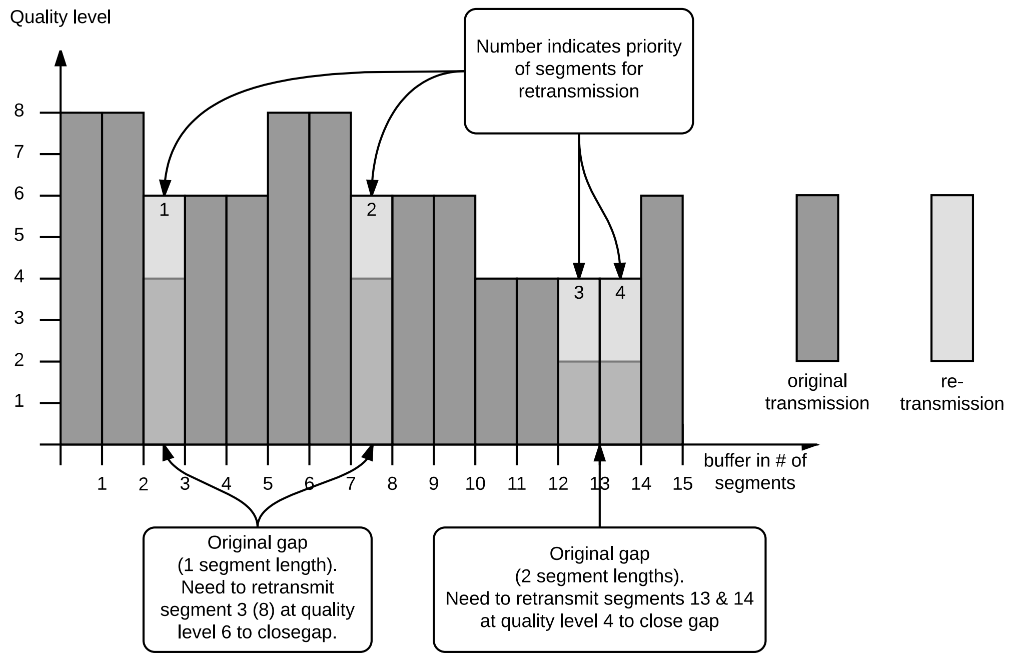

Quality level
8
7
6
5
4
3
2
1
Number indicates priority of segments for retransmission
1
2
3
4
original transmission
re-transmission
buffer in # of segments
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Original gap (1 segment length). Need to retransmit segment 3 (8) at quality level 6 to closegap.
Original gap (2 segment lengths). Need to retransmit segments 13 & 14 at quality level 4 to close gap

Figure 5.1: Example scenario for retransmissions. The QoE of this streaming session can be improved if, e.g., segments 3, 8, 13, and 14 are retransmitted in higher quality, assuming they arrive before their scheduled playout.

any point in time SQUAD shows that these reflect the recent quality decisions made by the adaptation algorithm, which, in turn, are based on the specific interpretation of the measured download rate and the corresponding buffer filling. Looking at the buffer filling in *retrospect* as in Fig. 5.1 (a) SQUAD identifies quality switches that are denoted as quality “gaps”. The emergence of these quality gaps is complex as it describes the instantaneous interaction of the adaptation algorithm with the buffer filling state and the download rate. In the following, we illustrate how to improve the QoE by filling some of these quality gaps. Fig. 5.1 shows a simplified example of segment qualities inside the player buffer with different possible gaps. SQUAD defines gaps as the downward variation from the quality level which negatively impact the QoE [91].

SQUAD’s current approach is based on HTTP/1.1, which does not allow the parallel transmission of original segments and the retransmission of segments in a better quality. HTTP/2 over TCP allows this parallel transmission but does not prevent HOL blocking to efficiently perform retransmissions. In contrast, HTTP/2 over QUIC does not suffer from such inefficiencies and gives the application maximum control of individual streams and we show how this can be used to improve the QoE of ABR streaming.

5.1.3 Analysis of Gaps in Streaming Sessions

Akamai [59] is the world's largest CDN provider that delivers 15%–30% of global Internet traffic. Its CDN contains over 150,000 edge servers distributed in 90+ countries and 1200 ISPs around the world. To motivate the retransmission of segments as described in Sect. 5.1.2, we analyze an anonymized trace collected from Akamai's video CDN. This trace contains video streaming session information for a 3-day period in June 2014. The ABR streaming traffic in this trace contains 5 million video sessions originating from over 200,000 unique clients who were served by 1294 edge servers around the world. For each streaming session, each individual segment request is logged, which allows us to reconstruct the quality of the segments received at the client. Fig. 5.2 gives an example for one such streaming session we randomly picked for better illustration. As shown in Fig. 5.2, this streaming session resulted in a series of gaps. These gaps are potential candidates for segment retransmission that could lead to less quality level changes and, thus, an improve QoE. In Fig. 5.2, we indicate that a retransmission of the segments in the later part of the stream could significantly impact the QoE.

In Fig. 5.3, we show the results of our analysis for the complete data set which has approx. 5 million sessions, and for the a subset that only includes sessions for mobile devices, which has approx. 0.1 million sessions. This figure shows the percentage of sessions that have one or more gaps. Considering all sessions in the data set, 36.19% of the sessions have at least one gap. These sessions could benefit from our segment

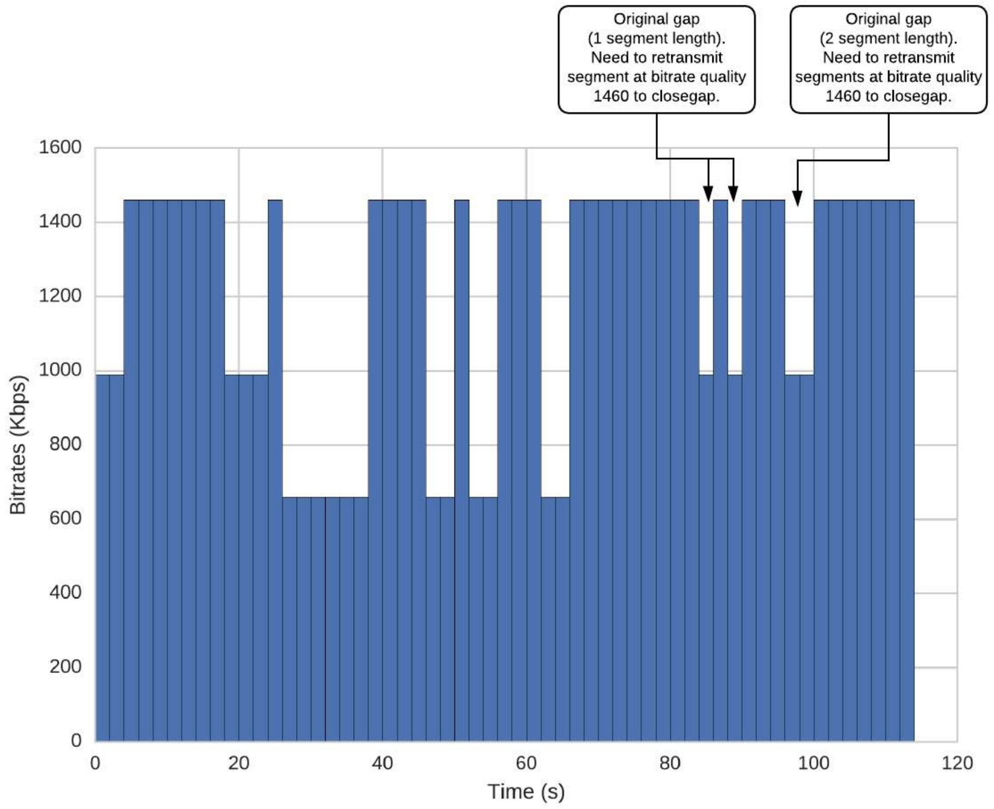


Figure 5.2: Original transmission of video stream from one randomly selected trace in the Akamai data set. The QoE of this video can be improved if the highlighted segments are retransmitted in higher quality, assuming they arrive before scheduled playout.

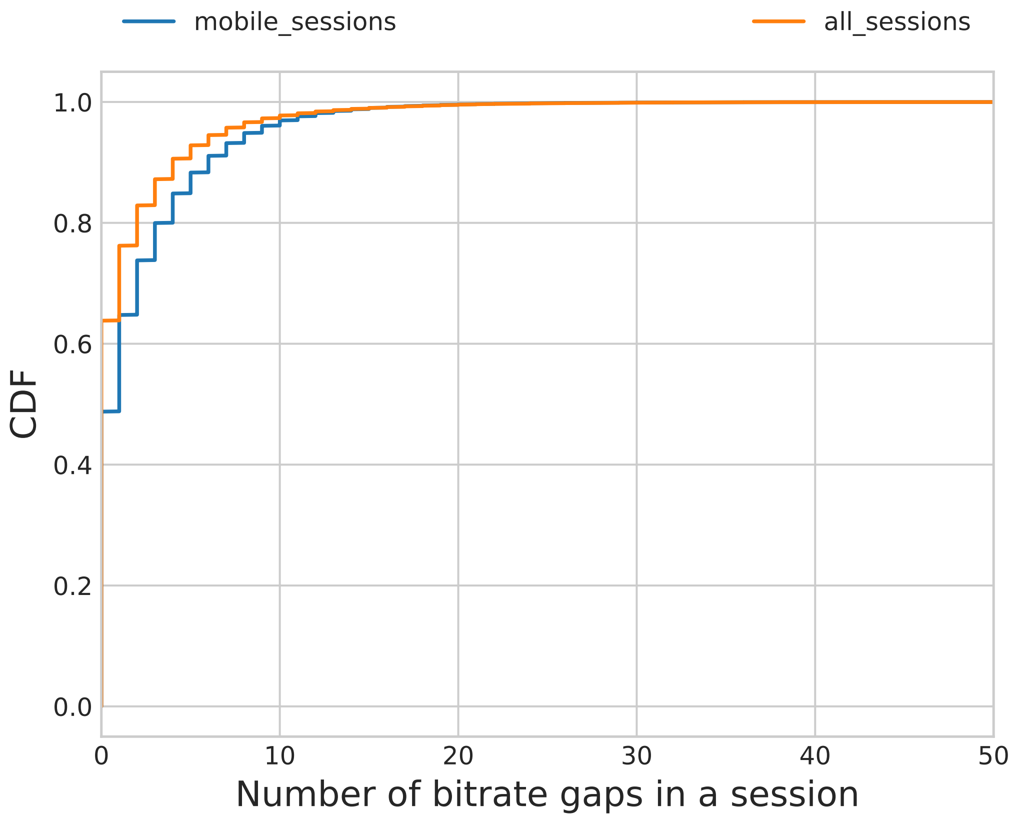


Figure 5.3: CDF for the number of sessions with one or more gaps for all sessions (orange) and mobile sessions (blue)

retransmission approach. We also analyzed how many of the sessions with mobile clients have at least one gap. As shown in Fig. 5.3, with 51.24% this ratio is even higher. Obviously, this increase in sessions with at least one gap is not too surprising since mobile/wireless clients are assumed to experience higher bandwidth fluctuations than stationary/wired clients.

5.1.4 Segment retransmisson over HTTP/2

After introducing the basics of segment retransmission for ABR streaming and outlining its drawbacks in the case of HTTP/1.1 in Sect. 5.1.2, we introduce the usage of this approach in the case of HTTP/2 in this section. We first give an overview on the advantages and disadvantages of using our retransmission approach in the case of HTTP/2 over TCP and QUIC, respectively. This is followed by a description of the implementation of our approach. Results from an evaluation of this approach are presented in Sect. 5.1.5.

5.1.4.1 Example

In the following, we compare a segment retransmission approach that is based on HTTP/2 over TCP (the current standard) with one that is based on HTTP/2 over QUIC. The TCP-based approach is shown in Fig. 5.4. Here, we show a specific scenario of retransmissions for ABR streaming. In this scenario, HTTP/2 over TCP allows the multiplexing of multiple requests within a single TCP connection. This feature makes this approach more efficient than our existing HTTP/1.1 solution, since original segment transmissions and retransmission can be performed in parallel. (In the case of HTTP/1.1, segments can either only be transmitted sequentially or a new TCP connection has to be established for the retransmissions). Despite the support of multiplexing several requests over a single TCP connection, this approach has several drawbacks. First of all, HOL blocking can lead to stalling. Such a case is indicated in Fig. 5.4, where the first retransmitted TCP segment is lost. All of the following (original and retransmitted) TCP segments will be blocked from delivery to the application layer until the lost TCP segment is successfully received. This HOL blocking, caused by a TCP segment retransmission, prevents original segments from being delivered to the buffer of the video player. This can result in an incorrect estimate of the segment download rate and consequently an unnecessary reduction in bit rate quality for the download of future original segments. In the worst case, this causes the drainage of the video player buffer, which in turn will stall the video playout.

In contrast, an HTTP/2 over QUIC approach is not impacted by HOL blocking. Fig. 5.5, shows the same segment transmission scenario as in Fig. 5.4. As opposed to the scenario shown in Fig. 5.4, the QUIC-based approach does not prevent the original datagrams from being delivered to the video player buffer if the first retransmitted UDP datagram is lost. This should lead to a significant reduction in the risk of stalling and misinterpretation of the download rate. In addition, the application can decide if the lost retransmitted UDP datagram should be retrieved again or not. This decision can be based on buffer fill level, position of the retransmitted segment in the buffer, and observed download rate. With the use of QUIC, the application can also determine at which rate segments should be downloaded. For example, pacing [5] can be applied for the retransmission of segments to assure that such transmissions only minimally interfere with the transmission of original segments.

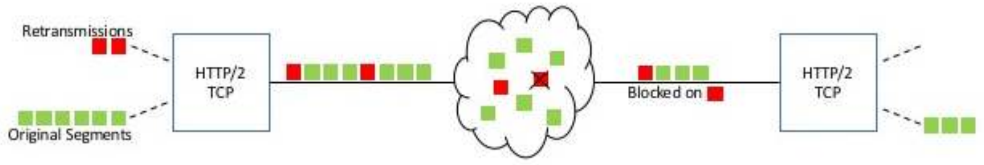


Figure 5.4: This figure shows a scenario of original and retransmitted segment transmission in the case of HTTP/2 over TCP. The first of the retransmitted TCP segments (red) is lost, which leads to HOL blocking at the receiver.

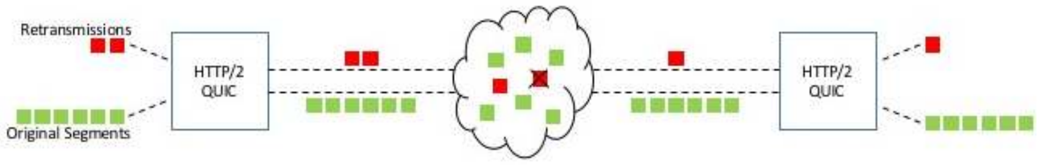


Figure 5.5: This figure shows a scenario of original and retransmitted segment transmission in the case of HTTP/2 over QUIC. In contrast to Fig. 5.4, the loss of a retransmitted UDP datagram (red) does not lead to HOL blocking and all original segments are delivered to the video player buffer.

5.1.4.2 Implementation

In this section, we give an overview of our implementation of ABR streaming that is based on HTTP/2 over QUIC that enables retransmissions of segments that have originally been transmitted in a low(er) quality (see Sect. 5.1.2).

5.1.4.2.1 SQUAD with HTTP/2 and HTTP/1.1

Since the multiplexing feature of HTTP/2 is unavailable in its predecessor, HTTP/1.1, the original version of SQUAD implements retransmission scheduling as a series of GET requests where at any given time there is only one outstanding request to the ABR streaming server. Intuitively, such a sequential implementation stalls the application pipeline and can lead to either conservative retransmission scheduling or a severe buffer drain. To prevent stalling in case of a severe drop in measured download rate, SQUAD implements retransmission abandonment, which cancels segment retransmission when we observe that the segment will not be downloaded on time. Our HTTP/2 implementation converts this sequential behavior into a parallel, multiplexed session of two simultaneous GET requests, where, at any given time there are a maximum of two possible streams active within a single connection. SQUAD is implemented as part of an open-source Python-based DASH player emulator, `AStream`.[3] For ease of integration, we use the Python-based HTTP/2 library, `hyper`[4], in order to implement two multiplexed GET requests for original and retransmission segment downloads. Additionally, we implement multithreading to allow transmissions on both HTTP/2 streams to proceed independently. We note that HTTP/2 still uses the same TCP connection which suffers from HOL blocking as explained in Sect. 5.1.4.1. and therefore, we also implement a SQUAD over QUIC approach which is introduced next. In order to make a fair comparison, we also adapt the original implementation of SQUAD to use `hyper` for making HTTP/1.1 requests.

5.1.4.2.2 SQUAD with QUIC

Similar to the experiment above, we implement multiplexed sessions for original and retransmitted segment downloads using QUIC. However, we include the use of IPC message streams with minimal overhead to com-

[3]`https://github.com/pari685/AStream`

[4]`https://github.com/Lukasa/hyper`

municate between the QUIC client (implemented in C++) and the AStream player (implemented in Python). Unlike HTTP/2 over TCP, QUIC does not suffer from HOL blocking and is designed to deliver data to the application as soon as they arrive at the receiver and a stream within a QUIC connection is not adversely affected by events that cause delay or loss of packets on a parallel, ongoing stream. At the time of this implementation, we used Chromium for Linux with QUIC version `Q043`. In order to provide support for multiplexed streams for SQUAD, we perform the following modifications on the `QUIC client`[5] code provided by Google: (i) we create and synchronize simultaneous streams within a single connection, (ii) we introduce IPC messaging not only to send commands between AStream and QUIC but also to provide intermediate chunk download rate measurements to the SQUAD ABR algorithm.

We note that this work does not focus on modifying SQUAD to perform optimally with a protocol such as QUIC but is instead intended as a study to evaluate the performance of SQUAD retransmissions over QUIC in order to determine if QoE can be improved with such an approach.

5.1.5 Evaluation Design

In this section, we describe a series of experiments, which are specifically designed to study the QoE performance of using QUIC and HTTP/2 for ABR video streaming with a focus on segment retransmission. We compare the results of these experiments with the baseline approach that uses HTTP/1.1. The server nodes (denoted as *Server1 - Server4* in Fig. 5.6) run a Caddy server (version=0.10.10) [13] with the experimental QUIC mode enabled such that the clients can stream DASH videos either over TCP or QUIC. We chose the Caddy server as it is a production server which is capable of simultaneously supporting QUIC, HTTP1.1, and HTTP/2 over TCP with

[5]`https://www.chromium.org/quic/playing-with-quic`

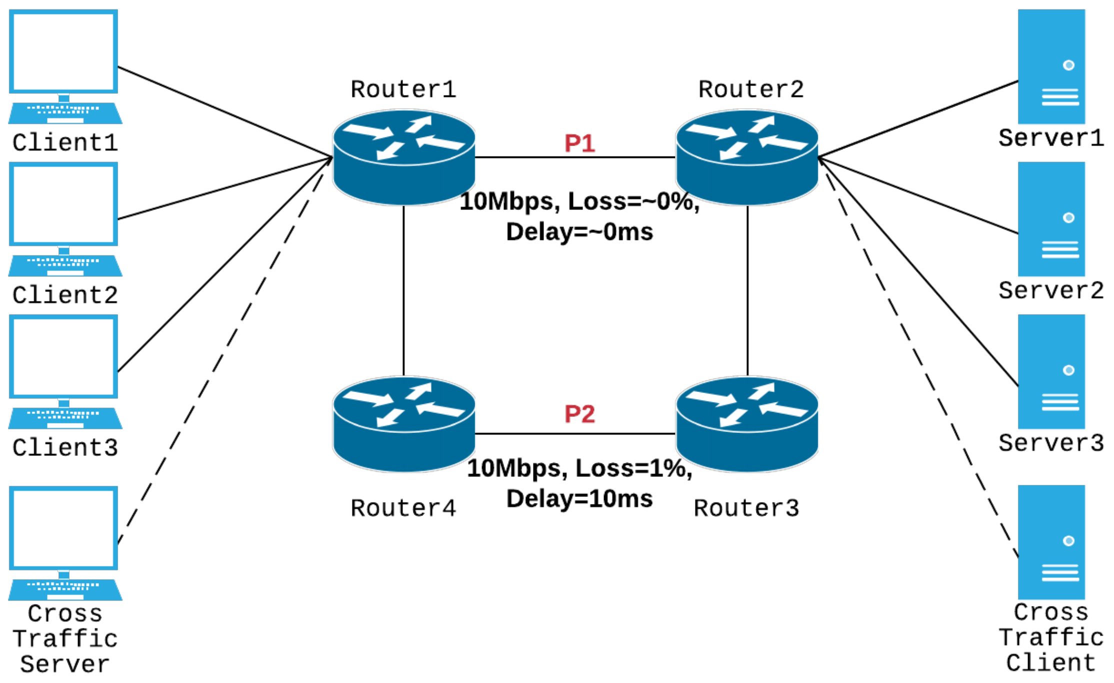


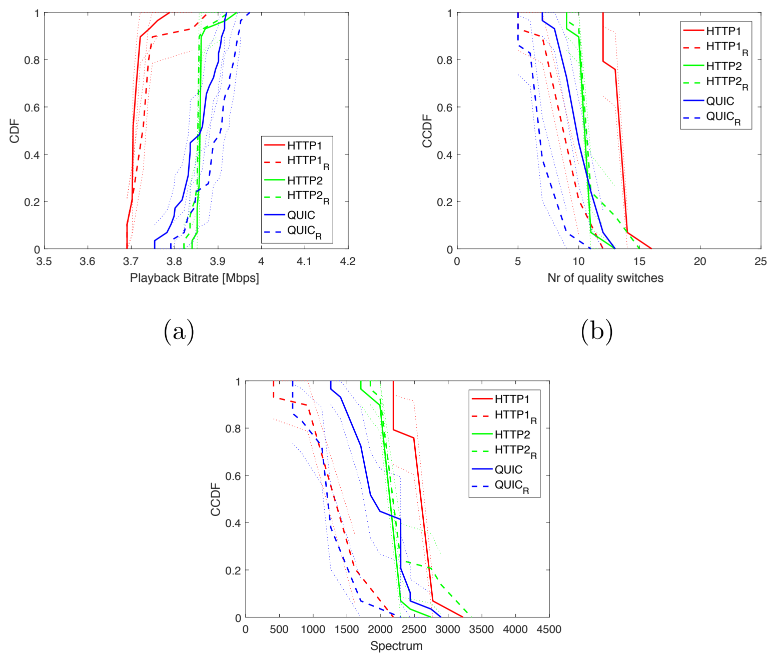
Figure 5.6: Cloudlab topology used for controlled experiments

TLS1.2. All experiments use an excerpt of the `BigBuckBunny` dataset [51] (unless stated otherwise) that comprises a 300s-long video with a 2s segment duration and the corresponding MPD file. We extended the MPD file by providing the size of each segment in each of the available quality levels.[6] The quality bitrates available in this MPD file are the following: {0.09, 0.13, 0.18, 0.22, 0.26, 0.33, 0.59, 0.79, 1.03, 1.24, 1.54, 2.48, 3.52, 4.21}Mbps. The client nodes run the SQUAD ABR algorithm [84] described above, which is implemented in a Python-based DASH player [44].

5.1.5.1 Testbed

For our controlled experiments, we use Cloudlab [70] which is a geographically distributed testbed for the development, deployment, and validation of cloud-based services. The CloudLab infrastructure consists of several different racks of varying compute and storage resources designed to provide isolated performance. The topol-

[6]We use segment sizes in the MPD file since this was introduced in AStreamer. This can easily be replaced by using byte ranges, which are available in real-world, ABR streaming solutions.



(a) (b)

(c)

Figure 5.7: Single Client Measurements - Rate Limited with UDP-Staircase cross traffic. QUIC has a significantly better overall Quality of Experience compared to HTTP/1.1 and HTTP/2, which is further improved by retransmissions. Note, subscript "R" denotes ABR segment retransmissions.

ogy shown in Fig. 5.6 consists of four clients and four servers connected by two paths **P1** and **P2** with the default set to **P1** unless stated otherwise. All nodes run vanilla Ubuntu 14.04 where all TCP related experiments use TCP Cubic. In order to account for statistical variance, every experiment in the controlled environment is repeated 30 times. For the single client experiments, we use *Client1* and *Server1* as the default pair and include other server and client pairs for parallel client cases.

5.1.5.1.1 Single Client: Rate Limiting with UDP In order to systematically compare the performance of HTTP/1.1, HTTP/2 and QUIC in a controlled environment, we use the `Iperf`[7] application to generate competing UDP traffic (denoted cross traffic) of varying amplitudes. The first set of experiments consists of repeating

[7]`https://iperf.fr/iperf-doc.php`

a stepwise variation of cross traffic where the duration of each step is 11s and varies as follows: {0-11s: 0Mbps, 12-23s: 3Mbps, 24-35s: 6Mbps, 36-55s: 9Mbps, 56-67s: 6Mbps, 68-79s: 3Mbps, 80-91s: 0Mbps} (then the pattern repeats until t=300s). Fig. 5.7 shows the CDF and CCDF along with 95% confidence intervals for upper and lower bounds of the QoE metrics described at the beginning of this section. In Fig. 5.7(a), we observe that QUIC clients have the highest average quality bitrate or AQB when compared to both HTTP/1.1 and HTTP/2. It is also worth noting that other QoE metrics such as number of quality changes ($\#QS$) and the Spectrum, H, are significantly improved with the use of QUIC retransmissions. Figure 5.8 shows results for the "W" cross traffic case where we use the `Iperf` application to generate competing UDP cross traffic that creates a "W" shaped bottleneck bandwidth and varies as follows: {0-20s: 9Mbps, 21-40s: 5Mbps, 41-60s: 9Mbps, 61-80s: 0Mbps} (then the pattern repeats until t=300s). Although, in terms of AQB, $\#QS$ and H, HTTP/2 clients appear to experience the best QoE, we observed that the clients also experience a relatively high rebuffering ratio, RB, of 4% while using HTTP/2 for ABR streaming. Since it is well known that the foremost objective of any ABR client streaming algorithm is to eliminate or reduce rebuffering, we conclude that QUIC, especially with the use of segment retransmissions, also performs significantly better than HTTP/1.1 and HTTP/2 for the "W" cross traffic case.

5.1.5.1.2 Single Client: Re-ordering and HOL Since packet reordering in the Internet is not uncommon [43], protocols for ABR streaming should be robust in the face of such reordering. Here, we study the ability of HTTP1.1, HTTP/2, and QUIC to recover from re-ordering of packets. This is the only experiment where we use the second path (denoted **P2** in Fig. 5.6) to carry video streams. In order to induce re-ordering of packets, we switch between a low latency, low loss path, **P1**, and a high latency, high loss path, **P2**, every second using SDN, namely the Open-Flow [54] implementation, which provides fine-grained, dynamic traffic engineering

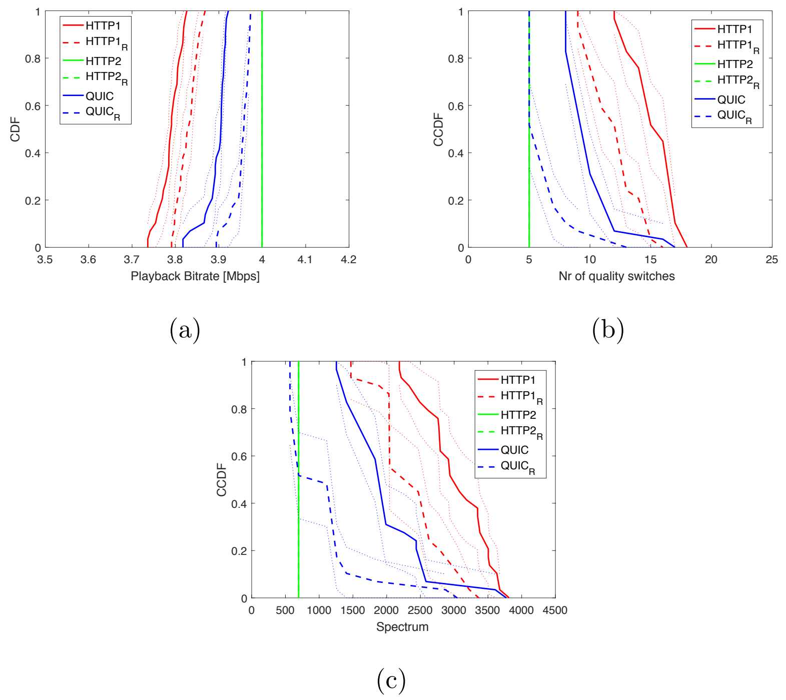


Figure 5.8: Single Client Measurements - Rate Limited with UDP-W cross traffic. QUIC has a significantly better overall QoE compared to HTTP/1.1. Although HTTP/2 sessions appear to be having a higher QoE, all clients experience 4% rebuffering. Note, subscript “R” denotes ABR segment retransmissions.

for application packets. As shown in Fig. 5.6, **P2** is characterized by 1% loss and 10ms delay implemented using `tc`[8] and `netem`[9] utilities. For the experiments presented in Sect. 5.1.5.1.1, we find that HTTP/2 is either comparable or marginally worse than HTTP/1.1 and QUIC. In the case of packet reordering (shown in Figure 5.9), we see that HTTP/2 performs significantly worse than QUIC and HTTP/1.1 . Not only is the *AQB* significantly lower with a high variation between runs, but also the rebuffering is as high as 10% where over 60% of clients experience an *RB* of 2.5%. Further analysis using the `tshark`[10] utility reveals that a HTTP/2 session experiences 9.5% fast TCP retransmits. In comparison, HTTP/1.1 experiences 7.1%, and QUIC sessions experience no UDP retransmissions since they use NACKs (c.f.

[8]`http://lartc.org/manpages/tc.txt`

[9]`http://man7.org/linux/man-pages/man8/tc-netem.8.html`

[10]`https://www.wireshark.org/docs/man-pages/tshark.html`

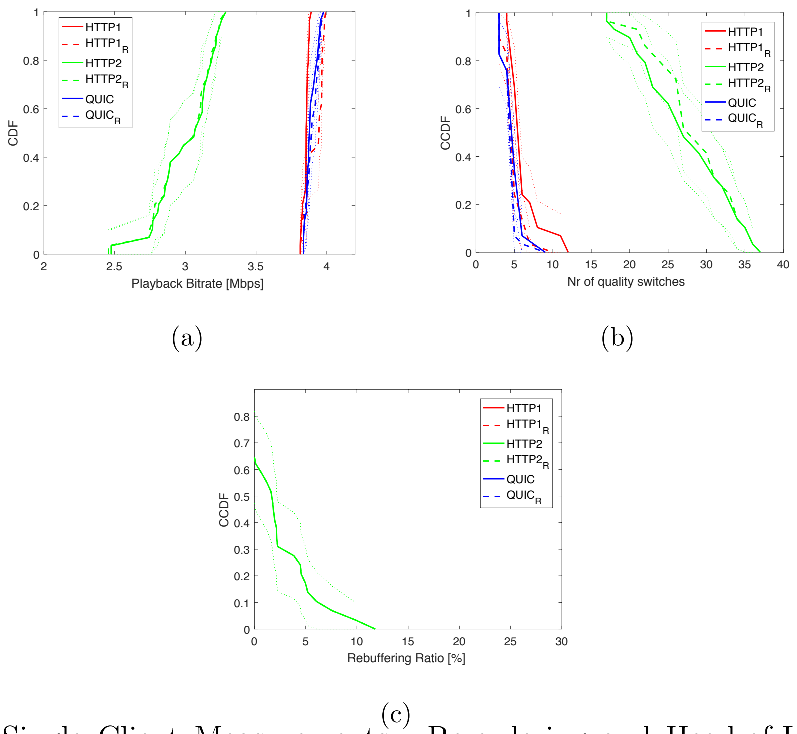


Figure 5.9: Single Client Measurements - Re-ordering and Head-of-Line Blocking. Re-ordering has an adverse effect on HTTP/2 causing significant degradation of QoE metrics, especially with respect to rebuffering which can be as high as 10% in spite of selecting lower quality bitrates as seen from (a). Note, subscript "R" denotes ABR segment retransmissions.

Sect. 2.3). More details on the `tshark` data can be found in [30] Additionally, QUIC uses a higher initial congestion window size=32 (the Linux default for TCP is 10) and also grows the window more aggressively, thus, allowing more unacknowledged bytes in flight. This results in a more reliable download rate measurement and a stable buffer level for the ABR client and consequently, a reduction in the quality variations $\#QS$ as observed in Fig. 5.9(b).

5.1.5.1.3 Parallel Clients: Competing Traffic For this experiment we use the additional client and server pairs (denoted as *Client2/Server2* and *Client3/Server3* in Figure 5.6) to initiate three simultaneous sessions of QUIC-based SQUAD clients. Although all three clients enjoy a smooth playback experience without rebuffering as shown in Figure 5.10, we note that the bandwidth sharing can result in unfair behavior in the case of ABR streaming sessions. This is contrary to the analysis presented by the authors of [45] where they observe that QUIC flows are fair to each other but only unfair to TCP flows when downloading a file. While we similarly observe that QUIC does tend to "starve out" TCP flows, we note that ABR streaming over QUIC with the use of retransmissions can result in unfair behavior for competing ABR streams since multiplexing due to retransmissions can occur at different points throughout the streaming session. In order to corroborate this analysis, we present the percentage of retransmissions in Table 5.1, which shows that the three clients experience varying number of ABR segment retransmissions per run. Since these retransmissions occur asynchronously, the clients observe different buffer levels and rate measurements throughout a streaming session. We also perform similar experiments with three HTTP/2 clients and observe that HTTP/2 shows a nearly equal distribution of *AQB* and closer inspection reveals that the *AQB* of *Client1* is 0.5Mbps higher on average as compared to the other two clients. Since TCP is more conservative about setting the initial congestion window size and has a less aggressive window growth it enables all three clients to have a "fair" share of the bottleneck bandwidth. Further details on the TCP-based experiments can be found in [30].

Table 5.1: ABR Segment Retransmissions for three parallel QUIC clients

	Client1	*Client2*	*Client3*
Average %Retransmissions	0.8±1.3	1.7±1.3	1.0±0.9

5.1.5.2 Internet

For the Internet measurements, we use Amazon EC2 virtual machines in Mumbai, India and Oregon, USA as servers and a client in the UMass Amherst campus network to perform inter-continental and intra-continental measurements, respectively. Here, we repeat each experiment 60 times to account for increased network variations in an uncontrolled environment. Since the bottleneck bandwidth during off-peak hours

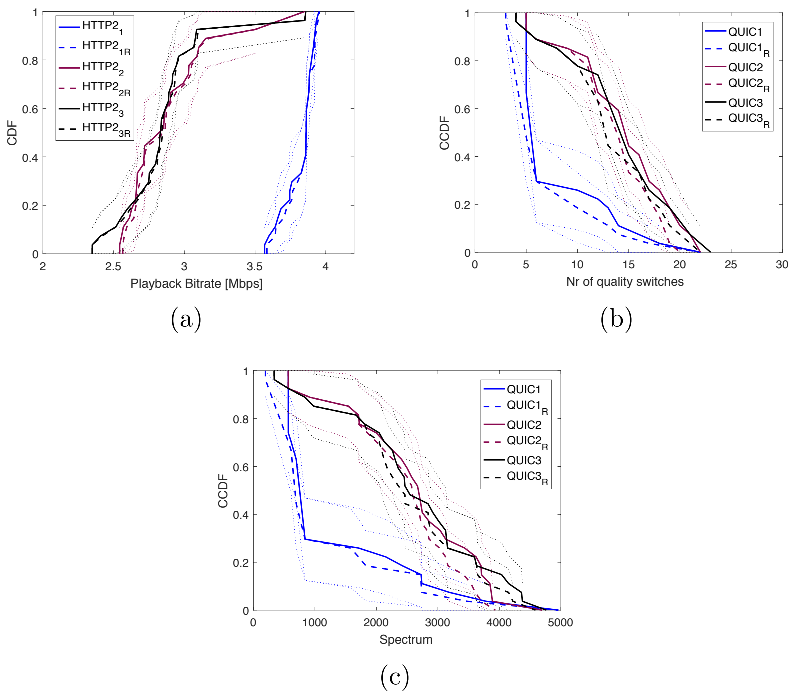
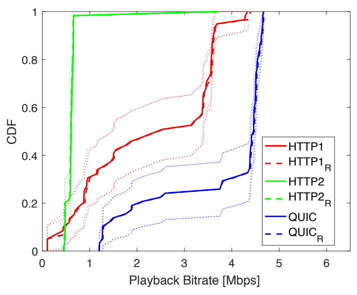
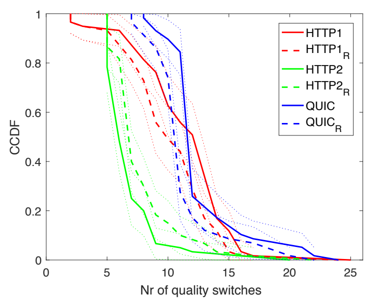


Figure 5.10: Parallel Client Measurements - Three QUIC Clients. Competing QUIC clients show an unfair behavior where two clients experience relatively similar QoE but one client has a significantly better QoE than others.

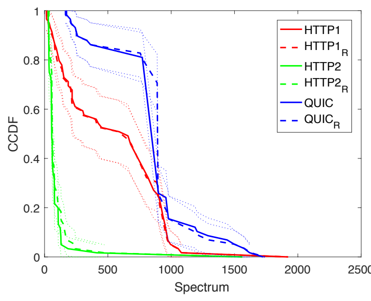
can be high, we use a different video dataset with higher qualities, `RedBull` [51], and modify the MPD to contain the following bitrates {0.10, 0.15, 0.20, 0.25, 0.30, 0.40, 0.50, 0.70, 0.90, 1.20, 1.50, 2.00, 2.50, 3.00, 4.00, 5.00, 6.00}Mbps for a video duration of 300s and a segment duration of 2s. Figure 5.11 presents results for measurements "in the wild" over inter-continental links from an EC2 web server located in India. The average quality bitrate (in Fig. 5.11(a)) is significantly higher for QUIC than HTTP/2 and HTTP/1.1. Fig. 5.11(b) shows that $\#QS$ is also reduced with the use of QUIC and HTTP/1.1 retransmissions indicating an overall high QoE. Table 5.2 shows QoE metrics for similar measurements conducted with the server located at EC2 in Oregon. Here, it is worth mentioning that all QoE metrics are comparable for HTTP/1.1 and QUIC where QUIC is marginally better than HTTP/1.1, but are significantly improved over HTTP/2 (for example, the average bitrate AQB is less than half of that obtained with HTTP/1.1 and QUIC). Since Internet traffic is predominantly



(a)



(b)



(c)

Figure 5.11: Internet Measurements - ABR streaming is performed over inter-continental links with the server at Amazon EC2 in India and the client on the US East Coast. QUIC far outperforms HTTP/1.1 and HTTP/2 in terms of QoE, i.e., provides significant improvement in Average Quality Bitrate while providing comparable reduction in the number of quality switches.

comprised of TCP flows, these results further reinforce the observations made in Sect. 5.1.5.1.2 for high delay, high loss paths with competing TCP traffic. Our results show that the use of QUIC results in better QoE in the case of inter-continental as well as intra-continental links, while the advantage compared to HTTP/1.1 is more significant in case of the former.

5.1.6 Conclusion

In this work, we conduct systematic experiments to analyze the performance implications of various HTTP/2 transport layer candidates on ABR streaming systems, particularly with respect to ABR segment retransmissions. We leverage the multiplexing feature of QUIC and HTTP/2 in order to efficiently implement parallel retransmissions in a higher quality with the objective of maximizing average quality bitrate while also minimizing bitrate variations throughout the duration of a stream-

Table 5.2: ABR Quality of Experience over the Internet: Amazon EC2 Oregon - US East Coast

	Internet: HTTP/1.1	Internet: HTTP/2	Internet: QUIC
AQB (Mbps)	5.31±0.1	2.12±0.6	5.31±1.9
AQB_R (Mbps)	5.66±0.1	2.13±0.6	5.44±0.2
$\#QS$	8.48±1.4	9.09±2.6	7.91±1.8
$\#QS_R$	3.82±2.1	6.98±2.5	5.81±1.7
H	490±213	552±280	445±299
H_R	242±312	447±255	351±273
RB_R(%)	0	0±10.8	0

ing session. We use a nearly isolated testbed setup in CloudLab and measurements "in the wild" to show that QUIC retransmissions provide a significantly better QoE than TCP in high latency, high loss networks while exhibiting comparable QoE in low latency, low loss networks.

5.2 Improving Resilience and QoE of ABR Streaming Sessions through Retransmissions using Multipath Transport layer Protocols

5.2.1 Introduction

We propose on extending our work on the multiple streams into multipath, to increase its resiliency and throughput, so as to account for link failure, for example due to severe weather/emergency, which we observed in the case of first two parts of the thesis.

5.2.2 Background

The MPTCP Linux kernel implementation [60] consists of 4 main blocks 1) the meta socket, which is the central abstraction of each MPTCP connection 2) the path manager, that decides on creation and removal of subflows 3) the congestion control, to insure TCP friendliness on shared bottleneck links, and finally 4) the scheduler,

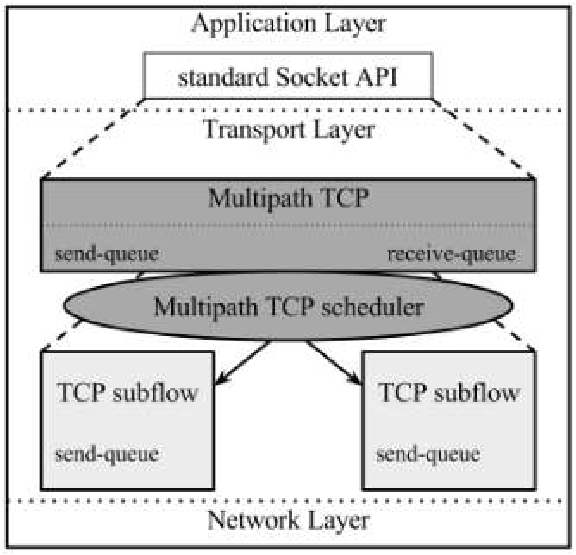


Figure 5.12: MPTCP architecture [62]

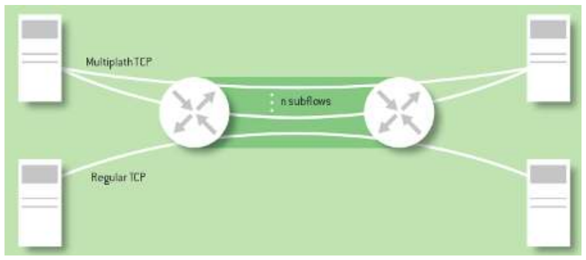


Figure 5.13: Coupled Congestion control [1]

which decides the subflow to be used for a packet. As seen from Figure.5.12, these blocks are closely tied and the latency depends on all of them.

Figure 5.13 shows us that, to insure fairness and responsiveness of MPTCP to other kind of traffic we use coupled congestion control as it can reduce the traffic sent over the bad link and increase the traffic over the better link to improve performance as shown in [57]. We focus on [68] BALIA (Balanced Linked Adaptation) as they found it to be better [63] than other coupled algorithms like OLIA and LIA. BALIA is only used in congestion avoidance part of the AIDM phase. The other phases like slow start, fast retransmit/recovery algorithms are the same as in TCP. BALIA is summarized as follows. Each source s has a set of routes r and each route r maintains

Table 5.3: MPTCP configuration

Parameter	Value
Congestion control	BALIA
Number of subflows	2 (1 on each path)
Path Manager	Full Mesh
Scheduler	Default (Low-RTT-First)

a congestion window w_r and measures its round-trip time τ_r. The adaptation for each ACK on router $r\epsilon s$ is eq 5.1 and for each packet loss on route $r\epsilon s$ is eq 5.2,

$$w_r \leftarrow w_r + \left(\frac{x_r}{\tau_r(\Sigma x_k)^2}\right)\left(\frac{1+\alpha_r}{2}\right)\left(\frac{4+\alpha_r}{5}\right), \tag{5.1}$$

$$w_r \leftarrow w_r - \frac{w_r}{2} min\{\alpha_r, 1.5\} \tag{5.2}$$

A the scheduler decides which subflow a packet is transmitted. Improper scheduling can introduce head-of-line-blocking or receive-window limitation, especially while using heterogeneous paths (which is the case in most real-world situations). This impacts performance and quality of experience [62]. Three types of schedulers are implemented as plug-able modules in MPTCP kernel: 1) Lowest-RTT-First(Default): This is the one that gives maximum end-to-end throughput, and is currently used in our experiments. It achieves this by sending data on the subflow with the lowest RTT until its congestion window is full. Then it transmits on the subflows with the next higher RTT. 2) Round-Robin: Selects one subflow after the other, using them in equal portioned time slices and in a circular order and without any priority. This is easy to implement and starvation-free. But,this causes high application delay in application-limited flows. Such delays have significant impact on delay-sensitive applications, as to mitigate delay-spikes they have to maintain large buffers which could cause buffer-bloat. 3) Redundant Scheduler: Redundantly sends packets on all available subflows. This approach trades throughput for latency. mp aware ABR [38]

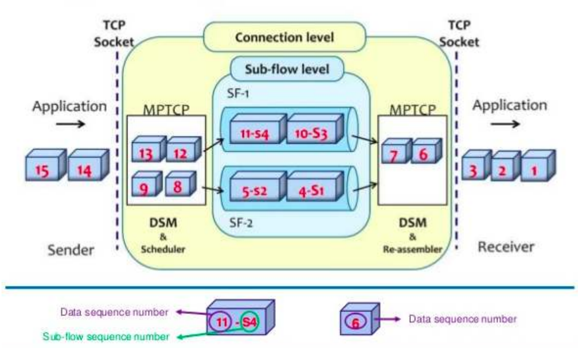


Figure 5.14: Sequence numbers in MPTCP

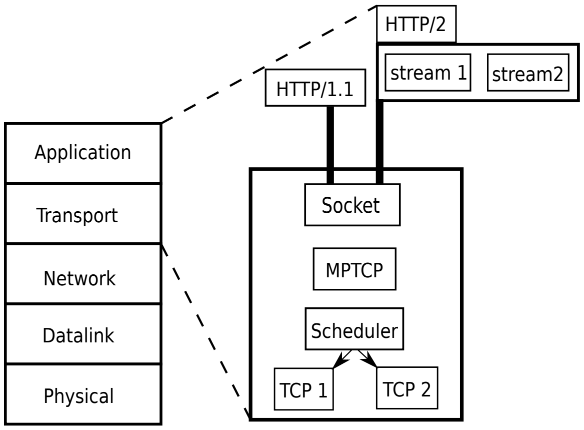


Figure 5.15: Protocol stack view

proposed and demonstrated MP-DASH framework for optimizing video delivery over WiFi.

To be compatible with the middle boxes, each subflow needs to have TCP segment numbering, so to facilitate the use of multiple interfaces MPTCP segment sequence numbers are used, as shown in Figure. 5.14.

5.2.3 Implementation

In this section we give an overview of our implementation of ABR streaming over MPTCP that is based on HTTP/2 that enables retransmissions of segments that have originally been transmitted in lower quality. Our implementation of the various protocols is shown in the Figure. 5.15.

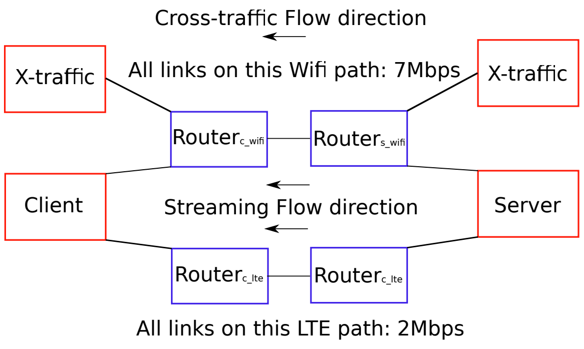


Figure 5.16: Cloudlab topology used for controlled experiments

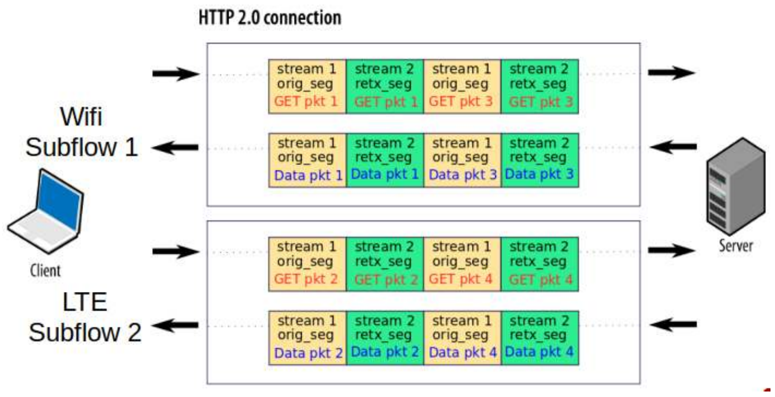

HTTP 2.0 connection
Wifi Subflow 1
stream 1 orig_seg GET pkt 1
stream 2 retx_seg GET pkt 1
stream 1 orig_seg GET pkt 3
stream 2 retx_seg GET pkt 3
stream 1 orig_seg Data pkt 1
stream 2 retx_seg Data pkt 1
stream 1 orig_seg Data pkt 3
stream 2 retx_seg Data pkt 3
Client
Server
LTE Subflow 2
stream 1 orig_seg GET pkt 2
stream 2 retx_seg GET pkt 2
stream 1 orig_seg GET pkt 4
stream 2 retx_seg GET pkt 4
stream 1 orig_seg Data pkt 2
stream 2 retx_seg Data pkt 2
stream 1 orig_seg Data pkt 4
stream 2 retx_seg Data pkt 4

Figure 5.17: HTTP/2 streams with MPTCP subflows

5.2.3.1 SQUAD on MPTCP with HTTP/2 and HTTP/1.1

We use the settings as described in table 5.3.

Our settings facilitate us to realize the following design in Figure. 5.17. The settings required, are described below.

Although, we use the Full Mesh path manager, which creates flows between each ip-pair involved. We disable IP_FORWARDING and configure routes in the kernel routing table such that, we have 1 subflow on each path shown in figure 5.16. As discussed in the above section 5.1.4.2, HTTP/1.1 doesn't allow out of order concurrent sessions on the same connection and this forces us to have the retransmission in the same pipeline, whereas HTTP/2 allows us to have multiplexed sessions where we use two simultaneous streams (one for the original segments and the other for retrans-

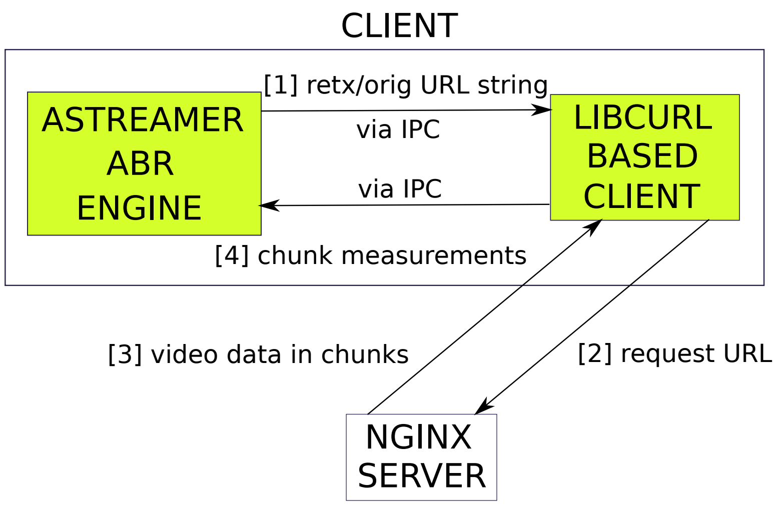


Figure 5.18: Flow diagram of the implemented code

missions). With the Python-based HTTP/2 library, hyper, we saw lower throughput performance as compared to HTTP/1.1, so we switched to an open-source, widely used library `libcurl`[11] based client (implemented in C++) for both HTTP/1.1 and HTTP/2. We show the flow of the code in Figure.5.18, which uses IPC message streams with minimal overhead to communicate intermediate chunk download rate measurements between the libcurl based client and the Astream player (implemented in Python).

5.2.4 Evaluation Design

In this section, we describe a series of experiments, which are specifically designed to study the QoE performance of using HTTP1/1.1 and HTTP/2 over MPTCP for ABR video streaming with a focus on segment retransmission. The server node in Fig. 5.16 run a nginx server (version=1.15.7) [12] HTTP/1.1 AND HTTP/2 enabled such that the clients can stream DASH videos. We chose the nginx server as it is a production server which is capable of simultaneously supporting HTTP1.1, and HTTP/2 over TCP with TLS1.2. All experiments use an excerpt of the `BigBuckBunny`

[11]`http://curl.haxx.se/download/curl-7.63.0.tar.bz2`

dataset [51] (unless stated otherwise) that comprises a 300s-long video with a 2s segment duration and the corresponding MPD file. We extended the MPD file by providing the size of each segment in each of the available quality levels.[12] The quality bitrates available in this MPD file are the following: {0.09, 0.13, 0.18, 0.22, 0.26, 0.33, 0.59, 0.79, 1.03, 1.24, 1.54, 2.48, 3.52, 4.21}Mbps. The client nodes run the SQUAD ABR algorithm [84] described above, which is implemented in a Python-based DASH player [44].

5.2.4.1 Testbed

For our controlled experiments, we use Cloudlab [70] which is a geographically distributed testbed for the development, deployment, and validation of cloud-based services. The CloudLab infrastructure consists of several different racks of varying compute and storage resources designed to provide isolated performance. The topology shown in Figure. 5.16 consists of two clients and two servers connected by two paths. All nodes run vanilla Ubuntu 14.04. In order to account for statistical variance, every experiment in the controlled environment is repeated 10 times.

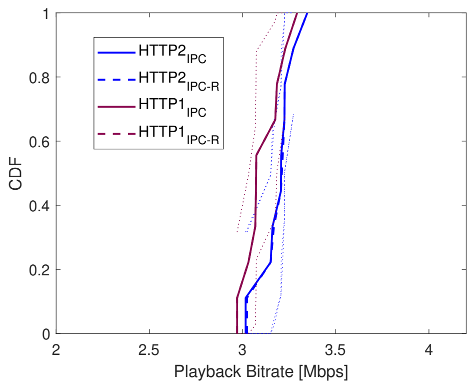
To motivate the need to use multi-path, we look at the experiment in which we use tcp with a single path and throttle the bandwidth with cross-traffic abruptly. We use only the WiFi path (not the LTE path) in the Figure. 5.16 .

In order to systematically compare the performance of HTTP/1.1 and HTTP/2 in a controlled environment, we use the `Iperf`[13] application to generate competing UDP traffic (denoted cross traffic) of varying amplitudes.

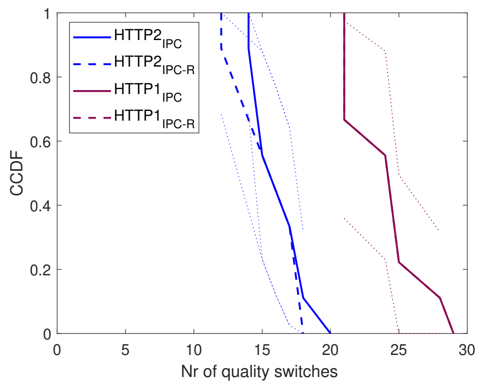
5.2.4.1.1 Single Client (TCP): Abrupt Limiting with UDP Figure.5.19 shows results for more abrupt ”W” cross traffic case where we use the `Iperf` applica-

[12]We use segment sizes in the MPD file since this was introduced in AStreamer. This can easily be replaced by using byte ranges, which are available in real-world, ABR streaming solutions.

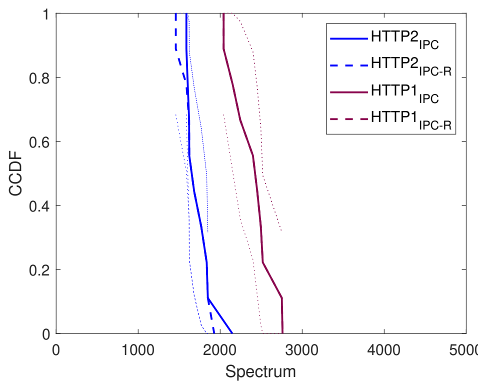
[13]`https://iperf.fr/iperf-doc.php`



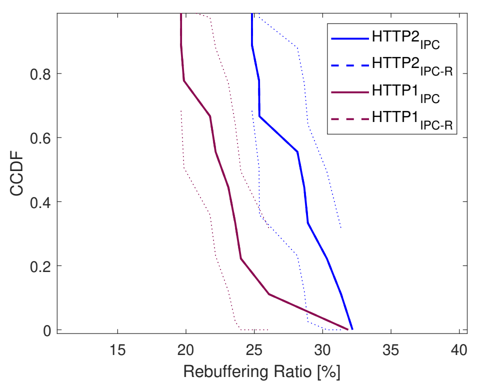
(a)



(b)



(c)



(d)

Figure 5.19: Single Client (TCP) Measurements - Rate Limited with UDP-W cross traffic. HTTP/2 has similar Quality of Experience compared to HTTP/1.1. Note, subscript “R” denotes ABR segment retransmissions.

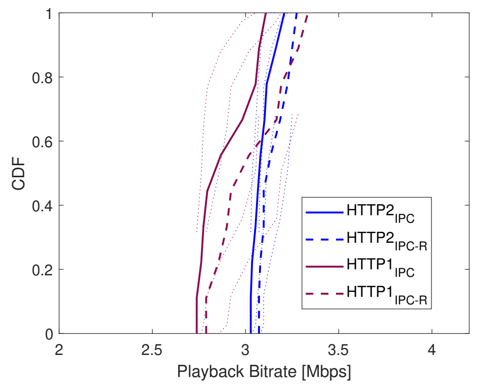
tion to generate competing UDP cross traffic that creates a ”W” shaped bottleneck bandwidth and varies as follows: {0-20s: 7Mbps, 21-40s: 5Mbps, 41-60s: 7Mbps, 61-80s: 0Mbps} (then the pattern repeats until t=300s). We see 20-30% rebuffering, which defeats the purpose of giving uninterrupted experience using ABR streaming. In terms of AQB, $\#QS$ and H, HTTP/2 client performs similarly HTTP/1.1.

Now, as we have seen that abrupt traffic on single path causes high rebuffering, we look into utilizing additional path and use both the WiFi and the LTE path in Figure. 5.16 by using MPTCP. We generate the cross-traffic only across the WiFi interface.

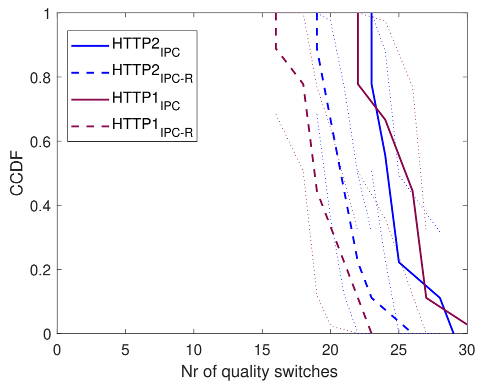
5.2.4.1.2 Single Client: Abrupt Limiting with UDP Figure.5.20 shows results for more abrupt "W" cross traffic case where we use the `Iperf` application to generate competing UDP cross traffic that creates the same "W" shaped bottleneck bandwidth as in the above mentioned single path experiment. Compared to single path experiment where we saw 20-30% rebuffering, we see no rebuffering in case of HTTP/2 and $\approx 0.1\%$ rebuffering in 2 runs of HTTP/1.1, and we also see significant improvement in terms of AQB, $\#QS$ and H. When comparing MPTCP experiments HTTP/2 client performs significantly better than HTTP/1.1 for the "W" cross traffic.

Figure 5.21 shows results for a bursty cross traffic (which we call "trace0") that throttles the WiFi interface and it varies as follows: {0-15s:0M, 15-17s: 7Mbps, 17-25s: 0Mbps, 25-36s: 7Mbps, 36-45s: 0Mbps, 45-46s: 7Mbps, 46-47s: 0Mbps, 47-48s: 7Mbps, 48-51s:0M, 51-62s: 7Mbps, 62-66s: 0Mbps, 66-83s: 7Mbps, 83-96s: 0Mbps, 96-105s: 7Mbps, 105-111s: 0Mbps, 111-121s: 7Mbps, 121-128s: 0Mbps, 128-137s: 7Mbps, 137-152s:0M, 152-154s: 7Mbps, 154-164s: 0Mbps, 164-189s: 7Mbps, 189-192s: 0Mbps, 192-209s: 7Mbps, 209-216s: 0Mbps, 216-232s: 7Mbps, 232-257s: 0Mbps, 257-278s: 7Mbps, 278-281s: 0Mbps, 281-291s: 7Mbps, 291-302s: 0Mbps, 302-304s: 7Mbps, 304-318s: 0Mbps, 318-334s: 7Mbps, 334-344s: 0Mbps, 344-348s: 7Mbps, 348-362s: 0Mbps }. In terms of AQB, $\#QS$ and H, HTTP/2 client performs significantly better than HTTP/1.1 for the "trace0" cross traffic case as well.

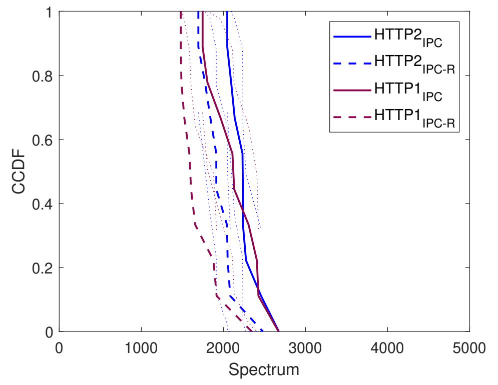
5.2.4.1.3 Single Client: Gradual Rate Limiting with UDP The first set of experiments consists of repeating a more gradual stepwise variation of cross traffic as follows: {0-15s: 0Mbps, 15-26s: 3Mbps, 26-37s: 5Mbps, 37-56s: 7Mbps, 56-67s: 5Mbps, 68-79s: 3Mbps, 80-91s: 0Mbps} (then the pattern repeats until t=300s). Figure.5.22 shows the CDF and CCDF along with 95% confidence intervals for upper and lower bounds of the QoE metrics described at the beginning of this section. In Figure.5.22(a), we observe that HTTP/1.1 clients have marginally high QoE metrics such as avg. bitrate, number of quality changes ($\#QS$) and the Spectrum, H, are



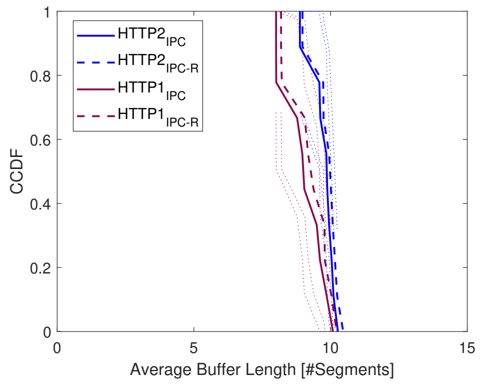
(a)



(b)



(c)



(d)

Figure 5.20: Single Client Measurements - Rate Limited with UDP-W cross traffic. HTTP/2 has a significantly better overall Quality of Experience compared to HTTP/1.1 , which is further improved by retransmissions. Note, subscript "R" denotes ABR segment retransmissions.

significantly improved with the use of retransmissions. But the avg. buffer length is higher for HTTP/2 as retransmissions happen in parallel and this is faster than the case of HTTP/1.1 where they occur serially.

5.2.5 Conclusion

In this work, we conduct systematic experiments to analyze the performance implications of HTTP/2 vs HTTP/1.1 over multipath-transport layer protocol MPTCP on ABR streaming systems, particularly with respect to ABR segment retransmissions. We leverage the multiplexing feature of HTTP/2 in order to efficiently implement

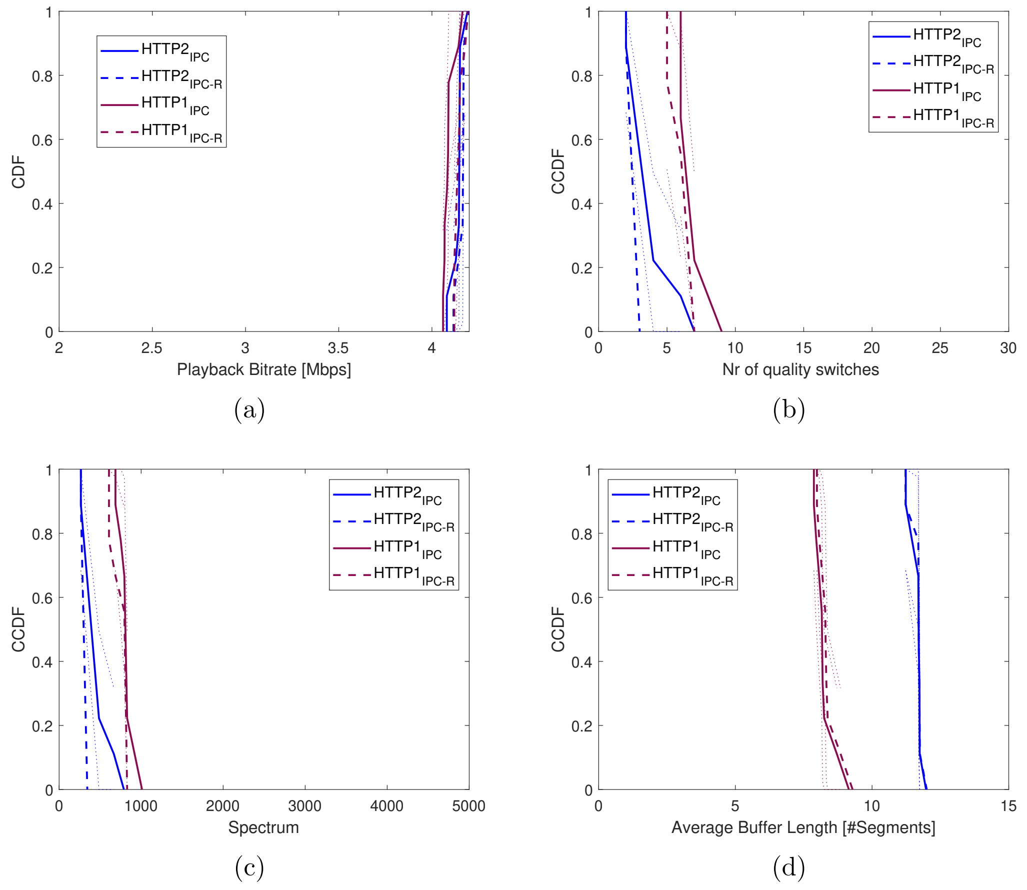


Figure 5.21: Single Client Measurements - Rate Limited with UDP-trace0 cross traffic. HTTP/2 has a significantly better overall Quality of Experience compared to HTTP/1.1 , which is further improved by retransmissions. Note, subscript "R" denotes ABR segment retransmissions.

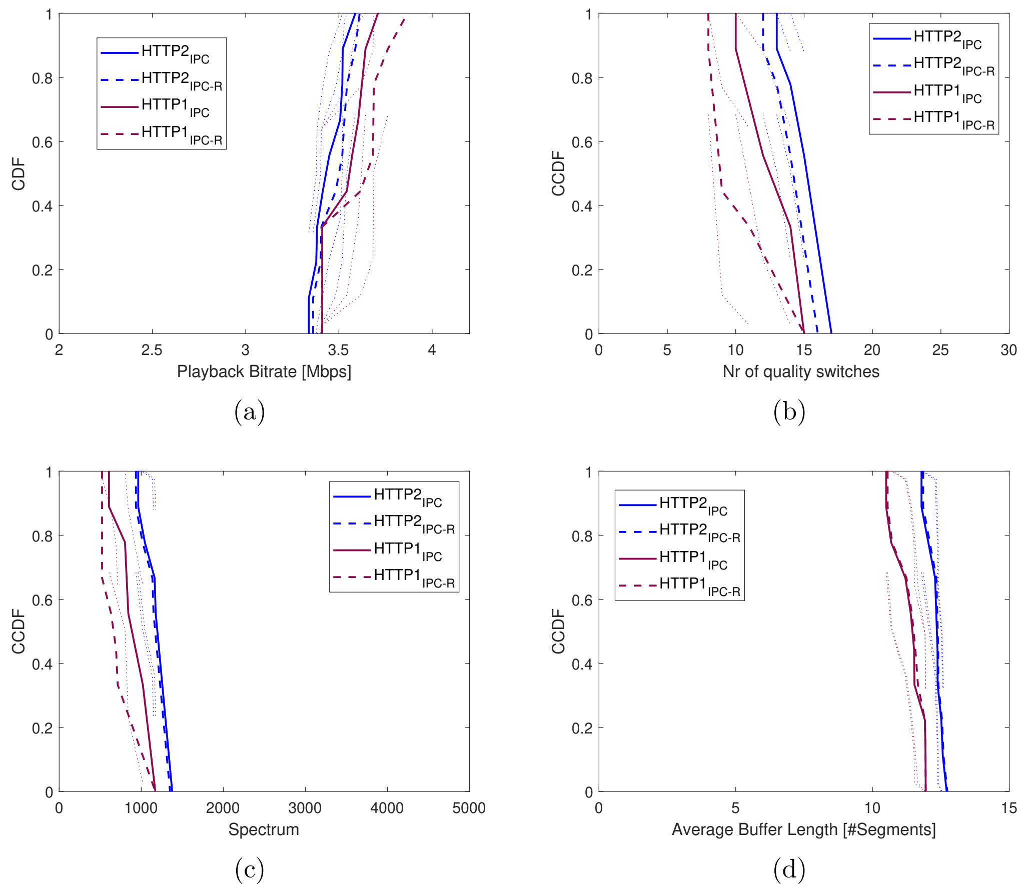


Figure 5.22: Single Client Measurements - Rate Limited with UDP-Staircase cross traffic. HTTP/1.1 has a marginally better Avg. bitrate, no. switches and spectrum compared to HTTP/1.1 , which is further improved by retransmissions. Note, subscript “R” denotes ABR segment retransmissions.

parallel retransmissions in a higher quality with the objective of maximizing average quality bitrate while also minimizing bitrate variations throughout the duration of a streaming session. We use a nearly isolated testbed setup in CloudLab and observed that HTTP/2 performs better in case of abrupt cross-traffic (which is generally seen in the "wild" internet) and is almost close to performance of HTTP/1.1 in case of a more gradual cross-traffic. But in all cases it has a higher avg. buffer attributing it to the use of parallel retransmissions. And in general retransmissions provide a significantly better QoE in low latency, low loss networks.

CHAPTER 6

CONCLUSION

In our work we analyzed a threefold approach, going from the case in which the infrastructure is totally down, to where it has been replaced by make shift low capacity mobile cellular base station. In the first part, we looked at dissemination of weather alerts specific to geographical areas. We compared an ICN based approach to an infrastructure-based approach that is less resilient in the case of disaster. In addition, we compared the performance of different message forwarding strategies in VANETs (Vehicular Adhoc Networks) using ICN. Our results show that ICN strategy outperforms the infrastructure-based approach as its 100 times faster for 63% of total messages delivered.

Then we looked at offloading as much traffic as possible from the mobile base stations, using device-to-device communication with a well fit ICN approach to fetch contents from nearby peers, increases the resiliency of the network in cases of outages and disasters. We were able to offload traffic from the backhaul network by up to 51.7%, suggesting an advantageous path to support the surge in traffic while keeping complexity and cost for the network operator at manageable levels.

Finally, we looked at ABR streaming with HTTP/2 and the two transport layer candidates, especially with respect to video segment retransmissions. It's multiplexing feature enabled use to successfully implement parallel retransmissions in higher quality with objective of improving QoE. Our results show that QUIC retransmissions provide a significantly better QoE than TCP in high latency. We further investigated the use of Multipath TCP with HTTP/1.1 and HTTP/2 and found that retransmissions provide significantly better QoE and that, using parallel retransmissions keeps the average buffer level higher and gives us better performance in abrupt competing-traffic.

As potential for future work, we know that MPTCP does not differentiate between the data it receives from the application layer .i.e. between the HTTP/2 logical streams, it has no control over which interface the data passed using these streams is sent. MPQUIC [27] is a smarter candidate that can differentiate between these streams, hence giving us more control over how to schedule packets related to these streams should be handles. Thus, this could allow an implementation for smart scheduling.

BIBLIOGRAPHY

[1] Decoupled from ip, tcp is at last able to support multihomed hosts. http://delivery.acm.org/10.1145/2600000/2591369/p40-paasch.pdf?ip=128.119.202.171&id=2591369&acc=OPEN&key=73B3886B1AEFC4BB%2E0404F0890BAA435B%2E4D4702B0C3E38B35%2E6D218144511F3437&__acm__=1547654113_c4fb38db40173442e31208186b1b7313. Accessed: 2019-01-16.

[2] Nakagami-m fast fading propagation loss model. https://www.nsnam.org/doxygen/classns3_1_1_nakagami_propagation_loss_model.html\#details. Accessed: 2017-07-17.

[3] NetFlix. http://www.netflix.com. Accessed: 2018-03-10.

[4] ns-3. https://www.nsnam.org/. Accessed: 2017-07-17.

[5] Packet Pacing in QUIC. https://groups.google.com/a/chromium.org/forum/#!topic/proto-quic/fRk0I98VSMk. Accessed: 2018-03-10.

[6] QUIC, a multiplexed stream transport over UDP. https://www.chromium.org/quic. Accessed: 2018-03-10.

[7] The role of mobiles in disasters and emergencies. http://braddye.com/gsm_disaster_relief_report.pdf. Accessed online: August 23, 2018.

[8] Usage of http/2 for websites. https://w3techs.com/technologies/details/ce-http2/all/all. Accessed: 2018-03-10.

[9] YouTube. https://www.youtube.com. Accessed: 2018-03-10.

[10] Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021. february, 2017, February 2018.

[11] Lte standard, July 2018.

[12] nginx web server. https://www.nginx.com/, Accessed 3-12-2018.

[13] Caddy web server. https://caddyserver.com/, Accessed 3-9-2017.

[14] Adamic, Lada A, and Huberman, Bernardo A. Zipf's law and the internet.

[15] Afanasyev, Alexander, Moiseenko, Ilya, and Zhang, Lixia. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, NDN, October 2012.

[16] Amadeo, Marica, Campolo, Claudia, and Molinaro, Antonella. Forwarding strategies in named data wireless ad hoc networks. *J. Netw. Comput. Appl. 50*, C (Apr. 2015), 148–158.

[17] Amadeo, Marica, Campolo, Claudia, and Molinaro, Antonella. A novel hybrid forwarding strategy for content delivery in wireless information-centric networks. *Computer Communications 109* (2017), 104 – 116.

[18] Amaral, L., Sofia, R., Mendes, P., and Moreira, W. Oi! - opportunistic data transmission based on wi-fi direct. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (April 2016), pp. 578–579.

[19] Angius, Fabio, Gerla, Mario, and Pau, Giovanni. Bloogo: Bloom filter based gossip algorithm for wireless ndn. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications* (New York, NY, USA, 2012), NoM '12, ACM, pp. 25–30.

[20] Aschenbruck, Nils, Ernst, Raphael, Gerhards-Padilla, Elmar, and Schwamborn, Matthias. Bonnmotion: A mobility scenario generation and analysis tool. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques* (ICST, Brussels, Belgium, Belgium, 2010), SIMUTools '10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 51:1–51:10.

[21] Ayad, Ibrahim, Im, Youngbin, Keller, Eric, and Ha, Sangtae. A practical evaluation of rate adaptation algorithms in http-based adaptive streaming. *Computer Networks 133* (2018), 90 – 103.

[22] Balk, A, Gerla, M, Sanadidi, M, and Maggiorini, D. Adaptive mpeg-4 video streaming with bandwidth estimation: Journal version. *vol 44* (2003), 415–439.

[23] Breslau, L., Cao, Pei, Fan, Li, Phillips, G., and Shenker, S. Web caching and zipf-like distributions: evidence and implications. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Mar 1999), vol. 1, pp. 126–134 vol.1.

[24] Carlucci, Gaetano, De Cicco, Luca, and Mascolo, Saverio. Http over udp: An experimental investigation of quic. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing* (New York, NY, USA, 2015), SAC '15, ACM, pp. 609–614.

[25] Crow, B. P., Widjaja, I., Kim, L. G., and Sakai, P. T. Ieee 802.11 wireless local area networks. *Comm. Mag. 35*, 9 (Sept. 1997), 116–126.

[26] d. S. Junior, D., Cardoso, I. F., and Momo, M. R. Tool-based mobile application applied to the monitoring system and flood alert. In *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems* (July 2015), pp. 348–351.

[27] De Coninck, Quentin, and Bonaventure, Olivier. Multipath quic: Design and evaluation. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies* (2017), ACM, pp. 160–166.

[28] Deng, Shuo, Netravali, Ravi, Sivaraman, Anirudh, and Balakrishnan, Hari. Wifi, lte, or both? measuring multi-homed wireless internet performance. In *Internet Measurement Conference 2014* (Vancouver, Canada, November 2014).

[29] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext transfer protocol – http/1.1, 1999.

[30] for double-blind review, Anonymized. TR - Improving QoE of ABR Streaming Sessions through QUIC Retransmissions. `https://drive.google.com/open?id=1E8PfspsHTTNMfRgA5iuY_SsiiOi53iio`, 2018.

[31] Ford, Alan, Raiciu, Costin, Handley, Mark, Barre, Sebastien, and Iyengar, Janardhan. Architectural guidelines for multipath tcp development. Tech. rep., 2011.

[32] Gilani, Z., Sathiaseelan, A., Crowcroft, J., and Pejović, V. Inferring network infrastructural behaviour during disasters. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)* (Jan 2016), pp. 642–645.

[33] Golaup, A., Mustapha, M., and Patanapongpibul, L. B. Femtocell access control strategy in umts and lte. *IEEE Communications Magazine 47*, 9 (September 2009), 117–123.

[34] Gomes, A., Braun, T., and Monteiro, E. Enhanced caching strategies at the edge of lte mobile networks. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops* (May 2016), pp. 341–349.

[35] Grassi, G., Pesavento, D., Pau, G., Zhang, L., and Fdida, S. Navigo: Interest forwarding by geolocations in vehicular named data networking. In *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (June 2015), pp. 1–10.

[36] H. Schulzrinne, S. Casner, R. Frederick, and Jacobson, V. RTP: A Transport Protocol for Real-Time Applications, 2017.

[37] Ha, Sangtae, Rhee, Injong, and Xu, Lisong. Cubic: a new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review 42*, 5 (2008), 64–74.

[38] Han, Bo, Qian, Feng, Ji, Lusheng, and Gopalakrishnan, Vijay. Mp-dash: Adaptive video streaming over preference-aware multipath. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies* (2016), ACM, pp. 129–143.

[39] Hayes, B., Chang, Y., and Riley, G. Omnidirectional adaptive bitrate media delivery using mptcp/quic over an sdn architecture. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference* (Dec 2017), pp. 1–6.

[40] Huang, T., Handigol, N., Heller, B., McKeown, N., and Johari, R. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proc. of IMC* (2012), pp. 225–238.

[41] Huysegems, Rafael, van der Hooft, Jeroen, Bostoen, Tom, Rondao Alface, Patrice, Petrangeli, Stefano, Wauters, Tim, and De Turck, Filip. Http/2-based methods to improve the live experience of adaptive streaming. In *Proceedings of the 23rd ACM international conference on Multimedia* (2015), ACM, pp. 541–550.

[42] Jacobson, Van, Smetters, Diana K., Thornton, James D., Plass, Michael F., Briggs, Nicholas H., and Braynard, Rebecca L. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies* (New York, NY, USA, 2009), CoNEXT '09, ACM, pp. 1–12.

[43] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., and Towsley, D. Measurement and classification of out-of-sequence packets in a tier-1 ip backbone. *IEEE/ACM Transactions on Networking 15*, 1 (Feb 2007), 54–66.

[44] Juluri, Parikshit, Tamarapalli, Venkatesh, and Medhi, Deep. SARA: Segment-aware Rate Adaptation Algorithm for Dynamic Adaptive Streaming over HTTP. In *IEEE ICC QoE-FI Workshop* (2015).

[45] Kakhki, Arash Molavi, Jero, Samuel, Choffnes, David, Nita-Rotaru, Cristina, and Mislove, Alan. Taking a long look at quic: An approach for rigorous evaluation of rapidly evolving transport protocols. In *Proceedings of the 2017 Internet Measurement Conference* (New York, NY, USA, 2017), IMC '17, ACM, pp. 290–303.

[46] Katsaros, K., Dianati, M., Tafazolli, R., and Kernchen, R. Clwpr – a novel cross-layer optimized position based routing protocol for vanets. In *2011 IEEE Vehicular Networking Conference (VNC)* (Nov 2011), pp. 139–146.

[47] Katsaros, K., Dianati, M., Tafazolli, R., and Kernchen, R. Clwpr — a novel cross-layer optimized position based routing protocol for vanets. In *2011 IEEE Vehicular Networking Conference (VNC)* (Nov 2011), pp. 139–146.

[48] Koponen, Teemu, Chawla, Mohit, Chun, Byung-Gon, Ermolinskiy, Andrey, Kim, Kye Hyun, Shenker, Scott, and Stoica, Ion. A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2007), SIGCOMM '07, ACM, pp. 181–192.

[49] Kurkowski, S., Navidi, W., and Camp, T. Constructing manet simulation scenarios that meet standards. In *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems* (Oct 2007), pp. 1–9.

[50] Langley, Adam, Riddoch, Alistair, Wilk, Alyssa, Vicente, Antonio, Krasic, Charles, Zhang, Dan, Yang, Fan, Kouranov, Fedor, Swett, Ian, Iyengar, Janardhan, Bailey, Jeff, Dorfman, Jeremy, Roskind, Jim, Kulik, Joanna, Westin, Patrik, Tenneti, Raman, Shade, Robbie, Hamilton, Ryan, Vasiliev, Victor, Chang, Wan-Teh, and Shi, Zhongyi. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (New York, NY, USA, 2017), SIGCOMM '17, ACM, pp. 183–196.

[51] Lederer, Stefan, Müller, Christopher, and Timmerer, Christian. Dynamic adaptive streaming over HTTP dataset. In *ACM MMSys* (2012), pp. 89–94.

[52] Mastorakis, Spyridon, Afanasyev, Alexander, Moiseenkox, Ilya, and Zhang, Lixia. Ndn technical report ndn-0028. Technical Report NDN-0028, NDN, January 2015.

[53] Mastorakis, Spyridon, Afanasyev, Alexander, and Zhang, Lixia. On the evolution of ndnsim: An open-source simulator for ndn experimentation. *SIGCOMM Comput. Commun. Rev. 47*, 3 (Sept. 2017), 19–33.

[54] McKeown, Nick, Anderson, Tom, Balakrishnan, Hari, Parulkar, Guru, Peterson, Larry, Rexford, Jennifer, Shenker, Scott, and Turner, Jonathan. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review 38*, 2 (2008), 69–74.

[55] Meisel, Michael, Pappas, Vasileios, and Zhang, Lixia. Listen first, broadcast later: Topology-agnostic forwarding under high dynamics. In *Annual conference of international technology alliance in network and information science* (2010), p. 8.

[56] Narasimhan, Priya, Drolia, Utsav, Tan, Jiaqi, Mickulicz, Nathan D., and Gandhi, Rajeev. The next-generation in-stadium experience (keynote). *SIGPLAN Not. 51*, 3 (Oct. 2015), 1–10.

[57] Nguyen, Sinh Chung, and Nguyen, Thi Mai Trang. Evaluation of multipath tcp load sharing with coupled congestion control option in heterogeneous networks. In *Global Information Infrastructure Symposium (GIIS), 2011* (2011), IEEE, pp. 1–5.

[58] NTT, Tokyo, Japan (February 2013). Great east japan earthquake and research and development for network resilience and recovery. itu workshop on e-health services in low-resource settings: Requirements and itu role. `https://www.nttdocomo.co.jp/english/corporate/ir/binary/pdf/library/presentation/110330/notice\_110330-1\_e.pdf`. Accessed online: August 23, 2018.

[59] Nygren, Erik, Sitaraman, Ramesh K., and Sun, Jennifer. The Akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev. 44*, 3 (Aug. 2010), 2–19.

[60] Paasch, Christoph, Barre, Sebastien, et al. Multipath tcp implementation in the linux kernel. Available from http://www.multipath-tcp.org.

[61] Paasch, Christoph, Detal, Gregory, Duchene, Fabien, Raiciu, Costin, and Bonaventure, Olivier. Exploring mobile/wifi handover with multipath tcp. In *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design* (2012), ACM, pp. 31–36.

[62] Paasch, Christoph, Ferlin, Simone, Alay, Ozgu, and Bonaventure, Olivier. Experimental evaluation of multipath tcp schedulers. In *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop* (2014), ACM, pp. 27–32.

[63] Peng, Qiuyu, Walid, Anwar, Hwang, Jaehyun, and Low, Steven H. Multipath tcp: Analysis, design, and implementation. *IEEE/ACM Transactions on Networking (ToN) 24*, 1 (2016), 596–609.

[64] Pesavento, D., Grassi, G., Palazzi, C. E., and Pau, G. A naming scheme to represent geographic areas in ndn. In *2013 IFIP Wireless Days (WD)* (Nov 2013), pp. 1–3.

[65] Piro, Giuseppe, Baldo, Nicola, and Miozzo, Marco. An lte module for the ns-3 network simulator. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques* (ICST, Brussels, Belgium, Belgium, 2011), SIMUTools '11, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 415–422.

[66] Raiciu, Costin, Barre, Sebastien, Pluntke, Christopher, Greenhalgh, Adam, Wischik, Damon, and Handley, Mark. Improving datacenter performance and robustness with multipath tcp. In *ACM SIGCOMM Computer Communication Review* (2011), vol. 41, ACM, pp. 266–277.

[67] Raiciu, Costin, Paasch, Christoph, Barre, Sebastien, Ford, Alan, Honda, Michio, Duchene, Fabien, Bonaventure, Olivier, and Handley, Mark. How hard can it be? designing and implementing a deployable multipath tcp. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (2012), USENIX Association, pp. 29–29.

[68] Raiciu, C., Handley M. Wischik D. Coupled multipath-aware congestion control. internet draft, internet engineering task force (january 2016).

[69] Rebecchi, F., de Amorim, M. Dias, Conan, V., Passarella, A., Bruno, R., and Conti, M. Data offloading techniques in cellular networks: A survey. *IEEE Communications Surveys Tutorials 17*, 2 (Secondquarter 2015), 580–603.

[70] Ricci, Robert, Eide, Eric, and The CloudLab Team. Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. *USENIX ;login: 39*, 6 (Dec. 2014).

[71] Saito, Hiroshi. Spatial design of physical network robust against earthquakes. *J. Lightwave Technol. 33*, 2 (Jan 2015), 443–458.

[72] Schulman, Aaron, and Spring, Neil. Pingin' in the rain. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference* (New York, NY, USA, 2011), IMC '11, ACM, pp. 19–28.

[73] Sodagar, I. The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia 18*, 4 (April 2011), 62–67.

[74] Sodagar, I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia 18*, 4 (April 2011), 62–67.

[75] Stenberg, Daniel. Http2 explained. *SIGCOMM Comput. Commun. Rev. 44*, 3 (July 2014), 120–128.

[76] Stewart, R., Xie, Q., Morneault, K., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and Paxson, V. Stream Control Transmission Protocol, 2017.

[77] Szabó, Géza, Rácz, Sándor, Bezzera, Daniel, Nogueira, Igor, and Sadok, Djamel. Media qoe enhancement with quic. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on* (2016), IEEE, pp. 219–220.

[78] Timmerer, Christian, and Bertoni, Alan. Advanced transport options for the dynamic adaptive streaming over http. *arXiv preprint arXiv:1606.00264* (2016).

[79] Tremblay, O., Dessaint, L. A., and Dekkiche, A. I. A generic battery model for the dynamic simulation of hybrid electric vehicles. In *2007 IEEE Vehicle Power and Propulsion Conference* (Sept 2007), pp. 284–289.

[80] ULC, Sandvine Incorporated. Global Internet phenomena report 2016, 2016.

[81] Vigneri, Luigi, Spyropoulos, Thrasyvoulos, and Barakat, Chadi. Streaming content from a vehicular cloud. In *Proceedings of the Eleventh ACM Workshop on Challenged Networks* (New York, NY, USA, 2016), CHANTS '16, ACM, pp. 39–44.

[82] Vulimiri, Ashish, Godfrey, Philip Brighten, Mittal, Radhika, Sherry, Justine, Ratnasamy, Sylvia, and Shenker, Scott. Low latency via redundancy. In *Proc. of CoNEXT* (2013), pp. 283–294.

[83] Wang, C., Rizk, A., and Zink, M. SQUAD: A Spectrum-based Quality Adaptation for Dynamic Adaptive Streaming over HTTP. In *Proc. of MMSys* (2016), ACM, pp. 1:1–1:12.

[84] Wang, Cong, Bhat, Divyashri, Rizk, Amr, and Zink, Michael. Design and analysis of qoe-aware quality adaptation for dash: A spectrum-based approach. *ACM Trans. Multimedia Comput. Commun. Appl. 13*, 3s (July 2017), 45:1–45:24.

[85] Wang, Lucas, Afanasyev, Alexander, Kuntz, Romain, Vuyyuru, Rama, Wakikawa, Ryuji, and Zhang, Lixia. Rapid traffic information dissemination using named data. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications* (New York, NY, USA, 2012), NoM '12, ACM, pp. 7–12.

[86] Wang, Lucas, Afanasyev, Alexander, Kuntz, Romain, Vuyyuru, Rama, Wakikawa, Ryuji, and Zhang, Lixia. Rapid traffic information dissemination using named data. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications* (New York, NY, USA, 2012), NoM '12, ACM, pp. 7–12.

[87] Xiao, Mengbai, Swaminathan, Viswanathan, Wei, Sheng, and Chen, Songqing. Evaluating and improving push based video streaming with http/2. In *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (2016), ACM, p. 3.

[88] Yuriyama, M., and Kushida, T. Sensor-cloud infrastructure - physical sensor management with virtualized sensors on cloud computing. In *2010 13th International Conference on Network-Based Information Systems* (Sept 2010), pp. 1–8.

[89] Zhang, Lixia, Afanasyev, Alexander, Burke, Jeffrey, Jacobson, Van, claffy, kc, Crowley, Patrick, Papadopoulos, Christos, Wang, Lan, and Zhang, Beichuan. Named data networking. *SIGCOMM Comput. Commun. Rev. 44*, 3 (July 2014), 66–73.

[90] Zhang, Lixia, Estrin, Deborah, and Burke, Jeffrey. Ndn technical report ndn-0001. Technical Report TR001ndn, NDN, November 2011.

[91] Zink, M., Schmitt, J., and Steinmetz, R. Layer-encoded video in scalable adaptive streaming. *IEEE Transactions on Multimedia 7*, 1 (Feb 2005), 75–84.