DESIGN AND IMPLEMENTATION OF REAL-TIME CLOUD-ASSISTED SYSTEMS

BY MEHRNAZ TAVAN

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Roy D. Yates and Professor Dipankar Raychaudhuri

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2017

ABSTRACT OF THE DISSERTATION

Design and Implementation of Real-Time Cloud-Assisted Systems

by Mehrnaz Tavan

Dissertation Director: Professor Roy D. Yates and Professor Dipankar Raychaudhuri

The emergence of various mobile devices (e.g., smart phones, smart wearables, tablets, etc.) has eliminated the location and timing constraints on access to a variety of services including social networking, web search, data storage, video streaming, and gaming. Limitations on resources in mobile devices such as battery life, storage, and processing power along with the scalability requirements of such services have motivated companies to rely on cloud computing. Cloud computing has created a new paradigm for mobile applications enabling access to computing and storage resources in various locations on demand. This has drastically transformed the quality of experience (QoE) for the users and increased the popularity of cloud computing based applications. This thesis studies the design and implementation of solutions for support of advanced real-time applications on mobile devices with stringent constraints on responsiveness and throughput using cloud systems.

One of the main advantages of cloud computing platforms is to provide a logically centralized but geographically distributed database with on demand access. This characteristic can be leveraged to develop a network layer solution for the problem of maintaining connectivity and global reachability for mobile nodes. In the second chapter of this thesis, the design and analysis of a connected vehicle architecture, called FastMF, optimized for the deployment and support of advanced real-time services is

examined. Motivated by current in-vehicle needs such as on demand video streaming and safety applications and with a look to a future with cloud-assisted autonomous driving and virtual reality goggles, we identify PHY, MAC, network, and transport layer requirements to support these services. Furthermore, we discuss the issues with current fixed-host/server IP architecture in providing service for mobile nodes and the shortcomings of current cloud systems in terms of reacting slowly to coarse-grained changes in the network.

To address these issues, we design a scalable, mobility-centric solution optimized for high dynamicity based on MobilityFirst architecture. Our solution aims at providing a mechanism for multi-hop bidirectional path discovery between vehicular nodes and Internet gateways and enhancing global reachability and seamless connectivity of vehicular nodes. To develop vehicular clustering for the purpose of obtaining the best network access and to increase cluster stability, we start with an analytical evaluation of vehicular mobility models followed by link lifetime analysis and description of mobility event during a contact between two cars. Our proposed distributed clustering algorithm forms tree-based clusters among nodes with similar mobility attributes. We integrate our proposed clusters into named service layer supported by globally available cloud servers. The distributed global name resolution service (GNRS) implemented in the MobilityFirst architecture along with efficient vehicular clustering enable scalable services with seamless connectivity and global reachability.

MobilityFirst services are provided mainly by routers with long-living links and relatively consistent connectivity to a network infrastructure. To support dynamic scenarios including connected vehicles, we enable nodes with short-living links and intermittent access to the network infrastructure to act as MobilityFirst routers and extend the MobilityFirst services to multiple-hops. The proposed scheme, assigns unique names to clusters of vehicles. These names are independent of the locations of the clusters and the interfaces. By aggregating vehicles in clusters, assigning unique identifiers to them, and extending GNRS service to ad hoc mode, mobile nodes can benefit from multi-homing and other services which are currently supported by MobilityFirst in a seamless way only for nodes with one-hop access to Internet. The proposed method can be added on top of IP or in an independent way as a network service for highly

dynamic scenarios. By large scale simulation of various representative scenarios including content delivery, web content retrieval, and store-and-forward service, we show the flexibility and efficiency introduced by our design and the improvement in throughput and reduction in delay that FastMF provides.

Another main advantage of the cloud computing platform is providing access to enhanced computing resources. This enables providing more computationally complex services such as cloud gaming through thin client devices. In this work, we investigate the design and analysis of a cloud gaming platform with optimized QoE for players. In cloud gaming, the game status updating and frame rendering tasks have migrated to cloud resources. The players receive the rendered game scenes which are streamed to their thin-client devices in real-time over the Internet. One of the main issues at stake is maintaining responsiveness while having a smooth display of the game. What makes this problem more challenging is that the existing solutions for video streaming is not designed for latency-sensitive interactive applications. Furthermore, QoE is a subjective metric and to the best of our knowledge, no quantitative modeling for QoE in interactive systems exists.

In this regard, we introduce a novel application layer solution which efficiently adapts itself to variations in the channel and server load. Our developed protocol performs online frame selection to dynamically reduce traffic rate and proactively conserve network resources for more fresh frames. This design performs optimization of responsiveness and display smoothness by evaluating the freshness of information perceived by player through video frames. Furthermore, we develop a mathematical analysis of interactive applications and provide an analytical model for interactive real-time systems. Using this model, we develop an objective QoE metric that suits the requirements of gaming systems. This includes the minimization of age of information that the client perceives subject to mitigating frozen screens and maintaining constant display frame rate and smoothness. Through conducting extensive analytical modeling and network simulation, we show the applicability of proposed model and protocol in improving the gaming experience for both single-player and multi-player games.

Acknowledgements

Looking back at the last six years of my life, I cannot be happier about my decision on pursuing PhD degree in Rutgers University. Struggling to find the right research problem, working on various and equally challenging topics, learning advanced maths, computer science, and engineering skills, and developing personality traits of researchers are all parts of the great achievements that I owe to Rutgers.

I have always been inspired by the depth of knowledge, advanced analytical insights, and high standards of professionalism and research methods of my advisor, Prof. Roy Yates to whom I would like to express my deepest gratitude. This thesis and more importantly, my growth into an independent researcher, would not have been possible without his constant guidance and support. Instead of instructing different steps of the solution, he helped me grow the ability to see the big picture, precisely articulate the main question in each problem, and rigorously tackle the research challenges. Furthermore, he always provided me with invaluable guidance on how to present my work and ideas.

When I joined MobilityFirst, I had mostly theoretical skills. I would like to express my greatest respect and admiration to Prof. Dipankar Raychaudhuri for trusting me and providing me with the support and opportunity to grow my programming skills and develop experimental abilities. Through his deep engineering insights, thorough knowledge on computer networking, and his great advice and support, I was able to transition into a different research area successfully.

Furthermore, I would like to thank Prof. Bajwa who provided me with his great support and help during the first research project that I worked on. Without his help, I was not able to work on an interdisciplinary topic between machine learning and information theory.

I would like to thank Prof. Predrag Spasojevic and Dr. Daniel Reininger for participating in my thesis committee and providing me with insightful comments and

encouragement. During my life in WINLAB, I was constantly asking questions from Ivan Seskar and I cannot thank him enough for all the great insight regarding wireless networks and career development skills that he provided. I would like to acknowledge the academic and technical support provided by Prof. Gajic and the staff at WINLAB and ECE department.

My life in Rutgers would not be as great without the support and help from great friends that I made from WINLAB, ECE, and other departments. In addition, I would not be able to reach this stage without the countless sacrifices that my family have made. Everything that I have accomplished and will ever achieve is due to their love and support.

Dedication

To any free spirit willing to fly thousands of miles away from home and climb the highest mountains in pursuit of science

Table of Contents

List of Tables

List of Figures

Chapter 1
INTRODUCTION

1.1 Overview

One of the key enabling factors in the evolution of mobile services is the cloud computing platform which offers the following advantages:

- The clients are not under the burden of creating and maintaining infrastructure or advanced in-device processing units for their computation needs.
- The globally distributed sites provide scalability and on-demand availability of the service.
- The logically centralized servers can aggregate and process data from multiple network entities involved in a service and make decisions knowing the dependencies among entities involved.

Cloud-assisted strategies face challenges related to networking requirements, mobility of the nodes, reliability of the links and computing resources, and inherent sensitivity of services to delay and congestion. Companies are hesitant to migrate their applications and services with stringent latency restrictions to cloud servers unless they are assured that the quality of experience (QoE) for their customers does not deteriorate.

In the remaining parts of this work, the intrinsic challenges and proposed solutions for two cloud-assisted applications, namely connected vehicles and cloud-assisted gaming, will be discussed.

1.2 Connected Vehicles

The growing popularity of smartphones and networkable vehicles has resulted in a significant increase in network utilization in challenging high mobility scenarios with

vehicular nodes. Throughout this work, a vehicular node can be a passenger's smartphone or a networking device installed in the vehicle. There are two main emerging categories of services [1] for vehicular nodes:

1. Intelligent Transportation Systems (ITS): The emergence of sensor-equipped vehicles with communication capabilities creates an opportunity to collect information about traffic congestion and road status in real-time by crowd-sourcing. After developing models based on gathered data, periodic or event-driven safety or traffic related messages can be disseminated among vehicles with relevant geographic or other attributes. Examples of such safety messages include emergency and traffic congestion warning, possible reroute options, sudden stop warning, notification on approaching emergency vehicle, lane changing assistance, and intersection coordination. In safety applications, most of the messages consist of only a single or very limited number of small packets. Often times, the same message will be broadcasted or multicasted to a group of vehicles.

2. In-vehicle Internet Access: Infotainment services [1] such as audio/video streaming, sharing front view of neighboring cars, sending point of interest notification, and updating maps along with automatic transfers of information between cloud servers and related in-vehicle devices for autonomous driving require high throughput links and long-living sessions. In this case, the data packets are mostly intended for a single vehicle or a very limited number of vehicles. Furthermore, most of the time, the messages consists of a stream of mid-size or large packets.

Providing reliable services in these scenarios is intertwined with intelligent usage of urban infrastructure (e.g., multi-homing) as well as leveraging available peer-to-peer WiFi links by forming ad hoc networks among vehicular nodes (i.e., vehicular ad hoc networks (VANETs)). Multi-hop forwarding within VANETs facilitates providing high quality services to nodes in weak coverage zones, compensating for the limited radio range in vehicles, or adopting more cost-efficient data transfer strategies (e.g., cellular offloading) [2]. Figure 1.1 illustrates an example of a smart city where VANETs help in exchanging data among vehicles using peer-to-peer WiFi links or providing indirect access to network infrastructure for cars without direct access.

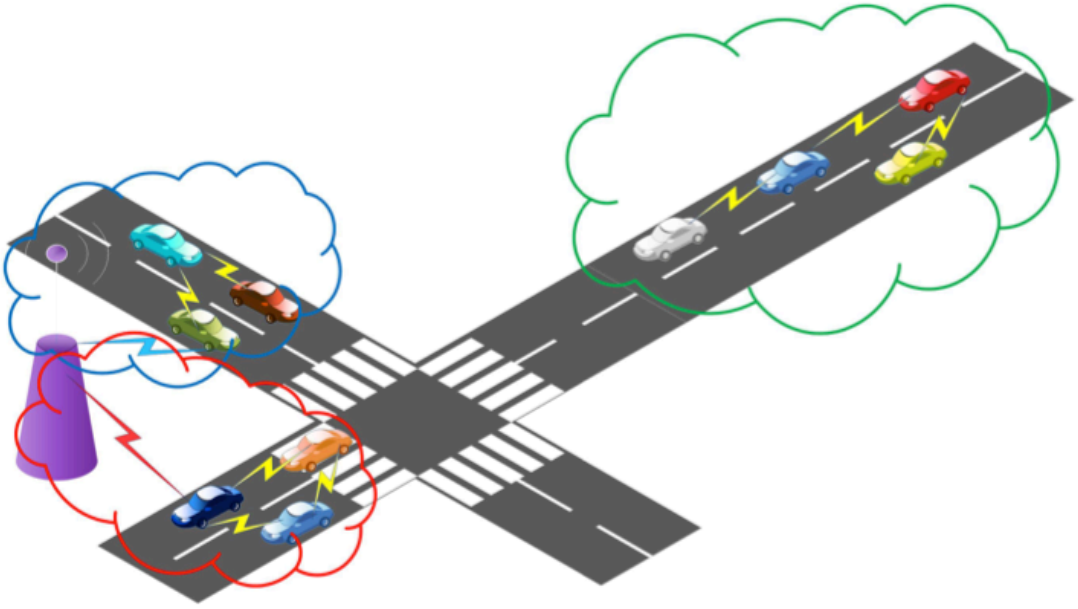

Figure 1.1: An example of a smart city with enhanced accessibility and reachability through intelligent usage of urban infrastructure as well as leveraging available peer-to-peer WiFi links

With the envisioned growth in vehicular infotainment deployment, the underlying network should support global reachability, multi-homing, and low latency design for mobile nodes. Seamless connectivity for vehicles faces various challenges:

1. **PHY/MAC layer**: Fast changing topologies, short living links, intermittent connectivity, unpredictable network loads, and potential broadcast storms impose challenges on connected car PHY/MAC layer design. In most current applications, location dynamicity triggers broadcast of control traffic. Consequently, VANETs with unstable links are often prone to congestion. Link stability is correlated with mobility characteristics of the vehicles and the road/city structure. Geographic proximity does not necessarily results in long living communication (e.g., links between cars in opposite lanes in a highway). The interference among nodes in close proximity limits the benefits of multi-hop communications. Most of the proposed solutions for node dynamicity are restricted to local information transfer (e.g., smart flooding using location information [3] and GeoServer assisted multicast [4]), dissemination of popular content, or delay-tolerant scenarios (e.g., carry and forward).

2. **Network layer**: The default assumption behind most of today's Internet protocols is exchange of data among fixed nodes using a host-centric approach where each node is identified with its IP address. The IP address depends on the location of node's network attachment point. However, with the shift in usage statistics in favor of mobile platforms, services, and devices, mobile entities often change their IP address multiple times while most of their services are still in progress. This further complicates maintaining the quality of service. Identification of nodes with IP addresses that are location-dependent has resulted in identity-location conflation problem in IP networks [5]. Most of the proposed solutions for issues incurred by IP are restricted to specific channels (e.g., cellular providers), using an overlay network, or using an application layer solution without cooperative support from the network [6, 7]. Application layer solutions incur overhead [8] and solutions with overlay network such as MobileIP [5] and NEMO [9] suffer from the routing bottleneck.

3. **Transport layer**: Current transport layer protocols revolve around the idea of pushing the related functionalities to the end-host. This concept has resulted in end-to-end reliability paradigm and flow/congestion control protocols such as TCP and its variants. The recovery from congestion/loss in these designs happens after an additional delay induced by end-to-end feedback. Mitigating congestion and packet loss through link-level information is more efficient due to faster discovery of such impairments. In addition, the recovery mechanism often consists of the session restart and retransmission of in-flight packets. Most of the recovery mechanisms are based on mitigating congestion since in the conventional fixed-node Internet, the main impairment is congestion [8]. As a result, these mechanisms are not applicable fully for mobile nodes. Unnecessary retransmissions due to large backlogs at the routers are another disadvantage of this design.

4. **Multi-homing**: Although in today's wireless world, it is very common for a device to be in the coverage area of different access technologies (e.g., WiFi+LTE, WiFi+Bluetooth, DSRC+LTE, etc.,) at the same time and be equipped with multiple network interfaces, multi-homing enabled systems still face major challenges imposed by network layer design. Identifying different active interfaces of

a device is a major hurdle since each interface has a different IP address. Current solutions are in transport or application layer and cannot be applied to the en-route entities specially bifurcation routers.

5. **Reachability**: One of the key requirements of maintaining seamless connectivity for mobile nodes is that the server must have access to the most recent network attachment point that the client device, through one hop or multiple peer-to-peer hops, is associated with. In current systems, only nodes with one hop connection to network infrastructure are directly reachable by other nodes. If a node switches the interface through which it is connected to the Internet, all ongoing sessions will be interrupted and the mobile node must reinitiate the query through new interface [8].

The prevalence of discussed issues among Internet services and applications has motivated numerous researchers to think outside the TCP/IP box and propose new network layer solutions and service abstractions [8, 10–12]. Furthermore, new technologies such as routers with storage, network function virtualization (NFV), and software defined networking (SDN) have helped in the realization of diverse designs [13].

In this work, we tackle the problem of providing an in-vehicle Internet access framework in presence of dynamic links through a comprehensive solution called FastMF.

After identifying the requirements in having in-vehicle Internet accessibility and the characteristics of the issues faced in connected vehicles, we developed a new distributed clustering optimized for high dynamicity. The objective of this distributed cluster formation algorithm is providing a mechanism for multi-hop bidirectional path discovery between vehicular nodes and Internet gateways. Under this algorithm, we observe significant improvements in terms of connectivity, overall throughput, and average latency.

Through exchange of location information with neighboring nodes, periodical estimation of the mobility parameters of neighbors, and detecting nodes with similar mobility attributes, vehicles form stable clusters. These clusters are adaptable to the mobility pattern of the nodes and network congestion conditions and can extend network connectivity. To tackle topology instability while mitigating control traffic overhead, clusters are formed in the shape of tree and react to coarse-grained changes in

links. The proposed distributed tree-based clustering algorithm obtains real-time flexibility with respect to channel status with no overhead cost. Each node has a partial view of the cluster sufficient for forwarding data. To optimize clusters based on not only stability but also connectivity to the infrastructure and to limit cluster size based on channel conditions, our clustering algorithm minimizes vehicle to infrastructure path length. The path length is measured using Expected Transmission Time (ETT) [14].

This is followed up by designing network service abstractions to solve the current shortcomings of TCP/IP networks in providing accessibility (ability of a mobile node to connect to the backbone Internet) and reachability (access to most recent address of a mobile device from any server) for mobile nodes. Our solution is in the realm of "named-object services" [8]. Specifically, we build on the global name resolution service (GNRS) implemented in MobilityFirst [8,13,15] which provides an in-network real-time distributed database of mappings between object IDs (GUID) and their current network addresses (NAs). In general, the NA of an entity is an identifier that corresponds to a routable network path to that entity. The NA can be a physical network address; however, it may also be the address of a subnet or even an autonomous system (AS), such that packets addressed with this NA will reach a router that can forward packets to the network entity. By assigning a GUID to each network entity, including devices, interfaces, and clusters of vehicles, we separate names from addresses of an entity. Furthermore, GNRS makes the network address for each entity available to the whole Internet. GUID enhances trustworthiness by using self-certifiable names. The end-hosts can authenticate each other by two-way challenge response procedure.

MobilityFirst is designed mainly for mobile nodes with single-hop and relatively consistent connectivity to a network infrastructure. For MobilityFirst to be applicable to vehicular networking, we need to address the following challenges: i) The routing schemes must be modified to reduce the control traffic overhead created by short lifetime links. ii) The GNRS operations must be revised to include service for nodes with no direct link to a network attachment point. iii) All named-object services must be extended to include a continuously changing cluster of nodes. iv) The reliability algorithm must adjust to situations with extremely intermittent links.

In the final step, we integrated clusters of vehicles with same mobility pattern into the MobilityFirst architecture to increase the session lifetime beyond one-hop coverage

interval. This is accomplished by assigning unique IDs to clusters and extending the naming service in MobilityFirst to support detached cluster members through indirect associations. Internet gateways will receive the identity of all members of a cluster through the messages sent by a subset of cluster members directly associated to them. These gateways periodically update GNRS of the identity of associated clusters and their members to enhance tracking of vehicular nodes. This extension to MobilityFirst provides all cluster members, including disconnected or poorly connected nodes, with an opportunity to obtain an indirect association to Internet gateways that provide a reachable NA for these nodes in the GNRS. In name-based architectures where IDs are globally unique and routable, this extension enhances global reachability.

Availability of the reachability information in real-time in the network layer extends the session lifetime specially for mobile nodes with intermittent direct Internet access. The name-based forwarding and in-network store-and-forward mechanisms help recover data, avoid retransmitting in-flight packets, resume data transfer through alternative paths without noticeable interruptions, and access content with less delay [11, 12] for nodes in poor coverage areas. While a mobile node cannot access Internet directly or indirectly, its in-transit data is stored in the last NA the node was mapped to in the GNRS. As soon as GNRS updates the mapping for the node, its data will be forwarded from old NA to new NA . To address the transport layer challenges, a hop-by-hop reliability mechanism is deployed. However, in case of high dynamicity, the checkpoint for reliability is the same as the network attachment point mapped for cluster.

Through simulation of a representative set of data delivery scenarios, we illustrate the capability of FastMF in reducing in-network load, avoiding congestion, and seamless delivery. Results from ns-3 experiments illustrate the improvements in throughput for downloading large files obtained by clustering and multi-hop transfer of data. In addition, experiments with interactive web-browsing scenarios indicate a significant delay reduction in various mobility scenarios.

1.3 Cloud Gaming

One of the most popular networked services is gaming [16]. Current popular games designed for traditional platforms such as desktop computers and gaming consoles, are

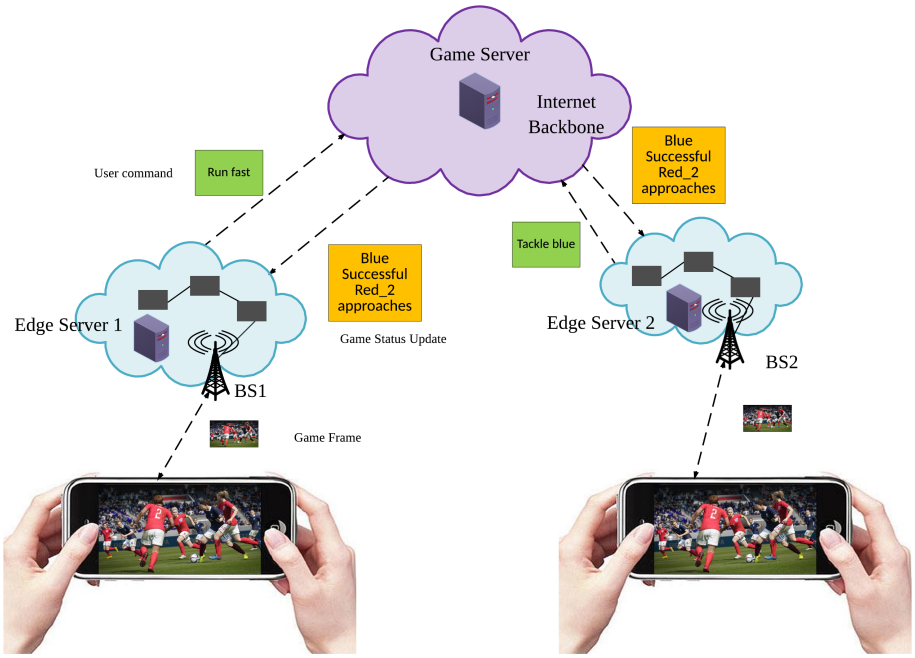

Figure 1.2: Cloud gaming system architecture (green packets are user commands and yellow packets are game status updates).

appealing to players due to elaborate frame rendering, immersive gaming experience, augmented functionalities, and dynamic scenes. We cannot expect to receive the same quality from more constrained platforms such as smartphones and tablets.

With recent improvements in the capabilities of cloud infrastructure and quality of wireless networks, mobile cloud computing systems have become an attractive choice for hosting computationally complex services including latency-sensitive online games [17–20]. In contrast to traditional platforms in which most of the computations are conducted in the client device, the majority of the complex tasks in cloud-assisted designs including game status updating, graphic rendering, and video encoding are run in one or multiple data centers. Mobile clients submit actions through an access network to a game server. The game server generates video frames at a constant frame rate. The rendered game scenes are streamed to thin client devices from cloud servers. At the mobile device, the display of these frames represent game status updates. Figure 1.2 illustrates an example of a cloud-assisted multi-player gaming system.

Cloud gaming has numerous advantages for both players and game developers [17,18]

including: a) *Cost reduction*: Migrating the game status computation and graphic rendering to a remote server enables players to use thin devices. b) *Platform independence*: A more diverse set of games can be played on the same device. c) *Piracy prevention*: The game source code is stored only in the game server. d) *Resource enhancement*: Servers have more processing power and memory as compared to mobile devices.

However, designing a cloud gaming system that provides a high QoE for players involves numerous challenges. QoE metric analysis is difficult due to its subjective nature and its dependence on various quantitative and content-based parameters. There are two important factors affecting QoE [21]. First, since game updates are streamed back to the thin client in the form of video frames, the design must maintain a high user-perceived video quality. A user's QoE is degraded when there are imperfections in the sequence of displayed frames; neither frozen frames nor frame rate changes are desirable. Second, the interactive and real-time nature of cloud gaming, particularly its dependence on the online generation of game updates, imposes stringent delay constraints on the system. In this setting, video playback delays of even a handful of frames constitutes an outage. Players are aggravated by delayed viewing of their commands. Short-duration frame freezes are also undesirable. Variations of latency, called jitter, can also have a drastic impact on QoE.

As an example, by increasing the frame rate in gaming systems, the perceptual quality of experience for players is supposed to improve [22]. However, high frame rate systems require high throughput in the range of 2-6 Mbit/s [22]. If the network cannot support this throughput, frame loss, frozen screen, and significant delays will happen which reduce the QoE significantly. If the frame rate is too low, the player will receive delayed and sparsely sampled game updates which has a dramatic impact on his/her gaming experience. Furthermore,the movements of game characters will appear as consecutive snapshots with no smoothness. This example shows that designs proposed to improve QoE in video streaming systems may not necessarily create the same perceptual quality in interactive systems. Furthermore, it shows the complex trade-offs and modeling challenges that exist in a cloud gaming system.

Similar to other distributed multimedia applications, network conditions have a significant impact on the user experience in cloud gaming systems. Buffering and adaptive bitrate (ABR) streaming protocols such as Dynamic Adaptive Streaming over HTTP

(DASH) and HTTP Live Streaming (HLS) [23] have been long used for masking the networking latency in video streaming. However, these methods cannot be conveniently applied to online gaming. Buffering, which is very effective in maintaining display smoothness in video streaming over channels with varying conditions, incurs significant delay and noticeable lag with respect to user commands in interactive applications. ABR operates based on encoding video segments of size at the order of seconds using different bit rates. Applying ABR to gaming which is developed in real-time, significantly increases system complexity and delay.

An important QoE metric in cloud gaming is the response time which corresponds to the elapsed time between the generation of an action and the displayed result of that action. In fact, the response time reflects the accumulated delay in different components including network and processing delays. In cloud gaming, the sensitivity of the users to response-time delay depends on the type of the game; a low-latency game such as a first person shooter, which is the focus of this work, requires a response time on the order of 100 ms [18, 24].

We try to tackle QoE evaluation by direct examination of the missing frame process. The users observe the games through a fixed frame rate video. We aim at maximizing the freshness of the data perceived by the client. In this work, we introduce a new model for low-latency (sub 100 milliseconds) gaming enabled by edge cloud game servers. We accomplish this goal in the following ways:

- We develop a Markov model which characterizes the frame delivery process in low-latency edge cloud gaming systems and reflects the player's watching experience. This model evaluates susceptibilities of highly interactive cloud gaming systems to lag, frame loss, and frozen screens.

- The main objective of a cloud gaming system is the "timely" update of players regarding the game status, where game updates are reflected in displayed frames. As a result, we follow the concept of age of information (AoI) introduced in [25] to characterize the system performance based on the impact of missing frames. AoI, as an objective QoE metric, focuses on application-layer events instead of transport-layer events. Based on the developed Markov model, we derive a simple formula for the average age of a tightly synchronized low-latency mobile gaming

system in which the inter-frame period is a significant contributor to the system latency. The freshness of frames reduces as they wait for medium access, retransmissions due to channel errors, backoff due to the activity of other wireless transmitters, and screen refresh instants. The age metric incorporates both the lag in response time and the latency associated with periodic framing. Status age measures frame freezes seen by the player when frames go missing. Our proposed metric, is in terms of both interaction delay and stream quality and captures the effect of stochastic network and processing delay variations on the game's responsiveness and video display quality.

- We develop a novel application layer protocol for real-time interactive systems to mitigate the impact of network latency, client device constraints, and server access limitations. Our protocol improves streaming quality and system responsiveness adapted to channel conditions. The adaptation indicates protocol's capability in proactively sensing lag-inducing events and dynamically adjusting the traffic load. In exchange, we are willing to introduce ideally sufficiently minor and isolated frame drops as long as the impact of these drops stays below the level of sensitivity of human eye and can be mitigated by the interpolation capability of human cognitive system. This protocol eliminates the aggregation of system-induced lags over multiple frames, avoids noticeable stalling events, and conserves the bandwidth for more fresh frames.

- We validate our system design by ns-3 simulation of low-latency edge cloud gaming system (single-player as well as multi-player). Our results prove that our analytical model provides an accurate performance characterization. In addition, we show how careful synchronization of the game server and the mobile client display can improve performance by roughly 20 percent.

1.4 Organization

The rest of this thesis is organized as follows. In chapter 2, we present our cloud-assisted solution for enhancing accessibility and reachability for poorly connected vehicles. In chapter 3, we present our analytical modeling, developed QoE metric, and designed protocol for low-latency (sub 100 milliseconds) edge cloud gaming.

Chapter 2

Connected Vehicles under Information-Centric Architectures

2.1 Introduction

The proliferation of mobile computing devices has motivated us to investigate and design network services for highly dynamic environments such as vehicular nodes. With the impairments mentioned in section 1.2 in consideration, we propose FastMF to tackle the problem of providing high-throughput in-vehicle Internet access in presence of dynamic links, uncertain network load, and insufficient infrastructure. Our proposed solution consists of integration of vehicular ad hoc networks (VANETs), formed by our proposed distributed cluster formation algorithm, into the Internet backbone, equipped with our proposed named-object services. Our distributed cluster formation algorithm is a mechanism for multi-hop bidirectional path discovery between vehicular nodes and Internet gateways.

Vehicles possess certain characteristics [26] which indeed support ad hoc network formation. First, their mobility patterns on streets and highways are well-structured, predictable, and more constrained compared to those for other Mobile Ad hoc Networks (MANETs) [1]. Second, adjacent vehicles moving on the same road typically follow correlated trajectories. Third, reliable location and velocity information is available using GPS. Finally, there is potential support from in-vehicle infrastructure including WiFi-enabled cars, passengers' smartphones, energy supplies for charging nodes, and various computationally powerful devices [1].

Vehicular nodes have speeds at least an order of magnitude higher than nodes in traditional MANETs. Practical studies of vehicular links have shown lifetimes around 10-20 seconds [26]. In fact, to implement multi-hop forwarding, there are well-recognized

VANET-specific challenges [26] including short-lifetime links and connection disruptions [9] due to fast changing topology, being prone to channel congestion due to exchanging more control data, the broadcast storm problem [6], and unpredictable network load due to random density of nodes.

Being designed for networks of fixed entities, the IP architecture is not well suited for dynamic scenarios [11, 15, 26] with intermittent connectivity, frequent changes in network address (NA), and unreliable wireless links [1]. In addition, mobile nodes in IP networks suffer from the identity-location conflation problem due to employing the same name to identify an interface and its NA [5]. In general, the NA of an entity is an identifier that corresponds to a routable network path to that entity. The NA can be a physical network address; however, it may also be the address of a subnet or even an autonomous system (AS), such that packets addressed with this NA will reach a router that can forward packets to the network entity. When a route fails, since IP is based on best-effort delivery policy, all in-flight data that had arrived at the previous network address of the mobile device are dropped.

The dependence of traditional Internet on fixed-host/server model resulted in a significant emphasis on pushing most of the functionalities, including TCP's reliability and congestion control mechanisms, to the end hosts [27]. The dependence of TCP on the IP address for identification of end-hosts, relating packet loss to congestion instead of link failure, induced delays due to slow start instead of switching the used interface, and session interruptions due to the change of end-host's IP address prove further the inadequacy of TCP for mobile nodes.

IP networks provide services to mobile nodes using solutions containing overlay networks or extensions applicable only for local exchange of short messages. In MobileIP, data is indirectly delivered to the mobile node through a pre-assigned home agent [5]. This indirect routing system generates considerable control traffic overhead (e.g., IP encapsulation) and induces extra latency. In IPV6, the end nodes receive updated addresses of mobile nodes from home agent. However, this information is not available to other en-route network entities [9]. NEMO [9], a mobility-friendly extension of MobileIP, does not perform well in vehicular scenarios and requires support from a complex infrastructure [9]. The Wireless Access in Vehicular Environments (Wave) protocol stack supports IP-less exchange of short safety and traffic related messages

without the need for connection setup operations. However, it needs to work alongside the TCP/IP stack for non-safety applications and large file transfers [27]. Furthermore, it does not support multi-hop data forwarding. Wave's physical layer, dedicated short range communications (DSRC), is not very scalable [28] and can only cover a limited radio range before being congested.

Most of the algorithms proposed for VANETs [6, 7] use an overlay solution that is strongly entangled with IP addressing. Recently, projects such as [8, 10–12] intend to redesign the Internet-wide architecture to satisfy the needs of mobile nodes (e.g., location-independent communication) in network layer. Based on these designs, alternatives for VANETs have emerged [9, 12, 15, 29, 30]. The structural shift in data forwarding paradigms introduced in these designs supports seamless connectivity for mobile nodes and multi-homing services which would otherwise face difficulty in traditional Internet architectures.

The Host Identity Protocol (HIP) [29] adds a naming layer between transport and network layers only at the sender and receiver nodes, and identifies each end host by a host ID. In [30], the Locator/Identifier Separation Protocol (LISP) assigns two separate IP addresses to each end host's name and NA and uses a name-to-address resolution service. In [12], various designs for dissemination of popular contents among vehicular nodes based on the Named Data Networking architecture (NDN) [10] are discussed. In NDN, the routers forward data based on content IDs. A major challenge in designing VANETs based on the NDN architecture is the scalability of content discovery and routing [12]. In [9], a location-based content discovery mechanism, called Navigo, is introduced for integration of VANETS into NDN architecture. In Navigo, the location of content producer, when discovered after initial flooding, is added to the data chunks and is used by intermediate nodes for guiding further requests for the same content. For efficient delivery of popular content, Navigo is an advantageous solution. However, for general improvement of in-vehicle Internet access including interactive web browsing or downloading personal files, the initial flooding to locate the data source produces significant overhead [12]. Furthermore, mobility of the data producer or consumer results in frequent repetition of data source locating procedure.

To provide seamless connectivity and efficient delivery in presence of dynamic links,

our designed protocol leverages an in-network real-time distributed database of mappings between object IDs and their current NAs (e.g., global name resolution service (GNRS) implemented in the MobilityFirst [8, 13, 15]).

In MobilityFirst [8], the concepts of identity-location separation and name resolution service introduced in LISP [30] and content ID in NDN were incorporated and evolved further by assigning globally unique identifiers (GUIDs) to all network entities and using logically centralized but physically distributed cloud servers to make the most recent mapping between ID and NA of any network entity accessible to other network entities. Compared to initial flooding mechanism in NDN, MobilityFirst leverages the available distributed cloud computing resources and builds GNRS to identify the location of data producers. Compared to HIP, where resolution only happened in the end host, any network entity have access to GNRS and can perform routing based on GUIDs. The concept of globally available name resolution service is very similar to Domain Name System (DNS) [31] which maintains the mapping between URL and IP address. However, DNS is in application layer and URLs are hierarchical whereas GNRS is in the network layer, and GUIDs are flat.

Using cloud servers for locating mobile nodes was previously proposed in GeoServers [4] framework. The servers keep track of the geographic location of vehicles in a limited area. Information maintained in GeoServers are used for local distribution of safety and traffic related messages. Our design enables global reachability, seamless connectivity, and possibility of transmitting over the globally optimum routes.

2.2 Contribution

In this work, we present the design and evaluation of FastMF, a framework for providing high-throughput in-vehicle Internet access and seamless connectivity [15]. FastMF extends both accessibility of the Internet for vehicular nodes and reachability of vehicular nodes from any remote server. Driven by the fundamental problems encountered due to short-living links, intermittent routes, identification and location conflation in IP networks, and domination of fixed-host/server paradigm in today's Internet, we propose a simple yet effective solution to provide access to advanced services for vehicular nodes with varying link conditions.

MobilityFirst provides seamless connectivity for mobile nodes through routers with enhanced functionalities and reliable attachment to the network infrastructure that are connected with each other through long-living links. Nodes with one-hop access to MobilityFirst infrastructure can leverage its services. If this one-hop access is lost, routers store in-flight data and wait until the node gains a new one-hop access. The following scenarios are satisfied fully by MobilityFirst: i) If the nodes have long-living high-throughput one-hop access to MobilityFirst infrastructure ii) If the nodes with poor one-hop access provide only delay-tolerant services or due to negligible relative mobility, can be fully satisfied by tethering. Otherwise, the detached nodes are hidden from GNRS and they cannot have seamless connectivity.

To provide seamless connectivity for vehicular nodes with intermittent links over one or multiple hops, we integrated clustering and ad hoc networking into the MobilityFirst architecture [32]. Our design enable vehicular nodes to function as MobilityFirst routers by forming clusters and reducing the impact of node dynamicity.

To improve in-vehicle Internet accessibility, FastMF develops a mechanism for multi-hop bidirectional path discovery between vehicular nodes and Internet gateways based on a distributed cluster formation algorithm. The characteristics of the PHY/MAC layer issues faced in connected vehicles lead us to develop solutions based on cluster-based multi-hop forwarding. Under FastMF, we observe significant improvements in terms of connectivity, overall throughput, and average latency for file delivery.

We design a novel clustering algorithm to take advantage of the MobilityFirst architecture and reduce the impact of node mobility on the functionality of the node as a router. The tree-based structure of clusters reduces the control traffic overhead induced in case of changes to the cluster memberships or shape. To form stable clusters adaptable to the mobility pattern of the nodes, stability of the links between pairs of vehicles are estimated using exchanged location, speed, and other mobility attributes. To extend network connectivity, clustering decisions over stable links are made based on the path length to the nearest network attachment point where path length depends on channel bandwidth and imposed latency.

Furthermore, our proposed clustering algorithm provides all cluster members, including disconnected or poorly connected nodes, with the opportunity to obtain an indirect association to Internet gateways and have reachable NAs in the GNRS. By

forming clusters of vehicles with similar mobility patterns, leveraging cluster to infrastructure links, and maintaining the mapping between node IDs and NAs in a logically centralized server, we provide the nodes without a direct Internet connection with the benefit of an indirect association to an Internet gateway. These detached nodes are otherwise hidden from the rest of the Internet. Furthermore, we extend the MobilityFirst functionality over routers with short-living links and no consistent link to infrastructure.

Internet gateways receive the identity of all members of a cluster through the messages sent by a subset of cluster members directly associated to them. These gateways periodically update GNRS on the identity of associated clusters and their members to enhance tracking vehicular nodes. In name-based architectures where IDs are globally unique and routable, this corresponds to global reachability. In case of interruption in a file transfer due to intermittent connection, this reachability information results in resuming the data transfer without noticeable delay and the need for the mobile node to resend queries. In case of intermittent connections and unreliable links, in-network storage helps recover data and access content with less delay and retransmission [11,12].

Our extended GNRS maintains the mapping between each network entity and a dynamic set of NAs. Consequently, the reachability of an entity through multiple interfaces is not only known to end points (the abstraction available today is through transport or application layer protocols), but also to all network entities on the data path. Thus, a bifurcation router at the branching point between interfaces can leverage cross-layer feedback on link quality to perform load balancing. Further details about our designed extensions to MobilityFirst are provided in section 2.6.1.

Simulation results illustrate the capability of FastMF in reducing in-network load, avoiding congestion, and seamless delivery. Results from ns-3 experiments illustrate the improvements in throughput for downloading large files derived by clustering and multi-hop transfer of data. In addition, experiments with interactive web-browsing scenarios indicate a significant improvement in delay in various mobility scenarios.

Although FastMF is based on the MobilityFirst architecture, it is applicable to any architecture with address and identity separation such as LISP [30]. In addition, our design principles for clustering in a distributed fashion are applicable to any architecture regardless of its networking layer design. In this work, MobilityFirst is integrated with IP architecture and network attachment points correspond to IP addresses.

2.3 Related Work

Traditionally, flooding has been the most simple yet reliable method of disseminating data over a collection of highly dynamic nodes with time-varying connections [3]. However, considering the adverse radio propagation conditions, the channel can become easily congested or the network may experience infinite retransmission loops. To avoid these problems, three main categories of data dissemination mechanisms were introduced for vehicular networks:

Location based: In geo-based routing algorithms such as GPSR and VADD [1], nodes periodically broadcast their geographical location. Forwarding decisions are made based on the geographical distance between neighbors of the node and the final destination (may consider extra information such as density of cars in an area and city maps). The location of the destination node is obtained through a node location service. Given the data source is unknown a priori or changes frequently, considerable control traffic overhead is needed for tracking the source. In addition, it is possible that neighbors geographically closer to the destination lack a network path to the destination [9].

Data mules: In intermittently connected scenarios [3], past information about the vehicles (e.g., mobility traces, destination, social media,...) are used to identify vehicles with high probability of contacting a disconnected or poorly connected node to serve as data mules. These algorithms require an extensive amount of past information to derive contact possibilities. For delay-sensitive scenarios, other alternatives are preferred.

Multi-hop forwarding: In delay-sensitive applications, extending Internet coverage by multi-hop forwarding of packets has been proposed as a solution. In these algorithms, nodes exchange information to gain partial overview about the topology of the local network and apply a routing algorithm to relay packets to/from client nodes from/to an Internet gateway. The routing algorithms can be divided into two groups: i) proactive (table driven) where nodes share the information on local topology with neighbors ii) reactive (on-demand) where upon a new data request, a route query is broadcast.

A classic approach to reduce the complexity of multi-hop forwarding using mobile nodes is to divide them into clusters of nodes with similar movement patterns [6] and thus more stable links. Clustering can improve the rate of successful packet delivery, provide a multi-hop path adaptive to mobility of nodes, and reduce network congestion

[6]. In this work, we focus on this method of data dissemination.

Various algorithms have been introduced for forming clusters in ad hoc networks including MOBIC [33], MobDhop [7], affinity propagation [6], (α, t)-based clustering, and DMAC [7]. These methods differ in the cluster formation criteria or maximum number of hops allowed between members. For instance, affinity propagation [6] maximizes the similarity among clustermates where similarity is measured based on their designed affinity metric which depends on the mobility parameters of vehicles.

In most clustering algorithms, the clusterhead manages the cluster membership of the nodes, routing, and forwarding packets within the cluster. Other cluster members periodically contact the clusterhead to provide it with updated topological information. The clusterhead might be the node with the lowest ID, highest degree, least variation in mobility, or an optimizer of some specified criteria [7]. For instance, in MOBIC [33], the node with lowest relative mobility to its neighbors becomes the clusterhead where relative mobility is measured based on logarithmic change of received power over time. When applied to nodes with group mobility behavior such as vehicles in highways, MOBIC is very likely to form efficient clusters.

Due to the level of dynamicity in VANETs, the rate of clusterhead change in VANETs is significantly higher than MANETs [6, 34]. This can result in frequent flooding of a cluster for the purpose of reassigning the clusterhead, providing a new clusterhead with member states, and reaffirming the existence of a clusterhead.

In [35], a multi-hop forwarding cross-layer (MAC and network) design is introduced for improving in-vehicle Internet access. In this algorithm, using their designed MAC layer protocol called VeMAC, each node assigns the subset of neighbors with high probability of having a network path to the destination as packet forwarders. The resulting algorithm provides a multi-hop path from a node to an Internet gateway. However, this algorithm does not improve reachability to the vehicular node from remote servers. Furthermore, for path discovery, no clustering is imposed; source routing (from a vehicular node) is applied, and the Internet gateway uses the path provided by the client node. In vehicular networks with high mobility, the source routing creates large overhead, the discovered path is usable for a very short time, and data retransmission happens very often [6, 34].

2.4 MobilityFirst Architecture Overview

The core architecture of the Internet is designed for interaction among fixed nodes. The naming and addressing creates many issues for enabling seamless mobility (since changing network attachment point causes changing identity) or multi-homing (since different interfaces have different identities). For vehicular nodes, the point of attachment to the Internet changes rapidly and providing seamless connectivity is challenging.

MobilityFirst [8] is one of the named-object architectures which features the separation of names from addresses and provides a scalable name-based service layer using GUIDs (see Figure 2.1). A GUID is a flat self-certifying identifier of a network entity (e.g., user, interface, device, service, content, context, and cluster of mobile nodes). A GUID is a consistent and long-lasting identifier. Each network entity is assigned a GUID by name certification services based on the cryptographic hash of entity's public key. A GUID is designed such that its owner can be authenticated using public/private keys [8] without using an external entity. This can be beneficial in VANET scenarios where traditional authentication settings (using a third-party server) creates substantial delays due to frequent disruptions.

GNRS as an in-network mobility management framework [8] eliminates the need for treating mobile nodes as special fixed entities and implementing mechanisms such as MobileIP's home and foreign agents [5]. The GNRS provides the basis for tracking nodes. When nodes move, they update GNRS of their most recent NAs. Furthermore, GNRS facilitates multi-homing by hosting the mapping between the GUID of an entity and the NAs connected to all active interfaces of the entity.

There are various design proposals for implementing GNRS as a globally distributed data structure. The original implementation, DMAP [13] consists of a local (per domain) and a global resolution server. The mapping between each GUID and its corresponding NAs are stored in K servers (for reliability purposes). The NAs of these servers are determined by applying hashing functions to the GUID. Through large-scale simulations, [13] showed that if the mappings are replicated at 5 servers chosen uniformly random, the 95% of latency to receive the NA for a queried GUID is at most 86 milliseconds. In GMap [36], the mappings are distributed based on the popularity and geographic location of the corresponding network entities in local, regional and global

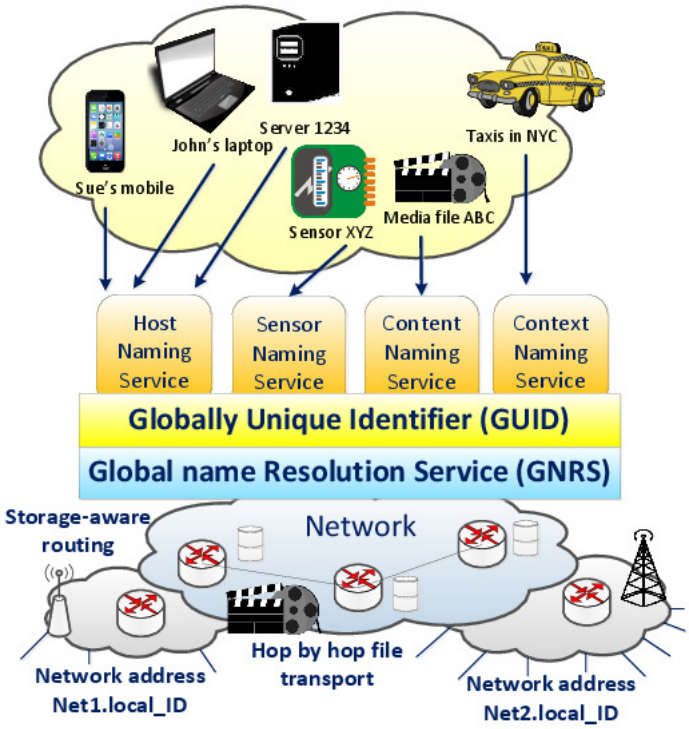

Figure 2.1: Separation of identity from address in the MobilityFirst architecture

servers. GMap, in general, has better latency results than DMap [36].

Support of hop-by-hop transport of large protocol data units (chunks) in the MobilityFirst routing protocol [13] along with having in-network storage for chunks in transit suit the VANET scenarios with frequent connection disruptions and varying link quality [11]. The hop-by-hop chunk transfer capability eliminates the extra delay incurred by end-to-end packet loss recovery or congestion detection. Sending a large chunk over short-living links in VANETs generally increases channel efficiency. In-network storage divides the burden imposed by frequent link disruptions among in-network routers and shifts that from end-points. It enables routers to make store vs. forward decisions per chunk based on the quality of the link to the next hop. In addition, it provides the system with multiple data sources which has been shown to reduce access latency [11]. The link state routing in MobilityFirst is an integration of delay tolerance capabilities into Open Shortest Path First (OSPF). To enable backward compatibility with IP-based systems, packet headers contain both the GUID and the IP address of the receiver.

An extensive discussion of the transport layer protocol in MobilityFirst appears in [8]. In MobilityFirst, the requested content is divided into multiple-packet chunks.

The size of the chunk is based on a compromise between forwarding overhead and available network resources [8]; here we use 500 KBytes chunks.

To Forward a chunk to its next hop, a node sends out a chunk synchronization packet (CSYN) containing the ID and size of the chunk. The next hop replies with a CSYN-ACK message that specifies which packets of the current chunk were received. This way, lost packets are detected and retransmitted following the same procedure. Upon complete reception of a chunk, each node begins forwarding the chunk to the next hop. If a node does not receive the CSYN-ACK within a pre-specified interval, it resends the CSYN message.

In case the client moves before receiving the requested chunk, the network layer assists by delegating the data transfer responsibility to the nodes holding in-transit chunks. If the chunk cannot be completely delivered by the link layer after some attempts, the CSYN will timeout. The node will store the chunk and schedule for its retransmission. The current chunk holder then periodically queries the GNRS for any NA update before attempting to deliver the chunk to its next hop.

2.5 Clustering Algorithm

In this section, we discuss our distributed protocol for forming clusters. Our goal is to design a dynamic clustering algorithm that builds on MobilityFirst to optimize the trade-off between the stability of a cluster and the control traffic overhead needed for both cluster formation and maintenance. The proposed clustering algorithm consists of a 3-stage VANET formation protocol. In the first stage, nodes exchange Link Probe (LP) messages to identify neighbors. In the second stage, nodes perform link stability assessment in order to exclude short-lived unstable links. In the final stage, nodes exchange Cluster Status (CS) messages to make clustering decisions and find the shortest path to an Internet gateway.

2.5.1 Neighbor Discovery

To describe the neighbor discovery and cluster formation algorithms, we classify nodes based on the following categories:

- Single: A node that does not belong to any cluster.

Message	Content	Size (Bytes)	Description
Link Probe (LP)	Node GUID, cluster GUID, current velocity, current location, correlation in recent w_{est} speed samples	14	All nodes (including gateways) periodically broadcast LP which is used to discover neighbors, and determine stable adjacent links.
Link Probe ACK	Node GUID, cluster GUID	8	Broadcasted by nodes who receive LP and have a stable link with LP owner and is used to evaluate link ETTs
Cluster Status (CS)	GUID, $\text{Path}_{\text{AP}}(A,t)$, Cluster GUID GUID of the cluster tree root GUIDs of descendants of A in the cluster GUID of A's parent node in the cluster	14+2N	Each node periodically broadcasts (as a multiple of the LP period) CS messages to provide neighbors with its updated clustering-related characteristics.

Table 2.1: Network layer messages for arbitrary node A which has N subtree members

- Disconnected Cluster Member (DCM): A node that belongs to a cluster but lacks a direct (one-hop) connection to an AP.
- Internet-Connected Member (ICM): A cluster member that has direct connection to an AP.

Each node periodically broadcasts LP messages to discover peers in its neighborhood, determine its stable adjacent links, and estimate the transmission delay over such links using ACKs received in response to LP messages (see Table 2.1). LP messages comprise the node GUID, the cluster GUID, and the mobility parameters (i.e., current velocity, current location, and estimated correlation in recent speed samples). The cluster GUID is a unique identifier assigned to a cluster and shared by all cluster members. Single nodes, although not a member of any cluster, announce a fresh self-made GUID as a cluster GUID in their transmitted messages. When a node becomes responsible for naming a cluster, this self-made GUID will be used (see section 2.6). Similarly, when a node becomes single by leaving a cluster, it creates a new cluster GUID. Instead of exchanging LP messages, nodes can obtain this information exploiting the lower layer beaconing functionality (e.g., 802.11p beacons provide speed, direction, and ID of the vehicle with 10 Hz frequency) [1]. APs similarly broadcast LPs that contain GUID and location of the AP. These messages can be incorporated into the AP's beacon messages.

Stability of a cluster is ensured by evaluating the stability of the links based on the mobility parameters of the nodes involved. For vehicular nodes, reasonably accurate measurements of location and speed can provide a basis for stability metrics. The accuracy of this information depends on LP broadcast period which is 0.3 sec (to avoid generating large overhead traffic). In this work, we determine the stability of a link between two nodes based on the difference between their velocities [6]. This follows

the stability analysis in highly dynamic scenarios where stable links with long lifetime occur more frequently among nodes with similar velocities.

2.5.2 Cluster Formation

The quality of a cluster is evaluated primarily based on the length of the path to the nearest AP, where the path length metric is expected transmission time (ETT) [14]. ETT of a link corresponds to the estimated time required to send a packet through the link successfully. For vehicular scenarios, ETT is ideally suited since both link packet loss (quality of uplink and downlink) and bandwidth (the effect of interference from other concurrent transmissions) impact the ETT. It dynamically adapts the cluster size with the channel conditions, and if the channel is congested, the cluster does not grow further. Nodes deduce the ETT of an adjacent link from the timing between their LP messages and the corresponding received ACKs from neighbors.

Each node periodically broadcasts (as a multiple of the LP message period) CS messages to provide neighbors with its updated clustering-related characteristics (see Table 2.1). Based on this knowledge, nodes form tree-based clusters in which the root of the tree is an ICM and each member of the tree is on its estimated shortest path to the AP. The cluster formation step is in fact a path discovery mechanism where each cluster member identifies its adjacent subtree and its next hop node towards the Internet gateway. By putting this restriction on the information that each node has, we thrive to minimize the amount of clustering overhead and make the protocol robust with respect to mobility. Figure 2.2 illustrates an example of a cluster containing B, D, E, F and the cluster status messages exchanged between B and D.

As specified in Table 2.1, the contents of a CS message transmitted at time t for an arbitrary node A, advertises the path length $\text{Path}_{\text{AP}}(A, t)$ to an AP as well as GUIDs of nodes in the adjacent subtree that ultimately enable construction of the full tree at the AP. To avoid issues related to looping, and to distinguish fresh and different paths to an AP from old paths advertised by subtree members of A, GUIDs of the parent and root of the tree are provided in the CS message. We also include a sequence number for the last update message generated by the root of the cluster tree that was incorporated in the $\text{Path}_{\text{AP}}(A, t)$.

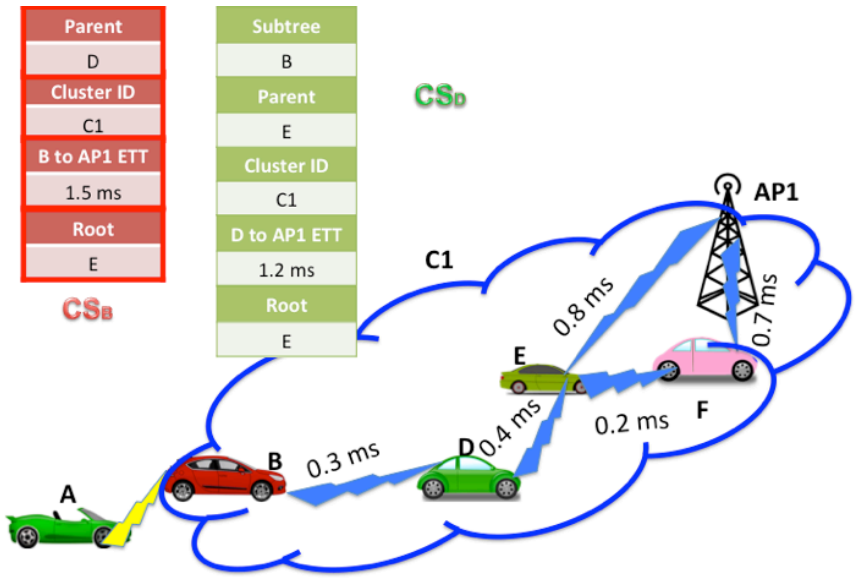

Figure 2.2: An example of a cluster and exchanged CS messages between B and D

Upon reception of CS messages, nodes update their estimate of the path from neighboring nodes to an access point, form clusters, or update their membership in a cluster accordingly.

The cluster formation process is as follows: Assume that nodes A and B identify the link (A, B), and have shared clustering information. In addition, assume that $\text{Path}_{\text{AP}}(A, t) > \text{Path}_{\text{AP}}(B, t)$. In this case, A will join B's cluster if

$$\text{Path}_{\text{AP}}(A, t) - \text{ETT}_{\text{thresh}} > \text{Path}_{\text{AP}}(B, t) + \text{ETT}_{(A,B)},$$

where $\text{ETT}_{\text{thresh}}$ is the minimum improvement in path length to an AP for which A will switch clusters. The hysteresis induced by $\text{ETT}_{\text{thresh}}$ is intended to prevent nodes from oscillating between clusters and generating excessive GNRS updates. When A decides to cluster with B, it uses the cluster GUID of B as its own cluster GUID. As such, A makes its clustering decision asynchronously and autonomously without further exchange of specific coordination messages with B. When A produces its next LP or CS, other neighboring nodes would detect the cluster change.

An arbitrary node A in cluster $\mathcal{C}$ detects loss of its network path to its clustermates by virtue of not receiving LP messages from the entire set of cluster neighbors after some prespecified interval. Hence, it resets its cluster GUID entry in its broadcast

messages to a fresh GUID to mark its departure from the old cluster. This indicates that nodes make the clustering decisions independent from each other.

Each node maintains a Neighbor table that summarizes the information received in LP and CS messages from adjacent nodes (including APs) containing their GUID, cluster GUID, link ETT, and path ETT, parent and root GUID and their subtree on their cluster tree. Cluster members can leverage the Neighbor table in the disconnected mode (when the cluster has no ICM). In this mode, access to the GNRS is denied, but a member can still communicate with others based on information in its Neighbor table. In addition, nodes in disconnected mode can advertise for content, context, and other supported services available in the cluster. The cluster topology is stored in a distributed way. Over time, new nodes join or old nodes leave the cluster and information is propagated to the root node.

Exchanges of CS messages with neighboring clustermates causes the aggregation and dissemination of the topology information (i.e., gaining knowledge about the subtrees of children) on the cluster trees. As such, upon receiving CS messages from root nodes, APs are provided with a list of all cluster members, which can be used for GNRS updating (see Section 2.6). Each cluster can be attached to zero, one or multiple Internet gateways. One of the novelties of FastMF is that an end host can send data to cluster members without knowledge of their cluster structure. After the cluster is established, each member forwards a received chunk if either the destination node is in the node's subtree or the destination node is the NA of the cluster.

In addition to enhancing connectivity, we take advantage of the broadcast nature of vehicular environments and the high possibility of connection among members by allowing these members to cache data addressed to other members for a limited time and forward to the intended clustermate when a path between them is available. A summary of the cluster formation algorithm is presented in Algorithm 1.

2.5.3 Cluster Maintenance

Nodes periodically scan their Neighbor tables, and discard expired neighbors from which no message was received for $\mathrm{TTL}_{\mathrm{neighb}}$ milliseconds. If the discarded neighbor was a parent node, clustering will be repeated. If a node loses its parent or all links to its cluster, it tries to repeat the clustering decision process where it searches through the

Algorithm 1 Cluster formation

procedure CLUSTER(node A)
  Input: Neighbor set N_A, Subtree set T_A, Access point set AP
  $//A \nsubseteq$ AP
  for all $B \subseteq N_A$ **do**
    if $\mathrm{ETT}_{A,B} \to \infty$ **then**
      continue to the next neighbor
    end if
    if $B \subseteq T_A$ **then**
      A is part of B's path to AP
      continue to the next neighbor
    end if
    if $((B, \mathrm{Parent}_A \nsubseteq \mathrm{AP})$ & $(B \neq \mathrm{Parent}_A)$ & $(\mathrm{Root}_A \nsubseteq \emptyset))$ **then**
      if $((\mathrm{Root}_A = \mathrm{Root}_B)$ & $(\mathrm{Seq}_{\mathrm{Root}_A} < \mathrm{Seq}_{\mathrm{Root}_B}))$ **then**
        B is showing stale path length
        continue to the next neighbor
      end if
    end if
    if $(B \subseteq \mathrm{AP})$ **then**
      if $(\mathrm{Path}_{\mathrm{AP}}(B) + \mathrm{ETT}_{A,B} < \mathrm{Path}_{\mathrm{AP}}(A))$ **then**
        $\mathrm{Parent}_A \leftarrow B$
        $\mathrm{Path}_{\mathrm{AP}}(A) \leftarrow \mathrm{Path}_{\mathrm{AP}}(B) + \mathrm{ETT}_{A,B}$
        return ;
      end if
    else if$(\mathrm{Path}_{\mathrm{AP}}(B) + \mathrm{ETT}_{A,B} < \mathrm{Path}_{\mathrm{AP}}(A) - \mathrm{ETT}_{\mathrm{thresh}})$
      if $\mathrm{cluster}_A = \mathrm{cluster}_B$ **then**
        $\mathrm{CmateBetterPath}_A \leftarrow B$
      else
        $\mathrm{NonCmateBetterPath}_A \leftarrow B$
      end if
    end if
  end for
  if $\mathrm{CmateBetterPath}_A \neq \emptyset$ **then**
    Assume $C \in \mathrm{CmateBetterPath}_A$ minimizes $\mathrm{Path}_{\mathrm{AP}}(.) + \mathrm{ETT}_{A,.}$
    $\mathrm{Parent}_A \leftarrow C$
    $\mathrm{Path}_{\mathrm{AP}}(A) \leftarrow \mathrm{Path}_{\mathrm{AP}}(C) + \mathrm{ETT}_{A,C}$
  else if $\mathrm{NonCmateBetterPath}_A \neq \emptyset$
    Assume $C \in \mathrm{NonCmateBetterPath}_A$ minimizes $\mathrm{Path}_{\mathrm{AP}}(.) + \mathrm{ETT}_{A,.}$
    $\mathrm{Parent}_A \leftarrow C$
    $\mathrm{Path}_{\mathrm{AP}}(A) \leftarrow \mathrm{Path}_{\mathrm{AP}}(C) + \mathrm{ETT}_{A,C}$
    $\mathrm{cluster}_A \leftarrow \mathrm{cluster}_C$
  end if
end procedure

Neighbor table for a node with a better path to an AP. The preference is given to current clustermates to minimize the number of cluster changes.

If a node exits the coverage area of other nodes, it will remain undecided for a specific interval. If it does not see any other path, it creates a new cluster. However, it still periodically reattempts to join neighboring clusters.

2.6 Integration of Highly Dynamic Nodes into MobilityFirst

A main step for improving in-vehicle Internet coverage is to make reachability information for disconnected or poorly connected nodes (their most recent indirectly associated network attachment points) globally available. CS messages broadcasted by ICMs provide the APs with the status of all cluster members based on ICM's most recent knowledge about the cluster (see Section 2.5.2). In other words, CS messages provide the seamless yet efficient status updating system required to combine ad hoc clusters and the in-network mobility management system (GNRS) without generating excessive control traffic overhead. APs collect CS messages from all associated nodes and detect changes in the status of members of attached clusters. When a change occurs, i.e., node joining or leaving a cluster, the AP submits an update message to the GNRS. Figure 2.3 depicts how our proposed FastMF algorithm improves the accuracy of mapped NAs specially for DCM nodes. Hence, FastMF enhances the reachability service and seamless connectivity in highly dynamic scenarios. Integration of structured VANETs into original MobilityFirst architecture minimizes the amount of time that DCM nodes are hidden from GNRS and the core Internet. .

GNRS maintains the mappings between cluster members and their corresponding cluster GUID. In addition, it keeps the mappings between cluster GUIDs and all of the cluster attachment points. When GNRS receives a query for a cluster member, it performs indirect mapping and returns all the NAs that the cluster is attached to.

Figure 2.4 illustrates an example of FastMF implementation in a highway with intermittent connections. In this example, clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ are already formed. Nodes V_1, V_4, and V_6 have one-hop connections to the AP, while V_2 and V_5 have multi-hop connections to the AP, and V_3 (a previous member of $\mathcal{C}_1$) is in the process of discovery and clustering.

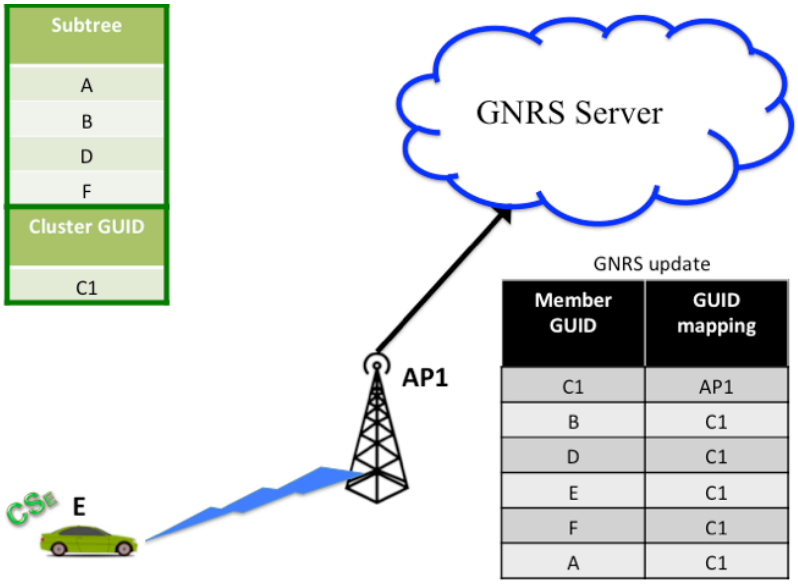

Figure 2.3: GNRS maintains the mappings for all members of $\mathcal{C}_1$.

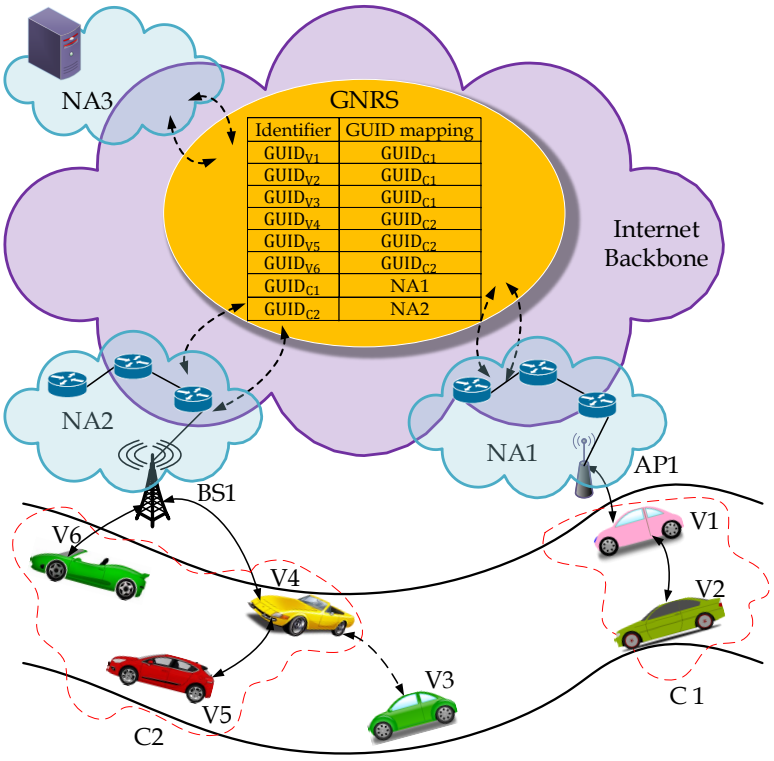

Figure 2.4: FastMF architecture

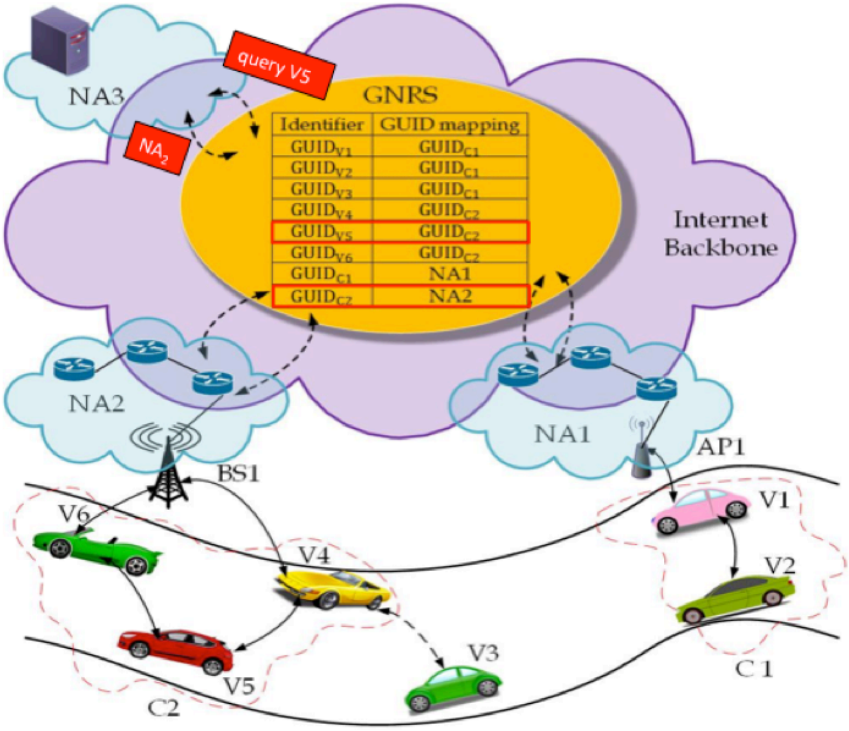

Figure 2.5: Remote server NA_3 queries the NA mapped to GUID V_5

To forward a packet to V_5, the remote server NA_3 sends a query to the GNRS to obtain the NA mapped to V_5 (see Figure 2.5). In the GNRS, the V_5 GUID is mapped to its corresponding cluster GUID $\mathcal{C}_2$. GNRS then proceeds to resolve the mapping for $\mathcal{C}_2$, and returns NA_2 to the remote server.

The remote server addresses the chunk using GUID of V_5 and NA_2. Within $\mathcal{C}_2$, any node whose subtree member list contained the GUID of V_5 forwards the chunk. Figure 2.6 illustrates the packet forwarding in backbone Internet and within the cluster.

Chunks addressed to V_3 are stored in NA_1 as long as V_3 remains in the disconnected area. NA_1 periodically queries GNRS to find the updated NA for V_3. NA_2 updates the GNRS as soon as V_3 reconnects through NA_2 (directly or through other members of $\mathcal{C}_2$). Upon receiving the updated NA, NA_1 forwards the stored chunks to the updated address. This eliminates the need for stored chunks to be retransmitted all the way from NA_3 to the NA. If one of the vehicular nodes such as V_2 needs to send a query to a remote server, the query will be forwarded over the cluster tree until it reaches NA_1.

To minimize the rate of updates sent to GNRS and changes within the cluster,

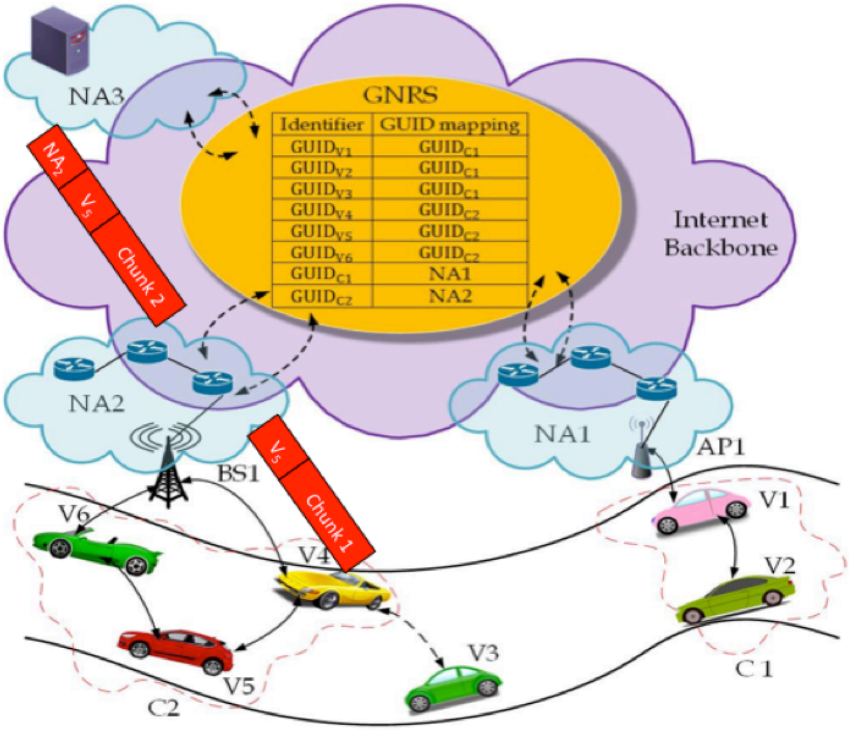

Figure 2.6: Forwarding chunks from remote server NA_3 to V_5

each cluster is named by a separate cluster GUID, rather than cluster identification alternatives such as the GUID of a member or the NA of an ICM clustermate.

2.6.1 Enhancements to MobilityFirst Forwarding Mechanism

Section 2.4 describes the hop-by-hop forwarding algorithm in MobilityFirst. In case of VANETs, nodes decide on forwarding a received chunk by analyzing its source and destination GUID. If the node is a member of the path between source and destination (e.g., the source belongs to the subtree of the node and the destination is an Internet gateway or vice versa), it will forward the chunk.

Various issues arise when applying original MobilityFirst forwarding mechanism to vehicular ad hoc scenarios. In this work, we designed and implemented mechanisms to extend the MobilityFirst routing functionality to nodes with short-living links and intermittent connection to the infrastructure and allow MobilityFirst accommodate nodes with high dynamicity or located in areas with partial coverage of infrastructure-based wireless network.

Link stability evaluation adapted to dynamicity: Due to high dynamicity in vehicular networks, we design new algorithms in FastMF to estimate link stability using mobility parameters of the nodes. More information can be found in Appendix A.

Indirect association for DCM nodes: In the original MobilityFirst, nodes detached from network infrastructure are hidden from the GNRS and the backbone Internet. As a result, for highly dynamic scenarios, the sessions are interrupted inevitably since maintaining a session requires client/server to have direct Internet access. The routers that are in charge of providing most of the MobilityFirst services are assumed to be connected with each other through long-living links and have a consistent attachment to the network infrastructure. In FastMF, we allow nodes with short-living links and intermittent network connection to function as MobilityFirst routers. As a result, the GNRS is updated with the most recent NAs for all nodes including DCMs. Furthermore, DCMs are not hidden from GNRS anymore and the stream of chunks towards them continue over multiple hops.

Extension of named-object based services: In FastMF, we extend the named-object based services to include nodes with high dynamicity or those in poor coverage areas. By integration of peer-to-peer links into MobilityFirst, FastMF enhances the capability of MobilityFirst in performing multihoming and maintaining seamless connectivity for nodes faster than Mobilityfirst's original design. To do so, we group network entities with similar mobility attributes, assign unique IDs to these groups, and provide GNRS with real-time update on NAs that such groups are reachable from.

Multiple CSYN-ACK: In the original MobilityFirst architecture, data chunks are exchanged among pairs of nodes and the chunk forwarder receives CSYN-ACK only from one node. In FastMF, multiple nodes may be eligible next hops and send CSYN-ACK for the same chunk (to avoid concurrent transmissions, each sends CSYN-ACK after a random delay). In the current implementation, all ACKs received during a specified interval after sending the chunk are aggregated and replied.

Extension of store-and-forward service: FastMF faces more resource limitations and higher randomness in network paths compared to MobilityFirst. As a result, FastMF leverages its own multi-hop forwarding capabilities to extend the store-and-forward mechanism in MobilityFirst. Based on the broadcast nature of transmissions in ad hoc networks, nodes other than the intended next hops may also receive a chunk. These

nodes do not reply with a CSYN-ACK to avoid multiple retransmissions of data and channel congestion. However, they cache the received chunk for a limited period of time and forward it in case of receiving requests for the stored chunk.

Reliability mechanism in VANETs: Due to the unreliability of links, inaccurate routing information, and intermittent connectivity in VANETs, FastMF has an additional client-to-gateway reliability mechanism to ensure the delivery of requested chunk to the client. As such, the client, upon complete reception of the requested chunk, sends an ACK which is forwarded to the client's current NA. The client's NA is responsible in part for ensuring the delivery of each requested chunk. If the designated NA does not receive the client-to-gateway ACK after a pre-determined time and if the client is still a member of one of NA's associated clusters, the NA will retransmit the chunk. If the client has left the associated clusters, the NA stores the chunk and queries the GNRS for the updated NA of the client. In addition to this strategy, based on the type of transferred chunks, we can implement a client-to-server mechanism using ACKs for each chunk or NACKs for lost chunks [8].

Mobile server: FastMF extends the data delivery services currently available in MobilityFirst to include highly dynamic data servers that are sending data over one-hop or multiple hops (e.g., Facebook Live, Periscope, etc.,). In these scenarios, the NA adjacent to the server's cluster stores a copy of transferred chunks, acts as a proxy for server, and is responsible for chunk delivery and end-to-end reliability with clients. This can mitigate the issues imposed by intermittent connections of the server.

Implicit ACK: If a node receives a data request for chunk i, it assumes that all previous chunks were completely delivered.

Flow control: Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) links in a cluster usually share the same channel bandwidth. FastMF extends the flow control mechanism in MobilityFirst to adapt the data transfer decisions to Internet access patterns of clusters. When a cluster has Internet access, the priority for channel use is given to V2I transmissions. In the disconnected area, nodes use the V2V links to transfer the stored data to the final destination.

Parameter	Amount
Highway radius	2000 m
Number of APs	12
Number of mobile nodes	200
Time step	0.001 sec
simulation time	2000 sec
AP coverage radius	125 m
Speed (one direction)	$\mathcal{N}\left(\mu(m/s), \sigma_v^2 = 5\right)$
T_{target}	$\exp(\lambda)$, $\lambda = 1/20\,\text{sec}^{-1}$
Beaconing period	200*Time step
Link probe period	0.3 *sec*
Standard	IEEE 802.11a

Table 2.2: Simulation parameters (test 1)

2.7 Performance Evaluation

In this section, we present the results from a proof-of-concept ns-3 simulation of FastMF. These results evaluate the quality of our designed clustering algorithm and enhancement in accessibility and reachability obtained by FastMF.

2.7.1 Simulation Setup

Our simulation contains vehicles with IEEE 802.11a wireless network interface operating at 6Mbit/s moving along a simulated path. The coverage radius for each node is 125 meters. The simulated path is a multi-lane circular highway with 2 km radius. Due to our link stability assessment algorithm, vehicles moving in opposite directions are ignored. The path is partially covered with uniformly spaced 802.11a access points. This provides intermittent connections for the vehicles as they move along the highway. Table 2.2 summarizes the basic simulation parameters.

To model the mobility of vehicles in sections 2.7.2-2.7.4, we used the Gauss-Markov mobility model [37]. In this model, each vehicle has a randomly selected target speed to demonstrate moving behaviors in different environments such as traffic jam, intersection, and highways. The target speed changes after an exponentially distributed random time T_{target}. The vehicle speed changes over time to converge to its target speed. To mimic the mutual interactions between vehicles, such as overtaking, traffic jam, and preferred paths, the distribution of target speed for neighboring nodes can

Test	Differences with test 1
2	Speed $\propto \mathcal{N}(\mu = 20m/s, \sigma = 5)$
3	Number of mobile nodes=100
4	Speed $\propto \mathcal{N}(\mu = 15m/s, \sigma = 5)$
5	Highway radius=1000 m

Table 2.3: Parameters modified in each test compared to test 1

be correlated. Table 2.3 points out those parameters that were modified in subsequent tests in sections 2.7.2-2.7.4. Additional analysis of the Gauss-Markov mobility model appears in Appendix A. For performance evaluation at scale in sections 2.7.5-2.7.7, we deploy mobility data derived from SUMO [38].

2.7.2 Overhead

The improvement in Internet connection using FastMF is appealing as long as the algorithm does not require exchange of a large amount of control messages. There are two sets of control traffic involved in this algorithm. The first set consists of LP, CS, and ACKs exchanged between nodes to form and maintain clusters (see Section 2.5.2). Figure 2.7(a) compares the clustering overhead per node for different tests. Considering that LP and CS messages are sent periodically, the difference in overhead between different tests is due to the different number of ACK messages and CS message sizes exchanged over different tests.

The second set of control traffic messages consists of periodic GNRS updates sent by APs (see section 2.6). To provide accurate reachability information in the GNRS, APs send the NAs for all associated (directly or indirectly) nodes. In vehicular settings, these updates are sent more frequently than general settings. This raises concerns related to scalability and amount of overhead required for maintaining acceptable precision in the GNRS. The total traffic generated by AP for updating each node in the GNRS is shown in Figure 2.7(b) for different tests. Figure 2.7(b) demonstrates that such precise reachability information (see Figure 2.13) requires a negligible control overhead. In addition, by comparing the number of updates required for each test, we can observe that clustering enables aggregation and summarizing of update information about vehicles and reduces update traffic size (compare test 1 and test 3).

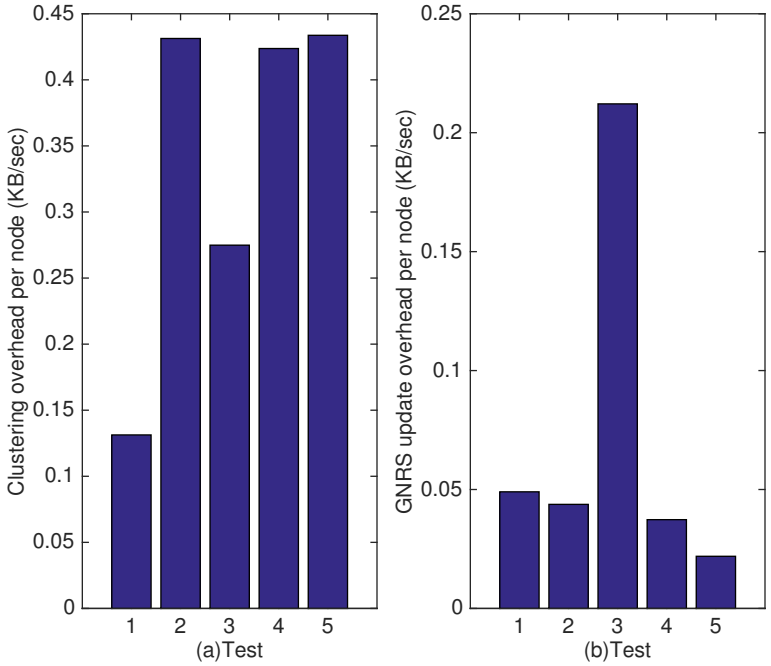

Figure 2.7: Control traffic overhead for various tests (a) overhead due to clustering (b) overhead due to GNRS update

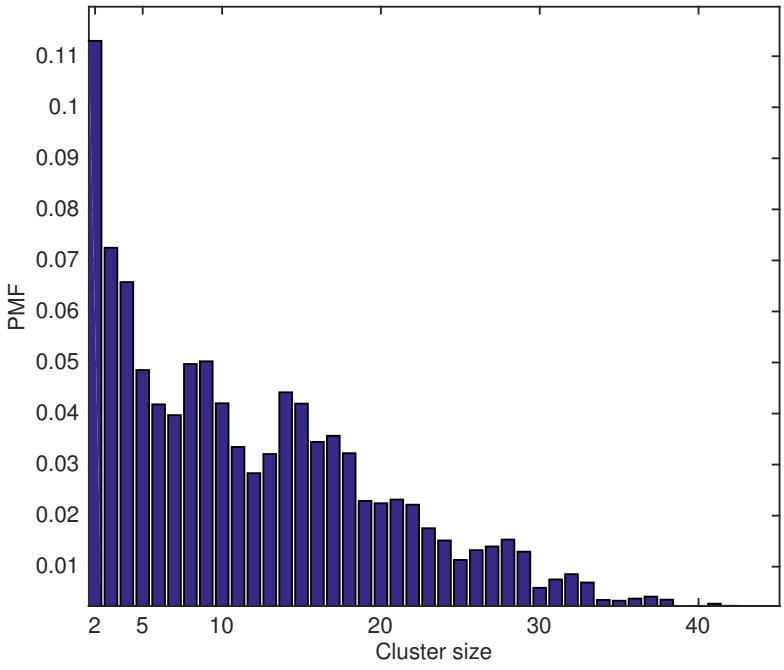

Figure 2.8: Distribution of the cluster size for test 1

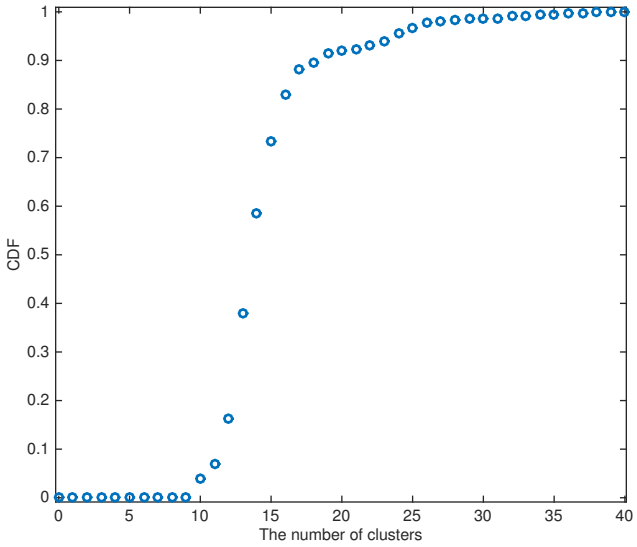

Figure 2.9: Cumulative distribution of the number of clusters for test 1

Cluster size has a direct impact on the size of clustering traffic overhead. Figure 2.8 illustrates the distribution of the cluster size per time unit. Approximately 97% of the time, clusters have sizes of less than 20. This indicates that the improvement in Internet connectivity can be achieved without a need to build large clusters. When the density was 100 nodes, around 45% of the clusters had just 2 nodes due to sparsity.

2.7.3 Cluster Stability

In highly dynamic scenarios, we expect the structure of the cluster to be robust against frequent topology changes. In this work, three metrics are used to evaluate the stability of the generated clusters.

The first metric is the number of clusters in each time step. If the clustering algorithm generates stable enough clusters, this number is low and remains steady during the simulation. Figure 2.9 illustrates the cumulative distribution function of the number of clusters in each time step. Based on this plot, approximately 99% of the time, we had at most 30 clusters for a set of 200 nodes. One of the parameters affecting the number of clusters is the ratio of the covered area to the total highway length. As more of the highway is covered with AP coverage, fewer clusters are generated.

The second metric is the average residency time of a node in a cluster illustrated

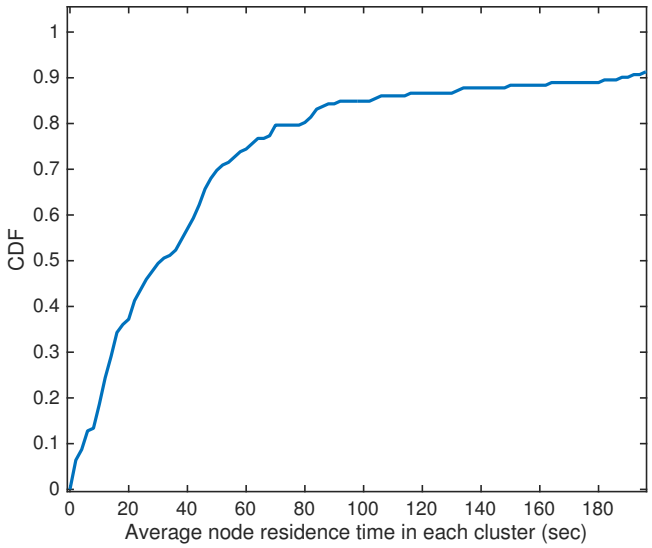

Figure 2.10: CDF for the average residence time in seconds of a node in a cluster (test 1)

in Figure 2.10 which indicates how much the decision of this node to evaluate a link as stable and join the cluster was suitable. The median residence time was roughly 32 seconds. The reasons for having long residence time are the threshold imposed for cluster switching (1) and having approximately fixed cluster structure in disconnected area.

The third metric is the average cluster lifetime, which is the length of an interval during which a cluster has at least two members. Figure 2.11 shows the cumulative distribution for cluster lifetime. The median cluster lifetime is 45 seconds. This shows that the generated clusters are robust enough to have long lifetimes.

2.7.4 Delay within a Cluster

One of the important quality of service metrics is the delay imposed by the network. Two important parameters affecting the end-to-end delay in this setting are GNRS query-response round trip time and the delay over ad hoc links. The former is approximately between 0.030 and 0.173 milliseconds [13]. Figure 2.12 depicts the CDF for the path length (in milliseconds) of any ad hoc node (either one-hop or multi-hop) to the nearest AP. This figure shows that the multi-hop path does not impose a huge latency on the packet and 94% of delays are less than 10 ms.

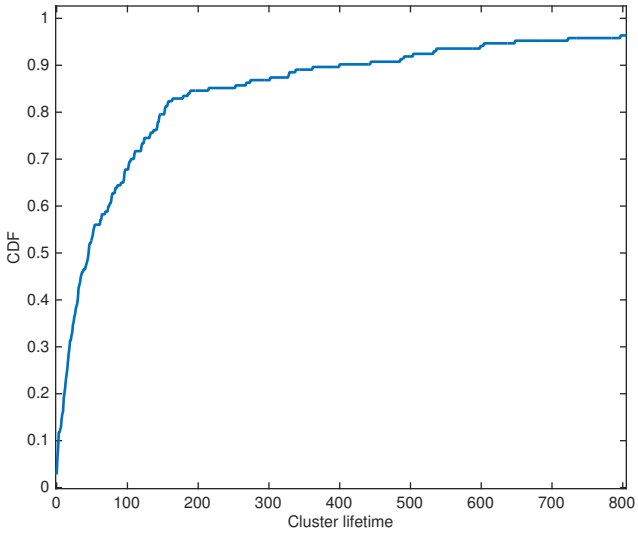

Figure 2.11: CDF of cluster lifetime in seconds

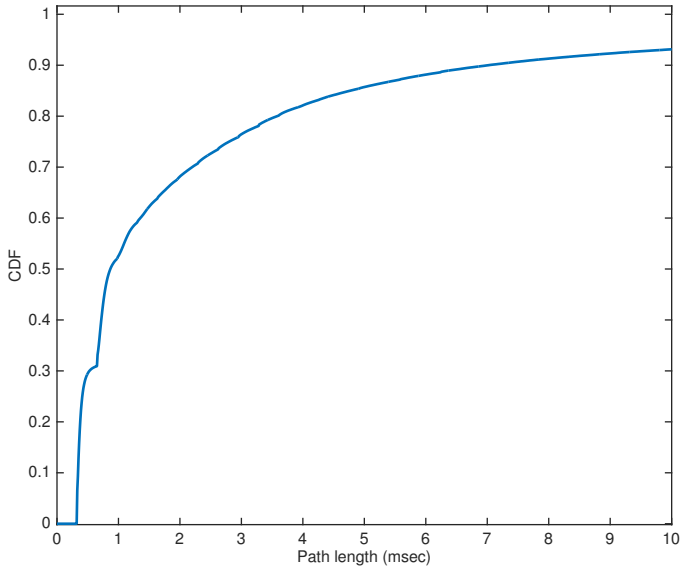

Figure 2.12: CDF for the path length (either one-hop or multi-hop) to NA

2.7.5 Connectivity Enhancement Results

Figure 2.13(a) illustrates the cumulative distribution function (CDF) of the percentage of time that a node has Internet connection (Accessibility). By comparing scenarios with and without multi-hop forwarding capability, the advantages of clustering and multi-hop forwarding in improving the uplink quality and extending the session lifetimes over multiple hops can be deduced. Clustering improves the opportunities for a highly dynamic node to connect to the Internet by a factor of 2 or 3 over its obtained connectivity with one-hop links. As illustrated in this figure, the average connectivity increased from 25% in the one-hop scenario to 56% in the multi-hop scenario with 100 nodes and 72% in the multi-hop scenario with 200 nodes. In addition, 60% of nodes were connected to the Internet at most 59% of simulation time in the scenario with 100 nodes and at most 74% of simulation time in the scenario with 200 nodes.

Figure 2.13(b) depicts the CDF of percentage of the time that the GNRS contains a correct mapping for a vehicular node (reachability). Similar to Internet connection percentage, the GNRS accuracy improves significantly by clustering.

Comparing Figures 2.13(a) and (b), we observe that the graphs for GNRS accuracy precisely follow their Internet connectivity counterparts. In other words, approximately as soon as a node obtains access to AP over one or multiple hops, the network attachment point is registered in GNRS and the node becomes reachable. In other words, nodes obtain seamless connectivity and reachability regardless of their connection quality or mobility dynamics. The minor difference between these graphs is because the APs aggregate the observed updates and periodically send them to the GNRS (GNRS update period is set to 1 second in this work).

2.7.6 Large File Download

In this section, we evaluate the performance of the designed system for a file retrieval scenario. At the start of the experiment, the client (a randomly chosen car) issues a fetch request for a content GUID_f, $\text{get}(\text{GUID}_f)$, without specifying the location of the server. This request reaches an access point connected to the client that queries the GNRS, receives the NA of the server with the content, and forwards the request message. In this experiment, we assume that the content owner is a remote server

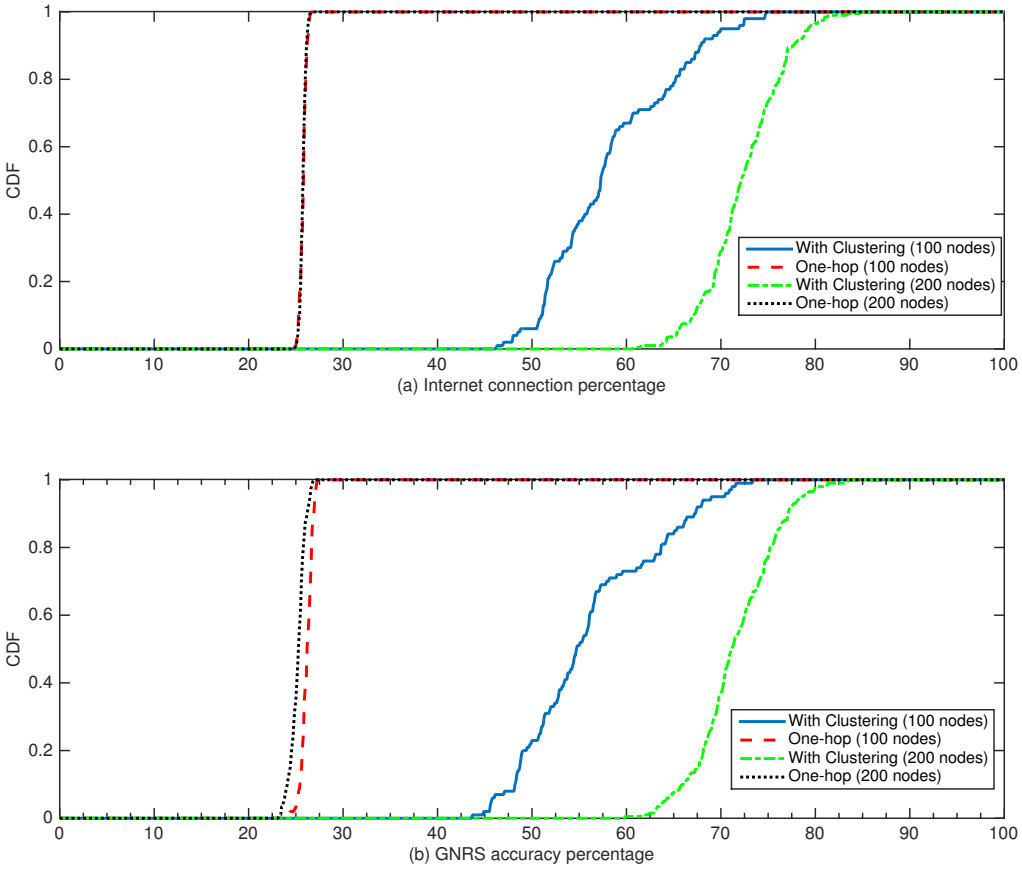

Figure 2.13: (a) accessibility: CDF for Internet connectivity percentage comparing one-hop scenario versus scenario with clustering (for 100 node and 200 node experiments) (b) reachability: CDF for GNRS accuracy percentage comparing one-hop scenario versus scenario with clustering (for 100 node and 200 node experiments)

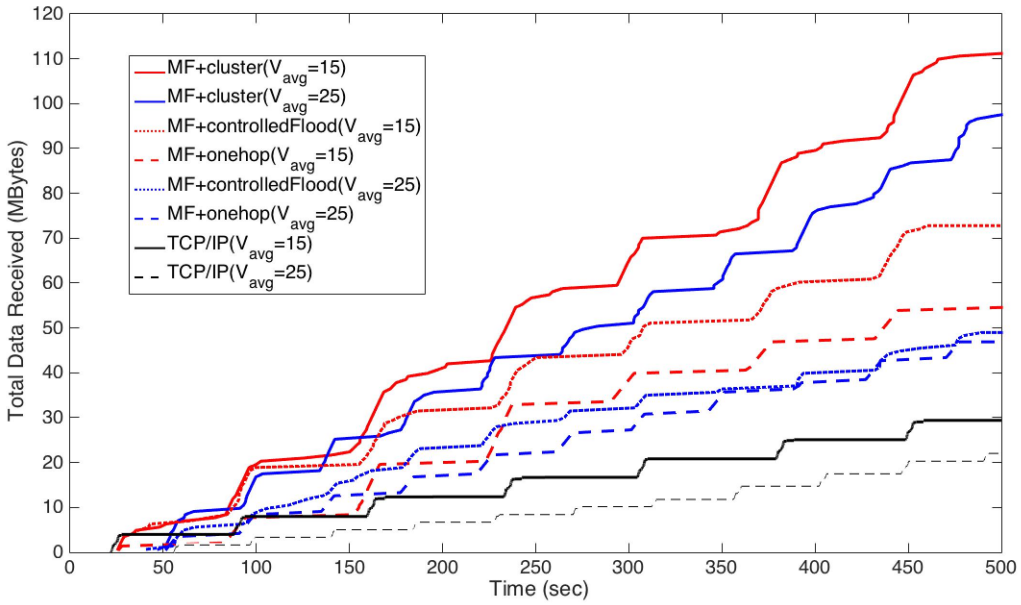

Figure 2.14: Aggregate throughput comparison of MF combined with clustering, original MF, MF with controlled flooding, and TCP/IP

which is connected to the Internet through a back-end wired network. Upon receiving the request, the server with the content initiates a flow of chunks to the client (0.5 MByte chunks every 100 milliseconds). The remote server queries the GNRS of the current NA of the client periodically (approximately every 3 seconds). The round trip time for GNRS query/update is between 30-170 milliseconds [13]. The nodes benefit from having the same ID moving among different APs which results in increasing the useful time spent in the coverage area of an AP.

Figure 2.14 shows the average accumulated received data over time by clients with intermittent connection. The total received data for two different average speeds are presented to evaluate the effect of mobility. In addition to our proposed protocol, we evaluate the basic MobilityFirst without clustering. Furthermore, we compare our protocol with a more reactive multi-hop data dissemination approach (i.e., with no neighbor information exchange), namely controlled flooding, integrated int MobilityFirst. In the controlled flooding scenario, the client device periodically broadcasts its query, which is rebroadcasted over all adjacent stable links. To avoid a broadcast storm, nodes do not rebroadcast a duplicate of a query received less than 1 second after the preceding query. When the query reaches the AP, the AP updates the GNRS about the NA of client node and forwards the query. When the AP receives the requested chunks, the same

rebroadcasting algorithm is used towards client. To compare the proposed protocol with current implementation of Internet access for vehicular nodes, we also examined the throughput of TCP/IP based Internet access over one-hop. In this scenario, the application layer after experiencing disconnection resumes sending the rest of the file instead of restarting.

Our current results are based on 1047 meters distance between APs, and 125 meters coverage radius for each AP. The flat horizontal parts of the curve in Figure 2.14 correspond to no connectivity areas. Due to the adaptation of flow control mechanism to connectivity pattern implemented in FastMF (see section 2.6.1), in the FastMF scenario, the throughput increases in disconnected areas.

In Figure 2.14, comparing the throughput in MobilityFirst-enabled scenarios with and without clustering, the significant improvement resulted from using clustering is illustrated. The clustering enhances the throughput due to extension of coverage area by cluster members, enhanced reachability, and multi-hop forwarding of cached data in disconnected area. This improvement is achieved regardless of the average speed which indicates the robustness of our clustering algorithm in vehicular scenarios. Furthermore, comparing the results of scenario with clustering and scenario with controlled flooding, we can see the effect of cluster formation and providing a data forwarding path adaptive to node dynamics on extending the session lifetime and improving throughput. These comparisons show that FastMF extends the routing functionality of MobilityFirst to vehicular nodes with intermittent connections to the Internet and short-living links. As expected, the disadvantage of TCP/IP systems results in a very low throughput compared with other MobilityFirst-enabled experiments.

2.7.7 Interactive Web-browsing

In this scenario, at each time 10% of the clients have an interactive session with a remote server. The clients initiate a request for a content with uniformly distributed size between 0.5 and 5 MBytes. After the requested data is completely received, the next request is issued after an exponentially distributed time (with mean 10 sec). These requests occur regardless of whether the node has a connection at that time and where the node is located.

Figure 2.15 depicts the CDF for data request completion times. Figure 2.15(a)

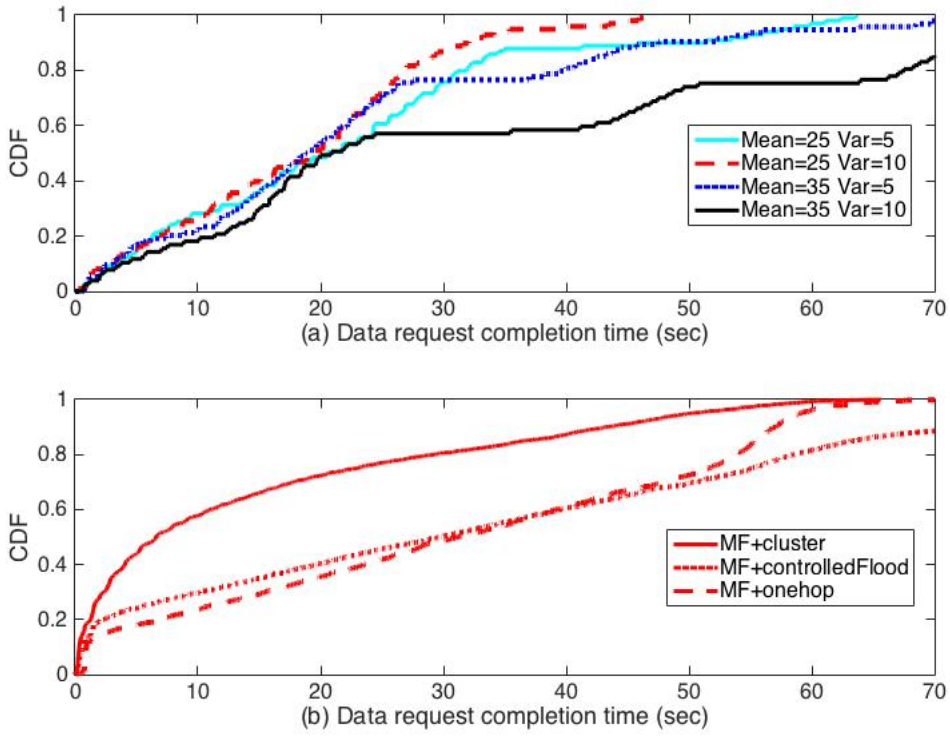

Figure 2.15: CDF of data request completion times in the interactive web-browsing scenario (a) effect of mobility parameters (in m/sec) on the performance of our clustering protocol (b) comparison of MF combined with clustering, original MF, and MF with controlled flooding with average speed of 15 (m/sec)

evaluates the effect of mobility parameters such as average speed (in m/sec) and speed variance on the performance of our clustering protocol. An interesting observation is that approximately 56% of requests are completed in 22 seconds regardless of mobility parameters. This proves the robustness of FastMF with respect to node mobility.

Figure 2.15(b) illustrates the significant reduction in interactive web-browsing delay when using clustering. If no multi-hop forwarding is applied, the mean and standard deviation of data request completion time is 30.76 and 20.78 sec respectively, while controlled flooding results in 39.4578 and 45.37 sec respectively and FastMF clustering results in 14.47 and 16.70 sec respectively. Furthermore, near 60% of requests are completed in 10 seconds using FastMF whereas other scenarios require 40 seconds. FastMF shows significant improvement due to extension of connection time obtained by clustering (see Figure 2.13), modifications to MobilityFirst, caching, and adaptive flow control.

2.8 Conclusion

In this chapter, we designed and evaluated algorithms for improving the network connectivity of vehicular nodes based on named object architectures such as MobilityFirst. Through simulating a highly disconnected scenario, we showed a significant improvement in in-vehicle Internet access, data throughput for large file download, and data request completion time for interactive web-browsing in the presence of traffic from other nodes. This improvement was achieved due to clustering, multi-hop forwarding, delay tolerant and storage-capable routing, indirect association to Internet gateways, and in-network mobility management. We enable nodes with short-living links and intermittent Internet access to perform routing functionalities of MobilityFirst. Thus, the MobilityFirst services were extended over multiple hops.

Chapter 3

Cloud Gaming

3.1 Introduction

Cloud computing systems have facilitated offering gaming services through thin devices regardless of the location of the clients. The cloud gaming architecture, shown in Figure 3.1, consists of the following stages [18]:

The user inserts its commands through the keyboard or touch screen of a mobile terminal. These commands are recorded by input manager and transformed into packets by input encoder in the client device. The gameplay commands are then submitted to the game server over the Internet.

In the game server, the user input processing unit buffers the received gameplay commands. During each scheduled server access event, the simulation of the game is resumed for a fraction, one, or multiple inter-frame intervals. The buffered inputs/actions of the players and the inputs of the artificial intelligence of the game (aka the game AI) induce changes in the game state. The game state evolves continuously at the game server but the updates in game status are recorded at a certain tick rate. The game server generates responses in the form of instructions for the frame renderer to construct game frames. We refer to these instructions as updates since they update the renderer (and ultimately the player) on the status of the game. The game status updates may contain far fewer bytes than the images that they describe.

Graphic rendering unit translates game server updates to video frames (i.e., images to be displayed on the screen of the mobile client) and frame encoding unit compresses the output frames and sends the results back to the user device. A display buffer at the mobile client maintains buffered frames and sends them to display unit at a fixed rate. A player observes the game via a full-motion video at a fixed nominal frame rate. Each video frame displayed at the mobile client represents a sample of the game status

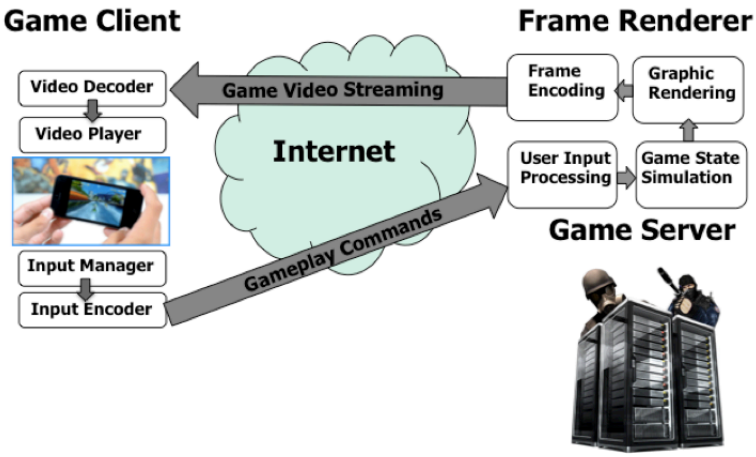

Figure 3.1: Cloud Gaming System Architecture

as provided to the player.

Our choice of transport layer for cloud gaming is UDP since state synchronization among endpoints through acks in TCP generates redundancy. Furthermore, the high throughput demand and stringent delay constraint in gaming is not compatible with packet retransmission and congestion control mechanisms in TCP. In streaming systems with UDP, packet loss creates "jump effect" since user actions may not get registered or game status updates may not arrive at the user device at a proper time.

One prominent factor in designing cloud gaming system architecture is the game genre. Games can be classified based on the type of offered user involvement [39]. First Person Avatar games (e.g., car racing games or first person shooter such as Half-Life, Counter-Strike, the Quake series, the Unreal Tournament series, and etc.) corresponds to the class of games in which the player views the game world from the viewpoint of his/her own avatar. These games are the most demanding in terms of responsiveness and [39] recommends latencies lower than 100 milliseconds. In Third Person Avatar games, the player sees the world from outside the body of his/her own avatar. These games have intermediate latency requirements (lower than 500 milliseconds) [39]. In Omnipresent (Real Time Strategy (RTS)) games, the player controls multiple elements in a large world. These games have more relaxed latency requirements (up to 1000 milliseconds). A summary of game genres is presented in table 3.1.

Game genre	Description
First Person Avatar	Player sees the world as his/her avatar (e.g., car racing or first person shooter games)
Third Person Avatar	Player sees the world from outside his/her avatar
MMORPG	Collaborative game among many people
Omnipresent (Real Time Strategy)	A strategic style of game playing involving planning and resource management

Table 3.1: Classifications of games

3.2 Problem Statement

The functionality enhancement presented by using cloud resources comes with the price of additional induced delay and constraint on data transmission. These limitations deteriorate the user experience. A key factor in success of cloud gaming systems is maintaining the viewer experience at the level of console games. Imperfections in the sequence of displayed frames will result in deviation of the display from a smooth presentation through reduced frame rates, frame freezings, and frame droppings. In the event of frame freezing, a frame is displayed for more than one frame interval. When a cloud gaming system has limited resources such as bandwidth, the displayed frame process will deviate from optimality. The resulting temporal artifacts create flickering (blinking), motion jerkiness, jittering, and reduce responsiveness [40].

In display flickering, adjacent frames at the same spatial location have different quality. In display jerkiness, changes in the frame rate results in a regular pattern in frames drops and movement of characters appear like a sequence of distinct snapshots. In display jitter, latency variance results in irregular and abrupt frame drops. The human eye can adapt to frame drops since the observed frames still have correlation and their content can be interpolated. In interactive applications such as gaming, lack of responsiveness results in inconsistency in the game status (e.g., a dead character starts shooting) which has an even worse impact on the gaming experience for players.

The impact of timing impairments (e.g., start-up latency, rebufferings, jitter, stutter, and bitrate changes) on QoE in video streaming systems has been studied before [41]. Rebuffering (pausing and resuming) is the most frustrating experience for video audience [41]. [42] investigated the QoE degradation due to frame rate reduction. They showed that having a long freeze is less disturbing to the viewers than a set of recurring

short stalling events. Furthermore, they concluded that as long as the system is experiencing temporal distortions (e.g. jitter), increasing bit rate and image quality does not improve QoE. Solutions focused on smart frame dropping and frame interpolation algorithms have been developed [40] for video streaming systems.

Considerable work has been accomplished to evaluate and improve the quality of bandwidth-efficient video streaming from cloud servers to user devices (e.g., Netflix, Hulu and YouTube); see, for example, [41,43]. To reduce the effect of network congestion and server limitations, large playback buffers and adaptive bitrate (ABR) streaming techniques are employed by popular video streaming systems such as Apples HTTP Live Streaming (HLS), Microsofts Smooth Streaming(SS), Adobes HTTP Dynamic Streaming, and Dynamic Adaptive Streaming over HTTP (DASH). In ABR streaming, the bandwidth depends on network conditions, number of packets in the buffer, screen resolution, and display refresh frequency.

In low-latency gaming, however, a large video buffer is a non-starter. For a 30 fps system, three buffered frames contribute 100 milliseconds delay. This latency constraint also precludes complex video compression methods [44] that code over multiple correlated frames. When image compression is restricted to individual frames, dynamic adjustment of the frame resolution based on client feedback could still be employed to facilitate smooth display. Practical techniques however use playback buffer backlog variations to signal rate changes [41] and thus are unsuitable for low-latency gaming.

Furthermore, network-layer QoS metrics such as packet throughput and delay do not precisely capture the viewer experience [17,18]. Packet delay may only loosely correlate with the timely delivery of a frame and it generally fails to describe the likelihood of missing and frozen frames. Inadequacy of these metrics in case of multi-player gaming where in addition to responsiveness, multiple clients must be working collaboratively, is even more clear.

In this work, we present a quantified representation of the user-perceived QoE of the cloud gaming system based on novel client-side measurements [45]. The users observe the games through a fixed frame rate video. A user's QoE is degraded when there are imperfections in the sequence of displayed frames. In here, we try to tackle QoE evaluation by direct examination of the missing frame process. Our proposed metric, is in terms of stream quality, and reflects the effects of network and processing delay

variability on a game's responsiveness. In contrast to video streaming systems, the value of each frame in gaming systems depends on the freshness of its contents and this value reduces over time.

As the main performance objective of a cloud gaming system is the "timely" update of players regarding the game status, we follow the concept of age of information introduced in [25] (see section 3.4.2) to characterize the system performance. Age of information approach measures system performance in terms of the average freshness of status updates. [46] used the age of information to optimize the status message generation process. Assuming that the game frames are status updates that are generated by game server, these packets form a queue and may wait for medium access, possibly retransmissions due to channel errors, backoff due to the activity of other wireless transmitters, and screen refresh instants until they can be displayed. In this system our design is not solely focused on minimizing bandwidth consumption or the delay in receiving frames. If the update rate is too low, the player will experience unnecessarily outdated status updates and lack of smoothness. On the other hand, If the update/frame rate increases beyond to enhance the smooth experience perceived by end users, the player will observe stream of stale and significantly lagged data and will be even more frustrated. Timely updating is not completely the same as minimizing the system delay or network traffic [46, 47].

In online gaming, a variety of predictive rendering techniques have been developed [48, 49] which enhances user-perceived timeliness at the expense of consistency, a measure of how accurately a video frame describes the true game state. In most of the proposed solutions, they still struggle with respect to response time minimization, graphical video encoding, network aware adaption, quality of experience (QoE) optimization, and cloud resource management. By contrast, we attempt to separate issues related to the depiction of the game state from the timeliness of the game state information at the client. Our approach is based on timeliness of the state information at the mobile client, rather than how that state is depicted for the client. While a game status update generated by the game server at time t is a snapshot of the game state at time t, networking and rendering delays imply that updates will be old by the time it is displayed at the mobile client. In particular we will use *status age* to measure how the game status, as displayed on the player's screen, lags the true game state.

In the most general setting, the mobile client, the game server, and the frame renderer are each entities connected by networks. In mobile gaming, the connection from the mobile client to the game server will include a wireless access link. In a conventional gaming system, the frame renderer is integrated in a relatively powerful client. In this case, the game server sends status updates across the wireless link to the mobile client. In our thin-client mobile setting, we consider two scenarios:

- The frame renderer may be integrated with the game server so that forwarded frames from the game server to the renderer takes negligible time. In this case, image frames are transmitted over the wireless link to the mobile client.
- The frame renderer may be a separate entity in the network. For example, the renderer may be located in an edge cloud server that renders image frames for many mobiles in the same local area [20].

3.3 System Model

Each of the network entities as well as the network interfaces may degrade the game performance. If the access network is slow, a player's actions will suffer queueing delay in the network. As the game server is likely to be processing game updates for multiple mobiles, possibly in multiple games, there may be queueing of actions at the game engine. Game updates, which may process multiple queued actions in a single frame, can have varying execution times as the task complexity can fluctuate. Randomness in game updates, in combination with randomness in the network can result in queueing of updates at the output of the game server. Randomness in the execution time required to render a frame can also result in both queueing at the input to the renderer and queueing of frames at the output of the frame renderer.

To analyze this complex situation, we propose to decompose the system. The "actions" submitted by the mobile are short instructions, possibly just a few bytes each in length. Furthermore the offered rate of these actions will be low; human responses at a rate of more than 100 actions per second are unlikely [50]. Thus, the traffic generated by the user actions will be small, on the order of 10 kb/s [17], relative to the data rates sustainable on even a moderate-rate access network. Thus in a well designed system, the delays in delivering these actions to the action queue at the game server should

be negligible, relative to either the delays in human response or the downlink delay of transmitted frames. The default setting for the game server is to process all queued actions for the player simultaneously.

Processing of the video frame requires a random service time that depends on the complexity of both the ongoing gameplay and the queued actions. The output of the game engine is a video frame or a set of instructions for rendering a video frame. We focus on the timely rendering of game server updates at the mobile display. Specifically, we assume updates are produced at a fixed frame rate f such that the mobile client displays a frame every $T = 1/f$ seconds. Specifically, for $k = 1, 2, \ldots$, the game server instantiates game status update k at time kT. That is, update k is the game server's authoritative update of the game state at time kT. We note that the game server may simulate the game evolution at a tick rate that is a multiple of f, but updates are generated only at the frame rate f.

In terms of the game state at the game server, user actions that arrive for processing in the interval $[(k-1)T, kT)$ are said to *occur* in that time interval. That is, latency in the user responses is assumed by the game server to be the latency of the user, rather than latency of the network in delivering those user actions. Under this model, consistency of the game state at the game server is maintained in that all actions that *occur* prior to time kT are incorporated in update k. That is, if the scheduler decides to generate a game update at time t, all actions in the queue at time are passed to the game update engine. Included in this set of actions is the virtual action of the game AI. If the action queue is empty, then only the virtual action of the game AI is processed.

The mobile client employs a time lag τ such that it schedules the display of frame k at time $T_k = kT + \tau$. In the interval $(kT, kT + \tau)$,

- the game server incorporates user inputs and game AI to produce update k;
- update k is sent, possibly through a network, to the frame renderer;
- the frame renderer generates the video frame k;
- frame k is sent to the mobile for display at time T_k.

If frame k is not ready to be displayed at time T_k, then the most recently received frame is displayed instead.

The mobile client can optimize the lag τ, subject to the limitations of its access network and frame renderer. We will see that the selection of the lag parameter τ can have a substantial effect on system age. In the context of a low-latency game, every frame matters, and the precise selection of the lag τ represents a tight (though intentionally lagging) synchronization of the game server and the mobile client.

In general, the game server sends game status updates via a network to the frame renderer. It is difficult to analyze this without making strong assumptions regarding both the network and the update delivery protocol. In prior status updating work, the disadvantages of queueing status updates have been noted in a variety of contexts [25,51]. Specifically, status age suffers if a service facility is processing old updates when newer updates are in the system. That is, the delivery of a newer update offers a larger reduction in status age and also obviates the need for discarding the older update. In short, the objective of the system is to display the most recently generated frame by avoiding the processing and queueing of old updates and frames at the expense of newer updates and frames.

To facilitate this goal, we adopt a form of preemptive stop-and-wait protocol for the forwarding of updates from the game server to the renderer. We describe this as an application-layer protocol, although its logic could be implemented at the transport layer. Specifically, the game server initiates the creation of update k at time kT. Each update may consist of multiple packets and contains the information to create one frame. After processing, the game server pushes update k to an output buffer that holds only the most recently generated update. If update $k-1$ is still in the output buffer at that time, it is preempted (i.e., discarded and replaced) by update k. The game server then attempts to send update k to the renderer. Update k remains in the game server output buffer until either a delivery acknowledgement is received from the renderer or it is preempted by update $k+1$. Note that no queueing occurs at the game server output buffer; only the most recently generated update is held in the buffer.

The logic of this preemptive service is assumed to be employed throughout the system. At the input to the frame renderer, update k is preempted if update $k+1$ arrives before update k is processed. At the network interface queue at the output of the frame renderer, frame k may be preempted by frame $k+1$. Ideally, frame k will be delivered to the mobile client in time for display at time T_k. Note, however, the frame

may still be useful to display even if it is received late. Specifically, the late arriving frame k will be displayed at time T_{k+1} if frame $k+1$ fails to be delivered on time. Thus, the system's effort to generate frame k may continue until time T_{k+1}.

3.4 The Update Age: An Analytic Model

To build a tractable analytic model, we assume that the time required for processing and sending update k plus rendering and sending frame k are described by a random variable Y_k that we refer to as the *update service time*. For the purpose of an analytic model, we further assume that the update service times $Y_1, Y_2, \ldots$ are independent and identically distributed (iid) sequence. In our system model, the role of Y_k in the display of frames is codified in these rules:

- At time T_k, the mobile displays its most recently buffered frame.
- If $Y_k \leq \tau$, then frame k will be displayed at time T_k.
- If $\tau < Y_k \leq T + \tau$, then frame k will be received by the mobile at time $kT + Y_k$. It will be displayed at time T_{k+1} unless it has been preempted by frame $k + 1$.

The idea behind this model is that until time T_k, the system makes its best effort to deliver and display frame k. However, starting at time $(k+1)T$, the system components give priority to the delivery of update/frame $k + 1$. In particular, the renderer may choose to terminate update k, either in rendering or in transmission, if its timely delivery to the mobile frame buffer appears to be unlikely.

This model is idealized in certain ways. First, if update/frame k is in transit in the network, protocol layering may preclude its immediate termination. If frame k is still in transit to the mobile client at time $(k+1)T$, it could ultimately be delivered and displayed at time T_{k+1}. Second, we note that backlogs in the network will generally result in dependencies among the delivery times Y_k. However, because the sending protocol has been designed to preclude the queueing of outdated frames, these dependencies will be chiefly the result of memory induced by network backlogs. In this case, even a high frame rate like 60 fps yields a new frame every 16 milliseconds, which, in the context of modern networks, may be sufficient to decorrelate the network response to successive packets.

Even with this simplified system model, basic tradeoffs between system configurations are not well understood. Where should the frame renderer be located? What frame rate optimizes the user experience? How should the lag τ be chosen? Here we analyze a status age metric to address these questions.

3.4.1 Notation

For random variable X, we denote the probability mass function (PMF) of X by $P_X(x) = \mathrm{P}[X = x]$, the cumulative distribution function (CDF) by $F_X(x) = \mathrm{P}[X \leq x]$ and the complementary CDF by $\overline{F}_X(X) = 1 - F_X(x)$.

3.4.2 Status Update Age Analysis

As described earlier, the age of information is a metric that evaluates the freshness of information observed by the client. It computes the time elapsed since the last instant that the client has received information from the server. The characteristics of age as a metric is different from throughput and delay. In a wireless network, by maximizing the frame rate, the throughput is maximized but since the update packets will form larger queues and induce extra delay, the client views stale information and the age increases. On the other hand, a very low frame rate reduces the waiting times in the queue but the client views stale information and the age increases. This trade-off shows the importance of devising a policy to optimize the waiting times in the queues.

Frame k represents the game status at its generation time kT. We characterize system performance by the status age process $\Delta(t)$. That is, if at time t the current displayed frame is k with timestamp kT, then the status age is $\Delta(t) = t - kT$. In the absence of a new frame being displayed, the status age $\Delta(t)$ grows with time. If frame k with timestamp kT is displayed at time $T_k = kT + \tau$, then $\Delta(T_k) = T_k - kT = \tau$ since frame k represents the game state τ seconds ago. The "age penalty function" [52] corresponds to the amount of dissatisfaction with respect to the data staleness.

Thus, the age process $\Delta(t)$ is given by the sawtooth function shown in Figure 3.2. Specifically, the age grows linearly at unit rate in the absence of a new frame while at time instance T_n, the age drops if a more recent frame is displayed. We now derive the *time-average* status update age $\Delta(t)$ using a graphical argument. Without loss of

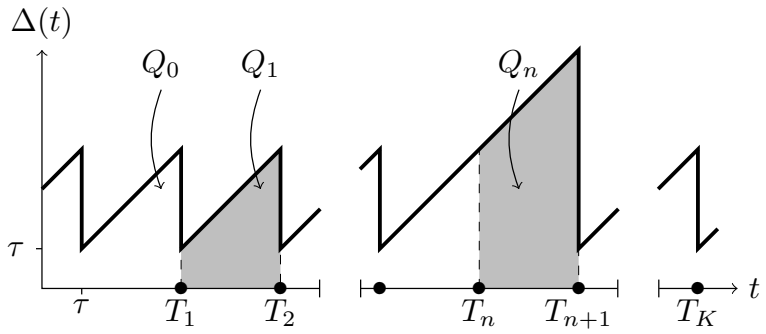

Figure 3.2: Example change in status update age $\Delta(t)$ (the upper envelope in bold) at the mobile client display. In this example, frames 1 and 2 are displayed on time, but frame n is lost. At time T_{n+1}, frame $n+1$ is displayed on time, which resets the age to τ.

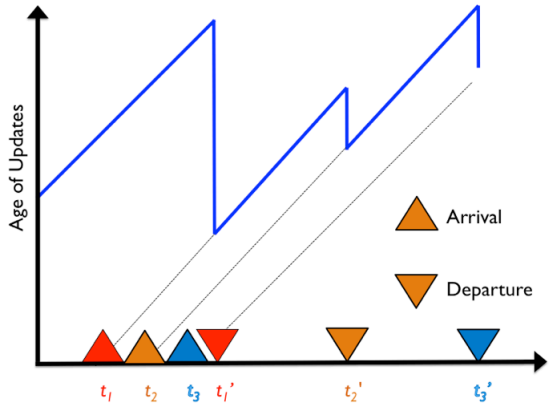

Figure 3.3: Age of client's information when frame rate is too high

generality, assume that we begin observing at $t = 0$ when frame $k = 0$ is displayed, so that $\Delta(0) = \tau$. Figure 3.3 corresponds to a scenario where the frame rate is too high. The large number of updates intensifies the congestion in the channel and as a result, the value that each frame brings will reduce.

The time-average age of the status updates is the area under the age graph in Figure 3.2 normalized by the time interval of observation. For simplicity of exposition, the observation interval is chosen to be $(\tau, KT + \tau)$. Over this K-frame interval, the average age is

$$\Delta^{(K)} = \frac{1}{KT} \int_{\tau}^{KT+\tau} \Delta(t)\, dt. \tag{3.1}$$

We decompose the area defined by the integral (3.1) into a sum of disjoint polygonal

areas $Q_0, Q_1, \ldots, Q_{K-1}$ (with Q_1 and Q_n highlighted in Figure 3.2.) This decomposition yields the average age over K frames

$$\Delta^{(K)} = \frac{1}{KT} \sum_{k=0}^{K-1} Q_k. \tag{3.2}$$

The area Q_k depends on the delivery of frames to the mobile display. At time T_k, let X_k denote the number of prior frames that have gone missing. That is, if frame k is displayed at time T_k, then no frames are missing and $X_k = 0$. If frames k and $k-1$ are missing, and frame $k-2$ is displayed at time T_k, then $X_k = 2$. From Figure 3.2, we see that each Q_k includes a right triangle of base and height T atop a base rectangle of width T and height τ. In addition, each missing frame at time k contributes a square of area T^2. This implies

$$Q_k = T^2/2 + \tau T + X_k T^2. \tag{3.3}$$

It follows from (3.2) that

$$\Delta^{(K)} = T\left[\frac{1}{2} + \frac{\tau}{T} + \frac{1}{K}\sum_{k=0}^{K-1} X_k\right]. \tag{3.4}$$

The time-average age is $\lim_{K\to\infty} \Delta_K$ and one would expect the normalized sum in (3.4) to converge to an average value of X_k. However, the missing frame process X_k has memory and modeling X_k is the key to characterizing the average age.

We first observe that $X_k = 0$ if $Y_k \leq \tau$. Similarly, $X_k = 1$ if $Y_k > \tau$ but $Y_{k-1} \leq T+\tau$. In principle, this process could continue indefinitely; that is, frame $k-i$ could be displayed at time T_k as long as $Y_{k-i} \leq iT + \tau$. In practice, however, delayed frames will be preempted. In order to model preemption, we assume that frame k will be preempted (i.e. discarded by the system) if $Y_k > (B-1)T+\tau$, where $B > 1$ is a backlog parameter. This constraint ensures that no more than B updates are kept in the system. Under this assumption, frame k may be displayed at times $T_k, T_{k+1}, \ldots, T_{k+B-1}$, but if it fails to be displayed by T_{k+B-1} then it will never be displayed. This implies

$$X_k = \min\left\{ i \geq 0 \;\middle|\; \begin{array}{l} Y_{k-i} \leq iT + \tau, \\ Y_{k-i} \leq (B-1)T + \tau. \end{array} \right\} \tag{3.5}$$

We note in (3.5) that the first condition ensures that frame $k-i$ is delivered by time T_k and the second condition enforces the buffering limit.

3.4.3 Missing Updates Markov Chain Analysis

Given that buffering outdated updates is generally a bad idea, we now present a Markov chain analysis of X_k process when $B = 2$. When $B = 2$, (3.5) simplifies to

$$X_k = \min\{i \geq 0 | Y_{k-i} \leq \min(1, i)T + \tau\}. \tag{3.6}$$

This implies

$$\mathrm{P}[X_k = 0] = \mathrm{P}[Y_k \leq \tau] \tag{3.7}$$

and for $j \geq 1$,

$$\begin{aligned}\mathrm{P}[X_k = j] = \mathrm{P}[&Y_{k-j} \leq T + \tau, Y_{k-j+1} > T + \tau, \\ &\ldots, Y_{k-1} > T + \tau, Y_k > \tau].\end{aligned} \tag{3.8}$$

With the assumption that the Y_k are iid, (3.7) and (3.8) imply that X_k has PMF

$$P_{X_k}(j) = \begin{cases} F_Y(\tau) & j = 0, \\ F_Y(T+\tau)\overline{F}_Y(T+\tau)^{j-1}\overline{F}_Y(\tau) & j \geq 1. \end{cases} \tag{3.9}$$

While (3.9) captures the marginal PMF of X_k, it does not fully reveal the Markov structure of the X_k process. In Appendix B, we show that X_k is described by the Markov chain shown in Figure 3.4 with transition probabilities

$$p_0 = \mathrm{P}[X_k = 0 | X_{k-1} = j] = F_Y(\tau), \tag{3.10}$$

and for $j \geq 1$,

$$p_1 = \mathrm{P}[X_k = 1 | X_{k-1} = j] = F_Y(T + \tau) - F_Y(\tau), \tag{3.11}$$

$$q = \mathrm{P}[X_k = j + 1 | X_{k-1} = j] = \overline{F}_Y(T + \tau). \tag{3.12}$$

The Markov chain is ergodic as long as $F_Y(T + \tau) > 0$. The stationary probabilities

$$\pi_j = \lim_{k\to\infty} \mathrm{P}[X_k = j] \tag{3.13}$$

satisfy

$$\pi_0 = \sum_{i=0}^{\infty} \pi_i F_Y(\tau) = F_Y(\tau), \tag{3.14}$$

$$\begin{aligned}\pi_1 &= \overline{F}_Y(\tau)\pi_0 + \sum_{i=1}^{\infty} \pi_i [F_Y(T+\tau) - F_Y(\tau)] \\ &= \pi_0 + (1-\pi_0)F_Y(T+\tau) - F_Y(\tau) \\ &= \overline{F}_Y(\tau)F_Y(T+\tau),\end{aligned} \tag{3.15}$$

and for $j \geq 1$,

$$\begin{aligned}\pi_j &= \overline{F}_Y(T+\tau)^{j-1}\pi_1 \\ &= \overline{F}_Y(\tau)F_Y(T+\tau)\overline{F}_Y(T+\tau)^{j-1}.\end{aligned} \tag{3.16}$$

As we would expect, (3.16) is consistent with the PMF of X_k given in (3.9). Ergodicity of the Markov chain implies that

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} \mathrm{E}[X_k] = \sum_{j=1}^{\infty} j\pi_j = \frac{\overline{F}_Y(\tau)}{F_Y(T+\tau)}. \tag{3.17}$$

The next claim then follows from (3.4) as $K \to \infty$.

Theorem 3.4.1. *The average age of the system with frame period T, lag τ, $0 \leq \tau < T$, and backlog limit $B = 2$ is*

$$\Delta_2(T,\tau) = T\left[\frac{1}{2} + \frac{\tau}{T} + \frac{\overline{F}_Y(\tau)}{F_Y(T+\tau)}\right].$$

Theorem 3.4.1 provides a simple characterization of the average in terms of the distribution of update delivery times. When the system is designed cautiously, $\overline{F}_Y(\tau) \approx 0$ and virtually all updates are delivered on time. In this case, Theorem 3.4.1 says $\Delta_2(T,\tau) \approx T/2 + \tau$ where $\Delta_2(T,\tau)$ measures how out-of-date the current video frame is when frames are delivered on time. We see that both the frame lag τ and the period T contribute. Specifically $T/2$ is the average age between periodic updates; when the

frame rate is low, the period T will be large. The frame lag τ is the response time of the system once a user action is incorporated in a game server update. The age metric incorporates both the lag in response time and the latency associated with periodic framing.

Despite the approximations made by the analytic model, we will see that it provides a surprisingly accurate calculation of the status age in a low-latency edge-cloud gaming system. For fixed T, we will see that Theorem 3.4.1 captures the complex way that the age varies with the lag τ. Nevertheless, Theorem 3.4.1 should be used carefully. Specifically, as the frame period T shrinks, the rate of updates/frames grows and queueing will cause the frame delivery time Y to increase.

3.4.4 Lag Periodicity of the Age

Theorem 3.4.1 describes the average age $\Delta_2(T,\tau)$ for $0 \le \tau < T$. One may wonder what benefit may be obtained by a lag $\tau \ge T$. To examine this, it is sufficient to consider lag $\tau = jT + \tau_0$ with $0 \le \tau_0 < T$. Here we refer to τ_0 as the local lag and j as the frame lag. At the mobile client, the display rule is that the most recently received frame is displayed at time $kT+\tau$. Thus, when $\tau = jT+\tau_0$, the most recently received frame at time $(k+j)T+\tau_0$ is displayed. However, as k and $k+j$ are arbitrary frame indices, we see the display rule depends only τ_0. That is, in the context of (3.5), the display rule at time T_k is to display the most recent frame $k-i$ such that $Y_{k-i} \le iT+\tau_0$. What changes, however, is that the frame lag j enables additional frames to be displayed rather than discarded. Specifically, the second condition in (3.5) requires

$$Y_{k-i} \le (B-1)T+\tau = (B+j-1)T+\tau_0. \tag{3.18}$$

To summarize, when $\tau = jT+\tau_0$, the X_k process is given by

$$X_k = \min\left\{ i \ge 0 \;\middle|\; \begin{array}{l} Y_{k-i} \le iT+\tau_0, \\ Y_{k-i} \le (B+j-1)T+\tau_0. \end{array} \right\} \tag{3.19}$$

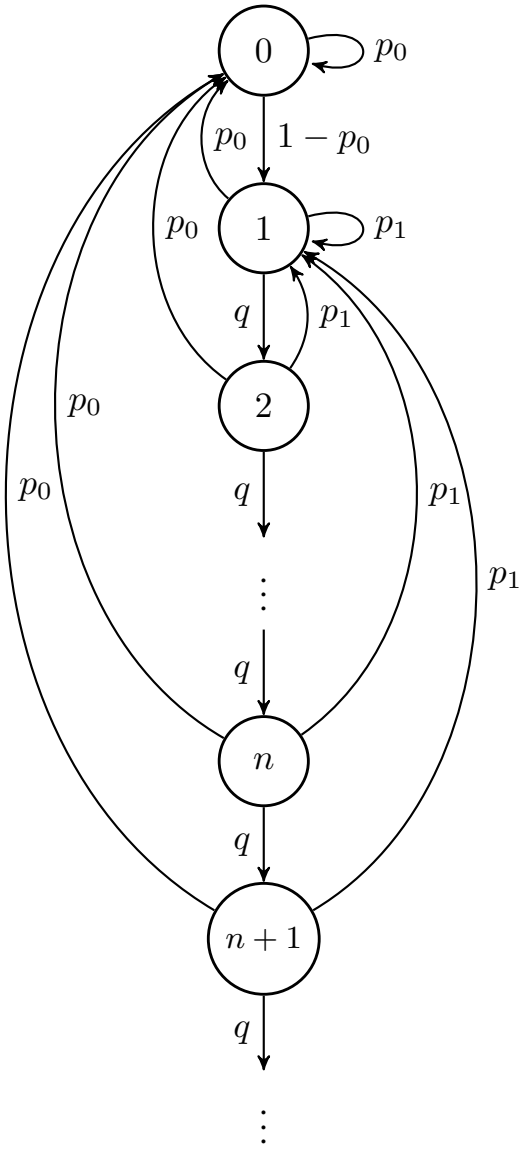

Figure 3.4: The discrete-time Markov chain X_k.

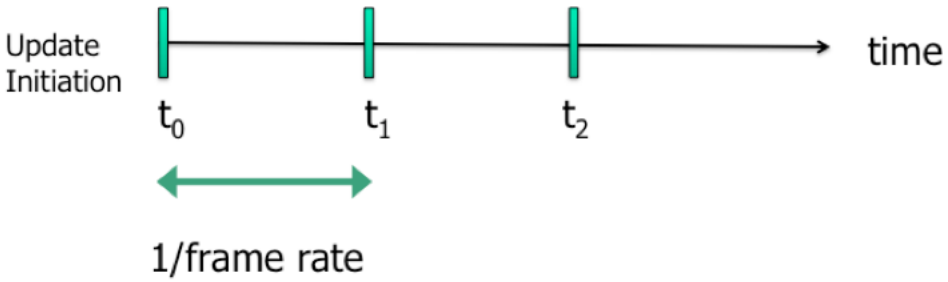

Figure 3.5: Time instance t_i corresponds to instances where server initializes game status updating

Comparing (3.19) and (3.5), we see that buffering limit B and lag $\tau = jT + \tau_0$ is identical to buffering limit $B + j$ and lag $\tau = \tau_0$. It then follows that

$$\Delta_B(T, jT + \tau_0) = \Delta_{B+j}(T, \tau_0). \tag{3.20}$$

When the system is well designed, $F_Y(BT + \tau) = 1 - \epsilon$ and almost all updates are delivered on time. In this case, the buffering limit B has almost no impact on system performance in that $\Delta_{B+j}(T, \tau_0) \approx \Delta_B(T, \tau_0)$. In such practical cases, the age is a periodic function of the lag τ. Henceforth we consider lags only in the interval $(0, T)$.

3.5 Timely Cloud Gaming Protocol Description

Based on the age of information metric introduced earlier, we developed a novel transmission scheduling protocol for improving responsiveness in cloud gaming streaming. Our protocol improves streaming quality and system responsiveness adapted to channel conditions. The adaptation indicates protocol's capability in proactively sensing lag-inducing events and dynamically adjusting the traffic load. In exchange, we are willing to introduce ideally sufficiently minor and isolated frame drops as long as the impact of these drops stays below the level of sensitivity of human eye and can be mitigated by the interpolation capability of human cognitive system. This protocol eliminates the aggregation of system-induced lags over multiple frames, avoids noticeable stalling events, and conserves the bandwidth for more fresh frames. This solution can be implemented in the application layer when transport layer is UDP.

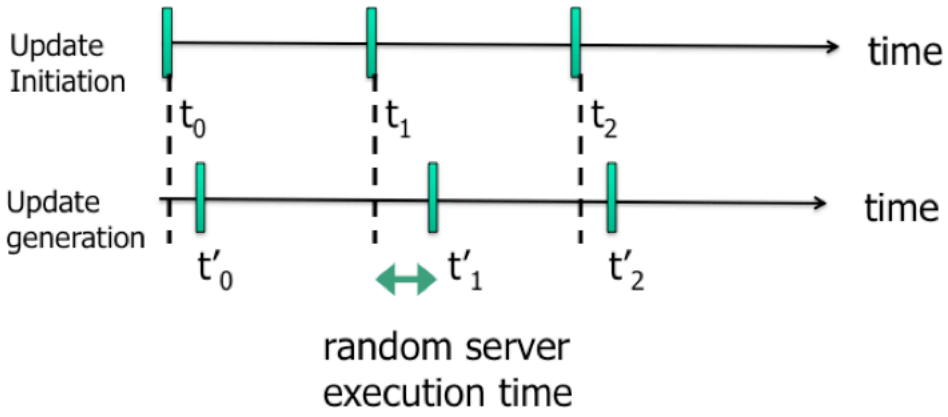

Figure 3.6: Time instance t'_i corresponds to instances where game status updating packets are generated

- Upon start of each scheduled server access event at times t_i illustrated in Figure 3.5, the simulation of the game is resumed and the time in the game world progresses for an inter-frame interval.

- Depending on the complexity of events occurred during game status updating interval and number of characters involved in each scene, a random processing time passes before game status updates are generated at t'_i in Figure 3.6.

- Based on the complexity of the frame, encoding complexity and available resources in the frame renderer, the compressed frames will be generated at random instants indicated by t''_i in Figure 3.7.

- The frames are streamed over the Internet, arrive at the client device and are decoded. $t_i + y_i$ instances in Figure 3.8 correspond to the moments when the frames are ready to be displayed. It is important to note that frames are not necessarily displayed at $t_i + y_i$ since the mobile client displays frames at a fixed frame rate. The ready-to-display frames are buffered in display buffer and wait until the screen refresh time arrives.

- To maintain the QoE for clients, one of the system requirements is having a constant screen refresh rate. $t_i + \tau$ in Figure 3.9 indicates the screen refresh times when τ is the intentional lag between server and client.

- Each video frame displayed at the mobile client represents a sample of the game status as provided to the player and reduces the age of his/her information. In

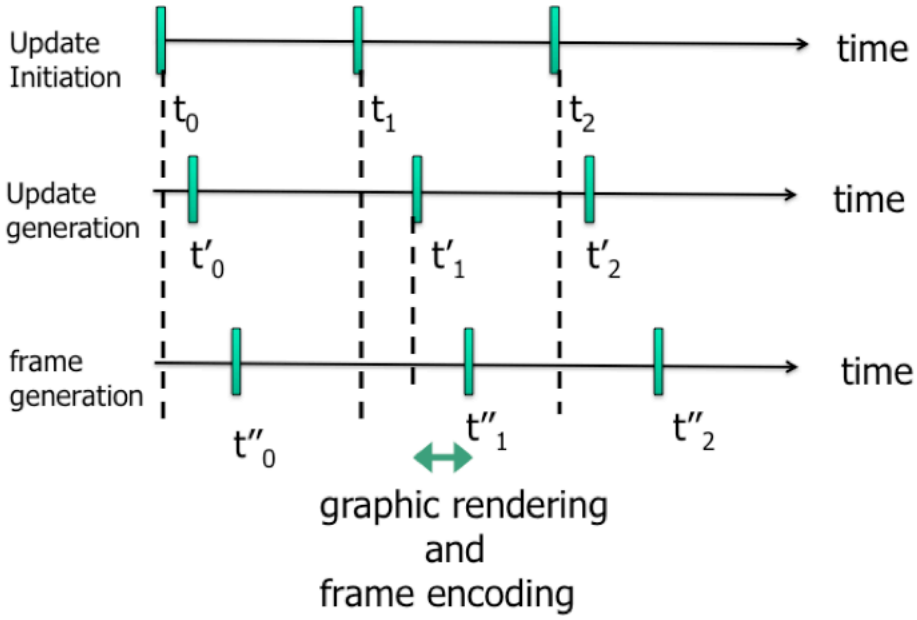

Figure 3.7: Time instance t''_i corresponds to instances where frames are generated

Figure 3.10, frame 0 was ready to be displayed before its intended display instant and as a result, the client's perception of the game is updated at $t_0 + \tau$.

- In Figure 3.11, frame 1 is not ready to be displayed before its intended display instant and as a result, the client's perception of the game is not updated at $t_1 + \tau$. This shows the sensitivity of the age of information metric to responsiveness in the game.

- In Figure 3.12, frame 1 is ready to be displayed before next display instant and as a result it remains in the display buffer. Since frame 2 is not ready to be displayed in its intended display instant, frame 1 is displayed and the client's perception of the game is updated at $t_2 + \tau$. However, the reduction in age is smaller than before to indicate the staleness of frame 1.

- In Figure 3.13, frame 1 is ready to be displayed before next display instant and as a result it remains in the display buffer. However, since the congestion in the downlink or server traffic was eliminated, frame 2 is ready to be displayed in its intended display instant. To avoid aggregation of such delays and propagation of that throughout the next multiple frames, frame 1 is dropped and frame 2 is displayed. The client's perception of the game is updated at $t_2 + \tau$ and the

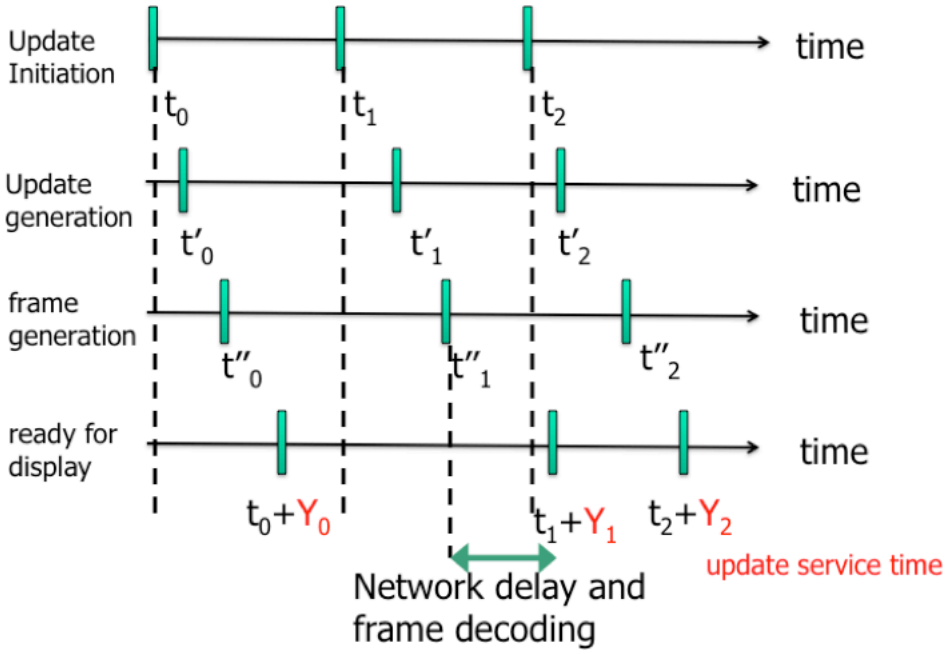

Figure 3.8: Time instance $t_i + y_i$ indicates the instances when the most freshly received frame is ready to be displayed

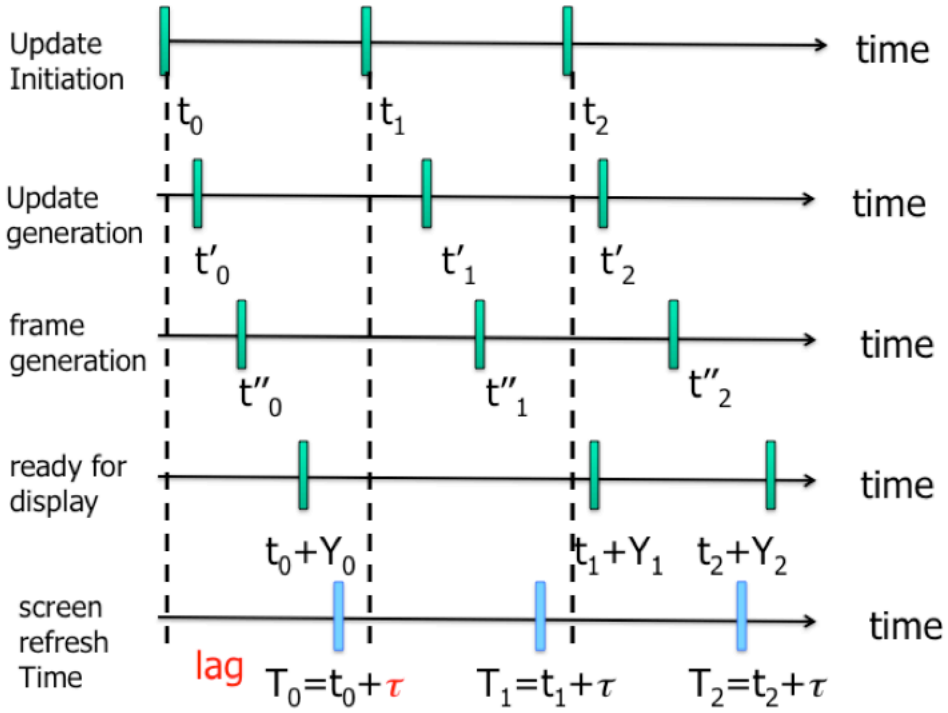

Figure 3.9: Time instance $t_i + \tau$ indicates the screen refresh times when τ is the intentional lag between server and client

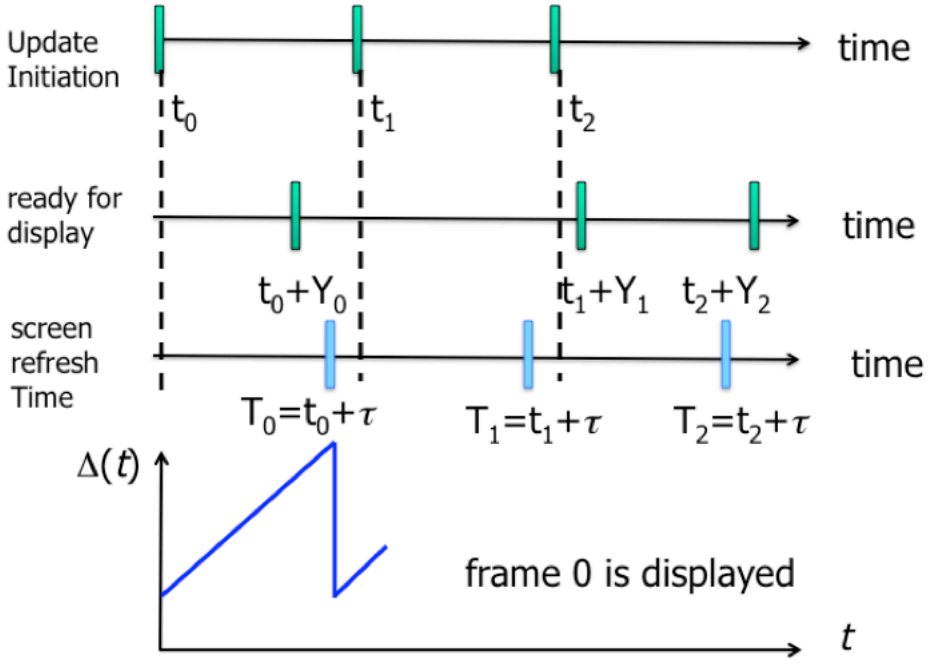

Figure 3.10: Variation of system age when a frame is displayed in its intended instant

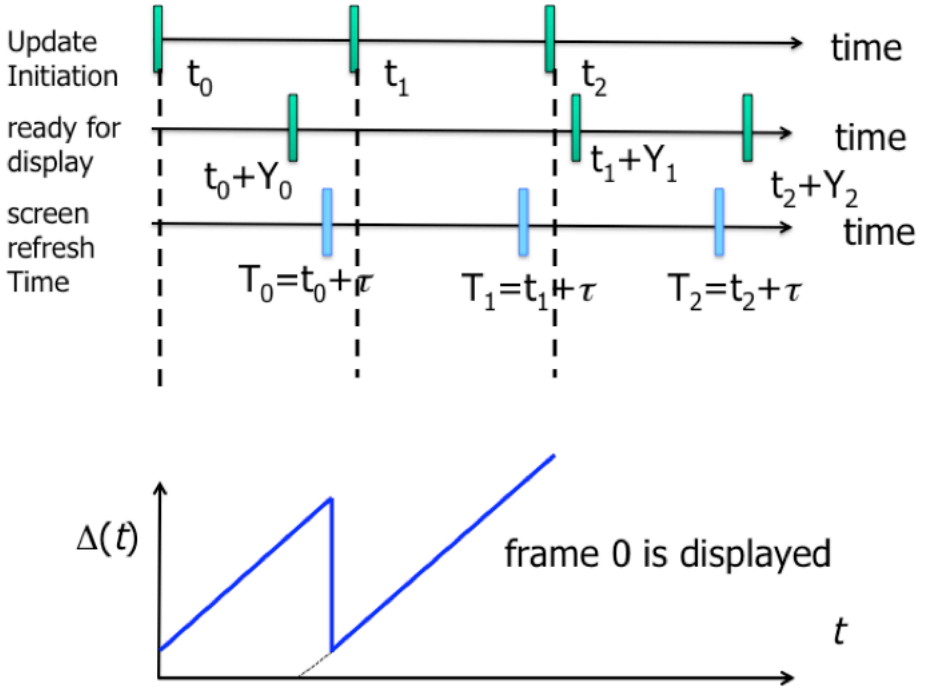

Figure 3.11: Increment in age of system age when a frame is not displayed in its intended instant

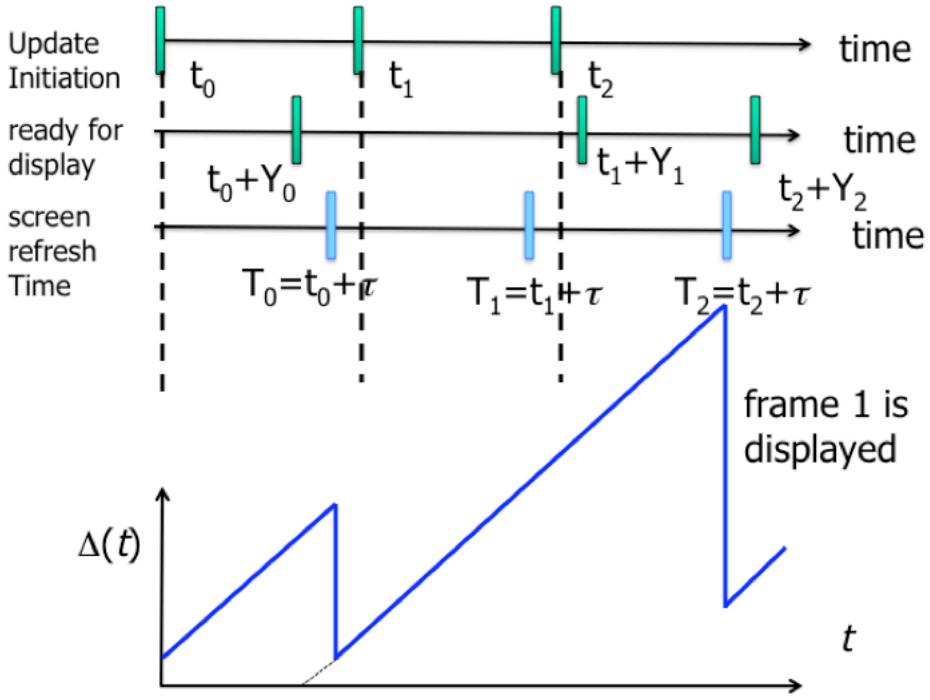

Figure 3.12: Variation of system age when an old frame is displayed at $t_2 + \tau$

reduction in age is larger than before to indicate the freshness of frame 2.

3.6 Evaluation

In this section we evaluate the role of the average age of the system as a metric in providing richer information on design and optimization of real-time interactive cloud-assisted applications. Our simulated cloud gaming system is illustrated in Figure 3.1.

To evaluate the proposed analytical model, we implemented a mobile cloud gaming scenario in ns-3 where the main parameters of this simulation are summarized in Tables 3.2 and 3.3. The simulation parameters are chosen to capture the complexity of cloud-gaming systems including bandwidth constraints, rendering, and encoding/decoding delays. In our simulations, we have considered both single-player games and geographically distributed multi-player games where in the multi-player scenario, different players submit their commands to the same game server, and the game server sends game status update messages to edge servers responsible for rendering the frames. Each client receives its frames from the edge server which is associated to.

Throughout this section, the term single-server refers to a scenario where processing

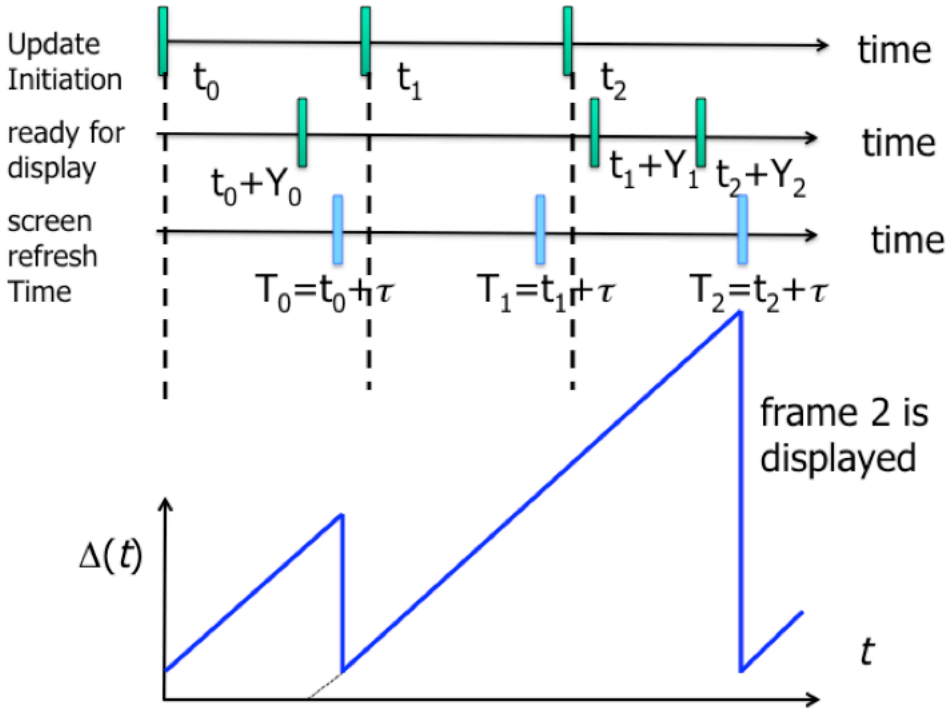

Figure 3.13: Variation of system age when an old frame is dropped and a more fresh frame is displayed at $t_2 + \tau$

tasks for both game status update and frame rendering are conducted in the same server. Similarly, the term multi-server refers to the alternative case where game status update is performed in a central server and frame rendering is done in edge cloud servers.

We start with a simple example in which the single-server execution duration has an exponential distribution with expected value $1/\mu$ and the transmission time is a constant y_0 such that $y_0 < T$. In this case, random variable Y has the shifted exponential distribution

$$F_Y(y) = \begin{cases} 0 & y < y_0, \\ 1 - e^{-\mu(y-y_0)} & y \geq y_0. \end{cases} \tag{3.21}$$

Similarly, in the multi-server scenario, if both game and rendering servers have exponentially distributed execution times each with expected value $1/\mu$ and the transmission time is a constant y_0 such that $y_0 < T$, then Y has the shifted Erlang distribution with

Parameter	Value
game-edge propagation delay	5 milliseconds, 1 Gbps link
game-edge bandwidth	1 Gbps
edge-client channel	802.11a with 6Mbps
command packet size	60 Bytes
frame size	4 KBytes
game status packet	1 KBytes
frame rate	30
command generation	Poisson process of rate 10
Simulation time	1000 sec
Transport protocol	UDP

Table 3.2: Simulation Parameters

Figure	analytical $F_Y(y)$
3.14	(3.21) $y_0 = 5.32$, $\mu = 1/14.5$
3.15	(3.22) $y_0 = 5.32$, $\mu = 1/7.95$
3.16	(3.22) $y_0 = 11$, $\mu = 1/4.9$
3.18	(3.21) $y_0 = 6.32$, $\mu = 1/16.6$
3.19	(3.22) $y_0 = 6.9$, $\mu = 1/8.1$

Table 3.3: Analytical $F_Y(y)$ parameters

shape parameter $k = 2$ such that

$$F_Y(y) = \begin{cases} 0 & y < y_0, \\ 1 - e^{-\mu(y-y_0)}\left(1 + \mu(y - y_0)\right) & y \geq y_0. \end{cases} \tag{3.22}$$

3.6.1 Analysis of the average age of the system

As indicated in section 3.4.2, the age of a system depends on target frame rate and the fine tuning parameter τ. Figures 3.14, 3.15, and 3.16 illustrate analytical and simulated average age of the system as a function of τ for lightly loaded systems. The system is lightly loaded in the sense that network delays can be approximated by the update/frame transmission times and that the game server and renderer do not have queued jobs. Figures 3.14 and 3.15 correspond to systems that have servers with exponential and Erlang execution interval distribution respectively. Figure 3.16 corresponds to a multi-player scenario with two separate exponential servers. In these systems, the randomness in Y_k derives solely from the randomness in processing and rendering. Comparing the analytical results with corresponding simulation graph, we

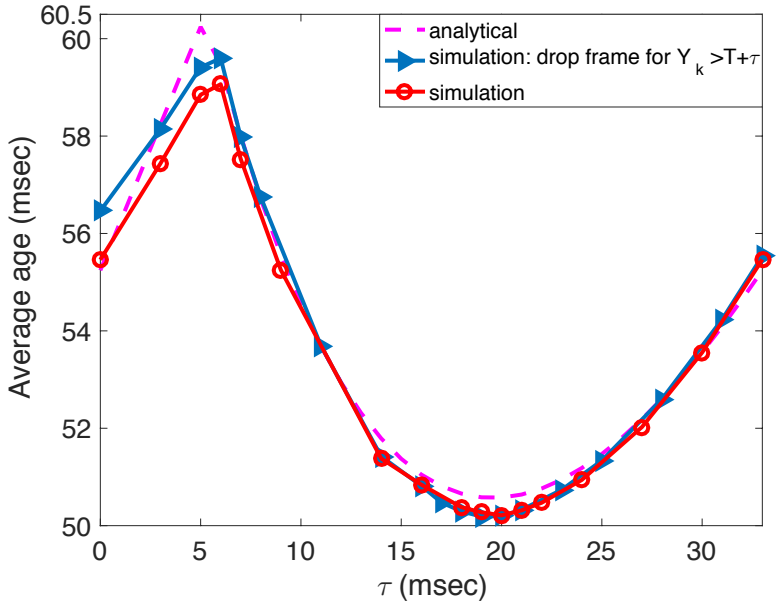

Figure 3.14: Average age of the system as a function of τ for exponential (mean 15 milliseconds) server

can see that Theorem 3.4.1 precisely describes the system.

In all three presented systems, the maximum age corresponds to having τ equal to the transmission delay. In other words, when the chosen τ consider service delay as 0, all arriving frames will be buffered and age for more than one frame period (interval between two consecutive frame display instants). If τ is less than the transmission delay, all arriving frames will be buffered but they age less since the upcoming frame display instant will be earlier. If τ is more than the transmission delay, the age of the system reduces since many frames will arrive before their corresponding display instant. As indicated in the presented results, depending on the frame rate and distribution of Y_k, there exists an optimal τ that minimizes the age. For τ larger than the optimal value, the performance will be degraded. By overestimating the system delay, most of the received frames will age unnecessarily and the average age of the system will grow.

As indicated in Figures 3.14, 3.15, and 3.16, the range of the values for system age is on the order of 50-63 milliseconds. Although this might look trivial, for delay sensitive applications, especially gaming systems, it would be considered large. As shown above, the age presents an interpretation for the system such that the client device can optimize

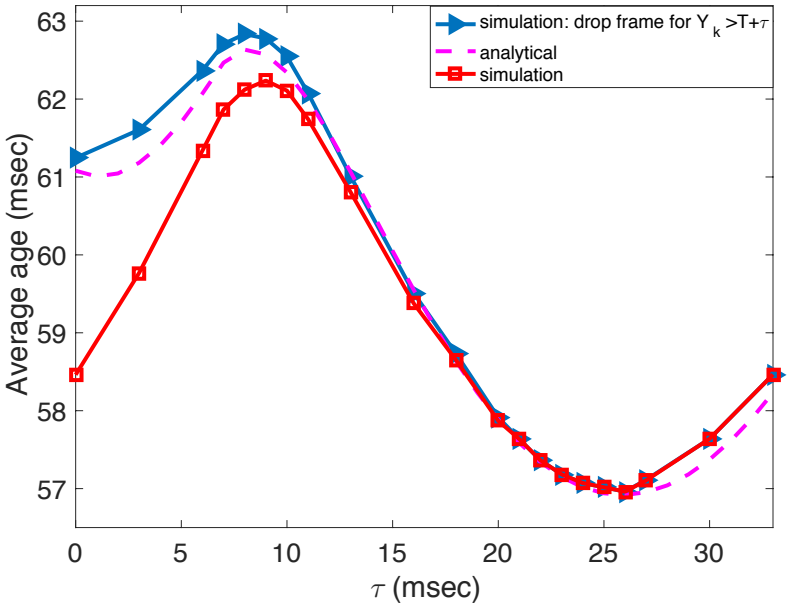

Figure 3.15: Average age of the system as a function of τ for Erlang $(k = 2, \mu = 0.1 \text{ msec}^{-1})$ server

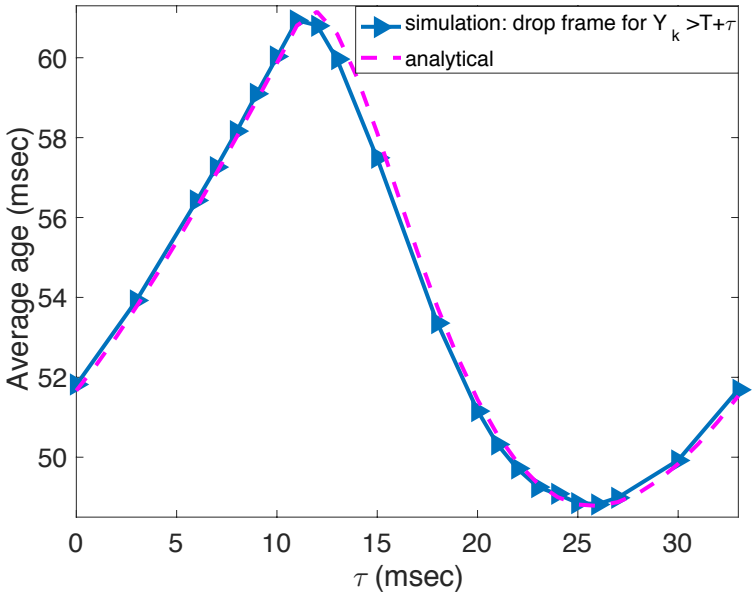

Figure 3.16: Average age of the system as a function of τ for Multi-player: game server Exponential (mean 10 milliseconds) and edge server Exponential (mean 5 milliseconds)

its synchronization with the server and adapt over time. This is possible for both single-player (Figures 3.14 and 3.15) and multi-player games (Figure 3.16).

In Figures 3.14 and 3.15, in addition to the $B = 2$ scenario, we simulated the $B \to \infty$ scenario called "soft" policy in which frames are dropped only if either server has sent a newer frame or receiver has received a newer frame. Comparing this soft policy with our model, we see that in lightly loaded systems for small values of τ, the soft policy will reduce age. However, for τ larger than the transmission delay, both policies correspond to the same amount of age. In fact, as indicated in section 3.4.4, the soft policy would correspond to having the age as a periodic function of τ with periods equal to frame display period.

Figure 3.17 illustrates the dependency of the average age on the system frame rate. We have compared lightly loaded channel with a busy channel for both $B = 2$ and soft policy systems. In all cases, low frame rates increase the average age of the system since the frames are sampling the game infrequently. As expected, the average age is higher in busy channels since frames suffer from queuing delay more. Furthermore, the age captures the compromise between frame rate and channel congestion. As an example, a system with 30 fps in 1.2 Mbps channel updates the clients with the same age as a 25 fps system in 6Mbps channel.

3.6.2 Effect of channel congestion on age

To investigate the effect of suboptimal network and server resources on age of the system, we simulated two scenarios with exponential servers where clients are sending with rate 30 fps but the available channel is limited to 2 Mbps and 1.2 Mbps. Figures 3.18 and 3.19 illustrate the resulting age as a function of τ along with the corresponding analytical curves. Comparing Figures 3.14 and 3.18, we can see that the designed frame selection protocol mitigates the impact of channel congestion and as a result, the performance of the system is not significantly degraded. Furthermore, the distribution of Y_k still can be approximated as shifted exponential and characterized by the analytical model. For τ larger than the channel transmission delay and smaller than the optimal τ, the distribution of Y_k is more similar to Erlang.

Comparing Figure 3.19 with Figures 3.18 and 3.14, we can see that in a more heavily congested scenario, the age of the system has increased. Moreover, since the frames for

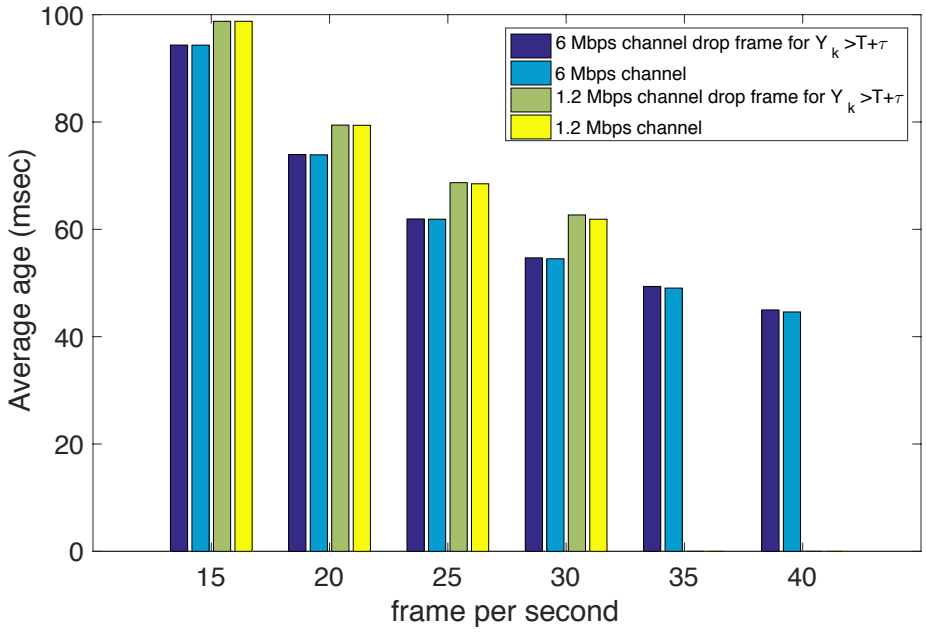

Figure 3.17: Average age of the system as a function of frame rate for various servers

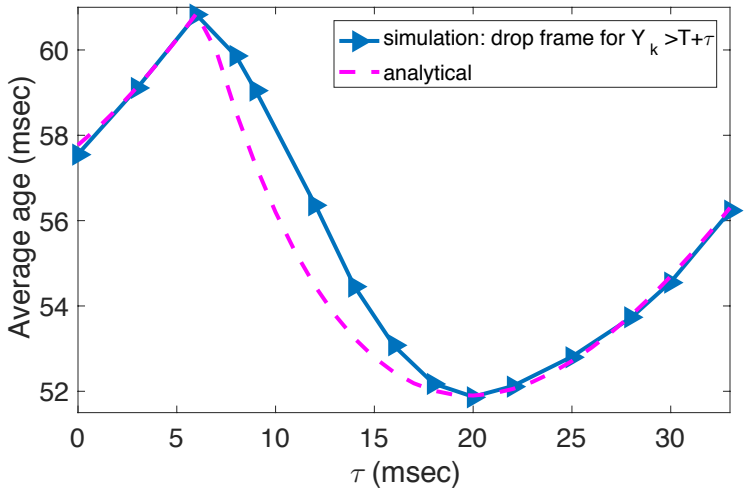

Figure 3.18: Average age of the system as a function of τ for 2 Mbps available channel

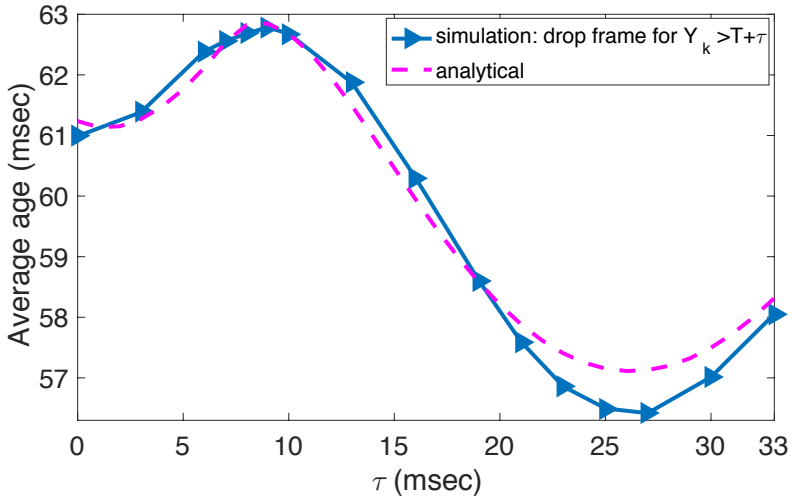

Figure 3.19: Average age of the system as a function of τ for 1.2 Mbps available channel

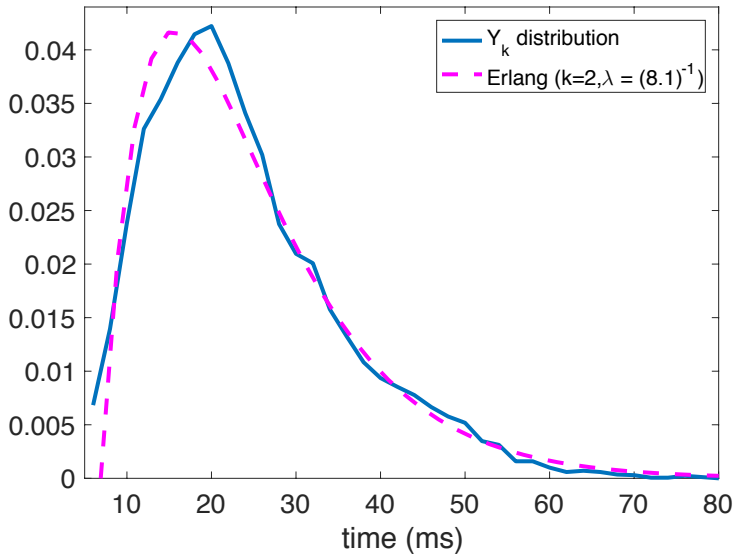

Figure 3.20: Distribution of Y_k for scenario with 1.2 Mbps available channel

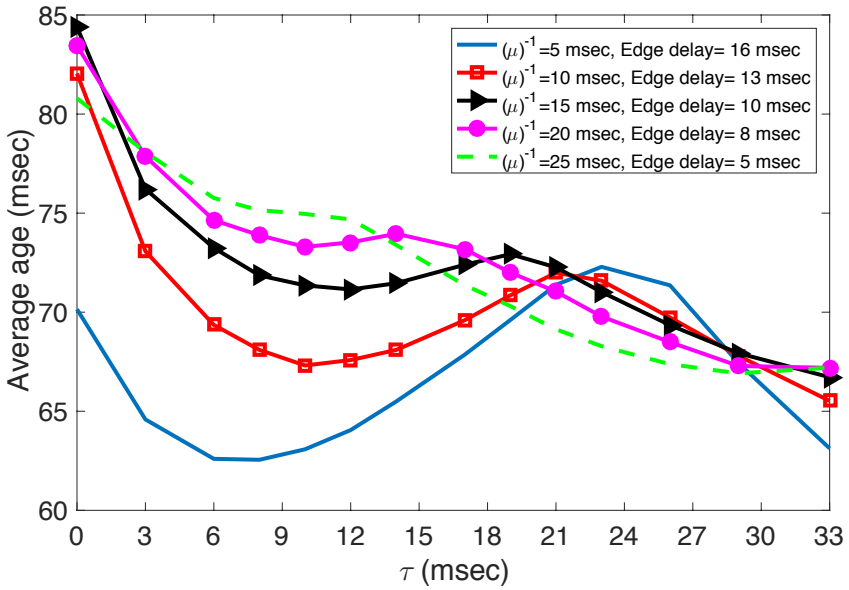

Figure 3.21: Average age of the system as a function of τ for different players of a geographically distributed multi-player game

different clients are queued in the network interface of the server, each game update experiences an exponential service time and an M/M/1 queue. Consequently, Y_k has Erlang ($k = 2$) distribution (see Figure 3.20).

3.6.3 Multi-player scenario

In this section, we simulated a multi-player scenario with geographically distributed set of edge servers where each player is associated to a different edge server with different processing power and edge delay (i.e., transmission time from edge server to the client terminal). Figure 3.21 illustrates the variations of age versus τ for different players. Each client terminal is able to use the tuning parameter τ to optimize its synchronization with the game server. Furthermore, this figure exhibits the potential of age of a system as an important tool for assigning game sessions to different edge servers since it incorporates various parameters that affect the subjective QoE for players.

3.7 Related Work: System Architecture

One of the first game streaming demos was provided by G-cluster in 2001 using PDAs over WiFi links [53]. [54] was one of the first academic works that introduced the cloud

gaming architecture and design specifics. By the late 2000's, companies such as OnLive and GaiKai were offering cloud gaming services [55]. OnLive's business model revolved around subscription services using geographically distributed game servers. Due to high incurred costs, by 2015 they closed the company. Gaikai used the cloud gaming platform to provide free game trials. The interested customers would then buy the game. Sony acquired Gaikai in 2012 and launched PlayStation Now in 2014 [56]. After analyzing the network traffic from OnLive game streaming systems, [57] concluded that its downlink is very similar to live videos. Furthermore, its traffic for first-Person, third-Person and omnipresent have the same features but different bitrates.

In [58], three main cloud gaming architectural frameworks were investigated: Remote Rendering (in the central game server or in edge servers), Local Rendering and Cognitive Resource Allocation. Browser games [59] are examples of local rendering where the gaming procedure is executed in the cloud servers and the game frames are rendered in the browser of mobile terminal using instructions from cloud servers. Such games usually rely on social networks (e.g., FarmVille on Facebook). The focus of this work is on edge-cloud rendering systems as described in [20]. Perhaps the most closely related system is the StreamMyGame (SMG) system evaluated in [17]. Just as in the experimental model evaluated in this work, the SMG system deploys a game server attached to the same local network as the client. Unlike this work, however, server processing times on the order of 300-400 milliseconds dominated the response time of the SMG system [17]. Instead of migrating all workload to a single server, [59] used a set of physically distributed servers and evaluated the impact of game type and content on cloud gaming design.

Some gaming frameworks are designed such that they can use unmodified game software in their cloud gaming systems whereas others try to modify the game's software to improve the performance [60].

3.8 Related Work: Performance Evaluation

To determine the perceptual QoE, both subjective and objective metrics have been developed. One of the subjective metrics is Mean-Opinion-Score (MOS) which corresponds to the results of surveying players and taking an average of their opinions [40].

One way to evaluate the QoE but not through explicit surveying is through user engagement with the game. Involvement of user can be evaluated through number of times playing a game and duration of each play [61] though this metric fails to evaluate how much player was focused on the game. These metrics capture more of the conditions of the game and game play situation. [62] used facial electromyography (FEMG) to evaluate the amount of satisfaction of game players for cloud gaming systems.

One of the main objective metrics is peak signal to noise ratio (PSNR) that measures the effect of spatial impairments in each frame but fails to consider the effect of temporal impairments. [63] evaluates the perceived jerkiness of a video as a function of display stalling length, video resolution, and the intensity of the displayed frame following the freezing event. The choice of these features and the impact (modeled by S function) that they have on jerkiness is determined based on the sensitivity of Human Visual System (HVS).

One of the main challenges in developing QoE metric for gaming systems is understanding the influential system characteristics. The subjective nature of QoE makes this task even more difficult. Analyzing network traces from running games, [64] evaluated the impact of network QoS (e.g., packet loss, latency, and jitter) on QoE (QoE was measured based on the duration of game sessions). They showed that RTT between 45-75 milliseconds decreases the length of the game session linearly. Furthermore, they found an even stronger correlation between game length and the standard deviation of latency (jitter). Maintaining a constant frame rate is proven to have a very important impact on the user performance in the game [65] where they recommended preserving frame rate at the cost of frame quality degradation. After investigating three different thin clients LogMeIn, TeamViewer, and UltraVNC, [21] reached the same conclusion about the effect of frame rate. They further showed that the frame quality has influence on QoE, though to a lesser degree. Through a more comprehensive analysis, [21] showed the role of frame rate, network delay, packet loss, contexts such as game genres and player skills, and graphic quality on QoE where network delay has the highest impact [62].

[66] decomposed the response delay of a gaming system into three parts: network delay, processing delay and playout delay. They measured the network delay and evaluated OnLive and StreamMyGame methods in masking the network delay.The network

delay consists of the uplink delay incurred when player's command is sent to the game server and downlink delay for returning game updates to the client. The processing delay corresponds to the time elapsed in server from receiving user command until updates are transmitted. The playout delay is the time frames are decoded and wait to be displayed. They concluded that OnLive modifies the resource allocation based on the game type and as a result has lower latency. In [17],they measured game delay as well and investigated the impact of scene complexity, size of updated regions, and screen resolution. In [67], the network load and traffic characteristics of cloud gaming and online gaming was compared in the context of packet inter-arrival time, size and inter-departure times. The [68] calculated the bandwidth usage of a variety of game genres and showed that it falls in the range of 3-4 Mb/sec regardless of player's action rate.

The sensitivity of the player to the responsiveness in the game depends on the genre of the game as well. For traditional gaming [39,69] proved that the sensitivity to latency will decrease in order for first person avatar, third person avatar, and omnipresent. In their experiments, they reported delays up to 100 milliseconds acceptable for the first person avatar genre and a delay up to 500 milliseconds can be acceptable to the third person avatar genre and a delay up to 1000 milliseconds can be acceptable to the omnipresent genre. [70] used FPS game "Call of Duty Modern Warfare 2" using subjective measures and divided the players based on their level of gaming experience and showed the 100 milliseconds latency threshold. [71] investigated delay in cloud gaming using subjective and objective measures. Their subjective tests were evaluated using their designed QoE metric. Their objective metric was the performance of player and their score. They show that the performance reduces up to 25% when delay is more than 100 milliseconds.

When network quality degrades, [72] showed that players are willing to sacrifice video quality (modified through quantization parameter) in favor of more responsive playing experience where network quality was evaluated using RTT jitter. [22] developed a scheduling and frame selection design for cloud gaming frame delivery over wireless networks to reduce the impact of delay based distortions. Outatime [60] is a predictive-based solution implemented on top of commercial games. It masks network latency and substitutes the lack of buffer with sending game frames predicted based on user-specific

behavior patterns one RTT in advance.

In the area of age of information, the analysis of the properties of queue-based models have dominated the emerging literature; see, for example [25, 51, 73–75]. However, these analytic models are approximations of real systems. A first effort to evaluate the accuracy of analytic models by simulation was provided in [76]. The ns-3 performance evaluation in Section 3.6 of the cloud gaming system is similar in spirit in that our analytic model is only an approximation of the gaming system with both networking and processing delays.

The assumption of iid delivery times in the analytic model is similar to the model in [74, 75] in which packets travel through a network on independent parallel paths. However, here the independence is motivated not by independent network paths but by a combination of independence in frame processing times and the temporal spacing of updates at a fixed frame rate.

3.9 Conclusion

In this work, we presented a quantified representation of the user-perceived QoE of latency-sensitive real-time cloud-assisted applications such as gaming based on missing frames. As the objective of a cloud gaming system is the "timely" update of players regarding the game status, we employed an age of information metric to characterize the system performance. The age metric incorporates both the lag in response time and the latency associated with periodic framing. Our designed metric, is in terms of both interaction delay and stream quality and captures the effect of stochastic network and processing delay variations on the game's responsiveness.

Based on the developed timeliness metric, we provide an analytical framework supported by extensive simulation for the problem of optimizing frame rate and lag synchronization of server and player. Based on the obtained results, age can be applied as a tool to synchronize game sessions based on the current status of the server. Furthermore, it can be used as a parameter in assigning edge servers to clients and designing resource allocation algorithms for cloud gaming systems.

We have suggested that user QoE in a low latency gaming system can be captured by the delayed or missed video frames, and proposed the average age of video frames as

a QoE metric. However, we observe that further study is needed to determine whether the average frame age is an effective measure of *user-perceived* QoE for various types of games with different sensitivity to latency. It may be that other metrics derived from the missing frame process X_k (e.g., the variance of X_k) better capture the user QoE. For example, if we say the system is in outage when $X_k > 0$, the average duration of outages may be a suitable metric. We note that our Markov chain analysis of the X_k process permits analytic consideration of these alternate metrics.

Chapter 4

Conclusion

The proliferation of mobile devices and cloud computing resources have contributed to the emergence of cloud-assisted applications. Cloud computing has created a new paradigm for mobile applications where computation and storage has migrated to centralized computing platforms in the cloud. This has drastically transformed the quality of experience for the users and increased popularity of cloud computing based applications. This thesis studied the design and implementation of solutions for support of advanced real-time applications with stringent constraints on responsiveness and throughput using cloud systems on mobile devices.

In the second chapter, we investigated the design and development of a network abstraction for the problem of maintaining connectivity and obtaining global reachability for vehicular networks using cloud servers as a global database of information with on-demand access. To address the challenges in in-vehicle communication including on-demand video streaming, safety applications, cloud-assisted autonomous driving, and virtual reality goggles, we determined the bottlenecks in PHY, MAC, network, and transport layers. Our scalable, mobility-centric solution, through "network abstractions" and clustering schemes with high resilience against failure, provided seamless connectivity, global reachability, and enhanced connection for vehicular nodes with high dynamicity and possibly no direct association to an access point.

To develop vehicular clustering for the purpose of obtaining best network attachment points along with cluster stability, we started with an analytical evaluation of vehicular mobility models that appears in Appendix A. This was followed by link lifetime analysis and description of mobility parameters during a contact between two cars. We derived a theoretical metric to evaluate the link lifetime that is mostly flexible with respect to mobility model. The vehicular clusters maintained tree structures that improve performance over conventional clusters, reduce control overhead and make them

adaptive to application requirements and network conditions.

This was followed with the extension of the naming service in MobilityFirst to enable support for highly dynamic scenarios. By aggregating vehicles in clusters, assigning unique identifiers to these clusters, and extending GNRS service to ad hoc mode, vehicles can benefit from multi-homing and other services available for fixed nodes in a seamless way. The proposed method can be added on top of IP or in an independent way as a network abstraction for highly dynamic scenarios. By simulation of various representative scenarios including content delivery, web content retrieval, and store and forward service, we show the flexibility and efficiency introduced by our design.

In chapter 3, cloud gaming was investigated where enhanced computing resources in the cloud servers results in delivery of high quality gaming experience through thin clients. In cloud gaming, the players receive the rendered game scenes which are streamed in real-time over network. One of the main issues at stake is maintaining responsiveness while having a smooth display of the game. What makes this problem more challenging is that the existing solutions for video streaming is not designed for latency-sensitive interactive applications.

In this regard, we introduced a novel application layer solution which supports flexibility of streaming with respect to the channel quality and efficiently adapts itself to variations in the channel. Our proposed protocol performed online frame selection to dynamically reduce traffic rate to conserve network resources. We presented the steps of this protocol and discussed system optimization based on responsiveness and display smoothness. Our final aim was optimizing quality of experience (QoE) for cloud gaming systems. This includes the minimization of age of information that the client perceives subject to improving QoE and maintaining constant display frame rate and smoothness. Through conducting extensive analytical modeling and network simulation, we showed the applicability of proposed model and protocol in improving the gaming experience. Using the proposed protocol, we can have a more steady game streaming.

Appendix A
Vehicular Movement Modeling

To form stable vehicular clusters for the purpose of providing high-throughput Internet access to the nodes with poor Internet connections, we study the characteristics of vehicular mobility models in highways and cities. Furthermore, we investigate the link stability evaluation metrics which are crucial to forming long-living clusters and eliminating redundant exchanges of control messages related to short-living links. Finally, after conducting an analytical study of the vehicular mobility models, we provide an insight into the relative mobility parameters of two cars connected with peer-to-peer WiFi links and conclude with a new probabilistic link stability assessment metric.

A.1 Analysis of Mobility Events

In the simplest mobility model, vehicles maintain a constant speed throughout their trips. Two vehicles may come into contact if the faster vehicle is located behind the slower car and passes over. One of the widely used metrics to evaluate stability of a link is introduced in Associativity Based Routing (ABR) protocol [77]. The ABR metric considers a link stable if the link lifetime is at least $2r/\triangle v$ where r is the coverage radius of each vehicle and $\triangle v$ is the difference between the most recent speed samples of the two nodes. [77] showed that for links younger than the ABR threshold, by increasing the time elapsed since the link formation instant, the residual link lifetime decreases. When links are older than the ABR threshold, the residual link lifetime is independent from the age of the link. Furthermore, [77] proved that the link lifetime distribution has exponential tail and its mode is approximately equal to the ABR threshold.

A more realistic mobility model for vehicles allows variations of speed over time. One of the main mobility models used for MANETs is Random Way Point (RWP) model [78] in which nodes randomly and independently from each other select their

next speed and direction (or destination). After reaching the destination, the nodes pause for a random time and choose new speeds and directions. Although RWP model captures the variations in speed and direction over time, in reality the cars move with more constraints since they follow roads and highways. One aspect of the mobility pattern of a vehicle in real life is the smooth variation in speed and direction such that consecutive samples of the velocity over time are correlated [79]. Due to the structure of the roads, sudden stops and sharp turns are not very prominent mobility events. To capture this change, [80] applied speed limits per lane in their proposed model.

If vehicular speeds are independent and identically distributed random variables that follow the Gaussian distribution with mean μ_v and standard deviation σ_v, $V_i(t) \sim \mathcal{N}(\mu_v, \sigma_v)$, the distribution of speed difference between any two vehicles will correspond to $\triangle V(t) \sim \mathcal{N}\left(0, \sqrt{2}\sigma_v\right)$. However, given that the two cars are entering the WiFi coverage area of each other, the speed difference between them approximately follows

$$f_{\triangle V_{\text{meeting}}(t)}(y) = \frac{|y| f_{\triangle V(t)}(y)}{\int |y'| f_{\triangle V'(t)}(y') dy'}, \tag{A.1}$$

where $f_{\triangle V(t)}(y)$ is the probability distribution for $\triangle V(t)$ in general and $f_{\triangle V_{\text{meeting}}}(y)$ is the probability distribution for $\triangle V(t)$ when nodes are meeting.

Figure A.1 compares the distribution of $\triangle V_{\text{meeting}}(t)$ derived from simulation data and the analytical distribution proposed in (A.1). Based on this figure, not only vehicles with significant speed difference, but also vehicles with approximately the same speed are very unlikely to meet. In fact, two vehicles are more likely to meet and form stable links if initially they have a considerable speed difference and after meeting, their speed difference reduces.

A mobility model that fully captures smooth correlated variations in movements is the Gauss-Markov mobility model [37, 81]. In this model, the jth speed sample v_j evolves as a convex combination of the previous speed sample v_{j-1} and target speed v_α as follows:

$$v_j = \zeta v_{j-1} + (1-\zeta) v_\alpha + \sigma_v \sqrt{1-\zeta^2} w_{j-1}, \tag{A.2}$$

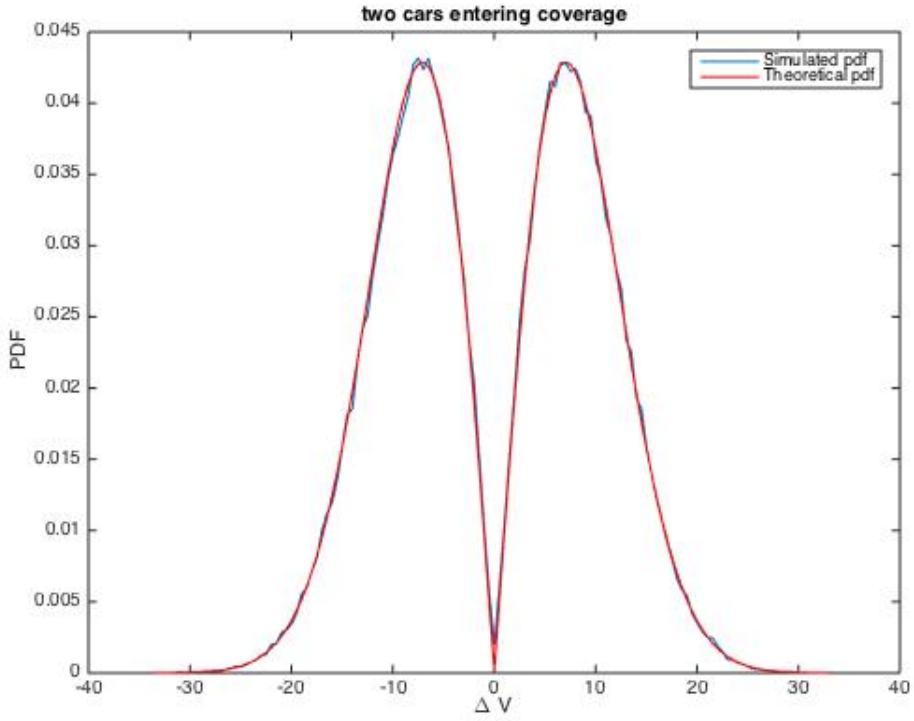

Figure A.1: probability distribution of $\triangle V_{\text{meeting}}$ where $V_i \sim \mathcal{N}(\mu_v, 5)$

where the asymptotic mean and variance of v_j are v_α and σ_v^2 respectively, ζ parameterizes the process memory, and w_{j-1} is an i.i.d. zero mean and unit variance Gaussian process such that w_{j-1} is independent of v_{j-1}. Figure A.2 illustrates the link lifetime distribution for vehicles moving based on (A.2) in a highway with average speed around 27 m/s. This figure highlights the high possibility of short lifetime links. The mode of the distribution is 27 seconds that is equal to the ABR threshold.

The target speed v_α is a random variable that allows the mobility model to adapt to different environments including intersection, highway, and traffic jam. To mimic the mutual interactions between vehicles, such as overtaking, traffic jam, and preferred paths, the distribution of target speed for neighboring nodes can be correlated. The time interval between two consecutive updates of target speed, called T_{target} is an exponentially distributed random variable (with average λ seconds). Starting from the time that a pair of cars meet, the time until one of the target speed changes is exponential of rate 2λ. Figure A.3 illustrates an example of the speed and target speed over time.

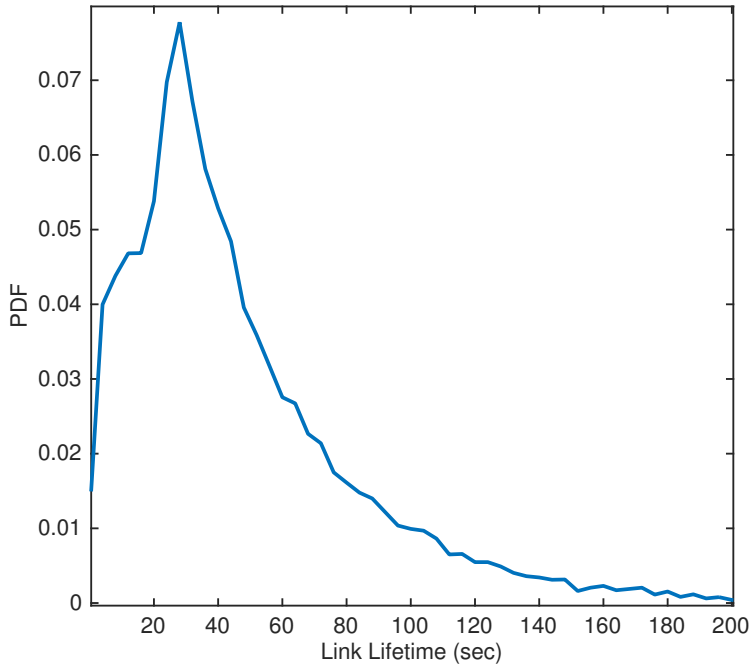

Figure A.2: Link lifetime distribution for a highway scenario with ABR threshold of 27 seconds.

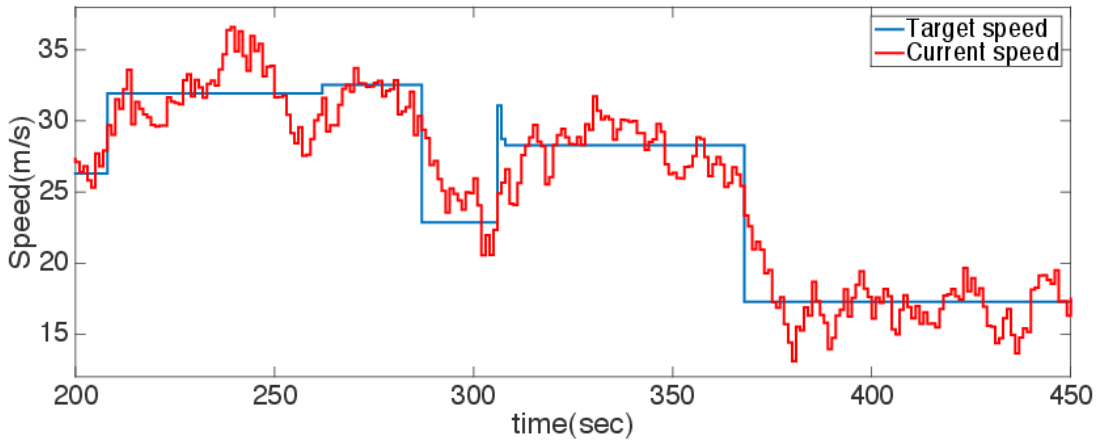

Figure A.3: A sample of the evolution of speed over time based on Gauss-Markov mobility model

A.2 Speed Process Dependencies

The accuracy of the link stability predictions that a car makes depends on the precision in estimation of future speed samples based on past speed samples. From (A.2), for an arbitrary node A, assuming constant target speed with distribution $V_\alpha \sim \mathcal{N}\left(\mu_\alpha^{\mathrm{A}}, \sigma_{v_\alpha^A}\right)$ and initial speed v_0,

$$v_j^{\mathrm{A}} = \zeta^{j,\mathrm{A}} v_0^{\mathrm{A}} + (1-\zeta^{j,\mathrm{A}}) v_\alpha^{\mathrm{A}} + \sigma_{v_\alpha^{\mathrm{A}}} \sqrt{1-\zeta^{2,\mathrm{A}}} \sum_{m=0}^{j-1} \zeta^{j-m-1} W_{m-1}^{\mathrm{A}}. \tag{A.3}$$

As j goes to infinity, the variance of v_j approaches $2\sigma_{v_\alpha^A}^2$. Assuming the speed of A and its neighbors follow independent and identically distributed random processes, we investigate the speed difference process between A and its neighbor B indicated by $\triangle V(j)$. The autocovariance of $\triangle V(j)$ given the target and initial speed of each node is

$$\begin{aligned}
C(j, j+\tau) &= E[(\triangle V(j) - \mu_{\triangle V(j)})(\triangle V(j+\tau) - \mu_{\triangle V(j+\tau)})] \\
&= E\left[\sigma_{v_\alpha}^2 (1-\zeta^2) \left(\sum_{m=0}^{j-1} \zeta^{j-m-1+\tau} \zeta^{j-m-1} W_{m-1}^{2,A} \right.\right. \\
&\quad \left.\left. + \sum_{m=0}^{j-1} \zeta^{j-m-1+\tau} \zeta^{j-m-1} W_{m-1}^{2,B} \right)\right] \\
&= 2\sigma_{v_\alpha}^2 (1-\zeta^2) \zeta^{2j-2+\tau} \left(\frac{1-\zeta^{-2j}}{1-\zeta^{-2}} \right) \\
&= 2\sigma_{v_\alpha}^2 \zeta^\tau (1-\zeta^{2j}).
\end{aligned}$$

Figure A.4 illustrates the autocovariance for $v_0 \sim \mathcal{N}(27,5)$, $v_\alpha \sim \mathcal{N}(27,5)$, and $\zeta = 0.5$. When j grows sufficiently, the speed difference process becomes wide sense stationary. Given the standard deviation of $\triangle V(j+\tau)$ process as

$$\begin{aligned}
\sigma_{\triangle V}(j+\tau) &= \sqrt{E[(\triangle V(j+\tau) - \mu_{\triangle V(j+\tau)})(\triangle V(j+\tau) - \mu_{\triangle V(j+\tau)})]} \\
&= \sqrt{E[\sigma_{v_\alpha}^2 (1-\zeta^2)(\sum_{m=0}^{j-1+\tau} \zeta^{2(j+\tau-m-1)} W_{m-1}^{2,A} + \sum_{m=0}^{j+\tau-1} \zeta^{2(j+\tau-m-1)} W_{m-1}^{2,B})]} \\
&= \sqrt{2\sigma_{v_\alpha}^2 (1-\zeta^2) \zeta^{2j+2\tau-2} \left(\frac{1-\zeta^{-2(j+\tau)}}{1-\zeta^{-2}} \right)} \\
&= \sqrt{2\sigma_{v_\alpha}^2 (1-\zeta^{2(j+\tau)})},
\end{aligned}$$

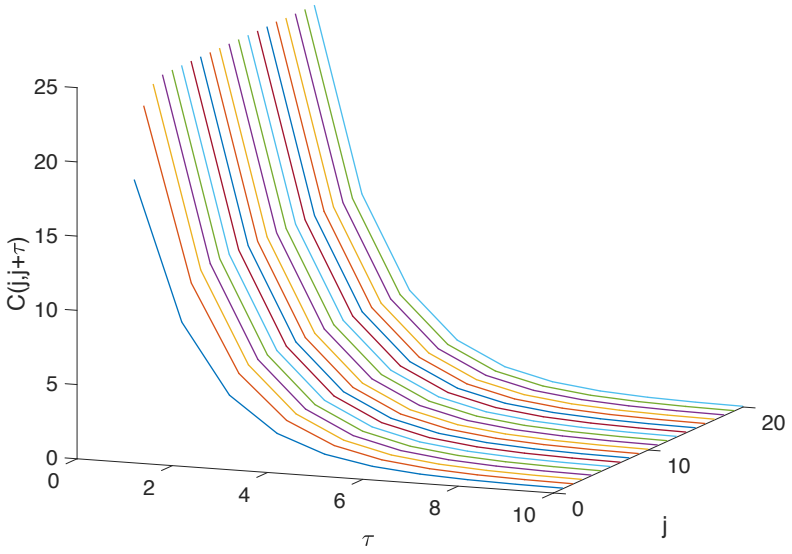

Figure A.4: The autocorvariance for speed difference process when $v_0 \sim \mathcal{N}(27, 5)$, $v_\alpha \sim \mathcal{N}(27, 5)$, and $\zeta = 0.5$

the autocorrelation coefficient for $\tau > 0$ is

$$\begin{aligned} \rho(j,\tau) &= \frac{C(j,\tau)}{\sigma_{\triangle V}(j)\sigma_{\triangle V}(j+\tau)} \\ &= \frac{2\sigma_{v_\alpha}^2 \zeta^\tau (1-\zeta^{2j})}{\sqrt{2\sigma_{v_\alpha}^2(1-\zeta^{2(j+\tau)})}\sqrt{2\sigma_{v_\alpha}^2(1-\zeta^{2j})}} \\ &= \frac{\zeta^\tau \sqrt{(1-\zeta^{2j})}}{\sqrt{(1-\zeta^{2(j+\tau)})}}. \end{aligned}$$

Figure A.5 illustrates the autocorrelation coefficient for $v_0 \sim \mathcal{N}(27, 5)$, $v_\alpha \sim \mathcal{N}(27, 5)$, and $\zeta = 0.5$. It can be seen that $\rho(j, \tau)$ between consecutive samples is significant, and when j increases, $\rho(j, \tau)$ is not a function of time any more. Figure A.5 shows that for estimating future samples of speed difference process, a limited number of past samples is sufficient.

A.3 Link Stability Evaluation

Defining link lifetime as the length of the interval that two cars remain in coverage area of each other, it can be formulated as

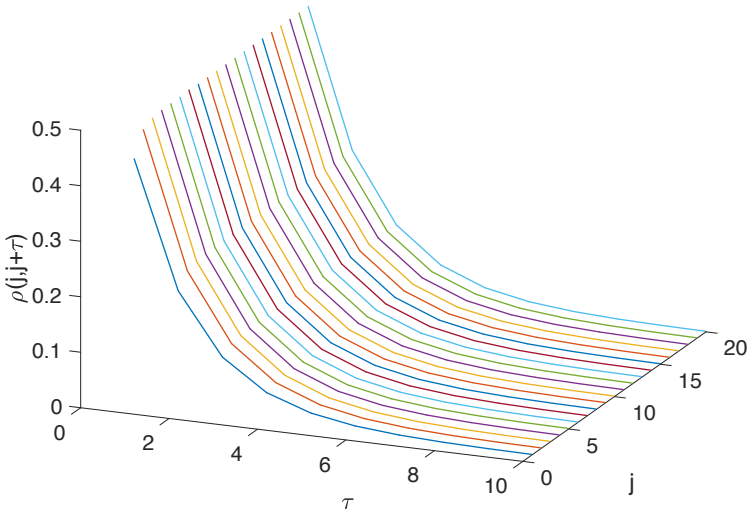

Figure A.5: The autocorrelation coefficient for speed difference process when $v_0 \sim \mathcal{N}(27, 5)$, $v_\alpha \sim \mathcal{N}(27, 5)$, and $\zeta = 0.5$

$$\max_{j \geq 0} j$$
$$\sum_{i=0}^{j} \triangle v_i \leq 2D_{\text{cov}}$$

where D_{cov} is the coverage radius. Link lifetime prediction depends on the accuracy of the estimation of the distance between two cars. In the rest of this section, we introduce three scenarios where simplifying assumptions help estimate the distance between cars.

Scenario 1: Assuming that the two cars have the same target speed and the target speed and ζ are known, the distance between cars at time j is

$$S(j) = \sum_{i=0}^{j} \triangle v_i - D_{\text{cov}}.$$

Peer-to-peer link exists if $-D_{cov} \leq S(j) \leq D_{cov}$. The estimated distance at time j is

$$\widehat{S(j)} = \sum_{i=0}^{j} \widehat{\triangle v}_i - D_{\text{cov}} = \sum_{i=0}^{j} \triangle v_{\text{meet}} \zeta^i - D_{\text{cov}} = \triangle v_{\text{meet}} \frac{1 - \zeta^{j+1}}{1 - \zeta} - D_{\text{cov}},$$

which asymptotically, when $j \to \infty$, is $\triangle v_{\text{meet}}/(1-\zeta)$ where $\triangle v_{\text{meet}}$ corresponds to the difference between the earliest speed samples of vehicles. The mean squared error in

the estimation of $S(j)$ will be

$$\begin{aligned}
\mathrm{MSE}[\widehat{\mathrm{S(j)}}] &= E\left[\left(\widehat{S(j)} - S(j)\right)^2\right] \\
&= E\left[\left(\triangle v_{\mathrm{meet}} \frac{1-\zeta^{j+1}}{1-\zeta} - \sum_{i=0}^{j} \triangle v_i\right)^2\right] \\
&= E\left[\left(\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{i-m-1}W_{m-1}^{A}\right.\right. \\
&\quad \left.\left. -\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{i-m-1}W_{m-1}^{B}\right)^2\right] \\
&= 2\sigma_\alpha^2\left(1-\zeta^2\right)\left[\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{2(i-m-1)}\right] \\
&= 2\sigma_\alpha^2\sum_{i=0}^{j}(1-\zeta^{2i}) \\
&= 2\sigma_\alpha^2\left[j+1-\frac{1-\zeta^{2j+2}}{1-\zeta^2}\right].
\end{aligned}$$

The $\mathrm{MSE}[\widehat{\mathrm{S(j)}}]$ increases unboundedly when $j \to \infty$. However, in this work, the performance of estimator in a limited time is of our interest. The baseline estimator (which assumes vehicles have constant speed) will be

$$\begin{aligned}
\mathrm{MSE}_{\mathrm{baseline}}[\widehat{\mathrm{S(j)}}] &= E\left[\left(\widehat{S(j)} - S(j)\right)^2\right] \\
&= E\left[\left(\triangle v_{\mathrm{meet}} j - \sum_{i=0}^{j} \triangle v_i\right)^2\right] \\
&= E\left[\left(\triangle v_{\mathrm{meet}} j - \triangle v_{\mathrm{meet}} \frac{1-\zeta^{j+1}}{1-\zeta}\right)^2\right] \\
&\quad +E\left[\left(\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{i-m-1}W_{m-1}^{A}\right.\right. \\
&\quad \left.\left. -\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{i-m-1}W_{m-1}^{B}\right)^2\right] \\
&= 2\sigma_\alpha^2\left[j+1-\frac{1-\zeta^{2j+2}}{1-\zeta^2}\right] + \left(j-\frac{1-\zeta^{j+1}}{1-\zeta}\right)^2 E\left[\left(\triangle v_{\mathrm{meet}}\right)^2\right].
\end{aligned}$$

Figures A.6 and A.7 compare the $\mathrm{MSE}[\widehat{\mathrm{S(j)}}]$ and $\mathrm{MSE}_{\mathrm{baseline}}[\widehat{\mathrm{S(j)}}]$ for $\zeta = 0.7$ and

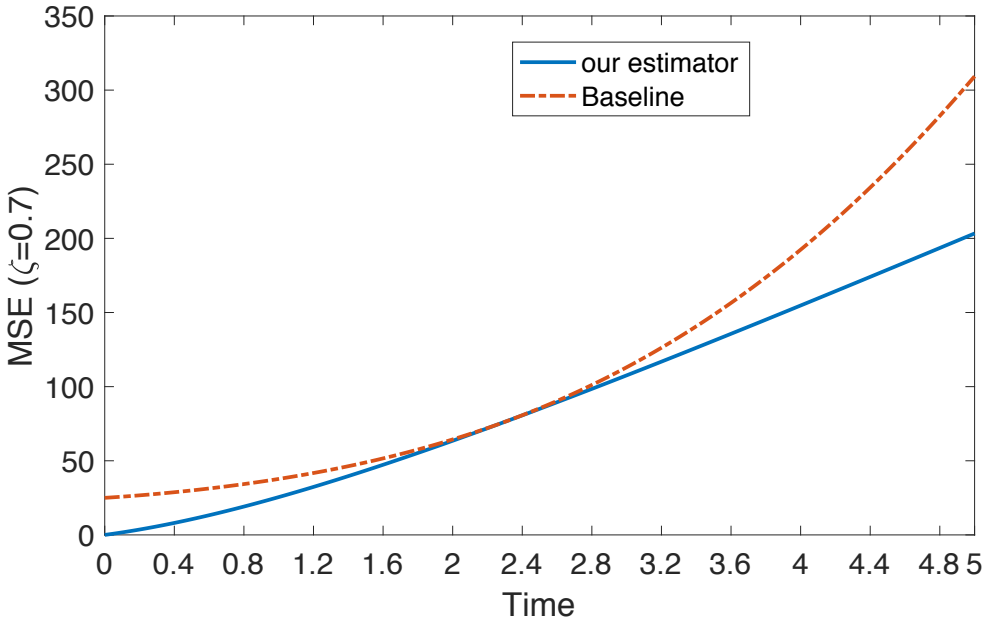

Figure A.6: $\widehat{\text{MSE}[\text{S(j)}]}$ and $\widehat{\text{MSE}_{\text{baseline}}[\text{S(j)}]}$ for $\zeta = 0.7$

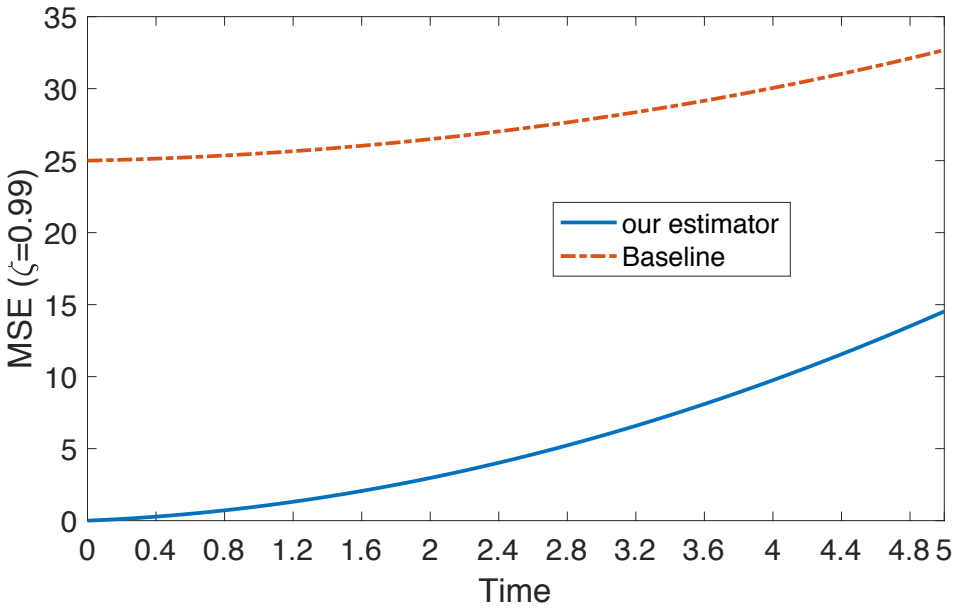

Figure A.7: $\widehat{\text{MSE}[\text{S(j)}]}$ and $\widehat{\text{MSE}_{\text{baseline}}[\text{S(j)}]}$ for $\zeta = 0.99$

$\zeta = 0.99$ respectively.

The probability that the actual distance is more than the asymptotic distance is

$$\begin{aligned}
&P\left(S(j) \geq \frac{\triangle v_{\text{meet}}}{1-\zeta}\right) \\
&= P\left(\sum_{i=0}^{j} \triangle v_i \geq \frac{\triangle v_{\text{meet}}}{1-\zeta}\right) \\
&= P\left(\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\zeta^{i-1}\left(\sum_{m=0}^{i-1}\zeta^{-m}W_{m-1}^{A} - \sum_{m=0}^{i-1}\zeta^{-m}W_{m-1}^{B}\right) \geq \frac{\triangle v_{\text{meet}}\zeta^{j+1}}{(1-\zeta)}\right).
\end{aligned}$$

Defining

$$L(i) = \sum_{m=0}^{i-1}\zeta^{-m}W_{m-1}^{A} - \sum_{m=0}^{i-1}\zeta^{-m}W_{m-1}^{B}$$

and

$$M(j) = \sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\zeta^{i-1}L(i),$$

we see that

$$L(i) \sim \mathcal{N}(0, \frac{2\left(1-\zeta^{-2i}\right)}{1-\zeta^{-2}}) \tag{A.4}$$

and

$$M(j) \sim \mathcal{N}\left(0, 2\sigma_\alpha^2\left[j+1-\frac{1-\zeta^{2j+2}}{1-\zeta^2}\right]\right). \tag{A.5}$$

This implies

$$P\left(S(j) \geq \frac{\triangle v_{\text{meet}}}{1-\zeta}\right) = Q\left(\frac{\frac{\triangle v_{\text{meet}}\zeta^{j+1}}{(1-\zeta)}}{\sqrt{2\sigma_\alpha^2\left[j+1-\frac{1-\zeta^{2j+2}}{1-\zeta^2}\right]}}\right).$$

Scenario 2: Assume that cars have different target speed but the target speed never changes and is known, the estimation of distance between cars at time j is

$$\widehat{S(j)} = \sum_{i=0}^{j}\widehat{\triangle v}_i = \triangle v_{\text{meet}}\frac{1-\zeta^{j+1}}{1-\zeta} + \triangle v_\alpha\left(\frac{\zeta^{j+1}-\zeta}{1-\zeta}\right).$$

The mean squared error in the estimation process will be

$$\begin{aligned}
\mathrm{MSE}[\widehat{\mathrm{S(j)}}] &= E\left[\left(\triangle v_{\mathrm{meet}}\frac{1-\zeta^{j+1}}{1-\zeta}+\triangle v_\alpha\left(\frac{\zeta^{j+1}-\zeta}{1-\zeta}\right)-\sum_{i=0}^{j}\triangle v_i\right)^2\right] \\
&= E\left[\left(\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{i-m-1}W_{m-1}^{A}\right.\right. \\
&\quad \left.\left.-\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{i-m-1}W_{m-1}^{B}\right)^2\right],
\end{aligned}$$

which can be further simplified as

$$\begin{aligned}
\mathrm{MSE}[\widehat{\mathrm{S(j)}}] &= 2\sigma_\alpha^2\left(1-\zeta^2\right)\left[\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{2(i-m-1)}\right]=2\sigma_\alpha^2\sum_{i=0}^{j}(1-\zeta^{2i}) \\
&= 2\sigma_\alpha^2\left[j+1-\frac{1-\zeta^{2j+2}}{1-\zeta^2}\right]. \qquad \text{(A.6)}
\end{aligned}$$

The mean squared error (A.6) can increase unboundedly and is not a function of the $\triangle v_{\mathrm{meet}}$. The MSE for the baseline estimator will be

$$\begin{aligned}
&\mathrm{MSE}_{\mathrm{baseline}}[\widehat{\mathrm{S(j)}}] \\
&= E\left[\left(\widehat{S(j)}-S(j)\right)^2\right] \\
&= E\left[\left(\triangle v_{\mathrm{meet}}j-\sum_{i=0}^{j}\triangle v_i\right)^2\right] \\
&= E\left[\left(\triangle v_{\mathrm{meet}}j-\triangle v_{\mathrm{meet}}\frac{1-\zeta^{j+1}}{1-\zeta}\right)^2\right]+E\left[\left(\triangle v_\alpha\left(\frac{\zeta^{j+1}-\zeta}{1-\zeta}\right)\right)^2\right] \\
&+ E\left[\left(\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{i-m-1}W_{m-1}^{A}\right.\right. \\
&\left.\left.-\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{i=0}^{j}\sum_{m=0}^{i-1}\zeta^{i-m-1}W_{m-1}^{B}\right)^2\right] \\
&= 2\sigma_\alpha^2\left[j+1-\frac{1-\zeta^{2j+2}}{1-\zeta^2}\right]+\left(j-\frac{1-\zeta^{j+1}}{1-\zeta}\right)^2 E\left[\triangle v_{\mathrm{meet}}^2\right] \\
&+ \left(\triangle v_\alpha\left(\frac{\zeta^{j+1}-\zeta}{1-\zeta}\right)\right)^2.
\end{aligned}$$

Figures A.8 and A.9 compare the $\mathrm{MSE}[\widehat{\mathrm{S(j)}}]$ and $\mathrm{MSE}_{\mathrm{baseline}}[\widehat{\mathrm{S(j)}}]$ for $\zeta = 0.7$ and $\zeta =$

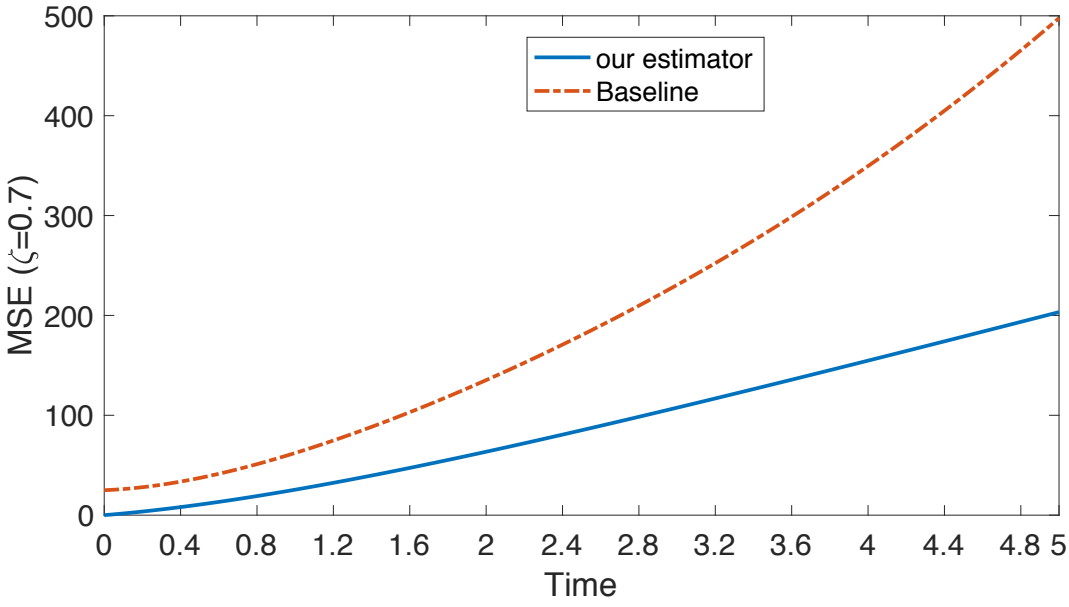

Figure A.8: MSE$[\widehat{S(j)}]$ and MSE$_{\text{baseline}}[\widehat{S(j)}]$ for $\zeta = 0.7$ and target speed difference is 10 m/s

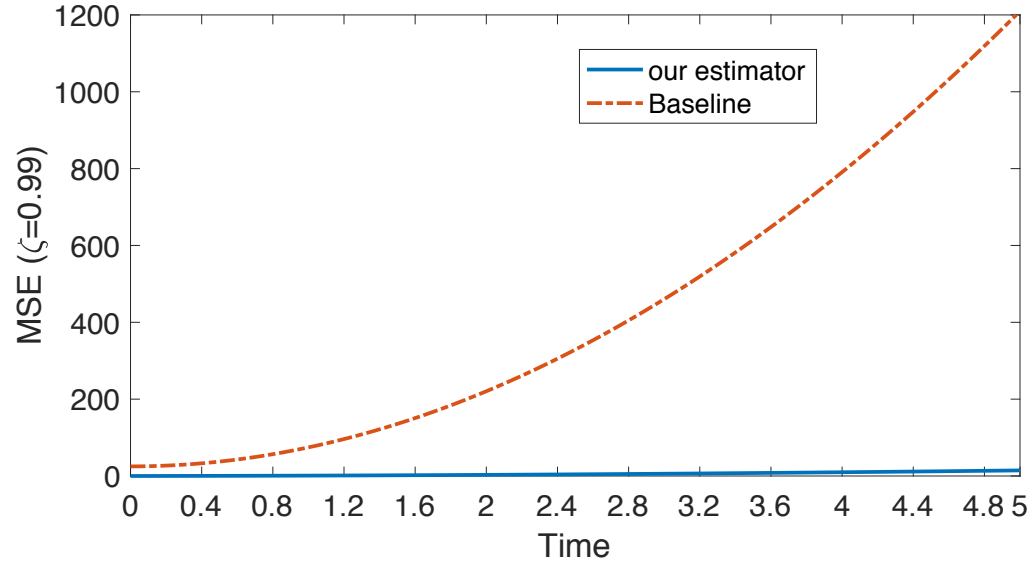

Figure A.9: MSE$[\widehat{S(j)}]$ and MSE$_{\text{baseline}}[\widehat{S(j)}]$ for $\zeta = 0.99$ and target speed difference is 10 m/s

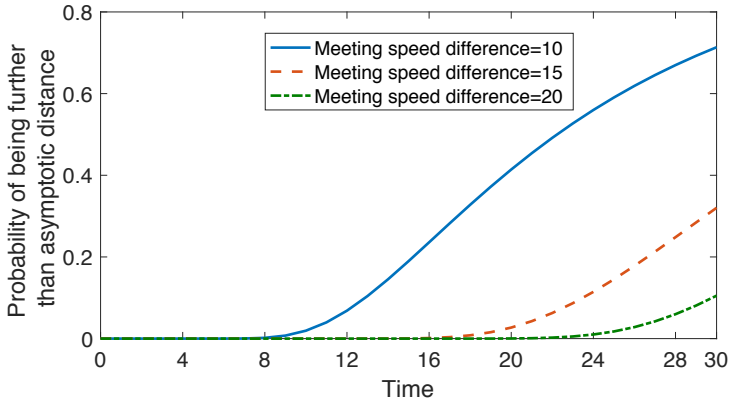

Figure A.10: the plot for $P\left(S(j) \geq \frac{\triangle v_{\text{meet}}}{1-\zeta} + \triangle v_\alpha \left(\frac{-\zeta}{1-\zeta}\right)\right)$ when cars have target speed difference of 10 m/s and $\zeta = 0.95$

0.99 respectively. The probability that the actual distance is more than the asymptotic distance is

$$
\begin{aligned}
& P\left(S(j) \geq \frac{\triangle v_{\text{meet}}}{1-\zeta} + \triangle v_\alpha \left(\frac{-\zeta}{1-\zeta}\right)\right) \\
&= P\left(\sum_{i=0}^{j} \triangle v_i \geq \frac{\triangle v_{\text{meet}}}{1-\zeta} - \triangle v_\alpha \frac{\zeta}{1-\zeta}\right) \\
&= P\left(\frac{\triangle v_{\text{meet}}}{1-\zeta} - \triangle v_\alpha \frac{\zeta}{1-\zeta} \leq \triangle v_{\text{meet}} \frac{1-\zeta^{j+1}}{1-\zeta} + \triangle v_\alpha \left(\frac{\zeta^{j+1}-\zeta}{1-\zeta}\right)\right. \\
&\left. + \sigma_{v_\alpha} \sqrt{1-\zeta^2} \left(\sum_{i=0}^{j} \sum_{m=0}^{i-1} \zeta^{i-m-1} W_{m-1}^{A} - \sum_{i=0}^{j} \sum_{m=0}^{i-1} \zeta^{i-m-1} W_{m-1}^{B}\right)\right)
\end{aligned}
$$

Based on (A.4) and (A.5),

$$
P\left(S(j) \geq \frac{\triangle v_{\text{meet}}}{1-\zeta} + \triangle v_\alpha \left(\frac{-\zeta}{1-\zeta}\right)\right) = Q\left(\frac{\frac{(\triangle v_{\text{meet}} - \triangle v_\alpha)\zeta^{j+1}}{(1-\zeta)}}{\sqrt{2\sigma_\alpha^2 \left[j+1-\frac{1-\zeta^{2j+2}}{1-\zeta^2}\right]}}\right)
$$

which is illustrated in Figure A.10.

Scenario 3: In the most general case, vehicles have different target speed which is unknown. Given $\mathbf{V}(L) = [v_L, v_{L-1}, \cdots, v_0]^T$ where L is a function of ζ, cars estimate v_α using $\hat{v}_\alpha = S_{(L+1)\times 1}^T \mathbf{V}(L) + d$ where S, and d are random variables. They exchange their

estimated target speed and estimate the distance based on the previous two scenarios.

A.4 Design of the New Link Stability Metric

Most of the current link stability metrics make the simplifying assumptions that i) vehicles move with constant speed, ii) when two cars meet, the contact event starts with one car entering the coverage area of the other car and leave from the other side of the coverage area. It does not consider other relative mobility events between vehicles. The probability that link lifetime, namely Life, is j seconds is defined as

$$P[\text{Life}] = P(D_{\text{cov}} \leq S(j)) + P(-D_{\text{cov}} \geq S(j)) \tag{A.7}$$

where

$$\begin{aligned}
P\left[D_{\text{cov}} \leq S(j)\right] &= P\left[2D_{\text{cov}} \leq \sum_{i=0}^{j} \triangle v_i\right] \\
&= P\left[2D_{\text{cov}} \leq \sum_{i=0}^{j} \left(\zeta^i \triangle v_0 + \left(1-\zeta^i\right)\triangle v_\alpha \right.\right. \\
&\quad \left.\left. + \sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{m=0}^{i-1}\zeta^{i-m-1}\triangle W_m\right)\right] \\
&= P\left[2D_{\text{cov}} \leq \triangle v_{\text{meet}}\frac{1-\zeta^{j+1}}{1-\zeta} + \triangle v_\alpha\left(\frac{\zeta^{j+1}-\zeta}{1-\zeta}\right)\right. \\
&\quad \left. + \sum_{i=0}^{j}\left(\sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{m=0}^{i-1}\zeta^{i-m-1}\triangle W_{m-1}\right)\right]
\end{aligned}$$

and

$$\begin{aligned}
&P\left[D_{\text{cov}} \leq S(j) | \triangle v_{\text{meet}}, \triangle v_\alpha\right] \\
&= P\left[2D_{\text{cov}} \leq \sum_{i=0}^{j}\left(\zeta^i\triangle v_0 + \left(1-\zeta^i\right)\triangle v_\alpha\right.\right. \\
&\quad \left.\left. + \sigma_{v_\alpha}\sqrt{1-\zeta^2}\sum_{m=0}^{i-1}\zeta^{i-m-1}\triangle W_{m-1}\right) | \triangle v_{\text{meet}}, \triangle v_\alpha\right] \\
&= P\left[2D_{\text{cov}} \leq \triangle v_{\text{meet}}\frac{1-\zeta^{j+1}}{1-\zeta} + \triangle v_\alpha\left(\frac{\zeta^{j+1}-\zeta}{1-\zeta}\right)\right. \\
&\quad \left. + \sum_{i=0}^{j}\left(\sigma_{v_\alpha}\sqrt{1-\zeta^2}\zeta^{i-1}\sum_{m=0}^{i-1}\zeta^{-m}\triangle W_{m-1}\right) | \triangle v_{\text{meet}}, \triangle v_\alpha\right].
\end{aligned}$$

Defining

$$L(j) = \sum_{i=0}^{j} \left(\sigma_{v_\alpha} \sqrt{1-\zeta^2} \zeta^{i-1} \sum_{m=0}^{i-1} \zeta^{-m} \triangle W_{m-1} \right),$$

since it is sum of correlated random variables, we re-write this sum as

$$L(j) = \sigma_{v_\alpha} \sqrt{1-\zeta^2} \sum_{m=0}^{j-1} \left(\triangle W_{m-1} \sum_{i=0}^{j-1-m} \zeta^i \right).$$

Since

$$W(m) = \triangle W_{m-1} \sum_{i=0}^{j-1-m} \zeta^i = \triangle W_{m-1} \frac{1-\zeta^{j-m}}{1-\zeta},$$

we see that

$$W(m) \sim \mathcal{N}(0, 2 \left(\frac{1-\zeta^{j-m}}{1-\zeta} \right)^2)$$

and

$$M(j) \sim \mathcal{N} \left(0, 2\sigma_\alpha^2 (1-\zeta^2) \sum_{m=0}^{j-1} \left(\frac{1-\zeta^{j-m}}{1-\zeta} \right)^2 \right).$$

This implies

$$P\left[D_{\text{cov}} \leq S(j) | \triangle v_{\text{meet}}, \triangle v_\alpha \right] = Q \left(\frac{2D_{\text{cov}} - \triangle v_{\text{meet}} \frac{1-\zeta^{j+1}}{1-\zeta} - \triangle v_\alpha \left(\frac{\zeta^{j+1}-\zeta}{1-\zeta} \right)}{\sqrt{2\sigma_\alpha^2 (1-\zeta^2) \sum_{m=0}^{j-1} \left(\frac{1-\zeta^{j-m}}{1-\zeta} \right)^2}} \right)$$

and similarly since $Q(x) = 1 - Q(-x)$,

$$\begin{aligned} P\left[-D_{\text{cov}} \geq S(j) | \triangle v_{\text{meet}}, \triangle v_\alpha \right] &= 1 - Q \left(\frac{-\triangle v_{\text{meet}} \frac{1-\zeta^{j+1}}{1-\zeta} - \triangle v_\alpha \left(\frac{\zeta^{j+1}-\zeta}{1-\zeta} \right)}{\sqrt{2\sigma_\alpha^2 (1-\zeta^2) \sum_{m=0}^{j-1} \left(\frac{1-\zeta^{j-m}}{1-\zeta} \right)^2}} \right) \\ &= Q \left(\frac{\triangle v_{\text{meet}} \frac{1-\zeta^{j+1}}{1-\zeta} + \triangle v_\alpha \left(\frac{\zeta^{j+1}-\zeta}{1-\zeta} \right)}{\sqrt{2\sigma_\alpha^2 (1-\zeta^2) \sum_{m=0}^{j-1} \left(\frac{1-\zeta^{j-m}}{1-\zeta} \right)^2}} \right). \end{aligned}$$

Finally, the CDF for link lifetime is

$$\begin{aligned} P\left[\text{Life} \leq \tau \right] &= P\left[\cup_{t=1}^{\tau} \left\{ S(t) < -D_{cov} \right\} \cup \left\{ S(t) > D_{cov} \right\} \right] \\ &\leq \sum_{t=1}^{\tau} P\left[\left\{ S(t) < -D_{cov} \right\} \cup \left\{ S(t) > D_{cov} \right\} \right]. \end{aligned} \tag{A.8}$$

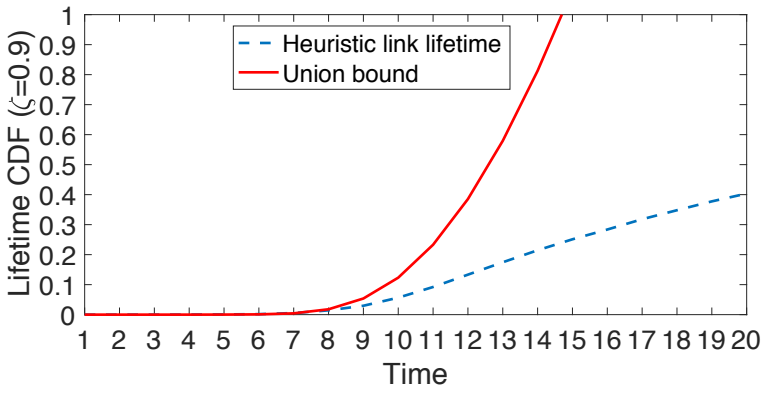

Figure A.11: Link lifetime CDF computed using the proposed union bound and heuristically when the meeting speed difference is 15 m/s

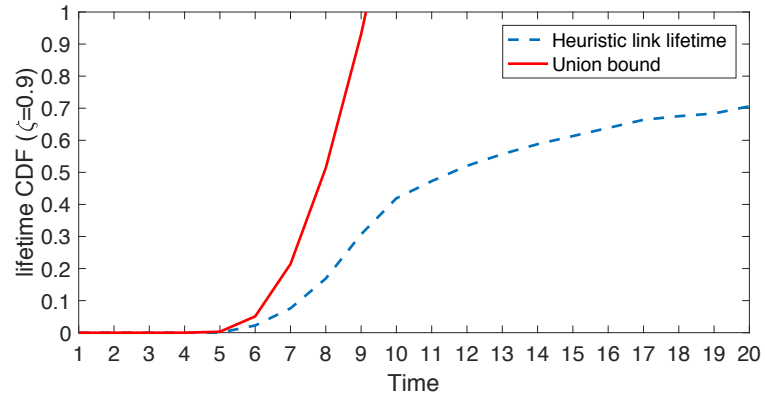

Figure A.12: Link lifetime CDF computed using the proposed union bound and heuristically when the meeting speed difference is 25 m/s

Figures A.11 and A.12 illustrate the upper bound on link lifetime CDF computed using the proposed union bound in (A.8) and compare the results with precise CDF computed heuristically. These plots belong to the links that the meeting speed difference between two cars is 15 m/s and 25 m/s respectively. The estimations using (A.8) are very close to the results of heuristic CDF for the first 6-8 seconds. This indicates the benefit of using the proposed algorithm to determine the link lifetime estimate. Vehicles can periodically repeat the estimation process using their most recent L samples to ensure the correctness of their link lifetime estimates.

Appendix B

Markov Chain Analysis

The X_k process evolves according to the following rules:

- If $X_k = 0$, then
  - If $Y_{k+1} \leq \tau$, then $X_{k+1} = 0$; otherwise, $X_{k+1} = 1$.
- If $X_k = j > 0$, then
  - If $Y_{k+1} \leq \tau$, then $X_{k+1} = 0$.
  - If $\tau < Y_{k+1} \leq T + \tau$, then $X_{k+1} = 1$.
  - If $Y_{k+1} > \tau$ and $Y_k > T + \tau$, then $X_{k+1} = j + 1$.

For $\tau \leq T$, we now build a Markov chain from these rules. First we observe that for all $j \geq 0$,

$$\mathrm{P}[X_{k+1} = 0 | X_k = j] = \mathrm{P}[Y_{k+1} \leq \tau] = F_Y(\tau). \tag{B.1}$$

Of course, this implies $\mathrm{P}[X_{k+1} = 1 | X_k = 0] = \overline{F}_Y(\tau)$. Furthermore, for $j \geq 1$,

$$\begin{aligned} &\mathrm{P}[X_{k+1} = j + 1] \\ &\quad = \mathrm{P}[X_k = j, Y_k > T + \tau, Y_{k+1} > \tau]. \end{aligned} \tag{B.2}$$

Note that $X_k = j \geq 1$ implies $Y_k > \tau$, and thus

$$\begin{aligned} &\mathrm{P}[X_{k+1} = j + 1 | X_k = j] \\ &= \mathrm{P}[Y_k > T + \tau, Y_{k+1} > \tau | X_k = j] \\ &= \mathrm{P}[Y_k > T + \tau | X_k = j] \\ &\qquad \times \mathrm{P}[Y_{k+1} > \tau | Y_k > T + \tau, X_k = j] \end{aligned}$$

$$
\begin{aligned}
&= \mathrm{P}[Y_k > T + \tau | X_k = j, Y_k > \tau] \, \mathrm{P}[Y_{k+1} > \tau] \\
&= \mathrm{P}[Y_k > T + \tau | Y_k > \tau] \, \mathrm{P}[Y_{k+1} > \tau] \\
&= \frac{\mathrm{P}[Y_k > T + \tau]}{\mathrm{P}[Y_k > \tau]} \, \mathrm{P}[Y_{k+1} > \tau] \qquad \text{(B.3)} \\
&= \overline{F}_Y(T + \tau), \qquad \text{(B.4)}
\end{aligned}
$$

where (B.4) follows from (B.3) because the Y_k are iid. With similar logic, we observe for $j \geq 1$ that

$$
\begin{aligned}
&\mathrm{P}[X_{k+1} = 1 | X_k = j] \\
&\quad = \mathrm{P}[Y_k \leq T + \tau, Y_{k+1} > \tau | X_k = j] \\
&\quad = \mathrm{P}[Y_k \leq T + \tau | X_k = j] \\
&\qquad \times \mathrm{P}[Y_{k+1} > \tau | Y_k > T + \tau, X_k = j] \\
&\quad = \mathrm{P}[Y_k \leq T + \tau | Y_k > \tau] \, \mathrm{P}[Y_{k+1} > \tau] \\
&\quad = \mathrm{P}[\tau < Y_k \leq T] = F_Y(T + \tau) - F_Y(\tau). \qquad \text{(B.5)}
\end{aligned}
$$

We observe that (B.1), (B.4) and (B.5) verify that X_k is described by the discrete-time Markov chain shown in Figure 3.4.

References

[1] U. Lee, R. Cheung, and M. Gerla, "Emerging vehicular applications," *Vehicular Networks: From Theory to Practice. Chapman and Hall/CRC*, 2009.

[2] N. Wang and J. Wu, "Opportunistic WiFi offloading in a vehicular environment: Waiting or downloading now?" in *IEEE International Conference on Computer Communications*, 2016.

[3] S. Panichpapiboon and W. Pattara-Atikom, "A review of information dissemination protocols for vehicular ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 3, pp. 784–798, 2012.

[4] L. Le *et al.*, "Infrastructure-assisted communication for car-to-x communication," in *Proceedings of the ITS world congress*, 2011.

[5] J. Wang, R. Wakikawa, and L. Zhang, "DMND: Collecting data from mobiles using named data," in *IEEE Vehicular Networking Conference*, 2010, pp. 49–56.

[6] B. Hassanabadi, C. Shea, L. Zhang, and S. Valaee, "Clustering in vehicular ad hoc networks using affinity propagation," *Ad Hoc Networks*, vol. 13, pp. 535–548, 2014.

[7] J. Y. Yu and P. H. J. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 1, pp. 32–48, 2005.

[8] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future Internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[9] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida, "Navigo: Interest forwarding by geolocations in vehicular Named Data Networking," in *IEEE WoWMoM*, 2015, pp. 1–10.

[10] V. Jacobson *et al.*, "Networking named content," in *International conference on Emerging networking experiments and technologies*, 2009.

[11] P. TalebiFard, V. C. Leung, M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for VANETs," in *Vehicular ad hoc Networks*. Springer, 2015, pp. 503–524.

[12] Y.-T. Yu, M. Gerla, and M. Sanadidi, "Scalable VANET content routing using hierarchical bloom filters," *Wireless Communications and Mobile Computing*, vol. 15, no. 6, pp. 1001–1014, 2015.

[13] T. Vu *et al.*, "DMAP: A shared hosting scheme for dynamic identifier to locator mappings in the global Internet," in *IEEE International Conference on Distributed Computing Systems*, 2012, pp. 698–707.

[14] P. M. Esposito, M. E. M. Campista, I. M. Moraes, L. H. M. Costa, O. C. M. Duarte, and M. G. Rubinstein, "Implementing the expected transmission time metric for olsr wireless mesh networks," in *Wireless Days, 2008. WD'08. 1st IFIP*. IEEE, 2008, pp. 1–5.

[15] M. Tavan, R. D. Yates, and D. Raychaudhuri, "Connected vehicles under named object architectures," in *Proceedings of the First ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*. ACM, 2016, pp. 60–61.

[16] (2016) Cisco Visual Networking Index: Forecast and Methodology, 2015–2020. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

[17] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 480–495, 2014.

[18] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Proceedings of the 11th annual workshop on network and systems support for games*. IEEE Press, 2012, p. 2.

[19] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, ""gaminganywhere: an open cloud gaming system"," in *Proceedings of the 4th ACM multimedia systems conference*. ACM, 2013, pp. 36–47.

[20] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "A hybrid edge-cloud architecture for reducing on-demand gaming latency," *Multimedia Systems*, vol. 20, no. 5, pp. 503–519, 2014.

[21] Y.-C. Chang, P.-H. Tseng, K.-T. Chen, and C.-L. Lei, "Understanding the performance of thin-client gaming," in *IEEE International Workshop on Communications Quality and Reliability (CQR)*. IEEE, 2011, pp. 1–6.

[22] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen, "Enabling adaptive high-frame-rate video streaming in mobile cloud gaming applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 1988–2001, 2015.

[23] T. Stockhammer, "Dynamic adaptive streaming over http–: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

[24] R. Shea, J. Liu, E. C. H. Ngai, and Y. Cui, "Cloud gaming: architecture and performance," *IEEE Network*, vol. 27, no. 4, pp. 16–21, July 2013.

[25] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE INFOCOM*, 2012.

[26] M. Gerla and L. Kleinrock, "Vehicular networks and the future of the mobile Internet," *Computer Networks*, vol. 55, no. 2, pp. 457–469, 2011.

[27] A. Baid *et al.*, "Enabling vehicular networking in the mobilityfirst future Internet architecture," in *IEEE WoWMoM*. IEEE, 2013, pp. 1–3.

[28] H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen, "Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 8, pp. 4628–4639, 2009.

[29] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host identity protocol," Tech. Rep., 2008.

[30] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, "The locator/ID separation protocol (LISP)," 2013.

[31] P. V. Mockapetris, "Domain names-concepts and facilities," 1987.

[32] M. Tavan, R. D. Yates, and D. Raychaudhuri, "Connected vehicles under information-centric architectures," in *IEEE Vehicular Networking Conference (VNC)*. IEEE, 2016, pp. 1–8.

[33] P. Basu, N. Khan, and T. D. Little, "A mobility based metric for clustering in mobile ad hoc networks," in *International conference on Distributed computing systems workshop*. IEEE, 2001, pp. 413–418.

[34] Y. Li, M. Zhao, and W. Wang, "Intermittently connected vehicle-to-vehicle networks: detection and analysis," in *IEEE Globecom*, 2011.

[35] H. A. Omar, W. Zhuang, and L. Li, "Gateway placement and packet routing for multihop in-vehicle Internet access," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 3, pp. 335–351, 2015.

[36] Y. Hu, R. D. Yates, and D. Raychaudhuri, "A hierarchically aggregated in-network global name resolution service for the mobile internet."

[37] B. Liang and Z. J. Haas, "Predictive distance-based mobility management for multidimensional PCS networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 718–732, 2003.

[38] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)-an open-source traffic simulation," in *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, 2002, pp. 183–187.

[39] M. Claypool and K. Claypool, "Latency and player actions in online games," *Communications of the ACM*, vol. 49, no. 11, pp. 40–45, 2006.

[40] K.-C. Yang, C. C. Guest, K. El-Maleh, and P. K. Das, "Perceptual temporal quality metric for compressed video," *IEEE Transactions on Multimedia*, vol. 9, no. 7, pp. 1528–1535, 2007.

[41] H. Nam, K.-H. Kim, and H. Schulzrinne, "QoE Matters More Than QoS: Why People Stop Watching Cat Videos," *IEEE International Conference on Computer Communications*, 2016.

[42] Q. Huynh-Thu and M. Ghanbari, "Impact of jitter and jerkiness on perceived video quality," in *Proc. Workshop on Video Processing and Quality Metrics*, 2006.

[43] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 325–338, 2015.

[44] J. Wu, C. Yuen, N. M. Cheung, J. Chen, and C. W. Chen, "Modeling and optimization of high frame rate video transmission over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2713–2726, April 2016.

[45] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming," in *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2017.

[46] R. D. Yates and S. Kaul, "Real-time status updating: Multiple sources," in *IEEE International Symposium on Information Theory*. IEEE, 2012, pp. 2666–2670.

[47] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *2015 IEEE International Symposium on Information Theory*. IEEE, 2015, pp. 3008–3012.

[48] C. Savery, T. Graham, and C. Gutwin, "The human factors of consistency maintenance in multiplayer computer games," in *Proceedings of the 16th ACM international conference on Supporting group work*. ACM, 2010, pp. 187–196.

[49] C. Savery, N. Graham, C. Gutwin, and M. Brown, "The effects of consistency maintenance methods on player experience and performance in networked games," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 2014, pp. 1344–1355.

[50] M. Claypool, "The effect of latency on user performance in real-time strategy games," *Elsevier Computer Networks*, vol. 49, no. 1, Sep. 2005.

[51] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1897–1910, April 2016.

[52] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Transactions on Antennas and Propagation*, 2017.

[53] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. C. Leung, and C.-H. Hsu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.

[54] P. E. Ross, "Cloud computing's killer app: Gaming," *IEEE Spectrum*, vol. 46, no. 3, pp. 14–14, 2009.

[55] "Gaikai," https://www.gaikai.com.

[56] "Psnow," https://www.playstation.com/en-us/explore/playstationnow/.

[57] M. Claypool, D. Finkel, A. Grant, and M. Solano, "Thin to win?: network performance analysis of the onlive thin client game system," in *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*. IEEE Press, 2012, p. 1.

[58] W. Cai, M. Chen, and V. C. Leung, "Toward gaming as a service," *IEEE Internet Computing*, vol. 18, no. 3, pp. 12–18, 2014.

[59] D. Mishra, M. El Zarki, A. Erbad, C.-H. Hsu, and N. Venkatasubramanian, "Clouds+ games: A multifaceted approach," *IEEE Internet Computing*, vol. 18, no. 3, pp. 20–27, 2014.

[60] K. Lee *et al.*, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2015, pp. 151–165.

[61] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[62] Y.-T. Lee, K.-T. Chen, H.-I. Su, and C.-L. Lei, "Are all games equally cloud-gaming-friendly?: an electromyographic approach," in *Proceedings of the 11th annual workshop on network and systems support for games*. IEEE Press, 2012, p. 3.

[63] S. Borer, "A model of jerkiness for temporal impairments in video transmission," in *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*. IEEE, 2010, pp. 218–223.

[64] K.-T. Chen, P. Huang, G.-S. Wang, C.-Y. Huang, C.-L. Lei *et al.*, "On the sensitivity of online game playing time to network qos." in *INFOCOM*, 2006.

[65] M. Claypool and K. Claypool, "Perspectives, frame rates and resolutions: it's all in the game," in *Proceedings of the 4th International Conference on Foundations of Digital Games*. ACM, 2009, pp. 42–49.

[66] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 1269–1272.

[67] M. Manzano, J. A. Hernandez, M. Uruenna, and E. Calle, "An empirical study of cloud gaming," in *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*. IEEE Press, 2012, p. 17.

[68] M. Suznjevic, J. Beyer, L. Skorin-Kapov, S. Moller, and N. Sorsa, "Towards understanding the relationship between game type and network traffic for cloud gaming," in *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–6.

[69] M. Claypool and K. Claypool, "Latency can kill: precision and deadline in online games," in *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. ACM, 2010, pp. 215–222.

[70] R. Amin, F. Jackson, J. E. Gilbert, J. Martin, and T. Shaw, "Assessing the impact of latency and jitter on the perceived quality of call of duty modern warfare 2," in *International Conference on Human-Computer Interaction*. Springer, 2013, pp. 97–106.

[71] M. Claypool and D. Finkel, "The effects of latency on player performance in cloud-based games," in *Network and Systems Support for Games (NetGames), 2014 13th Annual Workshop on*. IEEE, 2014, pp. 1–6.

[72] S. Jarvinen, J.-P. Laulajainen, T. Sutinen, and S. Sallinen, "Qos-aware real-time video encoding how to improve the user experience of a gaming-on-demand service," in *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, vol. 2. IEEE, 2006, pp. 994–997.

[73] S. Kaul, R. Yates, and M. Gruteser, "Status updates through queues," in *Conf. on Inf. Sciences and Systems*, Mar. 2012.

[74] C. Kam, S. Kompella, and A. Ephremides, "Age of information under random updates," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, 2013, pp. 66–70.

[75] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of message transmission path diversity on status age," *IEEE Trans. Info Theory*, vol. 62, no. 3, pp. 1360–1374, March 2016.

[76] C. Kam, S. Kompella, and A. Ephremides, "Experimental evaluation of the age of information via emulation," in *Military Communications Conference, MILCOM 2015 - 2015 IEEE*, Oct 2015, pp. 1070–1075.

[77] C.-K. Toh, "Associativity-based routing for ad hoc mobile networks," *Wireless Personal Communications*, vol. 4, no. 2, pp. 103–139, 1997.

[78] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile computing*. Springer, 1996, pp. 153–181.

[79] C. Bettstetter, "Smooth is better than sharp: a random mobility model for simulation of wireless networks," in *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*. ACM, 2001, pp. 19–27.

[80] S. Krauss, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *Physical Review E*, vol. 55, no. 5, p. 5597, 1997.

[81] M. Zhao and W. Wang, "Wsn03-4: A novel semi-markov smooth mobility model for mobile ad hoc networks." in *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*. IEEE, 2006, pp. 1–5.