The Dissertation Committee for Yousuk Seung
certifies that this is the approved version of the following dissertation:

Optimizing Mobile Multimedia Content Delivery

Committee:

Yin Zhang, Supervisor

Lili Qiu

Li Erran Li

Aloysius K. Mok

Gordon S. Novak Jr.

Optimizing Mobile Multimedia Content Delivery

by

Yousuk Seung, B.S., M.S.C.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2013

Optimizing Mobile Multimedia Content Delivery

Yousuk Seung, Ph.D.
The University of Texas at Austin, 2013

Supervisor: Yin Zhang

With the advent of mobile Internet the amount of time people spend with multimedia applications in the mobile environment is surging and demand for high quality multimedia data over the Internet in the mobile environment is growing rapidly. However the mobile environment is significantly more unfriendly than the wired environment for multimedia applications in many ways. Network resources are limited and the condition is harder to predict. Also multimedia applications are generally delay intolerant and bandwidth demanding, and with users moving, their demand could be much more dynamic and harder to anticipate. Due to such reasons many existing mobile multimedia applications show unsatisfactory performance in the mobile environment.

We target three multimedia content delivery applications and optimize with limited and unpredictable network conditions typical in the mobile Internet environment.

Vehicular networks have emerged from the strong desire to communicate on the move. We explore the potential of supporting high-bandwidth

applications such as video streaming in vehicular networks. Challenges include limited and expensive cellular network, etc.

Internet video conferencing has become popular over the past few years, but supporting high-quality large video conferences at a low cost remains a significant challenge due to stringent performance requirements, limited and heterogeneous client. We develop a simple yet effective Valiant multicast routing to select application-layer routes and adapt streaming rates according to dynamically changing network condition in a swift and lightweight way enough to be implemented on mobile devices.

Bitrate adaptive video streaming is rapidly gaining popularity. However recent measurements show weaknesses in bitrate selection strategies implemented in today's streaming players especially in the mobile environment. We propose a novel rate adaptation scheme that classifies the network condition into stable and unstable periods and optimizes video quality with different strategies based on the classification.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The demand for online multimedia data has grown explosively in recent years. Online multimedia traffic already accounts for more than half of Internet traffic as of 2011 and this number grows to over 80% if all forms of video traffic is included such as P2P video sharing, Internet TV, and VoD[36]. A popular video streaming application Netflix has become the biggest source of Internet traffic in North America in 2011 [45], and a recent report shows that Netflix and YouTube are the two top applications that show the highest share of peak period downstream Internet traffic of nearly 50% combined in North America in 2012[107]. Two-way multimedia applications such as video conferencing are also gaining popularity. Video conferencing market is expected to grow to $10 billion by 2015 [59] or 6 times bewteen 2011 and 2016 [36]. Although still accounts for a relatively small fraction of overall Internet traffic compared to multimedia, video communication traffic has been experiencing high growth as well [35]. Also it has become more accessible to non-enterprise users due to wider availability of capable devices such as smart phones or tablets.

Internet is rapidly going mobile and demand for affordable and faster Internet connectivity while moving is increasing. Although still majority of Internet traffic is non-mobile, the growth of mobile Internet traffic is explosive. Global mobile data traffic in 2011 is already 8 times greater than the total global Internet traffic in 2000, and is expected to grow 13 to 16 times bewteen 2012 and 2017[37]. Reports suggest that mobile Internet environment is also rapidly improving to catch the rapid growth of demand [95]. And Multimedia data dominates the traffic again in mobile environment as in the non-mobile environment. A recent report shows that Multimedia data already accounts for majority of mobile Internet traffic and will continue to grow at the highest rate of any mobile application category by 2017 [37]. More powerful mobile devices and faster wireless connectivity can boost the demand for higher quality multimedia data such as high definition video which would further accelerate already fast growth of mobile multimedia traffic.

In this dissertation we discuss how we can design a system that delivers multimedia traffic effectively over the Internet, specifically in mobile environment. We also present real systems that implement a set of novel ideas. In doing so we make the following contributions.

First, we develop a novel vehicular content distribution (VCD) system for enabling high-bandwidth content distribution in vehicular networks. In particular we present a content replication strategy that effectively exploits the synergy among the Internet, local wireless connectivity, vehicle relay con-

nectivity, and mesh connectivity among access points(AP). Moreover, we optimize replication through wireline network and wireless mesh networks based on predicted mobility and traffic demands, and opportunistically exploit the mobility of the vehicles to extend the converence of the Internet and mesh connectivity. The key contributions of VCD include optimized wireline and mesh replication, opportunistic vehicular replication, a novel algorithm for mobility prediction, and thorough evaluation through simulation, emulation, and testbed experiments.

Second, we propose a novel Valiant multicast routing (VMR) to support video conferencing. It can efficiently utilize capacity from all nodes by balancing the load among them by randomly selecting nodes according to their upload capacity. Compared with optimization based approaches, the randomized algorithm in VMR incurs much lower overhead, and is faster to adapt and more robust to measurement errors. We also leverage resources from other peers or CDN nodes to enhance the performance and implement a proximity-aware extension of Valiant multicast routing to account for heterogeneous delay.

Third, we discuss a novel video rate adaptation scheme for mobile networks. Adaptive bitrate technology has become a trend in today's video streaming platforms. However recent measurements show that rate adaptation strategies in today's commercial players do not handle challenges posed by wireless mobile networks properly. We develop a simple algorithm that classfies the current network condition into stable or unstable periods, and design novel video rate adaptation schemes to optimize video quality based on

the classification. We further extend our scheme to support joint optimization of video quality across multiple flows and explore per symbol SNR information to improve the accuracy of prediction in mobile networks.

1.2 Challenges

- **Multimedia data is bandwidth heavy.** Popular video streaming applications such as Netflix, Microsoft Smooth Streaming, and Adobe OSMF typically require at least 250 $\sim$ 500 Kbps of bandwidth or as high as 3Mbps [3]. This poses significant challenges for applications that transmit multimedia data over the Internet, and more so in mobile environment due to limited bandwidth.

- **Multimedia data is delay sensitive.** Playback of video will be interrupted or frozen unless data is timely delivered, and such playback freezes are known to have the most negative impact on user-perceived playback quality among other factors such as rate changes or lower bitrate [43]. Two-way multimedia applications such as video conferencing can be particularly more delay sensitive due to their interactive nature [120].

- **Mobile network is resource limited.** Despite increasingly wider availability of faster mobile Internet connectivity such as 4G LTE the average mobile Internet connection speed is still limited and is yet to surpass 1Mbps [37]. Moreover consumers pay per-use at a much higher

rate in the mobile environment compared to the broadband where people typically pay per-day and at a lower rate. Therefore challenges is twofold in that applications are not only limited by lower bandwidth but also by higher cost in mobile environment.

- **Mobile network is unpredictable.** Content delivery over the Internet benefits from predictable and stable network environment. Video streaming applications adaptively change streaming bitrate and reliable bandwidth prediction is critical. Mobile Internet environment is affected by unpredictable external factors such as mobility and interference which pose significant challenges for content delivery.

1.3 Approach

We target three types of applications based on the aforementioned motivations. i) Multimedia content distribution system in vehicular network, ii) multi-party Internet video conferencing application and iii) rate-adaptive online video streaming application. We discuss novel ideas on how above challenges can be overcome and build real systems to demonstrate the ideas.

1.3.1 Multimedia Content Distribution in Vehicular Network

We develop a replication strategy that effectively exploits the synergy among (i) Internet connectivity, which is persistent but has limited coverage and relatively low bandwidth, (ii) local wireless connectivity, which has high bandwidth but short contact duration, (iii) vehicle relay connectivity, which

has high bandwidth but high delay, and (iv) mesh connectivity among APs, which is persistent and has high bandwidth but low coverage. In particular, we optimize replication through wireline network and wireless mesh networks based on predicted mobility and traffic demands. Moreover, we opportunistically exploit the mobility of the vehicles to extend the coverage of the Internet and mesh connectivity. Even if only a small fraction of APs have Internet and mesh connectivity, by having the vehicles themselves relay content, one can potentially replicate content to a much larger number of APs. In essence, vehicle mobility has the potential to significantly increase the network capacity [51] and reduce future content access delay. Note that many mobile devices, such as smartphones, support the use of cheap external storage cards, which can help mitigate potential storage concerns regarding carrying traffic for other users in the system [113].

To this end, we develop a novel **V**ehicular **C**ontent **D**istribution (VCD) system for enabling high-bandwidth content distribution in vehicular networks. VCD consists of vehicles, APs with and without Internet access (some of which may form a mesh network), content server on the Internet (*e.g.*, Web servers), and a controller. Vehicles submit location updates and content requests to the controller via cellular links. The controller optimizes the replication strategy based on predicted mobility and traffic demands, and instructs the APs to carry out the replication strategy. To enhance reliability and scalability, the controller can be replicated on multiple nodes. APs are deployed along road sides (*e.g.*, at gas stations and/or coffee shops) to allow vehicles on the road

to opportunistically communicate with them. The APs prefetch content as instructed by the controller. Whenever a vehicle encounters an AP, the AP tries to send the requested content from its local storage if the content is available locally. Otherwise, the AP tries to fetch the content from an AP in the same mesh network if one is available. If no such AP is found, it fetches content from the Internet when it has Internet connectivity. In addition to sending the content that the vehicle itself needs, the AP may also send the vehicle content that can then be relayed to other APs, or fetch from the vehicle content that can be served to other passing vehicles later.

1.3.2 Randomizing Routing in Multi-Party Internet Video Conferencing

We propose a novel Valiant multicast routing (VMR) to support video conferencing. The salient feature of VMR is that it can efficiently utilize capacity from all nodes by effectively balancing the load among them. It achieves this by randomly selecting nodes according to their upload capacity. To achieve better load balance, we leverage the power of two random choices – for each packet a source randomly picks two candidate relay nodes according to their upload capacity and uses the one with the lower load as the final relay node. Compared with optimization based approaches, the randomized algorithm in VMR incurs much lower overhead, and is faster to adapt and more robust to measurement errors.

The traditional "power of two choices" has been shown effective when

we know the load of all nodes completely accurately (*e.g.*, [13]), but its performance degrades significantly with decreasing accuracy of load information (*e.g.*, [85, 86, 40]). In fact, how to effectively leverage the power of two choices in face of stale information is recognized as an important open problem in [87]. We develop a novel technique to explicitly account for the uncertainty in the load information.

We further develop three extensions. First, nodes in the wide-area network may have large delay and randomly selecting relays may result in high end-to-end delay (*e.g.*, letting a sender in US to use a node in China to relay to a destination node in US). We cluster nodes according to their proximity, and adapt VMR to take into account both bandwidth and delay.

Second, VMR works well when there is sufficient total resource. When the resource is insufficient, balanced load would still incur network congestion and packet losses. Therefore, in addition to relaying traffic through subscribers, we also use non-subscribing peers from the same session, or peers in different sessions, or CDN nodes as relay nodes, and extend VMR to leverage different types of relay nodes.

Third, if the total capacity of all nodes (including peers from other sessions and CDNs) is still insufficient, we develop a distributed rate adaptation scheme to quickly adapt to the current network conditions.

1.3.3 Dynamic Rate Adaptation in Online Video Streaming

We first collect and analyze a range of wireless traces under different mobility. These traces exhibit two distinct patterns: stable periods when the network performance can be predicted with reasonable accuracy and unstable periods when the network performance is completely unpredictable. These characteristics prompt us to first automatically classify the current network condition into one of these two periods and then develop rate adaptation schemes tailored to one of these periods.

To classify the current network condition, we detect change points online based on the network traces seen so far, and then classify the current condition into stable and unstable periods based on the gap between recent change points. Next we design a simple yet effective video rate adaptation for stable periods to maximize the amount of time that the client can stay before completely filling/draining the buffer by effectively utilizing the predicted network performance. To handle unstable periods, we develop a novel two-rate download scheme, which first fills the buffer with the lowest video rate to prevent outage. To avoid being overly conservative and under-utilizing network resources, we replace the downloaded content with a higher rate starting from the end of the buffer. In this way, we can minimize the chance that video is blocked due to buffer empty while effectively utilizing the network bandwidth.

We further extend our approach to support multiple clients. In particular, to avoid clients blindly competing against each other, which leads to unstability, unfairness, and under-utilization, we develop an approach to

jointly optimize the video quality across all clients. This not only allows us to use the aggregate bandwidth prediction as input, which is more stable and easier to predict than the bandwidth of individual video flow under competition, but also allows different flows to more effectively share the common network resources.

Finally, we observe that the effectiveness of video rate adaptation depends on the lower layer design. In particular, we investgate the interaction between video rate adaptation and MAC rate adaptation. We propose an effective enhancement to the existing MAC rate adaptation by utilizing the signal-to-noise (SNR) information not only in the preamble but also across all the data symbols in the frame. Our results show that this enhancement leads to more accurate prediction of future channel condition and a better MAC rate adaptation based on it. The benefit of this enhanced MAC rate adaptation also directly translates to the improvement in the video quality.

Chapter 2

Multimedia Content Distribution in Vehicular Network

Vehicular networks have emerged from the strong desire to communicate on the move [16, 17, 46, 124]. Car manufacturers all over the world are developing industry standards and prototypes for vehicular networks (*e.g.*, [23, 28, 116]). Existing works on vehicular networks often focus on low-bandwidth applications, such as credit card payment, traffic condition monitoring [29], Web browsing [16, 17], and VoIP [17]. We explore how to support high-bandwidth applications (*e.g.*, video streaming) in vehicular networks.

Challenges and opportunities: Cellular networks, despite good coverage, still have limited bandwidth and incur high cost. For example, many cellular service providers in US, like AT&T, T-mobile, Sprint, Verizon, charge around $60 per month for 5GB data transfer and $0.2/MB afterwards [88]. 5GB data transfer can only support 0.1Mbps for 111 hours (< 5 days)! The cellular service price in many other countries are similar or even higher [126]. Moreover, many mobile broadband providers restrict or limit large data exchanges, including streaming audio, video, P2P file sharing, JPEG uploads, VoIP and automated feeds [88]. According to the international poll of 2700 De-

vicescape customers [101], 81% smartphone users prefer Wi-Fi over 3G cellular for data services. Therefore there is strong need for supporting high-bandwidth applications in vehicular networks using Wi-Fi.

A natural way is to let a vehicle download content from the Internet when it meets an access point (AP) [17, 46]. However, it is challenging to meet high bandwidth requirement since vehicles often move at a high speed and thus the contact time between vehicles and APs tends to be short (*e.g.*, [29] reported that 70% of connection opportunities are less than 10 seconds). In addition, it is often expensive to provide dense high-speed Internet coverage at a large scale. As a result, if vehicles fetch desired content on-demand from the Internet during their contact with an AP, the amount of data fetched may be insufficient to sustain the data rate required by applications such as video streaming when vehicles are outside the communication range of any APs.

With recent advances in wireless technology, Wi-Fi capacity has grown rapidly and can be at least an order of magnitude higher than typical Internet access link connectivity. For example, IEEE 802.11n can offer up to 600Mbps PHY data rate using 4 antennas. We performed a measurement experiment using a laptop equipped with NetGear WNDA3100 on a vehicle communicating with a NetGear WNDR3300 AP deployed near the road. We got 4.6Mbps using 802.11b, 22.2Mbps using 802.11g, and 39.7Mbps using 802.11n (2x2 MIMO) on 2.4GHz frequency, and 56.1Mbps using 802.11n on 5GHz. In comparison, DSL throughput ranges between 768Kbps to 6Mbps [11], which is an order of magnitude slower. The gap between the wireline and wireless capacity is

likely to increase further (*e.g.*, due to the availability of new spectrum, such as whitespace, and advances in antenna and signal processing technology). Such large gap suggests that in order to enjoy high wireless capacity, we should proactively replicate content beforehand to the APs that a vehicle is likely to visit. While the idea of replication is natural, *how to replicate the content given the limited wireline and wireless resources and uncertainty in vehicular trajectory* is an open research question that we aim to address.

Approach and contributions: We develop a replication strategy that effectively exploits the synergy among (i) Internet connectivity, which is persistent but has limited coverage and relatively low bandwidth, (ii) local wireless connectivity, which has high bandwidth but short contact duration, (iii) vehicle relay connectivity, which has high bandwidth but high delay, and (iv) mesh connectivity among APs, which is persistent and has high bandwidth but low coverage. In particular, we optimize replication through wireline network and wireless mesh networks based on predicted mobility and traffic demands. Moreover, we opportunistically exploit the mobility of the vehicles to extend the coverage of the Internet and mesh connectivity. Even if only a small fraction of APs have Internet and mesh connectivity, by having the vehicles themselves relay content, one can potentially replicate content to a much larger number of APs. In essence, vehicle mobility has the potential to significantly increase the network capacity [51] and reduce future content access delay. Note that many mobile devices, such as smartphones, support the use of cheap external storage cards, which can help mitigate potential storage

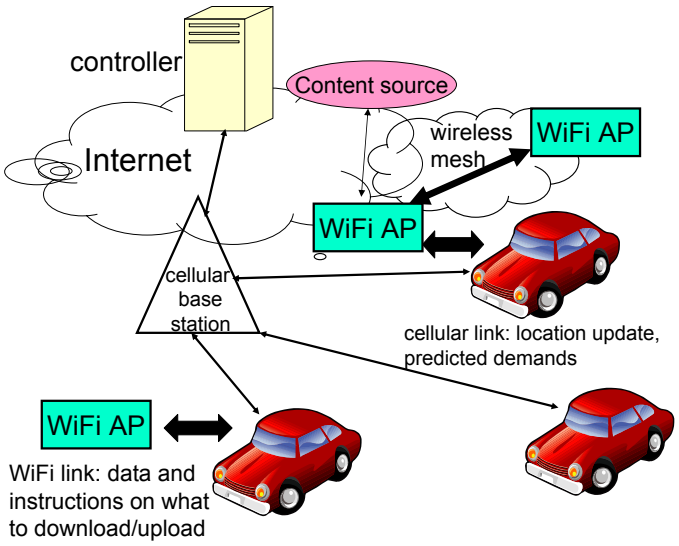


Figure 2.1: VCD architecture

concerns regarding carrying traffic for other users in the system [113].

To this end, we develop a novel **V**ehicular **C**ontent **D**istribution (VCD) system for enabling high-bandwidth content distribution in vehicular networks. As illustrated in Figure 2.1, VCD consists of vehicles, APs with and without Internet access (some of which may form a mesh network), content server on the Internet (*e.g.*, Web servers), and a controller. Vehicles submit location updates and content requests to the controller via cellular links. The controller optimizes the replication strategy based on predicted mobility and traffic demands, and instructs the APs to carry out the replication strategy. To enhance reliability and scalability, the controller can be replicated on multiple nodes. APs are deployed along road sides (*e.g.*, at gas stations and/or coffee shops)

to allow vehicles on the road to opportunistically communicate with them. The APs prefetch content as instructed by the controller. Whenever a vehicle encounters an AP, the AP tries to send the requested content from its local storage if the content is available locally. Otherwise, the AP tries to fetch the content from an AP in the same mesh network if one is available. If no such AP is found, it fetches content from the Internet when it has Internet connectivity. In addition to sending the content that the vehicle itself needs, the AP may also send the vehicle content that can then be relayed to other APs, or fetch from the vehicle content that can be served to other passing vehicles later.

VCD systems are easy to deploy in practical settings. For example, a vehicular service provider (VSP) can install its own APs and/or subscribe to existing wireless hotspot services. Since it is easier to place a stand-alone AP than hooking it up with Internet connection, VCD is designed to explicitly take advantage of APs with and without Internet connectivity. An AP without Internet connectivity is still useful since it can serve as a static cache, which vehicles can upload content that can be served to other passing vehicles in the future.

VSPs can offer content distribution service to taxis, buses, subways, and personal vehicles. We focus on taxis and buses that offer high-bandwidth content distribution as a value added service to their passengers. These vehicles have low-cost mobile devices on board for playing downloaded content. Such mobile devices can be installed by either the taxi/bus companies or VSPs.

Since the mobile devices can be powered by the vehicles, power consumption is not an issue. The mobile devices interact with APs and the VCD controller to report required information (*e.g.*, location update and predicted traffic demands) and follow their instructions.

The key contributions of VCD include:

- **Optimized wireline and mesh replication.** To fully take advantage of short contact time between APs and vehicles, we replicate content in advance to the APs that will soon be visited by the vehicle. A distinctive feature of our replication scheme is that it is based on optimization. Specifically, we explicitly formulate a linear program (LP) to optimize the amount of data that can be delivered before the deadline under the predicted mobility pattern and traffic demands subject to given resource constraints (*e.g.*, short contact time and limited link capacity). The formulation involves addressing challenging modeling issues and is a valuable contribution. In contrast, previous works either focus exclusively on protocol-level optimization of one-hop communication between vehicles and APs (*e.g.*, [17, 27, 29, 90]), or rely on heuristics to guide data replication [31], or completely ignore the resource constraints [42], which are crucial in vehicular networks. Our formulation is highly flexible and can support both wireline replication (Section 2.1.2) and mesh replication (Section 2.1.3). The formulation can be efficiently solved using standard LP solvers (*e.g.*, *lp_solve* [77] and *cplex* [39]) owing to modern interior-point linear programming methods.

- **Opportunistic vehicular replication.** To further extend the coverage of the Internet and wireless mesh networks, we develop vehicular replication to opportunistically take advantage of local wireless connectivity and vehicular relay connectivity (Section 2.1.4). Different from traditional vehicle-to-vehicle (v2v) communication, our scheme leverages the APs as the rendezvous points for replicating content among vehicles since vehicle-to-AP communication is easier to deploy and such contacts are generally easier to predict than v2v contacts.

- **A new algorithm for mobility prediction.** For our replication optimization algorithms to be effective, it is critical to predict the set of APs a vehicle will visit in a future interval with high accuracy. Given the high driving speeds, diverse and unpredictable road conditions, infrequent location updates, and irregular update intervals, accurately predicting mobility is challenging in vehicular networks. We develop a new mobility prediction algorithm based on the idea of *voting among K nearest trajectories (KNT)* (Section 2.2). We also implement several state-of-the-art mobility prediction algorithms based on Markov mobility models [110, 94]. Our evaluation in Section 2.4 shows that *KNT* achieves better prediction accuracy on our dataset.

- **Thorough evaluation through simulation, emulation, and testbed experiments.** We conduct trace-driven simulations to evaluate the performance of VCD using 500 taxis' traces in the San Francisco Bay Area

over the course of 30 days [26] and week-long traces of city bus in Seattle [108] (Section 2.5). Our results show that VCD is capable of downloading 3-6 times as much content as no replication, and 2-4 times as much content as wireline or vehicular replication alone; mesh replication further helps to improve throughput by up to 22%. The benefit of VCD further increases as the gap between wireless and wireline capacity enlarges and the AP density increases. In addition, we have developed a full-fledged prototype VCD system that supports real video streaming applications running on smartphones and laptops (Section 2.3, 2.6 and 2.7). We deploy our system in two wireless testbeds using 802.11b and 802.11n. Live road tests suggest that our system is capable of providing video streaming to smartphone and laptop clients at a vehicular speed. To further evaluate the performance of VCD at scale, we run the same AP and controller code as in the testbed together with emulated vehicles in the Emulab [44]. Our experiments show the efficiency of our implementation and that Emulab results closely follow the simulation results.

Chapter organization: The remainder of the chapter is organized as follows. We present our replication optimization techniques in Section 2.1 and our mobility prediction algorithm in Section 2.2. We specify our system design and implementation in Section 2.3. We evaluate the accuracy of mobility prediction in Section 2.4. We evaluate the performance of VCD through trace-driven simulation in Section 2.5, trace-driven emulation in Section 2.6, and testbed experiments in Section 2.7. We survey related work in Section 2.8.

2.1 Optimizing Replication

In this section, we first present an overview of our system, and then develop wireline, mesh, and vehicular replication.

2.1.1 Overview

At the beginning of every interval, the controller (shown in Figure 2.1) collects the inputs required for computing replication strategy. The controller computes the replication strategy during the current interval so that it can maximize user satisfaction during the next interval (Section 2.1.2). We use user satisfaction in the next interval as the objective since replication in the current interval is often too late to satisfy the traffic demands in the same interval. The controller then informs the APs of the replication strategy through the Internet or cellular network (in case the APs do not have Internet connectivity). We use cellular networks to send control messages as they are small. A vehicle performs the following actions during its contact with an AP:

- Step 1: The vehicle downloads the content according to the optimization results from this section.
- Step 2: After step 1, the vehicle may still have unsatisfied demand (*e.g.*, due to inaccurate prediction or insufficient capacity to replicate all the interesting content). The vehicle then downloads all the content that it is interested in and is also available locally at the AP.
- Step 3: Next, it downloads the remaining content that it is interested in from the AP's mesh network or the wireline network when the AP has

wireline connectivity.

- Step 4: Parallel to the Internet download, the vehicle can take advantage of wireless capacity by opportunistically transferring files to and from APs (Section 2.1.4).

2.1.2 Optimized Wireline Replication

Problem formulation: Our goal is to find a replication strategy that maximizes user satisfaction subject to the available network capacity. Specifically, we want to determine how to replicate files to APs during the current interval to maximize the amount of useful content that can be downloaded by vehicles when vehicles meet the APs in the next interval. To support delay sensitive applications, only content that is downloaded before the deadline counts and the other content that already missed the deadline is excluded from consideration for replication. This replication problem involves the following issues: (i) in what form to replicate the content, and (ii) how much to replicate for each file.

Applying network coding: To answer the first question, we note that directly replicating original content introduces two major problems. *First*, it is inefficient for serving multiple vehicles. Suppose multiple vehicles are interested in the same file and have downloaded different portions of the files before their contacts with an AP. If they visit the same AP, in order to satisfy all vehicles we need to replicate the union of the packets they need, which is inefficient. For example, vehicles 1 and 2 are both interested in file 1. Vehicle

1 has downloaded the first half and vehicle 2 has downloaded the second half before they encounter the AP. We need to replicate the complete file to satisfy both vehicles. *Second*, replicating original files is also unreliable. Consider a vehicle is expected to visit three APs but in fact it only visits two of the three APs, which is quite common due to prediction errors. If we just split the file into three and transfer one part to each AP, then the vehicle will not get the complete file. However, if we split the files into two and transfer one part to each AP, the vehicle still may not get the complete file since it may get two redundant pieces (*e.g.*, when it visits the two APs that both have the first half of the file).

We apply network coding to solve both problems. Specifically, we divide the original content into one or multiple files, each containing multiple packets. We use random linear coding to generate random linear combinations of packets within a file. With a sufficiently large finite field, the likelihood of generating linearly independent packets is very high [54]. For a file with n packets, a vehicle can decode it as long as it receives n linearly independent packets for it.

Network coding solves redundancy problems in the multiple vehicle case since each linearly independent packet adds value. In the above example of two vehicles, we only need to replicate one half worth of file content to satisfy both users, reducing bandwidth consumption by half. It solves reliability issue in the single vehicle case by incorporating redundancy. In the above example, we can split the file of interest into 2 and randomly generate 3 linear combinations

of these 2 pieces and replicate one to each AP. Since any two pieces are linearly independent with a high probability, the vehicle can decode the file once it gets any two pieces.

Note that we need network coding (not just source coding) in order to avoid redundancy during replication without fine-grained coordination. That is, APs should re-encode data before they replicate data to vehicles and other APs. For example, AP 1 has a complete file 1, and sends to vehicle 1 half the file, which is relayed to AP 2; similarly AP 1 sends half of the file 1 to vehicle 2, which relays it to AP 2. In order to avoid replicating duplicates to AP 2, AP 1 should re-encode the data before sending to the vehicles. In Section 2.3.2, we describe network coding cost and optimization.

Optimizing replication traffic: Using network coding, we transform the original problem of determining which packets to replicate into the problem of determining how much to replicate for each file. To solve the latter problem, we formulate a linear program, as shown in Figure 2.2. A few explanations follow. The first term in the objective function, $\sum_v \sum_f \sum_{a \in AP(v)} Q(v,f)D(v,f,a)$, quantifies user satisfaction, which is essentially the total traffic downloaded by a vehicle (before the deadline), denoted as $D(v,f,a)$, weighted by the probability for vehicle v to be interested in file f, denoted by $Q(v,f)$. The second term in the objective represents the total amount of wireline replication traffic. We include both terms to reflect the goals to (i) maximize user satisfaction, and (ii) prefer the replication that uses less traffic among the replication strategies that support the same amount of traffic demands. Since the first objective is

$\triangleright$ $Input:\quad Intv, WCap, InCap, OutCap, CT, AP, size, has, Q$

$\triangleright$ $Output:\quad x(f,i,a)$ *and* $D(v,f,a)$

Maximize: $\sum_{v}\sum_{f}\sum_{a\in AP(v)} Q(v,f)D(v,f,a) - \gamma\sum_{i\in I}\sum_{a\in A}\sum_{f} x(f,i,a)$

Subject to:

[C1] $\sum_{f} D(v,f,a) \leq WCap(a) \times CT(a,v) \qquad \forall v, a \in AP(v)$

[C2] $\sum_{a\in AP(v)} D(v,f,a) \leq size(f) - has(v,f) \qquad \forall v, f$

[C3] $D(v,f,a) \leq has(a,f) + \sum_{s\in I} x(f,i,a) \qquad \forall v, f, a \in AP(v)$

[C4] $\sum_{i\in I} x(f,i,a) \leq size(f) - has(a,f) \qquad \forall f, a \in A$

[C5] $\sum_{i\in I}\sum_{f} x(f,i,a) \leq InCap(a) \times Intv \qquad \forall a \in A$

[C6] $\sum_{a\in A}\sum_{f} x(f,i,a) \leq OutCap(i) \times Intv \qquad \forall i \in I$

Figure 2.2: Wireline replication optimization.

Optimizing wirelines replication, where v is a vehicle, f is a file, a is an AP, i is a node with wireline connectivity (which may or may not be an AP, e.g.*, a Web server), $Intv$ is an interval duration, A is the set of all the APs, I is the set of all the nodes with wireline connectivity, $AP(v)$ is the set of APs that vehicle v will visit, $Q(v,f)$ is the probability that v is interested in file f, $D(v,f,a)$ is the amount of traffic in file f vehicle v should download from AP a during a contact in the next interval, $x(f,n_1,n_2)$ is the amount of traffic in file f to replicate from node n_1 to node n_2 during the current interval, $CT(a,v)$ is average contact time of vehicle v at AP a, $WCap$ is wireless capacity, $InCap$ is incoming wireline access link capacity, $OutCap$ is outgoing wireline access link capacity, $has(n,f)$ is amount of file f a node n has, and $size(f)$ is the size of file f.*

more important, we use a small weighting factor γ for the second term just for tie breaking (i.e., when the first objective is the same, we prefer the one that has the lowest replication traffic). The value of γ should be small enough to ensure it does not dominate the first term, and our evaluation uses $\gamma = 0.001$. Note that in addition to optimizing the total downloaded traffic, it is also easy to support alternative metrics that are functions of downloaded traffic (*e.g.*, a linear approximation of proportional fairness, which balances between fairness

and total downloaded traffic [103]).

Constraint C1 in Figure 2.2 enforces that the total amount of traffic downloaded from an AP during a contact is bounded by the product of AP's wireless capacity and average contact duration. Constraint C2 ensures that the total content downloaded for each file does not exceed the total file size minus the amount of file the vehicle already has before the download. Constraint C3 encodes the fact that the amount of file the vehicle can download from an AP cannot exceed what AP already has plus what will be replicated to the APs through the wireline network during the current interval. Constraint C4 indicates that the total replication traffic in file f towards an AP is bounded by the file size minus the amount that the AP already has. Constraints C5 and C6 reflect the total replication traffic through the wireline network does not exceed the access link capacity. The formulation can support APs with and without wireline access by setting wireline capacity to zeros for APs without wireline access.

Obtaining input: As shown in Figure 2.2, we need $Intv$, $WCap$, $InCap$, $OutCap$, CT, AP, $size$, has, and Q. The $Intv$ is a control parameter that determines how frequently the optimization is performed. In our evaluation, we set $Intv$ to be 3 minutes, which gives a good balance between (i) achieving accurate mobility prediction and (ii) limiting the optimization overhead, since both (i) and (ii) decrease as $Intv$ increases. The next three inputs on link capacity—$WCap$, $InCap$, and $OutCap$—are known in advance and change infrequently. CT is estimated using historical data and only needs

to be updated infrequently. For ease of estimation, in our evaluation we set $CT(a, v)$ to be the average duration of all contacts from the trace. AP can be obtained by either letting a vehicle run a mobility prediction algorithm locally or having it send several of its recent GPS coordinates to the controller, which will perform mobility prediction. $size$, has, and Q are reported by the vehicles either through a Wi-Fi link during a contact with an AP or via a cellular link during other time. A vehicle predicts what future content to request based on the previous and current requests. For streaming content, it is relatively easy to predict as most users will request the subsequent frames. Demand prediction in general has been a well-researched problem in many domains [99, 16] and we can leverage existing solutions. Note that all the control information is small and can be easily compressed by sending delta from the previous update.

Using optimization results: To enhance robustness against errors in estimating the inputs, we use $x(f, i, a)$ and $D(v, f, a)$ to control the relative replication and download rates across different files using the weighted round robin scheduling. For example, if $x(f1, i, a) = 2 * x(f2, i, a)$, file 1 is downloaded twice as fast as file 2. In this way, we can still fully utilize network resources even if contact time or network capacity have estimation errors.

2.1.3 Optimized Mesh Replication

If some APs along the road are close together, they can form a mesh network. The mesh connectivity indicates that (i) we can now replicate content to the APs using mesh connectivity in addition to wireline connectivity, and

(ii) if a vehicle meeting an AP requests a file that the AP does not have, it is more efficient to fetch from its mesh network (if there is another AP having the file) than fetching via the slow wireline access link. A neighboring AP in the mesh network can have the file either due to explicit replication or opportunistically caching from earlier interactions.

To support (i), we make the following modifications to the replication formulation in Figure 2.2. Let $MCap(a', a)$ denote the capacity of a wireless link from AP a' to a in the mesh network, which can be different from the capacity of wireless links between vehicles and APs ($WCap$). Let $z(f, a', a)$ denote the amount of content to replicate from AP a' to a for file f through the mesh network. Let $ETX(a', a)$ denote the average number of transmissions required to send a packet from a' to a through the mesh (this can be easily estimated by measuring link loss rate using broadcast probes as in [38]). Our modifications include (1) adding $-\gamma \sum_f \sum_{(a',a) \in mesh} z(f, a', a)$ to the objective function to prefer the replication that uses less wireline and mesh replication traffic among the ones that support the same traffic demands, (2) adding $+ \sum_{(a',a) \in\ mesh} z(f, a', a)$ to the right-hand side of [C3] to indicate a node can download from AP a any content that is already available at a or replicated to a through either the wireline or mesh network, (3) adding the following two new constraints: $z(f, a', a) \leq has(a', f)$ and $\sum_{f,(a',a) \in\ mesh} \frac{ETX(a',a) z(f,a',a)}{MCap(a',a)} \leq 1$. The former constraint ensures AP a' cannot replicate more content than it has. The latter is interference constraint, which enforces that total active time of all mesh nodes cannot exceed 100% assuming

all nodes in the mesh network interfere with each other. Note that its left-hand side computes activity time by multiplying the replicated content by the expected number of transmissions normalized by the wireless capacity.

To support (ii), when AP a receives a request for a file that it does not have locally, it first tries to get from AP a' in the same mesh if the end-to-end throughput (approximated as $MCap(a', a)/ETX(a', a)$) is higher than the wireline access link; only when no such AP is found, does it fetch using the wireline access link.

2.1.4 Opportunistic Vehicular Replication

In addition to wireline and mesh replication, content can also be replicated using vehicles – a vehicle can carry content from one AP to another as it moves. This new form of replication is more effective than traditional vehicle-to-vehicle (V2V) replication, because V2V needs a very large number of vehicles to be effective whereas even a small number of APs can significantly enhance the performance by leveraging the Internet and mesh connectivity among them [18].

One way to support this new vehicular replication is to augment the LP formulation in Figure 2.2 with vehicular replication terms, which can produce wireline, mesh and vehicular replication as the final output. However, due to unpredictability in vehicular relay opportunity, we find the effectiveness of such optimization is rather limited. Interestingly, we find the following simple opportunistic vehicular replication scheme is effective.

Since the wireline fetch is bottlenecked by the slow access link, the wireless link is not fully utilized. Therefore, as mentioned in Section 2.1.1, parallel to the wireline fetch, a vehicle can take advantage of local wireless connectivity to exchange content with the AP. Such exchange has two benefits: (i) the vehicle can upload content to the AP, which can serve other vehicles later, and (ii) the vehicle can download files, which may serve its own demand in the future or the vehicle can relay the content to other APs for future service. To enhance effectiveness, we order the files to upload based on the expected future demand for the file at the AP, which is estimated as $\sum_{v:\ v \text{ visits } a} Q(v, f) demand(v, f)$, where $demand(v, f)$ is the expected size of file f vehicle v is interested in. While this vehicular replication is simple, our evaluation shows that it is highly effective.

2.2 Predicting Mobility

If we can predict the AP that a vehicle will visit, we can start replicating the required content to the AP well before the vehicle arrives so that the vehicle can enjoy high wireless bandwidth during its download. Predicting mobility for vehicles is challenging because (i) vehicles often move at high speed, which implies that there can be many possible next states and it is difficult to accurately predict transitions to a large number of next states, (ii) the GPS updates often have relatively low frequency (*e.g.*, once per minute) and tend to arrive at irregular intervals, and (iii) the road and traffic conditions are highly dynamic and difficult to predict.

To address the challenge, we develop a novel mobility prediction algorithm for vehicular networks: *K Nearest Trajectories (KNT)*. We also implement two existing algorithms based on Markov mobility models [110, 94]. In Section 2.4, we show that *KNT* achieves better accuracy on our dataset.

Algorithm: We observe that the mobility of vehicles exhibits unique structure – a vehicle follows the roads and only makes turns at the street corners or highway exits. This suggests that a good predictor should take into account the speed and direction in the previous interval as well as the underlying road structure. Our *KNT* algorithm is able to account for such information without requiring explicit knowledge about the detailed road map. Given a vehicle v_0 and current time t_0, the algorithm predicts the set of APs visited by v_0 in a future interval $[t_0+\Delta_1, t_0+\Delta_2]$ $(\Delta_2 \geq \Delta_1 \geq 0)$ in two steps:

1. *Finding K nearest trajectories.* Our algorithm first finds K existing mobility trajectories in a GPS location database that best match the recent mobility history of the given vehicle. Specifically, we maintain a database of past GPS coordinate updates: $\mathcal{D} = \{(v,t,c)\}$, where v is a vehicle, t is the time for the update, and c is the GPS coordinate. For any vehicle v and current time t, we define its mobility history MH as the set of GPS coordinates reported by v in the past δ seconds: $MH_v^t = \{c|(v,s,c) \in \mathcal{D} \wedge s \in [t-\delta, t]\}$. We also define a distance function between two trajectories: $f(MH_{v_0}^{t_0}, MH_v^t) = \sum_{c \in MH_{v_0}^{t_0}} \min_{d \in MH_v^t} \|c-d\|_2$, where $\|c-d\|_2$ is the Euclidean distance between the two locations specified by GPS coordinates c and d. Essentially, this distance function reflects the

total distance from each point on $MH_{v_0}^{t_0}$ to the closest point on MH_v^t. We then find K pairs of (v, t) that minimizes $f(MH_{v_0}^{t_0}, MH_v^t)$, i.e., the K nearest neighbors of (v_0, t_0).

2. *Voting.* For each of K nearest trajectories (v, t), we use linear interpolation (i.e., using a line to connect two adjacent points) to obtain its mobility trajectory in the future interval $[t + \Delta_1, t + \Delta_2]$. Based on this, we obtain the set of APs visited by v during that interval. We then report all the APs that are visited by at least T out of K nearest trajectories as the predicted set of APs that will be visited by v_0 during future interval $[t_0 + \Delta_1, t_0 + \Delta_2]$.

In step 1 above, to avoid computing $f(MH_{v_0}^{t_0}, MH_v^t)$ for all pairs of trajectories (which is expensive), we only compute for the trajectory pairs that are nearby. To quickly identify the trajectories that are close to the current one, we create an efficient index structure by (i) discretizing the GPS latitude-longitude coordinate space into $0.0001^\circ \times 0.0001^\circ$ grid squares, and (ii) storing all the (v, t) inside each grid square. Given (v_0, t_0), we start from its grid square and use expanded ring search to find C candidate points (v, t) residing in the same or nearby grid squares. We then find K nearest neighbors among these C candidate points.

To be general, our prediction algorithm intentionally does not exploit external knowledge (*e.g.*, certain vehicles have similar trajectory on different days, which may hold for some personal vehicles). When such information is

available, our prediction algorithm can potentially incorporate it when finding nearest trajectories to further improve the accuracy.

Parameter setting: Our algorithm has four control parameters: the number of nearest trajectories K, the number of candidate points C, the voting threshold T, and the mobility history duration H. In our evaluation, we keep $C = 32$, vary $T = 1, 2$, vary K from 2 to 12, and vary H from 60 to 180 seconds. Our results show that $(K = 4, T = 2, C = 32, H = 60)$ consistently give the best performance. We thus only report the results under this parameter setting.

2.3 VCD Implementation

We implement VCD in both Emulab [44] and our real testbed with smartphone and laptop clients. VCD consists of a controller, APs, content servers, and clients in vehicles. Emulab and testbed use the same controller, AP, and content server implementation, all of which are implemented as multi-threaded C++/Linux programs. They differ in client implementation. In Emulab, we implement a virtual vehicle program, which can emulate multiple vehicles, allowing us to conduct a trace driven emulation of all the vehicles in our trace using a few virtual vehicles. The client in the real testbed is implemented on both smartphones and laptops, which is described in Section 2.3.2.

2.3.1 System Overview

Communication between APs and controller: The APs and controller communicate with each other using TCP. As noted in Section 2.1.1, at the beginning of every interval the controller collects inputs, computes the replication strategy, and instructs content servers or APs to perform wireline and mesh replication at the desirable rates.

Communication between AP and vehicle: The communication between APs and vehicles uses UDP that sends data at close to the PHY data rate. When a vehicle contacts an AP, it sends a HELLO message that includes (i) a list of files and their sizes that it already has, (ii) the files it is interested in during the current and next intervals. Upon receiving the first HELLO message from the vehicle, the AP initiates data download to the vehicle according to the four steps described in Section 2.1.1. Meanwhile, the vehicle also sends buffered GPS updates (generated every 20 seconds in the testbed and every 1 minute in Emulab). Next, the AP determines a list of files for the vehicle to upload sorted in increasing utility as described in Section 2.1.4. The AP sends this list in a REQ message. Upon receiving the first REQ message, the vehicle initiates data upload to the AP. Both HELLO and REQ messages use soft state and are sent periodically once every control interval (100ms in testbed and 1s in Emulab). These messages also serve as heartbeats to the other party.

To achieve efficiency and reliability for data traffic, an AP applies network coding before sending the data it receives. In addition, servers leverage

a central dispatcher to distribute requests to an appropriate content server for load balancing.

2.3.2 Client Implementation

We implemented client on both Windows XP laptops and smartphones. We use HP Ipaq 910 Business Manager smartphones with Windows Mobile 6.1 Professional operating system, Marvell PXA270 416 MHz Processor, 128MB RAM, Marvell SDIO8661 802.11 b/g Wi-Fi card, and the .Net Compact Framework. Our implementation on smartphones uses OpenNet API, and that on Windows uses Managed Wi-Fi API. Implementing on smartphones introduces several challenges: (i) limited APIs and often inconsistent implementations, (ii) expensive I/O, (iii) limited system resources, and (iv) inability to implement existing wireless optimizations due to lack of low level access, which we address.

Handling expensive I/O: Since I/O on smartphones is around an order of magnitude slower than desktops, packets cannot be stored on the disk and read back on-demand for vehicular replication. For simplicity, we use an in-memory packet buffer with FIFO replacement policy. We further limit disk access during the contact with APs and push data to the disk only after the contact is over so that we can fully utilize the short contact time for data transfer.

Handling network coding cost: Due to the slow processor, thread scheduling and dynamic assignment of priorities are important. For example,

Batch size	110 packets		70 packets		35 packets	
Device	Phone	Desktop	Phone	Desktop	Phone	Desktop
Encoding	12.19s	0.0228s	4.79s	0.0088s	1.18s	0.0021s
Decoding	8.22s	0.017s	3.27s	0.0067s	0.809s	0.0012s

Table 2.1: Network coding benchmarks

network coding incurs much higher cost on the smartphone than on the desktop as shown in Table 2.1. We use packet size of 1230 bytes (i.e., the packet payload in our testbed implementation to ensure the maximum packet size is still within 1500 bytes (Ethernet MTU)). Our evaluation uses file sizes of 35, 70, and 110 packets, which correspond to minimum, median and maximum file sizes used in our experiments. To minimize the impact of decoding, we schedule the decoding thread at a low priority during a contact and increase its priority after the contact.

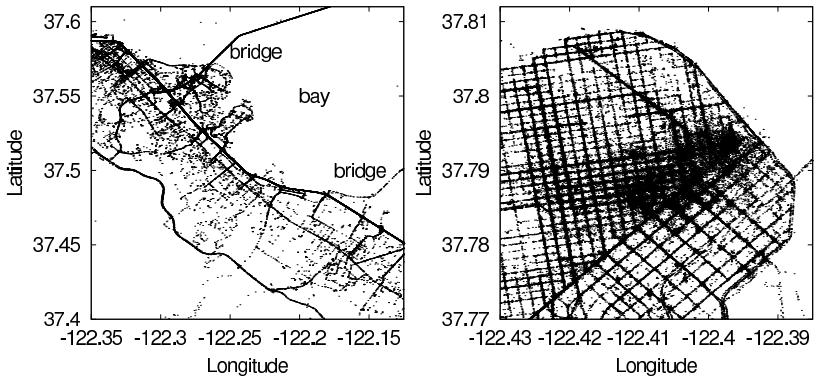
Connection setup: The ability to quickly establish connection to an AP is crucial. [25, 52] examine this problem in greater detail. In the context of smartphones, the problem becomes even harder since NDIS does not provide access to many low level parameters to implement the association optimizations proposed in the literature. Windows Mobile provides two ways to initiate connection to a Wi-Fi network programmatically, either through the wireless zero config (WZC) interface or by setting the appropriate NDIS OIDs. The association times using the WZC interfaces were around 3.0 sec, which is unacceptable in the vehicular network context. We therefore disable WZC and implement NDIS based association, which yields significantly lower association times. We also implement our own DHCP client and use the DHCP caching mechanism described in [25].

Our connection setup procedure is as follows. The smartphone scans for APs every 100 ms. When an AP is discovered, the smartphone waits for 3 RSSI readings greater than -91dB before trying to associate. We do not associate immediately because an association failure is expensive. The association procedure is retried up to 7 times with a short delay of 50ms between consecutive attempts. The various threshold values used in the scheme were chosen empirically. We report the association time and failures in Section 2.7.

2.4 Mobility Prediction Accuracy

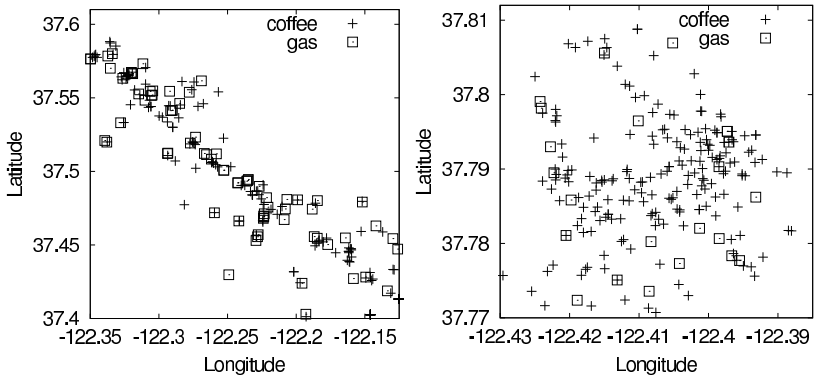
Mobility traces: We obtain real vehicular mobility traces from Cabspotting [26] and Seattle [108]. The former contain over 10 million GPS longitude and latitude coordinates for approximately 500 taxis in the San Francisco Bay Area over the course of 30 days (December 13, 2008 – January 13, 2009). The latter contains several week-long traces of city buses in Seattle during 2001. The bus system consisted of over 1200 vehicles covering a 5100 square kilometer area. The GPS coordinates are updated approximately once per minute for both Cabspotting and Seattle traces. Figure 2.3 (a) and (b) illustrate the vehicle locations along the highway and inside San Francisco. One can clearly observe the underlying street structure from taxis' GPS. Similar pattern was observed in Seattle traces.

AP locations: We consider two sets of locations for placing APs: (i) gas stations and (ii) coffee shops. We use Yahoo's Local Search API (version 3) [128] to obtain the longitude and latitude coordinates of 1120 gas stations and



(a) Taxi locations (on highways)

(b) Taxi locations (inside city)



(c) AP locations (on highways)

(d) AP locations (inside city)

Figure 2.3: Illustration of traces for mobility prediction.

1620 coffee shops in San Francisco Bay Area, as well as 618 gas stations and 738 coffee shops in Seattle. The average distance between two closest APs in the traces ranges between $345-589$ m and the median distance is $157-433$ m. There are quite a few APs whose distance exceeds 3500 m in all the four traces. The communication range between an AP and a vehicle is set to either 100 or 200 meters. We use these values because they approximate the communication ranges we measured from our vehicular testbeds using 802.11b and 802.11g,

respectively. To determine the contact period between a vehicle and an AP, we use linear interpolation to obtain the vehicle's mobility trajectory between two adjacent GPS location updates. Figure 2.3 (c) and (d) illustrate the locations for the gas stations and coffee shops in San Francisco.

Trace statistics: We analyze the traces and find that 23% – 40% of time the vehicles were parked or moved within 1 mile/hour, 70% of time they moved less than 11 – 15 miles/hour, and 90% of time they moved less than 25-27 miles/hour. Since most of the cabs are in the downtown area, they are bounded by the speed limits of the downtown area. We further study the contact duration and observe 70% of the contacts between a vehicle and an AP last less than 39-51 seconds when the communication range is 100 meters, and less than 54-82 seconds when the range increases to 200 meters. Such short contacts highlight the importance of replicating data in advance.

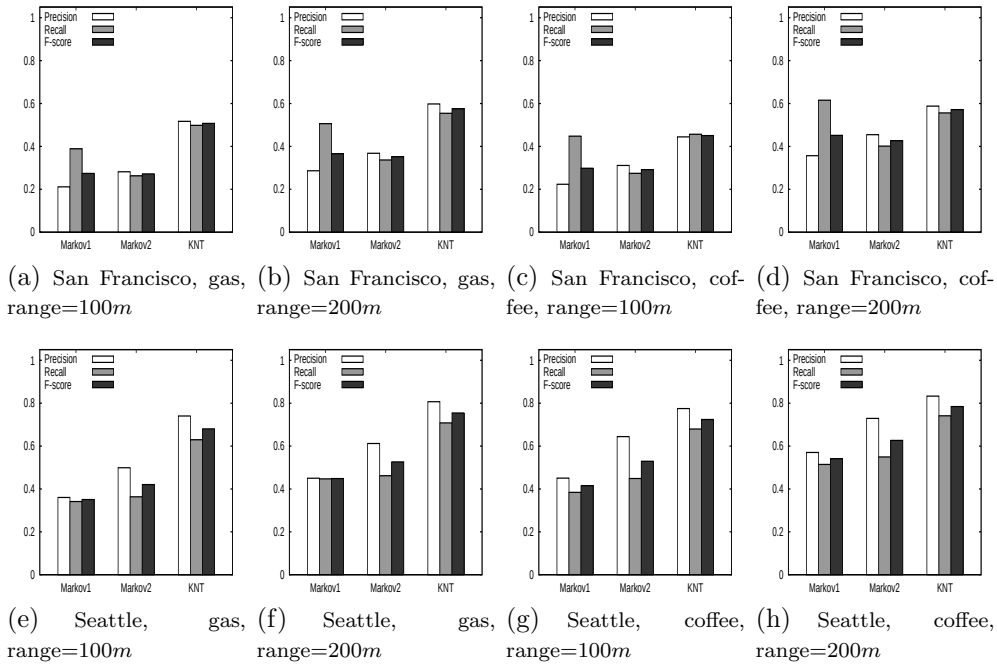
Baseline algorithms: For baseline comparison, we implement a variant of the mobility prediction algorithm in [94]. The algorithm is based on a second-order Markov mobility model. Each state has two sets of coordinates: the vehicle's location at time τ ago, and its current location. In our evaluation, τ is either 1 or 2 or 3 minutes. We deal with irregular GPS update intervals through linear interpolation. To avoid state space explosion, the algorithm discretizes the longitude and latitude coordinates into $0.001^\circ \times 0.001^\circ$ grid squares. The algorithm uses past mobility traces to learn the probability for a vehicle to transition into any new grid square given its last and current grid squares. Based on the transition probabilities, the algorithm identifies

the grid square that the vehicle is most likely to visit next, and uses the center of this grid square as the predicted new location for the vehicle after time τ. This procedure is repeated to make predictions further into the future. Based on the predicted locations, the algorithm applies linear interpolation to obtain the entire mobility trajectory and then computes the set of APs the vehicle is predicted to visit during a future interval. As in [94, 110], the algorithm falls back to a first-order Markov model when the second-order Markov model fails to make a prediction. Finally, we also implement the first-order Markov model as another baseline algorithm.

Metrics: We quantify the prediction accuracy using two metrics: (i) *precision*, i.e., the fraction of APs predicted by our algorithms are indeed visited by the vehicles in a future interval, and (ii) *recall*, i.e., the fraction of APs visited by the vehicles in a future interval are correctly predicted by our algorithms. In addition, we integrate precision and recall into a single metric called *F-score* [125], which is the harmonic mean of precision and recall: $F\text{-}score = \frac{2}{1/precision+1/recall}$. For all three metrics, larger values indicate higher accuracy.

Evaluation results: We consider the following prediction scenario as required by our replication optimization algorithm: *per-interval prediction*, which divides time into fixed intervals and the goal is to predict the set of APs that will be visited by a vehicle in the next interval. The prediction interval is set to 3 minutes, which matches the interval for periodic replication optimization. For each prediction algorithm we evaluate, we consider multiple

parameter configurations and choose the configuration that yields the best *F-score*. The results from Cabspotting traces use seven days of training data to predict the mobility on the eighth day, and results from Seattle bus traces use 5 days of training data to predict the sixth day as these traces have shorter duration.



(a) San Francisco, gas, range=100m (b) San Francisco, gas, range=200m (c) San Francisco, coffee, range=100m (d) San Francisco, coffee, range=200m

(e) Seattle, gas, range=100m (f) Seattle, gas, range=200m (g) Seattle, coffee, range=100m (h) Seattle, coffee, range=200m

Figure 2.4: Accuracy comparison of different mobility prediction algorithms.

Figure 2.4 shows the prediction accuracy when APs are placed at either gas stations or coffee shops and the communication range is either 100m or 200m. For the San Francisco taxi mobility trace (Figure 2.4 (a)–(d)), the *F-scores* of our algorithm (*KNT*) are 25-85% higher than those of the first-order Markov model (*Markov1*) and second-order Markov model (*Markov2*). For the

Seattle bus mobility trace (Figure 2.4 (e)–(h)), *KNT* outperforms *Markov1* and *Markov2* by 25–94% in terms of *F-scores*. In general, the absolute prediction accuracy for all three algorithms is higher for the bus mobility trace, because buses tend to follow fixed routes and are thus more predictable.

Finally, it is worth noting that in contrast to findings in [94, 110], *Markov2* does not significantly outperform *Markov1* in our evaluation. This suggests that with higher speed and less frequent GPS location updates, mobility prediction is more challenging in vehicular networks. As a result, solutions that perform better in less mobile environment do not necessarily perform better in vehicular networks.

Summary: The above results clearly show that our *KNT* mobility prediction algorithm consistently achieves good accuracy in vehicular networks. Later in Section 2.5, we further show that optimization based on our prediction results yields good performance in practice.

2.5 Trace-Driven Simulation

2.5.1 Simulation Methodology

We develop a trace-driven simulator for evaluation as follows. We first generate the contact traces based on the mobility traces, AP locations, and wireless communication range. When multiple vehicles contact an AP at the same time, we divide the original contacts into non-overlapping contacts, each of which has only one vehicle in contact with an AP. Such contact partitions can be easily realized in practice by letting the AP serve the new vehicle only

after it finishes serving the previous one. Similarly, when a vehicle is within the communication range of multiple APs, we also partition the contact into multiple non-overlapping intervals, each of which involves one AP. Another way to partition a contact between multiple vehicles and an AP or between multiple APs and a vehicle is to equally divide the contact time among multiple vehicles or multiple APs that are involved in the contact to mimic round-robin scheduling. The performance of these two types of partitions is similar, and we use the first partition in our evaluation.

We then feed the actual contact traces (after the above post processing), predicted contacts, and traffic demands to the simulator. The simulator updates the content at APs and vehicles based on the actual contacts, traffic demands, replication schemes, and wireless and wireline capacity at APs. We implement network coding for all data transfer to ensure only innovative packets (i.e., whose coding coefficients are linearly independent) are exchanged between APs and vehicles or among APs. We have a content server on the Internet, which has all the content, whereas all APs and vehicles are initialized with no content.

We compare (i) no replication, (ii) wireline replication alone, (iii) vehicular replication alone, (iv) both wireline and vehicular replication, (v) wireline, vehicular, and mesh replication (VCD). In all the schemes, a vehicle downloads content remotely from the Internet whenever the AP has Internet connectivity and the content is not available locally at the AP or mesh network.

To study the impact of traffic demands, we generate traffic demands

following either uniform or Zipf-like distribution. In both cases, for every interval, a vehicle randomly selects a specified number of files to request. In the uniform distribution, a file is uniformly drawn from the pool of the files that the vehicle has not requested previously. In Zipf-like distribution, the probability of requesting the ith file is proportional to $\frac{1}{i^{\alpha}}$, where i is the popularity ranking of the file and $i = 1$ indicates the most popular file. We set $\alpha = 0.4$ so that we can generate similar traffic load using both Zipf-like and uniform distributions and the performance difference is solely due to the difference in the distribution.

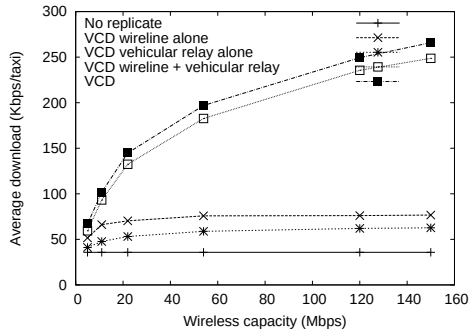
For delay sensitive applications, such as video, their performance depends on the amount of data received before the deadline. Therefore, we use average throughput per vehicle as our performance metric, which denotes the total demand that is satisfied before the deadline divided by the product of the number of vehicles and the entire trace duration (including the time without contacts with APs). The deadline is set to the end of the interval in which the demand is generated.

Our evaluation uses 2-hour trace, which exhibits similar contact characteristics as in the 1-day trace, shown in Section 2.4. Other default settings used in our evaluation include: 100-meter communication range between APs and vehicles, 500-meter communication range among APs (well within reach by many mesh routers [12, 84]), Zipf-like traffic demands, placing APs at coffee shops, all APs having 22 Mbps wireless link, half of the APs having Internet links with 2Mbps while the other half have no Internet connection. The con-

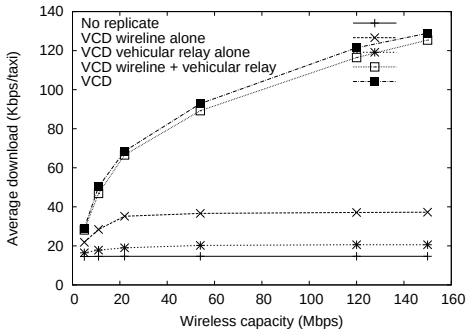
tent server has a 1 Gbps Internet link and zero wireless capacity to indicate that it is not directly reachable by vehicles. There are 1200 files in total. Each user requests 20 files every 3-minute interval, each file has 2K packets, which contains 1000 bytes. Every file represents either a video clip or one chunk in a larger video file (*e.g.*, We divide a large video file into smaller chunks and generate random linear combinations of packets within each chunk for efficient replication). We further evaluate the effects of changing these parameters.

2.5.2 Simulation Results

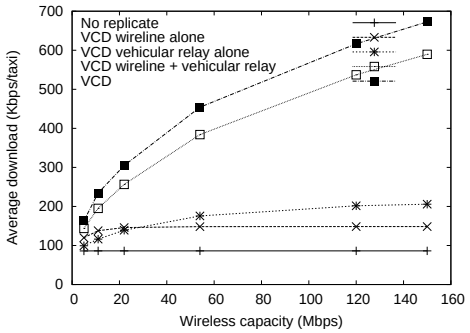
Varying wireless bandwidth: In Figure 2.5, we plot the total downloaded content as we vary wireless bandwidth from 5, 11, 22, 54, 120, and 150 Mbps. We make the following observations. First, in all cases VCD significantly out-performs the other schemes and its benefit increases rapidly with wireless capacity. Second, as we would expect, no replication performs the worst. Interestingly, its performance remains the same as we increase wireless capacity. This is because without replication APs often do not have content locally and the wireless download is bottlenecked by slow Internet access capacity. This further demonstrates the need of replication. Third, the performance of both wireline and vehicular replication alone initially improves with increasing wireless capacity and then tapers off. This is because limited Internet capacity prevents fully taking advantage of large wireless capacity. In comparison, harnessing both wireline and vehicular replication opportunities can effectively utilize the large wireless capacity when available. Adding



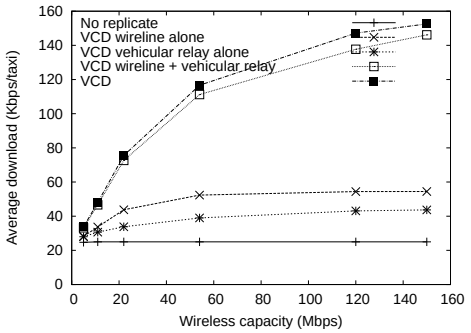
(a) San Francisco, coffee shops, range=100m



(b) San Francisco, **gas station**, range=100m



(c) San Francisco, coffee shops, **range=200m**

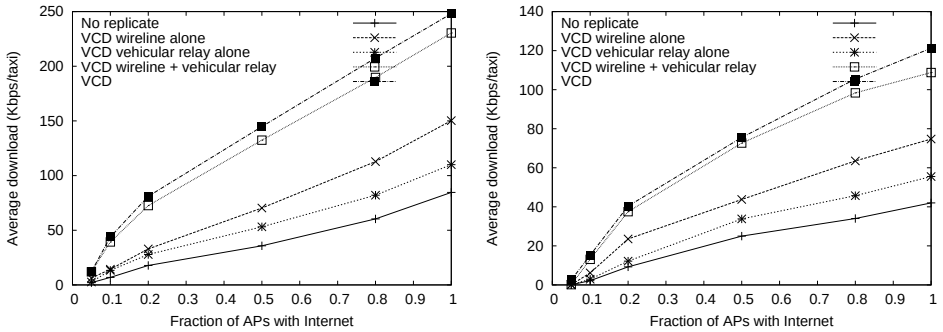


(d) **Seattle**, coffee shops, range=100m

Figure 2.5: Average throughput under varying wireless capacity. Average throughput of 50 vehicules under varying wireless capacity and Zipf-like traffic demands. The difference from the base configuration is in bold.

mesh replication further increases average throughput by 14-20% under high AP density (Figure 2.5(c)), and by 3-13% in low AP density. The benefit of mesh replication can be increased further if APs use high gain antennas or MIMO. Overall, at 22Mbps Wi-Fi capacity, VCD achieves 70 – 300 Kbps average throughput per vehicle depending on the AP density, which can support video streaming applications.

Varying fraction of APs with Internet connectivity: Next we



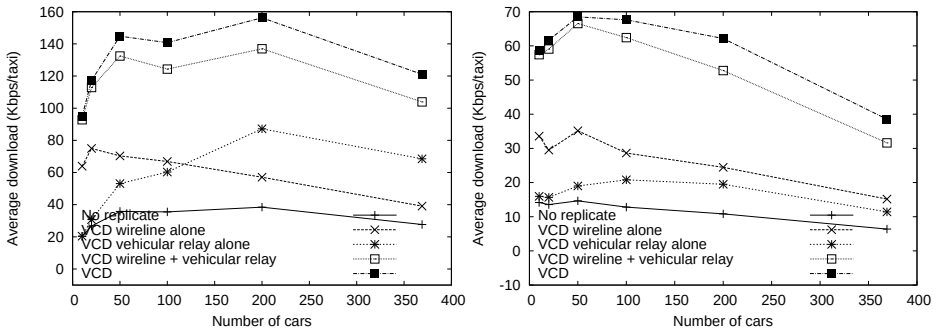
(a) San Francisco (b) Seattle

Figure 2.6: Average throughput under varying fraction of APs with Internet. Average throughput under varying fraction of APs with Internet (Zipf-like traffic, APs at coffee shops, range=100, 50 vehicles).

vary the fraction of APs with Internet connectivity. Figure 2.6(a) and (b) plot the average downloaded traffic in San Francisco and Seattle traces, respectively. As we can see, VCD continues to significantly out-perform the other schemes. In addition, the benefits of all types of replication increase with the fraction of APs that have Internet connectivity. The rate of such increase is faster for the replication schemes that involve wireline replication, since they explicitly take advantage of the new wireline capacity to push data.

Varying number of vehicles: To further evaluate the impact of degree of deployment, we vary the number of vehicles by randomly selecting a subset of vehicles from the traces. Figure 2.7 summarizes the performance results. We make the following observations. First, VCD continues to perform the best in all cases. Second, increasing the number of vehicles initially improves the average throughput because more content are available locally at APs due to previous requests coming from other users. In addition, increasing

the number of vehicles also creates more wireless relay opportunities. However, a further increase degrades performance due to increased contention for limited wireline and wireless resources. Third, the benefit of mesh replication increases with the number of vehicles. When we use all the vehicles in the two-hour traces, we find that the mesh replication helps to increase throughput by 17-22%. This is because increasing the number of vehicles increases vehicular relay opportunities and makes it more likely to have content available at nearby mesh nodes.

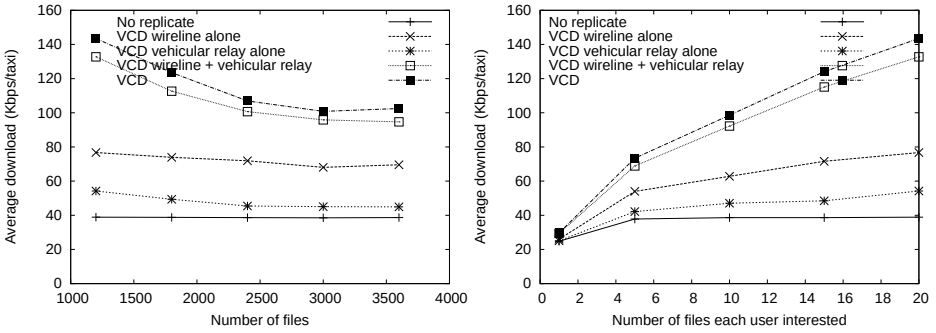


(a) coffee shops (b) gas stations

Figure 2.7: Average throughput under a varying number of vehicles (San Francisco, Zipf-like traffic, range = 100m).

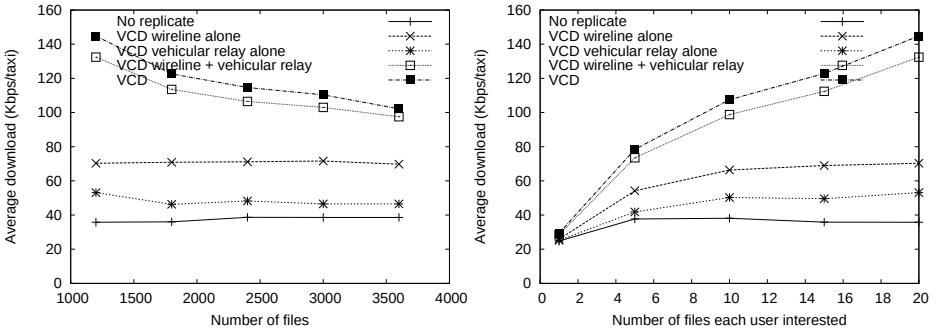
Varying traffic demands: Figure 2.8 shows the performance for uniformly and Zipf-like distributed traffic demand, respectively. As before, VCD performs the best in all cases. The performance of uniform and Zipf-like distributed traffic receives similar performance. Moreover, decreasing the total number of files tends to improve performance as demands are more concentrated and less replication is required to satisfy them. Finally, the replication benefit tends to increase with an increasing number of files requested by each

user. This is because when a user is interested in more content, it is more likely to have some locally available content that satisfies the user.



(a) Uniform: vary # files

(b) Uniform: vary # files per user



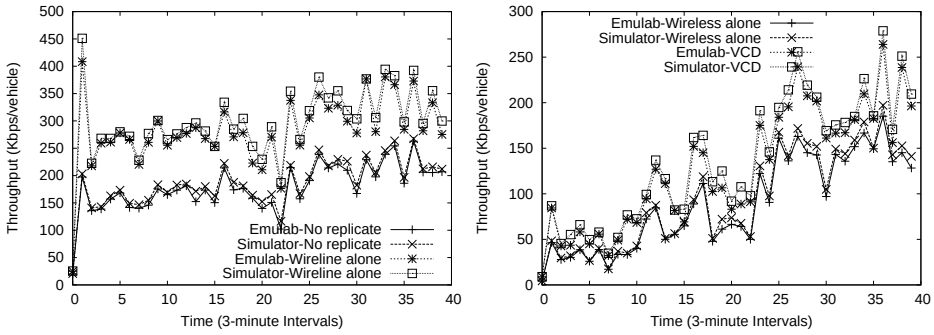
(c) Zipf-like: vary # files

(d) Zipf-like: vary # files per user

Figure 2.8: Average throughput under varying traffic demands (San Francisco, vehicle=50, range=100m, coffee shops).

2.6 Trace-Driven Emulation

The goal of our Emulab implementation is twofold: (1) validate simulation results, and (2) evaluate the performance of VCD at scale, which is hard to do in testbed experiments.



(a) All APs have Internet (b) 10% APs have Internet

Figure 2.9: Cross validation
Compares the performance in Emulab and simulation

2.6.1 Validation

To validate the simulation results, we compare them against those obtained from Emulab under identical settings. We consider the 30 most interactive APs from the trace contacting 100 vehicles. The radio range is 200m. Given limited machine availability on Emulab, we emulate multiple APs and vehicles on each machine. This limits the link capacity we can select per AP or per vehicle. Hence, our evaluation uses 1Mbps and 6Mbps as the Internet and wireless link capacities, respectively.

Figure 2.9 shows the average throughput for each interval in Emulab and simulator. In Figure 2.9(a), we consider that all APs have Internet connectivity and compare the simulation and emulation performance under no replication and wireline replication alone. We observe that the simulation results closely follow that of Emulab and the discrepancy between them is below 10%. Next we consider only 10% of the APs have Internet connectivity and

Packet type	Avg KB	% of total traffic
Controller to APs	192	0.006
APs to controller	1483	0.048
Content server to AP data	3078200	99.946
Vehicles to APs	49122	1.599
APs to vehicles data	3023100	98.401

Table 2.2: Average control message overhead per interval.

compare the performance for vehicular replication alone and VCD in both simulator and Emulab. In this case, since most APs are not connected to the Internet and there is no mesh connectivity, most content is replicated via vehicles. Figure 2.9 (b) shows that the simulation results match well with Emulab results: within 10% difference for both vehicular replication and VCD.

2.6.2 Micro-benchmarks

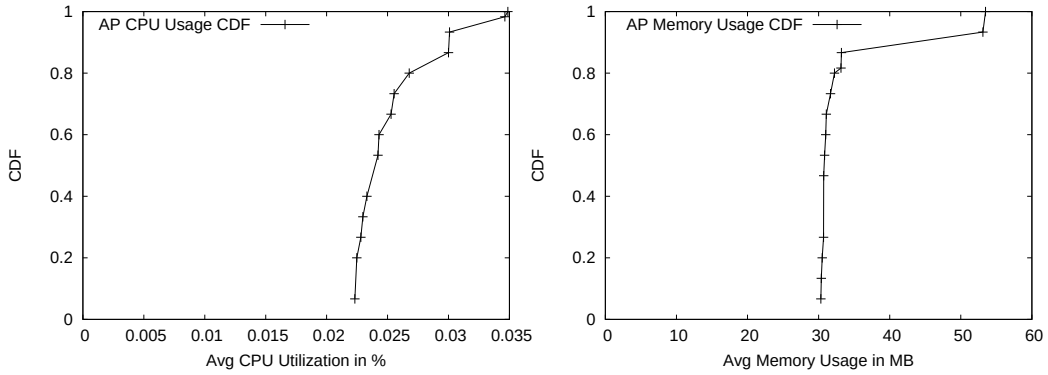
The following micro-benchmark results show that our implementation is efficient and light-weight even when operating at scale. We emulate the 120 most interactive APs and 317 vehicles from the trace.

Control message overhead: Table 2.2 shows the per-interval control message overhead. We observe that control messages constitute only 0.054% of the total wireline traffic exchanged amongst APs and between APs and the controller, and constitute only 1.6% of the total wireless traffic between APs and vehicles.

Controller efficiency: Next we evaluate the efficiency of the controller. On a 2.133GHz Xeon machine with 3GB RAM, average CPU and

	Average Latency (ms)
Pre-processing for LP	1307
LP Computation	6512
LP Result Processing	32
Total	7851

Table 2.3: Average controller processing delay per interval



(a) Average CPU Usage (b) Average Memory Usage

Figure 2.10: CDF of average CPU and memory usage at all APs.

memory usage of the controller is 2% and 38 MB respectively. The average latency at the controller is 7.8s, which is a small fraction of the 3-minute interval. Table 2.3 further shows the breakdown of the processing latency at the controller. The pre-processing stage involves predicting which APs will be visited and preparing input file for lp_solve. Out of 7.8s, the LP computation takes 6.5s. It is performed on Emulab using *lp_solve* [77] due to licensing issues with *cplex* [39], and the time can be further reduced if cplex is used instead.

AP load: Finally we evaluate the scalability of APs by running 120 instances of the AP on 2.133GHz Xeon machines with 3GB RAM. Figure 2.10

shows the CDFs of average CPU usage. we find that all APs have roughly the same usage, with each AP instance consuming only 0.01% CPU load and 33 MB of memory. Therefore it is light-weight.

2.7 Testbed Experiments

We evaluate our approach using two testbeds to understand its feasibility and effectiveness under realistic wireless conditions. The first testbed consists of 9 APs deployed in office buildings near the road. The APs are Linux desktops equipped with 802.11b radios, which are set to a fixed data rate of 11Mbps. The second testbed consists of 4 APs deployed outdoor equipped with 802.11n radios that use auto-rate. 802.11n radios use 2.4GHz frequency with a 20MHz band. In both testbeds, the APs have 1Mbps wireline access link connecting to the back-end content server. In the 802.11b testbed, 3 out of the 9 APs forms a mesh network as a 2-hop linear chain, whereas the 4 APs in the 802.11n testbed forms a mesh network with pairwise connectivity. In both testbeds, mesh communication takes place using additional 802.11b radios. We implement clients on both Windows Mobile Smartphones and Windows XP Laptops. Smartphone clients are used in 802.11b experiments and laptop clients are used in 802.11n experiments. Both clients ran a video streaming application during the car ride. The cars travelled around the testbed at 15 mph (speed limit). We expect that the driving speed does not significantly affect the performance when association time is small, because increasing speed reduces both on-time (i.e., contact time) and off-time (i.e., the time between

two consecutive contacts).

2.7.1 Connection Setup

802.11b: Due to deployment constraints, the placement of our 802.11b APs is not ideal: 4 of our APs were placed on the 3rd floor of buildings, limiting their range; and 3 APs were placed in high AP density areas, with 50-70 APs within their range, causing heavy interference. This deployment stress-tests our system. In our experiments during car rides, we were able to associate successfully for 65.2% of all attempts. Most of the failures came from the 3 APs deployed in the high AP density area: association success percentage was only 33.3% for these APs. In fact, even the Windows Mobile Wi-Fi manager utility experienced problems such as very long connection time and adapter freezing near these APs even without any movement. The other access points can successfully associate for 85.7% of the time. The association time in our experiments has minimum, median and maximum of 36ms, 844ms, and 14867ms, respectively. 70% of the associations finish within 2 seconds. We retry association up to 7 times and the median retry count is 1.

802.11n: In our 802.11n outdoor testbed, association success rate was 89.58% out of 48 attempts. The minimum, median and maximum association times were 48 ms, 162 ms, and 4086 ms, respectively. 80% of the associations finish within 246 ms and the median retry count was 1. The better results for 802.11n testbed were because (i) we used laptops as clients, (ii) APs were placed outdoor closer to vehicles, and (iii) MIMO in 802.11n improves received

	Download (kB)	Play time (sec)
No replication	29297	3662
Wireline	71930	8991
Wireline + Mesh	79440	9930
Full replication	92493	11562

Table 2.4: Throughput of wireline and mesh replication in the 802.11b testbed

	Download (kB)	Play time (sec)
No replication	16857	2107
Wireline	123175	15387
Wireline + Mesh	130827	16353
Full replication	136479	17060

Table 2.5: Throughput of wireline and mesh replication in the 802.11n testbed

signal strength.

2.7.2 Wireline and Mesh Replication

We implemented a video streaming application that can play H.264 videos (downloaded from APs) encoded at 64Kbps. We divide every video into multiple files and use network coding to generate random linear combination of packets within a file. Once enough packets are received for the file, the file is decoded and passed to the video player on the smartphone/laptop to play in proper order using the Windows Mobile media player plugin.

Tables 2.4 and 2.5 compare the performance of our optimized wireline and mesh replication with no replication and full replication at all the APs in 802.11b and 802.11n testbeds, respectively. We consider two performance metrics: total download size and total amount of time the video can play

	No replication		Wireless replication	
	Car 1	Car 2	Car 1	Car 2
AP1	0	0	Upload 780 pkts	Download 780 pkts,20 files
AP2	0	0	Download 1159 pkts, 20 files	Upload 1159 pkts

Table 2.6: Comparison between performance with and without vehicular replication.

(which is proportional to the download size). We report the averages over 3 runs. The full replication assumes every AP has all the files and serves as an upper bound. In both experiments, we follow the planned trajectory, which was fed as input to the controller. In 802.11b testbed, wireline replication alone and wireline plus mesh replication performs 2.45x and 2.7x better than without replication, respectively. In 802.11n testbed, the throughput of wireline and wireline plus mesh replication is 7.3x and 7.8x higher than without replication, respectively. This demonstrates the effectiveness of replication. Moreover, the benefit increases with wireless capacity. There is a gap between the performance of VCD and full replication, since the Internet bottleneck prevents complete replication of all the required files.

2.7.3 Vehicular Replication

To show the benefit of vehicular replication, we use the following setup. Car 1 follows the route $AP1 - AP2$, and Car 2 follows the route $AP2 - AP1$. Car 1 possesses files 1-20 and is interested in files 21-40, while car 2 has files 21-40 and is interested in files 1-20. Both $AP1$ and $AP2$ lack Internet and mesh connectivity. Therefore, without vehicular replication, neither car can get the content it is interested in and the total throughput is 0 under no replication, wireline replication alone, and mesh replication alone.

In comparison, VCD exploits the vehicular replication opportunity. When car 1 meets $AP1$, VCD finds that files 1-20 have highest utility because it predicts car 2 will visit $AP1$ soon and need these files. So $AP1$ instructs the car to upload them first. Similarly, car 2 uploads file 21-40 at $AP2$. When car 1 reaches $AP2$ it can download these files. Similarly, car 2 can download files 1-20 from $AP1$, leading to much higher throughput. Table 2.6 shows that both cars download their interested files in the actual road experiments.

2.8 Related Work

We classify related works into three areas: (i) vehicular networks, (ii) disruption tolerant networks (DTNs), and (iii) mobility and demand prediction.

Vehicular networks: A variety of novel techniques have been proposed to optimize various aspects of communications in vehicular networks. One class of works focus on techniques for optimizing one-hop communication between a vehicle and nearby APs. For example, CarTel project [29] proposes architectures for vehicular sensor networks, and develops a series of techniques to optimize association, scanning, data transport protocols, and rate selection. ViFi [17] proposes to take advantage of multiple nearby APs to improve communication with passing vehicles. [27] conducts in-depth study of various rate adaptation schemes in vehicular networks and proposes to select data rate based on a combination of RSSI and channel coherence time. [90] uses directional antennas to maximize the transfer opportunity between the

vehicle and the AP. These works are complementary to our work, which focuses on end-to-end performance of content distribution. We can potentially leverage these approaches to improve the performance of the last hop. With these enhancements, the gap between Internet and wireless capacity will further increase and make replication even more important. Another class of works consider changes to applications to support vehicular networks. For example, Thedu [16] transforms interactive Web search into one-shot request and response process to reduce access delay. While Thedu still requires connecting with the remote server, we replicate content to APs to eliminate the Internet bottleneck. The third class of work studies protocol issues. [42] proposes fast connection establishment, scripted handoffs, and prefetching at APs using HTTP range requests. Finally, there are a few works on vehicle-to-vehicle communication. For example, SPAWN [41] uses gossip for file transfer and CarTorrent [68] extends SPAWN and is implemented in a testbed. [31] treats vehicular networks as a special type of DTNs and focuses on leveraging vehicle to vehicle (V2V) communication to deliver content. As mentioned earlier, inspired by the analysis in [18], we leverage APs as the rendezvous points for replicating content among vehicles. We focus on optimizing content replication given limited wireline and wireless resources, which has not been studied earlier.

Disruption tolerant networks: Vehicular networks can be considered as a special type of disruption tolerant networks (DTNs) and benefit from advances in this area. Different from traditional DTNs, which focuses

on communicating with a specific node, we focus on content delivery. Epidemic routing [118] was initially proposed for DTNs, where any two nodes exchange messages whenever they meet. Recently, utility-based replication was proposed, where nodes replicate data over the best contacts according to some utility (*e.g.*, mobility history [62] or delay [15]). We leverage both utility-based optimization for wireline/mesh replication and target wireless replication to maximize effectiveness.

Mobility and demand prediction: There is a large body of literature on mobility prediction, ranging from coarse-grained prediction in cellular networks (*e.g.*, [5, 6, 73, 74, 100]) to more fine-grained prediction in Wi-Fi networks (*e.g.*, [94, 111]). In particular, [110] compares various predictors in literature and suggests that 2nd order Markov with a simple fallback mechanism (when there is no prediction) performs well. [47] builds mobility profiles for users and statistically predicts the next social hub the user will visit. [94] builds the user's customized mobility models on the devices themselves, and uses a second order Markov model to predict the connection opportunity and its quality of the device with an AP. [82] uses the past history to identify opportunities for media sharing in ad hoc DTNs. These works focus on low speed (*e.g.*, personal mobility). Vehicles travel much faster and make mobility prediction more challenging.

In this chapter, we do not study demand prediction, since it is a well-researched topic (*e.g.*, [99, 16]). We can leverage the existing work to enhance the effectiveness of VCD.

Chapter 3

Randomizing Routing in Multi-Party Internet Video Conferencing

Motivation: The popularity of multi-party Internet video conferencing has grown tremendously recently due to rapid advances in Internet technology, low-cost high-quality equipment, globalization, business continuity, and increasing economic pressures. The world video conferencing market is expected to reach $10 billion by 2015 [48].

In response to the rapidly growing demands for Internet video conferencing, significant advances have been made in both research community and commercial products. For example, there are numerous video conferencing products in the market, such as Skype [109], Cisco WebEx [122], AT&T Telepresence [10], Microsoft Windows Meeting Space [83], QQ [102], Google+ Hangouts [50], Apple FaceTime [8], just to name a few. In addition, significant research has been devoted to enhance the performance of video conferencing quality (*e.g.*, [7, 64, 70, 69, 67, 78, 4, 127]).

Challenge: Despite significant advances in Internet video conferencing technologies, supporting high-quality multi-party video conferencing at a low cost remains an important open challenge. The fundamental challenge lies in the potentially all-to-all nature of the communication in video conferencing. If

we blindly deliver every participant's video stream to all the other participants, then the total bandwidth requirement will grow quadratically with the number of participants. As the number of participants increases, the bandwidth requirement can quickly become prohibitive or simply infeasible due to each participant's limited access link bandwidth. The gap between the bandwidth requirement and access link capacity can only increase in the future due to the rapid increase in the demand for high-resolution videos and the much slower increase in the access link capacity.

Due to the above fundamental challenge, the existing solutions have to either constrain the forms of interactions among participants or incur high cost. For example, some solutions focus on audio or low-resolution video delivery (*e.g.*, QQ) or support only a limited number of participants (*e.g.*, Skype and iChat). Some products (*e.g.*, Webinar [123]) only allow one speaker to broadcast its video/audio streams while the other participants either only passively listen or just post text messages. Meanwhile, less restrictive commercial products are often quite expensive (*e.g.*, 33 cents/minute/person for WebEx and 32 cents/minute for Adobe Connect).

So an important research question is how to effectively support multi-party video conferencing at a low cost without constraining the rich form of interactions among participants.

Solution strategy and requirements: We believe the key opportunity in supporting multi-party Internet video conferencing lies in the nature of communication among participants. Even when the total number of partic-

ipants is large, each participant is typically only interested in receiving video streams from a small number of participants at any given time. For example, one is typically interested in watching the facial expressions and gestures of current and recent active speakers. One may pay more attention to close friends, family members, and colleagues in the conference Furthermore, in a video conference that is initiated among online video game players, a player is most interested in watching the nearby players in the game world. In general, it is tiring for an individual participant to simultaneously pay attention to a large number of other participants due to one's limited attention span. Moreover, the limited screens especially on the mobile devices, which are commonly used for video conferences, further constrain the number of participants that a user can see at any moment.

Therefore, there is no need to deliver video streams in an all-to-all fashion, which is either prohibitive or simply infeasible as the number of participants increases. Instead, our solution strategy is to deliver to each participant a set of video streams that he/she is *currently* interested in. However, doing so imposes a number of important requirements:

1. *Coping with traffic uncertainty.* Significant uncertainty exists in the traffic demands of multi-party video conferencing. Specifically, the set of video streams that a user is interested in change dynamically depending on the current active speakers or current interactions in a multi-party video game when a video conference is used to facilitate gaming.

Moreover, it is common for a user to be interested in viewing multiple participants, and it is prohibitive to enumerate all possible combinations of interested participants and optimize multicast trees based on the current set of interested participants. In addition, the video traffic rate of an individual video stream can also fluctuate rapidly over time.

Whenever traffic changes, the multicast routes should be updated. However, it is hard to compute new routes to optimize performance and avoid network congestion for the following reasons. First, demand and network conditions are unpredictable. Optimization based on precise knowledge of these information is fragile. Second, even with the precise information, optimizing routes in a distributed way is challenging due to uncoordinated decisions, which can easily cause flapping where everyone moves to the links with more spare bandwidth and later incur congestion and then move to the other links and creates new congestion. This is a well-known drawback of load-based routing. On the other hand, while centralized optimization coordinates routing changes across different nodes, it avoids flapping but incurs significant communication and computation overhead as well as creating a single point of failures. Therefore, we seek robust multicast routing that can work well under unpredictable traffic.

2. *Accommodating resource heterogeneity.* Network resources are heterogeneous across different clients and also fluctuate dynamically. If not careful, we can cause severe network congestion and serious performance degradation in some areas while under-utilizing resources in other areas.

3. *Providing performance guarantee.* If the subscribers in the session do not have enough resources, the video quality degrades significantly. How to combine resources from peers both in the same and different sessions and from a content distribution network (CDN) is an open issue.

4. *Supporting dynamic rate adaptation.* Finally, it is possible that even using all types of resources not all video sessions can be supported using the highest resolution. In this case, it is necessary to properly adapt streaming rates dynamically according to the current available resources.

Our approach: We propose a novel Valiant multicast routing (VMR) to support video conferencing. The salient feature of VMR is that it can efficiently utilize capacity from all nodes by effectively balancing the load among them. It achieves this by randomly selecting nodes according to their upload capacity. To achieve better load balance, we leverage the power of two random choices – for each packet a source randomly picks two candidate relay nodes according to their upload capacity and uses the one with the lower load as the final relay node. Compared with optimization based approaches, the randomized algorithm in VMR incurs much lower overhead, and is faster to adapt and more robust to measurement errors.

The traditional "power of two choices" has been shown effective when we know the load of all nodes completely accurately (*e.g.*, [13]), but its performance degrades significantly with decreasing accuracy of load information (*e.g.*, [85, 86, 40]). In fact, how to effectively leverage the power of two choices

in face of stale information is recognized as an important open problem in [87]. We develop a novel technique to explicitly account for the uncertainty in the load information.

We further develop three extensions. First, nodes in the wide-area network may have large delay and randomly selecting relays may result in high end-to-end delay (*e.g.*, letting a sender in US to use a node in China to relay to a destination node in US). We cluster nodes according to their proximity, and adapt VMR to take into account both bandwidth and delay.

Second, VMR works well when there is sufficient total resource. When the resource is insufficient, balanced load would still incur network congestion and packet losses. Therefore, in addition to relaying traffic through subscribers, we also use non-subscribing peers from the same session, or peers in different sessions, or CDN nodes as relay nodes, and extend VMR to leverage different types of relay nodes.

Third, if the total capacity of all nodes (including peers from other sessions and CDNs) is still insufficient, we develop a distributed rate adaptation scheme to quickly adapt to the current network conditions.

Contributions: Our major contributions are as follows: (i) a novel Valiant multicast routing with two random choices that can effectively cope with stale load information, (ii) a proximity-aware extension of Valiant multicast routing to account for heterogeneous delay, (iii) an approach to leverage resources from other peers or CDN nodes to enhance the performance, (iv) a

distributed rate adaptation scheme. Note that our load balancing algorithm is general and can be applied in many other load balancing contexts, such as hashing, circuit routing, and backbone routing. It can also be easily extended to more than two random choices.

The remainder of this chapter is organized as follows. We present our Valiant multicast routing and various enhancements in Section 3.1. We describe our implementation in Section 3.2. We evaluate our approach using real implementation and experiments in Section 3.3. We survey related works in Section 3.4.

3.1 Approach

We use Valiant load balancing (VLB) to perform overlay routing in video conferencing. We first introduce the basic Valiant load balancing. Then we enhance load balancing using the power of two choices while coping with stale information. We further describe how to handle clients with heterogeneous delay, utilize peers in other sessions and CDN nodes, and adapt streaming rates.

3.1.1 Valiant Multicast Routing (VMR)

Valiant load balancing: Valiant [119] first proposed Valiant load balancing for processor interconnection networks. Since then, Valiant load balancing has been applied to many other contexts, such as designing scalable routers [30, 65] and designing backbone networks [134]. In particular, Valiant

routing is attractive in backbone networks because it can minimize the maximum link load without the knowledge of the traffic matrices in advance, which is common in practice.

The basic idea of VLB is to probabilistically dispatch requests to the servers according to their capacity. A salient feature of VLB is that it balances the normalized load across different nodes. That is, let $\frac{L_i}{C_i}$ denote normalized load of node i, where L_i is node i's load and C_i is node i's capacity. The normalized load across all nodes is similar (i.e., $\frac{L_i}{C_i} \approx \frac{L_j}{C_j}$ for all pairs of nodes i and j).

Valiant multicast routing: We apply VLB to route video streams. Specifically, a traffic source, denoted as S, constructs a full mesh spanning over all its subscribers and itself. If the total capacity from all the nodes in the mesh cannot sustain S's streaming rate (i.e., $\sum_i C_i < n \cdot R$, where n is the number of subscribers to S and R is streaming rate), S can add other nodes (*e.g.*, other peers and/or CDN nodes) to the mesh to meet the minimum bandwidth requirement (Section 3.1.4).

S probabilistically selects a node as the next hop relay. The probability of selecting a node i is proportional to its upload capacity C_i (i.e., the probability is $\frac{C_i}{\sum_j C_j}$). The node may select itself as the next hop relay node using the same probability. The selected relay node then directly forwards to the final destination as specified by the traffic source. In this way, each data fragment traverses no more than two hops. When the source selects itself as a relay node, data is directly sent from the source to the subscribers (i.e., traversing

one hop). The upload capacity of access links is assumed to be bottleneck and measured as described in Section 3.2.

3.1.2 Robust VMR with Multiple Choices

Due to randomness, the normalized load of different nodes, defined as the ratio between the number of requests dispatched to the nodes and their capacity $\frac{L_i}{C_i}$ may not be balanced. In particular, given m packets to be routed, for each packet we randomly select one out of n relays. The maximum normalized load is $ln(n)/ln(ln(n))(1 + O(1))$ with a high probability.

To improve load balancing, we leverage the power of multiple random choices (i.e., randomly select d nodes and use the one with the lowest load as the relayer). [13] shows that using d random choices reduces the maximum normalized load to $ln(ln(n))/ln(d) + O(1)$ with a high probability.

Coping with stale load information: The above results on the power of multiple random choices hold only when the load is known completely accurately. In practice, the load is updated periodically and can easily get stale. The stale information can significantly degrade the effectiveness of this method. For example, a node may advertise a load value lower than others, attract higher amount of traffic thereafter, and soon become overloaded. However this node will keep attracting increased amount traffic until the next update of its load. This may cause significant load imbalance. Several previous works (*e.g.*, [85, 86, 40]) have used models and simulations to show the stale information can seriously degrade performance.

How to effectively exploit the power of two random choices in the presence of stale load information is considered as an important open problem [87]. [40] suggests to predict the load based on the previously published load and the new arrival and service rate and apply the predicted load instead of published load. In practice, the prediction accuracy is limited because we may not have a global view of the arrival rates (*e.g.*, a client only knows its own sending rate but not other clients' sending rate) and service rate fluctuates (*e.g.*, available bandwidth from a client may change over time). Therefore we propose to explicitly handle the uncertainty in the load information.

Two random choices under uncertainty: We propose the following novel yet intuitive approach to handle the uncertainty in the load information. Suppose we apply the "power of two choices" and hash a request to two nodes i and j. If we have up-to-date accurate information, then we will select either node i or node j based on whether $x_i < x_j$, where x_i and x_j are load of nodes i and j, respectively. Now to capture uncertainty and staleness of the load, we treat x_i and x_j as random variables instead of known constants. We should pick node i with probability $prob\{x_i < x_j\}$. This is because for each instantiation of the two random variables, we should pick the one with lower value.

Let m_i be the predicted load of node i. Let s_i be the standard deviation of the prediction error. Let x_i be the real load of node i. So we have: $mean(x_i) = m_i$, and $std(x_i) = s_i$. Assuming independence between x_i and x_j, we independently draw a random sample y_i and y_j from the probabilistic

distribution of x_i and x_j, respectively. We then choose node i if and only if $y_i < y_j$. In this way, the probability to choose node i is exactly $prob\{x_i < x_j\}$. In Section 3.3, we empirically show the load follows a Gaussian distribution in our experiments. We therefore draw y_i and y_j from Gaussian distributions.

Multiple random choices under uncertainty: We can generalize the above procedure to support d random choices in the presence of uncertain load information. Again, let x_i be a random variable that represents the current load at node i. Let m_i and s_i be its mean and standard deviation, respectively. Let $i_1, i_2, \cdots, i_d$ be the set of d nodes a request is hashed into. We independently draw a random sample y_{i_k} from each random variable x_{i_k}'s distribution (which is again assumed to be Gaussian). Let y_{i_*} be the smallest sample in set $\{y_{i_k} | k = 1, \cdots, d\}$. We then assign the new request to node i_*. The key distinction from classic VLB with multiple random choices is that we treat the current load as a random variable instead of a known constant. We then draw independent random load samples from the corresponding load distributions and select the node with the smallest load sample. In our experiments we assume x_i has Gaussian distribution because our empirical results in Section 3.3 suggest that Gaussian distribution is a reasonable approximation to the load distribution.

Estimating m_i and s_i: The only question that remains is how to estimate m_i and s_i. We use exponentially weighted moving average (EWMA) to estimate both m_i and s_i similar to TCP. Specifically, we initialize $s_{load} = x_i$ and $s_{dev} = x_i/2$, where x_i is the first sample. Upon each new x_i, we update

$s_{dev} = (1-\beta) * s_{dev} + \beta * |s_{load} - x_i|$ and $s_{load} = (1-\alpha) * s_{load} + \alpha * x_i$.

We use $\alpha = 1/8$ and $\beta = 1/4$ as in TCP. s_{load} gives an estimate of the $m_i = mean(x_i)$, while s_{dev} gives an estimate of mean absolute deviation (MAD) (i.e., $mean(|x_i - m_i|)$). Since x_i is assumed to follow a Gaussian distribution, $MAD \approx std * sqrt(2/\pi) = std * 0.79788456$. So the standard deviation s_i can be estimated as $s_i = s_{dev}/0.79788456$.

Dealing with large video fragments: We divide fragments into similar sizes to reduce the variability of job sizes. Fragments (*e.g.*, FLV tags) may vary significantly in sizes depending on their types. For example, the size of audio tags is typically constant and small whereas key-frame video tags can be more than 100 times larger than a typical audio tag. If relay node is selected for each fragment, the uneven fragment sizes can significantly increase load imbalance. Dividing fragments also helps to reduce delay. By dividing a large frame into smaller pieces, multiple nodes can be used to relay the frame in parallel and enjoy a higher aggregated upload capacity, thereby reducing the latency. If the same weak node is selected as a relayer, it only needs to deliver a piece instead of an entire frame to all the destinations. Since the piece is much smaller than the entire frame, the weak node can deliver the piece to all the destinations within the deadline.

3.1.3 Handling Heterogeneous Delay

Random relay selection works well when the delay between different nodes are homogeneous. When the delay between some nodes are large (*e.g.*,

conferencing between US and China), picking random two-hop paths may result in large delay (*e.g.*, a path from a node in US to a node in China and back to another node in US can be too long).

Therefore we cluster nodes according to their pairwise delay. We iteratively add a node to a cluster as long as its delay to other nodes in the cluster satisfies the deadline constraint. Otherwise, we create a new cluster involving the node.

Suppose there are E clusters. For each cluster that has at least one subscriber to the source, the source selects a relay node from the cluster according to the upload capacity of all nodes in the cluster and the portion of upload capacity of the source dedicated for the cluster. That is, the probability of selecting node i (including the source) as a relay node for the cluster is $\frac{C_i}{C_s+\sum_{j\in cluster} C_j}$, where C_s is the portion of upload capacity source dedicates for the cluster. We set this portion proportional to the number of subscribers in each cluster.

3.1.4 Utilizing Peers in Other Sessions and CDN Nodes

Our Valiant multicast routing can be easily extended to leverage resources from peers in other sessions and CDN nodes. In particular, a traffic source, denoted as S, constructs a full mesh spanning over all its subscribers and itself. If the total capacity from all the nodes in the mesh cannot sustain S's streaming rate, S adds helper nodes to the mesh until the total capacity meets the minimum requirement, where helper nodes include (i) other non-

subscriber nodes in the same video conferencing session, (ii) nodes in other video conferencing sessions, and (iii) CDN nodes. We add them in the order of (i), (ii), and (iii) to avoid unnecessarily asking help from nodes in other sessions or CDN nodes.

Figure 3.1 shows the pseudo code to construct the candidate pull, where $idle(P)$ indicates the idle capacity of a set of nodes in the mesh P and is estimated as the difference between the total upload capacity (as measured in Section 3.2) and the load, T is a threshold used to determine if there is enough idle capacity, and $n(A_i)$ is the number of nodes in the set A_i.

```
A0 = { peers w/ same session id and subscribed to source};
A1 = { peers w/ same session id but not subscribed to source};
A2 = { peers w/ different session id};
A3 = { CDN nodes};
P = { source node };
add all nodes in A0 to P ;
while (Idle(P) < T and n(A1) > 0)
{
   x = one random peer from A1
   add x to P
   remove x from A1
}
while (Idle(P) < T and n(A2) > 0)
{
   x = one random peer from A2
   add x to P
   remove x from A2
}
while (Idle(P) < T)
{
   add the closest CDN node from the source to P
}
```

Figure 3.1: Pseudo code of using other helper nodes.

3.1.5 Adapting Streaming Rate

In Section 3.1.4, we add enough nodes to ensure the total capacity is sufficient to stream the current video session. However, it is possible that we may not always have enough capacity even after utilizing all resources. In this case, we will have to reduce the fidelity of the video streams. Our goal is to reduce just enough to satisfy the capacity constraints.

We develop a simple and effective scheme to adapt video rates. Each source monitors its delivery rates to all its subscribers. If the delivery rate is low (*e.g.*, below 90%), it reduces the video rate to the current delivery rate. For example, if its delivery rate is 70%, it reduces the streaming rate to 70% of the current streaming rate. If the delivery rate is high enough (*e.g.*, above 95%) it tries to increase the video rate by a fraction of the idle capacity of its relay nodes. It does not completely consume all the available idle capacity because other nodes may also attempt to use some of the idle capacity for their streams. In addition, there may be an estimation error in the idle capacity, so it is better to be conservative.

We have the following requirements. First, it is undesirable for multiple clients to change the rate at the same time to avoid an abrupt change in total demand of throughput. Second, we want to avoid frequent fluctuations (*e.g.*, increase the rate; find capacity is insufficient and decrease the rate; and then increase again). Third, we prefer to choose the client that increases the streaming rate in the current round to decrease in the next round if necessary. To satisfy the first requirement, each source uses a random timer drawn from

a uniform distribution. The first one that makes the change will announce it, and the other nodes upon receiving the notification perform backoff for a random amount of time before changing their rates. To satisfy the second requirement, when a node increases its capacity unsuccessfully (i.e., the new delivery rate after the rate increase is low), the node doubles its backoff time so that the next increase needs to wait for a longer time. To achieve the third goal, if a node increases its streaming rate, it takes a shorter backoff time than others so that it can revert back to the previous rate before other peers do. Figure 3.2 shows the pseudo code.

```
if (deliveryRate < 0.9)
    targetStreamRate = meanStreamRate     deliveryRate;
else if (deliveryRate > 0.95)
    totalIdleCap = totalCap − currLoad;
    targetStreamRate =
      meanStreamRate + totalIdleCap / (numSubscribers + 1) × meanStreamRate / peakStreamRate
else
    targetStreamRate = meanStreamRate;
```

Figure 3.2: Pseudo code of streaming rate adaptation.

Note in practice it is not possible to adjust the bit rate precisely to the target rate. For example, in Flash, we can set maximum possible bandwidth, key frame frequency, and the quality factor as an integer value that ranges from 0 to 100, but cannot directly control the average sending rate. Therefore, we use the nearest possible setting whose maximum rate does not exceed the target streaming rate.

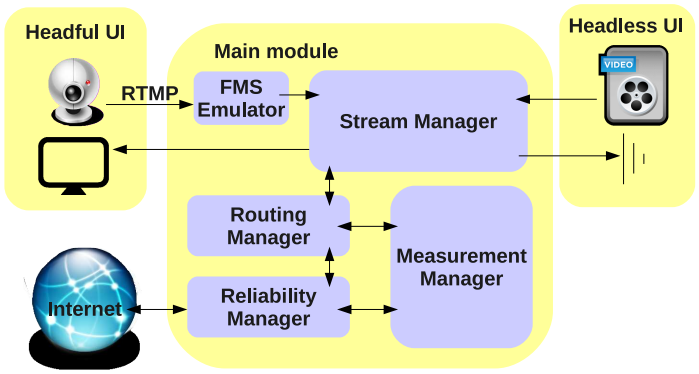


Figure 3.3: System architecture.

3.2 Implementation

We implement our prototype client in Java and Adobe Flash using around 25K lines of code. Flash is highly portable and platform independent. Flash provides a convenient generic interface to capture input data from a camera and a microphone. For live stream publishing, Flash supports one codec for video (Sorenson Spark) and two codecs for audio (NellyMoser and Speex), all of which are proprietary. For playback, Flash supports a variety of codecs and a container format called Flash Video is used for both stream publishing and playback. Flash video is considered the default online video format, and has been widely used by many content providers, such as YouTube [132], Hulu [58], NBC [91]. We also port our prototype to Android and run on a mobile device by embedding the headless UI module inside the system and disabling the reliability manager to reduce CPU overhead, while keeping the remaining components the same as the non-mobile version. Figure 3.3 shows the architecture of our system. Below we describe each component.

User Interface (UI) Module: A client captures input stream data, and plays back both its own stream and streams from other peers. Captured stream data is transmitted to the FMS emulator, and the FMS emulator can transmit multiple streams simultaneously. We implement two kinds of UI: (i) a "headful" client UI that interacts with an actual camera and a microphone, and (ii) a "headless" client UI that uses a pre-recorded file for large-scale experiments without GUI interface.

Headful client UI is implemented as an Adobe SWF file in ActionScript3 that runs in a browser. It captures video and audio from a camera and a microphone using Flash's Camera and Microphone objects, and transmits it to our FMS emulator using RTMP protocol. Headless client UI is implemented in Java that functions similarly to "headful" version except that it plays a pre-recorded file instead of streaming live captured data. Headless client UI does not need FMS emulator as it has direct access to the streaming data. Headless client UI loads and parses a FLV file and emulates a real client by releasing video and audio tags at the right pace to the FMS emulator. The headless UI receives incoming streams from the stream manager just as the headful UI does but without displaying the content. For convenience, our evaluation uses headless client UI.

FMS emulator: Flash does not support extracting the captured video and audio stream data directly and only allows it to be streamed to the Flash Media Server(FMS). We implement a simple FMS emulator that communicates with our headful client UI. This component functions as a simplified

FMS server by performing initial hand shaking with the headful client UI and receiving video and audio stream data from the UI over RTMP. FMS emulator then converts it into byte stream formatted in FLV and feeds it to the stream manager.

Stream manager: Stream manager consists of the reconstruction module and the fragmentation module. The reconstruction module reassembles video and audio tags in order, removes duplicates and late tags, and releases tags at an appropriate rate according to the timestamp of each tag. The fragmentation module fragments and defragments FLV tags at the application layer to improve load balancing and reduce delay (Section 3.1.2). To minimize overhead, we utilize an unused, 3-byte long "stream id" field in FLV for fragmentation. This field is used to record the fragment index as well as mark the last fragment of a single original FLV tag.

When stream data from a remote peer arrives, it is first processed by the fragmentation module for defragmentation. Once the original FLV tag bytes are defragmented, they are passed to the reconstruction module. Conversely local stream tags are sent to the reconstruction module, then fragmented by the fragmentation module, and finally released to the route manager for publication.

Route manager: We use application-layer routing and implement several routing algorithms, such as our Valiant multicast routing, broadcast trees, minimum spanning trees, and shortest path trees. Refer to Section 3.3.1 for details. All these routing schemes obtain the topology and subscription

information from the measurement module as input to construct routes.

Reliability manager: We use retransmissions to protect video tags and some of the control messages, and use FEC code to protect audio tags. Retransmission is simplified and minimized in our implementation compared to TCP due to time critical nature of stream data. We assume our system works in a resource-tight environment, so we prefer first-time transmissions over retransmissions, limit the number of retransmissions to one, and dynamically disable/enable retransmissions according to peer's resource utilization.

We use FEC encoding for audio messages because (i) audio messages are much smaller in size (*e.g.*, typically less than 200 bytes) and piggybacking FEC redundancy is much more efficient than retransmitting the entire messages and (ii) audio has more stringent delay constraints and FEC recovery is faster than retransmissions. We use Reed Solomon code [105] as our FEC code. For every k data blocks, FEC code will generate $(n - k)$ redundant blocks. Our implementation uses $n = 6$ and $k = 5$.

Measurement manager: Measurement manager measures and collects information from both network and application layers. At the network layer, it measures delay and bandwidth. At the application layer, it collects the peer subscription information and traffic statistics (*e.g.*, which peers subscribe to which other peers, number of bytes sent and received in various categories such as first-time transmissions and retransmissions, stream data or control data, and local data or relayed data). Most information is easy to collect, so below we only describe bandwidth measurement.

We measure network bandwidth using one-way delay. The approach is simple, light-weight, and requires no additional probing traffic other than existing control and data packets. Moreover, it can derive estimate using only a few data packets, and its accuracy improves over time. We further improve accuracy by aggregating estimation from multiple peers. Its accuracy is high especially for networks with tight bandwidth. Like TCP Vegas [24], it may under-estimate high bandwidth paths (*e.g.*, with Gbps). This is acceptable in our context since we do not have to use up all the link capacity.

Specifically, upon receiving a UDP message, a peer records the difference between the current local timestamp and the transmitter's timestamp in the message, and updates the minimum one-way delay between the two nodes and the corresponding size. Let t_{local} and t_m denote the local timestamp and the timestamp in the message, $minDiff$ denote the minimum one-way delay seen so far, $minSize$ denote the size of the message that corresponds to $minDiff$. $minDiff$ includes both propagation delay and clock difference between the two nodes. It estimates the bandwidth as

$$if\ (size > thresholdSize)$$
$$estBW = (size - minSize)/((t_{local} - t_m) - minDiff));$$

where $thresholdSize$ is set to 500 bytes in our evaluation.

Periodically each client reports measured estimated bandwidth back to the peer. It takes the maximum of bandwidth estimations reported by all peers as the estimated upload bandwidth, and announces it to the other peers.

We also allow a client to invalidate its current estimation whenever it notices network congestion (*e.g.*, a write() call fails), and informs all remote peers to immediately invalidate as well.

3.3 Performance Evaluation

3.3.1 Evaluation Methodology

Routing schemes: We compare the following schemes:

- Broadcast trees (BT): Each traffic source directly forwards its stream to all its subscribers.
- Minimum spanning tree (MST): Each traffic source constructs a minimum spanning tree involving only the nodes that subscribe to the stream.
- Shortest path tree (SPT): Each source sends traffic along the shortest path to all its subscribers. The traffic on the shared branch to multiple receivers is sent only once.
- Valiant multicast routing (VMR): We evaluate several versions of Valiant multicast routing. The first version (VMR-1 choice) simply chooses the relay node according to every node's upload capacity. The other versions (VMR-2 choices) randomly choose two relay nodes according to upload capacity and use the one with less load. VMR-2 choices include (i) robust VMR-2 choices, as described in Section 3.1.2, which explicitly captures the uncertainty in the load estimation, (ii) VMR-2

choices/previous estimates the load using the previously reported load, (iii) VMR-2 choices/EWMA estimates the load using exponential weighted moving average (EWMA) with 1/8 as the weight of a new sample as in TCP, and (iv) VMR-2 choices/Holt-Winter estimates the load using the Holt-Winters algorithm [63], a well known predictor that is more effective for dynamic time-series. The schemes (ii), (iii), and (iv) represent commonly used predictors. They aim to accurately predict the load and do not consider possible prediction errors. As we will show, the load is hard to predict accurately, so it is important to explicitly deal with errors, as in the robust VMR-2 choices.

Evaluation settings: We conduct our evaluation using machines in our department. We emulate the wide-area network delay and bandwidth by limiting its upload bandwidth and injecting additional delay to each link.

Unless otherwise stated, we randomly choose the bandwidth of a node from 10Mbps, 5Mbps, 2Mbps, 768Kbps and 512Kbps, and scale it to control the total available bandwidth in our experiments. The total bandwidth is scaled to 0.7 to 1 times the peak total streaming rate. We set pairwise delay uniform randomly from 0ms to 50ms.

A video conference session consists of 12 nodes in our experiment. We generate subscription by letting a subset of nodes to be subscribed by everyone. We vary the fraction of these speakers from 25% to 100% to capture that in real world conferences there are often a few frequent speakers and everyone

else usually listens.

Performance metrics: We use the loss rate as the performance metric. It is defined as the fraction of video tags that misses the deadline and the deadline is set to 400 ms. Moreover, when we evaluate the effectiveness of our approach using helpers (*e.g.*, other peers and CDN nodes), in addition to the loss rate, we also report the amount of traffic served by the helpers to ensure we do not over-utilize the helpers' resources. For each experiment setting, we run 5 times and report the average.

3.3.2 Performance Results

3.3.2.1 Comparison of Routing Schemes

Varying bandwidth: First we compare the performance of different routing schemes by varying the amount of total network resources (i.e., total upload capacity of clients). We vary the total available bandwidth in the system from 0.7 to 1.3, and fix the subscription rate to 0.25 (i.e., 3 clients are subscribed by all other clients in a 12-node video conference).

Figure 3.4 summarizes the results. We make four observations. First, VMR with 2 choices consistently performs much better than the other routing schemes. For example, in terms of loss rate reduction, VMR-2/Robust outperforms the broadcast tree (BT) by 75%-96%, minimum spanning tree (MST) by 63%-97%, and shortest path tree (SPT) by 62%-97%. The corresponding performance improvement of VMR-2/previous over BT, MST, and SPT are 52%-94%, 41%-96%, and 39%-95%, respectively. Similarly for VMR-2/EWMA

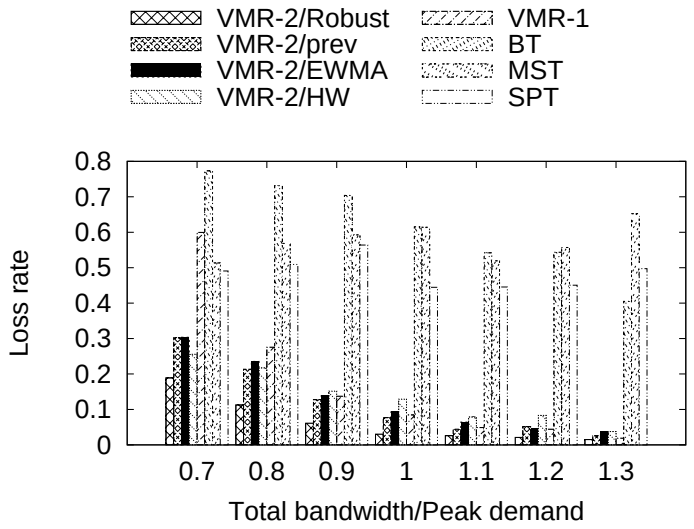


Figure 3.4: Loss rate of different multicast routing schemes under varying bandwidth (400 ms deadline).

and VMR-2/HW. The BT relies on a source to distribute its video to all its subscribers, so it is severely bottlenecked by the source's upload capacity. MST and SPT as well as other tree based schemes tend to cause congestion near the sources and incur high loss rate. In comparison, VMR balances load across all nodes to more efficiently utilize available resources and achieve much lower loss rates.

Second, VMR with 2 choices outperforms 1 choice. It is well known that Valiant load balancing with 2 choices is better than 1 choice. In our case, there is another reason that further increases the gap between the two versions. The source advertises its raw capacity minus the capacity reserved for sending its own stream as the capacity used for relay (i.e., advertised bandwidth is $Cap_{raw} - Cap_{reserve}$). Subtracting $Cap_{reserve}$ is necessary because the source

needs to transmit its own stream on the first hop as well as relay on the second hop while non-source nodes only need to relay on the second hop. However, due to variable video rates, there is error in $Cap_{reserve}$ estimation, which introduces error to the advertised capacity and in turn causes error to the derived dispatching ratio of relay requests to each node. For example, due to an increase in its own video rate, a source under-estimates $Cap_{reserve}$ and advertises a higher capacity than it has for relay, which will attract too many relay requests and cause congestion. In comparison, with 2 choices, even though we may pick such a node more often than it should in the first step of random selection (due to erroneous dispatching ratio), it is not actually used in the end due to its high load, thereby avoiding congestion. Therefore 2 choices can not only balance the load more evenly when capacity is known accurately (as shown in the previous works [13]), but also more robust to capacity estimation error.

Third, among different versions of VMR with 2 choices, our VMR-2/Robust performs the best. It outperforms VMR-2/prev by 40%-58%, VMR-2/EWMA by 37%-68%, VMR-2/HW by 26%-87%. The benefit is largest when the resource constraints are tight, because when we have enough resources, slight load imbalance are less likely to cause congestion. The other versions of VMR-2 choices do not perform as well because they treat the predicted load as completely accurate, but in reality prediction errors are inevitable regardless how hard we try, and such errors can degrade the effectiveness of load balancing. By explicitly taking into account of the prediction error,

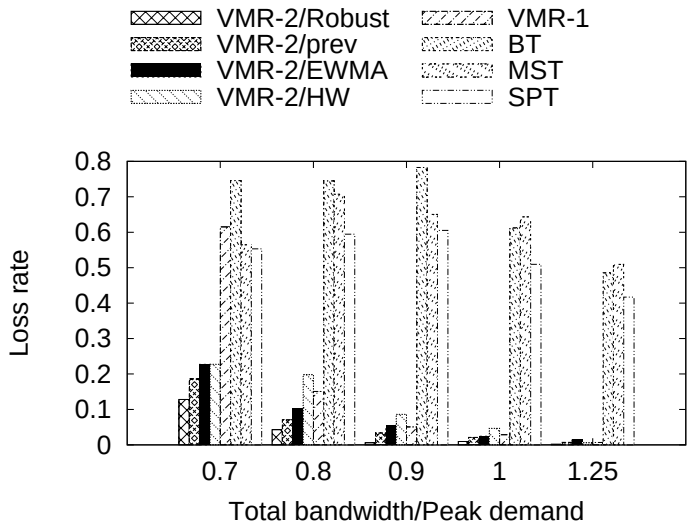


Figure 3.5: Loss rate of different multicast routing schemes under varying bandwidth (1 second deadline).

VMR-2/Robust yields better performance.

Fourth, the loss rate of even VMR-2/Robust is non-zero due to very tight deadline of 400 ms. A temporary surge in video rates can cause significant queue built-up and increase the total delay beyond 400 ms. To verify this, we also run experiments using 1 second deadline. As shown in Figure 3.5, the trend is similar but the loss rate of VMR schemes are even lower. VMR-2/Robust achieves 0.3% and 0.2% loss rate, respectively, under the bandwidth ratio of 1 and 1.25.

Furthermore, we test the normal distribution assumption used in drawing random samples in VMR-2/Robust. Figure 3.6 shows the qq-plot of the load of nodes fit very well with a straight line, indicating that the body of the distribution closely follows a normal distribution. The bottom-left tail of

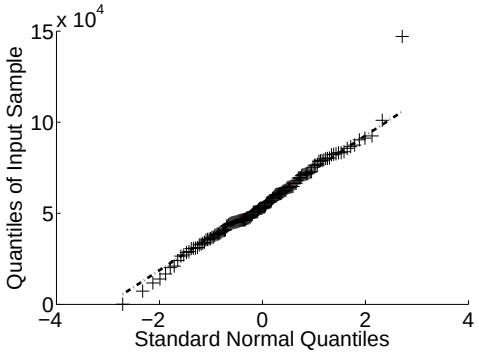


Figure 3.6: qqplot of load: the points fit well with a straight line, indicating they follow a normal distribution.

the distribution deviates more from the normal distribution, because the load distribution is truncated (i.e., no less than 0). The load from other nodes look similar and is omitted in the interest of brevity. In the future, we plan to better model the load distribution using a truncated Gaussian distribution. As described in Section 3.1, our approach can support any general distributions by sampling the load from the distribution obtained from the past measurements.

Varying subscriptions: Next we vary the subscription ratio from 0.25 to 1 (i.e., 3 to 12 frequent speakers subscribed by everyone), while the total bandwidth across all clients versus the peak demand is fixed to 1. As shown in Figure 3.7, the relative performance across different schemes is similar to the above. In particular, VMR-2/Robust continues to perform the best, and VMR based schemes performs much better than the tree based approaches. As the number of subscriptions increases, the loss rate of VMR-2/Robust remains low whereas the loss rates of tree based approaches are much higher with no clear trend because they are more sensitive to specific network topologies.

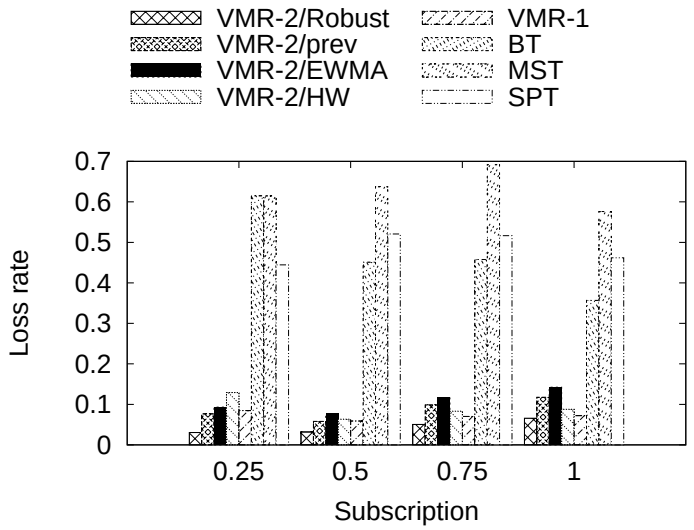


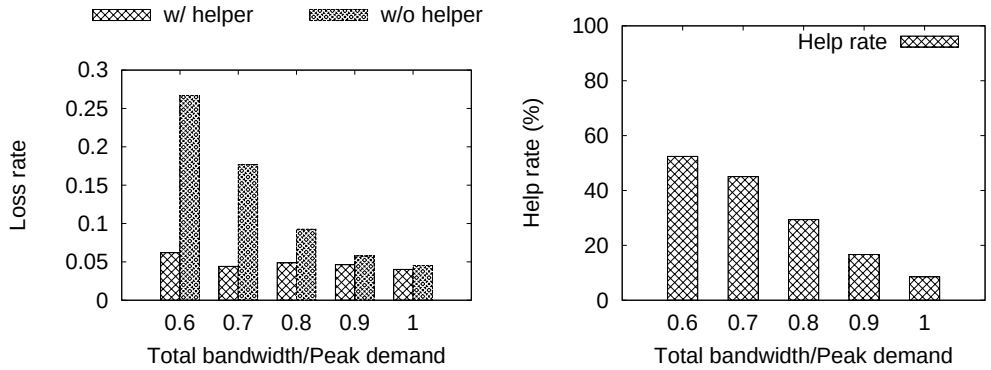
Figure 3.7: Loss rate of different multicast routing schemes as subscriptions vary.

3.3.2.2 Benefits of Helpers

When a session does not have enough bandwidth to support the conference, we obtain help from peers in other sessions and/or CDN nodes. Ideally we want the helper to cover the resource deficit, which is defined as the total demand minus the total available resources within the session. It is important that we do not over-use the helpers and only use them whenever necessary to cover the resource deficit. Therefore, we consider two performance metrics: the loss rate (as before) and how much traffic is carried by helpers.

CDN helper: In this experiment, we have one video conference session and a CDN node with high bandwidth. As shown in Figure 3.8(a), using the CDN significantly reduces the loss rate. The improvement is specially large when there is a large resource deficit. For example, when the bandwidth to

peak rate ratio is 0.6, the CDN helper reduces the loss rate from 27% to 6%. Moreover, Figure 3.8(b) shows the fraction of traffic going through the CDN helper. When the bandwidth to peak rate ratio is 0.6, 52% traffic goes through the CDN helper, whereas only 8% goes through CDN when the bandwidth to peak rate ratio increases to 100%. In the latter case, we still use a small amount of CDN resource due to our conservative idle capacity estimation. We can further reduce the CDN consumption by improving the estimation accuracy.



(a) Benefit of CDN helper (b) CDN help rate

Figure 3.8: Benefits of the CDN helper.

Peer helpers: Next we evaluate two sessions, where one session with insufficient resources (i.e., bandwidth to peak rate ratio is 0.7) and the other session with spare resources (i.e., the ratio is 6), and the two sessions together have enough resources to support their total demands.

The use of peer helpers improves the average loss rate of the weak session from 15% to 6%, while the strong session continues to enjoy the low

loss rate 5%. Meanwhile, we do not over-utilize the helpers: 53.4% traffic is carried by helpers, which is close to its resource deficit.

Both CDN and peer helper: Finally, we consider two sessions plus a CDN node, where one session has insufficient resources (i.e., bandwidth to peak rate ratio is 0.6) and the other has spare resources (i.e., the ratio is 1.1), but have insufficient resources combined to support the total demands. (So we need the CDN node to cover the remaining resource deficit.) By effectively utilizing both peer helpers and CDN helper, we achieve low average loss rates: 5% and 1.5%, respectively.

3.3.2.3 Handling Clients with Heterogeneous Delay

In this experiment, we evaluate the cases where clients have heterogeneous delay. We assign the delay between nodes as follow. We first assign nodes into different clusters; for two nodes within the same cluster, we set their delay by randomly drawing a sample from a uniform distribution between 0 and 50ms; for nodes belonging to different clusters, we assign their delay by randomly drawing a sample from a uniform distribution between 100 and 150ms. We vary the number of clusters from 1 to 4, and set the bandwidth to peak demand ratio to 1.

As shown in Figure 3.9, clustering significantly improves the loss rate by avoiding traversing long links twice. For example, clustering helps to reduce the loss rate from 13% to 7% with 2 clusters, and from 16% to 9.5% with 4 clusters. Note that as the client distribution forms more number of clusters

the loss rate grows higher because load is balanced within a cluster but not necessarily across clusters.

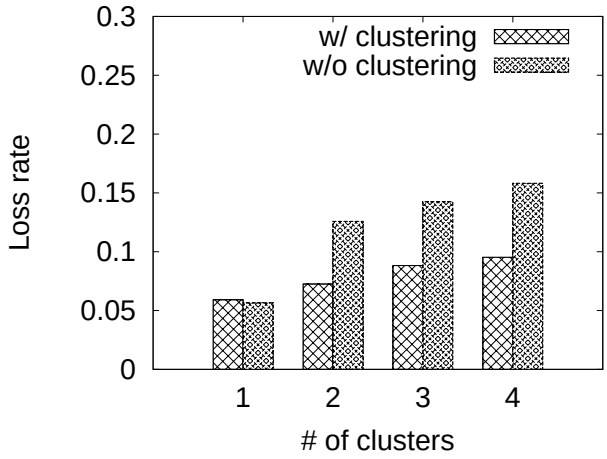
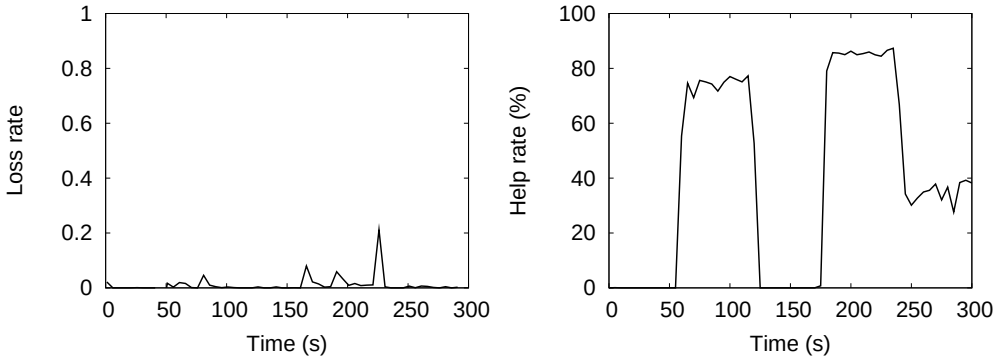


Figure 3.9: Benefit of clustering under heterogeneous delay.



(a) Loss rate (b) Help rate

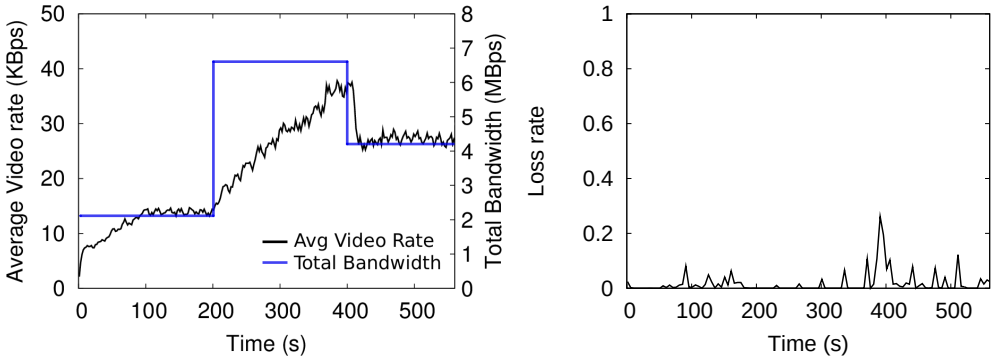
Figure 3.10: Impact of dynamic subscription.

3.3.2.4 Handling Dynamic Subscription Changes

A major advantage of VMR is its ability to quickly adapt to changing network conditions. In this experiment, we study the impact of dynamic

subscription changes, which is common for real video conferences as active speakers change from time to time.

We generate dynamic subscriptions by letting every client become an active speaker for 30 seconds at least once throughout the experiment and varying the number of simultaneous speakers from time to time. We also add one CDN helper and increase the total number of active speakers beyond what the session can support twice in the middle of the experiment. As shown in Figure 3.10(a), the loss rate is mostly close to 0, increases when the number of subscription increases, and quickly decreases close to 0 by leveraging the CDN to cover the resource deficit. Figure 3.10(b) shows the fraction of traffic going through the CDN helper, and confirms that our approach utilizes CDN helper's resource only when necessary.



(a) Bandwidth and average video rate (b) loss rate

Figure 3.11: Performance of streaming rate adaptation under varying bandwidth.

3.3.2.5 Streaming Rate Adaptation

Next we evaluate our streaming rate adaptation mechanism. We do that by increasing or decreasing the total upload bandwidth of the entire system, and record the delivery ratio and average video rates used. Figure 3.11 shows the result from one experiment, and the results from other experiments are similar and omitted in the interest of brevity. Figure 3.11(a) plots the total bandwidth and average video rate over time and Figure 3.11(b) shows the corresponding loss rate. As they show, our system responds quickly to bandwidth drops. Throughout the experiment, the average loss rate is 1.9%, and the only increase in loss rate occurred at the time when the bandwidth drops and it only lasted less than 20s and quickly decreases to 1.5% by reducing the streaming rate.

3.3.2.6 Performance on Mobile Devices

Finally we evaluate the performance on a mobile device. We use a Samsung Galaxy Player 4.0 device that runs on Android 2.3.5 with a 1GHz single-core Hummingbird CPU and vary the number of nodes subscribed to the mobile node, while letting the mobile node subscribe to 2 streams. We cap the streaming rate at 120Kbps due to the limited screen resolution of mobile devices. Figure 3.12 shows the loss rate of the mobile node's outgoing stream. Loss rates for all other outgoing streams are negligible and are not shown. We observe all the streams experience low loss rate (below 1%) when the number of out-going streams from the mobile is below 6. Afterwards, the loss rate of

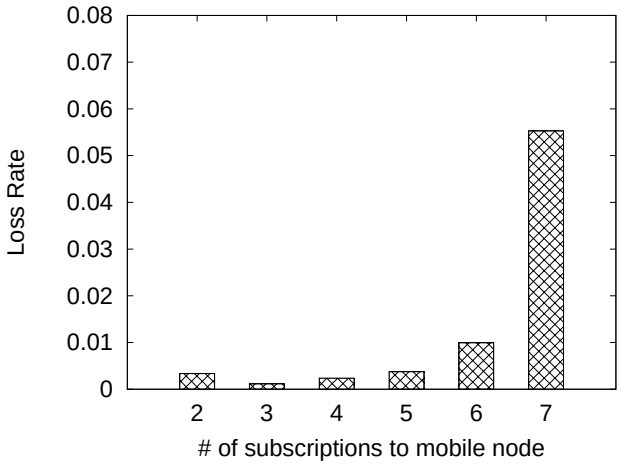


Figure 3.12: Loss rates of a mobile device's out-going streams.

the out-going stream increases due to CPU overhead. We expect this number to improve with further code optimization.

3.4 Related Work

Commercial products: Many commercial products provide audio or video conferencing. Due to their propriety nature, the available description typically describes the feature-set and not the techniques. So our discussion is based on our understanding of the public information about these products. Skype is one of the most popular conferencing softwares with 663 million registered users in Sept. 2011. However, the quality of two party video call is unsatisfactory from time to time. The performance of multi-party call is even worse due to a higher traffic demand, and its performance also degrades significantly for international calls. QQ is the most popular conferencing tool in China. Despite a large user population, it only supports low resolution video.

On the other end, Cisco WebEx technology offers high-end video conferencing by deploying a dedicated network of high-speed meeting switches and data centers across the world and having Cisco personnel provide 24x7 operational support. Despite its high cost, the use of server may introduce a long delay and result in poor performance [78]. Google+ Hangouts allows up to ten participants in a multi-way video call, but its purely server-based solution requires all participants to have good network connectivity to the servers in order to have good performance, which is not always possible.

Research work: Internet video conferencing and overlay multicast have both attracted lots of research.

[56] proposes an end system multicast protocol, called Narada. It overcomes the deployment issue of IP multicast by constructing overlay spanning trees. [60] builds a reliable multicast for content distribution in a large scale. [19] design a scalable application-layer multicast for low-bandwidth data streaming with many receivers. All the above schemes target a single source, so it is not applicable to video conferencing. [70, 127] propose different variants of application layer multicast for P2P video conferencing. [70] focuses on fairness, while [127] only supports audio. [69] casts the bandwidth sharing problem in multi-swarm multi-party P2P as a utility-maximization problem and develops distributed algorithms for intra-swarm and inter-swarm bandwidth allocation. [33] uses delay-bounded 2-layer trees for multicast routing and develops a distributed optimization algorithm for rate adaptation. However these optimization based approaches work well only on static subscription

while our Valiant based approach can better support dynamic subscriptions.

Valiant load balancing: Valiant load balancing has been widely used in many areas due to its simplicity and robustness under variable traffic. It is formulated as balls and bins problem, where bins represent servers and balls represent jobs. [13] proves the power of d choices and shows the maximum load is $log(log(n))/log(d) + \theta(1)$ with a high probability for uniform bins and balls. [22] extends to non-uniform bins, [114] extends to non-uniform balls. [85, 86, 40] point out the challenge of applying the power of two choices in practice due to stale load information, and [87] recognizes this as an important open problem, which we address.

Chapter 4

C3: Constancy-aware, Coordinated, Cross-layer Video Rate Adaptation for Mobile Networks

Background and motivation: A recent trend of online video streaming is Dynamic Adaptive Streaming over HTTP (DASH). It is an adaptive bit-rate streaming technology, where a video is fragmented into multiple small video chunks, each of which is encoded into multiple versions at different rates. Depending on the network condition, a client can dynamically switch rates for different video chucks. DASH became an International standard in 2011 and has been widely adopted by many large streaming providers, such as NetFlix, Microsoft, Apple, and Adobe.

The standard specifies the protocol design, but leaves the detailed adaptation algorithm to implementers. However, a number of recent measurement studies show serious weaknesses in the rate adaptation schemes implemented by today's commercial video players. While several new video rate adaptation schemes have been proposed, all the works so far focus on video rate adaptation in wireline networks. Wireless mobile networks pose significant new challenges to video rate adaptation. In particular, due to frequent wireless link fluctuation and mobility, wireless network condition is much more dy-

namic than the wireline network condition. All the existing rate adaptation schemes implicitly assume past network performance is a good indicator of the future performance and pick the video rate to closely track the network performance. However, such an assumption may not hold for wireless mobile networks. Unpredictable outages and network fluctuation are common in wireless mobile networks; and wireless interference and interactions among multiple wireless clients further complicate the problem. These challenges call for a new resilient video rate adaptation solution for wireless mobile networks to better protect against unexpected network changes.

Our approach: In this chapter, we propose C3, a *constancy-aware*, *coordinated*, *cross-layer* video rate adaptation scheme for wireless mobile networks.

1) *Constancy awareness.* We believe that in mobile networks it is crucial to explicitly cope with the constancy or the lack of constancy in network performance. By "constancy", we mean "in some fundamental sense the quantities measured are not changing" [133]. We collect and analyze a range of wireless traces under different mobility. These traces exhibit two distinct constancy patterns: (i) stable periods when the network performance can be predicted with reasonable accuracy, and (ii) unstable periods when the network performance is completely unpredictable. These characteristics prompt us to first automatically classify the current network condition into one of these two periods and then develop different rate adaptation schemes tailored to each.

To classify the current network condition, we detect change points online based on the network performance observed so far, and then classify the current condition into stable and unstable periods based on the gap between recent change points. Next we design a simple yet effective video rate adaptation algorithm for stable periods to maximize the amount of time that the client can stay before completely filling/draining the buffer by effectively utilizing the predicted network performance. To handle unstable periods, we develop a novel two-rate download scheme, which first fills the buffer with the lowest video rate to prevent outage. To avoid being overly conservative and under-utilizing network resources, we replace the downloaded content with a higher rate starting from the end of the buffer. In this way, we can minimize the chance that video is blocked due to buffer empty while effectively utilizing the network bandwidth.

2) *Coordination.* We further extend our approach to support coordination among multiple clients. In particular, to prevent clients from blindly competing against each other, which leads to instability, unfairness, and under-utilization, we develop an approach to jointly optimize the video quality across all clients. This not only allows us to use the aggregate bandwidth prediction as input, which is more stable and easier to predict than the bandwidth of individual video flow under competition, but also allows different flows to more effectively share the common network resources.

3) *Cross-layer adaptation.* Finally, we observe that the effectiveness of video rate adaptation depends on the lower layer design. In particular,

we investigate the interaction between video rate adaptation and MAC rate adaptation. We propose an effective enhancement to the existing MAC rate adaptation by utilizing the signal-to-noise (SNR) information not only in the preamble but also across all the data symbols in the frame. Our results show that this enhancement leads to more accurate prediction of future channel condition and a better MAC rate adaptation based on it. The benefit of this enhanced MAC rate adaptation also directly translates into improvements in the video quality.

Contributions: We make the following main contributions:

- Develop a scheme to automatically classify the current network conditions into stable and unstable periods (Section 4.1);
- Develop video rate adaptation schemes that are tailored to either stable or unstable periods to minimize playback freezes while maximizing the video rate and stability (Section 4.2);
- Design an optimization approach to maximize the video quality across multiple flows (Section 4.3);
- Enhance the MAC rate adaptation scheme by exploiting the SNR information across all the data symbols and integrate it with video rate adaptation (Section 4.4);
- Implement and evaluate the performance of our approaches using mobile

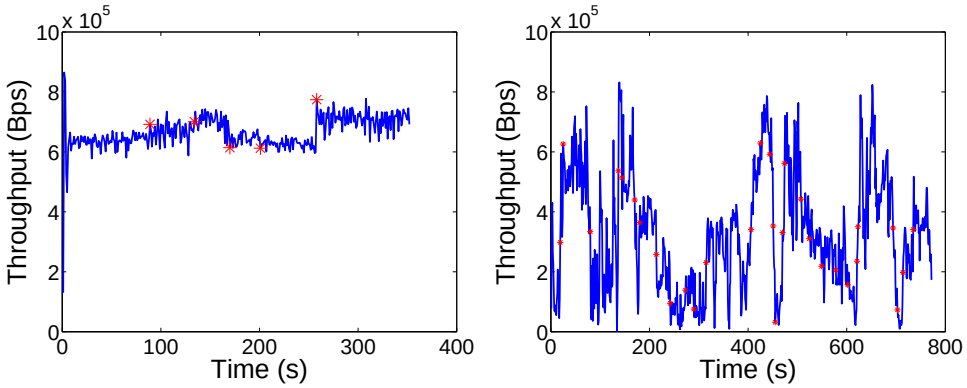
wireless networks under different mobility to demonstrate their effectiveness (Sections 4.5 and 4.6).

4.1 Mobile Constancy Analysis and Change Detection

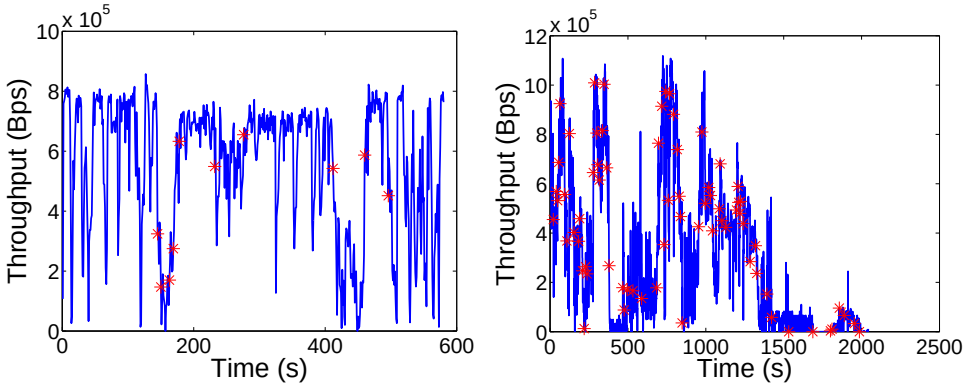
4.1.1 Mobile Trace Analysis

We collect a wide range of wireless traces under different mobility to analyze the constancy of network performance in mobile networks. By "constancy", we mean "in some fundamental sense the quantities measured are not changing" [133]. The level of constancy in network performance has significant impact on the effectiveness of different video rate adaptation schemes. To collect 3G traces, we connect a laptop to a 3G network capable phone using USB cable as a 3G modem. The laptop downloads a large file using FTP through 3G network and packets are captured using tcpdump. To collect WiFi traces, we connect the built-in wireless card in the laptop to a 802.11g router and use the same method to measure the TCP throughput.

Figure 4.1 shows the TCP throughput of 4 traces described in Table 4.1. We will explain about red dots later in this section. Figure 4.1 (a) shows the TCP throughput to a static device at night which is relatively stable due to no mobility and low network congestion. Figure 4.1 (b) shows the 3G throughput collected while walking and riding a bus in the daytime. Figure 4.1 (c) shows the throughput collected while driving at night that shows wider and longer fluctionations due to high mobility. Figure 4.1 (d) shows the throughput collected while walking and driving in the daytime. The throughput fluctuates



(a) Trace 1: *stable1* (b) Trace 2: *unstable1*



(c) Trace 3: *commute1* (d) Trace 4: *commute2*

Figure 4.1: TCP throughput in mobile environment and change point detection.

very widely in a highly unpredictable manner with intermittent periods where the throughput reaches close to zero due both to mobility and congestion.

These traces are some of the many traces we collect and analyze. In general, all the traces we analyze show two distinct constancy patterns: *stable periods* when the bandwidth varies in a small range and is relatively predictable, and *unstable periods* when the bandwidth varies significantly and is highly unpredictable. Such patterns are common in the mobile environment

due to user mobility, wireless interference, and congestion.

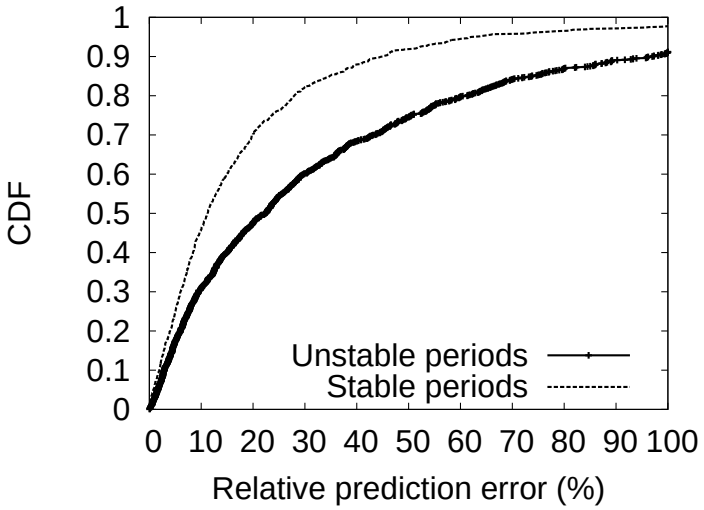


Figure 4.2: CDF of prediction errors in stable and unstable periods.

Any video rate adaptation scheme needs to predict the network throughput in the near future. We observe that prediction accuracy depends significantly on the types of periods the trace is in. Figure 4.2 shows that 80% of the prediction error is less than 26% in stable periods while the prediction error could be as high as 59% in unstable periods. Unlike in stable periods where we can expect high predictability in network throughput, in unstable periods the impact of mobility, network congestion, and the combination of both result in high prediction error.

Remarks: Existing video rate adaptation schemes do not differentiate between stable and unstable periods. They implicitly assume that the past performance is a reliable predictor for the future and apply a single algorithm to both constancy patterns. However, in mobile wireless networks, unstable and unpredictable periods are common, and a single algorithm is unlikely to

work well in both periods. When the network performance is stable and predictable, we can take advantage of the predicted performance to explicitly optimize video quality. On the other hand, when the network performance becomes unpredictable, such optimization is not effective and a more conservative approach to prepare for unforeseen bandwidth drop is necessary to avoid freezes. This observation motivates our design of a new video rate adaptation scheme.

4.1.2 Detecting Stable and Unstable Periods

To determine whether it is a stable period or unstable period, we first detect change points in the trace, and then determine the type of the period based on the gap between the current and previous change point. A small gap indicates an unstable period while a larger gap indicates a stable period.

Change point detection: One way to detect change points is to partition the trace into multiple segments and compute variance for each time window. However, variance is only meaningful when the window size is large enough, which makes too late to detect changes. Instead we use the following CUSUM based change point detection [81] that can quickly respond to changes:

- Compute the average of the original time series T.
- Subtract the average from all values of the time series and get T'.
- Calculate the cumulated sum and get a new time series S where $s_i = \sum_{j=0}^{i} t'_j$ and t'_j is the j-th value of T'.

- The point where the absolute value of the cumulative sum reaches its maximum is the candidate change point. The difference between the maximum cumulated sum and the minimum cumulated sum is the score of this change point.

- To test the statistical significance of the change point, the raw time series is permutated randomly for 1000 times, and steps 2-4 are repeated on the permutated time series. We compare the scores of the change points found in the permutated trace with the original score. If 99.5% of them are smaller than the original score, we claim the change point we find is significant.

- Recursively call change point detection to the segment between two adjacent change points until all change points are found.

To illustrate how the above algorithm works, consider a time series $T = 5, 5, 5, 1, 1, 1$. The average is 3. So T' is $2, 2, 2, -2, -2, -2$, and S is $2, 4, 6, 4, 2, 0$, where the maximum is 6 and the minimum is 0. Thus the candidate change point is at point 3 and its score is 6. We claim point 3 is a change point if it then passes the significance test. A reported change point indicates a shift in average bandwidth. Between two change points is a segment where there is no shift in the average bandwidth.

In Figure 4.1, the dots mark the detected change points in our real traces. We observe that the change points clearly differentiate the stable and unstable periods. Specifically, in unstable periods, the average bandwidth

shifts frequently, causing a change point to be detected every few seconds while stable periods do not involve a shift in the average, thus contain no change points. Based on this observation, next we propose to determine if it is a stable period based on how old is the last change point.

Classifying periods: First, to apply the change point detection in our system, we need to run this procedure in an online fashion. To do that, we recompute the change points every time a chunk is downloaded and a new bandwidth estimation is computed. However, recomputing all change points from the beginning is slow and unnecessary. So we only recompute using the past T chunks' information. Note the change points may change every time we compute due to the change in the computation window, but we only care about the latest change point because it tells the starting point of the current segment.

Second, to find an appropriate threshold, we plot the distribution of intervals between change points in Figure 4.3. We find that most change points are detected when the network is unstable. Specifically, 53% of the intervals are below 20s, indicating a shift in average happened while the previous average has only lasted for less than 20s. On the other hand, among the intervals that are longer than 20s, 68% are longer than 30s and 45% are longer than 40s; among the intervals longer than 40s, 77% are longer than 50s and 57% are longer than 60s. This suggests that a period that was stable for a sufficient amount of time are more likely to remain stable for at least that long. We leverage this observation and differentiate stable and unstable periods based

on how old the previous change point is. Specifically, whenever we finish downloading a chunk, we recompute the change points and calculate the gap between the latest change point and the current time. If the gap is greater than a certain threshold (which is set to 20 seconds in our evaluation), we claim the current period as stable , otherwise we classify the current period as unstable.

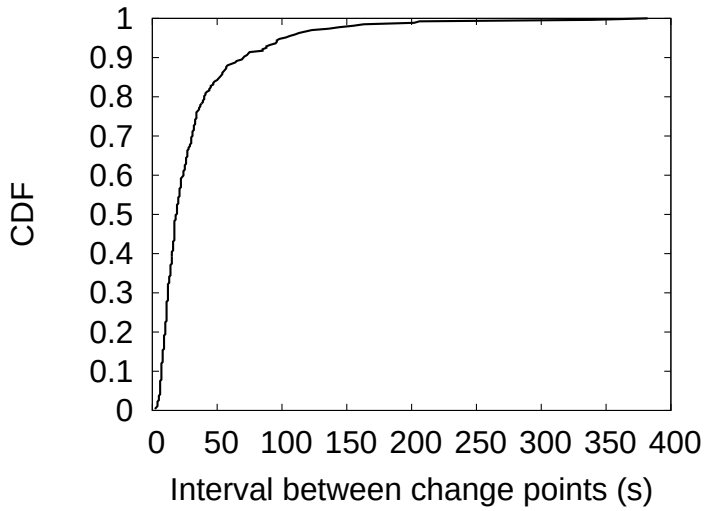


Figure 4.3: Distribution of interval between consecutive change points.

Assisting prediction: We also use the change point detection to help the bandwidth prediction. When a change point is detected, we notify the bandwidth prediction algorithm that the history before the change point should be discarded because the mean has shifted. This allows us to quickly adapt to the new bandwidth instead of letting the stale information to contaminate the prediction.

4.2 Constancy-aware Video Rate Adaptation for a Single Flow

In this section, we develop video rate adaptation schemes for a single flow. We have separate rate adaptation algorithms for stable and unstable periods. Our goal is to maximize Quality of Experience (QoE) of video. QoE depends on three main factors in the video streaming: (i) freezes: whether a video experiences a pause due to no content available in the buffer; and it is the most critical metric, (ii) normalized video rate: average video rate normalized by the network bandwidth, and (iii) stability: how frequently the video rate changes. Most commercial players do not buffer more than a few minutes of content in advance to avoid wasting downloads in case a user quits early or performs a seek. We also limit the buffer size in terms of playtime.

4.2.1 Adaptation during Stable Periods

A naive approach: In a stable period, the available bandwidth varies in a relatively small range and there is no significant shift in average bandwidth. To take advantage of the predictable bandwidth, a natural approach is to find the video rate that is immediately lower than the network bandwidth and stay at that rate. While this approach leads to minimum rate changes, it does not fully utilize the network unless the network bandwidth matches the video rate exactly, which is unlikely to happen. If the network bandwidth is higher than the video rate, the buffer will become full and afterwards the download rate is restricted to the video playout rate, which is lower than the

network rate and causes under-utilization.

A fix to the naive approach is to let the player switch to a higher rate once the buffer is full. When the selected video rate is higher than the network rate, the buffer will start to drain immediately. When the buffer level becomes low, the player can switch to a lower rate. This approach fully utilizes the network, but causes frequent rate changes.

Our approach: Based on the above observations, we propose the following rate adaptation algorithm that explicitly takes the current buffer size, the playback history, and the penalty of a rate change into consideration, while avoiding wasting the network resource due to a full buffer and avoiding video freeze due to an empty buffer.

We first design a scheme to select a rate that can sustain for the longest time without causing the buffer to become full or empty, but ignores the playback history. The idea of this scheme is illustrated in Figure 4.4. Specifically, based on the current buffer size, we evaluate all available rates and compute how long we can stay at that rate before the buffer becomes full or empty. For example, if we select R_1, which is the lowest rate, the buffer will quickly become full, whereas if we select a rate that is too high (*e.g.*, R_5), the buffer will quickly drain. The rate that can sustain for the longest time (*e.g.*, R_3 in Figure 4.4) is then selected.

This approach makes the buffer oscillate between a filling phase and a draining phase. When the buffer level is low it is preferred to fill the buffer

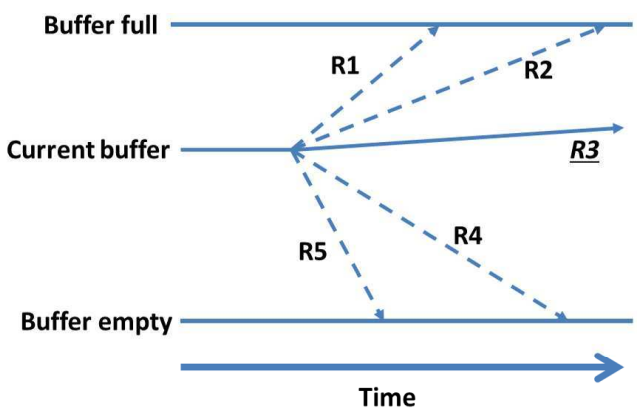


Figure 4.4: Video rate adaptation in stable periods.

with a rate lower than the network bandwidth as it takes longer. Similarly, when the buffer is close to full, video rates that are higher than the network bandwidth are preferred so that the buffer will enter the draining phase. In doing so, we keep the buffer level neither full nor empty so that we keep fully utilizing the network without freezes.

However this approach could cause frequent rate changes. The algorithm prefers a video rate that is close to the current network bandwidth and may switch to a different video rate as the network bandwidth changes dynamically.

To address this issue, we suppress frequent rate changes by only allowing a rate change when the current rate has been played over T seconds. We use $T = 120$ seconds according to the user study in [14]. Note that we always immediately switch to a lower rate to avoid freezes when the buffer is close to empty.

Figure 4.5 shows our pseudocode.

```
// r_0 is the lowest, r_n is the highest
availableRates = { r_0, r_1, ..., r_n }
N_est = estimated network rate
B_max = maximum amount of time can be buffered
B_cur = currently buffered amount of time
T_cur = amount of time current rate has been played
nextRate = currentRate
if( buffer is close to empty )
  nextRate = r_0
elseif( T_cur >= 120 seconds )
  nextRate = r_0
  If( N_est > r_0 )
    maxTime = (B_max – B_cur)/(N_est - r_0)
    foreach r in availableRates
      if(  r > N_est) estTime = B_cur/(r-N_est)
      elseif(  r < N_est) estTime = (B_max – B_cur)/(N_est– r)
      else estTime = + ∞
      If( estTime > maxTime )
          nextRate =  r
          maxTime = estTime
```

Figure 4.5: Pseudocode for rate adaptation in stable periods.

4.2.2 Adaptation during Unstable Periods

In unstable periods when the network is highly unpredictable, our primary goal is to protect against sudden rate drops and avoid freezes. The safest approach here is to always download at the lowest available rate. However that is too conservative and may cause a significant waste in network resource. We propose the following two techniques to address the issues:

Optimistic deferral. Upon entering an unstable period (i.e., when a new change point is detected), we do not immediately start to request at the

lowest rate. Instead, we keep requesting at the rate used at the end of the previous stable period until the buffer becomes lower than a certain threshold. The intuition is twofold. First, we have no knowledge of how long the unstable period will last. The fluctuation may last only briefly and the existing buffer accumulated so far may be able to absorb its impact and avoid a rate change. Second, the unstable network only suggests the bandwidth varies significantly but not necessarily is lower than the current video rate. We only need to switch to the lowest rate for protection if the current rate is indeed not sustainable.

Backward replacement. Once we switch to the lowest rate, it is possible that the buffer (in terms of time) quickly becomes full. To avoid wastage in resource and improve video quality, we replace the content in the buffer with a higher rate content backwards from the end of the buffer. We stop when the distance between playpoint and the chunk that will be replaced next becomes closer than N chunks to avoid the case where the playpoint reaches the chunk before the higher rate chunk is fully downloaded. In this case the partially downloaded chunk at the higher rate will be wasted and the lowest rate chunk will be played. We use $N = 3$ in our implementation.

An important question is which rate to use as the higher rate in this case. It is hard to optimize the selection due to the lack of predictability. We use the following heuristic in our implementation. We consider two candidates: the high rate of the previous window or the rate that is immediately below the current estimation. The first candidate can potentially improve the stability of the video in case we can replace all chunks in the current window before

the playpoint reachs the first chunk in the window. This benefit arises only when the network comes back to the previous rate, which may not happen very often. Moreover, if the last rate change is old enough, then having one more rate change does not affect the video quality. Thus we only choose the first candidate when these two conditions hold: (i) the current window has not been played yet, and (ii) we have been playing or buffering at the high rate of the previous window for less than 120 seconds at the end of the previous window. In case these two conditions do not hold, we use the rate that is immediately below the current estimation to avoid draining the buffer and being overly conservative.

Remarks: Natural questions one may ask is what is the cost associated with possibly downloading the same content at two rates and if it is worthwhile. We note that avoiding video freezes is the most critical concern. However, the reactive schemes in the existing works [115], which drop the video rate only upon too low buffer occupancy, can be too late to avoid freezes. In comparison, the proactive scheme as used here can more effectively prevent the freezes under unforeseen network performance drop or loss of connectivity. The cost of this proactive scheme is reasonable for the following reasons: (i) by using optimistic deferral, we only invoke the proactive protection when necessary; (ii) when it is invoked, if the network bandwidth is not much higher than the lowest video rate, the amount of resources available to be spent on the backward replacement is limited anyway; (iii) when the network resource is considerably higher than the lowest video rate, the cost of downloading

lowest video rate is small compared to downloading the higher video rate in the backward direction. Adjacent rates usually differ by at least 50% in most streaming players in which case the worst case cost is 40% (1/(1+1.5) = 40%). If the two video rates are not adjacent, the cost is even lower (*e.g.*, the highest vs. lowest video rates can differ by orders of magnitude).

4.3 Coordinated Video Rate Adaptation for Multiple Flows

So far we have focused on video rate adaptation for a single flow. When there are multiple video flows competing for the network resources, the existing rate adaptation schemes incur instability, unfairness between players, and bandwidth under-utilization [2]. Wireless interference further exacerbates the problem. In particular, letting the flows that share a common bottleneck adapt their video rates individually is ineffective for the following reasons.

First, the network bandwidth for each individual flow may fluctuate widely depending on the interactions between the flows even though the total aggregate bandwidth at the bottleneck is quite stable. This unnecessarily introduces unpredictability and causes instability and under-utilization as observed in [2]. Second, different flows compete for network bandwidth without knowledge of the objectives of other flows and cause sub-optimal performance. Different flows may have different objectives due to the types of their videos and their device display constraints (*e.g.*, smartphones vs. laptops) etc. Thus letting these flows individually optimize their own utility can deviate from the

global optimum and cause unfairness issues.

While there has been considerable work on optimizing throughput for multiple flows, the video quality optimization differs from throughput optimization in the following three aspects: (i) the video quality function is non-linear in terms of video rates, which take discrete values; (ii) the video quality does not only depend on the current performance but also how much the performance has changed from the past (since viewers prefer smoother and fewer changes). Therefore when network condition changes, the optimization should account for the amount of changes each video flow experiences. That is, we do not just optimize for one snapshot independently but optimize across multiple snapshots by taking into account the changes between the snapshots; (iii) the utility function of different flows can be very different in the mobile environment. For example, a too high video rate on a smartphone with a small screen not only has little improvement on the viewing quality, but also quickly drains the battery due to the higher computational cost. To address the issues, we develop a simple yet effective video rate adaptation scheme for multiple flows.

Specifically, we first measure and predict the total network bandwidth. Then based on the predicted network condition, we optimize the video quality across different flows as follow:

$$Max : \sum_i U_i(r_i, r_i')$$
$$\textit{subject to capacity constraints on } r_i$$

where r_i and r_i' are the network rates allocated to the i-th video flow in the

current and previous intervals, respectively, and $U(.)$ is the video quality that needs to take into account the freezes, video rate, and amount of rate change from the previous interval to the current interval. This optimization problem can be efficiently solved using fminunc in Matlab. Note that while we use the sum of utilities across all flows as an objective, our optimization can easily support alternative objectives, such as proportional fairness, defined as $\sum_i \log U_i$, to take into account both fairness and total utility [103], and other functions of $U(.)$.

The video server may perform this optimization. The optimization result r_i is not directly used as the video rate for the flow i, because the actual network bandwidth for the flow may not be exactly r_i and can be affected by other external factors which may not be predictable, such as interference, congestion, and signal quality. Instead, r_i is sent to the clients, which uses it as the predicted network bandwidth and feeds it as an input to the single-flow rate adaptation as described in Section 4.2 to determine the final video rate. In this way, each flow does not individually estimate and predict future bandwidth themselves, which not only has a higher error due to less degree of aggregation and uncoordinated competition among the flows but also sub-optimal partition of the bandwidth across the flows. Moreover, the single rate adaptation can apply the appropriate video rate adaptation scheme according to whether the current network condition is stable or unstable.

4.4 Cross-layer Rate Adaptation

In this section, we describe an enhancement to MAC-layer rate adaptation and then integrate the MAC-layer rate adaptation with the application-layer video rate adaptation.

4.4.1 MAC-layer Rate Adaptation

OFDM is widely used in Wi-Fi and cellular networks. In OFDM, data is transmitted by spreading them over multiple orthogonal subcarriers concurrently. Each subcarrier is essentially a narrowband channel. Due to frequency selective fading and narrowband interference, different subcarriers usually see different signal quality [49, 53]. Based on this observation, [53] proposes selecting the data rate based on effective SNR derived from the channel state information (CSI), which captures signal-to-noise ratio on each subcarrier. Specifically, it maps SNR of each subcarrier to bit-error-rate (BER), computes average BER across all subcarriers, and converts the average BER to effective SNR with the same BER. Finally it chooses the MAC-layer data rate based on the effective SNR.

The current effective SNR captures the signal variation across frequency. However, effective SNR is measured using a preamble and every frame gives only one SNR reading for each subcarrier. This information is rather coarse and can lead to significant prediction error, since in order to receive a frame correctly, both the preamble and the data symbols need to be received correctly.

Interestingly, not only can we derive SNR information from preambles, but also from data symbols. By combining the SNR information from both preambles and data symbols, we can obtain more fine-grained information, which can help us significantly improve prediction of the SNR in future frames and select a more appropriate data rate. We derive SNR from data symbols based on the distance between the received signal and the constellation point it maps to. Greater distance indicates a lower SNR. In fact, as shown in [9], their relationship can be expressed as :

$$SNR \approx \frac{P_0}{|S_r - S_t|^2} \tag{4.1}$$

where S_t is the transmitted signal, S_r is the received signal, and P_0 is the average power of all possible symbols of the chosen modulation.

The receiver only knows the decoded signal, denoted as S_d, which may or may not be equal to S_t depending on if the decoding is correct. Fortunately, S_d is usually equal to S_t since we generally select the data rate that allows most received symbols to be correctly decoded. Therefore the receiver can simply use $\frac{P_0}{(S_d - S_t)^2}$ to estimate SNR.

Given per symbol SNR in all the previous frames, we predict the mean and variance of SNR in a new transmission using a simple exponential weighted moving average (EWMA).

$$Z_{i+1} = \alpha X_i + (1 - \alpha) Z_i \tag{4.2}$$

where α is used as the weight of the most recent measured value and set to 0.8 in our evaluation.

Based on the predicted mean and variance of SNR in new frame, we select the MAC-layer data rate that maximizes throughput. This essentially means finding the data rate such that $maximize\ Throughput(rate, SNR_{mean}, SNR_{var})$.

$$\begin{aligned} &Throughput(rate, SNR_{mean}, SNR_{var}) \\ =&rate \times FDR(rate, SNR_{mean}, SNR_{var}) \end{aligned}$$

where FDR is the frame delivery rate at a given data rate. We can derive FDR based on the BER corresponding to the current SNR and the type of FEC that is used. For example, if Reed-Solomon FEC is used, the frame delivery rate is simply the probability that the number of erroneous bits is no larger than the maximal tolerable number of erroneous bits. Assuming the BER follows a normal distribution, then we have:

$$\begin{aligned} &FDR(rate, SNR_{mean}, SNR_{var}) \\ &= \Phi\left[\frac{r_{rate}/(2 \times FECSize) - BER_{mean}}{BER_{var}}\right] \end{aligned} \tag{4.3}$$

where $\Phi[x] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathsf{e}^{\frac{-u^2}{2}} du$ is the standard normal CDF, $FECSize$ is the number of bits in an FEC group, r_{rate} is the number of redundancy bits for the rate, and $r_{rate}/(2 \times FECSize)$ gives the maximum tolerable errors.

There has been previous work on leveraging the fine-grained SNR information (*e.g.*, [121]). Different from [121], which uses BER of the previous

frame to select the data rate based on a heuristic thresholding based approach, we recognize BER is a random process instead of a constant and estimate its mean and variance using EWMA. Moreover, we use the derived mean and variance to compute the rate to maximize throughput.

4.4.2 Integrate with Video Rate Adaptation

As we will show in Section 4.6, the rate adaptation based on per symbol SNR can lead to considerable improvement in bulk transfer throughput. We can translate the same benefit to videos by using the maximized throughput as the prediction for the future network bandwidth and feeding it as the input to the single-flow or multi-flow rate adaptation schemes.

4.5 Implementation

We develop a HTTP DASH system that implements our video rate adaptation algorithm along with other schemes. Our system is implemented in Java and consists of 8000 lines of code. It has two main components : DASH server and DASH client.

DASH server: The server is a light-weight HTTP server that handles HTTP GET requests from the client to request chunks. We implement a stand-alone HTTP server for greater flexibility in adding various functionality including partial chunk requests and request cancellation. For simulation the server throttles the bandwidth to the client according to the traces. The server consists of four modules, each of which runs as a separate thread. The main

module accepts incoming requests and manages the other modules. The worker module processes the requests, the network module throttles the outgoing bandwidth according to the mobile network traces, and the communication module exchanges control messages with the multiple flow optimization agent for multiple flow rate adaptation.

DASH client: The client implementation includes everything in a typical commercial video streaming client except the user interface. We implement various rate adaptation schemes for performance comparison and the client can select which scheme to run. The client is also multi-threaded, and each thread runs one client module. The main module manages the buffer and runs the algorithm. The communication module interacts with the multiple flow optimization agent. The downloading module is responsible for managing HTTP requests to the server, such as sending, preempting, resuming, or canceling a request. This module also collects network statistics, such as measured bandwidth.

Multiple flow optimization agent: This agent uses network traces as input, runs the optimization as described in Section 4.3, and transmits the result to the DASH servers and clients. The DASH clients will use the bandwidth allocation as the predicted future network performance and apply the single-flow rate adaptation as usual. In real deployment, this agent should also measure and predict aggregate network performance, which is used as input to the optimization procedure, instead of using the network traces.

4.6 Performance Evaluation

In this section, we evaluate the effectiveness of single-flow rate adaptation, multiple flow rate adaptation, and joint MAC-layer and video rate adaptation.

4.6.1 Evaluation Setup

We choose realistic configuration for our evaluation based on the analysis of popular commercial streaming applications [3]. We use 10 video rates ranging from 250Kbps to 8Mbps where rates are equally distributed in log scale. Video rates available in typical applications do not exceed 3Mbps to 4Mbps and we add a few higher video rates since some of the traces have higher data rates and we need video rates no lower than the highest data rates in the traces to compare the effectiveness of different schemes. The DASH chunk size is set to 3 seconds and the client specifies the video rate in every HTTP GET request sent to the server. Upon receiving a request, the server generates a chunk based on the bit-rate as done in [61]. The default buffer size of the client is set to 60 seconds.

We run our evaluation on Linux machines. For single flow evaluation, we run one instance of the server and the client where the bandwidth between the server and the client is throttled by the server according to the wireless trace. For multiple flow evaluation, we run all servers on a single machine and the multiple flow optimization agent sends control messages to the servers to control their outgoing bandwidth. In addition, the outgoing bandwidth of the

machine is throttled by a daemon with Linux's *tc* command according to the aggregate bandwidth in the wireless trace. Figure 4.6 shows the structure of our multiple flow evaluation.

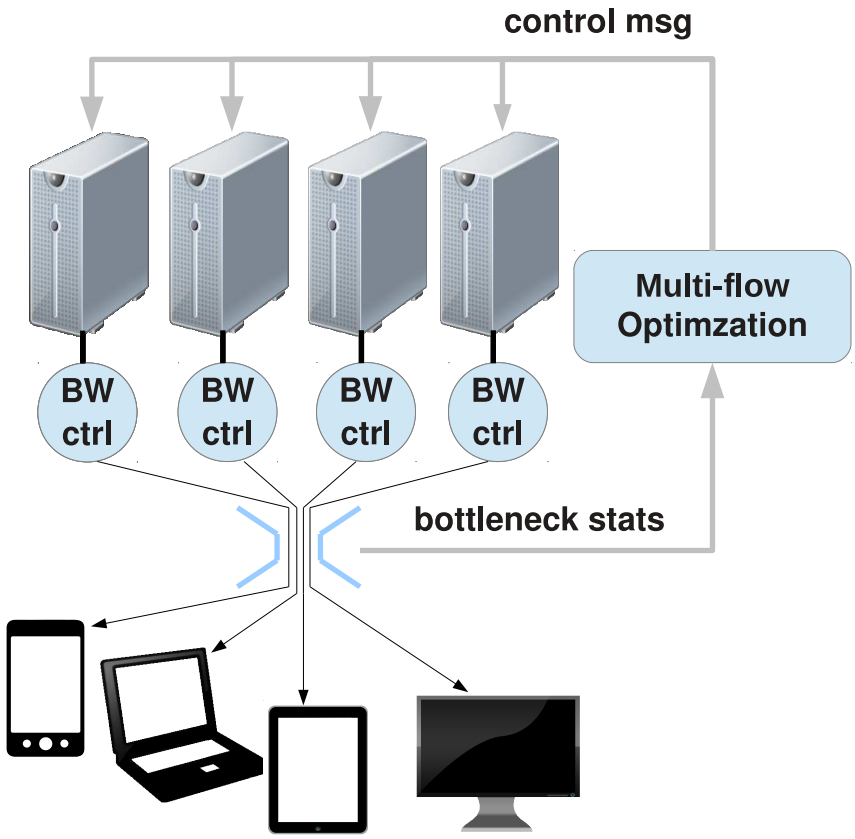


Figure 4.6: Multiple flow evaluation setup.

We collect a wide range of wireless traces as summarized in Table 4.1 for trace driven evaluation. The traces capture various mobile scenarios with different types of wireless networks. Traces are collected from tcpdump results of a large file download via FTP. *stable1* and *stable2* are traces for evaluation

Trace	Time	Description
stable1 *stable2*	960s	indoor, stationary
unstable1 *unstable2*	750s 440s	outdoor, daytime, walking & bus (part of a longer trace)
commute1 *commute2*	1020s 1989s	outdoor, walking & driving

Table 4.1: Description of traces

of our algorithm for stable periods that we acquire by shifting the same trace to different throughput ranges. *unstable1* and *unstable2* are used for evaluating our algorithm in unstable periods. They are obtained by taking unstable periods from two different long traces. *commute1* and *commute2* are the mixed traces that contain both stable and unstable periods. They are used to evaluate our combined video rate adaptation scheme, which automatically detects if the current network condition is stable or unstable and applies the rate adaptation scheme corresponding to the mode.

We quantify the video performance using the following three metrics:

- Normalized Rate: It is the average of playback bit-rate normalized by the network bandwidth. Bit-rate is counted 0 while playback is frozen for calculating average. It is defined as: $\frac{Average\ playback\ rate}{Average\ network\ throughput}$.

- Stability: We assume rate changes do not degrade the user experience if they are infrequent enough and use 2 minutes as the threshold value, as suggested in [14]. This gives us the following metric:

$$Min(1, \frac{Average\ rate\ change\ duration}{120 seconds})$$

- Freezes: This is the most critical metric. It is quantified as follows:

$$\frac{Amount\ of\ time\ spent\ frozen}{Total\ video\ length}$$

4.6.2 Video Rate Adaptation for a Single Flow

We first evaluate the performance of a single flow. We compare with the following baseline schemes:

- C3: Our scheme for both stable and unstable periods.
- C3/S: Our scheme for only stable periods.
- C3/U: Our scheme for only unstable periods.
- EWMA: It selects the rate that is immediately below the 60-second network bandwidth estimated using EWMA.
- BufSize: The buffer size oriented rate control scheme presented in [115]
- SS [112]: Approximated version of MS SmoothStreaming.
- Adobe [98]: An approximated version of Adobe OSMF.
- Akamai [1]: Approximated version of Akamai HD.
- Netflix [93]: Approximated version of Netflix.

[61] approximates the video rate adaptation schemes in commercial players using extensive experiments and determines the settings for these rate adaptation schemes. We use their approximations for SS, ADOBE, AKAMAI, and NETFLIX.

Performance in stable periods: We compare C3/S with the other schemes using traces *stable1* and *stable2*. As shown in Figure 4.7, C3/S shows $10 \sim 20\%$ higher normalized rate than the other schemes. This is because C3/S detects that the network conditions are stable and selects appropriate video rates that could be higher or lower than the estimated network throughput to fully utilize the network bandwidth. In comparison, the other schemes all limit the video rate to be lower than the estimated throughput and causes network under-utilization. Here since all schemes show high stability and no freeze, we omit these results for brevity.

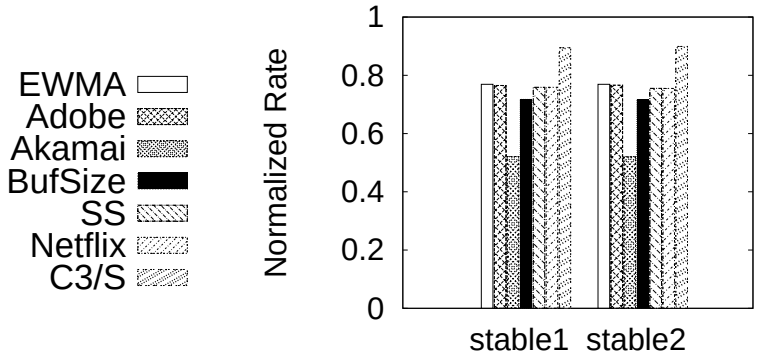
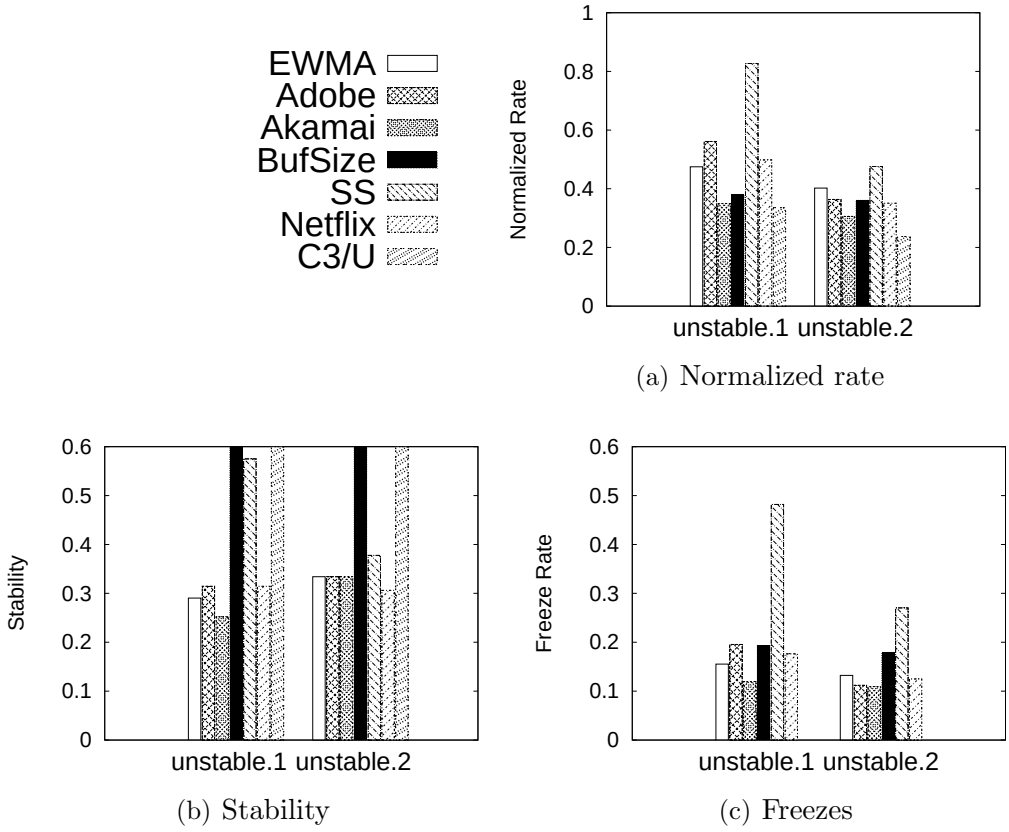


Figure 4.7: Normalized rate of different rate adaptations for a single flow with stable traces.

Performance in unstable periods: Next we compare C3/U with the other schemes using *unstable1* and *unstable2* traces, and plot the results in Figure 4.8. Note that C3/U is the only scheme that shows no freezes in both traces and no bar is shown for C3/U in Figure 4.8(c). The other schemes show significantly higher freezes, ranging 10% $\sim$ 48%. Freeze is the most critical metric to the video, and normalized rate and stability are secondary concerns. So C3/U is the most preferred scheme.



(a) Normalized rate

(b) Stability

(c) Freezes

Figure 4.8: Comparison of different rate adaptations for a single flow with unstable traces.

Performance with mixed traces: Now we consider mixed traces: *commute1* and *commute2*, which have both stable and unstable periods. We evaluate our bi-modal scheme, which automatically detects the type of the current period and invokes the corresponding rate adaptation scheme. periods. As shown in Figure 4.9, C3 shows the best performance overall.

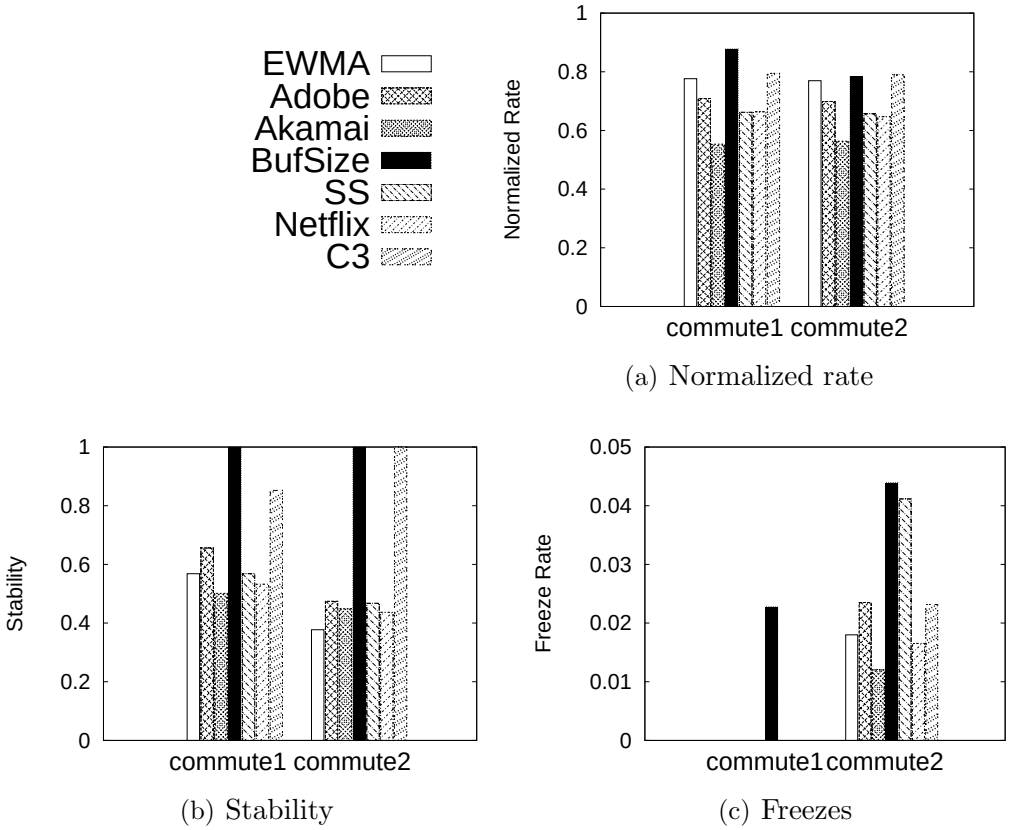
C3 shows no freezes and its stability and normalized rate are among the best in trace *commute1*. BufSize shows a 10% higher normalized rate and 15% higher stability, however its freeze rate is over 2%. In Trace *commute2*, C3 shows the best normalized rate and stability. Our scheme is not always the best in normalized rate because our adaptation algorithm for unstable periods tries to be conservative to protect against freezes. In these periods, other algorithms can achieve higher normalized rate by utilizing the network more aggressively, but they may cause higher freezes and the cost of freezes outweighs the benefit of a slightly higher normalized rate.

4.6.3 Video Rate Adaptation for Multiple Flows

Next we evaluate multiple flows that share a common bottleneck. We compare our optimization framework with the case when each flow predicts its future bandwidth based on its history and selects its own video rate independently.

We use the following utility function for our evaluation:

$$\frac{\sum_i \sum_t (log_{b_i} R_{i,t} - \alpha \left| R_{i,t} - R_{i,t-1} \right|)}{N \times T},$$



(a) Normalized rate

(b) Stability

(c) Freezes

Figure 4.9: Comparison of different rate adaptations for a single flow with mixed traces.

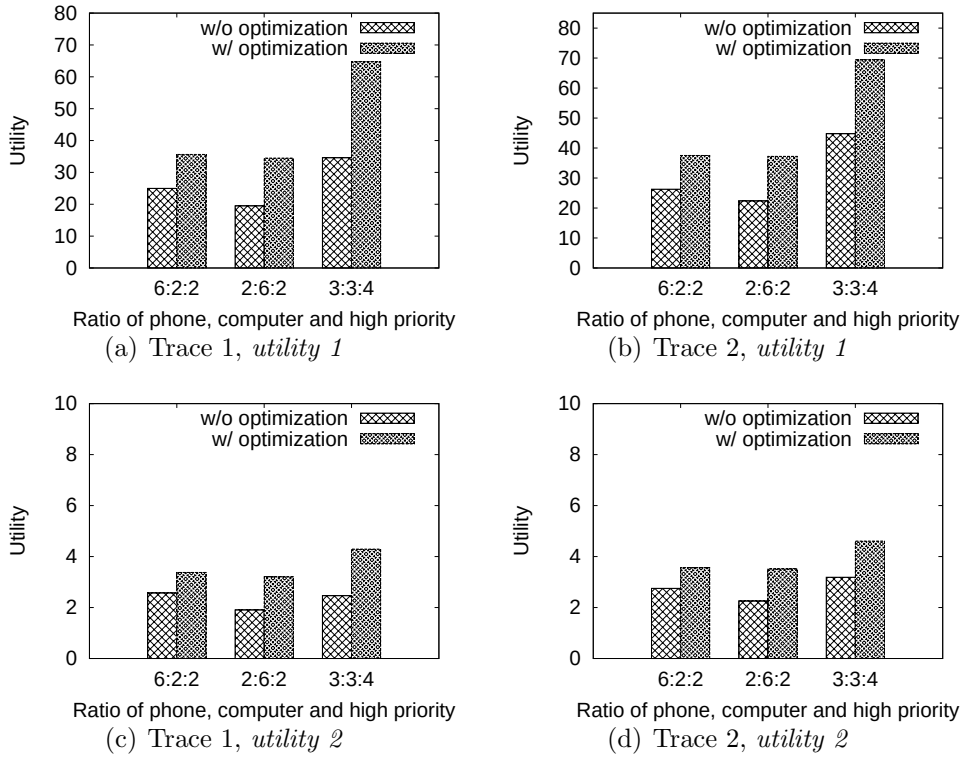
where α is selected to make the two terms have the same magnitude and set to 0.00001 in our evaluation. $R_{i,t}$ is the video rate of user i at time t. N is the number of users. T is playtime. We consider different devices competing with each other. Laptops have higher resolution screens, and can select among the lower 9 available video rates. Smartphones have lower resolution screen, and can select among the lower 6 rates. The higher priority clients, which are considered as paid members, can select all 10 video rates.

Intuitively, the utility function increases with the average video rate and penalizes the rate change. The utility increases logarithmically with the average video rate, and its base depends on the type of the devices. For example, given the same video, the perceived quality for a person watching on a laptop with a larger screen may be lower than a person watching on a smartphone that has a smaller display, in which case a larger b_i is needed for a laptop.

We evaluate with two different sets of b_i values. In *utility 1*, we choose $b_i = 2$ for a laptop and $b_i = 1.47$ for a smartphone so that they have the same utility when both get their maximum video rates, and $b_i = 1.02$ for the higher priority clients so that we prefer allocating bandwidth to them since this gives the largest increase in the utility. In *utility 2*, we choose $b_i = 4$ for a laptop, $b_i = 2$ for a smartphone, and $b_i = 1.5$ for a higher priority client. Our optimization scheme is general and can easily support other forms of utility functions, as described in Section 4.3.

We use two 3G network traces collected in an office and scale them up by 5 to provide reasonable aggregate bandwidth for 10 clients in this evaluation. Figure 4.10 summarizes the results as we vary the ratios of phones, laptops, and high priority users. As we can see, our optimization framework yields significantly higher utility. In *utility 1*, our scheme has 31.2% to 87.1% higher utility in trace 1, and 29.4% to 55% higher in trace 2. This is because our approach yields more accurate throughput prediction due to aggregation and optimized bandwidth allocation across different flows. In comparison, the

individual competition not only incurs much higher prediction error due to fluctuation arising from competitions and less degree of aggregation, but also sub-optimal bandwidth allocation, since the allocation is solely determined by the dynamics of competition and is not optimized. In *utility 2*, the improvement of our scheme ranges 29% - 75%. This shows our approach is general and can support different utility functions.



(a) Trace 1, *utility 1* (b) Trace 2, *utility 1*

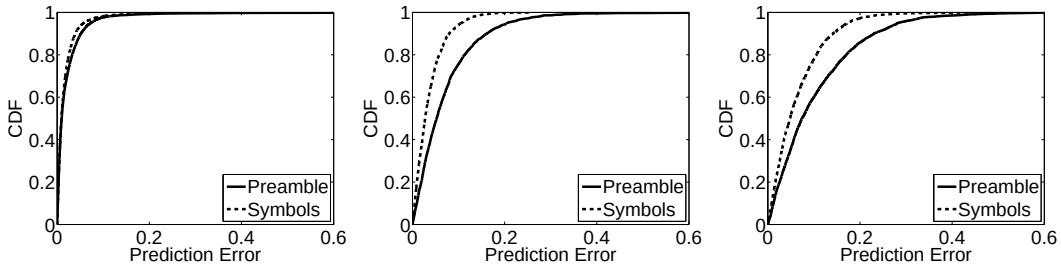
(c) Trace 1, *utility 2* (d) Trace 2, *utility 2*

Figure 4.10: Utility comparison of video rate adaptation for multiple flows.

4.6.4 Joint MAC-layer and Application-layer Rate Adaptation

Our evaluaton uses USRP traces. Since these traces have high throughput, we add 5 additional video rates, ranging from 11Mbps to 50Mbps to avoid video rates becoming the bottleneck. 50Mbps corresponds to the Blu-ray video quality.

First, we compare the accuracy of predicting next frame's effective SNR using our per symbol SNR versus the existing preamble SNR. Our scheme reduces the prediction error by 40-59%. The reduction is more significant for mobile traces since it is more important to capture the trend of the signal changes within a frame in this case.



(a) static trace (dist=0.1m) (b) static trace (dist=5m) (c) mobile trace

Figure 4.11: CDF of prediction errors

Next we evaluate the effectiveness of MAC rate adaptation. As Figure 4.12 shows, our scheme leads to 12-21% throughput increase due to the more accurate prediction of future channel conditions. As we would expect, the improvement is larger for mobile traces due to the higher improvement in accuracy.

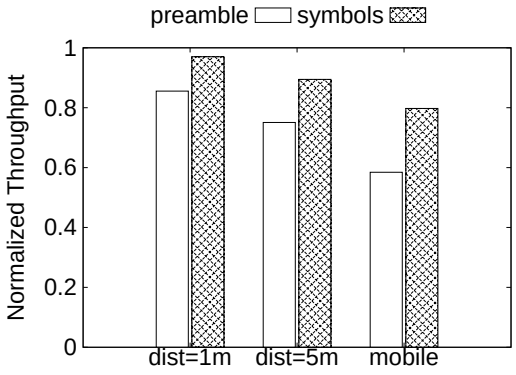
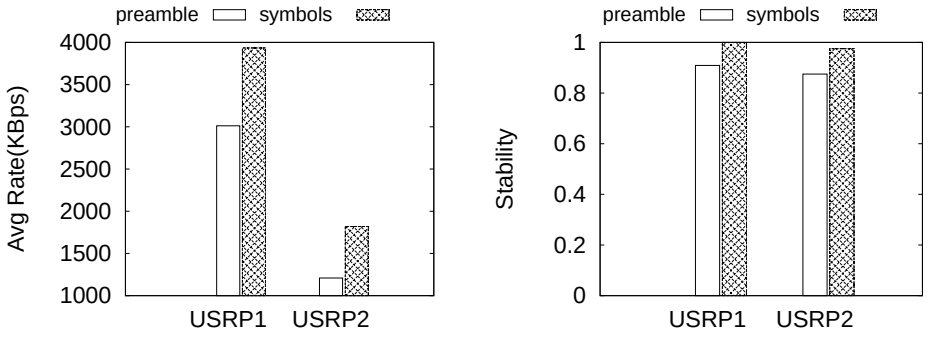


Figure 4.12: Throughput comparison

Finally, we compare the video rate and stability using the two schemes with two USRP traces. Figure 4.13 shows that our scheme increases the video rate by 30-50% and improves the stability by 9-11%.



(a) Rate (b) Stability

Figure 4.13: Comparison of performance with and without joint MAC-application layer rate adaptation

4.6.5 Summary of Results

To summarize, we implement and evaluate our video rate adaptation in a testbed using a range of wireless mobile traces, and quantify the benefit of

each of our proposed components. In particular, our *constancy-aware* single-flow rate adaptation yields 10∼20% higher video rates in stable periods by selecting appropriate video rates to effectively utilize the network. It is the only scheme that yields zero freezes in unstable traces. It also performs the best in the mixed traces with both stable and unstable periods by adapting rate based on different network condition. Our *coordinated* multiple video flow optimization is also effective. It improves the overall utility by 42 - 87% over the video rate selection by individual flows. Our *cross-layer* enhancement further increases video rate by 30-50%, and improves stability by 9-11%.

4.7 Related Work

Video rate adaptation has attracted considerable research work recently. There have been a series of measurement studies that reveal significant weaknesses in the rate adaptation implemented by today's commercial players. (*e.g.*, [3, 34, 43, 75, 2, 61]). For example, [3] measures two major commercial players (Smooth Streaming, Netflix) and one open source player (OSMF), and identifies major differences between the three players and significant inefficiencies in each of them. [34] experimentally investigates the performance of this new Akamai video service and reports that when the network has an abrupt change, it takes roughly 14 seconds before selecting the suitable vide rate. This is certainly too long and can cause significant interruptions in video streaming. [3] also identifies significant inefficiencies in the two major commercial players (Smooth Streaming, Netflix) and one open

source player (OSMF): they respond to the short-term changes too slowly and do not avoid unnecessary oscillations; in addition, they incur unfairness and sub-optimal performance when multiple video flows share the bandwidth. [43] presents a systemic analysis of the relationship between video quality metrics, content types, and quantitative measures of engagement and points out the need for robust bit-rate adaptation algorithm. [75] points out the network and delivery infrastructure that today's video streaming relies on is fundamentally unreliable and emphasizes the need of intelligent bitrate selection algorithm. [2] reports three performance problems in today's commercial players: player instability, unfairness between players, and bandwidth underutilization. [57] measures three popular video streaming services: Hulu, Netflix, and Vudu, and concludes picking a video streaming rate is hard due to the downward spiral effect arising from the inaccurate bandwidth estimate. [61] identifies additional limitations in the existing rate adaptation. [104] studies Netflix and YouTube to model their streaming strategies. For example, [3] identifies significant inefficiencies in the two major commercial players (Smooth Streaming, Netflix) and one open source player (OSMF): they respond to the short-term changes too slowly and do not avoid unnecessary oscillations; in addition, they incur unfairness and sub-optimal performance when multiple video flows share the bandwidth. [75] points out the network and delivery infrastructure that today's video streaming relies on is fundamentally unreliable and emphasizes the need of intelligent bitrate selection algorithm. [2] reports three performance problems in today's commercial players: player instability, unfairness between

players, and bandwidth under-utilization.

Motivated by the serious issues in the existing schemes, a number of video rate adaptation schemes have been proposed, such as QDASH [89], FESTIVE [61], client-side buffer-based adaptation [115], [55] and QAVA [32]. QDASH [89] reports that gradual quality changes have less impact on users' experience via experiments, and propose an adaptation algorithm that prefers gradual changes and combine it with available bandwidth measurement. Our design is guided by the same observation and reduces both number of rate changes and the gap between successive rates. [115] designs a single player video rate adaptation algorithm that uses the buffered video time as the feedback signal. However, their design is not tailored to the mobile environment and suffers when the bandwidth is highly unpredictable as shown in Section 4.6. FESTIVE [61] explicitly takes into account the competition between multiple players and designs techniques to improve fairness, stability and efficiency. In contrast to their distributed approach, where each player makes its own decision, we jointly optimize the aggregate performance across multiple flows, thereby achieving higher stability and performance. [55] takes a similar approach by implementing the bandwidth arbitration in the home gateway. Different from their approach, our optimization framework also considers the types of the client devices when optimizing the overall user experience. [72] proposes a smoothed throughput measurement algorithm, based on which they adopt a step-wise increase and aggressive decrease to find the rate that matches the network bandwidth. [71] proposes a method to determine

the segment duration in an HTTP DASH system and facilitate accurate and fast rate adaptation. [131] proposes using location information as an indicator of network capacity in vehicular environments by constructing a street-level bandwidth map. However, location is not the only factor that affects network bandwidth, and network congestion also has significant impact. Our change point detection can better track unexpected changes, and does not require map information. [79] considers mobile devices with limited native player capabilities and proposes using seek operation to perform rate adaptation. Their scheme is specific to devices not equipped with advanced players.

In summary, our work is inspired by the previous works. Different from these works, we specifically focus on video rate adaptation for wireless mobile networks by analyzing a range of mobile traces and applying the insights to design constancy-aware, coordinated, cross-layer video rate adaptation scheme.

Chapter 5

Conclusion

In this dissertation we discuss noval strategies to effectively deliver mutlimedia data over the Internet with mobility. We also present real systems based on such strategies and evalute through thorough simulation, emulation and testbed experiments.

We first present the VCD system that provides high-bandwidth content access to vehicular passengers by utilizing opportunistic connections to Wi-Fi access points along the road. VCD predicts which APs a vehicle will encounter in the future and proactively pushes content to these APs by leveraging both wireline and wireless connectivity. Using trace-driven simulation and Emulab-based emulation, we show that VCD is capable of downloading 3-6 times as much content as no replication and 2-4 times as much content as wireline or vehicular replication alone. The benefit further increases as the ratio between wireless and wireline capacity increases. We further develop a full-fledged prototype of VCD using two testbeds: a 9-AP 802.11b testbed and a 4-AP 802.11n testbed. Our experience suggests that VCD is an effective approach for vehicular content distribution.

We also propose a novel approach to support high-quality large video

conferences using the Valiant multicast routing. Our results show that our robust VMR/2 choices out-performs the broadcast tree by 75%-96%, minimum spanning trees by 63%-97%, and shortest path trees by 62%-97%. Among different versions of VMR, the robust VMR with 2 choices out-performs VMR with 1 choice by taking advantage of load information, and out-performs the VMR with 2 choices based on the load predicted from several widely used predictors. This shows that the load is hard to predict accurately and it is important to explicitly capture the uncertainty of load information. In addition, VMR can effectively incorporate locality information, leverage resources from other peers and CDN nodes whenever necessary, and adapt streaming rates according to the current network conditions. Our VMR/2 has applications beyond the video conferencing, which we plan to explore. We are also interested in having trials with real users.

Lastly we discuss a novel video rate adaptation scheme for mobile networks. We first analyze a range of wireless mobile traces and observe they show both stable and unstable performance over time. Motivated by these observations, we develop constancy-aware, coordinated, cross-layer rate adaptation scheme. Specifically, we automatically classify the current network condition into stable or unstable mode, and develop video rate adaptation for each mode. In addition, when multiple clients are streaming from the same video source and sharing the common bottleneck, we use a joint optimization scheme to maximize the overall quality across all clients. Furthermore, we propose a simple enhancement to the MAC rate adaptation by exploiting the SNR infor-

mation across the all data symbols as well as in the preamble, and integrate it with video rate adaptation. Our extensive evaluation shows the benefits of our single-flow rate adaptation, multi-flow rate adaptation, and MAC-layer enhancement in the wireless mobile networks. In the future, we are interested in conducting trials with real users and collecting user experience to further improve our design.

Bibliography

[1] Akamai. `http://www.akamai.com`.

[2] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen. What happens when http adaptive streaming players compete for bandwidth? In *Proc. of NOSSDAV*, 2012.

[3] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Multimedia Systems Conference*, 2011.

[4] Istemi Ekin Akkus, Oznur Ozkasap, and M. Reha Civanlar. Peer-to-peer multipoint video conferencing with layered video. In *Jounal of Network and Computer Applications*, Jan. 2011.

[5] Ian F. Akyildiz and Wenye Wang. The predictive user mobility profile framework for wireless multimedia networks. *IEEE/ACM Trans. Netw.*, 12(6), 2004.

[6] Abdul Rahman Aljadhai and Taieb Znati. Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications*, 19(10):1915–1930, 2001.

[7] M. Amad, Z. Haddad, L. Khenous, and K. Kabyl. A scalable based multicast model for P2P conferencing applications. In *ICUMT*, 2009.

[8] Apple FaceTime. http://www.apple.com/mac/facetime/.

[9] H. Arslan and H.A. Mahmoud. Error vector magnitude to snr conversion for nondata-aided receivers. *Wireless Communications, IEEE Transactions on*, 8(5):2694 –2704, may 2009.

[10] AT&T connect. http://uc.att.com.

[11] AT&T DSL. http://www.att.com/gen/general?pid=6431.

[12] AWE Mesh Router. http://www.nomadio.net/AWE_overview.pdf.

[13] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, pages 29:180–200, 1999.

[14] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. A quest for an internet video quality-of-experience metric. In *Hotnets*, 2012.

[15] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. DTN routing as a resource allocation problem. In *Proc. of SIGCOMM*, 2007.

[16] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. Enhancing interactive web applications in hybrid networks. In *Proc. of MobiCom*, Sept. 2008.

[17] Aruna Balasubramanian, Ratul Mahajan, Arun Venkataramani, Brian Levine, and John Zahorjan. Interactive WiFi connectivity for moving vehicles. In *Proc. of SIGCOMM*, 2008.

[18] Nilanjan Banerjee, Mark Corner, Don Towsley, and Brian Levine. Relays, base stations, and meshes: Enhancing mobile networks with infrastructure. In *Proc. of MobiCom*, Sept. 2008.

[19] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Overcast: Reliable multicasting with an overlay network. In *In Proc. of SIGCOMM*, 2002.

[20] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *In Proceedings of IEEE INFOCOM*, pages 1521–1531, 2003.

[21] Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, and Aravind Srinivasan. Resilient multicast using overlays. In *In Proc. of ACM Sigmetrics*, pages 102–113, 2003.

[22] Petra Berenbrink, Andr Brinkmann, Tom Friedetzky, and Lars Nagel. Balls into non-uniform bins. In *Proc. of IEEE IPDPS*, pages 1–10, April 2010.

[23] BMW car2car communication development. http://www.motorauthority.com/bmw–enlists-in-car-2-car-communications-development.html.

[24] Lawrence Brakmo, Sean W. O'Malley, and Larry L. Peterson. Tcp vegas: New techniques for congestion detection and avoidance. In *Proc. of ACM SIGCOMM*, 1994.

[25] Vladimir Bychkovsky, Bret Hull, Allen Miu, Hari Balakrishnan, and Samuel Madden. A measurement study of vehicular Internet access using in situ Wi-Fi networks. In *Proc. of MobiCom*, 2006.

[26] Cabspotting. http://www.cabspotting.com.

[27] J. Camp and E. Knightly. Modulation rate adaptation in urban and vehicular environments: Cross-layer implementation and experimental evaluation. In *Proc. of MobiCom*, Sept. 2008.

[28] Car2Car communication consortium. http://www.car-to-car.org.

[29] Cartel. http://cartel.csail.mit.edu/doku.php.

[30] C. S. Chang, D. S. Lee, and Y.-S. Jou. Load balanced Birkho-von Neumann switches, Part I: one-stage buffering. In *Proc. of HPSR*, 2001.

[31] B. B. Chen and M. C. Chan. MobTorrent: a framework for mobile internet access from vehicles. In *Proc. of IEEE INFOCOM*, 2009.

[32] Jiasi Chen, Amitabha Ghosh, Josphat Magutt, and Mung Chiang. Qava: Quota aware video adaptation. In *Proc. of ACM CoNEXT*, 2012.

[33] Xiangwen Chen, Minghua Chen, Baochun Li, Yao Zhao, Yunnan Wu, and Jin Li. Celerity: a low-delay multi-party conferencing solution. In *Proc. of ACM Multimedia*, pages 493–502, New York, NY, USA, 2011.

[34] L. De Cicco and S. Mascolo. An experimental investigation of the Akamai adaptive video streaming. In *HCI in Work and Learning, Life and Leisure*, 2010.

[35] CISCO. Cisco vni:forecast and methodology, 2008-2013. 2008.

[36] CISCO. Cisco vni:forecast and methodology, 2011-2016. 2011.

[37] CISCO. Cisco vni:forecast and methodology, 2012-2017. 2012.

[38] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, Sept. 2003.

[39] CPLEX. `http://www.ilog.com/products/cplex/`.

[40] Mike Dahlin. Interpreting stale load information. *IEEE Transactions on Parallel and Distributed System*, Oct. 2001.

[41] Shirshanka Das, Alok Nandan, and Giovanni Pau. Spawn: a swarming protocol for vehicular ad-hoc wireless networks. In *Proc. of VANET*, 2004.

[42] P. Deshpande, A. Kashyap, C. Sung, and S. Das. Predictive methods for improved vehicular wifi access. In *Proc. of ACM MobiSys*, 2009.

[43] Florin Dobrian, Asad Awan, Dilip Joseph, Aditya Ganjam, Jibin Zhan, Vyas Sekar, Ion Stoica, and Hui Zhang. Understanding the impact of video quality on user engagement. In *ACM SIGCOMM*, 2011.

[44] Emulab. http://www.emulab.net.

[45] Engadget. Study finds netflix is the largest source of internet traffic in north america. May 2011.

[46] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular content delivery using WiFi. In *Proc. of MobiCom*, Sept. 2008.

[47] Joy Ghosh, Matthew J. Beal, Hung Q. Ngo, and Chunming Qiao. On profiling mobility and predicting locations of wireless users. In *Proc. of REALMAN*, 2006.

[48] Video conferencing – a global strategic business report. `http://www.strategyr.com/Video_Conferencing_Market_Report.asp`.

[49] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[50] Google+ Hangouts. `http://www.google.com/tools/dlpage/res/talkvideo/hangouts/`.

[51] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. In *Proc. of INFOCOM*, Apr. 2001.

[52] David Hadaller, Srinivasan Keshav, Tim Brecht, and Shubham Agarwal. Vehicular opportunistic communication under the microscope. In *Proc. of MobiSys*, 2007.

[53] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *Proc. of ACM SIGCOMM*, 2010.

[54] Tracey Ho, Muriel Medard, Jun Shi, Michelle Eros, and David R. Karger. On randomized network coding, October 06 2003.

[55] R. Houdaille and S. Gouache. Shaping http adaptive streams for a better user experience. In *Proc. of ACM MMSys*, 2012.

[56] Yang hua Chu, Sanjay G. Rao, Srinivasan Seshan, and Hui Zhang. A case for end system multicast. In *in Proceedings of ACM Sigmetrics*, pages 1–12, 2000.

[57] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proc. of IMC*, 2012.

[58] Hulu. `http://www.hulu.com`.

[59] Infonetics. Enterprise telepresence and video conferencing equipment. 4Q 2011.

[60] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and Jr. James W. O'Toole. Overcast: Reliable multicasting with an overlay network. In *In Proc. of OSDI*, 2000.

[61] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proc. of ACM CoNEXT*, 2012.

[62] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. of ACM ASPLOS*, Oct. 2002.

[63] Prajakta S. Kalekar. Time series forecasting using holt-winters exponential smoothing. Dec. 2004. `http://www.it.iitb.ac.in/~praj/acads/seminar/04329008_ExponentialSmooth%ing.pdf`.

[64] Sinchai Kamolphiwong Karn Tirasoontorn and Suthon Sae-Wong. Distributed P2P-SIP conference construction. In *Proc. of Mobile Technology, Applications, and Systems*, 2008.

[65] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown. Scaling Internet routers using optics. In *Proc. of ACM SIGCOMM*, Aug. 2003.

[66] Kyu-Han Kim and Kang G. Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proc. of MobiCom*, Sept. 2006.

[67] Alexander Knauf, Gabriel Hege, Thomas Schmidt, and Matthias Wahlisch. A P2P virtualization for distributed adaptive conference management.

GI/ITG KuVS Fachgesprch: NGN Service Delivery Plat. & Svc. Overlay Networks, 2009.

[68] Kevin C. Lee, Seung-Hoon Lee, Ryan Cheung, Uichin Lee, and Mario Gerla. First Experience with CarTorrent in a Real Vehicular Ad Hoc Network Testbed. In *MOVE*, 2007.

[69] Chao Liang, Miao Zhao, and Yong Liu. Optimal bandwidth sharing in multi-swarm multi-party P2P video conferencing systems. *IEEE/ACM Transactions on Networking*, Dec. 2011.

[70] B. P. Lim, K. K. Ettikan, E. S. Lin, Truong Khoa Phan, Nam Thoai, E. Muramoto, and P. Y. Tan. Bandwidth fair application layer multicast for multi-party video conference application. In *Proc. of IEEE CCNC*, 2009.

[71] Chenghao Liu, I. Bouazizi, and M. Gabbouj. Segment duration for rate adaptation of adaptive http streaming. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–4, July.

[72] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. Rate adaptation for adaptive http streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, MMSys '11, pages 169–174, New York, NY, USA, 2011. ACM.

[73] George Liu and Gerald Maguire, Jr. A class of mobile motion prediction algorithms for wireless mobile computing and communication. *Mob.*

Netw. Appl., 1(2), 1996.

[74] Tong Liu, Paramvir Bahl, Senior Member, and Imrich Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks, Mar. 1998.

[75] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *Proc. of ACM SIGCOMM*, 2012.

[76] S.H. Low and D.E. Lapsley. Optimization flow control I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking, Vol 7*, pages 861–875, 1999.

[77] LP_SOLVE: Linear programming code. `http://www.cs.sunysb.edu/~algorith/implement/lpsolve/implement.shtml`.

[78] Chong Luo, Wei Wang, Jian Tang, Jun Sun, and Jiang Li. A multiparty videoconferencing system over an application-level multicast protocol. *IEEE Transactions on Multimedia*, Dec. 2007.

[79] K.J. Ma, Man Li, A. Huang, and R. Bartos. Video rate adaptation in mobile devices via http progressive download of stitched media files. *Communications Letters, IEEE*, 15(3):320–322, March.

[80] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the mac-level behavior of wireless networks. In *Proc. of SIGCOMM*, 2006.

[81] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. Detecting the performance impact of upgrades in large operational networks. In *Proc. of ACM SIGCOMM*, 2010.

[82] Liam McNamara, Cecilia Mascolo, and Licia Capra. Media sharing based on colocation prediction in urban transport. In *Proc. of MobiCom*, pages 58–69, 2008.

[83] Windows meeting space. `http://en.wikipedia.org/wiki/Windows_Meeting_Space`.

[84] Meraki MR58. `http://meraki.com/products_services/access_points/MR58/`.

[85] R. Mirchandaney, D. Towsley, and J. A. Stankovic. Analysis of the effects of delays on load sharing. *IEEE Trans. on Computers*, 1989.

[86] M. Mitzenmacher. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems*, 2000.

[87] Michael Mitzenmacher and Andrea W. Richa. The power of two random choices: A survey of techniques and results. *IEEE Transactions on Parallel and Distributed Computing*, 2001.

[88] Mobile Broadband Review 2010. `http://mobile-broadband-services-review.toptenreviews.com/`.

[89] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang. Qdash: A qoe-aware dash system. In *Proc of MMSys*, 2012.

[90] Vishnu Navda, Anand Prabhu Subramanian, Kannan Dhanasekaran, Andreas Timm-Giel, and Samir Das. Mobisteer: using steerable beam directional antenna for vehicular network access. In *Proc. of MobiSys*, 2007.

[91] NBC. `http://www.nbc.com`.

[92] Network devices and protocols: Windows DDK. NDIS library functions.

[93] Netflix. `http://www.netflix.com`.

[94] Anthony J. Nicholson and Brian D. Noble. Breadcrumbs: Forecasting mobile connectivity. In *Proc. of MobiCom*, Sept. 2008.

[95] Novarum. 2010: Guidelines of successfull large scale outdoor wi-fi networks. December 2009.

[96] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[97] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[98] Adobe osmf. `http://www.opensourcemediaframework.com/`.

[99] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve world wide web latency. *SIGCOMM Comput. Commun. Rev.*, 26(3), 1996.

[100] Pubudu N. Pathirana, Andrey V. Savkin, and Sanjay Jha. Mobility modeling and trajectory prediction for cellular networks with mobile base stations. In *MobiHoc*, 2003.

[101] Laptop & smartphone users prefer Wi-Fi to 3G; willing to pay for city-wide Wi-Fi. `http://www.teleclick.ca/2009/02/laptop-willing-to-pay-for-citywi` `i/`.

[102] QQ. `http://www.qq.com`.

[103] B. Randunovi and J. Y. L. Boudec. Rate performance objectives of multihop wireless networks. In *Proc. of INFOCOM*, Apr. 2004.

[104] A. Rao, Y.-S. Lim, C. Barakat, A. Legout, D. Towsley, and W. Dabbous. Network characteristics of video streaming traffic. In *Proc. of ACM CoNext*, 2011.

[105] Reed-solomon error correcting code. `http://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction#cite%_note-non-binary-0`.

[106] E. Rozner, A. P. Iyer, Y. Mehta, L. Qiu, and M. Jafry. ER: Efficient retransmission scheme for wireless LANs. In *Proc. of CoNext*, Dec. 2007.

[107] Sandvine. Global internet phenomena report, 2h 2012. 2012.

[108] Seattle Bus Traces. http://crawdad.cs.dartmouth.edu/meta.php?name=rice/ad_hoc_city.

[109] Skype. http://www.skype.com.

[110] L. Song, D. Kotz, R. Jain, and X. He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proc. of INFOCOM*, Mar. 2004.

[111] Libo Song, Udayan Deshpande, Ulas C. Kozat, David Kotz, and Ravi Jain. Predictability of WLAN mobility and its effects on bandwidth provisioning. In *Proc. of INFOCOM*, Mar. 2006.

[112] Microsoft smooth streaming. http://www.iis.net/downloads/microsoft/smooth-streaming.

[113] Memory cards. http://en.wikipedia.org/wiki/Comparison_of_memory_cards.

[114] Kunal Talwar and Udi Wieder. Balanced allocations: The weighted case. *Microsoft Research Technical Report*, 2008.

[115] Guibin Tian and Yong Liu. Towards agile and smooth video adaption in dynamic http streaming. In *Proc. of ACM CoNEXT*, 2012.

[116] Toyota and Honda vehicle-to-vehicle communication systems. http://www.motorauthority.c
and-honda-start-testing-vehicle-to-vehicle-communications-systems.html.

[117] Trends and challenges in video coding. http://www.slideshare.net/touradj_ebrahimi/futurevideo.

[118] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.

[119] L. Valiant and G. Brebner. Universal schemes for parallel communication. In *Proc. of STOC*, May 1981.

[120] VBrick. Delay and latency.

[121] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *Proc. of ACM SIGCOMM*, 2009.

[122] Cisco webex. `http:/www.webex.com`.

[123] Webinar. `http://www.gotomeeting.com/fec/webinar`.

[124] Wifi-taxi. http://www.wifi-taxi.com/cgv_en.htm.

[125] Wikipedia. F1 score. http://en.wikipedia.org/wiki/F1_score.

[126] Worldwide pricelist for iPhone 3G Plans. `http://www.unwiredview.com/2008/07/16/worldwide-pricelist-for-iphone-3g%-plans/`.

[127] Xiaotao Wu, Krishna Kishore Dhara, and Venkatesh Krishnaswamy. Enhancing application-layer multicast for P2P conferencing. In *Proc. of Workshop on Peer-to-Peer Multicasting*, Jan. 2007.

[128] Yahoo! Local Search Web Services. http://developer.yahoo.com/search/local/V3/localSear

[129] Mengkun Yang and et al. A proactive approach to reconstructing overlay multicast trees, 2004.

[130] Yaling Yang, Jun Wang, and Robin Kravets. Designing routing metrics for mesh networks. In *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, Sept. 2005.

[131] Jun Yao, S.S. Kanhere, and M. Hassan. Quality improvement of mobile video using geo-intelligent rate adaptation. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6, April.

[132] Youtube. `http://www.youtube.com`.

[133] Yin Zhang, Nick Duffield, Vern Paxson, and Scott Shenker. On the constancy of Internet path properties. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, November 2001.

[134] Rui Zhang-Shen and Nick McKeown. Designing a predictable Internet backbone network. In *Proc. of HotNets III*, Nov. 2004.