REALTIME STREAMING WITH GUARANTEED QOS OVER WIRELESS D2D NETWORKS

A Thesis

by

SUMAN PAUL

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2014

Major Subject: Computer Engineering

ABSTRACT

The increase in the processing power of mobile devices has led to an explosion of available services and applications. However, the cost of mobile data is a hindrance to the adoption of data intensive applications. We consider a group of co-located wireless peer devices that desire to synchronously receive a live content stream. Devices desire to minimize the usage of their B2D interfaces (3G/4G) to reduce cost, while maintaining synchronous reception and playout of content. While it might be possible for a cellular base station to broadcast or multicast live events to multiple handsets, such content would be restricted to a few selected channels, and only available to subscribers of a single provider. Utilizing both B2D and D2D (WiFi) interfaces enables users to pick any event of interest, and "stitch together" their B2D capacities regardless of provider support. Our objective is to enable users to listen or watch real time streams while incurring only a fraction of the original costs.

Our system setup is as follows. The real-time stream is divided into blocks, which must be played out soon after their initial creation. If a block is not received within a specific time after its creation, it is rendered useless and dropped. The blocks in turn are divided into random linear coded chunks to facilitate sharing across the devices. We transform the problem into the two questions of (i) deciding which peer should broadcast a chunk on the D2D channel at each time, and (ii) how long B2D transmissions should take place for each block.

The thesis studies the performance of a provably-minimum-cost algorithm that can ensure that QoS targets can be met for each device. We use a Lyapunov stability argument to show that a stable delivery ratio can be achieved using our mechanism. We show that the optimal D2D scheduling algorithm has a simple and intuitive form under reliable broadcast, which allows for easy implementation and development of good heuristics. We study this via simulations, and present an overview of the implementation on Android phones using the algorithm as a basis. Additionally, we design an incentive framework that promotes cooperation among devices. We show that under this incentive framework, each device benefits by truthfully reporting the number of chunks that it received via B2D and its deficit in each frame, so that a system-wide optimal allocation policy can be employed. The incentive framework developed is lightweight and compatible with minimal amounts of history retention.

The Android testbed used in the experiments consisted of multiple Google Nexus 4 phones. A modified version of Android Jelly Bean (v 4.3) was built in order to conduct the experiments which removes the limitation wherein the phone switches off its 3G data connection (B2D) whenever a known WiFi network (D2D) becomes available. Since the Nexus 4 devices are incapable of operating in ad-hoc mode, we used a WiFi network (without Internet connectivity) to emulate the D2D part. Hence, devices must use their 3G interfaces to receive chunks for the server (via the Internet). We present experimental results, and show that it would be possible to follow popular streams on hand held devices incurring only a fraction of the costs while achieving a high QoS.

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation for my advisor, Dr. Srinivas Shakkottai for his continued support and motivation throughout the duration of my master's thesis. I am more than thankful to him for giving me so much freedom while working on my thesis. Apart from being the perfect mentor and a constant source of inspiration, I am grateful to him for making my research experience all the more enjoyable and memorable.

I also wish to thank my friend and colleague, Rajarshi Bhattacharyya, without whose help and collaboration, this work would have taken much longer to accomplish.

I wish to thank my committee members who were more than generous with their expertise and precious time.

I also take this opportunity to thank my parents for always being supportive of my pursuits.

On a final note, I would like to thank God, who continues to make the impossible, possible.

NOMENCLATURE

D2D	Device to device
B2D	Base station to device
P2P	Peer to peer

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

There has recently been a steep increase in the use of smart, hand held devices for content consumption. These devices, such as smart phones and tablets, are equipped with multiple orthogonal wireless communication interfaces. Interfaces include expensive (both dollar-cost and energy) long-range base-station-to-device (B2D) interfaces (*e.g.* 3G/4G), as well as low-cost short-range interfaces like WiFi. More recently, the use of short-range interfaces such as WiFi-Direct and FlashLinQ for device-to-device (D2D) communication is starting to make an appearance. Simultaneously, there has been an explosion in available content, and it is expected that streaming of live events will play a big part in future demand [3].

We focus on live-streaming of content to multiple co-located devices, as shown in Figure 1.1. Here, data is generated in realtime by a server, and must be delivered to all the devices quickly in order to maintain the “live” aspect of the event. Each device is both capable of unicast communication via a B2D interface, and broadcast D2D communication with peer devices. Devices desire to minimize the usage of their B2D interfaces to reduce cost, while maintaining synchronous reception and playout of content. While it might be possible for a cellular base station to broadcast live events to multiple handsets, such content would be restricted to a few selected channels, and only available to subscribers of a single provider. Utilizing both interfaces enables users to pick any event of interest, and “stitch together” their B2D capacities regardless of provider support. Apart from content consumption in a social setting, such a scheme would also be valuable in an emergency response or battlefield setting.

In our model, content is generated in realtime in the form of *blocks*, with one block corresponding to a playout time called a *frame*. In other words, each device

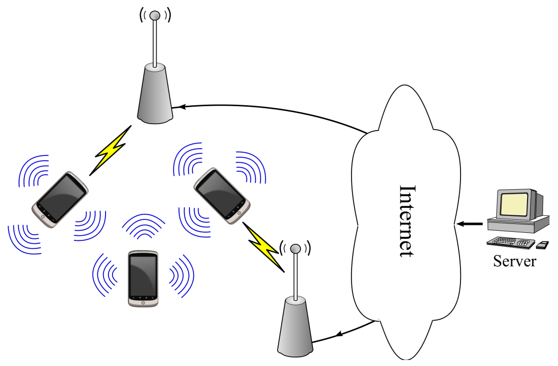

Figure 1.1: Hybrid wireless network. Each device can utilize both unicast base-station-to-device (B2D) as well as broadcast device-to-device (D2D) communication. The server generates blocks of information in realtime.

must receive one block of data within one frame of time for smooth playout. The blocks are divided into *chunks* for sharing via different interfaces. Coordination of chunk identities across all devices and all base-stations is near impossible, so a coding solution is needed for transmitting chunks. Hence, the server performs random linear coding [7, 17] across each block, and unicasts the resulting coded chunks to the devices via the Internet and through the B2D connections. Thus, each coded chunk can be thought of as an element in a vector space with the scalars in a finite field. The information at each device can then be represented by a matrix that contains all the vectors that it has received thus far, and the block of information can be decoded when this matrix is of full rank. Both the Internet and B2D links are lossy, but since the chunks are coded, there is no need for feedback and the server simply generates a new chunk for each transmission. The devices receive coded chunks from the server, and in turn transmit these chunks to each other over a potentially lossy D2D broadcast channel, again without feedback. If each device receives enough chunks to decode the block, it can play it, else it has to skip that block.

We illustrate the timing diagram for block transmission in Figure 1.2. Here, we divide time into *slots*, and each frame consists of T slots. Each block is divided into N chunks, and coding is performed over these chunks. We require all devices to synchronously play out the k^{th} block during the k^{th} frame. Since our application is that of *live* streaming, the k^{th} block would be available at the server to disseminate to devices only within some short interval before the k^{th} frame. Hence, we assume that the k^{th} block is available at the server not before frame $k-2$. Thus, devices receive chunks of block k using their individual B2D interfaces during frame $k-2$. These chunks may be disseminated over the broadcast D2D network during frame $k-1$, and must be played out in frame k. Device i is able to play out block k, only if it has received enough degrees of freedom to decode the N chunks of the k^{th} block by the beginning of frame k. Otherwise, i will be idle during this frame.

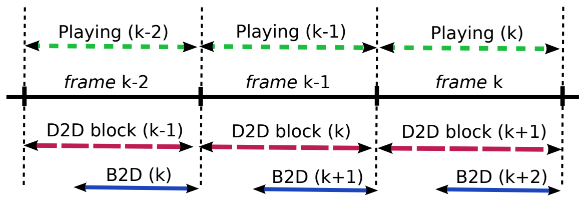

Figure 1.2: Sequence of transmissions over B2D and D2D networks. Each block must be delivered within two frames of its generation or lost.

We use a recent model of service quality for realtime wireless applications in which each station's QoS is parameterized by a *delivery-ratio*, which is the average ratio of blocks desired to the blocks generated [12]. For example, a delivery ratio of 90% would mean that 10% of the blocks can be skipped without noticeable impairment (which depends on the video coding used). We desire an algorithm that minimizes the usage of the expensive B2D connections, while maintaining an acceptable quality of service for each device. We note that the scenario is different from traditional peer-to-peer sharing in a wired network, wherein streaming with full chunk coordination using tree construction is the norm. In our situation, each device is independently connected to a base-station (perhaps using different service providers), and there is no coordination across base stations. The broadcast D2D medium is shared across the devices, and interference across the devices is a limiting factor.

The objectives of this work are three-fold. *First*, we would like to systematically design provably-cost-optimal chunk exchange algorithms that would minimize the usage of expensive B2D transmissions, while ensuring that quality constraints for live streaming can be met. *Second*, we desire to develop an application that implements our algorithms on Android smart phones in order to observe real-world behavior. *Third*, we would like to conduct performance analysis of our schemes on our Android application to understand how different parameters affect the algorithms. In what follows, we will describe these objectives in greater detail, and present our results in attaining each.

We also try to present an answer to the problem of why the devices should cooperate. It has been observed in deployed P2P systems, such as BitTorrent [4], that repeated interaction between peers enables them to cooperate via tit-for-tat strategies. Such strategies require that peers have fixed identities and retain histories of interactions with each other. However, such a scheme has several drawbacks. *First*, in a system with peer churn, each peer has to maintain histories for a large number of peers and calculate its optimal behavior with respect to each. *Second*, in a system of broadcast D2D transmissions, there is an incentive to try to obtain chunks without contributing to system welfare.

In short-range broadcast D2D networks, even if the energy cost of transmission is small enough to ignore, the opportunity cost of not being able to receive packets while transmitting could prevent collaboration. This cost can be high in the live streaming scenario, since receiving one additional chunk could mean the difference between success or failure in decoding a block. An ideal allocation algorithm would consider both the number of chunks that each peer device possesses and its deficit to ensure optimal system welfare. A fundamental problem is to incentivize peer devices to truthfully report their parameters so as to achieve such optimal welfare.

In addition to the above objectives, we also develop an incentive framework wherein each device truthfully reports the number of chunks that it receives via B2D and its deficit in each frame, so that a system-wide optimal allocation policy can be employed. Such an incentive framework should be lightweight and compatible with minimal amounts of history retention.

1.1 Related Work

The problem of exchanging random linear coded information while using the minimum number of transmissions over a broadcast D2D network (starting from some initial condition) was studied in [19, 15, 5]. There is also recent work on quick file transfer to a set of peer devices using a hybrid network, while minimizing the amount of B2D usage [1]. All consider a reliable model for the D2D links, and their common objective is to be able to decode a single block of information at the end. Unlike these papers, our objective and QoS metric—minimum cost timely synchronization of a live stream—is quite different.

There is a rich body of literature in the area of P2P multimedia streaming applications. Lava [20] and UUSee [14] are among the systems built based on the idea of network coding. While all the above models only consider unicast (wired) transmissions among peers, our D2D framework is one of the few models in this context that allows having broadcast transmissions over the D2D network.

Closest to our multimedia streaming model is [18] that investigates the problem of managing multiple interfaces for cooperatively sharing content over a D2D network. Unlike our block-by-block delay sensitive model for realtime streaming, they try to maximize a utility function of the average information flow rate achieved by the peers, which is relevant to elastic traffic such as file transfer or streaming of stored content.

The question of how to assign value to wireless broadcasts is intriguing. On the one hand, in [10] the game is a one-shot file transfer, and the objective is get nodes to reveal their true values of completion. On the other hand, [11] considers a problem of repeated interaction with time deadlines by which each node needs to receive a packet. Each node declares its readiness to help others after waiting for a while; the game lies in choosing this time optimally, and the main result is to characterize the price of anarchy that results. However, devices do not estimate the future while taking actions.

2. SYSTEM MODEL

2.1 System Model

We have M co-located peer devices in our system, denoted by $i \in \{1, \ldots, M\}$, all interested in synchronously receiving the same stream of data. The data source generates the stream in the form of a sequence of blocks. Each block is further divided into N chunks for transmission. We use random linear network coding over the chunks of each block, which implies that each coded chunk (a degree of freedom) is now a random linear combination (with coefficients in finite field F_q of size q) of the N original chunks in the corresponding block, and can be represented by a vector in F_q^N.

Time is divided into slots, and at each time slot τ, each device can receive up to one chunk using its B2D interface, and one using its D2D interface. We assume that the two interfaces are orthogonal. Also, both B2D unicasts and D2D broadcasts are made in a connection less fashion with no feedback. The probabilities of success when using each interface are different, and we can use these probabilities to reflect the relative throughput of the two interfaces.

Thus, each device has an expensive but unreliable B2D unicast channel to a base-station, whose success probability depends on the loss probability over the Internet as well as the condition of the B2D channel. For each device i, we model the chunk reception event for the B2D interface by a Bernoulli random variable with parameter β_i, independent of the other devices. Each device also has a low-cost D2D broadcast interface, and only one device (denoted by $u[\tau] \in \{1, \ldots, M\}$) can broadcast over

*Reprinted with permission from "Realtime Streaming with Guaranteed QoS over Wireless D2D Networks" by Navid Abedini, Swetha Sampath, Rajarshi Bhattacharyya, Suman Paul, Srinivas Shakkottai, *Mobihoc 2013*, pages 197-206, Bangalore, India, Aug. 2013, Copyright 2013 ACM 978-1-4503-2193-8/13/07.

the D2D network at each time τ. Losses over the D2D channel could happen due to collisions that occur as an overhead of distributed scheduling. Hence, we assume that a D2D broadcast is either received by all devices with a probability equal to δ, or lost[1]. Wireless devices could apply channel estimation techniques to determine their success probabilities β_i and δ, and we will develop algorithms assuming that these values are known. We will also study a system (similar to the ones investigated in [19, 15, 5]) in which the D2D broadcast is assumed to be completely reliable.

Since each D2D broadcast is either received by all devices or none, there is no need to rebroadcast any information received via D2D. Following this observation, it is straightforward to verify that the order of D2D transmissions has no impact on the performance. Thus, we only need to keep track of the number of chunks transmitted and received over the interfaces during a frame in order to determine the state of the system. We denote the total number of coded chunks of block k delivered to device i via the B2D network during frame $k-2$ using $b_i^{(k)}$. Also, $t_i^{(k)}$ and $r_i^{(k)}$ are used to denote, respectively, the total number of transmitted and received chunks of block k by device i via D2D during frame $k-1$ (*i.e.,* before frame k, which is the block k's play out time). Note that according to the model

$$\textstyle\sum_i t_i^{(k)} \leq T, \tag{2.1}$$

i.e., only one device can broadcast over the D2D network at each slot, and hence at most T transmissions can occur during frame $k-1$.

Let $\hat{n}_i^{(k)}$ denote the total number of coded chunks of block k possessed by device

[1]It is straightforward to extend our results and framework to a more general D2D model in which only a subset of devices receive each broadcast chunk, but at the cost of greater algorithmic complexity.

i at the beginning of frame k,

$$\hat{n}_i^{(k)} = b_i^{(k)} + r_i^{(k)}.$$

Hence, device i has a set of $\hat{n}_i^{(k)}$ vectors of coefficients corresponding to the coded chunks that it has received. We denote by $n_i^{(k)}$ (called the k^{th} *rank* of device i) the dimension of the subspace spanned by these vectors. In order to decode the original N chunks of block k at the beginning of frame k, device i must have $n_i^{(k)} = N$. Otherwise, i skips the block and will be idle during this frame. Let $R_i[k] \in \{0, 1\}$ denote the success or failure of the k^{th} block w.r.t i, *i.e.,* $R_i[k] = 0$ means device i is idle during frame k, and $R_i[k] = 1$ denotes the event of successful decoding of block k. Clearly, $R_i[k] = 0$, if $n_i^{(k)} < N$.

Each device i has a delivery ratio $\eta_i \in (0, 1]$, which is the minimum acceptable long-run average number of frames device i must playout. Hence, we require

$$\eta_i \leq \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}[R_i[k]]. \tag{2.2}$$

The objective of this paper is to find a scheme to coordinate both interfaces that would satisfy the delivery ratio requirement of all devices at the lowest cost of using B2D transmissions. Since the B2D and D2D transmissions corresponding to a particular block k occur during the disjoint frames $k-2$ and $k-1$, respectively, we will explore schemes that consist of two parts, namely:

1. **B2D Stopping Time:** We use a fixed number of transmissions on the B2D channel for each block and for each device. Hence, for each block k, device i receives some random number $\mathbf{b}_i^{(k)}$ of coded chunks via B2D at the beginning of frame $k-1$. By assumption, the values of $\mathbf{b}_i^{(k)}$ are independent over devices.

Also, since we fix the B2D usage in each frame, $\mathbf{b}_i^{(k)}$ are independent and identically distributed over blocks k for each device i. We need to determine the minimum stopping time such that delivery requirements can be met.

2. **D2D Scheduling Algorithm:** Given the B2D stopping time, each device receives chunks at the beginning of each frame in an *i.i.d* fashion according to an arrival process denoted by $\mathbf{b}_i$. We must determine a scheduling algorithm that can achieve the delivery requirements if at all it is possible to do so, given the D2D broadcast channel reliability.

In what follows, we will first solve the problem of D2D scheduling, and will then show how to select the B2D stopping time. We will do so under different assumptions on channel reliability and coding schemes.

2.2 D2D Scheduling Algorithm

Under the assumption of *i.i.d* B2D arrivals, we first need to determine whether the desired QoS metric $(\eta_1, ..., \eta_M)$ is achievable for the given arrival process $(\mathbf{b}_1, ..., \mathbf{b}_M)$. For large field sizes, distinct randomly coded chunks are linearly independent with high probability. In what follows, we assume that the field size q is large enough that we can ignore the effect of its finiteness on the performance of the linear coding.

Definition 1 *We say the QoS $(\eta_1, ..., \eta_M)$ is achievable, if there exists a feasible policy to coordinate the D2D transmissions such that, on average each device i successfully receives η_i fraction of the blocks before their deadlines.*

Since the B2D arrivals are assumed to be *i.i.d.* over frames, the achievability of the QoS metric can be evaluated based on the existence of a randomized stationary D2D policy as follows:[2]

[2]It is easy to verify this characterization of achievability, which is used in much recent work. For instance, a detailed proof can be found in [16].

Lemma 1 *The QoS $(\eta_1, ..., \eta_M)$ is (strictly) achievable if and only if there exists a D2D policy $\mathbf{P}^*$ such that,*

for each block k, given the B2D arrivals $b^{(k)} = (b_1^{(k)}, ..., b_M^{(k)})$, $\mathbf{P}^*$ chooses a feasible D2D schedule $t^{(k)} = (t_1^{(k)}, ..., t_M^{(k)})$ (satisfying (2.1)) with probability $\mathbb{P}\left(t^{(k)}|b^{(k)}\right)$, such that for each device i,

$$\mathbb{E}_{\mathbf{b}}\left[\sum_{t^{(k)}} \mathbb{E}\left[R_i[k] \mid t^{(k)}, b^{(k)}\right] \mathbb{P}\left(t^{(k)}|b^{(k)}\right)\right] > \eta_i \tag{2.3}$$

where $\mathbb{E}_{\mathbf{b}}[.]$ is expectation with respect to the B2D arrival processes.

2.2.1 Optimal D2D Scheme

In order to keep track of quality of service, each device maintains a so-called deficit queue. The length of this queue $d_i[k]$, for device i at frame k, follows the dynamic below

$$d_i[k] = d_i[k-1] + \eta_i - R_i[k]. \tag{2.4}$$

Recall that $R_i[k] = 0$ means device i is not successful in receiving the block k before its deadline, and hence is idle in frame k. In this case, the deficit value of this device increases by an amount equal to its delivery ratio η_i. Otherwise, $R_i[k] = 1$ and the deficit value decreases by $1 - \eta_i$. Therefore, the deficit queue can be thought of as a counter that captures the accumulated "unhappiness" of a device experienced thus far,

$$d_i[k] = k\eta_i - \sum_{l=1}^{k} R_i[l].$$

The evolution of these deficit queues can be understood by using a Markov chain $\mathcal{D}$, whose state at each step k is $([d_1[k]]^+, ..., [d_M[k]]^+)$, where $[a]^+ = \max\{a, 0\}$. If our D2D algorithm is such that $\mathcal{D}$ is stable (positive recurrent), then $\mathbb{E}[[d_i[k]]^+] < \infty$ for all k, and we will have

$$\lim_{K\to\infty} \frac{1}{K}\mathbb{E}[d_i[k]] \leq \lim_{K\to\infty} \frac{1}{K}\mathbb{E}[[d_i[k]]^+] = 0.$$

Hence, $\eta_i \leq \lim_{K\to\infty} \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}[R_i[k]]$, which implies that the QoS requirement of device i is satisfied. We next define a D2D scheme in Algorithm 1, whose optimality is shown in Theorem 2.

Algorithm 1 Optimal D2D scheme (unreliable broadcast)

At the beginning of each frame $k-1$: Given the B2D arrivals $(b_1^{(k)}, ..., b_M^{(k)})$ and the deficit values $([d_1[k-1]]^+, ..., [d_M[k-1]]^+) = (d_1, ..., d_M)$, solve the following maximization problem to find the optimal number of transmissions $(t^{(k)})^* = ((t^{(k)})^*_1, ..., (t^{(k)})^*_M)$:

$$\begin{gathered} \max \textstyle\sum_i d_i \mathbb{P}(\sum_{j\neq i} \min(\hat{t}_j^{(k)}, b_j^{(k)}) \geq N - b_i^{(k)}) \\ \text{s.t.} \\ \hat{t}_j^{(k)} = Bin(\delta, t_j^{(k)}), \quad \forall j \\ \textstyle\sum_j t_j^{(k)} \leq T \end{gathered} \tag{2.5}$$

Devices take turns (in any arbitrary order) to broadcast over the D2D network: Device i first broadcasts $\min(b_i^{(k)}, (t^{(k)})^*_i)$ of its initial coded chunks received from the B2D network. For the remaining $\max((t^{(k)})^*_i - b_i^{(k)}, 0)$ transmissions, device i randomly combines the initial $b_i^{(k)}$ B2D chunks.

Theorem 2 *The D2D scheme in Algorithm 1 is throughput optimal,* i.e., *it can satisfy* all *achievable QoS metrics* $(\eta_1, ..., \eta_M)$.

Proof In this proof, we will use the Lyapunov criterion [8] to show the stability of Markov chain $\mathcal{D}$. Consider the candidate Lyapunov function $V[k] = \frac{1}{2}\sum_i([d_i[k]]^+)^2$.

We will show that for any achievable QoS, the proposed D2D algorithm results in an expected drift $\Delta V[k] =$

$$\mathbb{E}\left[V[k] - V[k-1] \mid \text{state of the system at frame } k-1\right],$$

which is negative except in a finite subset of the state space. Hence, the Lyapunov Theorem implies that the Markov chain $\mathcal{D}$ is stable. Now, we present the details of the proof.

$$\begin{aligned}
&\Delta V[k] = \mathbb{E}\left[V[k] - V[k-1] \middle| [d_i[k-1]]^+ = d_i : \forall i\right] \\
&= \tfrac{1}{2}\mathbb{E}\left[\sum_i \left([d_i[k-1] + \eta_i - R_i[k]]^+\right)^2 \middle| [d_i[k-1]]^+\right] \\
&-\tfrac{1}{2}\sum_i (d_i)^2 \overset{(a)}{\leq} \tfrac{1}{2}\mathbb{E}\left[\sum_i (d_i + \eta_i - R_i[k])^2 - (d_i)^2\right] \\
&= \mathbb{E}\left[\sum_i d_i\left(\eta_i - R_i[k]\right)\right] + \tfrac{1}{2}\mathbb{E}\left[\sum_i (\eta_i - R_i[k])^2\right] \\
&\overset{(b)}{\leq} M/2 + \sum_i d_i\eta_i - \mathbb{E}\left[\sum_i d_i R_i[k]\right],
\end{aligned}$$

where (a) follows from $([X+Y]^+)^2 \leq ([X]^+ + Y)^2$, and (b) holds since $(\eta_i - R_i[k])^2 \leq \max((\eta_i)^2, (1-\eta_i)^2) \leq 1$.

In order to get a negative drift (except in a finite subset), we minimize the above upperbound. Hence, at the beginning of each frame $k-1$, for given deficit values $([d_1[k-1]]^+, ..., [d_M[k-1]]^+) = (d_1, ..., d_M)$ and any realization of the B2D arrivals $(b_1^{(k)}, ..., b_M^{(k)})$, we solve the following

$$\max_{t^{(k)}} \ \sum_i d_i \mathbb{E}\left[R_i[k] \mid t^{(k)}, b^{(k)}\right], \tag{2.6}$$

to find the optimal schedule $(t^{(k)})^*$. Note that policy $\mathbf{P}^*$ (in Lemma 1) randomly chooses a feasible $t^{(k)}$ according to some distribution $\mathbb{P}\left(t^{(k)} | b^{(k)}\right)$, hence one can easily verify that

$$\begin{aligned}&\textstyle\sum_i d_i \mathbb{E}\left[R_i[k] \mid (t^{(k)})^*, b^{(k)}\right] \geq \\ &\textstyle\sum_{t^{(k)}} \sum_i d_i \mathbb{E}\left[R_i[k] \mid t^{(k)}, b^{(k)}\right] \mathbb{P}\left(t^{(k)} | b^{(k)}\right).\end{aligned} \tag{2.7}$$

Taking expectation on both sides of the above inequality with respect to the arrival processes results in

$$\begin{aligned}&\textstyle\sum_i d_i \mathbb{E}\left[R_i[k] \mid (t^{(k)})^*\right] \\ &\geq \textstyle\sum_i d_i \mathbb{E}\left[\sum_{t^{(k)}} \left[R_i[k] \mid t^{(k)}, b^{(k)}\right] \mathbb{P}\left(t^{(k)} | b^{(k)}\right)\right] \\ &\overset{(c)}{>} \textstyle\sum_i d_i \eta_i \geq \sum_i d_i(\eta_i + \epsilon)\end{aligned} \tag{2.8}$$

for some small enough $\epsilon > 0$, where (c) follows from (2.3). By considering (2.8) in inequality (b), we conclude that using schedule $(t^{(k)})^*$ will result in

$$\Delta V[k] \leq \frac{M}{2} - \epsilon \sum_i d_i,$$

which is negative for large enough d_i values, and hence can stabilize the deficit queues.

We have just shown that we can satisfy any achievable QoS metric by using the schedule obtained by solving (2.6) for each frame. In what follows, we will show (2.6) is equivalent to the optimization problem (2.5) that appears in Algorithm 1.

Recall that $R_i[k] = 1_{\{n_i^{(k)} = N\}}$, where the indicator variable $1_{\{A\}}$ is equal to 1 if A is true, and 0 otherwise. Hence, $\mathbb{E}\left[R_i[k]\right] = \mathbb{P}\left(n_i^{(k)} = N\right)$. Suppose device j broadcasts $t_j^{(k)}$ chunks generated from random linear combinations of its initial $b_j^{(k)}$ chunks. Since each transmission is successful with probability δ, the number of successful transmissions by device j is distributed as a Binomial random variable $\hat{t}_j^{(k)} = Bin(\delta, t_j^{(k)})$ with parameters δ and $t_j^{(k)}$.

Now, we have assumed that the random linear coding is performed over a field of sufficiently large size q such that N distinct randomly coded chunks are linearly independent. Consequently, each successful transmission increases the rank of the receiving devices by one. Thus, the transmissions by device j introduces $\min(\hat{t}_j^{(k)}, b_j^{(k)})$ new degrees of freedom (DoF) at the other devices. Also, device i is full-rank (*i.e.,* $n_i^{(k)} = N$) if and only if it has received at least $N - b_i^{(k)}$ new DoFs from other devices, that is $b_i^{(k)} + \sum_{j \neq i} \min(\hat{t}_j^{(k)}, b_j^{(k)}) \geq N$. Therefore, we have

$$\begin{aligned} \mathbb{E}\left[R_i[k]\right] &= \mathbb{P}\left(n_i^{(k)} = N\right) \\ &= \mathbb{P}\left(b_i^{(k)} + \textstyle\sum_{j \neq i} \min(\hat{t}_j^{(k)}, b_j^{(k)}) \geq N\right). \end{aligned} \tag{2.9}$$

Substituting (2.9) in (2.6) results in the optimization (2.5).

We have just shown the throughput optimality of Algorithm 1 for the unreliable D2D case. However, we note that in practice we need a central entity with significant computational capabilities to implement this optimal algorithm, since it takes a combinatorial form. In the following section, we will consider the special case when D2D broadcast is completely reliable. We will show that the optimal algorithm under this case possesses an intuitively appealing form that lends itself to easy implementation. We also will develop a heuristic modification of this algorithm when the D2D broadcast is not reliable, which does not have the complexity of the optimal algorithm.

2.3 D2D Algorithm Under Reliable Broadcast

In this section, we assume that D2D broadcasts are always successfully received by all intended recipients. Of course, the constraint that only one device can broadcast at a time still applies. As before, we first model the B2D arrivals as *i.i.d.* random variables $\mathbf{b}_i$ for each device i, and focus on scheduling the D2D transmissions. Since broadcasts are deterministically successful, $\sum_{j \neq i} t_j^{(k)} = r_i^{(k)}$. In other words, the total number of D2D chunks received by device i is equal to the number of transmissions performed by all other devices. Also, each device i can transmit at most $b_i^{(k)}$ times during frame $k-1$, since any further transmissions by i will not add to other devices' information, *i.e.,*

$$t_i^{(k)} \leq b_i^{(k)}. \tag{2.10}$$

Consequently, it is easy to see that $n_i^{(k)} = \min\{N, \hat{n}_i^{(k)}\}$ holds for large field size q, *i.e.,* device i will obtain full-rank if it receives at least a total of N coded chunks from B2D and D2D interfaces.

Further, since the chunks received from the B2D interfaces are initially randomly coded, combining them further with random coefficients cannot improve performance. Hence, at time τ, the device chosen to transmit, $u[\tau]$, can simply broadcast any of the chunks received via its B2D interface, which it has not transmitted yet. We are primarily interested in the case where $N > T$, because otherwise there are enough number of time slots in each frame for devices to broadcast all N degrees of freedom and hence the optimal D2D scheme becomes trivial.

2.3.1 Achievability of QoS Metric

In Section 2.2, Lemma 1 implicitly characterizes the achievability conditions of a given QoS requirement $(\eta_1, ..., \eta_M)$, and can be applied for the reliable D2D case as well. However, we will see below that the QoS achievability condition can be determined explicitly based on a set of necessary and sufficient conditions for this case.

Devices have T slots in each frame to exchange the received B2D chunks. Hence, each device i can potentially recover block k, only if (i) it has received enough B2D chunks initially (i.e, $b_i^{(k)} \geq N - T$) and (ii) the whole system is full-rank at the beginning of the frame (i.e, $\sum_j b_j^{(k)} \geq N$). Therefore, for each device i we have,

$$R_i[k] \leq 1_{\{b_i^{(k)} \geq N-T,\ \sum_j b_j^{(k)} \geq N\}}. \tag{2.11}$$

Since $(b_1^{(k)}, ..., b_M^{(k)})$ is assumed to be identically and independently distributed across frames according to $(\mathbf{b}_1, ..., \mathbf{b}_M)$, from (2.2) we get the following necessary condition on the achievability of η_i :

$$\eta_i \leq \mathbb{P}\left(\mathbf{b}_i \geq N-T,\ \sum_j \mathbf{b}_j \geq N\right). \tag{2.12}$$

Let $N_s(b_1^{(k)}, ..., b_M^{(k)}) = \sum_i R_i[k]$ be the number of devices that successfully receive the whole block k, given the B2D arrivals $(b_1^{(k)}, ..., b_M^{(k)})$. The following lemma provides an upper bound on $N_s(b_1^{(k)}, ..., b_M^{(k)})$.

Lemma 3 *Given the B2D arrivals $(b_1^{(k)}, ..., b_M^{(k)})$, we have*

$$\begin{aligned} &N_s(b_1^{(k)}, ..., b_M^{(k)}) \\ &\quad \leq N_s^*(b_1^{(k)}, ..., b_M^{(k)}) \\ &\quad \equiv \frac{\min\left(|\mathcal{S}|(N-T), \left[\sum_i b_i^{(k)} - T\right]^+\right)}{N-T}, \end{aligned} \tag{2.13}$$

where $\mathcal{S} = \{i \in \{1, ..., M\} : N - b_i^{(k)} \leq T,\ b_i^{(k)} + \sum_{j \neq i} b_j^{(k)} \geq N\}$.

Proof Since $n_i^{(k)} \leq \hat{n}_i^{(k)}$, we have

$$R_i[k] = 1_{\{n_i^{(k)} \geq N\}} \leq 1_{\{\hat{n}_i^{(k)} \geq N\}} = 1_{\{b_i^{(k)} - t_i^{(k)} + \sum_j t_j^{(k)} \geq N\}}.$$

Therefore, we can solve the following maximization problem in order to find an upper bound on $N_s(b_1^{(k)}, ..., b_M^{(k)})$,

$$\begin{aligned} \max \ & \textstyle\sum_i 1_{\{b_i^{(k)} - t_i^{(k)} + \sum_j t_j^{(k)} \geq N\}} \\ \text{s.t.} \ & t_i^{(k)} \leq b_i^{(k)} \qquad \forall i \\ & \textstyle\sum_j t_j^{(k)} \leq T \end{aligned} \tag{2.14}$$

The first constraint implies $\sum_j t_j^{(k)} \leq \sum_j b_j^{(k)}$. Hence, to achieve the maximum objective we let $\sum_j t_j^{(k)} = \min(\sum_j b_j^{(k)},\ T)$.

We partition the set of devices $\{1, ..., M\}$ into sets $\mathcal{S}$ and $\mathcal{S}^c = \{1, ..., M\} \backslash \mathcal{S}$. Here, $\mathcal{S}^c$ is the set of devices which, either individually or collectively, have not received enough number of B2D arrivals to possibly recover the block, and $R_i[k] = 0$ for $i \in \mathcal{S}^c$.

Suppose $\sum_j b_j^{(k)} < N$. Then we have $N_s(b_1^{(k)}, ..., b_M^{(k)}) = |\mathcal{S}| = 0$. Otherwise $\sum_j b_j^{(k)} \geq N$, and with our assumption that $T < N$, the optimization in (2.14) can be rewritten as

$$\begin{aligned} \max \ & \textstyle\sum_{i \in \mathcal{S}} 1_{\{b_i^{(k)} - t_i^{(k)} \geq N - T\}} \\ \text{s.t.} \ & t_i^{(k)} \leq b_i^{(k)} \qquad \forall i \\ & \textstyle\sum_j t_j^{(k)} = \min(\sum_j b_j^{(k)},\ T) = T. \end{aligned} \tag{2.15}$$

The maximum value above can be shown to be

$$\left\lfloor \frac{\min\left(|\mathcal{S}|(N-T), \left[\sum_i b_i^{(k)} - T\right]^+\right)}{N - T} \right\rfloor.$$

Consequently, we obtain

$$N_s(b_1^{(k)}, ..., b_M^{(k)}) \leq \frac{\min\left(|\mathcal{S}|(N-T), \left[\sum_i b_i^{(k)} - T\right]^+\right)}{N-T}.$$

Now, from Lemma 3 and (2.2), since $(b_1^{(k)}, ..., b_M^{(k)})$ is *i.i.d* over frames, the following necessary condition on $\sum_i \eta_i$ can be obtained:

$$\sum_i \eta_i \leq \mathbb{E}\left[\lfloor N_s^*(\mathbf{b}_1, ..., \mathbf{b}_M)\rfloor\right]. \tag{2.16}$$

The following theorem summarizes our results.

Theorem 4 *The QoS metric $(\eta_1, ..., \eta_M)$ is achievable with respect to* i.i.d *B2D arrivals $(\mathbf{b}_1, ..., \mathbf{b}_M)$ if and only if the following conditions are satisfied*

$$\begin{array}{ll} (C1)\ \forall i: & \eta_i \leq \mathbb{P}\left(\mathbf{b}_i \geq N-T,\ \sum_j \mathbf{b}_j \geq N\right) \\ (C2) & \sum_i \eta_i \leq \mathbb{E}\left[\lfloor N_s^*(\mathbf{b}_1, ..., \mathbf{b}_M)\rfloor\right]. \end{array} \tag{2.17}$$

Further, we can show that for the symmetric case where $\eta_i = \eta$, and $\mathbf{b}_i$ are identically distributed for all devices i, the necessary and sufficient condition reduces to

$$(C2') \quad \eta \leq \tfrac{1}{M}\mathbb{E}\left[\lfloor N_s^*(\mathbf{b}_1, ..., \mathbf{b}_M)\rfloor\right]. \tag{2.18}$$

Proof The necessity part was shown in (2.12) and (2.16). To prove the sufficiency of these conditions, we will propose an algorithm in the next subsection which can fulfill any QoS constraints satisfying $(C1)$ and $(C2)$.

2.3.2 Optimal Reliable D2D Scheme

In this subsection, we propose a simple algorithm that can achieve any QoS metric satisfying the conditions in (2.17). As a result, $(C1)$ and $(C2)$ are sufficient conditions on achievability of a QoS metric (Theorem 4). Also, the proposed algorithm is throughput optimal.

We follow the same approach as in Section 2.2 to study the D2D system using deficit queues $d_i[k]$. Algorithm 2 describes an optimal D2D scheme that can stabilize these deficit queues for any achievable QoS metrics.

Algorithm 2 Optimal D2D scheme (reliable broadcast)

```
At the beginning of each frame $k-1$, given the arrivals $(b_1^{(k)}, ..., b_M^{(k)})$:
Partition the devices into sets $\mathcal{S} = \{i \in \{1, ..., M\} : N - b_i^{(k)} \leq T, b_i^{(k)} + \sum_{j \neq i} b_j^{(k)} \geq N\}$ and $\mathcal{S}^c$.
If $\mathcal{S} = \emptyset$, nobody can get full-rank. Otherwise,
Phase 1) Let all the devices in $\mathcal{S}^c$ transmit all they have initially received for the next $T_1 = \min\{\sum_{i \in \mathcal{S}^c} b_i^{(k)}, T\}$ slots.
If there exist time and a need for more transmissions,
Phase 2) Let each device $i \in \mathcal{S}$ transmit up to $b_i^{(k)} + T - N$ of its initial chunks.
Phase 3) While there exist time and a need for more transmissions, let devices in $\mathcal{S}$ transmit their remaining chunks in an increasing order of their deficit values.
```

Theorem 5 *The D2D scheme in Algorithm 2 is throughput optimal when the D2D transmissions are reliable.*

Proof As in the proof of Theorem 2, we use the Lyapunov criterion. Equivalent to the optimization in (2.6), we need to solve the following

$$\begin{aligned} \max\ & \textstyle\sum_i d_i R_i[k] = \sum_i d_i 1_{\{b_i^{(k)} + \sum_{j\neq i} t_j^{(k)} \geq N\}} \\ \text{s.t.}\quad & t_i^{(k)} \leq b_i^{(k)} \qquad \forall i \\ & \textstyle\sum_i t_i^{(k)} \leq T. \end{aligned} \tag{2.19}$$

The optimization in (2.19) is similar to the one in (2.14). Hence, we can apply a similar argument and verify that the following optimization problem is equivalent to (2.19),

$$\begin{aligned} \max\ & \textstyle\sum_i d_i z_i \\ \text{s.t.}\quad & z_i \leq 1_{\{b_i^{(k)} \geq N-T,\ \sum_j b_j^{(k)} \geq N\}} \qquad \forall i \\ & \textstyle\sum_i z_i \leq N_s^*(b_1^{(k)}, ..., b_M^{(k)}), \end{aligned} \tag{2.20}$$

where $N_s^*(b_1^{(k)}, ..., b_M^{(k)})$ is defined in (2.13).

Using the solution to the above maximization in the drift formula $\Delta V[k]$ will result in

$$\begin{aligned} \Delta V[k] &\leq M/2 + \textstyle\sum_i d_i \eta_i - \mathbb{E}\left[\max \sum_i d_i z_i\right] \\ &\leq M/2 + \textstyle\sum_i d_i \eta_i - \max \sum_i d_i \mathbb{E}\left[z_i\right]. \end{aligned} \tag{2.21}$$

From (2.20), we notice that $\mathbb{E}\left[z_i\right]$ must satisfy

$$\begin{aligned} \mathbb{E}[z_i] &\leq \mathbb{P}\left(b_i^{(k)} \geq N-T,\ \textstyle\sum_j b_j^{(k)} \geq N\right) \\ \textstyle\sum_i \mathbb{E}[z_i] &\leq \mathbb{E}\left[N_s^*(b_1^{(k)}, ..., b_M^{(k)})\right]. \end{aligned} \tag{2.22}$$

In Subsection 2.3.1, we have shown that an achievable QoS metric $(\eta_1, ..., \eta_M)$ needs to satisfy the above conditions. This suggests that for a strictly achievable

QoS metric $(\eta_1, ..., \eta_M)$, for which these conditions hold with strict inequalities, there exists some $\epsilon > 0$ such that

$$\max \sum_i d_i \mathbb{E}[z_i] \geq \sum_i d_i \eta_i (1+\epsilon). \tag{2.23}$$

Consequently, the drift in (2.21) reduces to

$$\Delta V[k] \leq M/2 - \epsilon \textstyle\sum_i d_i \eta_i. \tag{2.24}$$

Thus, for large enough deficit values d_i, the drift is negative. This means the D2D scheme implied by solving (2.19) at each frame can satisfy any achievable QoS metric, and hence is optimal. Furthermore, it proves the sufficiency of the conditions in Theorem 4 on the achievability of a QoS metric.

The argument showing that the D2D scheme in Algorithm 2 actually solves the optimization problem in (2.19) is straightforward, and follows from dividing devices into those that cannot complete (and so might as well transmit all that they have), then considering those that can complete (and hence can transmit a limited number of chunks), and finally considering those with the largest deficit (and so should stop transmitting after phase two).

2.4 Selection of B2D Stopping Time

In this section, we would like to determine the number of time slots that the B2D interface should be used by each device over each frame. We consider this as an offline stopping time problem, and will show how to calculate this value. Thus, we want to find $0 \leq T_B(i) \leq T$, which is the number of times in each frame that device i attempts to receive a coded chunk from the server using the B2D interface.

The B2D usage times should be such that the QoS target can be met with the lowest total cost $C(T_B(1), ..., T_B(M))$, where $C(.)$ is a general cost function. Note that the function could be linear $(\sum_i T_B(i))$, if usage-based costs are considered. Furthermore, it makes intuitive sense that devices that desire a higher QoS η_i, should contribute more to the system so as to maintain a level of fairness. This requirement can also be captured in the cost function, by simply choosing weights for each device that are dependent on its desired QoS value.

Recall that the B2D interface of each device i is assumed to be Bernoulli with success probability β_i. This means that the B2D arrivals $b_i^{(k)}$ to each device i are independently and identically distributed over frames k as a Binomial random variable $Bin(\beta_i, T_B(i))$ with parameters β_i and $T_B(i)$:

$$\mathbb{P}(b_i^{(k)} = a) = 1_{\{0 \leq a \leq T_B(i)\}} \binom{T_B(i)}{a} \beta_i^a (1 - \beta_i)^{T_B(i) - a}.$$

For the fully reliable D2D scenario, we can determine the achievability of a QoS requirement based on conditions $(C1)$ and $(C2)$ in Theorem 4. In order to achieve a given QoS metric $(\eta_1, ..., \eta_M)$, $T_B(i)$ values must be large enough such that these conditions are satisfied. Hence, the optimal values of $T_B^*(i)$ can be obtained by solving the following problem:

$$\begin{aligned} &\min C(T_B(1), ..., T_B(M)) \\ &\text{s.t.} \\ &0 \leq T_B(i) \leq T \qquad \text{for all } i \\ &\mathbf{b}_i = Bin(\beta_i, T_B(i)) \quad \text{for all } i \\ &(C1) \text{ and } (C2) \text{ in (2.17) are satisfied.} \end{aligned} \tag{2.25}$$

Note that the minimization problem in (2.25) does not have a simple form that can be solved efficiently. However, since the region for feasible $T_B(i)$ values is finite (*i.e.,*

$\{0, 1, ..., T\}^M$), and the values need to be determined only once for the given set of system parameters, we could conduct an exhaustive search to find the optimal $T_B^*(i)$ values. We can also improve the search algorithm by applying more efficient search methods like *branch-and-bound*.

In the case of the unreliable D2D channel, there is only an implicit achievability condition as given by Lemma 1. We therefore cannot analytically solve for the optimal B2D usage times. Since $T_B(i)$ values are to be calculated in an offline manner, we can run stochastic simulations of the optimal D2D scheme presented in Algorithm 1 in order to find the optimal values numerically.

3. INCENTIVE MECHANISM

We consider two different streaming settings as follows:

Static System Model: We consider a single D2D cluster consisting of co-located agents that are all interested in a common content stream. Examples include friends viewing a live event in a social setting, or a real-time update system for first responders. A fixed number of agents is assumed to be active at any one time, and devices enter and leave the cluster dynamically as agents arrive and depart.

Mobile User Model: We consider a large number of D2D clusters, each with a fixed number of agents, and with all clusters interested in the same content stream. Examples of such settings are sports stadia, concerts or protest meetings, where a large number of agents gather together, and desire to receive the same live-stream (replays, commentary, live video *etc.*). Devices move between clusters as agents move around.

We denote the total number of coded chunks of block k delivered to device i via the B2D network during frame $k-2$ using $b_i[k] \sim \zeta$. We call the vector consisting of the number of transmissions by each device over frame $k-1$ as the "allocation" pertaining to block k, denoted by $\mathbf{a}[k]$. Also, we denote the number received chunks of block k by device i via D2D during frame $k-1$ using $g_i[k]$. Due to the large field size assumption, if $b_i[k]+g_i[k]=N$, it means that block k can be decoded, and hence can be played out.

Each device i has a delivery ratio $\eta_i \in (0,1]$, which is the minimum acceptable long-run average number of frames device i must playout. It is straightforward to extend our results to the case where delivery ratios are drawn from some finite set of values. The device maintains a deficit queue with length $d_i[k] \in \mathbb{R}^+$, which keeps track of the current deficit. If a device fails to decode a particular block, its deficit increases by η_i, else it decreases by $1 - \eta_i$. The impact of deficit on the user's quality of experience is modeled by a function $c(d_i[k])$, which is convex, differentiable and monotone increasing. The idea is that as more and more blocks are skipped, user unhappiness increases more with each additional skipped block.

An allocation rule maps the values of the B2D chunks received and deficits as revealed by agents, denoted by $\boldsymbol{\theta}[k] := (\mathbf{b}[k], \mathbf{d}[k-1])$, to an allocation for that frame $\mathbf{a}[k]$. Given an allocation, agents have no incentive to deviate from it, since an agent that does not transmit the allocated number of chunks would see no benefit; those time slots would have no transmissions by other agents either. The question is: how do we incentivize the agents to reveal their states truthfully so that the constructed allocation can maximize system welfare?

Once we have a mechanism that promotes truth telling, any agent can calculate the transfer to be levied on each agent for that frame. In practice, we use a single trusted device to report these values to keep feedback to the server low. We visualize these transfers in terms of a token scheme constructed by the content provider. The provider issues these tokens to subsidize the operation of the D2D scheme, and in turn reduces the amount of server and communication capacity that it has to install. The agents benefit both from reduced usage of expensive B2D data, and also could reimburse tokens for purchasing content from the provider.

3.1 Static System Model

We start by describing the single cluster static system model which has M agents. At the end of a frame, any agent i can leave the cluster only to be replaced by an a new agent (also denoted by i) whose initial deficit is drawn from a (cumulative) distribution Ψ with support $\mathbb{R}^+$; in the analysis Ψ is assumed to have a point mass only at 0 and a bounded density function for the remainder. The regeneration event occurs with probability $\bar{\delta} = (1-\delta)$ independently for each agent, so that the lifetimes of the agents are geometrically distributed. As described in the previous section, we assume that the number of chunks received via B2D for agent i in frame k, denoted by $b_i[k]$, is chosen in an *i.i.d.* fashion according to the (cumulative) distribution ζ, with support $\mathbb{Z}^+$; one such distribution is the binomial distribution.

Given the system state at the end of frame $k = 0, 1, \ldots$, i.e., $\tilde{\boldsymbol{\theta}}[k] = (\mathbf{b}[k], \mathbf{d}[k])$ at frame k, our objective is to achieve

$$\begin{aligned} W(\tilde{\boldsymbol{\theta}}[k]) &= W(\mathbf{b}[k], \mathbf{d}[k]) \\ &= \min_{\{\mathbf{a}[l]\}_{l=k}^{\infty}} E\left\{ \sum_{l=k}^{\infty} \delta^{l-k} \sum_{i=1}^{M} c(d_i[l]) \right\} \end{aligned} \tag{3.1}$$

where $c(\cdot)$ is the holding cost function that is assumed to be strictly convex and monotone increasing, and $\mathbf{a}[\cdot]$ is the vector of allocations, i.e., number of transmissions for the different agents. It will, instead, be easier to work with the system state at the beginning of a frame $k = 1, 2, \ldots$, which we denote by $\boldsymbol{\theta}[k] := (\mathbf{b}[k], \mathbf{d}[k-1])$.

Given the packets received via B2D, the deficit queue values and the allocation $\mathbf{a}$, the deficit evolves as

$$d_i[k] = \big(d_i[k-1] + \eta_i - \chi_i(\mathbf{a}, \theta_i[k])\big)^+, \tag{3.2}$$

when the agent remains in the system and where $(\cdot)^+ := \max(0, \cdot)$. Here,

$$\chi_i(\mathbf{a}, \theta_i) = 1_{\{b_i + g_i(\mathbf{a}) = N\}} = \begin{cases} 1 & \text{if } b_i + g_i(\mathbf{a}) = N \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

where $\chi_i(\boldsymbol{\theta})$ is 1 if and only if agent i obtains all N coded chunks to be able to decode a block, $g_i(\mathbf{a})$ is the number of packets agent i can get during a frame under the allocation $\mathbf{a}$ (where we suppress the dependence of $\mathbf{a}$ on $\boldsymbol{\theta}$). If the agent leaves and gets replaced by a new one, then, as mentioned before, the new agent gets a deficit value chosen independently using distribution Ψ. Let $\Xi_i[k]$ be 1 if the agent i does not leave at frame k, and 0 otherwise. Let $\bar{\Xi} = 1 - \Xi$. Also, let $\tilde{d}_i[k]$ be chosen *i.i.d.* with the regeneration distribution Ψ. Then given $\theta_i[k] = (b_i[k], d_i[k-1])$ and allocation $\mathbf{a}[k]$, we have $\theta_i[k+1] = (b_i[k+1], d_i[k])$ where $b_i[k+1]$ is chosen *i.i.d.* with distribution ζ and

$$\begin{aligned} d_i[k] = {} & \bar{\Xi}_i[k+1]\tilde{d}_i[k] \\ & + \Xi_i[k+1]\big(d_i[k-1] + \eta_i - \chi_i(\mathbf{a}[k], \theta_i[k])\big)^+. \end{aligned} \tag{3.4}$$

For ease of exposition, set $v_i(\mathbf{a}, \boldsymbol{\theta}_i) = c\Big(\big(d_i + \eta_i - \chi_i(\mathbf{a}, \theta_i)\big)^+\Big)$ for all $i = 1, 2, \ldots, M$ if there is no regeneration and $v_i(\mathbf{a}, \boldsymbol{\theta}_i) = c(\tilde{d}_i)$ otherwise. Using these we can rewrite the system objective as

$$W(\boldsymbol{\theta}[k]) = \min_{\{\mathbf{a}[l]\}_{l=k}^{\infty}} E\left\{\sum_{l=k}^{\infty} \delta^{l-k} \sum_{i=1}^{M} v_i(\mathbf{a}[l], \theta_i[l])\right\}. \tag{3.5}$$

Following [2], we will consider the optimal Markov policy that chooses the allocation solely based on the current state $\boldsymbol{\theta}[k]$ so that the allocation chosen at time $k = 1, 2, \ldots$ will be one of the allocations that solves the Bellman equation depending on the state at time k, *i.e.*, solving for

$$
\begin{aligned}
W(\boldsymbol{\theta}) = \min_{\mathbf{a}} & \sum_{j=1}^{M} v_i(\mathbf{a}, \theta_i) \\
& + \delta \sum_{j'=0}^{M} \delta^{j'} \bar{\delta}^{M-j'} \sum_{\substack{\boldsymbol{\xi} \in \{0,1\}^M: \\ \sum_{l=1}^{M} \xi_l = j'}} E\left\{ W(\boldsymbol{\Theta}(\boldsymbol{\xi})) | \mathbf{a}, \boldsymbol{\theta} \right\},
\end{aligned} \tag{3.6}
$$

where $\boldsymbol{\Theta}(\boldsymbol{\xi})$ evolves as per the one-step transition function determined by the allocation $\mathbf{a}$ and current state $\boldsymbol{\theta}$ for all agents i with $\xi_i = 1$ and from the regeneration distribution otherwise, and then determining the optimal Markov control by choosing

$$
\begin{aligned}
\mathbf{a}^*(\boldsymbol{\theta}) \in \arg\min_{\mathbf{a}} & \sum_{i=1}^{M} v_i(\mathbf{a}, \theta_i) \\
& + \delta \sum_{j'=0}^{M} \delta^{j'} \bar{\delta}^{M-j'} \sum_{\substack{\boldsymbol{\xi} \in \{0,1\}^M: \\ \sum_{l=1}^{M} \xi_l = j'}} E\left\{ W(\boldsymbol{\Theta}(\boldsymbol{\xi})) | \mathbf{a}, \boldsymbol{\theta} \right\},
\end{aligned} \tag{3.7}
$$

and setting $\mathbf{a}^*[k] = \mathbf{a}^*(\boldsymbol{\theta}[\boldsymbol{k}])$. Using the optimal Markov policy we define the value for all agents other than i as

$$
\begin{aligned}
W_{-i}(\boldsymbol{\theta}[k]) &= E\left\{ \sum_{l=k}^{\infty} \delta^{l-k} \sum_{j \neq i} v_j(\mathbf{a}^*[l], \theta_j[l]) \right\} \\
&= \sum_{j \neq i} v_j(\mathbf{a}^*[k], \theta_j[k]) + \delta E\{W_{-i}(\boldsymbol{\theta}[k+1]) | \mathbf{a}^*[k], \boldsymbol{\theta}[k]\}.
\end{aligned} \tag{3.8}
$$

Similarly we can also define the value of agent i using the optimal Markov control policy as

$$
\begin{aligned}
V_i(\boldsymbol{\theta}[k]) &= E\left\{ \sum_{l=k}^{\infty} \delta^{l-k} v_i(\mathbf{a}^*[l], \theta_i[l]) \right\} \\
&= v_i(\mathbf{a}^*[k], \theta_i[k]) + \delta E\left\{ V_i(\boldsymbol{\theta}[k+1] | \mathbf{a}^*[k], \boldsymbol{\theta}[k] \right\}.
\end{aligned} \tag{3.9}
$$

We also define the objective in a fictitious system where agent i is absent from the system; this is constructed for every agent i with the state following the same dynamics as before: the $\mathbf{e}$s are chosen *i.i.d.* in every frame, and the deficits $\mathbf{d}$s follow the recurrence in (3.2). We will denote the allocations in this system by $\mathbf{a}_{-i}$, the state by $\boldsymbol{\theta}_{-i}$ and the objective by W_{-i}, respectively. The objective is given by

$$\begin{aligned} W_{-i}(\boldsymbol{\theta}_{-i}[k]) &= \min_{\{\mathbf{a}_{-i}[l]\}_{l=k}^{\infty}} E\left\{\sum_{l=k}^{\infty}\delta^{l-k}\sum_{j\neq i} v_j(\mathbf{a}_{-i}[l],\theta_j[l])\right\} \\ &= \sum_{j\neq i} v_j(\mathbf{a}^*_{-i}[k],\theta_j[k]) \\ &\quad + \delta E\left\{W_{-i}(\boldsymbol{\theta}_{-i}[k+1])|\mathbf{a}^*_{-i}[k],\boldsymbol{\theta}_{-i}[k]\right\}, \end{aligned}$$

where $\mathbf{a}^*_{-i}(\boldsymbol{\theta}_{-i})$ is the optimal Markov policy for this system when the state is $\boldsymbol{\theta}_{-i}$, and where conditional on $(\mathbf{a}^*_{-i}[k],\boldsymbol{\theta}_{-i}[k])$, $\boldsymbol{\theta}_{-i}[k+1]$ evolves depending on the set of agents that regenerate in a manner similar to that in (3.6).

The optimal allocation depends on the agents revealing their states truthfully. For incentive compatibility, the optimal allocation is incentivized using transfers ($p^*_i(\boldsymbol{\theta}[k])$ for agent i at time k) given by,

$$\begin{aligned} p^*_i(\boldsymbol{\theta}) = W_{-i}(\boldsymbol{\theta}_{-i}) - W_{-i}(\boldsymbol{\theta}) &= \sum_{j\neq i}^{M}[v_j(\mathbf{a}^*_{-i},\theta_j) - v_j(\mathbf{a}^*,\theta_j)] \\ &+ \delta\sum_{j'=0}^{M-1}\delta^{j'}\bar{\delta}^{M-1-j'}\sum_{\substack{\boldsymbol{\xi}\in\{0,1\}^{M-1}:\\ \sum_l \xi_l = j'}} E\left\{W_{-i}(\boldsymbol{\Theta}_{-i}(\boldsymbol{\xi}))|\mathbf{a}^*_{-i},\boldsymbol{\theta}_{-i}\right\} \\ &- \delta\sum_{j'=0}^{M}\delta^{j'}\bar{\delta}^{M-j'}\sum_{\substack{\boldsymbol{\xi}\in\{0,1\}^{M}:\\ \sum_l \xi_l = j'}} E\left\{W_{-i}(\boldsymbol{\Theta}(\boldsymbol{\xi}))|\mathbf{a}^*,\boldsymbol{\theta}\right\}, \end{aligned} \tag{3.10}$$

where $\boldsymbol{\Theta}_{-i}(\boldsymbol{\xi})$ is distributed via the one-step transition function given $\mathbf{a}^*_{-i}$ and $\boldsymbol{\theta}_{-i}$, and also the regenerations. It is easily seen using the optimality of the specific Markov policies $\mathbf{a}^*$ and $\mathbf{a}^*_{-i}$ that $p^*_i(\cdot)\geq 0$ and the net payoff of agent i when system is in state $\boldsymbol{\theta}$, *i.e.*, $V_i(\boldsymbol{\theta}) - p^*_i(\boldsymbol{\theta})$, is maximized by truthfully revealing the state irrespective of the strategy of the other agents (Theorem 1, Bergemann&Valimaki [2]). This then implies a sample-path-wise dominant strategy incentive compatibility.

As we see from (3.10), we need to keep track of all the information over all M agents in the cluster in order to determine the transfer function; in particular, $M+1$ value iterations will need to be carried out to determine the transfers for each agent. If there are many clusters, with agents moving between clusters, keeping track of information and determining transfers becomes computationally prohibitive. In order to solve this problem, we consider the mean-field approximation for the many cluster mobile system case in which we only need to keep track of one agent and all others are drawn from a given distribution function.

3.2 Mobile System Model

We develop the mean-field model by considering a large number of clusters and allowing mobility of agents between clusters in addition to possessing geometrically distributed lifetimes. In particular, in every frame randomly permute all the agents and then assign them to clusters such that there are exactly M agents in each cluster, all of which have the same delivery ratio η. Before proceeding, we first discuss the informational and computational savings that the mean-field model yields. For the system with mobility when the mean-field model does not apply, each agent will need to not only be cognizant of the values and actions of all agents, but also track their mobility patterns. Additionally, the mean-field model does not need to account for regenerations when determining the value functions and the optimal strategy as it used to determine the mean-field distribution.

There is, however, an important nuance that the mean-field analysis introduces: when there are a large number of clusters, each cluster sees a different group of agents in every frame with their states drawn from the mean-field distribution, but though each agent interacts with a new set of agents in every frame, it's own state is updated based on the allocations made to it, so that the viewpoints of the two entities are different and need to be reconciled while providing any incentives.

As before, we are given the system state $\boldsymbol{\theta}[k] = (\mathbf{b}[k], \mathbf{d}[k-1])$ at frame k, where we have collected together the state variables of all the agents in the system. Our mechanism then aims to achieve

$$W(\boldsymbol{\theta}[k]) = \min_{\{\mathbf{a}[l]\}_{l=k}^{\infty}} E\left\{\sum_{j=1}^{J}\sum_{l=k}^{\infty}\delta^{l-k}\sum_{i\in s_j[l]} v_i(\mathbf{a}_{s_j}[l], \theta_i[l])\right\}, \tag{3.11}$$

where $j = 1, 2, \cdots, J$ is the number of clusters in the system, $s_j[k]$ is the set of agents in cluster j at frame k and $\mathbf{a}_{s_j}$ is the allocation in cluster j. For agent i set $j_i[k]$ to be the cluster he belongs in during frame k, i.e., $i \in s_{j_i[k]}[k]$. Given the allocation in each cluster, if agent i does not regenerate, then his deficit gets updated as

$$d_i[k] = (d_i[k-1] + \eta - \chi_i(\mathbf{a}_{j_i[k]}[k], \theta_i[k]))^+. \tag{3.12}$$

As there are a large number of clusters, in the next frame there is a completely different set of agents that appear at any given cluster. The states of these agents will be drawn from the mean field distribution. Hence, from the perspective of some cluster l, the state of the agents in that cluster $\hat{\boldsymbol{\Theta}}_l$ will be drawn according to the (cumulative) distributions $[\otimes\rho^M, \otimes\zeta^M]$, with ρ pertaining to the deficit, and ζ pertaining to the B2D transmissions received by that agent. Note that the support of ρ is $\mathbb{R}^+$, while the support of ζ is $\mathbb{Z}^+$, and $\otimes$ indicates the $i.i.d$ nature of the agent states. Whereas from the perspective a particular agent i, the states of all the other agents in that cluster will be drawn according to $\hat{\boldsymbol{\Theta}}_{-i} \sim [\otimes\rho^{M-1}, \otimes\zeta^{M-1}]$. These facts will simplify the allocation problem in each cluster and also allow us to analyze the mean-field by tracking a particular agent.

First, we consider the allocation problem as seen by the clusters. Pick any finite number of clusters. In the mean-field limit, the agents from frame to frame will be different in the entire set, therefore the allocation decision in each cluster can be made in an distributed manner, independent of the other clusters; this is one of the chaos hypotheses of the mean-field model. This then implies that the objective in (3.11) is achieved by individual optimization in each cluster, *i.e.*,

$$W(\boldsymbol{\theta}[k]) = \sum_{j=1}^{J} W_j(\boldsymbol{\theta}_{s_j}[k]), \tag{3.13}$$

where $\boldsymbol{\theta}_{s_j}[k]$ is the state of the agents in cluster j at time k and

$$W_j(\boldsymbol{\theta}_{s_j}[k]) = \min_{\{\mathbf{a}_{s_j}[l]\}_{l=k}^{\infty}} \sum_{l=k}^{\infty} \delta^{l-k} \sum_{i \in s_j[l]} v_i(\mathbf{a}_{s_j}[l], \theta_i[l]). \tag{3.14}$$

Furthermore, the value function in the mean-field is determined by the first solving the following Bellman equation

$$\hat{W}(\hat{\boldsymbol{\theta}}) = \min_{\mathbf{a}} \sum_{i=1}^{M} v_i(\mathbf{a}, \theta_i) + \delta E\left\{\hat{W}(\hat{\boldsymbol{\Theta}})\right\}, \tag{3.15}$$

where $\hat{\boldsymbol{\theta}}$ is the M-dimensional state vector (with elements θ_i) and the future state vector $\hat{\boldsymbol{\Theta}}$ is chosen according to $[\otimes\rho^M, \otimes\zeta^M]$, and thereafter setting $W_j(\boldsymbol{\theta}_{s_j}[k]) = \hat{W}(\boldsymbol{\theta}_{s_j}[k])$ for every $j = 1, 2, \ldots, J$. This observation then considerably simplifies the allocation in each cluster to be the greedy optimal, *i.e.*, determine (multi)function

$$\mathbf{a}^*(\hat{\boldsymbol{\theta}}) = \arg\min_{\mathbf{a}} \sum_{i=1}^{M} v_i(\mathbf{a}, \theta_i), \tag{3.16}$$

and for $j = 1, 2, \ldots, J$ we set $\mathbf{a}^*_{s_j} = \mathbf{a}^*(\hat{\boldsymbol{\theta}}_j)$.

Next, we consider the system from the viewpoint of a typical agent i; without loss of generality, let $i = 1$. Any allocation results in the deficit of this agent changing according to (3.12) and the future B2D packets drawn according to ζ, whereas the state of every other agent that agent 1 interacts with in the future gets chosen according to the mean field distribution. Then the value function from the perspective of agent 1 is determined using

$$\begin{aligned}&\tilde{W}(1,(\theta_1,\hat{\boldsymbol{\theta}}_{-1})) = \\ &\min_{\mathbf{a}} \sum_{i'=1}^{M} v_{i'}(\mathbf{a},\theta_{i'}) + \delta E\left\{\tilde{W}(1,(\Theta_1,\hat{\boldsymbol{\Theta}}_{-1}))|\mathbf{a},\theta_1\right\},\end{aligned} \tag{3.17}$$

where $\hat{\boldsymbol{\theta}}_{-1}$ represents the states of all the agents in cluster except 1, $\hat{\boldsymbol{\Theta}}_{-1} \sim [\otimes\rho^{M-1}, \otimes\zeta^{M-1}]$, and for Θ_1, the deficit term is determined via (3.2) while the B2D term follows ζ. Using this, one can also determine the allocation that agent i expects his cluster to perform, namely, define the (multi)function

$$\begin{aligned}&\tilde{\mathbf{a}}(\theta_1,\hat{\boldsymbol{\theta}}_{-1}) = \\ &\arg\min_{\mathbf{a}} \sum_{i'=1}^{M} v_{i'}(\mathbf{a},\theta_{i'}) + \delta E\left\{\tilde{W}(1,(\Theta_1,\hat{\boldsymbol{\Theta}}_{-1}))|\mathbf{a},\theta_1\right\},\end{aligned} \tag{3.18}$$

and for agent i, set the allocation to be $\tilde{\mathbf{a}}(\theta_i,\hat{\boldsymbol{\theta}}_{-i})$.

Using the two allocations $\mathbf{a}^*$ and $\tilde{\mathbf{a}}$ we can write down the value of agent i from the system optimal allocation and the value of agent i in the allocation that the agent thinks that the system will be performing. For a given allocation function $\mathbf{a}(\cdot)$ (for the state of all agents in the cluster where i resides at present), we determine the solution to the following recursion

$$V(\mathbf{a}(\hat{\boldsymbol{\theta}}), \theta_1) = v_1(\mathbf{a}, \theta_1) + \delta E \left\{ V(\mathbf{a}(\Theta_1, \hat{\boldsymbol{\Theta}}_{-1}), \Theta_1) \right\}, \tag{3.19}$$

where Θ_1 follows the dynamics where the deficit is updated using (3.2) and the B2D term is generated independently, and $\hat{\boldsymbol{\Theta}}_{-1}$ is chosen using the mean-field distribution. Then by the optimal allocation, agent i gets $V(\mathbf{a}^*(\theta_i, \hat{\boldsymbol{\theta}}_{-i}), \theta_i)$ whereas from the perception of agent i he should get $V(\tilde{\mathbf{a}}(\theta_i, \hat{\boldsymbol{\theta}}_{-i}), \theta_i)$.

We will use the different value functions to define the transfer for agent i depending on the reported state variables in the cluster. Determine the function

$$\begin{aligned} p^*(\theta_1, \hat{\boldsymbol{\theta}}_{-1}) = {} & V(\mathbf{a}^*(\hat{\boldsymbol{\theta}}), \theta_1) - V(\tilde{\mathbf{a}}(\hat{\boldsymbol{\theta}}), \theta_1) \\ & + C(\hat{\boldsymbol{\theta}}_{-1}) - \sum_{i' \neq 1} V(\tilde{\mathbf{a}}(\hat{\boldsymbol{\theta}}), \theta_{i'}), \end{aligned} \tag{3.20}$$

where $C(\hat{\boldsymbol{\theta}}_{-1})$ is a function only of $\hat{\boldsymbol{\theta}}_{-1}$ and following the Groves pivot mechanism can be chosen to be the value of the greedy optimal allocation assuming that agent 1 is not present[1]. Then we set the transfer (the price charged) for agent i to be $p^*(\theta_i, \hat{\boldsymbol{\theta}}_{-i})$. The Groves pivot mechanism ensures that the last two terms together in (3.20) are non-negative; however, it need not hold that the first two terms together are non-negative. Additionally, since the optimal allocation from the viewpoint of agent i can differ from agent to agent and collectively need not even be a feasible allocation, even the sum of the first two terms over all the agents need not be non-negative.

[1]It can be argued similar to the development above that this is, indeed, the optimal allocation if agent i were absent.

3.3 Allocation Scheme

The basic building block of our mechanism is the per-frame optimal allocations that solve (3.11). We will now spell out the allocation in greater detail. We observe that, in case of more than one cluster, the allocation problem can be solved independently for each cluster.

The objective in this cluster is

$$\min_{\mathbf{a}} \sum_{i=1}^{M} v_i(\mathbf{a}, \theta_i) \tag{3.21}$$

$$= \min_{\mathbf{a}} \sum_{i=1}^{M} c((d_i[k-1] + \eta - \chi_i(\mathbf{a}[k], \theta_i[k]))^+) \tag{3.22}$$

Given the B2D arrivals $(b_1[k], ..., b_M[k])$, we partition the set of devices $\{1, ..., M\}$ into sets $\mathcal{S}$ and $\mathcal{S}^c = \{1, ..., M\} \backslash \mathcal{S}$, based on whether $b_i[k] + T - N \geq 0$ or not. Those agents that satisfy this condition can potentially receive enough chunks during the D2D phase that they can decode the block, whereas the others cannot. Hence, all members of $\mathcal{S}^c$ can potentially transmit their chunks in the allocation solving (3.22). Let $T_1 = \min\{\sum_{i \in \mathcal{S}^c} b_i[k],\ \mathrm{T}\}$. So we can devote the first T_1 slots of the current frame to transmissions from the devices in $\mathcal{S}^c$.

Let the number of transmissions made by agent i in allocation $\mathbf{a}$ be denoted by $x_i[k]$. We can write down the constraints that any feasible allocation $\mathbf{a}$ must satisfy as

$$\begin{aligned} &0 \leq x_i[k] \leq b_i[k] \qquad \forall\ i \in \mathcal{S} \\ &\textstyle\sum_{i \in \mathcal{S}} x_i[k] = T - T_1 \end{aligned} \tag{3.23}$$

Observe that each agent can transmit $b_i[k] + T - N$ chunks without affecting the above constraints (*i.e.,* it does not change its chances of being able to decode the block, as there is enough time left for it to receive chunks that it requires). We call these as “extra” chunks. Suppose that all extra chunks have been transmitted by time $T_2 < T$, and no device has yet reached full rank. At this point, all agents in the system need the same number of chunks, and any agent that transmits a chunk will not be able to receive enough chunks to decode the block. In other words, agents now have to “sacrifice” themselves one at a time, and transmit all their chunks. The question is, what is the order in which such sacrifices should take place?

Compare two agents i and j, with deficits $d_i > d_j$. Also, let $\chi \in \{0, 1\}$. Now, for either value of χ

$$d_i - (d_i - \chi)^+ \geq d_j - (d_j - \chi)^+.$$

Hence, since $c(.)$ is strictly convex and monotone increasing,

$$\int_{(d_i-\chi)^+}^{d_i} c'(z)dz \geq \int_{(d_j-\chi)^+}^{d_j} c'(z)dz \geq 0 \tag{3.24}$$

$$\Rightarrow c(d_i) - c((d_i - \chi)^+) \geq c(d_j) - c((d_j - \chi)^+) \geq 0. \tag{3.25}$$

Now, consider the following problem with $\chi_i, \chi_j \in \{0, 1\}$ under the constraint $\chi_i + \chi_j = 1$:

$$\min_{\chi_i,\chi_j} c(d_i - \chi_i) + c(d_j - \chi_j). \tag{3.26}$$

$$\Leftrightarrow \quad \max_{\chi_i,\chi_j} c(d_i) - c(d_i - \chi_i) + c(d_j) - c(d_j - \chi_j). \tag{3.27}$$

Then, from the above discussion, the solution is to set $\chi_i = 1$ and $\chi_j = 0$. Thus, comparing (3.26) and (3.22), the final stage of the allocation should be for agents to sacrifice themselves according to a min-deficit-first type policy. We find that in order to achieve long term stability of the deficit queues in a fully cooperative single cluster, the allocation rule must be as mentioned in Algorithm 2.

4. APPLICATION DESIGN

In this chapter, we describe the applications developed to implement the system described in Chapter 2. It consists of three applications as follows:

- Server Application: This application encodes the data and sends it to the devices over the Internet.
- Monitor Application: This application is mainly used to collect statistics and logs.
- Android Streamer Application: This application receives data from the server, shares it with other devices and plays the data if it could successfully decode it.

We begin the chapter by describing the process of rooting Android such that multiple interfaces work on it simultaneously followed by a description of the structure of the packets used by the applications.

4.1 Rooting Android

Android is an open source software stack whose development is led by Google. This enables us to access the source code of Android and modify it to suit our requirements. Our algorithm and the resulting application must be designed such that it can transfer as well as receive data simultaneously through the two orthogonal interfaces, viz. 3G/4G cellular interface and the WiFi interface. The normal or stock ROMs of Android available in the market today switch off their cellular interface whenever a known WiFi network is available in order to reduce data consumption costs over cellular network.

When there is a switch from one interface to another on a normal phone, the IPTables of one interface are deleted and replaced by the routing table of the other. Whenever the second interface is up on our modified version of Android, the IPTables of the first interface are retained and the IPTables of the new interface are added to the existing table. As a result, the phone has two default gateways.

The IPTable of a phone on stock Android looks as follows on a WiFi network and cellular network.

WiFi only:

```
default via 192.168.0.1 dev wlan0

192.168.0.0/24 dev wlan0 scope link

192.168.0.0/24 dev wlan0 proto kernel scope link src 192.168.0.125

192.168.0.1 dev wlan0 scope link
```

3G only:

```
default via 25.124.0.137 dev rmnet_usb0

25.124.0.136/30 dev rmnet_usb0 proto kernel scope link src 25.124.0.138

25.124.0.137 dev rmnet_usb0 scope link

208.54.70.50 via 25.124.0.137 dev rmnet_usb0
```

The IPTable of a phone running our modified version of Android on the same networks with both interfaces running is as follows

```
default via 192.168.0.1 dev wlan0

default via 25.124.0.137 dev rmnet_usb0

25.124.0.136/30 dev rmnet_usb0 proto kernel scope link src 25.124.0.138

25.124.0.137 dev rmnet_usb0 scope link

192.168.0.0/24 dev wlan0 scope link

192.168.0.0/24 dev wlan0 proto kernel scope link src 192.168.0.125

192.168.0.1 dev wlan0 scope link
```

```
208.54.70.50 via 25.124.0.137 dev rmnet_usb0
```

It can be seen that the IPTable of our rooted phone is a combination of the IPTables of the WiFi and 3G interfaces when both are being used. It is to be noted that the IPTable even has two *default* entries.

The advantage of using a phone such as Nexus 4 is that it runs on stock Android devoid of any manufacturer or carrier add-ons. We download the source code of Android from Google's repository [13]. The code we are concerned with in order for both the interfaces to work simultaneously can be found at "/frameworks/base/services/-java/com/android/server/ConnectivityService.java". We modify the portion of the code which specifies that an interface currently being used be torn down if another preferred interface becomes available. This modified code now retains information about both the interfaces. The code is then compiled with Nexus 4 specific options and packaged on a desktop running Ubuntu 13.10. We unlock the boot loader of the phone, reboot it into recovery mode and flash the newly compiled custom ROM.

4.2 Structure of the Encoded Packet

In this section we describe the three different types of packets present in the network, the server packet, the D2D packets and the monitor packets.

4.2.1 Server Packet

The structure of the packets sent by the server to the phones using their cellular interface is shown in Figure 4.1. These packets from the server have a header of size 12 bytes. Each of the fields *viz.*, static identifier, track number and deficit have size of 4 bytes. The static identifier is present in every packet in the system. It is used to identify if a packet is part of our streaming application. We use "12345" as the static identifier in our system. The block number is used to tell the application which block the data belongs to. This data is also used by the application to synchronize itself with the server and to switch blocks. The deficit field in the header is a dummy field and is included only for consistency with other packets in the network. The last field in the server packet is the part that contains the coded data. The size of this field can vary according to the frame duration and bit rate but usually is set to 1400 bytes in our system.

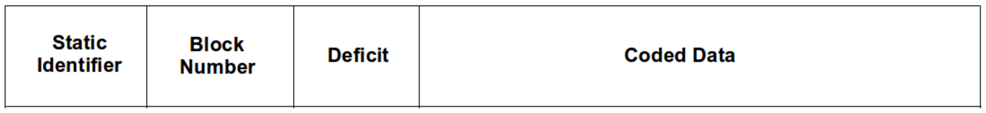

Figure 4.1: Structure of packet received from the server.

4.2.2 D2D Packet

The structure of D2D packets in our system is shown in Figure 4.2. The structure is very similar to the server packets with the only exception of a *Phone Identifier* field before the deficit field. We emulate churn in our system by making a phone reset every few frames with a predetermined probability. A phone reset essentially means that we reset the deficit of the phone to a random number between 0 and 5. Since the IP address of a phone doesn't change when the phone resets, in order to differentiate a phone before and after a reset we use a four letter alpha numeric character created by a random generator. We call this four letter alpha-numeric character the phone identifier. Every time a phone resets, it generates a new phone identifier. This combination of IP address and the phone identifier gives an unique identity to phone in between resets.

Static Identifier	Block Number	Phone Identifier	Deficit	Coded Data

Figure 4.2: Structure of P2P packets.

4.2.3 Monitor Packet

The third kind of packet in our network are the packets that are sent by the phones to the monitor application. The structure of a monitor packet is shown in Figure 4.3. These packets are used to collect logs and statistics from the system. Each of these monitor packets have 6 fields of size 4 bytes. The first field is the same as the two other types of packets. However we use a different identifier to differentiate it from other packets in the system. The second field contains the block number whose data is being carried by the packet. It is the same as the block number of the D2D packets and one behind the server packets in the system. The packets received field tells the monitor how many packets were sent to the phone by the server for this block. The phone identifier contains the four letter alpha numeric identifier for the phone. The deficit field contains the current deficit of the phone. The last field called the D2D packets transmitted tells the monitor how many packets were transmitted by the phone in the previous frame.

Static Identifier	Block Number	Packets Received	Phone Identifier	Deficit	P2P Packets Transmitted

Figure 4.3: Structure of monitor packets.

4.3 Server Application

The server application is responsible for encoding data and sending it to the requesting devices in our network. The first step in this process is encoding the data. As mentioned in the previous chapters, we use random linear coding to encode the data. In order to encode the data in our server application we use the NCutils library [9] written in Java that contains a set of functions that can be used to implement network coding techniques in applications. This Java library can be used to implement support of random network coding in a Java application. As soon as the server is started in our application, a file is read and encoded into multiple blocks and transmitted in order to simulate a real time stream.

The server application runs three processes, a *receive request thread*, a *timer thread* and a *transmission thread*.

- The *receive request thread* is used to initiate a connection with a client. This thread starts listening for new connection requests as soon as the application is started. The client first sends a connection request to the server on a known socket. The server application then saves the identity of the client in its database and initiates a connection.

- The *timer thread* is used to set the duration of a frame. The timer thread wakes up at the beginning of each frame and spawns a transmission thread. It then goes to sleep for the rest of the duration of the frame.

- The last and the most important thread in the server is the *transmission thread*. This thread inserts encoded data into a packet and appends appropriate headers before transmission to a client. Note that the data is already available in encoded form as this task is done as soon as the server is started. The transmission thread reads the database of the clients present in the system and transmits a random number of these packets to each of them. The random number is chosen within an upper and a lower limit which can be set when the server application is started.

4.4 Monitor Application

The monitor application helps us to collect useful data and statistics about the system. Every phone in our network tells the monitor about the number of packets it received from the server, its current deficit and the number of P2P packets it transmitted in the previous frame at the beginning of every frame. In addition, the monitor also tries to intercept all the D2D packets that are transmitted by the phones in order to keep track of which phones transmitted when and in what order. The monitor also tries to see if its possible to decode the data for the frame using the packets it has received.

4.5 Android Streamer Application

The Android streaming application runs on our rooted phone running a modified build of Android. This application is responsible for connecting with the server, sending and receiving data as well as playing the decoded data. It utilizes two sockets to exchange data, one to receive data from the server over the Internet and the other to send as well as receive data locally with others connected to the same access point.The application consists of three threads running in parallel, namely, the server thread, the D2D send and receiver threads. Also the media player service is called by our application and runs in the background.

Recall from Figure 1.2 that during Frame *k-2*, B2D for block *k*, D2D for block *k-1* and decoding and playout for block *k-2* occur simultaneously. Since we have no explicit time synchronization, the different threads need to determine the current frame number from the block number of the B2D transmissions. The server thread first connects to the server using its IP address upon which the server starts sending encoded data from the beginning of the next block. Say this is block *k*. As soon as the application receives a B2D packet with block number *k*, it assumes that the B2D transmission time for block *k-1* has ended and it advances to the next frame. At this instant, the D2D threads take over the block *k-1* (whose B2D transmission just ended in the server thread). This process is shown in Figure 4.4. The packets received from the server are stripped off their headers and the encoded data is added to a decoder unique to block *k-1*. Note that block *k-1* cannot be decoded yet since the decoding thread has not received enough degrees of freedom to do so. The raw encoded data is also saved in another data buffer to be used for transmission by the D2D thread.

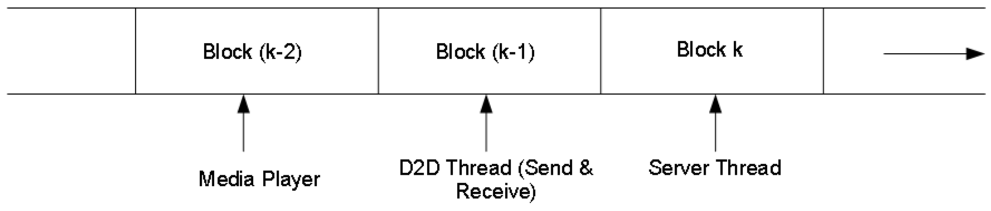

Figure 4.4: Flow of data to and from the B2D/D2D threads and the data buffers at the frame boundaries.

The D2D send thread picks up encoded data of block *k-1* from the data buffer, appends its own headers and transmits it using a different socket. The D2D receive thread meanwhile keeps listening for any transmissions from other devices. When a device should transmit is decided by following the scheduling policy described in Algorithm 3 which is based on Algorithm 2. As soon as a packet is received from another user in the network, the send thread is interrupted and the timers are reset.

Algorithm 3 Implementation of D2D scheduling and back-off

```
 At the beginning of each frame $k-1$
Partition the devices into sets $\mathcal{S} = \{i \in \{1, ..., M\} : \textit{Rank} \geq \textit{Subspace Size / 2}\}$ and $\mathcal{S}^c$.
While devices have packets received via B2D that have not yet been transmitted,
1) All the devices in $\mathcal{S}^c$ back-off between 1ms and 5ms before transmitting a packet.
2) All the devices in $\mathcal{S}$ back-off between 1ms and 15ms before transmitting their extra packets.
While there exist time, let devices in $\mathcal{S}$ back-off for a duration between 5ms and 15ms proportional to its own deficit and the maximum deficit observed before transmitting their remaining chunks.
Every time a D2D packet is received, the back-off timers are reset.
```

As D2D packets are received, they are stripped off their headers and the encoded data is added to the decoder for that block. The rank of the decoder keeps increasing as more and more packets are received and the data is written to a file. The decoded data of individual blocks are not written to separate files. On the contrary, the data from several blocks are grouped together to be written to a file. This is done because when the media player in Android finishes playing one such file and switches to the next, a lag can be noticed. This might produce a not-so-pleasant experience for the listener. If the media player switches files at the end of every block, it might degrade the quality of the experience. Hence we aggregate several blocks together before playing them. This aggregation does indeed induce a few seconds of delay which however, is consistent with current real time streaming standards.

4.6 Application for Adhoc Networks

An adhoc network makes it possible for devices to connect and communicate with each other without the need of an access point. This makes an adhoc network a device to device network in the true sense of the term. Hence it is our goal to deploy aur application on an adhoc network. An adhoc network needs a DHCP server to allocate IP addresses when a device connects to the network for the first time. However, once the devices have been alloted IP addresses, the DHCP server is no more required and the devices can communicate with each other directly. In case, we expect churn to be present in the network, we must make the DHCP server available at all times. The Nexus 4 phone comes with a Qualcomm Atheros WCN3660 chipset which makes it incapable of supporting the wireless drivers needed to establish connection to an adhoc network. We use Sprint HTC EVO phones that come with the Broadcom BCM4329 chipset to test our setup on an adhoc network. Since we do not have cellular service on the HTC phones, we simulate the B2D transmissions in our system. The HTC EVOs are rooted, and a custom recovery is installed on the phone. The HTC EVOs come with Android Gingerbread (v 2.3) and hence we download and install Cyanogenmod ROM built for this phone. CyanogenMod is an open source operating system for smartphones and tablet computers, based on the Android mobile platform. It is developed as free and open source software based on the official releases of Android by Google, with added original and third-party code [6]. The CyanogenMod ROM has the drivers prebuilt for our HTC phones to connect to an adhoc network. We create an adhoc network on a desktop and connect multiple phones to the network. We then switch off the network on the desktop and verify that the phones are still connected to the adhoc network. We run a modification of our application on this setup since the phones do not have a data connection. We split and encode the file to be played and store it on the phone itself. The phones are then initialized from these encoded files at the beginning of each frame. In this way, we actually emulate what happens on a phone with an actual data connection. The second step of D2D transmissions is similar to what has been mentioned in the previous section. The application was seen to show similar performance in an adhoc network.

5. EXPERIMENTAL RESULTS

In this chapter, we present experimental results that illustrate the performance of our Android application.

5.1 Properties of B2D and D2D Channel

We first test the capabilities of the B2D and D2D channels of our phones. In Table 5.1 we broadcast 150 UDP packets at regular intervals of 45ms, 35ms, 25ms, 15ms and 0ms each. It can be seen from the results that the delivery ratios are good even when the pauses between the transmissions are as low as 15ms. This indicates that the phones are capable of handling quite high transmission rates in our experimental setup.

Table 5.1: Delivery ratios for UDP broadcast on the D2D channel.

Blocks Sent	Sleep 45ms	Sleep 35ms	Sleep 25ms	Sleep 15ms	Sleep 0ms
150	150	145	145	145	25
150	144	145	145	142	24
150	146	144	145	143	24
150	145	145	145	145	25
150	150	145	145	145	51
150	150	145	141	143	27
150	150	144	138	144	26
150	149	133	145	145	24
150	145	135	127	145	51
150	148	130	145	145	58
150	150	145	145	141	24

Table 5.2 shows the delivery ratios for transmissions in B2D channel with different sleep times in between transmissions. It can be seen that the transmissions are quite

reliable because of the high delivery ratios.

Table 5.2: Delivery ratios for the B2D channel.

Blocks sent	Sleep 75ms	Sleep 65ms	Sleep 55ms	Sleep 45ms	Sleep 35ms
100	99	97	100	100	100
100	94	87	100	98	93
100	88	100	99	100	94
100	99	100	100	100	100
100	100	100	100	97	100
100	100	100	100	97	100
100	100	100	100	97	100
100	100	100	100	72	93
100	100	100	100	97	100
100	100	89	100	97	94

5.2 Stabilizing Nature of D2D Algorithm

We conduct trials to study the performance of our D2D algorithm, emulating the B2D part. The objective is to understand its behavior under a perfect B2D channel. We conduct experiments involving 3 to 5 HTC phones, using a standard MP3 audio file as the source, since playout is easily accomplished using the built in player on Android. The B2D channel in this case was emulated meaning that the phones had coded packets available to them at the beginning of a frame from a multimedia storage device on the phone itself implying a loss less B2D channel. An example trajectory of the (smoothed) deficit queue for a run with 3 phones, with each phone being initialized with 4 chunks in each frame, and desired delivery ratio 0.95 is shown in Figure 5.1. The deficit queue exhibits periodic decrease, and clearly does not increase to infinity, showing stabilization.

We create a cluster of 5 HTC phones running our application and conduct mul-

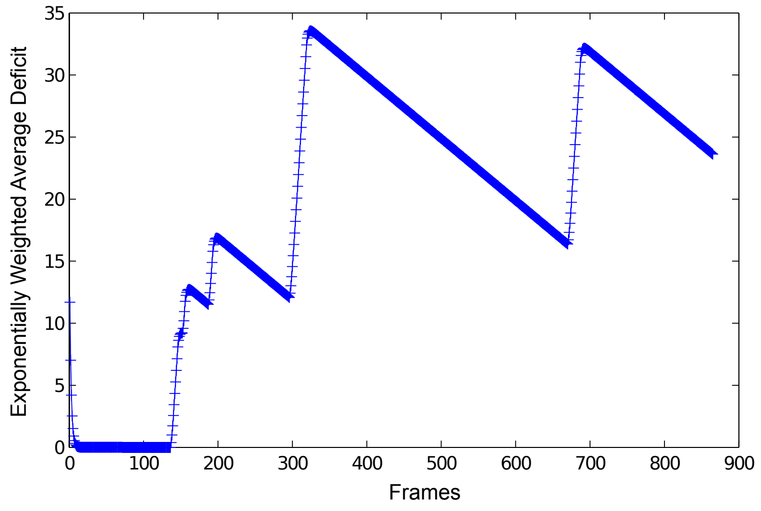

Figure 5.1: Sample trajectory of deficit. It is clear that the deficit queue is stable

tiple runs in order to evaluate the effect on QOS with increasing number of B2D initialization (emulated). The results of the experiments is shown in Table 5.3. We vary the number of B2D initialization from 2 to 5. In a cluster of 5 phones this means that the total number of coded packets available in the system varies from the bare minimum of 10 to 25 (Note that a phone needs 10 coded packets every frame in order to decode the frames). Due to the way with which we initialize the phones, these packets can be assumed to be unique in general. The results show that when the initialization is a bare minimum of 2 coded packets per frame or 10 coded packets in the system, the average QOS observed is quite low. However as the number of initialization per phone is increased, QOS rises quite sharply.

5.3 Deficit Distribution using Multiple Interfaces

We are now ready to conduct detailed experiments using both wireless interfaces simultaneously. The clients send connection requests to the server which has coded packets ready to be transmitted. The server selects a random number of packets to

Table 5.3: Delivery ratios achieved with different initialization and simulated B2D.

B2D	Ph 1	Ph 2	Ph 3	Ph 4	Ph 5	Avg
2	56.38	38.44	56.37	41.7	43.1	47.198
3	86.9	70.85	81.8	75.3	83.6	79.69
4	85.75	94.6	94.3	98.31	98.45	94.282
5	98.34	98.03	97.3	99.48	100	98.63

transmit to each phone from a range that is predefined by the administrator. We conduct the experiment varying this predefined range and note the quality of service received by each of the phones. The results in Table 5.4 show the expected trend of increasing quality of service as the phones are initialized with increasing number of packets in the B2D phase.

Table 5.4: Delivery ratios achieved with different initialization with 3G.

B2D	AT&T 1	AT&T 2	T-Mob 1	T-Mob 2	Avg
$2-4$	82.41	81.48	78.77	81.17	80.96
$3-5$	86.10	88.71	87.07	90.93	88.20
$4-6$	89.24	92.80	90.29	90.92	90.81
$5-7$	96.41	92.05	91.27	97.77	94.37

In our next experiment, we record the deficit values of each phone at the end of each frame. We then collect these deficit values into bins of a specified size encompassing the range of values of deficit observed. We count the frequency of the number of deficits in each bin. The normalized plot of the frequencies of each bin is shown in Figures 5.2 and 5.3. The figures show the distribution for a cluster of 4 phones initialized with 4 coded packets out of a total of 10 packets needed to decode a frame. We collect data for a total of 500 frames. Figure 5.2 shows the distribution for a bin size of 0.1 while Figure 5.3 shows the distribution for a bin size of 0.2. The application in this case resets randomly to 0 every 50 frames on an average. Hence, both the figures show the bars at 0 to be heavily biased. The figures show a decaying sinusoidal pattern indicating that some of the states are more prevalent than the others because of the type of transitions.

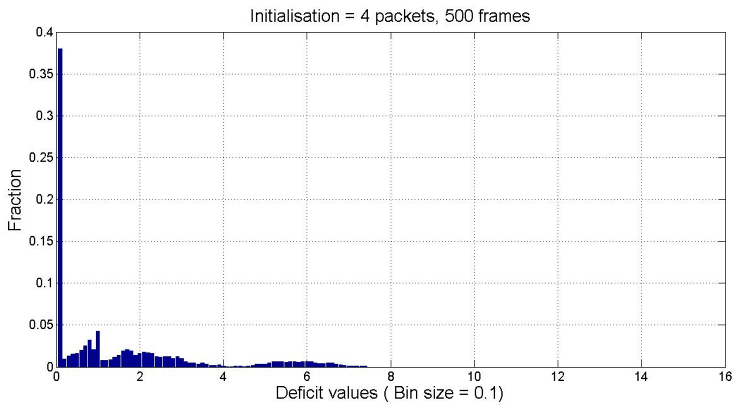

Figure 5.2: Distribution of deficit values for bin size=0.1.

We next change the application to reset to a random deficit between 0 and 5 and repeat the previous experiment with 3 phones.The results are shown in Figure 5.4. The figure shows that because the deficit resets to a random value between 0 and 5 and not always 0, the bias at 0 is reduced by a huge margin. Also we observe in the figure that because there are only 3 phones present in the system, the sinusoidal decaying curve peaks at different points than Figures 5.2 and 5.3 due to different dominant states.

We equip our monitor with a decoder so that it can also decode a frame if it receives sufficient number of coded packets. Since our monitor can listen to every D2D transmission, we count the number of transmissions seen by the monitor every

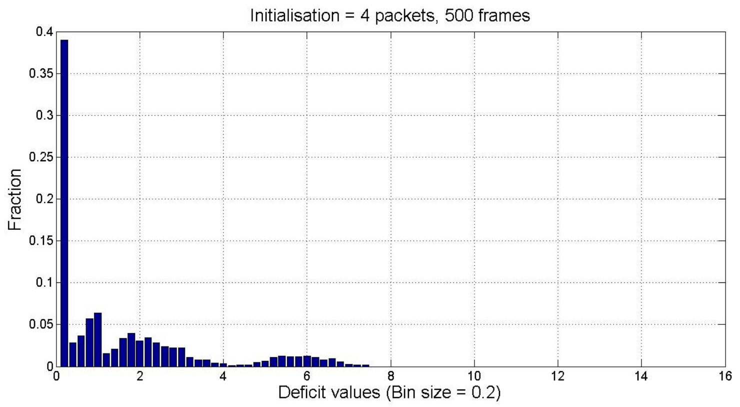

Figure 5.3: Distribution of deficit values for bin size=0.2.

frame as shown in Figure 5.5. We also track the rank of the monitor at the end of each frame as seen in Figure 5.5. It follows that the rank of the monitor closely follows the number of transmissions seen by it when the total number of transmissions is less than 10 and caps at 10 when transmissions are more than that. It confirms our assumption that the coded packets sent to each phone by the data server are indeed unique.

It must be noted that the monitor runs on a desktop which has hardware much superior to that in any of the phones. So even when the same transmissions are observed by a phone, the results might vary.

In Figure 5.6 we present the deficit trajectories of a cluster of 3 phones. The phones are initialized randomly with coded packets in the range 4-6. The frames at which the phones reset can be easily spotted out from the sharp rises and falls of deficit in the figure. It can be seen from the graph that the deficit values of the phones behave independently from each other and they generally have a tendency to decrease. It is safe to conclude from the trajectories that the system is necessarily stable.

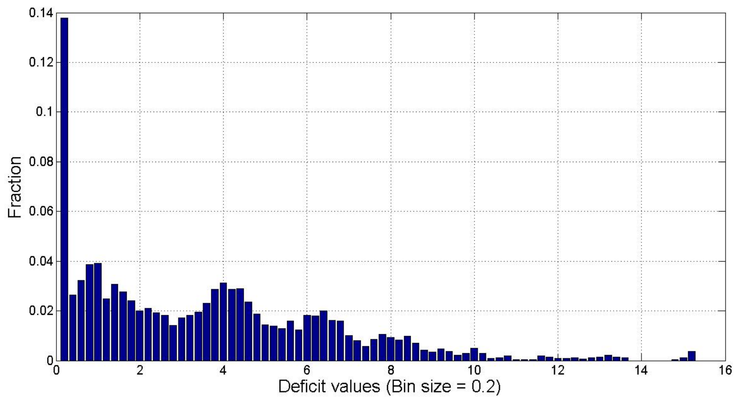

Figure 5.4: Distribution of deficit values for bin size=0.2 with no bias at 0.

5.4 Incentive Mechanism

We next present simulation results for the incentive mechanism discussed in Chapter 3. We compute the system value from the viewpoint of agent 1. Here, we suppose there are $M = 4$ agents in each cluster, and all have $\eta = \delta = 0.95$. Each agent needs to receive $N = 10$ packets to decode the block, and there are $T = 8$ time slots in each frame. We wish to determine the value function from the perspective of the cluster and from the perspective of agent 1, using (3.15), (3.17) and (3.19).

We make the following useful observations on determining the optimal allocations $\mathbf{a}^*$ and $\tilde{\mathbf{a}}$. It is straightforward to find $\mathbf{a}^*$, since it simply follows Algorithm 2. Now, consider $\tilde{\mathbf{a}}$. It is simple to see that it too would follow Phases 1 and 2 of Algorithm 2. Then, from the perspective of agent 1, after the completion of these two phases, there are only two classes of allocations–those in which he transmits and those in which he does not. Now, since all the other agents that agent 1 comes in contact with in the future are drawn from $[\otimes\rho^{M-1}, \otimes\zeta^{M-1}]$, the allocation should follow a greedy minimization with respect to the other agents. Thus, we only need consider two

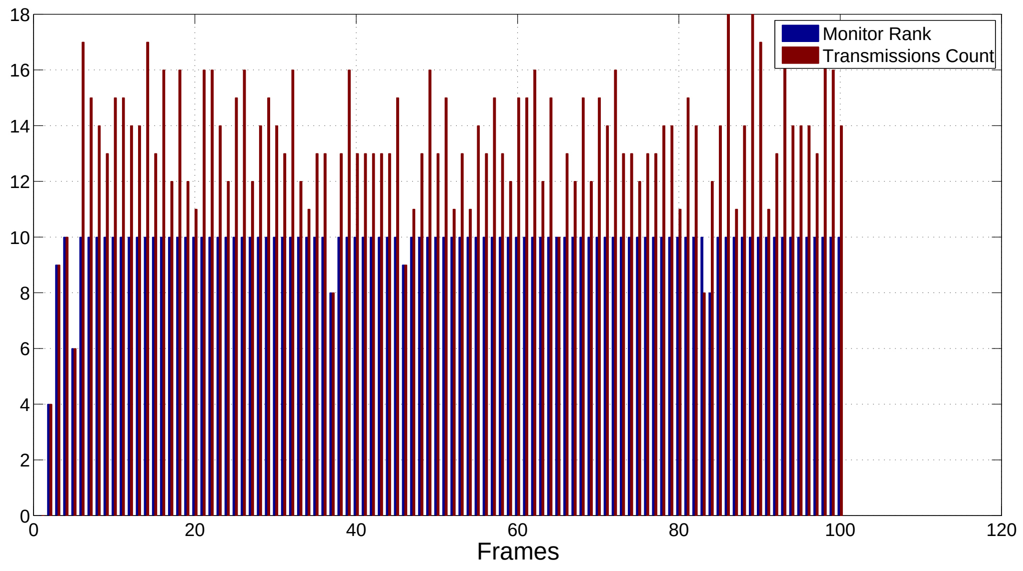

Figure 5.5: Rank achieved at the monitor.

allocations while conducting value iterations: min-deficit-first with agent 1 (identical to Phase 3 of Algorithm 2) and min-deficit-first without agent 1 (just set aside agent 1 in Phase 3 of Algorithm 2).

We first run the system according to Algorithm 3, and use the results to find the empirical deficit distribution, denoted by R. We find that deficit lies in the range $0 - 13$. We set the resolution of R as 0.5, so we have 26 potential values for deficit. For the number of B2D chunks received e, we take values 3, 4 and 5 (uniformly). Therefore, there are totally $26^4 \times 3^4$ states in the system. Using R to represent the MF deficit distribution, we run value iteration and obtain convergence after around 100 rounds; we present an example in Figure 5.7.

The empirical distribution of the average discounted over the lifetime of each device is shown in Figure 5.8. We see that average transfer centers around -1, meaning that the system is subsidized by the content provider. Agents are both

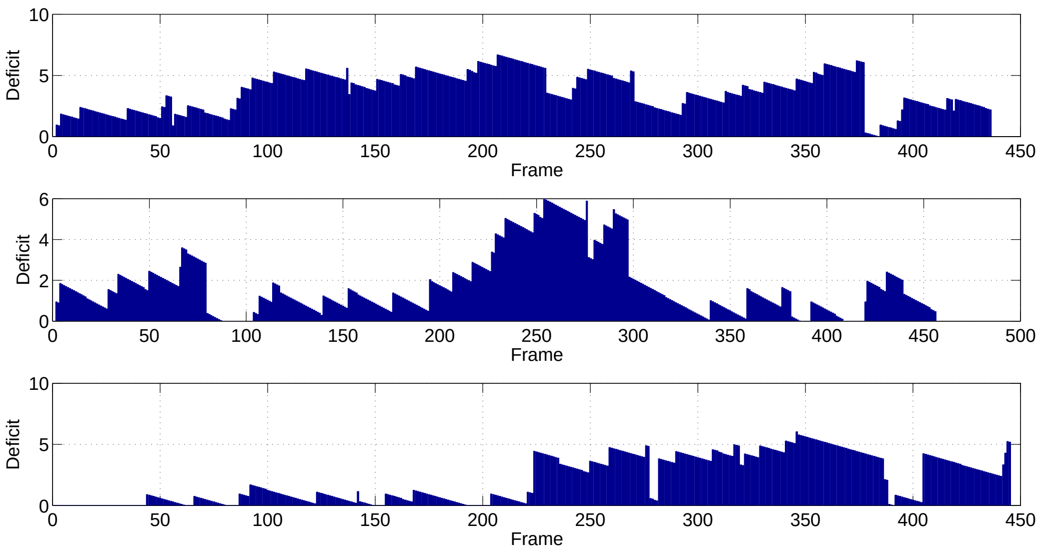

Figure 5.6: Deficit trajectory of phones in a cluster.

helped and help each other frequently.

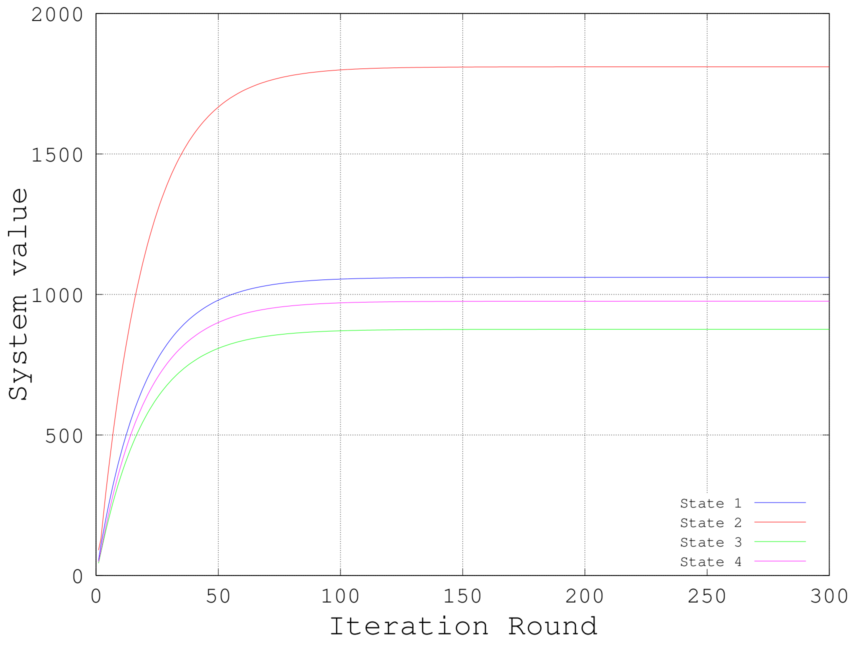

Figure 5.7: Convergence of value iteration

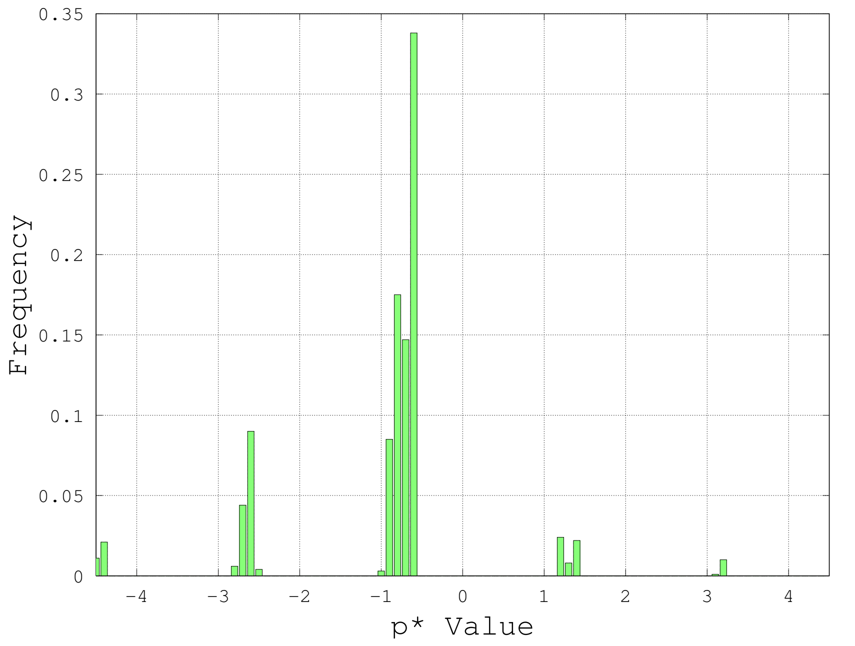

Figure 5.8: Transfer distribution from trace

6. CONCLUSION

We studied the problem of realtime streaming applications for wireless devices that can receive coded data chunks via both expensive long-range unicast and an inexpensive short-range broadcast wireless interfaces. QoS was defined in terms of the fraction of blocks recovered successfully on average.

We utilized a Lyapunov stability argument to propose a minimum cost D2D transmission algorithm that can attain the required QoS, in the case of unreliable and reliable D2D transmissions. We also showed how to calculate the minimum cost B2D usage per device. We showed how the infinite field size assumption under which the algorithms are derived does not have a significant impact on scaling and performance, and then showed how performance changes with parameters using simulations.

Additionally, we studied the problem of providing incentives for cooperation in wireless streaming networks. The system wide objective is to incentivize truth telling about individual user states so that a system wide cost minimizing allocation can be used. We developed such mechanisms for two models–a single cluster, and multiple clusters with user mobility. In both cases, the future states of agents must be estimated so as to provide the right incentives to users. Finally, we described our design decisions for an Android implementation.

REFERENCES

[1] N. Abedini, M. Manjrekar, S. Shakkottai, and L. Jiang. Harnessing multiple wireless interfaces for guaranteed QoS in proximate P2P networks. In *Proc. Intl. Conference on Communications in China*, pages 18–23, Beijing, China, Aug. 2012.

[2] D. Bergemann and J. Välimäki. The dynamic pivot mechanism. *Econometrica*, 78(2):771–789, 2010.

[3] Cisco. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015.* Cisco, February 2011.

[4] B. Cohen. Incentives build robustness in BitTorrent. In *Proc. WEIS*, Berkeley, CA, June 2003.

[5] T. Courtade and R.D. Wesel. Coded cooperative data exchange in multihop networks. *Arxiv preprint arXiv:1203.3445v1*, 2012.

[6] CynanogenMod. Android Community Operating System. http://www.cyanogenmod.org/, 2013. [Online; Accessed: Aug. 2013].

[7] S. Deb, M. Médard, and C. Choute. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. *IEEE Trans. on Information Theory*, 52(6):2486–2507, 2006.

[8] FG Foster. On the stochastic matrices associated with certain queuing processes. *The Annals of Mathematical Statistics*, 24(3):355–360, 1953.

[9] Google. Building for devices, Android Developers. http://source.android.com/source/building-devices.html, 2013. [Online; Accessed: Mar. 2013].

[10] I-H. Hou, Y-P. Hsu, and A. Sprintson. Truthful and non-monetary mechanism for direct data exchange. In *Proceedings of Allerton Conference*, pages 406–412, Monticello, IL, October 2013.

[11] I-H. Hou, Y. Liu, and A. Sprintson. A non-monetary protocol for peer-to-peer content distribution in wireless broadcast networks with network coding. In *Proceedings of WiOpt*, May 2013.

[12] I.H. Hou, V. Borkar, and PR Kumar. A theory of QoS for wireless. In *IEEE INFOCOM 2009*, Rio de Janeiro, Brazil, April 2009.

[13] L. Keller. Network Coding Utilities Library. https://code.google.com/p/ncutils/, Jan. 2011. [Online; Accessed: Dec. 2012].

[14] Z. Liu, C. Wu, B. Li, and S. Zhao. UUSee: Large-scale operational on-demand streaming with random network coding. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, San Diego, CA, March 2010.

[15] N. Milosavljevic, S. Pawar, S. El Rouayheb, Michael Gastpar, and Kannan Ramchandran. An optimal divide-and-conquer solution to the linear data exchange problem. In *Proc. of IEEE ISIT*, 2011.

[16] M.J. Neely. Energy optimal control for time varying wireless networks. *IEEE Trans. Information Theory*, 52(2):2915–2934, July 2006.

[17] A. ParandehGheibi, M. Medard, A. Ozdaglar, and S. Shakkottai. Avoiding interruptions-a QoE reliability function for streaming media applications.

IEEE Journal on Selected Areas in Communications, 29(5):1064–1074, May 2011.

[18] H. Seferoglu, L. Keller, B. Cici, A. Le, and A. Markopoulou. Cooperative video streaming on smartphones. In *Allerton*, pages 220–227, Monticello, IL, Sept. 2011. IEEE.

[19] A. Sprintson, P. Sadeghi, G. Booker, and S. El Rouayheb. A randomized algorithm and performance bounds for coded cooperative data exchange. In *Proc. of IEEE ISIT*, pages 1888–1892, Austin, TX, June 2010.

[20] M. Wang and B. Li. Lava: A reality check of network coding in peer-to-peer live streaming. In *INFOCOM 2007*, pages 1082–1090, Anchorage, AK, May 2007.