CrowdMAC: A Crowdsourcing System for Mobile Access

Ngoc Do, Cheng-Hsin Hsu, Nalini Venkatasubramanian

CrowdMAC: A Crowdsourcing System for Mobile Access

Ngoc Do[1⋆], Cheng-Hsin Hsu[2⋆⋆], and Nalini Venkatasubramanian[1]

[1] Dept. of Information and Computer Science, University of California, Irvine, USA
[2] Dept. of Computer Science, National Tsing Hua University, Hsin-Chu, Taiwan

Abstract. Staggering growth levels in the number of mobile devices and amount of mobile Internet usage has caused network providers to move away from unlimited data plans to less flexible charging models. As a result, users are being required to pay more for short accesses or underutilize a longer-term data plan. In this paper, we propose CrowdMAC, a crowdsourcing approach in which mobile users create a marketplace for mobile Internet access. Mobile users with residue capacity in their data plans share their access with other nearby mobile users for a small fee. CrowdMAC is implemented as a middleware framework with incentive-based mechanisms for admission control, service selection, and mobility management. CrowdMAC is implemented and evaluated on a testbed of Android phones and in the well known Qualnet simulator. Our evaluation results show that CrowdMAC: (i) effectively exercises the trade-off between revenue and transfer delay, (ii) adequately satisfies user-specified (delay) quality levels, and (iii) properly adapts to device mobility and achieves performance very close to the ideal case (upper bound).

Keywords: Crowdsourcing, wireless networks, resource allocation optimization

1 Introduction

Recent years have witnessed a dramatic increase in the number of mobile Internet users due to the tremendous popularity of smartphones and tablets; market forecasts point out that although only 13.3% worldwide cellular users have smartphones in 2011, the ratio is expected to reach 31.0% by 2016 [4]. In some regions, the number of smartphone users actually has exceeded that of feature phone users at the time of writing. For example, more than 46% of U.S. adults own smartphones in early 2012 [6], in Japan the smartphone penetration rate exceeds 95%. A key use of smartphones is to gain access to the Internet. Market reports place the number of mobile Internet users at 1.2 billion worldwide; the National Communications Commission of Taiwan reports that 68% of cellphone

⋆ N. Do and N. Venkatasubramanian are partially supported by National Science Foundation, grants #1059436 and #1057928.

⋆⋆ C. Hsu is partially supported by the National Science Council (NSC) of Taiwan, grant #100-2218-E-007-015-MY2.

users opt for data plans [3]. The staggering number of data plan users forces cellular service providers to deploy costly infrastructure and purchase expensive spectrum, so as to maintain quality-of-service. The resultant traffic surge has also backfired at the mobile users, as the cellular service providers have moved away from unlimited data plans to tiered services [5], and may consider time-dependent pricing [18], which in turn may increase user's monthly bill.

Existing data plans often require 1- to 3-year contracts, and may not have too many options in terms of monthly traffic quotas. This results in low quota utilization, e.g., the worldwide average unused data quota is as high as 61% [4]. Studies indicate that 48.6% of AT&T data plan users incur very low monthly traffic (lower than the least expensive 300-MB data plan), and 81% of these users have a residue quota of more than 100 MBs every month [8]. The aforementioned statistics reveal that: (i) light mobile Internet users may find the contracts and relatively high data plan quotas less appealing, and (ii) other mobile Internet users may have residue data plan quotas. We argue that these two types of users could form a virtual community or marketplace, similar to My Virtual Neighbor [7], and share the resources with each other.

In this paper, we present *CrowdMAC*, a *crowdsourcing* solution for providing on-demand mobile Internet access. Crowdsourcing refers to open platforms, that enable "loose sharing of resources between undefined publics" that may be human or online. Crowdsourcing platforms often incorporate human participation [15] allowing entities to outsource tasks to individuals and gain information from the collective processing. In CrowdMAC, mobile users in need of Internet access (or network connectivity in general) leverage the ability of other nearby users to provide access to the resource – i.e., mobile Internet. In particular, light mobile Internet users may completely avoid data plans, and *hire* other mobile Internet users with residue resources (e.g., data plan quotas, battery) to transfer data to/from the Internet for them. The hired mobile Internet users are referred to as *mobile Access Points (mobile APs)*, and the hiring mobile Internet users are referred to as *mobile devices* throughout this paper. The mobile devices and mobile APs communicate with each other via short-range wireless networks, such as WiFi ad hoc, WiFi Direct [2] and Bluetooth, and hence do not incur significant traffic overhead over the cellular networks. This kind of sharing is called *tethering*, and is widely used among mobile devices belonging to the same user. Some cellular service providers, including Verizon Wireless, Clearwire, and Sprint sell dedicated mobile gateways [23], which essentially are mobile APs. Cooperative use of multiple access networks (Cellular, WiFi, Bluetooth, and etc.) is becoming increasingly feasible; enabling rich mobile applications using such hybrid access networks is a current topic of research [11, 13, 14, 16, 19, 21].

Matching mobile devices with nearby mobile APs is not an easy task – the following challenges arise in creating a meaningful incentive-based design that ensures robust data transfer despite dynamics of mobile users and connectivities.

1. How does a mobile AP make an admit/reject decision upon receiving a request from a mobile device? Admitting a larger number of requests brings

higher revenues, but causes buffer overflow and a longer end-to-end delay for file transfer. Higher delays may turn users away from the system.
2. How does a mobile device select a mobile AP and a corresponding service (Cellular or WiFi network)? Services charge different fees and provide different qualities of service.
3. How do mobile devices and mobile APs deal with uncertain channel conditions caused by mobility? The system has to solve the situation where a mobile device is moving out of its range before its file is completely transferred to the mobile AP. Similarly, a mobile AP's direct Internet access link may be disconnected due to mobility.
4. How does the system handle the associated security issues and legal implications of sharing mobile access? How does this scheme fit into the ISP/network provider ecosystem?

In this paper, we focus on the mechanisms to enable crowdsourced mobile access to the Internet where mobile APs with direct access are able to offer connectivity to mobile devices without easy direct access (i.e., challenges 1, 2 and 3). In particular, CrowdMAC implements incentive-based mechanisms for admission control, service selection and mobility handling in a distributed middleware framework described in Section 2. The proposed mechanisms are implemented directly on off-the-shelf Android devices and require no changes to the underlying network boxes such as cellular base stations and WiFi Hotspots. Challenge 4 (i.e., security and integration with ISPs/providers) is a topic of future work; we discuss our views on this in Section 7.

2 CrowdMAC: Architecture and Approach

In this section, we describe the design principles and architecture of the CrowdMAC crowdsourcing system that enables mobile devices to hire mobile APs to upload and download data.

2.1 Hardware and Network Architecture

Fig. 1 illustrates the hardware components in the proposed system architecture that include the following parties.

- **Mobile Device**: A device wishes to upload/download a file, but currently does not have a *last connection* to the Internet.
- **Mobile AP**: A mobile AP is also a mobile device which possesses a last connection(s) to the Internet via cellular base stations or WiFi Hotspots. We assume that mobile APs possess data plans with network service providers, and are thus able to send/receive control messages to/from the Internet. In our proposed system, a mobile AP is willing to transfer data for a mobile device for fees. In Fig. 1, A, C, and D are mobile APs willing to transfer data for mobile device B.

  A mobile AP may have one or more last connections over either cellular base stations or WiFi Hotspots. The last connections correspond to different

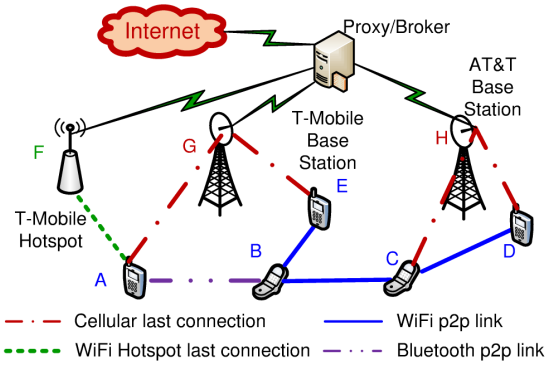


Fig. 1. Network architecture of the proposed system.

service providers, which charge the mobile AP at various prices. Moreover, transferring data over different wireless networks consumes diverse local resources, including battery levels and CPU cycles. Hence, each mobile AP may set a different price for transferring data over each last connection. Because a mobile AP may simultaneously transfer data over multiple last connections, it can concurrently offer multiple transfer **services**. Hence, by service, we refer to a specific last connection of a mobile AP.

– **Proxy/broker**[3]: The system consists of a proxy/broker located behind last connections that keeps records of data amount transferred by mobile APs for mobile devices and the corresponding payments. It ensures that data is transferred from/to the Internet through mobile APs in totality and correctly.

Note that although C in Fig. 1 has a last connection, it may access the Internet through mobile AP D at times of poor connectivity. Nonetheless, we assume that a smartphone can be either a mobile device or a mobile AP, but not both simultaneously in this paper.

2.2 Software Architecture

CrowdMAC is envisioned as a distributed middleware system that resides on mobile devices, mobile APs, and the network proxy/broker. A mobile AP may provide concurrently multiple services, each for a last connection. Fig. 2 depicts the key software components/modules in CrowdMAC: (i) AAA Module, (ii) Control Plane Module, (iii) Data Plane Module, and (iv) Connection Manager. Fig. 2 also illustrates the operational workflow of a CrowdMAC session. Without loss of generality, we illustrate the system workflow using a data upload scenario, and minor modifications required for a download session.

[3] Multiple proxies/brokers can be used to scale the system. Proxies/brokers can be offered by: (i) a service provider, (ii) an alliance of multiple providers, or (iii) a third-party company.

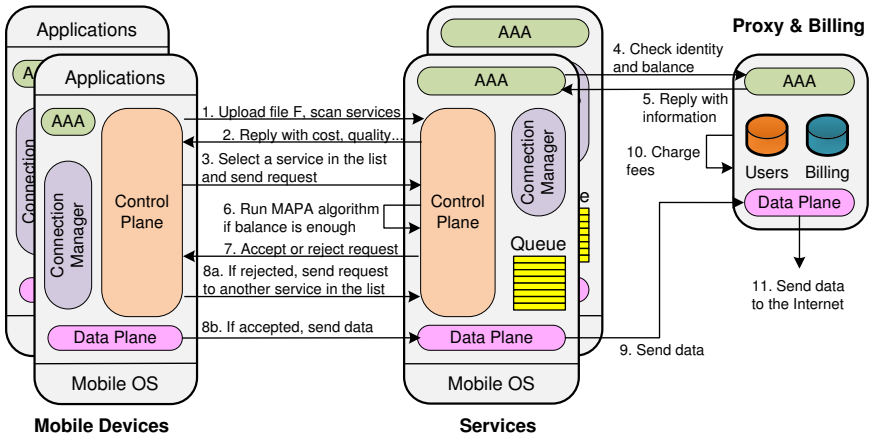


Fig. 2. Components and workflow (upload case) in the middleware system.

AAA Module: This module maintains information about users in the system and their mobile access sessions via databases at the proxy/broker. Mobile devices and mobile APs register themselves with the proxy/broker and provide identity information stored in database *User*. During operation, logs are maintained about the amount of data transferred by mobile APs for mobile devices and the corresponding monetary fees in database *Billing*. Our initial implementation uses a prepaid option. Mobile devices (i.e., users) make a monetary deposit to their account before they can use services provided by mobile APs. Prior to transferring data for a mobile device, the mobile AP accesses the proxy/broker to verify if the mobile device is able to cover the fees for the transfer.

Control Plane Module: The control plane modules at different nodes cooperate to establish and maintain connectivity between a mobile device that requests access and the mobile APs that enable this access for a fee. When a mobile device wishes to use a service provided by a mobile AP, it first establishes a connection to the mobile AP through a discovery process in the Control Plane. Mobile devices and mobile APs discover one another as follows. When a user has a file to upload, the Control Plane at the mobile device executes a *service scan* by broadcasting a one-hop message with the size of the file. The Control Plane at the surrounding mobile APs respond to the scan message with the corresponding fees to transfer the file to the Internet. The price of each service is a function of the last connection cost and the local resource consumption dependent on the data amount transferred. It can be manually set by the mobile AP owner. Once the mobile device finds a list of services, it will select a service in the list and send a request containing its identity stored in its AAA to the corresponding mobile AP to use the service. If there is no service found, the mobile device will wait for some time before repeating the scan process. Upon receiving a request from the mobile device, the Control Plane communicates with the proxy/broker

through the AAA component to authenticate the user. The identity information provided in the user request is used to verify the mobile device's balance at the billing server. If the balance cannot cover the fees, the request is rejected. Otherwise, the Control Plane goes further one more step. It makes a decision if it should admit or reject the request by running an admission control algorithm, *MAPA*, and then replies the decision to the requesting mobile device.

Data Plane Module: If the mobile device requesting service receives an *admit* response, its Control Plane triggers its Data Plane to start transferring data. Otherwise, it will remove the service out of the list, and choose another service to send the request. In the case of an admitted upload request, the mobile device reads the file and begins transmitting data packets to the Data Plane at the mobile AP. The Data Plane at the mobile AP maintains **a FIFO Queue** to buffer the receiving packets and transmits the buffered packets to the proxy/broker over its last connection. The proxy/broker then transmits the packets to the Internet. Once the whole file is successfully uploaded, the Data Plane at the mobile AP sends a confirmation back to the mobile device.

Connection Manager: This component manages network interfaces available on the device. It notifies the Control Plane and the Data Plane the availability of surround mobile devices and APs and the breakage of connections.

3 CrowdMAC Admission Control

We next present an admission control algorithm, called **MAPA** (Mobile AP Admission control) that allows a service to admit or reject requests from mobile devices based on the characteristics of the incoming requests, their potential to generate increased revenue for the service and the current set of ongoing commitments made by the service. Key design criteria of the admission control algorithm include: (a) maximizing long term revenue (measured as average revenue over time); (b) ensuring overall stability of the system (implying no buffer overflows at the service); and (c) providing a distributed and practical implementation.

We characterize the problem and develop an algorithm to enable admission control decision using a Lyapunov optimization framework. The Lyapunov approach is well suited to creating an operational framework for the mobile access crowdsourcing problem in this paper since it provides: (a) a meaningful theoretical underpinning for stability analysis of the dynamic execution environment and (b) the inherent queuing theoretic based modeling that is well suited to the network transmission problem. Using Lyapunov framework for designing admission control algorithms has been studied in [17, 20] for cellular base stations and WiFi Hotspots where resources are not an issue. Our problem considered here is much more challenging because mobile APs are resource constrained, and thus ensuring stability of CrowdMAC must consider this fact.

3.1 Basic Models and Problem Formulation

Consider a network consisting of a set of mobile APs M, a set of mobile devices D, and a set of wireless links L. Let us also assume that time is divided into slots

of t units and requests are made by mobile devices to mobile APs for wireless services in a specific time slot. The following terminologies are used in developing the problem.

Services. Each mobile AP m provides a set of services S_m – which in our case represents disparate last hop connections. Let S be a set of all services in the network. When mobile device d wishes to upload a file to the Internet, it selects a service s_m provided by mobile AP m and waits for an accept or reject decision from s_m.

Monetary costs. Mobile device d, who wishes to upload/download a file with size f_d through a service, must pay a price or cost, which is a function of f_d. In this paper, this is the revenue earned by the service and we consider this as monetary cost. If d selects service s_m provided by mobile AP m, d will be charged a cost denoted by $C(s_m, f_d)$. Typically, $C(s_m, f_d)$ incorporates two subcosts: (a) a local resource consumption cost, $C_r(s_m, f_d)$, that captures the cost incurred due to use of resources at m , and (b) a provider cost, $C_p(s_m, f_d)$, paid by the mobile AP to the network provider (e.g., telco) for covering the data plan cost of s_m's last connection[4]. Hence, $C(s_m, f_d) = C_r(s_m, f_d) + C_p(s_m, f_d)$.

The exact choice of the cost function is immaterial to our admission control technique. Our system works well with any non-decreasing concave function $C(s_m, f_d)$. For example, energy consumption, a dominating portion of the local resource cost for mobile devices, $C_r(s_m, f_d)$, can be modeled as a non-decreasing concave function of the amount of transmitted data [12]. Similarly, a mobile AP m with a 200 MB, \$10 data plan may employ a linear function such as $C_p(s_m, f_d) = \frac{10 f_d}{200(1024)^2}$ to represent the provider cost.

Workload arrival and departure. A mobile AP m may have many mobile devices requesting to use its services for file upload/download. We assume that each mobile device has only one file to upload/download at a time, and the maximum file size is F_{max}. A mobile device approaches m when it wants to use m's services, and departs when it completes the transmission or moves out of m's range. Let $R_{s_m}(t)$ denote the set of mobile devices requesting to use service s_m provided by m in time slot t, and $A_{s_m}(t)$ denote the total workload requested from service s_m in time slot t. That is, $A_{s_m}(t) = \sum_{d \in R_{s_m}(t)} f_d$. Assume that $A_{s_m}(t) \leq A_{max} \quad \forall s_m, t$, where A_{max} is the maximum possible workload at any mobile AP. We assume that mobile device arrival rate to m is i.i.d over time slot. Every time slot, s_m admits a set of mobile devices $r_{s_m}(t) \in R_{s_m}(t)$ with an admitted workload of $a_{s_m}(t) \leq A_{s_m}(t)$. The time average data amount admitted to s_m is defined: $\bar{a}_{s_m} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{a_{s_m}(\tau)\}$.

If a mobile device moves out of m's range and has not completed transmission of the file, it finds another mobile AP to upload the remaining file. We let $g_{s_m}(t)$ be the residual data that was not sent via s_m due to disconnections between the mobile AP and mobile devices. Assume that $g_{s_m}(t) \leq g_{max} \quad \forall s_m, t$. The time average residual data not sent via s_m is: $\bar{g}_{s_m} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{g_{s_m}(\tau)\}$.

[4] For simplicity, we consider the provider cost also includes cost paid for the proxy/broker provider.

Network capacity. For each link l in L, let $\mu_l(t)$ and $\theta_l(t)$ denote the amount of data transferred through link l and the maximum capacity of link l at time slot t. Thus, $\mu_l(t) \leq \theta_l(t)$. Value $\theta_l(t)$ depends on channel quality while $\mu_l(t)$ depends on the available data for transfer and $\theta_l(t)$. We assume that channel quality is i.i.d over time slots. We denote θ_{max} as the maximum channel capacity under any quality channel, i.e., $\theta_l(t) \leq \theta_{max} \quad \forall l, t$.

We denote $\mu_{s_m}^{out}(t)$ as the amount of data transmitted out of service s_m in time slot t. $\mu_{s_m}^{out}(t) = \mu_l(t)$ if l is s_m's last connection for the upload case, or $\mu_{s_m}^{out}(t) = \sum_{d \in \Gamma_{s_m}(t)} \mu_{m-d}(t)$ for the download case where $\Gamma_{s_m}(t)$ denote a set of mobile devices that s_m is serving. The time average outgoing data amount from s_m is defined as: $\bar{\mu}_{s_m}^{out} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_{s_m}^{out}(\tau)\}$.

We define the total incoming data amount to s_m at time slot t as $\mu_{s_m}^{in}(t) = \sum_{d \in \Gamma_{s_m}(t)} \mu_{d-m}(t)$ for the upload case, or $\mu_{s_m}^{in}(t) = \mu_l(t)$ where l is the last connection for the download case. We denote ϑ_{max} as the maximum incoming data amount to any service at any time slot, i.e., $\mu_{s_m}^{in}(t) \leq \vartheta_{max} \quad \forall s_m, t$. The time average incoming data amount at s_m is defined as: $\bar{\mu}_{s_m}^{in}(t) \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_{s_m}^{in}(\tau)\}$.

Problem formulation. Given a set of mobile devices $R_{s_m}(t)$ arriving at service s_m and requesting to use service s_m with workload $A_{s_m}(t)$, s_m determines a set of mobile devices $r_{s_m}(t)$ to be admitted for using the service such that:

$$\text{max:} \quad \bar{C}_{s_m} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{d \in r_{s_m}(\tau)} \mathbb{E}\{C(s_m, f_d)\}; \tag{1a}$$

$$\text{st:} \quad \bar{\mu}_{s_m}^{in} \leq \bar{\mu}_{s_m}^{out}; \tag{1b}$$

$$\bar{a}_{s_m} \leq \bar{g}_{s_m} + \bar{\mu}_{s_m}^{in}. \tag{1c}$$

Objective function (1a) indicates that s_m attempts to maximize the time average revenue while maintaining its system's stability over time as shown in constraint (1b). Constraint (1c) is to ensure that admitted data will be transmitted to s_m unless the mobile device leaves the range of the mobile AP.

3.2 Our Proposed Admission Control Algorithm - MAPA

To solve the problem (1a)-(1c), we design an admission control algorithm using the Lyapunov framework. To ensure the two constraints (1b) and (1c), we maintain a real queue and a virtual queue for each service s_m in S presented below.

Real queue and the system stability - Constraint (1b): Data transmitted to/from mobile devices from/to s_m is stored at a queue before it can be transmitted further. Each service has one real queue to store request data; let $Q_{s_m}(t)$ denote the queue backlog, which is the number of bytes in that queue, at the beginning of time slot t for service s_m. The queue backlog's evolution is expressed as:

$$Q_{s_m}(t+1) \triangleq \max(Q_{s_m}(t) - \mu_{s_m}^{out}(t) + \mu_{s_m}^{in}(t), 0). \tag{2}$$

We define **stability** of our system as finite average queue backlog:

$$\bar{Q}_{s_m} \triangleq \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{Q_{s_m}(\tau)\} < \infty. \tag{3}$$

Notice that if (3) is maintained then constraint (1b) is satisfied. In real implementations, the real queue at mobile devices can be ignored, as a mobile device may read files on-demand rather than prefetching them into memory.

Virtual queue and the admitted data - Constraint (1c): To satisfy constraint (1c) without breaking the Lyapunov framework, we employ the concept of a virtual queue that captures the projected load over time based on requests admitted by the service. Each service s_m maintains a virtual queue $U_{s_m}(t)$ that is just a software counter, i.e. does not hold actual data packets. $U_{s_m}(0)$ is set to 0 and $U_{s_m}(t)$ evolves over time as follows:

$$U_{s_m}(t+1) \triangleq \max(U_{s_m}(t) - g_{s_m}(t) - \mu^{in}_{s_m}(t) + a_{s_m}(t), 0). \tag{4}$$

Lemma 1 *If the virtual queue is stable, i.e.,*

$$\bar{U}_{s_m} \triangleq \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{U_{s_m}(\tau)\} < \infty. \tag{5}$$

then constraint (1c) is satisfied.[5]

We define **a revenue weight** V that works as a control parameter providing a trade-off between revenue and stability. By employing the Lyapunov framework with the real queue, the virtual queue, and parameter V, we translate the original problem (1a)-(1c) to another optimization problem that can be solved by 3 tasks presented in the following algorithm.

The MAPA algorithm, summarized in Algorithm 1, solves the problem in a distributed manner, and can be directly deployed on off-the-shelf devices without requiring modifications of the existing softwares at cellular base stations and WiFi Hotspots. It works as follows. Every time slot, every service s_m performs three tasks: (1) determining which mobile devices requesting to use s_m will be admitted; (2) determining if mobile devices that s_m is serving should transmit data to s_m's real queue or not; and (3) transmitting data out of the real queue $Q_{s_m}(t)$ whenever the queue is not empty.

To do task (1), service s_m needs to solve the problem (6a)–(6b). It does that by going through every mobile device d in $R_{s_m}(t)$, and calculates $e_d = VC(s_m, f_d) - U_{s_m}(t)f_d$. If $e_d \geq 0$, then d is admitted. Otherwise, d is rejected. Therefore, the procedure's complexity is $O(n)$.

Service s_m performs task (2) to control congestion. In the case of admitted upload requests, every time slot, if s_m observes that the real queue backlog is larger than the virtual queue backlog, s_m broadcasts a *STOP* message to

[5] The proofs of our lemmas and theorems throughout this paper are omitted without breaking the flow, due to the space limitations.

Algorithm 1 Admission Control and Transmission Schedule

1. **Admission Control Procedure** : Every time slot t, service s_m selects a set of mobile devices $r_{s_m}(t)$ from $R_{s_m}(t)$ such that the following maximization condition is satisfied:

$$\text{max:} \quad \sum_{d \in r_{s_m}(t)} (VC(s_m, f_d) - U_{s_m}(t) f_d); \tag{6a}$$

$$\text{st:} \quad r_{s_m}(t) \in R_{s_m}(t). \tag{6b}$$

2. **Transmission Schedule for Incoming Data**: Every time slot t, s_m schedules transmission for incoming data such that if $Q_{s_m}(t) - U_{s_m}(t) > 0$, it does not allow mobile devices to transmit data to the service as well as download data from the Internet.
3. **Transmission Schedule for Outgoing Data**: Every time slot t, s_m always transmits data out of the service if its real queue has data to send.

request mobile devices not to transmit data in the current time slot. In the case of admitted download requests, the service generates threads to download data packets from the Internet and put the packets into the real queue. At the beginning of a time slot, if there is congestion at the real queue, the service stops the threads from downloading during the time slot.

3.3 Performance Analysis for the MAPA Algorithm

Let's denote $C^*_{s_m}$ as the maximum time average revenue of the problem (1a)–(1c) achieved by some stationary randomized algorithm. We first show that MAPA achieves a time average revenue $\bar{C}$ arbitrarily close to $C^*_{s_m}$.

Theorem 1 (Revenue lower bound) *Our MAPA algorithm achieves a lower bound of the time average revenue:*

$$\bar{C}_{s_m} = \lim_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \sum_{d \in r_{s_m}(\tau)} \mathbb{E}\{C(s_m, f_d)\} \geq C^*_{s_m} - \frac{\beta}{V}, \text{ where} \tag{7}$$

$$\beta = \frac{1}{2}[\max(\theta_{max}, \vartheta_{max})^2 + \max(g_{max} + \vartheta_{max}, A_{max})^2]. \tag{8}$$

Next, we show there is a trade-off O(1/V, V) between the achieved revenue and the quality-of-services in the MAPA algorithm.

Theorem 2 (Worst case bounds on real and virtual queues) *MAPA provides worst case bounds of the service's real and virtual queue size:*

$$U_{s_m}(t) \leq V y^*_{s_m} + A_{max} = U^*_{s_m}; \quad Q_{s_m}(t) \leq V y^*_{s_m} + A_{max} + \vartheta_{max} = Q^*_{s_m}, \tag{9}$$

where y_{s_m} is a service based function with respect to f_d ($f_d \leq F_{max}$) defined as $y_{s_m} = \frac{C(s_m, f_d)}{f_d}$ and $y^*_{s_m}$ is the maximum value of y_{s_m}, $Q^*_{s_m}$ and $U^*_{s_m}$ indicate the maximum queue backlog of s_m's real queue and virtual queue.

The above two theorems illustrate the trade-off between the time average revenue and the worst case queue bounds. If service s_m increases its time average revenue O(1/V), then the size of both real and virtual queues increases O(V). For example, when queue sizes reach infinity, s_m achieves the time average revenue arbitrarily close to the optimal value.

However, note that a large queue size may lead to high end-to-end delay. The following lemma shows that MAPA leads to a linear relationship between the end-to-end delay and the backlogs of the real and virtual queues.

Lemma 2 (Relating queue size and delay) *Let's denote the current backlogs of the real and virtual queues as $Q^o_{s_m}$ and $U^o_{s_m}$, respectively. Assume that the incoming and outgoing data rates $\mu^{in}_{s_m}$ and $\mu^{out}_{s_m}$ of service s_m are unchanged over time slots. The MAPA algorithm achieves the worst case delay T to send all data in the real and virtual queues to the Internet:*

$$T = \hat{t} + \hat{T} \qquad \textit{slots, where} \tag{10}$$

$$\hat{t} = \max(0, \lceil \frac{Q^o_{s_m} - U^o_{s_m}}{\mu^{out}_{s_m}} \rceil); \quad \hat{Q}^o_{s_m} = \max(0, Q^o_{s_m} - \hat{t}\mu^{out}_{s_m}); \tag{11}$$

$$\hat{T} = \begin{cases} \lceil \frac{\hat{Q}^o_{s_m} + U^o_{s_m}}{\mu^{out}_{s_m}} \rceil & \textit{if } \mu^{in}_{s_m} \geq \mu^{out}_{s_m}; \\ \lceil \frac{\hat{Q}^o_{s_m}}{\mu^{out}_{s_m}} + \frac{U^o_{s_m}}{\mu^{in}_{s_m}} \rceil + 2 & \textit{if } 2\mu^{in}_{s_m} > \mu^{out}_{s_m} > \mu^{in}_{s_m}; \\ \lceil \frac{U^o_{s_m}}{\mu^{in}_{s_m}} \rceil & \textit{if } \mu^{out}_{s_m} \geq C\mu^{in}_{s_m} (C \geq 2). \end{cases} \tag{12}$$

Selection of parameter V: We have shown that V controls the trade-off between time average revenue and the delay. Two heuristics to select V are:

- *Bounding memory consumption*: Let M^* be memory bound at a service. Per Theorem 2, we have $Q^*_{s_m} = Vy^*_{s_m} + A_{max} + \vartheta_{max} \leq M^*$, which enables us to pick a V value satisfying the memory bound.
- *Delay quality*: The delay quality is a quality-of-service metric for a service to deliver all data currently admitted in the virtual queue and buffered in the real queue to the Internet. Let T_{s_m} be the bounded delay quality, service s_m wishes to offer. If T_{s_m} is small, the amount of data admitted and buffered is small. That leads to a short transfer delay to deliver data to the Internet and a low revenue for the service. According to Lemma 2, T_{s_m} is the upper bound of T. In the worst case, we have $Q^o_{s_m} = Q^*_{s_m}$ and $U^o_{s_m} = U^*_{s_m}$. We set the minimum data rate incoming to s_m to be $\mu^{in}_{s_m}$ and the minimum outgoing data rate from the service to be $\mu^{out}_{s_m}$. Plugging $Q^*_{s_m}$, $U^*_{s_m}$, $\mu^{out}_{s_m}$ and $\mu^{in}_{s_m}$ into Lemma 2, we estimate V by (9) such that T is bounded by T_{s_m}.

4 Handling Mobile Device and Mobile AP Mobility

Due to mobility, connections between mobile devices and services as well as last connection may be lost. Fundamental to our approach is the ability to monitor the liveness of a link. For a service provided by the mobile AP, once it detects

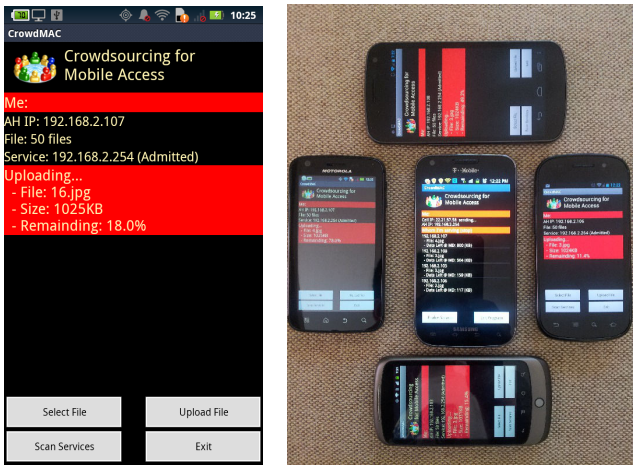
the disconnection, it removes the mobile device from the list of the devices it is serving, updates the virtual queue, and drops the mobile device's packets out of the real queue. For the mobile device, it stops using that service, and scans for another service. Advanced technologies such as WiFi Direct [2] on Android OS 4.0.4 support APIs to detect the disconnection quickly. For other network technologies, we broadcast HELLO messages to detect the breakage. In addition to liveness monitoring, CrowdMAC incorporates the following techniques to support continuity of upload/download under mobility conditions.

Technique 1 - File Segmentation for easy management and recovery: We divide each large file into smaller sized chunks and transmit each chunk independently. This is to reduce the transfer time and consequently enable successful delivery of chunks, despite connectivity changes. Once chunks have been sent, missing packets and chunks are retransmitted.

Technique 2 - Service Selection and Mobility Awareness: When multiple mobile APs are available, a mobile device can choose one from them using one of the following strategies:

1. *Cost Based Service Selection*: The mobile device simply chooses the service with the lowest cost.
2. *Delay Quality Based Service Selection*: The mobile device picks the service with the best delay quality (as defined in Section 3.3).
3. *Mobility Based Service Selection*: The mobile device chooses a service based on reliability of the link from itself to the service. We assume that mobile devices and APs travel using a well known Random Waypoint model. Link reliability is determined by estimating link duration that indicates how long a link is likely to last. The mobile device picks the service with the longest link duration. The duration of link l denoted as $\delta_l(t)$ can be estimated using existing efforts, such as Qin and Zimmermann [22] using an analytical model to estimate the link duration based on GPS readings.

Technique 3 - Incentives to Constrain Mobility during file transfer: We employ charging schemes to motivate mobile devices and mobile APs to constrain mobility during file transfer. In file download scenarios, consider a situation in which a mobile device d requests to download a file f_d. The file has been downloaded in the real queue, but just part of that has been sent to the mobile device because the mobile device moves out of the service's range. This is unfairness to the mobile AP because it had to pay for the Internet access cost $C_p(s_m, f_d)$ to download the file. To address this issue, the mobile AP divides the file into multiple smaller chunks. Each time when the service starts downloading a chunk c_d, the mobile device will be charged a fee $C_p(s_m, c_d)$. Once the mobile device fully receives the whole chunk from the service, it pays the remaining fee $C_r(s_m, c_d)$. If the service fails to download the chunk, it returns $C_p(s_m, c_d)$ to the mobile device. Note that if the service does not receive a confirmation of receiving a chunk from the mobile device, it will not download the next chunk. With this mechanism, both sides, the mobile device and the service, have motivation to maintain the link stability until the whole chunk is successfully downloaded. In file upload scenarios, incentive provision is simpler. Fees are charged on what the



(a) (b)

Fig. 3. Our Android based testbed: (a) a screenshot of our system on a mobile device and (b) an experiment with 5 different types of mobile phones.

service uploads through the broker/proxy. While the mobile device is motivated to keep the link stable to complete the transfer, the mobile AP is incentivized to keep the last connection stable to earn the revenue.

5 Testbed Implementation: A Proof-of-Concept

Our Testbed and Settings. We implement our middleware on an Android based testbed which consists of a Linux server and multiple Android smartphones located in the University of California, Irvine. An interesting thing is our implementation does not depend on synchronized timers among mobile devices and mobile APs. The admission control procedure in Algorithm 1 runs once for each request, rather than in every single time slot. That is, whenever a mobile device has a file to transfer, it sends a request to a mobile AP. That mobile AP then invokes the MAPA algorithm right away, and immediately sends back the accept/reject decision. Also, the transmission scheduling procedure in Algorithm 1 keeps track of the backlogs of virtual and real queues on mobile APs. Once a mobile AP sends a STOP message, the corresponding mobile device stops transmitting data to that mobile AP. Since MAPA does not require synchronized timers, it can be readily implemented in the CrowdMAC system.

We deploy our system on multiple types of Android phones: a Samsung Galaxy SII as a mobile AP connecting to the T-Mobile network, and four mobile devices including Google Nexus, Nexus S, HTC Nexus One and Motorola Atrix. The phones run Android OS versions from 2.3.6 to 4.0.4. They are placed in a labroom with no mobility and connect to each other over a WiFi peer-to-peer network using WiFi Tether [1]. The mobile AP is equipped with a data plan of

$30 for 5 GB, and it charges $5 per GB for local resource consumption. The T-Mobile network is measured to have bandwidth between 311 and 748 Kbps throughout our experiments. In each experiment, each mobile device uploads 50 equal-sized files. For each file, a mobile device sends a request to the mobile AP and waits for its decision. If the request is rejected, the mobile device waits for 2 secs before it resends the request. The mobile device skips the current file if the its request is rejected for 5 times. Fig. 3 presents our system's GUI and the phones in an experiment.

Experimental Results. We consider two metrics (i) *Revenue*, which is the total revenue generated by the service in each experiment, and (ii) *Transfer delay*, which is the average per-file transfer time, from the instant a request is admitted to the instant when the last packet is transferred.

Reliable transfer. We repeat the experiments with various parameter $V \in [0, 1500]$, and we consider two file sizes: 512 KB and 1 MB. Across all experiments, we find that all the admitted requests lead to successful file uploads. This confirms that the proposed CrowdMAC system and MAPA algorithm do work.

Trade-off between revenue and transfer delay. Fig. 4 presents the revenue and transfer delay under different V. This figure shows that smaller V leads to fewer admitted requests, and thus lower revenue. Furthermore, fewer admitted requests means lighter-loaded networks, and thus shorter transfer delay. For example, with $V = 10$, the mobile AP only admits half of the 512 KB files at the end; while with $V = 1000$, the mobile AP admits all the requests. Fig. 4 shows the effectiveness of V: the MAPA algorithm supports wide ranges of revenues and transfer delays.

User-specified delay quality level. We next evaluate the V selection heuristic proposed in Sec. 3.3. We consider the heuristic that maps a delay quality level, between 45 and 135, to a suitable V value. Fig. 5 presents the revenue and transfer delay achieved by various delay quality levels. We make two observations. First, higher delay quality level leads to more admitted requests, which in turn results in high revenue and transfer delay. Second, Fig. 5(b) reveals that the average transfer delay is always smaller than the delay quality level specified by the user throughout our experiments.

6 Simulation Based Evaluation

6.1 Settings

We implement our middleware system in Qualnet 5.02 [9]. In simulations, we use WiMAX to simulate last connections from mobile APs to the Internet and use IEEE 802.11 in ad hoc mode to simulate links among mobile devices and mobile APs. We configure the WiMAX range to cover all mobile APs, and we set the 802.11 range to be 120 m. Two-Ray model is used as the propagation model and UDP is used as the transport protocol in our simulations. Simulation time is set long enough such that all files can be completely transferred.

We follow the current data plan prices of T-Mobile [10] to design the cost functions. We define three cost functions: (1) $S_1 = \frac{10 \times 100x}{2 \times (1024)}$ cent/KB ($10 for

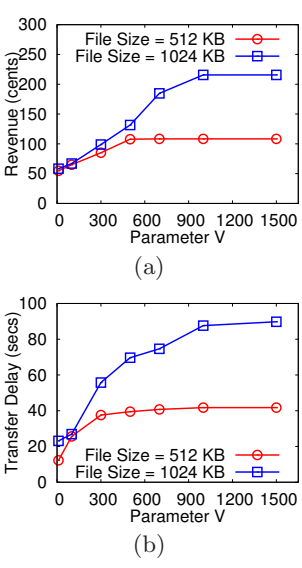


Fig. 4. Diverse Parameter V: (a) revenue and (b) transfer delay.

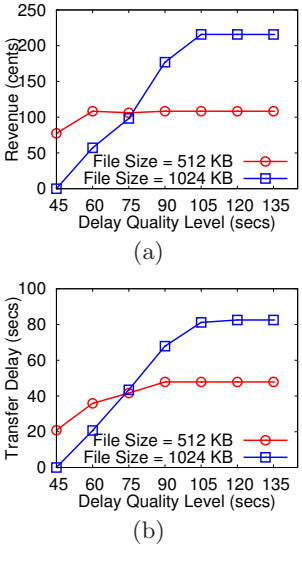


Fig. 5. Diverse user-specified delay quality level: (a) revenue and (b) transfer delay.

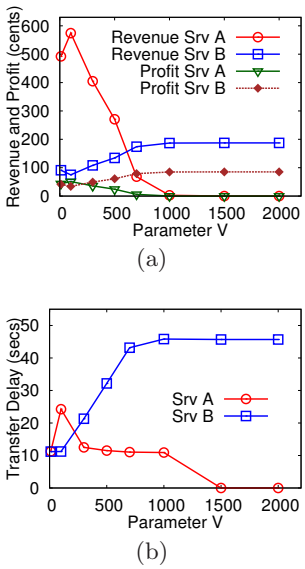


Fig. 6. Trade-off between: (a) revenue and profit, and (b) transfer delay.

200 MB), (2) $S_2 = \frac{30 \times 100x}{5 \times (1024)^2}$ cent/KB ($30 for 5 GB), and (3) $S_3 = \frac{5 \times 100x}{(1024)^2}$ cent/KB ($5 for 1 GB) where x is the amount of transferred data in KB. S_1 and S_2 cover the Internet access cost, and S_3 covers the local resource consumption cost. Each mobile AP may decide to employ one of the cost functions, or adapt a linear combination of two of them.

In Section 5, we examine our system with networks with a single service. In this section, we investigate our system's performance with more than one service, so as to exercise two different service selection strategies. In addition to the two metrics described earlier, we consider metrics: (i) *Profit*: the total monetary cost (C_r) charged by a service for its local resource consumption after each simulation, (ii) *Cost*: the total cost paid by a mobile device on average for each file, (iii) *End-to-end delay*: the average delay per file from the instant a mobile device scans services to the instant when the last packet reaches destination; (iv) *Number of interruptions*: the average number of disconnections during each file transfer, and (v) *Overhead*: the ratio between the total traffic amount and the total file size. We ran each simulation five times with different random seeds, and plotted the averages of the results obtained in all runs. If not otherwise specified, we consider all mobile devices to select the service based on the cost. By default, we use the same V value for all mobile APs. In the following two sections, we consider the static and mobile scenarios, respectively.

6.2 Evaluation under Static Scenarios

We configure a network with a WiMAX base station, two mobile APs, and seven mobile devices. Each mobile AP offers a service, and we call them service A and B, respectively. Service A employs a more expensive cost function of $S_1 + S_3$, and B employs a cheaper cost function of $S_2 + S_3$. All mobile devices are in the range of both mobile APs. In each simulation, every mobile device sequentially transfers 50 equal-sized files. We conduct each simulation with two file sizes: 512 KB and 1 MB. Due to the space limitations, we present the results with 512 KB files if not otherwise specified.

Generality of the MAPA algorithm. We compare MAPA against two baseline algorithms: *ALL* and *ONE*. In ALL, a service always accepts request from mobile devices no matter how much workload it is carrying on. Hence, it aims to maximize its revenue. In ONE, a service always rejects request unless it is serving no request. Thus, ONE is designed for service to provide best service quality to one request. Different from ALL and ONE, MAPA employs a parameter V. We vary $V \in [0, 2000]$, and repeat the simulations with MAPA. We find that ALL and ONE achieve similar revenue, profit, and end-to-end delay as MAPA with $V = 10$ and $V = 2000$, respectively. Hence, MAPA is a general algorithm for diverse quality-of-service needs.

Trade-off between revenue and transfer delay. We plot the resulting profit and transfer delay in Fig. 6. Fig. 6(a) shows that, with a small V, service A achieves higher revenue due to its higher Internet access cost. This in turn leads to higher workload and longer transfer delay as illustrated in Fig. 6(b). With a larger V, service B achieves higher revenue because its maximum workload is raised. With $V > 1000$, service B accepts all requests and achieves the highest possible revenue, at the expense of long transfer delay. Fig. 6 reveals the trade-off between revenue and transfer delay.

V's implication on profit. Fig. 6(a) reveals that service B makes higher profit than A when $V \in [100, 2000]$. This can be attributed to B's lower Internet access cost. In fact, with $V \in [100, 600]$, service B makes higher profit even when A achieves high revenue. With $V \in [0, 100]$, service A makes higher profit, because B has saturated its maximum workload. Service A has a larger maximum workload due to its higher cost.

User-specified delay quality level. We consider the heuristic, presented in Sec. 3.3, which maps a delay quality level to a suitable V value. We vary the delay quality level of B from 45 to 172 secs. Service A sets its delay quality level to be half of B's. Fig. 7 presents the simulation results, in which x-axis is the delay quality level of service B. This figure shows that when the delay quality level of B $\leq$ 60 secs, mobile devices completely avoid service A. This is because the delay quality level for A is too short for 512 KB files. Once the delay quality is large enough, service A starts to receive requests, and its revenue and profit increase. At the delay quality of 75 secs (i.e., 150 secs on the x-axis), service A accepts all requests. That leads to the saturated revenue and transfer delay at service A as shown in Figs. 7. We make another observation: the transfer

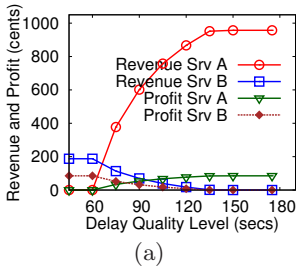


Fig. 7. Diverse delay quality level: (a) revenue and profit, and (b) transfer delay.

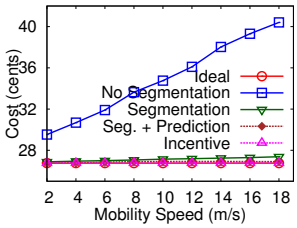


Fig. 8. Per-device total cost under different speeds.

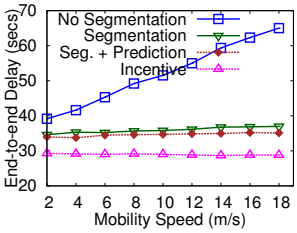


Fig. 9. End-to-end delay per file under different speeds.

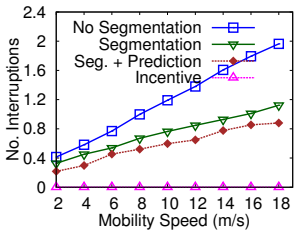


Fig. 10. Number of interruptions under different speeds.

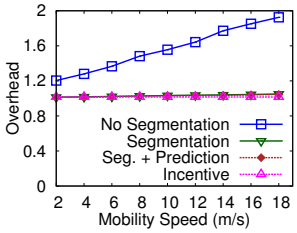


Fig. 11. Traffic overhead of our system.

delay is always lower than the delay quality level specified by the users in our simulations. This reveals the effectiveness of our heuristic.

6.3 Evaluation under Mobility Scenarios

Mobile users may move around when they are participating in CrowdMAC. To evaluate the performance of our system under user mobility, we set up a 700×700 m^2 network with 40 devices. Among these devices, there are 8 stationary mobile APs connecting to a WiMAX base station and offering Internet access services. The other mobile devices travel with Random Waypoint model unless otherwise specified and download 50 files with a size of 512 KB. A mobile device downloads only one file each time, and starts the next download 30 secs after it finishes the previous one. Mobile APs are randomly placed in the network such that they cover about 65% of the network area. Some mobile APs are placed overlapped such that a mobile device can have a chance to discover more than one mobile AP. We configure all mobile APs to use the same cost functions S_2 and S_3. Unless otherwise specified, mobile devices select service using cost based strategy.

Impact of techniques handling mobility. We repeat the simulations with different mobility handling techniques, and plot the results in Figs. 8–11. The considered techniques are:

1. *Ideal*: The expected cost under no exceptions, such as link breakages. This is the minimum cost a mobile device could possibly pay.
2. *No Segmentation*: No specific technique is applied. Each file is transferred in its entirety.
3. *Segmentation*: The performance achieved by using technique File Segmentation for handling mobility.
4. *Segmentation + Prediction*: The performance of the combination of two techniques: File Segmentation and Mobility based Service Selection.

Fig. 8 shows that No Segmentation pays the highest cost up to 42 cents to download the files, i.e., 16 cents higher than Ideal, at the speed of 18 m/s. It is because the increase of the speed leads to a higher link breakage frequency and a higher amount of data which is downloaded to the real queue yet transferred to the mobile device. The Segmentation technique significantly improves the performance. Segmentation leads to the cost close to Ideal, only 1 cent higher to download 25 MB at the high speed of 18 m/s. Segmentation + Prediction even reduces the cost more, approximately equal to Ideal at that speed.

Fig. 9 shows that the end-to-end delay of No Segmentation is approximately twice higher than that of Segmentation. Segmentation + Prediction further reduces the end-to-end delay to 2 secs shorter than that of Segmentation. The higher end-to-end delay can be attributed to more link breakages due to the mobility. We plot the number of interruptions in Fig. 10, which confirms our conjecture. Last, Fig. 11 reports the relative overhead achieved by different techniques. This figure clearly shows that CrowdMAC produces negligible traffic overhead. It also reveals the effectiveness of File Segmentation.

Impact of the incentive mobility. We have shown that CrowdMAC works well even when mobile devices follow a random mobility model. Next we consider a more realistic mobility model, denoted as *Incentive Mobility Model*. In this model, the mobile devices, after selecting services, try to keep the links between themselves and the services stable, rather than moving fast and causing link breakage. This is done in order to ensure continuity for ongoing services that have already been charged. Mobile devices that are not transferring files follow the Random Waypoint model.

We plot the system performance under the Incentive Mobility Model in Figs. 8–11, labelled by *Incentive*. CrowdMAC achieves much better performance under this realistic mobility model. In particular, Incentive: (i) achieves the same minimum cost as Ideal, (ii) more than 10 secs shorter end-to-end delay than Segmentation + Prediction, and (iii) never suffers from interruptions.

7 Related Work and Concluding Remarks

Modern mobile phones have been extended to incorporate multiple networking interfaces beyond traditional cellular and WLAN capabilities; this enables them to form opportunistic networks using technologies such as WiFi Direct [2], WiFi Tether [1] and Bluetooth. Opportunistic networks complement infrastructure networks with new capabilities to support throughput enhancement [13, 21] and

peer-to-peer cellular traffic offloading [14,16,19]. Efforts have explored techniques to enhance throughput for mobile devices that suffer from low cellular data rates by selecting proxies that are in fact mobile devices with high cellular link rates [13,21]. Recent research has also proposed solutions [14,16,19] for mobile devices to cooperatively disseminate data. For example, in [14,19], nearby mobile devices cooperate to stream live videos; in these schemes, selected mobile devices are scheduled to receive part of videos over cellular networks and relay the received data to other mobile phones over WiFi peer-to-peer networks.

To our best knowledge, CrowdMAC is the first effort to developing a crowdsourcing system to motivate mobile users for sharing Internet access. We showed how to design and implement a middleware framework that incorporates a Lyapunov based admission control algorithm for mobile APs to serve multiple requesting mobile devices optimally and stably, strategies for mobile devices to select mobile APs appropriately, and techniques for handling device and AP mobility. CrowdMAC has been implemented and evaluated on a testbed of diverse Android phones with varying capabilities indicating feasibility of the approach. Our experimental and simulation results show that CrowdMAC: (i) effectively exercises the trade-off between revenue and transfer delay, (ii) adequately satisfies user-specified (delay) quality level, and (iii) properly adapts to device mobility and achieves performance very close to the ideal case (upper bound).

Future Work: The Lyapunov based approach in this paper lends itself well to a practical implementation. One can also consider an alternate game theoretic formulation that models the interaction between mobile devices and mobile APs as a multiparty game. The ability to embed the game theoretic approach into a real system in a dynamic setting is challenging – this is our future work.

To realize a broader and wide-scale deployment of our crowdsourcing scheme, our future work will also address two key concerns. The first concern is that of security. Mechanisms are required to protect a users' file as it passes through arbitrary (and potentially untrusted) nodes, networks and middle-boxes; this includes protection from DOS attacks and from malicious services. How to leverage cryptographic techniques to provide such end-to-end security is our current aim. The second concern is the need for a tighter integration of the proposed scheme into the ISP/provider ecosystem so as to mesh with the business objectives of telcos and service providers. Creating localized networks to support data exchange is becoming a commodity technology. We believe that the ability to crowdsource/share local wireless access can offer a new perspective that may change the scale and scope of mobile data delivery just as VoIP has changed the landscape of telephony today.

References

1. Android WiFi tether. `http://code.google.com/p/android-wifi-tether/` (2009)
2. Wi-Fi certified Wi-Fi Direct: Personal, portable Wi-Fi that goes with you anywhere, any time. `http://www.wi-fi.org/Wi-Fi_Direct.php` (2010)
3. National communications commission. `http://www.ncc.gov.tw/` (2011)

4. Traffic and market data report. http://hugin.info/1061/R/1561267/483187.pdf (2011)
5. Why Verizon dropped its unlimited data plan (and what you can do about it). http://moneyland.time.com/2011/06/23/why-verizon-dropped-its-unlimited-data-plan/ (2011)
6. 46% of American adults are smartphone owners. http://pewinternet.org/~/media//Files/Reports/2012/Smartphone%20ownership%202012.pdf (2012)
7. My virtual neighbor. http://www.myvirtualneighbor.com/ (2012)
8. Nearly half of AT&T subscribers would pay less by switching to a metered plan. http://tinyurl.com/86kcgyj (2012)
9. Qualnet network simulator. http://code.google.com/p/android-wifi-tether/ (2012)
10. T-Mobile data plan. http://www.t-mobile.com/shop/plans/mobile-broadband-plans.aspx (2012)
11. Balasubramanian, A., Mahajan, R., Venkataramani, A.: Augmenting mobile 3G using WiFi. In: Proc. of MobiSys. pp. 209–222. San Francisco, USA (2010)
12. Balasubramanian, N., Balasubramanian, A., Venkataramani, A.: Energy consumption in mobile phones: A measurement study and implications for network applications. In: Proc. of IMC. pp. 280–293. Chicago, IL (2009)
13. Bhatia, R., Li, L., Luo, H., Ramjee, R.: ICAM: Integrated cellular and ad hoc multicast. IEEE Transactions on Mobile Computing 5(8), 1004–1015 (2006)
14. Do, N., Hsu, C., Jatinder, S., Venkatasubramanian, N.: Massive live video distribution over hybrid cellular and ad hoc networks. In: Proc. of IEEE WoWMoM. pp. 1–9. Lucia, Italy (2011)
15. Franklin, M., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: CrowdDB: Answering queries with crowdsourcing. In: Proc. of ACM SIGMOD. pp. 61–72. Athens, Greece (2011)
16. Han, B., Hui, P., Kumar, V., Marathe, M., Shao, J., Srinivasan, A.: Mobile data offloading through opportunistic communications and social participation. IEEE/ACM Transactions on Mobile Computing 11(5), 821–834 (2012)
17. Huang, L., Neely, M.: The optimality of two prices: Maximizing revenue in a stochastic communication system. IEEE/ACM Transactions on Networking 18(2), 406–419 (2010)
18. Joe-Wong, C., Ha, S., Chiang, M.: Time-dependent broadband pricing: Feasibility and benefits. In: Proc. of IEEE ICDCS. pp. 288–298. Minneapolis, MN (2011)
19. Keller, L., Le, A., Cici, B., Seferoglu, H., Fragouli, C., Markopoulou, A.: MicroCast: cooperative video streaming on smartphones. In: Proc. of ACM MobiSys. Lake District, United Kingdom (2012)
20. Lotfinezhad, M., Liang, B., Sousa, E.: Optimal control of constrained cognitive radio networks with dynamic population size. In: Proc. of IEEE INFOCOM. pp. 1–9. San Diego, CA (2010)
21. Luo, H., Meng, X., Ramjee, R., Sinha, P., Li, L.: The design and evaluation of unified cellular and ad-hoc networks. IEEE Transactions on Mobile Computing 6(9), 1060–1074 (2007)
22. Qin, M., Zimmermann, R.: Improving mobile ad-hoc streaming performance through adaptive layer selection with scalable video coding. In: Proc. of ACM Multimedia. pp. 717–726. Augsburg, Germany (2007)
23. Yeh, S., Talwar, S., Wu, G., Himayat, N., Johnsson, K.: Capacity and coverage enhancement in heterogeneous networks. IEEE Wireless Communications 18(3), 32–38 (2011)