COMBINE: Leveraging the Power of Wireless Peers through Collaborative Downloading

Ganesh Ananthanarayanan
Microsoft Research
ganeshan@microsoft.com

Venkata N. Padmanabhan
Microsoft Research
padmanab@microsoft.com

Lenin Ravindranath
Microsoft Research
leninr@microsoft.com

Chandramohan A. Thekkath
Microsoft Research
thekkath@microsoft.com

ABSTRACT

Mobile devices are increasingly equipped with multiple network interfaces: Wireless Local Area Network (WLAN) interfaces for local connectivity and Wireless Wide Area Network (WWAN) interfaces for wide-area connectivity. The WWAN typically provides much wider coverage but much lower speeds than the WLAN. To address this dichotomy, we present COMBINE, a system for collaborative downloading wherein devices that are within WLAN range pool together their WWAN links, significantly increasing the effective speed available to them.

COMBINE makes a number of novel contributions over prior work in this area, including: (a) a framework of incentives for collaboration that addresses several practical issues including the unification of monetary and energy costs, and on-the-fly estimation of the energy cost of communication in a system in operation; (b) a protocol for collaborative group formation and workload distribution that is energy efficient and adaptive to fluctuations in network conditions; and (c) an application-level striping procedure that eases deployment by avoiding the need for special-purpose proxies in the infrastructure. We present experimental results based on the prototype we have implemented that show encouraging speeds-ups with COMBINE.

Categories and Subject Descriptors:
C.2 [Computer-Communication Networks]: Distributed Systems

General Terms:
Measurement, Performance, Design, Security.

Keywords:
Wireless networks, bandwidth aggregation, incentives, peer-to-peer, multi-homed.

1. INTRODUCTION

Mobile devices such as laptops, smartphones, and PDAs are increasingly being equipped with multiple wireless network interfaces. These include one or more wireless LAN interfaces (e.g., 802.11, Bluetooth) and wireless WAN interfaces (e.g., GPRS, UMTS). This allows devices to have a choice of radios to use *separately*. Since the WLAN offers much higher speeds (a few to tens of Mbps) than the WWAN (tens to hundreds of Kbps), the conventional wisdom is to use the WLAN interface when in range of a WLAN (e.g., at a WiFi hotspot), but settle for the much lower speeds offered by the WWAN interface at other times. Despite the proliferation of WiFi hotspots, their coverage is still quite limited (e.g., outside the city center or on trains and buses). Furthermore, even in locations with WiFi coverage, policy issues may impede a user's ability to take advantage of it (e.g., the user may not be a subscriber of the hotspot service provider and so may have to set up yet another billing relationship). Such a mismatch is less likely to arise on the large footprint WWAN because the user's own provider is often within reach.

In this paper, we present *COMBINE*, a system for collaborative downloading that uses both the WLAN and the WWAN in *combination*, in an attempt to bridge the range-speed dichotomy. Nodes in close vicinity use the high-speed WLAN to discover each other, form a collaboration group, and stripe traffic across their WWAN links, increasing the effective WAN download speed available to any one node. All of these steps happen automatically, with user involvement limited to setting policy.

While pooling together WWAN links has been considered in prior work (see Section 2), doing so among a set of collaborating but uncoordinated nodes raises several challenges that we believe have not been adequately addressed. *COMBINE* makes a number of novel contributions in this regard:

1. **Cost modeling:** By contributing its WWAN bandwidth for the benefit of its peers, a node typically incurs both a monetary cost (e.g., WWAN charges) and an energy cost. It is important that these costs be accounted for as peers provide/seek help to/from each other. *COMBINE* includes techniques for infering the energy cost on-the-fly and unifying it with the monetary cost.

2. **Accounting:** The costs computed above form the ba-

sis of a market wherein nodes buy and sell WWAN bandwidth resources. *COMBINE* includes a lightweight and secure accounting scheme to keep track of the credits earned or debits incured as nodes buy and sell bandwidth.

3. **Collaboration group formation:** Assuming two or more nodes wish to collaborate, *COMBINE* includes an energy-efficient protocol for them to rendezvous with each other, exchange their bids, and form a collaboration group.

4. **Striping protocol:** Once a collaboration group has been formed, *COMBINE* uses an adaptive workload distribution algorithm to farm out work across the participants in the collaboration group. Also, *COMBINE* uses HTTP byte-range requests to effect the striping, which trades off generality for improved deployability compared to prior work that has focused on striping at the network layer and consequently required special-purpose proxies in the infrastructure (e.g., [11, 16]).

We believe that these contributions of COMBINE complement prior work. For example, although *COMBINE* is based on HTTP-level striping, there are many aspects of the system (e.g., cost modeling and workload distribution) that could be applied in a setting where striping is done at the network layer instead.

We have prototyped *COMBINE* on Windows XP and evaluated its performance on laptop-class devices equipped with 802.11b WLAN NICs and GPRS WWAN NICs. We have encouraging results indicating near-linear speedups for group sizes of up to 5 nodes that were tested.

The remainder of the paper is organized as follows. In Section 2, we survey related work and discuss how *COMBINE* relates to it. We present an overview of *COMBINE* in Section 3 and then turn to a detailed discussion of the components of the system in the sections that follow. We present our cost model in Section 4 and accounting scheme in Section 5. We discuss the collaboration protocol in Section 6 and consider security issues in Section 8.1. We present an experimental evaluation of our *COMBINE* prototype in Section 7. We conclude in Section 9.

2. RELATED WORK

Considerable research has gone into improving download performance. One stream of research focuses on throughput enhancement by a more efficient and aggresive use of a single network interface. Download performance can be accelerated by opening multiple connections from the client to various replicas or mirrors of websites [15, 19].

The evolution of devices with multiple network interfaces has opened up a whole new area of research where researchers tried to overcome the limitations in the bandwidths of a single WWAN interface using bandwidth aggregation. The work broadly follows two paths: one where the bandwidths of all the WWAN links attached to a single device are aggregated and the second where multiple devices collaborate to aggregate their individual bandwidths.

Horde [16], pTCP [9], and Snoeren et. al. [22] focus on mechanisms for inverse multiplexing (i.e., striping packets) across multiple WWAN links to avoid problems such as TCP packet reordering. Since these systems assume that the multiple WWAN links are attached to the same device, the is-

sues of how to accomplish local communication and provide incentives for cooperation are not considered. MAR [18] makes use of the multiplicity of the wireless networks available by dynamically instantiating new channels based on traffic demand, aggregating the bandwidth and dynamically shifting load from poor quality to better quality channels.

Other systems have considered coordinating communications from multiple mobile computing devices. PRISM [11] consists of a network-layer inverse multiplexer (PRISM-IMUX) at the proxy and a congestion-control mechanism (TCP-PRISM) at the sender side. Sharma et. al. [20] propose an application-aware and channel-adaptive architecture for flow assignment over multiple links [20]. MOPED deals with group mobility where a user's set of devices collaborate to appear as a single entity in the Internet [5].

But all these systems have side-stepped the issue of incentives and accounting, and simplified the problem of local communication by either assuming that the same device owns all the devices (e.g., MOPED [5]), the presence of a separate aggregation router (e.g., MAR [18]), or that the improved performance alone is sufficient incentive for cooperation (e.g., Handheld Routers [20], PRISM [11]). The mere promise of improved performance may not be a sufficient incentive for cooperation because of the ephemeral association between mobile nodes (see Section 5). While UCAN [13] considers secure crediting, it uses the WLAN to increase the reach of the WWAN rather than for bandwidth aggregation, and more importantly, it ignores a key resource, viz. energy.

The notion of providing incentives to nodes in an ad hoc network for a collaborative activity has been presented in the context of forwarding in ad hoc networks [4, 14]. Forwarding in ad hoc networks, however, is somewhat different from the collaboration we consider here. In ad hoc networks, nodes rely on each other to communicate amongst themselves. In a mobile community, nodes rely on each other not for basic connectivity, but for performance improvements.

Peer selection in a free market scenario has been done for optimizing speed in the context of streaming video [3, 8]. But we are not aware of any prior work that optimizes throughput in the context of collaborative communities for downloading data.

We contrast our work with the previously mentioned solutions on two fronts. One, while all of the above systems operate at the transport or network layer, our solution operates at the application level. While working at the lower layers offers the advantage of application independence, it also requires infrastructure proxies to split and splice flows, which may be a deployment hurdle. Also, we believe that application level changes are simpler to implement compared to any other lower-level changes. Two, our system has a practical energy-cognizant incentives scheme and a secure accounting mechanism that enables the collaborators to be assured of easy and secure redemption of credit and hence encourages collaboration.

We believe that elements of previous work (e.g. TCP enhancements proposed in [11]) can be adapted in *COMBINE* for performance enhancements and vice versa. Notably, if devices participating in *COMBINE* have with multiple WWAN links attached to them, they could apply the bandwidth aggregation techniques proposed in Horde [16] at their local ends. Likewise, while *COMBINE* focuses on the web download application, elements of it like the group

formation protocol and the accounting mechanisms can be combined with a more general application.

3. OVERVIEW OF COMBINE

COMBINE enables a mobile device to utilize the WWAN links of devices in its vicinity to boost the effective WWAN speed available to it. The motivating application is large downloads, which would likely benefit the most from a bandwidth boost. Given the growing market for mobile music and video download (e.g., [2]), we believe that the speedup provided by collaborative downloading would significantly enhance user experience. In fact, we believe it would be in the interest of WWAN providers to encourage such sharing among users on their networks, for it would help improve user experience through software means, without the need for expensive hardware upgrades.

We consider a setting where a requester (termed *initiator*) seeks to utilize the WWAN links of one or more *collaborators*. A collaborator contributes its WWAN bandwidth only when it is not in local use. Even so, since the collaborator would incur a cost in contributing its unused WWAN bandwidth, there is the need for incentives to offset this cost. The cost comprises both the monetary tariff imposed by the WWAN provider and the opportunity cost of expending battery energy (i.e., expending energy on behalf of a peer might diminish the ability of a user to use the device for their own purpose).

In *COMBINE*, each collaborator estimates its cost of providing help and communicates it as its price to the initiator. (In general, the collaborator could, based on user policy, scale up or down its price relative to its cost to reflect how aggresive it wants to be as a seller.) In turn, the initiator compares the bids offered by multiple collaborators, picks the ones that are low enough, and proceeds to form a collaboration group.

Given device mobility and the consequent ephemeral nature of association between devices, a collaborator cannot count on the initiator being in its vicinity at a later time when the collaborator needs help. To address this issue, *COMBINE* includes an accounting mechanism, wherein the initiator issues signed IOUs to its collaborators, who then redeem these by contacting an accounting server at some later point in time.

In the process of exchanging bids and forming a collaboration group, a key challenge is in enabling the initiator to rendezvous with potential collaborators in its vicinity in an energy-efficient manner. Since this process happens occasionally and at unpredictable times, it unacceptable to have the potential collaborators be constantly listening for initiator requests. Instead, *COMBINE* uses a combination of a low-power radio (e.g., Bluetooth) and periodic wakeups to achieve energy efficiency, while trading off a little in group formation speed.

Once a collaboration group has been formed, the initiator uses an adaptive work-queue algorithm to distribute work across the collaborators. This algorithm adapts to dynamic variations in WWAN link speed across the collaborators and also to local traffic at the collaborators, which takes priority over *COMBINE* traffic.

COMBINE uses HTTP byte-range requests [6] to stripe traffic across multiple WWAN links. While this HTTP-level mechanism is not as general as prior link- or network-level striping schemes, it avoids the need for a special-purpose proxy in the infrastructure to split and splice flows. We believe this is a significant advantage from a deployment viewpoint.

4. MODELING COST

It is important that the cost of sharing bandwidth be modeled appropriately, so that the offered price is high enough to adequately compensate the collaborator but not so high as to be unattractive to the initiator.

There are two principal costs that a collaborator incurs in helping the initiator. The first is the cost of transferring data on the WWAN link for which the WWAN service provider extracts a fee. This fee depends on the tariff structure imposed by the service provider and may depend, in general, on factors such as the user's service plan and the time of day. Nonetheless, given a tariff structure, one can estimate the cost of WWAN usage for transferring a given amount of data, although some projection of anticipated future usage may be needed (say, based on past usage patterns) to account for tiered pricing. Also, if flat rate pricing is being used, there may not be an incremental cost to additional data transfer. Nevertheless, a user would want to amortize their monthly dues over the expected (or typical) monthly usage. For our purposes in this paper, we assume that the WWAN tariff is known and is a uniform rate per unit data.

The second cost is the opportunity cost of expending battery energy on behalf of a peer. Battery energy is a meager resource for mobile devices, so by expending it in this manner, a collaborator increases the risk of running out of battery energy for its own use.

We begin by presenting a simple framework for quantifying energy cost and unifying it with monetary cost. We then turn to estimating energy usage on-the-fly.

4.1 Unifying Monetary and Energy Costs

Quantifying energy cost and unifying it with monetary cost presents an interesting challenge. While both energy and money are valuable resources, they are quantified in different units that need to be reconciled. Furthermore, the opportunity cost of expending energy would depend not just on the amount of energy expended but also the likelihood that the expenditure would deprive the user the use of their device. After all, if the battery is likely to be recharged before it fully drains, the user would not really incur an opportunity cost. Finally, the opportunity cost would be a function of the utiliy of a mobile device for its user. It would be lower for a user who is idling compared to a user who needs to have use of their device at any cost.

In *COMBINE*, we model the opportunity cost as a function of the fraction of battery battery energy remaining (BR), which falls in the range $[0, 1]$. The smaller BR is, the more precarious the device's energy state is, and so the more valuable battery energy is likely to be to the user. In our current design, we use $1/BR$ as the opportunity cost, which is consistent with the inverse relationship between BR and the opportunity cost.

To unify the opportunity cost of expending energy with the monetary cost (MC) of performing a data transfer, we scale MC by the opportunity cost $1/BR$. Hence the total cost is $TC = MC/BR$. Since BR is dimensionless, TC is also expressed in monetary units, which is convenient from the viewpoint of accounting.

In the common case, a collaborator performs a WWAN

transfer on behalf of the initiator. So the monetary cost, MC, is a function of the tariff rate and the WWAN transfer size (e.g., simply a product of the two if the tariff rate per byte is uniform). However, it is possible that the collaborator has an up-to-date copy of the requested file in its cache, in which case a WWAN transfer is not needed. Nevertheless, since the collaborator had presumably incurred the cost of a WWAN transfer at some point in the past, we compute MC to be what the WWAN tariff *would* be for a transfer of the corresponding size. (In general, we could amortize this cost over multiple requests for the same cached file, but we do not do so in *COMBINE*.)

Finally, we accommodate variations in the utility of a mobile device for its user by scaling up or down TC based on user input. As elaborated in Section 8.2, the scaling factor, K_s, is set to 1 by default. A user who is very keen not to deplete their battery would set K_s to a large value whereas another user without a pressing need for their device (or who expects to be recharging it in the near future) may choose to set K_s to a low value, in an attempt to make their bids attractive and earn credits for future use (Section 5).

4.2 Estimating Battery Depletion

The procedure discussed in Section 4.1 to compute the total cost, TC, depends on the current level of battery remaining, BR, which can be queried directly from the OS. However, computing TC is this manner would only be appropriate for data transfers that are small enough that BR does not change significantly on account of the data transfer itself. While the data transfer chunk size used in the workload distribution scheme discussed in Section 6 satisfies this requirement, it is still desirable to be able to estimate the energy usage (in terms of battery depletion) for longer transfers. This would allow estimating how much more expensive a collaborator's bid is likely to become in the future, allowing the initiator to make a more informed choice when it is seeking collaborators for a sustained period and wants to minimize churn in its collaboration group.

Accordingly, during group formation, the initiator seeks not only the current bid from a collaborator but also information on how the bid is likely to change for every additional (large) unit of data transferred through the collaborator (say every MB of data). Knowing the size of the file to be downloaded (determined, say, using an HTTP HEAD request), the initiator can then evaluate the suitability for each collaborator for sustained participation.

There has been much work on measuring and characterizing the energy consumption of network activity in mobile devices. However, the focus has been on measuring this in controlled settings, possibly using special instruments to measure the current drawn by the NIC [23, 21]. While this may be accurate, it is not suitable for use in devices in the field, especially as their characteristics (e.g., the capacity of the battery to hold charge) change over time.

In contrast, *COMBINE* seeks to estimate the energy consumption of network activity based on observations of the device in operation. We use a simple linear model for the battery depletion: $BD = (time\_elapsed * BD_t) + (bytes\_sent\_or\_recd * BD_d)$. BD is the total battery depletion, expressed as a fraction of the full battery capacity. BD_t is the battery depletion per unit time and it reflects the cost of keeping the device, including the NIC, turned on. BD_d is the battery depletion per unit data sent or received (clubbing both

together simplifies the model and is also supported by measurement data showing that the costs of wireless transmission and reception are similar).

While the device is in operation, we sample the volume of network activity (i.e., the count of bytes sent or received on a NIC, as reported by `netstat`) and the battery remaining at various times. Using these samples, we apply multivariate linear regression to estimate BD_t and BD_d.

Since the energy characterics of the WLAN and the WWAN NICs are likely to be different, we need to estimate the battery depletion rate separately for each. However, to avoid complicating our model too much, we focus our sampling on periods of *COMBINE* activity, when both NICs are turned on. (After all, we are only interested in estimating energy usage during such periods.) So we introduce separate variables, BD_{d_WLAN} and BD_{d_WWAN} to represent the battery depletion per unit data. However, we retain a single BD_t that reflects the battery depletion per unit time on account of keeping the devices, including both NICs, turned on. Our model therefore becomes: $BD = (time\_elapsed * BD_t) + (bytes\_sent\_or\_recd_{WLAN} * BD_{d\_WLAN}) + (bytes\_sent\_or\_recd_{WWAN} * BD_{d\_WWAN})$.

5. ACCOUNTING

For an incentive scheme based on the cost model from Section 4 to work, we need an accounting mechanism to keep track of payments made by initiators to the collaborators they recruit.

5.1 Requirements

There are several properties that a practical accounting scheme should ideally have:

1. *Storing credits:* Given the mobility of nodes and the ephemeral association between them, a collaborator who provides help cannot count on the initiator being available to return the favor at a different time and place. So the accounting mechanism should be able to store credits for future use; a real-time tit-for-tat scheme as in BitTorrent would not suffice.

2. *Cheat-Proof:* The accounting scheme should be able to limit the damage caused by an initiator who fails to provide the promised compensation to a collaborator (e.g., by using counterfeit money for payment) or a collaborator who fails to provide the promised help.

3. *Privacy:* The accounting scheme should not leak information on the identities of the collaborator and the initiator to each other, or to a third party.

4. *Flexibility:* The accounting scheme should be flexible enough to accommodate real money as well as artificial forms of currency.

5. *Efficiency:* The computational and communication overhead of accounting should be low, ideally insignificant compared to the cost of the primary task at hand, viz., data transfer. This is especially important given the resource limitations of mobile devices.

5.2 Existing Alternatives

We first consider the possibility of using an existing digital payment system to enable an initiator and its collaborators to exchange payments.

Electronic funds transfer (EFT), which is widely supported by banks, is one possibility. However, EFT is cumbersome in practice (e.g., the collaborator would have to share its bank account information with the initiator). It also imposes a significant overhead on the collaborator (especially when fine-grained payments are made) in confirming the EFT with its bank online, to prevent cheating by the initiator.

An alternative would be to use digital cash (e.g., [1]), possibly adapted for efficient micropayments (e.g., [7]). The main advantage of such systems is their privacy, which is equivalent to that of paper cash. However, double spending is a serious risk, preventing which requires either the overhead of online communication with the bank that issued the cash or the provision of a secure hardware device (e.g., smartcard) at the clients to detect or prevent duplication.

5.3   Accounting in COMBINE

In *COMBINE*, we employ a credit-based scheme that offers protection against double spending without requiring (expensive) online communication with a server in the infrastructure. Our scheme leverages a central authority to issue public/private key pairs, and the corresponding certificates, to each user upon presentation of a proof of identity (e.g., a credit card). This happens at registration time, say when a user installs and activates the *COMBINE* software on their client. There is also a trusted accounting server, which keeps track of the credits/debits accrued by each user. The central authority that issues keys could be colocated with the accounting server.

When an initiator finds a collaborator's bid to be acceptable, it initiates the process of having the collaborator download content for it. As explained in Section 6, requests are issued to each collaborator in chunks. Together with the request for a chunk, the initiator includes a signed note of credit, termed an IOU, indicating the amount of payment being made for that chunk. The collaborator accumulates these IOUs during the collaboration session.

At a later time, perhaps when it has better connectivity (say on a high-speed WLAN), it transmits these to the accounting server for redemption. The accounting server credits the collaborator's account for the corresponding amounts, provided the IOUs have not already been redeemed. The IOUs include an expiration time, which helps limit the amount of history the accounting server needs to maintain to detect duplicate redemptions, while still giving the payee (i.e., the collaborator) sufficient latitude in communicating with the accounting server only when convenient. The asynchronous redemption mechanism for the IOUs has the advantage of being resilient to any temporary outages in the accounting infrastructure (e.g., it is not uncommon for an otherwise highly available banking website to be down for maintenance once in a while). Also, the lack of dependence on an online accounting infrastructure makes it possible to achieve a quick and initial small-scale deployment of our system (say, with a homegrown accounting system).

Although the IOUs are signed by the initiator, the collaborator needs the assurance that these are original, not duplicates that have already been used as payment. So at the start of a collaboration session, the collaborator conveys a nonce of its choosing to the initiator and asks that this be included (and signed) in all IOUs issued to it during the session.

An IOU does *not* identify the payee, i.e., the collaborator, for privacy reasons, as discussed in Section 5.4 below. So to prevent both an eavesdropper from stealing and redeeming the IOU (thereby denying the rightful payee the opportunity to redeem it), and the initiator who just granted an IOU from redeeming the IOU itself before the collaborator, the nonce supplied by the collaborator is augmented to include a one-way hash $h(x)$ where x is known only to the collaborator. For an IOU to be redeemed, the accounting infrastructure insists that it be accompanied by an x that matches the hash in the IOU.

To summarize, an IOU is of the form: $IOU = \{key_{pub}, amount, h(x), seq, exp, sign_{k_{priv}}\}$, and includes the payer's public key (key_{pub}), the amount of credit ($amount$), the nonce with a one-way hash supplied by the payee ($h(x)$), a sequence number that is incremented for each additional IOU issued with the same nonce (seq), the expiration date of the nonce (exp), and the signature generated with the payer's private key ($sign_{k_{priv}}$).

5.4   Refinements

The accounting scheme presented above satisfies many of the requirements noted in Section 5.1. The IOU is a credit note that remains valid beyond the ephemeral association between an initiator and a collaborator. Its construction, and the assumed existence of a PKI, give the payee the assurance that it is valid, without having to communicate with the accounting server online. (We defer to Section 8.1, discussion of the converse problem, viz., protecting the initiator from a collaborator who fails to perform the service for which it has been paid.) Also, IOUs are in no way limited to monetary credits; the same mechanism could be used to exchange and accrue arbitrary forms of credits (e.g., energy credits). Note that the mention of a credit card in Section 5.3 is only as a form of identity and does not necessarily imply monetary payments.

Nevertheless, there are a couple of issues that call for refining the basic accounting scheme. While we present the ideas here, these refinements have not been incorporated into our implementation as of this writing.

One issue is that a payee may accumulate a large number of IOUs from a payer during a long collaboration session. To avoid having to redeem these individually, the payee can, at any point in the collaboration session, request the payer to issue a new, aggregate IOU to replace a large number previously issued IOUs. The new IOU is made *conditional* on none of these prior IOUs having been redeemed. Rather than list each individual prior IOU, the new IOU just includes the nonce and sequence number range of the IOUs that it is conditioned on.

Another issue is that, compared to digital cash schemes, we sacrifice privacy. The payer does not remain anonymous since it needs to sign the IOUs and present the public key certificate issued by the central authority. While the certificate by itself does not permit the payee to identify the payer (e.g., learn the latter's name), it does give the payee the opportunity to track the payer across multiple collaboration sessions, even if spread over space and time, which is clearly undesirable.[1]

---

[1]We assume that steps have been taken to prevent tracking using more basic information such as MAC addresses of wireless NICs.

Furthermore, when the payee redeems an IOU, the accounting service can deduce that the payer (who is identified in the IOU) and the payee (who is redeeming the IOU) were likely in each other's vicinity at some point. The ability of the accounting service to spy on users in this manner is again undesirable. [2]

To make tracking by payees more difficult, we issue each user with multiple, but a fixed number, of public/private key pairs and certificates. The user can then pick at random which key pair to use when signing IOUs at various times, thereby hiding its tracks to an extent.

To prevent the accounting service from spying on user associations, we use a technique inspired by random forwarding chains used in anonymous communication systems (e.g., [17]). Consider a payer A who issues an IOU, IOU_A, to a payee B. Rather than redeem the IOU by itself, B forwards it on to a randomly chosen peer, C, who then redeems it at some point in the future. In return for the IOU_A forwarded to it, C returns to B a new IOU, IOU_C signed by C for the same amount as IOU_A. So effectively the transaction is neutral for both B and C, as it should be.

As a result, the accounting server will not be in a position to establish any association between A and B. While it might deduce an association between B and C (or A and C), there is in fact no privacy-sensitive association between B and C. Indeed, C is just happens to be a *COMBINE* user whom B picked at random and communicated with over the wide-area network at some later point in time. This is in contrast to the privacy-sensitive association between the original payer (A) and the original payee (B), viz., the likelihood that they were both in the same location and engaged in a collaborative download session.

6. PROTOCOL DESIGN

We now turn to the mechanics of forming a collaboration group and performing a collaborative download. We discuss three components: a protocol for mobile devices to form groups, a scheme to distribute work amongst the group, and a mechanism for low-level data transport and connection management to fetch data from servers, which are oblivious to the collaboration.

6.1 Group Formation

Group formation is the process by which the initiator identifies a set of collaborators. This is clearly a critical first step in doing collaborative downloads, but to the best of our knowledge, prior work has largely ignored it.

While the local connectivity offered by the WLAN would, in principle, make it a suitable means for a mobile device to rendezvous with other devices in its vicinity, energy considerations complicate matters. Figure 1 shows the high rate of energy depletion on a typical mobile phone with its 802.11 interface always turned on. This high energy depletion motivates mobile devices to switch off their 802.11 cards or put them in a power saving mode to conserve battery, particularly in environments where there are no Wi-Fi access points. Since this is our environment of interest, our protocol cannot depend on WLAN cards being switched on in

---

[2]We believe that it is *not* a serious problem, in itself, that the accounting service learns the payer's identity from an IOU, for that only reveals that the payer performed collaborative downloading. It does not, in itself, reveal the payer's location or which user(s) the payer was in the vicinity of.

anticipation of group formation requests, which complicates the rendezvous process. Indeed, groups in *COMBINE* are created opportunistically by the initiator, without advance notice to potential collaborators.
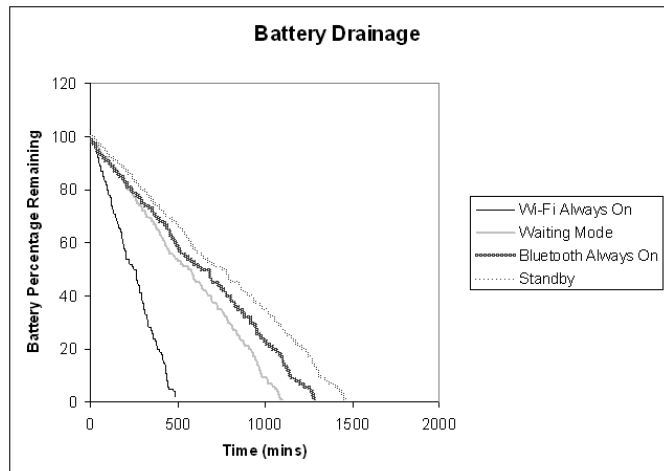
Figure 1: Power depleted over time by an i-mate PocketPC with Bluetooth and 802.11 networks in varying power states. Standby refers to the condition where both networks are completely shutdown but the device is otherwise powered on. Wi-Fi Always On refers to the condition when the 802.11 alone is powered on. Bluetooth refers to the condition when the Bluetooth network alone is powered on. Waiting Mode refers to the condition when the 802.11 network interface alone is powered up once every 500 milliseconds and stays on for 50 milliseconds.

Furthermore, group formation must work correctly when mobile devices move in and out of range: It must tolerate non-responsive initiators and collaborators, as well as multiple simultaneous initiators.

To enable devices to rendezvous yet conserve energy, *COMBINE* devices, by default, keep their WLAN cards in a low-energy mode called the *waiting* mode. In this mode, devices periodically wake up their WLAN cards and broadcast an *I-am-Alive* message and await a response for a certain period of time before shutting off their WLAN card.

The initiator keeps its WLAN card always switched on and listening for *I-am-Alive* messages. It collects *I-am-Alive* packets for a total time of T_G (set to 1 second in the experiments reported in Section 7), to form a group as described below. Our waiting mode is inspired by the standard Power Saving Mode (PSM) in IEEE 802.11 NICs [10]. However, unlike PSM, the waiting mode does not depend on any interaction with wireless access points. It also offers the flexibility of less frequent wakeups than standard PSM, thereby helping conserve energy.

Each collaborator computes the cost of using its resources (see Section 4) and sends a bid in the *I-am-Alive* message to the initiator. The bid contains the price of using the collaborator and an estimate of the WWAN download speed it is able to offer. The initiator first determines the set of collaborators that has an acceptable cost and then distributes work to this group in a way that optimizes the data transfer.

Group Formation Algorithm

The detailed group formation protocol for 802.11 is as follows:

1. Each node i periodically wakes up its WLAN card and broadcasts an *I-am-Alive* message with the following information and waits for time T_A.

    (a) TC_i: The cost to the initiator for downloading one unit of data using this node. Section 4 describes how each node calculates this cost.

    (b) B_i: The WWAN speed that the node expects to be able to offer to the initiator. Note that this is only an estimate of the WWAN speed and could be inaccurate. We describe in a later section how we deal with these inaccuracies.

2. On receiving an *I-am-Alive* message, the initiator responds with a *CCHECK* message containing the following information.

    (a) The URL of the file it needs to download.

    (b) The time $(T_G - T_C)$ after which it will reply to the node with a collaboration acknowledgement. T_C is the amount of time currently elapsed since the start of the group formation process. Depending on the length of $T_G - T_C$, the node can decide to conserve energy by turning off its WLAN NIC until just before the appointed time.

3. On receiving a *CCHECK* message, the device checks its local cache for the URL mentioned in the message. If it has it in its cache and and is up-to-date (as determined by an "if-modified-since" HTTP request [6]), it unicasts a reply to the initiator informing it of the availability of the file. Otherwise, the node takes no action, but keeps its WLAN card on for the specified time period awaiting a message from the initiator.

4. After time T_G, the initiator evaluates all the *I-am-Alive* messages and selects the list of collaborators using the method described below.

5. The initiator sends out a *CACK* message to all the selected collaborators informing them of the fact that they have been enlisted in the collaboration group.

If the initiator does not receive any *I-am-Alive* message within time T_G, it resets its timer and starts over. This happens a specified number of times after which the initiator aborts its group formation attempt. If the nodes do not receive any *CACK* message within the time specified in the *CCHECK* message, they switch back to waiting mode. At the end of the group formation mechanism, the initiator has the list of all the nodes that are selected for collaboration.

A collaborator may receive *CCHECK* and subsequent *CACK* messages from distinct initiators. The protocol does not prohibit collaborators from responding to these messages, but in our current implementation, collaborators respond only to the first initiator.

While we have focused on using an 802.11 WLAN for group formation, mobile devices that have an 802.11 interface often also have a Bluetooth interface. Although Bluetooth is unlikely to be suitable for high bandwidth data transfer, it has one virtue that favors its use for group formation. The energy drain of Bluetooth is much lower than that of 802.11 as is evident from Figure 1. Thus, unlike 802.11, it may be reasonable for the Bluetooth interface to be turned on all the time, even if no groups are formed for extended periods. However, the group formation algorithm would need to be adapted given the smaller range of Bluetooth compared to 802.11 and the point-to-point rather than broadcast nature of Bluetooth connectivity. We plan to look into this in future work.

Group Selection Criteria

Notice that the initiator must choose a set of collaborators in Step 5 above. It does this based on one of two algorithms described below. In the following discussion we assume that the user wants to download a file of size F and he/she is willing to incur a cost of C to do so. Also recall that each *I-am-Alive* serves as a bid from a potential collaborator and contains a values for TC and B, i.e., the cost to the initator and the bandwidth it will get by using that collaborator.

#1: Threshold-based group selection: Our first algorithm is based on a simple thresholding scheme. Given F and C, the initiator calculates the value $\widehat{TC} = C/F$, the maximum per byte cost threshold. All nodes whose bids contain a TC value less than $\widehat{TC}$ are selected and sorted in the descending order of their WWAN speeds (also contained in their bids), and the first n nodes are selected as collaborators (with n chosen suitably to limit the size of the collaboration group). This conservative choice of collaborators guarantees that the overall cost will not exceed C regardless of how the workload is distributed across the collaborators.

#2: Opportunistic group selection: Our second scheme is less conservative. Rather than restrict ourselves only to nodes whose per byte cost is under the C/F threshold, we are open to recruiting some collaborators that are more expensive. Nevertheless, through a suitable workload distribution, we ensure that the overall cost still does not exceed C. As we show in Section 7, this less conservative approach could yield significant performance gains.

This scheme is based on formulating group selection as an optimization problem. The goal of the optimization is to minimize the total time taken to download F bytes of data subject to the cost constraint C. If each collaborator i, working in parallel, downloads x_i bytes with bandwidth B_i, the total time taken to download the file is the maximum of $\{x_1/B_1, x_2/B_2, ..., x_N/B_N\}$, where N is the number of distinct *I-am-Alive* packets received by the initiator. We determine optimum values of $x_i, i = 1..N$ so that we minimize the total time subject to the constraints:

$$\sum_{i=1}^{N} TC_i x_i \le C \qquad (1)$$

$$\sum_{i=1}^{N} x_i = F \qquad (2)$$

$$x_i \ge 0, i = 1..N \qquad (3)$$

The nodes with non-zero x_i values are selected for the collaboration group. The remaining drop out of the protocol.

6.2 Work Distribution

We have implemented two work distribution strategies: a

work-queue algorithm and an opportunistic algorithm. As will be clear below, these two algorithms are intended to be used in conjunction with threshold-based group selection and opportunistic group selection, respectively.

#1: Work-Queue Algorithm

In this algorithm, the initiator gets the total size of the file to be downloaded and forms a work-queue with fixed equal-sized byte ranges of the file. The total file size is obtained by querying the server, say with an HTTP HEAD request. Collaborators query the initiator and pick up the next available item from the work-queue, download the amount of data as specified in its work-item and return it to the initiator. Each collaborator picks up more work when it is done with its current work item, and keeps working until the queue is empty.

Thus the work performed by each collaborator is proportional to its WWAN speed, without the initiator having to allocate work explicitly. Also, since the initiator does not allocate work, the work-queue algorithm is only suited for use with the conservative threshold-based group selection. Otherwise, we cannot guarantee that the overall cost constraint will be satisfied.

The amount of data in a work-item, called the *chunk size*, needs to be picked appropriately. Too large a chunk size would result in the algorithm being less agile to changing speeds and also result in a high amount of salvaging in case of any of the collaborators going down. On the other hand, too low a chunk size will place considerable overheads for every WWAN download and adversely affect the throughput. Our experiments, detailed in Section 7, indicate that a chunk size of 200 KB is appropriate for HTTP downloads.

#2: Opportunistic algorithm

The opportunistic work distribution algorithm follows directly from the opportunistic group selection algorithm discussed above. It uses the same optimization framework with one key difference — rather than solve the optimization problem and compute the work allocation (viz. the x_i values) just once for the entire file, it is solved repeatedly over smaller *partitions* of the file. The initiator divides the file of size F into fixed-size partitions of size p bytes each and apportions to each partition a cost budget of $(C/F)p$.

Furthermore, rather than persist with the bandwidth estimates, B_i, provided by the collaborators initially, we compute an exponentially weighted moving average of the actual bandwidth observed for each collaborator during the course of the download. These observed bandwidth values (OB_i) are used when solving the optimization problem for later partitions.

Unlike the work-queue algorithm, the initiator explicitly allocates work to the collaborators to ensure that the cost constraint is satisfied despite the presence of expensive collaborators. Computing work allocation on partitions rather than the entire file makes the work distribution more adaptive to dynamic fluctuations in bandwidth.

Failure Handling and Adaptation

We ensure that *COMBINE* adapts well to changing conditions, both to provide good performance and to avoid stomping on non-*COMBINE* traffic.

As noted in Section 3, *COMBINE* only seeks to utilize *unused* bandwidth at the collaborator nodes. Therefore, it needs to detect when a previously idle WWAN link has become busy, and then back off. We have implemented a simple busyness detector that compares the volume of bytes sent/received on the WWAN interface with the volume of traffic sent/received by *COMBINE*. If the difference is large, it implies that there is a significant amount of non-*COMBINE* traffic. Therefore, *COMBINE* stops using that WWAN link (beyond the current chunk, if any, in progress) until the link falls idle again.

A collaborator node could unexpectedly slow down or fail, either because of the above backoff procedure or because of other network dynamics (e.g., it may move out of the range of the initiator). *COMBINE*'s work distribution procedure needs to be agile enough to adapt to these.

The work-queue algorithm accommodates such fluctuations and failures by design. The slowdown or failure of a collaborator would mean that the affected node would automatically slow down or stop the process of picking up additional work items. So other collaborators would automatically pick up a larger share of the workload.

In the opportunistic work distribution algorithm, the initiator estimates how long it should take for a collaborator to complete the work allocated to it. If a much longer time has elapsed but the collaborator still has not finished its work, the initiator asks it for its status (i.e., what fraction of the download has completed). The initiator then allocates the remaining work on the pending chunk to another collaborator, assuming there is one that is expected to complete the remaining download faster.

6.3 Data Transfer and Connection Management

Apart from forming a collaborative community and distributing work to its members, we must still arrange for a way for each collaborator to fetch its share of data from the site providing content.

We have chosen to implement the data transfer protocol at the HTTP level; specifically we exploit the byte-range request defined in HTTP/1.1 [6] and that is supported by most web server implementations. Using HTTP has the benefit that it does not require specialized proxies in the infrastructure to split and splice flows (e.g., the inverse multiplexers described in prior work). This eases deployment.

However, an issue that arises with HTTP-level striping is that the server providing the content may require session establishment and the exchange of session-level information before it will serve up content. Maintaining session semantics can be problematic. Collaborators must now either share a single session with the initiator, or collaborators have to initiate multiple sessions simultaneously. This may run into problems because, for instance, the server may disallow session information (e.g., an HTTP cookie) to be used from multiple (collaborator) IP addresses concurrently. We can get around this difficulty by having the requests through all collaborator routed through a common web proxy in the infrastructure, which would then present a single IP address to the server. Note that web proxies are "standard" infrastructure and in not *COMBINE*-specific, unlike the specialized network-layer proxies required in prior work (e.g., [11]).

Another issue is that the initiator may have to share session-level secrets (e.g., HTTP cookies) with the collaborators, which it may be unwilling to do. However, this problem can be alleviated by limiting cookie lifetime.

Our current implementation of *COMBINE* does not address these session-level issues. However, this is the subject of ongoing work.

7. EXPERIMENTAL EVALUATION

The COMBINE system has been implemented on Intel laptops. The laptops are equipped with an Intel Pentium 4 processor with a GB of RAM running Windows XP SP2. The laptops are equipped with both IEEE 802.11 cards and CDMA-based 3G data connections. The laptops use the Sierra Atlantic PCMCIA wireless cards as their WWAN link. All the laptops are equipped with D-Link DWL-AG132 USB 2.0 dongles. This is an 802.11 a/b/g radio based on the Atheros chipset.

We have also implemented the system with basic functionalities on Pocket PC (PPC) devices (195 MHz OMAP850 processor with 64 MB RAM). For diversity, we used two i-mate and one HP iPAQ PPCs . The PPCs used the GPRS connections for WWAN connectivity and Wi-Fi for local connectivity.

While the system implementation on laptops is complete, the opportunistic work-distribution algorithm and busyness detection modules have not been implemented in the PPCs as of this writing. We also report microbenchmarks to measure the impact of the compute-intensive tasks (e.g., digital signature operations) on the PPCs.

7.1 Group Formation

We overload the SSID field of the Wi-Fi beacons for group formation. Potential collaborators overload the SSID field of the Wi-Fi beacons (*I-am-Alive* messages) with information about their cost (TC) and their WWAN speed. The initiator continuously collects unique *I-am-Alive* beacons for a specific period and after the group selection, sends out Wi-Fi beacons (*CACK*) with the SSID field containing the list of selected collaborators. The selected collaborators detect this beacon and join the IEEE 802.11 ad hoc network with this SSID.

In our implementation, the initiator listens to *I-am-Alive* beacons for 4 seconds. The total time taken for the entire group formation process including the formation of the ad hoc network averages under 8 seconds for both the laptop as well as the PPC implementations.

7.2 Optimal Chunk Size

In the work-queue algorithm, the collaborators download chunks of data as specified in the work-items. Since there is a significant overhead associated with initiating and closing an HTTP connection, it is important to arrive at an optimal value of the chunk size. While very high chunk sizes affect the agility and failure-handling capability of the algorithm, very low chunk sizes places a significant overhead and hence adversely affects the throughput.

We evaluated the HTTP throughput for varying data sizes by measuring two parameters, the time taken for the download excluding the initial connection establishment phase and the total time taken. While the former gives an estimate of just the data transfer rate, the latter corresponds to the effective throughput. We observed that the difference in these two values tended to zero as the amount of data downloaded increases. Figure 2 plots the reduction in throughput due to the initial connection establishment for HTTP downloads. Notably, the slope of the graph sharply

decreases as the download size just exceeds 100 KB and the loss in throughput is very small (less than 10 kbps) if the download size is over 200 KB. Hence we assumed 200 KB to be a reasonably optimal chunk size for any collaborator to download via HTTP.
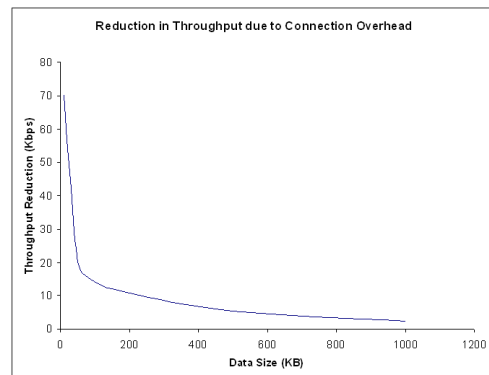
Figure 2: Reduction in HTTP Throughput due to the Connection Overhead for Varying Download Sizes

7.3 HTTP Throughput and Speed-up

We measured the throughput and speed-up of COMBINE by downloading an 8 MB file via HTTP from a data source on the Internet. We used the work-queue algorithm to distribute the workload among the collaborators and the HTTP byte-range request format to request for parts of a file from the server. The baseline for our the measured HTTP throughput is the initiator's individual throughput.

True to intuition, the speed-up values increased proportionately with the number of nodes.[3] Figure 3 shows the speed-up and Table 1 shows the throughput values as the size of the community is increased, for both the laptop as well as the PPC implementations. The results indicate similar speed-up numbers for both the PPC as well as the laptop implementations. Note that a collaboration group size of one is the case where the initiator downloaded the file without any collaboration.

7.4 Comparison of Work-Queue and Optimized Work Distribution Algorithms

The main advantage of the opportunistic algorithm over the simple work-queue model is its ability to utilize even those nodes in its WLAN whose costs are not strictly lower than the initiator's admissible cost. Among a given set of nodes, the opportunistic scheme could potentially enlist more collaborators and thereby achieve higher throughput.

To validate this, we implemented the opportunistic work-distribution algorithm with community sizes of 5 and 3. Let G be the set of the nodes with TC values greater than the admissible value, $\widehat{TC}$. We fixed the size of G to be 2 and

---

[3]Linear increase in throughput would be a reasonable approximation for a small number of devices connecting to the same cell tower. With TDMA networks (e.g., GSM, which also encompasses GPRS), aggregation would buy you access to more TDMA slots as well as multiple channels. Definitely there are limits to the number of slots or channels that could be used, but we wouldn't come anywhere close to it for the scales that we consider in COMBINE.
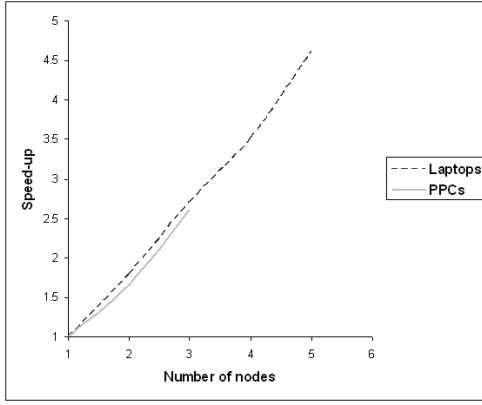
Figure 3: Variation of the speed-up of *COMBINE* with varying collaboration group size (including the initiator)

Number of nodes	Laptops (kbps)	PPCs (kbps)
1	95	77
2	170.25	127.82
3	256.57	200.97
4	333.55	-
5	438.77	-

Table 1: Variation of the Throughput of *COMBINE* with varying collaboration group size (including the initiator)

1 respectively for the two communities. Let $\widehat{TC_G}$ be the average value of TC of the members in G. We vary the ratio of $\widehat{TC_G}/\widehat{TC}$ and measure COMBINE's throughput. Figure 4 plots the throughput of *COMBINE* as this ratio. increases. We use a file size of 8 MB and a partition-size, p, of 1 MB.

The work-queue algorithm is not affected by the variation in G because it never considers nodes in this set for distributing work. The horizontal line in Figure 4 is indicative of this. On the other hand, the opportunistic work-distribution algorithm is capable of using nodes in the set G as long as their costs are not significantly above the admissible cost, $\widehat{TC}$. Our results show that up to a factor of 2, the opportunistic work distribution is appreciably superior to the work-queue algorithm. For higher ratios, the opportunistic algorithm starts allocating lesser work to the members in the set G and utilizes them minimally to meet the cost constraint. In the limit the algorithm behaves as though the nodes in G did not exist.

7.5 Agility and Adaptation

In this section, we evaluate *COMBINE's* behavior in the presence of variability in collaborator's network characteristics.

Specifically, when a collaborator's WWAN link quality is degraded we want to ensure that *COMBINE* can automatically reapportion the work among collaborators.

We consider the case of an initiator using the work-queue model to distribute work to four collaborators. We artificially slowed the download rate of a single collaborator and observed the adaptability of the system. Figure 5 shows the
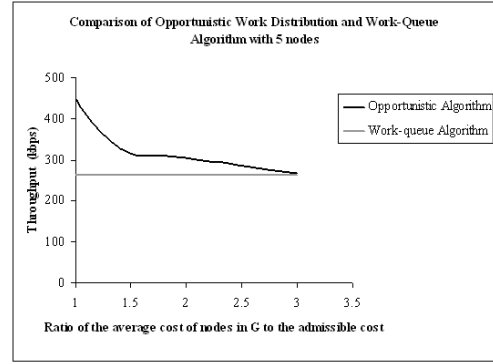

Figure 4: Comparison of the throughput of the opportunistic work distribution algorithm with the work-queue algorithm as the ratio $\widehat{TC_G}/\widehat{TC}$ varies in a five-node scenario

effect of progressively slowing a single collaborators from full-speed down to 0.75, 0.5, 0.25 and 0.1 of its original rate. The vertical dotted lines indicate the slow down events. We observe that even as the throughput of the slow node progressively decreases, the average throughput of the healthy nodes is invariant and they automatically start servicing more chunks from the work-queue.
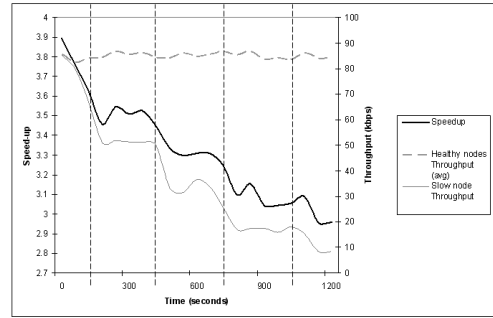

Figure 5: Agility of COMBINE with progressive reduction in WWAN speed of one of the collaborators

Prioritization of local work: *COMBINE* aims to leverage only the unutilized WWAN bandwidths of the collaborators to achieve throughput enhancement and hence it is crucial that there is a mechanism in place that automatically prioritizes local WWAN activity over *COMBINE* activity, as discussed in Section 6.2. We simulated this by intentionally downloading a file locally at a collaborator during the course of *COMBINE's* activity. Figure 6 illustrates the efficacy of our prioritization module. As soon as we see a surge in the local WWAN activity, there is a drop in the amount of data downloaded by the collaborator as a part of *COMBINE*.

7.6 The Impact of Cryptography

Recall that the initiator generates IOUs to the collaborator using cryptographic signing techniques for security. This section describes the impact of cryptographic operations on the throughput and battery energy. We evaluate the impact of signing an IOU on the i-mate PPC by way of a microbenchmark.
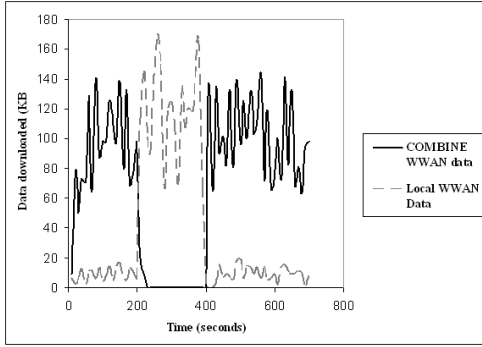
Figure 6: Prioritization of local WWAN activity over COMBINE activity

We assume an IOU to be about 256 bytes long. To sign an IOU, the initator (i.e., the mobile phone) calculates a hash of the IOU, and then signs the hash with its private key. Verification involves calculating the hash of this message and decrypting it with the public key.

Previous work indiate that Elliptic Curve Digital Security Algorithm (ECDSA) has a better performance, especially on mobile devices, compared to traditional cryptosystems like RSA because of the reduced key sizes [12]. The security of ECDSA with a key length of 160 bits is equivalent to a 1024-bit RSA encryption and we believe that this is sufficient in practice.

We measured the time taken and power expended for the signing and verification operations of the digital signatures on the i-mate Pocket PC by averaging over 1000 iterations. We used SHA_1 for hashing and ECC for the encryption and decryption operations. Table 2 show our results.

We now show by a simple calculation that the energy overhead due to the IOU's is negligible on the device. Our experiments indicate that the maximum transfer possible on the WWAN in a single lifetime of a battery is approximately 150 MB. If we assume that an IOU is generated every 200 KB (optimal chunk size), a device generates a total of 750 IOU's. It can be easily verified that the total cost for the generation of these 750 IOU's is a very small fraction (0.75%) of the total battery capacity (3.7 V, 1250 mA-h). Hence we conclude that our IOU scheme does not place a significant overhead on the device.

Table 2: Power Consumption of an i-mate Pocket PC for cryptographic operations with ECDSA

	Energy (mJ)	Time (ms)
Verifying	1.43	6.79
Signing	166.5	280

7.7 Estimating Battery Depletion

We evaluate the effectiveness of the simple model presented in Section 4.2 in estimating battery depletion. We focus here on the case where the i-mate PPC only has a WWAN NIC. Recall that the battery depletion model is:
$$BD = time\_elapsed * BD_t + bytes\_sent\_or\_recd * BD_d.$$
We first estimate BD_t and BD_d using controlled "cali-

bration" experiments. To estimate BD_t, i.e., the battery depletion per unit time, we power on the device and its WWAN NIC (but do not perform any network transfers) and measure how long it takes for battery to fully discharge (i.e., go from 100% to 0% battery remaining). BD_t is then estimated as $1/discharge\_time$.

Then, to estimate BD_d, i.e., the battery depletion per unit data, we perform a similar calibration experiment with one crucial difference: the WWAN NIC is engaged in a continuous download at full throttle. We record how long it takes for the battery to fully discharge and also the amount of data received during this period. Using BD_t and the discharge time, we compute how much of the discharge was due to the device and the WWAN being simply turned on over this period. The remaining discharge is attributed to actual data reception; dividing it by the amount of data received yields an estimate of BD_d.

Next, we examine how accurately BD_t and BD_d can be estimated based on observations made while the device is engaged in "COMBINE-like" network activity rather than the steady and controlled workload of the calibration experiment. The COMBINE-like workload comprises large bursts of data transfer (1-5 MB in size at random), interspersed with idle periods ranging up to 10 minutes in duration. We record the battery remaining at the beginning and end of each burst. We report results from two runs of this experiment conducted a day apart. The two runs included 27 and 32 bursts of data transfer activity, respectively.

Experiment	BD_t (per second)	BD_d (per KB)
Calibration	1.1E-5	5.1E-6
Run #1	2.4E-5	4.4E-6
Run #2	2.4E-5	4.2E-6

Table 3: Estimates of battery depletion per second and per KB.

As reported in Table 3, the estimates of BD_t and BD_d obtained from the COMBINE-like runs match those obtained from the calibration experiment reasonably well. More importantly, the estimates obtained from one run match those from the other very well. We verified that the estimates of BD_t and BD_d from run #1 yield accurate estimates of battery depletion during various subsets of run #2 (i.e., sequence of bursts together with the intervening idle periods). We do not show these results here due to space limitations.

These experiments with the WWAN suggest promise in our ability learn the battery depletion characteristics simply by observing the device in operation.

8. DISCUSSION

We briefly discuss security and user interface issues pertaining to COMBINE.

8.1 Security

While collaborative downloading offers a significant performance benefit, it also raises several security issues. We briefly discuss these and ways of addressing them that could be incorporated into COMBINE.

First, there is the issue of *privacy*, with regard to leaking information on a user's activity to collaborators and/or the

accounting system. For example, a collaborator might learn which URLs or files have been accessed by the initiator. However, despite the IOUs signed by it, the initiator can still effectively remain anonymous and difficult to track, as discussed in Section 5.4. So the collaborator does not gain much since it cannot tie back the URLs and files to any specific user.

Second, there is the issue of *confidentiality*. For example, the initiator might be downloading a music file that is only available to subscribers. Existing end-to-end encryption techniques, employed independently of *COMBINE*, would help. For example, with the collaborators operating as web proxies, using SSL would shut them out of the content just as it would any other web proxy.

Finally, there is the issue of ensuring the *authenticity* of the content downloaded through the collaborators. In *COMBINE* an initiator issues an IOU for a chunk of data in advance of a collaborator actually downloading and supplying the data. A collaborator could cheat, for instance, by failing to perform the requested download or by returning bogus content instead of expending WWAN bandwidth to download the real content. Even if the the damage caused to the initiator by any one such instance of cheating is small, a malicious collaborator can accumulate a large volume of ill-gotten credits from multiple initiators over time.

Addressing this problem requires a combination of (a) certification of content blocks by the source (just as systems such as BitTorrent would need) so that the initiator can tell whether it has received valid content, and (b) a reputation system to blacklist persistent cheaters. Such blacklisting is facilitated by the availability of an identity infrastructure, which helps identify the cheater (e.g., the initiator can request proof of identity from each collaborator at the start of a collaboration session) and a trusted accounting service, which can record complaints against a cheater. Even if the cheater has multiple identities, as noted in Section 5.4, the accounting service can group together the set of identities associated with any user, since it had issued them in the first place. Also, the assumption is that users would need to go through a relatively heavyweight process to register with the system and receive a certificate (e.g., they might have to set up a billing relationship with a provider or produce a valid credit card). So it would be expensive and difficult for a blacklisted user to rejoin the system repeatedly.

8.2 User Interface

It is desirable for *COMBINE* to operate autonomously, without requiring user input on an ongoing basis. However, since users may expend valuable resources and/or accrue monetary charges, we would still want to give them the *option* of exercising control. We present here our initial ideas on the design of a simple user interface for *COMBINE*.

The UI would comprise elements that inform the user and those that allow the user to exercise control. In the former category are "dials" that show the user (a) how much credit/dues they have accrued, (b) the speedup obtained by using *COMBINE*, and (c) the amount of resources expended (or remaining), in particular battery power. The user can view these dials as and when desired, and use the information presented to drive how they control the operation of *COMBINE*.

To enable user control, the UI could include two sliders, one each to control the aggressiveness of selling and buying

bandwidth, respectively. The slider settings are translated to numerical factors, K_s and K_b, to control the buying and setting of resources, respectively. These factors are initially set to a neutral value (say 1) to ensure reasonable operation by default. The K_s factor is used to scale down or up the price computed in Section 4, to make collaborator more or less willing sell its resources. At the extreme, K_s is set to infinity, which means that the seller is unwilling to sell for any price. The factor K_b operates analogously, with the base price that the buyer (viz., the initiator) is willing to pay pegged to the cost it would incur (in terms of bandwidth and battery) were it to do the download by itself. At the extreme, K_b is set to zero, which means that the initiator is unwilling to pay any price, effectively opting out of collaborative downloading.

We could also overlay demand/bid information from the neighborhood on the sliders so that the user knows how aggressive they would need to be to make a deal. This could be the basis of a game for (idle) users looking to earn some extra money by parlaying their unused resources.

8.3 WWAN Service Provider

COMBINE provides a way for devices to achieve a higher effective bandwidth purely through software means and without requiring expensive infrastructure upgrades on the part of the WWAN service providers. We believe that the increased throughput would translate into a better user experience for the users and result in more data downloads across the WWAN, thereby directly benefiting the service providers.

One way for the service providers to play a more active role in COMBINE could be by serving as the accounting authority. This is relatively simple because it already has billing relationships with the users and hence has the identity infrastructure. Redemption of IOUs could be in terms of additional talk-time or data limits.

9. CONCLUSION

Collaborative downloading offers the potential for significant performance gains by utilizing WLAN and WWAN links in combination. This paper has described the design, implementation, and initial performance from a prototype calld *COMBINE*. Our experience with the system suggests that it offers excellent scaling as well as good adaptability to changing network conditions.

COMBINE provides a framework to provide practical economic incentives to collaborators. It also makes few demands on the wired and wireless infrastructure than what is already present. We believe these aspects make the ideas in our system easy to deploy.

Acknowledgements

10. REFERENCES

[1] ECash. *http//www.ecash.com*.
[2] YouTube coming soon to cell phones. *http://news.com.com/YouTube+coming+soon+to+cell+phones/2100-1032_3-6138794.html*, Nov 2006.

[3] M. Adler, R. Kumar, K. Ross, D. Rubenstein, D. Turner, and D. Yao". Optimal peer selection in a free market peer-resource economy. In *Second Workshop on Economics of Peer-to-Peer Systems*, June 2004.

[4] S. Buchegger and J.-Y. L. Boudec. Performance analysis of the CONFIDANT protocol: Cooperation of nodes — fairness in dynamic ad-hoc networks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*. IEEE, June 2002.

[5] C. Carter and R. Kravets. User Devices Cooperating to Support Resource Aggregation. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, pages 59–69. IEEE, June 2002.

[6] R. Fielding et al. Hypertext transfer protocol – HTTP/1.1: RFC 2616. *http://www.ietf.org/rfc/rfc2616.txt*, June 1999.

[7] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro. The Millicent Protocol for Inexpensive Electronic Commerce. In *The World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, Dec. 1995.

[8] R. Gupta and A. K. Somani. Compup2p: An architecture for sharing of computing resources in peer-to-peer networks with selfish nodes. In *Second workshop on the Economics of Peer-to-Peer systems*, June 2004.

[9] H.-Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidth on mutli-homed mobile hosts. In *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 83–94, Sept. 2002.

[10] IEEE Std 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications. IEEE Computer Society LAN MAN Standards Committee., Aug. 1999.

[11] K.-H. Kim and K. G. Shin. Improving TCP Performance over Wireless Networks with Collaborative Multi-homed Mobile Hosts. In *MobiSYS '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 107–120. ACM Press, June 2005.

[12] K. Lauter. The advantages of elliptic curve cryptography for wireless security. In *IEEE Wireless Communications*, Feb. 2004.

[13] H. Luo, R. Ramjee, P. Sinha, L. Li, and S. Lu. UCAN: A Unified Cellular and Ad-hoc Network Architecture. In *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Sept. 2003.

[14] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pages 255–265, August 2000.

[15] A. Miu and E. Shih. Performance analysis of a dynamic parallel downloading scheme from mirror sites throughout the internet. Technical report, MIT LCS, Dec. 1999. Term paper.

[16] A. Qureshi and J. Guttag. Horde: Separating Network Striping Policy from Mechanism. In *ACM/Usenix MobiSys*, June 2005.

[17] M. K. Reiter and A. D. Rubin. Anonymity Loves Company: Anonymous Web Transactions with Crowds. *CACM*, Feb. 1999.

[18] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. MAR: A Commuter Router Infrastructure for the Mobile Internet. In *ACM/Usenix MobiSys*, June 2004.

[19] P. Rodriguez, A. Kripal, and E. W. Biersack. Parallel-access for mirror sites in the internet. In *IEEE Infocom 2000*, Mar. 2000.

[20] P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin. Handheld Routers: Intelligent Bandwidth Aggregation for Mobile Collaborative Communities. In *BROADNETS '04: Proceedings of the First International Conference on Broadband Networks*, pages 537–547. IEEE, Oct. 2004.

[21] E. Shih, P. Bahl, and M. Sinclair. Wake On Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *Mobicom*, Sept. 2002.

[22] A. C. Snoeren. Adaptive Inverse Multiplexing for Wide-Area Wireless Networks. In *IEEE Global Internet Symposium*, Dec. 1999.

[23] M. Stemm and R. H. Katz. Measuring and Reducing Energy Consumption of Network Interfaces in Handheld Devices. In *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Science,*, Aug. 1997.