Unity: Collaborative Downloading Content Using Co-located Socially Connected Peers

Prateek Jassal, Kuldeep Yadav, Abhishek Kumar, Vinayak Naik, Vishesh Narwal, Amarjeet Singh,
Indraprastha Institute of Information Technology, Delhi (IIIT-D)
New Delhi, India
Email: {kuldeep, naik}@iiitd.ac.in

***Abstract*—**

Large proliferation of mobile phone applications result in extensive use of data intensive services such as multimedia download and social network communication. With limited penetration of 3G/4G networks in developing countries, it is common to use low bandwidth 2G services for data communication, resulting in larger download time and correspondingly high power consumption.

In this paper, we present a system architecture, *Unity*, that enables collaborative downloading across co-located peers. *Unity* uses short range radio interfaces such as Bluetooth/WiFi for local coordination, while the actual content is downloaded using a cellular connection. *Unity* is designed to support mobile phones with diverse capabilities. End-to-end implementation and evaluation of *Unity* on Android based phones, with varying workload sizes and number of peers, show that *Unity* can result in multi-fold increase in download rate for the co-located peers. We also describe architecture of cloud-based *Unity* which uses principles of mobility prediction, social interactions, and opportunistic networking to make collaboration more pervasive and useful.

I. Introduction

Majority of new cellular subscriptions today are coming from developing countries. As per recent statistics, number of 3G subscribers in China are only 14% of the total 1 billion cellular subscribers[1], while in India, it is only 2% of over 893.8 million subscribers[2]. Low penetration of $3G/4G$ networks is attributed to higher setup cost, limited number of supporting handsets and expensive data plans. As a result, many people still use $2G$ based data connection which is widely deployed and accessible on most of the phones. In countries like India and Egypt, over 50% of users access Internet from mobiles only[3] and China has more mobile Internet users than PC[4]. Recent Opera report shows[5] that, mobile users download content (mostly multimedia) from the Internet using $2G$ network and top handsets used were found to be feature phones.

Most advanced 2G technology (EDGE) can provide download throughput of up to 48 KBps. We performed an experiment to measure the throughput of 2G data connection in wild by repeatedly downloading a MP3 song of about 5MB size on five different phones, used by volunteers for a week. The median download throughput achieved by the EDGE network across two operators was about 18 KBps while the variation was from 4 KBps to 28 KBps. We also observed many instances of failed downloads - approx. 22% downloads failing for Operator A and approx. 42% for Operator B. Low throughput and failed downloads are primarily due to two major reasons - variable wireless conditions (low RSSI) and variable load on cellular networks. As a result downloading content (especially multimedia) on EDGE results in several limitations - 1) Higher time to download; 2) Excessive power consumption due to low throughput (cellular radio is switched ON irrespective of data speed [4]); and 3) Poor user experience due to frequent failed downloads. Additionally, there is a lack of appropriate systems that can assist users while downloading content in limited bandwidth conditions offered by $2G$.

In this paper, we present *Unity*, a system that allows multiple co-located phones (peers) to collaborate in downloading a commonly desired content (workload). In *Unity*, co-located peers participate to individually download different parts of the desired workload and then share their downloaded part with each other such that everyone gets the complete content at the end. Mobile phone users are typically expected to be part of several social gatherings during the day at different places i.e. home, workplace, and even while commuting [6]. The key idea of *Unity* is to use these social meetings to provide a collaborative environment with a usable interface to enable faster downloads of content desired by all (or most) of the participating peers. Almost all the phones, including feature phones, have one or more small range radio technologies, such as WiFi, Bluetooth, NFC. *Unity* leverages one of the available short range technologies for coordinating and local sharing of workload parts amongst peers while each of the part is downloaded by individual peer from the Internet using cellular connection. Specific contributions of this paper are as follows:

1) We present a system, *Unity*, (shown in Figure 1) that exploits multiple radio interfaces available in common phones for collaborative download of mutually desired content by co-located socially connected peers.
2) We implement and discuss different variants of *Unity* based upon A) Local communication interface - using WiFi (*Unity-WiFi*), using Bluetooth (*Unity-Bluetooth*) and B) Collaborative mechanisms - *Unity-Default* in which workload is equally divided among peers and *Unity-Adapt* that adapts to changing cellular network conditions. All these variants are essentially different components of final *Unity* system.

[1] http://goo.gl/QvMJ9 [2] http://goo.gl/ZGbUj [3] http://goo.gl/oR8Au
[4] http://goo.gl/lYkr9 [5] Opera Mini is a widely used browser in mobiles

978-1-4244-9529-0/13/$31.00 ©2013 IEEE

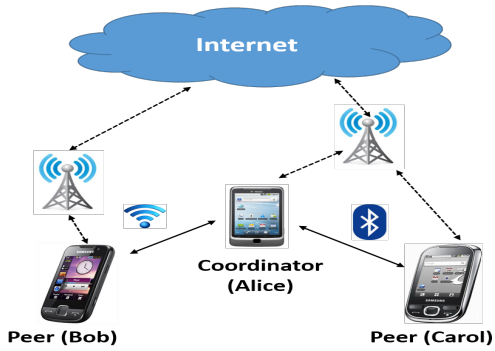


Fig. 1: Design of proposed system *Unity* , a group of mobile phones users collaborates with each other to download a single workload which are of interest to all of them

3) We implemented *Unity* on Android-based phones for proof-of-concept and evaluated its effectiveness with different workload sizes and varying number of collaborating peers. Our findings show that *Unity* can increase the download rate by a factor of 3 when used by 4 peers as compared to the best download rate amongst all peers.
4) Finally, we present *Unity-Cloud* - a cloud-based architecture to extend *Unity* using user's redundant mobility profiles and their social network. *Unity-Cloud* also minimizes the power consumption for a desired download by predicting the best time (when the Internet bandwidth is the best) to download.

II. System Architecture and Implementation

System architecture of *Unity* is guided by various capabilities of commonly available phones. Several implementation aspects are described subsequently with the design details. Initial implementation of *Unity* was built for Android phones as they continue to grow in countries like India and at the same time, provide rich networking API support essential for our implementation. *Unity* works in two different modes for the participating phones i.e. a phone can either act as a coordinator or a peer. The coordinator initiates the download, recruits phones in the geographical vicinity that are willing to participate in the download process, and coordinates all the communication with the participating peers. The peer connects to the coordinator, downloads a part of the desired workload and shares it with the coordinator. The coordinator, within itself, also runs a peer instance to share the download workload.

Usage Scenario: Let us now explain the utility of *Unity* through a usage scenario, as shown in Figure 1. Three friends *Alice*, *Bob* and *Carol*, are all interested in the multimedia content "V" and decide to use *Unity* for collaborative downloading as follows:

1) *Alice* decides to be the initiator and therefore launches *Unity* coordinator mode. She finds the URL of "V" and feeds it into *Unity*. *Bob* and *Carol* launch the peer mode of *Unity* .
2) After *Alice* chooses a network interface available within all three of their devices e.g. Bluetooth/WiFi, *Unity* launches device discovery on the selected network interface to find nearby peers - *Bob* and *Carol*, and recruit them as peers.
3) From the given URL, *Unity* in the coordinator mode finds the size of "V" using the HTTP protocol request and equally divide the file workload among the three peers by communicating each one of them the URL address with block information[6].
4) On receipt of content download request from coordinator, peers start downloading the assigned blocks using their cellular connection.
5) On complete download of the assigned individual block, each peer sends it to the coordinator who combines all the blocks to make the desired content "V".
6) Coordinator also communicates the remaining blocks (from the received and downloaded blocks) to each peer. As a result, each participating peer gets access to the whole file after combining the received blocks with its own downloaded block.

A. *Modules of* Unity*:*

We have kept design of *Unity* modular so that it can easily run on phones with different capabilities. A brief description of different modules in *Unity* is as follows:

- User Interface: One of the main design principle of *Unity* is to abstract out various complexities of the system from the user and provide a usable interface which can be used for collaborative downloading. This module is responsible for showing different screens to user based on the selected mode i.e. Coordinator or Peer. Figure 3a shows the home screen of *Unity* for coordinator which accepts different parameters from the user to get started.
- Local Networking Module: *Unity* needs frequent message passing and data sharing among different peers. This modules enables seamless data sharing and message passing between different peers and coordinator using P2P data transfer technologies such as WiFi and Bluetooth. This module is invoked by the controller module whenever there is a need to do data exchange across devices. A detailed description of working and implementation of this module is presented in Section II-B.
- Downloader Module: The main task of this module is to connect to the Internet using a cellular connection and download desired content. This module is invoked by the controller module in both modes, while passing URL address and byte ranges as an input. This module also provides functionality for updates on download and cellular speed status to user interface to make it interactive. If download of a workload got failed in between, it restarts the download of a workload from the point it got failed.
- Controller Module : This module has different functionalities for coordinator and peer modes. It is used to invoke different modules in both modes of *Unity*. For instance,

[6] It denotes the number of bytes with start and end byte to be downloaded

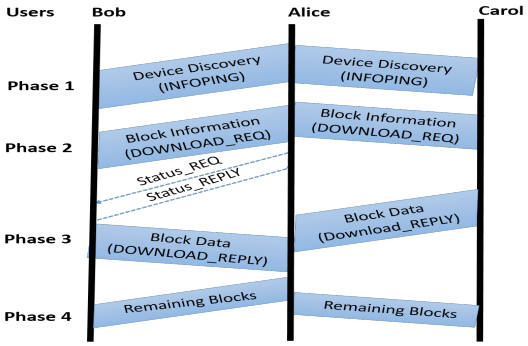


Fig. 2: Sequence diagram of various control and data exchanges between different phones in *Unity-WiFi*

based on user choice, it selects collaboration policy i.e *Unity-Adapt* or *Unity-Default* (described in Section II-C) and computes the download size for each peer. Additionally, in the peer mode, it invokes local networking module on completion of download to transfer the content to the coordinator.

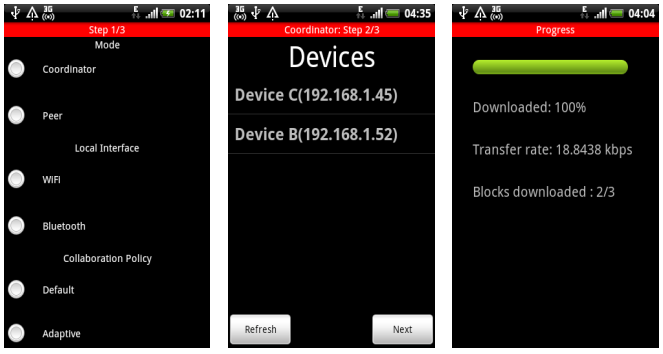
B. Local Networking

Unity's local networking module may use WiFi and Bluetooth based on its availability on the phones. However, WiFi and Bluetooth have different networking stack in the phones and thus, require completely different and independent implementation.

1) Unity-WiFi: This is a variant of *Unity* which uses WiFi for local communication. WiFi (802.11) supports two different modes: Infrastructure and Adhoc. In infrastructure mode, two or more WiFi enabled devices have to use a intermediate WiFi access point (AP) to communicate between them because AP is used for routing of data packets. In adhoc mode, two different devices can directly (i.e. P2P) communicate with each other without any AP. Android started supporting WiFi Adhoc mode after OS version 4.0, popularly called as WiFi-Direct. There are large number of phones, with prior Android OS versions such as 2.2 or 2.3[7]. Building our system with WiFi-Direct would have eliminated more than 70% of the total Android based phones. Further, WiFi adhoc mode results in higher energy consumption as all the peers have to stay awake and send beacons to exchange data whenever required.

As an alternative of WiFi adhoc mode, we use a novel utility provided by Android called as WiFi hotspot, primarily designed for sharing the Internet connection of the phone with other devices such as a laptop. WiFi hotspot utility is available on all version of Android which are running Android 2.3 or beyond. WiFi hotspot utility uses 802.11 infrastructure mode which turn the phone as WiFi AP and other phones[8] can connect to it. For simplicity, let us assume that coordinator is acting as WiFi AP and all other phones connected to it are different peers. Figure 2 shows various control and data exchanges between a *Unity* coordinator and two *Unity* peers, following is corresponding description:

[7] http://goo.gl/d0C5D [8] Android 2.3 based AP can support up to 6 connected devices whereas Android 4.0 supports up to 7



(a) Home Screen (b) Device Discovery (c) Progress Status

Fig. 3: Different screens for *Unity* coordinator: (a) shows the different modes and variants of *Unity*, (b) Peers running *Unity* peer mode and (c) Transfer rate of downloading and blocks received from other devices

1) The phone, which is running *Unity* coordinator creates WiFi AP and other peers connect to it as clients.
2) As shown in Figure 2, coordinator launches device discovery to discover all connected peers and exchange a few control messages with them individually to get information such as peer name. (Refer Phase 1)
3) After device discovery step, block information and URL is passed on to all the peers using a control message and each of them start downloading their block from Internet. By default, Android uses WiFi AP functionality for enabling tethering and it may happen that peers start downloading using coordinator's data connection. To force the peers to use their own data connection, we change the connection priority during download. (Refer Phase 2)
4) Coordinator can also check the status of block download in between by sending a status request.
5) On download completion, peers send their data blocks to the coordinator. On receipt of blocks from all the peers, coordinator sends the remaining blocks for each peer to them. Peers then merge the received blocks with downloaded blocks to get the complete content. (Refer Phase 3 and 4)

Coordinator, acting as WiFi AP, will be awake for whole download duration while the peers can operate in power saving mode (PSM) which consumes negligible energy [10] or even turn off their WiFi to save energy when not in use. As shown in Figure 1, star topology where one phone, acting as coordinator, communicates with all the other phones results in smaller local communication bandwidth as compared to the distributed architecture. *Unity-WiFi* requires that at least one person in the group should have a phone with WiFi AP capability and all other phones should have WiFi.

2) Unity-Bluetooth: To enable *Unity* on feature phones, we also developed a Bluetooth based local networking module. Bluetooth only supports adhoc P2P connection. In case of *Unity*, coordinator runs Bluetooth server instance and the peers run Bluetooth client instance. All control and data

exchanges in *Unity-Bluetooth* happen in the same order as *Unity-WiFi*. In the device discovery phase, each *Unity* peer creates a bluetooth socket with a service record[9] and listens for incoming connections whereas *Unity* coordinator connects with them subsequently and exchanges information such as peer name as shown in Figure 2. Bluetooth server stores all the UUIDs with peer names for future communication. Unlike *Unity-WiFi*, it does not require changing data priority on different peers.

C. Collaboration Schemes

As described in controller module, *Unity* has two different collaboration schemes - *Unity-Default*, *Unity-Adapt*. *Unity-Default* divides the desired workload of size d into equally sized blocks. If there are n devices participating in the download, block size, to be downloaded by each peer, will be d/n. This scheme has advantage in terms of fairness as all the collaborating peers will incur equal amount of data connection expense. However, in cellular network conditions, it is usual that some nearby peer may be experiencing poor cellular network conditions resulting in low download rate. In such cases, this scheme will result in increased waiting time for *Unity* coordinator and other peers due to the peer who is experiencing low throughput. Our experiments showed that for large downloads, this incremental wait could be several minutes thus correspondingly increasing the energy consumption as well.

To reduce this waiting time, *Unity* uses an algorithm, which adapts to changing network conditions termed as *Unity-Adapt*. For a workload of size d, *Unity-Adapt* divides it into equally sized blocks of size k[10]. *Unity* coordinator assigns each peer a single block to download at a time and the peer is expected to request another block to download whenever it finishes downloading 80% of the assigned block. *Unity* coordinator will keep on allocating the blocks dynamically until all the blocks are assigned. Thereafter, *Unity* peers will send all the downloaded blocks to *Unity* coordinator together to minimize control overhead and frequent connections.

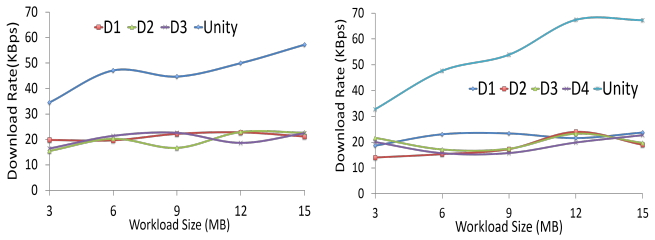
III. Evaluation

We now present results from extensive evaluation of *Unity* while running on Android phones. First, we define some of the evaluation metrics for *Unity*. *Total download time* is the time taken by *Unity* to collaboratively download a workload and deliver it to all the collaborating peers. From *total download time*, we compute *effective download rate* which is equal to workload size divided by *total download time*. Our evaluation experiment consists of four Android phones, three of them manufactured by HTC and one by Samsung. All the phones were running Android 2.3.3 OS.

A. Download Rate vs Workload Size:

To evaluate download rate in *Unity* with varying number of collaborating devices and varying workload sizes, we

[9] *Unity* has a common service name and unique UUID number for each peer
[10] Value of k in this case is typically greater than the number of collaborating devices



(a) Download rate with 3 devices (b) Download rate with 4 devices

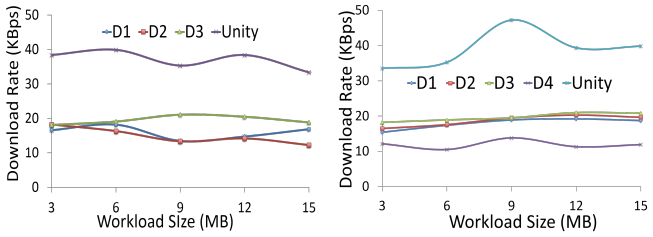
Fig. 4: Download rate of *Unity-WiFi* with different workloads and total 3 and 4 collaborating devices, D1, D2, D3, and D4 represents the individual device's estimated download rate.

downloaded five different workloads i.e. 3 MB, 6 MB, 9 MB, 12 MB, 15 MB with default collaboration policy. Number of collaborating devices were varied from 2, 3 and 4 for each of the workload. For each download instance, the download rate of individual devices are computed from the time taken by them to download the assigned workload and effective download rate of *Unity* is computed as defined in metrics above. In case of *Unity-WiFi* with 3 devices, as shown in Figure 4a, *effective download rate* increases linearly with workload size. *Unity* download rate is comparatively low for smaller workloads as local communication overhead for collaboration across different peers takes significant time. However, with increasing workload size, this overhead becomes negligible. Similar trends were observed in the case of *Unity-WiFi* when used with 4 different devices, as shown in Figure 4b.

For comparison purpose, we define a *baseline download rate* which is equal to the highest download rate among peer devices assuming that a given peer would have downloaded the complete workload at the same rate as it performed while downloading part of the workload. In the case of 4 devices, *Unity-WiFi* makes download faster by a factor of approx. 1.5 for smallest workload (3 MB) and a factor of approx. 3 for the largest workload (15 MB), as compared to the *baseline download rate*. In the case of *Unity-Bluetooth* with 4 devices, download rate increased by a factor of 1.8 for the smallest workload and a factor of 2 for the largest workload as compared to the *baseline download rate* (refer Figure 5b). In *Unity-Bluetooth*, download rate of *Unity* increased marginally when workload size is increased mainly due to the higher overhead with Bluetooth. We also observed that some of the devices in *Unity-Bluetooth* experiments downloaded with a slower rate resulting in higher *total download time* and smaller improvements in *effective download rate*.

B. Overhead Comparison:

Total download time for *Unity* consists of workload downloading time from internet, local sharing amongst collaborating devices and merging the shared workloads. It is useful to accurately quantify the overhead caused by different *Unity* operations i.e. local networking and merging, and compare them with the *total download time*. For this purpose, we ran three instances of workload (12 MB) using *Unity* and collected logs with high resolution time intervals for these activities.



(a) Download rate with 3 devices (b) Download rate with 4 devices

Fig. 5: Download rate of *Unity-Bluetooth* with different workloads and number of collaborating devices, D1, D2, D3, and D4 represents the individual device's estimated download rate.

Average overhead % across the 3 instances for *Unity-WiFi* and *Unity-Bluetooth* is shown in Figure 6.

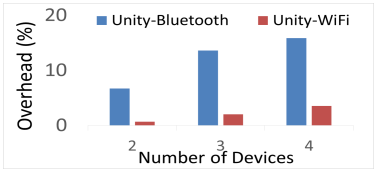


Fig. 6: Overhead % comparison between *Unity-WiFi* and *Unity-Bluetooth*

We observed that much of the overhead in *Unity* is dominated by local networking module for exchanging control and data messages across different devices. As a result, overhead % using WiFi is smaller than using Bluetooh due to the corresponding difference in data transfer rates (WiFi: 1.5 - 2 MBps and Bluetooth: $450 - 480$ KBps) [10]. This difference in data rate also explains the reason for lower increase of download rate in *Unity-Bluetooth* as shown in Figure 5.

C. *Measuring Impact of* Unity-Adapt

Due to variable cellular network conditions, one or more devices may download at a lower rate in *Unity* thereby increasing the overall download time. As an instance, in Figure 5b, device $D4$ downloaded with slower rate as compared to the other 3 devices. To avoid such a situation, *Unity-Adapt* divides the whole workload into smaller block sizes and keeps assigning them to the collaborating peers based on their download rate. Empirically, we observed that block size equal to 1 MB works well in *Unity* and used it for these experiments. With 3 collaborating devices, we gave *Unity* a workload of 12 MB to download in three different instances. Across all of these instances, average downloads rate of the three devices were 5.94 KBps, 8.14 KBps and 10.54 KBps for D1, D2 and D3 respectively. On an average, *Unity* without adaptation downloaded the whole workload in approx. 692 seconds. However, when using *Unity-Adapt*, *total download time* was reduced to approx. 505.78 seconds resulting in approx. 27% improvement. Additionally, the workload downloaded by a peer on an average was representative of their download rate i.e. D1 (3 MB), D2 (4 MB) and D3 (5 MB).

IV. Cloud-based *Unity*

Unity provides a platform for collaborative downloading of content of common interest to mobile peers. However, it requires users to explicitly ask their friends if they would be interested in the same content and would be willing for collaboration. Further, *Unity* requires all the collaborating peers to be in geographical proximity for the whole duration. Simultaneous download by geographically close peers at the same place and time may result in lower download rate from the cellular connection as well as consume more battery. We extend current architecture of *Unity* with the help of the Cloud to address some of these limitations and make collaboration more useful and pervasive. An architecture of *Unity* with the Cloud i.e. *Unity-Cloud* is shown in Figure 7. The cloud acts as a control information gateway among different mobile peers interested in collaboration.

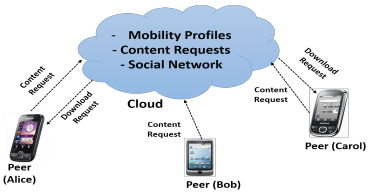


Fig. 7: Architecture of the Cloud-based *Unity*

The cloud primarily stores mobility profile of the users, their social network and various content requests from different users. Intuitively, people tend to visit same places and meet with many people in their social network regularly across several days. This is also validated empirically by prior research [6]. *Unity-Cloud* builds mobility profiles of people using heterogeneous location data sources such as Cell ID and WiFi as described in our previous work [2] and uses this profile to predict interaction place and time of the peers. The mobile client of *Unity-Cloud*, closely coordinates with the cloud in building mobility profile for the users by sensing location data and submitting content requests to the cloud in addition to what was done before [2]. Following is a use case scenario of *Unity-Cloud*:

1) A user submits a request for desired content to the *Unity-Cloud* with a time deadline. The cloud aggregates content requests from all the registered users.
2) *Unity-Cloud* helps the user in forming a peer group, from within her social network, with other users who are interested in downloading the same content and are expected to meet within the specified deadline. Expected meeting time and duration is calculated using users mobility profiles as mentioned in [2].
3) Once the user forms a peer group, *Unity-Cloud* communicates a request to download part of the desired content to all the collaborating peers. Each collaborating peer downloads the content whenever it finds good cellular conditions to download.
4) When all the participating peers are in geographic proximity (as calculated using their mobility profiles), the cloud will send them notifications to share their downloaded blocks with each other.

One of the salient feature of *Unity-Cloud* is that each peer downloads content whenever it experiences good cellular conditions. While predicting cellular throughput is hard given

the variable network conditions, *Unity-Cloud* uses some optimizations such as downloading in the night only or download when RSSI is good to result in improved throughput and energy conservation [3].

Unity-Cloud extends the architecture presented in Section II by helping users in finding friends who are interested in the same content as well as uses predictable mobility of the users for forming peer groups. Moreover, *Unity-Cloud* will result in lower energy consumption as it optimizes content download using better cellular conditions. We leave the evaluation of cloud-based *Unity* to subsequent work where we are planning to do a user study to evaluate the impact and effectiveness of *Unity-Cloud* in real-world.

V. Related Work

Several systems have been proposed targeting enhanced web access performance using support from multiple co-located peers. Eric et al [9] designed a framework for bandwidth sharing using markov decision processes while taking other parameters such as network conditions into account. However, they evaluated their framework only in simulation whereas our objective is to be build a working system. Cool-Tether [4] presents a cloud proxy based system which builds WiFi hotspot using multiple smart phones in vicinity to provide high speed data rate on a laptop client. Cool-Tether minimizes the energy consumption of smart phones by sending/receiving bursty HTTP traffic coordinated by a stripper module running on the laptop for uplink traffic and a cloud based proxy for downlink traffic. Cool-Tether improves upon COMBINE [8], a similar architecture proposed earlier that attempts to increase the web access performance without giving any consideration to energy consumed in the mobile devices. Both of these systems are different from *Unity* from both the architecture perspective and the target application scenarios. While COMBINE and CoolTether work on the assumption that all the peer devices are owned by or closely associated with the user, in the case of *Unity*, all the collaborating peers (devices) will benefit by downloading the mutually desired content. Recent work, MicroCast [7] targets the specific problem of video streaming using multiple nearby phones. Unlike *Unity*, MicroCast is implemented using features of custom ROM and does not address issues which come from implementation on off the shelf phones.

VI. Conclusion

Multiple people, specifically those who have similar interests typically inferred by social network or geographic proximity, have overlapping interests in desired content such as multimedia songs and videos. However, most often they tend to download the same content individually from Internet. In this paper, we presented a system *Unity* that enables collaboration between co-located and socially connected users to download mutually desired content from Internet. *Unity* is implemented as a complete system for Android and is evaluated for effectiveness on different workload sizes and varying number of collaborating devices. *Unity* user will benefit by incurring lower costs for data connection as well as multi-fold increase in download time.

While this work is focused on using limited bandwidth connection (2G) to download content from Internet, architecture and implementation of *Unity* would work in the same manner if some of collaborating peers have access to high bandwidth connection such as 3G. In such scenarios, *Unity-Adapt* requests peers with higher bandwidth to download larger chunks of the content thus resulting in further performance improvement. We are currently in the process of developing *Unity* for other mobile OS platforms and working to distribute it through corresponding app stores. We are also working on a user study to understand collaboration preferences of users through its usage in real-world.

Acknowledgements

Kuldeep Yadav is supported by a PhD Fellowship from Microsoft Research, India. This research is partially sponsored by a research grant from Microsoft Research, India.

References

[1] Yao, Jun, Salil S. Kanhere, and Mahbub Hassan. "An empirical study of bandwidth predictability in mobile computing." Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization. ACM, 2008..

[2] Yadav, Kuldeep, Vinayak Naik, Amarjeet Singh. MobiShare: Cloud-enabled Opprtunistic Content Sharing among Mobile Peers. Technical Report IIITD-TR-2012-009.

[3] Schulman, Aaron, Vishnu Navda, Ramachandran Ramjee, Neil Spring, Pralhad Deshpande, Calvin Grunewald, Kamal Jain, and Venkata N. Padmanabhan. "Bartendr: a practical approach to energy-aware cellular data scheduling." In Proceedings of the sixteenth annual international conference on Mobile computing and networking, pp. 85-96. ACM, 2010.

[4] Sharma, Ashish, Vishnu Navda, Ramachandran Ramjee, Venkata N. Padmanabhan, and Elizabeth M. Belding. "Cool-Tether: energy efficient on-the-fly wifi hot-spots using mobile phones." In Proceedings of the 5th international conference on Emerging networking experiments and technologies, pp. 109-120. ACM, 2009.

[5] Bayir, Murat Ali, Murat Demirbas, and Nathan Eagle. "Discovering spatiotemporal mobility profiles of cellphone users." In World of Wireless, Mobile and Multimedia Networks & Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a, pp. 1-9. IEEE, 2009.

[6] Vu, Long, Quang Do, and Klara Nahrstedt. "Jyotish: Constructive approach for context predictions of people movement from joint wifi/bluetooth trace." Pervasive and Mobile Computing 7, no. 6 (2011): 690-704.

[7] Keller, Lorenzo, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. "Demo: Microcast: cooperative video streaming on smartphones." In Proceedings of the 10th international conference on Mobile systems, applications, and services, pp. 463-464. ACM, 2012.

[8] Ananthanarayanan, Ganesh, Venkata N. Padmanabhan, Lenin Ravindranath, and Chandramohan A. Thekkath. "Combine: leveraging the power of wireless peers through collaborative downloading." In Proceedings of the 5th international conference on Mobile systems, applications and services, pp. 286-298. ACM, 2007.

[9] Jung, Eric, Yichuan Wang, Iuri Prilepov, Frank Maker, Xin Liu, and Venkatesh Akella. "User-profile-driven collaborative bandwidth sharing on mobile phones." In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, p. 2. ACM, 2010.

[10] Friedman, Roy, Alex Kogan, and Yevgeny Krivolapov. "On power and throughput tradeoffs of WiFi and Bluetooth in smartphones." In INFOCOM, 2011 Proceedings IEEE, pp. 900-908. IEEE, 2011.