Opportunistic, Collaborative and Synchronized, Proximal Device Ecology

Eyal Toledano

B.A. in Computer Science, The Interdisciplinary Center, May 2003

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, in partial fulfillment of the requirements for the degree of Master of Science in Media Arts and Sciences at the Massachusetts Institute of Technology

September 2013

© 2013 Massachusetts Institute of Technology. All Rights Reserved.

Signature redacted

Author
Eyal Toledano
Program in Media Arts and Sciences
August 9, 2013

Signature redacted

Certified by
Andrew Lippman
Senior Research Scientist
Program in Media Arts and Sciences

Signature redacted

Accepted by
Prof. Patricia Maes
Associate Academic Head
Program in Media Arts and Sciences
Opportunistic, Collaborative and Synchronized, Proximal Device Ecology

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning on August 9, 2013
in partial fulfillment of the requirements for the degree of Master of Science in Media Arts and Sciences at the Massachusetts Institute of Technology

Abstract

CoSync is an on-device software framework for coordinating proximal consumer electronic devices in order to create a synchronized, opportunistic and collaborative device ecology. The CoSync device ecology combines multiple stand-alone devices and controls them opportunistically as if they were one distributed, or diffuse, device at the user’s fingertips. CoSync interconnects these devices without requiring their users to coordinate in advance.

CoSync is inspired by the observation that people participate in serendipitous collaborations with those around them, both friends and strangers; proximal networking compliments the well-established client-server model with a more society like device ecology.

This thesis is centered on the design, implementation, and evaluation of CoSync. CoSync is architected to be platform independent and an Android specific reference implementation is provided. Several use cases are demonstrated, including device discovery, opportunistically using another device as network carrier connection, real-time media sharing, co-creation of media such as multi viewpoint video, and close synchronization between devices taking collective action, e.g. coordinating the flash on multiple cameras, and recording synchronized sound and images using separate devices.

This thesis work was partially supported by a research grant from Qualcomm Corporation, the Media Lab Digital Life Consortium and the Communications Futures Program

Thesis Supervisor
Andrew Lippman
Senior Research Scientist
Program in Media Arts and Sciences
Opportunistic, Collaborative and Synchronized, Proximal Device Ecology

Eyal Toledano

The following people served as readers for this thesis:

Signature redacted

Thesis Reader

Henry Holtzman
Research Scientist
Program in Media Arts and Sciences

Signature redacted

Thesis Reader

David P. Reed, Ph.D., Chief Scientist, TidalScale, Inc.
## Contents

### Acknowledgments

9

1 **Introduction**

10

Proximal Connectivity Trend and Challenges 10

Opportunistic Proximal Wireless Networks 11

The Social Challenge 12

Media Creation, Sharing and Bandwidth Challenge 13

The Connectivity Challenge 14

Business Model Challenges 17

Framework Prototypes and Demonstrations 17

Contributions 20

Lessons Learned 21

Thesis Outline 22

2 **Background and Related Work**

23

CoSync Framework Domains 23

Connectivity Sharing 23

Media Sharing 28

Synchronization and Remote action 29

3 **The Framework**

32

High Level Software Requirements 33

Software Design Principles 33

Software Architecture 35

4 **Framework Applications**

46

Airmobs - Opportunistic Connectivity Sharing 48

CoCam Media Sharing 59

CoSync Flash 63

CoSync Video 66

5 **Evaluation**

69

Connectivity Sharing Evaluation 69

Media Sharing Evaluation 70

Synchronization Method Evaluation 74

Synchronized Actions Evaluation 77

### Bibliography

82
Acknowledgments

First, I would like to thank my advisor, Andrew Lippman, for his guidance and support throughout the research process. I would also like to thank my thesis readers Henry Holtzman and David P. Reed for their critique and thoughtful comments.

A special big thanks goes to Dan Sawada, Grace Woo and Szymon Jakubczak who advised, contributed, helped and developed parts of the framework, It was an honor and a pleasure to work with you guys.

Special thanks for the members of the Viral Spaces Group for their continued critique, advice, experience and friendship; In particular I would like to thank Jonathan Speiser, Travis Rich, Polychronis Ypodimatopoulos and Dawei Shen.

Thanks to the amazing Deborah Widener for her support, facilitation, organization and endless sense and sensibility.

I would like to acknowledge the generous support from our Media Lab sponsor Qualcomm. Specifically I would like to thank Vince, Steve Lisi and Chong Lee.

Finally, I would like to thank my dear friends Natan Linder and Rony Kubat for fun and educational times, Miki and my family for all the support they have given me during my studies.
1 Introduction

Proximal Connectivity Trend and Challenges

There is a clear trend of mobile devices that include proximal connectivity interfaces. These communication interfaces improve the user experience when communicating with other local devices and with the Internet.

This trend started with the introduction of Bluetooth for hands-free interaction with mobile phones and simple data transfer of user contact information or image files. Since then, additional Bluetooth profiles were included in consumer devices (though many, such as the PAN profile, are still not available in all devices).

Wi-Fi has been the cornerstone of this trend. The availability of Wi-Fi in consumer devices has opened the closed gardens of the mobile operators' network and enabled every device with an affordable, easy to use connection to the web. For example in the past when a user wanted to download a ringtone he had to access his operator portal only and could not download any song freely from the web. Users now can now use VOIP, chat, text messaging and other valuable services that may compete directly with the mobile operator services.

Fig 1 Proximal Networking major technologies

This trend is even expanding with Wi-Fi Ad-Hoc available in every iOS device as integral part of the operating system, and available in some
experimental Android devices. Wi-Fi tethering is now available on Android devices and on jail-broken iOS devices. These technologies allow more devices to connect wirelessly than ever before.

New emerging standards for proximal connectivity are recently implemented in latest versions of mobile operating systems such as WiFi Direct\(^2\) that potentially allows a direct connection between every two mobile devices without the need of an infrastructure Wi-Fi access point. WiFi Miracast\(^3\) enables sharing streaming media and sharing screens across devices that support this new standard.

Other standards are emerging that are not part of the WiFi / Bluetooth ecosystem. NFC\(^4\) (near field communication) is a new standard that functions as tap-like card swipe or very short-range communication. Android OS adapted this standard on many devices. Products such as Google Wallet and Android Beam use this capability. Another proximal discovery and connectivity standard, LTE Direct\(^5\), is still under development within the 3GPP standard organization and will hopefully complete the range for networks available with its ability to establish cellular range P2P connections.

These technologies commonly provide the ability to discover, connect and transfer data. They are not focused on making a common interface for such features and providing higher-level services such as network synchronization, grouping and protocols for coordination and collaboration. This thesis contributes some of these much needed services on top of the basic connectivity provided by each technology, including an extensible unified API that covers connectivity over both WiFi and Bluetooth.

**Opportunistic Proximal Wireless Networks**

Proximal networks are comprised of devices that are linked via proximal wireless links. These networks are commonly implemented by, one or multiple of, the connectivity standards described above and primarily linked via P2P links. These networks are sometimes referred as Mobile Ad-Hoc Wireless Networks\(^6\) (MANET), the major difference is that proximal wireless networks do not require an infrastructure free connectivity and can rely on the existence of some infrastructure. The proximal nature of these networks is derived from the connectivity standards as each dictate a proximity requirement to successfully establish communication between the network nodes.

Opportunistic networks\(^7\) are created around the context of a specific opportunity. Examples of opportunistic context are: a shared event that people engage with by taking pictures; chatting or sharing files between different people with common content interest profiles that are present at the same location.

Opportunistic networks based on proximal links are good candidates to address the limitations of content sharing between uncoordinated users. These networks can improve the user experience by using proximal wireless links with higher bandwidth, instead of mobile data.
infrastructure, and reduce the cost associated with mobile data when sharing contents.

These opportunistic networks are pervasive in their nature as network nodes can join or leave the opportunistic context, nodes mobility and batteries can cause disconnection at any given time. The ubiquity of mobile phones their embedded sensors, their social applications and their ability to support multiple communications interfaces make them excellent candidates for creating opportunistic proximal wireless networks.

The Social Challenge

The “World of Facebook” is a term sometimes used to describe how the world looks like from the Facebook® perspective. In this world, we upload images and share statuses with our friends that are out of our immediate context. Facebook has introduced a timeline component that tries to provide time context to shared media, although the upload time not the actual time and physical location of the event is usually recorded. New and complementing services to Facebook provide users with tools for social discovery. These tools help them discover who is around them so they can get connected, interact with each other and “friends” each other in mainstream social services such as Facebook or LinkedIn.

Compared to individuals who do not use Social Network Services (SNS), users of SNS are exposed to more people and to more activities. Yet, interaction through the “World of Facebook” is limited only to people who are connected to each other a priori through the social graph. People who happen to be at the place at the same time, but are not connected to each other on Facebook, will not be able to share their experiences through the Facebook platform. Moreover, while connections on SNS must be set up in advance for users to share experiences, the actual sharing occurs post factum: a user first takes a picture or a video clip, and then shares it with people he or she already knows.

First we should identify and acknowledge that a centralized SNS (that requires all the people to registered and connected to the service in advance) cannot satisfy some of the social opportunities and sharing needs that are local, proximal, immediate and opportunistic between people who share the same social context but are not connected to the same services and may not know each other.

The challenge is therefore to build tools to enable an a priori model of sharing. In this new model sharing is the default and can happen spontaneously among people who do not necessarily know each other. This new model should not be require its users to use the same social services, since they share the same social context and can benefit and enjoy such spontaneous interactions. Another challenge is in extending the sharing model to a collaboration model. Collaboration is possible by providing tools that allow people to engage in a synergetic co-creation of digital media.
This thesis addresses both these challenges and tries to provide a first step in realizing a solution. Addressing the first challenge by enabling a sharing model that is a priori, opportunistic, local and proximal without the dependency on a specific SNS. This principle is demonstrated by the CoCam app for proximally sharing pictures. Addressing the second challenge of collaboration, co-creation and engagement is demonstrated by the CoSync app that allow users to collaboratively create multi-viewpoint videos.

Hopefully this thesis work will facilitate the creation of future interactions in our viral spaces. The technical elements of the framework are described in chapter three and the applicative examples and use cases are described in chapter four.

**Media Creation, Sharing and Bandwidth Challenge**

Parallel to the increasing popularity of content sharing services, smartphones and tablet devices have massively increased their penetration. According to the IDC Worldwide Mobile Phone Tracker, a total of 491 million devices were shipped worldwide in the 2011 fiscal year alone. Smartphones and tablets equipped with high quality cameras, sensors and communication interfaces, enable millions to create rich media at anytime, anywhere, and share it with everyone. Smartphone users are now always connected and capable of making use of rich media services during actual events, share and consume content in real-time.

Sharing user-generated media on various services using the cellular network has an associated cost and burden on the cellular infrastructure. According to "An Empirical Study on 3G Network Capacity and Performance", in conditions of fully loaded cellular network, latency for 3G data services can increase beyond one second, and unpredictable behavior, such as dropped data calls, is exhibited. These limitations may not be an issue for basic email or web browsing, but may heavily impact the user experience of streaming rich media contents in real-time. User experience is very important, but 3G networks are not good enough.

Today users primarily use Internet based social services for sharing social interactions. The social challenge presented in the previous section is joined with a bandwidth and cost limitations on users that wish to create and share media using their mobile phones.

As more devices penetrate the market and more users use these devices to create and share media across services the demand for network bandwidth increases. If the network providers could not meet this demand by supplying more bandwidth and increasing the capacity of the network, the impact on user experience will be even greater.
This thesis explores one possible approach to meet this challenge by using proximal links that are free, high speed and battery efficient means of communication. This thesis provides a framework that is able to share rich media without the latency and costs associated with consuming and sharing media off the Internet via mobile data links.

**The Connectivity Challenge**

Mobile devices have become a ubiquitous means of communication in the last decade. In many countries around the world mobile device penetration surpass 100%, each person has on average at least one mobile device\(^\text{13}\). The high penetration rate created a market for services and applications for mobile devices—in particular smartphones and tablets—of which many require constant Internet connectivity. Mobile users have therefore become accustomed to a 24/7 cellular network connection, anytime anywhere.
However, quality of service may differ significantly across locations. In many areas, customers of one mobile operator enjoy a great service while those of another experiences dropped calls and broken data sessions. Unfortunately, Mobile network operators are commonly competitors, they do not share infrastructure and host other operators’ devices transparently. A synergetic solution by the operators, namely roaming without roaming costs for cases were one operator can provide service to customers of another operator that is out of service, is not in sight.

The research community has been addressing the broader issue in the context of absent or failing infrastructure. In many cases, research focuses on the mobile device technology. Notable is the cognitive radio\textsuperscript{14}, and MANET\textsuperscript{15} research. Both designed a multi-hop device-to-device network that eventually routes messages through to the Internet or to the destination device.
Fig 4 Cognitive radio setup diagram (source Sciencedirect)

However, both MANET and cognitive radio are limited in their scalability, as a result of the complexity and inefficiency of their routing protocols. It is challenging to establish and maintain a multi-hop route when mobile devices are roaming or are turned off and on. Routing table updates are exponential in the number of devices in the network.

This limitation can be mitigated by the tethering approach applied in this thesis. The basic assumption is that all devices are only one hop away from the Internet and the routing device is sharing its Internet connection and only provides access to devices that are directly connected to it via proximal wireless links (i.e. one hop away). Although this may address the need to provide Internet access to a device near by, tethering cannot be infrastructure-free communication between devices. This approach depends on infrastructure to be one hop away.

Tethering, and specifically WiFi tethering, provides a compelling way for users to overcome the lack of synergy between operators. It also provides access to each other when one user is out of network coverage and the other can provide network access. This feature, alongside with other proximal wireless discovery technologies can facilitate discovery, connection and interaction in our immediate spaces and allow for interactions that were not possible until now. The technical and applicative elements are described in details in chapters three and four.

This thesis work will present a framework that can facilitate such synergy between users, overcome some of the connectivity and Internet access limitations using a WiFi tethering approach. This thesis demonstrates this with a mobile application called Airmobs that is released in conjunction to this thesis work.
Business Model Challenges

This thesis touches upon some business ecosystem and business model challenges. In the United States, mobile operators restrict the use of the mobile tethering feature on mobile phones. This feature turns the mobile phone into a Wi-Fi hotspot that other users and devices can connect to. This hotspot uses the mobile data Internet link as its uplink and routes the connected users though this link to the Internet. Many operators demand additional payment to allow the user to use this feature with their already-paid-for mobile data plan, just for allowing (tethering) other devices though the user’s cellular connection, even if these are his or her personal devices.

The Airmobs prototype is a part of this thesis work. This prototype, reported by the NewScientist18 and Bloomberg Businessweek19, contributed to a discussion about the ownership of data plan bits. With Airmobs, a user can allow other users to get Internet access though his or her device’s 3G/4G interface and receive credit that will allow him to tether through other people’s devices when access the Internet is needed though unavailable due to a lack of network coverage. The notion that “A bit is a bit” is the center of the conflict with mobile operators that view the act of tethering to other users as a resale of their services, which is strictly prohibited, meaning that a bit generated on the device is legitimate and a bit routed though the device is not i.e. “a bit is not a bit”. This notion conflicts with the approach of this framework and prototype app that claims that a person is owner of his data plan and can decide who can be routed though. In this approach “a bit is a bit” and there is no difference between a bits originated on the user’s device or routed though, as long as the user allows for it.

This thesis suggests that connectivity can be improved, costs reduced, and user experience improved with the use of proximal links and tethering. But these benefits may conflict with the business model of some of the mobile operators.

Other challenges such as privacy and security are addressed, though are not the focus of this thesis. Some features of privacy and security are implemented in the CoSync framework.

Framework Prototypes and Demonstrations

We have developed seven applications that demonstrate and explore the space of proximal collaborative and synchronized mobile devices. The insights we derived from these prototypes can stir discussion on a set of design principles for the CoSync framework.

We provide a brief description of the prototypes below.

Airmobs

Airmobs creates a local mobile community to allow free sharing of Internet access among diverse 3G and 4G data accounts. We created an app where anyone can advertise available bits (bandwidth) and their
willingness to let other Airmob members tether through them to the Internet. Users might do this if they are near the end of their data plan and need additional data, or willing to share since they have data surplus in their plan and they want to use it before it expires. A website tracks the evolution of the community and tracks the biggest donators and users of the system. To date, this app works on Android devices. It is designed to be open and community-based. We may experiment with market credits for sharing airtime and other features.

![Fig 5 Airmobs users (provider and customer)](image)

### CoCam

CoCam is a self-organizing network for real-time camera image collaboration. Like all camera apps, it works by using the familiar point and shoot interaction. Once an image is shot, CoCam automatically joins other media creators into a network of collaborators. The system handles network discovery, creation, grouping, joining, and leaving automatically in the background, letting users focus on participation in the event. We use CoSync framework and a 3G-negotiation service to create a network for real-time media sharing. CoCam also provides multiple views that make the media experience more exciting—such as appearing to be in multiple places at the same time. The media is immediately distributed and replicated in multiple peers, thus if a camera phone is confiscated, other users have copies of the images.

![Fig 6 CoCam demo sharing images in real-time](image)

### CoSync video

Viewing the same scene from multiple viewpoints is an interesting and fun way to consume video media. This is possible using professional video software with manual editing or a real-time director. CoSync Video is a demo using six independent video sources that are synchronized and capture the same scene at the same time. The viewing station allows switching between each camera without editing, continuing almost exactly at the same frame.
CoSync audio

In many cases, we want to use the speakerphone to play a tune, but the speaker was designed for hands free conversation not as a loud speaker for an audio system. In addition, the physical and battery limitations of the mobile device impact the quality and volume of audio being produced by the speakerphone. CoSync Audio synchronizes multiple speakers to play together the same tune at the same time, overcoming volume limitations and improving the audio quality and surround effects.

CoSync Camera Flash

The mobile phone camera is immediately available to take pictures, especially when the picture-taking opportunity was not planned in advance. Many social picture opportunities happen indoors in poor light conditions such as a party or a dark room. The mobile phone camera performance at low light is poor and the camera flash in some occasion may make quality even worse; as a result we lose precious moments. CoSync Camera Flash allows the use of multiple devices as synchronized remote wireless flashes that are triggered by the device taking the picture. The resulting images have less motion-blur, are not blinding to the subject and subjectively have more dramatic and accurate colors.

LipSync

This application demonstrates two technologies developed in the Viral Spaces group: CoSync and VRCodes. It was developed to show the capabilities of VRCodes to detect a human invisible code off a screen using a mobile phone. The demo shows an interactive broadcast of famous monologues available in many different languages to multiple users at the same time. Users direct their smartphones to “tune in” to their desired language, either audio or closed captioning, by simply pointing their phones at the relevant part of the screen. The thesis framework contributed the precise synchronization between the video
and the audio streams to create a seamless experience, where the user's natural motions give voice to moving lips.

Fig 9 Syncing personal device audio to public screen, by pixels.io and VRcodes

CoSync Camcorder Synched Microphones

Professional video capture uses personal microphones (often wireless) that capture the audio of a remote person, or a dedicated soundman that follows the scene with a boom microphone. When shooting a video with a mobile device from a few feet away, it is very difficult to get high quality audio. CoSync allows synchronized devices to function as remote microphones and a display system that presents the video and can switch from one audio source to the other without editing. This allows tuning into different concurrent conversations captured in the same video.

Contributions

This thesis makes contributions in the design and implementation of opportunistic proximal collaborative and synchronized applications. The described CoSync framework is being used beyond the scope of this thesis by other projects in the Media Lab such as the Mobile P2P project conducted in Information Ecology research group.

Mobile P2P framework

- A P2P library for network creation, discovery, connection and peer connections over proximal links supporting Bluetooth and WiFi with a unified API.

Proximal Collaboration

- Coordinating proximal groups, organizing peers into a network and switching between groups opportunistically.

Synchronization and Action API

- Synchronization API.
- Remote Synchronized Action API.

Reference Applications

- Design and implementation of a suite of novel mobile synchronized collaborative applications.
This work can stimulate a wider discussion about synergetic value of mobile device collaboration in proximal spaces, to improve connectivity, media creation and consumption, user experience, expression and creativity. This work is one point in the path to realize the vision of viral spaces and device ecologies in general, by enabling all the mobile devices around us to be used as if they are one omnipotent device at the users fingertips.

Lessons Learned

In this thesis we have learned that there are many user level benefits of using proximal networking as a basis for application level features, beyond connectivity only. We have learned that it is possible to augment one-device capabilities by using other devices nearby with a common synchronized collaborative framework, as demonstrated by the remote camera flash. We learned that new features stem from such capabilities may interest the users and allow new kinds of creativity and creation.

Proximal networking in general and the CoSync framework in particular are a cheap, quick and simple way to examine, experiment and prototype user level collaboration, co-creation and media features. These features, if found useful, may later influence the future design of systems or become part of the infrastructure or provided by future versions of operating systems. For example if exact synchronization between devices becomes a key enabler of a new kind of video audio distributed system, after prototyping it with CoSync, it may become part of the cellular network software stack and APIs. Adding such a feature is a process that may be complex and expensive to incorporate in the cellular standards without validating user interest and need first.

We have learned that an overall approach is need not a specific single feature is king. This framework includes sub-domains of proximal network technology that at first seemed disjoint. We found it critical to have all elements at application level; connectivity, discovery, sharing collaboration, synchronization enabled with the relevant protocols to take actions (same action or different actions) at a planned and controllable (nearly) at real-time.

We have learned that the mobile infrastructure and algorithms we examined are still lacking the accuracy that is needed in order to provide a stable user experience. For example in some cases we can achieve sync accuracy that can support multiple devices playing music together and at times the same system and algorithm result in a sync gap that completely destroys the user experience as the audio emitted by different proximal devices is not synced. We should find ways to improve stability and best performance before releasing such features to a wider use.

There may be other user considerations that were not examined such as security, privacy and ownership of media created jointly. It may be important to look at these issues as follow-up work. Another technical aspect that should be investigated and incorporated into future
versions of such framework is proximal localization. Investigate how the ability to accurately localize a proximal device influences user level features and the possible applications. A future framework that is opportunistic collaborative synchronized can also localized.

**Thesis Outline**

This thesis is divided to five chapters.

**Chapter 1: Introduction** - provides a high level review of the current challenges in proximal collaborative synchronized mobile application this thesis addresses and a brief overview of the CoSync project.

**Chapter 2: Background and Related Work** - provides motivation for this thesis. It covers the relevant theory and the previous work in mobile collaboration and relevant background for the applications of the CoSync project.

**Chapter 3: CoSync** - describes the software framework features, design and API. Covers proximal connectivity sharing feature, real-time media-sharing feature and Synchronization and remote action protocols and implementation.

**Chapter 4: Applications** - description of the CoSync applications suite including UX and flow. Proximal connectivity demonstrated by Airmobs. Opportunistic grouping and real-time media sharing demonstrated by CoCam. CoSync Video, CoSync Audio and CoSync Flash demonstrate synchronization and remote action features.

**Chapter 5: Evaluation** summarizes the evaluation of the CoSync framework, user studies, technical evaluation and lessons learned.
2 Background and Related Work

CoSync Framework Domains

This framework tries to address the major elements needed to develop a mobile application that targets media creation, social collaboration and interaction in an opportunistic proximal space.

Since this framework domain is extensive we have included tools and modules that address the different sub-domains of the framework. The demo applications although utilize the framework on demonstrating one of the major capabilities of the framework.

Fig 10 - Framework - Proximal Networking sub-domains

The relevant background for each domain is following below.

Connectivity Sharing

Available Solutions

Connectivity sharing has been experimented with and developed in research projects and commercial products. The following is a quick survey of these services.
Fixed Commercial Connectivity Sharing Solutions

Fon\textsuperscript{20} is a service by British Telecom (BT), partnered with other worldwide telecom operators, that aims to provide the world's first global WiFi service, crowdsourced from WiFi hotspots provided to Fon users. As a member of the Fon community, customers agree to share a little bit of WiFi at home, and get free roaming at Fon Spots worldwide in return. As a connectivity-sharing product, Fon is considered to be very successful; it has over five million Hot Spots across the globe, mainly in England.

XFINITY WIFI\textsuperscript{21} and AT&T\textsuperscript{22} provide access to their users to thousands of WiFi hotspots, leveraging their respective networks to provide nationwide access to their customers when they are away from home. This is not a true community but a community-like feature that is common to all the service provider users.
Whisher network has been created by people sharing their WiFi connection and provides its users with a software application that helps them get free WiFi access all around the world.

**Fixed Open Connectivity Sharing Solutions**

Guifi.net is a telecommunications network. It is open, free and neutral because it is built through a peer-to-peer agreement where everyone can join the network by providing a connection, and therefore, extend the network and increase connectivity.

There are many other examples for community wireless networks around the world. One of the most significant ones is the Australian community aggregated by wireless.org.au

LifeNet is an infrastructure-free wireless network based on openWRT to provide a local mesh. This is a fixed network since it is primarily based on a modified version of a fixed access point. LifeNet is opensource, but since it is targeting infrastructure-free applications, it is not commonly available.

Shareair.net and easywifi are services that locate and connect to WiFi networks that are scattered worldwide. These services aggregate and update information about these networks and facilitate the connection by providing a pc utility and a mobile app that search these networks and perform the actual connection.

**Commercial Mobile Connectivity Sharing Solutions**

Open Garden is a commercial solution for crowdsourcing connectivity. This service is functioning as a tethering application and claims the ability to augment bandwidth by utilizing multiple links. The unique structure of open garden is that they aim to use links provided by parent WiFi networks as well as mobile tethering connections from near by mobile devices. Open garden is designed as a commercial solution and not as a community and there is no social incentive for participating in this network.

Karma is a mobile hotspot device designed for personal connectivity. This device has a special community feature where the owner shares part of the bandwidth with other karma users. The more data a user shares the more he earns. This is a very similar model to Airmobs but requires an additional dedicated device to function as the mobile Wi-Fi hotspot.

**Open Mobile Connectivity Sharing Solutions**

This thesis demonstrates Airmobs, an app that aims to provide an open, community based, connectivity sharing mobile application. There is other research and prior work that examine various aspects of mobile connectivity sharing, described below.
Research prior work

The roofnet\(^{30}\) project at MIT and later Meraki\(^{31}\) created a multi-hop mesh WiFi network where some of the nodes acted as gateways, sharing their Internet connections. Users of the roofnet could simply install hardware and the existing network nodes would collaborate to route traffic to the Internet. Meraki has extended this concept to include a community of users using their cloud managed WiFi access points that can share their Internet connections.

The Link-alike\(^{32}\) project at MIT has examined the fundamental requirements of a system that aggregates upstream broadband connections in a neighborhood using wireless communication between homes, and proposed a scheme to address the challenges of such aggregated wireless environment with multiple Internet connections shared.

"Wireless Community Networks: Public Assets for 21st Century Society" by Elaine Lawrence et al.\(^{33}\) reports a survey conducted to learn about the motivations, efforts and participation in Wireless Community Networks in Greece, Australia and US. The research concludes that the community members were motivated by the concept of wireless community network and acted on behalf of their common interests. The research confirms that the community members were indeed realizing the notion of collective action as defined by Olson Mancur\(^{34}\).

According to Olson the logic of collective action argues that individuals in any group attempting collective action will have incentives to "free ride" on the efforts of others if the group is working to provide public goods. Individuals will not "free ride" in groups that provide benefits only to active participants.

Further Olson explains that pure public goods are goods, which are non-excludable and non-rivalrous (one person’s consumption of the good does not affect another’s, nor vice-versa). Hence, without selective incentives to motivate participation, collective action is unlikely to occur even when large groups of people with common interests exist.

Mobile WiFi connection sharing is a public good in the sense that it is non-excludable. In the sense that a person who downloads the app and provides a shared link does not exclude another person from downloading the same app and using a shared link or sharing his own link. But it is excludable in the sense that the user sharing his connectivity cannot consume the data consumed by the shared link client. Another important aspect is that creates exclusion is the fact that the battery is consumed as a result of sharing a link.

From the literature we can understand that the case of mobile connectivity sharing may be more complex technically and economically than fixed connectivity sharing applications. This is demonstrated in many community initiatives and commercial products that are now available globally.
"CrowdMAC: A Crowdsourcing System for Mobile Access" by Ngoc Do et al (2012) addresses the challenge of mobile data sharing by means of crowdsourcing connectivity. This research focuses on (1) the admission control algorithm of the access point node (sharing entity), (2) service selection by the mobile device with different fees associated with multiple services (using entity), and (3) the mobility and uncertainty of such links. This research was simulated on Qualinet system and a test bed of multiple Android smartphones.

CrowdMAC also suggests a cost model that presents the associated cost of using possible shared links by a consumer device (a mobile device using the shared link). There is an assumption that some cost model can exist, and is integrated within the admission control feature.

Another research that developed an economical model for determining mobile tethering cost/price is "Price-based Tethering for Cooperative Networking" by Jisub Lee at al. This research constructed a mobile tethering-based cooperative network. It required collecting every user's information about cellular traffic price, traffic demand and traffic capacity (i.e. bandwidth constraint). After that, each user's cellular traffic demand should be allocated to other users at a specific pattern that can minimize the total cellular cost. According to the optimal traffic load, the optimal tethering market price that can maximize the total utility of all users was determined.

Why a user should cooperate and share the paid data connection? Why should he allow others to exploit his resources and risk his privacy? These are the major questions commonly asked by researchers, engineers and economists researching and developing mobile connectivity. The works described above take a clear economical approach and present several different economic models in attempt to answer this question and provide motivation to share mobile data connectivity. The social aspect of mobile connectivity sharing is less explored; this thesis, and the work described below start to look at the social motivations and the community action that may result from social incentive to cooperate.

The recent work "Cooperative Networks: The Mobile Tethering Game" by Mihai Constantinescu et al (2012) takes a slightly different approach. It proposes an analysis of cooperation for data sharing and the behavioral aspects involved in the process of decision making. This analysis applies game theory principles and models in order to inquire what makes the cooperation work, and what are the economical implications required to build a cooperative network.

The game theory used to analyze this strategic situation is defined as the mobile tethering game. In this game the players have two strategies: cooperate (share) or defect (not share). The goal this work is to investigate what the user's payoffs may be and the dependency of the payoffs on the user's preferences.

This game considers influential factors impacting the payoffs such as security and quality of service, and cost factors such as energy consumption. It also considers benefit factors such as reduced fees, or
the accomplishment of a goal that impacts the payoffs of the tethering game. Constantinescu et al. attempted to quantify the costs and benefits based on generic use cases and construct the payoff matrix of the mobile tethering game.

From this game we learned that players do not always play the dominant strategy equilibrium. Often they play the cooperative strategy pair and are not as rational as game theory assumes, or may not always base their actions on self-interests. Altruism can prevail, but players do not spontaneously choose the rational solutions of the game and rather tend to employ heuristic rules.

This thesis is based on the conclusion of the previous research work, focusing on the social incentive for sharing links. The starting point of connectivity sharing in this work is based on the conclusion that common action is possible and was observed in fixed wireless community networks around the world.

Furthermore, this work acknowledges the difference and complexity introduced by the mobile environment and attempts to address these issues. We emphasize common action and discourage free riders. We assume that motivation for sharing bandwidth is increased by reflecting the user his community standing, and providing a visual feedback for his community credit.

We also assume that the future value of bartering excess data with potential ability to use other community members' data when in need of Internet access will motivate downloading and installing the app.

Lastly, we assume that if the system does not require user involvement it will reduce friction and promote automatic behavior, and as a result, will facilitate sharing network links. This last feature of automatic background activation of the connectivity-sharing feature is not only a user experience enhancement; it is a game changing design feature. As previous work defined the mobile tethering game as one that has two possible strategies (share or defect) this feature minimizes or even eliminates the ability of a user to defect since the system is deciding when and whether to activate the tethering interface. Hence this thesis builds on previous research and tries to look forward to how we can build better tools for opportunistic mobile communities.

**Media Sharing**

Social-aware Content Sharing Demo describes a file sharing application over an opportunistic network based on user profile information. Upon sharing content, it examines the user's sharing interests (genre of content the user intends to share), habits, social contacts and mobility patterns. This particular application is built on top of Haggle, which is a platform for content-centric data distribution on opportunistic networks. Though it focuses on sharing existing contents in a delay-tolerant fashion, it demonstrates the potential benefits of utilizing opportunistic networks for sharing files and user-generated content in a seamless fashion.
CallTheWeb\textsuperscript{41} describes an experiment designed to understand technical feasibility and user perception of mobile video interaction. The experiment results revealed that the 3G network (as of 2008) was not satisfactory for real-time video sharing from the user's perspective, largely due to the bandwidth limitations. The results also indicated that general user preference for Wi-Fi over 3G for real-time content sharing. Moreover, users in the experiment reported that they would prefer the sharing feature integrated within the camera application, and desired the categorization of contents based on events in order to ease the search process. As this experiment demonstrates, it is clear that relying on the 3G network for real-time media content sharing is not entirely feasible from the user-experience perspective.

BlueTorrent\textsuperscript{42} explores a way of sharing contents among proximal nodes using Bluetooth as a network interface. It proposes a file swarming technique, which enables it to restructure the original content from small blocks retrieved from multiple nodes. While Bluetooth consumes less energy than Wi-Fi, and is more suited specifically for P2P communication, the potential bandwidth limitations are likely to become a bottleneck upon real-time content distribution of rich media.

MOVi\textsuperscript{43} is a framework that exploits the opportunistic use of P2P links for optimizing on-demand video streams within a mobile ad-hoc network. MOVi is based on the assumptions that other nodes within the ad-hoc network may become "seeds" for providing portions of the same video feed provided by the centralized server. Upon enabling the on-demand stream function, MOVi proposes a concept of centralized status-tracking and control with decentralized data distribution. As a result of simulations, it has demonstrated that the method provides shorter video startup delays, and increases the capacity of simultaneous reception by multiple nodes.

**Synchronization and Remote action**

**Network Time Protocols**

A concise summary of the state of the art of network time for synchronization is provided in the ACM Communications article "Toward Higher Precision"\textsuperscript{44}.

**NTP** (Network Time Protocol RFC 1059) is the standard protocol for timekeeping in the Internet, with many servers deployed worldwide. This algorithm uses a real-time source as time input to the server that is later connected by clients' wishing to adjust their internal clock to the server's reference clock. In many cases the time source is either accurate GPS receiver or an atomic clock.

The advantage of this method is in its simplicity, as it is supported by most operating systems and has implementations that are freely available. The protocol supports both UDP and TCP transport and can be implemented in the local network and the Internet. Internet implementations are more common.
The disadvantages of this method arise from the assumptions of the synchronization algorithm. It assumes the network delays are symmetric and the upstream message has the same propagation delay as the downstream message. In many home networks the upstream enjoys less bandwidth, hence from time to time greater delays than the downstream, this contributes to synchronization gaps between the master (server) and the slave (client). The algorithms may also suffer from fluctuations in the network and provide different time offset results at different sync attempts to the same server. Lastly if the server is busy (highly unlikely since this is a very lightweight protocol) delays and inaccuracy can increase.

The main disadvantage of the NTP sync method is that applications depend on connection to the Internet and must agree in advance on using the same NTP server, as using different servers generate big differences in time sync.

PTP (Precision Time Protocol IEEE standard 1588 version 1 and 1588-2008 version 2) is a time keeping synchronization algorithm that was designed initially for LAN and physically connected devices. This algorithm later evolved to support modern LAN topologies, in many cases still wired networks with specific hardware support. In order to achieve the precision requirements of this algorithm specific modifications of the network hardware is needed, as in NTP the algorithm is based on a request and reply messages, unlike NTP additional message is sent and it is initiated by the master device, the timekeeper and not the client.

The hardware-based PTP changes the timestamps on the network packet as it leaves the NIC hardware. This is supported only in a few wired NICs and does not exist for regular standard computer hardware. For this reason a hardware-assisted PTP algorithm was developed. In this algorithm instead of modifying the timestamp within the packet by the hardware of the NIC, the NIC software driver would report the timestamp of the packet leaving the NIC. This allowed an improvement of the basic PTP protocol usually referred as software assisted PTP. This version of PTP is implemented by adding two more update messages that send an update message with the correct timestamp of the original message.

The advantage of this algorithm is that it is very suitable for local and proximal networks as the master broadcast its existence and can be selected dynamically.

The disadvantage of these solutions for achieving the synchronization precision requirement of the CoSync framework is that they are impossible to implement in mobile device environment as mobile hardware does not supporting PTP and the driver does not notify when a specific packet has left the NIC.

For the purpose of this framework I have developed two versions of PTP. The first implementation is a straightforward PTP message exchange that begins with the master broadcast. The second version,
called UPTP, attempts to reproduce the software assisted PTP without a driver support. This version assumed that after the send function, each packet has left the device and its time can be used as the updated timestamp in the following update message.

GPS Time provided by the Location API’s of the mobile GPS device can potentially provide the same accurate time for devices under the same satellite coverage.

The advantage of this method is in its simplicity, as time is provided in each GPS update and can be read without special interpretation.

The limitations of this method are (1) proximal devices should have access to the GPS signal (usually outdoors), (2) Setup time is very slow until a reliable GPS update is received, and (3) the mobile API caches GPS requests, and the time reported to the application can be obsolete (although in latest 4.2.2 version of the Android OS, a new method is provided to query the “age” of the GPS update in the system).

In this framework I have evaluated the use of GPS time synchronization by using the Android OS API provided by the classes LocationManager, LocationListener and Location.getTime() method.

BCN is a beacon based time synchronization algorithm I have designed to be evaluated against the above standard algorithms. The major difference of this algorithm from all other LAN and WAN based algorithms is that it is a simple single message protocol with only the master clock broadcasting its timestamp and a delta from last broadcast.

This algorithm assumes that the difference between the clocks encapsulate two values. One is the real difference between the clocks and the message propagation delays. Both these values are indistinguishable and are stored in the offset variable.

The second assumption is that the master is sending a periodical broadcast; hence the next time a message arrives can be predicted.

The third assumption is that the network delay varies from messages that have long delays and messages with very short delays. This algorithm is designed to detect the messages that have shorter delays than expected and adjust the difference between the slave clock and the master clock.

Since this algorithm adjusts its offset every time a shorter delay occurs, it will potentially converge a synchronization delay that is as long as the shortest end-to-end delay. If such delay is short enough it may perform according to the framework synchronization requirements (that is evaluated in chapter 5). In this case the advantage would be that it is a simple algorithm with single periodic message.

The algorithm pseudo code:

1. receive message
2. current \leq \text{local timestamp} \\
3. \text{delta} \leq \text{message.delta} \\
4. \text{if first message} \\
   \text{offset} = \text{current} - \text{message.timestamp} \\
   \text{last} = \text{current} \\
5. \text{else if last + delta} > \text{current} \\
   // \text{arrived before expected hence we need to correct} \\
   \text{correction} = \text{last} + \text{delta} - \text{current} \\
   \text{offset} = \text{offset} - \text{correction} \\
   \text{last} = \text{current} \\
6. \text{else} \\
   \text{last} = \text{last} + \text{delta} \\

Other related work using synchronized time is presented by Shunshuke Kurumatani et al\textsuperscript{46} demonstrating the use of time synchronization for accurately estimating a synced action on different clients performed potentially at the same time based. This research is based on server side calculations.

3 The Framework

The CoSync framework was developed through several iterations and prototypes to better understand the requirements, limitations and challenges of the mobile device environment.

In this chapter we describe the technical implementation details for the CoSync framework developed in the course of this thesis work. We describe the framework sub-systems and their technical details.

While this chapter focuses on the technical details of the CoSync software framework, it sets the ground for the next chapter, which deals with applications of the CoSync framework in different mobile scenarios. The two chapters are closely related, and together they form a complete description of the design, implementation and demonstration of the CoSync framework.
High Level Software Requirements

The software requirements for the CoSync framework are briefly described.

Embedded
- The entire software stack should be able to run on embedded boards, especially mobile phones.
- The entire code base should have a small footprint so it fits the requirements of a mobile app
- Low runtime memory demands

Layered and Modular
- The system should be developed in a modular fashion, simple to integrate and use

Extensible / Portable
- Provide interfaces to extend the system in its respective layers and replace or add implementations to its core elements
- Support portability of the codebase across different mobile devices and personal computers

Connected
- Should be implemented with multiple network interfaces to accommodate different applications needs.
- Support proximal network creation, peer discovery and connection.
- Support data transmission between peers.

Software Design Principles

The framework design should be modular; its implementation should be independent of operating system. The software design should apply proper separations between protocols and communications interfaces, as well as between application layer and utilities. The framework should be scalable and parallelizable.

Modularity, Separation and Polymorphism

Modularity: The framework is designed to allow a user to use all or portions of the framework elements without dependency on all other elements. For example: a user that requires just synchronization can use the SyncService class that provides the system with the ability to sync without the need for SyncManager class that is responsible for sending and receiving synced actions.
Implementation independence: The framework allows multiple implementations of main code elements to coexist by defining interfaces between the core elements. For example, SyncService is independent of the implementation of the actual SyncProtocol that is used to negotiate the sync information and conclude the sync state.

Polymorphism and reuse: The framework provides multiple implementations of some of the core elements to allow modularity and reuse of higher-level objects. For example, Connection is the interface class to any method of communication and is implemented over Bluetooth and WiFi classes with specific implementations that allow broadcast as well as unicast.

OS Portability
OS specific features are abstracted. These abstract classes facilitate portability and are located in the os.port package. Classes that implement these abstract classes for a specific OS are also located in a sub package os.port.<os_name>. For example, the Android OS implementation is located at os.port.android. Applications should use the relevant classes according to their specific OS implementation.

Parallelism
The framework uses multiple threads for its services and communications and is safe to be used by multiple application threads. This allows a simpler API and more efficient use of system resources. For example, each ProtocolService class holds two threads for receiving and sending messages for its specific protocol, and each BTConnection class holds a reader thread to allow quick response to incoming data over Bluetooth.
Software Architecture

System Block Diagram:

![System Block Diagram]

Features Diagram:

![Features Diagram]

Components Overview:

The CoSync framework is designed as a layered architecture, with the following layers and components:

- **Applications** - front-end opportunistic collaborative and synchronized mobile applications
- **Core Services** - framework components, interfacing with lower level utilities exposing high level protocols and service managers
Utilities – wrapping OS capabilities and providing convenient utilities for accessing the hardware.

The following sections will describe specific core elements of the CoSync framework, their design and implementation details.

Protocols Architecture

Design

The protocol architecture is designed around a generic interface that represents a generalized protocol class for handling periodic messages and processing incoming messages. This interface is used by services that are executing a protocol. These services provide an interface to the protocol features to higher level APIs and applications that wish to use these protocols.

Classes

Protocol – Interface representing a protocol that can send periodic messages and process messages. The protocol interface includes timeout support for protocols that have timeout requirements for incoming messages. The protocol interface is independent of actual implementation of the protocol messages and the implementation of the communication channel.

SyncProtocol – This abstract class implements a protocol that is responsible for sending and processing synchronization messages. This
class is holding the sync state, clock offset from the master and a
listener for notifying a sync state change.

**PTPSyncProtocol** – This class implements a specific sync protocol
similar to PTP (or NTP).

**BeaconSyncProtocol** – This class implements a specific sync
protocol syncing to a half duplex fixed interval beacon.

**DiscoveryProtocol** – This class implements a broadcast based
discovery protocol based on publishing request for network peers to
update their address information

**SyncedActionProtocol** – This class implements a protocol for
requesting an action and sending its parameters to the relevant remote
devices for a synced execution at a specific time in the future.

### Connections Architecture

![Connections Architecture Diagram]

**Design**

The purpose of the connection interface is to provide the ability to
communicate with other peers, without dependency on the actual
implementation of the communication interface. This class supports
WiFi and Bluetooth as well as unicast and broadcast messages.

### Classes

**Connection** – interface for sending and receiving messages providing
local peer information and remote peer info for this specific connection.
This interface also supports timeout for blocking connections.
**UDPConnection** – A connection that supports sending and receiving unicast UDP messages. This class is usually used for UDP based protocols over WiFi connections.

**UDPBroadcastConnection** – A UDP connection that supports sending and receiving broadcast messages.

**BTConnection** – A connection that is implemented over Bluetooth RFCOMM interface to a specific pre-connected Bluetooth device.

**ConnectionReader** – A thread class that is responsible for reading from the specific Bluetooth connection it is attached to for minimizing the delays in reading Bluetooth data.

**BTBroadcastConnection** – A class responsible for maintaining all live BTConnections, broadcasting a message to all connections, and blocking until receiving a message from one of the connections.

**Services Architecture**

![Services Architecture Diagram]

**Design**

The purpose of the service architecture is to simplify the creation and execution of a service. The services interface with the application or higher level API’s and provide a specific functionality such as discovery service, sync service or remote activation service. CoSync services use a protocol to communicate with the same services hosted by peer devices. The services also associate using a protocol class and a connection class that provides the needed communication channel for the protocol messages. Although the connection is provided to the protocol by the service it is abstracted and only the application actually decides what is the appropriate communication interface to use.
Classes

ProtocolService – An abstract class responsible for handling a protocol over a connection and interfacing to higher level classes and the application layer.

SyncService – This class is responsible for running the sync protocol over a given connection. The sync service holds the sync state and notifies its clients using a state changed event interface. This class also includes utility functions to convert local time to synced time.

SyncManager – This class is responsible for running the synchronization action protocol. This class facilitates sending a remote action request to be executed by a specific peer at a specific synced time in the future. The action is sent along with action parameters. The action manager can send an action to a specific peer, a broadcast action to all peers, or an action plan i.e. different actions to different devices to be executed at the same time. This class holds a reference to an action handler object that is responsible to respond and execute incoming remote action requests.

DiscoveryService – This class is responsible for discovering peer information as peers join and leave the proximal network. This class uses a discovery protocol to request network nodes to provide it with relevant peer information that can be later used by the above services to communicate with these peers.

Service, Protocol, Connection

Design

The service, protocol, connection model is a design pattern that separates entities that function together to accomplish a specific application goal. This design pattern allows specialization of each of the
classes around a specific implementation of the capability needed. As a result, it is possible to replace the implementation of each of the classes without affecting the other entities in this model.

The entities in this model represent three different aspects of distributed applications: the service a runtime execution unit that interfaces the application and higher-level APIs; the protocol responsible for the specific messages passed between the peers to accomplish a goal; the connection the specific communication channel used to communicate the protocol messages executed by the service.

This model is reused by all CoSync framework services including discovery, sync and action protocols.

**Sync Architecture**

![Sync Architecture Diagram]

**Design**

This architecture is designed to simplify the implementation of new sync protocols and improving existing implementations without changing the higher-level services and action manager that interface the application layer.
Sync Application

Design

The SyncApplication architecture is designed to provide an easy way for applications to interface with the framework by aggregating all the needed services together. Since this structure is common to almost all the applications using CoSync, SyncApplication class allows the common code to be reused across different applications.

The framework also notifies applications about important state changes and incoming action requests by registering listeners that in case of relevant events, are used to communicate the event or state change to the application.
Sync Sequence Diagrams

Master Sync Sequence

Sequence Description:

A device that is in the MASTER sync state executes the following sequence. The SyncManager starts a SyncService that runs two threads: a sender and a receiver. According to the specification, the SyncProtocol class (in this example a PTPSyncProtocol) sends a periodic MASTER_SYNC message with a timestamp. The receiver thread receives reply messages from remote peers trying to synchronize to this master device as followers. The reply message sent by such as slave device is SLAVE_DELAY with the timestamp representing the time the slave has replied to the master. The protocol processes the SLAVE_DELAY message and sends a SLAVE_DELAY_RESPONSE message in return with a new master timestamp.

This sequence repeats on each periodic message sent according to the specific SyncProtocol period attribute, and conclude with the ability of slave devices to infer and calculate their clock offset from the master clock.

Slave Sync Sequence
Sequence Description:

This sequence is the same sequence from above but from the point of view of a slave device.

The slave device does not initiate any message, but waits for a master to broadcast a MASTER_SYNC message that triggers the process of syncing with that master. In case such a message does not arrive until a defined timeout, the slave device understands that there is no master on the network and changes its state to become a master and initiate MASTER_SYNC messages. When a slave receives and processes the MASTER_SYNC message, it sends a SLAVE_DELAY message with a new timestamp. Upon receiving a SLAVE_DELAY_RESPONSE message from the master, the slave changes its state to synced and updates its offset from the master clock.

Grouping and ClusterManager

Pre-association Discovery

Before devices can use proximal links to communicate with each other, they should be associated with the same network. As different devices advertise their proximal presence in multiple different ways, they should also agree on the advertisement parameters.

It is very common to have multiple WiFi networks at the same physical location. Since devices that wish to communicate proximally may be located in the same place but associated with different networks it is not possible to use common discovery protocols such as bonjour or multicast DNS.

One common option is to use Bluetooth service discovery. This option is excellent for many applications but suffers from a few critical
limitations. First, the effective Bluetooth range is substantially smaller than the effective WiFi range, so devices that can communicate over proximal WiFi links will not discover each other and will not communicate as a result of using Bluetooth as pre-association discovery method. The second critical limitation is that user active involvement is needed when a device wishes to become discoverable and a popup is prompt to the user. Another issue is that this mode is limited by a timeout. Both limitations are making it problematic to use Bluetooth in a background service. For example a service such as pre-association discovery that always runs in the background, or other services that wishes to communicate without explicit user involvement.

Another option is to use WiFi SSID broadcast and WiFi network scan for devices pre-association discovery. Devices are “discoverable” if they publish a network that can be scanned and connected to by other devices. This method has some limitations as well, for example: a device that publishes an SSID is essentially becoming an access point. It can be visible to other devices that perform a WiFi network scan, but the device itself cannot scan for other devices. Hence, if two devices are both in “discoverable” mode they cannot discover each other. Another limitation emerges when a device switches to discoverable mode, it leaves the WiFi network that it was previously associated with, this can impact the user experience since it is not known how much time it will wait until being discovered and complete a proximal interaction.

CoSync proposes a pre-association and grouping mechanism that may suffer less from the above limitations and can serve as a basis for the creation of opportunistic networks.

This mechanism makes two assumptions. First, devices are always connected to the Internet and can access a central coordination server. Second, devices have access to physical location via WiFi networks (BSSID and RSSI) or GPS positioning. In this way the coordination server can make an initial grouping decision.

The coordination server can assign the network setup information to the candidate devices that can later connect and establish a proximal group. The details of this process are described below.
The CoSync framework’s central server manages and coordinates individual nodes and network groups, while enabling the actual content sharing to occur over a P2P proximal link (WIFI or BT). The nodes interact with the server individually through the ClusterManager in order to join to the most relevant network group and receive periodic updates on the status of other nodes. This architecture is not infrastructure-free as it relies on the availability of Internet access (though the 3G interface) to the coordination server. However, keeping video streams and image transfers strictly over WIFI P2P links causes the data bandwidth to the server to be comparatively small in comparison to a server that replays the actual communication between the peers. Considering the fact that centralized node management would reduce the uncertainty of opportunistic networks.

The ClusterManager acquires the location information (latitude and longitude) and Wi-Fi networks fingerprint (SSID, BSSID & RSSI) from scanning. It then sends this information periodically to the coordination server, which makes the grouping decision for the nodes. The server associates the device with the most relevant proximal group. If the assigned group already exists, the network middleware (Wifiglue) attempts to join in to the AP node. If the group is new and does not exist, the device will initiate a new network by starting a mobile access point and become the AP node itself. The network middleware also keeps track of the current members of a group, and handles the cases when nodes join or leave the group.

**Coordination Server Command Flow**

When users start CoSync ClusterManager, it automatically probes the location and network fingerprint and sends this data as payload within a JOIN command to the server. If the server finds a relevant group
within the node’s range, it sends a reply to the device with the group information (SSID and WPA passcode). Using this information, the device attempts to associate with the AP node, and join the group. In the case where multiple relevant groups exist, the group with the nearest AP node will be prioritized. If the server does not find a relevant group, it creates and registers a new group, the server reply with the SSID and WPA passcode of the new group. In this case, the node would initiate its mobile access point with the given information, and enable other nodes to join, when they come in range.

After the joining process completes, ClusterManager periodically issues an UPDATE command to the server. This message includes the latest location and WiFi scan data. The server, in return, replies with the latest list of group members with their private IP address.

When the user decides to leave the group, or physically steps away from the network range, ClusterManager automatically issues a DETACH command to the server. This removes the node from the group, and notifies the other nodes of the group in the next UPDATE command response.

In case of the user deciding that this group is not relevant and wishes to change group, ClusterManager issues a CHANGE command to the server. The server first marks the node irrelevant to the current group and then searches for other groups that may be more relevant but still within the desired proximal distance. The server replies with the new group information (same as JOIN command).

In case that AP node has left the group or got lost, ClusterManager automatically issues a LOST command to the server. The server replies with the closest (in terms of signal strength) AP node candidate to replace the lost AP node. The assigned AP node initiates a mobile access point, and others attempt to associate with it.

In the case a regular node loses its link while the AP node is still present and alive, the server requests the node to retry and associate again with the same AP node. If the nodes succeeds it notifies the server by issuing a RECOVER command.

4 Framework Applications

The following applications use the framework described in this thesis work to synchronize and remote activate different features of devices near by. The framework developed attempts to provide the user with the impression that remote devices he controls are physically attached to his own device.

For simplicity this framework is broken into many different smaller applications, each demonstrating a specific use case where a user may
want to synchronize and control remote devices as additional hardware. Some of these applications are described in the introduction section. Below I will describe and focus on specific applications that represent specific challenges of this framework addresses. These challenges were also evaluated later in the evaluation section.

- **Airmobs** – Opportunistic Connectivity Sharing, crowd sourcing connectivity from users near by that are sharing their mobile internet link opportunistically to other users and in return for contributing part of their data plan they can use the application to connect though other users when they are in need of internet connectivity.

- **CoCam** – A priori proximal sharing of created media. Users share the images they create via P2P links opportunistically without the need be connected to a central service.

- **CoSync Flash** – synchronized and controlled remote flash with strobing effect. This application uses multiple other mobile devices as a wirelessly controlled flash. The application can program the flash behavior: flickering, constant, delay, duration and takes advantage of this ability to move and locate the remote flashes at different positions to generate images in difficult light conditions.

- **CoSync Video** – synchronizing and remotely control array of six mobile camcorders pointing to the same scene from different points of view. This application was developed together with a web-viewer that allows the user to switch from one point of view to any other point of view at any time during playback without the need for editing.

- **CoSync Audio** – synchronized remote audio playback, demonstrated using six speakerphones of six different mobile devices playing together a tune. This application allows the user to overcome the volume limitations of a single speaker and take advantage over available hardware to produce a laud speaker effect.

**Demo Applications Goals**

**Demonstrate the Framework**

In order to develop, evaluate and explain the different features of CoSync framework, I have developed a demonstration suite of applications. The use of multiple applications also tests the generic nature of the API and allows designing and developing the framework for reuse across applications.

**Device Ecology**

The approach for all the applications is to be able to take advantage of whatever devices are available currently. Since the framework is designed to automatically discover and attach to devices nearby, these
applications can take advantage of these resources in a dynamic and scalable way. E.g., the CoSync Audio can use two or five or more devices to play a tune together and when more devices enter the network it can also use these devices.

**Synchronization**

The synchronization is a major challenge as making sure the application can control the timeline in which actions are performed. Since all these applications perform actions across multiple different devices, the applications should be able to accurately control the time in which these actions are performed.

**Remote Activation**

Remote activation of a specific feature is a part of all the framework applications. The framework takes care of distributing the synchronized action requests and the results from the different remote devices.

**New Media Creative Experience**

It would benefit the user to show the potential of crowd sourcing different proximal hardware that is available and shared by other nearby users. All these applications demonstrate a new kind of creative media experience that is no longer bound by the capabilities of the stand-alone device he owns but by the aggregated capabilities of the devices around him. E.g., with CoSync Flash users can take pictures in low light without blinding the subject or over exposing and saturating the image.

**Airmobs - Opportunistic Connectivity Sharing**

Airmobs is a mobile application that uses proximal connectivity and the pre-association layer provided by CoSync framework (without a server) to share and discover shared Internet connections.

Airmobs creates a community that allows mobile users to share their data plan opportunistically with other users that need connectivity and are located nearby. This application allows sharing connectivity in the background without explicit user involvement.

**Application Goals**

**Synergy:**

Airmobs allows the creation of synergy between users that have reached their data plan cap and require connectivity to the Internet at broadband speeds, and users that will not use their entire data plan and may lose the unused data allocation at the end of the monthly billing cycle.

Airmobs logs the amount of data shared by a user; since that user is part of the Airmobs community he can use any other community member as an access point when his data plan has reached its cap.
This synergy helps users not lose the value of their spare data; they can invest the amount of data not used for a future value of having connectivity provided by other community members in months they over use their monthly data plan.

It is very expensive and difficult for a mobile network operator to provide 100% of coverage for its users. Different network operators have different gaps in coverage. In many cases, one operator could potentially host or roam a user of its competitor, but this rarely or never happens.

Users whose network may be provided by different network operators are part of the Airmobs community. As a result of the core connectivity-sharing capability, users of an operator that has coverage at a specific location can provide network access to users of a competing mobile operator that are out of coverage. This synergy is not possible between competing operators but it can be possible between users of the same Airmobs community, especially when sharing behavior can be reciprocated at a later time and a different place.

There are built-in limitations in radio technology that create gaps in network coverage. It may be impossible to establish a link from a mobile device to a cellular tower in spots like street corners behind a large building or at the bottom of an underground passage. Proximal radio links can bridge these gaps since they can establish a local link to another person standing nearby that is not sheltered by radio interference (for example a few feet away in the middle if the street or at the top of the underground stairs). The Airmobs community can create such synergies among its members and overcome some of the mobile radio limitations.

**User Experience**

The data-sharing application should be unobtrusive for the users who are sharing their data plan. Users will become annoyed if interrupted every time someone wants to use their device as access point it will quickly annoy the users, especially if this is done repeatedly several times during the day.

Another important aspect of user experience is that an application that uses device resources for the sake of other remote users should not deplete critical resources. In this application the critical resources are battery and data. The application should avoid the case in which user of the mobile device functioning as access point cannot make a call or use his or her device because the battery was depleted servicing other Airmobs community members. Airmobs should not deplete the user’s data plan; it should use up to a quota specified by the user without using his entire cap.

Users who are sharing should be able to see which devices are connected through their device and be able to see in real time the amount of data shared. This feedback needs also to be subtle, unobtrusive and optional for the user to view but easy to access.
Users should have an easy and simple way to stop sharing or connect to a shared connection. There should be relevant notifications that reflect the status of the application.

**Cost and Business Structure Discussion**

One of the goals of this project is to provoke a discussion around the current mobile operator perspective or their network, their business model, service model and the structure of the users’ contracts.

Some operators view this act of sharing connectivity, performed by Airmobs and other Tethering applications, as resale of their service. These operators (especially Verizon wireless in the US) specifically object to such applications and require the user to pay an additional fee to have such a feature enabled. In some cases these operators even modify the software of the mobile device to block this option programmatically.

This position of mobile operators raises the question: why does a network operator distinguish between bits that originated from the access point node and bits originated by another device transferred by the mobile access point node? This essentially is a business decision made solely by the mobile operators to restrict and discriminate bits.

By creating a community around sharing data plan, Airmobs is challenging this business model and suggesting that users will be better served if bits are not discriminated. Our hypothesis is that by allowing such a feature, users will have better connectivity in cases they are out of network coverage or when their data plan is over and are likely to be more satisfied from this service.

Other questions are provoked around the high costs of data roaming. The high cost of data roaming essentially renders this service unusable to many of the users traveling abroad. The Airmobs community is global, hence when a user from abroad is establishing a link to a local sharing device, he or she is able to connect to the Internet and avoid the roaming costs.

We believe that if mobile operators adopt Airmobs as a model for their users, the operators can reduce costs of roaming, potentially reduce the complexity and cost of their networks, and provide an overall better user experience. Some of the mobile operators have already adopted such a model for stationary access points to offload and improve user experience (e.g. BT’s Fon).
The Airmobs application has three states: idle, customer and provider. The clients connect to the WiFi access point started and published by the provider device and though the 3G/4G interface of the provider device to the Internet.

**Software Design**

**High Level Class Diagram:**

The Airmobs app is composed of two layers: the application layer, providing UI for interaction and service layer that provides the functionality and monitoring deamons.
Application Layer

WelcomeScreen:

This class manages the main user interface for manually connecting as a client ("customer") device or a switching to provider mode. This screen also allows the user to view current connectivity status and how much data was shared by the provider or used by the customer. This screen also provides a navigation option to other application screens.

StatisticsScreen:

This screen aggregates statistics about the usage of Airmobs and shows a pie chart with the percentage of data used and data shared by
the local device for each month, a selector allows the user to browse the history of previous months.

SettingsActivity:

This screen allows the user to set the Airmobs settings. These settings include the battery level below which Airmobs will not provide network to customer devices, and the quota of data the user is willing to share. The users may also configure their own data plan cap and billing cycle to ensure that Airmobs does not cause the user to pass his or her cap.

Service Layer

MonitorService - This class is responsible for gathering various phone and sensors data for monitoring the phone state (more details below). This information is provided to the EngineLogic class that deduces the Airmobs state.
**EngineLogic** - This class receives all inputs provided by the MonitorService class and maintains internal state of the Airmobs app. This class decides how to respond to state changes such as connectivity-lost, WIFI-connected or battery-level-is-low. Once the state is changed, this class notifies state listeners that register for state change notification.

**ConnectivityManager** - This class wraps around the Wifiglue middleware and uses that API to start a mobile access point, receive notifications of devices connected, provide their MAC and IP addresses, or to connect to a specific Airmobs access point as a client and notify that this connection was established.

**BillingService** - This class keeps track over the data used and shared by every device. Each sharing session is logged in two tables: one for sessions that this device acted as a customer and the other table for sessions that this device acts as provider. This information is uploaded to a tracking server once in a while to provide a global view of the entire Airmobs community.

**Client** - This class connects to the server class and authenticates the Airmobs client at the application level, although WIFI WPA2 security is enabled, the password can be hacked, so for this purpose airmobs clients authenticate with the server after it successfully established WiFi connection. In the case of devices using the Airmobs network without the app (i.e. by hacking the password) this session will be limited by time. After successful authentication, this class sends periodic keep-alive messages that help maintaining the connection state.

**Server** - This class is responsible for connecting with the client classes, for convenience this class uses a worker class that checks the authentication and returns ACK messages for the keep-alive messages sent by the client. In cases where the user is taking part in the experiment (see web monitor in the Airmobs application section), a live monitoring message is sent once the server is activated with the operator name and id to which this device is connected. Upon connection with a client, another message is sent to the live monitor to signal this connection with the id and name of the customer device (that is tethering though this provider device). When a client is lost, left the network, or disconnected, a remove message is sent that removes that linked device from the live monitor view.
This class diagram explains the architecture of the various monitoring classes and how the information provided by different monitoring classes is integrated by the MonitorService and interpreted by the EngineLogic class.

**MonitorService** - This class wraps around and Android Service class that facilitates the execution of a background process, this service runs independent of the application’s processes on its own process and can be accessed using an API object. The API object hides the complexity of IPC. The Android SDK generates proxy and stub objects that implement remote method call.

This class holds the authentication service and client that handle application level authentication. It also includes a list of monitor listener objects—mainly UI objects that would like to be notified on change in the monitoring states such as devices being connected, the data used/shared change and the current role state of the app.

The monitor service holds instances of various different monitor classes and provides a unified reporting mechanism that is implemented over the Android message handling system. The handler of these messages is the EngineLogic that gets the aggregate of all the different messages generated by the different monitor classes.

**BatteryMonitor** - This class monitors the current state of the battery and compares it with the settings provided by the user. Once the battery state is below the user-defined threshold, this class will send a BATTERY_LOW message that will be interpreted by the EngineLogic class to decide an action. When EngineLogic received a BATTERY_LOW message it will automatically close the WiFi tethering interface (including all clients connected) and switch back to Idle state in order to conserve battery.
**DataMonitor** - This class checks the traffic statistics of the device and how many bytes have been routed through each of the network interfaces. The class updates the amount of data used in real time and provides a total measure when a connectivity sharing session is complete. This class monitors the amount of data used in real-time when the shared data use is close to the user-defined cap or when the user is near his data plan cap. When data usage approaches its limit, this class will send a **DATA_LOW** message. When the EngineLogic class receives this message it switches state back to **Idle** which turns off the WiFi tethering interface.

**ConnectivityMonitor** - This class monitors the connectivity state of the phone. It checks the WiFi state (is it connected, is the AP on, is it Airmobs AP, is it ON or OFF) and will issue relevant messages when the connectivity conditions have changed. It also monitors the cellular network state, including the signal strength and the availability of cellular data and the device roaming state. In cases that the cellular data is available and not roaming, this device is a good candidate to provide network. The ConnectivityManager sends these messages to the MonitorService that provides its message queue to the EngineLogic for aggregate state decision-making.

![Airmobs state diagram](image)

**MotionMonitor** - This class uses the accelerometer data to try and infer motion, it uses pitch detection to find walking pattern inside a window of filtered samples. Once substantial motion is detected for a period of time (someone walking or running) this call will notify the MonitorService that such a motion was detected and a message will be relayed to the EngineLogic. A stationary device is assumed to be a better candidate to provide network than a device that is on the move.

**Data Tracking**

**Persistent Data Tracking server:**

For the purpose of learning the sharing behavior and tracking how the community shared data at a global level, we implemented a data-tracking server.
The data-tracking server has a database containing two tables. Each table corresponds to one of the Airmobs sharing states. The shared data table tracks the amount of data shared and to which consumers this data was shared. The provider nodes publish the information. The consumed data table tracks the consumer’s reports of data used though another Airmobs provider node.

The format for publishing this data to the server is by sending a list of JSON objects, each object representing a single session of providing or consuming shared data. The JSON object has the following format.

```json
{
    operator: "name",
    operator_id: "cellular id",
    customer: "mac address",
    provider: "mac address",
    data: bytes number,
    time: start time,
    duration: seconds,
}
```

The list of the above objects is sent to the server via HTTP POST message with the list as the payload data for the HTTP POST.

The URL for reporting shared data tracked to the tracking server is http://airmobs.media.mit.edu/data/shared

The URL for reporting used shared data by a client (consumer) is http://airmobs.media.mit.edu/data/consumed

The data-tracking server maintains an administrator website that can be access to view the current data tracked in html form format. The admin URL is http://airmobs.media.mit.edu/data/admin

The following classes are responsible for reporting data tracking information to the server.
LogUploader - This class is responsible for setup, starting and stopping the tracking upload process.

UploadAlarmBroadcastReceiver - This class is a broadcast receiver that handles a periodic check if there is data to upload that was written to the database between upload periods.

DataTracker - This class reads the local phone database for shared and consumed data records, selects the records that were not uploaded and prepares them in a list for the UploadAlarmBroadcastReceiver to upload to the server data-tracking server during the next upload period.

Application UX

The application can change states according to a background monitoring decision or manual user request. If a device has no cellular data connectivity it cannot change state to provider; it can only be in idle and customer states.

The background service decides dynamically whether to change its state and display a notification to the user in the service notification. A corresponding icon reflecting the state of the application is shown to the user even when the application is not running in the foreground.

No interruption of the user is required for the application to provide connectivity to other users. A user can optionally start the application to view its current state with full details (who is connected, for how long, how much data was shared).

User notification is important since the goal of this application is to provide opportunistic collaboration and synergy. If user involvement is required, the connectivity will be planned and not opportunistic.
CoCam Media Sharing

CoCam is a camera application that automatically discovers, connects and share images in real-time with nearby mobile users that use the same application.

Application Goals

New Sharing Modality

Web based services, such as Facebook, that enable users to share images and video clips have become popular. These services originally focused more on the experience of sharing rich media contents after the event (post-hoc) and less on the real time experience of sharing images and video clips during the event. These web services act as separate silos and rarely allow users to discover and collaborate with users of other services, though they may have participated in the same event. Even for users who use the same service, it is difficult to collaborate and share contents, as the user needs to tag content (with text labels, geo-tags, etc.) for others to find and consume. In most cases, users must be connected on the social graph beforehand in order to access the shared content. Moreover, users may need to filter shared media that is not relevant to the context of the shared event. As a result, sharing and consuming contents with the relevant people requires the users to invest time and effort.

This application investigates a new sharing model. Instead of sharing images with people you already know (friends / social network) that are out of your context after the event (post-hoc), this application allows the users to share images with people they do not know that are copresent during the event in real time.
Opportunistic Network

Opportunistic networks based on proximal radio are good candidates to address the limitations of content sharing between uncoordinated users. These networks can improve the user experience by using proximal wireless links with higher bandwidth instead of mobile data infrastructure, and reduce the cost associated with mobile data when sharing contents. These networks can relax the requirements for tagging, searching and filtering content since the relevance can be inferred from the opportunistic context.

Opportunistic networks are created around the context of a specific opportunity. Examples of opportunistic contexts are: a shared event that people engage with by taking pictures; sharing files between different people with common content interest profiles; relaying a message between different people whose geographical locations intersect in a way that generates a path between the source of the message and its destination.

These opportunistic networks are pervasive in their nature as network nodes can join or leave the opportunistic context. Another element that makes these networks dynamic is the nodes mobility and batteries that can cause disconnection at any given time. The ubiquity of mobile phones and their ability to support multiple communications interfaces make them excellent candidates for creating opportunistic networks.

This application demonstrates the ability of the CoSync framework to create such opportunistic networks, allowing users to share media over proximal links.

Discovery and Relevance

It is difficult to infer the relevance of a group of devices that are joining to a network in an opportunistic way. It is possible to have joined a group though there is another group nearby that is more relevant. There may be cases where a specific user sends non-relevant information or media to the group; the group should be able to mark this user and ignore him.

The demonstrated use-case is a trade show demo with multiple groups taking pictures at the same proximal area. The users have an option to change group. If there are other groups in proximity, the coordination server of the CoSync framework will notify the ClusterManager to switch group and effectively switch wifi networks.

By allowing the user to transparently switch groups in the background without impacting or interrupting their experience users can select which group is relevant without the need for the system to guess relevance, a difficult or impossible task to accomplish programmatically.
Internet Offload to Proximal Links

Sharing user-generated media on various services using the cellular network has associated cost and increases the burden on its infrastructure. According to "An Empirical Study on 3G Network Capacity and Performance"\(^4\), in conditions of fully loaded cellular network, latency for 3G data services can increase to beyond one second, and unpredictable behavior is exhibited such as data calls being dropped unexpectedly. These limitations may not be an issue for basic email or web browsing, but it may heavily impact the user experience of streaming rich media content in real-time. While there are many ways to load balance the mobile Internet traffic, ad-hoc networking with proximal radio could be one of the solutions for enabling rich media content sharing with others near by, without applying unnecessary loads on the mobile infrastructure.

Simplicity

The application is designed with simplicity in mind. It uses the ability of CoSync framework to coordinate, create and connect to a network without requiring the user to manually handle network setup parameters, network names and passwords, service URL's and hashtags.

Operating the mobile device when creating media, sharing it and consuming shared media is a difficult and sometimes complex. Traditionally, these tasks are separated into different applications designed separately for creating, sharing, or consuming media. This application provides a simple point and shoot design that simplify this complexity.

Application Design and Implementation

Class Diagram:

**Peervideo** - is the main class responsible for presenting a point and shoot application with CoCam additional features. This class delegated to classes that deal with three aspects of the application. Camera
(picture taking, presenting preview video, creating a video stream), image file sending and receiving, and network discovery, creation and management.

**JpegFileSender** – a simple sender class that has internal thread to manage asynchronous file sending.

**JpegFileReceiver** – a simple receiver class that uses a worker thread class named Receiver to accept and receive incoming files shared.

**CameraPreview** – this class handles the access to the camera and the presentation of a full screen preview.

**CameraPicture** – this class is responsible to take a picture when the user desires.

**ClusterManager** – this class is responsible to communicate with the coordination server and provide two relevant location indicators. The location as extracted from the LocationCoordinator class (basic lat/long coordinates taken from the mobile GPS API) and WifiScanner that creates a network scan list as indication of location. These inputs are sent to the coordination server that keeps track over all the network peers and assign them into clusters or groups. Each group gets a specific network name and password. One of the group members is selected by the cluster manager to become an access point node by using the WifiApManager class; others connect to it using the WifiConnector class. Further description of the protocol and roles of the ClusterManager are described above in chapter 3.

**Application UX:**

[Image of CoCam UI]

The UI is designed as a simple point and shoot application with additional buttons that handle collaboration. The first button is group join-leave toggle that allows the user to decide whether he wants a collaborative experience or not. The second is privacy button allowing the user to still be part of a group and receive images but not send images to other peers when in private mode. Another button handles the relevance of switching a group when the incoming content is not
relevant or when a user wishes to explore other groups in his proximal area.

Incoming shared images appear on the top left as thumbnails (becoming full screen when selected) and a camera shooter button for taking images.

**CoSync Flash**

*Wireless Synchronized Remote Programmable Flash for Mobile Phone Cameras*

**Introduction and Motivation**

Mobile phone camera performance is very poor in low light conditions. On the other hand, the onboard LED flash may overcompensate for a short distance night shot resulting in overexposure; without the flash a very low shutter speed results in motion blur. In contrast, professional camera setups use multiple remotely triggered flashes to create dramatic and compelling images. The disadvantages of such professional setups are that they are very expensive, require specialized equipment, need to be carried around and take time to assemble. Hence opportunistic capture is rarely possible and seldom do we see such setups outside of the photographer studio.

Based on the framework presented in this thesis, CoSync Flash uses multiple mobile phones as remote synchronized wirelessly controlled flashes that can be individually programmed and controlled, and jointly triggered when the main device camera takes a picture. The goal of this application is to try to bridge the gap between the expensive professional multiple flash setup and the availability of mobile phones around us.

The advantage of the suggested application is that it allows a more dynamic and flexible setup; i.e., multiple flash modes such as strobing, controlling the flash timing duration and delay.

**Motivation**

In low light conditions the mobile phone camera uses its on board flash. This may cause various problems such as:

- Overexposure - LED light is saturating the image especially reflected from bright surfaces and people's faces.
Fig 16 Over exposed normal mobile flash

- Motion blur - caused by the long exposure time needed by the mobile phone camera.

Fig 17 Motion blur normal camera flash

- Blinding the picture subject - caused by the location of the LED flash that is positioned directly in front of the camera subject.

Fig 17 Blinding front facing LED flash
Limited light capacity – even with flash the picture subject may be too far away to produce visible image.

So why even use such a limited, ill performing camera?

The mobile phone camera is a more immediate way to take pictures especially when the picture taking opportunity was not planned in advance. Social picture opportunities happen indoors in the evenings in poor light conditions such as a party or a dark restaurant. As a result, we lose precious moments due to mobile camera limitations.

Based on this thesis framework the app uses multiple devices in synchronized collaboration for taking a better low light condition image. Further investigation beyond the scope of this thesis may address the computational camera aspects of using mobile phones as multiple synchronized flashes and multiple cameras. Another important investigation should address the automatic measuring and focus when multiple external flashes take part in capturing an image. This synced multi-flash, multi-camera capability in mobile phone system is introduced for the first time.

Implementation and Design

CoSync Apps UI Design

CoSync applications share the same UI design with special features enabled for different use cases. On the right side icons represent user actions such as take picture, record video, record audio, play audio etc. All user actions can be done in synchrony with other devices around. On the top left side are the remote action settings when the local action is different than the remote action.

For example if a user wants to take a picture with his camera and other device as flash he will press the top left flash icon to access remote flash control settings menu. Another example would be a user that wants to record a video with other devices functioning as remote microphones he will press the top-left microphone icon to access the remote settings of the microphones on other devices.
CoSync Video

Wireless synchronized multi viewpoint video creation and web based display.

Introduction and Motivation

There are many tasks that are just too difficult to perform on our own, but with the right collaboration setup these tasks become achievable and even fun.

It is very challenging to shoot a scene with video cameras from multiple views. Not only that the user needs to control multiple cameras in order to create the video, later he needs to synchronize all the clips and edit a cut version that only than can be shared by with other uses. Even after sharing the clip the viewers may want to see the scene differently but the result cut is static since it was edited.

CoSync video tries to create a co-creation experience where multiple users can join and create a single video that is actually multiple synced clips that can be later shared and viewed. The viewers now also get to participate in the creation since they can switch views at any time during playback without editing and continue watching the video.
Implementation

This app is using the framework for all its remote activation, discovery and synchronization and adds the ability to record a video upon synced action request.

Since the action of recording a video requires the mobile phone operating system to allocate hardware and software resources such as codecs and memory, it is a relatively complex and long action to complete. More importantly different devices take different setup time to start recording a video. Even if the sync is perfect the actual video start time may differ from device to device. Contrary to starting a video recording, stopping a recording is relatively quick and is more reliable as synced action.

The CoSync Video app encodes the video time from the end of the video to the start and so is the name of the video file that includes the device id and the end video synced timestamp. All videos are then uploaded to the presentation server to a folder named according to the video start recording synced action time that is shared by all devices.

The UI is similar to other CoSync applications that include camera and presented in the above section named “CoSync Apps UI Design”.

Demo Setup

For the purpose of the demo, six mobile devices were mounted on a simple rig pointed at the same scene from different angles.

Fig 20 CoSync video demo setup


**Demo Display Station**

The display station is designed as a web page first presenting a thumbnail image per each collaborative video taken (i.e. a folder of synced videos).

Once a thumbnail is pressed the specific multi-view video is presented to the user and he can start playback. At any given moment, the user can press the video and start transitioning between the different video angles. Pressing the button again will immediately continue the video from the angle selected. This selection can change at any time over and over again, while not requiring post synchronization or editing of all the clips.
5 Evaluation

Connectivity Sharing Evaluation

For the purpose of evaluating the connectivity sharing I have developed two tools. The first is a web monitor tool (http://airmobs.media.mit.edu:8888/static/history.html) that records some of the data sharing sessions conducted with Airmobs. These are records of users who opted-in to join the experiment (a checkbox in the app settings that must be checked). The second is a data-tracking server that actually tracks the sharing data amount and sharing data sessions. Since the app was released the server recorded 316 sharing sessions.

Airmobs Web Monitor

A red circle marks devices that are sharing a connection. A blue circle marks devices that is consuming a shared connection and tethering trough the shared connection. A gray circle is a mobile operator that finally provides the internet connection. We can see that there are about 13 different operators world wide that are reported by their users who joined the experiment and compose this image. Note that mobile operators that are present twice in this image have two different operator IDs that is possible with MVNOs, and big operators.
Airmobs Installations

The following graph is total application installations taken from the android play store app page.

Media Sharing Evaluation

User Study Results:

This CoCam app demonstrated the technical feasibility of real-time media sharing over opportunistic self-organizing network.

A study was conducted to evaluate the user experience of CoCam and to better understand the users' needs, habits and expectations upon a new model of real-time content sharing in the context of events.
The user study was conducted under a simulated demo pavilion scenario, commonly seen in professional conferences. Users were requested to explore the pavilion, and record photos of the demos using CoCam as they browse through. For the purpose of the study, twelve students (aged between 19 and 35, mixed gender) were selected. The students were separated into three different groups of four users. After a short period of exploring the pavilion demos, the students were asked to answer a web-based questionnaire that evaluated their experience. They were also asked to comment freely about different aspects of their live content sharing habits and their experience during the study. The time period given to the users in the study was designed to be slightly shorter than the time a single person needs to document comfortably all the demos in the pavilion.

From the user study we hoped to learn what are the habits and expectations of users from real-time image sharing. We asked questions that would be indicative to learn how important is privacy and prior social connection with other people in the context of sharing and receiving shared images. We evaluated whether the UI designed for the application was usable and intuitive enough for the task of creating and sharing images. We hope to show that the users perceive this as efficient and time saving method for covering and documenting a public event. We would also like to show that the image content was relevant and interesting and that incoming images were not so intrusive.

Users were asked whether they usually share images during live public events Fig 24. 58% of the users answered that they rarely or never share images at live events and 33% answered that they share often or always. Users were also asked whether they would like to get images taken by other peoples at the same event. 67% answered that they would often like and only 17% answered that they would rarely like to get images taken by others.

Fig 24 [Real-Time sharing habit and expectation: 1. Do you usually share images during event? 2. Would you like to get images other people took at the same event?]

In regards to receiving shared images 42% of the users surveyed prefer to receive shared content from their friends whereas for 33% being friends with the person that shared the content received is not very important. Regarding users sharing their own images taken during a
live public event, 50% don't mind disclosing their content publicly whereas 42% prefer not to disclose the images they took publicly,

Fig 25 Privacy and prior social connection:
1. Is it important for you to be friends with whoever shared images with you?
2. How much do you care about disclosing images you take during public event?

Fig 25. When evaluating efficiency Fig 26, 58% of the users perceive this method of sharing images during live public event as a method that increased their coverage of the event, 33% only to a medium extent and 8% to a little extent. On the other hand when we asked whether they think that sharing images during event saved them time to achieve desired coverage of the public event, we found that 75% of the users found this method to be time saving only to medium or little extent.

Fig 26 Efficiency:
1. Did sharing images during event increase your coverage?
2. Did sharing images during event saved you time to achieve coverage?

Evaluating relevance, Fig 27. 42% found the images received during the user study to be relevant compared to only 17% found the images shared by other users to be not relevant.
The user experience of using the CoCam app in the user study was found by 58% of the users to be not disruptive or disruptive to a little extent.

75% of the users found the CoCam user interface to be intuitive and 50% of the users felt that they received images in a timely fashion, Fig 28.

Discussion:

The results indicate that the majority of the users are not sharing images during events. On the other hand, there is also a majority of users that would like to get other people’s images from the same event. This finding raises a potential gap in content supply vs. demand. This gap may be attributed to the difficulty and complexity of engaging in sharing during an event. This finding strengthens the motivation to provide tools that allow users to easily share created media with others that seek to consume it. Another finding may suggest that social connection is still very important for the users as more users reported that it is important for them to be friends with whoever shared images with them than users that didn't care about prior social connection. Somewhat in contrast CoCam can provide value exactly in situations where social connections and services are not available within the opportunistic context. Users may value better coverage of a public
event, as CoCam is perceived to assist in that task. It was expected that the users perception on time saving will match their perception on increased coverage, but only the minority perceived this method to be time saving to a large extent or more. This may suggest that further improvements may be needed in order to provide a time saving experience (such as automatic upload to various social services). The familiarity of the point and shoot interface is a key factor in making the UI clear and intuitive to the majority of users. Even though additional information was presented that is not part of the original camera app (such as incoming images and group status). This may lead to future inclusion of real-time social features within the native smartphone camera app while still keeping the application clear and intuitive to use.

**Synchronization Method Evaluation**

**Time Based Synchronization Algorithms Evaluated**

There are many timekeeping algorithms that are used to synchronize devices to the same clock. The different requirements drive different implementations. For the purpose of this framework we have evaluated the following algorithms and protocols for time keeping and synchronization. Description of these protocols is provided in the background chapter two.

The following describe the implementation used to sync and time based sync of mobile devices:

1. **NTP** - The version of NTP client evaluated is a class by Google android SnmpClient that is periodically re-sync with an Internet NTP server. Some of the NTP servers reject repeated requests from the same client, as it is not common to re-sync every few seconds. For the purpose of this algorithm I used “time.fu-berlin.de” which allows repeated NTP requests.

2. **PTP** - For the purpose of this framework I have developed two version of PTP. The first implementation straightforward forward PTP message exchange sequence is described in chapter three sync sequence diagram.

3. **UPTP** - Is a second version of PTP attempting to reproduce the software assisted PTP without actual driver support. This version assumed that the timestamp after invocation of socket.send() method is similar to the timestamp of the packet leaving the device. If this assumption is true an updated message can be sent according to the software assisted PTP. This implementation records this time and sends two update messages as MASTER_UPDATE following the MASTER_SYNC message and SLAVE_DELAY_REQUEST_UPDATE after the SLAVE_DELAY_REQUEST.

4. **GPS Time** - the GPS time provides a common time source to any device that receives GPS updates. In this framework I have evaluated the use of GPS time for accurate sync. This is accomplished by using the Android OS APIs provided by the classes LocationManager, LocationListener and
Location.getTime() method. Updates are requested from the mobile device location manager every 50ms and if a change of 10cm in the location detected occurs. These settings are not common since for location tracking application one would like farther apart updates (commonly one to five seconds and 1 meter).

5. BCN – Beacon broadcast sync algorithm, assumes that at least one beacon message will be transferred in a very short network delay. The algorithm attempts to detect this message by its arrival sooner than expected. Once this message is detected the offset to the master can be updated to reflect a minimal sync delay. The algorithm evaluated is described in chapter two in details.

**Evaluation Method**

A special evaluation application was developed based on the framework. This application was used to evaluate the above algorithms.

The application presents a timer to the user. The main screen has a sync button that activate the SyncManager Service and specific SyncProtocol selected for the test. The user can press play to start the timer in sync on both (2 or more) devices and press again stop in order to stop the timer in sync.

![Image of a mobile device with a timer application]

The start / stop action is recorded by a third device camera; I used a laptop webcam, to provide a joint time reference. The video is later analyzed manually using a video editor to record the sync gap in frames between the devices. Since the camera captured 24 fps video the shortest sync delay that can be detected is one frame delay that equals to 41.66ms (1000/24). This synchronization level < 50ms is good for most applications (including below synced flash). Another evaluation is needed to test if the sync is good enough for multi-source synced audio playback since its requirements are sync gap < 30ms. This case is evaluated separately below with higher accuracy (refer to audio sync evaluation section)
Results

The results below describe the synchronization gap between the devices in milliseconds. We can clearly see that PTP NTP and UPTP outperform the other algorithms BCN and GPS based.

<table>
<thead>
<tr>
<th>Sync Gap</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTP</td>
<td>0.00</td>
<td>41.67</td>
<td>125.00</td>
</tr>
<tr>
<td>GPS</td>
<td>375.00</td>
<td>491.67</td>
<td>583.33</td>
</tr>
<tr>
<td>UPTP</td>
<td>0.00</td>
<td>58.33</td>
<td>166.67</td>
</tr>
<tr>
<td>NTP</td>
<td>0.00</td>
<td>16.67</td>
<td>83.33</td>
</tr>
<tr>
<td>BCN</td>
<td>416.67</td>
<td>508.33</td>
<td>625.00</td>
</tr>
</tbody>
</table>
Discussion

For further evaluation and improvement we can see that it will be better to focus on the leading algorithms NTP, PTP and UPTP. The minimum of all these algorithms is 0 frame gap indicating that they can perform < 41.6ms sync. All of the algorithms used are periodical sync algorithms that have noise hence all can be improved by adding a Kalman filter to reduce fluctuations and noise as suggested by Aggelos.

Since PTP is outperforms UPTP and has fewer messages in its protocol, it is preferred over the latter. Since NTP requires a NTP server a connection to the Internet and pre-configured and agreed server it is less suitable for opportunistic scenarios of collaboration that this framework strive to support. As a result, I decided to use PTP to support the framework use cases, removing the need for any configuration and Internet connection, while still gaining a relatively good performance.

Synchronized Actions Evaluation

Synchronized Flash Action

One of the most interesting performance questions is how well is the camera synchronized with external flash devices without any cable or hardware synchronization method.

The evaluated question is: what is the minimal flash duration that can be captured by another device camera?

I have conducted the following experiment: one device camera captures another device flash. The images are captured and the flash fires according to the synchronized time. If the camera device captures image with the flash clearly on – the camera and the flash are synchronized.

By reducing the flash duration time again and again until we can no longer detect the presence of flash in the captured image we can deduce what is the minimal synchronization (hence the minimal duration of flash to be detected).

Results

The experiment was repeated with flash durations beginning at 1000ms reducing duration iteratively until the flash was not detected. The experiment resulted in successful flash detection durations down to 50ms. Below 50ms, the flash was not detected.. This result consistent with other synchronized action evaluations conducted for this framework which point to about 40-50ms of synchronization accuracy between different devices.

Validation

The below images are taken for comparison by two different applications using the same mobile phone camera.
The images on the left were taken by our CoSync Flash camera applications and the images on the right were taken by the phone camera application. The phone camera application mode was full auto. I used the phone camera app as the baseline for validating the results of the collaborative camera application. Since evaluating image results is very much dependent on personal preference and opinion, we have only conducted and informal user study with a group of colleagues and students. To properly answer the question below, a larger survey should be conducted comparing multiple images of the same scenes taken by the same person with two different apps.

The evaluation question presented was: which image you prefer and why?

The feedback received from a few potential users in the course of the this informal user study was that they all agree that there is some improvement in making the images more dramatic with wider range of colors, less flat, less blurry, less saturated, softer but more granular

**Result Demos**

![Image 1](image1.png)  ![Image 2](image2.png)

Synchronized Remote Audio Action

This is the most challenging use case to evaluate any synchronization technology. Trying to play a sound from two different sources in sync as if it is played from a single source has been a challenging task, especially for wireless speakers.

In addition to the challenging task the mobile phone environment present additional difficulties. The mobile phone has non-deterministic
performance at the user interface level because of background actions and other applications that execute in parallel.

**Evaluation Method**

The experiment begins when two synchronized mobile phones start to play together. Each device plays a repeated on and off sequence of 500 ms of DTMF tone (0/1). The sequence is saved as a PCM wave file to ensure a sequential playback of the sample by both devices at the same rate. Each device is recorded on a different channel with a stereo microphone (converted headphones). This allows easy separation of sources since each device will have a specific stereo channel (right and left). Using two different DTMF tones helps detecting which device is assigned with which channel as these tones can be easily separated by their DTMF frequency.
**Results**

I have evaluated three different sync algorithms in order to select the best one for this purpose. The algorithms evaluated were NTP (by Google SntpClient class), PTP (self implemented) and UPTP (self implemented).

The recorded audio was analyzed using the Audacity application and the gap in synchronization was logged in the following table and chart.

<table>
<thead>
<tr>
<th>Sync gap in milliseconds</th>
<th>PTP</th>
<th>NTP</th>
<th>UPTP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>95</td>
<td>210</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>80</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>95</td>
<td>5</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>45</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>20</td>
<td>55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gap (ms)</th>
<th>PTP</th>
<th>NTP</th>
<th>UPTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>20</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Avg</td>
<td>63</td>
<td>72</td>
<td>69.4</td>
</tr>
<tr>
<td>High</td>
<td>95</td>
<td>210</td>
<td>185</td>
</tr>
</tbody>
</table>

The human auditory system is evolved also to localize sound. This ability to detect audio source location is based on our ability to detect different time of arrival of sound to our left and right ears. This effect is called the precedence effect, also known as the “Haas effect” or the “law of the first wavefront,” is the psychoacoustic effect in which the
apparent location of a sound is determined largely by the localization cues from the initial onset of the sound [53,54].

The synchronization difference should be smaller than 30ms in order for the brain to unify different sources into a perceptible single source (one big virtual speaker). If the audio sources are not synced within 30ms the brain can detect the difference between these sources (Haas Effect). In delays bigger than 30ms the sound becomes muffled and later echoes are perceived followed by complete separation of sources.

**Discussion**

From the results we can see that all algorithms can perform within the required benchmark. We can observe that all algorithms also perform poorly and do not satisfy the need for consistent performance below 30ms sync delays.

I have selected to use a self-developed version of PTP for audio sync since the spikes between the lowest and the highest values are low but at the same time the performance is inconsistent. It is possible that users may hear echoes or multiple sources not playing in sync.

Since many sync algorithms sync again and again every few seconds or microseconds. These algorithms, in many cases, cannot distinguish between different sync results that may be better or worse in terms of sync gap. It is desired to design a future algorithm that is aware of its own performance, and re-sync only if it results in a better sync.

All methods have noise that can be modeled and the fluctuations can be smoothened by applying a Kalman filter over the result time offset as suggested by Aggelos.

As a result of this experiment the framework adopts PTP as the synchronization algorithm for audio sync as it is local and not internet dependent on a specific NTP server and the master sync node can rotate between the nodes as they enter and exit the proximal network. Still, the user experience is not stable enough to provide this feature to all mobile devices without specific support by the operating system to improve sync results.
Bibliography

1 http://www.wi-fi.org/knowledge-center/white-papers/mobile-ad-hoc-networking-wi-fi-certified%E2%84%A2-ibss-wi-fi-protected-setup%E2%84%A2

2 http://www.wi-fi.org/discover-and-learn/wi-fi-direct

3 http://www.wi-fi.org/wi-fi-certified-miracast%E2%84%A2

4 http://www.nfc-forum.org/home/


8 http://www.facebook.com

9 http://viral.media.mit.edu/#vision


15 Wolfgang Kiess, Martin Mauve, A survey on real-world implementations of mobile ad-hoc networks, Ad Hoc Networks, Volume 5, Issue 3, April 2007, Pages 324-339, ISSN 1570-8705


18 http://www.newscientist.com/article/mg21729015.900-bandwidthsharing-app-brings-connectivity-to-all.html


20 http://www.btfon.com

21 http://www.comcast.com/wifi

22 http://www.att.com/shop/wireless/wifi.htm

23 http://www.whisher.com/

24 http://guifi.net/en/

25 http://www.thelifenetwork.org/

26 http://www.shareair.net/

27 http://www.easywifi.com/

28 http://opengarden.com/about.html

29 https://yourkarma.com/


31 http://www.meraki.com

33 Lawrence, E.; Bina, M.; Cujak, G.; El-Kiki, T., "Wireless Community 

34 Olson, Mancur [1965]. The Logic of Collective Action: Public Goods 
and the Theory of Groups (Revised edition ed.). Harvard University 

CrowdMAC: a crowdsourcing system for mobile access. In Proceedings 
of the 13th International Middleware Conference (Middleware '12), Priya Narasimhan and Peter Triantafillou (Eds.). Springer-Verlag New 

36 Jisub Lee; Chenglong Shao; Heejun Roh; Wonjun Lee, "Price-based 

37 Mihai Marius Constantinescu, Ertan Onur, Harry Bouwman, Miodrag 
Djurica, Mark De Reuver, and Yunus Durmus. 2012. Cooperative 
networks: the mobile tethering game. In Proceedings of the seventh 
ACM international workshop on Mobility in the evolving internet 
arquitecture (MobiArch '12). ACM, New York, NY, USA, 41-42.
DOI= 10.1145/2348676.2348687

38 R. McCain. Game Theory: A Nontechnical Introduction to the Analysis 

39 M.Conti,F.Delmastro,andA.Passarella,“Social-awarecontentsharing in 
opportunistic networks,” in Sensor, Mesh and Ad Hoc Commmuni-
cations and Networks Workshops, 6th Annual IEEE Communications Society 

40 J. Su, J. Scott, P. Hui, J. Crowcroft, E. De Lara, C. Diot, A. Goel, M. H. 
Lim, and E. Upton, "Haggle: seamless networking for mobile 
applications," in Proceedings of the 9th international conference on 
Ubiquitous computing, 2007, pp. 391-408.

41 J. K. Nurminen, O. Karonen, L. Farkas, and T. Partala, "Sharing the 
experience with mobile video: a student community trial," in Proceed-
ings of the 6th IEEE Conference on Consumer Communications and 
Networking Conference, 2009, pp. 1197–1201

Cooperative content sharing for bluetooth users,” in Fifth Annual IEEE 
International Conference on Pervasive Computing and 
Communications, 2007.


52 http://audacity.sourceforge.net/


54 William A. Yost and George Gourevitch, Eds., Directional Hearing, chapter The precedence effect, Springer-Verlag, 1987
