EFFICIENT COMMUNICATION FOR MOBILE DEVICES IN THE NEW ERA

A Dissertation Presented

by

YING MAO

Submitted to the Office of Graduate Studies,
University of Massachusetts Boston in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2016

Computer Science Program

CHAPTER 1

ABSTRACT

EFFICIENT COMMUNICATION FOR MOBILE DEVICES IN THE NEW ERA

MAY 2016

YING MAO

Ph.D., UNIVERSITY OF MASSACHUSETTS BOSTON

Directed by: Professor Bo Sheng

Smartphone has become one of the most revolutionary devices in the history of computing. With various kinds of applications, the scope of smartphone has been significantly broadened in the past few years including almost every aspect in our daily life. However, due to the limited on-board resources such as CPU, storage, network bandwidth and battery power, smartphones and the mobile network serving them bring new challenges that have not been encountered in the traditional computing and networking environments.

This dissertation focuses on the research areas of improving the network architecture and enhancing the current applications on smartphones. It mainly investigates the areas in the following two directions for three representative categories of mobile services.

- In the first direction, the dissertation aims to develop new communication models for smartphone Ad-Hoc networks to achieve efficient communication in the proximity. It is motivated by the fact that smartphone Ad-Hoc networks can help improve the current **location-based services** and propel new applications. Moreover, the new communication models provide complementary alternatives to the traditional infrastructure-based wireless networks.

- In the second direction, the dissertation focuses on improving the other two categories of services, **cloud storage services** and **real-time video streaming services** for mobile devices. In the field of cloud storage, we introduce

doi:10.20944/preprints202007.0461.v1

a cloud-assisted approach to provide a set of advanced file operations, such as encryption, decryption and compression, on smartphones. Furthermore, by utilizing the on-board Near Field Communication(NFC) module, we develop an algorithm to securely share the files between mobile devices. For the services of real-time video streaming, we propose approaches that identify a user's status by analyzing the accelerometer data and then, dynamically adjust the buffer mechanism to save network bandwidth on smartphones.

The proposed communication models and enhanced applications have been intensively evaluated with both experiments and simulations. Compared to the prior work, this dissertation has identified a few important problems for the efficient communication on mobile devices and provided novel solutions to improve the performance.

doi:10.20944/preprints202007.0461.v1

CHAPTER 2

INTRODUCTION

With recent advances, smartphones have become one of the most revolutionary devices nowadays. According to a report from Pew Research Center in 2015, about 64% of U.S. consumers own a smartphone and 39% of the total Internet traffic is consumed by mobile devices. Consequently, a variety of applications have been developed to meet users' demands in all aspects. Today's smartphones have gone far beyond a mobile telephone as they have seamlessly dissolved in people's daily life in all perspectives by providing all kinds of services through mobile devices.

Among the various services on mobile devices, this dissertation studies three popular and representative categories, (1) Location-based services; (2) Cloud-storage services; (3) Real-time video streaming services.

2.1 Research Problems and Challenges

In this section, we identify the research problems and challenges for each of the three representative mobile service categories.

2.1.1 Location-based Services

Location-based services for mobile devices have attracted a large volume of users [1–6]. The current location-based services, however, are still built upon the client-server architecture which incurs some unavoidable issues. Let us consider an example where a department store in a mall tries to deliver a flyer file to a nearby shopper. In the current architecture, the following steps are required: (1) the store hosts the file on its server; (2) the user has to install the store's applications; (3) the user connects to the Internet in the mall and reports his location to the store's server; (4) the server delivers the flyer file to the user. This process involves a few representative drawbacks that an Ad-Hoc network can help address. First, a user cannot discover nearby data or information (step 2). He has to register for each and every service he is interested in. With an Ad-Hoc network, the user can browse or receive all the unknown services nearby as long as they transfer information on a common channel (such as WiFi). Second, step 3 and step 4 require the Internet connection which may not be always available, e.g., subway stations, crowded and congested areas, and the areas with infrastructure failures. Not to mention that when the store and user are close to each other, the data transfer going through the Internet may not be necessary and could incur additional costs to the user. Third, step 3 requires an accurate indoor localization scheme.

After investigating the problems of current client-server architecture, we argue that Ad-Hoc network model is a complementary alternative that can effectively help solve all the above issues. With an Ad-Hoc network, be able to hear the signals from the store is a best evidence that the user is close to the store. Therefore, constructing a mobile Ad-Hoc network (MANET) with hop-by-hop communication to carry local data traffic is desirable in practice.

However, the current routing protocols used in MANETs suffer from two major problems. First, it is costly and inefficient to establish a path from the source to destination. Traditional MANET routing protocols either pay a high cost for maintaining routing tables or flood a request message in the entire network for on-demand path discovery. Both categories require a large number of messages to be delivered for establishing a path. The second problem resides in the path recovery protocol which is usually triggered by an observed failure and proceeded by repeating the path establishment process. In practice, however, this recovery process is slow.

2.1.2 Cloud-storage Services

The cloud related services has been widely deployed in many applications, such as edge computing [7–9], deep learning [10–12], big data processing [13–15], resource management [16,17]. In this field, we consider the mobile application of cloud-storage service, which is a recent emerging technology for mobile devices. Representative products include iCloud [18], Dropbox [19], Box.com [20], Google Drive [21], and etc. [22, 23]. Basically, each user holds a certain remote storage space in cloud and can access the files from different devices through the Internet. Synchronization and file consistence are guaranteed in these cloud-storage services. When smartphones become popular, it is ineluctable for users to couple cloud storage service with their smartphones. However, users and developers have encountered specific challenges due to the limitations of smartphones. First, the storage capacity of a smartphone is limited compared to regular desktops and laptops. Second, the network bandwidth of the cellular network is limited. At this point, major U.S. mobile networks carriers rarely provide unlimited data plans and the service scalability is limited by fundamental constraints. Finally, energy consumption is a critical issue for smartphone users.

With the above constraints in mind, most existing smartphone applications for cloud-storage service follow one important design principle of not keeping local copies of the files stored in cloud because smartphones may not have sufficient space to hold all the files, and downloading those files consumes a lot of bandwidth and battery power. Instead, only meta data is kept on smartphones by default. Though this design is efficient, it limits the capabilities of the applications. Some file operations that can be easily done with local copies become extremely hard, if not impossible, for smartphone users, e.g., compressing files and transferring files to another user.

2.1.3 Real-time Video Streaming Services

Streaming video is another popular service for smartphone users. Most of the popular stream video providers such as Youtube, Netflix, and Hulu have developed mobile applications to serve their clients. However, designing mobile applications for video streaming faces new challenges that do not exist in traditional wired Internet. One of the most critical issues is the network connection. Compared to the wired network users, a mobile user's network bandwidth is much limited. All the common connections for mobile users such as WiFi, 3G, and 4G have significantly lower throughput and the link qualities are heavily affected by environmental factors such as obstacles and distance to infrastructure nodes. In addition, user mobility is another unique feature for mobile users. When a user is mobile, the wireless link quality could be highly dynamic and a user can be temporarily disconnected in a certain region. Along with the movement, a user could also trigger the handoff protocol to switch the associated infrastructure node. All these dynamics in a wireless network serving mobile users make the design of streaming video mobile applications more challenging. Video streaming is a real-time service and extremely sensitive to the change of network conditions. Any network jitter or delay could pause the playback of the video clip. In addition, network cost is another issue that should be considered when developing a mobile app for streaming videos. On one hand, end-users may want to consume bandwidth as little as possible when watching the video because the video delivery may incur a cost, e.g., 3G or 4G, and additional energy consumption. On the other hand, more importantly, the service providers may want to save the network bandwidth cost while serving the end-users.

A rule of thumb to achieve is to deliver only the necessary video data to the users. With mobile users, it is more feasible to manage to achieve this goal because the mobile streaming video applications are usually developed by the service providers. Compared to the means of accessing online video in the traditional network, e.g., via a general web browser, mobile streaming applications enable the service providers with more flexible functions to manage the data delivery from the source server to the end users. However, how to identify the changes of link quality and to predict the trend the changes of a mobile user remains a challenge.

2.2 Dissertation Contributions

To address the above research problems, this dissertation designs, develops and evaluates several novel communication models and enhanced applications. In the rest of this section, we briefly discuss the each of the three representative mobile service categories and summarize the contributions in each field.

Location-based services: In this field, our objective is to use MANETs to provide an alternative network architecture to the infrastructure-based client-server communication model. With this target in mind, we make the following contributions in this dissertation.

(1) We propose long-range radio assisted communication model(**LAAR** [24, 25]) for regular data transmission where a long-range, low cost, and low rate radio is integrated into smartphones to assist regular radio interfaces such as WiFi and Bluetooth. LAAR uses the long-range radio to carry out small management data packets to improve the routing protocols. Specifically, we develop new schemes to improve the efficiency of the path establishment and path recovery process in the on-demand Ad-Hoc routing protocols.

(2) For local message dissemination, we build a system upon a new communication model, called passive broadcast (**PASA** [26]). It is a connectionless and receiver-initialized model where each node periodically *scans* other nodes in the communication range and obtains their data if available. The representative carriers of PASA in reality include Bluetooth and WiFi-Direct, both of which define

a mandatary 'peer discovery' function to fetch basic information about nearby devices. This function can be easily extended to implement passive broadcast mechanism without modifying the existing network protocol stack.

(3) Based on PASA, we build a Mobile Message Board (**MMB** [27]), a location-based service for smartphone users to post and share messages in a certain area. Our algorithm design focuses on the message management on each phone considering its own schedule of turning the wireless device on and off. We present algorithms for two different cases to maximize the availability of the messages.

(4) We implement and evaluate our solutions on the Android smartphone testbed through small scale experiments. Furthermore, we collect the results from simulations for large scale settings to validate the scalability of our proposed approaches.

Cloud-storage services: Our basic idea to enhance cloud-storage services is to launch a cloud instance to assist users to accomplish advanced file operations. By using the resources of the instance, smartphone users will significantly reduce the bandwidth consumption for file operations. To implement this idea, we develop **Skyfiles** [28] system for smartphone users to manage their files in cloud storage with more capabilities. The main contributions of Skyfiles lie as follows.

(1) It extends the available file operations for mobile devices to a more enriched set of operations including downloading, compressing, encrypting, and converting operations.

(2) It includes a protocol for two smartphone users to transfer files from one's cloud storage space to the other's cloud storage.

(3) It includes secure solutions for all the above operations to use shared cloud instances, i.e., instances created by other users.

Real-time video streaming services: In this category, we target on improving video prefetch/buffer mechanism that is commonly used in video streaming services. We develop an efficient and dynamic video buffer control scheme named (**DAB** [29]) that tries to keep a smooth playback with minimum data delivered to the user by adjusting the buffer size. DAB contains the following contributions.

(1) It takes the mobility of smartphone users into consideration and combines the signal strength as well as accelerometer measurements to dynamically adjust the buffer size.

(2) It is implemented on Android smartphones to play Youtube videos and evaluated with real users' mobility traces.

2.3 Dissertation Organization

The rest of this dissertation is organized as follows. In Chapter 2 and 3, we investigate two new efficient Ad-Hoc communication models for smartphones as well as a suite of new techniques to significantly improve the efficiency. Both models target on reducing the cost of route establishment and maintenance, while one model is for bulk data communication and the other model is for short message dissemination. Specifically, we present **LAAR** for bulk data communication in Chapter 2 and discuss the details in protocol design, system implementation and performance evaluation. Chapter 3 presents **PASA** for short message dissemination and discusses the system model, problem formulation, parameter optimization and performance evaluation. In Chapter 4, we propose **MMB** system for nearby smartphone users to share messages with each other based on PASA communication model. MMB provides an alternative to traditional infrastructure-based client-server architecture for location-based services. **Skyfiles** is proposed in Chapter 5 to improve the cloud-storage services by utilizing a cloud instance. Chapter 6 is our work of **DAB** that enhances the real-time video streaming services. Finally, we conclude the dissertation in Chapter 7.

CHAPTER 3
LONG-RANGE RADIO ASSISTED AD-HOC NETWORKS

In this chapter, we study smartphone-based Ad-Hoc networks to support applications that require interactions and communications in the proximity. It is motivated by the fact that location plays an extremely important role in mobile applications. With an Ad-Hoc network, hearing the signals from the other users or service providers is the best evidence that the user is close to the each other. Therefore, constructing a mobile Ad-Hoc network (MANET) with hop-by-hop communication to carry local data traffic is desirable in practice. However, the current routing protocols used in MANETs suffer from two major problems. First, it is costly and inefficient to establish a path from the source to destination. Traditional MANET routing protocols either pay a high cost for maintaining routing tables or flood a request message in the entire network for on-demand path discovery. Both categories require a large number of messages to be delivered for establishing a path. The second problem resides in the path recovery protocol which is usually triggered by an observed failure and proceeds by repeating the path establishment process. In practice, however, this recovery process is slow. routingtable

To address the problems, we investigate a new *long-range radio assisted* Ad-Hoc communication model for smartphones as well as a suite of new techniques to significantly improve the performance of path establishment and recovery. We have prototyped this model on commercial Android phones by integrating additional long-range radio chips. Specifically, we adopt XE1205 [30] which features low cost (<\$30), low power ($10 \sim 20$mA current), and a communication range of 1.6 miles. However, as a tradeoff, its data rate is low (tens of bps) unsuitable for bulk data transmission. We propose to use this additional radio channel for control and management messages while data communication is still carried by WiFi or Bluetooth.

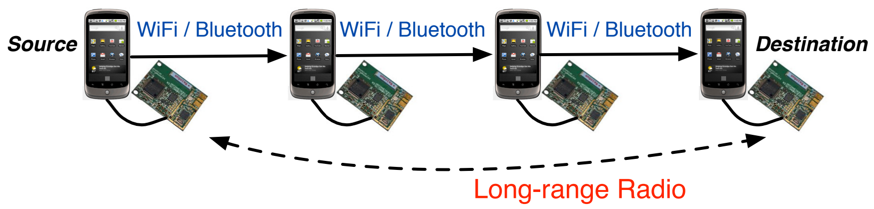


Figure 3.1: Long-range Radio Assisted Model

3.1 Related Work

Generally, there are two types of routing protocols in MANETs. One type is proactive protocols such as DSDV [31], OSLR [32]. In these protocols, each node maintains one or more tables with routing information to every other node in the network. The other type is on-demand protocol (reactive), such as DSR [33], AODV [34]. In on-demand protocols, the routes are created as required. In this chapter, we mainly focus on the on-demand routing protocol design.

A lot of innovative approaches in this area have been studied to improve the network performance from different aspects. For example, in LQSR [35], the route is selected based on link quality metrics, such as expected transmission count (ETX), per-hop RTT, and per-hop packet pair. However, LQSR only works with static setting and fails to deal with topology changes. Frey et al. [36] focus on geographic routing to overcome topology changes with mobility. Another important aspect is to utilize multiple resources on one node to achieve better performance. For example, [37] attempts to use multi-channel on one node and proposes a hybrid channel assignment strategy. Multiple resources on one device make the routing protocols more flexible. MR-LQSR [38], AODV-MR [39] and Extended-DSR [40] assumes that each node is equipped with multiple radio interfaces. MR-LQSR uses a new metric named weighted cumulative expected transmission time(WCETT) to provide better route selection. The AODV-MR uses the multi-radio interfaces communication to improve spectrum utilization and reduce interference. Extended-DSR attempts to address limited capacity and poor scalability problem by taking advantage of multi-radio feature.

Our work is closely related to [26, 38–42]. However, in their settings, each node is equipped with multiple 802.11 wireless cards. And they focus on addressing issues of interference and channel allocation. In our case, the two interfaces work on different frequencies. We mainly focus on utilizing the collaborations of these radios to boost the network performance. In our previous work [24], we investigate the route construction and as a following work, we design the whole protocol.

3.2 Background and Problem

We target on the routing problem in a MANET where a source node aims to transfer data to a destination node. Each node in our setting is a smartphone and the phone-to-phone Ad-Hoc communication is carried out by WiFi or Bluetooth interface. We adopt on-demand Ad-Hoc routing protocols such as DSR and AODV, where each node does not maintain stateful link information and a path is established only when the source intends to transfer data to the destination. Compared to proactice protocols, on-demand routing protocols are more suitable in a dynamic network considering user mobility. Here we introduce the details of some key components in the traditional on-demand Ad-Hoc routing protocols and list their drawbacks which motivate our new design.

Path Establishment: Establishing a path from the source to destination is a basic and important step in Ad-Hoc routing. The basic design in the prior work is to let the source flood a *route request message* (RREQ) to the entire network until one of them reaches the destination. Then the destination will send a *route replay message* (RREP) to the source tracing back the transmission path of the RREQ. Once the RREP is received by the source, a routing path is successfully established. This flooding-based scheme, however, is costly in two aspects. First, the RREQ message is broadcast by a node to all its neighbors in omni-direction. Most of the messages will never reach the destination. Although RREQ message is often confined with a time-to-live (TTL) parameter, it still causes a large number of useless messages transferred which consume energy of each node and yield wireless signal interferences in the MANET. In addition, the exchange of RREQ and RREP takes a round-trip time with hop-by-hop delivery. Considering the interference and processing time at each relay node, this initial delay could degrade the throughput performance especially when transferring small amounts of data.

Path Cache: Path cache is often included in the MANET routing protocols to avoid unnecessary path establishments. Each node stores the paths from itself to other nodes into a *route cache* based on the RREQs or RREPs it overhears. In the path establishment, when an RREQ arrives, the node will first check its route cache. If there exists a path to the destination in the route cache, the node will reply to the source without propagating the RREQ. In this scheme, the validity of the cached paths is crucial to the performance. Because of the node mobility, the cached paths may not be available when the node intends to use them. In the traditional routing protocols, the validity of the cached paths is never checked. When a node attempts to use a cached path and later finds the path is stale, the source will have to re-establish another path.

Path Recovery: Path recovery is another important component in MANET routing protocols. When a node on the path moves out of the transmission range of its neighboring hops, a link is broken and the path recovery protocol will be triggered to establish another path. In typical MANET routing protocols, the node that fails to receive a link layer acknowledge detects the broken link and sends a *route error message* (RERR) back to the source. The source will first search its route cache for an alternative route to the destination. If no alternative is found, the source initializes a new path establishment process. In addition, any node that receives or overheads an RERR message will delete all the routes in the cache that contain the broken link. Path recovery inherits the issues we have mentioned for path establishment and path cache.

Our Problem Setting: In this chapter, we aim to develop an efficient MANET routing protocol based on the integration of a long-range radio interface that helps address the above issues. Specifically, we consider a MANET consisting of smartphone nodes and each smartphone is equipped with two heterogeneous wireless interfaces: one is the regular wireless radios such as WiFi and Bluetooth, and the other is a new long-range radio. According to our prototype, the long-range radio has a much longer communication range (up to miles) than WiFi or Bluetooth. Its power consumption is extremely low which is suitable for smartphones. However, the network bandwidth of the long-range radio is significantly lower than the regular radios. More details of the hardware characteristics will be introduced in Section 3.4. In our solution, the long-range radio will be used to broadcast small management packets while the data transfer is still carried by the regular radios in a hop-by-hop fashion. In addition, considering the coverage of the long-range radio, this chapter targets at the local data transfer where the source and destination can directly reach each other over the long-range radio. Our goal in this chapter is to use the new long-range radio assisted communication model to improve the performance of MANET routing protocols. Specifically, with the new long-range radio interface, we aim to reduce the message flooded in the entire network, decrease the time overhead of establishing or recovering a path.

3.3 System Design of LAAR

In this section, we present our solution LAAR which mainly consists of an efficient path establishment protocol and path recovery protocol. In addition, we develop a new route cache management scheme that serves both path establishment and recovery protocols.

3.3.1 Path Establishment

In this subsection, we discuss the motivations and technical details of path establishment in LAAR system.

3.3.1.1 Motivations

Our design of the path establishment process includes the following two major new techniques.

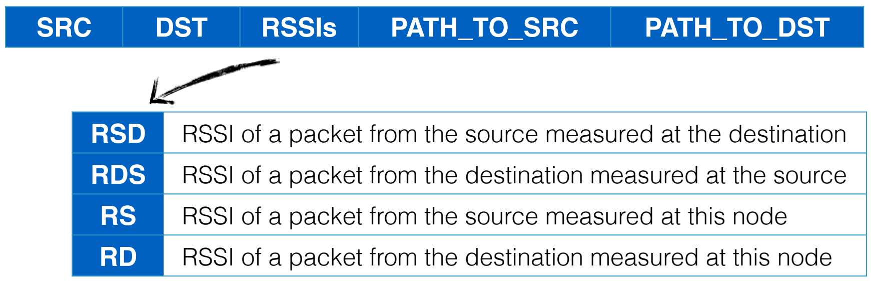
Bi-directional Route Request Flooding: Traditionally, the route request message (RREQ) is flooded from the source towards the destination. In our problem setting, the source and destination are directly connected over the long-range radio. Thus, before flooding the RREQ message, the source can use the long-range radio to notify the destination about the upcoming communication session. An then, the destination will participate in this process as well. In out solution, therefore, both the source and destination flood the RREQ message towards each other. When a node receives both request messages implying a path has been established, it can send an announcement message through its long-range radio.

RSSI-guided Flooding: Traditionally, the RREQ messages are forwarded to all directions and most of them are wasted. In our solution, we confine the region involved in the flooding process by considering the received signal strength (RSSI) of the packets sent over the long-range radio. The basic intuition is that for any communication session, only the nodes "between" the source and destination should be involved in propagating the RREQ messages. For example, if a node is further away from the destination than the source, it should not forward the RREQ for the session. Our solution provides relevant RSSI information to each node to help determine if the node should participate in the flooding process.

3.3.1.2 Complete Path Establishment Protocol:

Our path establishment protocol includes three phases. In the first phase, we develop a new *three-way handshake protocol* to enable the bi-directional route request flooding and prepare for a *RSSI-guided flooding*. In the second phase, the source and destination each broadcasts an RREQ and all the participating nodes propagate the RREQs as in the traditional routing protocols. The third phase is for announcing an established path. We assume that each node maintains a regular routing table that hosts the routing information for all the active communication session it participates. The routing table contains at least three columns: the source node ID (SRC), the destination node ID (DST), and an ordered list of node IDs that represent the path to the destination (PATH_TO_DST). After a path is successfully created, each node in the path will add a new entry in its routing table for this session.

In our solution, each node keeps an additional table, called *preparation table*, to help decide whether the node will participate in a path establishment process. The structure of a preparation table is illustrated in Fig. 3.2. Compared to the routing table, each entry in the preparation table includes some more fields such as PATH_TO_SRC and four RSSI values.



SRC	DST	RSSIs	PATH_TO_SRC	PATH_TO_DST

RSD	RSSI of a packet from the source measured at the destination
RDS	RSSI of a packet from the destination measured at the source
RS	RSSI of a packet from the source measured at this node
RD	RSSI of a packet from the destination measured at this node

Figure 3.2: Preparation table structure

Next, we present the details of the three phases where the preparation tables will assist to eventually update the routing tables on the path from the source to destination.

Phase I - Three-way handshake: Assume the source S tries to send data to the destination D and they are within each other's communication range over the long-range radio. S first sends out an **INIT** message including S and D's IDs via the long-range radio. The combination of the source and destination's IDs $< S, D >$ uniquely identifies a communication session. Once receiving the message, D sends an **INIT-ACK** message back to S. Besides the source and destination's IDs, this message also includes the RSSI of the INIT message indicated by $R_{S \to D}$. Finally, S sends the last handshake message **INIT-FIN** including the RSSI of the INIT-ACK message ($R_{D \to S}$). The structures of these three messages are illustrated in Fig. 3.3.

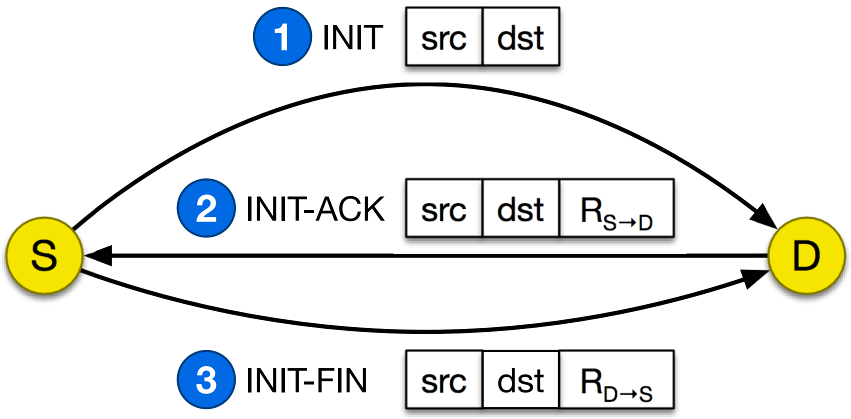


Figure 3.3: Three-way handshake protocol

When overhearing the three-way handshake messages, every node that is neither source or destination applies the following Algorithm 1. Basically, the preparation table hold the candidate sessions that may be added into the routing table later. When a node receives an INIT message (lines 3–4), it adds a new entry into the preparation table recording the new session as well as the RSSI of this message (RS). INIT-ACK and INIT-FIN messages confirm the awareness of the incoming path establishment process at the source and destination, and also help enforce the RSSI-guided flooding. When receiving an INIT-ACK message (lines 5–11), the node first searches its preparation table for the matching session with <src, dst>. If a matching entry E is found, the node will compare the recorded RS value with the RSSI value ($R_{S\to D}$) in the INIT-ACK message. This entry E will be removed from the preparation table if E.RS$< \beta \cdot R_{S\to D}$ where $\beta \in (0,1)$ is a threshold depending on the signal prorogation model. This step filters out the nodes that are further away from the source node than the destination. If E.RS$\geq \beta \cdot R_{S\to D}$, the node will update the entry E by setting E.RD value to be the RSSI of this INIT-ACK message. Similar steps are applied when processing an INIT-FIN message (lines 12–18).

Algorithm 1 Process Three-way Handshake Messages

```
1: function Receive(msg):
2: Read msg.src, msg.dst, and measure the RSSI of the message indicated as msg.rssi
3: if msg is an INIT message then
4:     Add a new entry {SRC=msg.src, DST=msg.dst, RS=msg.rssi} into the preparation
       table
5: else if msg is an INIT-ACK message then
6:     Search the session <msg.src, msg.dst> in the preparation table
7:     if there exists an entry E for the session then
8:         if E.RS< $\beta\cdot$ msg.$R_{S\to D}$ then
9:             Remove this entry E from the preparation table
10:        else
11:            E.RSD=msg.$R_{S\to D}$ and E.RD=msg.rssi
12: else if msg is an INIT-FIN message then
13:     Search the session <msg.src, msg.dst> in the preparation table
14:     if there exists an entry E for the session then
15:         if E.RSD value is null or E.RD< $\beta\cdot$ msg.$R_{D\to S}$ then
16:             Remove this entry E from the preparation table
17:         else
18:             Update the entry by setting E.RDS=msg.$R_{D\to S}$
```

Phase II - Bi-directional RREQ Flooding: In phase II, both the source and destination will start flooding an RREQ message towards each other. Essentially, a node will participate in flooding an RREQ only if its preparation table contains an entry for the session of the received RREQ. In our solution, an RREQ message includes source/destination IDs, a TTL value, the nodes it has traversed (i.e., the path), and an additional field indicating the origin of the message, i.e., from the source node or destination node. The following Fig. 3.4 illustrates the message structure. Having received an RREQ, each node applies the following Algorithm 2. First, it

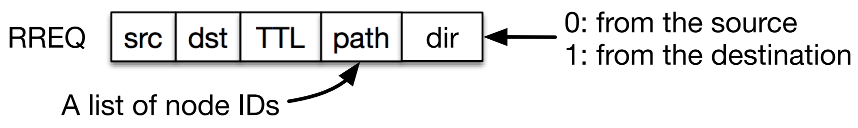


Figure 3.4: RREQ message format

searches its preparation table for the session this RREQ represents. If there exists an entry for the session, the node will add its own ID in the field of the path and further broadcast the RREQ if the TTL is not expired. Meanwhile, the node will add the path included in the RREQ into either E.PATH_TO_SRC or E.PATH_TO_DST according to the origin of the RREQ. If the node finds that both E.PATH_TO_SRC and E.PATH_TO_DST have been filled, it will move to Phase III to announce the established path.

Algorithm 2 Process RREQ Messages

```
1:  function Receive(msg):
2:  Read msg.src and msg.dst, and search the preparation table
3:  if there exists an entry E for the session then
4:      if msg.dir indicates msg is from the source then
5:          if E.PATH_TO_SRC = null then
6:              E.PATH_TO_SRC = msg.path
7:      else
8:          if E.PATH_TO_DST = null then
9:              E.PATH_TO_DST = msg.path
10:     if E.PATH_TO_SRC and E.PATH_TO_DST are defined then
11:         SendAnnouncement(E)
12:     else if msg.TTL>0 then
13:         Broadcast a new RREQ {msg.src, msg.dst, msg.TTL-1, msg.path+NodeID,
            msg.dir}
```

Phase III: Path Announcement Once a node receives the RREQ messages for the same session from both sides, it will broadcast an announcement message

(**ANNO**) via the long-range radio with a complete path from the source to destination. An ANNO message contains only three fields: the source (src), the destination (dst), and the full path from the source to destination. After receiving the ANNO message, every node will no longer forward the RREQ message for this session (by removing the entry for the session in the preparation table). In addition, each node checks the path and adds the session to its routing table if it is listed in the path.

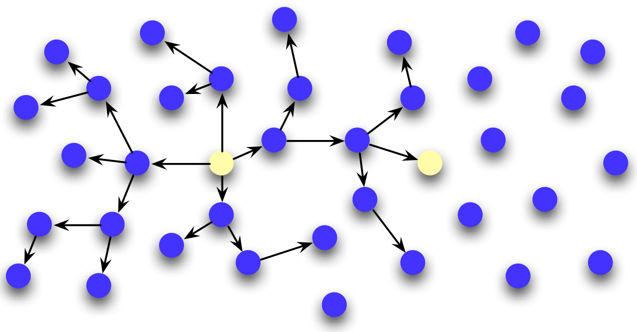


Figure 3.5: Traditional Path Establishment (TTL=4)

Fig. 3.5 and Fig. 3.6 show a comparison between traditional path discovery and our long-range radio assisted path discovery. The two orange nodes are sender S and receiver D. Fig. 3.5 shows the request message flooding with TTL (time to live) set to 4. The shortest path from sender to receiver is 3-hop long and in this partial topology,

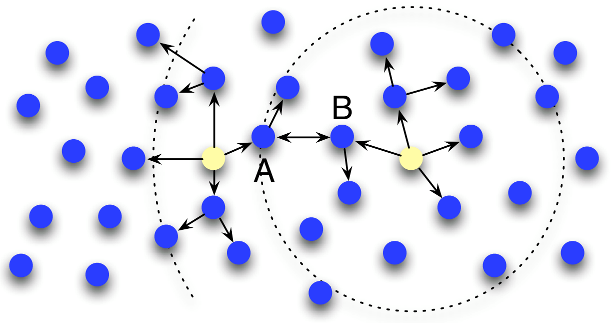


Figure 3.6: Long-range Radio Assisted Path Establishment

14 nodes broadcast the request when it reaches the receiver. Fig 3.6 illustrates the benefits of bi-directional flooding and RSSI filtering. In this example, the request is propagated from both sender and receiver and the path is established in the second round of broadcast, i.e., when node A and B broadcast their received requests. With the handshake messages including RSSI information, we assume the dotted circle and arc centered at the receiver define the region where the RSSI of the receiver's packets is similar, i.e., $R_{D \to S}$. Assume the nodes on the left side of the dotted arc have RSSIs (R_D) smaller than $\beta \cdot R_{D \to S}$. Thus they will node forward the RREQ message. Only 7 nodes broadcast the RREQ message in Fig 3.6 when the path is established.

3.3.2 Path Recovery

Path recovery is a critical component in MANETs because of the dynamic network topology caused by user mobility. We develop an efficient path recovery protocol in LAAR with the following two new techniques. Due to the page limit, we omit the detailed pseudo codes for the protocols.

Partial Path Recovery: In the prior work, once a node detects a broken link, the notification will be sent back to the source by an RERR message, and then the source will launch a new path establishment process. Therefore, any single link failure will lead to a complete path establishment process which is not efficient in practice, especially if the broken link is shared by multiple active sessions. An example is shown in Fig. 3.7. In the traditional MANET routing protocols, while node V moves away causing a broken link, node U will send three RERRs to the sources which will further start three path establishment processes.

In our partial path recovery solution, the node who detects the failure will notify the sources with a single RERR over the long-range radio and start path establishment processes with the destinations. Referring to the example in Fig. 3.7, node U will attempt recover the paths from U to D1 and D2. If successful, node U will notify the sources about the recovered paths over the long-range radio. Meanwhile, each source node also sets a timer once receiving an RERR. If the detecting node cannot recover the path to the destinations before the timers expire, the sources will initialize the path establishment process with the destinations.

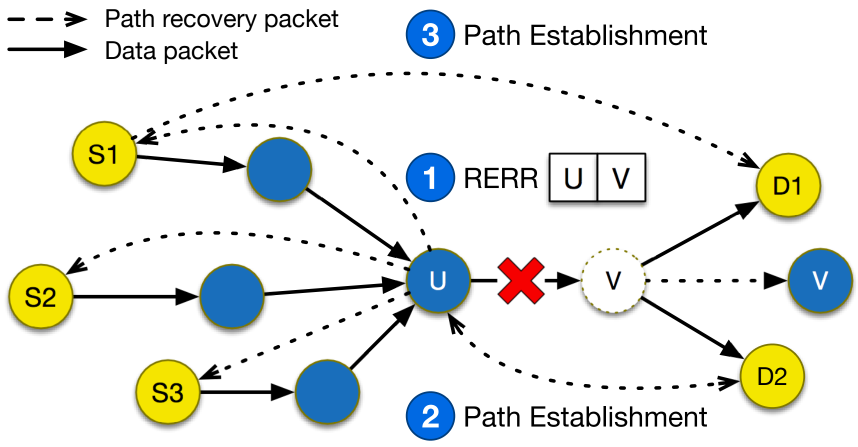


Figure 3.7: An example of partial recovery: 3 active sessions (S1,D1), (S2,D2), and (S3,D2) share a link U→V. Node V moves away and the link U→V is broken.

Proactive Path Recovery: The other new technique we develop is to proactively start path recovery protocol before any link is broken. The basic idea is to detect weak or about-to-break links based on each phone's mobility. Considering smartphones being the mobile nodes in our setting, we particularly use the accelerometer and RSSI measurements to determine if a node is moving away from a path it belongs to. If a node detects high movements or poor RSSIs from its neighbors, it will notify the neighboring nodes about the possible departure. Then a path recovery process will start when the node still carries out the data transfer. Once a new path is established, the neighboring nodes will update their routing tables to bypass the departing node.

In practice, both accelerometer and RSSI readings are dynamic, and may not accurately reflect the user movements. We develop a heuristic algorithm that combines these two measurements to indicate if a node will cause a broken link soon. Algo-

Algorithm 3 Proactive Path Recovery

```
1: $RL$: avg RSSI level, $SL$: avg speed level, $C = 0$
2: When a new packet is received, update $R$
3: if $RL$ is $GOOD$ then
4:     $C = 0$; return;
5: else if $RL$ is $POOR$ then
6:     $C = 0$; Start the recovery process;
7: else
8:     Measure the user's moving speed $SL$
9:     $C = \begin{cases} C + \Delta_H & : \text{if } SL \text{ is } HIGH \\ C + \Delta_M & : \text{if } SL \text{ is } MEDIUM \\ C + \Delta_L & : \text{if } SL \text{ is } LOW \end{cases}$
10:    if $C \geq \tau$ then
11:        $C = 0$; Start the recovery process;
```

rithm 3 illustrates the detailed process when a packet arrives. Specifically, we define three discrete levels for RSSI values, {*GOOD, FAIR, POOR*}. While a *GOOD* RSSI indicates a stable link, a *POOR* RSSI will trigger a path recovery process. When a *FAIR* RSSI is received, our solution will start to periodically measure the accelerometer. Our intuition is that a highly mobile user with *FAIR* RSSIs is likely to cause a broken link. Similar to RSSI measurements, we use three levels, {*HIGH, MEDIUM, LOW*}, represent a user's moving speed. Our algorithm use a variable C to track the user speed accumulation. According to the speed level, we increase C with heuristic values (line 9, $\Delta_H > \Delta_M > \Delta_L$). When C exceeds a threshold τ, the path recovery process will be started.

3.3.3 Route Cache Management

In MANET routing protocols, every node records the known paths in a route cache to avoid the delay of path establishment. For both path establishment and path recovery protocols, route cache plays an important role. When processing an RREQ message, a node will first check its route cache and if a matching path to the destination is found, the node will reply to the source without further flooding the RREQ. However, the paths in the route cache are not verified until the node decides to adopt them. In a MANET, the link conditions are dynamic and the known paths may not be stable. Using stale paths will cause path recovery once a broken link is detected and yield a worse performance than not using the route cache.

In our solution, we address this issue by removing invalid paths in the cache based on the overheard packets. Two types of packets will trigger a cleansing of the route cache. First, if a node receives a broadcast RERR packet over the long-range radio indicating a broken or about-to-break link, it will search its route cache and eliminate all the paths that contain the link specified in the RERR. Second, every node will listen to the active data transmissions over WiFi or Bluetooth from the neighboring nodes even if the packets are not designated to it. By sniffing these packets (e.g., from node j to node k), node i can measure the RSSI and estimate the quality of the link $j \rightarrow i$. If the RSSI is in the category $POOR$, node i will remove all the paths in its cache that contain the link $j \rightarrow i$.

Complete Protocol in Path Recovery: Here is the complete protocol. When the recovery process is initialized by node i, it first checks its path cache to see if there is another route to the destination. If an alternative route is found, node i sends a **recovery message** that contains the new route back to the source. When the source receives the **recovery message**, it will use this route for further transmission.

If there is no available route to the destination in the cache, node i will send a **route error message** to the source, and it will start the path establishment process with the three-way hand-shake protocol with the destination. When a path is successfully established from node i to the destination, node i will re-assemble a complete path from the source to the destination. This new path will be included in a **re-assemble reply** message which will be sent to the source by node i over the long-range radio.

Upon receiving the **route error message** from detected node, the source first suspends the transmission. Then, it sets a timer and waits for a **re-assemble reply** message. If a **re-assemble reply** message arrives, the source will use the new path for the rest of transmissions. If the timer expires with no **re-assemble reply** message, the source will start a path establishment process to find a path to the destination.

3.4 System Implementation

In this section, we introduce our implementation of LAAR with off-the-shelf devices. In our prototype, we attach a TinyNode [43], which includes a long-range radio transceiver, Xemics XE1205 [30], to an Android smartphone. XE1205 operates on 915Mhz and feature low cost, low power consumption, and a communication range of 1.6 miles. We have integrated the long-range radio into assorted phones including HTC Magic phone, Nexus One phone, and Nexus 4 phone. We use PL2303 [44] USB-to-Serial bridge controller to connect TinyNode and smartphone (through either ExtUSB or MicroUSB port).

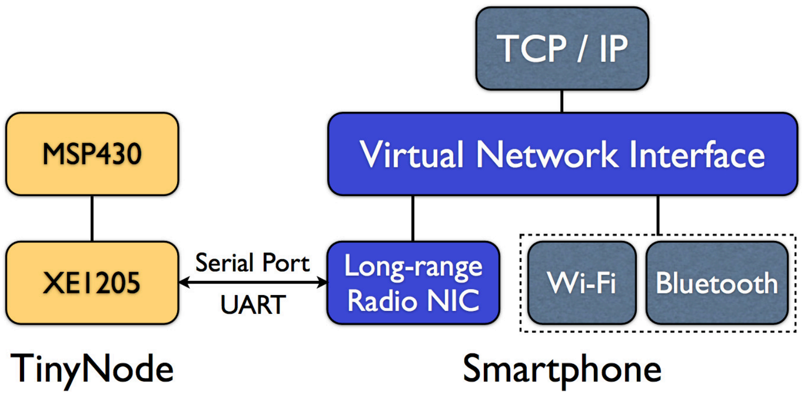


Figure 3.8: Software Architecture

Software support includes programs on both smartphones and the external devices. Fig. 3.8 illustrate the design architecture with TinyNode. We have customized Android kernel and developed user space programs on smartphones to support dual radio communication. Basically, the USB port of a phone is recognized as a serial UART device (Universal Asynchronous Receiver/Transmitter) and a device file for it is created under '/dev/'. User programs can communicate with the USB port by reading from or writing to the new device file. Communication between a TinyNode and smartphone is built on a module deployed on both sides. We have implemented data-link level protocol over this serial link (UART) communication including basic mechanisms such as checksum and retransmission. In addition, we use TUN/TAP device driver [45] to create a virtual network interface and change the routing policy on phones such that all incoming and outgoing traffic will pass through the virtual interface. In our solution, TUN is used for routing, while TAP is used for creating a network bridge. Then we have developed programs in TUN/TAP driver to process each packet. Our prototype smartphone is able to dispatch each packet to different network interfaces, either WiFi, Bluetooth, or the long-range radio. Fig. 3.9 shows two prototype smartphones equipped with TinyNode conducting a ping test with dual radio model.

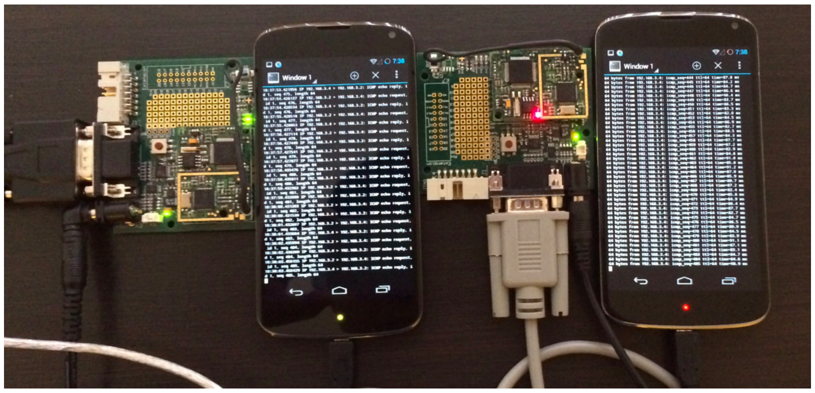


Figure 3.9: Demonstration of the Dual Radio Model

3.5 Performance Evaluation

In this section, we evaluate LAAR and compare it with the conventional MANET routing protocols. The results are drawn from the experiments on basis of a small scale network and NS2 [46] simulation on basis of a large scale network. Our major performance metrics are overhead, number of messages transferred, and network throughput.

We compare LAAR with DSR [33], DSR-R0, and AODV-ERS [47]. DSR-R0 is the default implementation of DSR in NS2 and improves DSR with a *ring-zero search* scheme in the path establishment. Ring-zero search aims to reduce the overhead by firstly sending an RREQ with TTL=0. If the sender and the receiver are direct neighbors, the path would be quickly established. Otherwise, upon a timer expires, the sender will send another RREQ with a regular TTL value. AODV-ERS is an enhanced version of AODV [34] with expanding ring search, where the sender broadcasts the RREQ for multiple rounds each with an incremental TTL value. The process terminates when the destination is reached.

Workloads: We consider brochure dissemination application for our evaluation. We collect a set of real brochure files for our tests considering the following cases where a MANET could help disseminate the files. *(1) Advertisements in a mall*: The stores in a mall may want to attract nearby customers by delivering their advertisements or coupons. *(2) Subway map and schedule*: Wireless signals are often poor in subway stations or tunnels. With an effective MANET, the subway administrator can simply deploy a standalone WiFi device to deliver map or schedule files to the commuters without any infrastructure support. *(3) Crowded events*: In an event with a large number of attendees, the infrastructure-based network may have scalability issue because of the limited capacity. [1] With a MANET setting, the attendees can easily check the schedule of shows and other information without connecting to the Internet. Our evaluation uses the sample workload in the following Table 3.1.

[1]For example, it was reported that more than 3 millions people attended the 86th Annual Macy's Thanksgiving Day Parade and the explosive users' demand within central park west area caused serious congestions in mobile networks.

Table 3.1: Brochure Dissemination Workload

Case	Content	Format	Size
1	Homedepot 20% off coupon	PDF	213KB
2	MTA(New York) Map	PNG	344KB
3	Target Black Friday 2014	HTML	915KB
4	MBTA(Boston) Schedule	PDF	1.2MB
5	AT&T Cyber Monday Sale 2014	PDF	2.1MB
6	NYC Thanksgiving Parade	PDF	2.8MB
7	Nordstrom Anniversary Sale 2014	PDF	4.2MB
8	Mall of America Direction	SWF	5.2MB

3.5.1 Experimental Results

First, we build a small scale Ad-Hoc network consisting of 6 Android smartphones equipped with the long-range radio. The phone-to-phone Ad-Hoc mode is supported with WiFi and WiFi-Direct. In our experiments, the smartphones are placed at the fixed positions, i.e., the mobility is not considered. We mainly evaluate the data throughput and the performance of the path establishment protocol. The hop distance between the source and destination ranges from 1 to 5.

Fig. 3.10 plots the experimental results with the workload in Table 3.1. The bars show the time consumption of the transmission in each case versus the length of the path. Apparently, the overhead grows along with file size and path length. For example, disseminating 2.1MB (case 5) and 4.2MB (case 7) files takes 4.178s and 6.424s respectively for a 2-hop path. The overheads are increased to 25.575s and 30.134s for a 3-hop path. We observe that a MANET is effective for delivering up to a few Megabytes of data to nearby nodes. For a large file over a long path, e.g., case 8 (5.2MB) with a 5-hop path, the overhead may not be acceptable for users. In practice, considering the dense user population and possible user content sharing, we expect a short path length for any communication session. In addition to the overall performance, we also evaluate the breakdown overhead and try to answer the following questions.

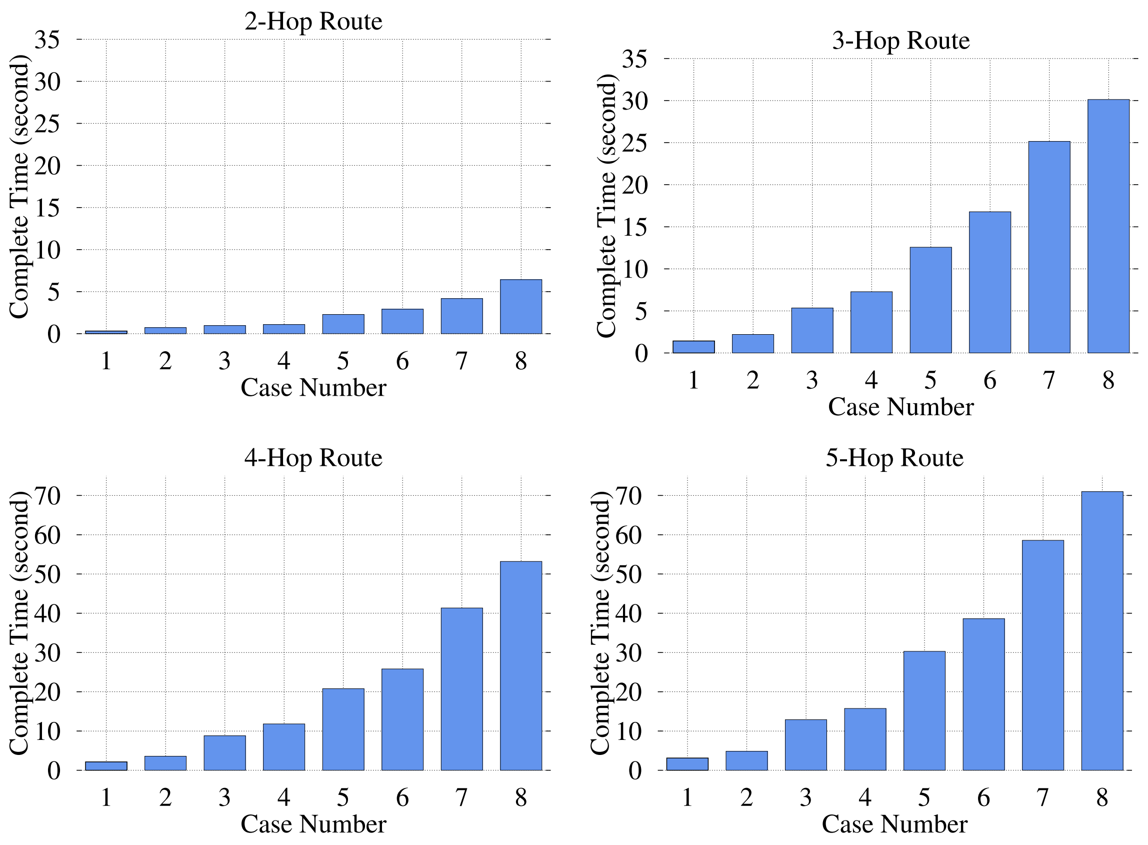
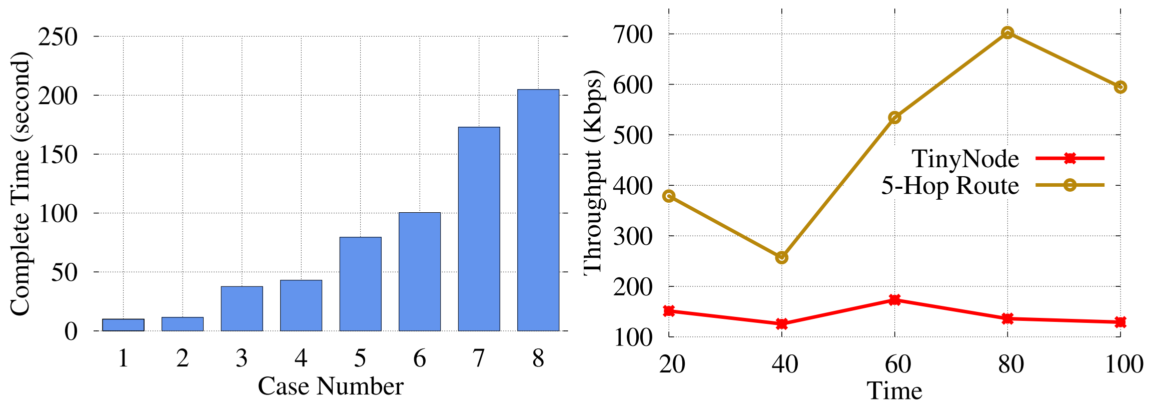


Figure 3.10: The experimental performance of overhead with different path lengths

Can we use the long-range radio for data delivery? The protocol design could be much simplified if the long-range radio can carry out the data transmission. We have conducted the same experiments with direct transmission between two TinyNode devices. The results are shown in Fig. 3.11a. Compared to Fig. 3.10 , the time consumption with the long-range direct link is much higher. Fig. 3.11b further compares the throughput of direct long-range radio link with hop-by-hop transmission along a 5-hop path. In this experiment, we use "iperf" tool to record the throughput every 20 seconds. We observe that hop-by-hop delivery yields a much higher throughput (with a high variance) even over a long path. Overall, we conclude the long-range radio works well for small management packets, but is not suitable for bulk data transmission.



(a) Direct link **(b)** Direct link v.s. 5-hop path

Figure 3.11: TinyNode data delivery performance

3.5.2 Simulation Results

In addition to experiments, we conduct simulation with NS2 to evaluate LAAR in a large scale network.

3.5.2.1 Simulation Settings

In the simulation, we consider the brochure dissemination application in a mall. We run the simulation in following two settings.

- Single store: In this setting, there is only one store trying to send out brochures to the nearby shoppers. We assume that the store periodically broadcasts a short message including a link to the brochure file over the long-range radio. The users can use the link to fetch the brochure. We assume that N users receive the short message and $\alpha \in [0, 1]$ portion of them will be interested in it, i.e., $\alpha \times N$ users will download the brochure.

- Multiple stores: In this setting, there are multiple senders in the mall. Similar to the previous setting, the senders first use periodical short messages over the long-range radio to notify the users.

The parameters in NS2 are set as follows. First, we adopt two-ray ground reflection model and constant speed propagation delay model for wireless signal prorogation. In addition, each node in our LAAR protocol is set with two radios. We modify the NS2 to support two wireless interfaces. The frequency of the long-range radio is set to be 915MHz, and the communication range is configured to be 2500m in receiving (RX) and 3000m in carrier sensing (CS). The other regular radio (short range) is configured to work at 2.4GHz, and the RX and CS ranges are set to be 50m and 100m respectively. For the results shown in this chapter, β is set to 0.9 for the RSSI-guided flooding.

The users in the simulations follow a manhattan grid mobility model [48] with a maximum moving speed of 2m/s. At an intersection, the probability of going straight is 0.5 and taking a left or right is 0.25 each. We generate mobility traces with different numbers of users, and in each trace, users randomly select the initial positions inside a store or on a corridor. For all the tested protocols, we set RREQ's default TTL to 5 if applicable. For speed level with user mobility, the three discrete values in Algorithm 3 are defined LOW (<0.5m/s), MEDIUM ([0.5,1.5)m/s), and HIGH (>1.5m/s). In addition, $\Delta_H = 2$, $\Delta_M = 1$, $\delta_L = 0$, and $\tau = 3$.

3.5.2.2 Single Store

In this setting, we choose Nordstrom (case 7) as our sender and conduct the simulations with different values of the parameters N and α. For each particular setting, we randomly generate 100 mobility traces for tests, and present the average values in the following figures.

Path establishment: First, we set N to 200 and 300, and change the value of α to control the total concurrent sessions in the network. The overhead performance of initial path establishment is shown in Fig. 3.12.

In our setting, the average number of neighbors is 23.9 and 34.3 for $N = 200$ and $N = 300$ respectively. The dense topology can lead to a serious congestion in the existing routing protocols, for example, as shown in Fig. 3.12b, to construct 9 ($3\% \times 300$) concurrent sessions, DSR, DSR-R0 and AODV-ERS uses 514ms, 598ms and 459ms, respectively. However, in LAAR, the overhead of path establishment remains low, because our design reduces the number of messages transferred mitigating the effect of congestion.

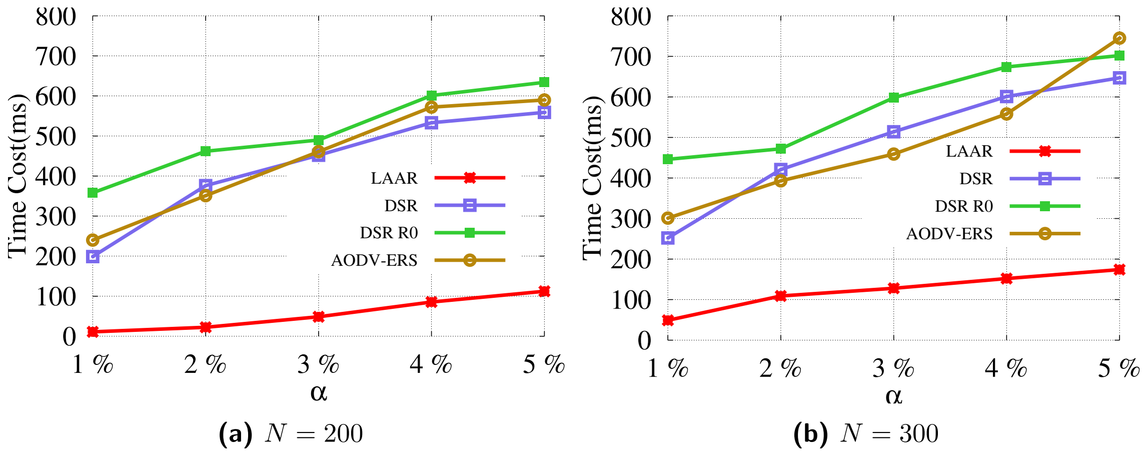


Figure 3.12: Average overhead of path establishment (single store)

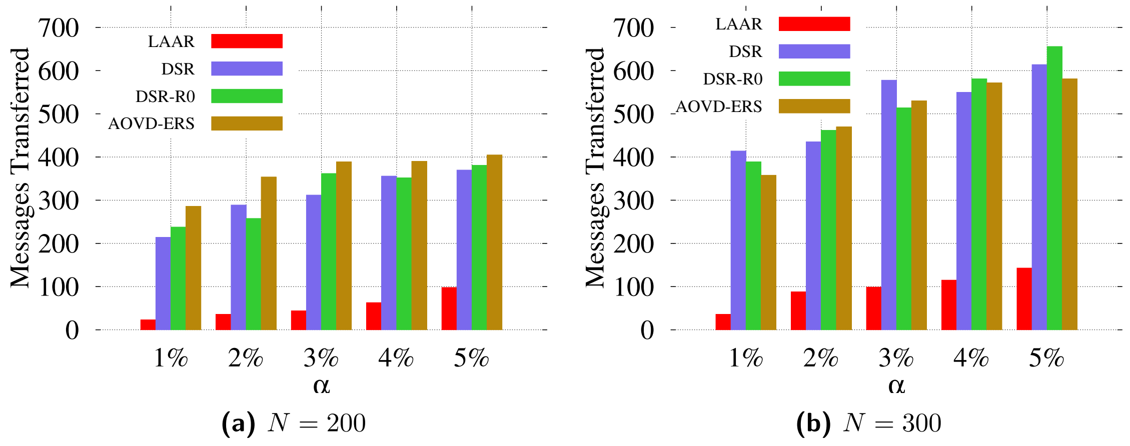
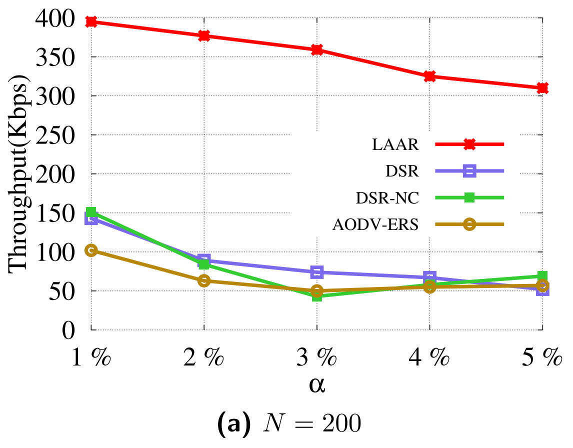


Figure 3.13: Average number of messages transferred to establish path in single store: varying α with 200, 300 users

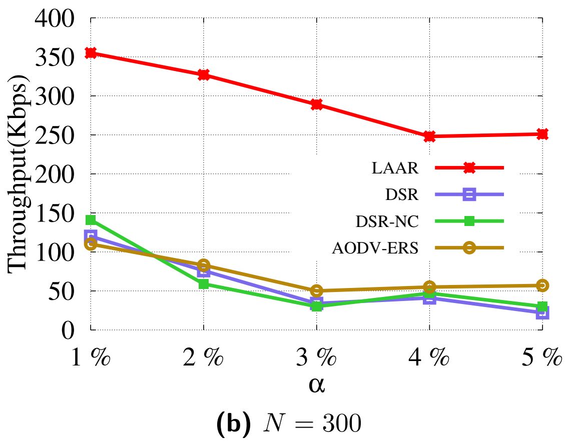
Fig. 3.13 illustrates the number of messages (RREQs) transferred in the entire network. The bars indicate a similar trend in all the protocols. Our solution LAAR significantly outperforms the other three protocols.

Overall throughput: We use throughput as an overall performance metric taking full mobility trace and link breaks into consideration. Since DSR and DSR-R0 use the same path recovery protocol, we do not include DSR-R0 in this test. Instead, to better study the impact of stale routes in the cache, we evaluate a DSR protocol that does not use route cache.

The results are compared in Fig. 3.14. LAAR maintains a high throughput with different α. For example, with $N = 300$ and $\alpha = 3\%$, the throughputs of DSR, DSR-NC, AODV-ERS and LAAR are, 34.1, 30.7, 50.8, 289.1Kbps. LAAR's throughput is more than five times the throughput of the second best protocol, AODV-ERS. The major reason of the significant improvement is the efficient path recovery protocol in LAAR. Our solution greatly reduces the overhead and congestion during a recovery process, and also improves stability of the selected path and the effectiveness of the route cache.



(a) $N = 200$

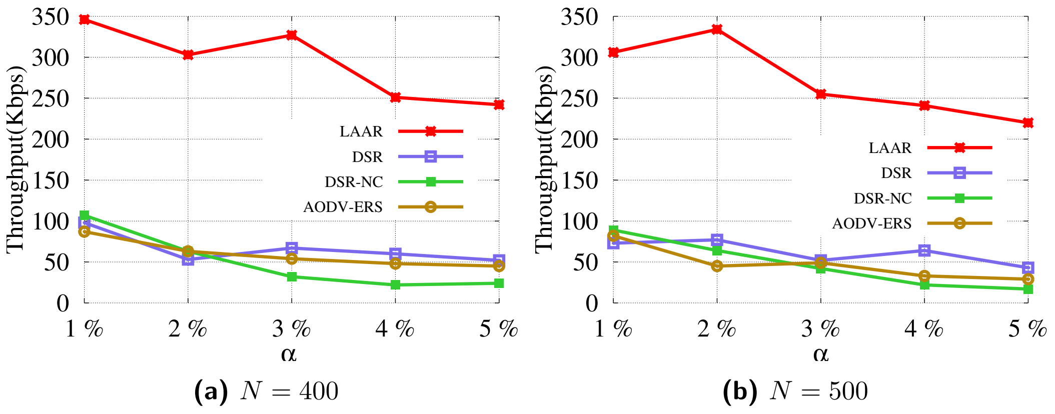


(b) $N = 300$

Figure 3.14: Average throughput in single store

3.5.2.3 Multiple Stores

Finally, we test with three stores, Target (case 3), AT&T (case 5) and Nordstrom (case 7) as our senders. Each store tries to disseminate its brochure listed on Table 3.1. In our configuration, the α for each store's brochure is the same. Thus, the total number of transmissions in the network is $3 \times \alpha \times N$. We collect the throughput from each transmission session and show the average result for each different α value.



(a) $N = 400$ **(b)** $N = 500$

Figure 3.15: Average throughput in multiple stores (3 stores)

Overall throughput: Fig. 3.15 plots the throughput with different values of α when $N = 400, 500$. Obviously, LAAR performs the best among the four tested protocols. For example, in Fig. 3.15a, the throughputs of LAAR, DSR, DSR-NC, AODV-ERS are 327.4, 67.0, 32.5, 54.4Kbps with $\alpha = 3\%$, respectively. We also find that the throughput of LAAR is not always inversely proportional to the increase of α. For instance, in Fig. 3.15b, the throughputs are 306.1, 334.4Kbps for $\alpha = 1\%, 2\%$. With more users involved in the transmission, our techniques of proactive path recovery and route cache management wii be more effective helping improve the throughput performance.

Fig. 3.16 shows the throughput with different number of users (N). The values of α in Fig. 3.16a and Fig. 3.16b are set to 5% and 15%, respectively. Again, LAAR outperforms the other three protocols.

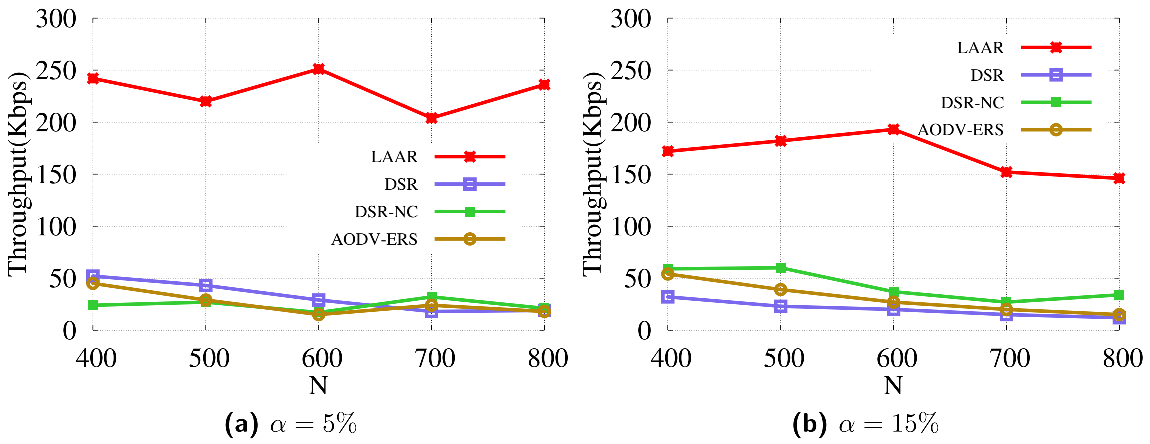


Figure 3.16: Average throughput in multiple stores (3 stores)

3.6 Summary

This chapter presents LAAR, a new dual radio model for smartphone-based Ad-Hoc networks. We integrate a long-range radio to help improve the performance of path establishment and recovery which are critical components in the routing protocols. The experimental and simulation results show that LAAR dramatically improves the performance.

CHAPTER 4

PASSIVE DELIVERY IN AD-HOC NETWORKS

In Chapter 3, we focus on short message dissemination in the proximity. It represents a category of applications with high potential if communication between nearby devices are well supported. For example, a user may want to share his recent tweets or facebook messages with other people sitting in the same room; a police car on site of car crash may disseminate the accident information to other cars within one mile distance; a student in library may chat with his friend in another classroom via instant messenger; a bunch of sport fans may want to share the comments with each other on the same game they enjoyed; a victim of crime or natural disasters may want to ask for help when there is no cellular network or not cannot use it. The current location-based service architecture is still based on a centralized client-server model, where a user submits his location to a server and obtains the customized data he needs. This conventional model quite limits the application scope and may hinder wide deployment of location related mobile applications because of the following disadvantages. First, it requires Internet connection even when a sender and receiver are adjacently located, which will unnecessarily increase the Internet traffic burden and users' bandwidth cost. With this requirement, in addition, applications are not robust against catastrophic infrastructure failures. Furthermore, a data consumer has to have prior knowledge of the data providers, e.g., the URL of the server. There is no general channel for users to browse all available service resources nearby without registering for each and every one of them.

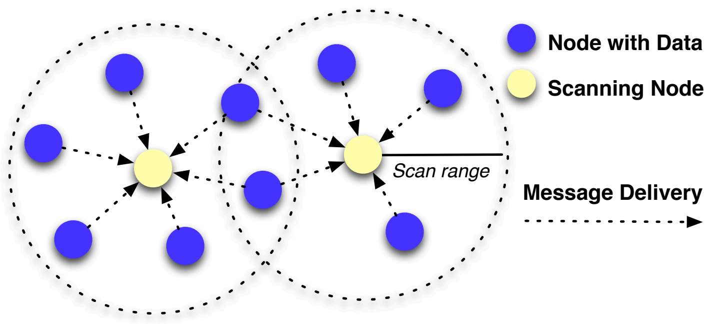


Figure 4.1: Passive Broadcast Model

We argue that Ad-Hoc network model is a complementary alternative that can effectively help solve all the above issues. In practice, however, creating and maintaining a direct link between two nearby devices which is the building block of an ad-hoc network is costly. For example, both Bluetooth and WiFi-Direct require a slow initial phase of discovering nearby peers and handshaking to establish a connection. It is especially inefficient for transferring a small amount of data. In addition, connected link may bring with it a security risk because one device might be able to access all exposed services on the connected device (e.g., in Bluetooth). In this chapter, we build a local data dissemination system upon a new communication model, called *passive broadcast*. It is a connectionless and receiver-initilized model where each node periodically *scans* other nodes in the communication range and obtains their data if available (see Fig. 4.1, i.e., each scan is a many-to-one communication. The representative carriers of *passive broadcast* in reality include Bluetooth and WiFi-Direct, both of which define a mandatary 'peer discovery' function to fetch basic information about nearby devices. This function can be easily extended to implement passive broadcast mechanism without modifying the existing network protocol stack. In passive broadcast, the cost for establishing and maintaining direct links is negligible and our experiments show that the communication range is expanded compare to the established connection over the same radio interface. In addition, the feature of fetching messages from multiple nearby devices is desirable for applications that spread messages in the proximity.

4.1 Related Work

This work is related to prior research on mobile social networks, overlay P2P dissemination, and delay tolerant networks. Information dissemination has become more and more important in mobile social networks, such as [49–51], which aim at data dissemination in resource-constrained opportunistic networks, broadcasting from superusers and ferrying messages in intermittently connected mobile networks. Point&Connect [52] implements pointing gestures of moving one device towards another in order to enable spontaneous device pairing. Musubi [53] provides a decentralized trusted social services on personal mobile devices. And BubbleRap [54] utilizes group membership information to improve standard unicast routing. In [55], the authors propose a gossiping-based approach, where each node forwards a message with some probability, to reduce the overhead of the routing protocols. 7DS [56] is developed to address network disruption problem in mobile networks by providing store-carry-forward communication. However, it only concerns data as store-carry-forward manner for disruption-tolerant applications which is very limited. [57] proposes to use data diffusion to reduce the query delay in DTNs. They use theoretical models to analyze the data diffusion process and compare the performance of the their proposed diffusion schemes in terms of diffusion speed and query delay. In addition, there is other prior work that helps better understand characteristic of DTNs such as [58–61]. For example, [58] is the rst to study multicast in DTNs from the social network perspective. They study multicast in DTNs with single and multiple data items, investigate the essential difference between multicast and unicast in DTNs, and formulate relay selections for multicast [59] studies human contact-based traces and designs SNAMD scheme that makes use of such information to efficiently deliver multicast messages. The authors in [60] compared the asymptotic performance of Interest-Based forwarding and from both the theoretical and experimental point of view. And SimBet [61] first introduces social network analysis in the context of delay tolerant networks. It uses ego-centric centrality and its social similarity. However, the focus of our problem is based on a different communication model and our objective is different. Passive broadcast is a receiver-initialized connectionless communication based on many-to-one delivery in each scan. Moreover, we target on a distributed and collaborative solution that efficiently disseminate messages in proximity.

One project closely related to our passive broadcast model is called Dythr [62] which lets a phone broadcast a WiFi hotspot with the SSID being the message. This method actually is from the opposite direction of 'active' delivery as every node frequently injects messages into the wireless channel. In [63], Huang et al. proposed PhoneNet. This method uses a central server to establish links between devices connected to a WiFi network and then allows devices connected on local networks to connect directly. In [64], the authors use Bluetooth service discovery protocol to find common interests between two users. Similar to our proposed work, no connection is established and each user stores the keyword about his interest in Bluetooth service ID. However, this work is for two-user communication while our problem is set in a multiple user environment and our goal is to determine each device's schedules to achieve the efficiency.

4.2 System Model

Our target problem in this chapter is to enable nearby smartphone users to share information. In this section, we introduce basic components and sketch of our solution.

4.2.1 Communication Model

Our solution is based on the new *passive broadcast* communication model, where each sender buffers its broadcast data locally and the receiver initializes the transmission and fetches all available data from nearby nodes. In this model, when having a message to deliver, a node puts it in a buffer, but does not know when the message will be sent. On the other hand, each node periodically *scans* other nodes in the communication range and obtain the data in their buffers if available, i.e., each scan is a many-to-one communication.

We have implemented this model based on the mandatary 'peer discovery' function in both Bluetooth and WiFi-Direct. In the rest of this chapter, we take Bluetooth as a platform instance to introduce our solution. Basically, we use the field of 'device name' to carry target payload data. When a user intents to send a message, he assigns the message to his phone's Bluetooth device name. When other phones conduct peer discovery, the message will be sent over. The length of device names is usually limited, e.g., a Bluetooth's device name in Android can be up to 248 bytes. A large message can be fragmented to fit in and a phone can periodically change the device name to rotate multiple messages or fragments.

4.2.2 Smartphone Operations and States

Assume there are n smartphone nodes ($\{p_1, p_2, \ldots, p_n\}$) and this chapter considers a network model where all the phones are within each other's Bluetooth communication range. There are two basic operations for each smartphone p_i, *scan* and *update message*. The first operation is the regular peer discovery process while the second one is to change its own device name to a new message. *Update message* operation can be finished instantly. But *scan* process has a long overhead. For Bluetooth, according to the standard and our experiments, a scan operation usually takes $10 \sim 12$ seconds to finish. Therefore, we further define two states for each phone: when a phone is conducting *scan* (peer discovery), it is in **scanning phase**; otherwise, it is in **idle phase**. For each message, we define its *active period* as the duration when the message is available for scanning, i.e., after the message is put on the device name and before it is replaced by the next message. If a phone finishes a complete scan during a message's active period, the message will be surely scanned by the phone. If the active period starts or ends during a scanning process, the phone has a certain probability to receive the message. We will present detailed analysis later in Section 4.4.

We assume that the scan schedule has been pre-configured by each phone based on its own performance concerns, e.g., power consumption and urgency of getting new messages. We use T_i to represent p_i's *scan interval* which is defined as the interval between the end of the prior scan and the beginning of the next scan, i.e., the length of p_i's idle phase. Additionally, we use U_i to indicate the message *update interval* of p_i, i.e., p_i changes its device name once every U_i time units. Different from T_i, U_i's value can be dynamically changed as it does not incur any extra computation overhead or power consumption. Furthermore, we define S as the length of scanning phase, which is a constant for all phones.

In our solution, each phone and each active message has a unique identifier defined as follows:

- Phone ID: In this chapter, we use the MAC address as each phone's identity. When scanning nearby devices, a phone automatically obtains their MAC addresses and is able to recognize these phones later.

- Message ID: Each message can be identified by its owner's MAC address and a local index number. For example, aa:bb:cc:dd:ee:ff.12 represents a message from the phone with a MAC address of aa:bb:cc:dd:ee:ff and its index number on that phone is 12. In our solution, each index number is incremental with new messages and set up to 255 after which it will be reset to 0.

4.2.3 Message Format

In our solution, the device name is divided into two segments, header and payload. Similar to other network protocol, 'header' field contains control and management data and 'payload' stores the messages being broadcast. The header includes the following fields:

- Scan interval T: Each phone p_i uses one byte to represent its own scan interval T_i in the unit of second.
- Index range of active messages: Each phone uses two bytes to specify the index number range of its active messages. We assume each user apply the policy that only the most recent messages are considered for dissemination, e.g., the most recent 10 messages, and the messages generated in the past 5 hours.
- Message reception feedback: Each phone includes a *message reception feedback* to indicate the current state of the reception. Our solution uses a simple bitmap to represent this feedback information. Namely, every message is flagged by one bit in the feedback, '1' indicates 'received' and '0' means 'absent'. We assume that a function f can convert a MAC address to a numeric value, thus all phones can be ordered according to their MAC addresses. In the bitmap feedback, the messages from different phones are ordered by their owners' MAC addresses.

The following Table 4.1 lists some notations we use in the rest of the chapter.

n/p_i	number of smartphones/the i-th phone
T_i/U_i	scan interval / update interval of p_i
S	execution time of one scan
t_i	execution time for p_i to receive all messages
f	a function that converts a MAC address to a numerical value
k_i	number of messages generated by p_i
M	set of all message IDs in the network

Table 4.1: Notations

4.3 Message Dissemination with Passive Broadcast

In this section, we present our solution PASA that disseminates local messages based on passive broadcast model. We further provide numerical analysis to derive the optimal parameters for our solution.

4.3.1 Problem Formulation

Recall that we consider all the phones $\{p_1, p_2, \ldots, p_n\}$ are within each other's communication range. Each phone p_i holds k_i messages to propagate to other phones. Assume each smartphone is aware of other phones' scan intervals after an initial scan. Without loss of generality, we sort all smartphones in the ascending order of their scan intervals, i.e., $\forall i, j \in [1, n]$, if $i < j$, then $T_i < T_j$. Let t_i represent the time phone p_i spends in receiving all the messages. Our objective in this chapter is to minimize $\sum_i t_i$.

The essential goal of our algorithm is to determine the update interval for each phone so that all phones can collaborate to disseminate all messages quickly. Out solution is based on an important property defined in the following Theorem 1.

Theorem 1. *If a phone set its update interval to be $T_n + 2 \cdot S$, its message can surely be scanned by all other phones.*

Proof. For any phone p_i ($T_i \leq T_n$), apparently there must be a complete scan during a period of $T_n + 2 \cdot S$. Thus, the message will certainly be scanned by every phone. □

4.3.2 Design of PASA

We present a solution with two stages. In the first stage, every phone simply sets U_i to be a constant α and rotates its own k_i messages. In the second stage, each phone set $U_i = T_n + 2 \cdot S$ and iteratively update the device name by one of the messages generated by its own or received from the first stage. The strategy of updating messages, however, varies depending on each phone's status during the second stage. We classify all phones into two categories and each category executes a different algorithm. The first category is called *complete phones* which includes all the smartphones that have received all $\sum_{i\in[1,n]} k_i$ messages. The other category, *incomplete phones*, is the complement of the first stage and includes the phones that have not received all messages. As the update interval is fixed as $T_n+2\cdot S$, our design focus is message updating algorithm in the second stage. In the rest of this section, we develop two algorithms, one for incomplete phones and the other for complete phones.

Our basic intuition is to let all the smartphones assign the *most wanted* messages to the device names so that they can help other phones to speed up their second stages. In addition, we intent to avoid duplicate messages in the second stage. It is apparently inefficient if multiple phones put the same message on their devices' names. In the stretch of our solution, we give incomplete phones higher priority to select messages as the messages that incomplete phones can contribute in the second stage are limited. Complete phones, on the other hand, will estimate incomplete phones' choices and select other desired messages to serve in the second stage.

We use M to represent the set of all message IDs in a given network, i.e., in our setting, M includes $\sum_i k_i$ items. Assume all message IDs in M are sorted according to a pre-defined numerical conversion and let m_i represent the i-th item in M. Each phone is aware of M after the initial scan. In addition, each phone maintains a two dimensional matrix MR to indicate the message reception status in the network, where $MR_{ij} = 1$ if phone p_i has received message m_i (otherwise $MR_{ij} = 0$). For each phone, this matrix MR is built upon the feedbacks from other phones. With the assistance of MR, each phone can derive sufficient information for the next stage. First, each phone can identify all the complete phones and incomplete phones, i.e., p_i is a complete phone if $\sum_j MR_{ij} = |M|$. Let IP and CP respectively represent the set of incomplete phones and complete phones each phone constructs. Second, each phone knows which messages it has received are needed by other phones. We let each phone p_i build a set of *candidate messages*, indicated by C_i, including all the messages that are useful for some other phones and could be put on the device name in the next stage. By definition, each message $m_j \in C_i$ must satisfy $MR_{ij} = 1$ and $\sum_h MR_{hj} < n$.

In addition, we assume each phone has a different *priority* value based on the phone ID. Our algorithm will use this priority value to avoid unnecessary duplicate message selections from multiple phones. If a message is chosen by multiple phones, only the one with the highest priority will put it on the device name and the others have to yield and select other messages. Specifically, we assume a pre-defined the function f could convert a phone ID to a numerical value for comparison, i.e., p_i has a higher priority than p_j if $f(p_i) > f(p_j)$.

Message update for incomplete phones: In our solution, an incomplete phone applies the following Algorithm 4 in the second stage.

Algorithm 4 Choose the Message for Incomplete Phones

```
1: Identify all incomplete phones (construct $IP$)
2: Sort the set $IP$ in the descending order of each phone's priority
3: for each incomplete phone $p_i$ do
4:     Construct its candidate set $C_i$
5:     Sort $C_i$ in the ascending order of the number of phones having received the
       messages, i.e., for each message $m_j$, the value of $\sum_h MR_{hj}$.
6: for $i = 1$ to $|IP|$ do
7:     Let $p_j$ be the $i$-th phone in $IP$, choose the first message of $C_j : msg_j = C_j[1]$
8:     for each incomplete phone $p_h \in IP$ do
9:         Remove $C_j[1]$ from $C_h$
```

In Algorithm 4, each incomplete phone first identifies all incomplete phones (the set IP) and sorts them based on their priority values. Additionally, each phone forms the candidate sets of all incomplete phones using the information from MR (Lines 3–6). The candidate messages are also sorted by the number of phones that have received them (Line 5). In our algorithm, therefore, the first message in each candidate set is the *most wanted* messages. The main message selection process is in Lines 7–12. We use msg_i to record the choice of p_i. The loop starts from the phone with the highest priority to the one with the lowest priority. Within the loop, each phone picks the first message in the candidate set for updating message in the second stage. Upon a message msg_j is selected, all candidate message sets are updated by removing msg_j to avoid redundant message selections. Eventually, for a phone p_i, msg_i will be the message to be put on the device name. The time complexity of Algorithm 4 is bounded by $O(n^2)$.

Message update for complete phones: The algorithm for a complete phone to update message is similar to Algorithm 4. Due to the page limit, we omit the pseudo-codes here. Each phone needs to construct the set of all complete phones CP and derive the candidate message set for each of them. An extra step for a complete phone is that it has to execute Algorithm 4 before making its own choice. All the messages that have been selected by incomplete phones will be eliminated from complete phones' candidate sets. The remaining messages in the candidate set are also sorted according to the number of phones that need them. The same as in Algorithm 4, each phone selects the header message in the sorted list of candidate messages.

4.3.3 Parameter Optimization

In this subsection, we analyze the performance of the solution presented above and derive the optimal value of α. Essentially, we aim to express the objective as a function on α.

Our analysis is based on a general function $\mathcal{P}(x, y)$ defined to represent the probability that a phone with scan interval of x can receive a message from another phone with update interval of y. In another word, $\mathcal{P}(T_i, U_j)$ is the reception probability for p_i to receive a message from p_j. We will present how to calculate $\mathcal{P}(x, y)$ in Section 4.4.

We use the following Theorem to estimate the execution time for each phone to receive all messages, i.e., the value of t_i.

Theorem 2. *The execution time for phone p_i to receive all messages is expected to be*

$$t_i = kmax \cdot \alpha + r \cdot (T_n + 2 \cdot S),$$

where $kmax$ is larges value of k_i, and let $c_1 = \sum_{k_i<kmax} \frac{(kmax-k_i)\cdot\alpha}{T_n+2\cdot S}$, $c_2 = (1 - \prod_i \mathcal{P}(T_i,\alpha)) \cdot (1 - \frac{c_1}{\sum_i k_i}) \cdot \sum_i k_i$, r is the minimal value that satisfies the following condition,

$$\sum_{j\in[0,r-1]} \frac{(n-1)^2}{c_2 - (n-1)\cdot j} = (1 - \mathcal{P}(T_i,\alpha)) \cdot \sum_{j\neq i} k_j - \frac{c_1}{n}$$

Proof. Let us consider the time point when the last phone finishes its first stage, i.e., the phone with the most messages. Let $kmax = \max\{k_i\}$ Each phone p_i has missed $(1 - \mathcal{P}(T_i,\alpha)) \cdot \sum_{i\neq j} k_j$ messages from their owners. However, some phones have broadcast messages in their second stage which guarantees a success delivery at all other nodes. In total, there have been $c_1 = \sum_{k_i<kmax} \frac{(kmax-k_i)\cdot\alpha}{T_n+2\cdot S}$ messages in the second stage. We assume they evenly contribute to each phone's collecting process, i.e., each phone obtain at least $\frac{c_2}{n}$ missing messages.

For each message m, the probability that it has not been scanned by all the phones when the last phone enters the second stage is

$$(1 - \prod_i \mathcal{P}(T_i,\alpha)) \cdot (1 - \frac{c_1}{\sum_i k_i}).$$

Therefore, there are $c_2 = (1 - \prod_i \mathcal{P}(T_i,\alpha)) \cdot (1 - \frac{c_1}{\sum_i k_i}) \cdot \sum_i k_i$ total candidate messages that can be put on device names in the second stage. Assume each phone picks a distinct message in the first round of second stage. For a particular message that p_i needs, there is a probability of $\frac{n-1}{c_2}$ to be scanned after the first round. In the second round, the expected number of total candidate messages becomes $c_2 - n$ and each message p_i needs has a probability of $\frac{n-1}{c_2-(n-1)}$ to be collected. This process is repeated until p_i obtains all the messages. Therefore, p_i is expected to use r rounds in the second stage to collect all messages, such that

$$\begin{aligned}&\sum_{j\in[0,r-1]} \frac{n-1}{c_2-(n-1)\cdot j}\cdot(n-1)\\ =\ &(1-\mathcal{P}(T_i,\alpha))\cdot\sum_{j\neq i} k_j - \frac{c_1}{n}\end{aligned}$$

Eventually, the execution time for phone p_i is $t_i = kmax\cdot\alpha + r\cdot(T_n + 2\cdot S)$. □

Since the value of α is upper-bounded by $T_n + 2\cdot S$, we can enumerate all possible values and derive the best value leading to the minimum $\sum_i t_i$ according to the above Theorem 2.

4.4 Analysis of Message Reception Probability

In this section, we analysis the message reception probability $\mathcal{P}(x, y)$ which is a critical building block for deriving the optimal parameters in our solution.

Based on Theorem 1, $\mathcal{P}(x, y) = 1$ if $y \geq x + 2\cdot S$. The following analysis is under the condition of $y < x + 2\cdot S$. The message reception probability depends on ont only the two input parameters x and y which indicate the length of scan interval and update interval (the message's active period), but also the schedule of scanning and update processes, i.e, the start points of the phone's (p_i) scanning phase and the message's (m) active period. Therefore, we need to consider several basic possible cases as follows.

- **Case A**: The active period of m is within p_i's scanning phase. The probability that p_i can receive m is $\frac{y}{S}$.

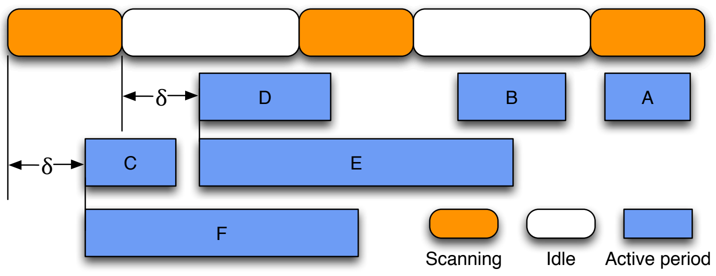


Figure 4.2: Cases A∼F

- **Case B**: The active period of m is within p_i's idle phase. The probability that p_i can receive m is 0.

- **Case C**: The active period of m starts during p_i's scanning phase and ends in the consecutive idle phase. In this case, the reception probability depends on the different between the beginning of the scanning phase and that of m's active period. Let $\delta \in [0, S]$ indicate that offset. Thus, the probability that p_i can receive m is $1 - \frac{\delta}{S}$.

- **Case D**: The active period of m starts during p_i's idle phase and ends in the consecutive scanning phase. This case is similar to Case C and the reception probability depends on overlap of the scanning phase and m's active period. Let $\delta \in [0, S]$ be the difference between the start of the scanning phase and the end of the active period. The probability that p_i can receive m is $\frac{\delta}{S}$.

- **Case E**: The active period of m starts in p_i's idle phase and ends in p_i's next idle phase. In this case the reception probability is 1.

- **Case F**: The active period of m starts in p_i's scanning phase and ends in p_i's next scanning phase. In this case, p_i could possibly receive m in both scanning phases. Let $\delta \in [0, S]$ represent the different between the start of the first scanning phase and that of m's active period. Then the probability that p_i receives m in the first scanning phase is $1 - \frac{\delta}{S}$. In the second scanning phase, the overlap with the active period, i.e., between the start of the second scanning phase and the end of m's active period, is $y - x - (S - \delta)$. Thus, the probability that p_i receives m in the second scanning phase is $\frac{y-x-(S-\delta)}{S} = \frac{y-x+\delta}{S} - 1$. Combing these two scanning phases, therefore, the probability that p_i can receive m (in at least one of the scanning phases) is

$$\begin{aligned} & 1 - (1 - (1 - \frac{\delta}{S})) \cdot (1 - (\frac{y-x+\delta}{S} - 1)) \\ = \;& 1 - \frac{\delta}{S} \cdot (2 - \frac{y-x+\delta}{S}) \end{aligned}$$

- **Case G**: The active period of m starts in p_i's idle phase and ends in the second consecutive scanning phase. The reception probability is obviously 1.
- **Case H**: The active period of m starts in p_i's scanning phase and ends in the second consecutive idle phase. The reception probability is also 1.

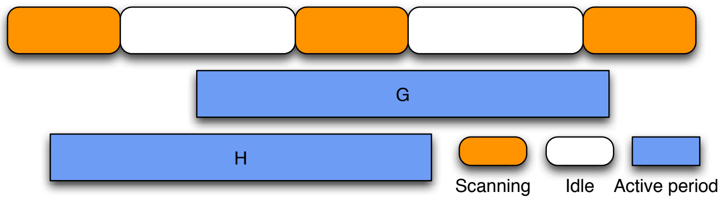


Figure 4.3: Case G and Case H

Above all, we derive the value of $\mathcal{P}(x, y)$ under thre different conditions in the following Theorem 3, Theore 4, and Theorem 5.

Theorem 3. *When* $y \in [x + S, x + 2 \cdot S)$

$$\mathcal{P}(x, y) = 1 - \frac{1}{x+S} \sum_{\delta \in [0, x+2\cdot S-y)} \frac{\delta}{S} \cdot (2 - \frac{y-x+\delta}{S}))$$

Proof. We consider a scanning phase followed by a idle phase as a repeating round for phone p_i and the message active period could start at any point in this round. When $y \in [x+S, x+2\cdot S)$, as shown in Fig. 4.4, we can classify all possible scenarios

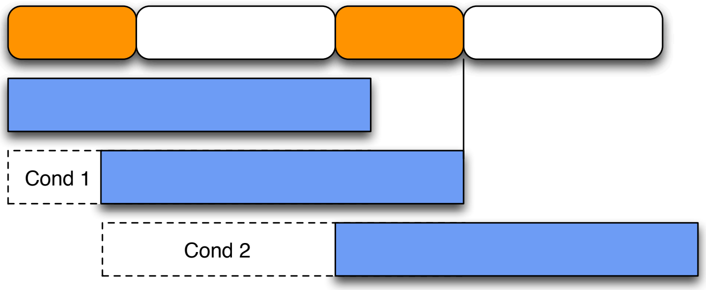


Figure 4.4: $y \in [x+S, x+2\cdot S)$

into two conditions based on the starting point of m's active period. In Condition 1, the active period starts in p_i's scanning phase with a offset from the beginning of the scanning phase to a point where the active period ends at the end of the next scanning phase. Condition 2 is the complementary of Condition 1. Apparently, the reception probability under Condition 2 is 1 because the active period covers a complete scanning phase (Case H). On the other hand, Condition 1 refers to Case F above and the difference from the start of the scanning phase ranges from 0 to $u = x+2\cdot S-y$. Therefore, the reception probability is

$$\begin{aligned}\mathcal{P}(x,y) &= (1-\frac{u}{x+S})\\ &\quad +\frac{1}{x+S}\sum_{\delta\in[0,u)}(1-\frac{\delta}{S}\cdot(2-\frac{y-x+\delta}{S})))\\ &= 1-\frac{1}{x+S}\sum_{\delta\in[0,u)}\frac{\delta}{S}\cdot(2-\frac{y-x+\delta}{S}))\end{aligned}$$

□

Theorem 4. *When* $y \in [x, x+S)$

$$\mathcal{P}(x,y) = \frac{1}{x+S}(\sum_{\delta\in[0,v)} \frac{min\{S-\delta, y\}}{S}$$
$$+ \sum_{\delta\in[v,S)} (1 - \frac{\delta}{S}(2 - \frac{\delta+y-x}{S})) + x - \frac{v^2+1}{2\cdot S}.$$

Proof. Let SP_1 and SP_2 be the closest scanning phase before and after the message active period starts, i.e., SP_1 belongs to the round, but SP_2 does not. Under the condition of $y < x+S$, the message active period could overlap with either SP_1 or SP_2 or both of them. Let δ be the offset of the message active period compared to the start of a round. We derive $\mathcal{P}(x,y)$ based on the analysis of the following three cases:

(1) The message active period only overlaps with SP_1, but not SP_2. This case refers to Case A and Case C and δ ranges from 0 to $S+x-y$, the message reception probability is $\frac{min\{S-\delta,y\}}{S}$.

(2) The message active period only overlaps with SP_2, but not SP_1. This case refers to Case D, Case E and Case G. The message reception probability is $\frac{\delta+y-S-x}{S}$ when $\delta \in [S, x+2\cdot S-y]$ (Case D). The message reception probability is 1 when $\delta \in [x+2\cdot S-y, x+S]$. Therefore, let $v = x+S-y$, the message reception probability is

$$\sum_{\delta\in[S,v+S)} \frac{\delta-v}{S} + \sum_{\delta\in[v+S,x+S)} 1$$
$$= \frac{(v+2\cdot S-1)\cdot v}{2\cdot S} - \frac{v^2}{S} + (x-v) = x - \frac{v^2+1}{2\cdot S}$$

(3) The message active period overlaps with both SP_1 and SP_2. In this case, δ ranges from $x+S-y$ to S. The lengths of the overlaps with SP_1 and SP_2 are $S-\delta$ and

$\delta + y - x - S$ respectively. Referring Case F, the message reception probability is,

$$\begin{aligned} & \sum_{\delta}(1-(1-\frac{S-\delta}{S})(1-\frac{\delta+y-x-S}{S})) \\ = & \sum_{\delta}(1-\frac{\delta}{S}(2-\frac{\delta+y-x}{S})) \end{aligned}$$

Therefore, when $y \in [x, x+S)$,

$$\begin{aligned} \mathcal{P}(x,y) = & \frac{1}{x+S}(\sum_{\delta\in[0,v)}\frac{min\{S-\delta,y\}}{S} \\ & +x-\frac{v^2+1}{2\cdot S}+\sum_{\delta\in[v,S)}(1-\frac{\delta}{S}(2-\frac{\delta+y-x}{S})) \end{aligned}$$

□

Theorem 5. *When* $y \in (0, x)$

$$\begin{aligned} \mathcal{P}(x,y) &= \frac{1}{x+S}(\sum_{\delta\in[0,v)}\frac{min\{S-\delta,y\}}{S} \\ &+\sum_{\delta\in[v,S)}(1-\frac{\delta}{S}(2-\frac{\delta+y-x}{S})). \end{aligned}$$

Proof. The proof is similar to the above proof of Theorem 4. Under the condition of $y < x$, the message active period could overlap with either SP_1 or SP_2. Let δ be the offset of the message active period compared to the start of a round. We derive $\mathcal{P}(x, y)$ based on the analysis of the following two cases:

1. The message active period only overlaps with SP_1. This case refers to Case A and Case C and δ ranges from 0 to $S + x - y$, the message reception probability is $\frac{min\{S-\delta,y\}}{S}$.

2. The message active period only overlaps with SP_2. This case refers to Case D and Case E. The message reception probability is $\frac{\delta+y-S-x}{S}$ when $\delta \in [S, x+2\cdot S-y]$ (Case D).

Therefore, following the above proof of 4, when $y < x$,

$$\begin{aligned}\mathcal{P}(x,y) &= \frac{1}{x+S}\Big(\sum_{\delta\in[0,v)}\frac{min\{S-\delta,y\}}{S}\\ &\quad+\sum_{\delta\in[v,S)}(1-\frac{\delta}{S}(2-\frac{\delta+y-x}{S}))\end{aligned}$$

□

4.5 Implementation and Evaluation

In this section, we will first introduce the system implementation of our PASA system and then present the performance evaluation results from both smartphones experiments and simulation.

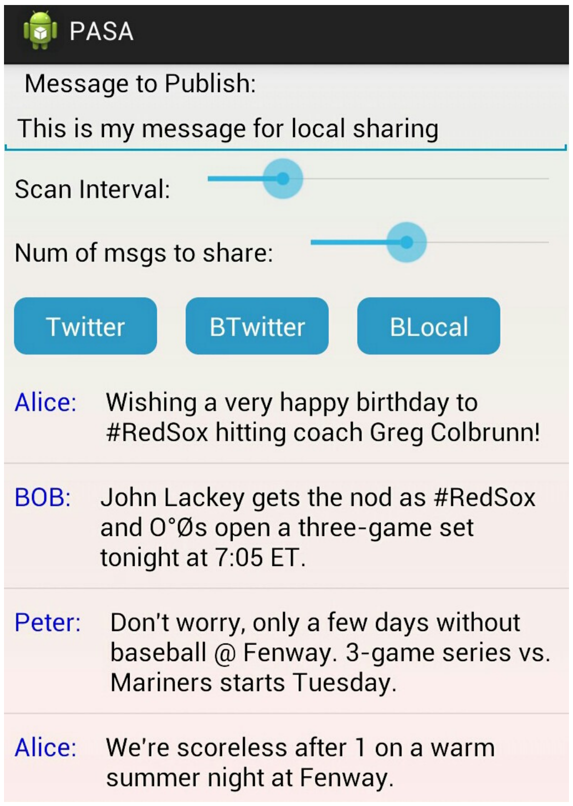
4.5.1 System Implementation

We choose Android operating system as our development environment, and have implemented PASA on different brands of smartphones, including Asus(Nexus 7), LG(Nexus 4) and Samsung(Galaxy Nexus). Our prototype system is built on Android Jelly Bean(specific version 4.2.2), API level 17. PASA system does not require any change on Bluetooth driver or Android kernel.

In our experiments, we integrate PASA with Twitter to demonstrate application functions. A user can choose to pull online tweets for local dissemination or push local messages to his Twitter account. As a result, when a user starts PASA for the first time, he needs to authorize the application to use his Twitter account and configure his own profile(e.g. name, number of messages to share and length of idle duration).

Fig.4.5 and Fig.4.6 show the configure setting and demo of our PASA system. The system mainly performs the following operations:

- Input and publish message to Twitter, nearby users or both (Twitter, BLocal and BTwitter on Fig.4.5 correspondingly).
- Set or change configurations.
- Scan the nearby users to receive messages.
- Generate and transmit feedback to inform other users which messages it currently holds.
- View one specific local user's shared message.



This is my message for local sharing

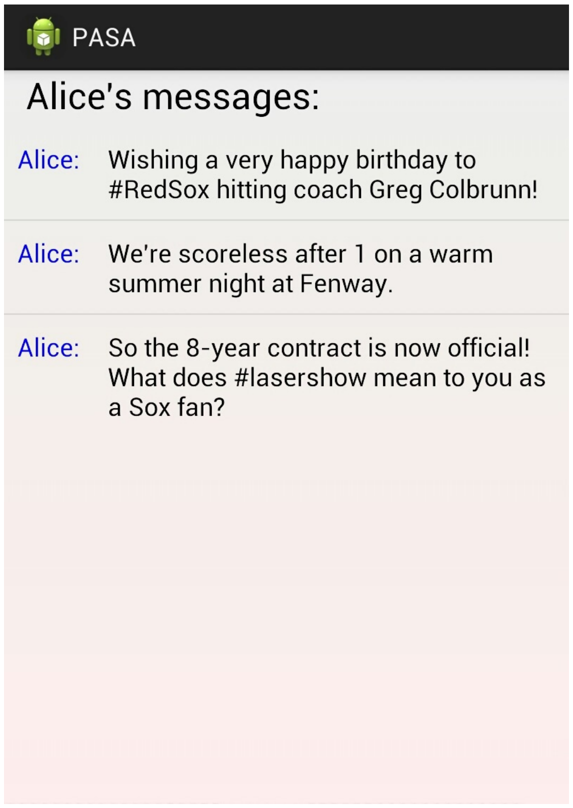
Alice: Wishing a very happy birthday to #RedSox hitting coach Greg Colbrunn!

BOB: John Lackey gets the nod as #RedSox and O°Øs open a three-game set tonight at 7:05 ET.

Peter: Don't worry, only a few days without baseball @ Fenway. 3-game series vs. Mariners starts Tuesday.

Alice: We're scoreless after 1 on a warm summer night at Fenway.

Figure 4.5: PASA: Main page



Alice: Wishing a very happy birthday to #RedSox hitting coach Greg Colbrunn!

Alice: We're scoreless after 1 on a warm summer night at Fenway.

Alice: So the 8-year contract is now official! What does #lasershow mean to you as a Sox fan?

Figure 4.6: PASA: Message list page

4.5.2 Performance Evaluation

In this subsection, we present the performance evaluation on both Android smartphones(small scale) and simulation(large scale). The key of PASA is to, in different configure setups, quickly disseminate and gather local users' messages. Thus, the most important metric is the average complete time among all smartphones in the network.

4.5.2.1 Experimental Results

We set up a small testbed with 5 Android phones running our program. We consider the following four different scenarios for our evaluation.

- Scenario 1: All smartphones have the same scan interval(T_i) and number of available messages(k_i);
- Scenario 2: They have different T_i but the same k_i;
- Scenario 3: They have different k_i but the same T_i;
- Scenario 4: Both T_i and k are different for them.

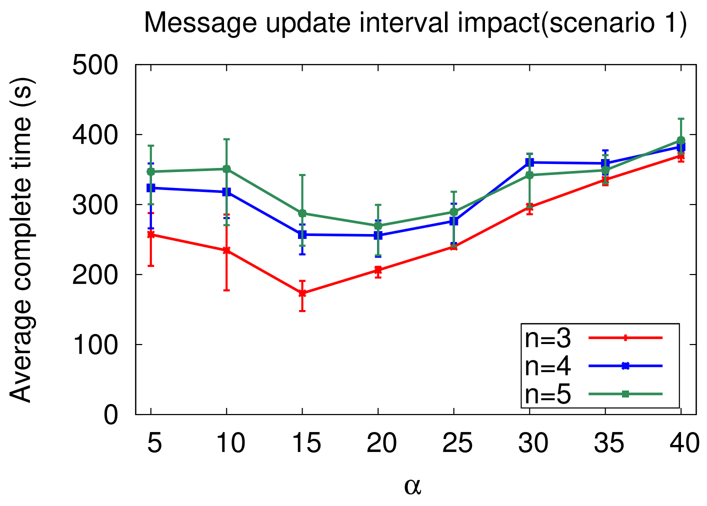


Figure 4.7: Average complete time with different α, ($T_i = 5, k_i = 10$)

Fig. 4.7 shows a result of Scenario 1. In this setting, each smartphone has 10 messages($k_i = 10$) to share and their scan intervals is 5s($T_i = 5$). We present the average complete time with an error bar showing the maximum and minimum complete time. We set α to a set of discrete values ranging from 5s to 40s with a interval of 5s. As we can see from the figure, with 3 phones, the best value of α is 15; with 4 or 5 phones the best results come with $\alpha = 20$. Intuitively, a smaller α causes low message reception rate in the first stage leading to more messages disseminated in the second stage; on the other hand, a larger α increase the execution time of the first stage. Therefore, there are pivot points in all three curves. Our analytical model derives the optimal values of α to be 18, 19, 19 respectively for $n = 3, 4$ and 5. This estimation matches well with the experimental results which shows our analysis is accurate and effective, thus can help users find the best value of α for the PASA system. Another observation from Fig. 4.7 is that when α is small, the average complete time of each phone varies a lot. because the expected message reception probability is low which indicates that it is likely to miss such a message during the first stage. For example, when $\alpha = 10$, the difference between the complete time of the first complete phone and that of the last one is 108s, 66s and 122s for $n = 3, 4, 5$, respectively. But when $\alpha = 40$, these three differences become 14s, 18s, 50s. The main reason is that when α is larger, more messages are delivered during the first stage which aligns each phone's progress.

Fig. 4.8 shows a result for Scenario 2, where $k_i = 10$ and T_i is set to 10,15,20,25,30 respectively. We find from the figure that when α is small, the average complete time is varying rapidly. For example, with 5 phones ($n = 5$), the average complete time is 544s when $\alpha = 10$. It increases to 569s when $\alpha = 15$, and drops to 502s when $\alpha = 20$. Based on the experiments, the best tested values of α are 25,30,30 for $n = 3, 4, 5$. Our theoretical model derives 26, 26, and 30 as the optimal values which are very close to the experimental results.

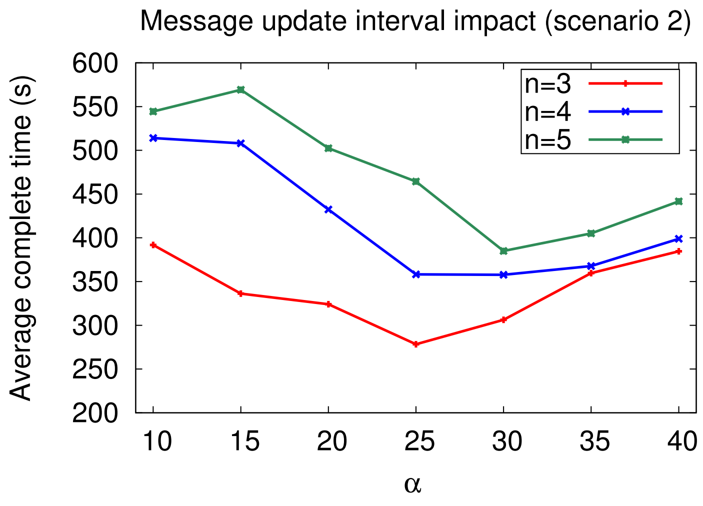


Figure 4.8: Average complete time with different α, $k = 10$ for all smartphones and $T_i = 10, 15, 20, 25, 30$ respectively.

Fig. 4.9 and Fig. 4.10 illustrate the results for Scenario 3 and Scenario 4 in our evaluation. In Fig. 4.9, $T_i = 10$ for all phones and k_i is set to 10, 15, 20, 25 for each phone respectively. In the tested cases, the best value of α with minimum average complete time is 20 for both $n = 3$ and $n = 4$. Our analysis again derives very close values of 23 and 24 for $n = 3$ and $n = 4$. In Fig 4.10, we show the results of different T_i and k_i. Given a particular value of α, the average complete time in the experiment with $n = 4$ is much larger than $n = 3$. Comparing with $n = 3$, the network with $n = 4$ has $T_4 = 25$ which results in large difference of average complete time. Phone p_4 with $T_4 = 25$ is more likely to miss messages during the first stage and thus needs more time to collect messages during other phones' second stage. Our theoretical optimal values of α are 25 and 45 for $n = 3$ and $n = 4$ which match the results in Fig. 4.10 very well.

4.5.2.2 Simulation Results

To evaluate a large scale network, we conduct simulation to test the performance of PASA. Our goal is to study the impact of the number of phones on the system

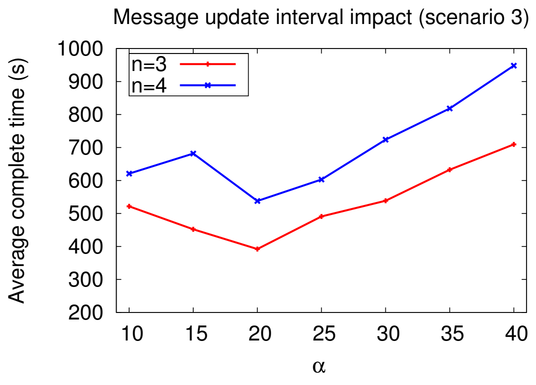


Figure 4.9: Average complete time with different α, while $T_i = 10$ for all smartphones and $k_i = 10, 15, 20, 25$ respectively.

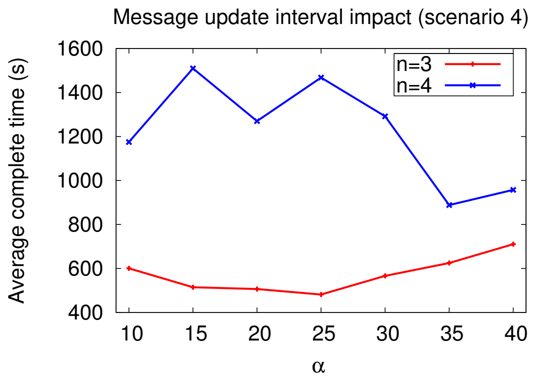


Figure 4.10: Average complete time with different α, while $T_i = 10, 15, 20, 25$, $k_i = 25, 20, 15, 10$ respectively.

performance. Fig.4.11 shows the average complete time with $k_i = 10$ and $T_i = Random[5, 35]$ over different number of smartphones in the network. The number of phones ranges from 5 to 50, and the average complete time fluctuate within 500 to 600 which is very small comparing with the number of total messages increase. This is caused by the many-to-one communication during each scan. Increasing the number of phones does not increase the complete time much which indicates passive broadcast is certainly a scalable model.

Fig. 4.12 shows the average complete time for 10, 15, 20, 50, 80 smartphones. We find that for $\alpha \geq 30$, the five curves are almost coincide with each other. This is because the update message interval is large enough for everyone to receive all messages during the first stage. From our theoretical model, the optimal value for these settings are 24, 26, 22, 23, 23 for $n = 10, 15, 20, 50, 80$. Comparing with the Fig. 4.12, our model can find a good value of α for PASA system in these settings.

Fig. 4.13 shows the result when $k = 10$ for everyone and each phone randomly choose a T_i in $[5, 35]$. In our tested cases, for $n = 10, 15, 20, 50$ the maximum T_i is 29, 28, 21, 35 respectively. The theoretical optimal value of those four settings are 20,

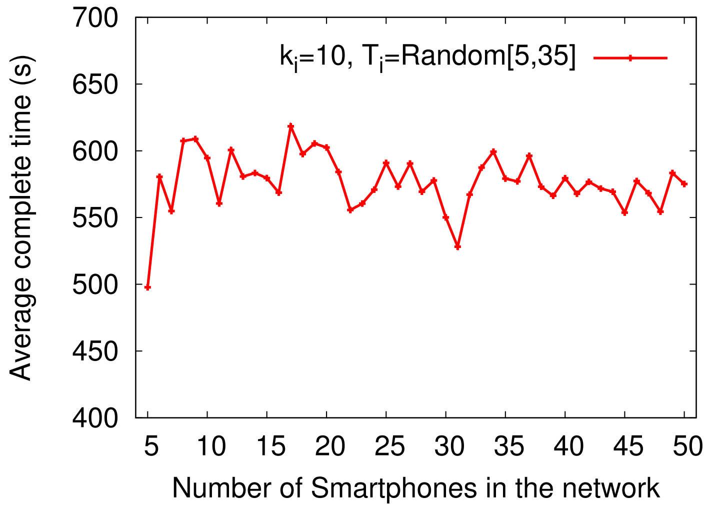


Figure 4.11: Average complete time with different numbers of phones (n), while $k_i = 10$ for all smartphones and $T_i = Random[5, 35]$

26, 25, 33. As we can see from the figure, our analysis provides a very good guideline for choosing a value for α.

Finally, Fig. 4.14 shows a general scenario where a user's k_i and T_i are randomly chosen from a given range, 5 to 30 in our setting. The following table 4.2 shows the specific setting of our test. With the details of each setting, our theoretical model gives the optimal value of α as 29, 25, 26, 34.

	$n = 10$	$n = 15$	$n = 20$	$n = 50$
$\max(k_i)$	30	30	27	30
$\max(T_i)$	29	30	28	30

Table 4.2: Scenario 4 parameter settings

4.6 Summary

This chapter presents PASA, a smartphone Ad-Hoc network based communication model for a local message dissemination system. We present a two-stage protocol for propagate messages and provide in-depth analysis to derive the optimal update

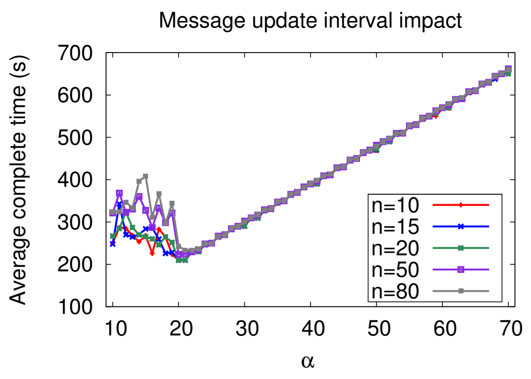


Figure 4.12: Average complete time with different α, while $T_i = 10, k_i = 10$ for all smartphones(Scenario 1)

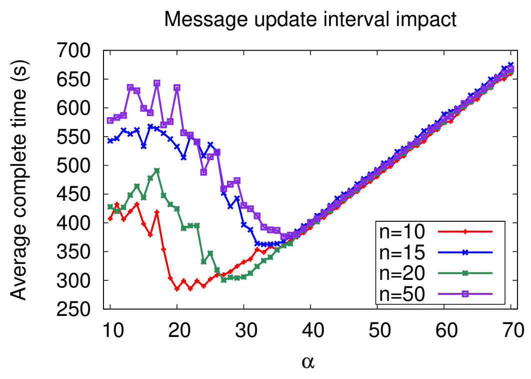


Figure 4.13: Average complete time with different α, while $k_i = 10$ for all phones and $T_i = Random[5,35]$(Scenario 2)

interval value. We have implemented our solution on Android phones and evaluated it with experiments and simulation. The results show that PASA system is effective and efficient for disseminating local messages.

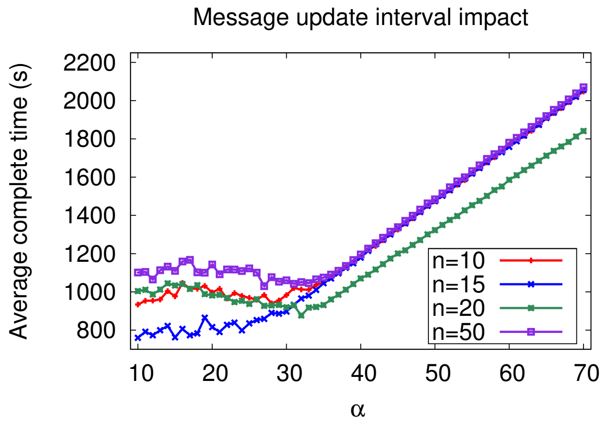


Figure 4.14: Average complete time with different α, while $k_i = Random[5, 30]$ and $T_i = Random[5, 30]$

(Scenario 4)

CHAPTER 5

MOBILE MESSAGE BOARD: LOCATION-BASED MESSAGE DISSEMINATION

In this chapter, we propose a novel message dissemination system, called Mobile Message Board (MMB), for nearby smartphones to share messages with each other based on Ad-Hoc communication model.

The message dissemination in the current social networks follows the client-server architecture. For example, when a Facebook/Twitter user or a forum user posts a message, the message is uploaded to a central server and hosted there. The server will check the user's friend list or followers and forward the message to them, or a user will browser the forum and view the post. This framework, however, does not exploit the strong but hidden link in a social network, the location. Generally, people in the proximity may share common interests. For example, the recommendation of a restaurant may not be interesting to the friends in another city, but could be very useful for a tourist nearby. In addition, a user may want to discover the events around him by browsing/searching the messages posted by other people nearby (probably strangers) without registering every interest with a server.

In addition, the client-server architecture does not work for some applications. For example, one of the major motivating applications in this chapter is the Rogue Access Point detection. We consider in a public area, there might be some maliciously installed WiFi Access Points (AP) that lure users to connect to them. There have been solutions in the literature that can detect different types of rogue APs. The problem we consider is that if a user detects the existence of some rogue APs, how can he effectively notify other users in the area. With the traditional client-server model, the user can report the rogue APs to a server, and then other users, after recognizing their locations, can fetch this information from the server. However, this solution requires other users to connect to the Internet first in order to check with the server, and the connection to a rogue AP may have been established before contacting the server. Once connected with a rogue AP, the rest of the Internet traffic cannot be trusted. The rogue AP may redirect, alter, and manipulate the packets. Therefore, rogue AP detection may fail if it relies on the connection to a server.

We believe that phone-to-phone communication in the Ad-Hoc network model is an appropriate alternative to help local message dissemination. The ad-hoc model can effectively link nearby users even if they are all strangers to each other. In the previous chapter, we have develope, PASA [26], a connectionless communication model built upon Bluetooth or WiFi-Direct that is suitable for short message dissemination. In this chapter, we use PASA model to build a mobile message board for host messages in a particular area. We consider that each smartphone may have its own schedule to turn the Bluetooth/WiFi-Direct device on or off because of its energy strategy. The main challenge is how to select smartphones to host the messages so that the messages can be available in the system as long as possible. Ideally, we hope the messages are always accessible, i.e., when a new user joins the system at any time he can certainly read all the messages. However, when no smartphone keeps its wireless device always on, there is no guarantee of each message's availability. Our goal is to design appropriate strategies for each phone to manage the messages it hosts to maximizing the chance a user can read all the messages.

5.1 Related Work

In the research of intermittently connected networks, for example, Delay Tolerant Networks and Opportunistic Networks, a number of approaches have been proposed in different areas including link construction, data forwarding and content distribution and security [52–55, 61, 63–65].

To provide a secure network, Musubi [53] proposed a decentralized trusted social services on personal mobile devices. All the communication, in Musubi, is supported using public key encryption thus leaking no user information to a third party. Focusing on enhancing the P2P connection, Point&Connect [52] implements pointing gestures of moving one device towards another in order to enable spontaneous device pairing. In addition, BubbleRap [54] utilizes group membership information to improve standard unicast routing. However, all these works are based on a network connected with Internet or P2P techniques, like WiFi-Direct and Bluetooth, that requires network construction and management.

In PASA [26], the authors introduce a new communication model named passive broadcast that utilizes the peer discovery process in P2P techniques to initialize and manage the ad-hoc network. This approach bypasses the real connection between the users by using the device name as an information carrier. The proposed Mobile Message Board (MMB) system takes advantage of the passive broadcast. However, relative to PASA that focuses on spreading the information, the MMB system mainly concerns about extending the activation period of the messages on the board.

One application of the MMB system is rogue access detection. Rogue Access Point is a wireless access point that has either been installed at a public area, like shopping malls and airports, without explicit authorization from a local network administrator, or has been created by a naive user for convenience. Multiple solutions have been proposed to prevent user from Rogue Access Point [66–70]. These solutions attempt to address the Rogue Access Points based on a centralized server that analyzing wireless traffic and reporting suspicious devices. However, those approaches require users to query their servers to check the database or connect to the access point first. In our scenarios, users may not or very costly have Internet access, for example, in airports while roaming abroad. If users connect to the Rogue Access Point first, they have already exposed to the attackers.

5.2 Problem Formulation

In this chapter, we aim to develop a mobile message board (MMB) system without the support of the Internet infrastructure. We define a target area that hosts the message board, such as an airport terminal or a train station, a dining area with several restaurants, and a building with public hotspots. We assume that the users in the target area are within each other's communication range and they collaborate to host a virtualized message board. A user can leave messages on the board that will be shared with other nearby users, but are effective only in this area.

5.2.1 Ad-Hoc Communication Between Phones

Our problem setting and solution are based on the new connectionless communication model, *passive broadcast*, which has been introduced in [26]. Basically, every node conducts a periodical *scan* to fetch data from nearby nodes if available. When a node intends to send a message, it puts the message in a local buffer which will be sent to the nearby nodes later during their scanning processes. This communication model requires no connection to be established between any two nodes, thus is extremely efficient for message dissemination.

In this chapter, the connectionless model is implemented based on the mandatary 'peer discovery' function in Bluetooth and WiFi-Direct. Specifically, we apply the field of 'device name' to carry the messages. When sending a message, the user will replace the Bluetooth or WiFi-Direct device name with the target messages. The messages will be delivered to other nearby phones when they conduct peer discovery.

We have prototyped this new model on commercial phones. According to our experiments, it does not affect regular Bluetooth/WiFi-Direct functions. For example, a phone using this message delivery model can still pair with other Bluetooth devices such as a headset or a keyboard.

5.2.2 States And Operations Of Each Phone

According to the working procedure of Bluetooth module on the smartphone, in general, there are two states of each phone, **Bluetooth ON** and **Bluetooth OFF**. During the **ON** state, the users are willing to contribute to the local Mobile Message Board system through the ad-hoc communication with the others devices nearby. However, in the **OFF** state, the Bluetooth module is turned off to save energy, suggesting it cannot send or receive any messages in the system. In our setting, we assume each user already configures his own states schedule according to the energy budget and other user-specific factors.

The operations of a smartphone in the system includes: initial scan, message update, assignment and delegation.

- **Initial scan:** Initial scan is the phase when all the users in the system are performing the peer discovery process to collect the parameters from nearby devices such as the state schedules and the limit of message each user can host. Based on the experiment in [26], the initial scan usually takes approximately 12.8 seconds.

- **Message update:** The connectionless communication model utilizes the device name. However, the device name usually has a limited length, e.g., a Bluetooth's device name in Android can be up to 248 bytes. Multiple short messages can be merged into one "device name". But a large message will have to be fragmented to fit in and a phone can periodically update the device name to rotate multiple messages or fragments. In our solution, we use a special character string as the prefix to distinguish the device names that adopt our scheme from the regular ones.

- **Message assignment:** Since each phone has its own states schedule, it is possible that the contributor who wants to share the message in the Mobile Message Board has very limited **ON** time. In order to yield a longer coverage or availability of a message so that others are more likely to receive it, the message owner will assign it to the other phones nearby that can help disseminate the message. The owner has to analyze each phones' state schedule and compare it with its own schedule to choose the best relay phones.

- **Message delegation:** This operation is conducted when a phone is leaving the target area of the MMB. It has to choose a set of phones in the system to delegate the active messages that are currently held by itself. This delegation keeps the messages remain active even when the owner and relay phones leave the system. To choose the right delegated phones, it compares the schedule with the current relay phones that hold the same message and others who are available to hold new messages. Message delegation is also used to improve the active length of the messages.

5.2.3 Objective And Constraints

In this chapter, our goal is to develop efficient algorithms for *message assignment* and *message delegation* to maximize the availability of the active messages on the MMB. Specifically, we assume there are n users, $U = \{u_1, u_2, \ldots, u_n\}$, and k active messages, $M = \{m_1, m_2, \ldots, m_k\}$. Assume each user u_i follows a pre-configured schedule to periodically turn its device on and off. Let T_i be the duration for which u_i keeps its device on and I_i be the interval of two consecutive active periods. We use $L_i = T_i + I_i$ to represent the *cycle length* of each user u_i. Let $x_{ij} \in \{0, 1\}$ be the indicator of whether the user u_i hosts the message m_j. According to the schedule of each user u_i, we use $y_i(t)$ to denote the status of the user at the time point t, i.e., if u_i's device is turned on, then $y_i(t) = 1$; otherwise $y_i(t) = 0$.

Let L_{LMC} be the least multiple common of all the users' cycle lengths of their schedule, i.e.,

$$L_{LMC} = LMC(L_1, L_2, \ldots, L_n).$$

For each message m_j, let $c_j(t)$ indicate if the message is active at time t, and w_j be its weight indicating its importance or urgency. In addition, we include two threshold parameters: each phone can host at most τ distinct messages, and each message can have up to θ replicas in the system.

Therefore, our problem is formulated as follows:

$$\begin{aligned}
&\text{maximize} \sum_{j\in[1,k]} w_j \cdot \frac{\sum_t c_j(t)}{L_{LMC}} && (5.1)\\
s.t.\quad &\forall i, j, \sum x_{ij} \cdot y_i(t) \geq c_j(t) && (5.2)\\
&\forall i, \sum_j x_{ij} \leq \tau && (5.3)\\
&\forall j, \sum_i x_{ij} \leq \theta && (5.4)\\
&x_{ij} \in \{0, 1\} && (5.5)
\end{aligned}$$

The problem is NP-hard (due to the page limit, we omit the proof here.), and in the next section, we'll present our algorithms based on greedy strategy.

5.3 Mobile Message Board

In our solution, each user uses its phone's "device name" to carry the messages on the mobile message board (MMB) and enable the communication protocols with other users. Particularly, three categories of data are hosted on the device name:

- Payload messages: These messages are posted on the MMB and supposed to be accessible to nearby users.
- Control messages: These messages are part of particular communication protocols developed in our solution.

- Header: The header contains general profile information about the user such as the scanning schedule.

Since the length of the device name is limited, our solution allocates a fixed segment for each category of data. We assume that each user device can host at most θ payload messages with the knowledge of the average length of a message. In addition, each user reserves a certain space to hold one control message (the size of the control message will be introduced later when we present the protocols). The rest of the space on the device name will be allocated to the header information.

The following Fig. 5.1 illustrates an example of the data hosted on the device name. All the devices participating our protocols will use a special prefix in the beginning of the device name in order to distinguish themselves. The payload messages are listed with their owner and weight information separated by "##". In our solution, each device is identified by two bytes which is the hashed value of its MAC address. The control message starts with a list of recipients which is followed by the message content. Different from the payload messages, the control messages in our protocols usually are not broadcast messages, but with certain target recipients. Finally, the header contains its ID and the system parameters which include the state schedule and the maximum number of messages it can host.

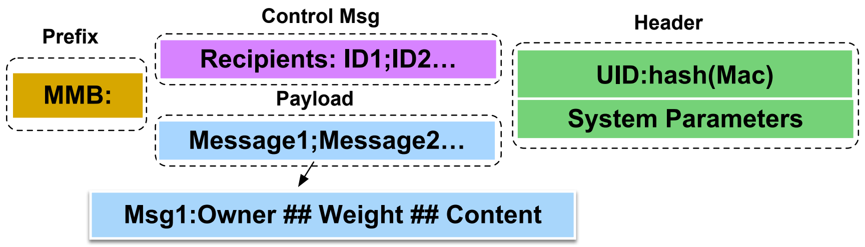


Figure 5.1: Message Format carried by device name

The following Table 5.1 lists the notations we will use in the rest of this chapter. In the following subsections, we present our algorithms in two different cases depending on if a coordinator node exists in the system.

u_i / m_j / w_j	the i-th user / the j-th message / the weight of m_j
T_i / I_i	the length of "on" period / "off" period of u_i
x_{ij}	the indicator of whether u_i hosts m_j
$y_i(t)$	the indicator of whether u_i is on at time t
$c_j(t)$	the indicator of whether m_j is available at time t
τ / θ	the limit of # of msgs per user / # of replicas per msg
AN_i	the set of *accessible neighbors* of u_i
HU_n	the set of messages that u_n currently hosting

Table 5.1: Notations

5.3.1 With A Coordinator

A *coordinator* in our system is defined as a node that is aware all other nodes' schedule. In another word, it has successfully scanned all other nodes' information during the initial scan. Specifically, the coordinator recognizes the $y_i(t)$ for all $u_i \in U$. When such a coordinator exists in the system, it will be responsible for assigning the messages to each smartphone and maximize the total weights as shown in objective 5.1.

We develop an algorithm 5 for the coordinator to complete the assignment. The basic intuition is to apply greedy strategy to select a subset of nodes to host each message. First, it calculates the least multiple common of all the users' cycle lengths of their schedule and combines all the active messages into a set named msgs (lines 1–2). For every message $m_j \in msgs$, it initializes the candidate smartphone set, $\mathcal{S}_j$, which contains all the available users, the selected user set, $\mathcal{A}_j$, which is empty initially and activation function, $\mathcal{F}_j$, that indicates whether the message m_j is discoverable in the MMB system (lines 3–4). The main part of the algorithm is a while loop which will enumerates every message. The loop terminates when all the messages are

Algorithm 5 Centralized Message Assignment

```
1: $L_{LMC} = LMC(L_1, L_2, \ldots, L_n)$
2: msgs=$\{m_1, m_2, \ldots m_k\}$
3: for message $m_j$ do
4:     $\mathcal{S}_j = \{1, 2, \ldots, n\}$, $\mathcal{A}_j = \{\}$, $\mathcal{F}_j[1..LMC] = 0$
5: while msgs $\neq \phi$ and $\cup \mathcal{S}_j \neq \phi$ do
6:     $OPT = i* = j* = 0$
7:     for message $m_j$ in msgs do
8:         for $i \in \mathcal{S}_j$ do
9:             $c = 0$
10:            for $t = 1$ to $L_{LMC}$ do
11:                if $\mathcal{F}_j[t] = 0$ and $y_i(t) = 1$ then
12:                    $c = c + 1$
13:            if $c \cdot w_j > OPT$ then
14:                $OPT = c \cdot w_j$, $i* = i$, $j* = j$
15:    $\mathcal{A}_{j*} = \mathcal{A}_{j*} + i*$
16:    $CM_{j*} = CM_{j*} + 1$, $CU_{i*} = CU_{i*} + 1$
17:    if $CM_{j*} = \theta$ then
18:        $msgs = msgs - m_{j*}$
19:    if $CU_{i*} = \tau$ then
20:        for message $m_j$ do
21:            $\mathcal{S}_j = \mathcal{S}_j - u_{i*}$
22:    for $t = 1$ to $L_{LMC}$ do
23:        $\mathcal{F}_{j*}[t] = \mathcal{F}_{j*}[t] \mid y_{i*}(t)$
```

assigned ($msgs = \phi$), or all the phones are fully loaded ($\mathcal{S}_j = \phi$). The parameters OPT, $i*$ and $j*$ are temporary variables where OPT stores the current optimum for this message m_j; $i*$ is the current selected user id and $j*$ currently message id (lines 5–6). For every candidate in the $\mathcal{S}_j$, the coordinator calculates the activation period indicator, c, and the total weight, $c \times w_j$. If the $c \times w_j$ is larger than the current optimum OPT, then it updates the temporary parameters (lines 8–14). When m_j is assigned to u_i, it updates the m_j's selected user set, $\mathcal{A}_j$, the counter of currently hosting message, CM_j^* and the counter of currently hosting users, CU_i^* (lines 15–16). The algorithm monitors these two counters CM_j^* and CU_i^* to check if they reach the preset thresholds and updates the $msgs$ and $\mathcal{S}_j$ sets when needed (lines17–21). Finally, the coordinator refreshes activation function for m_j (line 22).

5.3.2 Without A Coordinator

In some scenarios, a coordinator may not exist in the system because of the misalignment of the state schedules among all the nodes. In this subsection, we present a solution for the MMB without a coordinator. Our solution includes two stages: initial assignment and message delegation, with the objective of maximizing the message's active period.

The initial assignment is performed by every message owner. Since the owner may have very limited ON period (according to its state schedule), whenever a user generates a message, it runs through the initial assignment to choose the best holders for it.

Algorithm 6 Initial Assignment (by the message owner u_o)

```
1:  users = $\{u_i \mid CU_i < \tau$ and $u_i \in AN_o\}$, $\mathcal{A} = \{\}$, $CM = 0$
2:  while users $\neq \phi$ and $CM < \theta$ do
3:      $OPT = i* = 0$
4:      for user $u_i$ in users do
5:          c=0
6:          for $t = 1$ to $L_{LMC}$ do
7:              if $\mathcal{F}_j[t] = 0$ and $y_i(t) = 1$ then
8:                  $c = c + 1$
9:          if $c > OPT$ then
10:             $OPT = c$, $i* = i$
11:     $\mathcal{A} = \mathcal{A} + u_{i*}, CU_{i*} = CU_{i*} + 1$, $CM = CM + 1$
12:     if $CU_{i*} = \tau$ then
13:         users = users - $u_{i*}$
```

Algorithm 6 describes the details of the initial assignment stage. Firstly, the owner, u_o, initializes the selected user set, $\mathcal{A}$ and counter of replicas, CM. In addition, it needs to combine its accessible neighbors into a set named $users$. This set excludes the users that have already reached their maximum number of messages threshold (line 1). When the set $users$ is non-empty and CM is still under the threshold, for every $u_i \in users$, it computes the activation period if u_i is chosen to be a message holder by using the message's activation function in a L_{LMC}. The algorithm continues until it finds the largest c among all the $u_i \in users$ (lines 2–10). After choosing the candidate u_i from AN_{u_0}, we update selected user set and the counter of replicas for this message. Moreover, we add u_i^* to the selected user set and refresh the number of hosted messages of u_o. Finally, it keeps monitoring whether u_i has meet its limit. If it is, we remove it from the candidate user set. (lines 11–13).

Algorithm 7 Message Delegation (by the selected u_o)

```
 1: users = $\{u_i \mid CU_i < \tau$ and $u_i \in AN_o\}$, $\mathcal{A} = \{\}$, $CM = 0$
 2: $holders_{m_j} = \{u_n \mid m_j \in HU_n\}, OPT$
 3: while $holders_{m_j} \neq \phi$ and excludes $u_o$ do
 4:   for Every $u_n \in holder_{m_j}$ do
 5:     $n = n^*$
 6:     for $t = 1$ to $L_{LMC}$ do
 7:       if $y_i(t) = 1$ then
 8:         $\mathcal{F}_j(t) = 1$
 9:   $holders_{m_j} = holders_{m_j} - u_{n^*}$
10: while users $\neq \phi$ and $CM < \theta + 1$ do
11:   for Every $u_i$ in users do
12:     for $t = 1$ to $L_{LMC}$ do
13:       if $y_i(t) = 1$ then
14:         $\mathcal{F}_j(t) = 1$
15:     $OPT^* = \sum_1^{LMC} \mathcal{F}_j(t)$
16:     if $OPT^* > OPT$ then
17:       $OPT = OPT^*, DU = u_i$
```

Without the coordinator's assistance, the owner may fail to choose the best users to host the message due to the limit of accessible neighbors. In this case, after the initial assignment, the selected users keep tracking to see if there is any better user that can hold the message and lengthen the total active time.

Algorithm 7 illustrates how u_o delegate the message m_j to a better user. First, it initializes the users set, the selected user set and message m_j's current holder set which is the set $\mathcal{A}$ from algorithm 6 (lines 1–2). Then, we calculate the activation function , $\mathcal{F}_j(t)$, without u_o's hosting (lines 3–10). Next, the algorithm computes whether the active time becomes longer if one of the users in u_o's available neighbors is chosen to host m_j. After enumerating all the available neighbors, we select the user that can return the maximum OPT (lines 11–13).

5.4 Implementation And Evaluation

In this section, we will first introduce the system implementation of our MMB system and then present the performance evaluation results from our experiments and simulation.

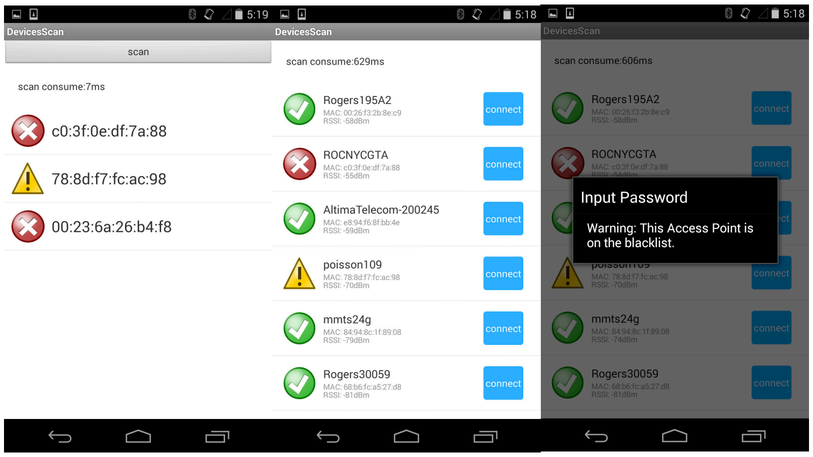
5.4.1 System Implementation

In the implementation, we take advantage of the MMB system in the application of warning the Rogue Access Point (RAP) in an airport when users are not able to or very costly to connect to the Internet. In this scenario, users report the RAP by changing their Bluetooth name and publish the information to the MMB. When a user finds a possible RAP, it changes its Bluetooth name to "MMB:Mac:X", where MMB is a prefix that is used to filter out unrelated Bluetooth devices, Mac is the RAP's MAC address and X stands for 1 or 2 which indicates rogue access point or slow speed access point.

We implement the prototype as an Android application for smartphone. However, for off-the-shelf phones, as long as they have Bluetooth modules and can change their namespaces, they can contribute to the system by reporting the suspected access points. Fig. 5.2 is a set of screen shots of the Android application. Upon opening the application, the users need to initiate scanning to discover the nearby MMB warning messages (the left figure of Fig. 5.2). Then, they can start discovering the active nearby Wi-Fi hotspots and the reported access points are clearly marked (the middle figure of Fig. 5.2). When a user attempts to click "connect" button, the warning message will pop up (the right figure of Fig. 5.2).

5.4.2 Environment Settings

In this subsection, we discuss the details of the environment settings in both experiments and simulations.



scan consume:7ms
c0:3f:0e:df:7a:88
78:8d:f7:fc:ac:98
00:23:6a:26:b4:f8

scan consume:629ms
Rogers195A2
MAC: 00:26:f3:2b:8e:c9
RSSI: -58dBm
ROCNYCGTA
MAC: c0:3f:0e:df:7a:88
RSSI: -55dBm
AltimaTelecom-200245
MAC: e8:94:f6:8f:bb:4e
RSSI: -59dBm
poisson109
MAC: 78:8d:f7:fc:ac:98
RSSI: -70dBm
mmts24g
MAC: 84:94:8c:1f:89:08
RSSI: -79dBm
Rogers30059
MAC: 68:b6:fc:a5:27:d8
RSSI: -81dBm

scan consume:606ms
Rogers195A2
MAC: 00:26:f3:2b:8e:c9
RSSI: -58dBm
ROCNYCGTA
MAC: c0:3f:0e:df:7a:88
RSSI: -54dBm
Input Password
Warning: This Access Point is on the blacklist.
MAC: 78:8d:f7:fc:ac:98
RSSI: -70dBm
mmts24g
MAC: 84:94:8c:1f:89:08
RSSI: -74dBm
Rogers30059
MAC: 68:b6:fc:a5:27:d8
RSSI: -81dBm

Figure 5.2: Rogue Access Point warning application of MMB system

5.4.2.1 Experiments

In the experiments, we use 6 Android smartphones that include LG Nexus 4, Nexus 5, Google Nexus 7 and Samsung Galaxy S4. During the experiment, we distribute the smartphones on the desks in our lab which is around $20m^2$ to form a fully connected network. The state schedule, T_i and I_i, is preloaded in each phone. In addition, other user settings, like messages and thresholds, are pre-configured.

5.4.2.2 Simulations

We use the Crawdad dataset from Dartmouth [71] for the simulation. The particular dataset we use is Sigcomm 2009 trace that contains Bluetooth encounters, opportunistic messaging, and social profiles of 76 users of MobiClique [72] application. Based on the length of encounter time and activity, we filter out the users who only appear in a short period and lack of activities. Then, there are 52 users left whom we consider as the valid clients. We drive the encounter length and number of social messages from the dataset as the parameters. In the simulation, we randomly pick up the users from the valid clients pool. Due to lack of the Bluetooth OFF state data, we assign each user a OFF length randomly so that each user has a state schedule. Finally, the parameters are fed to our simulator to examine the Mobile Message Board system.

5.4.3 Performance Evaluation

In this subsection, we present the evaluation results from both Android smartphones experiments (small scale) and simulation (large scale). We consider the following four different cases for our evaluation.

- **Case 1**: Only one message generated by one user in the MMB system.
- **Case 2**: Multiple messages generated by the same user.
- **Case 3**: One message generated by multiple users.
- **Case 4**: Multiple messages generated by different users.

The major performance metric we exam is the Activation Rate (AR) for each message. For each $m_j \in M$, its activation rate is defined as,

$$AR_j = \frac{\sum_{t\in[0,L_{LMC}]} c_j(t)}{L_{LMC}} \tag{5.6}$$

where, according to Table 5.1, $c(t)$ is the indicator of whether m_j is active at time t and L_{LMC} is the function that returns the least multiple common of all the users' state schedules in the MMB system.

5.4.3.1 Experimental Results

We manually set the required state schedule and randomly choose the limit number of messages per user in the range from 1 to 3 for each phone as shown in Table 5.2. In the experiments, we mainly focus on the parameters', τ and θ, impact on activation rate. (the parameters' definitions are in Table 5.1). As for the message owners, we randomly pick from the six candidates for different cases.

u_{id}	T	I	τ	u_{id}	T	I	τ
1	10	10	3	2	15	10	1
3	15	25	1	4	10	20	1
5	15	20	2	6	10	15	2

Table 5.2: Parameters

Fig. 5.3 plots the average activation rate per message under different θ. As the figure displays, if the $\theta = 1$, it suggests each message only has one host. Except case 3, the activation rate of the other three cases is very low (less than 0.6). The reason lies in the fact that for the case 1, 2, and 4, each message can only has one host controlled by θ. However, in case 3, since different users generate the same message, it can bypass the limit of θ. The activation rate grows along with the replicas increasing since the more hosting users get involved, the less OFF state is in each period.

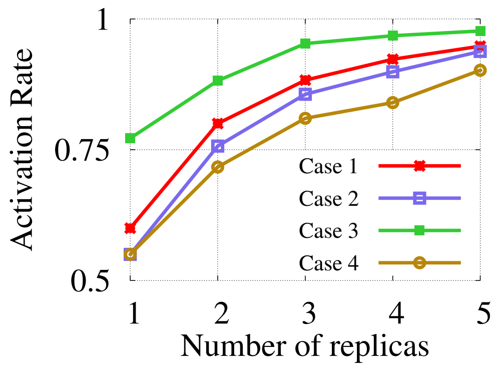


Figure 5.3: Activation Rate versus number of replicas (θ)

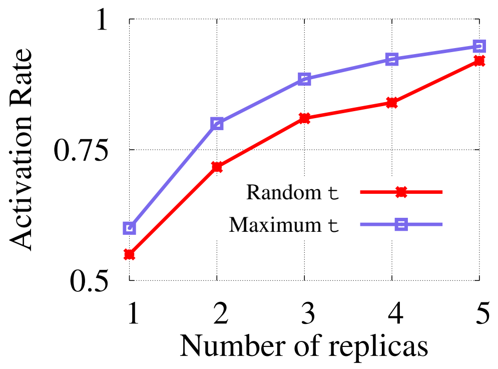


Figure 5.4: Comparison of random and maximum τ values (case 4)

Then we focus on the impact of τ which controls the maximum number of messages each user can host. As shown in Table 5.2, τ is randomly selected from 1 to 3. Fig. 5.4 compares the case 4 under the random setting above and the maximum setting that assigns every user the limit of 3 messages. The figure indicates that the maximum τ setting can improve the activation rate. The difference of state schedules results in the improvement since the user with largest ON state portion, user 2, can hold more messages (1 v.s 3).

5.4.3.2 Simulation Results

In the simulation, we use the parameters that derived from the trace. However, it lacks the records of Bluetooth OFF state and messages replicas. Therefore, we randomly pick the OFF state from 5 to 50 and the number of replicas from 1 to 5.

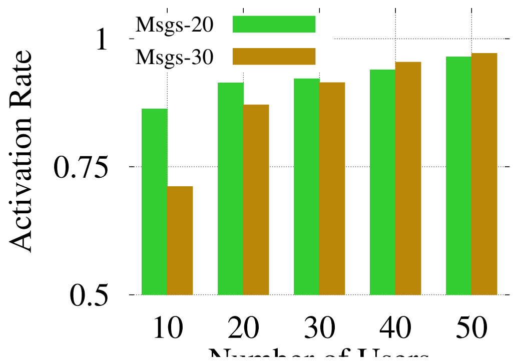


Figure 5.5: Activation Rate (case 4) with random number of replicas

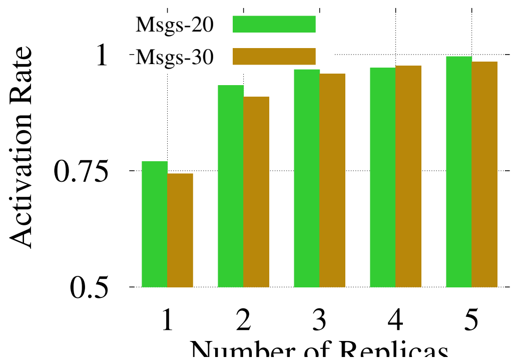


Figure 5.6: Activation Rate (case 4) with fixed number of replicas and 30 users

Fig. 5.5 illustrates activation rates (with coordinator) under different number of users. In addition, the number of messages is set to 20, 30. Consequently, the activation rate grows with the user size increase. The more users are in the system, the more choices are for each message. The figure also implies a fact that the activation rate is higher for 20 messages than 30. It can be attributed that, with fixed user size, less messages in the system results in more choices for each message.

Fig. 5.6 plots the activation rate (with coordinator) under different fixed replicas (all messages has the same θ) with 30 users. Since the user size is fixed, the activation rate goes up along with the replicas. Moreover, the Msg-20 experiment has a higher activation rate than Msg-30. It reflects the fact that with the same user group, the smaller message set size results in more choices for each message.

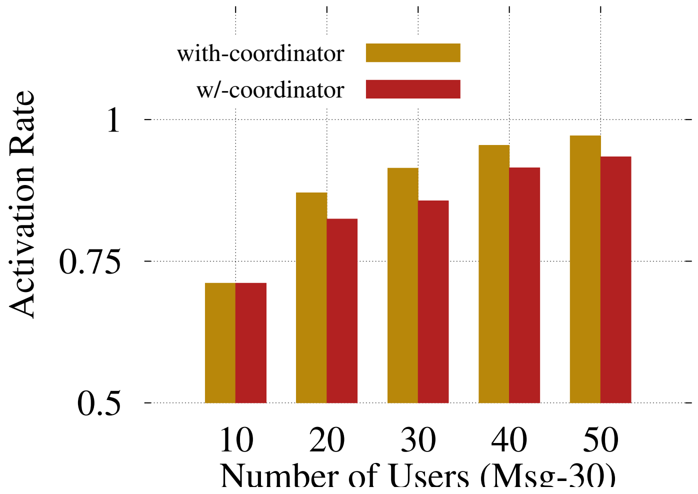


Figure 5.7: Activation Rate (case 4) with fixed number of replicas and 30 messages

Next, we compare the two approaches, with and without the coordinator. To compensate the absent of coordinator, in the second approach, the message, after initial assignment, will be delegated to other users if it can improve the activation period. It takes some time for the delegation process. Therefore, the simulation is configured to keep running for two periods (two L_{LCM}). Fig. 5.7 compares the two approaches with Msg-30 under different size of users. As shown on the figure, two approaches achieve the same performance with 10 users. This is because the central message assignment by the coordinator and initial message assignment by the owners choose the same holders for the messages. Without the coordinator's assistance, the second approach fails to choose the optimal users for some messages in the other settings that results in the decrease of average activation rate in the system. However, message delegation can compensate some of the decrease. For example, in user 20 setting, the average activation rate is 0.77 after message initial assignment. The overall activation rate improves to 0.82 in two periods after the delegation.

5.5 Summary

This chapter presents a Mobile Message Board (MMB) system for smartphone users to share messages in a target area. Our solution is built on ad-hoc communication model without the support from the Internet. We present algorithms that appropriately manage the messages on each participating phone to maximize the message availability in the system. In addition, we have implemented our solution on off-the-shelf phones. Our evaluation based on experiments and simulation shows that our system is efficient and effective for disseminating messages in the proximity.

CHAPTER 6

SKYFILES: EFFICIENT AND SECURE CLOUD-ASSISTED FILE MANAGEMENT

In this chapter, we develop Skyfiles which is a system for smartphone users to manage their files in the cloud storage. Our basic idea is to launch a cloud instance to assist users to accomplish the file operations. It is motivated by the fact that the cloud instance is inexpesive and sometimes free. For example, Amazon Web Service (AWS) [73] provides 750 free Micro instance hours per month. By using the resources of the instance, smartphone users will significantly reduce the bandwidth consumption for file operations. Skyfile still does not require users to keep local copies of files, but provides advanced and secure operations.

6.1 Related Work

In the past few years, cloud computing and storage technology gained increasing attention in mobile applications.In [74], a survey studies offloading computation with the objective of extending smartphone battery life. MAUI [75], Cuckoo [76] and ThinkAir [77] implement an Android framework on top of the existing runtime system. These three systems are easy to deploy because they only need to access to the program source codes, and they do not require modifying the existing operating system. They provide a dynamic runtime system which can, at runtime, decide whether a part of an application is better to be executed locally or remotely.

Besides offloading computation, SmartDiet [78] aims at offloading communication-related tasks to cloud in order to save energy of smartphones. The authors of [79] investigate Dropbox users to understand characteristics of personal cloud storage services on mobile device. Their results show possible performance bottlenecks caused by either the current system architecture and the storage protocol. In [80], the authors focus on the impact of virtualization for Dropbox-like cloud storage systems. Another related area in the prior work is to use cloud to enhance mobile device security and privacy. CloudShield [81] presents an efficient anti-malware smartphone patching with a P2P network on the cloud and CloudShield can stop worm spreading between smartphones by using cloud clones. The authors of [82] propose the Confidentiality as a Service (CaaS) paradigm to provide usable confidentiality and integrity for the bulk of users who think the current security mechanisms are too complex or require too much effort for them. In their paradigm, CaaS separates capabilities and requires less trust from cloud or CaaS providers, leverages existing infrastructure and can performs automatic key management. They test CaaS on facebook, dropbox and some popular service providers.

6.2 Background

This subsection introduces background information about cloud storage service. Most of our experiments in this paper are conducted on Dropbox platform. Thus, this section regards Dropbox as a representative service provider and briefly describes the Dropbox architecture and functions for user applications.

As a leading solution in personal cloud storage market, Dropbox provides cross-platform service based on Amazon Simple Storage Service(S3) for both desktop and mobile users. The architecture of Dropbox follows a layered structure. At bottom level, Amazon S3 infrastructure provides a basic interface for storing and retrieving data. The above layer is Dropbox core system which interacts with S3 storage service and serves higher level applications. The top layer is the official Dropbox application and a set of APIs for developers to build third party applications.

To manage third party applications, Dropbox assigns each of them an unique *app key* and *app secret*. When a user launches an application, Dropbox server follows the OAuth v1 [83] for authentication. Fig. 6.1 shows the basic authentication process. When launched by a user, the third party app contacts the Dropbox server to obtain a one time request token and request secret(step 2,3). Then, the app uses them to forma redirect link and present the link to the user(step 4). When accessing this link, the user will be prompted to login his Dropbox account and the Dropbox server will verify the redirect link and the user's login information. After a successful login, the server will return an *access token* and *access secret* to the application (step 6), which grants access permissions on the user's data stored on Dropbox.

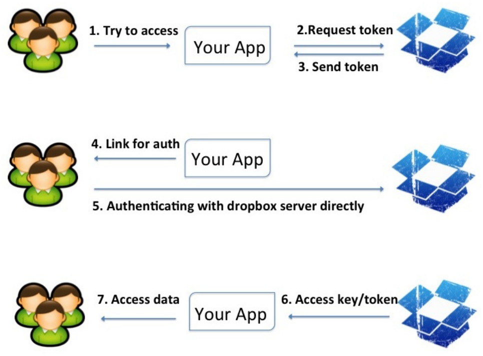


Figure 6.1: Authentication Procedure

6.3 Basic Solutions

In this section, we present our basic solutions in Skyfiles for supporting cloud file managements for mobile devices. We first describe the framework of Skyfiles and cloud-assisted file operations. Then, we focus on the file transfer between two users' cloud storage in Skyfiles. Our objective is to accomplish file operations with minimum network bandwidth assumption.

6.3.1 Framework And Basic Cloud-assisted Operations

In Skyfiles, each mobile device is associated with a cloud storage account. Similar to other related apps, Skyfiles by default does not keep local copies of the files stored in cloud because of the storage limit and bandwidth consumption. Instead, Skyfiles maintains a *shadow file system* on the mobile device which includes the meta information of the files stored in cloud. This local file system is built on service provider's APIs and synchronized with the cloud storage.

Skyfiles supports two categories of file operations. The first category is the basic file operations that are commonly available in service providers' APIs such as creating/deleting/renaming a file. The second category is a new set of advance file operations that require assistance from a cloud instance. Skyfiles recognizes the category each operation request from a user belongs to and processes it accordingly. The first category will be handled by regular API function calls. For the second category, Skyfiles will create a cloud instance and then forward the file operation request to the instance for processing. In particular, we have designed the following four cloud-assisted operations in Skyfiles:

- **Download:** This operation allows a user to download files directly to his cloud storage. Given the location of the target files such as URLs, the conventional way of downloading is to first obtain the files on user devices and then synchronize with/upload to the cloud storage. In Skyfiles, the cloud instance will fetch the files and then upload them to the user's cloud storage. Thus the downloading and uploading will not consume mobile device's bandwidth.

- **Compress:** This operation enables a user to compress existing files or directories stored in cloud. If user devices hold local copies of the target files, the operation can be easily accomplished and the generated compressed file can be uploaded to the cloud storage. In Skyfiles or other similar apps for mobile devices, however, the actual file contents are not available. Thus we design an interface that the user can select the target files based on the local shadow file system with meta data and then the compressing operation is forwarded to a cloud instance for execution. The instance will fetch the specified files from the cloud storage, compress them, and upload the compressed file back to the cloud storage.

- **Encrypt:** This operation is similar to compress and does not exist in current apps that do not keep local file copies. In Skyfiles, the user can choose the target files and the cipher suite including the cryptographic algorithm and key. The encryption operation will be sent to a cloud instance. Similarly, the cloud instance will download the target files from the user's cloud storage, encrypt them, and send the ciphertext back to the cloud storage.

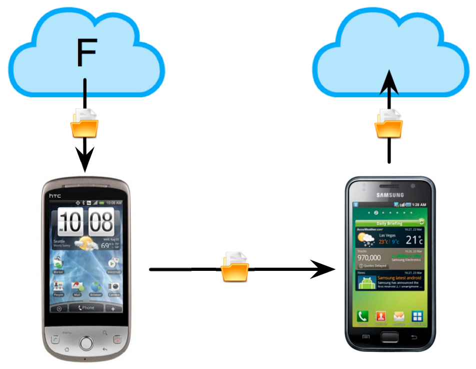
- **Convert:** The last operation is particularly for media files such as pictures and video clips. When a user wants to view a picture stored in cloud, he has to download it to his smartphone. Nowadays, high-resolution picture files could be very large, but a smartphone user may not benefit from it because of the limited screen size. In Skyfiles, therefore, a user can specify an acceptable resolution when viewing a picture and the request will be processed by a cloud instance. The original picture will be downloaded to the instance and then converted to a smaller file according to the user-specified resolution. Finally, the converted picture is sent to the user.

Overall, we develop the above set of advance file operations for mobile devices which are impossible to achieve without local file copies. In Skyfiles, a cloud instance is launched to assist a user to accomplish these file operations. During the execution of the file operations, the cloud instance will periodically send heartbeat messages to the smartphone to report the progress and status. The smartphones only consumes a small amount of bandwidth for exchanging control messages with the cloud storage server and the cloud instance.

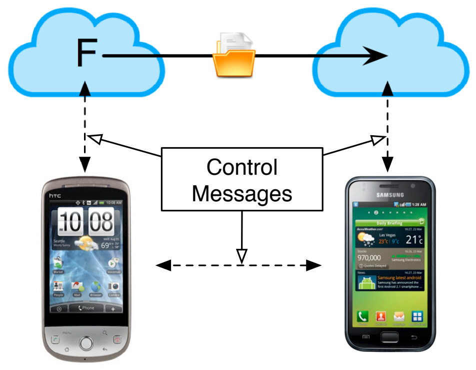
6.3.2 File Transfer Between Users

In this subsection, we present an important feature of Skyfiles which allows file transfer between users. While most cloud storage services allow a user to share files with another user, copying files across different user spaces is not supported. However, file sharing between users cannot substitute file transfer (make a copy). With file sharing, a user's actions on the shared files will affect other users. For example, if a user deletes the shared files, all the other users lose those files too. In this subsection, we consider the file transfer between user spaces illustrated in Fig.6.2. Assume two users carrying smartphones with data plan subscription meet with each other and both have storage spaces on cloud. One user (as the sender) wants to transfer files in his cloud storage to the other user's (as the receiver) cloud storage. We aim to develop an efficient solution to support this feature with as little network bandwidth consumed as possible. In the conventional solution, the sender can download the target files to his smartphone and send them to the receiver's phone through Internet or short range connection such as Bluetooth and NFC. Upon receiving the files, the receiver's phone can upload or synchronize them with the cloud storage. This solution, however, is not efficient in terms of bandwidth consumption, especially when the target files are large, e.g., bunch of pictures and video clips, as the sender has to download the entire files and the receiver has to upload all the files.

In Skyfiles, we solve the problem by following the same design principal that utilizes a cloud instance to assist users to transfer files between their cloud storage spaces. Basically, a cloud instance is initialized and plays a role of relay node. It fetches the target files from sender's cloud storage space and then forward them to the receiver's cloud storage. In this means, both sender and receiver's smartphones do



(a) Conventional Solution



(b) Skyfiles

Figure 6.2: File Transfer between Two Users

not have to hold a local copy of the target files and the bandwidth is consumed only by control messages between smartphones and the cloud instance/cloud storage server. The basic design of using a cloud instance, however, is challenging in practice when no trust has been established between the sender and receiver. The cloud instance can be created by either the sender or receiver. In either case, it is not a secure solution for the party who does not own the instance because the cloud instance will need to obtain the security credentials of cloud storage from both sender and receiver to complete the file transfer. Therefore, the owner of the cloud instance will be able to access the cloud storage space of the other user which could breach data privacy and lead to other malicious operations.

To address the above issue, in Skyfiles, we develop a solution that requires a cloud instance from both sender and receiver. In particular, we implement the following Algorithm 8 to accomplish the file transfer. Assume user U_A is trying to send a file F (F can also represent a set of files) to user U_B. Let F_{src} be the location of F in U_A's cloud storage and F_{dst} be the destination location that U_B will put in his cloud storage. In our protocol description, U_A and U_B also represent the users' smartphones. The following Algorithm 8 presents the major steps for file transfer. First, U_A starts a cloud instance (I_A) and uploads the security credentials for accessing his cloud storage space to the instance. U_A's request also includes the source file location F_{src} and an *intermediate file location* URI_F (uniform resource identifier) which indicates where to store F on the instance. The cloud instance will use the security credentials to download the target file F to its local disk. At this point, I_A needs to make F accessible to user U_B. It first sends U_A the intermediate file location URI_F. Then, I_A can set F publicly available or create a guest account and set the permissions of F so that only the guest account can access it. In the latter case, the security information for logging as the guest account, such as login password or identity file, needs to be sent back to U_A as well. After that, the steps on U_A's side have been completed. Then, U_A needs to notify U_B necessary information for accessing F. Since this step of communication includes sensitive information, Skyfiles adopts NFC protocol to securely deliver URI_F and optional login information from U_A's smartphone to U_B's phone. At receiver's side, U_B also starts a cloud instance I_B which obtains F from I_A based on URI_F. Finally, I_B uploads F to F_{dst}. Here, both sender and receiver start a cloud instance to behave as their agents. The data transfer of F is between cloud instances and cloud storage servers which does not consume bandwidth of users' smartphones. Meanwhile, the security credentials for accessing cloud storage are only sent to the instance created by the same owner. Thus, in Skyfiles, file transfer between two users' cloud storage is efficient and secure.

Algorithm 8 File Transfer from U_A to U_B

```
1: $U_A$ starts a cloud instance $I_A$
2: $U_A \rightarrow I_A$ (cellular network) : $U_A$'s security credentials to access his cloud storage
   space, source file location $F_{src}$, and intermediate file location $URI_F$
3: $I_A$ downloads $F$ from $U_A$'s cloud storage and stores it at $URI_F$
4: $U_A \rightarrow U_B$ (NFC) : $URI_F$ (intermediate file location on $I_A$)
5: $U_B$ starts a cloud instance $I_B$
6: $U_B \rightarrow I_B$ (cellular network) : $U_B$'s security credentials to access his cloud storage
   space, $URI_F$, and destination file location $F_{dst}$
7: $I_B$ copies $F$ from $I_A$ ($URI_F$) to its local storage
8: $I_B$ uploads $F$ to $U_B$'s cloud storage space ($F_{dst}$)
```

6.4 Solutions With Shared Cloud Instances

The solutions presented above efficiently enable smartphone users to manage their files on cloud by launching cloud instances for assistance. In practice, however, the following two issues may hinder the deployment of the proposed solutions. First, initializing a cloud instance incurs a significant overhead, e.g., the overhead ranges from 15 seconds to 30 seconds in our experiments in Section6.5. In reality, such a long delay is not suitable for some file operations which require instant response and may negatively affect users experience. The second issue is the cost of launching cloud instances. Although cloud service is inexpensive, frequently starting cloud instances may still increase the cost of users. High cost could be caused by users' misconfiguration and incorrect design of Apps. Users can certainly monitor the usage and charge on their cloud service account to avoid unexpected costs. But it could greatly limit the features of Skyfiles.

In this section, we propose an enhancement for Skyfiles that solves the overhead and cost issues by allowing users to share cloud instances with each other. It is motivated by the fact that cloud service providers charge the instance service at a certain time granularity. For example, AWS, Microsoft Azure [84] and HP Cloud [85] charge the usage of cloud instance at the granularity of an hour. For regular file operations, it is a excessive time period. If a user starts a cloud instance for a file operation, he does not have to terminate the instance when the operation is done. The instance can be kept running until an additional cost is about to be charged. For example, assume a service provider charges the instance service in the time unit of an hour, when a user starts an instance and finishes his file operations in the first 5 minutes, the instance he started can stay active for another 55 minutes without extra cost for him. During this idle time period, if the instance can serve other users or other file operations from the same user, the overhead and cost will be both reduced. While benefiting the performance, the design principal of sharing instances among users incurs challenge for security. First, it is risky for a user to upload his security credentials of cloud storage account to other users' cloud instances. The owner of the instance may monitor and catch the security credential, and gain the access to the user's cloud storage space. Second, when open to public, the shared cloud instances may be used by malicious users to launch attacks.

In Skyfiles, we have developed a framework for sharing instances which requires a trusted server. This server maintains a list of available cloud instances for sharing and coordinates the users who request instances and share instances. Skyfiles applies the following two basic policies to address the security concerns. First, an instance is shared in the form of launching a background service and accepting requests from other users, rather than allowing other users to login and execute arbitrary programs. Second, when using a shared instance, a user does not upload the security credentials of his cloud storage account in plaintext, but in an encrypted format. In this way, the owner of the instance can not gain the access to the tenant user's cloud storage space and any user's privileges on shared instances are limited to the specified file operations.

Specifically, there are three types of entities in our design, a trusted server S, a user U_A who wants to conduct file operations on a shared instance, and an available cloud instance I owned by another user U_B. The trusted server holds a binary program P that can be running on a shared instance to provide Skyfile services to other users. Once a user (U_B) decides to share his instance (I), the instance will contact the server and forwards the basic information about I such as operating system, hardware setting, and the time left for sharing. The response from the server is an executable binary P, where a secret key k_P is embedded. We assume k_P is protected by program obfuscation techniques and I_B is not able to derive k_P from P. In addition, the server will periodically change k_P and re-compile the binary. Upon receiving P, I will execute P as a service and be ready to accept other users' requests. The server, on the other hand, adds I_B into the list of available instances for sharing. Finally, each shared instance I can set a scheduled task to automatically shut down the instance before the additional charge is incurred. During the shutdown process, I also notifies the server S which will consequently remove I from the list of available instances for sharing.

$I \rightarrow S$: willing to share;

$S \rightarrow I$: program P embedded with a key k_P;

I executes P;

S adds I into the list of available instances

Single User Operations: When a user (U_A) requests to use a shared instance to conduct operations on his files on cloud, he needs to first contact the trusted server S. S will generate a random seed R_A for the requesting user and apply a hash function H on k_P and R_A to generate another key k_A,

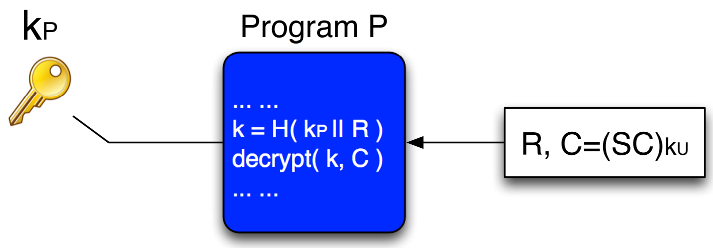


Figure 6.3: Structure of Service Program P: secret k_P is embedded in P which can generate k_U and decrypt ciphertext C.

$$k_A = H(k_P||R_A).$$

The server then chooses a shared instance from the list to serve U_A and it could be an interactive process that involves U_A's opinion. Assume I is selected, the server sends $\{R_A, k_A\}$ and the IP address of I back to the user U_A. Next, U_A will encrypt the security credentials of his cloud storage using key k_A and upload the ciphertext and R_A to the shared instance I. The security credentials will be decrypted in the execution of P, which takes R_A as an input parameter and apply the same hash function H on k_P and R_A.

$U_A \rightarrow S$: request a shared instance;

$S \rightarrow U_A$: I, R_A and k_A;

U_A encrypts security credentials SC_A with k_A, $\{SC_A\}_{k_A}$;

$U_A \rightarrow I$: R and $\{SC_A\}_{k_A}$.

It is possible that multiple users share the same instance in which case the server S will assign different random number R and key k.

File transfer between users: In Skyfiles, two users can also request a shared instance for transferring files between their cloud storages. Following the design in

Section 6.3, the sender will initialize the process and request a shared instance from the server. Compared to the single user operations, file transfer requires both sender (U_A) and receiver (U_B) to send the security credentials of their cloud storage account to the shared instance. In addition, the sender needs to notify the receiver the instance assigned by S. The following Algorithm 9 shows the major messages exchanged in our design. We assume that the server holds a pair of public key/private key, indicated by k_S^+/k_S^- and the public key is known by U_A and U_B.

Algorithm 9 File Transfer with a Shared Instance

```
1: $U_A \rightarrow S$: request a shared instance
2: $S \rightarrow U_A$: $I$, $R_A$, $k_A$, and $\{U_A, I\}_{k_S^-}$
3: $U_A$ encrypts security credentials with $k_A$, $\{SC_A\}_{k_A}$
4: $U_A \rightarrow I$: $R_A$, $\{SC_A\}_{k_A}$, $F_{src}$, and $URI_F$
5: $I$ downloads $F$ from $F_{src}$ and stores it at $URI_F$
6: $U_A \rightarrow U_B$: $I$, $URI_F$, and $\{U_A, I\}_{k_S^-}$
7: $U_B \rightarrow S$: $U_A$ and $I$
8: $S \rightarrow U_B$: $R_B$ and $k_B$
9: $U_B$ encrypts security credentials with $k_B$, $\{SC_B\}_{k_B}$
10: $U_B \rightarrow I$: $R_B$, $\{SC_B\}_{k_B}$, $URI_F$, and $F_{dst}$
11: $I$ uploads $F$ from $URI_F$ to $F_{dst}$
```

Sender U_A initializes the request by contacting the server S. In the response, besides the same information for single user operations, the server sends an additional certificate back $\{U_A, I\}_{k_S^-}$, which is a signature of the requesting user's ID and the assigned instance I. U_A will upload the encrypted security credentials to I as well as the source file location (F_{src}) and intermediate file location (URI_F). Then U_A will notify the receiver U_B the shared instance I and the location of the target files (URI_F). This message is attached with the certificate from S so that the receiver can verify the shared instance I is legitimate. Next, the receiver U_B sends the server a request with (U_A, I). After verifying there exists a shared instance I serving U_A, the server will send back R_B and k_B to U_B so that U_B can encrypt his security credentials in the same way as U_A. Eventually, U_B uploads the encrypted security credentials and the intermediate file location (URI_F) and destination file location (F_{dst}) to the shared instance I.

6.5 Performance Evaluation

In this section, we present the evaluation results of Skyfiles based on experiments. We have implemented Skyfiles system on Android with Dropbox [19] storage service and tested it on Google Nexus smartphone running Android 4.1. For cloud-assisted operations, we use the service provided by Amazon Web Service (AWS EC2) [73] and all the experiments are conducted on the Micro instance (613 MB memory, up to 2 EC2 Compute Units). The major performance metrics we consider are time overhead and bandwidth consumption. To measure the first metric, we insert *System.currentTimeMillis()* java function into the Skyfiles codes to catch the timestamps and calculate the time overhead for each step we are interested in. Measuring bandwidth consumption for a file operation is more challenging. In our experiments, we first connect smartphones to a WiFi access point (AP) for Internet access. A Linksys WRT54GL wireless router running DD-WRT [86] serves as the AP in our tests. For a particular file operation, we start a tcpdump [87] session to record all data packets sent from and received by the smartphone. Then, we use a typical analyzer to retrieve the transferred data size from the trace records. To make the measurement more accurate, we additionally set the firewall on the smartphone to block network traffic from all the applications except Skyfiles. For each particular setting, we conduct five independent experiments and the average values are reported in this section.

6.5.1 Basic File Operations

We first present the bandwidth consumption of basic file operations implemented by official Dropbox APIs. In this test, we create a new Dropbox account with a folder "test" containing 1000 text files (22 bytes each). The operations we tested are 1. log into dropbox; 2. create/delete a folder (under the root directory); 3. create/delete/rename a file (under “test”); 4. enter/leave a folder (“test”).

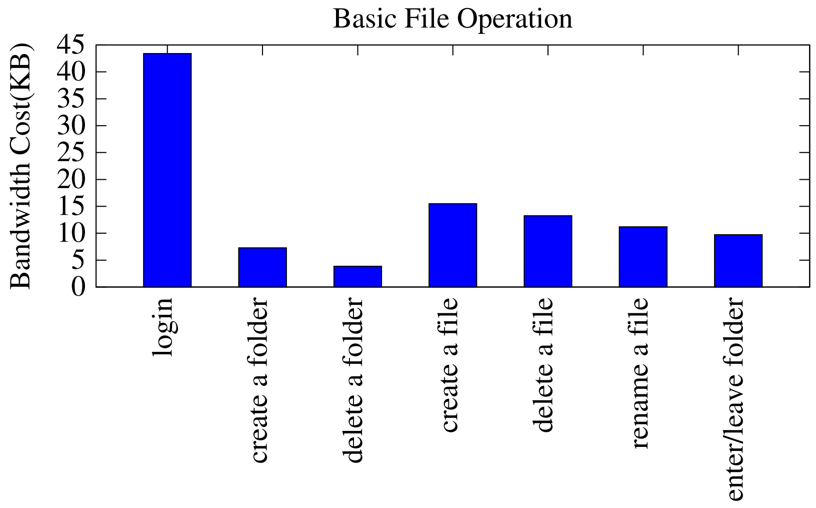


Figure 6.4: Bandwidth Consumption of Basic File Operations

For each file operation, Dropbox server requires security credential to be attached and the communication is based on SSL. As shown in Fig. 6.4, the login process consumes the most bandwidth because the interaction for authentication and Dropbox APIs need to recursively fetch meta data to synchronize/update the local shadow file system. Creating and deleting an empty folder consumes 7.3K and 3.9K bytes respectively which are the minimum costs among the tested operations. Creating and deleting a text file is similar to the previous case. When tested in the folder “test,” it certainly incurs more bandwidth cost (15.5K and 11.2K bytes). The reason is that the folder contains 1000 other files and once a change is made in the folder, Dropbox APIs will re-fetch the list of files in it. Finally, when a user enters the folder and then leave, this operation costs 9.7K bytes bandwidth which is slightly lower than creating/deleting a file. The dominant factor is still fetching the entire file list.

6.5.2 Cloud-assisted Advance File Operations

In this subsection, we evaluate the cloud-assisted file operations in Skyfiles, particularly download and compression. Due to the page limit, we omit the results for encrypt and convert operations. The performance of encrypt operation is similar to that of compress operation. Skyfiles supports these operations with user-created cloud instances and shared instances. The difference on the performance is that using shared instances saves the initial cost (mainly the time overhead) of starting a cloud instance. Therefore, we first present the overhead of starting a cloud instance and then show the performance of these operations under the assumption that a cloud instance has been available. The workload we use for testing includes 4 sets of files: one picture (16M bytes), five pictures (83M bytes), and two video clips (63M and 127M bytes).

Overhead of starting a cloud instance: We conduct five groups of tests in this experiment at different time of a day. Each group contains 5 individual operation of starting a AWS Micro instance. The operation ends when the user is able to log into the instance. The following Fig. 6.5 illustrates the results of the average value and variance. Overall it is a time-consuming process as all the tested cases spend more than 15 seconds in starting an instance. After sending the request, the user has to wait a long time until the cloud instance to be ready. Amazon service spends most of the time in allocating necessary resources for the instance and booting it. In the rest part of this subsection, the performance overhead does not include the initial phase of starting an instance, thus it is for the case of using shared instance. If the user starts his own cloud instance, the extra overhead could range from 15 seconds to 30 seconds based on Fig. 6.5.

Time overhead of download operation: In this test, we let the cloud instance download the files in our workload and upload them to our Dropbox storage space. The target files are hosted in one of our servers. As we can see from Fig. 6.6, up-

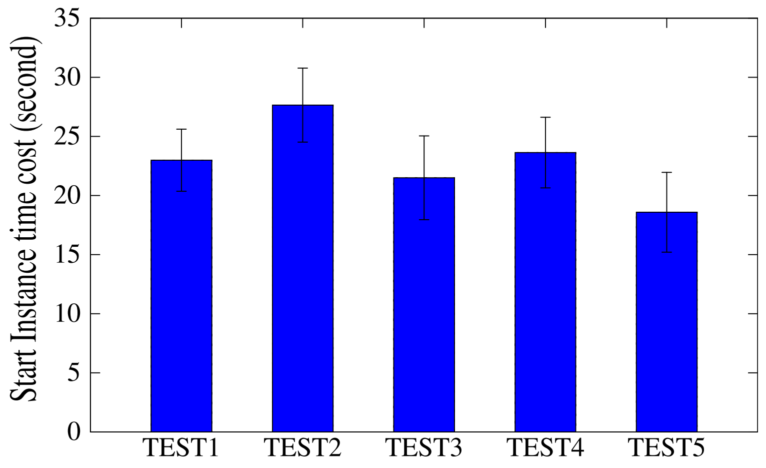
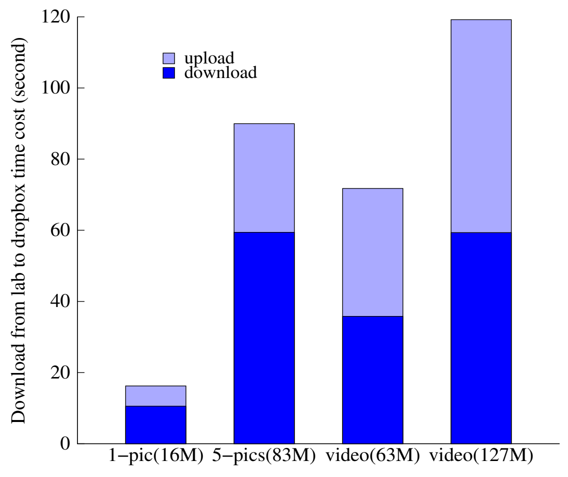


Figure 6.5: Overhead of Starting a Cloud Instance

load/download overhead is roughly proportional to the file size. Uploading is faster than downloading because the instance we use (AWS EC2) and the Dropbox service (AWS S3) belong to the same cloud service provider. Overall, the transmitting rate is around 1M bytes per second which is much faster than downloading files to the smartphone and then uploading them to Dropbox cloud storage via cellular network.



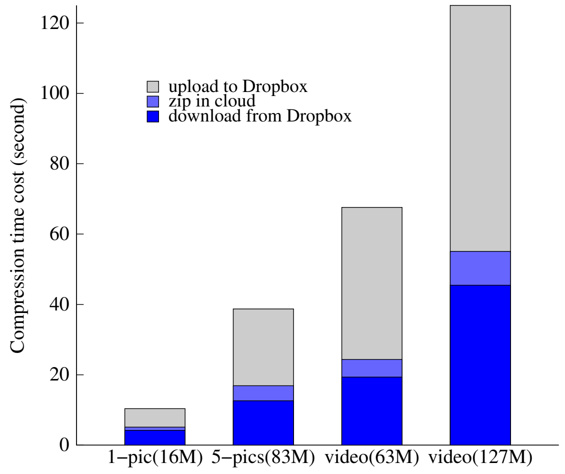


Figure 6.6: Download file time cost **Figure 6.7:** Compress file time cost

Time overhead of compress operation: In this experiment, we test compression operations on Dropbox files. Particularly, we use gzip to compress the files downloaded to the cloud instance and then to upload the compressed file back to Dropbox cloud storage. Fig.6.7 shows the breakdown time overhead of this operation. Uploading process costs the most and followed by downloading and compression process. Downloading becomes faster than uploading because the source files are hosted on Dropbox. This operation in Skyfiles is fast compared to compressing files locally stored on smartphones. For example, compressing one picture (16M) and 5 pictures (83M) cost 10.4s and 38.7s in total. The compressed files in these two cases are 7.7M and 40.0M bytes.

Bandwidth consumption of advance file operations: The bandwidth consumptions of advance file operations are similar as only control messages are being exchanged. We only show the performance of compress operation in Fig.6.8 due to the page limit. Uploading process cost which includes control messages is very small, 3.86kB, 3.95kB, 3.79kB and 3.31kB, respectively. The downloading bandwidth varies by different file sizes and most of it is consumed by our periodical heartbeat messages reporting the status of the operation. Even with the reporting messages included, the bandwidth consumption is always lower than 190K bytes regardless of the target file size.

6.5.3 File Transfer Between Users

The performance of file transfer is similar to the above advance file operations. The process is the same as the downloading phase in *compress operation* followed the uploading phase in *download operation*, i.e., target files are downloaded by an instance from Dropbox storage (sender's account) and then uploaded back to Dropbox storage (receiver's account). The time overhead and bandwidth consumption are very close to those in advance file operations such as compression and encryption. When

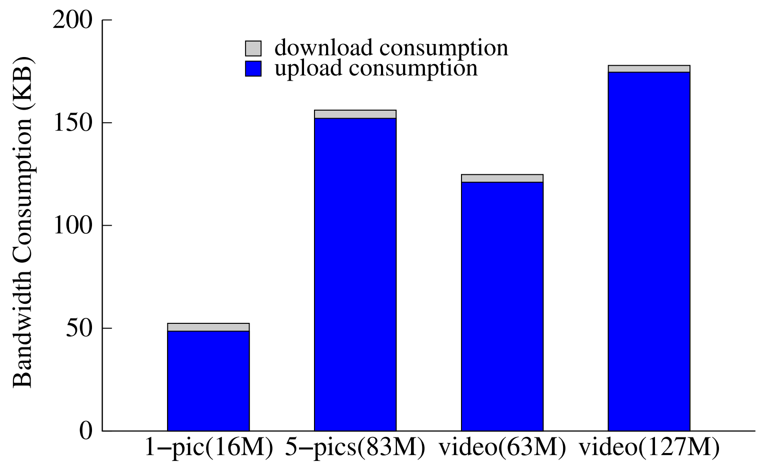


Figure 6.8: Compress file bandwidth cost

using shared instances, the extra costs for communicating with the trusted server is negligible compared to the total performance. The following Table 6.1 lists the time overhead and bandwidth consumption of file transfer (both sender and receiver) with a shared instance. The binary program hosted by the server is 4.9M bytes in our implementation.

	1-Pic(16M)	5-Pics(83M)	Video1	Video2
Download time	8.5s	40.7s	31.4s	59.3s
Upload time	7.3s	25.8s	30.1s	59.8s
Bandwidth	63.4K	245.8K	221.2K	505.9K

Table 6.1: Performance of File Transfer (shared instances)

6.6 Conclusion

This chapter presents Skyfiles which uses cloud instances to help smartphone users execute operations on the files stored in cloud. Skyfiles does not keep local copies of the files and includes an extended set of file operations such as compress and encrypt operations. In addition, Skyfiles supports file transfer between users. Finally, all the file operations in Skyfiles can be optionally executed on shared instances. Our secure protocols protect Skyfiles from disclosing users' security credentials of their cloud storage.

CHAPTER 7

DAB: DYNAMIC AND AGILE BUFFER-CONTROL FOR READL-STREAMING VIDEOS ON MOBILE DEVICES

In this chapter, we target on video prefetch/buffer mechanism that is commonly used in video streaming services. We develop an efficient and dynamic video buffer control scheme that tries to keep a smooth playback with the minimum data delivered to the user by adjusting the buffer size. Our solution especially considers the change of link quality and predicts the trend of change based on the measurement of signal strengths and accelerometer. The solution includes schemes that monitor and analyze the measurements, and an algorithm that decides the buffer size based on the measurements. We have prototyped our solution on Android smartphones and evaluated the performance with experiments. The results show that our solution significantly improves the performance in terms of playback smoothness and buffer efficiency.

7.1 Related Work

This work is related with research on online streaming, video streaming on smartphones and mobile quality of experience. Gill et al. [88] examine usage patterns, file properties, popularity and referencing characteristics, as well as transfer behaviors of YouTube, and compare them to traditional web and media streaming workload characteristics. While Cha et al. [89] first provide an in-depth study of YouTube and other similar User Generated Content(UGC) systems and their user's behaviors. After the study on user's behaviors, Cheng et al. [90] look at YouTube.com and the characteristics of its videos. The authors understand that YouTube has millions of videos and try to point out the problems that it's causing, like network traffic cost per bandwidth. By studying the YouTube system, Krishnappa et al. [91] propose a reordering approach for related list which leads to a 2 to 5 times increase in cache hit rate compared to an approach without reordering the related list. As a result of the increased hit rate, 5.12% to 18.19% reduction is found in server load or back-end bandwidth usage. Adhikari et al. [92] build a measurement infrastructure by using PlanetLab nodes with the goal to understand the YouTube system architecture. Recent work has also been done for other streaming platforms such as Krishnappa et al. [93] for Hulu and Adhikari et al. [94] for Netfilx.

The above related work mainly focuses on measurement of streaming platforms to understand the architecture, user behavior and improvement of the system performance at server side. As the usage of smartphone blooming, more research work has been done about consuming online video on mobile devices. For instance, Finamore et al. [95] compare YouTube traffic generated by mobile devices (smart-phones,tablets) with traffic generated by common PCs (desktops, notebooks, netbooks). The approach Aquarema [96] enables application specific network resource management and thereby improves the user quality of experience. By managing the streaming process, Shen et al. [97] can minimize the sleep and wake penalty of cellular module and at the same time avoid the energy waste from excessive downloading for energy efficient smartphone video playback applications. Staehle et al. [98] present YoMo which can constantly monitor the YouTube application by comforting and detecting the stalling of the video. Ma et al. [99] provide an overview of the current mobile content delivery ecosystem and discuss the expanding role of HTTP-based mobile video delivery. Metzger et al. [100] study video stream buffering and playback control. They claim that choosing the right buffering model can make a huge difference on the quality of the playback process. However, unlike our work, these previous papers are either focus on back-end system architecture improvement, or front-end, PC smartphone, static quality of experience assurance. Our work, on the contrary, takes mobility into consideration, which is an obvious and important feature of smartphone users.

7.2 Dynamic And Agile Buffer Control

In this section, we present our solution DAB, a dynamic and agile buffer control algorithm for mobile devices. The basic idea is to dynamically change the buffer size for prefetching streaming videos according to the link quality and user mobility. The objective is to provide smooth video playback with the minimum bandwidth consumption for downloading video contents.

Essentially, the optimal size of the video buffer on smartphones depends on the video quality and wireless link quality. Video quality indicates the amount of necessary video data for a smooth playback to be prefetched by the player. Apparently, higher quality videos require more data prefetched in the buffer, thus preferring a larger buffer size. Once a video clip is chosen to be player, the quality of the video specifies the minimum requirement of the buffer size, which can be considered as a given parameter for the player. In this chapter, we are focusing on the second factor of wireless link quality while holding the minimum threshold of the buffer size defined by the video quality.

The link quality refers to the network bandwidth capacity for supporting video streaming. We aim to develop an algorithm to accommodate the current link quality by adjusting the buffer size. First of all, all our discussions are for the scenarios where link quality is sufficiently good to support the video rate. The buffer control is ineffective in the application if the link quality is poor because when the video content consumption is faster than the prefetch rate, the buffer will eventually become empty regardless of the buffer size causing a pause of the play. Ideally, the buffer should hold the video data just a little ahead of the current position to guarantee the smooth playback, and then prefetch new contents at the same rate as video rate. For mobile users, however, it is extremely hard to accurately keep an appropriate buffer size because the link quality is highly dynamic due to signal propagation and user mobility. Mobile device may even be temporarily disconnected (e.g., handoff between APs). Intuitively, for a static user, when the link quality is adequate, i.e., the bandwidth is much higher than the video rate, the buffer size can be set to the minimum requirement specified by the video quality. When the bandwidth is close to the video rate, we may need to buffer some more data considering small fluctuations of the link quality. For mobile clients, more importantly, we need to estimate the trend of the link quality. The basic idea is to buffer more data when the link quality is becoming worse and vice versa. In this chapter, we develop a new buffer control scheme DAB (Dynamic and Agile Buffer-control), which considers the current link quality and predicts the trend of change to decide the buffer size. DAB uses the readings of RSSI (Received Signal Strength Indicator) and accelerometer from smartphones to indicate the link quality and user mobility respectively. In the rest of this section, we introduce three basic components of our solution: measurement of RSSIs, measurement of the accelerometer, and the buffer-control mechanism.

7.2.1 Measurement of RSSIs

Most of WiFi devices provide an interface for applications to obtain the values of RSSIs which indicate the signal strengths from the associated AP. This measurement can help represent the current link quality and predicate the trend of change in our solution. Conceptually, RSSI is an effective numeric indicator for the current link quality, i.e., a higher RSSI value indicates a better link quality. In theory, RSSI can also be directly mapped to other metrics regarding the link quality such as signal-noise-ration and bit error rate for a particular modulation. Essentailly, RSSIs at the receiver's side are supposed to be reversely proportional to the distance between the sender and receiver, i.e., a client closer to an AP should have larger values of RSSIs. In practice, however, RSSI values are not precise and quite dynamic. Besides the distance, other factors may also affect the measurement of RSSIs, such as blocking objects, wall reflection, sensitivity of WiFi antenna or nearby interferences. Table 7.1 shows experimental RSSI values measured at the same place. In this experiment, we continuously measure the AP's RSSIs for 100 times with an interval of 500 ms. The values in Table 7.1 are unstable with quite large fluctuations, especially when the client is close to the AP.

Distance	Max(dB)	Min(dB)	Avg(dB)	Std(dB)
3 meters	-40	-61	-51.980	6.635
6 meters	-55	-64	-62.939	3.651
9 meters	-68	-76	-71.384	2.838
12 meters	-74	-81	-76.263	2.368

Table 7.1: RSSI measurements at a given location

In our solution, inspired by [101], we use a *RSSI index* which is defined as a range of RSSI values to represent the RSSI measurement. Our intuition is to mitigate the effect of fluctuation with coarse-grained RSSI measurements. With a span of 10dB, we divide RSSI measurement into 5 intervals with index 1 to 5, *index 1*: [$-\infty$,-80]; *index 2*: [-80,-70]; *index 3*: [-70,-60]; *index 4*: [-60,-50]; *index 5*: [-50, $+\infty$]. In this chapter, RSSI index is used to represent the current link quality as well as the trend of change. We monitor a series of consecutive RSSI index values and estimate the client's moving direction. We assume the client is leaving the associated AP if indexes become larger. Similarly, if the indexes get smaller, we suppose the user is moving towards the AP. More details will be introduced in Section 7.2.3.

7.2.2 Measurement of Accelerometers

Nowadays, accelerometer is a common sensor on smartphones. The accelerometer sensor can continuously measure the acceleration values on the smartphone in the directions of X (lateral), Y (longitudinal) and Z(vertical). Our solution uses the accelerometer readings as another alternative to detect the movement of a client and further estimates his moving speed to help predict the change of link quality. Specifically, we develop two schemes to analyze the accelerometer readings. The first one quickly checks if a user is static or moving. And in case the user is moving, our second scheme, which involves more computations, further derives the moving speed of the user.

Detect whether a User is Moving: To detect whether a user is moving, we only consider the average amplitude of the three directions. The faster the user moves, the greater the average amplitude is. We set a threshold τ for the average amplitude to distinguish if a user is static (sitting or standing) or moving (walking or running). Fig. 7.1 shows the experimental results with ten users. During the experiments, each user holds the smartphone for one minute in two different states, static and moving. From the experiment, when users are moving, the average amplitude of accelerometer is 1.034 with the highest value is 2.44 and the lowest value 0.86. When users are static, the average amplitude is only 0.1 ranging from 0.2 to 0.02. So we set the threshold τ with a heuristic value of 0.35 (m/s^2).

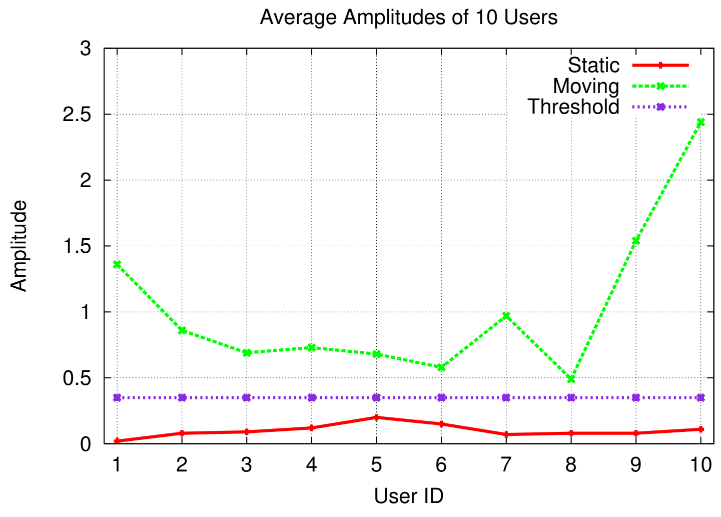


Figure 7.1: Average amplitudes of 10 users for 1 minute

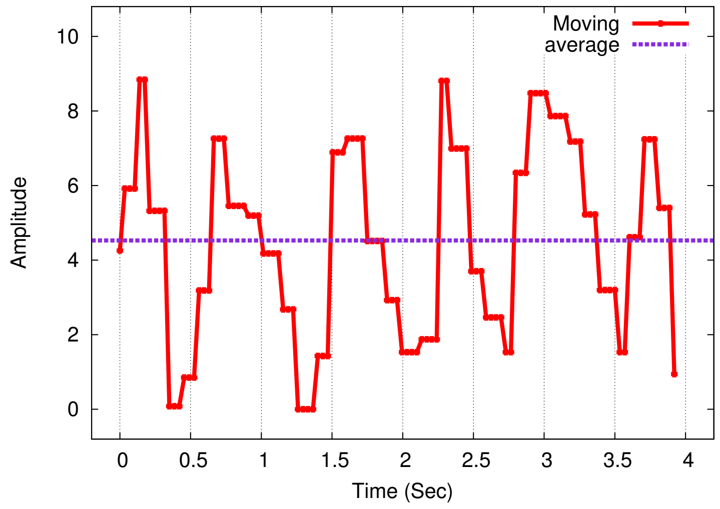


Figure 7.2: Average amplitudes statistics of people moving

Estimate a User's Moving Speed: In order to set an appropriate size of video cache, it's not enough to just predict whether a user is leaving the access point. We also need to estimate how soon it will happen. Thus once our solution detects a user is moving, it will use accelerometer to estimate the user's moving speed. We define five discreet levels to represent the speed (level 0∼4), where level 4 represents the highest speed and level 0 indicates a static user.

Specifically, the moving speed is proportional to the stride frequency and the length of one step. Let S be the moving speed, F be the stride frequency (steps/sec) and L be the length of one step. Then S can be calculated as $S = F \cdot L$. We assume that L is nearly a constant and known as a prior knowledge. Then our focus is to calculate the stride frequency based on the accelerometer measurements. In our approach, We continuously detect the average amplitude $\bar{A}$ of accelerometer every t_m seconds and add the value to a list $\mathcal{A} = \{\bar{A}_1, \bar{A}_2, \bar{A}_3, ..., \bar{A}_N\}$. Obviously, $\bar{A}_1$ is the value of average amplitude at time t_m and $\bar{A}_N$ is the one at time $N \cdot t_m$. Fig 7.2 shows an example of all values in $\mathcal{A}$. The peaks of amplitudes happen on the foot strike and the troughs of amplitudes happen on the foot lifting at the highest position. So the time interval of two peaks or two roughs indicates the time interval for one step, represented as C. For example, assume A_i and A_j represent two consecutive peaks. The time interval of one step should be $C = (j - i) \cdot t_m$. Then, $\frac{1}{(j-i) \cdot t_m}$ should be the stride frequency F. Once we get the stride frequency of a user, we round it up to the nearest whole number as its corresponding speed level. From the experiments, the stride frequency of a user is seldomly larger than 4 steps/sec. So we set four speed levels for moving users in our algorithm, i.e., the level index indicates the number of steps per second.

Algorithm 10 illustrates the details of deriving the speed level (variable SL) of a user. We use avg to represent the average value of all elements in the list $\mathcal{A}$. In line 2, if avg is less than τ, the user is static and the algorithm return 0 as the value of SL. Otherwise, we use $\mathcal{P}$ and $\mathcal{T}$ to represent the sets of indexes of all peaks and troughs respectively. A_h and A_l represents the temporary highest/lowest values in the list $\mathcal{A}$ and I_h and I_l represent the indexes of them, i.e., $A_h = \mathcal{A}[I_h]$ and $A_l = \mathcal{A}[I_l]$. Lines 4 and 5 update the current highest/lowest value and mark its index. In line 6, the algorithm determines that the trend of values is going down. In this case, the current highest value will be set as a peak and the index I_h will be added to $\mathcal{P}$. Similarly, line 7 finds the troughs in the wave and the index of each trough will be added in $\mathcal{T}$. Once we get $\mathcal{P}$ and $\mathcal{T}$, lines 8$\sim$13 are to calculate the average time interval of each two consecutive peaks/troughs. Finally, the algorithm derives the average stride frequency and calculate the user's speed level(SL) in line 14. Fig. 7.3 shows the examples of accelerometer readings that lead to different speed levels.

7.2.3 Dynamic and Agile Buffer-control

In this subsection, we present our Dynamic and Agile Buffer-control (DAB) scheme that combines the measurements of RSSIs and accelerometer and adaptively adjust the buffer size on-the-fly. Comparing RSSI and accelerometer readings, we observe

Algorithm 10 Monitor Users' Moving Speed Level

```
1: Initial: $L_h = L_l = 0, A_h = A_l = avg, \mathcal{P} = \{\}, \mathcal{T} = \{\}, D_p = D_t = 0$
2: if $avg < \tau$ then return 0
3: for $i = 1$ to $N$ do
4:     if $\mathcal{A}[i] > A_h$ then $A_h \leftarrow \mathcal{A}[i], I_h \leftarrow i$
5:     if $\mathcal{A}[i] < A_l$ then $A_l \leftarrow \mathcal{A}[i], I_l \leftarrow i$
6:     if $(\mathcal{A}[i-1] > avg)$ and $(\mathcal{A}[i] < avg)$ then $\mathcal{P} \leftarrow \mathcal{P} + \{I_h\}, A_h \leftarrow avg$
7:     if $(\mathcal{A}[i-1] < avg)$ and $(\mathcal{A}[i] > avg)$ then $\mathcal{T} \leftarrow \mathcal{T} + \{I_l\}, A_l \leftarrow avg$
8: for $m = 1$ to $|\mathcal{P}| - 1$ do
9:     $D_p += (\mathcal{P}[m+1] - \mathcal{P}[m]) \cdot t_m$
10: $\bar{C}_p = \frac{D_p}{|\mathcal{P}|-1}$
11: for $n = 1$ to $|\mathcal{T}| - 1$ do
12:     $D_t += (\mathcal{T}[n+1] - \mathcal{T}[n]) \cdot t_m$
13: $\bar{C}_t = \frac{D_t}{|\mathcal{T}|-1}$
14: $F = \frac{2}{\bar{C}_p + \bar{C}_t}$,    $SL = min(\lceil F \rceil, 4)$,    return $SL$
```

the following differences. First, RSSI values are more fluctuating. As we have shown in Section 7.2.1, there could be a large variance among the RSSIs measured at the same location. Accelerometer measurements, on the other hand, are more consistent and can accurately reflect a user's movement. In addition, accelerometer readings are barely affected by environmental factors. Second, accelerometer values can be monitored continuously while RSSIs are usually measured at discrete time points. As an internal sensor, accelerometer can be accessed any time by any program. Our solution still periodically checks the accelerometer values, but the the interval between two consecutive readings is set to be very small. RSSIs, however, are only available upon the arrivals of packets such as data packets or beacon messages from the APs. Thus RSSIs are not instantly available when the program needs it and the interval between two consecutive RSSI measurements is larger than the interval that we can set for accelerometer readings, e.g., beacon messages from an AP are broadcast with an interval of 100ms. The third difference between RSSI and accelerometer measurements is that RSSIs more directly reflect the link quality while accelerometer readings only indirectly indicate the link change. RSSIs are the measurement of wireless signals. Despite of unavoidable inaccuracy due to environmental factors and hardware

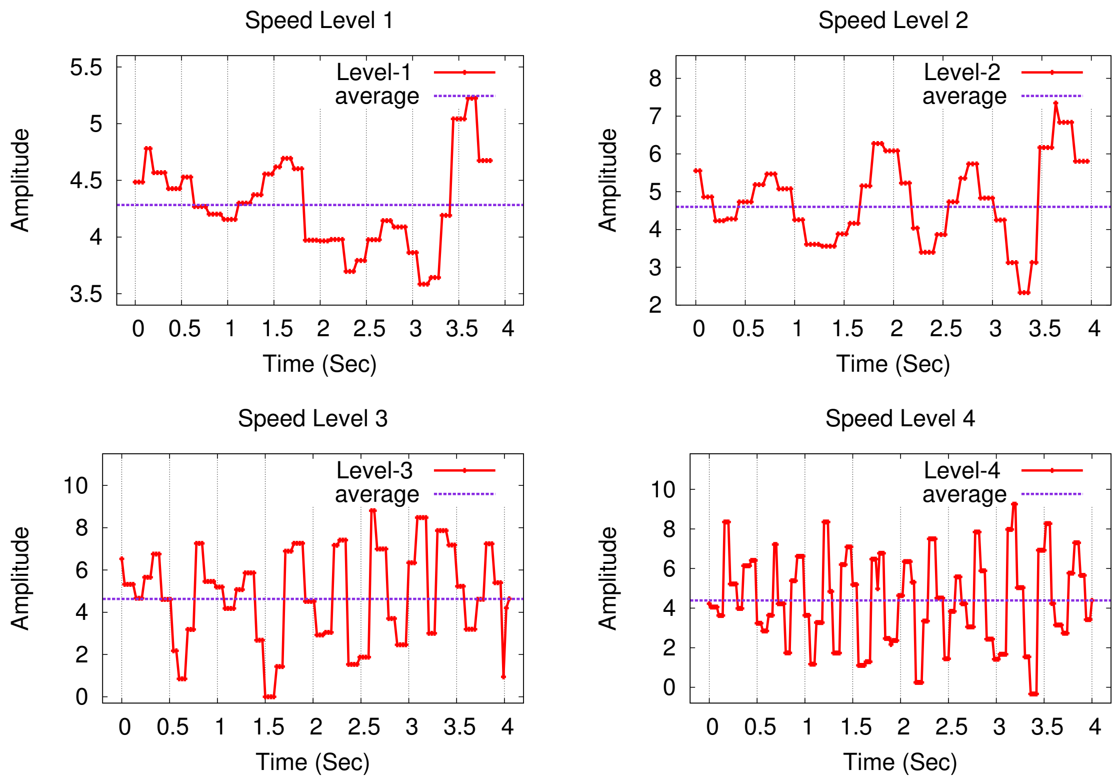


Figure 7.3: Accelerometer statistics in different speed levels

sensitivity, RSSIs at a user's side represent the quality of the downlink (from the AP to the user). Conceptually, accelerometer only measures the movement of a user which is not necessarily linked to the quality of wireless signal reception. For example, a user may move back and forth around a particular location, in which case the accelerometer readings are high, but the link quality probably remain the same.

Considering the above characteristics of RSSI and accelerometer measurements, we develop an algorithm that predict the change of link quality and dynamically adjust the buffer size. Specifically, our main idea is to continuously monitor the accelerometer values (Section 7.2.2) and once a movement is detected, we will further measure the RSSIs (Section 7.2.1). The new buffer size is decided based on both measurements.

Algorithm 11 Dynamic and Agile Buffer-control

```
 1: Initial: $B = B_{min}, \mathcal{R} = \{\ \}$
 2: if $S > 0$ then
 3:     for $i = 1$ to $m$ do
 4:         Measure the RSSI value and store it in $\mathcal{R}[i]$
 5:     $u = \text{argmin}_{i \in [1,m]} \mathcal{R}[i], \quad v = \text{argmax}_{i \in [1,m]} \mathcal{R}[i]$
 6:     if $u > v$ then
 7:         $B = B \times S^{-\alpha} \times \beta^{-(\mathcal{R}[v]-\mathcal{R}[u]) \cdot \mathcal{R}[u]}$
 8:     else if $v > u$ then
 9:         $B = B \times S^{\alpha} \times \beta^{(\mathcal{R}[v]-\mathcal{R}[u])/\mathcal{R}[v]}$
10: else
11:     if $B_u < 0.9 \times B$ then
12:         $c = c + 1$
13:         if $c \geq W$ then $B = 0.9 \times B, \quad c = 0$
14:     else $c = 0$
```

Algorithm 11 shows the details of our DAB scheme. Let B be the buffer size and given the video quality of a clip to be played, let B_{min} be the minimum buffer size required for a smooth play (as discussed in Section 7.2). We use $\mathcal{R}$ to represent the set of RSSI indexes (Section 7.2.1) and S to indicate the most recent accelerometer-based speed level (Section 7.2.2). Initially, $\mathcal{R}$ is empty and RSSI measurement is disabled. Our algorithm only continuously accesses the accelerometer and derive the value of S. Once $S > 0$ (line 2), which indicates a user starts moving, our solution will start to monitor the RSSIs and collect m measurements. The derive RSSI index values are stored in $\mathcal{R}$. In line 5, the algorithm finds the minimum and maximum values in $\mathcal{R}$ and stores their indexes in the variable u and v respectively. When $u > v$, we suppose the link quality is getting better; otherwise ($u < v$), the link quality is degrading. In lines 7 and 9, our algorithm adjusts the buffer size correspondingly. We use two parameters $\alpha \in (0, 1)$ and $\beta > 1$ to define two heuristic functions to change the buffer size. Intuitively, when the link quality is improving (line 7), we should reduce the buffer size towards B_{min}. The reduced amount of buffer size should be proportional to the moving speed (S) and the increase of the RSSI ($\mathcal{R}[v] - \mathcal{R}[u]$). In another work, the faster a user moves or the more improvement is on RSSI index, the smaller the resulting buffer size is. Similarly, when the link quality gets worse, our algorithm increases the buffer size considering the moving speed and the gap between the best and worst RSSI indexes. The difference on the exponents of β in line 7 and line 9 is for another design intuition that slowly increases the buffer size and quickly decreases it in the appropriate circumstances. Finally, we also consider the actual usage of the buffer during the video play. Let B_u be the size of the actual data stored in the buffer, apparently $B_u \leq B$. If B_u keeps smaller than B, which implies that the utilization of B is not full, our algorithm will reduce the buffer size B. In Algorithm 11, we use 0.9 as a threshold for checking the utilization of the buffer. We use a count variable c to record the number of consecutive occurrences of $B_u < 0.9 \cdot B$. When the value of c exceeds another threshold W, the algorithm will adaptively shrink the buffer size to be $0.9 \cdot B$ (line 13).

7.3 Implementation And Evaluation

In this section, we first introduce the system implementation of our solution and then present the performance evaluation results based on smartphone experiments.

7.3.1 System Implementation And Experiment Setup

We implement our solution DAB on Android platform and deploy it on an assorted set of phones with different manufactures including LG, ASUS, and Samsung. Specifically, in the implementation, we use vitamio [102], an open multimedia framework for android, as our base development platform. Youtube is the video service provider in our experiments. Additionally, to better evaluate our solution, we implement two commonly used buffer mechanisms: Flip-Flop(FF) and Maximum(Max), for comparison. With Flip-Flop mechanism, the video player uses a preset value as the fixed buffer size, such as 10 seconds or 2MB. Whenever the buffer is full, it stops video prefetching. In Maximum buffer mechanism, the player simply keeps downloading data until all the video data has been buffered.

Finally, in order to thoroughly test our application in different network conditions, we attach the android phones to a router running DD-WRT [86], a Linux-based firmware. On DD-WRT, we use tc(traffic control) command to limit the maximum connection speed on certain device. To simulate the mobility, the users continuously move around within the router's communication range, backward and forward.

7.3.2 Performance Evaluation

To measure the smoothness of the playback, the performance metric of our application is defined as *total stalling duration equation*

$$\Delta_t = card\{ \ i \mid i \in [1, n], \ C_{BS_i} - C_{PS_i} = 0 \ \} \times t_i \tag{7.1}$$

where C_{BS_i} and C_{PS_i} are the i-th current-buffered size and current-played size. When $C_{BS_i} - C_{PS_i} = 0$, the playback is stalling. The application records those values with an interval of 500ms which is represent by t_i in the above equation. By multiplying the interval and the number of recorded stalling cases, we get the total stalling duration which is denoted by Δ_t.

Another objective of our solution is to efficiently use the buffered bytes so that a user can save bandwidth and server can reduce the traffic load. To quantify this target, we define the two parameters E_i and R_i in Eq. 7.2 and Eq. 7.3, where T_i is the recording timestamp of the i-th record, E_i is the buffer efficiency at T_i and R_i is the average buffer redundancy rate from 0 to T_i which indicates the extra bandwidth cost on this video and, obviously, the lower the better it is.

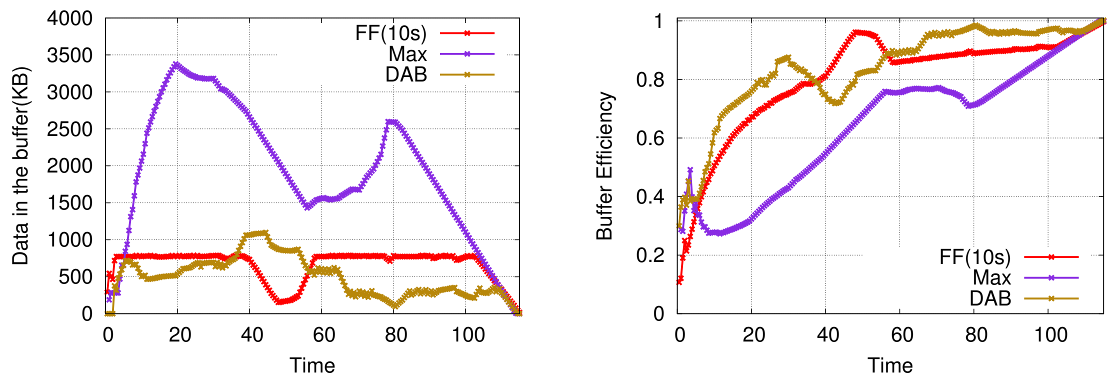
$$E_i = \begin{cases} \frac{C_{BS_i}}{C_{PS_i}} \ , & if \ C_{BS_i} \neq C_{PS_i} \\ 0 \ , & otherwise \end{cases} \tag{7.2}$$

$$R_i = \frac{\sum_{i \in [1,n]} (C_{BS_i} - C_{PS_i})}{T_i} \tag{7.3}$$

In this subsection, we present the evaluation result of our proposed solution. For Algorithm 11, we set $\alpha = 0.3$ and $\beta = 1.1$. For the compared FF method, we set the buffer size as 10 seconds. In our experiments, we use a 115-second Youtube video clip which is 9.38MB as the source. In each test of the three buffer mechanisms, users follow a similar mobility pattern(moving towards and forwards). Particularly, in our proposed solution we maintain a window of 100 consecutive accelerometer readings with an interval of 10ms and if a user movement is detected, we will start to record 5 RSSI readings with an interval of 500ms.

We first test our application under a good connection where the bandwidth on the router is 6MB from service provider. Fig. 7.4a plots the values of $C_{BS_i} - C_{PS_i}$ which indicate Δ_t by the number of zeros on the x-axis. From the figure, we can observe that all three buffer mechanisms perform well with no stalling ($\Delta_t = 0$).

Fig. 7.4b compares the buffer efficiency E_i. We consider the commonly observed user behavior that a user often stops playing a video clip before it ends. The axis x indicates the time when a user stops the video. As we can observe from Fig. 7.4b, if a user stop watching at the very beginning of the video, E_i of DAB, FF and Max mechanisms are all below 50%. For example, at the 10-th second, the DAB, FF and Max's E_{10} values are 63.8%, 49.5% and 27.8%. From 0-th to 38-th second, DAB outperforms FF and Max. From time 29-th to 43-th second, the efficiency of DAB



(a) Buffered Data Size **(b)** Buffer Efficiency

Figure 7.4: Experiments with a good connection (6MB wired bandwidth on the router)

decreases because during this period, the algorithm detects that the user is about to leave the transmission range, and then it allows the player to increase the buffer size in order to prefetch more data. Furthermore, the redundancy rate at the 10th second for DAB, FF and Max, R_{10} are 50.7Kb/s, 87.3Kb/s and 206.4Kb/s, respectively. Our solution reduces 33.4% redundancy rate during these 10 seconds. Therefore, in this scenario with a good link connection, our solution DAB can deliver perfect quality of service to user($\Delta_t = 0$), at the meantime, we achieve high efficiency E at beginning of the video which can help user save bandwidth and reduce traffic load at the server side.

Since most users suffer from slow network connection, we also test our application with a maximum downloading speed 200Kb/s. Fig 7.5 presents the result of this scenario. Clearly, from the figure, FF and Max suffer from stalling during video playback. The total stalling duration (Δ_t) for DAB, FF and Max in this scenario are, 4, 12, and 3.5 seconds, respectively. Particularly, FF mechanism stalls 5 times, for 5.5s, 1s, 0.5s, 2s and 3s each time. While DAB and Max mechanism only stalls once for 4s, 3.5s. In this case, our DAB solution and Max perform similarly and much better than FF mechanism. However, considering the average redundancy rate from time 0 to 25, comparing to Max mechanism, our solution DAB reduces it by 59.7%.

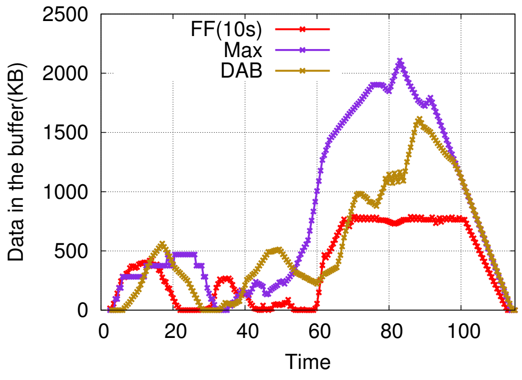


Figure 7.5: Connected to 200Kb/s downloading speed

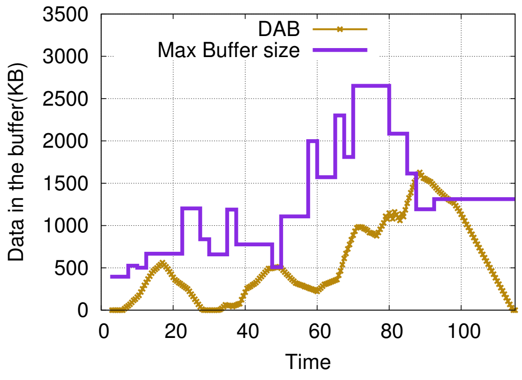


Figure 7.6: Buffer size trend change during the movement

Fig 7.6 shows the buffer size trend of our proposed DAB along with the calculated maximum buffer size during the 115-second test. As we can see from the figure, at the beginning, DAB successfully detects the user's connection conditions and movements. The adjusted buffer size does not limit the actual bytes in the buffer. However, from 80-th to 100-th second, the calculated buffer size helps the user reduce buffer size and improve the efficiency.

7.4 Conclusion

This chapter presents DAB, a dynamic buffer-control scheme that adaptively adjusts the video buffer size based on the measurement of RSSI and accelerometer. Our solution predicts the change of link quality and correspondingly changes the buffer size to help maintain a smooth playback while minimizing the bandwidth consumption. The experimental results have shown that our solution is superior to typical buffer schemes.

CHAPTER 8

CONCLUSION

8.1 Dissertation Summary

This dissertation studies improving services on mobile devices. We consider three representative categories of mobile services: location-based services, cloud-storage services and real-time video streaming services.

The first half of the dissertation focuses on the location-based services. It provides complementary alternatives to the traditional infrastructure-based network architecture. Based on the idea of Ah-Hoc networks, the proposed communication models improve the data transmission in proximity. Specifically, Chapter 2 presents LAAR that utilizes an additional hardware, long-range radio interface on mobile devices, to help construct, recover and maintain the paths in the Ad-Hoc networks. In particular, we propose bi-direction and RSSI-guided route request flooding techniques to reduce the overhead and broadcast messages in the path establishment phase. In addition, we introduce partial and proactive path recovery techniques to accelerate the link break recovery process. Moreover, LAAR improves the efficiency of route cache management by removing invalid paths in the cache based on the overheard packets by the long range radio. We demonstrate and evaluate the LAAR through a location-based flyer distribution application on smartphones. Next, Chapter 3 presents PASA that aims at local message dissemination. Based on the existing communication modules, such as Bluetooth and WiFi-Direct, PASA uses their “peer discovery” functions to scan the nearby users and utilizes their “device name” fields to carry the information, such as Twitter or Facebook statuses. We provides a detailed analysis, including theorems and proofs, on the message reception probability for parameter optimization. Additionally, a two-stage protocol guarantees the information to be received by every user in the system. Based on PASA communication model, a mobile message board application has been proposed in Chapter 4 for smartphone users to post and share information within a certain area. We develop algorithms to choose the best message host combinations and maximize the activation rate for each messages.

The second half of the dissertation investigates the other two representative mobile services, cloud-storage services and real-time video streaming services. Skyfiles is introduced in Chapter 5, which focuses on enriching the file operations for cloud-storage services, such as Dropbox and iCloud. It utilizes a cloud instance to assist the advance operations on the files. The direct interactions between cloud instance and cloud-storage service providers significantly reduce the amount of data transmissions on mobile users. To improve real-time video streaming services, we present DAB in Chapter 6. The DAB application takes mobility and signal strength into consideration and dynamically adjusts the buffer size to avoid unnecessary bandwidth cost.

8.2 Future Work

The research in this dissertation can be extended in several directions:

- In Chapter 2, we present LAAR, which introduces a new long-range radio to smartphones. Through taking advantages of the new hardware, LAAR dramatically boosts the system performance of the Ad-Hoc networks. It would be interesting to adopt the system design to the drone networks, which is another emerging research area with open challenges.

- In Chapter 3 and 4, we present PASA, a connectionless communication model for local message dissemination. Currently, both PASA communication model and its application MMB only consider one-hop networks where every user can communicate with others directly. The performance of PASA in a multi-hop network could be a natural extension of this research project.

- Chapter 5 focuses on the cloud-storage services. Skyfies application uses a cloud instance to provide advance file operations. However, cloud instance has more capabilities than file operations. It can be extended to empower mobile devices by offloading computational-intensive tasks to cloud instance, such as picture stitching tasks for virtual reality and augmented reality.
- DAB is presented in Chapter 6 to dynamically adjust the buffer size for real-time video streaming services. The evaluation in this work only focuses on the user side. However, the design of DAB can also benefit the server side. It could be another direction to investigate the impact of the buffer mechanisms at the server side and develop the algorithms to reduce the server load.

BIBLIOGRAPHY

[1] Foursquare. http://www.foursquare.com.

[2] Facebook Places. http://www.facebook.com.

[3] Yelp. http://www.yelp.com.

[4] Waze. http://www.waze.com.

[5] SCVNGR. http://www.scvngr.com.

[6] UBER. http://www.uber.com.

[7] Anil Acharya, Yantian Hou, Ying Mao, and Jiawei Yuan. Edge-assisted image processing with joint optimization of responding and placement strategy. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1241–1248. IEEE, 2019.

[8] Anil Acharya, Yantian Hou, Ying Mao, Min Xian, and Jiawei Yuan. Workload-aware task placement in edge-assisted human re-identification. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2019.

[9] Hank H Harvey, Ying Mao, Yantian Hou, and Bo Sheng. Edos: Edge assisted offloading system for mobile devices. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2017.

[10] Yuqi Fu, Shaolun Zhang, Jose Terrero, Ying Mao, Guangya Liu, Sheng Li, and Dingwen Tao. Progress-based container scheduling for short-lived applications in a kubernetes cluster. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 278–287. IEEE, 2019.

[11] Wenjia Zheng, Yun Song, Zihao Guo, Yongchen Cui, Suwen Gu, Ying Mao, and Long Cheng. Target-based resource allocation for deep learning applications in a multi-tenancy system. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2019.

[12] Wenjia Zheng, Michael Tynes, Henry Gorelick, Ying Mao, Long Cheng, and Yantian Hou. Flowcon: Elastic flow configuration for containerized deep learning applications. In *Proceedings of the 48th International Conference on Parallel Processing*, pages 1–10, 2019.

[13] Ying Mao, Victoria Green, Jiayin Wang, Haoyi Xiong, and Zhishan Guo. Dress: Dynamic resource-reservation scheme for congested data-intensive computing platforms. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 694–701. IEEE, 2018.

[14] Jiayin Wang, Teng Wang, Zhengyu Yang, Ying Mao, Ningfang Mi, and Bo Sheng. Seina: A stealthy and effective internal attack in hadoop systems. In *2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 525–530. IEEE, 2017.

[15] Jiayin Wang, Yi Yao, Ying Mao, Bo Sheng, and Ningfang Mi. Fresh: Fair and efficient slot configuration and scheduling for hadoop clusters. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 761–768. IEEE, 2014.

[16] Ying Mao, Jenna Oak, Anthony Pompili, Daniel Beer, Tao Han, and Peizhao Hu. Draps: Dynamic and resource-aware placement scheme for docker containers in a heterogeneous cluster. In *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2017.

[17] Ying Mao, Bo Sheng, and Mooi Choo Chuah. Scalable keyword-based data retrievals in future content-centric networks. In *2012 8th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pages 116–123. IEEE, 2012.

[18] iCloud. `https://www.icloud.com/`.

[19] Dropbox. `http://www.dropbox.com`.

[20] Box. `http://www.box.com`.

[21] Google Drive. `https://drive.google.com/`.

[22] Microsoft SkyDrive. `https://skydrive.live.com/`.

[23] Ubuntu One. `https://one.ubuntu.com/`.

[24] Ying Mao, Jiayin Wang, Bo Sheng, and Mooi Choo Chuah. Laar: Long-range radio assisted ad-hoc routing in manets. In *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, pages 350–355. IEEE, 2014.

[25] Ying Mao, Jiayin Wang, and Bo Sheng. Building smartphone ad-hoc networks with long-range radios. 2015.

[26] Ying Mao, Jiayin Wang, Joseph Paul Cohen, and Bo Sheng. Pasa: Passive broadcast for smartphone ad-hoc networks. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, pages 1–8. IEEE, 2014.

[27] Ying Mao, Jiayin Wang, and Bo Sheng. Mobile message board: Location-based message dissemination in wireless ad-hoc networks. In *2016 International Conference on Computing, Networking and Communications (ICNC)*, pages 1–5. IEEE, 2016.

[28] Ying Mao, Jiayin Wang, and Bo Sheng. Skyfiles: Efficient and secure cloud-assisted file management for mobile devices. In *Communications (ICC), 2014 IEEE International Conference on*, pages 4202–4207. IEEE, 2014.

[29] Ying Mao, Jiayin Wang, and Bo Sheng. Dab: Dynamic and agile buffer-control for streaming videos on mobile devices. *Procedia Computer Science*, 34:384–391, 2014.

[30] Xe1205. `http://www.semtech.com/images/datasheet/xe1205.pdf/`.

[31] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, SIGCOMM '94, pages 234–244, 1994.

[32] Olsr. `http://www.ietf.org/rfc/rfc3626.txt`.

[33] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181, 1996.

[34] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *THE 2ND IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS*, pages 90–100, 1997.

[35] Richard Draves, Jitendra Padhye, and Brian Zill. Comparison of routing metrics for static multi-hop wireless networks. *SIGCOMM Comput. Commun. Rev.*, 34(4):133–144, August 2004.

[36] Hannes Frey. Scalable geographic routing algorithms for wireless ad hoc networks. *Network, IEEE*, 18(4):18–22, 2004.

[37] Pradeep Kyasanur and Nitin H. Vaidya. Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 10(1):31–43, January 2006.

[38] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom '04, pages 114–128, 2004.

[39] Asad Amir Pirzada, Ryan Wishart, and Marius Portmann. Multi-linked aodv routing protocol for wireless mesh networks. In *GLOBECOM*, pages 4925–4930. IEEE, 2007.

[40] Saad Biaz, Bing Qi, Shaoen Wu, and Yiming Ji. In *Evaluation of Multi-Radio Extensions to DSR for Wireless Multi-Hop Networks*, pages 65–69.

[41] Y Cheng-Ren et al. Configuring cloud-integrated body sensor networks with evolutionary algorithms. In *Proceedings of the 9th International Conference on Body Area Networks*, BodyNets '14, pages 271–278, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[42] Yi Cheng-Ren et al. Leveraging evolutionary multiobjective games for configuring cloud-integrated body sensor networks. In *Soft Computing and Intelligent Systems (SCIS), 15th International Symposium on*, pages 630–636, Dec 2014.

[43] Tinynode-584. `http://tinynode.com/?q=product/tinynode584/tn-584-868`.

[44] Pl2303. http://www.prolific.com.tw/US/newsdetail.aspx?news_id=29.

[45] Tun/tap. http://en.wikipedia.org/wiki/TUN/TAP.

[46] Network simulator 2. http://www.isi.edu/nsnam/ns/.

[47] Woonkang Heo and Minseok Oh. In *FGCN (2)*, pages 128–132.

[48] Manhattan mobility model. http://en.wikipedia.org/wiki/Manhattan_mobility_model.

[49] Chiara Boldrini, Marco Conti, and Andrea Passarella. Contentplace: social-aware data dissemination in opportunistic networks. In *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '08, 2008.

[50] Jialu Fan, Jiming Chen, Yuan Du, Wei Gao, Jie Wu, and Youxian Sun. Geocommunity-based broadcasting for data dissemination in mobile social networks. *IEEE Trans. Parallel Distrib. Syst.*, 24(4), April 2013.

[51] Bahadir K. Polat, Pushkar Sachdeva, Mostafa H. Ammar, and Ellen W. Zegura. Message ferries as generalized dominating sets in intermittently connected mobile networks. In *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp '10, pages 22–31, 2010.

[52] Chunyi Peng, Guobin Shen, Yongguang Zhang, and Songwu Lu. Point&Connect: intention-based device pairing for mobile phone users. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, 2009.

[53] Ben Dodson, Ian Vo, T.J. Purtell, Aemon Cannon, and Monica Lam. Musubi: disintermediated interactive social feeds for mobile devices. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 211–220, 2012.

[54] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11), November 2011.

[55] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Trans. Netw.*, 14(3):479–491, 2006.

[56] Arezu Moghadam, Suman Srinivasan, and Henning Schulzrinne. 7ds - a modular platform to develop mobile disruption-tolerant applications. In *Proceedings of the 2nd International Conference on Next Generation Mobile Applications, Services and Technologies*, 2008.

[57] Yang Zhang and Jing Zhao. Social network analysis on data diffusion in delay tolerant networks. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '09, pages 345–346, New York, NY, USA, 2009. ACM.

[58] Wei Gao, Qinghua Li, Bo Zhao, and Guohong Cao. Multicasting in delay tolerant networks: a social network perspective. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '09, pages 299–308, 2009.

[59] Mooi Choo Chuah. Social network aided multicast delivery scheme for human contact-based networks. In *Proceedings of the 1st Annual Workshop on Simplifying Complex Network for Practitioners*, SIMPLEX '09, 2009.

[60] Josep Díaz, Alberto Marchetti-Spaccamela, Dieter Mitsche, Paolo Santi, and Julinda Stefa. Social-aware forwarding improves routing performance in pocket switched networks. In *Proceedings of the 19th European conference on Algorithms*, ESA'11, pages 723–735, Berlin, Heidelberg, 2011. Springer-Verlag.

[61] Elizabeth M. Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, 2007.

[62] S. Jakubczak. Dythr. http://szym.net/dythr/.

[63] Te-Yuan Huang, Kok-Kiong Yap, Ben Dodson, Monica S. Lam, and Nick McKeown. PhoneNet: a phone-to-phone network for group communication within an administrative domain. In *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*, MobiHeld '10, 2010.

[64] Adam C. Champion, Zhimin Yang, Boying Zhang, Jiangpeng Dai, Dong Xuan, and Du Li. E-smalltalker: A distributed mobile system for social networking in physical proximity. *IEEE Trans. Parallel Distrib. Syst.*, 24(8), August 2013.

[65] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *14th ACM MOBICOM*, San Francisco, CA, September 2008.

[66] AirDefense. http://www.airdefense.net/.

[67] AirWave Management. http://www.airwave.com/.

[68] CiscoWorks Wireless LAN Solution Engine. http://www.cisco.com/c/en/us/products/cloud-systems-management/ciscoworks-wireless-lan-solution-engine-wlse/index.html.

[69] AirMagnet. http://www.airmagnet.com/.

[70] NetStumbler. http://www.NetStumbler.com/.

[71] Crawdad Dartmouth. http://crawdad.org/index.html.

[72] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, and Christophe Diot. Mobiclique: Middleware for mobile social networking. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, pages 49–54, New York, NY, USA, 2009. ACM.

[73] Amazon Web Service. `http://aws.amazon.com/`.

[74] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. A survey of computation offloading for mobile systems. *MONET*, 18(1):129–140, 2013.

[75] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 49–62, 2010.

[76] Roelof Kemp, Nicholas Palmer, Thilo Kielmann, and Henri E. Bal. Cuckoo: A computation offloading framework for smartphones. In *MobiCASE*, pages 59–79, 2010.

[77] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *INFOCOM*, pages 945–953, 2012.

[78] Aki Saarinen, Matti Siekkinen, Yu Xiao, Jukka K. Nurminen, Matti Kemppainen, and Pan Hui. Smartdiet: offloading popular apps to save energy. In *SIGCOMM*, pages 297–298, 2012.

[79] Idilio Drago, Marco Mellia, Maurizio M. Munafo, Anna Sperotto, Ramin Sadre, and Aiko Pras. Inside dropbox: understanding personal cloud storage services. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, IMC '12, pages 481–494, 2012.

[80] Haiyang Wang, Ryan Shea, Feng Wang, and Jiangchuan Liu. On the impact of virtualization on dropbox-like cloud file storage/synchronization services. In *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*, IWQoS '12, pages 11:1–11:9, 2012.

[81] Marco Valerio Barbera, Sokol Kosta, Julinda Stefa, Pan Hui, and Alessandro Mei. Cloudshield: Efficient anti-malware smartphone patching with a p2p network on the cloud. In *P2P*, pages 50–56, 2012.

[82] Sascha Fahl, Marian Harbach, Thomas Muders, and Matthew Smith. Confidentiality as a service – usable security for the cloud. In *Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, TRUSTCOM '12, pages 153–162, Washington, DC, USA, 2012. IEEE Computer Society.

[83] OAuth v1.0. `http://oauth.net/core/1.0a/`.

[84] Microsoft Azure. `http://www.windowsazure.com`.

[85] HP Cloud. `https://www.openssl.org/`.

[86] Dd-wrt. http://www.dd-wrt.com/.

[87] TCPDUMP. `http://www.tcpdump.org/`.

[88] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 15–28, New York, NY, USA, 2007. ACM.

[89] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 1–14, New York, NY, USA, 2007. ACM.

[90] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Understanding the characteristics of internet short video sharing: Youtube as a case study. *CoRR*, abs/0707.3670, 2007.

[91] Dilip Kumar Krishnappa, Michael Zink, Carsten Griwodz, and Pål Halvorsen. Cache-centric video recommendation: an approach to improve the efficiency of youtube caches. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, pages 261–270, New York, NY, USA, 2013. ACM.

[92] Vijay Kumar Adhikari, Sourabh Jain, Yingying Chen, and Zhi-Li Zhang. Vivisecting youtube: An active measurement study. In *INFOCOM*, pages 2521–2525, 2012.

[93] Dilip Kumar Krishnappa, Samamon Khemmarat, Lixin Gao, and Michael Zink. On the feasibility of prefetching and caching for online tv services: a measurement study on hulu. In *Proceedings of the 12th international conference on Passive and active measurement*, PAM'11, pages 72–80, Berlin, Heidelberg, 2011. Springer-Verlag.

[94] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *INFOCOM*, pages 1620–1628, 2012.

[95] Alessandro Finamore, Marco Mellia, Maurizio M. Munafò, Ruben Torres, and Sanjay G. Rao. Youtube everywhere: impact of device and infrastructure synergies on user experience. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 345–360, New York, NY, USA, 2011. ACM.

[96] Barbara Staehle, Matthias Hirth, Rastin Pries, Florian Wamser, and Dirk Staehle. Aquarema in action: Improving the youtube qoe in wireless mesh networks. In *Baltic Congress on Future Internet Communications (BCFIC)*, 2 2011.

[97] Hao Shen and Qinru Qiu. User-aware energy efficient streaming strategy for smartphone based video playback applications. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, pages 258–261, San Jose, CA, USA, 2013. EDA Consortium.

[98] Barbara Staehle, Matthias Hirth, Florian Wamser, Rastin Pries, and Dirk Staehle. Yomo: A youtube application comfort monitoring tool. Number 467, 3 2010.

[99] K.J. Ma, R. Bartos, S. Bhatia, and R. Nair. Mobile video delivery with http. *IEEE Communications Magazine*, 49:166–175, April 2011.

[100] Florian Metzger, Albert Rafetseder, David Stezenbach, and Kurt Tutschku. Analysis of web-based video delivery. In *FITCE International Congress*, Palermo, 2011.

[101] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. *SIGCOMM Comput. Commun. Rev.*, 40(4):159–170, August 2010.

[102] Vitamio framework. http://www.vitamio.org/en/.