Supporting Heterogeneous Networks and Pervasive Storage in Mobile Content-Sharing Middleware

Daniel J. Dubois[1,3], Yosuke Bando[2,1], Konosuke Watanabe[2,1], Arata Miyamoto[2,1],
Munehiko Sato[1], William Papper[4,1], V. Michael Bove, Jr.[1]
[1]MIT Media Lab, Cambridge, MA, USA; [2]Toshiba Corporation, Tokyo, Japan;
[3]Imperial College London, London, UK; [4]Washington University in St. Louis, MO, USA
ddubois@media.mit.edu, yosuke1.bando@toshiba.co.jp, konosuke.watanabe@toshiba.co.jp, arata.miyamoto@toshiba.co.jp,
munehiko@media.mit.edu, wpapper@wustl.edu, vmb@media.mit.edu

***Abstract*—Sharing digital content with others is now an important part of human social activities. Despite the increasing need to share, most sharing operations are not simple. Many applications are not interoperable with others, require an Internet connection, or require cumbersome configuration and coordination efforts. Our idea is to simplify digital content sharing on mobile devices by providing support for self-organizing heterogeneous networks and pervasive storage. That is, mobile devices can spontaneously connect to each other over a mixture of different available networks (e.g., 3G/4G, WiFi, Bluetooth, etc.) without requiring an explicit user action of network selection or mandatory Internet access. Moreover, indirect communication can be further augmented by pervasive storage. Mobile devices can store shared content on it, which can later be automatically downloaded by other devices in proximity, thus allowing location-based sharing with minimal coordination even when devices are not in the same location at the same time. This paper shows how these technologies can be incorporated into mobile content-sharing middleware to simplify sharing operations among mobile devices without any modification to commercially available devices or applications. In particular, (*i*) we provide an implementation of our approach as extension modules for existing content-sharing middleware, (*ii*) we present two example applications built on top of it, and (*iii*) we demonstrate our approach through experiments in representative situations.**

Keywords*—*Content sharing; proximity sharing; heterogeneous networks; pervasive storage; throwboxes; mobile peer-to-peer; opportunistic networks; store-and-forward; delay-tolerant networks.

I. Introduction

Sharing experiences with others is an important part of human social activities, and sharing content using mobile devices is becoming increasingly popular in this digital age. There are many popular means of sharing digital content, ranging from old – but still used – methods such as e-mail attachments, MMS, and Bluetooth file transfer, to new methods such as AirDrop, DropBox, WhatsApp, Viber, and the always growing social network services that provide their own cloud-based sharing services (e.g., Facebook, Google+). However, despite their popularity, these services do not necessarily cover all the sharing needs that a mobile user has. To cite some examples, the sharing capabilities of most popular apps and social networks (e.g., WhatsApp, Viber, Facebook Messenger) require an Internet connection to actually deliver messages (the same also applies to e-mail). Other methods that only rely on proximity such as Bluetooth and AirDrop require an active direct connection to start the sharing process. Both of these limitations are not desirable. For instance, with methods that require Internet access, people who cannot afford data plans and do not live in areas where there is free WiFi will rarely be able to share anything through the Internet in a reasonable amount of time. On the other hand, people that want to use proximity-based methods have to face other problems such as the lack of interoperability, and tedious configurations and coordination. For example, applications using Bluetooth for proximity-based sharing may not be able to talk with others that use only WiFi technologies. Some applications require users to select a device that is currently in range and active, thus not allowing them to choose a device that has been previously seen and that might appear again. Having to wait for the appearance of a user is an example of coordination complexity that affects the ease of use of many proximity-based sharing applications.

The goal of this paper is to address the limitations described above by providing applications with support for *heterogeneous networks* [12] to increase interconnectivity potential, and *pervasive storage* [19] to enhance delayed sending in commercial mobile devices. Heterogeneous networks have been defined in several ways in literature, but in our context we define them as *a class of networks in which different elements may be connected to each other using different communication methods*. An example of such a network is this: let us consider three mobile devices A, B, and C, in which link AB is a WiFi connection, while link BC is a Bluetooth connection. To the best of our knowledge, it is still a complex task to allow A to share something with C without an active user intervention in B to save and re-share it using a different communication method. We address this problem by having a sharing mechanism that is transparent with respect to inhomogeneous networks. As for pervasive storage, we define it as *any network device that can be accessed for reading and writing by nearby mobile devices*. One example of pervasive storage is a WiFi SD card, which provides read/write primitives to nearby devices through a WiFi connection. However, any wirelessly networked device may act as pervasive storage if properly configured, such as a smart TV or a hard drive connected to a capable wireless access point. We use these pervasive storage devices to enable offline sharing among devices in the same location, but not simultaneously. For example, a person can enter a conference room equipped with pervasive storage, and share a file with a colleague that will be in that room later. This works like a digital version of a sticky note, and does

This work has been partially funded by Toshiba Corporation, and by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement n. 629982 (SPANDO). Special thanks to Henry Holtzman, Dhairya Dand, and Eun-young Lim for their assistance with the early stages of the work.

not require any strong assumption such as the availability of a free Internet connection, complex configurations, or human coordination.

The support of heterogeneous networks increases the connectivity potential of content-sharing applications because pieces of information can travel from one device to another in a peer-to-peer fashion independently from the type of available networks. Moreover, data can be shared with devices that are not currently connected directly, since data can be temporarily stored in nearby mobile devices until the intended recipient is in range. Another desirable property of heterogeneous networks is that a pair of devices may be connected using more than one network at the same time; in such a case, it is possible to exchange data using the connection that optimizes the overall communication in the network, for example by minimizing latency or maximizing transfer speed, depending on the policy adopted. If we further introduce support for pervasive storage, we can have an intermediate storage device that can retain shared information until the intended recipient appears. This can be useful in situations in which there is no direct link between a source and a destination, or when an application wants to use a location-based sharing approach rather than destination-based. Scenarios in which this approach is useful include, but are not limited to, the following situations: (*i*) sharing pictures when traveling abroad with friends, while not everyone has cellular roaming or access to WiFi hotspots; (*ii*) sharing in rural areas or in subway cars that have no cellular network; (*iii*) sharing help requests over partially disabled pre-existing networks after a natural disaster; (*iv*) sharing with people in proximity without being socially connected to them (e.g., they are not friends in a social network and their phone number and email address are not known); and (*v*) sharing information to a location rather than to a given set of people, for example at a temporary event like a conference or a concert. An indoor visual example with both heterogeneous networks and pervasive storage technologies employed can be seen in Figure 1.

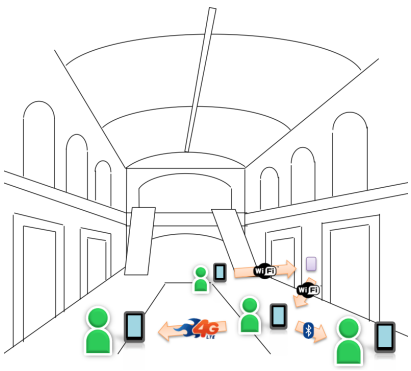


Fig. 1: Content can be shared seamlessly via different communication technologies. It can also be stored on a pervasive storage device (depicted in purple), and re-shared to other nearby devices.

What we deliver in this paper is an approach and an implementation to support the technologies mentioned above on a proximity-based content-sharing platform: we provide new extension modules for existing middleware, new applications, and results of evaluations run on real devices. The paper is organized as follows. In Section II, we discuss related research. Section III describes our approach, and in particular how it was designed and implemented. In Section IV, we show some real example applications and a device that can be used as pervasive memory. Section V demonstrates our approach through experiments in representative situations. Finally, Section VI concludes the paper.

II. Related Work

Content sharing in proximity for mobile devices is a hot topic studied in multiple research areas. In the area of systems and software, academic contributions such as Haggle [16], MobiClique [11], MundoCore[1], and ShAir [4] propose middleware and standards for sharing content in mobile environments. The same concepts are also becoming established in the industry sector in applications such as OpenGarden [10] and AllJoyn [13]. Protocols and algorithms for creating opportunistic networks in proximity include [3], which allow devices to create a temporary ad-hoc network to each other for proximity-based sharing. Another class of approaches, called *infostation paradigm* [15], creates ad-hoc networks by relying on scattered nodes that act as temporary access points. All these works focus on methodologies for sharing resources among different mobile devices with minimal user configuration and interaction. In this paper, we have the same goal, but our focus is on introducing the use of wireless heterogeneous networks and pervasive storage devices to further enhance the sharing capabilities of nearby mobile devices.

Wireless heterogeneous networks in literature have been defined in several different ways, depending on the context. The characterizing element of this class of networks is the intrinsic diversity of its elements, which might be characterized by differences in network access technologies, connectivity, radio-frequency spectra, and so on. These networks are now becoming increasingly popular because they can be used to reduce communication costs, to optimize network capacity, to improve the quality of service, to increase network coverage, and so on [12]. Surveys on this area have been recently published [5, 7], but most of the problems addressed focus on finding handover mechanisms for 4G networks and several other solutions to optimize the infrastructure for providers of broadband services. Beside these successful results, little effort has been made in using the same concepts at a consumer application level. In particular, the possibility to adaptively combine different existing proximity-based communication technologies such as Bluetooth and WiFi to improve the user experience is an area that has not been fully explored yet.

For what concerns the use of pervasive storage for proximity-based communication, a recent research roadmap paper [2] explains the need for such solutions. Existing literature contains some effort in the use of dedicated devices as pervasive storage, which are also known as *throwboxes* [19]. They have been mainly employed to improve context-awareness and the performance of *store-and-forward* networks [18]. In GSTAR [9], the authors propose a solution for improving store-and-forward routing by using pervasive storage as a temporary buffer when the destination of shared content is not currently available. Finally, another class of solutions uses cloud-enabled approaches, where pervasive storage is used as a mirror for cloud storage (e.g., [20]). These existing solutions typically require either that pervasive devices used as storage have some degree of computation capability and configuration, or the presence of an existing infrastructure such as access to cloud computing systems.

In summary, our approach is based on the combination of

existing research on proximity-based content-sharing middleware, the idea of heterogeneous networks, and the adoption of throwbox-style pervasive storage, where our novelty and main contribution lie in providing extended middleware and example applications that exhibit the following three properties at the same time: (*i*) handling proximity-based content-sharing between mobile devices and pervasive storage devices seamlessly on unmodified off-the-shelf devices without mandatory Internet access, (*ii*) exploiting pervasive storage devices as permanent data repositories that can re-share data to new, previously unknown, mobile devices, and (*iii*) combining the use of heterogeneous networks with pervasive storage.

III. Approach and Implementation

Since platforms for mobile data sharing already exist, we provide support for heterogeneous networks and pervasive storage by extending one of the platforms, for which we choose ShAir [4] for its extensibility. Although we use it as an example for simplicity, this approach can be easily adapted and repeated on alternative middleware whose components are separated in a similar way.

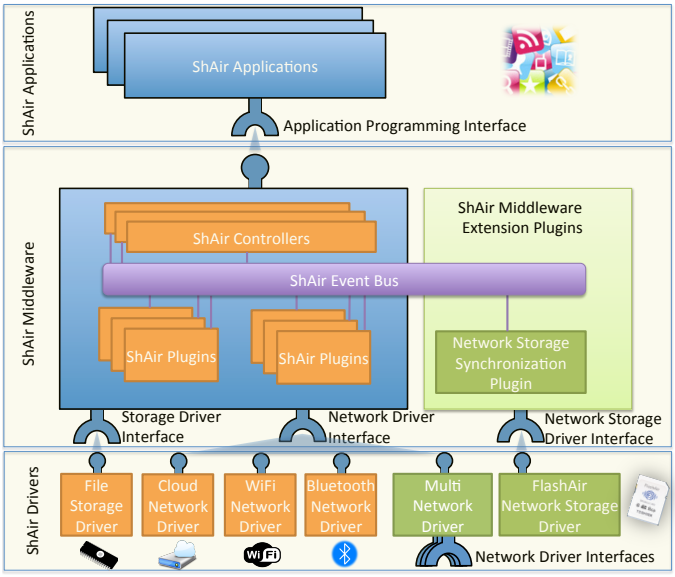


Fig. 2: Extended ShAir architecture. Our extension modules are highlighted in green.

A. Background on Content-sharing Platform ShAir

To show how our approach can be implemented in practice, we give a brief overview of ShAir. ShAir provides an abstraction to application developers for sharing information. Its architecture is simple and composed of three levels, as shown in Figure 2. The first level is the *application level*, which is the part of any ShAir-enabled application that interacts with its underlying middleware through an application programming interface (API). This means that, even if the middleware implementation changes, as long as the API does not change, there will be no need to modify any application. The second level is the *middleware level*. It is composed of: *ShAir controllers*, which provide the actual implementation of the API; *ShAir event bus*, which manages event-based communication among different middleware parts; and *ShAir plugins*, which contain the decision-making logic of ShAir. ShAir plugins make decisions on how to share content and on how to interact with the controllers as well as with *drivers*, which we explain next. The third level is the *driver level*, which provides the middleware with access to device-specific hardware and networking capabilities. This access is provided through *driver interfaces*, which are implemented by device-specific modules called *drivers*. Similarly to the application level, the middleware level does not need to be modified even if drivers change, as long as replacement drivers implement the same driver interface. ShAir comes with two driver interfaces and their respective Android implementations: the *storage driver interface* and the *network driver interface*. The storage driver interface is implemented by the *file storage driver*, which enables access to the Android file system. The network driver interface is used for device-to-device communication and has three implementations: (*i*) *WiFi network driver*, which allows devices to interact with all the devices connected to the same WiFi network without using a network gateway; (*ii*) *Cloud network driver*, which allows devices to interact with all the devices connected to a particular cloud messaging server through any available Internet connection; (*iii*) *Bluetooth network driver*, which allows devices to interact with all the other Bluetooth devices in proximity that have been previously paired[1]. The three green blocks in Figure 2 show the extension modules we have added to support heterogeneous networks and pervasive storage, as we will explain in the following subsections.

B. Heterogeneous Network Extension

The purpose of this extension is to allow the content-sharing platform to support multiple network technologies at the same time. This is done by adding a new driver that abstracts multiple networks as a single network. In the ShAir platform, this is realized by introducing a new network aggregator driver, called *multi-network driver*, which implements the network driver interface and can be instrumented with multiple drivers each of which again implements the network driver interface, as shown in Figure 2. This driver accepts an arbitrary number of existing network drivers together with a policy for making decisions when a device can be reached with more than one connection. As an example policy, we have implemented a *low latency policy*, which constantly estimates the latency of each link using a simple ping-based protocol, and gives higher priority to the one with the lowest latency. This policy can be modified and replaced with more sophisticated ones in the future. The goal of the policy is to self-organize paths of content sharing in such a way that the whole heterogeneous network optimizes a given property (e.g., latency or speed). In our example, the network self-organizes so as to have reduction of latency as an emergent property. Since our ShAir modification consists in only one additional driver, there is no need to modify existing applications that use ShAir, except for the part that instantiates the middleware and its drivers.

C. Pervasive Storage Extension

The purpose of this extension is to permit the use of pervasive storage as a common repository that hosts shared content, and then makes it available to intended destinations later, as soon as they appear. This extension is realized by introducing two new components: (*i*) a middleware plugin responsible for sharing and retrieving data to and from pervasive

[1]We need to pair devices because, for security reasons, interaction with non-paired devices is not currently allowed in commercial devices.

storage devices and (*ii*) a driver for accessing pervasive storage devices. To do this on the ShAir platform, we have added, as shown in Figure 2, (*i*) a new middleware plugin called *network storage synchronization plugin*, which implements a newly-defined *network storage driver interface* to access pervasive storage devices and (*ii*) a new driver called *FlashAir network storage driver*, which implements the network storage driver interface using FlashAir WiFi SD card [17] as a pervasive storage device.

The network storage synchronization plugin is responsible for listening to events in which local content on the mobile device is shared/unshared/updated, and then for executing the appropriate upload, download, or removal operations on pervasive storage. Moreover, the plugin listens to events in which a new nearby pervasive storage device is detected: in such a case, the plugin fetches a list of files on the newly-detected pervasive storage, downloads all the files intended for the mobile device, removes the files from the pervasive storage that have been previously unshared on the mobile device, and finally uploads the files that have been previously shared or updated on the mobile device to the pervasive storage.

The FlashAir network storage driver is an implementation of the network storage driver interface, which is used by the network storage synchronization plugin to interact with pervasive storage devices. It provides primitives to: (*i*) detect pervasive storage devices in proximity; (*ii*) list the files stored in a pervasive storage device; and (*iii*) upload, download, and delete a file to and from a pervasive storage device. We provide an implementation of this driver that uses the commercial FlashAir WiFi SD card as pervasive storage. Since FlashAir works as a WiFi access point and runs a web server, accessing its content can be implemented using HTTP requests [6].

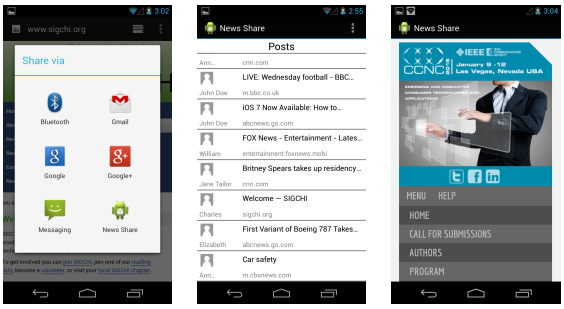
IV. Applications

The support of heterogeneous networks and pervasive devices can provide additional means for interaction and information sharing among mobile device users. The applicability of our approach spans from location-based leisure applications (e.g., photo sharing, news/blog sharing, dating services, etc.) to more serious applications (e.g., distributing critical software updates, broadcasting critical alerts, emergency response, etc.). This section presents two such examples to provide concrete representative usage scenarios, which are made possible with the proposed middleware extensions. We also show an example implementation of pervasive storage.

A. News Article Sharing

People in the same place are likely to have a similar interest in location-specific news. Based on the extended middleware, we implemented an application shown in Figure 3, called News Share, for sharing news articles (or any website) mainly with co-located people. It allows users to share websites that they think are interesting to the local community while browsing under Internet connections. Once articles are marked shared, the application parses, downloads, and stores the websites in the local storage on the mobile device, so that they can be directly transferred to other users and pervasive storage devices without requiring Internet access.

Figure 4 shows the usage scenario for this application. While under cellular networks, a user can browse websites and mark them shared, or obtain articles shared by other News Share users through the cloud. After the user gets on



(a) (b) (c)

Fig. 3: Screenshots of the News Share application. (a) User can select "News Share" option to share websites from a browser while online. (b) The home screen shows posts from nearby users and pervasive storage devices. Proximity-based sharing happens even while offline. (c) User can select a post to view the associated website stored in the mobile device.

a subway car where Internet connections are unavailable, the articles are sent to a pervasive storage device installed on the car via proximity-based WiFi. People getting on the same car later can receive them, and further relay them to other nearby people via Bluetooth. One of the people gets off the car, and goes to a cafe, where he shares the articles with his friend, which in turn can be shared with people coming in later through a pervasive storage device installed in the cafe. All of these communications are performed by calling the common middleware APIs for sharing content without specifying network types.

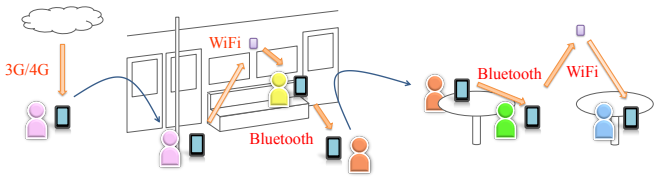
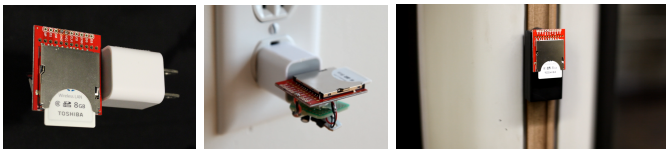


Fig. 4: Usage scenario for the News Share application.

As seen in the cafe scenario, even in cases where Internet connections are available, pervasive storage devices serve as digital sticky notes on which people can leave shared articles and messages, and thus they can be viewed as a simple tool for providing location-based services without requiring explicit location information from mobile devices such as GPS coordinates. To make installation of pervasive storage as simple as possible, we use a WiFi-enabled SD card FlashAir as mentioned earlier, because of its small size and weight (2 grams), and its low cost (around $30). As it operates with DC 3V power supply, we attached a USB AC adapter and an additional voltage regulator, so that it can be readily used by only plugging it into an electrical outlet, as shown in Figure 5(a)(b). Alternatively, two AA batteries can be used instead for short-term usage as shown in Figure 5(c).

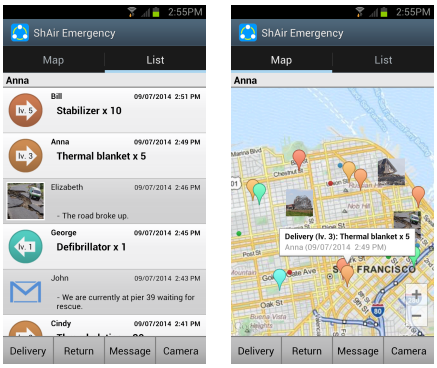
B. Emergency Response

In an emergency situation such as an earthquake or hurricane, oftentimes people will be in desperate need of communication means, but at that very moment, cell towers may be down. We implemented an application that was designed to allow affected people to submit help requests, and let people know each other's situations by sharing messages and pictures describing their surroundings, as shown in Figure 6.



(a) (b) (c)

Fig. 5: Example implementations of pervasive storage. (a) WiFi SD card attached to an AC adapter. (b) It can be used by plugging into an electrical outlet with no further configuration. (c) Alternative implementation using two AA batteries.



(a) (b)

Fig. 6: Screenshots of the emergency response application. (a) Users can submit requests for relief supplies. Pictures and messages can also be shared. (b) Shared items can also be shown on a map if GPS coordinates are associated.

The considered scenario is depicted in Figure 7. Help requests are relayed through people's devices via Bluetooth until they get picked up by a flying drone hovering over the area and providing WiFi access. The drone can further relay the requests to the mission control center using a directional antenna capable of long-distance (up to a few kilometers) communication [14]. At the direction of the mission control center, requested items will be delivered [8]. Shared messages and pictures can also be sent to the mission control center to increase situational awareness, while at the same time they can be stored in pervasive devices set up temporarily with battery power as shown in Figure 5(c), serving as emergency digital bulletin boards. The application worked in cooperation with the airborne and mission control systems provided by the respective authors of [8, 14].

Fig. 7: Usage scenario for the emergency response application.

V. Evaluation

In this section we analyze how the extended middleware works on real mobile devices when sharing data of different sizes in different situations.

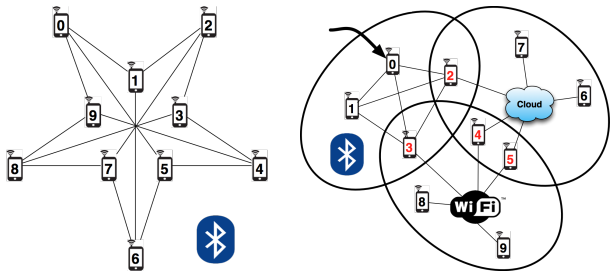
A. Setting

To perform this evaluation, we use the following tools combined in different ways in three representative scenarios:

- Ten smartphones from Samsung and Sony equipped with Android 4.1 or higher. These smartphones do not have a SIM card and are WiFi and Bluetooth capable.
- An evaluation application powered by the extended ShAir middleware discussed in Section III. This application allows us to share files of arbitrary sizes and to log the progress of file transfer with respect to time. The application automatically subscribes to any content, meaning that each phone running the application is the intended destination of any content that is shared by the application itself.
- An 8GB WiFi SD card, which is the first generation FlashAir from Toshiba.
- An 802.11n WiFi router connected to a wired 1Gbit/s Internet access provided by MIT.
- A physical Linux server running RabbitMQ enterprise messaging system connected to the internal MIT network. It allows a configurable artificial delay in order for us to simulate high-latency situations.

1) Setting of Scenario 1: Single Network: This scenario is used to analyze the baseline behavior of our evaluation application when the ShAir middleware is used without our extension, meaning that the use of more than one network technology at a time (heterogeneous network) and pervasive storage access are not available in this scenario.

Under this scenario, we perform three experiments. In the first experiment (`Sc1-WiFi`), we configure the evaluation application using the WiFi network driver, and then connect all the phones to our WiFi router. The WiFi network driver permits direct connections among the phones connected to the same access point without using its Internet connectivity. In the second experiment (`Sc1-Cloud`), we keep all the phones connected to our WiFi router, but we use the Cloud network driver instead. The Cloud driver works by using our RabbitMQ to exchange messages, therefore it would work with any other kind of Internet connection (including one based on 3G/4G). To make this second experiment more difficult and different from the first one, we configured the artificial delay of RabbitMQ to 1000ms. In the third experiment (`Sc1-Bluetooth`), we connected all the phones via Bluetooth as in Figure 8(a), which is a topology of degree 3 that minimizes the maximum distance between two phones. We had to use this topology instead of connecting all the phones to each other because the current Bluetooth stack in our phones exhibited excessive instability when heavily used with more than three phones connected.



(a) (b)

Fig. 8: Connection topologies in the experiments. (a) Experiment `Sc1-Bluetooth`. (b) `Sc2-Partitioned`.

2) Setting of Scenario 2: Heterogeneous Network: This scenario is used to analyze how our approach works when

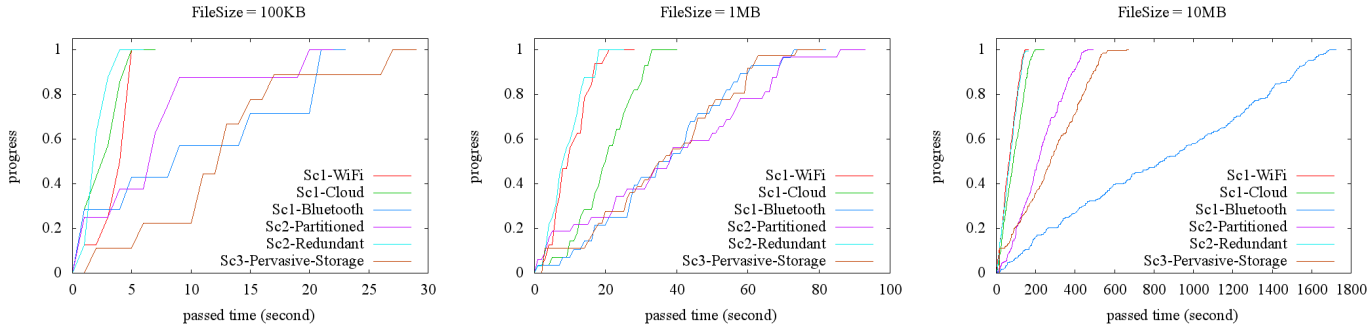


Fig. 9: Experiment results.

using heterogeneous networks, that is, multiple networks at the same time. In the first experiment (`Sc2-Partitioned`) for this scenario, we have phones that only use the WiFi driver, phones that only use the Cloud driver, phones that only use the Bluetooth driver, and finally phones use a combination of those drivers. The exact arrangement can be seen in Figure 8(b), which shows the three different network partitions. As in the previous scenario, phones with either WiFi or Cloud driver are connected to our WiFi router. The phones using a combination of drivers (depicted with red numbers) act as gateways for re-sharing content across network partitions characterized by different technologies. By running this experiment, we simulate a situation in which phones in a given location have access to one or more networks (e.g., some phones have only Internet access, some phones have only WiFi access, and some phones have both accesses). It is important to notice that, in this experiment, we have only one possible technology that connects any two phones, therefore the decision-making mechanism of our approach is not used. In the second experiment of this scenario (`Sc2-Redundant`), we configure the evaluation application with both the WiFi and Cloud drivers, so that all the phones are connected to each other using two redundant links characterized by different technologies. The purpose of this experiment is to see the decision-making capability of our approach. In both experiments, RabbitMQ is configured with a delay of 1000ms, as in Scenario 1.

3) Setting of Scenario 3: Sharing from Pervasive Storage: In this scenario, we performed an experiment (`Sc3-PervasiveStorage`) to analyze the impact of pervasive storage on the sharing experience of users. We configure the evaluation environment in the same way as `Sc2-Partitioned`, so we have phones with different technologies connected in different ways without redundant links. The only difference of this experiment is that one Bluetooth phone has the pervasive storage driver enabled and its WiFi radio is associated to the WiFi SD card. With this experiment, we show how data stored in the WiFi SD card is distributed to all the phones in the system.

B. Evaluation Parameters

For each experiment described above, we performed three experiment variations by sharing files containing random data of three different sizes: 100KB, 1MB, and 10MB. The smallest size represents short messages or any other kind of alert of the real world. The medium size represents typical MMS-style shared information (e.g., medium resolution pictures, large text files, and audio). The large size represents other heavier multimedia content (e.g., high-resolution pictures, short videos, and applications). A file was initially shared by only one random phone for each experiment, except for Experiment `Sc2-Partitioned`, which used a random phone that had only the Bluetooth driver enabled; and `Sc3-PervasiveStorage`, which used the WiFi SD card. The file size is the only input parameter that we changed for each experiment variation, while the main output parameter is the percentage of delivery of the file with respect to time, which we simply call *percentage of delivery metric*. Given that each phone is capable of measuring the percentage of completion for the shared file they have to receive, the percentage of delivery metric is the average completion per phone, excluding the phone that actually shared the file first.

C. Results

1) Scenario 1: In this scenario, we ran three experiments with WiFi, Cloud, and Bluetooth, respectively, and each of them was repeated with 100KB, 1MB, and 10MB file sizes. Figure 9 shows the percentage of delivery with respect to time. We consider these results as a baseline and benchmark to demonstrate that the middleware and network overhead added by our approach is negligible compared to the results without our approach. To make the experiments more realistic, the experiments were carried out in an area with a high number of pre-existing WiFi networks, therefore it is reasonable not to expect the performance to reach the theoretical maxima of the network technologies we used.

2) Scenario 2: This scenario is to show that our support for heterogeneous network has two benefits: (1) it is able to merge networks with different technologies; (2) it is able to optimize the distribution time when redundant links are present. From the Figure 9, we can see that in the 100KB and 1MB variations for Experiment `Sc2-Partitioned`, we were able to converge in a finite time that is similar to the Bluetooth experiments, while in the 10MB variation, the convergence is between the WiFi/Cloud experiments and Bluetooth experiments. The reason is that, since this experiment uses all the networks, it benefits from higher speeds of WiFi and Cloud partitions, but it also suffers from the slower speed of the Bluetooth partition. However, these benefits only appear when the size is sufficiently large. By looking at the `Sc2-Redundant` experiments of Figure 9, we can see that they look very similar to the fastest WiFi experiments (`Sc1-WiFi`), which means that the heterogeneous network driver is able to sense the higher delay of the Cloud alternative and chooses the lowest latency option, which is the WiFi driver that does not require Internet access. To give some statistical validity to the behavior in this situation, Figure 10 shows the

averages and the error bars that represent the standard deviation of Experiment `Sc2-Redundant` over 10 runs. As the figure shows, the results confirm that our approach is stable[2] and still behaves similarly to the fastest WiFi experiments.

3) Scenario 3: In this scenario, we perform Experiment `Sc3-PervasiveStorage`, which evaluates the time it takes for a file to move from a WiFi SD card to all the phones configured in the same way as in Experiment `Sc2-Partitioned`, where the SD card is connected to one of the devices in the Bluetooth partition pointed to by the arrow in Figure 8(b). As we see from Figure 9, the files of all the three sizes were successfully transferred in times that were similar to or, at most, slightly longer than those in Experiment `Sc2-Partitioned`. The slight delay we can see in the 10MB variation is the time that is needed to transfer the data from the WiFi SD card to the first phone. In the 100KB and 1MB variations, no clear delay can be observed, given the corresponding variances observed in Figure 10. From this result, we can say that transferring content from or to a pervasive storage device just adds its transfer time to the regular performance as seen in Scenario 2.

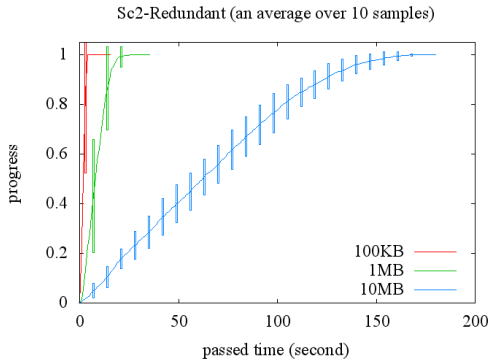


Fig. 10: Experiment `Sc2-Redundant` over 10 runs.

VI. Conclusions

In this paper, we have shown how the combined support for heterogeneous networks and pervasive storage in mobile content-sharing middleware allows for simple and seamless sharing operations. In particular, we have shown that the proposed approach works in a transparent way with respect to existing networks, does not require modifications of commercial devices or existing applications, and it is easy to manage both from the end user perspective and from the application developer perspective. By using our approach, shared data can move from a source mobile device to a destination device without the need for the use of the same network technology on both ends, and there is no need for a recipient to be active during a sharing event, since it can receive the content later from a pervasive storage device. We have implemented extension modules for the ShAir proximity-based content-sharing middleware to realize our approach. We have also implemented news sharing and emergency response applications on top of the extended middleware, together with a prototype of a pervasive storage device based on the FlashAir WiFi SD card. Finally, we have demonstrated through experiments that our extended middleware spontaneously converges when used in partitioned networks connected with different communication technologies, and is able to optimize itself in the presence of redundant links.

From the results of the experiments, we have seen that the time needed to share moderately-sized content is short enough to prove that the system can find immediate and practical use with unmodified, inexpensive commercial devices that many people currently own. However, there is still a limitation to the current system in handling large volumes of data (e.g., video files) due to congestion in wireless frequency bands. We plan to address this issue by introducing smarter routing and communication protocols. In addition, we would like to include other types of mobile and pervasive devices in our evaluation other than smartphones and WiFi SD cards, such as smart watches, smart TVs, and smart printers.

[2]Stability is clear in the case of a file size of 10MB, but is not obvious in the case of 100KB and 1MB, since actual transfer time tends to be smaller than sharing latency.

VII. References

[1] E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser. Mundocore: A light-weight infrastructure for pervasive computing. *Pervasive and Mobile Computing*, 3(4):332–361, 2007.

[2] M. Conti, S. K. Das, C. Bisdikian, et al. Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber-physical convergence. *Pervasive and Mobile Computing*, 8(1):2–21, 2012.

[3] D. J. Dubois, Y. Bando, K. Watanabe, and H. Holtzman. Lightweight Self-organizing Reconfiguration of Opportunistic Infrastructure-mode WiFi Networks. In *IEEE SASO '13*, 2013.

[4] D. J. Dubois, Y. Bando, K. Watanabe, and H. Holtzman. ShAir: An Extensible Platform for Wireless Resource Sharing on Mobile Devices. In *ESEC/FSE '13*, pages 687–690. ACM, 2013.

[5] H. ElSawy, E. Hossain, and M. Haenggi. Stochastic geometry for modeling, analysis, and design of multi-tier and cognitive cellular wireless networks: A survey. *IEEE Comm. Surveys Tut.*, 15(3):996–1019, 2013.

[6] Fixstars Corp. FlashAir Devel. https://flashair-developers.com/en/.

[7] D. Lopez-Perez, I. Guvenc, G. de la Roche, et al. Enhanced intercell interference coordination challenges in heterogeneous networks. *IEEE Wireless Comm.E*, 18(3):22–30, June 2011.

[8] P. J. Mosterman, D. E. Sanabria, E. Bilgin, K. Zhang, and J. Zander. A heterogeneous fleet of vehicles for automated humanitarian missions. *Computing in Science and Engineering*, 16(3):90–95, 2014.

[9] S. C. Nelson, G. Bhanage, and D. Raychaudhuri. Gstar: Generalized storage-aware routing for mobilityfirst in the future mobile internet. In *ACM MobiArch '11*, pages 19–24, New York, NY, USA, 2011. ACM.

[10] Open Garden Inc. Open Garden. http://opengarden.com/.

[11] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. Mobiclique: middleware for mobile social networking. In *ACM WOSN '09*, pages 49–54, 2009.

[12] S. ping Yeh, S. Talwar, G. Wu, N. Himayat, and K. Johnsson. Capacity and coverage enhancement in heterogeneous networks. *IEEE Wireless Comm.E*, 18(3):32–38, June 2011.

[13] Qualcomm Innovation Center, Inc. AllJoyn. https://www.alljoyn.org/.

[14] V. J. Sheth, M. Bokshi, E. Pruett, C. Drotar, Y. Wan, S. Fu, and K. Namuduri. A testbed for multi-domain communication networks using lego mindstorms. In *AIAA Infotech@Aerospace Conference*, 2013.

[15] T. Small and Z. J. Haas. The shared wireless infostation model: A new ad hoc networking paradigm (or where there is a whale, there is a way). In *ACM MobiHoc '03*, pages 233–244, New York, NY, USA, 2003. ACM.

[16] J. Su, J. Scott, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M. Lim, and E. Upton. Haggle: Seamless networking for mobile applications. In J. Krumm, G. Abowd, A. Seneviratne, and T. Strang, editors, *UbiComp '07*, volume 4717, pages 391–408. Springer, 2007.

[17] Toshiba Corporation. FlashAir: SD Card with Embedded WLAN. http://www.toshiba-components.com/FlashAir/.

[18] M. Xiao, J. Wu, and L. Huang. Community-aware opportunistic routing in mobile social networks. *IEEE Trans. on Computers,*, 63(7):1682–1695, July 2014.

[19] W. Zhao, Y. Chen, M. Ammar, et al. Capacity enhancement using throwboxes in dtns. In *IEEE MASS 2006*, pages 31–40, Oct 2006.

[20] W. Zheng, P. Xu, X. Huang, and N. Wu. Design a cloud storage platform for pervasive computing environments. *Cluster Computing*, 13(2):141–151, 2010.