Personal Clouds: Sharing and Integrating Networked Resources to Enhance End User Experiences

Minsung Jang†, Karsten Schwan†, Ketan Bhardwaj†, Ada Gavrilovska†, and Adhyas Avasthi‡,[1]

†Georgia Institute of Technology, {firstname.lastname}@cc.gatech.edu
‡Cisco Systems, adavasth@cisco.com

Abstract— **End user experiences on mobile devices with their rich sets of sensors are constrained by limited device battery lives and restricted form factors, as well as by the 'scope' of the data available locally. The 'Personal Cloud' distributed software abstractions address these issues by enhancing the capabilities of a mobile device via seamless use of both nearby and remote cloud resources. In contrast to vendor-specific, middleware-based cloud solutions, Personal Cloud instances are created at hypervisor-level, to create for each end user the federation of networked resources best suited for the current environment and use. Specifically, the Cirrostratus extensions of the Xen hypervisor can federate a user's networked resources to establish a personal execution environment, governed by policies that go beyond evaluating network connectivity to also consider device ownership and access rights, the latter managed in a secure fashion via standard Social Network Services. Experimental evaluations with both Linux- and Android-based devices, and using Facebook as the SNS, show the approach capable of substantially augmenting a device's innate capabilities, improving application performance and the effective functionality seen by end users.**

Keywords— mobile computing; workload offloading; capability sharing; cloud computing; management of distributed resources; Android; virtualization; social network services

I. Introduction

Smart phones and other mobile devices are operating in increasingly rich settings that include both nearby sensors and machines, and the remote cloud. By leveraging and interacting with such potentially cooperative resources, mobile device capabilities can be improved, and device users can gain enhanced interactions with their current environments.

This paper presents the Personal Cloud abstraction, which along with its runtime and its Cirrostratus implementation with the Xen hypervisor, delivers to each application a Personal Cloud instance -- *PCloud* -- coupling their mobile devices with network-attached resources. In contrast to cloud-only services like Apple's Siri and those that do not clearly distinguish between nearby and remote cloud resources such as MAUI [3], CloneCloud [4], ThinkAir [20], [21], and [22], a PCloud can service end users even when remote cloud resources are not present and/or difficult to access due to insufficient network connectivity or expensive to use via 3G/4G connections. This is because a PCloud can also run on available and free-of-charge user-owned machines in the home or on other cooperative machines.

Further benefits obtained from using local, user-own nearby vs. remote cloud resources are rapid access to such nearby devices and improved privacy, e.g., by users retaining complete control over their data rather than placing it into the cloud (e.g., voice, pictures, and etc.). The outcome is that a PCloud no longer limits mobile applications to run on single devices. Instead, it exploits the capabilities of the variety of devices available in most homes, offices, and elsewhere. Advantages derived from using PClouds vs. single devices include the followings:

Combined and augmented abilities. While mobile devices are imbued with many built-in sensors, the interpretation of sensor outputs can benefit from increased computational abilities and from data captured previously and/or stored elsewhere, e.g., in nearby desktop PCs or in the cloud. An example explored in our work is face recognition, using a camera on a smartphone (i.e., a local sensor) to capture images, but leveraging other network-accessible resources for computationally intensive work, and to deal with the fact that recognition accuracy also depends on the extent of the face database.

Improved usability. While the small form factors of mobile devices restrict their display and keyboard sizes, this is not the case for the large-display TV in a user's home (or his friend's) and the keyboard attached to his home desktop machine. We demonstrate new and secure methods for accessing and using/sharing such capabilities available on nearby networked devices that are owned by users and/or their friends or other cooperative parties.

Increased 'scope'. Referring to the fact that a device's limited storage and current context (e.g., physical location) can benefit from data resident in the remote cloud and captured by other devices, storage aggregating data from the remote cloud and all user-owned devices is shown capable of delivering improved accuracy and utility for services offered to end users. An example is the aforementioned face recognition service.

High availability. Studies [1-5] show that applications on battery-operated devices can gain performance and availability and extend their battery lives by offloading computationally expensive tasks from local to remote resources. PCloud, therefore, offers ways to seamlessly tie applications running on battery-operated devices with both nearby and remote resources. In this paper, we show how a neighborhood-watch

[1] This work was done while the author was working for Nokia Research Center North America.

application can gain performance and accuracy benefits from running across such networked sets of machines.

PCloud is similar in spirit to vendor-specific solutions for integrated use of shared devices, such as Apple's AirPlay and Microsoft's Smart Glass as well as DLNA, but in contrast to those solutions operating only across 'compatible' vendor-certified entities, PClouds have no such constraint, by using a simple model of device interaction realized at a level of abstraction 'below' that of vendor-specific software, i.e., at the system level. Specifically, the Cirrostratus implementation of PCloud operates as a set of extensions of the Xen hypervisor, including arbitrary other Xen-based devices and also interacting with non-virtualized entities like Android-based smart phones, via device-resident 'agents'.

Generalizing our earlier work on 'device clouds' [1], Personal Clouds make the following new technical contributions:

- They manage and dynamically compose distributed networked resources that are from both local/personal and remote/public devices and machines, where any of those entities can be active participants in running the services desired by end users, and all resources in the PCloud are available to the applications being run. The neighborhood watch application in the paper, for instance, takes advantage of PCloud to run its face recognition service both with and without remote cloud connectivity, albeit at different levels of fidelity based on where and how it runs.
- Device participation in any PCloud is guided by permissions and policies controlled through social network services (SNS), thus making it possible to share devices owned by different end users and/or residing at different locations. This is done in a privacy-preserving manner, via a system level service for authentication and authorization that uses Facebook's 'Friends' lists to look up and encode the relationships of users' participating devices. With such SNS-defined access policies, Cirrostratus can extend and alter a user's PCloud without the need for direct and repeated user interaction or consultation.
- PClouds protect end user privacy by tagging data – like photos – with meta-data about the devices on which it is captured, the users to which it belongs, and other such semantic information. This makes it possible to automatically 'sync' data across of all of the devices owned by some PCloud user, e.g., with a privacy-protected repository maintained on her home desktop vs. storing such data in some remote SNS not controlled by her. The user can then selectively upload photos from the repository to the SNS, and/or she can use a PCloud-provided service that gathers photos from the SNS, e.g., from those who are encoded as 'friends' and have expressed their intent to share their pictures on Facebook.
- The PCloud runtime tracks the availability of networked resources and decides what resource should be granted to a PCloud instance to meet current demand. This is guided by policies in Cirrostratus aware of current device capabilities and network conditions. An example is a face recognition service located on three different platforms – a mobile device, a nearby home desktop, and a remote cloud – with different location choices determined by current network connectivity and machine load, thus providing different levels of performance to PCloud users.

Since PCloud participants can range from entire virtual machines running on servers or in the cloud to agents operating on low end devices, PCloud instances can operate across the wide spectrum of low end devices, to high end non-virtualized or virtualization-capable mobile devices to server systems. We demonstrate the utility of this generality with both micro-benchmarks and realistic use cases. The first use case implements neighborhood-watch functions. It uses face recognition to distinguish neighbors from friends from others, where Cirrostratus orchestrates its use of network-accessible resources in the home and/or on remote cloud machines. Using local resources offers rapid response, but may suffer in terms of accuracy due to limited volumes of face data on the home machine. Additional use of remote machines improves accuracy by drawing on the cloud's global scope and data. The second use case shares devices and content among friends, where the SNS serves to establish secure interactions across friends' devices. The applications enabled by this functionality include playing a video on say, a friend's large-screen display, capturing sound from other devices, and playing sound where desired.

Experimental evaluations in Section V demonstrate the utility of leveraging multiple nearby resources for enhancing the capabilities of single mobile devices. Performance results, for instance, show a 53% reduction in execution time for a compute-intensive service on a PCloud vs. on a single device in addition to a 74% saving in energy consumption. Results also show how the use of 'friend' relationships present on social network services make it possible to securely employ diverse devices in PClouds: a ten-fold increase in data scope via SNS-maintained data results in improved accuracy for the outputs of certain data-intensive tasks.

II. Motivating Use Cases

This section describes us how Personal Clouds can leverage surrounding and cloud resources to provide end users with enhanced application functionality and performance.

A. Neighborhood Watch

A neighborhood watch is a community group organized by residents to forestall crimes and vandalism. Tasks include watching for suspicious activities and persons, notifying each other of such events, alerting neighbors about relevant local issues, and reporting certain events to external parties like the police. In this context, a face recognition service can be used to classify persons seen in the neighborhood as residents vs. others, e.g., using images taken by neighborhood or home cameras and stored in homes' image repositories. Such a service should be always-on, even when Internet connectivity is not present or when power is out, the latter requiring its ability to run on single battery-driven devices. At the same time, face recognition accuracy heavily depends on the size of the dataset being used [8], with small sample data for each face, i.e., a narrow 'scope' of data, resulting in over-fitting and making recognition unlikely. Also important is the computational capacity available for running these codes, with their compute requirements directly related to database size. Further, when images are seen, it must detect and identify potential threats in a timely manner. This makes the network bandwidth and latency between cameras and computing resources running the service critically important.

PCloud's functionality supporting services like neighborhood is the following. First, the face recognition service can, of course, run on a single device, using the photos resident on it, but second, it can obtain a wider scope of data (e.g., people's faces in this case) than those existing on a single user's device by also accessing a user's home image repository, as well as using remote data available via SNS (or even in neighbors' home repositories). Third, the service can benefit from both home machines' and remote cloud resources' computational capacities depending on local vs. remote network conditions. Section V describes additional implementation details about how PClouds backed by Cirrostratus permit the face recognition service to run with these diverse configurations.

B. *Display Sharing among Friends*

With high quality video clips and photos with multi-million pixels, recent smart phones have become equipped with high resolution displays and high performance processors. Small screen sizes continue to hamper user experiences, however. PClouds permit mobile devices seamless access to nearby large screen displays. Our previous work demonstrated this capability [1], but it required all personal cloud devices (e.g., display, keyboard, and processors) to be virtualized and owned by the same user. PClouds extend device sharing to also include non-virtualized devices and those owned by friends. Specifically, users can share any capability of any nearby device (e.g., a large display in this case), enabled by the use of social network services (SNS): the Cirrostratus infrastructure implementing this functionality ensures (i) that the device issuing a sharing request is actually owned by a designated friend who is identified via SNS, (ii) that the request recipient is the intended PCloud, and (iii) that no one else is currently using the chosen target device (i.e., the display). Secure, access-controlled sharing is realized with a security service implementing the X.509 specification and using the SNS to maintain 'friend' relationships; the devices currently supported include a user's home displays, storage devices, keyboards, and machines, devices in friends' homes (even image repositories they wish to export), and remote cloud resources.

III. PERSONAL CLOUD – SOFTWARE ARCHITECTURE

PClouds are realized as a multi-level software architecture designed to operate within dynamically changing external environments. Fig. 1 depicts the main components of the proposed architecture.

Applications. An application is a set of services running on a PCloud instance. An example is a media player consisting of a storage service holding a content file, a decoding service generating a video stream from the file, and a screen service projecting the video stream. The PCloud API permits cloud instance creation, query of and access to currently available services, and the construction of new services for a given one.

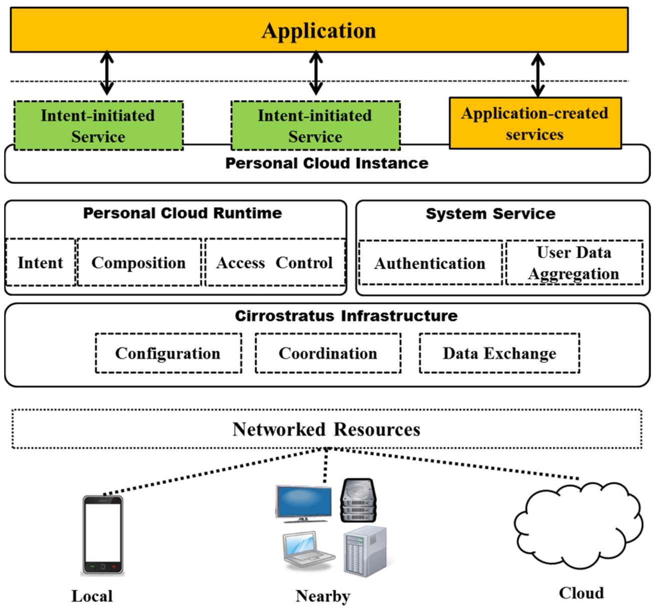


Figure 1. Personal Cloud Architecture

Services. Applications running in some PCloud instance can transparently access both local and remote services. This approach is similar to that of the Android Service implementation in the sense that it separates computing intensive or hardware access parts of applications from their main execution flows.

PCloud Instance. Each PCloud presents the illusion of a single machine. Its resources may be distributed across some set of networked devices, but they are accessed in exactly the same fashion as done by traditional applications (e.g., `open`, `read`, and `write` system calls). An instance's lifecycle transitions from its initialized, to started, to terminated states.

PCloud Runtime. The runtime brings up a PCloud instance for an application, with its 'intent' interface used to describe resource requirements, e.g., a large screen with HD resolution, presence of a face recognition service, etc. It also allocates resources for the services being asked for a given instance.

System Services. These are used to authorize applications to run, and for actions requiring global knowledge, such as authentication and a data storage with global scope, so they directly interact with Cirrostratus.

Cirrostratus. These set of extensions of the Xen hypervisor discover and monitor both locally and globally network-reachable resources, in order to maintain a pool of distributed resources accessible to each end user. The PCloud runtime draws resources from this pool and combines them to establish and maintain PCloud instances.

A. *Applications*

Designed to support existing programing models and minimize code development, a PCloud application operates much like one running on a single device or machine. It runs a simple preamble to create a PCloud instance, then runs its services, the latter able to use all of the resources in PCloud.

It is the responsibility of Cirrostratus and its runtime to ensure resource availability in lieu of changing network conditions and resource use by others.

B. *Services*

PCloud services are like those described in the Intel processing framework [24], but differ in implementation in that they use local 'service proxies' for the services that are run remotely. Specifically, for each remotely run service, the runtime instantiates a local proxy that reveals how to reach the remote service, what service is required for dependency, and what resources are needed to run it. State information is maintained, as well, relayed by the runtime to the actual remote service location. The current API has operations to enumerate, attach, use, and detach services, and to create and launch new ones. System services differ from others in that they are always present and available.

C. *The Personal Cloud Instance*

The 'resources' presented in a PCloud instance are either services or devices, and for both types of participants, Cirrostratus exposes and maintains their *capabilities*. For instance, a desktop machine can separately expose devices like its disk storage, processing capacity (CPU), and large display. This means that a PCloud application can run across say, the CPU of a laptop, the storage on a desktop, the keyboard of a smartphone, and a separate large-screen display of a TV, all as if it were running on a single platform offering all of those device resources.

Other than running the aforementioned preamble, the use of hypervisor-level technologies in Cirrostratus makes it possible for entirely unmodified single-machine applications to run across such virtual platforms. However, users may wish to impose constraints on the platforms being constructed. For this reason, the PCloud runtime provides additional APIs for an application to guide a construction of such platforms: the 'Intent' runtime module and API can be used to submit a list of required services along with their required capabilities. An example is the 're-wiring' of a user's display device, to say, move the depiction of a streaming video currently being viewed by the user on his laptop screen to his large home TV display. A PCloud instance supporting this example may continue to use the exact same laptop resources as before, and simply add the large home TV as a screen resource.

D. *The Personal Cloud Runtime*

The PCloud runtime's tasks are (i) to establish a PCloud instance with services per a user's request or 'intent' – the Intent module, (ii) to decide which resources are most suitable for running requested services – the 'Composition' module, and (iii) to manage the certificate of the device owner for authentication – the 'Access Control' module. The Intent module builds a list of services requested by an application and determines the types of resources needed to run them. Based on its inputs, the Composition module interacts with the Cirrostratus layer to allocate resources, the latter mapping the best-fit resources to requested services, and it establishes a PCloud instance with such resources and services.

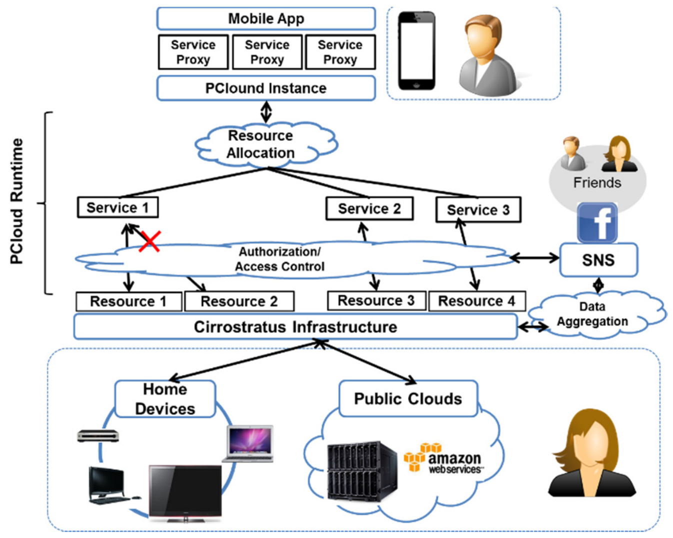


Figure 2. Using a Personal Cloud between people identified via a SNS

The Access Control module works with the authentication system service checks access permissions for the resources requested for some specific user.

E. *System Services*

System services provide global information to other PCloud components and to user-created applications and services. Two default current system services are: (i) the authentication service and (ii) the user data aggregation service. They both interact with social network services (SNS) to authenticate the users of resources joining a PCloud platform, permit/deny requests for resources, and they also use the SNS as a repository of user data for the applications running with a PCloud instance. Authentication leverages the SNS, and aggregation is a special storage service that assembles data about users from both their local disks and the remote clouds, to create a virtual storage service for each cloud instance. The purpose is for the services running on that cloud to have access to the data they need. Like others, system services also use local proxies.

F. *Cirrostratus*

Cirrostratus extends or earlier work on Stratus and its 'device clouds' [1] in three substantial ways. First, Stratus device clouds required all active cloud participants to be fully virtualized platforms running the Xen hypervisor. Cirrostratus also permits non-virtualized devices like Android-based smartphones to be active participants. Second and more important is the use of SNS to implement the access control methods needed for device sharing across 'friends' and for access to friends' data, the latter increasing the data scope available to individual participants. Third, Cirrostratus is able to use remote cloud services (e.g., Amazon EC2) as PCloud resources, thus substantially increasing PCloud functionality and capabilities. Fourth, for all such resources, it is the responsibility of Cirrostratus to maintain status information of about participants' devices, via active device and network monitoring. The 'Coordination' module carries out these tasks. Finally, Cirrostratus' 'transport' layer, termed 'Data Exchange' in Fig. 1, is a straightforward extension of the data exchange protocol developed for Stratus.

Fig. 2 illustrates an example of PCloud use between two users. It illustrates how the aforementioned components relate to others so that users can leverage their own network-reachable resources as well as share others, in a secure fashion, by interacting with a SNS

IV. Select Implementation Detail

Cirrostratus supports the Linux and Android systems running on native or on virtualized hardware. This section presents the detail about these implementation needed to explain the performance results shown in Section V.

A. Cirrostratus

We review interesting aspects of the Data Exchange, Coordination, and Composition modules.

Coordination. Cirrostratus uses the master/slave model to manage the resource pool available for use by PClouds. Its current implementation elects the one with the largest performance index (defined later) as the master among nearby devices. Battery-operated devices are precluded from that role, because of the master's computationally intensive workload and the need for high availability. Slaves contribute resources with certain capabilities.

Configuration. The configuration module runs on all participating Cirrostratus devices. It uses static data as well as online monitoring to gather information about the capabilities of available device resources, and it provides such information to the master for storage in a central PCloud repository of device capabilities. There are also 'knobs' available to device owners permitting them to control their devices' degrees of participation, including preventing certain device resources from being used (e.g., a device's display). Finally, there are resource quotas to control device usage. The naming scheme used in configuration translates `<scr1.dt2.pcloud3.bob>` to the fact that Bob has the screen1 connected to the desktop2 in the personal cloud managed by PCloud3. A single user may have multiple names, according to his location, say in the home or office. The configuration module also supports resource discovery. That is, if a new device joins any PCloud, this module on the device reports the device's capabilities to the master of that PCloud upon discovery and registration.

Data Exchange. Data is exchanged via a publish-subscribe transport termed M-Channel [1][15]. M-Channels create a software layer for applications to communicate with others transparent to underlying physical connections being used.

B. Resource Allocation

Resources are allocated to services. The current implementation uses a simple allocation heuristic guided by summary data about resource capabilities changed dynamically. Allocation operates within stable allocation epochs, assuming that (i) the status of resources and the network connection are unlikely to change during the current epoch, and (ii) if changes occur, they will persist during the next allocation epoch. We have not yet considered multi-epoch allocation planning and/or the evolution of allocations across multiple epochs in place sophisticated allocation methods. We currently approximate the status of each computational resource by computing its performance index (PI) and its network connectivity. When N is the number of processor cores in a device, and ω is a weight assigned to its processor architecture, the PI is defined as follows:

$$PI = MemorySize + \omega\sum_{n=1}^{N}\{ClockFreqofCore_n \times (1 - Util\_Core_n)\}$$

MemorySize: RAM size of a resource in MB.
ClockFreqofCore$_n$: Operational frequency of a core in MHz.
Util_Core$_n$: Current utilization (0-1) of a core

As the metric for resource allocation, we can then estimate the execution time, $T_{estimation}$, of a service on any resource as:

$$T_{\text{computiation}} \cong \frac{Workload\ size\ being\ processed\ by\ a\ service}{Performance\ Index}$$

$$T_{network} \cong\ Elapsed\ time\ to\ transmit\ data\ over\ the\ network$$

$$\therefore T_{estimation} \cong T_{computation} + T_{network}$$

PCloud composition for an application with some set of services, then, operates as follows. When the application requires computation-related services, the composition module interacts with Cirrostratus to obtain the performance index (PI) and network connection status of each candidate computational resource. Each service is then mapped to the resource with the largest PI, as its initial allocation. Subsequent runtime monitoring uses the service proxy to obtain additional detail about service execution, including the volume of data transmitted over the network and the elapsed finish time for service requests. Allocations are changed when service times substantially and consistently exceed previously observed values; we have not yet investigated rigorously the dynamic methods needed for runtime re-allocation. Our alternative focus has been on showing PClouds to be viable and useful for realistic application.

C. System Services

For brevity, we elide further detail about the implementation of the PCloud runtime, instead describing those system services critical to the ability of PClouds to span distributed resources.

1) Authentication Service

The authentication service is implemented as a Facebook app, and like other services, its proxy resides within each PCloud instance. A user must install this app on her Facebook account and create a Facebook group with the label "DeviceShare." By then listing appropriate Facebook friends, she indicates her willingness to share the capabilities of devices in her own PCloud with others in this group. Once the app is installed on her Facebook account, the server hosting the app generates and holds her certificate, based on the X. 509 standard.

The following example illustrates the authentication process. Alice and Bob are Facebook friends and each install the application on their accounts. One day, Alice visits Bob's house and wants to use his large display to share pictures with him that reside on her phone. By running the Facebook app (the authentication system service), Alice is shown the list of people who are willing to share their resources with her.

Choosing Bob prompts the server to send out her request to his PCloud along with Alice's certificate. The access control module in the runtime then verifies Bob's resources she is allowed to access. In terms of access rights, our current implementation grants access to resources based on group membership, either owner or guest. In this case, Alice belongs to the guest group, while Bob is in the owner group. Alice's phone also receives the IP address of Bob's network, along with Bob's certificate. With this information, the data exchange module in Bob's Cirrostratus establishes a connection to Alice's phone, whereupon her phone then becomes a 'guest' participant of Bob's Cirrostratus PCloud.

2) *User Data Aggregation Service*

The service used to collect user data consists of the aggregate total of a user's remote and local storage. Its remote part extracts photos and user-written tags from SNS accounts listed as 'friends', while the local part provides storage to participants of Cirrostratus, e.g., so that it can aggregate all photos from participating devices. The remote store is comprised of a Facebook app and a remote storage service. If a user installs this app on her account, it collects photos along with tags from friends' accounts that open their photos in albums for friends or for the public to see. Since on Facebook, tags of photos usually include information regarding who appears in photos, the app can send both the photos and such meta-data to some remote storage site. For the local synced storage implementation, we use an rsync-based tool and SparkleShare [23].

D. *Implementation Setup*

1) *Virtualized Devices*

We use an Intel Atom mobile platform with the Xen 4.1 hypervisor, running Linux and Android virtual machines. The control domain of Xen hosts the PCloud runtime and its other necessary components. For both Linux and Android on guest domains, M-Channels link guest actions with PCloud components.

2) *Non-virtualized Devices*

Cirrostratus uses 'agents' to interact with non-virtualized mobile devices, such as actual Android phones or tablets, using Google's Android 4.0 Ice Cream Sandwich and 4.1 Jellybean. The agent is a service module on the Java service layer that essentially emulates the functionality resident in the control domain (Dom0) of virtualized systems.

V. Experimental Evaluation

The two case studies described in Section II are the basis for demonstrating the utility and performance of the Personal Cloud approach and implementation.

A. *Neighborhood Watch with Face Recognition*

1) *Test-bed Configuration*

The resource pool managed by Cirrostratus consists of two desktops (`N1, N2`, respectively), an m1.large EC2 instance (`EC2`) as well as a mobile device. Further, the neighborhood watch application on a mobile device uses a camera connected via Android on our virtualized Intel Atom platform. For non-virtualized devices, the same software employs a camera installed on the Nexus 7 tablet.

TABLE I. Participants in the neighborhood watch.

Name	**The number of recognized faces**	
	Faces in Facebook	***Standalone phone***
A	32	10
B	23	16
C	24	9
D	35	11
E	14	5

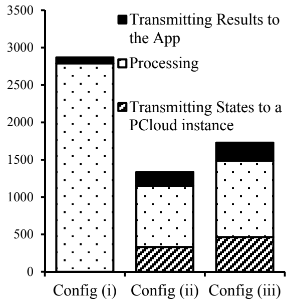
The face recognition service operates in three different configurations: (i) on a standalone mobile device, (ii) on a PCloud instance with a set of ambient devices, and (iii) simultaneously using a remote server running on the Amazon EC2 instance as well as (ii). Each configuration performs the tasks of the neighborhood watch application, which include taking a photo, detecting and identifying faces on the photo, returning results to a user, and starting all tasks over again every 3 seconds. In addition, we introduce two cases in which execution environments differ in terms of network status and resource utilization. In exclusive execution mode, the neighborhood watch is the only application that gains access to all resources given by the PCloud, whereas in mixed execution mode, other applications are allowed to simultaneously access such resources. We use OpenCV 2.4.2 with the Eigen-face algorithm [6] to build this application.

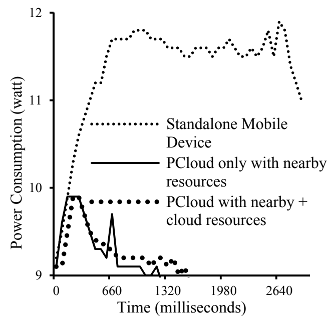
2) *Evaluation Scenario*

When some user monitors a street close to his house, he runs the neighborhood watch app on his mobile device. The app captures peoples' images on the street periodically (every 3 seconds) and determines who they are. These result in capacity queries sent to each configuration of the face recognition service: (i) located on the device, (ii) a user-owned device, and (iii) a remote server. For this use case, our current selection heuristic is to initially choose the nearby service unless the remote one, i.e., the Amazon instance, offers a performance index that is twice as high. If a nearby service is not available, the heuristic attempts to work with a remote cloud service. Using the compute power on a mobile device is the last resort.

3) *Results: The 'Scope' of Data*

Five fellow students helped run the experiment by installing the required Facebook app for the data aggregation service, resulting in the remote store containing all photos and tags on their respective Facebook accounts for identified friends, the idea being to emulate several parties living in the same neighborhood. We assume that a user A, in TABLE I, has a mobile phone participating in the PCloud. The numbers of faces listed in TABLE I use photos from the data aggregation service attached to all participants (i.e., A, B, C, D, and E) since all of them are in the 'DeviceShare' group in Facebook. The numbers mentioned in TABLE I. refer only to those that are sufficiently clear to permit our face recognition software to correctly identify faces. Note that the total number of distinct faces accumulated by the data aggregation service is 105, after removing duplicated results, thereby demonstrating its ability to provide a wider scope with a broader set of data set than that available on any specific device.





(a) User-experienced response times (b) Power consumption

Figure 3. (a) shows user-experienced response time in milliseconds for each photo under different configurations, while (b) shows power consumption in watts per configuration.

4) *Results: Performance*

We first evaluate the elapsed time needed to identify a person's face via the neighborhood watch app. In the mixed execution environment, the neighborhood-watch app sends a face recognition service request every 3 seconds for 5 minutes. Then the PCloud runtime and Cirrostratus build a PCloud instance and allocate a resource for the service either on one of the nearby devices (`N1` or `N2`) or on the cloud (`EC2`). Fig. 3 is a snapshot of the elapsed time to process one request. As seen in Fig. 3(a), the service located on the nearby resource, Config. (ii) (Nearby), shows the best response time when each service can identify a given face. With Config (ii), transmitting states and user data consumes 331ms out of the total elapsed time of 1335ms (about 24.7% of time). This includes the time needed for the PCloud runtime to construct a PCloud instance to meet the requirement of the Face recognition service before transmitting them. However, most of the time is influenced by the network connection status: the time is less for nearby devices, compared to using the remote cloud, due to the network delay.

Fig. 3(b) shows that power consumption is substantially higher when the neighborhood watch runs on a single device. This is in stark contrast to results shown for the 'nearby' and 'cloud' configurations: the energy consumption on the mobile device of Config. (ii) and (iii) (Cloud) compared to (i) (Standalone) is reduced by 74% and 77%, respectively. Even if we omit CPU usage on the mobile device for each configuration, it still follows the power consumption plot in Fig 3(b): this means that power savings are due to the fact that compute-intensive face recognition functions can be successfully offloaded to a suitable computing resource.

During the experiment, resource allocation takes place 19 times, due to changes in network condition and resource availability, caused by other workloads being run. Fig. 4 shows the user-experienced total response time taken to obtain recognition results. If the resource is located on the cloud, network delay is three times larger than when using the nearby resource, as seen in Fig. 5, but the increased computational capabilities of the cloud successfully offset that additional delay. Fig. 6's comprehensive comparisons show that for this application, user-experienced response times are almost identical no matter what resource is given to the service.

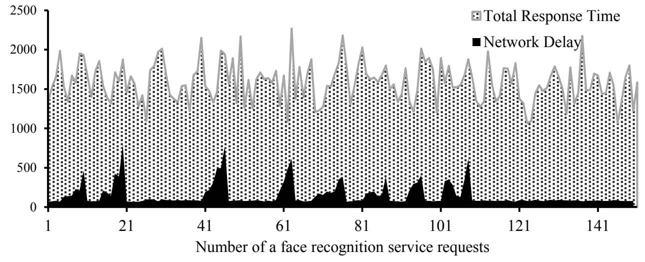
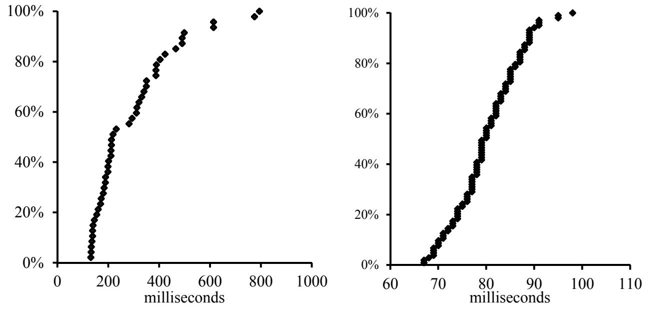


Figure 4. User-experienced total response times in milliseconds for face recognition requests sent every 3 seconds for a 5 minute duration.



(a) Service on the cloud (b) Service on nearby devices

Figure 5. Cumulative distribution functions of network delay when the service runs on the cloud (*EC2*) or on a nearby device (*N1,N2*), respectively.

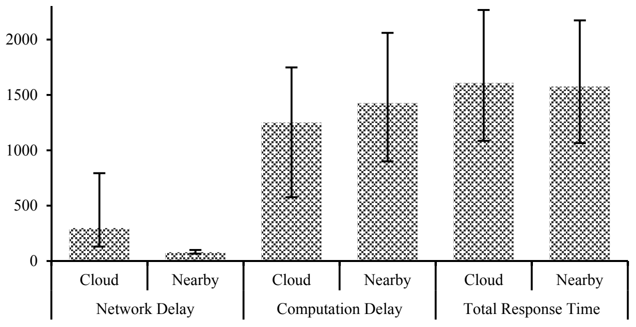


Figure 6. Average delays in milliseconds

B. *Display Sharing*

1) *Testbed Configuration*

We evaluate the costs of projecting a small screen on a smartphone to a large nearby display. We use a 23-inch monitor with full HD resolution (1900 x 1080) attached to a desktop PC, participating in PCloud interacted with the master in Cirrostratus, all connected to the mobile device via 802.11n.

Our Atom platform and the Nexus 7 offer 1024 x 600 and 1280 x 800 resolution, respectively. The authentication service runs on an Amazon t1.micro instance hosting a Facebook app.

2) *Evaluation Scenario*

Suppose that Alice shows video clips of her newborn baby when she visits Bob's house. With a PCloud instance, if a PCloud environment is set up at his home, she can easily depict this clip, as long as both Alice and Bob are friends on Facebook. In this case, Alice asks Bob to share the large TV with her, and he places her into the "DeviceShare" group in his Facebook account. Assuming she already installed the Facebook app for the authentication service, when she starts the app, Facebook leads her to a server that hosts the app. Now the app shows a list of people who are willing to share their device's capabilities by putting her on their "device share" group. She selects Bob on the list.

If this is her first use of the authentication app, the authentication service generates her certificate based on PKI (X. 509 specification). The service also provides the IP address of Bob's Cirrostratus master along with her private/public key pair. Authentication is completed when the PCloud software in Alice's phone starts a connection with Bob's Cirrostratus master, with all subsequent communications transmitted via a secure M-Channel connection. Using this connection, Cirrostratus is allowed to share with Alice's device a list of all permitted capabilities for guest users.

Our current implementation only supports two groups—the owner group and the guest group, where only a member of the owner group can adjust the capabilities being shared. The Personal Cloud software in her phone, then, chooses the desired display, lets the capability service know about its selection, and finally, starts transmitting the mobile phone's screen to an appropriate PCloud instance. Before the transmission takes place, the PCloud runtime in Bob's house initiates a PCloud instance including the large TV and begins with the remote display service on the instance. Then the desktop machine starts its VLC receiver as the remote display service to project Alice's screen on its display.

3) Evaluation Results

TABLE II shows the elapsed time from when Alice sends a request for display sharing for the first time to Bob's Cirrostratus. In this case, Alice has to first create her certificate. As seen in TABLE II, it takes 1471.5ms to be ready to project the screen on her mobile phone to Bob's display, likely superior to a case in which Alice and Bob try to wire these devices manually. Further, with PKI and SSL connections obtained by the authentication service, such sharing is done in ways that respect access rights and provide end user security.

C. Discussion

Interesting to note about the neighborhood watch application is that while local resources offer low latency response, the increased data scope of remote resources can offer improved accuracy, and in addition, for this application, the substantially faster remote resources can also hide some of the additional network delay seen for requests to the remote cloud vs. local machines. Such tradeoffs are a general attribute of the PCloud approach, encouraging future work combining performance indices capturing such tradeoffs, with networking data, and with indications of what may be most important to end users when services are run, i.e., user 'intent'. In fact, it is because of the flexibility offered by PClouds that such tradeoffs become possible, permitting mobile devices to opt for 'best-fit' resources in their current environments. The PCloud approach supports such actions via continuous system monitoring to understand the capabilities offered by currently reachable resources (the performance index in this case) and network connectivity.

TABLE II. ELAPSED TIME FOR DISPLAY SHARING

Task (message)	**From**	**To**	**Time (milliseconds)**	**Dev.**
Initiate a certificate	MO	S	88.2	27.6
Return the certificate and key pairs	S	MO	213	35.3
Authentication	All	All	405	41
Send a display sharing request	MO	SM	140	66.3
Return a list of available capabilities	SM	MO	293	117.3
Notify a selection of a display that wants to use	MO	SM	179	55.8
Initiate a VNC connection	SM	MO	153.3	80
Total Elapsed time			1471.5 ms	

MO: a mobile device, S: the authentication service, S M: a Cirrostratus master

Display sharing via PClouds demonstrates seamless and secure ways to leverage nearby resources, supported by SNS-based authentication and access control. Offline certificates [18] can be used when remote SNS services cannot be reached. More generally, efficient operation disconnected from remote clouds is an obvious advantage provided by PClouds, for which we are currently considering additional Cirrostratus enhancements that can distinguish important from less important services when a user wishes to run multiple services on potentially scarce nearby resources.

VI. RELATED WORK

ThinkAir [20] proposes dynamic resource allocation on the server side to maximize parallelism for offloaded workloads. MAPCloud [19] models mobile application as a location-time workflow (LTWs) and maps LTWs onto online resources with 2-tier clouds. Our work also considers two layers of resources, nearby and remote clouds, but PClouds can share fine-grain device capabilities among participants as well as perform compute offloading. HomeOS [1] pursues goals similar to those of PCloud. Like PCloud, it permits users to formulate and enforce desired policies across all HomeOS devices, but it does not exploit SNS to enable the access control and consequent rich methods for device sharing present in PCloud. SBone [13] uses social networks to create an overlay over existing social network graphs for enabling devices to share content, state, and computational resources. Cirrostratus shows the utility of leveraging efficient local area networks, and it can compose a new abstract at the finer granularity of individual device 'capabilities'. We leverage well-known authentication technology, with the innovative approach of using the Facebook SNS as an authentication service. Our approach implements a framework similar to [17], which suggests using Google's OpenSocial and Facebook's Connect to provide APIs for web-based social network applications, allowing users to carry their identities across applications and devices.

Using remote resources to augment the compute performance of mobile devices has been studied since [9]. Recent research like MAUI, CloneCloud, Mars [7], and Cuckoo [5] address the question of how to find the compute intensive portions of an application and move those between battery-operated devices and high-performance backend servers. The focus is on remote cloud servers, however, so these systems do not have functionality that describes and can exploit the varied capabilities present in the many and often heterogeneous devices present in the nearby cyber-environments targeted by our work. Similar to our own previous work, Cloudlets [16] use small servers as nearby computing resources to augment mobile devices, via virtual machine migration, but the system does not support non-virtualized devices. [12] suggests use cases similar to ours, arguing that the use of smartphones heavily depends on context, in particular, on other devices and places, and on the situations users are experiencing.

VII. Conclusions and Future Work

Personal Clouds are shown capable of substantially augmenting the capabilities of mobile devices, and they can alleviate their limitations, including lack of performance, limited battery lives, and constrained form factors. They can also deal with the restricted the 'scope' of the data currently resident on each device. In contrast to remote cloud services used by mobile devices, PClouds can also augment device capabilities through the use of nearby devices. The outcome is not only increased storage and computational capacities, but also the creation of entirely new functionalities not available from remote services, such as the ability to present images on large displays, the potential to share content not resident in remote clouds, and others. PClouds leverage Social Network Services (SNS), i.e., those provided by Facebook, for authentication, access control, and for secure device interaction. Their hypervisor-level realization includes fully virtualized devices as well as non-virtualized mobile devices. Performance evaluations show PClouds capable of augmenting device capabilities to improve their performance as well as enhance the functionality seen by end users.

Our ongoing and future work is exploring several additional dimensions of service sharing and offloading. First, we are exploring offloading computational services to accelerator devices, like those accessible via OpenCL. Second, we are experimenting with useful device-device interactions, including those in which a user controls his large-screen display by manipulating his smart phone's screen. Third, also needed are additional evaluations in actual home environments with less mature Internet or Wi-Fi connectivity.

Acknowledgements

We gratefully acknowledge this work was funded in part by Intel via the Intel Science and Technology Center for Embedded Computing (ISTC-EC), and by additional support provided by a gift from Nokia Corporation.

References

[1] Minsung Jang, K. Schwan., "STRATUS: Assembling Virtual Platforms from Device Clouds," 2011 IEEE International Conference on Cloud Computing (CLOUD), pp.476-483, July 2011

[2] M. R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices," in Proceedings of the 9th Mobisys, 2011, pp. 43–56.

[3] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in Proceedings of the 8th Mobisys, 2010, pp. 49–62.

[4] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in Proceedings of the sixth conference on Computer systems, 2011, pp. 301–314.

[5] R. Kemp, N. Palmer, and T. Kielmann, "Cuckoo: a computation offloading framework for smartphones," Intl. Conf. on Mobile Computing,, 2010.

[6] M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of human faces," Pattern Analysis and Machine Intelligence, vol. 12, no. 4, 1990.

[7] A. Cidon, T. M. London, S. Katti, C. Kozyrakis, and M. Rosenblum, "MARS: adaptive remote execution for multi-threaded mobile devices," in Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds, 2011, p. 1.

[8] M. Guillaumin, "Is that you? Metric learning approaches for face identification," In Twelfth International Conference on computer Vision, 2009.

[9] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," in ACM SIGOPS Operating Systems Review, 1999, vol. 33, no. 5, pp. 48–63.

[10] R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden, "Wishbone: Profile-based partitioning for sensornet applications," in Proceedings of the 6th USENIX symposium on Networked systems design and implementation, 2009, pp. 395–408.

[11] S. Kannan, A. Gavrilovska, and K. Schwan, "Cloud4Home -- Enhancing Data Services with @Home Clouds," 2011 31st ICDCS, pp. 539–548, Jun. 2011.

[12] T. Matthews, J. Pierce, and J. Tang, "No smart phone is an island: the impact of places, situations, and other devices on smart phone use," Research Report RJ10452, IBM, vol. 10452, 2009.

[13] S. Pravin, N. Nath, I., V. Ananthanarayanan, L. Han, "SBone: Personal Device Sharing Using Social Networks," Technical Report DCS-tr-666, Rutgers University, 2010

[14] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: can offloading computation save energy?," Computer, vol. 43, no. 4, pp. 51–56, 2010.

[15] S. Kumar, V. Talwar, V. Kumar, P. Ranganathan, K. Schwan. "vManage: Loosely Coupled Platform and Virtualization Management in Data Centers," In 6th ICAC, Barcelona, Spain, 2009

[16] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. 2009. "The Case for VM-Based Cloudlets in Mobile Computing," IEEE Pervasive Computing 8, 4, 14-23

[17] A. V. Ramachandran and N. Feamster, "Authenticated out-of-band communication over social links," In Proceedings of the first workshop on Online social networks - WOSP '08, p. 61, 2008.

[18] D. Bauer, D. Blough, and D. Cash, "Minimal information disclosure with efficiently verifiable credential,". In Proceedings of the 4th ACM workshop on Digital identity management (DIM '08). pp. 15-24, 2008.

[19] M.R. Rahimi,; N. Venkatasubramanian, S. Mehrotra, A.V. Vasilakos, "MAPCloud: Mobile Applications on an Elastic and Scalable 2-Tier Cloud Architecture," Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on , vol., no., pp.83,90, 5-8 Nov. 2012

[20] Kosta, S.; Aucinas, A.; Pan Hui; Mortier, R.; Xinwen Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," INFOCOM, 2012 Proceedings IEEE , vol., no., pp.945,953, 25-30 March 2012

[21] X. Qiu, H. Li, C. Wu, Z. Li, and F. C. M. Lau, "Cost-minimizing dynamic migration of content distribution services into hybrid clouds," 2012 Proceedings IEEE INFOCOM, pp. 2571–2575, March 2012.

[22] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," 2012 Proceedings IEEE INFOCOM, pp. 2716–2720, March 2012.

[23] SparkleShare, http://www.sparkleshare.org

[24] http://software.intel.com/sites/landingpage/perceptual_computing/documentation/html/

[25] C. Dixon, R. Mahajan, S. Agarwal, A. J. Brush, B. Lee, S. Saroiu, and P. Bahl. 2012. "An operating system for the home," In Proceedings of the 9th USENIX NSDI'12. USENIX Association, Berkeley, CA, USA, 25-25