SPiCa: A Social Private Cloud Computing Application Framework

Chii Chang\textsuperscript{a}, Satish Narayana Srirama\textsuperscript{b}, Sea Ling\textsuperscript{c}

\textsuperscript{a}chii.chang@acm.org, Institute of Computer Science, University of Tartu, Estonia.
\textsuperscript{b}srirama@ut.ee. Institute of Computer Science, University of Tartu, Estonia.
\textsuperscript{c}chris.ling@monash.edu. Faculty of Information Technology, Monash University, Australia.

Abstract

Mobile devices are capable of acting as smart assistances not only to serve their users but also to collaborate with each other remotely via wireless Internet to accomplish common goals. The latter is achieved by establishing a Social Private Cloud (SPC). SPC is a cluster formed by a scalable group of social network participants using their mobile devices that are capable of providing their resources to accomplish computational tasks. In this paper, we propose a workflow-based SPiCa framework that enables task delegation in SPC. In order to support adaptive task scheduling based on resource availabilities, a resource-aware task scheduling scheme has been proposed and implemented as a proof of concept. The evaluation demonstrates that the framework is capable of dynamically reacting to runtime changes in order to adjust the task delegation process.

Keywords: Mobile device cloud, mobile Web service, application framework, resource-aware, task delegation.

1. Introduction

In the last decade, numerous approaches \cite{1, 2, 3, 4} have been proposed to utilise mobile devices (in particular—smartphones) to provide software services. The emerging technologies granted mobile devices advanced capabilities in terms of computational power, interoperability and connectivity seamlessly in ubiquitous environments. The devices are now capable of acting as smart assistances not only to serve their direct users but also to collaborate with each other remotely via wireless Internet to accomplish common
goals [5, 6].

Today, the hardware specifications of high-end mobile devices can compete with many common desktop computers. The high performance mobile devices are capable of establishing a distributed computing network group to collaborate for some tasks [1]. For example, mobile users may partition complex computational tasks into several smaller task chunks from their mobile devices and distribute the task chunks to the mobile devices of their real world friends when their friends’ mobile devices are in idle (or in charge) mode assuming resources of these devices are capable of handling the allocated tasks. Furthermore, popular cloud services have extended these mobile devices’ resource capabilities. The computational power of a mobile device is no longer limited to its physical hardware but also include its interconnected cloud utility services.

Previously, researchers [7, 8, 9] have proposed a number of frameworks to enable mobile device-based collaboration. However, because of the limited computational powers of early mobile devices, past approaches were based on centralised solution or relying on stationary mediators. Besides the bottleneck and single point of failure issues of centralised architecture, the stationary mediators require additional hardware settings, which are less flexible. As mobile technologies evolved, today it is possible to realise the mobile host-based collaboration without the assistance from stationary nodes [1].

Distinct from the existing standalone mobile agent-based approaches, in this paper, we introduce Social Private Cloud (SPC), a service-oriented mobile computing environment to enable task collaboration using mobile and cloud resources within social group users. In SPC, a mobile device represents a participant. They are free to join and leave the network. The resources of each mobile device include its physical hardware and its backend cloud services such as services deployed on Google App Engine\(^1\), its associated cloud storage such as dropbox\(^2\) or MEGA\(^3\), as well as its direct connected participants. In other words, a SPC is a cluster established by a scalable social group of mobile devices that are capable of providing their resources to accomplish computational tasks.

---

1. https://appengine.google.com/
2. https://www.dropbox.com/
3. https://mega.co.nz/
SPC can be applied in numerous applications. For example, in Mobile Social Network in Proximity (MSNP) [10], SPC lets participants to perform collaborative trustworthy service discovery process towards enhancing the performance. In Mobile Crowdsensing [11], SPC can balance the resource usage of the mobile participants. Similarly, SPC can also be applied in establishing collaborative processes in the Internet of Things for Disabilities scenario [12].

Enabling SPC faces challenges in resource management. The connectivity of mobile devices is not guaranteed and the resource availability of devices varies on different hardware models and their associated cloud services. Distributing tasks to participants cannot be done equally but one needs to consider the availability of resources. In this paper, a workflow-based framework—Social Private Cloud computing application framework (SPiCa)\(^4\) is proposed to enable dynamic task delegation in SPC based on resource-awareness. SPiCa is capable of analysing the performance and resource availability of task participants at runtime and performing a cost-efficient task scheduling strategy to delegate computational tasks among SPC participants. The framework is based on service-oriented architecture with standard technologies including W3C standard-based Web service technologies and OASIS BPEL4WS [13]. The evaluation of the prototype demonstrates that the framework is capable of dynamically reacting to runtime changes in order to accomplish the goal.

The paper is organised as follows. Section 2 provides an overview of Social Private Cloud environment and the proposed SPiCa framework architecture. Section 3 describes the proposed resource-aware adaptive task scheduling schemes. Section 4 describes how to apply the SPiCa framework in an SPC scenario using a case study. In Section 5, a preliminary prototype evaluation demonstrates the performance of SPiCa. In Section 6, a number of related works have been reviewed and compared with the proposed SPiCa framework. Finally, the paper is concluded in Section 7.

2. System Design

2.1. Overview of Social Private Cloud

Figure 1 illustrates the conceptual architecture of SPC. In SPC, every mobile device hosts an agent. The term—agent is derived from the soft-

\(^4\)Spica is the brightest star of Virgo constellation.
The agents communicate via mobile Internet based on standard mobile Web service protocols and are capable of collaborating for computation tasks that are described by the workflow process definition documents. Each agent has three types of resources—hardware of mobile device, remote cloud services and its Direct Connect Nodes (DCN), which is a group of directly connected external agents. Each agent’s available resources are profiled in a document called Resource Profile (RP) which is shared among DCN members.

A member of an agent’s DCN may have its own DCN, whose members are indirectly connected to the current agent. Agents which are indirectly connected to the current agent are Indirect Connect Nodes (ICN). For example in Figure 1, DCN of Agent R has three members—X, Y and Z. Agent X has three connected nodes in its DCN—Agent A, B and C, which are members of Agent R’s ICN. Suppose Agent R scheduled three tasks to Agent X, Y and Z. Since Agent X has three other members in its DCN, Agent X can further reconfigure its task to three tasks and schedule the tasks to its DCN members.

Agents maintain their RP and the RP are shared among the agents’ DCN either by direct interaction or via their cloud storages. Suppose Agent R intends to seek help from its SPC for a resource intensive task. Agent R can first identify who are the members that can participate in the collaboration based on the RP. Afterwards, Agent R can define a workflow process definition document and then launch the workflow in which the participated SPC members will accomplish the task by following the workflow definition. Since

<http://www.w3.org/TR/ws-arch/>
each agent has remote cloud services, the task scheduled to the SPC member can be accomplished by the SPC member either using the hardware resource of mobile device or by its cloud utility service.

Note that it is possible for a member of SPC to be a stationary node. However, in this work, we do not assume there is always a resource-rich stationary node exist in the SPC for the stable message brokering. Hence, in the following content, we only focus on mobile participants.

2.2. SPiCa Framework

In order to enable the SPC environment, we propose the Social Private Cloud application framework—SPiCa. In SPC, each agent is a SPiCa node. SPiCa lets an agent establish its SPC and manage how it can delegate its computation tasks to its SPC using a cost-efficient scheme.

SPiCa is a Web service standard-based service-oriented architecture framework. It supports loose coupling and enables the auto-scalability of task offloading processes based on the resources derived from its SPC.

![Figure 2: SPiCa framework architecture](image)

Figure 2 illustrates the high-level architecture of SPiCa. It consists of three main groups: public cloud, SPC-DCN and SPiCa Node, which represents the agent that has SPiCa embedded in its device. Public cloud represents the remote cloud service resources that are associated with the SPiCa node. In general, the SPiCa node has cloud utility service for computation
offloading and cloud storage for data replication or data delegation. In this work, we do consider that the cloud utility service has limited capability and availability [14, 15]. If the cloud utility service is not feasible for task delegation, the task will be assigned to SPC. SPC-DCN represents the DCN of the SPiCa node. The right hand side of the figure shows the components of the SPiCa node, which are described below.

**Mediator** serves two functions:

- It manages the message routing between local host components in SPiCa node. For example, a request message handled by the Request Handling Module will be forwarded to Workflow Management Module (WMM). The WMM may also send request messages to States Manager (to be described in the later paragraph) via the Mediator in order to retrieve the required data for the design of workflow process definition.

- It also enables the interaction between the SPiCa node and its external resources. The interaction technology is based on standard Web service protocols. In our prototype, the communication protocol is via RESTful Web services.

**Request Handling Module (RHM)** handles the request from either the end-user’s application or other components. RHM sends the request message to WMM to find the corresponding Abstract Workflow for the request and the Abstract Workflow will be forwarded to the Coordinator. The Abstract Workflow describes a workflow in an abstract level in which the entities involved in workflow activities are not defined. Instead, for each activity, the Abstract Workflow only defines the input and output message types. The Activity Manager will define the entities for each activity during workflow execution, which is known as late binding in Web service domain.

**Communication Management Module (CMM)** manages the information needed for interacting with external resources. The information is described in a distributed hash table (DHT) that contains: the IP addresses of DCN members, the URL of their cloud services and the replicated service description metadata (e.g., WSDL) of DCN members’ resources. CMM may also consist of additional components for Trust, Privacy and Security management. These components are considered as future works that combine the schemes from [10] to support privacy and security in SPC to prevent malicious nodes from joining SPC or to determine abnormal activities in SPC.
Note that in this paper, we assume that we have applied one of the solution in the literature [16, 17] for the mobile IP addressing issue.

**Resource Management Module (RMM)** consists of:

- **Resource Manager** which provides the information needed to interact with local host resources and external resources. It maintains a list of Resource Profiles of DCN in SPC.

- **States Manager** which provides information regarding the hardware states of the mobile device and the states of cloud services. Resource Manager collects the states data from States Manager.

**Workflow Management Module** consists of:

- **Workflow Process Definition (WPD) Manager** which maintains a collection of WPD documents. It can define and retrieve a corresponding workflow document from local host dataset for a request message.

- **Coordinator** which is the controller of the workflow process. It monitors the states of workflow process and creates Activity Manager instances to handle workflow tasks.

- **Activity Manager** which handles an activity of workflow based on the resource availability. If the resources of the SPiCa node that the Activity Manager belongs to are not sufficient to handle the task, Activity Manager will collaborate with WPD Manager to define a task delegation WPD and execute the WPD as a sub-workflow.

3. Resource-Aware Adaptive Task Scheduling

In the following content of this paper, we will use Business Process Model and Notation (BPMN) [18] to describe the processes of the proposed system. BPMN was chosen because the core of SPiCa is a BPEL workflow and BPMN can also be used to describe BPEL workflow models [19].

The key contribution of this paper is the task delegation model for SPC, which is accomplished by the workflow-controlled resource-aware adaptive task-scheduling scheme. The proposed task-scheduling scheme is divided in two sections corresponding to two different types of workflow task scheduling for task delegation: sequential task delegation and parallel task delegation. The terms we used in the following paragraphs are:
• **Task**, which is handled by Activity Manager, is usually a sub-workflow (or is termed as a subprocess task in BPMN) in SPiCa because a task usually involves a set of sub-tasks such as configure a Web service client, use the Web service client to invoke a service provider to retrieve a file, deliver the file to a specific local host location using Mediator.

• **Activity**, which represents a runtime process of a workflow task. When a task is handled by a resource, it is called an activity.

### 3.1. Sequential Task Delegation

In sequential task delegation, a set of sequential tasks has been scheduled to DCN members. In case a powerful DCN member can handle the entire sequence, the tasks will be described in a single workflow and will be accomplished by a single DCN member. Conversely, the sequential tasks can be re-defined as a number of sub-workflows and each sub-workflow will be accomplished by different DCN member.

![Figure 3: Activities handled by local host components](image)

Figure 3 illustrates a partial workflow described in a BPMN choreography diagram. In this workflow, an agent N1 initiates a workflow, which consists of two main activities in WPD. AM1 (Activity Manager 1) uses COM1 (local host component 1) as the resource to process activity 1. After activity 1 is completed, AM1 sends the result generated by activity 1 to AM2 (Activity Manager 2). AM2 then uses COM2 (local host component 2) to process activity 2.

The above example shows a case in which all activities will be handled by local host components of the mobile device (N1). Figure 4 illustrates the case in which the activities have been scheduled to DCN members. As the figure shows, AM1 of N1 will be created for sending task to N2 in which AM1 of N2 will handle the activity 1 by using COM1 of N2. After the activity 1 is completed, AM1 of N2 will send the result to N3 in which AM1 of N3 will handle activity 2.
The two examples above lead to a question: *How does the coordinator decide which DCN member is suitable to handle the task?*

The following steps describe the proposed scheme for the scheduling of sequential task delegation.

**Step 1: Identify candidate participants**

**Definition 1: Task Profile**—TP. TP is a tuple \((ID, RP)\) where \(ID\) denotes the unique identification of the task; \(RP\) is a set of required resource properties for executing the task, in which \(RP = \{rp_j : 1 \leq j \leq N\}\). Each \(rp_j\) consists of resource property ID \((ID_j)\), the resource property type \((type_j)\) and the required resource property value \((v_j)\).

**Definition 2: Available Resource Property Profile**—AR. Each DCN member has an AR. AR = \(\{ar_k : 1 \leq k \leq N\}\). Each \(ar_k\) consists of resource property ID \((ID_k)\), resource property type \((type_k)\) and the available value of the resource property \((av_k)\), which is defined based on user preference.

Since each task requires a set of available resource properties, a DCN member can be assigned as the candidate participant for the task only when all the available resource properties of the DCN member match the minimum available resource property requirement of the task. To determine a DCN member is the candidate of a task or not, the following scheme is used.

Let \(H\) be a set of DCN members. \(H = \{h_l : 1 \leq l \leq N\}\). The AR of \(h_l\) is denoted by \(AR_l\). Let \(CH^i\) be the candidates of task \(i\). We generate \(CH^i\), the candidates of task \(i\) such that:

\[
CH^i = \{h_l \in H : \forall \forall av_k^l \in AR_l \\
\text{s.t. } av_k^l \geq v_j^i, \ ID_k^l \equiv ID_j^i\}
\]  

(1)

where \(v_j^i\) is the required resource property value of \(type_j^i\) for task \(i\). Once the \(CH^i\) is generated, the next step is to identify the best node for the task \(i\) by comparing the performance scores of candidates.
Step 2: Calculate the performance of helper

The performance scores are computed based on the process score and the transaction time score. The process score is based on the computation process power a candidate can provide. It is either provided by the mobile device hardware or by the cloud utility service of the candidate. The transaction time score is based on the overall transaction time for the entire process when the candidate is handling the task, the shorter the better.

Process Specification Score

Definition 3: Process Specification—$PS$. Each member of SPC has a $PS$ that represents its computation process power. $PS_l = \{ps^l_m : 1 \leq m \leq N\}$ denotes a set of elements of candidate $l$’s $PS$ where $ps^l_m$ consists of an identification of the element ($ID^l_m$); e.g., CPU and RAM and the value of the element ($v^l_m$).

Instead of using the specific number for each type of element (e.g., CPU=800MHz), SPiCa utilises Fuzzy Set [20] to normalise the element values in order to ease the comparison of the score of each participate node. The fuzzy value of an element $x$ (denoted by $\tilde{v}^x_m$) is computed by:

$$\tilde{v}^x_m = \frac{v^x_m}{\sum_{h_l \in CH} v^l_m} \quad (2)$$

Let $spec_x$ be the $PS$ score of candidate $x$, which is computed by:

$$spec_x = \sum_{ps^l_m \in PS_x} \sum_{ps^m \in PS} \tilde{v}^x_m \cdot w^m \quad (3)$$

where $w^m$ denotes the weight of $ps^m$. Workflow initiator can define the weight of resource elements based on certain rules. For example, the workflow initiator can define the weight of CPU resource as 0.4 (40%) and the weight of RAM as 0.6 (60%). If the rule was not defined, each resource element will have an equal weight.

Transaction Time Score
Let $P_{l,i}$ be a set of transaction path when task $i$ has been scheduled to candidate $l$. $P_{l,i} = \{p_{o}^{l,i} | 1 \leq o \leq N\}$. The maximum transaction time ($T_{l,i}$) for candidate $l$ to handle task $i$ is computed as below:

$$T_{l,i} \approx \max_{p \in P_{l,i}} \{T_{p_{o}}^{l,i}\}$$ (4)

where $T_{p_{o}}^{l,i}$ denotes the transaction time of path $p_{o}$.

Let $D_{i}$ be a set of maximum transaction times of all candidates when they are scheduled to handle task $i$. $D_{i} = \{T_{l,i} : 1 \leq l \leq N\}$. Let $L = \min \{T_{i}\}$. The na"ive transaction score of candidate $l$ for task $i$ ($NTS_{l,i}$) is calculated as below:

$$NTS_{l,i} = \begin{cases} 1 & \text{iff } T_{l,i} \equiv L \\ (L + 1) - T_{l,i} & \text{otherwise} \end{cases}$$ (5)

The normalised fuzzy value of the final transaction time score of candidate $x$ is computed as below:

$$TS_{x,i} = \frac{NTS_{x,i}}{\sum_{T_{l,i} \in D_{i}} NTS_{l,i}}$$ (6)

**Step 3: Overall performance score**

Finally, the overall performance score of candidate $x$ for task $j$ is computed as below:

$$perf_{x,i} = spec_{x} + TS_{x,i}$$ (7)

If a candidate has been computed as the best candidate for more than one task, the system needs to decide which task should be scheduled to the candidate. Let $G$ be a set of possible scheduling settings in which $G = \{g_{q} : 1 \leq q \leq N\}$. Each $g_{q}$ represents a setting regarding to which task is scheduled to the candidate. $g_{q}$ consists of a task ID ($tID_{q}$) and the overall performance score of the candidate when the task is scheduled to the candidate ($v_{q}$). Let $g_{Max} = \max_{g_{q} \in G} v_{q}$, and $ID_{g_{Max}}$ be the task ID of $g_{Max}$. We allocate the scheduling setting that gives the highest score for candidate $x$. Hence, the task that is to be scheduled to the candidate $x$ ($tID_{x}$) will be $ID_{g_{Max}}$.

### 3.2 Parallel Task Delegation

This section describes how parallel task delegation is scheduled in SPC. This approach is suitable when the task involves a large number of same type of transactions or processes with different sources. For example, a new peer
in MSNP [10] intends to retrieve a large number of reputation rating data (RRD) from the cloud storage of MSNP peers for determining the trustworthiness of a content provider. Since the process may cause the transaction overhead issue for the new peer, it is feasible to split the task into a number of parallel subtasks and delegate them to the new peer’s SPC.

The proposed scheme for parallel task delegation follows the following steps:

1. The task will be partitioned equally based on the minimum resource availability of the task participants. For example, the task participant who has the lowest performance score can handle three subtasks effectively, then the task will be partitioned based on such setting.

2. A task participant who has a higher performance score can handle multiple subtasks.

Suppose a task TA needs to be partitioned into N number of subtasks. Therefore, \( TA = \{ta_c : 1 \leq c \leq N\} \). \( ta_c \) requires a set of minimum available resource properties—\( E_c \). \( E_c = \{e^c_z : 1 \leq z \leq N\} \). \( v^c_z \) is the value of \( e^c_z \). \( type^c_z \) denotes the type of \( e^c_z \).

Let \( H \) be a set of DCN members (\( H = \{h_l : 1 \leq l \leq N\}\)) inherited from the previous description in last section in which each \( h \in H \) has an AR (see Definition 2). Therefore:

\[
e^c_z = \min_{h_l \in H} \{av^l_k \in AR_l : type^c_z \equiv type^l_k\} \quad (8)
\]

Based on (8), the \( E \) of \( ta \) can be generated, which is the minimum resource requirement of \( ta \).

The number of \( ta \) each \( h_l \) can handle depends on the resource availability given by the \( h_l \).

Let \( STA_l \) be the number of subtasks that need to be scheduled to a \( h_l \in H \).

\[
STA_l = \min_{e^c_z \in E_c, ar^l_z \in AR_l} \left\{ \frac{av^l_k}{v^c_z} \cdot \frac{1}{EX} \right\} \\
\text{s.t.} \quad k = 1...n; \quad z = 1...n; \quad type^l_k \equiv type^c_z \quad (9)
\]

where \( EX = 0...1 \). The \( EX \) number is required for supporting flexibility when considering a DCN member may not be reachable due to mobility-related issues (e.g., disconnected). In such a case, the corresponding task,
which has been assigned to the member, will be assigned to a different member. By applying \( EX \), each DCN member can be scheduled extra tasks if needed at runtime.

Finally, the total number of partition for \( TA \) (denoted by \( \text{size}_{TA} \)) will be \( \sum_{h_i \in H} STA_i \). Therefore, the task will be split equally into \( \text{size}_{TA} \) number of sub-tasks. Then, the task participant who have more resource availability can handle more than one \( ta \). The one that has least available resources will handle only one \( ta \). The task scheduling will be done asynchronously.

4. Case Study

In this section, we use a scenario to explain how SPiCa manages the task delegation in SPC. In the following paragraphs, we use BPMN to describe the processes in SPiCa.

Firstly, we explain the general behaviour of the Activity Manager. In order to enable dynamic task reconfiguration, each Activity Manager will follow the process workflow as shown in Figure 5.

As Figure 5 shows, when a task has been assigned to the Activity Manager, the Activity Manager will first retrieve resource information from RMM component (mark 1). Afterwards, the Activity Manager will identify whether the agent’s own resources can handle the task. If available resources are not sufficient to handle the task, the Activity Manager will use WPD Manager to create a new subprocess (a task delegation workflow; mark 3) and execute the subprocess. Otherwise the Activity Manager will directly execute the current workflow.

The scenario is based on a Mobile Social Network in Proximity (MSNP) [14, 15] environment (see Figure 6). In this scenario, a content advertiser peer (marked as Initiator in the figure) has found a number of content consumer
Figure 6: Scenario environment

(a) Original task

(b) Single task delegation as Sequential Delegation

(c) Original tasks from the Initiator

Figure 7: Workflow configuration
Figure 8: Resource-aware parallel task delegation
peers (CCP) who are potentially interested in the content it provides. Considering the mobility issue, all the CCPs prefer to receive advertisement via their cloud storage proxies, such a preference requires the Initiator to send the advertisement via the mobile Internet. Meanwhile, the Initiator realises that its resources are not sufficient to perform the advertisement process (e.g., unstable Internet connection speed). Hence, the Initiator decides to delegate the task to its SPC. In the scenario, the Initiator has only one DCN member (DCN_X) and three ICN members (ICN_1, ICN_2 and ICN_3) in its SPC. It can only delegate the entire process to DCN_X.

Figure 7a illustrates the original process, which has been defined as one subprocess activity in BPMN. The activity consists of 6 sub-tasks corresponding to the content advertisement processes with CCP. Each sub-task corresponds to one synchronised HTTP interaction.

Figure 7b illustrates the task delegation workflow. For the Initiator, two new workflow activities have been defined—Delegate task and Receive result—in order to delegate the original process to DCN_X.

As Figure 6 shows, DCN_X has three members in its SPC. Assume DCN_X has suddenly noticed that its battery-life is below 50%. Hence, DCN_X intends to further delegate the process to its SPC members, which are ICN of the Initiator.

Figure 7c illustrates the original workflow DCN_X receives from the Initiator. In order to delegate the process efficiently to SPC members, DCN_X reconfigures the workflow.

Figure 8 illustrates the new workflow created by DCN_X with tasks delegated to its SPC members. Suppose each member in DCN_X’s SPC has different resource availability. Based on the proposed schemes described in the previous section, DCN_X partitions the process into three workflows in which ICN_1 receives a workflow that contains only one task, ICN_2 handles two tasks and ICN_3 handles three tasks. Each ICN will send response messages to DCN_X after their tasks are completed.

5. Preliminary Experimental Evaluation

We have implemented and evaluated the SPiCa prototype for proof-of-concept. This section describes the details of the evaluation. The primary objective of the prototype evaluation is to validate the proposed resource-aware task delegation using BPEL workflow for SPC.
Since the task delegation/offloading between mobile device and cloud service has been covered in our previous works [14, 15], the following content focuses on the task delegation between the member nodes of SPC.

5.1. Implementation

The core of SPiCa is the mobile-device-hosted BPEL workflow engine and mobile Web service. The SPiCa prototype has been implemented as an iOS application with following main components:

- **BPEL workflow engine.** We have implemented a customised prototype BPEL workflow engine with the proposed resource-aware task scheduling mechanism. The XML-formatted BPEL documents are processed by using GDataXML\(^6\) class provided by Google. Currently, the prototype workflow engine is capable of processing \(<\text{sequence}/>, <\text{flow}/>, <\text{invoke}/>\) and \(<\text{assign}/>\) tags.

- **Mobile Web service provider.** SPiCa nodes are capable of communicating automatically. Such a mechanism can be realised by implementing the HTTP Web servers on them. We have utilised CocoaHTTPServer (yoojin version)\(^7\) to support such a need.

Other components were implemented using native libraries of iOS SDK. In order to evaluate the proposed resource-aware task delegation schemes, we deployed the SPiCa prototype on an iPhone 4S, an iPod Touch 4th generation (iTouch 4), an iPad 2 and five iPhone 4 devices. We used two different Internet connections—Wi-Fi and 3G. The speed of the WiFi was around 29Mbps at the time of testing, and the speed of 3G Internet was around 1Mbps.

5.2. Performance Score Analysis

The goal of this test is to demonstrate how SPiCa node measures the performance score of DCN. We used four SPiCa nodes deployed on an iPhone 4 (as the Initiator described in Section 4), an iTouch 4, an iPhone 4S and an iPad 2 (as the DCNs of the Initiator).

Figure 9 illustrates the performance score calculation result influenced by the importance weight of resources. In the setting of this testing, all

\(^6\)https://code.google.com/p/google-api-objectivec-client/

\(^7\)https://code.google.com/r/yoojin-cocoahttpserver-iphone/
the SPiCa nodes were using the WiFi Internet connection (with 29Mbps speed). As the figure shows, when different weights of the resources have been assigned, the performance score of each node can be changed. It means that when the process does not rely highly on the RAM, the score of iTouch 4 node can out-perform the iPhone 4. Hence, if the process does not rely highly on the RAM, delegating the process to the iTouch 4 node is also an efficient option, although such a case in the real world is less probable.

Within the three DCN, the iPad 2 node gives the best performance. It is because fundamentally, iPad 2 has the best hardware specification.

Figure 9: Performance score measuring

![Performance Score](image)

Figure 10: Performance of sequential task delegation

Figure 10 illustrates the timespan comparison when the process is performed by different nodes. In this test, we defined 50 tasks in BPEL workflow. Each task requires a synchronised HTTP communication with external Web service. In the figure, ‘Solo’ represents the process not delegated to any
(a) Parallel task delegation based on equal number of tasks

(b) Performance comparison between equally delegation and resource-aware strategy. (same network speed)

(c) Performance comparison between equally delegation and resource-aware strategy. (iPhone4S on 3G)

Figure 11: Task delegation performance
DCN. It shows that when the process highly involves network communication with the external service provider, delegating the process to the nodes that have similar hardware specification could not improve the performance much. However, considering the battery power consumption aspect, delegating the process to the node that has similar performance score can still save battery life.

5.3. Resource-Aware Task Delegation

This section describes the test based on the scenario in Section 4. In this test, the Initiator intends to send advertisement to the cloud storages of different number of MSNP content consumers. The process has been defined as a set of parallel tasks in BPEL workflow. At runtime, the Initiator will re-configure the workflow in order to delegate a different number of tasks to its DCN. In the setting, each task is given the same resource requirement.

The test consists of two parts. In the first part, we show how much the task delegation can improve the overall performance by partitioning the tasks equally based on the number of DCNs. In the second part, we show that how much resource-awareness can improve the overall performance compared to the equal-partition-based delegation.

Figure 11a illustrates the comparison between three different settings. ‘Solo’ denotes the entire process was performed by the Initiator without delegation. 2 DCN and 5 DCN respectively represent how many DCN members are involved in the task delegation. For example, when there are 10 parallel tasks defined in the workflow, in the 2 DCN setting, each DCN member will handle 5 tasks. We tested this use case by using different number of parallel tasks defined in the workflow. In this test, we only used iPhone 4 devices as DCN members. As the result shows, the more DCN members participate in the task delegation, the less timespan it requires.

In the second part, we compared the equal partition-based delegation with the resource-aware strategy. In this test, we used iPhone 4 as the Initiator and the three DCN members were deployed on iTouch 4, iPhone 4S and iPad 2. All the devices were connected to the Internet via WiFi. The resource-aware strategy is based on the proposed schemes described in Section 3. Figure 11b illustrate the performance comparison between the two delegation schemes. As the figure shows, the resource-aware strategy can slightly improve the overall performance. The difference between the two schemes is quite small because ultimately the task performance was highly affected by the Internet
connection speed, since all the devices were using the same WLAN for the Internet connection.

In order to further show the difference between the equal delegation and the resource-aware strategy explicitly, we changed the Internet connection of iPhone 4S to 3G network, which has an average speed of 1Mbps. Based on this setting, the result (see Figure 11c) clearly shows that the resource-aware delegation can improve the overall performance of the parallel task delegation.

5.4. Discussion

In the previous section, we described the prototype evaluation of SPiCa. The evaluation results show that the proposed resource-aware task delegation schemes can provide an efficient way to enable SPC.

One major concern of SPiCa is the battery consumption. We have performed a test on the battery usage of SPiCa node on an iPhone 4S. Figure 14 illustrates the battery usage comparison between the phone that is operating SPiCa in the background (With SPiCa) and the phone that is not operating SPiCa in the background (Default).

![Figure 12: Battery consuming comparison](image)

The fundamental idea of SPC is to let an Initiator delegate their tasks to their friends’ devices when their friends’ devices are not in-use and their battery status is in-recharge mode or the battery level is still high (e.g., 90%). However, we aim to move forward to such a pre-requisite that a new approach can effectively enable SPC without constraining the prerequisite status of the participant devices. We consider such an approach as one of our future research directions.
6. Related Works

6.1. Task Delegation in Mobile Cloud Applications

Mobile cloud task delegation can be classified into three types: distant, proximal and mobile peer-to-peer cluster.

In a classic design of distant task delegation/offloading, the computation tasks of mobile devices are delegated to one or more cloud services. In order to augment the resource management, a common approach is to utilise a central mediator service to organise the delegation process between the mobile application and the cloud services. Such an approach faces the potential single point of failure issue because it assumes that the mediator service is always ready and available.

Proximal-based task delegation derives from the concept of the virtual machine (VM)-based cloudlet [21]. In VM-based cloudlet environment, there are resource rich cloudlet machines situated in many WiFi areas. VM-based cloudlets can also interact with distant cloud services to enhance their computational power. Mobile applications can delegate their tasks to the nearby VM-based cloudlets instead of directly interacting with distant cloud services. Such an approach can improve the overall performance of task delegation processes. However, it can be costly to provide and maintain the VM-based cloudlet machines compared to the distant-based approach, which does not require extra physical machines.

Mobile peer-to-peer cluster (MP2P cluster)-based task delegation has been applied in the Extended Mobile Host Complex Web service Framework (EMHCWF) [1]. Instead of delegating tasks to resource rich desktop machines, the authors of EMHCWF proposed an approach to enable task delegation to a ‘mobile device cloud’ that is established by a cluster of mobile devices via the Internet. In order to profile the tasks for the computation offloading to a collaborated mobile cloud, the authors of EMHCWF have proposed Complex Partial Distribution (CPD) scheme. The CPD scheme is able to analyse the hierarchy of the service from the service description metadata towards decomposing the service into fragments.

SPiCa can be seen as an extension of EMHCWF with two additional features—standard-based workflow engine and the fuzzy set based resource-aware task scheduling. The two new features enhance the interoperability and flexibility of the MP2P cluster-based task delegation.
6.2. Mobile Cloud Workflow Systems

Workflow Management Systems (WfMS) have been applied in numerous Mobile Cloud Computing (MCC) applications [1, 14, 15, 22, 23] to provide efficient management of complex composite services. Workflows in MCC can be classified into two basic types:

- **User-Configurable Workflow (CWf)** refers to a workflow pattern being used as part of a request message to trigger a sequence of activities from a system. The workflow pattern is either pre-defined or is defined at runtime based on certain criteria.

- **Non-configurable Workflow (NWf)** refers to a static workflow predefined by the system administrator, such that the user has less control over the specification of the workflow itself.

While applications in each execution mode can be distinguished in terms of user configurability—as either CWf or NWf, based on the application domain, these workflows can be either executed from a remote cloud service or executed by a mobile application that has an embedded workflow engine. They are termed Remote Central Server as a Workflow Manager (RCWfM) and Mobile Requester as a Workflow Manager (MRWfM) respectively.

6.2.1. Remote Central Server as a Workflow Manager (RCWfM)

In a classic design of WfMS for MCC, a remote central server (RCS) will be used as the workflow manager for mobile client-side applications. We use the term—RCWfM to describe such a server. Workflows in RCWfM-based WfMS can be either CWf or NWf:

- **CWf**—RCWfM-based CWf has been applied in providing customisable Software as a Service (SaaS). SaaS provides an easy way for end-users not only to deliver the isolated applications, but it can also provide the advanced composite services to end-users in which users can design their own workflows to compose a set of remote Web service and to organise how the Web services behave in order to accomplish their goals without any programming skills. For example, the cloud-based development application—IVO [23] provides an environment which lets non-programmers create and use customised context-aware mobile cloud applications.

- **NWf**—The fundamental purpose of RCWfM-based NWf is the same as common Web-based composite services associated with workflow engines. An example is the offloading to nearby stationary machine in wireless local area
network approach that was proposed by [22] for a cloudlet-based computational offloading application. The primary objective of their work is to reduce energy consumption of mobile devices when the devices are participating in a workflow process initiated by a remote computer via the Internet. The authors considered wireless network communication as one of the main factors that influence the energy usage of mobile devices when they are participating in a workflow. Hence, the mobile devices can consider offloading the tasks to cloudlets, which are formed by computers provided by the WLAN providers. Cloudlets provide the virtual machine mechanism allowing mobile devices to offload their task.

6.2.2. Mobile Requester as a Workflow Manager (MRWfM)

Similarly, workflows in MRWfM-based WfMS can be either CWf or NWf.

**CWf**—A typical example of MRWfM-based CWf is AMSNP [14, 15]. In AMSNP, workflow engines are embedded in mobile devices. When the workflow engine receives the request, it will search for pre-defined corresponding workflow pattern in local memory and execute the workflow. Each workflow task is managed by a task agent component. When a workflow task is to be performed, the corresponding task agent will check the resource availability (e.g., CPU and RAM usage, battery life) of the mobile device. If the resources of mobile device have reached a certain level based on the proposed cost-efficient algorithm, the task will not be performed in the mobile device. Instead, the task will be offloaded to a cloud service that is capable of processing the task.

**NWf**—EMHCWF [1] is a typical example of the offloading to mobile peer-to-peer cluster via the Internet, which can be classified as MRWfM-based NWf. The framework was designed to manage the CMC in which multiple mobile hosts are involved in the workflow execution. On the other hand, the entities involved in AMSNP are only of two kinds: one mobile host (which is the requestor itself) and remote stationary cloud services.

Table 1 summarise the classification of mobile cloud workflow systems.

<table>
<thead>
<tr>
<th></th>
<th>RCWfM</th>
<th>RC/MRWfM</th>
<th>MRWfM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWf</td>
<td>IVO  [23]</td>
<td>SPiCa</td>
<td>AMSNP [14, 15]</td>
</tr>
<tr>
<td>NWf</td>
<td>VM Cloudlet [22]</td>
<td>n/a</td>
<td>EMHCWF [1]</td>
</tr>
</tbody>
</table>

Table 1 summarise the classification of mobile cloud workflow systems. Distinct from previous works, SPiCa is a hybrid system between RCWfM
and MRWfM in which every participating nodes in SPiCa-based SPC environment are capable of configuring and processing standard-based workflow tasks. Furthermore, since SPiCa is based on standard technologies, it can provide flexible scalability in terms of dynamically increasing/decreasing the number of participants.

7. Conclusion and Future Work

In this paper, we have proposed a standard-based workflow-controlled system for collaborating task delegation in mobile device established social private cloud environment. The proposed SPiCa framework utilises the resource-aware task delegation scheme to optimise process scheduling. The preliminary prototype evaluation has provided the proof-of-concept in which the proposed scheme can enhance the workflow based task delegation based on the runtime resource analysis.

Currently, we continue validating the SPiCa framework and intend to address the following issues in our future work:

- In this paper, we have not described how SPiCa profiles the more complex computational tasks into workflow patterns in detail. It is because we considered that task profiling requires further investigation as an individual research topic. Hence, in the future work, we aim to propose a proper solution for task profiling.

- As mentioned in Section 5.4, operating SPiCa can consume a lot of battery life of mobile devices. There is a need to reduce the energy consumption while the system can still remain in the decentralised topology to avoid single point of failure. We are investigating potential solutions such as cloud messaging.

- Although the standard-based BPEL workflow approach enhances the scalability and interoperability of the system, the XML based BPEL workflow is considered heavy weight. We are still studying different light-weight solutions that can provide a “lighter” workflow process environment and also support the best interoperability.

- We also consider to introduce a queuing management component in SPiCa to handle interactions among large number of mobile participants.
8. Acknowledgments

This research is supported by the European Regional Development Fund through the EXCS, Estonian Science Foundation grant PUT360 and Target Funding theme SF0180008s12.

References


