Service-Oriented Heterogeneous Resource Sharing for Optimizing Service Latency in Mobile Cloud

Takayuki Nishio
Graduate School of
Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku,
Kyoto, Japan
nishio@i.kyoto-u.ac.jp

Ryoichi Shinkuma
Graduate School of
Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku,
Kyoto, Japan
shinkuma@i.kyoto-u.ac.jp

Tatsuro Takahashi
Graduate School of
Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku,
Kyoto, Japan
ttakahashi@i.kyoto-u.ac.jp

Narayan B. Mandayam
Wireless Information Network
Laboratory (WINLAB),
Rutgers University
671 Route 1 South
North Brunswick, NJ, USA
narayan@winlab.rutgers.edu

ABSTRACT

Fog computing is expected to be an enabler of mobile cloud computing, which extends the cloud computing paradigm to the edge of the network. In the mobile cloud, not only central data centers but also pervasive mobile devices share their heterogeneous resources (e. g. CPUs, bandwidth, content) and support services. The mobile cloud based on such resource sharing is expected to be a powerful platform for mobile cloud applications and services. In this paper, we propose an architecture and mathematical framework for heterogeneous resource sharing based on the key idea of service-oriented utility functions. Since heterogeneous resources are often measured/quantified in disparate scales/units (e.g. power, bandwidth, latency), we present a unified framework where all these quantities are equivalently mapped to “time” resources. We formulate optimization problems for maximizing (i) the sum of the utility functions, and (ii) the product of the utility functions, and solve them via convex optimization approaches. Our numerical results show that service-oriented heterogeneous resource sharing reduces service latencies effectively and achieves high energy efficiency, making it attractive for use in the mobile cloud.

Categories and Subject Descriptors

C.2.4 [**Computer-Communications Networks**]: Distributed Systems

MobileCloud'13, July 29, 2013, Bangalore, India.
Copyright 2013 ACM 978-1-4503-2206-5/13/07 ...$15.00.

Keywords

cloud computing, mobile cloud, fog computing, heterogeneous resource sharing, service-oriented

1. INTRODUCTION

Recent advances of cloud computing platforms and mobile devices have given rise to mobile cloud computing (MCC), which is a new paradigm for mobile applications and services that promises to have a strong impact on our lifestyle. Initially, MCC mainly focused on improving computing power and storage capacity of mobile nodes by outsourcing tasks to more powerful cloud data center [1].

Mobile cloud architectures can be roughly classified into two types[2, 3]. The first type of architecture is the agent-client based architecture, where only a central data center provides resources (e.g. central processing unit (CPU) and storage) for mobile devices and processes the tasks necessary for implementing a service. In this setting, mobile devices just use cloud resources and do not contribute any services [4]. The second type of architecture is a cooperation-based architecture, where not only central data centers but also mobile devices share their resources and support services. Such architectures can be widely varied as well as the most powerful due to the sheer numbers of devices available to take part in the cloud. The underlying concept behind such architectures is also referred to as fog computing [5], which extends the cloud computing paradigm to the edge of the network.

The cooperation-based mobile cloud is the most interesting and visible research area of MCC at present. The recent performance advances and diversification of mobile devices bring a lot of ‘heterogeneous resources’ into local networks such as high-performance CPUs, high-speed Long Term Evolution (LTE) connections, high volume storages, and multiple-sensor information. These correspond respectively to computational resources, communication resources, storage resources, and information resources. These heterogeneous resources can be leveraged opportunistically making cooperation-based cloud computing enabled by advanced mobile devices, a powerful platform for supporting a wide variety of applications.

Several technical issues still need to be solved to fully realize the cooperation-based mobile cloud. In this work, we focus on the core problem of how to coordinate the sharing of heterogeneous resources between nodes. Conventional approaches for achieving heterogeneous resource sharing usually coordinate tasks (such as numerical calculations or even download of data) without consideration of the specific service that is being provided [2, 3]. In such task-oriented sharing, heterogeneous resources are often measured/quantified in disparate scales/units (e.g. power, bandwidth, latency), and tasks are allocated to optimize certain metrics based on the the disparate scales/units. Consider the example of a navigation service composed of the tasks of calculating the optimal route and downloading map images. Task-oriented sharing for such a service minimizes the processing (computing) time for each task. However, the navigation service has the peculiar feature that even if the route calculation is completed in just a few microseconds, users can not enjoy the service until they complete the downloading tasks. In this case, it makes no sense to optimize computational resources; perhaps the optimization would waste the computational resources even. Instead, what would have been better is the optimization of a service-oriented utility function that better captures the benefits of such optimization.

In this paper, we propose an architecture and mathematical framework for heterogeneous resource sharing based on the key idea of service-oriented utility functions. The proposed system model is shown in Figure 1. In our architecture, a resource coordinator orchestrates tasks and resources service-by-service to maximize the utilities of nodes for services. Since heterogeneous resources are often measured/quantified in disparate scales/units (e.g. power, bandwidth, latency), we present a unified framework where all these quantities are equivalently mapped to "time" resources. We formulate optimization problems for maximizing (i) the sum of the utility functions, and (ii) the product of the utility functions, and solve them via convex optimization approaches.

This paper is organized as follows. Section 2 introduces the system model as well service-oriented utility functions. The proposed numerical model is discussed in Section 3. We discuss optimization of resource sharing in Section 4. In Section 5, scenarios for numerical analysis are discussed, and numerical results are presented. We conclude the paper in Section 6.

2. SYSTEM ARCHITECTURE

2.1 Assumptions

We assume that nodes use services through applications installed in them. Nodes request resources service by service. A service is composed of multiple tasks such as computing and data downloading.

Figure 2 depicts an example of a task flow for a service. Nodes want to complete the tasks included in the service as soon as possible. To process tasks, nodes need to use their resources. The set of resources that node i has is given as $R_i = \{R_i^m : m \in M\}$, where the label m indicates the specific type of resource from the set of all available resources M. Further, the set of the sizes of tasks that node i has to process is given as $T_i = \{T_i^l : l \in L\}$, where the label l denotes the specific type of task from the set of all required tasks L. In other words, node i uses an appropriate amount of resources to accomplish the task l of size T_i^l. It is possible that accomplishing two different tasks may require the same type of resources in which case the specific type of resource has to be shared between these two.

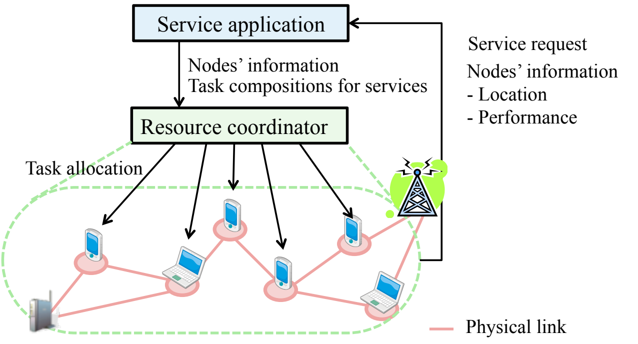
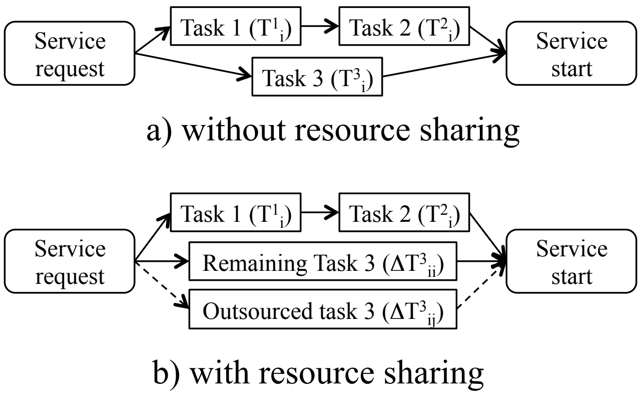


Figure 1: Service oriented mobile cloud with fog computing

Figure 2 (b) shows an example of services where node i requests additional resources and node j shares its resources with node i. A task of T_i^l can be separated into smaller portions and, when node j shares resources with node i, node i outsources some portions of the task to node j. The small portion of the task outsourced to node j is denoted as ΔT_{ij}^l ($\Delta T_{ij}^l \geq 0$). Node i has to process the remaining (not outsourced) tasks ΔT_{ii}^l. Note that the relationship between T_i^l and ΔT_{ij}^l is $T_i^l = \sum_{j \in N} \Delta T_{ij}^l$ where N is the set of all nodes. Outsourcing portions of a task reduces the processing time for the task accordingly.



a) without resource sharing

b) with resource sharing

Figure 2: Example of flow of tasks for service. Value in bracket is the amount of the task.

2.2 System model

An exemplary model of the architecture is shown in Figure 3 where neighboring mobile nodes are connected to other nodes and form a local network for resource sharing by using short-range wireless connections such as WiFi in ad-hoc mode and/or Bluetooth. Messages for resource sharing such as resource requests, task instructions, and the results for the tasks, are transmitted via the local network. Some of

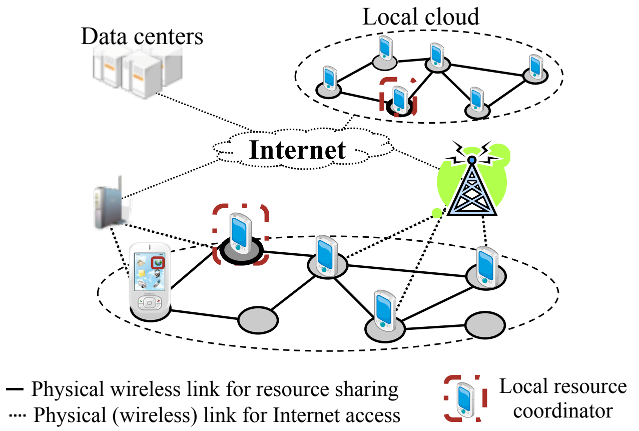

Data centers
Local cloud
Internet
— Physical wireless link for resource sharing
···· Physical (wireless) link for Internet access
Local resource coordinator

Figure 3: System model

the nodes have wireless Internet access using possibly 3G, LTE and WiFi connections.

Neighboring node in a local network form a group called a local cloud. Nodes share their resources with other nodes in the same local cloud. A local resource coordinator (LRC) is elected from the nodes in each local cloud. The coordinator manages resource requests and allocates tasks to the nodes in the local cloud or data center in the Internet if necessary. The coordinator should be elected in accordance with the connectivity to the local network, CPU performance and battery life time. In this paper, we focus on resource coordination and have thus omitted the algorithms for the formation of a local cloud and selecting the LRC. Further, we assume that the short range connections necessary to support formation of any networks for resource sharing are not bandwidth restrained. We also study resource allocation only under the scenario of sharing of computation resources, communication resources and information resources.

In our architecture, nodes can be resource users and resource providers. Figure 4 illustrates an example of messaging between two nodes and a coordinator. Here, suppose that node i and node j are requesting resources, and node i is outsourcing its tasks for node j in accordance with some coordination procedure. The messages are exchanged sequentially as follows: (1) the nodes send request messages to the coordinator; these messages include what and how much resources a node has, what and how much tasks the node has to complete for a service, and MAC and IP addresses of the node. (2) The coordinator allocates the tasks in order to maximize the utilities of nodes for the services, which is proposed in next section. The coordinator notifies each node what and how many tasks the node should process; (3) node i sends an instruction message to node j in accordance with the coordination result. The message includes necessary information to process tasks of node i; (4) node j processes the tasks in accordance with the instructions from node i; (5) node j sends the results of the tasks to node i; and (6) node i constructs the service based on the results of tasks.

2.3 Service-oriented unified utility function

We model the utility of nodes as a function of a service latency. The reason why we focus on service latency is that the latency is a straightforward measure of the decrease in the quality of experience for the service [6]. We define service latency as the duration from when the first task of a service being processed starts until all the tasks for that service are finished. We define the utility function of node i as $U_i(t_i)$, which is a monotonically decreasing function such as $-a \cdot t_i + b$, where t_i is the service latency of node i. Figure 5 shows the service latency and the utility as a function of outsourced tasks. As we can see in Figure 5, increasing the quantity of outsourced tasks increases the gain of the node for the resource sharing. Here, t_i depends on $R_i, T_i^l, \Delta T_{ij}^l, \Delta T_{ji}^l \quad (j \in N$, and $l \in L)$.

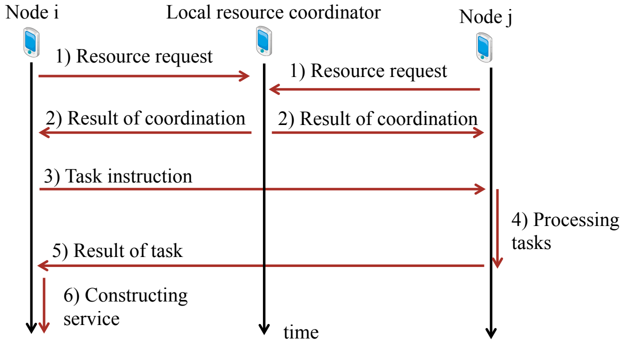


Figure 4: Coordination flow

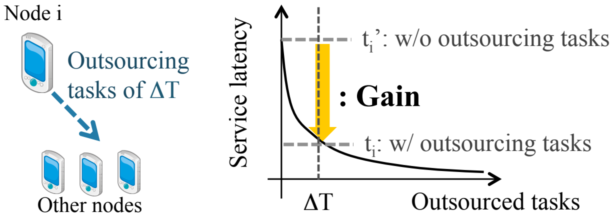


Figure 5: Gain of resource sharing

Using the unified utility function enables us to compare the value of heterogeneous resources in the same dimension of time. It could be a powerful metric for heterogeneous resources since we can treat any kind of resource as long as the resource can reduce the service latency. For example, a high-performance CPU reduces the latency for processing computational jobs, and high-speed wireless access reduces the latency for downloading data. A smart algorithm can reduce the latency for processing a computational task. Content can also be a resource, which can reduce the latency for obtaining the same content from the Internet. Sensor information can reduce latencies. For example, GPS information can be used to localize a node instead of other localization techniques that may be more computationally intensive and require completion of several communication tasks.

3. NUMERICAL MODEL FOR SERVICE-ORIENTED RESOURCE SHARING

In this section, we discuss the proposed architecture in terms of service latency and energy consumption, which are important factors for mobile users.

3.1 Service latency

Figure 6 shows the different scenarios possible for how tasks may need to be accomplished in providing a service. For a service (a), we can easily calculate the latency as the sum of the latencies for all tasks. For a service (b), we also can calculate the latency as the pointwise maximum of the latencies for all tasks. However, a service generally consists of multiple sequential tasks as shown in Fig 6 (c). Here, we define a sequence as a set of tasks which have to be processed in order. As shown in Figure 6 (d), we can define the service (c), identified as s, as $Q_s = \{Q^u_s : u \in U_s\}$, where the label u indicates the specific identification of the sequence from the set of all sequences included in the service U_s. Since the latency of each sequence can be calculated as the sum of the latency for processing the tasks included in the sequence, the service latency of node i t_i is written as

$$t_i = \max_{u \in U_s}\Big(\sum_{l \in Q^u_s} t^{m_l}_i\Big), \qquad (1)$$

where $t^{m_l}_i$ is the latency required for node i to complete task l when using the resources appropriate for the task $R^{m_l}_i$.

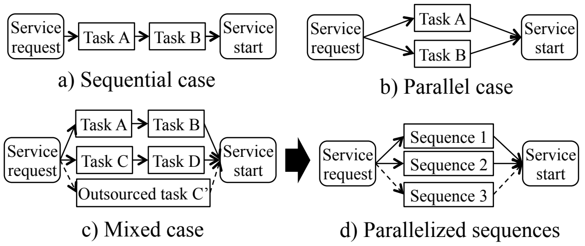


Figure 6: Task sequences for service

The $t^{m_l}_i$ should be defined appropriately for the resources m_l. For computational and communication resources, we define the latency as $T^l_i / R^{m_l}_i$. This definition of latency is accurate, since, for example, the processing time of computational operations T^l_i when using a CPU that performs $R^{m_l}_i$ operation per second, and the downloading time of a data size of T^l_i with throughput of $R^{m_l}_i$ bps could be calculated using the definition.

Next, we discuss information resources. We regard these resources as alternative resources to communication and computational resources. For example, when a map image is required for a service, if a node caches the image, the node can use it without downloading the image from the Internet. Suppose that in a navigation service, if a node caches a route calculation result, the node can leverage that result instead of calculating the route. If node i has alternative information for a task T^l_i, the node can finish the task without processing the task using resources $R^{m_l}_i$. Let us define

$$\delta^{li}_j = \begin{cases} 0 & (I^l_i \subseteq R^{info}_j); \\ 1 & (I^l_i \not\subset R^{info}_j), \end{cases} \qquad (2)$$

where I^l_i is an alternative information resource for task l and R^{info}_j denotes the set of all information resources for node j. Using Eq. (2), we define the latency for a task as

$$t^{m_l}_i(T^l_i) = \delta^{li}_i \frac{T^l_i}{R^{m_l}_i}. \qquad (3)$$

3.2 Procedure for sharing resources

As shown in Figure 4, in order to share resources, nodes have to exchange messages and process tasks for other nodes. For example, suppose that node i outsources a set of tasks ΔT^l_{ij} to node j. The instruction message from node i instructs node j how to process each task in the set and node j processes the tasks using its resources $R^{m_l}_j$. When it finishes processing the tasks, node j sends the results of the tasks back to node i. We define the latency for an outsourced task t^l_{ij} as the duration from when the first message is sent until the result is received.

The latency t^l_{ij} is written as

$$t^l_{ij} = t^{m_l}_j(\Delta T^l_{ij}) + t_t(D^l_{ins}(\Delta T^l_{ij})) + t_t(D^l_r(\Delta T^l_{ij})) + \Delta t^{m_l}_j, \qquad (4)$$

where $t_t(D)$ indicates the time needed to transmit a message via short-range communications, the length of which is D; $D^l_{ins}(\Delta T^l_{ij})$ is the data size required for instructing the node how to process the task ΔT^l_{ij}; $D_r(\Delta T^l_{ij})$ is the data size of results for task ΔT^l_{ij}; $\Delta t^{m_l}_j$ is an additional latency that depends on when node j starts to process task T^l_{ij}. In this paper, we simply define them as

$$D^l_{ins}(T^l_{ij}) = w^l_{ins}, \quad D^l_r(T^l_{ij}) = w^l_r T^l_{ij}, \quad \text{and} \quad t_t(D) = D/\theta, \qquad (5)$$

where w^l_{ins} and w^l_r are weight parameters and θ is the throughput of a short-range wireless link.

Outsourced tasks and the other remaining tasks can be processed in parallel. Then, the latency for task l becomes

$$t^{m_l}_i = \delta^{li}_i \cdot MAX(t^{m_l}_i(\Delta T^l_{ii}), t^l_{i1}, t^l_{i2}, ..., t^l_{iN}), \qquad (6)$$

3.3 Trade-off between gain and energy consumption

As we mentioned in Section 1, we intuitively know that sharing resources with nodes close to you should be more efficient and effective than sharing with distant nodes.

There is a trade-off between the gain from cooperating with distant nodes and the energy consumption. Increasing the number of cooperating nodes might enable the nodes to obtain a greater gain, since the nodes would share a larger amount of remote resources. To increase the number of cooperating nodes, the nodes must improve their communication range. Suppose that there are three nodes i, j and k. The distances between each pair of nodes are d_{ij}, d_{jk}, and d_{ik}, respectively. Here, d_{ik} and d_{jk} are larger than d_{ij}. In order for node i to share resources with node j, node i has to transmit messages to node j. In that case, the communication range of node i has to be longer than d_{ij}. If node i and j cooperate with node k, their communication ranges have to be longer than d_{ik} and d_{jk} respectively. Nodes i and j can obtain greater gain by sharing resource with node k, while they require a larger energy for transmission to obtain more communication range.

Energy is consumed when nodes send/receive data. The values $E_{tx}(D, d)$ and $E_{rx}(D)$ represents the consumed energy in sending and receiving a message, where D is the data size of sent or received data, and d_r is the required communication range to share resources with a node, which is almost the same as the geographical distance between a sender and a receiver. $E_{tx}(D, d)$ and $E_{rx}(D)$ monotonically increase as D and d increase. In particular, they increase exponentially as d increases while maintaining the signal to noise ratio (SNR) [7]. Energy is also consumed when nodes process tasks. $E^{m_l}(T^l)$ is the consumed energy for processing the task T^l while using resource m_l. It also monotonically increase as the amount of tasks increases. Here, we simply define $E^{m_l}(T^l) = e_{m_l} \cdot T^l$, $E_{rx}(D) = e_{rx} \cdot D$, $E_{tx}(D, d) = e_{tx} \cdot (d)^2 \cdot D$, where e_{m_l} indicates the energy consumed in a second for fully usage of the resource m_l; e_{rx} indicates the energy consumed to receive a data size of 1 Byte; e_{tx} indicates the energy consumed to send a data size of 1 Byte to a node distance of 1 m. Then, the energy consumed by node i when sharing resources with node j is

$$\begin{aligned} E_{ij} &= \sum_{l \in L_{ji}} \{e_{rx} w^l_{ins} + e_{m_l} \Delta T^l_{ji} + e_{tx} (d_{ij})^2 w^l_r \Delta T^l_{ji}\} \\ &\quad + \sum_{l \in L_{ij}} \{e_{tx} (d_{ij})^2 w^l_{ins} + e_{rx} w^l_r \Delta T^l_{ij}\}, \end{aligned} \tag{7}$$

where L_{ij} is the set of labels for tasks outsourced to node j by node i. In addition, node i has to complete the remaining tasks. Then, the total energy consumption of node i becomes

$$E_i = e_{m_l} \sum_{l \in L} (\Delta T^l_{ii}) + \sum_{j \in N, j \neq i} E_{ij}. \tag{8}$$

4. RESOURCE SHARING OPTIMIZATION

We consider a case where a coordinator ideally knows all information such as R^m_i and T^l_i. The coordinator instructs every node to allocate their tasks to other nodes in order to i) maximize the sum of the gains of all nodes or ii) maximize the product of gains of all node. First we discuss the general case. Then, we explicitly formulate an optimization problem for a specific case where two nodes shares resources.

4.1 Objective

4.1.1 *Maximizing sum of gains in utility*

The simplest objective is to maximize the sum of the gains of all nodes. It is written as

$$\text{objective:} \quad \max_{\Delta T_{ij}\ (i,\, j \in N)} \sum_{i \in N} (U_i(t_i) - U_i(t'_i)), \tag{9}$$

where $\Delta T_{ij} = \{\Delta T^l_{ij} : l \in L\}$ and t'_i is a service latency when node i does not share resources. If we can define the utility function as $U_i(t_i) = -a \cdot t_i + b \quad (t_i > 0)$, the problem to maximize the sum of the gains in utility of all nodes is equal to the problem of maximizing the sum of reduced service latencies. Then, the objective can be written as,

$$\text{objective:} \quad \max_{\Delta T_{ij}\ (i,\, j \in N)} \sum_{i \in N} (t'_i - t_i). \tag{10}$$

As described in [9], a nonnegative, nonzero weighted sum of convex (concave) functions is convex (concave). Then, if t_i is convex or concave for ΔT_{ij}, the objective is concave or convex for ΔT_{ij}, respectively.

4.1.2 *Maximizing Product of gains in utility*

An objective of the optimization problem should be

$$\text{objective:} \quad \max_{\Delta T_{ij}\ (i,\, j \in N)} \prod_{i \in N} (t'_i - t_i). \tag{11}$$

This objective is based on the idea of the Nash bargaining solution, which is a solution that brings Pareto efficiency and proportional fairness. However, the objective is not necessarily convex, which makes the optimization problem hard to solve.

4.2 Constraints

Next, we discuss constraints. In this paper, we consider constraints for task amounts and incentives on service latency and energy consumption. The constraints are written as:

subject to:

$$T^l_i = \sum_{j \in N} \Delta T^l_{ij} \text{ (for any i and l)}, \tag{12}$$

$$\Delta T^l_{ij} \geq 0 \text{ (for any i,j and l)}, \tag{13}$$

$$E^{th}_i > E_i - E'_i \text{ (for any i)}, \tag{14}$$

$$t'_i > t_i \text{ (for any i)}, \tag{15}$$

where E^{th}_i is threshold for energy consumption and E'_i is energy consumption of node i without resource sharing.

Eqs. (12) and (13) mean that the size of tasks should be positive, and nodes cannot outsource more tasks than the nodes have. Eq. (14) and Eq. (15) mean that node i is motivated to join the resource sharing only if its additional energy consumption is kept smaller than its threshold E^{th}_i and if the resource sharing reduces its service latency. Why we consider the constraints is that we can expect that nodes do not share their resources without ensuring an incentive for the sharing. Resource sharing can reduce not only service latency but also energy consumption [8]. These reductions can be incentives for resource sharing, but we expect that these two do not go together since there are trade-offs, as discussed in Section 3.3.

From Eq. (8), E_i and $\sum_{j \in N} \Delta T^l_{ij}$ are the sum of an affine function. Then constraints except for t_i are evidently convex or concave for ΔT^l_{ij}. If we use objective (i) and t_i is convex or concave, the optimization problem becomes a convex optimization problem, which can make optimization easier than the general case since any local solution must be a global solution.

4.3 Case study: Resource sharing in two-node case

We discuss a simple case where a node shares computational, communication and information resources with another node to minimize the latency for a service composed of a computational task and a communication task.

We consider a scenario where two nodes 1 and 2 share their computational and communication resources, the amounts of which are indicated by R^1_i and R^2_i respectively (i = 1 or 2). A common service requested by nodes 1 and 2 consists of a computational task and a communication task, the amounts of which are T^1_i and T^2_i, respectively (i = 1 or 2.) The latency of each task is assumed to be T^l_i / R^l_i as described in Section 3.1.

Here, we assume that these two tasks can be processed in parallel. In this case, as discussed in Section 3.1, the service latency of node i when they do not share resources becomes

$$t_i = MAX(\delta^{1i}_i \frac{T^1_i}{R^1_i}, \delta^{2i}_i \frac{T^2_i}{R^2_i}). \tag{16}$$

From Eqs. (5) and (6), when node i and j share their resource, the latency for a task l of node i is

$$\begin{aligned} t^l_i &= \delta^{li}_i MAX(\frac{T^l_i - \Delta T^l_{ij} + \Delta T^l_{ji}}{R^l}, \\ &\quad t_t(D_{ins}(T^l_{ij})) + \delta^{li}_j \frac{T^l_{ij}}{R^l_i} + t_t(D_r(T^l_{ij}))). \end{aligned} \tag{17}$$

In this scenario, from (17), either ΔT^l_{ij} or ΔT^l_{ji} should be 0 at the optimal equilibrium. We define Δ^l_{ij} as

$$\Delta^l_{ij} = \begin{cases} \Delta T^l_{ij} & (\Delta T^l_{ji} = 0); \\ -\Delta T^l_{ji} & (\Delta T^l_{ij} = 0). \end{cases} \tag{18}$$

Then

$$\Delta T^l_{ij} - \Delta T^l_{ji} = \Delta^l_{ij}. \tag{19}$$

From Eq. (19), Eq. (17) can be described as

$$t^l_i(\Delta^l_{ij}) = \begin{cases} \delta^{li}_i MAX\{\frac{T^l_i - \Delta^l_{ij}}{R^l_i}, \\ \frac{w_{ins}\Delta^l_{ij}}{\theta} + \delta^{li}_j \frac{\Delta^l_{ij}}{R^l_i} + \frac{w_r \Delta^l_{ij}}{\theta}\}. & (\Delta^l_{ij} > 0); \\ \delta^{li}_i \frac{T^l_i - \Delta^l_{ij}}{R^l}. & (\Delta^l_{ij} \leq 0). \end{cases}$$

This function is convex, since, as proved in [9], if f_1 and f_2 are convex functions then their pointwise maximum f, defined by $f(x) = max\{f_1(x), f_2(x)\}$, is also convex.

Next, we discuss the energy consumption. From Eq. (8), the energy consumption of node i when sharing resources is

$$E^l_i = \begin{cases} \delta^{li}_i\{e^l \cdot (T^l_i - \Delta^l_{ij}) + e_{tx} \cdot (d_{ij})^2 \cdot w_{ins}\Delta^l_{ij} \\ +e_{rx} \cdot w_r \Delta^l_{ij}\} & (\Delta^l_{ij} > 0); \\ \delta^{li}_i e^l \cdot (T^l_i - \Delta^l_{ij}) - e_{rx} \cdot w_{ins}\Delta^l_{ij} \\ -e_{tx} \cdot (d_{ij})^2 \cdot w_r \Delta^l_{ij} & (\Delta^l_{ij} \leq 0). \end{cases}$$

The function is continuous and obviously convex for $\delta(I^l_i, R^{info}_i)$ $= 0$ or 1.

Using the above functions, we can write the optimization problem as:

$$\text{objective:} \quad \max_{\Delta^1_{ij}, \Delta^2_{ij}} G_i + G_j \tag{20}$$

subject to:

$$\begin{aligned} G_i &= MAX(\delta^{1i}_i \frac{T^1_i}{R^1_i}, \delta^{2i}_i \frac{T^2_i}{R^2_i}) - MAX(t^1_i(\Delta^1_{ij}), t^2_i(\Delta^2_{ij})), \\ G_j &= MAX(\delta^{1j}_j \frac{T^1_j}{R^1_j}, \delta^{2j}_j \frac{T^2_j}{R^2_j}) - MAX(t^1_j(-\Delta^1_{ij}), t^2_j(-\Delta^2_{ij})), \\ G_i, G_j &\geq 0, \\ T^l_i &\geq \Delta^l_{ij} \geq -T^l_j \quad (\text{l} = 1 \text{ and } 2), \\ E^{th}_i &> E^1_i - e_1 T^1_i + E^2_i - e_2 T^2_i. \end{aligned}$$

$MAX(t^1_i(\Delta^1_{ij}), t^2_i(\Delta^2_{ij}))$ is convex.[1] Then, G_i and G_j are evidently concave functions. As described in [9], a nonnegative, nonzero weighted sum of convex (concave) functions is convex (concave). Then, $G_i + G_j$ is concave. As mentioned above, other constraints are also convex. Therefore, the optimization problem is a convex optimization problem.

While we have discussed the 2-node case with 2 tasks thus far which lends itself to formulation as convex optimization, the extension to the general case of N-nodes with L-tasks may not necessarily yield such convex optimization formulations. We instead outline a heuristic approach below that will be an aspect of further study. Specifically, a heuristic for the N-node optimization problem can be to use subsets of the 2-node optimization problems, where these subsets may be chosen according to various criteria as described below: minimizing the average geographical distance between nodes, maximizing the average reduced service latency, or maximizing the product of reduced service latencies.

5. NUMERICAL EXAMPLES

5.1 Scenario setup

To illustrate our approach, we consider a scenario where nodes share their communication and computational resources. The nodes want to use a navigation service. For the service, node i has to download load information and map images, and calculate the optimal route. We assume here that the load information is already stored in the nodes. Then a latency for downloading the load information is 0. The total data size of the map images is T^1_i and the number of operations for calculating the route is T^2_i. Node i downloads data at a throughput of R^1_i and calculates computational tasks at the speed of R^2_i. We assume that nodes can process the image downloading and route calculation simultaneously. In this case, an optimization problem for the resource sharing becomes the same as that of (20) when $\delta^{li}_i = \delta^{lj}_j = 1$ for any l.

Here, we assume that the local resource coordinator mentioned in Section 2.2 ideally solves the optimization problem discussed in Section 4 and determines the optimal allocation of tasks; however, the constraints for energy consumptions are not considered in order to observe the adverse effects of geographical distance. We solve this non linear optimization problem by using the generalized reduced gradient technique [10].

We describe the parameters used in the study in the captions of specific figures. These parameters are chosen considering the performance of actual CPUs and wireless access using 3G, LTE, and Wi-Fi. [11, 12, 13]

5.2 Comparative Evaluation

We compare our service-oriented approach with a numerical upper bound and a task-oriented approach.

5.2.1 Numerical upper bound

The numerical upper bound is obtained when the sum of reduced service latencies is maximized without considering incentives for nodes. In particular, in this scenario, we solve the following optimization problem for 2 nodes i and j as:

$$\text{objective:} \quad \max_{\Delta^1_{ij}, \Delta^2_{ij}} G_i + G_j \tag{21}$$

subject to:

$$\begin{aligned} G_i &= MAX(\frac{T^1_i}{R^1_i}, \frac{T^2_i}{R^2_i}) - MAX(t^1_i(\Delta^1_{ij}), t^2_i(\Delta^2_{ij})), \\ G_j &= MAX(\frac{T^1_j}{R^1_j}, \frac{T^2_j}{R^2_j}) - MAX(t^1_j(-\Delta^2_{ij}), t^2_j(-\Delta^2_{ij})), \\ T^l_i &\geq \Delta^l_{ij} \geq -T^l_j \quad (\text{l} = 1 \text{ and } 2). \end{aligned}$$

5.2.2 Task-oriented

In the scenario of a task-oriented method, the coordinator first allocates communication tasks for minimizing the average latency for downloading map images and then allocates computational tasks for minimizing the average latency for calculating the optimal route. Then, the task-oriented method first solves the optimization problem of Eq.

[1] We define $t^1_i(\Delta^1_{ij}, \Delta^2_{ij}) = t^1_i(\Delta^1_{ij}) + 0 \cdot \Delta^2_{ij}$ and $t^2_i(\Delta^1_{ij}, \Delta^2_{ij}) = t^2_i(\Delta^2_{ij}) + 0 \cdot \Delta^1_{ij}$. These functions are convex since they are the sum of convex functions and zero. Then their pointwise maximum of $MAX(t^1_i(\Delta^1_{ij}, \Delta^2_{ij}), t^2_i(\Delta^1_{ij}, \Delta^2_{ij}))$ also has to be convex. Since $MAX(t^1_i(\Delta^1_{ij}, \Delta^2_{ij}), t^2_i(\Delta^1_{ij}, \Delta^2_{ij}))$ is equal to $MAX(t^1_i(\Delta^1_{ij}), t^2_i(\Delta^2_{ij}))$, $MAX(t^1_i(\Delta^1_{ij}), t^2_i(\Delta^2_{ij}))$ is also convex.

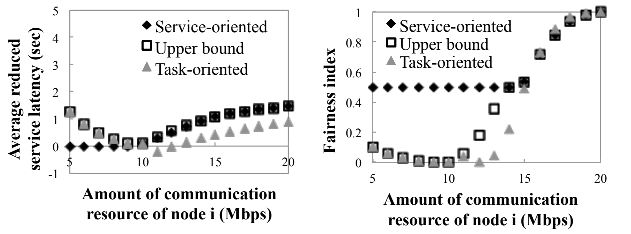


Figure 7: Average reduced service latency and fairness index as a function of communication resource of node i. Nodes i and j share computational and communication resources. Parameters are as follows: d_{ij} is 1 m; T^1 = 5 MByte; R_i^1 is variable; R_j^1 is 10 Mbps; T^2 is 50 G floating-point operation; R_i^2 is 12 G floating-point operation per second (FLOPS); R_j^2 is 27 G FLOPS.

(22) with $l = 1$ and determines Δ_{ij}^1. After that, it solves the optimization problem of Eq. (22) with $l = 2$ and determines Δ_{ij}^2.

$$\text{objective:} \quad \max_{\Delta_{ij}^l} G_i^l + G_j^l, \tag{22}$$

$$\begin{aligned} \text{subject to:} \quad G_i^l =& \;\; T_i^l / R_i^l - t_i^l(\Delta_{ij}^l), \\ G_j^l =& \;\; T_j^l / R_j^l - t_j^l(-\Delta_{ij}^l), \\ T_i^l \geq& \;\; \Delta_{ij}^l \geq -T_j^l. \end{aligned}$$

5.3 Numerical results

5.3.1 *Basic performance*

We observe the average reduced service latency and a fairness index. The fairness index is defined as

$$F(G) = \frac{(\sum_{i \in N} G_i)^2}{|N| \sum_{i \in N} G_i^2}, \tag{23}$$

where $G = \{G_i : i \in N\}$. The fairness index becomes lower than $1/N$ only when Δt_i $(i \in N)$ includes a negative value, which means the resource sharing works to the detriment of some nodes.

Figure 7 plots the average reduced service latency and fairness index as a function of the communication resource of node i in the two node case. As we can see in Figure 7, when node i does not have enough communication resources, nodes do not share any resources in the service-oriented method. This is because, as we can see from the result of the fairness index, when node i has few communication resources, node j increases its service latency by providing both its computational and communication resources to node i. When node i has communication resources grater than 10 Mbps, the average reduced service latency with resource sharing using the service-oriented method is almost the same as the upper bound and is larger than that of the task-oriented method.

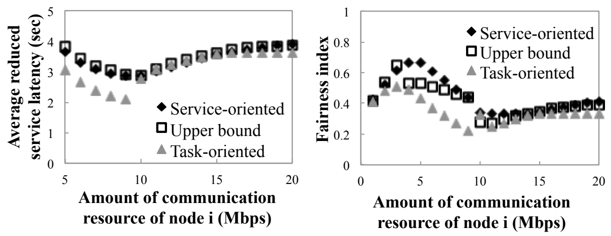


Figure 8: Average reduced service latency and fairness index as a function of communication resource of node i. Three nodes share computational and communication resources. Parameters are as follows: d_{ij}, d_{jk} and d_{ik} are 1 m; T^1 = 5 MByte; R_i^1 is variable; R_j^1 and R_k^1 are 10 Mbps; T^2 is 50 G floating-point operation; R_i^2 is 12 G FLOPS; R_j^2 is 27 G FLOPS; R_k^2 is 4 G FLOPS;

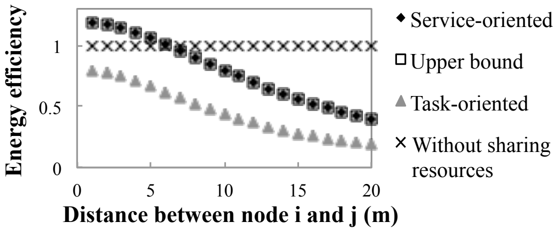


Figure 9: Energy efficiency as a function of distance between nodes i and j. Two nodes share computational and communication resources.

Next, we consider a 3-node example,[2] where nodes i, j, and k share their communication and computational resources. Figure 8 plots the average reduced service latency and fairness index as a function of the communication resource of node i. In this case, the average reduced service latency with the service-oriented method is almost the same as that of the upper bound and is larger than that of the task-oriented method. The fairness index of the service-oriented method is always larger than that of other methods. This result suggests that, as discussed in Section 3.3, if the number of nodes increases, the service-oriented method would reduce service latency more than the other methods without reducing the fairness index.

5.3.2 *Energy consumption*

Next, we discuss the effect of geographical distance. We define the energy efficiency as

$$\frac{(\sum_{i \in N} t_i') \cdot (\sum_{i \in N} E_i')}{(\sum_{i \in N} t_i) \cdot (\sum_{i \in N} E_i)}, \tag{24}$$

where E_i is the energy consumed for a service of node i while sharing resources. $\Delta t_i'$ and E_i' are the service latency and the energy consumption for a service of node i without sharing resources, respectively. If the energy efficiency becomes lower than 1, nodes should not share their resources. Figure 9 plots the energy efficiency as a function of distance between nodes i and j. We can see that the service-oriented method is more energy-efficient than the task-oriented one. However, when the distance becomes large, energy efficiency of the service-oriented method drops lower than 1. This means that, as we mentioned in Section 3.2, we should limit the sharing of resources while considering geographical distances.

[2] We easily extend two node case to three node case by considering following simple assumption: when node i receives two tasks Δ_{ji}^l and Δ_{ki}^l, the latency of these tasks are $t_{ji}^l = t_{ki}^l = (\Delta_{ji}^l + \Delta_{ki}^l)/R_i^{m_l}$.

6. CONCLUSION

In this paper, we proposed an architecture and mathematical framework for heterogeneous resource sharing based on the key idea of service-oriented utility functions. Since heterogeneous resources are often measured/quantified in disparate scales/units (e.g. power, bandwidth, latency), we presented a unified framework where all these quantities were equivalently mapped to ”time” resources. We formulated optimization problems for optimizing various metrics based on such service-oriented utility functions and solved them via convex optimization approaches. Our numerical results show that the service-oriented heterogeneous resource sharing reduces service latencies effectively and achieves high energy efficiency, making it attractive for use in the mobile cloud. While the numerical scenarios studied in the paper have considered only the cases of 2 or 3 nodes, the approach outlined here is extensible to larger sets of nodes. The problems of scalability and distributing these algorithms as the number of nodes increase are of interest as topics of future study.

Acknowledgements

This work is supported in part by the National Institute of Information and Communications Technology (NICT), Japan, under the new generation network R&D program for innovative network virtualization platform and its application.

7. REFERENCES

[1] “Mobile Cloud Computing Forum,” http://www.mobilecloudcomputingforum.com/.

[2] Guan, L., Ke, X., Song, M., and Song, J., 2011, A survey of research on mobile cloud computing, Proc. of the 10th IEEE/ACIS International Conference on Computer and Information Science, pp. 387-392.

[3] Dinh, H.T., Lee, C., Niyato, D., and Wang, P., 2011, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Communications and Mobile Computing.

[4] Bifulco, R. , Brunner, M., Canonico, R., Hasselmeyer, P., and Mir, F., 2012, Scalability of a mobile cloud management system, Proc. of the first edition of the MCC workshop on Mobile cloud computing, p. 17.

[5] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S., 2012, Fog computing and its role in the Internet of things, Proc. of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16.

[6] Reichl, P., Egger, S., Schatz, R., and D’Alconzo, A., 2010, The logarithmic nature of QoE and the role of the weber-fechner law in QoE assessment, Proc. of IEEE International Conference on Communications, pp.1-5.

[7] Friis, H.T., 1946, A note on a simple transmission formula, Proc. of the Institute of Radio Engineers.

[8] Kumar, K., 2010, Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?, Computer, vol. 43, no. 4, pp. 51-56.

[9] Boyd, S., and Vandenberghe, L., 2004, Convex Optimization,” Cambridge, U.K.: Cambridge Univ. Press, 2004.

[10] Lasdon, L.S., Waren, A.D., Jain, A., and Ratner, M., 1978, Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming, ACM Transactions on Mathematical Software, vol. 4, no. 1, pp. 34-50.

[11] Henning, J.L., 2000, SPEC CPU2000: measuring CPU performance in the New Millennium, Computer, vol. 33, no. 7, pp. 28-35.

[12] Mahesri, A. and Vardhan, V., 2005, Power Consumption Breakdown on a Modern Laptop, Proc. of the 4th International Workshop on Power-Aware Computer Systems, vol. 3471, pp. 165-180.

[13] Balasubramanian, N., Balasubramanian, A., and Venkataramani, A., 2009, Energy consumption in mobile phones, Proc. of the 9th ACM SIGCOMM conference on Internet measurement conference, pp. 280-293.