IPP Hurray!

www.hurray.isep.ipp.pt

Technical Report

QoS-based Surrogates Selection and Service Proposal Formulation in Offloading Environments

Luis Nogueira

Luis Miguel Pinho

TR-050703

Version: 1.0

Date: July 2005

QoS-based Surrogates Selection and Service Proposal Formulation

Luis NOGUEIRA, Luis Miguel PINHO

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: {luís, lpinho}@dei.isep.ipp.pt

http://www.hurray.isep.ipp.pt

Abstract

Resource constraints are becoming a problem as many of the wireless mobile devices have increased generality. Our work tries to address this growing demand on resources and performance, by proposing the dynamic selection of neighbor nodes for cooperative service execution. This selection is infuenced by user's quality of service requirements expressed in his request, tailoring provided service to user's specic needs. In this paper we improve our proposal's formulation algorithm with the ability to trade off time for the quality of the solution. At any given time, a complete solution for service execution exists, and the quality of that solution is expected to improve overtime.

QoS-based Surrogates Selection and Service Proposal Formulation in Offloading Environments

Luís Nogueira, Luís Miguel Pinho
IPP Hurray Research Group
Polythecnic Institute of Porto, Portugal
{luis,lpinho}@dei.isep.ipp.pt

Abstract

Resource constraints are becoming a problem as many of the wireless mobile devices have increased generality. Our work tries to address this growing demand on resources and performance, by proposing the dynamic selection of neighbor nodes for cooperative service execution. This selection is influenced by user's quality of service requirements expressed in his request, tailoring provided service to user's specific needs. In this paper we improve our proposal's formulation algorithm with the ability to trade off time for the quality of the solution. At any given time, a complete solution for service execution exists, and the quality of that solution is expected to improve overtime.

1 Introduction

Users, using any of his heterogeneous mobile devices (e.g. laptop, PDA or cell phone), want to access any application, anytime and anyware. However, the strict constraints on their resources such as memory capacity, CPU speed and battery power makes it very challenging to run complex applications. At the same time, rapid advances in processor and wireless networking technology are introducing resource constrained mobile devices into distributed computing. As such, the presence of high-performance nodes in the neighborhood of these resource constrained devices exposes new opportunities in the efficient implementation of network-wide applications.

Such an heterogeneous environment demands a global approach to achieve efficient resource usage that constantly adapts to devices' specific constraints, nature of executing tasks (hard real-time, soft real-time, non real-time) and dynamically changing system conditions (task migration, mobility). Researchers have been proposing and optimizing several techniques for resource management in resource constrained devices. In [6, 8, 12, 13] all explore the use of computation offloading to a remote machine for power (and performance) gains. They conclude that the efficiency of an application execution can be improved by careful partitioning the workload between the device and its neighbors. Optimal application partitioning depends on the tradeoff between the computation workload and the communication cost. However, a method for finding and selecting the best service providers among the set of neighbor nodes is still missing.

Also, to the best of our knowledge, previous work in offloading do not take into consideration quality of service (QoS) issues, according to end-users' preferences. QoS is considered an important user demand, receiving wide attention in real-time research. Unfortunately, in most systems, users do not have any real influence over the QoS they can obtain, since service characteristics are fixed when the systems are initiated. Furthermore, users can differ enormously in their service requirements as well as applications in the resources which need to be available. Therefore, there is an increasing need for customizable services that can be tailored to user's specific requirements. A QoS negotiation model is the key to build predictable, gracefully degradable middleware services for real-time applications [1].

In [11] we proposed a system where heterogeneous nodes organize themselves into a coalition, dictated by computational capabilities. Changes in the availability of resources introduces a problem for distributed applications, since the optimal (or even a good) mapping of the application depends on the system and its status.

We are primarily interested in dynamic scenarios where new tasks can appear while others are being executed, the processing of those tasks has associated real-time execution constraints, and service execution can be performed by a coalition of neighbor nodes. Such scenarios prevent the possibility of computing optimal resource allocations before execution. Instead, nodes should negotiate partial, good-enough service proposals that can be latter refined if time permits.

The main contributions of this paper are the improved fundamentation and motivation for our proposed framework, the further refinement of the coalition formation process, and a new algorithm for service proposal's formulation that trades off time by quality of the solutions.

The rest of this paper is organized as follows. The next section provides a brief description of the problem we are trying to solve. Section 3 presents our distributed offloading framework. The QoS-based surrogates selection and coalition formation for cooperative service execution is described in section 4. Section 5 details how a particular node proposes a specific QoS level for service execution, according to user's multi-dimensional preferences expressed in his request. Section 6 concludes this paper.

2 Problem description

A user wants to execute a resource intensive application (for example, a video-conferencing application) on a resource constrained mobile device, such as a PDA. The audio streams in this application have multiple QoS considerations: the sampling rate of the audio data, the resolution (number of bits) of each audio sample and the end-to-end latency of the audio stream. Similarly, the video streams must deal with multiple QoS dimensions: the video frame rate, the size of the video window, the number of bits per pixel and so on. User's QoS constraints are described through a semantically rich model (see [11] for details). This model allows users to express in their service requests a range of QoS levels they can accept from available providers. However, the selection of the QoS level it will actually receive depends on the provider. As such, the computation workload and communication cost may change with different execution instances and different users. Correct decisions on program partitioning must be made at run time when sufficient information about workload and communication requirements become available [14].

We envision an environment where many such time-critical, real-time and non-real-time applications each with multiple QoS dimensions co-exist in a system with a finite set of resources. During loaded periods, the system may not have sufficient resources to deliver user's required quality in every application along each of its QoS dimensions. Hence, decisions must be made by the underlying offloading partition scheme to split the resource intensive application into tasks and to select the best surrogates for task execution, such that a global objective is maximized.

In [5] the authors consider the adaptation of offloading trigerring and the efficient application partitioning as the two important decision-making problems for adaptive offloading. However, they do not address the problem of surrogate selection among available neighbors.

Considering the existence of a run-time application partitioning algorithm [14, 5], the problem is to select the best service providers among available neighbors, forming a coalition for cooperative application execution, taking user's QoS requirements into account.

Given the set of tasks to offload, resulting from the partitioning algorithm, the system as a whole must seek the maximization of the QoS constraints associated with tasks. This will be achieved via the formation of a temporary group of nodes, which, due to its higher flexibility and agility, is capable of effectively respond to new, challenging, requirements. Various groups of nodes may have different degrees of efficiency in tasks' execution performance due to different capabilities of their members. As such, task allocation should be done with respect to those differences.

3 Distributed offloading framework

In this paper we consider a system where wireless/mobile nodes may dynamically enter the range of each other, and of wired infrastructures (even clusters of nodes[4]), opportunistically taking advantage of resource availability, distributed across neighbor nodes, forming temporary coalitions for service execution, considering QoS-aware applications.

It is clear that such coalition formation presents very significant challenges, especially at the architectural level. Major developments are required in the fields of communications protocols, data processing and application support. Our goal is to develop an architecture which enables the creation of a new generation of nodes that can effectively network together, providing a flexible platform for the support of distinct network applications.

Figure 1 presents the structure of the proposed framework.

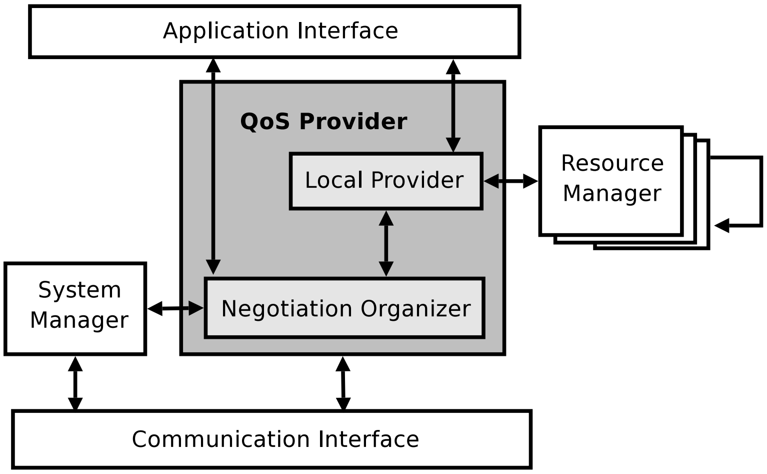

Figure 1: Framework structure

In this model, QoS-aware applications must explicitly request the service execution to the underlying QoS framework, thus providing explicit admission for controlling the system, abstracting from existing underlying distributed middleware and from the operative system. The model itself abstracts from the communication and execution environments.

Central to the behavior of the framework is the *QoS Provider*, which is responsible for processing both local and distributed resource requests. Rather than reserving resources directly, contacts the *Resource Managers* to grant specific resource amounts to the requesting task.

Within the QoS Provider, the *Negotiation Organizer* is responsible for the coalition formation process, atomically distributing service requests, receiving individual nodes' proposals and deciding which node(s) will provide the service. We consider the existence of an atomic broadcast mechanism in the system, guaranteeing that all nodes receive the same service requests and proposals in the same order.

The *Local Provider* is responsible for replying to negotiation requests and for maintaining the state of node's resource allocations and services provided.

The *System Manager* maintains the overall system configuration, detecting nodes entering and leaving the system, coalition operation and dissolution.

The *Resource Manager* is a module that manage a particular resource. This module interfaces with the actual implementation in a particular system of the resource controller, such as the device driver for the network, the scheduler for the CPU, or by software that manages other resources (such as memory). Although we consider a collaborative environment, proper resource usage must be monitored in run time[3], in order to decide based on actual resource usage of the system and not only on the resource usage assumptions of requesting services.

One important issue is the ability of Resource Managers to use each other, in order to allow systems to be built supporting QoS requirements either from the point of view of the user (e.g. high quality), of applications (e.g. frame rate) or of the system (e.g. period and cost).

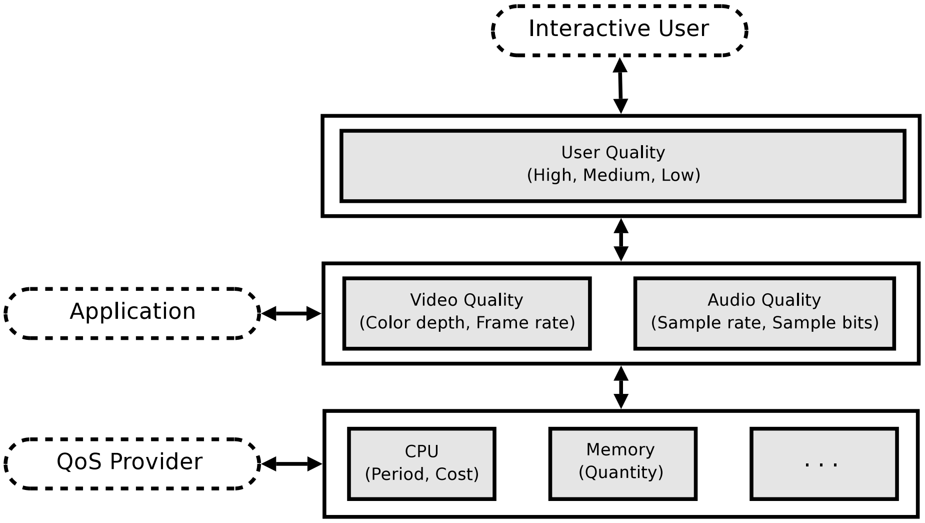

Figure 2: Resource Managers Layering

As an example, a particular system may provide the Resource Managers layering of figure 2. An interactive user application can be more friendly and easier to use by providing only high-level user perceived quality, while other applications could be programmed to use application-related QoS constraints. Finally, service providers collect service requirements at the system level.

4 Surrogates selection

Consider a network with several nodes, each one with its own particular set of resources R_i. There will be several tasks T to be cooperatively executed. Each application has specific QoS constraints, defined by the user, and will compete for the finite set of resources R.

Let Q_i be the set of QoS constraints associated with task T_i. Each Q_{kj} is a finite set of quality choices for the j^{th} attribute of dimension k. This can be either a discrete or continuous set.

The objective of each coalition is to fulfill the resource allocation requests from users. Every member of the coalition provides some of the requested resources, according to its own operation constraints. We view the distributed resource allocation problem as a cooperative process among nodes.

Let $N = 1, \ldots, n$ denote the set of possible nodes available for the application to request resources from. Let P denote the set of user's preferences on the several QoS dimensions for an application, with its independent tasks T. The basic coalition formation problem can be described as:

Given a set of nodes N and a resource allocation demand enforced by P they have to satisfy, if the resource demand cannot be satisfied by a single node or when a single node handles the request inefficiently, it is necessary for the nodes in the network to cooperate to fulfill the resource demand. With cooperation between nodes, by forming coalitions among themselves, resources can be allocated by splitting application's tasks by a subset of N.

4.1 Coalition formation

The coalition formation process should enable the selection of individual nodes that, based on their own resources and availability, will constitute the best group to satisfy user's QoS requirements. In such a scenario, the adopted automatic negotiation mechanism has to be powerful enough to negotiate over requirements described through multiple attributes, which imply that the negotiation process must be enhanced with the capability to both evaluate and formulate multi-attribute proposals. Also, the negotiation process has to be able to deal with inter-dependencies in individual node's proposals, reaching a coherent solution.

Our coalition formation algorithm, seeking distributed QoS optimization, is described in figure 3.

1. On the source node N_i, the *QoS Provider* broadcasts the description of each task T_i, user's QoS constraints Q_i as well as a timeout T_i for proposals reception.
2. Every N_j which is able to satisfy the request, formulates a proposal according to the local QoS optimization algorithm, and replies to N_i with proposal P_j and its local reward W_j, resulting from its proposal acceptance.
3. The *QoS Provider* at N_i evaluates all received proposals for T_i and selects the own that offers the values closer to user's QoS constraints Q_i.
4. N_i offloads task T_i to the winning node.

Figure 3: Coalition formation algorithm

As detailed in [11] we impose a preference order over QoS dimensions, its attributes and values in user's request. Therefore, we consider a proposal to be formulated through the relative decreasing importance ($K = 1 \dots n$) of a set of QoS dimensions. Furthermore, for each dimension a relative decreasing importance order of attributes is also specified ($i = 1 \dots attr_k$), where k is the number of attributes of dimension K. In other words, dimensions and attributes identified, in user's request, by lower indexes are more important than dimensions and attributes identified by higher indexes. Please note that k, i are not the identifiers of dimensions and attributes in the QoS requirements representation for a specific domain, but their relative position in user's service request.

It can be very difficult to the user to specify absolute numeric values to quantify all the QoS dimension's attributes. Imposing a preference order over the dimensions, its attributes and their values encodes user's preferences in a qualitative way.

All admissible proposals are evaluated according to user's QoS preferences specified in service request. A proposal is admissible if it can satisfy all the QoS dimensions requested by the user. For each K QoS dimension, an weighted sum of the differences between user's preferred values and the values proposed by a specific node to that dimension's attributes evaluates proposal's distance to user's request. For that, the degree of acceptability of each proposed attribute's value when compared to the request one is determined, considering continuous and discrete domains.

The best proposal is the one that contains the attributes' values more closely related to user's preferences, in all QoS dimensions.

5 Service proposal formulation

All entities that participate in the QoS negotiation must provide sufficient resources to try to fulfill these QoS requirements. It is therefore the responsibility of each individual *QoS Provider* to map QoS constraints to resource requirements, and then reserve resources accordingly (resource reservations are made through *Resource Managers*).

In [11] we proposed an algorithm for proposals' formulation whose run time depended on the size of the set of tasks being executed and on the number of QoS dimensions and attributes of a particular domain. Algorithms such as Constant Bandwidth Server (CBS)[2], Greedy Reclamation of Unused Bandwidth Server (GRUB)[9], and Idle-time Reclaiming Improved Server (IRIS)[10] were developed to efficiently handle soft real-time requests with a variable or unknown execution behavior under EDF scheduling policy. To avoid unpredictable delays on hard real-time tasks, soft tasks are isolated through a bandwidth reservation mechanism, according to which each soft task s gets a fraction of the CPU and it is scheduled in such a way that it will never demand more than its reserved bandwidth, independently of its actual requests. This is achieved by assigning each soft task a deadline, computed as a function of the reserved bandwidth and its actual requests. If a task requires to execute more than its expected computation time, its deadline is postponed so that its reserved bandwidth is not exceeded. As a consequence, overruns occurring on a served task will only delay that task, without compromising the bandwidth assigned to other tasks. By isolating the effects of task overloads, hard tasks can be guaranteed using classical schedulability analysis. In this paper we improve our proposal's formulation algorithm with the ability to trade off time for quality, and to continue after it has been halted (because its budget has exhausted). Nodes with less workload can reclaim more time to find a solution and, as such, present a solution closer to user's request.

We clear split the formulation of a new proposal for service execution in two different scenarios. The first one involves guaranteeing the new task without changing the level of service of previously guaranteed tasks. The second one, due to node's overload, demands service degradation in existing tasks in order to accommodate the new requesting task. Our local QoS optimization (re)computes the set of QoS levels for all local tasks, including the new requested one. Offering QoS degradation as an alternative to task rejection has been proved to achieve higher perceived utility [1].

At any given time, a complete solution for service execution exists, and the quality of that solution is expected to improve overtime. The algorithm iteratively work on the problem of service proposal and produces results that improve in quality over time. Instead of a binary notion of correctness of the solution the algorithm returns a proposal and a measure of its quality. The local reward measures how useful this result will be for the user, according to its service request.

When it is not possible to find a valid solution for service execution within a given time presented in request for proposals' reception, then no proposal will be sent to the requesting node. This way, we encourage load balancing.

Requests for task execution arrive dynamically at any node. To guarantee the request locally, the *QoS Provider* executes the local QoS optimization algorithm described in figure 4. In order to make its proposal each *QoS Provider* contacts the required *Resource Managers* for resource availability. Conventional admission control schemes either guarantee or reject each request, implying that future requests may be rejected because resources have already been committed to previous requests. We use a QoS negotiation mechanism that, in cases of overload, or violation of pre-run-time assumptions [3] guarantees graceful degradation.

The algorithm terminates when the time for the reception of proposals has expired (this time is sent in request), when it finds a set of QoS levels that keeps all tasks schedulable and the local reward can not be further improved, or when it finds that, even at the lowest QoS level for each task, the new set is not schedulable. In this case the new arrived task is rejected.

As described above, each task T_i has user defined QoS dimensions constraints Q_i. Each Q_{kj} is a finite set of n quality choices for the j^{th} attribute, expressed in decreasing order of preference, for all k QoS dimensions.

The algorithm always improve or maintain the quality of the solution as it has more time to run. This is done by keeping the best schedulable solution so far, if the result of each iteration is not always proposing a schedulable set of tasks.

Contrary to the work reported in [1] the reward of providing service at one of the different pre-defined QoS levels is not specified in user's request. Here, the provided QoS level is dynamically built according to user's accepted values for each dimension expressed in request, in decreasing preference order. As such, the reward depends on the number, and relative importance, of QoS dimensions being served closer to user's requirements, and is calculated by:

```
Start by selecting the worst requested QoS level in all $k$ dimensions, $Q_{kj}[n]$, for the new arrived
task $T_a$. Maintain level of service for previously guaranteed tasks.

  1. If the new set of tasks is not schedulable jump to step 2
       • For each QoS dimension in $T_a$ receiving service at $Q_{kj}[m] > Q_{kj}[0]$
           – Determine the utility increase resulting from upgrading attribute $j$ to $m-1$
           – Find level $T_{max}$ whose increase is maximum and upgrade it to the $m-1$'s level
           – Repeat step 1
  2. Select the QoS level in all $k$ dimensions, $Q_{kj}[m]$, for task $T_a$, resulting from step 1. Find
     minimal service degradation to accommodate $T_a$ in step 3.
  3. While the new set of tasks is not schedulable
       • For each task $T_i$ receiving service at $Q_{kj}[m] > Q_{kj}[n]$
           – Determine the utility decrease resulting from degrading attribute $j$ to $m+1$
           – Find task $T_{min}$ whose decrease is minimum and degrade it to the $j+1$'s level
```

Figure 4: Local QoS optimization

$$r = \begin{cases} max & \text{if task is being served at } Q_{best_j} \\ & \text{for all dimensions} \\ max - \displaystyle\sum_{j=0}^{n} w_j * penalty_j & \text{if } Q_{jk} < Q_{best_j} \end{cases} \qquad (1)$$

In equation 1 *penalty* is a parameter that decreases the reward value. This parameter can be defined by the user according to its own criteria and its value increases with the distance for user's preferred value.

6 Conclusions

Wireless nodes may need to cooperate with neighbor nodes in order to fulfill certain services. Given a set of tasks to be executed with associated QoS constraints, we consider situations where a service is assigned to a group of nodes. Service allocation to several nodes is necessary when the processing cannot be performed by a single node or when a single node performs it inefficiently.

Various groups of nodes may have different degrees of efficiency in service execution performance due to different capabilities of their members. As such, service allocation should be done with respect to those differences. Our coalition formation algorithm selects the nodes that offer the solution that includes values closer to user's multi-dimensional QoS requirements. Those requirements are described through a semantically rich QoS specification interface for multidimensional QoS provisioning, allowing the user, and applications, to define fine-grained service requests.

In this paper we improve our proposal's formulation algorithm with the ability to trade off time for quality, and to continue after it has been halted (because its budget has exhausted). Nodes with less workload can reclaim more time to find a solution and, as such, present a solution closer to user's request. At any given time, a complete solution for service execution exists, and the quality of that solution is expected to improve overtime.

References

[1] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin. Qos negotiation in real-time systems and its application to automated flight control. *IEEE Transactions on Computers, Best of RTAS '97 Special Issue*, 49(11):1170–1183, November 2000.

[2] Luca Abeni and Giorgio Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, Madrid, Spain, December 1998.

[3] Ricardo Barbosa and Luis Miguel Pinho. Mechanisms for reflection-based monitoring of real-time systems. In *Work-In-Progress Session of the 16th Euromicro Conference on Real-Time Systems*, June 2004.

[4] Michael Ditze, Berta Batista, Eduardo Tovar, Peter Altenbernd, and Filipe Pacheco. Workload balancing in distributed virtual reality environments. In *1st Intl. Workshop on Real-Time LANs in the Internet Age, Satellite Event to the 14th Euromicro Conference on Real- Time Systems*, 2002.

[5] Xiaohui Gu, Alan Messer, Ira Greenberg, Dejan Milojicic, and Klara Nahrstedt. Adaptive offloading for pervasive computing. *IEEE Pervasive Computing Magazine*, 3(3):66–73, 2004.

[6] Ulrich Kermer, Jamey Hicks, and James Rehg. A compilation framework for power and energy management on mobile computers. In *14th International Workshop on Parallel Computing*, pages 115–131, 2001.

[7] Chen Lee, John P. Lehoczky, Daniel P. Siewiorek, Ragunathan Rajkumar, and Jeffery P. Hansen. A scalable solution to the multi-resource qos problem. In *IEEE Real-Time Systems Symposium*, pages 315–326, 1999.

[8] Zhiyuan Li, Cheng Wang, and Rong Xu. Computation offloading to save energy on handheld devices: a partition scheme. In *Proceedings of the 2001 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 238–246. ACM Press, 2001.

[9] Giuseppe Lipari and Sanjoy Baruah. Greedy reclamation of unused bandwidth in constant-bandwidth servers. In *Proceedings of the 12th Euromicro Conference on Real-Time Systems (Euromicro-RTS 2000)*, pages 193–200, Stockholm, Sweden, 2000.

[10] Luca Marzario, Giuseppe Lipari, Patricia Balbastre, and Alfons Crespo. Iris: A new reclaiming algorithm for server-based real-time systems. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04)*, page 211, Toronto, Canada, 2004.

[11] Luís Nogueira and Luís Miguel Pinho. Dynamic qos-aware coalition formation. In *Proceedings of the 19th IEEE International Parallel Distributed Processing Symposium (IPDPS 2005).* IEEE Computer Society, 2005.

[12] Mazliza Othman and Stephen Hailes. Power conservation strategy for mobile computers using load sharing. *SIGMOBILE Mobile Computing Communications Review*, 2(1):44–51, 1998.

[13] Alexey Rudenko, Peter Reiher, Gerald J. Popek, and Geoffrey H. Kuenning. Saving portable computer battery power through remote process execution. *Mobile Computing and Communications Review*, 2(1):19–26, 1998.

[14] Cheng Wang and Zhiyuan Li. Parametric analysis for adaptive computation offloading. In *Proceedings of the ACM SIGPLAN 2004 Conference on Programming Language Design and Implementation*, pages 119–130. ACM Press, 2004.