Technical Report

Iterative Refinement Approach for QoS-aware Service Configuration

Luis Nogueira

Luis Miguel Pinho

TR-061001

Version: 1.0

Date: October 2006

Iterative Refinement Approach for QoS-aware Service Configuration

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: {luis, lpinho}@dei.isep.ipp.pt

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

Abstract

In heterogeneous environments, diversity of resources among the devices may affect their ability to perform services with specific QoS constraints, and drive peers to group themselves in a coalition for cooperative service execution. The dynamic selection of peers should be influenced by user's QoS requirements as well as local computation availability, tailoring provided service to user's specific needs. However, complex dynamic real-time scenarios may prevent the possibility of computing optimal service configurations before execution. An iterative refinement approach with the ability to trade off deliberation time for the quality of the solution is proposed. We state the importance of quickly finding a good initial solution and propose heuristic evaluation functions that optimise the rate at which the quality of the current solution improves as the algorithms have more time to run.

ITERATIVE REFINEMENT APPROACH FOR QOS-AWARE SERVICE CONFIGURATION

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

Abstract In heterogeneous environments, diversity of resources among the devices may affect their ability to perform services with specific QoS constraints, and drive peers to group themselves in a coalition for cooperative service execution. The dynamic selection of peers should be influenced by user's QoS requirements as well as local computation availability, tailoring provided service to user's specific needs. However, complex dynamic real-time scenarios may prevent the possibility of computing optimal service configurations before execution. An iterative refinement approach with the ability to trade off deliberation time for the quality of the solution is proposed. We state the importance of quickly finding a good initial solution and propose heuristic evaluation functions that optimise the rate at which the quality of the current solution improves as the algorithms have more time to run.

1. Introduction

The amount of data produced by a variety of data sources and sent to end systems to further processing is growing significantly. There are several examples of sensors being installed to continuously measure environmental properties and disseminate data streams. These applications are pushing the limits of traditional data processing infrastructures [15]. The challenges become even more critical when coordinated content analysis of stream data from multiple sources is necessary [3]. This calls for an architecture that supports the distribution of the processing task to different nodes in order to be able to cope with increasing resource requirements.

At the same time, quality-aware processing of those data streams is increasingly being 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 to perform a service with a specific level of quality. Therefore, there is an increasing need for customisable services that can be tailored to user's specific requirements [14]. A QoS negotiation model is the key to build predictable, gracefully degradable services for real-time applications [1].

This paper addresses the growing demand on resources and performance requirements by allowing resource constrained devices to cooperate with more powerful or less congestioned neighbour nodes to meet resource allocation requests and handle stringent constraints, opportunistically taking advantage of global network resources and processing power.

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 neighbour nodes. Such scenarios may 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. Moreover, taking the cost of decision-making into account is not an easy task, since the "optimal" level of deliberation varies from situation to situation. It is therefore beneficial to build systems that can trade off computational resources for quality of results.

We propose and evaluate new anytime algorithms for coalition formation and service proposal formulation with the ability to trade off deliberation time by the quality of the solutions. The proposed algorithms can be interrupted at any time and provide a solution and a measure of its quality. This quality is expected to improve as the run time of the algorithms increase. A higher adaptation to changing conditions in dynamic environments is thus introduced by allowing flexibility in the execution times of the algorithms.

The conformity of both algorithms with the desired properties of anytime algorithms and the validation through extensive simulations of the design decisions of our approach is detailed in [11]. The achieved results emphasise our believe that use of anytime algorithms for coalition formation and service proposal formulation significantly improve the ability of our framework to adapt to changes in a dynamic heterogeneous environments.

2. Related work

The quality of the outputs may depend on the available amount of resources. For example, in multimedia applications, higher network and CPU bandwidth produces better audio and video quality, at higher resolutions and/or higher frame rates. As such, researchers have been proposing and optimising several techniques for resource management in resource constrained devices.

Computation offloading to a remote machine has been explored to achieve power and performance gains [6, 7, 13]. The authors conclude that the efficiency of an application execution can be improved by careful partitioning the workload between a device and a fixed neighbour. Optimal application partitioning depends on the trade off between the computation workload and the communication cost. However, a method for finding and selecting the best subset of service providers among the set of neighbour nodes is still missing. Also, to the best of our knowledge, previous work in offloading do not take into consideration QoS constraints imposed by users in their service requests. Since different users can access multiple devices at the same time, supporting users' QoS preferences in service execution is a key issue.

Work on applications' decomposition into tasks has, for example, been reported in [3, 9, 16]. Interpretation of QoS constraints and consequent mapping on resource parameters as been described, for example, in [12, 4, 5]. We focus on proposing a generic model that enables a distributed QoS-aware service allocation, with the ability to adapt to dynamically changing system conditions.

Our preliminary work [10] proposes a system where heterogeneous nodes organise themselves into a coalition for cooperative service execution, dictated by computational capabilities. However, the assumption that the algorithms can have all the time they need to compute their outputs was used.

The work on anytime algorithms [2, 17] recognises that the computation time needed to compute optimal solutions will typically reduce the overall utility of the system. An anytime algorithm is an iterative refinement algorithm that can be interrupted and asked to provide an answer at any time. It is expected that the quality of the answer will increase (up to some maximum quality) as the anytime algorithm is given increasing time to run, offering a trade off between the quality of the results and computational requirements. Associated with an anytime algorithm is a performance profile, a function that maps the time given to an anytime algorithm (and in some cases input quality) to the quality of the solution produced by that algorithm.

3. Time-bounded coalition formation

A 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 Q associated with service S. The anytime approach proposed here extends the algorithm introduced in [10] by allowing it to return many possible approximate answers and a measure of their qualities for a given input of service proposals to evaluate. Those service proposals are sent by neighbour nodes, in reply to a cooperative service execution request with associated user's QoS constraints that this node is not able to fulfil by itself.

We consider a user's service request to be formulated through the relative decreasing importance of a set of QoS dimensions [10]. Furthermore, for each dimension a relative decreasing importance order of attributes, and possible values for each attribute, is also specified. As a result, the user is able to express acceptable compromises in QoS and their relative importance.

All admissible proposals are evaluated according to user's QoS preferences, measuring the distance between requested and proposed values [10]. The best proposal is the one that contains the attributes' values more closely related to user's preferences, in all QoS dimensions.

Time-bounded coalition formation implies trying to quickly find a good initial solution and gradually improve that solution if time permits. The selection of the next candidate proposal to be evaluated from the set of available proposals should be done in a order that maximises the expected improvement in solution quality. It is necessary to make a trade off between search effort and solution quality explicitly in the heuristic selection of the next candidate proposal so that we can optimise search effort directly, rather than relying in arbitrary proposal evaluation. As such, for each task T_i we select the next candidate proposal P_{ki} from the set of received proposals P_i to be evaluated for task T_i, as *the one sent by node N_k that has the greatest local reward R_k*.

The local reward R_k is an indicator of node's local QoS optimisation, according to the set of tasks being locally executed and their QoS constraints. We claim that the local reward achieved by a node should be used to guide the coalition formation process, since nodes with higher local reward have a higher probability to be offering service closer to user's request under negotiation.

The anytime coalition formation algorithm, seeking distributed QoS optimisation, is described in Algorithm 1. Since the formation of a coalition is aimed at maximising the benefits associated to a cooperative service execution, the quality of each generated coalition can be

measured by using the evaluation values of the best proposals for each service's task.

$$Q_{coalition} = \left\lfloor \frac{|coalition|}{|S|} \right\rfloor * \sum_{i=1}^{|coalition|} \frac{1 - Best_{P_i}}{|coalition|} \quad (1)$$

For an empty set of proposals the quality of the coalition is zero. Note that the quality of the coalition is also zero, if there are not any proposals for one or more tasks T_i of service S.

Algorithm 1 Iterative coalition formation

for each $T_i \in S$ **do**
  Select next candidate proposal P_{Ki}, maximising local reward
  $E_{P_{ki}} = evaluate(P_{ki})$
  if $E_{P_{ki}} - Best_{P_i} > \alpha$ **then**
    $Best_{P_i} = E_{P_{ki}}$
    Update coalition with N_k for task T_i
  else if $0 < E_{P_{ki}} - Best_{P_i} < \alpha$ and $R_{P_{ki}} > R_{Best_{P_i}}$ **then**
    $Best_{P_i} = E_{P_{ki}}$
    Update coalition with N_k for task T_i
  end if
end for

The algorithm continues, if time permits, to evaluate received service proposals trying to improve the quality of the current solution. It is possible that another node, while achieving a lower local reward, proposes a better service for the specific request under negotiation. The service proposal formulation algorithm, described in the next section, always suggests the best solution for a particular user, even if it has to degrade the provided level of service of previous existing tasks. It is the responsibility of the coalition formation algorithm to select between similar proposals (whose evaluation values differ in less than α) those nodes that achieve higher local rewards, promoting load balancing.

The algorithm terminates when it finds that the quality of a coalition cannot be further improved or the local reward of each node that belongs to that coalition is maximum.

4. Time-bounded service proposal formulation

Requests for cooperative service execution arrive dynamically at any node. Each user's request is formulated as a set of acceptable multi-dimensional QoS levels in decreasing preference order. To guarantee the

request locally, the node executes a local QoS optimisation algorithm described in Algorithm 2. 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 guarantees graceful degradation. In our model, guaranteeing a user's request is the certification that the service will be provided in *one* of the QoS levels expressed in the request.

A service configuration proposed for a specific task T_i will achieve a reward r_i determined by the proximity of the proposal with respect to the QoS preferences specified in user's service request. Its value is maximum if the task is being served at the highest requested level in all QoS dimensions. Otherwise, it is affected by a penalty factor that increases with the distance for user's preferred values [10].

As introduced in the previous section, each node sends along with the service proposal a measure of global satisfaction resulting from its proposal acceptance. The local reward R_j expresses a degree of global satisfaction for all the users that have tasks being executed by a particular node N_j, with specific QoS levels. For a node N_j, the local reward $R_j = (\sum_{i=1}^{n} r_{T_i})/n$ achieved by a set of tasks is determined combining the reward of each task being locally executed as a measure of global satisfaction of the proposed solution.

Unless all tasks are executed at their highest QoS level, there is a difference between the actual local reward achieved by the currently selected QoS levels and the maximum possible local reward that would be achieved if all local tasks were executed at their highest requested QoS level. This difference can be caused by either resource limitations, which is unavoidable, or poor load balancing, which can be improved by sending actual local rewards in service proposals, and selecting, for proposals with similar evaluation values, those nodes that achieve higher local rewards. Selecting the node with higher local reward for similar service proposals, not only maximises service satisfaction for a particular user, but also maximises global system's utility, since a higher local reward clearly indicates that the previous set of tasks being locally executed had to suffer less QoS degradation in order to accommodate the new task.

In [8], it was demonstrated that the QoS optimisation problem involving multiple resources and multiple QoS dimensions is NP-hard. An optimal solution based on dynamic programming and an approximation scheme based on a local search technique was presented. However, the computation time needed to find an optimal solution can reduce the overall utility of the system. In addition, the deliberation cost is dependent

on local resources' availability and user's QoS constraints. Therefore, it is beneficial to build systems that can trade the quality of results against the cost of computation [17].

The proposed anytime algorithm considers two different scenarios when formulating a service proposal. 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 optimisation (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].

The algorithm iteratively work on the problem of finding a feasible service configuration that maximises user's satisfaction and produces results that improve in quality over time. Equation 2 shows how the quality of each generated feasible configuration Q_{conf} is calculated by considering the reward achieved by the service proposal configuration for the new arriving task r_{T_a}, the impact on the provided QoS of previous existing tasks and the value of the previous generated feasible configuration Q'_{conf}. Initially, Q'_{conf} is equal to zero.

$$Q_{conf} = \left(r_{T_a} * \frac{\sum_{i=0}^{n} r_{T_i}}{n} \right)^{(1-Q'_{conf})} \tag{2}$$

When a new service request arrives, the algorithm starts by maintaining the QoS levels of previously guaranteed tasks and by selecting the worst requested QoS level, for all dimensions, for the new arrived task. As such, the reward of the initial service configuration for the new task is low (the exact value is determined by the penalty factors used in a particular system), affecting node's local reward. On the other hand, the impact of this new task on the provided level of previously existing tasks is inexistent. Also, this initial solution is the service configuration that has a higher probability of being feasible, considering the new arrived task. The algorithm continues to improve the quality of the initial solution, conducting the search for a better feasible solution in a way that maximises the expected improvement in solution's quality. When there are enough resources the algorithm selects, from the set of possible upgrades, the next configuration that*maximises the reward achieved by the new arrived task*. When QoS degradation is needed, it selects the configuration that *minimises the decrease in local reward*.

At each iteration the algorithm produces a new service configuration that may not be feasible due to local resources availability and user's QoS

constraints expressed in request. Since a service proposal can only be considered useful within a feasible set of configurations, the algorithm, if interrupted, always returns the best found feasible solution. However, each intermediate configuration, even if not feasible, is used to calculate the next solution, minimising search effort.

When the new task can be accommodated without degrading the QoS of previously existing tasks, the algorithm incrementally selects the configuration that maximises the increase in obtained reward, according to user's QoS preferences expressed in his request. When QoS degradation is needed, the algorithm incrementally selects the configuration that minimises the decrease in obtained reward of all tasks.

Algorithm 2 Iterative service proposal formulation

Each task T_i being locally executed has associated a set of user QoS constraints Q^i.
Each $Q^i_{kj} = \{Q^i_{kj}[0], \ldots, Q^i_{kj}[n]\}$ is a finite set of n quality choices for the j^{th} attribute of the k^{th} QoS dimension associated with task T_i, expressed in decreasing order of preference.

Step 1: Improve QoS level of the new arrived task T_a
Select the worst requested QoS level, in all j attributes of all k dimensions, $Q^a_{kj}[n]$, for task T_a.
Maintain level of service for all previously guaranteed tasks.
while the new set of tasks *is* feasible **do**
  for each k QoS dimension in T_a receiving service at $Q^a_{kj}[m] > Q_{kj}[0]$ **do**
    Determine the utility increase by upgrading attribute j to $m-1$
    Find *maximum* increase and upgrade attribute to the $m-1$'s level
  end for
end while
Step 2: Find global minimal degradation to accommodate T_a
Select for all k dimensions of task T_a the final result of Step 1, $Q^a_{kj}[m]$
while the new set of tasks *is not* feasible **do**
  for each task T_i receiving service at $Q^i_{kj}[m] > Q^i_{kj}[n]$ **do**
    Determine the utility decrease by degrading attribute j to $m+1$
    Find task T_{min} whose reward decrease is *minimum* and degrade attribute j to the $m+1$'s level
  end for
end while

The algorithm terminates when the time for the reception of proposals has expired (this time is sent in user's request), when it finds a set of

feasible QoS levels and the quality of the solution can not be further improved, or when it finds that, even at the lowest QoS level for each task, the new set is not feasible. In this case the new arrived task is rejected. When it is not possible to find a feasible solution to include the new task within available time, the node continues to serve existing tasks at their current QoS levels and does not send any service proposal to the requesting node.

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

5. Conclusions and future work

Resource constrained devices may need to cooperate with neighbour nodes in order to fulfil complex services, with specific user's QoS constraints. Given a set of tasks to be executed, we consider situations where a service is assigned to a group of nodes for cooperative execution in a dynamic heterogeneous environment.

This paper proposes algorithms for coalition formation and service proposal formulation with the ability to trade off deliberation time for quality of results. At each iteration, the search of a better solution is guided by heuristic evaluation functions that optimise the rate at which the quality of the current solution improves overtime. These capabilities are essential for successful operation in dynamic real-time environments, as it may not be feasible to compute an optimal answer before providing a solution for a cooperative service execution.

The proposed anytime algorithms significantly improve the ability of our framework to adapt to changes in dynamic environments by allowing flexibility in the execution times of the algorithms. A complete integration in the existing framework is under development.

Acknowledgments

This work was supported by FCT, through the CISTER Research Unit (FCT UI 608) and the Reflect project (POSI/EIA/60797/2004).

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] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 49–54, 1988.

[3] Viktor S. Wold Eide, Frank Eliassen, Ole-Christoffer Granmo, and Olav Lysne. Supporting timeliness and accuracy in distributed real-time content-based video analysis. In *Proceedings of the 11th ACM international conference on Multimedia*, pages 21–32. ACM Press, 2003.

[4] K. Fukuda, N. Wakamiya, M. Murata, and H. Miyahara. Qos mapping between user's preference and bandwidth control for video transport. In *Proceedings of the 5th International Workshop on Quality of Service*, pages 291–302, New York,USA, 1997.

[5] Vera Goebel and Thomas Plagemann. Mapping user-level qos to system-level qos and resources in a distributed lecture-on-demand system. In IEEE Computer Society, editor, *Proceedings of The 7th IEEE Workshop on Future Trends of Distributed Computing Systems*, page 197, 199.

[6] 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.

[7] 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.

[8] Chen Lee, John Lehoczky, Dan Siewiorek, Ragunathan Rajkumar, and Jef Hansen. A scalable solution to the multi-resource qos problem. In *20th IEEE Real-Time Systems Symposium*, pages 315–326, 1999.

[9] L. Marcenaro, F. Oberti, G. L. Foresti, and C. S. Regazzoni. Distributed architectures and logical-task decomposition in multimedia surveillance systems. *Proceedings of the IEEE*, 89(10):1419–1440, October 2001.

[10] Luís Nogueira and Luís Miguel Pinho. Dynamic qos-aware coalition formation. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, Denver, Colorado, April 2005.

[11] Luís Nogueira and Luís Miguel Pinho. Time-bounded distributed qos-aware service configuration in heterogeneous cooperative environments. Technical report, IPP Hurray Research Group. Available at http://hurray.isep.ipp.pt/, January 2006.

[12] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A resource allocation model for qos management. In *Proceedings of the 18th IEEE Real-Time Systems Symposium*, page 298. IEEE Computer Society, 1997.

[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] Sven Schmidt, Thomas Legler, Daniel Schaller, and Wolfgang Lehner. Real-time scheduling for data stream management systems. In *17th Euromicro Conference on Real-Time Systems (ECRTS'05)*, pages 167–176, 2005.

[15] Michael Stonebraker, Ugur Cetintemel, and Stan Zdonik. The 8 requirements of real-time stream processing. *SIGMOD Record*, 34(4):42–47, 2005.

[16] 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.

[17] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *Artificial Inteligence Magazine*, 17(3):73–83, 1996.