On Computing the Utility of a Network of Devices*

W. Ke, P. Basu, S. Abu Ayyash, and T.D.C. Little
Department of Electrical and Computer Engineering, Boston University
8 Saint Mary's St, Boston, MA 02215, USA
617-353-9877
{*ke,pbasu,saayyash,tdcl*}*@bu.edu*

MCL Technical Report No. 03-12-2003

Abstract–Pervasive computing is characterized by speculated, unbuilt applications and features. From what we can envision today, these have wildly varying characteristics with associated network performance needs. We argue that conventional performance metrics such as bandwidth, throughput, and latency have insufficient richness to quantify desired properties for these future networked applications, and propose a utility metric more suitable for the characteristics of the applications and the pervasive computing network infrastructure. The utility metric can be derived from the tasks (applications) that the devices can execute. Therefore, key research challenges that need to be addressed in this context are: (1) building an application specification model that allows us to categorize applications into distinct types, and (2) formulating a mapping function that allows us to determine the feasibility region (number of instances) of each type of applications that the devices can execute, given pre-specified performance bounds and (3) computing aggregate and marginal utilities of a network with respect to the tasks that need to be executed.

Keywords: Distributed Application Specification, Utility of a Network of Devices

*This work was supported by the NSF under grant No. ANI-0073843. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

Technological advances have brought many computing devices to our everyday lives. We rely on them to perform a plethora of activities (personal information management, communication, scheduling, etc.), but with the increase in the resources available in the devices, soon it becomes apparent that many of the activities performed by one device overlap with the functionalities present in another, e.g., a cellphone carries a phone book and a PDA carries an addressbook with cellphone numbers. During the same time that such devices became popular wireless networking technologies such as IEEE 802.11 [1] and Bluetooth [2] also found their way to the devices. The coupling of the communication technology and computing devices in fact opens the door for us to exploit the presence of the variety of forms and types of services in this pervasive computing environment.

In a continual effort to harness the full potential of such an environment, the research community has persistently laid down research principles that argued for the distinction between the *tasks we want done* and the *devices (hosts) that execute them* [3, 4, 5, 6]. Parallel to this principle is the understanding of a need for data-centric paradigms [7, 8, 9] instead of the old host-centric paradigms. The separation between the tasks and specific "devices" can be further seen in [10], for web-enabled services, and in [11], for programming languages. Service Location Protocol [12] and JINI [13] also can be used to bring to reality the concept that what matters is not finding any *specific* device, but how to find *a* device that is capable of accomplishing the task in mind and enabling it to do so in a seamless manner.

Much work is still needed in terms of enabling inter-operability among such a diverse set of devices, communication technologies and operating systems, but already services in the current wired network can be seen as adopting a different paradigm, such as the development of grid computing [14], in which what is sought is computational power, storage capacity or communication capacity, apart from any specific host. Proposals to attach wireless devices to the grid [15] have already been made.

The evolution of this paradigm, which we loosely term "task" paradigm, after [3, 6] brings then a necessity for a specification of "task." Such specification should define what a task is, how tasks are formed and what rules "govern" their behavior. Furthermore, this specification eventually must fully support the nuances of context for task execution. Currently research is made to define such specification for web services [10, 16], employing high level abstractions methods [17, 18]. With such high level of abstraction and potential complexity, it becomes extremely difficult for a human being to determine whether any given task abstraction can be executed by a network of devices.

Given the current increase in the number of embedded devices and research into sensor

networks, one can envision systemlevel design questions such as: which resources to increase/replenish in the sensor network, which new resources to add to the environment, and which resources can be discarded. These are issues related to the *utility of the network*. If what matters to the user are the tasks that can be executed, and not any specific devices, then the utility of a network of devices must be defined in terms of the tasks that it can execute. And the marginal value of a single device must be computed with respect to the increase (decrease) of the number of tasks the network can support with the device's addition (removal). We believe that quantification of this utility in terms of known factors can answers the questions mentioned above.

In this paper we present this challenge to the research community: how can we evaluate the utility of a set of devices for a user given that value is found not in the hardware specification of the devices, but on the tasks that can be executed by them? We show how through proper research direction on task and application specifications this problem can be tackled.

Related Work The separation of tasks from the devices (hosts) has been visited in [3, 5, 6, 7], but these works provide the arguments for the separation principle and/or provide approaches to solve the technical aspects of the issue, not being specifically concerned with the utility aspects. Market oriented models of resource allocation have been presented in [19, 20, 21, 22], among others. Our challenges focus on computing a metric in pervasive computing environments that support the complexity of the task paradigm. In Sec. 2 we show how utility issues and the task paradigm are related. We also offer an intuitive description of our approach to utility computation. Sec. 3 lays down the foundational work regarding the task paradigm, addressing various research issues, and presents a formal way of showing how to leverage research results to solve the utility computation problem. We conclude then in Sec. 4.

2 A First View of Utility

2.1 Utility in Everyday Life

The evolution of software application paradigms will enable (as mentioned in [3]) a user to perform complex tasks which require multiple hosts coordinate their services. This means that we cannot characterize application needs in terms of pure CPU cycles, storage capacity or communication capacity only, but instead, an amalgam of hosts, each satisfying a certain collection of attributes (representing a subtask), must be ensembled to execute the application. If we consider that the user is mobile and the application must "follow the leader," each "subtask" must "move" with the user, and, in the process, access different hosts and

take on the characteristics (computational, communication, storage and I/O) of the host in which it "resides." It may even be necessary for a "subtask" which resided in a single host to be split among multiple hosts when the user roams into an environment where there is no single device that can support it. Conversely, if these subtasks are led into an environment that has one powerful device that can support all of them, then they may be executed from the same host.

As an example of the need to accommodate different devices when executing an application, consider a simple video conferencing application [4]. In the office, a user may engage in video conferencing through a multimedia workstation. As the user is on the move, the video conference may be downgraded into a simple phone conference. When the user arrives home, the home theater system may pick up the video conferencing functions, or if it has been taken by other family members and visual information is needed, the cellphone, a digital camera and a PDA may collaborate together to provide the full audio/visual aspect of the conference.

How then, would this new application paradigm play into everyday scenarios? Consider:

1. If a company provides a pervasive computing environment and is interested in upgrading its set of available devices, not only should it perform upgrades in raw computational power or storage, but also in the diversity or capacity of devices that support different applications.

2. A user buying a Bluetooth enabled hands-free headset suddenly can use the headset to make phone calls (dialing through voice recognition), listen to CDs from the Bluetooth enabled stereo system (low audio quality, but very convenient, again controlling the selections through voice recognition software that resides in a network enabled desktop computer), check the weather from radio or TV stations (only the audio part), and even browse online recipes through an audio interface filter/driver. And all this hands free! What is the utility of the headset? What is the utility of the voice recognition software?

3. A conference organizer who wants to support a pervasive computing environment for the general public must provide devices that can support popular tasks, while providing enough bandwidth resource for all of them.

4. The "utility" of a sensor network must in turn be evaluated by the possible outcomes of coordination among various neighbor sensors, which can effectively increase the capacity of the sensing task, e.g., higher sampling rates, or allow different sensing tasks, e.g., using temperature sensors and pressure sensors in a balloon to determine its altitude.

In the scenarios above we illustrate how the task abstraction influence utility decisions with respect to a network of devices. Note that it is precisely because tasks can potentially be accomplished by many different devices and in many different ways that the decision of adding a single device (or provisioning of devices for others) must be made having the other devices and the commonly used tasks in mind. Our solution to the utility problem must therefore, start with the task specification.

2.2 An Intuitive Approach To Measuring Utility

From the examples given above, it is intuitively obvious that with the task paradigm, computation of the utility of a device must incorporate things it can do (i.e., tasks it can execute) when it is in the presence of other devices. Therefore:

1. The utility of a device to a user changes not only because of a function of its own static attributes but is actually also a function of the presence of other task enabling devices nearby. If a device D belongs to a set of devices S, and D is present and needed in all the tasks S can support, then D's marginal utility is extremely high.

2. This dependence on others neighboring devices also means that at run-time, as neighboring devices are busy executing other applications and become unavailable, a device D's utility drops, even if it is completely idle and no hardware changes occurred.

For now we shall occupy ourselves with computation of the utility of the network of devices to the user when there are no other users present. Run-time scenarios and multi-user resource allocation that maximizes aggregate utility are research topics that need be addressed in the future.

Intuitively speaking, each task can be accomplished through a different number of applications (in this paper, we consider "task" to be an abstraction that does not need to be tied to any attribute(s) present in physical devices; "application specification" represents a set of real attributes and relationships between them to realize the task and an "application" indicates a software program that is executed on top of devices). For each application we apply a pre-specified embedding algorithm that selects the device(s) which will run the application. In this way we propose to establish a mapping from a task to a *point* in the "realizable" application space, i.e., the application suggested to execute the task was successfully embedded (found suitable devices and in sufficient number) in the network, given certain performance constraints.

If we go through all points of the "realizable" application space and sum the associated user utility value, we obtain a utility metric for the network of devices. Note that although in the

general sketch delineated above the whole Earth's connected network components could be theoretically encompassed in the utility computation, a scheme that does not search in such fine-grained manner in the realizable application space is better suited for large networks.

3 Computing Utility

3.1 Tasks

A task can be loosely defined as an expression of the desire of the user, and should reflect the user's wishes accurately. A user who wishes to print to a laser printer may be indifferent to color or black-and-white outputs but may not want to have his work printed in a dot matrix. In addition, the user may not object to having his print out sent to the printer on the same floor, but would mind very much having to collect his documents 20 floors up. Also, the user may have no problems switching from a videophone call to a regular telephone call, but would definitely "hang" if the "call" was being conducted through the user's non-multimedia enabled PDA connected via a network printer. However, the same user might not mind making such call at all if possible if he locked himself out of his laboratory and had only access to the PDA and that printer.

The examples above show the complexity in expressing the task accurately and capturing all the possible nuances of the user's intent, based on the context. One difficulty lies in an almost seemingly infinite context possibilities that can change the task being executed.

Assuming that nuances can be captured, what are some other problems related to task specification? Suppose the user has an ebook in the PDA and would like to print it out, but the Postscript printer does not recognize the format of the ebook. If both the PDA and the printer are network enabled, a logical solution is to seek a filter that can translate the ebook format into one recognized by the printer and form the following *ebook printing task*: PDA $\rightarrow$ (ebook format)-to-(PS format) $\rightarrow$ printer. This path of reasoning leads to some questions:

1. How are such composite tasks built?

2. What is the lowest level (if any) definition of a task? In other words, are there *atomic* tasks (tasks that cannot be decomposed into smaller tasks)?

3. If there is another filter, which reverts the printer recognized format back to the ebook format, and suppose we add in sequence: *PDA* $\rightarrow$ *(ebook format)-to-(PS format)* $\rightarrow$ *(PS format)-to-(ebook format)* $\rightarrow$ *(ebook format)-to-(PS format)* $\rightarrow$ *printer*, would this mean that we have an "extra" task the network can execute?

4. What if there were two filters of different capacities (e.g. 300 dpi and 600 dpi), would there be two different tasks?

The answer to the first question may be found in work explored in building ontologies and process ontologies [4, 18, 23] for web-services. Their aim is to provide meaning to services found in the web, and allow their composition and reuse. These works may provide the starting point to allow automatic task composition, given a set of task specifications. The answer to the second question is actually subjective, and it is part of the design. Real specifications, however, should depend on the physical attributes of the devices on which the task will be executed. A highly detailed specification may be completely unnecessary (and may be detrimental to efficient use) if there are no equivalent physical hosts in the universe of possible devices that can execute the tasks. We are implicitly defining here the notion of *atomic* tasks [6, 23], which do not admit decomposition. A second aspect of defining the level of detail of tasks is the ease with which tasks are translated into specific application requirements. More will be discussed in the next section, when we talk about application specifications.

As for the third question, in principle we could say that we have an "extra" task the network can accomplish, but which would take more resources from the network and would accomplish the same results as a simple *ebook printing task* defined above. Thus, considering that our goal is eventually to obtain an application specification that can be used to measure the utility of a network, such "extra resource consuming equivalent tasks" must be *reduced* to *equivalent* formulations that consume the least amount of resources. Regarding the fourth question, note that *dpi* is in fact an attribute of an instance of a filter task. Such difference in attributes do not prevent the task from being executed and therefore we argue that there is in fact only one task. We are implicitly claiming that, unless formulated in the task specification itself, QoS differences do not introduce fundamental changes to a task but simply influence the maximum performance level that a specific instance of a task may achieve.

3.2 Application

If we consider the task to be a specification of a user's intent, then an application specification is an attempt to describe a task's intent in a more concrete fashion, by specifying the attributes of physical hosts necessary for task completion.

Application specifications must also support the operation of composition and decomposition, so that a complex application may be separated into simpler ones. The notion of *atomicity* is supported, and, just like tasks, the level of atomicity should be dependent on the specific devices. Also like discussed in the task specification section, a composition of

application specifications must satisfy minimum resource consumption requirements. However, whereas task specification may not bring any specific "attribute" values (e.g. 300 dpi), application specifications always bring a range of acceptable attributes. Furthermore, the operations of composition/decomposition must preserve original QoS constraints.

From the paragraph above we see that an "application specification" defines a set of resources in terms of measurable attributes (computational capacity, specific I/O capacity, etc) which devices in a network can match. A "task specification," on the other hand, defines a set of units of work that must be performed by a set of resources satisfying certain conditions. Thus a key research challenge lies in designing a proper *translator*, which can express the same *meaning* of a task specification in terms of application specifications. Work related to ontologies has been suggested as building blocks for task specifications (3.1). These may also be used as building blocks for giving application specifications the semantics needed. Further work in merging and translation [24, 25] of ontologies may then be leveraged as starting research point. Eventually the goal should be to have the process automated to the point in which a system designer only offers the atomic specifications, and composite application specifications build automatically when a task is given[1].

Some further points related to the discussion of applications:

- Physical hosts in the network are described according to application specifications. In other words, a physical device is described as a collection of atomic application specifications it can support, along with all the rules of composition possible. Such rules should come as a result from the research challenge mentioned in the preceding paragraph. In the absence of such rules a device may need to hold a collection of application specifications it can support.

- A suitable model for the applications (and not just the specification) can (or should) be adopted to verify whether the task specifications and application specifications are correct, i.e., yielding expected results. Potential models can be found in [26, 27].

- The context switches a task may experience (moving from office to car) during its execution lifetime must be reflected in the potential change of applications. Thus a model for the application must capture this possibility of time limits.

- True implementation of the applications is another research challenge, for ideally we should be able to provide a code that can be "merged" when there is enough resources and "split" to many devices when one powerful devices is not present. Alternatives to the ideal case can be the utilization of application libraries that will be downloaded into

[1]Other AI techniques may be needed to achieve this automation.

different devices as they become available and only state mapping and maintenance are required.

3.3 Mapping Functions

Having derived separate application specifications that can accomplish the task, the next step is *mapping* (or embedding) each application to devices that meet its specification. Application specifications can be represented as a graph, where nodes represent attributes of required devices and edges represent the communication requirements [6].

The problem of embedding a general graph onto a network that minimizes dilation[2] is related to subgraph isomorphism, which is known to be NP-Complete [28]. Thus potentially optimal solutions may exist only for specific graph types (e.g. trees [29]) while other mappings need to use heuristics. Needless to say, good heuristics for this mapping function are essential. Poor heuristics may result in unsuccessful execution of a task. A significant impact of the lack of an optimal solution is then a dependency of the utility metric on the mapping function. This unfortunately cannot be avoided, unless there is consensus regarding which mapping function to employ, or a metric other than dilation that would allow optimal solutions. Regarding the mapping function, we have the following remarks:

- Specific locations of the devices at embedding time influences the results;
- Offline computation of the utility of a network of devices may employ complex embedding algorithms. However, at run-time, with cross traffic and no resource sharing, such embedding may not be possible, and accordingly, the network's utility to the user will drop. Estimating dynamic behavior of the network's utility to the user and optimizing the network in terms of the aggregate utility (over multiple users) is another research challenge.

3.4 An Aggregate Utility Formulation

In this section we develop a mathematical formulation of the aggregate utility of a pervasive network of devices. Let a current snapshot of the network be denoted by $G = (V, E)$ where V is the set of devices in the network and E is the set of links that exist between them. Devices in V provide atomic services and can be represented by: $V = \{d_1, d_2, d_3, \ldots, d_N\}$. Let $D_t = \{t_1, t_1, t_3, \ldots, t_k\}$ denote the set of types of services that are provided by devices in V. Let $T = \{T_1, T_2, \ldots, T_n\}$ be the set of application specifications (corresponding to user requested tasks) that can be supported on these devices. Now each application specification

[2]Dilation is the average stretch of an edge in the application specification graph on the network of devices

$T_i(i : 1 \rightarrow n)$ may need services from device of types $t_{i_1}, t_{i_2}, \ldots, t_{i_l}$ and each service type $t_j(j : 1 \rightarrow k)$ may be used for $T_{j_1}, T_{j_2}, \ldots, T_{j_m}$. If n_j is the number of instances of devices offering services of type t_j in the network, we have $\sum_{j=1}^{k} n_j t_j = N$.

We now define a mapping function ψ on the set of application specifications T as follows: $\psi : T \rightarrow S$ where $S \subset V$. For example,

$$\psi(T) = \psi(\{t_{i_1}, t_{i_2}, \ldots, t_{i_l}\}) = \{d_{i_1}, d_{i_2}, \ldots, d_{i_l}\} \tag{1}$$

A_ψ denotes a mapping algorithm that yields the above mapping function in a systematic manner. If all devices are shareable, ψ can map multiple applications to the same set of devices. However, there are likely to be *constraints* in the mapping process, for example, a device may not be shareable by multiple applications or it may not be used by more than a certain number of them concurrently. In general we represent all such constraints by C.

Let $\Theta : T \times V \times (A_\psi, C) \rightarrow P$ be a function computing the performance indices obtained after mapping a given set of application specifications T on devices in V using the mapping algorithm A_ψ while obeying the constraints C. The performance indices are denoted by $P = \{p | p \in \Re^+\}$.

Now consider an average user U in the system who derives a set of utilities Q from given set of tasks that can be performed on the network. It is given by $Q = \{q_1, q_2, \ldots, q_n\}, \forall i q_i \in \Re^+$. A high utility value for a certain application specification T_j means that U derives a high utility from executing the corresponding task. This can either be specified by the user if it denotes the importance of the task to him/her or can be learnt historically from frequency of usage patterns.

Computing Aggregate Utility Given P and Q, the aggregate utility γ of the entire network of devices with respect to the set of application specifications can be computed by

$$\gamma(V, A_\psi, C) = \sum_{t \in \tau} \{\Theta(\tau, V, (A_\psi, C))Q(\tau)\} \tag{2}$$

In other words, aggregate utility of a network reflects the performance that an average user can expect to get from the network during the execution of a desired set of tasks.

Computing Marginal Utility Marginal utility of a certain device D is the gain in aggregate utility of the network due to the addition of that device to the network. From Equation 2 we can derive marginal utility μ as follows:

$$\mu(D) = \gamma(V \cup D, A_\psi, C) - \gamma(V, A_\psi, C) \tag{3}$$

A high value of $\mu(D)$ means that the addition of a device D to the network increases the value of the network significantly with respect to the possibility of execution of a given set of tasks. On the other hand a low value of $\mu(D)$ indicates that addition of a device does not result in any value addition with respect to task execution. This information can be put to good use by the administrator of the network (in case of a conference like scenario where a part of the network is managed by the conference organizers) to perform capacity planning. However, if the network is not managed by anybody, this information can still be used effectively. We show later in this section how this can be achieved.

A major challenge lies in the efficient computation of the aggregate and marginal utilities of the network with respect to a certain set of user tasks and their importance to the user. Specific challenges include:

1. The efficient computation of a good mapping function while obeying the constraints in C.

2. Computation of the function $\Theta(.)$ which can involve assessing the current conditions in the network in order to give a reasonably accurate picture of the performance.

Use of Utility Computation in Unmanaged Networks In an unmanaged network, there are no administrators but the users themselves. Since users want to perform a set of tasks on such networks according to their profiles, information regarding network application performance can be valuable to them to predict if there are enough resources available in the network in order to perform their set of tasks. This can be achieved in an on-demand fashion. Suppose user U_1 is assessing the network resources for suitability of task execution. Meanwhile users U_2 and U_3 who both possess certain resources that U_1 needs enter the network. If there is a quick and message efficient way of computing the marginal utility of U_2 and U_3 towards U_1, then the latter can assess the network again and deduce from the new utility computation whether the increase in marginal utility was enough for task execution. Naturally, in order to achieve this we need to be able to compute γ and μ incrementally without utilizing much network resources in the process. We recognize that this is indeed a major challenge. However, if certain approximations are made in the mapping algorithm, γ and μ can be computed incrementally without altering the current mapping for the existing devices.

4 Conclusion

Our challenge to the research community is to present a formulation of the utility of a network of devices that can be used to gauge the priority given to each resource, as their addition to, replenishment in, or removal from the network is considered. However, we argue that such formulation of utility cannot be tied only to metrics like bandwidth, storage or CPU cycles, but must take into account the ability of the network of devices of accomplishing user "tasks." Therefore we propose that an important step to tackle the problem is to design an application specification model that can support the full expressiveness of any task abstraction, while containing enough information that allows mapping functions to determine if an application specification can be executed by a set of devices, given the constraints. Selection of efficient mapping functions is crucial because these in effect dictate whether an application can be executed on the set of devices. The utility metric of the network then becomes a function of both the number of instances mapped from all the application space and the mapping function itself.

References

[1] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai. "IEEE 802.11 Wireless Local Area Networks". *IEEE Communications Magazine*, 35(9):116–126, September 1997.

[2] Bluetooth Consortium. URL. `http://www.bluetooth.com`.

[3] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman, and D. Zukowski. "Challenges: An Application Model for Pervasive Computing". In *Proceedings of the 6th ACM MobiCom Conference*, Boston, MA, August 2000.

[4] G. Banavar and A. Bernstein. Software infrastructure and design challenges for ubiquitous computing applications. *Communications ACM*, November 2002.

[5] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. "The Design and Implementation of an Intentional Naming System". In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP)*, Kiawah Island, SC, December 1999.

[6] P. Basu, W. Ke, and T. D. C. Little. "A Novel Approach for Execution of Distributed Tasks on Mobile Ad Hoc Networks". In *Proceedings of the IEEE Wireless Computing and Networking Conference (WCNC)*, Orlando, FL, March 2002.

[7] M. Esler, J. Hightower, T. Anderson, and G. Borriello. "Next Century Challenges: Data-Centric Networking for Invisible Computing The Portolano Project at the University

of Washington". In *Proceedings of the 5th ACM MobiCom Conference*, Seattle, WA, August 1999.

[8] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. "Next Century Challenges: Scalable Coordination in Sensor Networks". In *Proceedings of the 5th ACM MobiCom Conference*, Seattle, WA, August 1999.

[9] G. P. Picco, A. L. Murphy, and G-C. Roman. LIME: Linda meets mobility. In *Proc. 21st International Conference on Software Engineering (ICSE 1999)*, pages 368–377, Los Angeles, CA, USA, May 1999.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[11] C. Borcea, C. Intanagonwiwat, D. Iyer, P. Kang, A. Saxena, U. Kremer, and L. Iftode. Spatial programming using smart messages: Design, implementation and evaluation. Technical Report DCS-TR-490, Rutgers University, October 2002.

[12] E. Guttman. "Service Location Protocol: Automatic Discovery of IP Network Services". *IEEE Internet Computing*, July 1999.

[13] Sun Microsystems: Jini Technology Core Platform Specification. URL. `http://www.sun.com/jini/specs`.

[14] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *nternational J. Supercomputer Applications*, 15(3), 2001.

[15] T. Phan, L. Huang, and C. Dulan. Challenges: Integrating mobile wireless devices into the computational grid. In *Proceedings of the eighth annual international conference on Mobile computing and networking*, pages 271–278, Atlanta, Georgia, USA, 2002. ACM Press.

[16] S. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent System. Special Issue on the Semantic Web*, 16(2), March/April 2001.

[17] M. Uschold and M. Grüninger. Ontologies: Principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, June 1996.

[18] D. Fensel, F. van Harmelen, I. Horrocks, and D. L. McGuinness an d P. F. Patel-Schneider. Oil: An ontology infrastructure for the semantic web. *IEEE Intelligent System. Special Issue on the Semantic Web*, 16(2), March/April 2001.

[19] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Special Issue on Grid Computing Environments, The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, May 2002.

[20] J. Bredin, R. T. Maheswaran, Ç. Imer, T. Başar, D. Kotz, and D. Rus. A game-theoretic formulation of multi-agent resource allocation. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 349–356. ACM Press, June 2000.

[21] Y. Amir, B. Awerbuch, and R. S. Borgstrom. A cost-benefit framework for online management of a metacomputing system. *The International Journal for Decision Support Systems*, 28(1-2):155–164, April 2000.

[22] B. Chun and D. Culler. User-centric performance analysis of market-based cluster batch schedulers. In *2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002)*, May 2002.

[23] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. Daml-s: Web service description for the semantic web. In *Proceedings of the first International Semantic Web Conference (ISWC 2002)*, Sardinia, Italia, June 2002.

[24] D. L. McGuinness. "Conceptual Modeling for Distributed Ontology Environments". In *Proc. Eigth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues (ICCS 2000)*, Darmstadt, Germany, August 2000.

[25] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2), 1993.

[26] E. A. Lee. Overview of the Ptolemy Project. Technical report, University of California, Berkeley, March 2001. Technical Report Technical Memorandum UCB/ERL M01/11.

[27] E. A. Lee and T. M. Parks. Dataflow process networks. In *Proceedings of the IEEE*, volume 83, May 1995.

[28] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Company, San Francisco, CA, USA, 1979.

[29] P. Basu, W. Ke, and T. D. C. Little. A Novel Approach for Execution of Distributed Tasks on Mobile Ad Hoc Networks. Technical report, Boston University, July 2001. MCL Technical Report No. 07-22-2001.