Integrated Security and Quality of Service Support in E-Commerce

Kyoung-Don Kang
Department of Computer Science
University of Virginia
kk7v@cs.virginia.edu

Abstract

One of the main challenges against successful e-commerce in an open environment is security and QoS (Quality of Service) support. A security compromise can lead to the failure of an e-business in the worst case. At the same time, an e-commerce system needs to process service requests with a certain service quality. Security and QoS support have hardly been studied in an integrated manner despite the increasing demand. To address this problem, we extend the role-based access control model, which is a popular access control model used to ensure only authorized users are allowed to access computational resources, to support the required security and QoS properties in a single framework. Our extended model, called QoS-aware Role-Based Access Control model (QRBAC), provides a novel notion of QoS-aware roles to let a system administrator directly specify resource quotas, e.g., the CPU cycle and network bandwidth quotas, needed to process service requests with the required QoS. We introduce the notion of system status as part of the access control model to detect overloads or violations of the access control policy. When overloaded, QRBAC can dynamically adjust the permission (in terms of resource quotas) and the role activation rate (the rate at which roles are actually assigned to their permissions), if necessary, to reduce resource consumptions. It can also revoke the permissions assigned to suspicious roles violating the access control policy to protect well-behaving roles. In this way, QRBAC can support the required QoS and security properties in terms of access control. In this paper, we motivate the need for integrated security and QoS support in one framework, and present a high level description of QRBAC. We also show, via examples, how QRBAC can provide security and QoS support in e-commerce.

Keywords: *Security, QoS, Role-Based Access Control*

1 Introduction

The growth of e-commerce has been dramatic. It has been reported that e-commerce trades totaled $132 billion in 2000−more than double the $58 billion reported in 1999 [1]. Behind the increasing impact of e-commerce on the economy, there are growing concerns about security and quality of service. E-commerce is usually performed in an open environment in which all possible security problems co-exist. On the other hand, e-commerce applications are sensitive to QoS such as the average response time. For example, late processing of service requests, possibly due to overloads, caused nearly $420 million in losses in 1999 [5]. Many potential clients may leave e-commerce sites when the service response time is longer than a certain time such as 5 sec [2].

Security and QoS research have *separately* produced abundant, successful results. However, security and QoS support can not be handled separately in e-commerce. A completely secure e-commerce system, if any, may not be useful if it significantly violates the required QoS. Alternatively, a well developed QoS management scheme without proper security support can be easily compromised by a malicious user launching a denial of service (DoS) attack, which may exhaust all system resources.[1] Due to the lack of available resources, the required QoS can be significantly violated resulting in loss of profit. Without integrated security and QoS support, a system administrator should specify and monitor the required security and QoS properties separately even if a system supports both security and QoS. However, separate configuration and maintenance of security and QoS can be computationally expensive and error-prone.[2]

[1] Other types of attacks, e.g., stack overflow attacks exploiting program flaws, are beyond the scope of this paper. We assume critical e-commerce systems are implemented properly to avoid well known attacks exploiting program flaws.

[2] In the worst case, security and QoS requirements may pose conflicting requirements due to misconfiguration. This can be a serious problem because it is known that many security vulnerabilities are due to incorrect system configuration [9].

To bridge the gap between security and QoS support, we present a novel role-based access control model, called QoS-aware Role-Based Access Control model (QRBAC), to support QoS and security in a single, seamless framework. Role-based access control model (RBAC) [11] is one of the most efficient access control schemes used to ensure that only authorized users can access certain resources (and information).[3] A role is a semantic entity, which is the basis of an access control policy. Using RBAC, a system administrator can create roles considering the job functions performed in an organization, e.g., client, teller, and finance manager roles in a bank, and grant permissions (access authorizations) to these roles. An administrator can assign users to the roles considering their job responsibilities and qualifications. Compared to existing access control mechanisms such as the access control list, RBAC significantly reduces the difficulty to configure and maintain an access control mechanism. For example, if a teller in a bank is promoted as a finance manager, the administrator can easily reassign her to a new role of the finance manager instead of revoking all access permissions previously assigned to the teller job and reassigning permissions necessary to perform the job of a finance manager. RBAC can efficiently handle these problems because permissions assigned to roles, unlike user memberships in roles, tend to change relatively infrequently [6]. Further, RBAC supports well known security principles such as least privilege and separation of duties (SoD). Each role receives only those permissions required to perform the job functions. Several roles, e.g., client and teller roles, are mutually exclusive; that is, a user should not be allowed to take client and teller roles at the same time.

Although RBAC significantly reduces the maintenance cost of an access control mechanism and supports important security principles, it is not expressive enough for e-commerce applications that need to process service requests in a QoS-aware manner. Using RBAC, an administrator can not specify the resource quotas needed for roles, e.g., catalog browsing roles, to support the required QoS, e.g., the average response time and/or minimum acceptable quality of catalog images.[4] Therefore, the required QoS may not be supported, especially when overloaded. Further, an access control mechanism based on RBAC does not provide direct support for resource usage monitoring/policing to prevent malicious roles from exhausting system resources, incurring DoS.

QRBAC introduces a novel notion of *QoS-aware roles* to let an administrator directly specify the required resource quotas for e-commerce roles to support the required service quality. QRBAC defines the notion of system status as part of the access control model. In this way, an access control mechanism can directly monitor the system status to detect overloads, if any. We extend the original principles of least privilege and SoD to consider resource consumptions in a QoS-aware e-commerce system; that is, we consider a role has violated the least privilege if it has overused its quotas. Consequently, other roles may have to receive the reduced QoS due to the lack of computational resources. This violates the original intention of performance isolation among roles, which we consider a SoD problem in QoS-aware e-commerce applications. To support the extended notions of least privilege and SoD, in addition to the original notions, QRBAC can dynamically adjust the permissions in terms of resource quotas assigned to certain roles such that they receive the degraded QoS, e.g., reduced image quality, when overloaded. QRBAC can dynamically decrease the activation rate, i.e., the number of roles actually given their permissions to use resources over a certain period of time, of relatively less critical roles if the overload conditions continue after the QoS degradation described before. Further, the access control mechanism can immediately revoke the permission of a role, which causes DoS by significantly exceeding the allocated resource quotas specified in the permission, to protect other well-behaving roles. In this way, QRBAC can support the extended principles of least privilege and SoD, while supporting the required QoS in a single framework.

In Section 2, background information about RBAC, a basis of our work integrating QoS and security support in a single framework, is given. In Section 3, QRBAC is presented in detail. A realistic example of using QRBAC in an e-commerce system is also given to show the applicability of QRBAC. Section 4 discusses the related work. Section 5 concludes the paper and discusses the avenues for future work.

2 Role-Based Access Control Model

Figure 1 shows the basic RBAC model called RBAC_0 in [11]. Users are assigned to roles representing job functions, e.g., teller or finance manager roles. Each role is associated with specific permissions describing the access rights authorized for a member of the role. Using roles between users and permissions, rather than directly assigning users to permissions, it significantly reduces the difficulty of configuring and maintaining the access control mechanism as discussed before.

The RBAC_0 model consists of the following components [11]:

- U, R, P, and S represent users, roles, permissions, and sessions, respectively;

[3] In this paper, we assume users are already authenticated, e.g., via passwords or mutual authentication mechanisms such as Kerberos [10], in which users and service providers can authenticate each other.

[4] In this paper, we mainly consider roles e-commerce service entities providing specific services, e.g., catalog browsing and purchase.

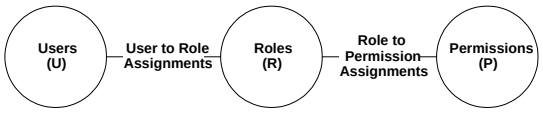

Figure 1. RBAC$_0$: A Basic Role-Based Access Control Model

- UR ($\subseteq U \times R$) represents a many-to-many user to role assignment relation, which is a subset of the Cartesian product $U \times R$;
- RP ($\subseteq R \times P$) is a many-to-many role to permission assignment relation, which is a subset of the Cartesian product $R \times P$;
- $user : S \rightarrow U$ is a function mapping each session s_i to the single user $user(s_i)$ for the lifetime of the session; and
- $roles : S \rightarrow 2^R$ is a function that maps each session s_i to a set of roles $roles(s_i) \subseteq \{r|(user(s_i), r) \in UR\}$, while s_i has the permissions $\cup_{r \in roles(s_i)}\{p|(r,p) \in RP\}$.

User-to-role assignment (UR) and role-to-permission assignment (RP) are many-to-many relations. Therefore, one user can have many roles and one role can have many members. Further, one role can have many permissions and the same permission can be authorized for many roles.

In $RBAC_0$, a user needs to establish a session during which she can activate a subset of roles she belongs to. A user can activate multiple roles in one session and she has the union of all permissions that are assigned to the activated roles. Each session is related to one user, but one user can open multiple sessions at the same time, if necessary.

In RBAC$_0$, the (original) principles of least privilege and SoD are left to the user's discretion. If a user has a powerful role, she can use that role to perform her tasks possibly violating the least privilege principle. In addition, there is no constraint on user to role assignment. Therefore, a user simultaneously can take several roles, e.g., teller and client, which should be mutually exclusive. Extended versions of RBAC$_0$ presented in [11] provides the notion of constraints−key motivation behind RBAC−to describe least privilege and SoD properties.[5] However, they are neither powerful enough to support the (extended) principles of least privilege and SoD in QoS-sensitive e-commerce applications. To our best knowledge, there is no existing extension of RBAC that can directly support QoS and security requirements at the same time.

[5] In the remainder of this paper, we assume RBAC supports the original principles of least privilege and SoD. Therefore, we mainly focus on the issues related to extending RBAC to support the extended principles of least privilege and SoD.

3 QoS-Aware Role-Based Access Control Model

In this section, we present QRBAC that extends RBAC to directly specify resource quotas and support the dynamic adjustment of permissions and role activations, if necessary, to support the required QoS and security properties in a specific e-commerce application. Further, we give a realistic example of applying QRBAC for QoS and security support in e-commerce.

3.1 Definition of QRBAC

In designing QRBAC, we consider a typical QoS management architecture, which may include performance/resource usage monitoring, QoS degradation, admission control, and differentiated service schemes, similar to [3, 8, 13]. QoS management schemes usually aim to support a certain performance, e.g., average response time, or to limit the resource usage to be below a certain threshold to avoid overloads [3, 13], which can cause QoS violations. To detect overloads, a QoS management scheme usually monitors the system performance or resource usage. When the system is overloaded, the QoS can be degraded to reduce the workload. For example, a QoS-aware web server can only provide textual information under overload conditions. (Image data can be serviced when the load decreases.) Admission control can be applied to the incoming requests when the system is severely overloaded. In addition, users can be classified into several classes considering their importance such as the payments. When overloaded, the system can differentiate the service, e.g., by differentiating the service rate among the classes, considering the relative importance of service requests. Note that QRBAC supports all these QoS management mechanisms for the general applicability of QRBAC to support the required QoS within the role-based access control mechanism. Further, an administrator can selectively use the features of QRBAC to support the QoS and security properties needed for the specific e-business that she administers.

Definition 3.1 *The QRBAC model has the following components:*

- U_Q, R_Q, P_Q, S_Q, L, and C: QoS-aware users, roles, permissions, sessions, QoS levels, and service classes, respectively;
- *QoS-aware users* ($\in U_Q$) can be classified according to their importance, if necessary, to differentiate

the service especially when overloaded. A function $class\_of\_user : U_Q \rightarrow C$ maps each user u to a single service class i where $1 \leq i \leq k$ and k is the number of the service classes. Note that $U_1 \cap ...U_i \cap ...U_k = \emptyset$ and $U_Q = U_1 \cup ...U_i \cup ...U_k$ where U_i is the set of users belonging to the service class i. A violation of this constraint can be considered an intrusion in which a malicious user tries to subvert the differentiated service scheme, if any (i.e., $k \geq 2$).

- *QoS-aware roles* ($\in R_Q$) can be classified to support users with different importance, if the service is differentiated. A function $class\_of\_role : R_Q \rightarrow C$ maps each role r to a single service class i. Thus, $R_1 \cap ...R_i \cap ...R_k = \emptyset$ and $R_Q = R_1 \cup ...R_i \cup ...R_k$ where R_i is the set of roles belonging to the service class i. A violation of this constraint can be considered an intrusion, similar to the potential user-to-class assignment subversion described before.[6]

- A *QoS-aware permission* ($\in P_Q$) is a list of resource quotas $(q_1, ..., q_m)$ available for the roles belonging to one service class when there are m (≥ 1) types of resources mainly considered in an e-commerce system. For example, one permission can specify the CPU utilization, memory space, and disk/network bandwidth quotas allocated for one service class. Given the resource quotas, the system can support a certain QoS level, e.g., in terms of average response time. Therefore, a function $QoS\_level : P_Q \rightarrow L$ maps each permission to a single QoS level assuming that roles assigned to the permission do not exceed the resource quotas. By enforcing the resource quotas specified in each permission, the system can prevent one service class from exhausting all resources, denying service to the other class(es). QRBAC also defines $P_{one\_user} = a \times (q_1, ..., q_m)$ and $P_{one\_role} = b \times (q_1, ..., q_m)$ where a and b ($0 \leq a, b \leq 1$) are specified by the system administrator. P_{one_user} and P_{one_role} are necessary to prevent that a single user or role, providing a specific service, exhausts all resources denying service to other users or roles.

- $user : S_Q \rightarrow U_Q$ is a function mapping each *QoS-aware session* s_i to the single user $user(s_i)$ for the lifetime of the session. A user needs to open a session to request a service, similar to RBAC. Further, a session s_i is associated with a certain timeout value t and it will be closed after a predefined idle time to service other users, if any.

[6]Several roles can provide the same service such as real-time stock quotes with different QoS, e.g., average response time, according to the users' importance. Instead of having separate roles providing the same service for different classes, one can dynamically create separate instances of a single role providing the service, for example, with different scheduling priorities. We reserve this for future work.

Definition 3.2 *The QRBAC model has the following QoS-aware user-to-role and role-to-permission assignments:*

- UR_Q ($\subseteq U_Q \times R_Q$) represents a many-to-many QoS-aware user-to-role assignment relation, which is a subset of the Cartesian product $U_Q \times R_Q$. Note that $class\_of\_user(u) = class\_of\_role(r)$ for each user and role pair $(u, r) \in UR_Q$. Therefore, users are mapped to the roles in the same service class and vice versa, if the service is differentiated.

- RP_Q ($\subseteq R_Q \times P_Q$) represents a many-to-many QoS-aware role-to-permission assignment relation, which is a subset of the Cartesian product $R_Q \times P_Q$. Note that we assign all class i roles to one permission. Thus, class i roles share the resource quotas authorized for that class. In this way, QRBAC greatly simplifies the configuration and maintenance of the access control mechanism, while policing the resource consumption of an individual user or role to keep it from exhausting resources as described in Definition 3.1.

To support the required security and QoS properties, UR_Q and RP_Q are further subject to the system status constraints defined next.

Definition 3.3 *To detect potential overloads, the QRBAC model defines the following system status:*

- System status $V = \{v_1, v_2, ..., v_m\}$ is the set of current resource utilization measured periodically, e.g., at every minute, for m (≥ 1) different types of resources. At a measurement instant, V is compared with the set of utilization thresholds $\Theta = \{\theta_1, \theta_2, ..., \theta_m\}$ defined for QoS management. If $v_i > \theta_i$, e.g., CPU utilization $\geq$ 80%, the system can be considered overloaded.

- V is replaced with the list $(V_1, ..., V_k)$ when the service is differentiated among k (≥ 2) classes. V_i represents the resource usages measured in class i. The system can be considered overloaded if $v_{i,j} > \theta_{i,j}$ where $v_{i,j}$ is the class i's usage of resource j and $\theta_{i,j}$ is the corresponding threshold ($1 \leq i \leq k$ and $1 \leq j \leq m$).

There are alternative approaches to monitor the system status and detect overloads. For example, overloads can be detected if the measured (per-class) average response time is longer than the required (per-class) average response time. Note that we neither try to cover all possible approaches for overload detection, nor require a system administrator to use a specific approach. Instead, we let the administrator choose an appropriate method considering the

specific QoS management scheme used for the e-business system that she is administering. Therefore, in the remainder of this paper we may refer to overload conditions without specifying the exact overload detection scheme for presentation purposes. Once detected, overloads can be managed as follows.

Definition 3.4 *QRBAC provides the following dynamic permission and role activation rate adjustment, if necessary, to manage overloads and support the (extended) principles of least privilege and SoD:*

- When the system is overloaded, QRBAC can dynamically adjust RP_Q to degrade QoS, if necessary, by decreasing the resource quotas specified in (a) certain permission(s). For example, under overload the permission associated with a catalog browsing role, which supports relatively less important users, can be adjusted to provide low quality images of merchandise, if necessary, to reduce workloads.
- Admission control can also be applied to newly incoming service requests when the system is still overloaded after the QoS degradation described before. To support admission control, QRBAC can decrease the acceptance rate of incoming session open requests until the system is not overloaded anymore. Consequently, the role activation rate is decreased for those users.

QRBAC can support the least privilege and SoD principles as long as the per-class permission (P_Q), single user permission (P_{one_user}), and single role permission (P_{one_role}) are enforced, while avoiding overloads. This is because roles in one class use only the allocated quotas without adversely affecting the QoS perceived by other service class(es). Further, each user or role does not deny service to other roles without exceeding the maximal available quota for a single user or role. Therefore, QRBAC can support security and QoS in a single, seamless framework.

3.2 An Example of an QRBAC Application

To show the applicability of QRBAC to e-commerce, we consider a simplified online shop, in which clients may browse catalogs, quote prices, and buy items. The main objective of QoS management is to limit the system utilization to be below a certain threshold to avoid overloads and process service requests in a timely manner, as a result. For the clarity of presentation, we assume the network bandwidth is the bottleneck resource whose utilization should be closely monitored.

As shown in Figure 2, we assume the QoS management scheme provides two service classes for the clarity of presentation. Similarly, we assume there is only one adaptable QoS dimension, i.e., product image quality, with two QoS levels, i.e., high and low. Further, all roles for catalog browsing, price quote, and purchase have two versions to support the premium and basic class, respectively (Definition 3.1).

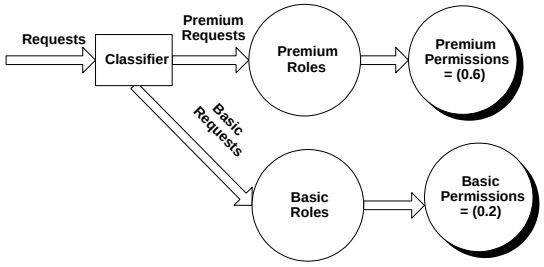

Figure 2. An Application of QRBAC

As shown in Figure 2, the Classifier classifies an incoming service request, e.g., based on the user id or IP address, into the premium or basic class and assigns it to a role providing the requested service for the users in the corresponding service class (Definition 3.2). All roles in the premium (basic) class are associated with the premium (basic) permission, which allocates 60% (20%) of network bandwidth.

When the bandwidth quota of the basic class is violated, i.e., the bandwidth consumption of the basic class $V_2 > 20\%$, the system is considered overloaded (Definition 3.3). The system can adjust the permission to degrade the image quality for the basic class, thereby reducing the bandwidth consumption (Definition 3.4). If V_2 is still higher than 20% after the QoS degradation, the activation rate of the basic class roles can be decreased possibly in proportion to $V_2 - 20\%$.[7] When the premium class consumes more than 60% of the total bandwidth, QoS degradation and admission control are applied in a similar manner. Further, the resource utilization of individual users and roles are monitored for policing purposes. In this way, both classes only receive the least privilege considering the current system status and their resource consumptions. Thus, the performance of each class is isolated from the other class, while one user or role is kept from exhausting resources. In addition, maintenance of QoS management and access control schemes is simple and efficient. Using QRBAC, an administrator can simply modify the premium and basic permissions to reallocate the bandwidth between these classes.

From this example, we can observe that QRBAC can let an administrator directly specify resource quotas and support the required QoS and principles of least privilege and SoD by applying Definitions 3.1– 3.4.

[7] Determining a specific degree of session activation rate adjustment depends on a specific QoS management and admission control scheme.

4 Related Work

Covington et al. have recently presented the Generalized Role-Based Access Control model in the context of the smart home [4]. They introduce the novel notion of environment roles to capture the security related conditions of the environment where access requests are issued. An access request to a certain object modeled as an environment role, e.g., an electric appliance, will only be granted if the request is issued from a specific location at a certain time. Although environment roles can be conditionally activated, their access control model does not support resource quota specification, dynamic adjustment of the permission and role activation rate under overloads, and resource usage monitoring/policing for QoS management as QRBAC does.

Joshi et al. have presented the Generalized Temporal Role-Based Access Control model [7], which can allow an administrator to specify a comprehensive set of temporal constraints for access control. In their scheme, a role can be configured to get enabled during a certain time period, e.g., between 9am and 5pm for a daytime doctor in a hospital, to limit the resource usage. However, their model is not directly applicable to e-commerce because users, especially B2C users, may request services at any time. Therefore, defining a time interval for the activation of a role representing certain users is very hard, if at all possible. Also, they do not consider the dynamic adjustment of the permission/role activation rate and resource usage monitoring as we do.

Spyropoulou et al. have considered the problem of calculating the cost for security service [12]. Different security mechanisms, e.g., various encryption algorithms, may support different quality of security service, while consuming different amounts of computational resources. Ideally, an administrator can select an appropriate security mechanism considering its cost and benefit. However, the quantitative cost-benefit analysis of security service remains an open problem yet. Considering the cost, the overhead of QRBAC can be minimal compared to commodity access control mechanisms that do not apply the notion of roles. QRBAC could incur very little overhead for configuration and maintenance, since it significantly reduces the number of permissions to manage by mapping a group of roles in one class to a single permission. In addition, the adjustment of the permission and role activation rate for QoS management is handled gracefully supporting the principles of least privilege and SoD extended for secure, QoS-aware e-commerce applications.

5 Conclusions and Future Work

Integrated security and QoS support has attracted relatively little research interests despite the increasing importance. In this paper, we address this problem by extending the role-based access control model to let a system administrator explicitly specify resource quotas needed for QoS support. Our extended role-based access control model, QRBAC, supports both the security and QoS properties via QoS-aware roles, QoS-aware permissions, and dynamic adjustment of the permission and role activation rate, if necessary. In this way, QRBAC can minimize risks of resource exhaustion attacks. QRBAC can also support important security principles such as least privilege and separation of duties, while supporting the required QoS. We have presented a theoretic foundation of QRBAC and shown the applicability of QRBAC via examples. The work presented in this paper is an initial result of our on-going research aiming to integrate security and QoS support in a single, seamless framework. In the future, we will further extend QRBAC to let a role create multiple instances mapped to different permissions, if necessary, to differentiate the service. In addition, we plan to investigate the semantics of QoS-aware role hierarchies in the context of e-commerce.

References

[1] Active Media Research, Real Numbers behind 'Net Profits. http://www.activmediaresearch.com/.

[2] N. Bhatti, A. Bouch, and A. Kuchinsky. Integrating User-Perceived Quality into Web Server Design. In *the 9th International World Wide Web Conference*, 2000.

[3] N. Bhatti and R. Friedrich. Web Server Support for Tiered Services. *IEEE Network*, 13(5):64–71, September 1999.

[4] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. In *ACM Symposium on Access Control Models and Technologies*, 2001.

[5] E-Commerce Statistics. http://www.zdnet.com/ecommerce/stories/main/0, 10475, 2636088, 00.html.

[6] D. F. Ferraiolo, D. M. Gilbert, and N. Lynch. An Examination of Federal and Commercial Access Control Policy Needs. In *NIST-NCSC National Computer Security Conference*, pages 107–116. National Institute of Standards and Technology, 1993.

[7] J. B. D. Joshi and E. Bertino and A. Ghafoor. Temporal Hierarchies and Inheritance Semantics for GTRBAC. In *ACM Symposium on Access Control Models and Technologies*, 2002.

[8] K. D. Kang, S. H. Son, and J. A. Stankovic. Differentiated Real-Time Data Services for E-Commerce Applications. *Electronic Commerce Research*, 3(1-2), January/April 2003. Combined Special Issue: Business Process Integration and E-Commerce Infrastructure.

[9] S. McClure, J. Scambray, and G. Kurtz. *Hacking Exposed*. McGraw Hill, 4th edition, 2003.

[10] B. C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications*, 32(9), Sept. 1994.

[11] R. S. Sandhu. Role-based access control models. *IEEE Computer*, 2, Feb. 1996.

[12] E. Spyropoulou, T. Levin, and C. Irvine. Calculating Costs for Quality of Security Service. In *15th Computer Security Applications Conference*, 2000.

[13] R. Zhang, T. F. Abdelzaher, and J. A. Stankovic. Kernel Support for Open QoS-Aware Computing. In *IEEE Real-Time/Embedded Technology And Applications Symposium*, 2003. To appear.