Institut Eurécom[1]
Mobile Communications Department
2229, route des Crêtes
B.P. 193
06904 Sophia Antipolis
FRANCE

Research Report RR-08-214

A Markovian Performance Model for Secure Service Discovery Systems

7 March 2008

Slim Trabelsi, Guillaume Urvoy-Keller , and Yves Roudier

Tel: (+33) 4 93 00 81 00
Fax: (+33) 4 93 00 82 00
Email: {Slim.Trabelsi, urvoy, Yves.Roudier}@eurecom.fr

[1] Eurecom's research is partially supported by its industrial partners: BMW, Bouygues Télécom, Cisco Systems, France Télécom , Hitachi Europe, SFR, Sharp, STMicroelectronics, Swisscom, Thales

Abstract

SOA supported pervasive applications introduce new threats to service discovery. Securing service discovery impacts performance and scalability, yet the additional performance and scalability costs of solutions proposed so far have not been evaluated analytically nor by simulation. This paper addresses this problem by introducing a Markovian application layer performance model for centralized and decentralized secure service discovery, which is then validated through simulation.

1 Introduction

The deployment of ubiquitous computing systems, as envisioned by [1] for instance, and the trend towards Service Oriented Architectures will undoubtedly generalize the need for discovery mechanisms as essential components for locating ambient and location-based services. Service discovery in a network can be implemented in two manners, first using a decentralized architecture relying on point to point (broadcast) or point to multipoint (multicast) communication, and second using a centralized architecture based on an identified registry relied upon by users and servers to facilitate discovery request matching. The choice of the appropriate architecture to enable an efficient service discovery highly depends on the deployment environment (LAN, wireless or ad-hoc communications, Internet, VPN, etc.) and on parameters like the expected number of users and services, the type and amount of resources available (CPU, memory …), and the power consumption. The performance of discovery mechanisms has already been studied through simulation. [2] and [3] present an evaluation of the performance of post-query discovery strategies in ad-hoc networks in which the authors test five strategies with the DSR and DSDV routing protocols. [4] introduces a service discovery performance model that makes it possible to predict discovery service failure and overloading in real time. This work presents simulation results that suggest that a decentralized architecture yields better robustness than a centralized one. These studies aim at getting a better understanding of the phenomena observed during discovery like message loss, faults, delays, or saturation, so as to select the most efficient service discovery mechanism for a given application. Numerous service discovery standards like WS-Discovery, Jini, UPnP, SLP, or UDDI (see Table 1) have also been proposed in recent years, even though their performance has not been assessed analytically to our knowledge. Security in these standards is usually limited to recommendations about classical message authentication and integrity protection, thereby implicitly restricting discovery to known services. This approach falls short for taking into account the increasing use of discovery in open ubiquitous computing scenarios with numerous new threats [5] [6]. We presented security mechanisms to deal with such issues in [7] and [8]. This paper introduces Markovian models that aim at assessing the impact at the application level of introducing security mechanisms, for both centralized and decentralized service discovery. Focusing on the application level, i.e., neglecting network artifacts such as delay or losses enables us to delineate network effects from the impact of security mechanisms in terms of processing overhead for the nodes in the system.

Table 1. Discovery Protocols diffusion mechanisms

	Centralized	Decentralized	Unicast	Multicast	Broadcast
SLP	Yes	Yes	No	Yes	No
UPnP	No	Yes	No	Yes	No
UDDI	Yes	No	Yes	No	No
Jini	Yes	Yes	Yes	Yes	No
SDP Bluetooth	No	Yes	No	No	Yes
WS-Discovery	Yes	Yes	Yes	Yes	No
Salutation	Yes	No	Yes	No	No

This paper is organized as follows: Section 2 describes security and networking assumptions that are accounted for by our model. Section 3 details our Markovian model whose validation is then described in Section 4.

2 Modeling Secure Service Discovery

We essentially focus on Service Oriented Architectures (SOA), which introduce a loosely coupled interaction model that serves as a basis to define protocols and procedures to interconnect different application systems or software components. SOA mainly consists of services, which are software wrapped components providing elaborate functions (e.g., database access, data processing, business logic…), and of clients, which are requesting such services through the exchange of messages. These two types of players rely on a standardized interface to communicate but do not necessarily share the same implementation platforms (programming language or OS). With the emergence of new dynamic networks, discovery techniques are being adapted in order to cope with the pervasive deployment of services onto this new infrastructure. This in particular stresses the need to locate and combine services to achieve a given task in an unknown environment. In this respect, service discovery is evolving from a simple brokering mechanism to a central composition component, with increasingly demanding security requirements, and whose performance should be better understood.

2.1 Service Discovery Basis

The main players of the discovery phase are: the service requester (client), which can be a human user or software and the service provider (server), which represents the entity providing one or multiple services that can be accessed by the clients.

Centralized Discovery. Centralized discovery approaches rely on a registry which plays the role of yellow pages, and which clients can refer to. The registry (or repository) is a database containing descriptions and references to some available services. Servers publish their services by contacting a registry, while clients discover published services by requesting a registry. A service advertises its capabilities (a set of attributes describing the service) to the registry, which will store them for a certain amount of time. A client contacts the registry to find a service by sending a request containing service preferences, which the registry tries to match with the most suitable provider found from the stored advertisements. In that approach, registries have to be considered by the services and the clients as a third trusted party.

Decentralized Discovery. Limiting service discovery to registry supported architecture that many standards SOA based services have adopted in their implementations is reductive in terms of network architecture and equipments (e.g., need to deploy specific equipments like registries). An alternative approach to service discovery exists that relies on peer to peer advertisements between services and clients (point to point and point to multipoint). In such an approach, clients discover services by broadcasting their requests to their neighborhood, and if one of the neighbors features the requested service, it will directly respond; the neighbor may otherwise forward a request to its own neighborhood. This mechanism is used for instance by the P2P-based Web Service Discovery system (PWSD) [9], which relies on the Chord P2P protocol to perform the service discovery over the internet.

Secure Service Discovery. The first approach to securing service discovery is to rely on an infrastructure for establishing the trustworthiness of clients and services. In the work by Zhu et al. [10], each participant to the discovery protocol is located behind a trusted proxy that sets up trust relationships through key exchanges with other proxies. [12] suggests instead the use of a central entity that combines the roles of a Certificate Authority and registry, and help clients and servers to set up a trust relationship and established secure channels between each another. Concerning privacy Zhu et al [11] proposed a Bloom filter based matching aims at hiding private information related to client and services that could be exchanged during a service discovery process. These solutions are not adapted to decentralized architectures and focus more on servers than client security, contrary to [7] and [8] on which we focus. We detail those solutions along with their model hereafter.

2.2 Centralized Discovery

Security. Our security solution designed for a centralized configuration relies on security policies provided by clients and services. Registries have to be considered by services and clients as a trusted third party whose role is no more limited to a basic matchmaker, but it evolves to a security guarantor. In this configuration [7], clients and services first establish a secure connection (e.g., SSL) with the registry to protect the confidentiality of the exchanged messages. Servers can restrict the discovery of their services to only certified users by specifying a security discovery policy to be enforced by the registry. Clients are also able to restrict the matching scope to some certified services by specifying a security discovery policy also enforced by the trusted registry. Both clients and servers have to provide credentials issued by a known authority that can be used by the registry to authenticate them during the policy verification phase. More details on this architecture can be found in [7].

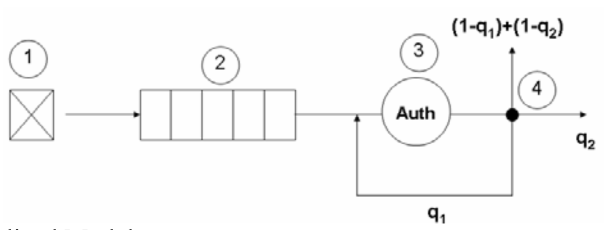

Fig. 1. Centralized Model

Model. Figure 1 presents the processing phase of a secure client service request at the registry for a centralized configuration in case of a single-threaded registry, a sole thread is in charge of all processing steps.

The discovery process consists of these steps:

1. Client service discovery requests arrival: requests are assumed to be generated according to arrival process with a rate λ.
2. Buffering: The registry can temporarily store the requests to be processed by the central unit. Messages are served in a FIFO manner.
3. Request processing: the registry first matches a client request with the service profiles available locally. The matched service will be authenticated in order to verify its compliance with the security policy provided by the client. If the verification is successful, the registry also has to further authenticate the client in order to verify its compliance with the security policy provided by the service. The corresponding service time is a random variable with a mean value $1/\mu$.
4. Probabilistic decisions (acceptance or rejection): q_1 is the probability that a service matches with a client request and also be compliant with its policy. q_2 is the probability that a client be compliant with this service policy.

2.3 Decentralized Model

Security. The security solution proposed in [8] for a decentralized configuration relies on a particular usage of the Identity Based Encryption mechanism [13]. The server advertises its service capabilities by multicasting its profile to the entire network. Clients can cache service information or ask for a specific service by multicasting its requests to all available servers and only concerned services will respond to him. With no possible reliance on any third party in ad-hoc configurations, clients and servers now must assure their own secure service discovery using a particular encryption scheme. In [8], Attribute Based Encryption (ABE) [14] was adopted to make it possible for a server to encrypt its service description according to the restrictions imposed to users (i.e., only a class of users holding corresponding private keys will be able to access to services information). Clients also can use the same encryption mechanism in order to protect their request messages from unauthorized servers (i.e., only a class of servers is able to decrypt the request).

Model. Authentication and policy verification computing time are now replaced with encryption and decryption time. The fact that the request is routed using multicast adds complexity to the event handling. In a decentralized architecture, nodes usually have limited capacities as compared to a registry: For this reason, we considered in our model that servers do not buffer new requests when they are busy. In Figure 2, the execution takes this order:

1. Client service discovery request arrival: requests are generated according to an arrival process with rate λ.
2. Servers message processing: all the available servers are contacted by the client via multicast. Each of these servers has to decrypt the messages in order to authenticate and access to client's request. The time to decrypt is assumed to be a random variable with a mean value $1/\mu_1$.
3. Service authentication: q_1 is the probability to successfully decrypt a client request. In case of success the server has to encrypt the response message to the client.
4. Client authentication: q_2 is the success decryption probability of a client.

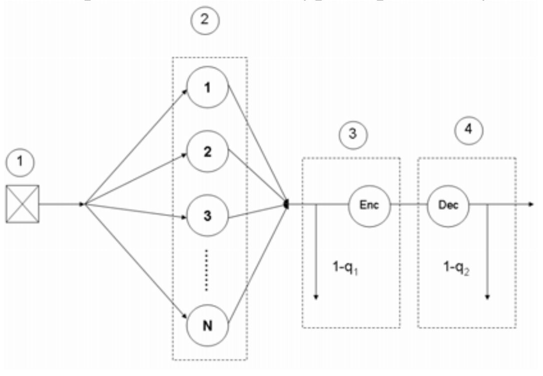

Fig. 2. Decentralized Model

2.4 System Model Assumptions

We make the following assumptions concerning the service demands and the processing for the service requesters and service providers.

- *Processing Time.* As described above, servers and clients must both perform some tasks during discovery. In the distributed configuration, processing time is essentially dedicated to encryption and decryption tasks. This processing time is variable for each message (message length, key length, padding size …), and also variable for the same message and the same encryption/decryption key. This variability is exemplified in [18], in which we can observe an event independent duration time for Encryption/ Decryption actions. We model processing time in our decentralized model as an exponentially distributed random variable with mean $1/\mu$. In the centralized configuration we observed that the processing time is strongly correlated with the policy size (number of conditions/attributes to be checked) and does not vary for a given policy size. We assume enough diversity in the distribution of policy sizes to model the processing time as a random variable that we further assume to be exponentially distributed for mathematical tractability in our Markov models.

- *Inter-arrival Time.* We assume clients request to follow a Poisson process with rate λ.
- *Traffic class.* For a decentralized scenario, some servers could be more popular than others. In this case, the matching probability with the clients request is higher for popular services. To model this popularity, traffic classes could be used to distinguish between these services. The model described in this paper assumes that all services have the same popularity, or to put it differently, focuses on the performance of an average client.

3 Markovian Model

In this section we present Markovian models for the centralized and decentralized secure service discovery systems. We use networks of queues to model both approaches.

3.1 Markovian centralized Model

For each request, the CPU of the registry is assumed to perform one or two authentication and policy verification cycles (since we assume that the registry is adopting a mono-threading strategy for the request processing). The first cycle corresponds to service authentication, while the second one corresponds to client authentication. We model these two cycles using a bi-dimensional Markov chain (see Figure 3).

The first dimension of the Markov chain (A) represents the number of requests stored in the cache and the number of requests currently processed (0 or 1).The second dimension of the Markov chain (B) is a Boolean representing the request in the second processing cycle. If $B = 1$, the parameter A represents the number of requests in the cache. If $B = 0$, A represents the number of requests in the cache plus one request in the first cycle processing state. For instance, the left upper state in Figure 3 corresponds to $A = 0$ and $B = 0$.

Markov Chain. Figure 3 is the Markovian representation of the centralized system outlined in Figure 1. Client requests are entering the system according to a Poisson process with rate λ. The parameter A is the first to be incremented. After an authentication and verification first cycle (exponential with rate μ), the system moves to the second authentication and verification cycle with a probability q_1 ($B = 1$) or the client is rejected with a probability ($1-q_1$). If $B = 1$ and a new request reaches the registry, only A will be incremented

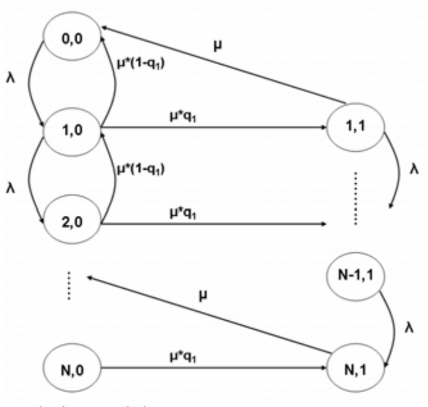

Fig. 3. Centralized Markov Chain Model

Numerical Resolution. The bi-dimensional Markov chain described above is not easy to resolve using balance equations for the stationary distribution. We used a transition rate matrix and transition rate diagram to resolve numerically the system with the Gauss-Seidel method. The transition rate matrix Q is written by:

$$Q = \begin{bmatrix} -\sum_{j\neq 1} T_{ij} & T_{12} & \cdots\cdots & T_{1j} \\ T_{21} & -\sum_{j\neq 2} T_{2j} & \cdots\cdots & | \\ | & \cdots\cdots & \cdots\cdots & | \\ T_{i1} & \cdots\cdots & \cdots\cdots & | \end{bmatrix}$$

Where T_{ij} is the transition rate between state i and state j
Q can be decomposed as:

$Q = L + U - D$; where D is a diagonal matrix, L is the lower part of Q and U is the upper part of Q. In a stationary regime the steady state probability vector P can be written as follows:

$$\begin{aligned} &P.Q = 0 \Rightarrow P.D = P.U + PL \\ &\Rightarrow \\ &P^{(n)} = \left(P^{(n-1)}L + P^{n}U\right)D^{-1} \\ &\Rightarrow \\ &P_j^{(n)} = \frac{1}{\sum_{i\neq j} T_{ji}}\left[\sum_{i<j} P_i^{(n)}T_{ij} + \sum_{i>j} P_i^{(n-1)}T_{ij}\right] \end{aligned} \tag{1}$$

The steady state vector P is used to calculate the different performance parameters of the system, including the rejection rate, the server usage rate, the mean number of users, the acceptance rate, the authentication rate, etc.

3.2 Markovian Decentralized Model

Since we do not account for network effects, especially delay and losses, we assume that each client request reaches all available servers simultaneously, through some multicast communication scheme. For each request arrival the number of busy servers is equal to the total number of servers in the system. Each server independently processes the request. After decrypting the message, a server will generate a response, encrypt it, and send it to the client.

Markov Chain.

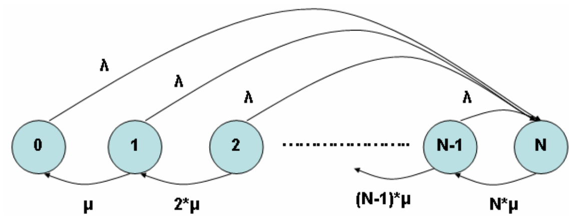

Fig. 4. Decentralized Markov Chain Model

The Markovian chain in Figure 4 represents parts 1 and 2 of the model described in Figure 2. Each state of this linear Markov chain represents the number of occupied servers. Part 3 and 4 of the Figure 2 are represented by the two states Markov chains in Figure 5.

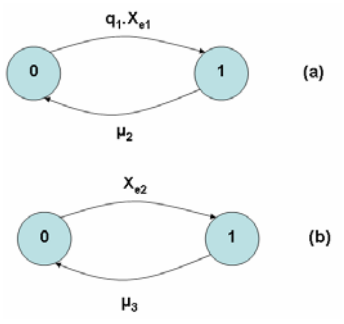

Fig. 5. a – Encryption Markov chain; b - Decryption Markov chain

The request arrival rate Xe_1 represents the output rate of the linear chain in Figure 4. The encryption Markov chain (Figure 5-a) is used to evaluate the impact of the server encrypting time on the system. Xe_2 represents the rate at which encrypted messages can be sent from servers. The decryption Markov Chain (Figure 5-b) is used to evaluate the impact of the decryption action performed by the client when it receives the encrypted response from the server.

Numerical resolution. The Markov chain representing the system is linear. For this reason, it is easy to calculate the steady state probability vector P using balance equations:

$$\begin{cases} p(0)\lambda = p(1)\mu \\ p(1)(\lambda + \mu) = p(2).2\mu \\ \text{..........} \\ p(n-1)\big((n-1)\mu + \lambda\big) = p(n)n\mu \end{cases}$$

The steady state probability vector P can be written:

$$p(i) = \prod_{k=1}^{i} \left(\frac{(k-1)\mu + \lambda}{k\mu} \right).p(0) \tag{2}$$

With

$$\sum_{i=0}^{n} p(i) = 1 \Leftrightarrow p(0) = \frac{1}{\sum_{i=1}^{n} \prod_{k=1}^{i} \left(\frac{\lambda + (k-1)\mu}{k\mu} \right) + 1} \tag{3}$$

4 Validation and Evaluation

In this section, we consider several scenarios to compare the results obtained with our simulator (described below) and with the Markovian models presented in Section 3. For a complete validation, we have considered a large set of the system parameters by varying values like arrival rates, processing time, acceptance probabilities … These scenarios are not necessarily realistic but they aim to provide as complete as possible validation of our analytical models.

4.1 Java Simulator

In order to study the behavior of a system under various conditions, simulation is usually considered as a realistic solution to provide the expected performance measurements. A lot of network-oriented simulator tools are available but they are not really adapted to model security mechanisms (like encryption, authentication, access control…).For this reason we implemented our own event-driven simulator in Java using the SSJ [15] Java library for stochastic simulation. This library provides methods for generating random variables, computing different measures related to probability distributions, performing goodness-of-fit tests.

The simulator is configured according to the models described in Sections 3.2 and 3.3. The request source is represented by a generator that creates a new Client message structure entering the system every arrival time period (according to a Poisson process). In the centralized configuration the registry processor is represented by random variable generator that generates a uniform processing time values. In the decentralized configuration an exponential time generator is used for the same purpose. All the events of the simulator are collected by a scheduler that memorizes the arrival time of each client, the processing time of each request, the number of rejected requests, the number of successful matchings. All the data acquired with the scheduler are reused to compute the performance parameters described in the next section.

4.2 Rejection Rate

In Figures 7 and 8, we compare the average rejection rate representing the probability for a client request to be rejected from a server before the processing phase. Rejection occurs when all the servers in the decentralized model are busy, i.e.,

$$R_d = P(N) \tag{4}$$

And when N places of the cache of the registry in centralized model are occupied, i.e.,

$$R_c = P(N,0) + P(N,1) \tag{5}$$

After setting the processing rate time μ to 0.2 (5 seconds on average to process a message), we varied the request arrival rate λ from 0.1 to 0.3 (10 seconds to 3.33 seconds of inter-arrival time) with an increment step of 0.01 in order to study different cases of system load: for instance, 0.1 corresponds to a light load while 0.3 corresponds to a heavy load. We also varied the number of servers and the buffer size at the registry (5, 10, and 20 places). The authentication probabilities (q_1 and q_2) are constant and equal to 0.5.

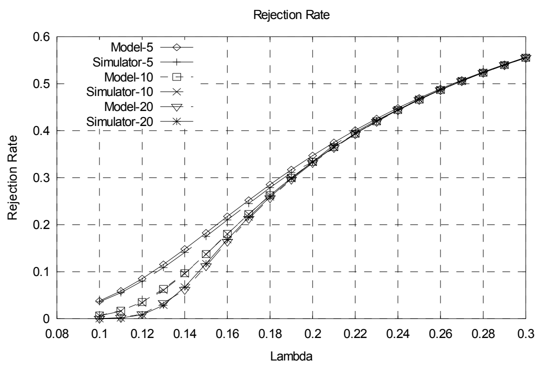

Fig. 6. Rejection rate in a centralized architecture

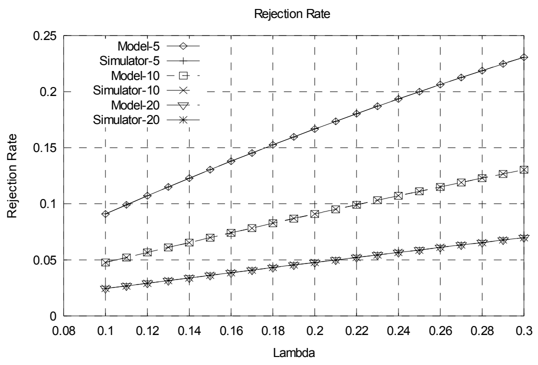

Fig. 7. Rejection rate in a decentralized architecture

We observe from Figure 6 and 7 a perfect matching between the rejection rate measured by the simulator and the one computed by the Markovian model (the margin error is 0.07 %). For a distributed discovery model, the evolution of the rejection rate is linear: this behavior is due to the fact that for every sent request, all the servers become busy at the same time.

This means that a system administrator is able to predict in advance under which conditions his secure discovery system might become overloaded based on the behavior described through Equations 4 and 5, and which configuration is more suitable to ensure a better availability.

A straightforward observation of Figures 6 and 7 could lead to the conclusion that rejection rate in centralized architecture is always higher the one in decentralized model. However this comparison is misleading as in reality a registry should be much more powerful than a server in a decentralized architecture, and action performed by registries are less expressive in terms of computing resources.

4.3 Server and Resource Usage Rate

Using the same scenario as in the previous section, we now provide a comparison between the usage rates of the servers for both architectures, in order to increase the accuracy of validation tests. To obtain a meaningful comparison between the distributed model (S servers where S > 1) and the centralized model (1 server but N slots in the queue), we focused on the resource usage time and not on the server usage time (the proportion of time the resources are busy). With decentralized discovery, this usage time is equal to:

$$U_d = \sum_{k>0}^{S} \frac{k.p(k)}{S} \tag{6}$$

where S is the number of servers in the distributed system. And for a decentralized architecture, with N the maximum capacity of the registry we obtain:

$$U_c = \sum_{k>0}^{N} \frac{k.p(k)}{N} \qquad (7)$$

Figure 8 and 9 illustrate a perfect matching between the resource usage rate measured by the simulator and the one computed by the Markovian model (with less than 0.05% of discrepancy). We notice that the usage rate in the decentralized model is independent from the resource size. This is due to the multicast allocation technique that balances the resource occupation. A system administrator should therefore be able to dimension the resources deployed in the system based on the behavior described in Equations 6 and 7. Buffer size can be optimally adjusted to the traffic in a centralized scenario and an optimal number of replicas of services (provided by the same server) can be deployed to ensure a good quality of service.

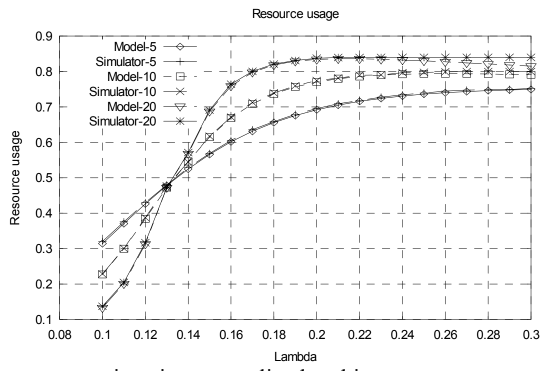

Fig. 8. Resource usage comparison in a centralized architecture

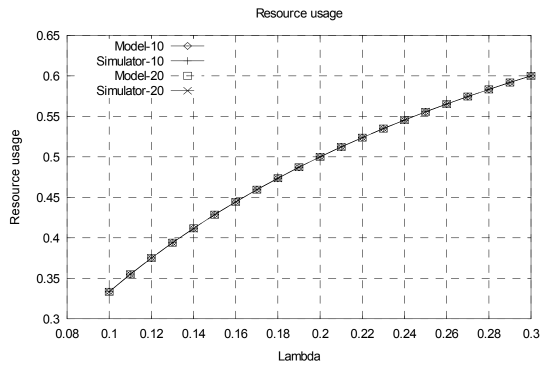

Fig. 9. Resource usage comparison in a decentralized architecture

We notice in the Figure 8 that the resource usage rate lower in 20 slots buffer when $\lambda < 0.13$ and becomes higher when $\lambda < 0.13$. This is however misleading as the number of free slots remains higher in any for a 20 slot buffer in any circumstances. For instance when $\lambda = 0.1$, in a 20 slot buffer, 17.6 places are free while 3.5 places are available in a 5 slot buffer. But if $\lambda = 0.$ 4 places remain free 3 in a 20 slot buffer as compared to 1.5 for a 5 slot buffer.

5 Conclusion

This paper introduced analytical models to assess the impact of security mechanisms used in both centralized and decentralized secure service discovery architectures. This is the first such analytical study of this problem to our knowledge. Results provided by our Markovian models are extremely important to determine whether a centralized or decentralized strategy should be used to deploy services. They make it possible to undertake a systematic study of the robustness, efficiency, resource consumption, fault tolerance, message size, or acceptance rate in a SOA architecture, these performance parameters being easily computed thanks to the analytic approach. We are currently working towards improving the modeling of security aspects of discovery, in particular the efficiency of ABE key combinations. We also plan to study the combination of our application level models with network level model developed either for specific network environments, e.g. ad hoc networks [17], or specific communication schemes, e.g. network level multicast [16].

References

1. Weiser, M. :The Computer of the 21st Century. Scientific American, vol. 265, no. 3, pp. 66–75 (1991)
2. Luo, H., L., Barbeau, M.: Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks. Second Annual Conference on Communication Networks and Services Research pp. 61-68 (2004)
3. Barbeau, M., Kranakis, E.: Modeling and Performance Analysis of Service Discovery Strategies in Ad Hoc Networks. International Conference on Wireless Networks pp. 44-50 (2003)
4. Dabrowski, C., Mills, K.L., Rukhin, A.L.: Performance of Service-Discovery Architectures in Response to Node Failures. Software Engineering Research and Practice 2003: 95-104
5. Trabelsi, S., Roudier, Y., Pazzaglia, J.C.: Discovery: Threats and solutions. 2nd Conference on Security in Network Architectures and Information Systems, Annecy, France (2007)
6. Trabelsi, S., Roudier, Y., Pazzaglia, J.C.: Service discovery: Reviewing Threats and Security Architectures. Research Report RR-07/197 (2007)
7. Trabelsi, S., Gomez, L., Roudier, Y.: Context-Aware Security Policy for the Service Discovery. Symposium on Security in Networks and Distributed Systems (SSNDS) Niagara Falls, Canada (2007)
8. Trabelsi, S., Pazzaglia, J.C, Roudier, Y.: Secure Web service discovery: overcoming challenges of ubiquitous computing. 4th IEEE European Conference on Web Services, Zurich - Switzerland (2006)
9. Garofalakis, O. et al: Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?. 15th ACM Conference on Hypertext and Hypermedia, Santa Cruz, USA (2004)

10. Zhu, F., Mutka, M., and Ni, L.: Splendor: A secure, private, and locationaware service discovery protocol supporting mobile services. First IEEE International Conference on Pervasive Computing and Communications, pp. 235–242, (2003)
11. Zhu, F., Mutka, M., and Ni, L.: Prudent exposure: A private and user centric service discovery protocol. 2nd IEEE International Conference on Pervasive Computing and Communications, Orlando, USA (2004)
12. Czerwinski, S.E., et al: An Architecture for a Secure Service Discovery Service. MobiCom, Seattle, WA (1999)
13. Shamir, A.: Identity-based cryptosystems and signature schemes. Advances in Cryptology, Lecture Notes in Computer Science, Vol. 196, pp. 47-53,Springer-Verlag (1984)
14. Goyal, V. et al: Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. 13th ACM Conference on Computer and Communications Security, Alexandria, USA (2006)
15. L'Ecuyer, P., Meliani, L., Vaucher, J. G. : SSJ: a framework for stochastic simulation in Java. Winter Simulation Conference (2002)
16. Özkasap, Ö. : Performance Study of a Probabilistic Multicast Transport Protocol. Elsevier Science - Performance Evaluation Journal, 57/2, pp. 177-198, (2004)
17. Ari, I., Jethani, N., Rangnekar, A., Natarajan, S. : Performance Analysis and Comparison of Ah-Hoc Routing Protocols. CMSC691T Mobile Computing Project Report, (2000).
18. Hengartner, U. Steenkiste, P.: Exploiting Hierarchical Identity-Based Encryption for Access Control to Pervasive Computing Information. In proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks, (2005).