A PERFORMANCE BASED APPROACH TO SELECTING A SECURE SERVICE DISCOVERY ARCHITECTURE

SLIM TRABELSI, GUILLAUME URVOY KELLER, YVES ROUDIER

Institut Eurecom, 2229 route des Crêtes, BP 193, 06904 Sophia-Antipolis, France

Service discovery, an essential building block of nomadic and ubiquitous computing applications, needs to be secured to be effectively deployed. Centralized and decentralized approaches have been proposed to this end. This paper analyzes the application layer secure matching function using a Markovian performance model in order to analyze various deployment scenarios. This study outlines the determinant parameters that should be evaluated for selecting one out of these two architectures in order to ensure a scalable and efficient service discovery.

1. Introduction

The deployment of ubiquitous computing systems, as notably envisioned by Mark Weiser [1], 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 broadcast or 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 of user devices. 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, while [4] introduces a service discovery performance model that makes it possible to predict discovery service failure and overload in real time. Numerous service discovery standards like WS-Discovery, Jini, UPnP, SLP, or UDDI have also been proposed in recent years, even though their performance has not been assessed analytically to our knowledge. This approach falls short for taking into account the increasing use of discovery in open ubiquitous computing scenarios with numerous new threats [5] [6]. This paper analyzes security mechanisms introduced in [7] and [8] to deal with such issues using performance results obtained out of a Markovian models detailed in [9]. While this model does not take into account low-level network artifacts such as delay or losses, it can assess the impact of introducing security mechanisms at the application level and the resulting processing and traffic overhead incurred. It thus makes it possible to reason about the preferred architecture to ensure an efficient and scalable deployment of secure service discovery depending on specific scenarios.

2. Secure Service Discovery Models

Service discovery involves a service requester (client) and a service provider (server), the latter providing one or multiple services that can be accessed by the clients. In traditional discovery approaches, security is usually limited to recommendations about classical message authentication and integrity protection, thereby implicitly restricting discovery to known services. Ubiquitous computing has prompted standard-independent studies aimed at securing service discovery using an infrastructure for establishing the trustworthiness of clients and services. For instance, Zhu et al. [10] assume that participants to the discovery protocol are located behind a trusted proxy that sets up trust relationships through key exchanges with other proxies. [12] instead suggests the use of a central entity that combines the roles of a Certificate Authority and of a registry, and that helps clients and servers to set up a trust relationship and to establish secure channels with each another. These solutions are not adapted to decentralized architectures and address server rather than client security, contrary to [7] and [8] on which we focus. These security solutions being destined to a wireless environment, they aim at protecting discovery systems against illegal access to discovery messages which the adversary can get. Similarly to these solutions, we consider that brute force DoS and signal jamming are out of scope of our adversarial model. The following sections introduce queuing models of these two approaches.

2.1. *Centralized Discovery Model*

Centralized discovery approaches rely on a registry which plays the role of a trusted third party in charge of enforcing security policies provided by clients and services. Clients and services send their discovery policies to the registry that will be in charge of judging whether a discovery matching is secure.

The queuing model depicted in Figure 1 represents the processing phase of a secure client service request at the registry for a centralized configuration, considering a sole thread is in charge of all processing steps.

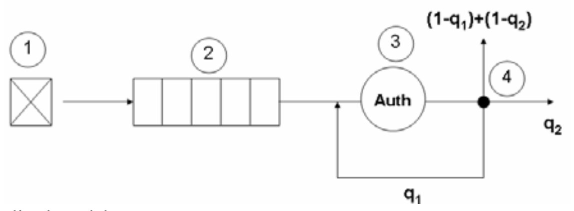

Figure 1: Centralized Model

The discovery process consists of four 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/μ.
4. Probabilistic decisions (acceptance/rejection): q1 is the probability that a service matches with a client request and be compliant with its policy, q2 the probability that a client be compliant with this service policy.

2.2. *Decentralized Discovery Model*

The security solution proposed in [8] for a decentralized configuration relies on a particular usage of the Attribute Based Encryption mechanism [11]. This mechanism is used by clients to protect their requests by encrypting them according to a particular policy of disclosure of client/service profile attributes.

The queuing model depicted in Figure 2 represents the decentralized discovery scheme in which some computing time is now allocated to encryption and decryption. Requests are routed using multicast, which adds complexity to event handling. In such a decentralized architecture, nodes usually have limited capacities as compared to a registry. For this reason, we considered in our model that a server does not buffer new requests when it is busy. The execution proceeds as follows:

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/µ1.
3. Service authentication: q1 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: q2 is the success decryption probability of a client.

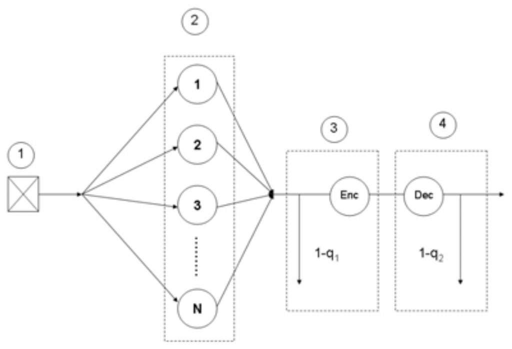

Figure 2: Decentralized Model

3. Matching Probabilities

The probabilities q1 and q2 described above represent the probability for a client or a service to obtain a successful matching (including authentication, access control, decryption) with security policies protecting the access to resource profiles. This probability depends on the number of elements of the systems and on the volume of vocabulary known by each element. The vocabulary volume is the amount of data knowledge related to a certain domain. For example, a subset of the medical vocabulary (scanner, radiology, dermatology, cardiology etc.) can be related to the services deployed inside a hospital building or the roles of users (surgeon, patient, etc.). In analogy to such concepts, we define a vocabulary as the global set of possible identities or roles in a system. The subset of this vocabulary is represented by the group of identities and roles existing in the system.

The probability to match an element (client or au server), represented by a group of attributes *x* in a system, is defined by the probability *P* that these attributes belong to the subset vocabulary *C* part of the general vocabulary *V*:

$$P(x) = \frac{\binom{x}{C}}{\binom{x}{V}}; size(x) \geq 1 \tag{1}$$

4. Performance Analysis

Markov chains corresponding to the two queuing models were detailed and validated with a continuous time simulator in [9]. This mathematical model makes it possible to study the performance parameters of service discovery and to determine whether a centralized or a decentralized strategy should be adopted.

The performance study detailed below answers the following determinant questions for selecting one of the two secure solutions to service discovery: in which conditions is the request rejection rate better or worse? Which model is able to serve the largest number of clients? What is the fastest approach? What is the impact of a variable number of servers in the system? What is the impact of the matching probabilities on the performance?

4.1. *System Setup*

Table1. Values of the input variables used in the tests.

	Centr1	Decentr1	Centra2	Decentr2	Centr3	Decentr3	Centr4	Decentr4
λ	0.5 → 40	0.5 → 40	0.5 → 40	0.5 → 40	0.5 → 40	0.5 → 40	0.5 → 40	0.5 → 40
μ1	14.28	2.5	14.28	2.5	14.28	2.5	14.28	2.5
μ2	-	20	-	20	-	20	-	20
V	10	10	10	10	20	20	20	20
C	5	5	8	8	16	16	5	5
q1	0.5	0.5	0.8	0.8	0.8	0.8	0.25	0.25
Buffer Size	2-5-10	-	2-5-10	-	2-5-10	-	2-5-10	-

This relies on measurements obtained on real systems as previously published. Four test scenarios are described in Table 1. The Attribute Based Encryption/Decryption duration are excerpted from [11] according to set values for *x={1,2,3}*. We experimented ourselves with XACML policy reasoning and enforcement as detailed in [7]. The arrival rate of client requests, number of services, vocabulary size, and matching probability q1 are variable in our tests.

4.2. *Rejection Rate*

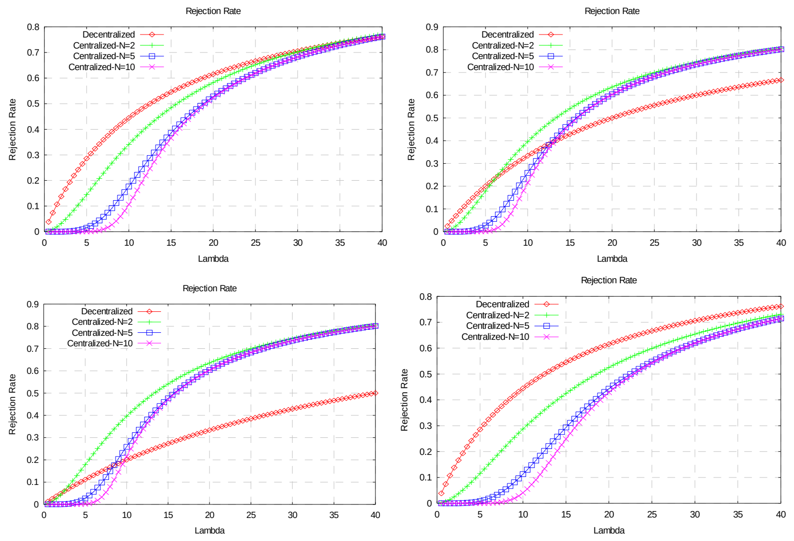

Figure 3: Rejection rate curves for the four test scenarios

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.,

$$R_d = P(N) \tag{2}$$

and when N slots of the registry cache are occupied in the centralized model,

$$R_c = P(N,0) + P(N,1) \tag{3}$$

Figure 3 shows that the rejection rate due to a lack of resources is invariant for the centralized model in case of a fixed buffer size; in contrast, as the buffer size increases, the rejection rate reduces. Regarding the decentralized model, Figures 3-a, 3-b, and 3-c show that the rejection rate is strongly dependent on the number of servers deployed. As the number of servers increases, the rejection rate decreases. Figure 3-d shows that probability q1 does not affect the rejection rate for the decentralized model but clearly impacts the rejection rate for the centralized model. We can conclude that the decentralized system is more suitable in a system with a large number of servers.

4.3. *Average Number of Users in the System*

The average number of users Q present in the system is the temporal mean N(t) of the number of users observed in the system over period [0,T].

$$Q(T) = \frac{1}{T}\sum_{n} n \cdot T(n,T) \tag{4}$$

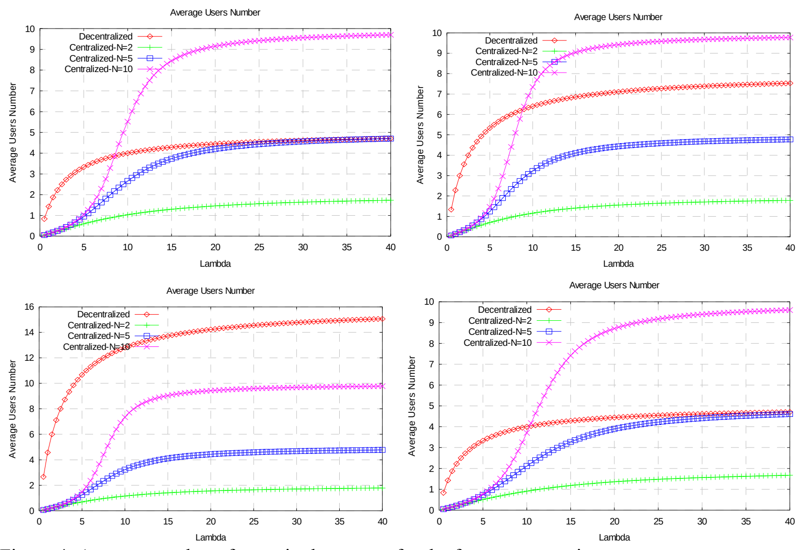

Figure 4: Average number of users in the system for the four test scenarios

In the centralized Markov chain model, Equation (4) can be written as:

$$Q_c = \sum_{n=1}^{Nbuffer} n \cdot \left(p(n,0) + p(n,1)\right) \tag{5}$$

In the decentralized Model this equation becomes:

$$Q_d = \sum_{n=1}^{Nservers} n \cdot p(n) \tag{6}$$

Figure 4 illustrates the capacity to serve requests. In the centralized system, it is proportional to the buffer size and to the matching probability (the bigger the matching rate, the longer requests stay in the system). In the decentralized system, the number of users served is proportional to the number of servers and the matching probability does not affect the number of users in the system.

4.4. *Service Time Duration of a Request in the System*

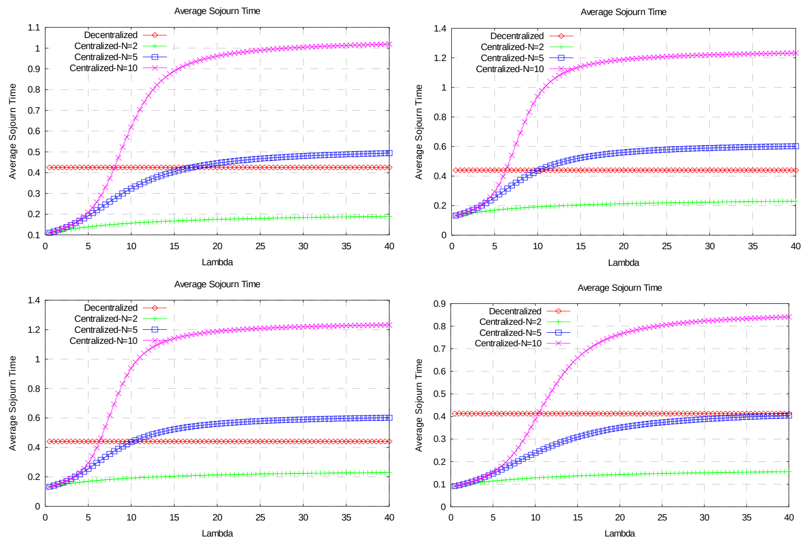

Figure 5: Service time duration of a request in the system for the four tests scenarios

The lifetime R of a request in the system is the mean time spent by the requests accepted and processed during time period [0,T]. This rate can be computed using Little's Law that states that "*the long-term average number of customers in a stable system Q is equal to the long-term average arrival rate X multiplied by the long-term average time a customer spends in the system R*" [13]:

$$R = \frac{Q}{X} \tag{6}$$

where X is the product of the probability of at least a user being served and of the Processing Rate. For the centralized model, the service time rate R is:

$$R_c = \frac{Q_c}{\sum_{n=1}^{Nbufeer} \left[p(n,0).(1-q1)+p(n,1)\right].\mu} \tag{7}$$

For the decentralized model, the service time rate R is:

$$R_d = \frac{Q_d}{\sum_{n=1}^{Nservers} \mu . p(n)} \qquad (8)$$

As depicted in Figure 5, the service time duration is strongly related to the matching probability in the centralized model. As this probability increases, the longer it takes to process the request. With a small buffer size, the system delivers a quicker response although with a high rejection rate. In contrast, the decentralized model exhibits a constant service time in every situation: this is due to the facts that tasks are distributed between servers.

4.5. *Summary*

Table 2 summarizes the tradeoffs from the performance study presented above: it lists the effects of a change in the infrastructure on the performance parameters that influence the quality of service of the service discovery functionality. One important result of this study is that no single approach can satisfy the requirements of all deployment scenarios.

Table 2: Performance summary (C : Centralized, D: Decentralized, + : increase, - : decrease, = : unchanged value)

Performance parameters:	**Reject**		**Number of users**		**Service Time**	
	C	D	C	D	C	D
Increased Buffer Size	-	=	+	=	+	=
Increased Matching Probability	=	=	-	=	+	=
Increased Number of Servers	=	-	=	+	=	=

5. Conclusion

This paper evaluated the performance of the secure service discovery function, a basic building block in nomadic and ubiquitous computing, as deployed with two approaches: a centralized one, requiring the setup of a trusted third party, the registry; a decentralized one, relying only on a decentralized matching by services themselves. To our knowledge, this is the very first evaluation of the performance of service discovery that takes into account the cost of secure matching and that compares these two different techniques using a Markovian model. This study provides determinant elements for selecting and fine-tuning either of these approaches in order to ensure the scalability of service discovery according to the application scenario deployment parameters. We plan to evaluate the impact of blind DoS attacks on the system by considering a malicious traffic class representing fake encrypted messages.

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. Trabelsi, S., Urvoy-Keller, G., Roudier, Y.: A Markovian performance model for secure service discovery systems. Rapport de recherche RR-08-214. (2008)
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. Hengartner, U. and Steenkiste, P.: Exploiting Hierarchical Identity-Based Encryption for Access Control to Pervasive Computing Information. Proc. of First IEEE/CreateNet International Conference on Security and Privacy for Emerging Areas in Communication Networks (2005), Athens, Greece, pp. 384-393.
12. Czerwinski, S.E., et al: An Architecture for a Secure Service Discovery Service. MobiCom, Seattle, WA (1999)
13. Little, J. D. C.: A Proof of the Queueing Formula $L = \lambda W$" Operations Research, 9, (1961), pp. 383-387