

# Adopting Trust Negotiations: To Negotiate or Not To Negotiate?

Arne Koesling  
L3S Research Center  
Hannover, Germany  
koesling@L3S.de

Daniel Olmedilla  
L3S Research Center  
Hannover, Germany  
olmedilla@L3S.de

## Abstract

*Open distributed environments require that agents who are not known to each other must be able to interact. A new authorization scheme called trust negotiation has emerged allowing two strangers to iteratively and bilaterally establish trust. This scheme has been applied to different environments such as the Web, P2P networks or Grid environments. However, it is not yet clear what impact, implies its integration into running systems what leads to a lack of adoption. This papers investigates the overload produced by the integration of trust negotiation techniques and shows how negotiations might, under some assumptions, imply only a small increase on the network use in comparison with the benefits it provides.*

## 1 Introduction

Open distributed environments such as the Web, P2P networks or Grid environments require that two entities that have not yet had any transaction in common are able to interact. Traditional access control relies on entities identities and therefore it is not suitable for transaction among strangers. Instead, entities properties (e.g., project membership, being a student or holding German citizenship) can be used for trust establishment. Trust negotiation [18] is a bilateral process in which credentials stating each party's properties are exchanged iteratively according to specified policies, allowing for the trust between such two entities to increase after each iteration. Trust negotiation has been successfully applied to environments such as the Web [19, 6], P2P networks [16], Web Services [13, 14] and Grid [15, 5], enhancing the authorization schemes they provide. However, although the important benefits of this approach have been acknowledged it has not yet been widely adopted. One of the main reasons for that situation is that it has not been thoroughly investigated the consequences of its integration

in the above mentioned environments. Therefore, it is difficult to predict with accuracy what the impact of its use implies.

Different policy languages for trust negotiations have been created [2, 10, 6, 1, 3], most of them based on data-log semantics and reasoners. In comparison with traditional authorization mechanisms, reasoning for policy compliance requires extra resources including memory and time. However, recent results on policy reasoning performance [8] and the fact that some of the policy languages allow for PTIME reasoners (e.g., see PROTUNE [3]) shows that policy reasoning will not be the bottleneck when performing negotiations. Instead, the lack of adoption and therefore due the lack of real world policies supporting negotiations makes that time and network use in unbounded negotiations have not yet been investigated.

This paper addresses this gap and investigates how the adoption of negotiations affect the overall network use and what the overhead is. Since no actual data exists about trust negotiations in real life we simulated the performance of negotiations under different assumptions and parameters extracted from current distributed environments such as the Web. Our simulations demonstrate that even though the adoption of negotiations increase the load of the network, using simple query-answering optimizations dramatically reduces it.

The rest of the paper is organized as follows: Section 2 introduces trust negotiations and the problems their adoption may imply. The simulation algorithm used in this paper is described in detail in Section 3 and the results of the simulations are described and discussed in Section 4. Finally, Section 5 concludes the paper and outlines some future work.

## 2 Trust Negotiation and Network Load

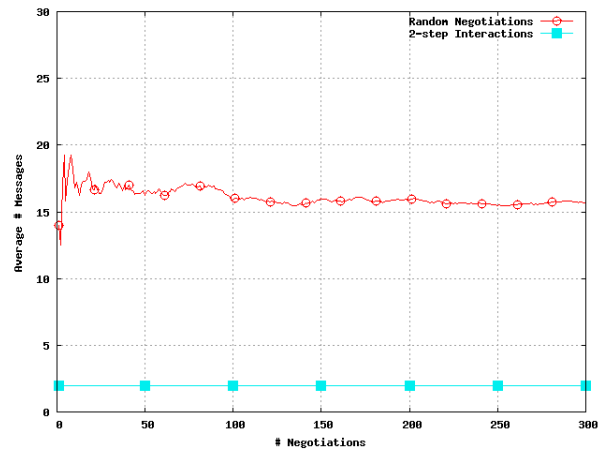
Trust negotiation is a mechanisms that allows two strangers to establish trust by bilaterally and iteratively ex-

changing policies and credentials. This process addresses the requirements of open distributed environments, for which identity-based mechanisms do not scale. However, its adoption implies an impact on the overall network usage and length of negotiations. In a conventional case with pre-registered users and a simple authentication mechanism based on user and password, authorization transactions typically consist of two messages: a request from the client and a response from the server with either access granted or access denied. However, allowing negotiations imply that the length of a negotiation, and therefore the number of messages exchanged, is not known in advance. Suppose Alice wants to buy a book from an on-line store. The following describes the messages exchanged:

1. Alice sends the request to access the book to the store.
2. The store requests her credit card to complete the transaction.
3. Alice is willing to release her credit card only to companies member of the Better Business Bureau.
4. The store releases its membership credential.
5. Alice releases her credit card.
6. The store grants access to the book.

In this example, six messages are exchanged between Alice and the store. However, they could potentially have been many more if both parties' policies were more complex. Figure 1 shows a comparison among a traditional 2-step interactions and a negotiation-based case in which the length of negotiations follow a purely random distribution with an upper bound of 30 messages. Of course, since there is no real data about what distribution negotiations follow, we may not be sure whether this comparison is accurate but it definitely represents how the latter potentially produces a big impact on the environment in which it is integrated.

However, trust negotiations were especially designed for trust establishment among strangers. Therefore, after a first transaction in common, two entities are not stranger to each other anymore and part of, or possibly the whole negotiation may be reused in future requests. For example, a server may remember for a certain period of time those credentials that were disclosed by the client, therefore not needing to request them again in future requests made by the same client (e.g., as Amazon remembers the payment information given by a client in order to be able to offer its "1-click ordering"). Similarly, a client negotiating with a server may remember the policies (and possibly credentials) of the server that must be satisfied for the kind of transaction requested (e.g., Amazon policies for buying a book are the same independently of the book being bought). This way, next time



**Figure 1. Two step interactions vs. random negotiations**

the client makes a request to the server, it may proactively attach the required credentials with the request. These two kinds of cache may allow for strong reduction in the number of messages exchanged in a distributed environment where negotiations are integrated.

The following sections present a simulation model that allows inputting different parameters (e.g., negotiation distributions) in order to represent different negotiation situations and estimates how many messages are interchanged for each one of them.

### 3 Simulation Algorithm

Simulating the behavior of the use of negotiations in a distributed environment is a difficult task since no actual data exists in order to accurately represent all the elements required. For that reason, we specified our model with a large set of parameters that permit us to perform simulations under a different set of assumptions.

Our simulation model observes the number of messages exchanged during a large number of negotiations in a distributed environment. For that, a set of servers and clients are specified. In order to simplify the model, it is assumed that only clients initiate requests and they send them only to servers. Therefore, negotiations in which third entities are involved (e.g., via delegation mechanisms) are not considered in this model but left for future work. Furthermore, without loss of generality, each server provides exactly one service<sup>1</sup>.

<sup>1</sup>A server providing multiple services may be easily modelled as several servers with one service each. As a matter of fact, grouping services into servers would even improve the results of this paper since cache mech-

- $S \equiv$  set of providers/servers, and  $C \equiv$  set of consumers/clients
  - $D \equiv$  distribution of messages per negotiation: random, normal or powerlaw
  - $ksp \equiv$  probability of a client sending a request to a known server (re-visitation rate)
  - $known\_servers(c) \equiv$  set of providers known by a client  $c$
  - $cache@client(c) \equiv$  set of servers cached by a client  $c$
  - $cache@server(s) \equiv$  set of clients cached by a server  $p$
- (1) For each negotiation
  - (2) select randomly  $c \in C$
  - (3) select  $s \in \begin{cases} known\_servers(c) & \text{with probability } ksp \\ S \setminus known\_servers(c) & \text{with probability } 1 - ksp \end{cases}$
  - (4) if  $s \in cache@client(c)$  then
  - (5)      $\#messages = 2$
  - (6) else
  - (7)     if  $c \in cache@server(s)$  then
  - (8)          $\#messages = 2$
  - (9)     else
  - (10)          $\#messages = generateNumberMessages(D)$
  - (11)          $update(cache@server(s), c)$
  - (12)          $update(cache@client(c), s)$

**Figure 2. Simulation Algorithm**

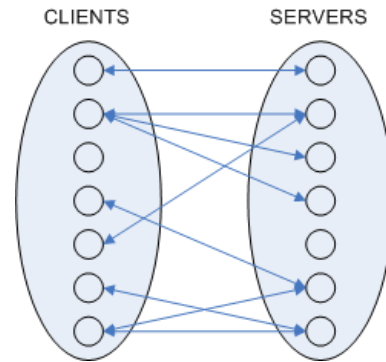
One of the main assumptions in our simulation algorithm (depicted in Figure 2) is that negotiations are required only for first interactions between a given server and client, that is, when they are strangers. During such first interaction credentials and policies may be cached by both clients and servers, using the “cache@client” and “cache@server” respectively. The “cache@client” (C@C) stores the policies from a given server allowing the client to proactively send the credentials together with the request, therefore transforming subsequent requests from a full negotiation in a 2-messages interaction. The “cache@server” (C@S) allows a server to store the credentials provided by a client so they are not requested again for new requests. This way, the policy of the server is satisfied with such credentials and again<sup>2</sup>, a full negotiation is transformed in a 2-messages interaction<sup>3</sup>. Our simulation receives as input the size of both caches, where 0 represents no cache. In order to deal with limited cache sizes, a simple least recently used (LRU) strategy has been adopted. If a cache is full, it overwrites the entry having the oldest timestamp (set at insertion time and updated when used) with the new entry to add.

Techniques like caching can only show reasonable ef-

anisms would be shared among services.

<sup>2</sup>Note that we do not deal here with policy evolution but assume that the policy of the server does not change as long as it is in the cache.

<sup>3</sup>Similarly to caching and reusing stored credentials and policies, the server may also hand out a trust ticket comparable to [9] to grant access to the client for a later re-visitation.



**Figure 3. Clients-servers interactions**

fect, if clients often return to a specific server and service. This occurs typically on the Web where users often return to web pages [17, 4] and therefore we include this re-visitation rate (a probability of a client sending a request to a known server) as an input to the algorithm. Therefore, a list of “visited servers” is kept at the client (similar to a bookmark list), in order to select among those “known” servers and the ones that are not.

The algorithm followed for the simulations is depicted in Figure 2. Given all the parameters (all the values given to inputted parameters during the simulations are extracted from actual data on the Web and will be discussed in the next section) the following process is performed for each

C@C Size	Re-visitation rate	C@S Size		
		11	55	110
0	49%	9.09	8.49	7.81
	61%	9.02	8.22	7.28
	81%	8.88	6.93	6.04
	92%	8.78	6.93	4.99
1	49%	8.93	8.45	7.79
	61%	8.79	8.13	7.27
	81%	8.46	7.33	5.94
	92%	8.23	6.71	5.01
5	49%	8.03	7.90	7.49
	61%	7.55	7.37	6.81
	81%	6.43	6.10	5.24
	92%	5.62	5.12	4.04
10	49%	7.10	7.08	6.90
	61%	6.35	6.34	6.09
	81%	4.67	4.64	4.31
	92%	3.42	3.38	3.04
15	49%	6.36	6.35	6.31
	61%	5.51	5.51	5.45
	81%	3.76	3.76	3.70
	92%	2.69	2.69	2.66

**Table 1. Results for combinations of varying parameters using a powerlaw distribution for negotiations.**

negotiation of the simulation. A client is selected randomly from the set of existing clients. A server is selected from either the set of known servers or the unknown servers according to the re-visitation probability, and using a zipf distribution in order to reflect different degrees of popularity for our servers (like on the Web, some servers typically have many more visitors than others)<sup>4</sup>. If the server is found in the “cache@client” or in the “cache@server”, then the negotiation consists of only two negotiations exchange. Otherwise, a new negotiation is “performed” assuming that all negotiations performed follow a given distribution (e.g., random, normal or powerlaw). After the negotiation is performed, the caches are updated accordingly. We simulate a large number of negotiations and calculate the average number of messages exchanged in order to measure the network load.

<sup>4</sup>The whole set of servers are created at start-up covered by the zipf distribution. Therefore, there are no two separate zipf functions for servers on the known server list and for the remaining list.

## 4 Simulation Results

In order to represent a real distributed environment we used Web statistics on number of clients and web servers. [12] measured the amount of web servers in Internet to 96,854,877 at the beginning of October 2006. According to [7] the estimated number of people having access to the Internet was 1,086,250,903 worldwide for September 2006. These figures let us infer that there exist a ratio of 1:11 between servers and clients. Keeping such a ratio, all our simulations used 1,100 clients and 100 servers.

The size of caches at both client and server are important resources. Servers typically have more resources than client computers and therefore we assume that their caches are larger. In the simulations we run, we used caches of size 0, 1, 5, 10 and 15 (that is, no cache, 1%, 5%, 10% and 15% of available servers respectively) for the client and caches of size 11, 55 and 110 (1%, 5% and 10% of total amount of clients respectively) for all the servers. We also run simulations where not all the servers share the same cache size but instead they have sizes among a range following the same zipf distribution used for its selection. Those cases will be clear from the context by using ranges such as 11-22, 55-110 and 110-220 for the cache size.

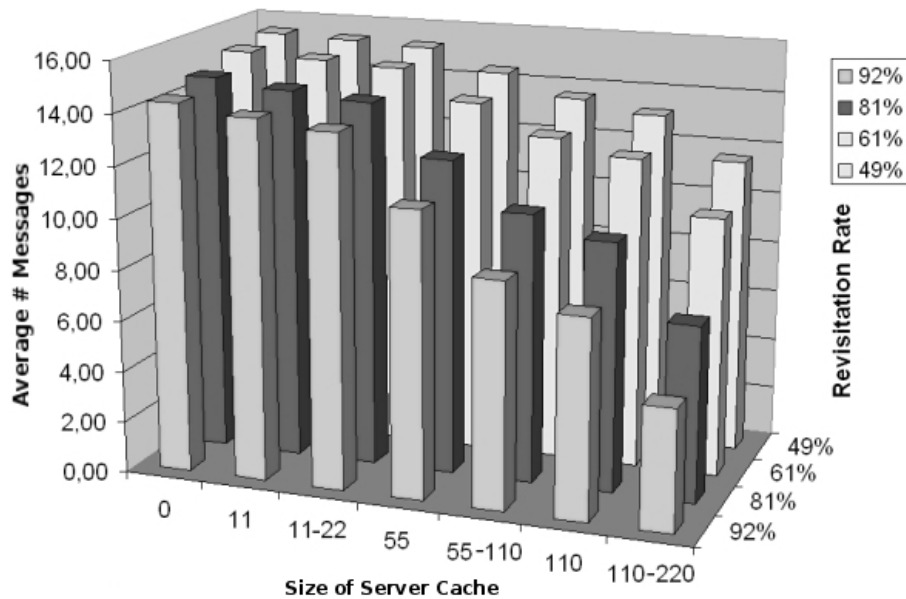
Re-visitation to known servers (or re-invocation of known services) is a hard parameter to estimate. There do not exist studies to hint what those values may be in real world. However, studies on web page re-visitation rate [17, 4] show different experiments in which such rates varied between 49% and 92%. Although we would be more interested in web server revisitation rate (a server offering one service has a single policy protecting the server, independently of how many pages the server provides) we use those values as equivalent to our server/service revisitation rate<sup>5</sup>. We use 49%, 61%, 81% and 92% as input values for our simulations.

Finally, if there is no hit in any cache between a selected client and a server, a negotiation must be performed. Since there does not exist a large number of negotiation policies allowing for its proper analysis, we assumed that such negotiations may follow different distributions. In our negotiations we used random, normal and powerlaw as plausible distributions.

We performed simulations with 1,000,000 negotiations each for all possible combinations of the parameters introduced above. Some results are depicted in Figure 4 and Table 1.

Figure 4 shows how increasing the size of the server

<sup>5</sup>While web page revisitation would treat two different books at Amazon as two different pages, still the policy protecting its access would be the same for the whole Amazon site. Therefore, we would expect the server revisitation rate to be much higher than the web page revisitation rate.



**Figure 4. Combined effect of server cache and re-visitation rate for a fixed client cache of size 1 and powerlaw negotiation distribution.**

cache and the re-visitation rate improves the overall exchange of messages during the negotiations. Interesting is that varying the cache size for servers according to their popularity show similar results to a uniform cache size for all servers (e.g., comparing 55-110 and 110). The reason is that most of the overall hits produced by caches at servers are provided by the most popular servers since they receive most of the requests. In addition, as expected, a higher re-visitation rate implies a reduction on the total number of messages exchanged.

Table 1 reflects how increasing the cache size at both server and client sizes help to improve the overall load, specially if higher re-visitation rates hold. However, interestingly, it can be observed that using and increasing a “cache@client” strongly improves the average number of exchanged messages, even overriding the “cache@server”. Furthermore comparing all results obtained, we observed that by using a cache at the clients, the average number of messages drop down to 2.66 for the highest re-visitation rate.

As a summary, varying different parameters in the simulation helped us to better understand the impact of negotiation under given assumptions. The following facts summarize our observations:

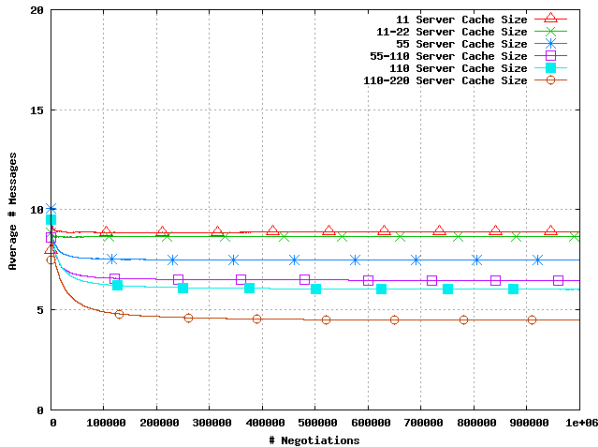
**Negotiation distribution.** Our simulations showed that although there were not really big differences between

random, normal and powerlaw distributions, still the later gave better results. The reason for such a small differences among them is that the weight of the distributions is strongly diminished due to the caching effect and the high re-visitation rates (at least 49%). This is interesting because it may allow to elaborate on general strategies for negotiations without having to specify them differently for different kind of policies or negotiation distributions.

**Cache at server.** As expected, increasing the size of the cache stored at servers reduces the overall number of messages exchanged (see Figure 5).

**Cache at client.** Similarly as with the cache at server side, increasing the size of cache at client side reduces the overall number of messages exchanged. However, interestingly, when using this cache at client side, it seems to override the cache at the server (see Figures 6(a) and 6(b)). This is interesting because it proves to be much more effective and relatively small caches in each client browser may dramatically reduce the impact of adopting trust negotiations in a distributed environment.

**Re-visitation rate.** Finally, as expected, the re-visitation rate plays a crucial role in the overall simulations resulting on decreasing the overall number of messages



**Figure 5. Simulations with varying server cache size, no cache at the client, 81% revisitation rate and powerlaw negotiation distribution.**

exchanged when it increases. However, a relatively low re-visitation rate such as 61% (as explained in Section 3, a server revisitation rate is expected to be much higher than the web page revisitation values that we use in this paper) already dramatically reduces the overall impact of negotiations to a third. For higher rates, the reduction is of a fifth for 81% (an average of only 3 messages per negotiation) or an eight for 92% (almost only 2.5 messages average per negotiation).

## 5 Conclusion and further work

Although it is known and acknowledged that trust negotiations bring in many advantages to existing distributed environments, its adoption has not yet taken place. One of the main reasons for that is that the impact of such an adoption has not been extensively investigated. Although some recent results showed that the impact produced by the integration of policy compliance reasoners in the authorization process may be acceptable, there has not been any study showing what the impact in the overall network may imply. This is particularly interesting in environments such as the Grid in which network bandwidth is a precious resource and delays produced by sending messages and waiting for answers may be too costly.

This paper presents a set of simulations analyzing different re-visitation rates, distributions and types and size of caches. These simulations show how the impact of integrating negotiations may not be as high as it might have been

thought at the beginning. In fact, for environments in which high re-visitation rates hold, the integration of trust negotiations produces a really small overhead.

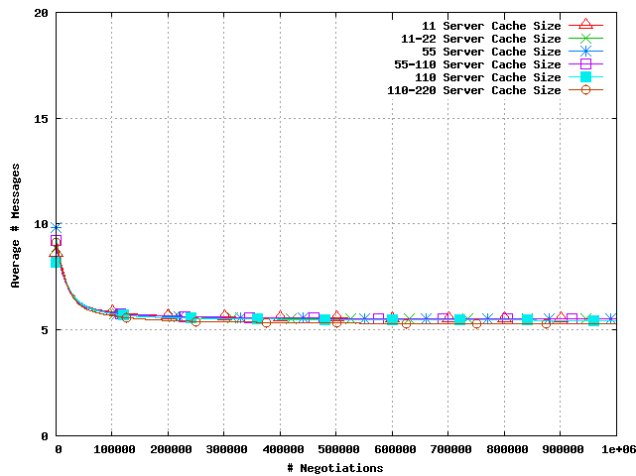
Although we performed an extensively large number of simulations, there are still quite some assumptions that need to be further investigated. For example, our ratio among clients and servers resembles that of web clients and servers in 2006. Different values may be required for different environments such as P2P networks in which the roles of client and server are tightly joined. In addition, negotiations involving more than two parties are not considered in this paper and may be required to be further investigated as well as more advance techniques (e.g., for caching) may be explored. Furthermore, privacy may be an issue when storing client or server policies and credentials. Usage control policies (also known as sticky policies [11]) for different credential types and policies might be needed in such a case. Finally, this paper studies the impact of integrating negotiations from a network perspective. However, trust negotiations systems are more exposed to denial of service attacks than simpler authorization mechanisms. Therefore, techniques for adapting the servers (e.g., stop negotiations if a DoS attack is detected) or off-loading to trusted servers must be investigated.

## Acknowledgments

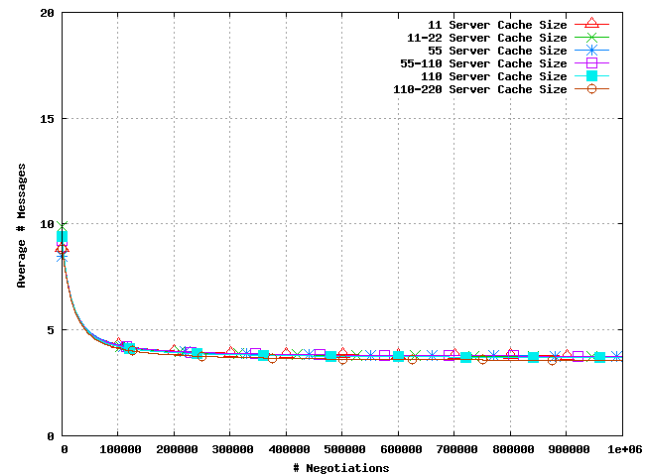
This work was partially funded by the European Commission and by the Swiss State Secretariat for Education and Research within the 6th Framework Programme project REVERSE number 506779 (cf. <http://reverse.net>).

## References

- [1] M. Becker and P. Sewell. Cassandra: Distributed access control policies with tunable expressiveness. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, pages 159–168. IEEE Computer Society, June 2004.
- [2] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 134–143. ACM Press.
- [3] P. A. Bonatti and D. Olmedilla. Driving and Monitoring Provisional Trust Negotiation with Metapolicies. In *6th IEEE POLICY 2005*, pages 14–23, Stockholm, Sweden, jun 2005.
- [4] A. Cockburn and B. McKenzie. What do web users do? an empirical analysis of web use. *International Journal of Human-Computer Studies*, 54:903–922, 2002.
- [5] I. Constandache, D. Olmedilla, and F. Siebenlist. Policy-driven negotiation for authorization in the semantic grid. In *IEEE International Workshop on Policies for Distributed*



(a) 61% revisitation rate



(b) 81% revisitation rate

**Figure 6. Simulations with varying server cache size, fixed client cache size, and powerlaw negotiation distribution.**

*Systems and Networks (POLICY 2007)*, Bologna, Italy, June 2007. IEEE Computer Society.

- [6] R. Gavrioloie, W. Nejdl, D. Olmedilla, K. E. Seamons, and M. Winslett. No Registration Needed: How to Use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web. In *1st European Semantic Web Symposium (ESWS 2004)*, volume 3053 of *LNCS*, pages 342–356, Heraklion, Crete, Greece, may 2004. Springer.
- [7] Internet world stats homepage. <http://www.internetworldstats.com/stats.htm>, 2006.
- [8] A. J. Lee, M. Winslett, J. Basney, and V. Welch. Traust: a trust negotiation-based authorization service for open systems. In *11th ACM Symposium on Access Control Models and Technologies*, pages 39–48, Lake Tahoe, California, USA, June 2006. ACM.
- [9] J. Li, J. Huai, J. Xu, Y. Zhu, and W. Xue. TOWER: Practical Trust Negotiation Framework for Grids. *Will be published in: Proceedings of ESScience Conference in Amsterdam*, 2006.
- [10] N. Li and J. C. Mitchell. Rt: A role-based trust-management framework. In *Third DARPA Information Survivability Conference and Exposition (DISCEX III)*.
- [11] M. Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services, 2003.
- [12] Netcraft october 2006 web server survey. [http://news.netcraft.com/archives/web\\_server\\_survey.html](http://news.netcraft.com/archives/web_server_survey.html), 2006.
- [13] D. Olmedilla, R. Lara, A. Polleres, and H. Lausen. Trust Negotiation for Semantic Web Services. In *1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, volume 3387 of *Lecture Notes in Computer Science*, pages 81–95, San Diego, CA, USA, jul 2004. Springer.
- [14] L. Olson, M. Winslett, G. Tonti, N. Seeley, A. Uszok, and J. M. Bradshaw. Trust negotiation as an authorization service for web services. In *Proceedings of the 22nd International Conference on Data Engineering Workshops*, page 21, Atlanta, GA, USA, Apr 2006. IEEE Computer Society.
- [15] T. Ryutov, L. Zhou, C. Neuman, N. Foukia, T. Leithead, and K. E. Seamons. Adaptive trust negotiation and access control for grids. In *6th IEEE/ACM International Workshop on Grid Computing*, 2005.
- [16] S. Staab, B. K. Bhargava, L. Lilien, A. Rosenthal, M. Winslett, M. Sloman, T. S. Dillon, E. Chang, F. K. Hussain, W. Nejdl, D. Olmedilla, and V. Kashyap. The pudding of trust. *IEEE Intelligent Systems*, 19(5):74–88, 2004.
- [17] L. Tauscher and S. Greenberg. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human Computer Studies*, 47(1):97–138, 2001.
- [18] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. DARPA Information Survivability Conference and Exposition, IEEE Press, Jan 2000.
- [19] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. Negotiating trust on the web. *IEEE Internet Computing*, 6(6):30–37, 2002.