BambooTrust: Practical scalable trust management for global public computing

Evangelos Kotsovinos
Deutsche Telekom Laboratories
Ernst-Reuter-Platz 7
10587 Berlin, Germany
evangelos.kotsovinos@telekom.de

Aled Williams
Cambridge University Computer Laboratory
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
aledwilliams@cantab.net

ABSTRACT

Global public computing platforms, such as PlanetLab, grid computing systems, and XenoServers, require facilities for managing trust to allow their participants to interact effectively in an open and untrusted environment. In this paper, we describe BambooTrust, a practical, high-performance distributed trust management system for global public computing platforms. We present our peer-to-peer architecture, based on the XenoTrust model and the Bamboo distributed hash table. We describe the initial BambooTrust implementation and deployment, and demonstrate that the system performs and scales more than adequately well by means of experimental evaluation.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; H.3.5 [**Online Information Services**]: Data sharing; H.3.3 [**Information Search and Retrieval**]: Clustering

Keywords

Reputation systems, incentives, honesty detection, trust management, peer-to-peer systems

1. INTRODUCTION

The XenoServer project [9] is building a global network of servers, which undertake the safe execution of potentially untrusted computation on behalf of clients. Servers perform accounting for resources consumed and ultimately charge clients accordingly. Utility computing initiatives[1], grid computing projects[2], and PlanetLab [13] provide — or plan to provide — similar facility. We collectively term distributed platforms that allow the deployment of third-party computation — potentially in exchange for money or services — *global public computing* systems.

Such systems are facing a number of threats stemming from their *open* and *public* nature. For instance, servers may be unreliable, overcharging clients or not running their programs faithfully, or even trying to extract clients' secrets. Also, clients may try to avoid paying their bills, or to run programs with anti-social or illegal effects.

We discussed the threat model to public computing platforms in [4], and described why the trust management requirements they impose are distinct from the ones of *on-line marketplaces*[3] and *peer-to-peer file-sharing applications* — mainly scaling well, supporting very frequent feedback submissions, and facilitating multidimensional trust. In light of those requirements, we proposed XenoTrust [5], a high-level paradigm for the design of decentralised trust management systems for global public computing, and Pinocchio [6] for providing participation incentives.

In this paper, we will analyse the architecture and implementation of BambooTrust, a practical, scalable distributed trust management system based on the XenoTrust model and the Bamboo distributed hash table[4]. BambooTrust is designed and implemented to facilitate:

- **Performance.** The reputation system needs to be able to retrieve reputation information quickly, providing reasonably predictable perceived quality of service to the users.
- **Scalability.** The sheer volume of reputation information that needs to be stored and processed in a global-scale setting requires a highly scalable reputation system to handle it.
- **Efficiency.** As the size of the distributed reputation system increases to cope with increasing workload, it is important to do so without incurring prohibitive amounts of network traffic and turning the underlying network into a bottleneck.
- **Load balancing.** The reputation system has to be able to spread its storage and processing load relatively evenly among its nodes to improve uniformity of service and enhance reliability.

[1] `http://www.utilitycomputing.com`
[2] `http://www.gridcomputing.com`
[3] `http://www.ebay.com`
[4] `http://bamboo-dht.org/`

SAC'06 April, 23-27, 2006, Dijon, France.

We outline the XenoTrust model, describe Bamboo DHT, and discuss related work in Section 2. We analyse the BambooTrust architecture in Section 3, and demonstrate its effectiveness and scalability by means of experimental evaluation in Section 4. Finally, in Section 5 we present our conclusions and outline areas of future work.

2. RESEARCH CONTEXT

2.1 XenoTrust

Global public computing platforms comprise clients and servers, collectively termed *participants* or *users*. After interactions, participants assess the performance of other participants in dimensions such as reliability, honesty, and throughput. Using these assessments they form *statements* and submit them to the XenoTrust system. A statement is a unit of reputation information; it expresses the value that a certain *subject's* performance has on a specific dimension, as rated by another participant. For instance, "participant Y says that participant X's reliability is 40%".

For scalability and performance, XenoTrust facilitates the aggregation of reputation information. Instead of querying the system every time information is required, users submit persistent queries, termed *rule-sets*, specifying how aggregation is to be performed — e.g. "the average reputation of participant X according to participants Y and Z". These are stored in XenoTrust and evaluated periodically. For scalability, participants can request to be asynchronously notified when these evaluations change more than a *trigger* value instead of having to continuously poll the system.

The computation that XenoTrust undertakes is expected to increase as the system size grows. Centralised or clustered architectures such as the one used by Google [8] and other search engines typically work well in environments where reads are far more frequent than writes, and writes only happen for quantities of several megabytes. In our case, writes are much more frequent and happen for small quantities of data — reputation statements. Also, in centralised approaches machines are financed and maintained by a single organisation, and often are co-located. We believe that a highly distributed system — such as an open distributed hash table (DHT) — is more suitable for allowing the dynamic scaling of the trust management system to cope with high update rates from globally distributed clients.

Furthermore, using a distributed reputation system instead of a central supercomputer or cluster fits better with the global public computing model, as BambooTrust nodes can themselves be flexibly deployed on PlanetLab nodes or XenoServers. Additionally, using a DHT provides transparency and node anonymity. Also, some of the available DHTs — such as Bamboo, discussed below — facilitate effective handling of the *churn* caused by continuous and arbitrary node arrivals and departures.

2.2 Bamboo DHT

The Bamboo distributed hash table system [14] is based on the Pastry [15] routing geometry, which performs lookups in $O(logN)$ hops. Each Bamboo node's data is replicated on a set of nodes called its *leaf-set*, providing improved fault tolerance. Also the modified Pastry algorithm that Bamboo uses has been proven to handle churn very effectively.

Bamboo uses the Staged, Event-Driven Architecture [16], a single-threaded programming style running a small number of threads (usually one per CPU) processing individual events from an event queue. This concurrent programming method has been proved to scale better than thread-based methods.

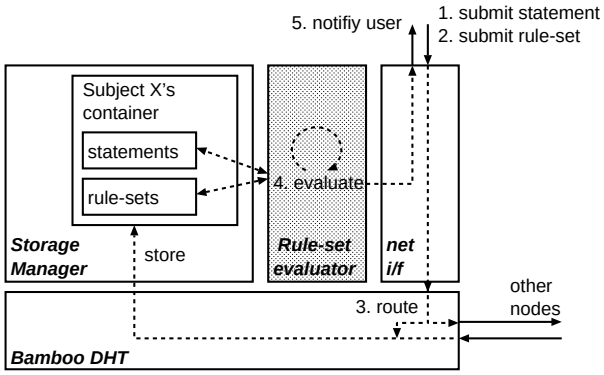

Figure 1: Architecture of a BambooTrust node

Bamboo exports the standard DHT `put(key, value)` and `value = get(key)` interfaces for adding and retrieving data to and from the system. It performs routing of data to the node with the numerically closest node id to the data key. Interfaces are also provided to access and manipulate the data stored within a node. An iterator provides functionality for retrieving all the node's data in key order.

2.3 Related work

eBay, the popular auction site, employs a centralised reputation management system where participants submit ratings about each other by means of a *feedback forum*. Participants' performance is rated as positive, negative, or neutral after each transaction. A participant's trust rating is computed as the sum of the positive feedbacks minus the sum of negative feedbacks.

Yu and Singh propose a framework for combining reputation information from a distributed set of users [17], based on the submission of referrals to acquaintances. Yenta [7] and Web of Trust [11] provide trust models and computational methods for distributing and aggregating reputation information.

Several other trust models and implementations have been developed over the last ten years [10, 12, 2, 3]. The trust model closest to XenoTrust is the one proposed by Aberer and Despotovic [1].

All these approaches focus on trust models, algorithms for aggregating trust information, and mechanisms for interpreting reputation feedback. We have not been able to find any other deployed systems that transparently distribute the *computation* and *storage* of reputation for performance, scalability, load balancing, and efficiency.

3. ARCHITECTURE

BambooTrust is built as a peer-to-peer system; it consists of a number of *nodes*, which are identical in terms of the functionality they perform. This provides ease of implementation, as well as important fault tolerance and scalability benefits, discussed in Section 4.

The architecture of a BambooTrust node is shown in Figure 1. Users communicate with a node using a *network in-*

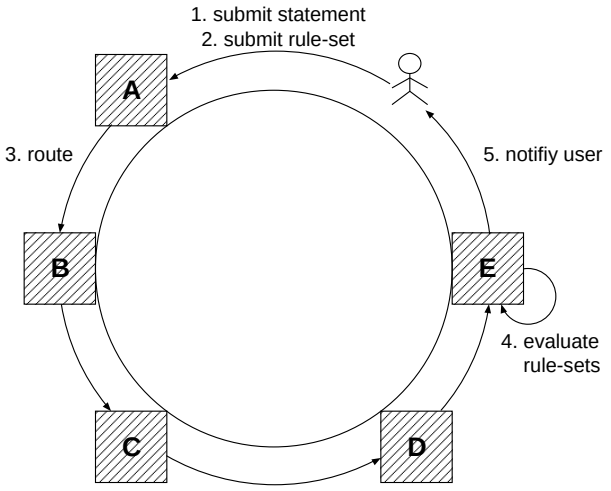

Figure 2: BambooTrust operation (nodes A-E)

terface it exports, in order to submit statements or rule-sets. Bamboo DHT determines whether a statement or rule-set is to be stored locally, or forwarded to another node. A node's *storage manager* is responsible for storing statements and rule-sets locally. The routing and transfer of rule-sets and statements between BambooTrust nodes is managed by the Bamboo DHT. The results of rule-sets are periodically calculated by the *rule-set evaluator*, which may trigger user notifications accordingly. The operation of the system is described in more detail below.

3.1 System operation

BambooTrust users can *submit* statements or rule-sets to any BambooTrust node — operations 1 and 2 in Figures 1 and 2. Statements and rule-sets get routed to the node that is responsible for storing them — operation 3.

Rule-sets are periodically evaluated on the node on which they are stored — operation 4. If the result is such that the user needs to be notified, a message is sent asynchronously to the user — operation 5.

3.1.1 Statement and rule-set submission

Statements and rule-sets can be submitted by users to any BambooTrust node — operations 1 and 2 in Figures 1 and 2. Discovery of BambooTrust nodes can be achieved with the help of a third-party service, such as a search engine, a web site, or a an on-line database — this is not part of the work presented in this paper.

The fact that all BambooTrust nodes can receive, route, and process rule-sets and statements is important for ensuring the fault-tolerance of the system. It also improves responsiveness, as a nearby, low-latency node can be chosen by a user.

A statement or rule-set has to be *signed* using the private key of the participant who is submitting it, for authentication. BambooTrust nodes *retrieve* participants' public keys — necessary for message verification — by communication with the global computing system's authority component — for example PlanetLab Central (PLC) in PlanetLab, or XenoCorp in the XenoServer platform. Statements and rule-sets are also *timestamped* to avoid replay attacks.

3.1.2 Routing and storage of data

Each subject — i.e. participant for whom statements are submitted — is allocated a *container* on a BambooTrust node, as shown in Figure 1. All statements and rule-sets related to a subject are stored in its container. Containers are replicated on more than one nodes for fault-tolerance, and nodes store several containers.

When a statement or rule-set is submitted by the user to a node, it needs to be *routed* to the nodes that hold its containers. BambooTrust uses a *hash* of the subject identifier as the key passed to Bamboo DHT for routing the data. Bamboo then forwards and stores the data in the appropriate node and container — operation 3 in Figures 1 and 2. If there is no container for a subject when a statement or rule-set is submitted, a new one is created.

Rule-sets and statements for the same subject are stored on the same node. As a rule-set can only involve one subject, the need for message passing between nodes during rule-set evaluation is effectively nullified — as all statements for the rule-set's subject reside in the same node. This proves to be an effective strategy, as the number of subjects is far greater than the number of BambooTrust nodes. In the opposite case a different distribution scheme might have to be considered.

3.1.3 Rule-set Evaluation

Each node periodically evaluates all rule-sets it stores locally — operation 4 in Figures 1 and 2. The *rule-set evaluator* module cycles through the containers present on the node, evaluating all rule-sets of each container based on the relevant statements it stores. A statement is relevant if it rates the same property that the rule-set evaluates — e.g. "performance" — and if the advertiser is included in the list of participants' whose opinions are to be considered. Each rule-set is evaluated in turn so there is no conflict between rule-sets that reference the same statements.

For example, a server would evaluate a rule-set in its container translating to "the average reputation of participant X according to participants Y and Z" using all statements in X's container submitted by Y or Z and rating X's reliability — such as "participant Y says that participant X's reliability is 40%".

The *atomic* rule-sets employed by the XenoTrust model require simple calculations, such as for calculating the maximum, minimum, or average values of sets of statements. More complex rule-sets can be composed by sequences of atomic ones, as shown in [5].

3.1.4 User Notification

When submitting a rule-set, a user is effectively registering with a *publish/subscribe* facility that BambooTrust provides. Each rule-set states a trigger value to ascertain when an event needs to be triggered to notify the user of significant changes in its result. A rule-set also contains an IP address and port where the user expects to receive such updates.

To ensure that only authenticated BambooTrust nodes can submit such updates to the user, BambooTrust nodes must *sign* messages using a common private key. The user, having BambooTrust's public key, is able to verify the message. Messages are also *timestamped* to avoid replay attacks, before being sent to the user — operation 5 in Figures 1 and 2.

RS	Avg(ms)	Min(ms)	Max(ms)	Std dev
1	2.22	1	8	1.04
2	2.25	1	7	1.04
3	2.33	1	8	1.05

Table 1: Time (average, minimum, and maximum) required to compute rule-sets 1,2, and 3, and standard deviation of time measurements (experiment repeated 100 times)

4. EVALUATION

It is important that BambooTrust processes reputation information with high performance and scales well to accommodate large numbers of participants. Furthermore, as the system grows, load should be balanced between the nodes, and traffic on the underlying network should be kept to a minimum.

4.1 Experimental setting

The experiments were carried out on 56 3.2GHz Pentium 4 machines with 1GB RAM and 160GB SATA disks. The software was developed in Java 1.5.0 and executed on a Windows XP platform. Nodes were connected using a 100Mbps network, and were physically co-located.

We generated 95,000 random statements and 5,000 rule-sets on 500 subjects, and added them to the system by communication with BambooTrust nodes. Tests were run on different BambooTrust network sizes, starting from a single node and going up to 56, in increments of four nodes.

4.2 Performance

It is important that BambooTrust can evaluate rule-sets quickly in order to ensure efficient operation as the system scales to include large numbers of statements and rule-sets. It is also desirable that rule-set computation time is fairly constant to allow for a predictable perceived quality of service on the user side.

We used a the data set described above and submitted the following rule-sets to the system: 1. *"the maximum value of participant X's reputation according to all participants"*, 2. *"the average value of participant X's reputation according to participants Y and Z"*, and 3. *"the standard deviation of participant X's reputation according to all participants"*. Each rule-set was evaluated 100 times.

Table 1 shows the average, minimum, maximum, and standard deviation of computation times. The results show that rule-sets are computed quickly, and that their computation time is fairly constant. This suggests that BambooTrust can provide good perceived quality of service to users.

4.3 Scalability

Adding more BambooTrust nodes to the system should result to a near-linear performance improvement, in order to allow the system to grow to cope with increasing participation. We measured the system *throughput* as the number of nodes increased, keeping the amount of data — i.e. statements and rule-sets — constant, by monitoring and timing the periodical evaluation of rule-sets at each node.

Figure 3 shows the statistics aggregated and plotted for the system as a whole showing the average rule-sets evaluated per-second as a function of the number of nodes present in the BambooTrust network. The performance of the system improves linearly with the number of nodes present. This is a very encouraging result, demonstrating that BambooTrust can easily scale to accommodate increasing numbers of participants.

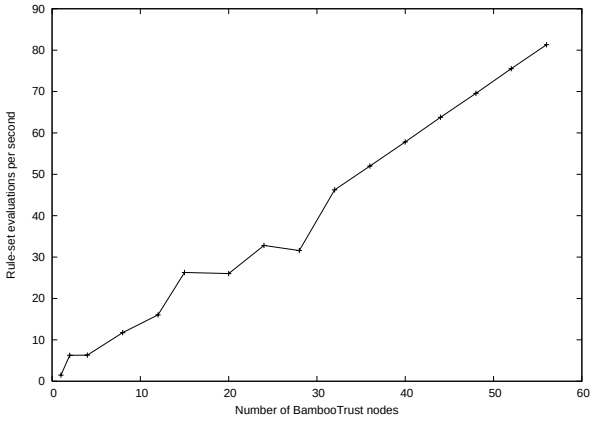

Figure 3: Total system throughput as number of BambooTrust nodes increases

4.4 Efficiency

While the system has been shown to scale well in terms of processing throughput achieved, it is also crucial to demonstrate that it can do so *efficiently*, without generating prohibitive amounts of network traffic.

Our system uses Bamboo DHT for the communication between nodes. According to Bamboo documentation, internode management traffic increases logarithmically with the number of nodes present in the Bamboo network.

In this experiment we measured the total bytes of data sent and received by each BambooTrust node, as the system size was increasing. We found that, as expected, all generated traffic between BambooTrust nodes was due to Bamboo DHT management operations, as rule-sets always operate on statements stored in the same node. Our experiment also confirmed that the increase in Bamboo management traffic as the network size increases is logarithmic, as expected — as described in [14], the leaf-set size is logarithmic of the network size and management traffic is only to and from nodes in the leaf-set.

This is an important result, as it demonstrates that new nodes can be added on demand to allow the system to scale without quickly turning its network into a bottleneck.

4.5 Load balancing

For effective distribution of computation, the load must be spread relatively evenly between BambooTrust nodes. This experiment recorded the number of rule-sets and statements — collectively termed *items* — stored at each node. There were 56 nodes present and a total of 100,000 items, as described in Section 4.1.

As Figure 4 shows, BambooTrust achieves an even spread of data amongst nodes, ensuring a respectively even workload and effectively distributing the storage and computation. This is attributed mainly to the effectiveness of our hashing function mapping subject identifiers to Bamboo keys.

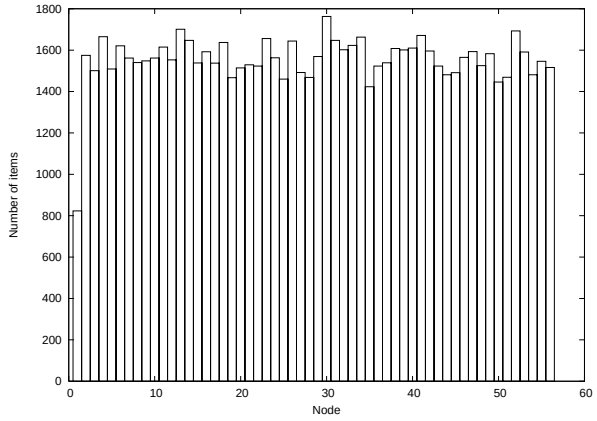

Figure 4: Number of items (statements and rule-sets) stored and evaluated at each node

5. CONCLUSIONS AND FUTURE WORK

In this paper we described BambooTrust, a practical, scalable, high-performance distributed trust management system based on the Bamboo DHT and the XenoTrust model. We presented the statement submission and routing, rule-set evaluation, and asynchronous user notification mechanisms facilitated. Furthermore, we demonstrated that the system performs, scales, and balances load more than adequately well, incurring a very low network traffic penalty.

In the future, we plan to deploy and test BambooTrust in existing global public computing platforms, such as PlanetLab and XenoServers. We also plan to investigate algorithmic optimisations for rule-set computation, as well as mechanisms to detect or prevent the submission of malicious statements.

Acknowledgements

We would like to thank Jon Crowcroft and Tim Moreton for their valuable suggestions. We also wish to thank the anonymous reviewers for their constructive feedback while preparing the final version of this paper.

6. REFERENCES

[1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *CIKM*, pages 310–317, 2001.

[2] T. Beth, M. Borcherding, and B. Klein. Valuation of Trust in Open Networks. In *Proceedings of the 3rd European Symposium on Research in Computer Security – ESORICS '94*, pages 3–18, 1994.

[3] A. Chavez and P. Maes. Kasbah: An Agent Marketplace for Buying and Selling Goods. In *Proceedings of the 1st International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technologies (PAAM'96)*, 1996.

[4] B. Dragovic, S. Hand, T. Harris, E. Kotsovinos, and A. Twigg. Managing Trust and Reputation in the XenoServer Open Platform. In *Proceedings of the 1st International Conference on Trust Management*, pages 59–64, Heraklion, Crete, Greece, May 2003. Also published in Springer-Verlag Lecture Notes in Computer Science (LNCS), Volume 2692, pp. 59-74.

[5] B. Dragovic, E. Kotsovinos, S. Hand, and P. Pietzuch. XenoTrust: Event-Based Distributed Trust Management. In *Proceedings of the 2nd IEEE International Workshop on Trust and Privacy in Digital Business (DEXA Workshop)*, pages 410–414, Sept. 2003.

[6] A. Fernandes, E. Kotsovinos, S. Ostring, and B. Dragovic. Pinocchio: Incentives for honest participation in distributed trust management. In *Proceedings of the 2nd International Conference on Trust Management (iTrust 2004)*, pages 63–77, Oxford, UK, Mar. 2004. Also published in Springer-Verlag Lecture Notes in Computer Science (LNCS), Volume 2995, pp. 63-77.

[7] L. N. Foner. Yenta: a Multi-Agent, Referral-Based Matchmaking System. In *Proceedings of the 1st International Conference on Autonomous Agents (AGENTS '97)*, 1997.

[8] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, 2003.

[9] S. Hand, T. L. Harris, E. Kotsovinos, and I. Pratt. Controlling the XenoServer Open Platform. In *Proceedings of the 6th International Conference on Open Architectures and Network Programming (OPENARCH)*, Apr. 2003.

[10] A. Josang. The right type of trust for distributed systems. In *Proceedings of the New Security Paradigms Workshop*, 1996.

[11] R. Khare and A. Rifkin. Weaving a Web of Trust. *World Wide Web Journal*, 2(3):77–112, 1997.

[12] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[13] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I)*, Oct. 2002.

[14] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference (USENIX '04)*, June 2004.

[15] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001.

[16] M. Welsh, D. E. Culler, and E. A. Brewer. SEDA: An Architecture for Well-Conditioned, Scalable Internet Services. In *Proceedings of the 18th Symposium on Operating Systems Principles (SOSP '01)*, Oct. 2001.

[17] B. Yu and M. P. Singh. Distributed reputation management for electronic commerce. *Comput. Intelligence*, 18(4):535–549, 2002. Special Issue on Agent Technologies for Electronic Commerce.