Hierarchically Federated Registration and Lookup within the perfSONAR Framework

Jason Zurawski [*], Jeff Boote [†], Eric Boyd [†], Maciej Glowiak [‡], Andreas Hanemann [§], Martin Swany [*], and Szymon Trocha [‡]

{zurawski, swany}@udel.edu, {boote, eboyd}@internet2.edu, {mac, szymon.trocha}@man.poznan.pl, hanemann@dfn.de

Abstract

A widely-distributed network monitoring system requires a scalable discovery mechanism. The “Lookup Service” component of the perfSONAR framework is able to manage component registration, distill resource data into tractable units, and respond to queries regarding system and performance information.

A model of organizing and distributing information is presented to support both dynamic environments where services frequently change as well where different administrative configuration requirements exist. These interactions are accomplished by forming “federated hierarchies” to share information amongst the various logical overlays.

1 Introduction

Advanced national and regional networks motivate interest in the deployment of status monitoring applications [4, 6]. This software ensures availability and efficiency within the infrastructure by monitoring various performance metrics. To unify the often distributed task of monitoring, the concept of a Service Oriented Architecture (SOA) offers the ability for specialized, autonomous services to join under a common access scheme. Thus, it is possible to separate the roles of monitoring, storage, processing, and visualization of data into specialized service instances [7, 9].

The role of a registration and discovery service involves maintaining information related to the set of available resources. The success of this service depends on keeping the available information current in order to provide an accurate view of the health and size of the framework. This goal becomes challenging in dynamic environments where the lifetime of a service may vary.

This paper presents a solution to to the problem of extending a registration and discovery mechanism (referred to as the “Lookup Service”, or LS) to function in dynamic situations that may span multiple administrative domains. We illustrate the complexities required to implement this idea, outline our solution, and present the results of empirical testing.

The paper is structured as follows: Section 2 features an overview of *perfSONAR*. Section 3 presents some related solutions to the resource discovery problem. Section 4 shows implementation details. Section 5 features results of experimentation. We conclude our work with Section 6.

2 perfSONAR Overview

perfSONAR is a performance monitoring tool designed to aid in the discovery, troubleshooting, and solution of network performance problems potentially between numerous networks. This framework is comprised of entities meant to collect, store, analyze, and deliver data using well-defined protocols.

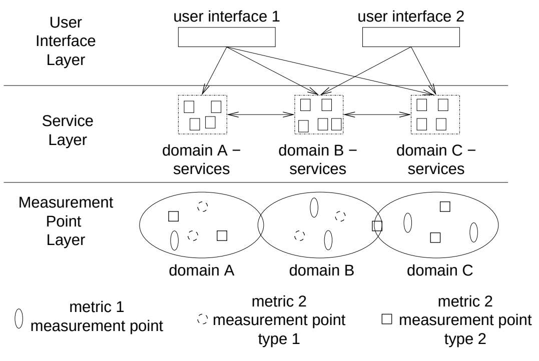

Figure 1. perfSONAR Measurement Framework

The general monitoring infrastructure implemented by *perfSONAR* is illustrated in Figure 1. The Measurement Points (MPs) are the base layer of this framework and have the responsibility of measuring as well storing distinct network metrics. The Service Layer [5] is the middle layer of the system that is divided across administrative boundaries. The User Interface Layer consists of visualization tools which present the data in customizable ways. In addition, this layer allows users to perform tests using services available at the lower layers of the framework.

[1]**Department of Computer and Information Sciences, University of Delaware**, Newark, DE 19716, USA

[2]**Internet2**, 1000 Oakbrook Drive, Suite 300, Ann Arbor MI 48104, USA

[3]**Poznan Supercomputing and Networking Center**, Noskowskiego 12/14, 61-704, Poznan, Poland

[4]**German Research Network (DFN)**, c/o **Leibniz Supercomputing Center**, Boltzmannstr. 1, 85748 Garching, Germany

The function of the LS is to accept registration requests from services in *perfSONAR*. As each component updates its information, other components and clients may locate these deployed services via queries. All service descriptions and network metrics are defined using XML schema and encoded into XML. Storage of these descriptions is performed in native XML databases and querying is performed using XPath and XQuery.

The initial LS was focused on single domain functionality; all services interacted with the same LS instance. This single server approach does not facilitate information sharing between deployments. Our work investigates an architecture for cooperation of LSs, both within and amongst administrative domains. A primary goal of *perfSONAR* is to allow "federations" between disjoint monitoring deployments. Federated systems are designed to work together while tolerating different configurations and policies.

Federated relationships between instances requires changes in the presentation and storage of information. Sharing data may require omitting items due to policy; thus the ability to efficiently transform data, yet still have it maintain meaning becomes critical. Because the information resides as XML, distillation is performed through XSLT [11] transformation.

3 Related Work

Most distributed frameworks offer mechanisms to aid in resource discovery and information integration. While there are many entries in this realm, the unique requirements of *perfSONAR* have mandated the design of a new system.

Globus has developed the Monitoring and Discovery System (MDS) [3] which is able to summarize resources and federate with related monitors. This service contains monitoring features that exceed our requirements; each component of *perfSONAR* is required to perform self registration to remain autonomous.

MonALISA features the Clarens [2] monitoring system to handle issues such as security, registration and discovery, and monitoring. The abundance of offered features will go unused or become duplicated by the remainder of *perfSONAR*, as the framework offers small specialized services.

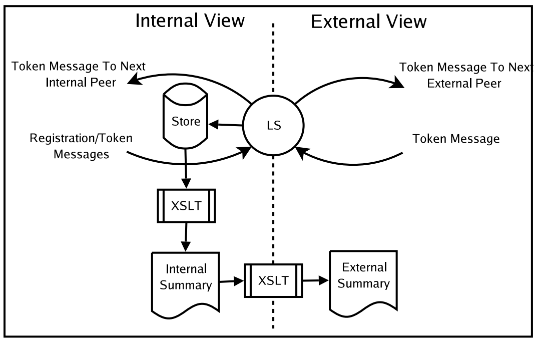
Internal View
External View
Token Message To Next Internal Peer
Token Message To Next External Peer
Store
LS
Registration/Token Messages
Token Message
XSLT
Internal Summary
XSLT
External Summary

Figure 2. Dual roles for each LS

Universal Description, Discovery and Integration (UDDI) [10] is a specification for publishing and locating information. UDDI defines an information framework that enables description and classification of organizations, related services, and the technical details about the interfaces of the exposed web services. While general and widely accepted, [1] has found that dynamic applications requiring constant information updates can cause performance and scalability issues.

The concepts and primitives offered by these solutions and the many others would be an adequate start, but a single solution could not be found that would not require extensive modification to suit our needs. The LS is required to be nimble when operating; data storage must remain fast and reliable. As such we proceed with an intention to maintain a small, dedicated resource and discovery service that communicates and stores information natively as the other *perfSONAR* services already operate.

4 Implementation

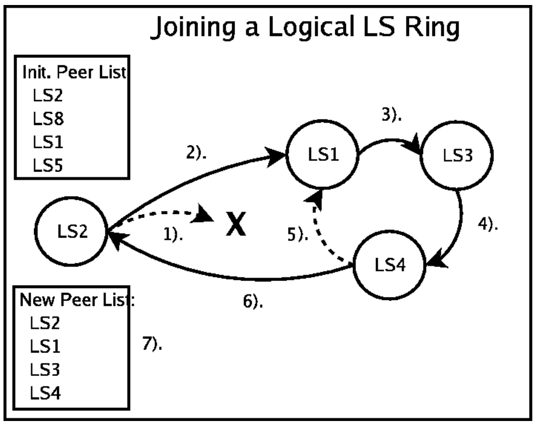

Figure 3. An LS joining a logical ring

The current LS is the basis for further development; this entity contains operations able to accept data, perform queries, manage resources, and purge expired data. The ability to exchange messages between service peers, summarize and store registered data for sharing, and handle queries for distributed information are necessary for success. A central part of communicating with peers is an algorithm to form and maintain communication within logical groups. Also, methods for the selection of hosts to perform leadership roles within the groups must be defined.

The organization of LS instances can be viewed in two ways: a flattened group of peers versus that a hierarchy where designated "leaders" communicate across the group boundaries. Both methods will take part in a token passing protocol with the goal of exchanging summarized information. This communication model allows leaders to communicate as part of two logical rings: "inter-domain" peers and "intra-domain" peers. To better visualize this distinction, consider Figure 2.

Data summarization must be both flexible and efficient. The exchange of large data sets is unacceptable; succinct breakdowns of information will offer a compact and complete description of what services have to offer. We define our summarization procedure in terms of XML Stylesheet Language Transformations (XSLT). The level of summarization performed in each group is controlled by the policy decisions that exist across the federations. By using different stylesheets, the LS deployments may implement various policies to accomplish data propagation. The data we wish to minimize via transformation is formatted in the NM-WG [8] style; a full description of which is presented in [12].

LS instances maintain two sets of summaries; an inter-domain summary of local peers as well as a similar intra-domain summary that reflects the global information view. During queries, each LS instance must examine locally first. If found, the answer will be returned at once, otherwise the answer can be sought with two different methods: the message can be forwarded to a group leader on behalf requester or a response containing a reference to a group leader could be returned. The initial results presented in this work are for the latter (i.e. iterative) query scheme. Future development will allow for a configurable query behavior.

The algorithm used to dictate membership in peer groups must be run each time an LS begins to function. Each LS instance maintains a list of peers that for now is statically configured but in time could be created by automatic discovery methods. This procedure is outlined in Figure 3. The initial list contains many deployments which may or may not be active. Actions 1 and 2 of the algorithm call peers aiming to get a response. After contact is established, lists at the peers are updated. Actions 3, 4, 5, 6, and 7 show how the peers recognize the new addition by distributing the information through the overlay, thus completing a cycle.

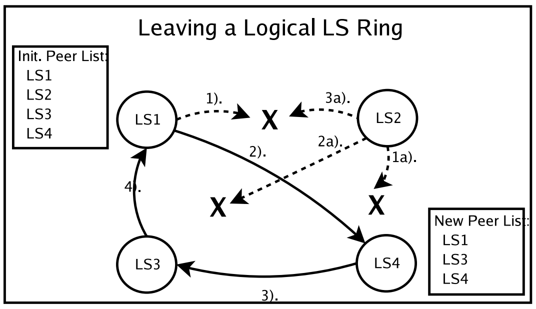

Figure 4. An LS becoming disjoint from a logical ring

The algorithm is subject to various rules regarding timeouts to prevent the loss of a token. If instances are not "heard from" for a length of time a new token will be created and passed, skipping the LS instance that may have disappeared. The situation of node failure is described in Figure 4. Action 1 shows a failure in passing the token. When an LS cannot reach the next available peer, the list of peers is checked to find out the next available service. Action 2 sends a new message to the next in line, making sure to remove the unreachable LS from the list. The failing LS cannot contact anything on its peer list and will gradually lengthen the time between contact attempts as demonstrated in actions $1a$, $2a$, and $3a$. The remainder of the ring that is able to communicate is able to do so in Actions 3 and 4 using the updated peer list.

Designating group leaders is handled by a simple "election" algorithm. Each time a token arrives containing a peer list, the leader is known by examining the ID value of the peers, and choosing the smallest. These ID values can be manipulated by system administrators adding configuration "salt" to help specific instances function in this role. To handle the potential failure of a leader, intra-domain data can be cached by the second or third place winner of the leader election. The failure of a leader does not cause a major disruption in service; within a single local cycle the new leadership will be established and queries will be honored due to the caching procedure.

5 Experimental Results

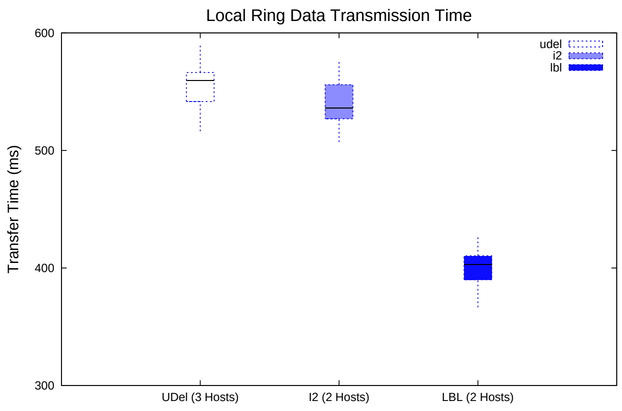

Figure 5. Information propagation in a LAN

We present a subset of the experimentation done during the development process to test performance of the modified *perfSONAR* LS. The experiments focus on feasibility of information exchange and delivery in distributed environments. These tests are small by comparison to a dedicated perfSONAR deployment but provide a decent performance picture. The experimental environment consists of three local area network (LAN) configurations joined via wide area (WAN) links. The LANs where located at the University of Delaware, Internet2, and Lawrence Berkeley National Laboratory.

The first results demonstrate the performance observed when exchanging information between peers over the LAN as well as the WAN connections. The experiment tests how long it takes for the registration of a service to a single LS to propagate throughout the two group organizations.

Figure 5 displays the transmission time for information in

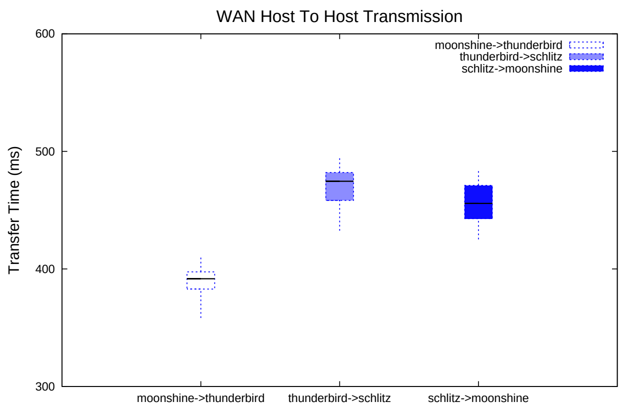

Figure 6. Information propagation host to host in a WAN

each of the three LAN environments. The observations are based on transmitting a complete summary of services offered within the LAN, decoded from the original data stores on each host by using XSLT translation. The size of the summary file to be transmitted ranged from 7KB to 9KB, depending on the number of deployed services and the effectiveness of the transformation. Additionally, Figure 6 illustrates the total time required for information to propagate on the WAN. The size of the WAN summary file ranged from 3KB to 6KB, depending on the size of the LAN summary files involved. The results are acceptable based on bandwidth and latency observations made on the LAN and WAN links, but have been omitted for brevity.

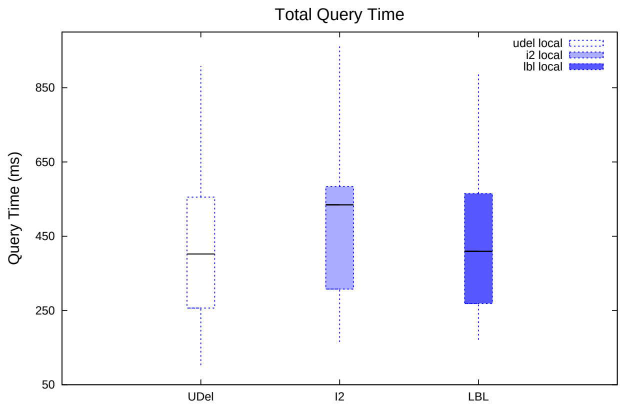

Figure 7. Query times at each domain

The second results demonstrate the performance of the iterative query scheme. A client will query an LS for specific data and be returned either data, or contact information of additional LS instances. The latter case will trigger additional queries. A series of queries was performed (where the results may or may not be available somewhere in the network of LSs). Queries for 50 resources were requested from each domain. Instances requiring multiple queries (in the case of 2 LS replies, or a redirect to another LS) will add to the time required of a single query.

Figure 7 displays the results of these queries. The results indicate that query time will remain similar (although variable as demonstrated by the height of both the boxes and whiskers) across domains. This is a clear indication that summarization will benefit overall query performance across a federation.

6 Conclusion

This work has discussed the problems and solutions related to implementing a multi-domain Lookup Service to manage components of the perfSONAR framework. While still in the early stages of development and deployment, experimentation has shown that performance is within the expected boundaries for service instances separated by a LAN or WAN. Additional work to provide query functionality and options to ease the burden of federating resources will be explored in future iterations of this work.

References

[1] E. Benson, G. Wasson, and M. Humphrey. Evaluation of uddi as a provider of resource discovery services for ogsa-based grids. In *Parallel and Distributed Processing Symposium, IPDPS 2006*, Rhodes Island, Greece, April 2006.
[2] Clarens. `http://clarens.sourceforge.net`.
[3] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. 2001.
[4] IST-MOME (Monitoring and Measurement Cluster) project web site. `http://www.ist-mome.org`.
[5] A. Hanemann, J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. Lapacz, M. Swany, S. Trocha, and J. Zurawski. Perfsonar: A service oriented architecture for multi-domain network monitoring. In *Service-Oriented Computing - ICSOC 2005, LNCS 3826, Springer Verlag*, pages 241–254, Amsterdam, The Netherlands, December 2005.
[6] S. Leinen, M. Przybylski, V. Reijs, and S. Trocha. Testing of traffic measurement tools, September 2001.
[7] H. Newman, I. Legrand, P.Galvez, R. Voicu, and C. Cirstoiu. Monalisa: A distributed monitoring service architecture. La Jola California, Mar 2003. CHEP 2003.
[8] Network Measurements Working Group (NM-WG). `http://nmwg.internet2.edu`.
[9] Performance focused Service Oriented Network monitoring ARchitecture. `http://www.perfsonar.net`.
[10] UDDI. `http://www.uddi.org/`.
[11] Xml Stylesheet Language Translation. `http://www.w3.org/TR/xslt`.
[12] J. Zurawski, M. Swany, and D. Gunter. A scalable framework for representation and exchange of network measurements. In *IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, Barcelona, Spain, March 2006.