

DOOR: A Distributed Object-Oriented Repositories for Network Management*

Alan Bivens, Patrick H. Fry, Li Gao, Mark F. Hulber and Boleslaw K. Szymanski

Department of Computer Science

Rensselaer Polytechnic Institute, Troy, NY USA 12180-3590

Email: {bivenj, fryp, gaol, hulber, szymansk}@cs.rpi.edu

1 Introduction

As networks grow larger and more complicated, their management also becomes more difficult, especially when there is no central authority to coordinate management. This paper describes a Distributed Object Oriented Repositories (DOOR) for facilitating management of large networks. This framework serves as a middleware between a collection of independent network management agents and network nodes.

At the core, DOOR provides a secure, efficient, and fault tolerant method for the acquisition, management, aggregation, and caching of network data and objects. This requires the use of mobile agents and a dynamic interface to support management and collection protocols such as LDAP and SNMP. We are working with a 100K node network that can't be managed using traditional strict hierarchical approaches. In the first phase of this project, described in this paper, we are primarily supporting the collection, aggregation, and management of SNMP data and related objects.

*This work is supported in part by DARPA grant F30602-97-C-0274

Through DOOR, network performance visualization and other “upstream” application components can channel requests for objects and data. Requests outside the domain of a repository are managed and forwarded to an appropriate cooperating repository. Regardless whether the request is in or outside the domain of a repository, the request is managed and responded to in a manner virtually transparent to the requester. Through several mechanisms, the DOOR system takes steps to meet the temporal requirements of the requester. If recent data is available in the DOOR, the object and its data are immediately provided to the requester. Otherwise, the DOOR, having been configured with the tools to collect this type of object, either communicates with an existing agent or launches an appropriate agent in order to provide the object to the requester. In anticipation of frequent requests from one or multiple sources, the DOOR system may proactively collect network data in order to meet requests in an efficient manner.

This paper focuses on the first phase of the development of our DOOR infrastructure, emphasizing the following components:

- Data Management Requirements
- DOOR Architecture
- Implementation Details
- Preliminary Timing Results

In this extended abstract we will focus on motivation of our work and the basic assumptions of our implementation, leaving the description of the implementation and performance results for the full paper.

| | Interval | Utilization | | Seconds per Request |
|-------|----------|-------------|-------|---------------------|
| 7200s | 1s | 111s | .0154 | .0154 |
| | 10s | 31s | .0043 | .0430 |
| | 15s | 34s | .0047 | .0708 |
| | 30s | 18s | .0025 | .0750 |
| | 60s | 9s | .0012 | .0750 |

Figure 1: Impact of SNMP Data Polling on Network Router

2 Data Management Requirements

As we experience the growth of networks we also see the proliferation of support at various levels to manage these networks and the activity on them. Directory services such as LDAP and SNMP have been developed, used, and grown over time to collect and disseminate information about networks. As the scale of the network management effort grows, the collection processes and management efforts themselves can put a significant load on the networks being managed.

Our goal is to minimize the negative effects of obtaining network data, while offering much more functionality than existing systems. Figure 1 shows the impact of polling SNMP data from a router. We intend to keep this impact minimal by channeling access to this data through repositories.

Network management places complex requirements on the physical location of the network data. Three major factors are performance, availability, and bandwidth usage. Performance, in this case, has to do with client queries and agent updates. For the clients and agents, their repository must be “nearby” in the sense of the physical layout of the network. This would imply the repository resides on the same subnet or is at most a few hops away. It’s not realistic to expect a repository to reside on every subnet. One repository per some small set of physically close subnets should be

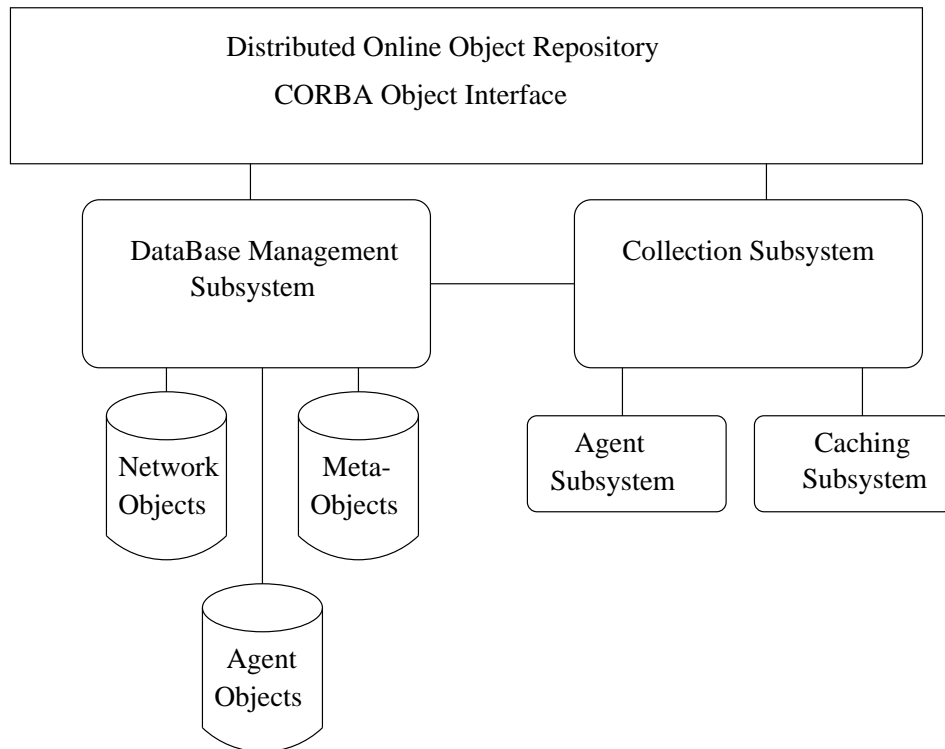


Figure 2: Internal Overview of Distributed Online Object Repository (DOOR)

sufficient.

For performance reasons, the optimal location of the repository would be the network region for which it is holding data. However, data for a network region should be available during periods where that region is unreachable. Therefore, the repository should be somewhere nearby without actually residing in the region.

Another important issue is bandwidth usage. One of the main goals of this project is to provide its services with an absolute minimum impact on the network. It is likely that clients will frequently request data for networks not within the domain of its local repository. In this case, the client sends the request to its local repository and it is the repository's responsibility to retrieve the data from the nonlocal repositories.

From the information model standpoint, the repository to repository communication is forward-

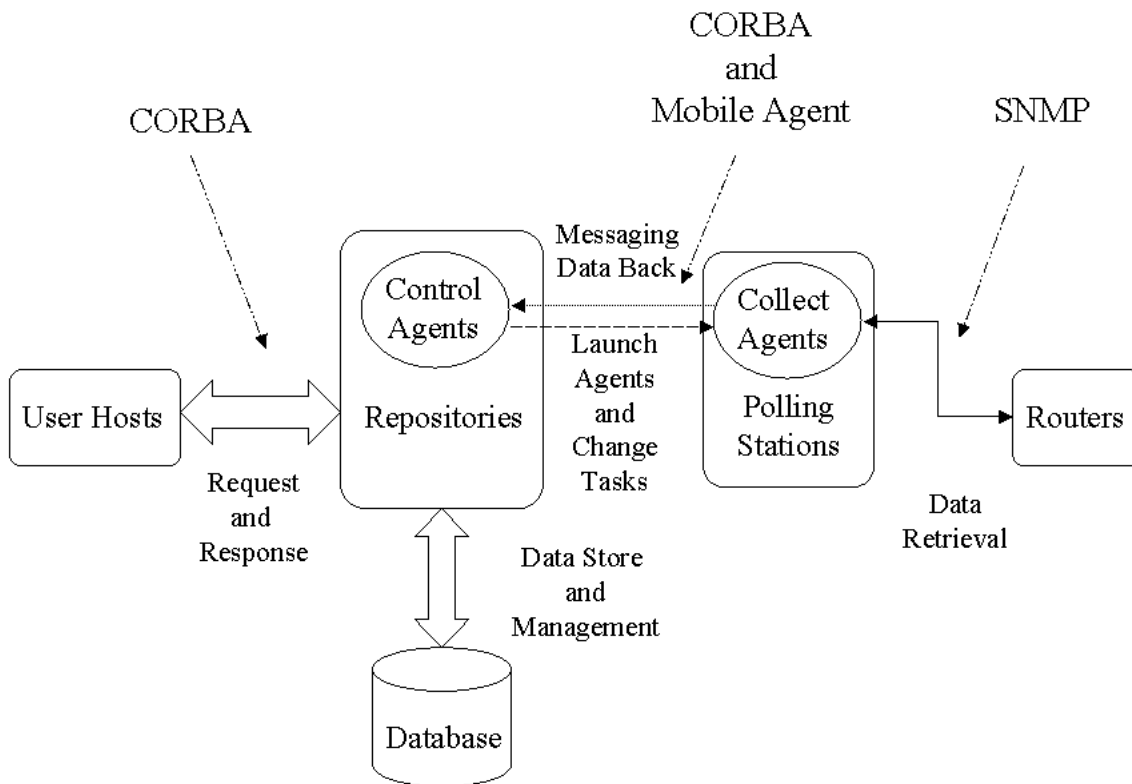


Figure 3: Door Architecture

ing client requests between repositories and receiving the data from the source repository to send back to the client.

3 DOOR Architecture

The DOOR system uses several components to retrieve and process network data. The purpose of each component will be given here, whereas the specifics, including what is actually used, will be described in the next section. The necessary components are listed below:

- Client Interface

- Repository
- Mobile Agents
- Polling Station
- Storage System

The Client interface just needs to be a communication device with the repository. It could be a small stand-alone application or part of a larger application which could involve visualization or other “upstream” application components. It communicates with only one local repository.

The Repository is responsible for the controlling of agents and coordinating requests from different clients as well as other repositories.

The Mobile Agents are sent from the repository to a polling station to request the actual data from the destination object (typically a router.)

A Polling Station is a critical component, because it would be difficult to send an agent to sit on the router itself to request data. Many routers use a custom operating which we don’t know how to use. If we did know how to use it, we wouldn’t know how to upload to it. If the last two details were known, we would have to know them for every type of router. Even if this were possible, it could force an extra strain on the router. Therefore, we pick a machine very close to the router in which an agent can sit and poll the router with minimum network traffic.

The Storage System is not strictly necessary to have a DOOR system functioning. In the final version of the system it likely will be a database used to store values retrieved with the agents. The storage system could be included in the repository to provide historical data to the clients.

4 Summary

Large networks consisting of thousands of nodes can't be managed using the strict hierarchical approaches of the past. In this paper we discuss proactive network management and the role played by our Distributed On-line Object Repositories (DOOR) in the integration and support of the collection of the network management data. We outline the requirements which the network management projects place on DOOR and discuss our implementation plans. Our hope is to improve our prototype system for this network management project and then identify ways we can extend DOOR to support other applications requiring distributed databases.