A Scalable Architecture for Network Fault Diagnosis in the Knowledge Plane

George J. Lee GJL@MIT.EDU
Peyman Faratin PEYMAN@MIT.EDU
Steven Bauer BAUER@MIT.EDU
MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, Cambridge MA, 02139 USA

1. Introduction

The Internet enables a wide variety of devices to communicate with one another, but both the distributed administration of the Internet and the complex interactions among failures make problems difficult to diagnose. To address this problem, we propose an architecture for distributed network fault diagnosis that allows data providers, diagnosis providers, and users to exchange information in a standard way. We see this diagnostic architecture as one aspect of the Knowledge Plane (KP)(Clark et al., 2003), a platform which will enable automated network diagnosis, management, configuration, and repair. A general architecture for distributed diagnosis will allow different diagnosis providers, such as intrusion detection systems and domain-specific failure diagnosis systems, to interoperate and exchange data from different sources such as network monitors and Internet tomography systems. Designing such an architecture is difficult because it must be able to support a wide range of data and diagnostic methods as well as scale to large networks with unpredictable network faults potentially affecting millions of users. In this paper we describe an approach for addressing these challenges using an extensible ontology and a scalable routing protocol and present the results of some preliminary experiments.

There has been some related work in networks for exchanging diagnostic information. Wawrzoniak et al. developed Sophia, a system for distributed storage, querying, and processing of data about networks (Wawrzoniak et al., 2004). Thaler and Ravishankar describe an architecture for diagnosing faults using a network of experts (Thaler & Ravishankar, 2004). Gruschke describes how an event management system can use dependency graphs for diagnosis (Gruschke, 1998). Unlike previous work, we consider how to handle large volumes of diagnostic requests by reasoning about the effects of individual network failures.

2. An Agent Architecture for Fault Diagnosis

Our goal is to design a network architecture that allows distributed diagnosis for a wide range of faults. As a starting point, we consider how to diagnose reachability faults in which a user cannot complete a network task because they cannot reach some destination. Reachability faults may result from a variety of causes, including physical network cable disconnection, network misconfiguration, access provider failures, routing failures, and software failures.

In order to automatically collect data about faults, perform diagnosis, and convey diagnoses to users affected by faults, we developed a diagnostic architecture comprising a network of intelligent agents. Each agent may request information from other agents and respond to such requests. In this network architecture, **user agents** make requests for diagnosis, and **diagnosis agents** request data from **data agents** in order to respond to user agent requests.

3. Scalable Routing Using Aggregation

Our architecture must address the issue of scalability. In large networks such as the Internet, a single serious network fault may affect millions of users. To successfully diagnose such faults, our architecture uses aggregation to greatly reduce the number of messages required for diagnosis. Instead of performing a full diagnosis for every request, an agent may determine that multiple requests may be aggregated and answered using existing knowledge.

In order to maximize the effectiveness of aggregation, agents route requests and responses according to the Internet autonomous system (AS) topology. For each AS, there exists a diagnostic agent that knows about failures within that AS. A diagnostic request from a user agent first travels to the diagnostic agent for the user's access provider AS. If the diagnostic agent does not have enough information to respond, then it forwards the request to the agent for the next AS along the AS path towards the unreachable destination. When an agent has enough information to produce a diagnosis, it sends a response that travels along the reverse path of the request, allowing each agent along the path to store the data in the response. Routing along the AS path ensures that if a fault occurs somewhere along the

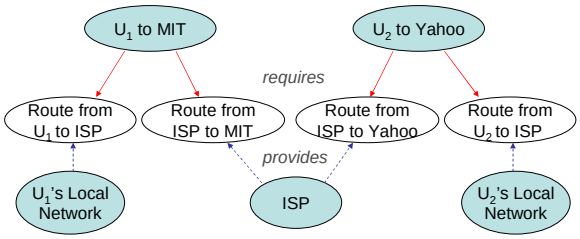

Figure 1. Inferring failures using a dependency graph

AS path, data is collected from every agent with information about a fault while minimizing the average number of requests each agent handles.

In order to aggregate effectively, however, an agent must be able to determine whether it can satisfy multiple requests using existing knowledge. To address this problem, we propose an ontology that allows agents to describe the dependencies between different network components and infer all the network failures that may result from a failed component. This ontology defines **components** and **dependencies**, where components may *require* multiple dependencies. A dependency is satisfied if and only if there exists a component that *provides* that dependency. A component functions if and only if it has not failed and all the dependencies it requires are satisfied. This dependency graph is distributed: each diagnostic agent may define its own local dependency model and incorporate additional dependency information from other agents.

Figure 1 depicts a simple example of how an agent might use a dependency graph for aggregation. The shaded and unshaded ellipses represent components and dependencies, and the solid and dotted lines indicate required and provided dependencies, respectively. To reach their destinations, users U_1 and U_2 must be able to reach their ISP, and their ISP must be able to reach their respective destinations. Since both users share the same ISP, if their ISP fails, then a diagnostic agent can infer that neither user can reach their desired destinations. An actual dependency graph would also model dependencies among many other types of networking components, including DNS, network applications, and link-layer components.

To illustrate the potential benefit of aggregation, we conducted simulations of our routing protocol using an actual AS topology consisting of 8504 nodes using data from Skitter[1]. Figure 2 plots the average number of requests an agent receives based on its distance from the agent of the AS responsible for the failure, where the point "0 hops" represents the responsible agent. In this simulation, the failure affects 1,000,000 users, each of whom makes a diagnostic request. Without aggregation, the responsible agent

[1]http://www.caida.org/tools/measurement/skitter/

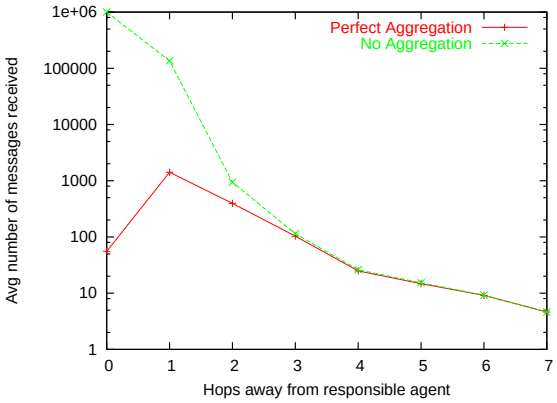

Figure 2. Message Distribution

receives one request from every user agent. With perfect aggregation, if agents can accurately respond to requests using existing knowledge, the agent for the responsible AS only receives at most one request from each of its neighboring agents. The benefit of aggregation diminishes farther away from the responsible agent because the distant agents receive fewer requests and hence have fewer opportunities for aggregation. This plot shows that an effective ontology and intelligent aggregation can greatly reduce the average number of requests agents process.

4. Conclusion and Future Work

In this paper we proposed an architecture for distributed network fault diagnosis that represents data using an extensible ontology and routes requests and responses using aggregation to reduce the average number of requests an agent must process. We are currently developing a prototype of this architecture and plan to conduct more experiments to determine how well it performs under a variety of realistic failure cases.

References

Clark, D. D., Partridge, C., Ramming, J. C., & Wroclawski, J. T. (2003). A knowledge plane for the internet. *Proceedings of SIGCOMM '03*.

Gruschke, B. (1998). Integrated event management: Event correlation using dependency graphs. *Proceedings of DSOM '98*.

Thaler, D. G., & Ravishankar, C. V. (2004). An architecture for inter-domain troubleshooting. *Journal of Network and Systems Management, 12*.

Wawrzoniak, M., Peterson, L., & Roscoe, T. (2004). Sophia: an information plane for networked systems. *SIGCOMM Comput. Commun. Rev., 34*, 15–20.