Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 80 (2007) 896–917
The Journal of Systems and Software
www.elsevier.com/locate/jss

Understanding failure response in service discovery systems

C. Dabrowski *, K. Mills, S. Quirolgico

National Institute of Standards and Technology, Information Technology Laboratory, 100 Bureau Drive, Mailstop 8970, Gaithersburg, MD 20899-8970, United States

Received 1 March 2006; received in revised form 21 November 2006; accepted 23 November 2006
Available online 17 January 2007

Abstract

Service discovery systems enable distributed components to find each other without prior arrangement, to express capabilities and needs, to aggregate into useful compositions, and to detect and adapt to changes. First-generation discovery systems can be categorized based on one of three underlying architectures and on choice of behaviors for discovery, monitoring, and recovery. This paper reports a series of investigations into the robustness of designs that underlie selected service discovery systems. The paper presents a set of experimental methods for analysis of robustness in discovery systems under increasing failure intensity. These methods yield quantitative measures for effectiveness, responsiveness, and efficiency. Using these methods, we characterize robustness of alternate service discovery architectures and discuss benefits and costs of various system configurations. Overall, we find that first-generation service discovery systems can be robust under difficult failure environments. This work contributes to better understanding of failure behavior in existing discovery systems, allowing potential users to configure deployments to obtain the best achievable robustness at the least available cost. The work also contributes to design improvements for next-generation service discovery systems.
Published by Elsevier Inc.

Keywords: Distributed systems; Robustness; Service discovery

1. Introduction

Various teams designed and implemented a first generation of (competing) service discovery systems that enable distributed components to find each other without prior arrangement, to express capabilities and needs, to compose into collections, and to detect and adapt to changes. Each specific design defines a system structure, along with protocols for discovery, monitoring, and recovery. Some designs assume a specific underlying communication technology, some designs focus on one application domain, and some designs were conceived to operate over Internet protocols and to support many applications.

In this paper, we investigate the architectures and behaviors underlying Jini Networking Technology[1] (Arnold, 1999), Universal Plug and Play (UPnP) (Universal Plug and Play Device Architecture, 2000), and the Service Location Protocol (SLP) (Guttman et al., 1999) when subjected to various failures. Elsewhere (Dabrowski et al., 2005), we present a generic model encompassing the designs of these systems and we identify performance issues that could arise. While this previous work considers system behavior absent failures, here we explore the relative ability of discovery systems to cope with different types and intensities of failure.

We reported preliminary results in various conference papers (Dabrowski and Mills, 2001; Dabrowski et al.,

[1] Certain commercial products or company names are identified in this paper to describe our study adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor to imply that the products or names identified are necessarily the best available for the purpose.

* Corresponding author. Tel.: +1 301 975 3249; fax: +1 301 948 6213.
E-mail address: cdabrowski@nist.gov (C. Dabrowski).

0164-1212/$ - see front matter Published by Elsevier Inc.
doi:10.1016/j.jss.2006.11.017

2002a,b, 2003); however, this paper improves upon earlier work in two ways. First, we extend the scope of our results to cover three architectures (two-party, three-party, and adaptive), three failure scenarios (configuration restoration, service acquisition and maintenance, and consistency maintenance), four failure types (power failure and restart, node failure, communication failure, and message loss), and a set of failure detection and recovery techniques at three levels (transport protocols, discovery protocols, and application logic). Second, we increase the amount of data collected and analyzed to obtain better estimates for performance metrics at high failure rates.

This paper contributes to the understanding of service discovery systems. First, this paper characterizes robustness of discovery systems under difficult failure environments. This paper further identifies and discusses the most significant design and configuration decisions that influence robustness. Second, this paper identifies specific design and deployment decisions that could lead to diminished robustness. Third, this paper quantifies the relative cost associated with specific decisions. Overall, the information provided here should contribute to better understanding of failure behavior in existing discovery systems, allowing potential users to configure deployments to obtain the best achievable robustness at the least available cost. Further, results and discussions presented here have contributed to design improvements in the next generation of discovery systems (Sundramoorthy, 2006; Sundramoorthy et al., 2004).

This paper also contributes experimental methods to study robustness in distributed systems. First, we introduce and apply metrics to quantify relative robustness and cost at the application level for various scenarios. Second, we present a technique to decompose aggregate robustness into detection and recovery latency. Using this technique, we show how similar robustness can be achieved through different behaviors arising from particular design choices. Our methods can be adopted, adapted, or extended by other researchers to investigate failure response in distributed systems – a topic due for increased study.

We begin (in Section 2) with a synopsis of existing work comparing and contrasting service discovery systems. Most previous work focuses on functional comparisons, on means for translating among discovery systems, or on improving existing designs. Our own related work (Dabrowski et al., 2005; Bowers et al., 2003; Mills and Dabrowski, 2003; Rose et al., 2003; Mills et al., 2004; Tan and Mills, 2005) attempts to unify designs for several existing discovery systems, and investigates performance problems arising when such systems are deployed at large scale.

In Section 3, we survey the design and function of service discovery systems. We introduce a model to convey concepts across selected systems. Using our model, we describe how discovery operates under UPnP (a two-party architecture, where clients issue multicast queries to find services), Jini (a three-party architecture, where clients consult a directory to find services), and SLP (which is a three-party architecture that can adapt to become a two-party architecture). We also describe two mechanisms (polling and notification) used by discovery systems to maintain consistent information among distributed replicas. The architectures, discovery procedures, and consistency maintenance mechanisms described in Section 3 form the basis for scenarios, experiments, and results recounted in later sections.

In Section 4, we introduce selected types of failure that can impede a distributed system and we discuss selected techniques to detect and recover at three layers. At the lowest layer, transport protocols may include detection and recovery mechanisms (e.g., acknowledgments, retransmissions, and exceptions). In the middle layer, discovery protocols typically include some detection and recovery mechanisms (e.g., heartbeats and soft state). At the top layer, applications may take recovery actions in reaction to exceptions raised by transport protocols. Interactions among these detection and recovery techniques can become quite intricate and difficult to understand.

In Section 5, we describe our experiment methodology, consisting of six steps: (1) constructing (simulation) models reflecting structure, behavior, and deployments of selected service discovery systems, (2) incorporating failure models into the simulations (3) devising scenarios and related metrics to quantify robustness and cost, (4) simulating scenarios for selected configurations over a range of failure rates, (5) collecting, analyzing, and plotting data from simulations, and (6) investigating unexpected results and anomalies. In Section 6, we describe the design and results for our experiments: (1) restart after power failure, (2) service acquisition and maintenance impeded by node failures, and consistency maintenance impeded by (3) communication failures and (4) message loss. We report results from these four experiments, which encompass 30 configurations. For each experiment, we explain the scenario and failure model, define metrics, present results, outline findings, and discuss unexpected outcomes. We close in Section 7 with a précis of our findings and contributions.

2. Related work

Emergence of various specifications for service discovery systems, coupled with the anticipated importance of discovery functionality in future distributed systems, has stimulated significant interest in understanding similarities and differences among competing designs. Most existing comparisons focus on architecture, features, and function. A few comparisons also consider programming differences, because most discovery systems are conceived as middleware to support distributed applications. Bettstetter and Renner (2000) compare SLP, Jini, UPnP, and Bluetooth with respect to architecture, function, and features, and consider underlying requirements for programming languages, operating systems, and network protocols. The comparison is expressed using concepts and termino-