Volume 08 Issue 04 Published, November 17, 2004 ISSN 1535-864X

Intel® Technology Journal

Toward The Proactive Enterprise

Towards an Autonomic Framework: Self-Configuring Network Services and Developing Autonomic Applications

A compiled version of all papers from this issue of the Intel Technology Journal can be found at:
http://developer.intel.com/technology/itj/index.htm

Towards an Autonomic Framework: Self-Configuring Network Services and Developing Autonomic Applications

Brian Melcher, Information Services and Technology Group, Intel Corporation
Bradley Mitchell, Intel Communications Group, Intel Corporation

Index words: autonomics, Eclipse, network services, self-configurability, toolkit

ABSTRACT

Autonomic services in the enterprise are becoming more and more of a requirement in all types of networked environments. With an ever-increasing number, type, and complexity of network services available to individual computing systems comes an increasing complexity in establishing and maintaining the configurations of these services. Many network services are already self-configuring today, but this capability is not yet universally available for the broad spectrum of network services or networked environments. Without wide-reaching network services self-configurability, the benefits of reduced management complexity will remain unrealized.

We begin by examining existing network services configuration technologies and identifying incomplete or inconsistent capabilities for dynamically self-configuring these network services. We present for consideration the requirements of an architecture for dynamically self-configuring network services that drives enhanced yet simplified capabilities both to end users and to IT technicians and engineers in a corporate IT environment, as well as to roaming wireless users and home networks.

We then continue by examining the practical implementation of autonomic network service configuration. Purely autonomic systems cannot easily be built today due to a lack of comprehensive framework support. However, substantial pieces of autonomic technology exist in forms suitable for early adoption. Specifically, we focus on the IBM Autonomic Computing Toolkit*, an open set of Java*-class libraries, plug-ins, and tools created for the Eclipse development environment. The IBM Autonomic Computing Toolkit represents a modern framework for enterprise software integration. We examine the toolkit's standard interfaces and data formats to identify its applicability to network services configuration problems. We conclude with a summary of findings and recommendations for prospective enterprise developers and integrators of autonomic toolkits.

INTRODUCTION

Network services discovery is a significant aspect of today's network infrastructure. In today's network environments, network services configuration information is dispersed among a variety of information repositories, and the relationship between the storage and consumption of that configuration information is often managed through programs, procedures, or protocols specially developed for the specific environment at hand. As the number of end-user systems joining the network grows and the number and variety of network services grows, the complexity of and demand for a solution to manage this relationship between the configuration information and its consumption likewise grows.

The Need for Self-Configurability

Automating the management of network services configurations is becoming ever more a requirement across the entire spectrum of computing networks. The large networks in a corporate Information Technology (IT) or Internet Service Provider (ISP) environment are becoming too complex to manage configurations on an individual system-by-system basis. The sheer number of systems requires an increasing number of staff to manage them; proportional growth of staff-to-systems is undesirable, if not outright impractical. Additionally, individual hands-on system management increases the likelihood of errors being introduced into the environment. Mobile computing puts an increasing demand on reconfigurability as the system roams between

* Other brands and names are the property of their respective owners.

connectivity points. Home networks are becoming more commonplace as the number of computers in the home increases; yet, those same home networks must be easy to maintain in order for the typical home computing consumer to be able to manage them. In each of these environments, the end-user system has varied degrees of manageability control–that is, those who manage and control the network–and network services configurations may or may not have management control over the end-user system.

Self-configurability with respect to network services is the capability of a system to configure its own network-based services and applications in response to the needs of the user and the environment the system finds itself in. As the needs of the user change or the environment the system is in changes, that end-user system would recognize the change, understand the impact, and respond by reconfiguring itself accordingly. Flexibility of location, a wide range of administrative control, and the need for varied rates of dispersal of changed configuration information across the environment, all complicate the task of autonomic network services configuration behavior.

Structure of this Paper

In this paper we first review current-day network services configuration technology, identifying existing capabilities as well as incomplete or inconsistent capabilities in autonomic network services configuration behavior. We present for consideration an architecture that supports autonomic configuration of network services for a plethora of computing environments: a corporate IT environment, a mobile user hopping from one access point to another, and a home computing network whether or not it is connected to the Internet.

Second, we demonstrate development of autonomic applications using the IBM Autonomic Computing Toolkit in the Eclipse development environment. Created by the Eclipse Foundation, a consortium backed by Intel, IBM, and others, Eclipse provides a modern, extensible environment for software development. This toolkit specifically supports general-purpose problem determination, installation, and user access features that correspond to network service configuration tasks at a very high level. We examine details of the toolkit to develop practical recommendations for utilizing it in the IT, roaming user, and home network environment. Most recommendations apply generally to various other classes of autonomic problems.

NETWORK SERVICES CONSUMPTION PROBLEMS

Autonomic Network Services Solution Goals

We discuss the goals, considerations, and demands of self-configurability of network services in four types of network environments:

- A corporate IT environment.
- A wireless network provider with roaming mobiles.
- An Internet-connected home network.
- An isolated home Local Area Network (LAN).

We consider these four environments to span the range of network environments: these environments would push the limits of any solution for self-configurability of network services, and, therefore, any solution that applies to these networks would apply to network solutions in between.

In IT environments, one goal of self-configuring network services is to move away from individual system configuration management to policy management. This approach brings a higher level of abstraction to management by introducing a policy from which the configuration is derived, allowing the automation components of the infrastructure to apply these derived configurations to the individual systems across the environment. Policy management frees IT personnel from the role of sustain, maintain, and fix. Additionally, with the system itself deriving the configuration from a set of policies, this eliminates the human-error factor when configuring by hand.

More and more users are mobile, jumping from a wireless connectivity point to another as they travel from airport, to coffee shop, hotel, on-site at another corporation, or across a college campus. Eventually network services and connectivity for mobile computing should become as seamless as cell-phone usage going from one cell coverage area to another, or from one provider's region to another. In any mobile environment, user systems will come and go and network services will also come and go. As such, there can be no preconceptions or expectations by either party: the network and available services are foreign to the end-user system, and the end-user system is likewise foreign to the network and infrastructure.

In the home environment, multiple computers are becoming more commonplace. Home systems may be connected to just each other and otherwise isolated, or connected to the Internet. Either way, in this home environment usually the set of systems and services does not rapidly change, but there should be no expectations of establishing or sustaining a complex solution.

Usage Areas

The IT Environment

A corporate IT environment today consists of a mix of fixed and mobile, both wired and wireless, systems where IT personnel have end-to-end management control–that is, control of the configuration information repositories (the *back-end* systems), the system consuming said information (the *front-end* or *end-user* systems), and all the infrastructure in between. An additional characteristic is the technical expertise by the IT personnel for the entire breadth of the environment: the back-end, front-end, and infrastructure; and the standard images, sometimes referred to as *gold* images, deployed across the environment.

With this end-to-end system management (or, *tightly coupled* management) programs, procedures, and protocols specific to the environment can be put in place to manage the dispersal and consumption of any network services self-configuration information. In an IT environment where central administration of network services configuration is needed or desired, this manner of a gold image with pre-configuration or specialty-developed self-configuration utilities can more readily be introduced than it could be in other environments to completely address the problem. As a new service is established, a companion utility is developed and deployed in parallel with the service. Even with mobile users within the IT environment, such environment-specific utilities can still be employed to achieve network services self-configurability. Being mobile may require a more complex solution, but can be achieved.

In an IT environment, uses for network services self-configurability are varied: they range from day-to-day usage as mobile systems roam about the corporation (such as between buildings, sites, or sub-domains), to managing introduction of new services and retirement of existing services, to crises event management with distribution of anything from a security patch or anti-virus signatures data file, to a Web proxy configuration or mail relay blacklist across the entire environment.

Any solution applicable for an IT environment can require control of the end-user and/or back-end systems. It must also allow for policy management to drive individual end-user system configuration, and must allow for rapid policy and configuration changes to be introduced and dispersed across the environment.

The Roaming User

A roaming wireless user is in contrast to the tightly coupled end-user system in the IT environment. Here, the end-user system may not be under any degree of control by those administrating the infrastructure. Environment-specific utilities, as in the IT environment, could be introduced to provide self-configurability for that specific wireless neighborhood. However, it cannot be expected that this utility would be installed by a roaming wireless consumer. There is the initial question of trust and security of that utility: why would you trust a utility made available in a wireless hotspot in the middle of some airport or hotel. One question is whether it is a legitimate service or someone masquerading as a legitimate service in order to intercept the network connection. Further, and more applicable to the autonomic network services configuration problem here, as the user roams from one provider to another, the number of environment-specific utilities required on that mobile system increases, increasing the likelihood of conflicts or system instability.

With a roaming wireless user, the system being disconnected from the network must be considered. Any network discovery and self-configuration methods must recognize this disconnected state and allow for a stable and functional (as much as possible) computing system. Any solution must not require any degree of control on the end-user system, but can require establishment of back-end systems. A higher level policy management solution will greatly facilitate numerous systems coming and going, but the demand for an event-driven "push" of new configurations is less than with an IT environment.

Home Networks–Connected and Isolated

In the home network, both Internet-connected and an isolated LAN, there is not necessarily the IT know-how to create an environment-specific utility or even to deploy and configure a well-known solution. With computers now becoming ubiquitous in the home environment, the demand for autonomic network services configuration increases. In addition to the technical know-how, if a solution requires additional infrastructure it increases the physical cost of that solution. As such, an optimal solution would not require any additional infrastructure, services, or configuration. The solution should be transparent yet automatic, as if working magically with the existing collection of end-user class systems.

With the connected home network, there is a connection to the Internet at large through a service provider. The solution space for the connected home network is distinctive from the isolated home network in that it could be reliant upon the ISP to provide a certain level of autonomic network services configuration. The requirements for technical know-how and in-home infrastructure remain the same.

Solution Space Boundaries and Conditions

From the network usage areas described above we can create a series of checks by which we can evaluate the appropriateness of possible solutions:

- *No back-end infrastructure*. As indicated for the home network environment.
- *No technical know-how required*. Also as indicated for the home network environment.
- *No tightly coupled management*. For the wireless ISP and home networks.
- *Dynamic addition/removal of end-user systems*. For the IT environment and wireless service provider.
- *Dynamic addition/removal/change of services*. For the IT environment.
- *Universality*. Implemented by many OS and network equipment vendors.
- *Rapid change deployment*. For the IT environment.

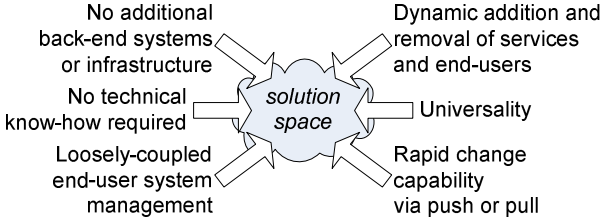

Figure 1: Network environment solution conditions

A wide range of network services should be considered for this problem of providing autonomic network services, for if the broadest scope is considered, a more robust solution will be forthcoming. These network services include hostname resolution, proxy (for FTP, Web, telnet), logging, Network Time Protocol (NTP), mail storage, mail relay, nearest printer, nearest patch distribution, and user authentication servers. Typically, what this configuration entails is a reference to another system, and then once the end-user system has been pointed to the service server, the end-user and server systems can communicate directly between each other.

The additional checks for varieties of network services should be as follows:

- *Nearest proxy server*. Can the nearest firewall/proxy server for FTP, Web and/or telnet be determined? This may be a list of systems, not just a single system, with each equally capable of providing service, but only one utilized for a transaction.
- *Nearest printer*. Can the nearest printer be determined? This is different than the *nearest proxy* check, as *nearest printer* is based upon geographic location whereas *nearest proxy* is based upon network location.
- *NTP server*. This is similar to the *nearest proxy server* and *mail relay server* checks: a list of equally capable systems is provided. This check differs in that all list members may be used for each transaction; however, only one is required. In the previous checks, only one system is used per transition.
- *Mail relay server*. Can the nearest mail relay server be located, a similar check as *nearest proxy*?
- *Mail server*. Can the mail server appropriate for this user be located? This condition is very different than the *mail relay server* check. A mail relay server is a system that receives and forwards e-mail (perhaps scanning for viruses or checking blacklists in the process); a mail server is the repository of a users e-mail inbox. Since a mail server provides a stateful service it requires locating the specific server with the appropriate state for the user, in this case, the state being the mail repository itself. The *mail relay* check differs in that it is stateless and readily interchangeable between peers.
- *Logging server*. This is much the same as the stateful *mail server* check in that a system may need to consistently report to the same logging server.
- *Nearest patch distribution server.* An authenticated patch distribution server must be located to provide quick and timely updates, particularly important in a tightly coupled environment.
- *Nearest user authentication server.* This is also for a tightly coupled environment, but for a stateful service in that the same server is employed time and again.

NETWORK SERVICES CONSUMPTION–STANDARDS, SOLUTIONS, AND CHALLENGES

Solution Methods

Self-configuration functionality can be implemented in a variety of methods. Functionality can be implemented locally, employing locally maintained policies, protocols, or programs wholly on the end-user system. Or, service configurations can be learned from the environment itself. These are the policies and protocols established in the infrastructure to facilitate the end-user system in learning, that is, updating itself, about the environment, both upon initial introduction to the environment as well as when the environment changes. Also, learning from the environment could come from peer systems already in the environment and successfully configured to operate within it. Learning from the environment can also stem from a global application of network resource configurations, specifications, policies, programs, procedures and/or protocols.

Intentional Hostname Naming Conventions

A long-standing approach to providing connectivity to network services, a pseudo-autoconfiguration method if you will, is the use of an intentional hostname naming convention. Services are provided by a system with a well-known hostname; for example, a mail relay is "mailhost" and a name service server is "ns."

This convention requires initial configuration on the end-user system to use this well-known hostname, but once configured, the act of reconfiguring network services occurs silently. And in actuality, on the end-user system there is no reconfiguration: the name resolution process provides the reconfiguration of the network service. Such a solution can be directly employed in IT and ISP environments with a managed back-end infrastructure.

With this convention the "short" hostname is used, not the Fully Qualified Domain Name (FQDN). For an IT environment using a gold image method of system installation, when that image is deployed across multiple domains, the service server name is resolved to the server within the local domain just like any other short hostname. A roaming mobile user would see the same effect when going from one service provider to another, providing there is consistency in the naming convention between service providers.

There are several shortcomings to this method. First, this convention is optional and therefore consistency between wireless ISPs cannot be guaranteed. Second, this does not provide an automated solution for a home network: it can facilitate a solution for an Internet-connected home network given the ISP's participation, but to the LAN internal to the home, this provides no solution.

Due to these shortcomings, this method would work best only in a tightly coupled environment such as an IT environment. But even here there are further shortcomings: this method doesn't meet several of our checks. Geographic information isn't available to address *nearest printer*. Further, *mail server* is not addressed due to how name resolution typically occurs: if there are multiple physical systems with the same hostname, name resolution may load-balance requests in a manner that returns the address of a different physical system for each name lookup query, rendering it useless for any stateful service that is distributed across a server farm.

DNS and DHCP

Some institution of autonomic network services comes from Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP), both long-standing network name-service protocols. DNS provides hostname resolution through a distributed hierarchy of DNS servers. DHCP enables systems to dynamically configure their own IP address under the conditions and policies allowed by a DHCP server.

DNS

Use of DNS aliases is an instantiation of an intentional hostname naming convention and facilitates service portability. On the back-end the service proper can be relocated from one machine to another and, with the use of a DNS CNAME, end users are directed to the new service location as DNS is updated. This has commonly been done with hostnames such as "mail" and "www." Proposals recommend additional records to the DNS tables–beyond MX (mail exchange) to WKS (well-known services, which include FTP and www). However, this is still not seen as a long-term solution by even the Request for Comments (RFC) authors [3]. For the purposes of autonomic network services, we only need to communicate to the end-user systems within the immediate network. Additional DNS records would express this information externally as well, which may be an unacceptable information leakage; however, this can be remedied with a separate DNS zone available only internally. This internal zone increases the technical complexity and infrastructure required for the solution, but this separation of DNS zones is usually a requirement for IT and ISP environments segmented with firewalls from the Internet at large and should not limit its consideration as a solution for autonomic network services.

DHCP

DHCP provides dynamic network services through a software agent on the client and an infrastructure to service requests. DHCP is a well-established protocol and both client and server services are available on most operating systems today. Given this wide availability, DHCP could be well positioned to become the foundation for an autonomic network services configuration solution. Extensions to DHCP provide for additional network services, providing the DHCP client with DNS resolvers and a DNS search path [7]. Moreover, DHCP itself is an extensible protocol that could be utilized to provide pointers to additional network services configuration information.

While DHCP is an extensible protocol, these extensions only define packet content; Application Programming Interfaces (APIs) that utilize the additional DHCP-provided information are left undefined. This is a key reason DHCP is not a solution to the problem of autonomic network services. While this lack of APIs could be accommodated if the application itself implemented (at least partial) DHCP capability, this only transfers the problem and doesn't really resolve it. This method of introducing DHCP capability directly into the applications that utilize this DHCP-provided information

is a part of the environment-specific utilities discussed earlier, suitable only for tightly coupled environments.

Many of our solution space boundary checks can be met through careful policy construction, but this itself doesn't meet the *no technical know-how* check. With regard to *no back-end infrastructure* with the connected home network, DHCP is available in firewall/gateway appliances designed specifically for connecting up home networks, meaning this is not much of a limitation. However, the *dynamic addition and removal of services* check is not met with DHCP: an administrator must change the DHCP configuration policy when a service is added or removed.

Network Information Service (NIS/NIS+)

Network Information Service (NIS), and its security-enhanced follow-up NIS+, provide for database-like queries about information services. An NIS domain consists of a set of tables, and within those tables is a keyword-to-answer mapping. The NIS tables and mappings must be preconfigured, thus failing the *no technical know-how* check. Also, NIS does not lend itself to a dynamic environment of services coming and going, failing another check. Due to its one-to-one mapping nature, it also fails the *nearest printer* check, and it suffers the *mail server* problem in the same way as DNS. Further limiting NIS is that it is principally a UNIX*- and Linux*-only solution, failing the *universality* check.

NIS does have a unique characteristic. In DHCP, APIs do not exist to access new information types, but the APIs in NIS are generic enough to handle new information services: only the new table needs to be created.

Directory Service (LDAP)

Lightweight Directory Access Protocol (LDAP) [5] is a protocol for locating resources on a network through a hierarchical directory repository. Commonly used for authentication and as a replacement for NIS, LDAP and its information model are extensible, and as such can be considered for autonomic network services. As the information model is extensible, the type of information which can be included in the directory service can be used to meet practically all of the network services checks: network locality-based services of *nearest proxy server* and *mail relay server*; stateful services like *mail server* and *logging server*; and even geographical locality-based services of *nearest printer,* if the directory service information model contains the right data. LDAP includes APIs, allowing applications to interface with the information in the dynamic directory service. However, like DHCP and NIS, LDAP requires an infrastructure preconfigured to service requests, failing the *no technical know-how* and *no infrastructure* checks. Unlike NIS, LDAP can provide for dynamic responses and provides for service announcements, addressing the *mail server* and *logging server* checks. However, LDAP is still limited in that it does not provide for dynamic, spontaneous discovery, and as such does not lend itself as a solution.

Service Location Protocol (SLP)

Service Location Protocol (SLP) is a protocol that provides a framework for discovery and selection of network services, eliminating the need for many static network services configurations for network-based applications [4]. SLP is intended for an enterprise network with shared services (as opposed to a global network) fitting into the IT, ISP, and home environments.

With this protocol, end-user systems attempt to locate a service. The services themselves, as they come on-line, advertise their services and can communicate directly with end users or with a central directory agent. In this fashion it meets the *dynamic addition and removal of services* and *dynamic addition and removal of end-user systems* checks. APIs allow access to SLP data, and for non-SLP capable applications, SLP proxies can be employed. Like LDAP, SLP can provide dynamic responses and address many of the network services checks, again provided the directory agent is established with the right data.

DHCP options [6] and use of DNS Service Location Resource (SRV) records [10] have been proposed that would facilitate the discovery of the SLP directory agent, which would facilitate the use of SLP and increase its autonomic capabilities by reducing the *not tightly coupled* hurdle. SLP is a promising protocol and has been implemented in products from several vendors [15]. It does not, however, yet meet the *universality* check.

Limitations of Today's Solutions

Intentional hostname naming has existed for many years, and if it was a solution capable of providing autonomic network services for the broad spectrum of network services and networked environments, this would not even be a topic of continued research. The creation of DNS, and more specifically the MX and later the SRV records, facilitated service availability through intentional hostname naming. DHCP introduced dynamic hostname naming of end-user systems, but it is quite limited in facilitating network service discovery. DHCP, NIS, and LDAP all require an infrastructure and technical know-how to implement and sustain each services' own configuration. SLP also requires an infrastructure, but its implementation demands on technical know-how are less than those with LDAP; plus the demand can be reduced if

* Other brands and names are the property of their respective owners.

services register themselves, reducing the burden of activating the SLP Service Agent. SLP is not widely available: it is too soon to determine if an SLP-based solution is viable for autonomic network services configuration.

All these methods have limitations in one form or another, meaning that with any one implementation of these existing technologies and their implementation, a single universal architecture for autonomic network services has not yet arrived. However, a combined used of these protocols can provide a comprehensive solution. For instance, an end-user system using DHCP could acquire DNS name server information, and then with the information from DNS SRV records locate a directory service to subsequently locate configuration information for network services. Other combinations of protocols can also be applied to bridge the gaps that the participant applications have when standalone. These solutions certainly work, and, considerations of availability and reliability aside, they are complex architectures and, accordingly, do not meet the *no technical know-how* and *no back-end infrastructure* checks. Therefore, these protocol combinations are not beneficial to the unconnected home network. Further, even in the technical know-how rich IT and ISP environments, these solutions can more readily be employed where one protocol is built upon another, layer upon existing layer over time. To apply them to a new environment would incur high installation and maintenance costs.

TOWARDS AUTONOMIC NETWORK SERVICES

A Solution Architecture

To meet the *no back-end infrastructure* check suggests a solution in the direction of a peer-to-peer self-discovery mechanism. However, an IT environment requires a high-level policy control and rapid change capability, suggesting a client/server architecture. These two vectors seem contradictory, even mutually exclusive.

Consider a peer-to-peer self-discovery mechanism with a priority schema. By being peer-to-peer, there is no need for a back-end infrastructure, which satisfies the needs of a home environment. Then in the tightly coupled IT environment, we introduce a system we call a *services broker*, upon which policies are managed directly by IT personnel. In this priority schema, the services broker has a higher priority than the class of end-user systems: the services broker will "shout" while end-user systems "whisper." If so desired, the priority of end-user systems could be set to nearly zero (with priority defined as the higher the number, the higher the priority). Or, it could be set to zero (or "off"), in essence transitioning the original peer-to-peer architecture into a traditional client/server architecture–but all within the same solution implementation.

This is not, however, a traditional client/server architecture. Due to the priority-schema peer-to-peer structure this architecture has an ability for dynamics, fault tolerance, and self-healing built right into it. The outage of a services broker can be tolerated by end-user systems being able to utilize a configuration from any one of the collection of services brokers. In an "end user at zero" model, the end-user system cannot reference any of its peers, necessitating that end-user systems seek out other services brokers in the environment. In an "end user at near-zero" model, in an outage of all services brokers, any end-user system can listen to the "whisper" of another peer and discover what limited services may still be available.

In either a "zero" or "near-zero" end-user model, we gain an additional key benefit: the ability to rapidly introduce a new service or service configuration, a crucial feature to address crisis and denial-of-service situations. One can "seed" configuration solutions simply by introducing a system with a newer configuration (the "fix") into the environment. Even in a total services broker outage, an IT administrator can configure the new service on his or her own end-user class system and temporarily elevate the priority of that configuration, spreading the new configuration by "whispering" louder than other end-user systems.

Such a priority-based schema could be implemented onto an LDAP or SLP infrastructure, even maintaining compatibility with existing infrastructure deployments. SLP is considered a state-of-the-art directory services protocol and a strong choice for future management of network services [16], but not likely to displace existing LDAP infrastructures.

With these considerations we have presented the following as requirements for an architecture for autonomic network services:

- Priority-based peer-to-peer structure, allowing for a traditional peer-to-peer architecture that can transition into a client/server architecture.
- Service consumer self-discoverability of service providers and services brokers.
- Rapid introduction of new services and services configurations.

SOFTWARE TOOLKITS FOR AUTONOMIC NETWORK SERVICES

To fully address issues of autonomic network services, an autonomic framework is necessary. To reach this goal, "we need to rethink the structure of the system and the application software (and the tools that help build them)" [12].

While some features of autonomic services are best implemented in computer hardware, typical autonomic solutions for the enterprise also demand software support. Administration of traditional client/server and n-tier systems can benefit substantially from having autonomic services implemented in the operating system or higher-level software applications. A fully autonomic-aware Web database system, for example, should be capable of identifying and repairing certain classes of database integrity errors to minimize impact to future user sessions.

Unfortunately, to make enterprise systems fully autonomic requires significant effort. Modern enterprise architectures tend to feature numerous applications and components supplied by different vendors. These systems increasingly distribute the software, hardware, and data storage functions geographically across networks. Additionally, individual hardware and software components have grown exponentially in complexity over the past few years as processing speeds and development tools have improved.

Software toolkits deliver required autonomic capabilities utilizing standard APIs, data formats, and network protocols. A toolkit provides reusable, standards-based software to reduce these efforts of software development and integration. Examples of functions ideally supported in an autonomic software toolkit include the following:

- Event logging APIs and data formats.
- Issue alerting mechanisms.
- Network discovery mechanisms.
- Data migration and conversion utilities.
- Interfaces for extensibility and integration with third-party software.

In the following sections we explore functionalities of the IBM Autonomic Computing Toolkit. Though not a complete enterprise solution, this toolkit provides a practical framework and reference implementation for incorporating autonomic capabilities into software systems.

THE IBM AUTONOMIC COMPUTING TOOLKIT

The IBM Autonomic Computing Toolkit comprises class libraries, plug-ins, and tools for the Eclipse development environment. To support both development and execution, the toolkit depends on specific versions of the Java Runtime Environment* (JRE). In the following sections we describe each software component of the toolkit.

General Concepts

The toolkit is based on the dual concepts of Managed Resources and Autonomic Managers. Managed Resources can represent end-user computers, other network nodes, or individual software components running on a device. Resources monitor their environment and are capable of detecting and reporting events to an Autonomic Manager. Managed Resources also take administrative actions in response to Autonomic Manager requests.

Autonomic Managers oversee the operation of Managed Resources. Managers implement administrative policy and business logic to facilitate and coordinate optimal operation of resources. All direct communication between managers and resources is handled via Java interfaces. Although the toolkit is architected to accommodate distributed managers and resources, the current implementation of manageability interface APIs supports only limited forms of communication on the local device.

Common Base Events

The IBM Autonomic Computing Toolkit defines a standard data format called the Common Base Event. Common Base Events are Extensible Markup Language (XML) structures ("blobs") that define a standard data format for communicating events. Common Base Events provide a convenient mechanism to centralize and correlate events from disparate applications.

Each instance of a Common Base Event may define the software component that generated the event, a location of the event (such as short or fully qualified hostnames), the time of the event, and a description of the situation or scenario leading up to the event.

Generic Log Adapter

One way to generate Common Base Events is through the Generic Log Adapter (GLA). Runtime support in Autonomic Managers utilizes the GLA to convert data from existing log files of legacy applications to the Common Base Event format. For each type of log file to

* Other brands and names are the property of their respective owners.

be supported by the GLA, a developer must implement parsing, formatting, output, and other objects tied together by a configuration file. The toolkit provides an Eclipse plug-in called the Adapter Rule Editor to simplify creation of these configuration files.

Log/Trace Analyzer

The Log/Trace Analyzer is a simple implementation of an Autonomic Manager. Administrators use this analysis tool to graphically view and correlate event log files.

Associated with the Log/Trace Analyzer in the toolkit is support for "symptom databases." These databases consist of XML files that encode possible resolution actions for Common Base Events. A toolkit plug-in allows administrators to build symptom databases according to their policies.

Resource Model Builder

The Resource Model Builder generates data models of monitored resources. Resource models define event types, polling intervals, thresholds, and actions to take when thresholds are crossed. These models employ industry-standard Common Information Model (CIM) classes for holding resource properties. The Resource Model Builder also supports CIM and Windows Management Instrumentation (WMI) standard Managed Object Format (MOF) files in addition to custom scripts.

Automated Management Engine

The Automated Management Engine (AME) hosts deployed resource models. This engine executes resource model scripts within a control loop. It also stores operational data in an embedded local database. AME contains a CIM Object Manager (CIMOM) extensible via Engine APIs.

Integrated Solutions Console

The Integrated Solutions Console (ISC) is a Web-based console user interface. ISC implements centralized management of autonomic capabilities using a WebSphere Application Server as the supporting infrastructure. An Eclipse plug-in supports development of add-on console components. The console supports user interaction in environments where full enterprise management console integration is not in place.

APPLICATIONS IN NETWORK SERVICE CONFIGURATION

In the following sections, we describe several approaches for integrating the IBM Autonomic Computing Toolkit into existing network service configuration technologies to address the usage models described earlier.

Toolkit Integration for Client-Server Environments

At a conceptual level, the IBM Autonomic Computing Toolkit can be integrated in a straightforward fashion with IT client/server network architectures. Each network device can be modeled as a managed resource, and dedicated server or gateway nodes can be configured as Autonomic Managers (clustered as necessary). CIM/WMI support allows easier integration with legacy enterprise management data stores. Specific versions of Java Virtual Machines (JVM*s) can be targeted for enterprise deployment across multiple platforms as needed.

However, the toolkit's current implementation prohibits this design approach, as manager-resource interfaces enable only local (intra-device) communication. Until supported extensions for inter-device communication are available in future versions of the toolkit, workarounds or extensions for the IT environment must be designed. One possible workaround involves adding remote monitoring support to Managed Resources. This approach entails extending the current toolkit architecture with a new "Remote Managed Resource" component as shown in Figure 2.

* Other brands and names are the property of their respective owners.

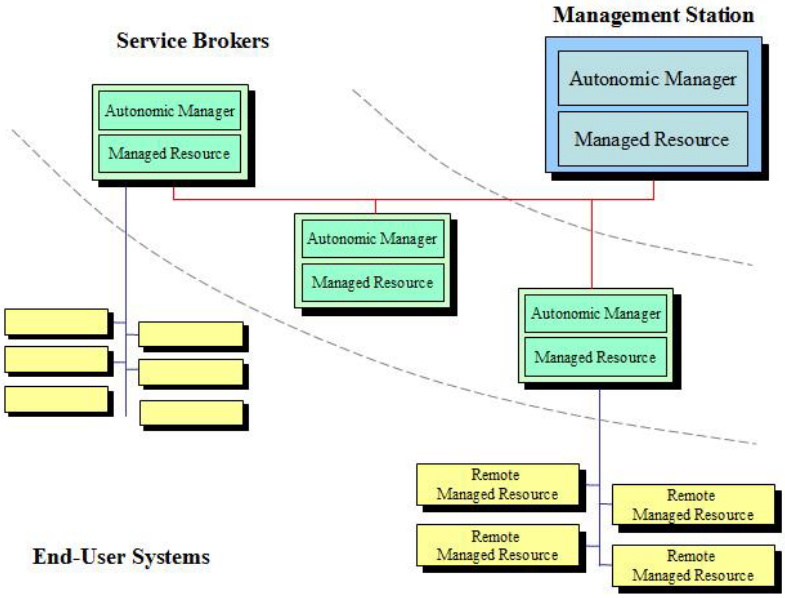

Figure 2: Extended Eclipse autonomic toolkit architecture

In this proposed architecture, Remote Managed Resources are lightweight implementations of toolkit managed resources. These remote resources incorporate the following functions:

- Monitoring and control capability for software and/or hardware on the local device.
- Support for discovery of toolkit Managed Resources and Autonomic Managers.
- Retrieval of control instructions remotely from managers.

To interact with ordinary toolkit resources, remote resources initiate all requests to parent nodes, called toolkit Service Brokers. Toolkit service brokers are simply Autonomic Managers with extensions to support remote resources. They represent a single-service implementation of the general-purpose "services broker" functionality discussed earlier. To pass event logs or alerts to a service broker, for example, remote resources push Common Base Events and other data up to a previously discovered broker. Likewise, to obtain configurations and instructions, a remote resource requests and pulls data from the manager.

These one-way discovery and communication processes can be implemented in IT environments through XML-RPC (Remote Procedure Call) or similar HTTP-oriented protocols. Protocol conventions for decoding configurations and actions must be established between managers and resources. This architecture enables management of remote resources by establishing data-driven monitoring and configuration procedures.

Toolkit Integration for Peer-to-Peer Environments

Attempting to fit the IBM Autonomic Computing Toolkit into a peer-to-peer architecture for network services poses several new challenges. Peer-to-peer architectures utilizing the toolkit demand that any device be capable of taking on Autonomic Manager responsibilities when needed. It follows that each peer must be granted access to the resource models, symptom databases, and event logs deployed in that environment. Besides the increased configuration burden, this architecture also places increased computational demands on nodes that may not be fully equipped to handle the additional overhead.

One method for implementing peer-to-peer toolkit support entails periodic synchronization of all Autonomic Managers. This approach requires a method to verify that each Autonomic Manager configuration is up-to-date as well as a mechanism to trigger propagation of the manager's policy data when needed. The toolkit's architecture must be extended to provide such support. As noted earlier, however, the resource overhead incurred by running an Autonomic Manager may exceed the capability of some peer-to-peer devices. Alternatively, these devices can be configured as lightweight Remote Managed Resources.

Toolkit Integration with Network Services

As described above, developers can, with effort, integrate the IBM Autonomic Computing Toolkit into various client/server and peer-to-peer networks. In IT environments, Autonomic Manager functionality fits

logically as an add-on to existing DHCP, DNS, LDAP and other server platforms. Deploying Remote or full Managed Resource technology to the client-side likewise can be centrally administered using SLP or another available discovery protocol to tie the system together.

On home networks, the difficulty of enabling autonomic services increases. As home "routers" and "entertainment gateways" continue to expand in popularity, these devices become the natural host for Autonomic Manager functions. However, these devices do not generally support a "push" model for deployment, and many home administrators will refuse to install toolkit components on their managed devices manually. Additionally, networked printers, game consoles, and low-end home computers will generally not meet minimum system requirements to serve as toolkit Managed Resources. Given suitable policy setup on the router, the toolkit offers a promising solution for the myriad configuration problems that afflict home networks today. For example, an Autonomic Manager could detect and heal mis-configured workgroup names, wireless encryption settings, and other security settings in addition to basic DHCP setup.

Mobile and roaming wireless users also benefit from features of the IBM Autonomic Computing Toolkit. By pre-installing a laptop or other mobile device with Remote Managed Resource components, the device can discover Autonomic Managers as needed to update DHCP and other network service configurations. During times when the mobile device is not connected to the network, the system can rely on configuration/policy information cached in the local Autonomic Manager. Wireless Internet Service Providers (WISPs) may also leverage the toolkit to build universal sign-on and automated billing services.

AUTONOMIC COMPUTING AND ENTERPRISE MANAGEMENT FRAMEWORKS

Autonomic functions represent the latest step in the natural evolution of enterprise system, network, and storage management capabilities. In the 1980s, Simple Network Management Protocol (SNMP) introduced basic monitoring and control functions for enterprise IP networks. SNMP supports basic configuration, logging, and alerting mechanisms. Unfortunately, SNMP did not define standard resource models for managed devices, hindering its adoption in cross-platform networks.

In the 1990s, Intel, as part of the Desktop/Distributed Management Task Force (DMTF), actively developed and promoted the Desktop Management Interface (DMI) standard. DMI offers a more sophisticated monitoring, configuration, and control framework that enhances support for end-user usage models. Specifically, DMI includes predefined objects for modeling PC client resources and system events.

Subsequently, the DMTF also developed specifications for CIM, which eases the burden of software development associated with DMI. WMI in turn incorporates CIM into standard management software for Windows* platforms. Finally, in the late 1990s, Intel and other companies jointly developed the Intelligent Platform Management Interface (IPMI). Whereas SNMP, DMI, and CIM/WMI function on top of an operating system and a network protocol stack, IPMI provides management capability at a lower level, monitoring platform hardware attributes like voltages, fan speeds, and temperatures, and working independently from the operating system.

Autonomic technologies complement each of these other management standards. Specifically, the IBM Autonomic Computing Toolkit leverages the CIM Object Model and can work with MOF files as indicated earlier. Through the Generic Log Adapter, the toolkit also can aggregate data generated by other software components tied to these standards. In general, autonomic services operate at the next higher level of the management solution stack. It follows that the ability of autonomic solutions to support self-configuring and self-healing depends on the availability of these standard management technologies as well as the extent of instrumentation deployed.

IBM AUTONOMIC COMPUTING TOOLKIT SUMMARY

The IBM Autonomic Computing Toolkit enables developers to add self-configuring and other autonomic capabilities to their software. Capabilities center on the ability to model and monitor resources, implement policies for configuring and controlling those resources, and manage log data using standard cross-application and cross-platform mechanisms. The toolkit attempts to leverage and retain compatibility with existing enterprise management standards. It is a framework best utilized as a reference implementation by early adopters to assess the impact of autonomic technology on their environment.

SUMMARY

In this paper we presented existing network services configuration technologies, identifying current capabilities and shortcomings for dynamically self-configuring network services. We proposed requirements for additions to these existing technologies to address these shortcomings to provide a fully capable autonomic

* Other brands and names are the property of their respective owners.

network service for several types of networked environments. We also described the IBM Autonomic Computing Toolkit, an application development suite that provides software developers with a technology to develop autonomic applications, including dynamically self-configuring network services.

ACKNOWLEDGMENTS

We thank Shu-min Chang, Susan Harris, Marian Lacey, Ray Mendonsa, and Tim Verrall for their valuable technical and editorial contributions to this paper.

REFERENCES

[1] Gulbrandsen, A. and Vixie, P., "A DNS RR for Specifying the Location of Services (DNS SRV)," *IETF* RFC 2052*, Oct 1996.

[2] Droms, R., "Dynamic Host Configuration Protocol," *IETF* RFC 2131*, March 1997.

[3] Hamilton, M. and Wright, R., "Use of DNS Aliases for Network Services," *IETF* RFC 2219*, Oct 1997.

[4] Viezades, J., Guttman, E., Perkins, C., and Kaplan, S., "Service Location Protocol," *IETF* RFC 2165*, June 1997.

[5] Wahl, M., Howes, T., Kille, S., "Lightweight Directory Access Protocol (v3)," *IETF* RFC 2251*, Dec. 1997.

[6] Perkins, C. and Guttman, E., "DHCP Options for Service Location Protocol," *IETF* RFC 2610*, June 1999.

[7] Smith, C., "The Name Service Search Option for DHCP," *IETF* RFC 2937*, Sept. 2000.

[8] T'Joens, Y., Hublet, C., and De Schrijver, P., "DHCP Reconfigure Extension," *IETF* RFC 3203*, Dec. 2001.

[9] Klensin, J., "Role of the Domain Name System (DNS)," *IETF* RFC 3467*, Feb. 2003.

[10] Zhao, W., Schulzrinne, H., Guttman, E., Bisdikian, C., and Jerome, W., "Remote Service Discovery in the Service Location Protocol (SLP)," *IETF* RFC 3832*, July 2004.

[11] Ganek, A.G. and Corbi, T.A., "The dawning of the autonomic computing era," *IBM Systems Journal*, vol. 42, no. 1, 2003, pp. 5-19*.

[12] Bantz, D.F., Bisdikian, C., Challener, D., Karidis, J.P., Mastrianni, S., Mohindra, A., Shea, D.G., and Vanover, M., "Autonomic personal computation," *IBM Systems Journal*, vol. 42, no.1, 2003, pp. 165-176*.

[13] Balakrishnan, H. "Resource Discovery Using an Intentional Naming System," Nov. 1999, http://nms.lcs.mit.edu/talks/stanford-netseminar/*.

[14] Konstantinou, A. V., Florissi, D., and Yemini, Y, "Towards Self-Configuring Networks," *DARPA Active Networks Conference and Exposition*, May 2002, http://www1.cs.columbia.edu/dcc/nestor/nestor-dance-2002.pdf*.

[15] Alex, H., Kumar, M., and Shirazi, B., "Service Discovery in Wireless and Mobile Networks," http://crewman.uta.edu/psi/download/Mohan_Shirazi/ServiceDiscovery.pdf*.

[16] Perkins, C., "SLP White Paper Topic," May 1997, http://playground.sun.com/srvloc/slp_white_paper.html*.

[17] Jacob, B., Lanyon-Hogg, R., Nadgir, D. and Yassin, A., *A Practical Guide to the IBM Autonomic Computing Toolkit*, IBM International Technical Support Organization, http://www.redbooks.ibm.com/redbooks/SG246635/*.

AUTHORS' BIOGRAPHIES

Brian Melcher, GCUX, RHCE, is a senior UNIX/Linux security engineer in Intel's Information Services and Technology Division. His technical interests include UNIX and Linux security and operating system internals. Brian received a Master's degree from the University of Arizona and a Bachelor's degree from the University of Illinois at Urbana-Champaign. His e-mail is brian.a.melcher at intel.com.

Bradley Mitchell is a validation manager and senior software engineer in Intel's Flash Products Group. His technical interests include automation software, high-performance computing, and wireless networking. He holds one patent with two additional patents pending. Bradley received a Master's degree from the University of Illinois at Urbana-Champaign and a Bachelor's degree from M.I.T. His e-mail is bradley.mitchell at intel.com.

For further information visit:

developer.intel.com/technology/itj/index.htm