Distributed Event Correlation and Self-Managed Systems

Jean-Philippe Martin-Flatin
CERN, IT Dept.
1211 Geneva 23, Switzerland
jp.martin-flatin@ieee.org
http://cern.ch/jpmf/

Abstract

The systems that currently provide Internet users with sophisticated e-business services struggle to offer high availability guarantees while being managed as regular systems, because management architectures rely on the manager-agent paradigm and domain-based distribution. In this paper, we argue that e-business information systems should instead be structured as self-managed systems: they need to detect problems by themselves, work out the cause of each problem, and take corrective actions independently of any external entity. They should not depend on external managers for their efficiency and robustness. To achieve this, we propose an organizational model that allows management architectures to take into account the specific requirements of self-managed systems, and investigate how to distribute event correlation in the presence of self-managed systems.

Keywords: systems management, event correlation, dependency graphs, self-managed systems, e-business services, distributed systems.

1 Introduction

E-business services are now part of day-to-day life for many Internet users. Leveraging the success of online retail, and more generally the business-to-customer (B2C) trade paradigm, which requires interactive customers, many companies have recently moved (or are migrating) to the business-to-business (B2B) paradigm, whereby distant applications interact directly without the intervention of any person. It seems likely that the proportion of business that is done electronically will continue to grow in the next few years.

As e-business services become increasingly sophisticated, the underlying information systems (i.e., the physical machines, Web applications, databases, etc.) also grow in complexity. We have now reached a point where most service providers want to guarantee highly available services to their customers, but a number of them cannot afford full-fledged fault-tolerant systems to offer such guarantees: they expect management systems to cater for that. Unfortunately, today's management architectures cannot fulfill this new requirement: the manager-agent paradigm and domain-based distribution were never meant for that.

To alleviate this problem, we argue that e-business information systems should evolve into *self-managed systems*. They should detect problems by themselves, work out the cause of each problem, and take corrective actions independently of any external entity. Occasionally they can still interact with managers (e.g., when they receive policy updates or end-to-end monitoring data), but they should not depend on external entities for their performance and stability (as assumed by the manager-agent paradigm). In this position paper, we propose an organizational model that allows management architectures to take into account the specific requirements of self-managed systems, and study how to distribute event correlation in the presence of self-managed systems.

The rest of this paper is organized as follows. In Section 2, we explain why current management architectures are not suitable for self-managed systems. In Section 3, we propose a new way to distribute event correlation in the presence of self-managed systems. In Section 4, we investigate dynamic dependency graphs. Last, we conclude in Sectio n5.

2 Problem Statement

Structuring e-business information systems as self-managed systems poses problems with current management architectures, especially for distributing event correlation. Let us highlight the main ones.

2.1 Manager-Agent Paradigm

The cornerstone of today's management architectures is the manager-agent paradigm. For scalability reasons, the organization to be managed is split into management domains (e.g., on a geographic basis). Each domain is controlled by a *manager*; this application (or, by extension, the host running it) performs all the "smart" tasks that need to be done to manage entities in its domain, including managing faults, configurations, accounting, performance and security [3]. Domains may overlap but let us ignore this case for the sake of clarity.

Within each domain, the managed entities are called *agents*. They are entirely under the responsibility of their manager and provide it with mere instrumentation data: Object IDentifiers (OIDs) in the Simple Network Management Protocol (SNMP), or Common Information Model (CIM) objects in Web-Based Enterprise Management (WBEM). SNMP and WBEM are the two predominant management architectures used to date in the Internet world.

2.2 Monitoring

To ensure the smooth operation of their businesses, most organizations regularly monitor their systems, networks, applications and services in order to detect system/network faults or application/service failures (*reactive management*) and anticipate problems (*proactive management*). A simple example of network monitoring consists in regularly polling the network interfaces of all devices within a management domain. Service Level Agreement (SLA) monitoring consists in regularly checking that customers get the quality of service they pay for (e.g., by impersonating a customer, sending a fake service request and measuring the time it takes for this request to be honored).

When a problem is detected, an event (i.e., the software representation of an alarm) is generated. Some software or hardware entities are capable of automatically sending events to a management application. Others require an external monitor to check their health on a regular basis and send events on their behalf. Today's monitors usually pull instrumentation data off the agents (this is known as *polling*), but they can also rely on a push model for receiving data [10].

2.3 Event Correlation

When they reach the domain manager, all these events are processed by a component called the *event correlator.* Event correlation is an automated process that enables administrators to find, among many events, those revealing critical problems that cannot be ascribed to other issues (*root cause analysis*). Of particular importance is the detection of the few problems that have an adverse effect on the stability and performance of critical services (e.g., an e-business application server for an online retail shop).

Event correlation is the "smart" part of a management application. It can rely on a number of techniques [8]: state transition graphs (finite state machines), rule-based reasoning, binary coding (codebooks), case-based reasoning, probabilistic dependency graphs (Bayesian networks, belief networks), model-based reasoning, neural networks, etc. In today's management platforms, several techniques are often used in conjunction.

Event correlators interact with other components that take corrective actions (e.g., restart an application or reboot a network device), inform customer care that a given problem has been solved, update trouble-ticket systems, etc.

2.4 Distributed Event Correlation

Event correlation becomes particularly complex (thus interesting) in environments where all management applications are integrated. The number of system-, network-, application- and service-related events to correlate can grow very large, beyond a million per day, which poses challenging scalability problems.

Several people proposed to address this issue by distributing event correlation (see Section 5). The most popular approach is to distribute it hierarchically, with one event filter per management domain (these filters operate very fast and are resilient to event bursts) and one event correlator for the entire organization (this correlator receives only filtered events and can perform complex computations).

2.5 Event Flooding Problem

The main problem of this type of distribution is event flooding. For instance, Mansouri-Samani [9] assumes that an event filter is in charge of at most thousands of managed objects (OIDs in the SNMP world, CIM objects in the WBEM world). This assumption is fine when the managed entities (hosts, devices) within the event filter's domain are mostly network devices: a single network device rarely requires the monitoring of more than 10 managed objects, and we seldom find more than hundreds of managed entities in a given domain. But this assumption does not hold anymore for sophisticated e-business information systems.

For instance, let us consider an e-business server consisting of a single physical machine. The service provider is committed to delivering a certain quality of service with this server, which requires detecting and correcting problems at an early stage. To determine the causes of problems occurring on this server, it is necessary to monitor the status of all well-known sources of problems, that is, most physical resources (CPU, memory, RAID disks, etc.) and most software resources (middleware, components, applications, etc.). A summary of the current business activity is also needed: number of service requests received per second, honored per second, etc. As a result, a single e-business server often requires the monitoring of hundreds (sometimes thousands) of managed objects. This processing can take up all the resources of an event filter.

Now, let us assume that this e-business server actually consists of a cluster of 10 powerful machines all located at the same site, within the same domain (and thus under the responsibility of a single manager). If each machine requires the monitoring of hundreds or thousands of managed objects, the total flow of events to the manager could easily overwhelm a standard management platform in the Internet world.

Last, let us consider the case of an e-business server that consists of 10 powerful geographically distant machines. Problems occurring within this server may be intertwined with network problems between any two parts of this distributed system. Service-, system- and network-level monitoring systems therefore have to be thoroughly integrated. Many (all?) management platforms in the Internet world would struggle to manage such a demanding distributed system in parallel to the other agents in the domain.

To make sure they can offer guarantees of service availability, service providers sometimes use dedicated management platforms for managing their most complex e-business information systems; but this approach defeats the purpose of integrated management. We need a better solution.

2.6 *Latency Problem*

Another problem associated with the type of distribution described in Section 2.4 is latency. Correlating cross-domain user complaints or SLA failures with temporary instabilities in the network or faults in geographically dispersed applications is difficult enough to distribute. But the problem is even worse for e-business information systems, which have strict SLA constraints and sometimes cannot cope with the latency of today's event correlators when the latter run on remote managers.

3 New Organizational Model

To alleviate this problem, we propose to distribute event correlation down to the level of self-managed systems. The idea is to view e-business servers as autonomous systems capable of taking independent corrective actions, just like autonomous robots or intelligent agents in distributed artificial intelligence.

This requires a change in the organizational model that currently underlies both SNMP and WBEM. In a management architecture, the organizational model defines the way different entities interact and share the management work load [10].

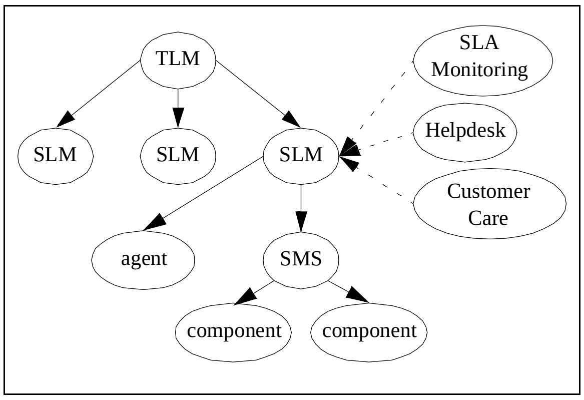

Figure 1: Hierarchical Management with Self-Managed Systems

As usual with hierarchically distributed management, we have one Top-Level Manager (denoted TLM on Figure 1) for the entire organization and an arbitrary number of layers of Sub-Level Managers (SLMs), each in charge of a management domain within the organization [10]. For the sake of clarity and without undermining the generality of our approach, let us assume that we have only one layer of sub-level managers.

The top-level manager is in charge of enforcing organization-wide policies and solving cross-domain problems (especially boundary problems). It runs an event correlator called the *TLM event correlator*.

Within each management domain, we have one sub-level manager, a number of "dumb agents" as they are called in the Internet world (remember that the manager does everything "smart" in the manager-agent paradigm), and optionally some self-managed systems (which need not necessarily be e-business servers).

Each sub-level manager runs its own event correlator (called the *SLM event correlator*). On one side, it communicates with the top-level manager (e.g., to receive new rules). On the other side, it interacts with dumb agents, polling them for monitoring data and/or receiving events (e.g., SNMPv1 traps, SNMPv2 notifications or CIM events). Dumb agents are entirely controlled by their sub-level manager.

So far, we have described a fairly standard organizational model in case management is hierarchically distributed. The novelty of our proposal stems from the fact that each Self-Managed System (denoted SMS on Figure 1) runs its own event correlator (called the *SMS event correlator*), which is in charge of correlating all the events generated within this system.

If all the constituents of the self-managed system are located inside a single management domain, this system is under the responsibility of the sub-level manager in charge of this domain (this is the case depicted in Figure 1). Otherwise, it is directly under the responsibility of the top-level manager. This convention determines which manager sends it input from customer care, new rules, etc.

When performing cross-domain event correlation or triggered investigations (e.g., in case the execution of a rule requires the retrieval of more management data via polling), the top-level manager does not distinguish between sub-level managers and self-managed systems directly under its responsibility. Self-managed systems therefore need to be able to answer requests from the TLM and SLM event correlators.

One advantage of letting a self-managed system correlate its own events is locality: most events need not be propagated upward to the sub-level manager because their impact is limited to the self-managed system. Another advantage is that latency is significantly reduced (see next).

4 Dynamic Dependency Graphs

To perform root-cause analysis on the events that occur within it, a self-managed system needs to have an internal "model of the world", that is, a model of its own subsystems, coarse-grained components and fine-grained components. Ideally, it should know the states of all its constituents and all the hardware and software dependencies between them, at any time.

Unfortunately, this is not possible. States are updated in pseudo real-time (e.g., every 10 minutes in case of polling), not in real-time, so its knowledge of the states of its constituents may be obsolete. Dependencies change over time, some of them frequently, so keeping an up-to-date model of all dependencies is wishful thinking. Last but not least, the purpose of a self-managed system is to deliver a service, not to manage itself, so the amount of resources dedicated to self-management should remain reasonable.

Consequently, just as a manager with the agents in its domain, a self-managed system can only afford to maintain a limited knowledge of the states of its constituents and the dependencies between them.

The good news is that self-managed systems can exploit the locality of their event correlator to refine their internal models as they progress during the investigation of a given problem. The reduced latency that results from local polling makes it possible to refine a dependency graph or capture dynamic dependencies on demand—something that is rarely achievable when event correlation is performed by a remote manager.

Another advantage of locality is that it is easier to cope with event bursts. If a hardware or software component supports a debugging mode to temporarily provide very detailed information to the event correlator, the self-managed system may decide to temporarily switch on this mode. This is much more risky with an external domain manager.

A potential drawback of having event correlation performed by a self-managed system is that we lose the end-to-end vision of a sub-level manager. This problem can be overcome by configuring a monitor to send end-to-end service monitoring statistics to the self-managed system that offers this service (e.g., on Figure 1, the SLA monitor for the self-managed system is the sub-level manager one level up). An alternative to configuration is publish-subscribe.

5 Related Work

A number of authors have already investigated the distribution of event correlation. Some proposals focus on correlating network events with the network topology [4] or intrusion detection [7]. Others propose general-purpose languages [2][9]. Some people split the fault management engine into components that can run on different hosts, but none of these components can itself be split and distributed [1][2]. Others claim to support distributed monitoring but only have a single event correlator fed by multiple monitors or hosts [6].

One of the most advanced proposals published to date is GRACE [5]. This event correlation service is more mature than ours, but it does not address the specific issue of self-managed systems.

6 Conclusion

In this paper, we have proposed to structure e-business information systems as self-managed systems. To make it possible, we have described how to modify the organizational model of current management architectures. We have also showed how to distribute event correlation in the presence of self-managed systems.

In the future, we plan to implement this proposal and demonstrate its potential. It would be interesting to quantify the latency reduction when a self-managed system correlates its own events, but also the latency increase when a higher-level manager performs cross-domain event correlation.

Acknowledgments

The author thanks Emil Lupu and Joe Sventek for their feedback on early versions of this paper.

References

[1] E. Akbas, “System Independent and Distributed Fault Management System”, in *Proc. ISCC 2003*.

[2] E. Al-Shaer, *A Hierarchical Filtering-Based Monitoring Architecture for Large-Scale Distributed Systems*, Ph.D. thesis, Old Dominion University, USA, 1998.

[3] CCITT (now ITU-T), *Recommendation M.3400. TMN management functions*, ITU, October 1992.

[4] C.S. Chao, D.L. Yang and A.C. Liu, “An Automated Fault Diagnosis System Using Hierarchical Reasoning and Alarm Correlation”, *Journal of Network and Systems Management*, Vol. 9, No. 2, June 2001.

[5] G. Jakobson, M. Weissman, L. Brenner, C. Lafond and C. Matheus, “GRACE: Building Next Generation Event Correlation Services”, in *Proc. NOMS 2000*.

[6] J. Joyce, G. Lomow, K. Slind and B. Unger, “Monitoring Distributed Systems”, *ACM Transactions on Computer Systems*, Vol. 5, No. 2, May 1987.

[7] C. Kruegel, T. Toth and C. Kerer, “Decentralized Event Correlation for Intrusion Detection”, in *Proc. ICISC 2001*.

[8] L. Lewis, *Managing Business and Service Networks*, Kluwer, 2001.

[9] M. Mansouri-Samani, *Monitoring of Distributed Systems*, Ph.D. thesis, Imperial College, UK, 1995.

[10] J.P. Martin-Flatin, *Web-Based Management of IP Networks and Systems*, Wiley, 2002.