J Netw Syst Manage (2007) 15:425–445
DOI 10.1007/s10922-007-9082-9

On the Evolution of Management Approaches, Frameworks and Protocols: A Historical Perspective

George Pavlou

Published online: 13 October 2007

Abstract Network, system and service management has evolved into an important scientific discipline over the last 20 years. While management problem solving is expected to continue ad infinitum, one would have expected that, after 20 years of research and standardization, an agreement would have been reached regarding a common management framework and protocol. But despite relevant progress and various available solutions, there seems to exist a permanent quest for the all encompassing next generation management technology. This paper looks at the evolution of management approaches, frameworks and protocols over the last 20 years, proposes a relevant taxonomy, presents the salient features of the representative tecnologies and discusses relevant issues. The purpose of this paper is to document historically this evolution, highlight important design choices and explain the how's and why's behind the various frameworks and technologies. The paper is concluded with a summary and future outlook.

Keywords Manager–agent · Distributed objects · Web-based management · Management by delegation · Mobile code

1 Introduction

Network, system and service management has evolved into an important scientific discipline over the last 20 years. Communication management systems encompass the following major aspects. First, the architecture regarding the structure of management applications that communicate in order to solve a particular problem, for example centralized, distributed hierarchical, distributed cooperative, etc.

G. Pavlou (✉)
Department of Electronic Engineering, Centre for Communication Systems Research, University of Surrey, Guildford, Surrey GU2 7XH, UK
e-mail: g.pavlou@surrey.ac.uk

Second, the way in which information about managed resources is modeled and accessed in a distributed fashion, for example protocol-based with simple variables, protocol-based object-oriented, distributed object based, etc. And third, the problem solving logic and algorithms of management applications.

Previous and current research has been addressing all these aspects. While architectural and information modeling/distributed access aspects can be agreed and standardized, management problem solving is not subject to agreements in order to leave degrees of freedom to management solution providers. In fact, relevant research work is expected to continue ad infinitum as different networking environments emerge with new management needs, providing fertile soil for applying new problem solving techniques.

But one would have expected that after more than 20 years of relevant research, development and evolution, an agreement would have been reached by now regarding management information modeling and distributed management information access. There have been standardized solutions that were deployed and used in the real world, for example OSI System Management (OSI-SM) for telecommunication environments and Simple Network Management Protocol (SNMP) for enterprise and network provider IP-based environments, but there always seems to exist a permanent quest for the "all encompassing next generation management technology".

This paper looks at the evolution of management approaches, frameworks and technologies over the last 20 years, it proposes a relevant taxonomy and discusses the salient features of the representative ones. We start from the early days of procedural technologies, we move to the manager–agent model and associated technologies, we discuss distributed object and service interface approaches and also address management by delegation in various forms i.e., moving management logic close to the data it requires. The purpose of this paper is to document historically this evolution, highlight important design choices and explain the how's and why's.

The remainder of this paper has the following structure. Section 2 looks at technology evolution from a historical perspective. Section 3 presents the fundamental management models, i.e., the manager–agent model and the distributed object/service interface model; it then proposes a taxonomy of management approaches, frameworks and protocols. Section 4 presents the fundamental aspects of key technologies, including OSI-SM, SNMP, Common Object Request Broker Architecture (CORBA) and Web Services, and discusses issues behind their successes, failures and future potential. Section 5 discusses briefly related work. Finally, Section 6 presents a summary and outlook for the future.

2 Technology Evolution

2.1 Procedural Approaches

Network management has always been a key aspect of communication networks but remote "soft" manipulation of managed devices only emerged as an important

aspect in the 1980s. The first generation remote management approaches were procedure-based, with protocol primitives being specific to a particular application, for example *enable_interface(ifId = 2)* and *disable_interface(ifId = 2)* as opposed to a generic object-oriented approach in which the relevant operations could be *set(objName = if2, state = enable)* and *set(objName = if2, state = disable)*. This "procedural" approach continued to find supporters until the early 1990s, despite the advent of generic protocol manager–agent approaches, as it is detailed in the next subsection.

In the days before object-oriented distributed object platforms (i.e., in the late 1980s to early/mid-1990s), the Open Software Foundation (OSF) produced specifications for the Distributed Computing Environment (DCE) and the Distributed Management Environment (DME). These were non object-oriented software platforms for distributed computing and management respectively, and there was significant hype behind them at the time and behind the OSF DME in particular. But the latter was caught between the protocol-based manager–agent approaches of the late 1980s and the fully object-oriented distributed platforms that emerged in the mid-1990s, never seeing any significant uptake despite the original promises. [1] describes eloquently the reasons behind its demise, being appropriately titled "Icarus, Alice and the OSF DME". We may conclude that *remote procedure call* approaches lost to *remote method call* ones, the latter reflecting a cultural evolution in software design, that of object-orientation.

2.2 Object-Oriented Generic Protocol Approaches

The first object-oriented approach to management originated from ISO in the mid to late-1980s. ISO standardized an approach for managing OSI intermediate systems (switches/routers) and end systems (computers or terminals), hence the term OSI-SM [2]. This approach was also adopted by ITU-T and formed the key interface technology for the Telecommunication Management Network (TMN) [3]. OSI-SM proposed the manager–agent model, in which the resources of a managed system are modeled through managed objects to which collective access is provided by an agent. We discuss OSI-SM in some detail later as a representative technology, but it is worth mentioning that it exhibits an object-oriented information model, sophisticated selective information access capabilities through scoping/filtering, a powerful event model and a set of generic functions to support common types of management activities, i.e., the Systems Management Functions. On the other hand, it is fairly complicated technology tied to OSI protocols, so it has only found use in telecommunication environments for the management of synchronous transmission networks, the public switched telephone network and cellular networks, where it is still in use today.

While OSI-SM standardization was in progress, an initial thought in the Internet community was to map it onto TCP/IP protocols in order to use it for the management of IP-based devices, i.e., routers, end-systems, etc. This resulted in the CMIS/P Over TCP/IP (CMOT) approach [4], which uses a lightweight presentation protocol that runs directly over either TCP or UDP, or in the full CMIP OSI upper

layer stack over TCP using the RFC1006 method. In fact, the Internet management information for managing the TCP/IP protocol suite was converted into the equivalent OSI-SM one and there has been at least one implementation of this approach as part of the first version of the OSIMIS OSI-SM software platform. But CMOT never caught up because Internet management emerged, so OSI-SM has been confined to the management of telecommunication environments.

The Internet community had in parallel been working towards a simpler protocol and information model that would be easily implementable without posing significant overhead to managed devices. SNMP [5] and associated information model was finalized around 1990 and was an instant success, being deployed in all IP-capable managed devices and used initially for enterprise management. In addition, as IP became the dominant networking technology in the 1990s, SNMP has been increasingly used for the management of Internet Network Provider IP networks. SNMP uses a not fully object-oriented "variable-based" information model and a small set of operations. The original version omitted a few important aspects which were added by subsequent versions: proper emulation of creation and deletion through the set operation and the addition of operations for bulk data retrieval and reliable event delivery. While SNMP's simplicity contributed substantially to its wide deployment, the lack of transaction support and security, which are essential for configuration management, has resulted in it being used mostly for monitoring. It should be mentioned that version 3 added a proper security framework but this came too late, when vendor-specific approaches were being used for configuration management. The IETF has finally decided not to evolve SNMP any further in 2002 [6] and a XML-based approach is currently being standardized for configuration management, as it will be discussed later.

Given SNMP's problems with configuration management, the IETF resource allocation working group came up with the Common Open Policy Service for Provisioning (COPS-PR) [7] in the late 1990s/early 2000s. COPS-PR supports policy configuration interactions between a policy decision point (PDP) and policy enforcement points (PEPs) within managed devices. It uses TCP-based reliable transport and supports atomic transactions and security. Its data model is similar to the SNMP-SMI and is known as the Policy Information Base (PIB). The key problem with COPS-PR is that, while it is quite similar to SNMP, it does not maintain backwards compatibility with it while at the same time it does not fix other important SNMP shortcomings, for example the rudimentary information model. As a result, despite the original hype it has never seen any significant uptake in the real world.

2.3 Distributed Object Approaches

One key aspect of the remote procedure call approaches is the fact that they standardize a programming language dependent Application Programming Interface (API) in addition to the remote invocation protocol that supports interoperability. This API shields the programmer from the underlying network details and harnesses the problems of distributed access, allowing non-network programmers to develop

distributed applications. With the advent of object orientation in the mid-1980s and its wide adoption thereafter, remote procedure call approaches naturally evolved to distributed object ones. In a similar fashion to the previous effort of OSF's DCE, the Object Management Group (OMG) produced the CORBA [8] which can be thought as the representative vendor-neutral distributed object approach.

In CORBA, object interfaces support inheritance and are accessed in a distributed manner through stub objects that are present in the client's address space, harnessing to a large extent the problems of distribution. Unlike the manager–agent model, every object is a "first class citizen" of the supporting infrastructure and its interface is accessed separately. While this model was devised for general purpose distributed applications, it is possible to use it for network management by modeling managed objects as CORBA object interfaces. As such, the communications management community spent significant effort researching into potential approaches for using it for network management. Some of these efforts were fed back to OMG who enhanced CORBA with a sophisticated event notification service that supports filtering and also with the lightweight portable object adaptor that can in principle support large object populations within a single device.

Despite the original vision that CORBA would evolve into the prevailing technology for distributed network management and the support by some vendors in the telecommunication domain, this approach never caught up in a significant scale. The key reasons behind this is that CORBA does not support bulk data retrieval, it is relatively resource hungry and, more important, because technologies such as Java RMI and more recently Web Services have been directly challenging it in the quest for the all-encompassing distributed management technology.

The Java Remote Method Invocation (JRMI) is another distributed object approach that Sun Microsystems built into their extremely successful Java programming language. The key difference between CORBA and JRMI is that the former supports bindings to many programming languages, including C, C++, Java and Smalltalk, while JRMI is tightly bound to Java, and this could be a problem for real-time applications. An additional difference is that CORBA has been enhanced with sophisticated services that are required by management applications, for example event notifications, while JRMI does not include services of similar sophistication. JRMI has been used for network management mostly in the research domain and it is also been challenged by the emerging Web Services technology.

2.4 Management by Delegation and Mobile Code Approaches

In all the frameworks and technologies described so far, information in managed devices is accessed by a manager application through remote operations that retrieve information or receive events for gaining awareness (monitoring) and change information for intrusive (configuration) management. A totally different approach pioneered in the early 1990s is management by delegation [9]: instead of performing management tasks through remote operations, code is uploaded to managed devices in order to perform these operations locally and, for example, inform the manager

regarding the consolidated device state when monitoring or verify that the actual changes have been done in case of configuration management.

This approach is more difficult with platform-dependent programming languages such as C and C++, requiring "elastic servers" that can host new code that has access to the server's managed objects through a well-defined API. Management by delegation became easier with scripting languages such as Tcl that appeared in the early 1990s and even easier with execute-anywhere languages such as Java in the mid to late-1990s. In practice, there are two approaches of management by delegation: manager–agent based approaches in which the upload, activation, monitoring, suspension and termination of scripts and programs is controlled through support managed objects handled by an agent; and a more liberal approach in which mobile objects migrate and execute close to other local objects to which they need access—in this case, a distributed platform supporting mobile code is necessary.

There have been two manager–agent based delegation approaches. The Scripting Management Information Base (MIB) [10] supports the delegation of scripts or even of compiled programs through an SNMP agent and the Command Sequencer [11] supports the delegation of scripts to an OSI-SM agent. Both approaches have been the result of research in the mid to late 1990s, see [12] for the research work and potential uses of the scripting MIB and [13] for early work that demonstrated delegation in the context of OSI-SM. But neither approach has been used much in real world management scenarios.

The advent of Java gave rise to the emergence of mobile agent platforms in the late 1990s and many researchers considered their use in network management, with an early survey of potential applications presented in [14]. There exist two key approaches in using mobile agents for network management as classified in [15]: *constrained mobility*, where the agent is told beforehand the itinerary of which network locations to visit; and *full mobility*, where the agent senses the environment and makes autonomous decisions regarding its next move. A special case of constrained mobility involves the itinerary of just one hop and is equivalent to management by delegation, but this is supported more efficiently by the manager–agent delegation approaches rather than a full-blown mobile agent platform. Despite the amount of research work regarding the use of mobile agents for network management, there have never been "killer applications" demonstrating the use of full agent mobility and, as a result, mobile agent approaches have never been used in the real world.

2.5 XML and Web-based Approaches

The eXtensible Markup Language (XML) that emerged in the mid-1990s allows one to define arbitrary structures through Data Type Definitions (DTDs) and hence it can be used as a textual encoding mechanism for application protocols, object and interface specifications, etc. Given that most enterprise applications store data in XML formats, XML-based network management technologies are considered particularly attractive given the potential easy integration of XML-based

management data with that of other applications. In addition, combining XML with Web-based approaches is beneficial given the ubiquitous low cost infrastructure of the latter and the proven scalability and security features.

The first XML Web-based management approach was the Distributed Management Task Force's (DMTF) Web-Based Enterprise Management (WBEM) [16] that emerged in the mid to late 1990s and the target applications were system management for desktop computers but also network management. A key aspect of the WBEM framework is the Common Information Model (CIM) [17] which provides generic classes from which application-specific information models are derived, in a similar fashion to the TMN Generic Network Information Model [18]. The DMTF information model uses the Managed Object Format (MOF) as a language for formal managed object specification, together with UML class diagrams. The manager–agent model was adopted with a management protocol known as CIM-XML that produces XML encodings of CIM object method request and responses and transports them over HTTP/TCP. The protocol provides a simple operation request/response that emulates a remote method call and also a multiple operations request/response, in which multiple remote method calls are bundled in a request and response accordingly, supporting bulk retrieval and atomic configuration. There also exists a CIM protocol mapping to the Lightweight Directory Access Protocol (LDAP) [19] for use in the DMTF's Directory Enabled Networks (DEN) framework. While CIM and its extensions, for example the Policy CIM (PCIM), have been used in various applications, there has not been much use of the CIM-XML over HTTP protocol for system or network management.

The second XML-based management approach emerged in the early 2000s when IETF started standardization work to produce a framework and protocol that would address SNMP's shortcomings for configuration management, namely transaction support and security. As already discussed, because of these limitations configuration was done using vendor-specific approaches, for example Cisco's CLI or Juniper's Junoscript. The latter is XML-based and the ideas behind it formed the basis of IETF's work towards the Network Configuration protocol (NetConf) [20]. The novel idea in NetConf in comparison to other management protocols is that configuration is not done by remotely manipulating settable managed object attributes or variables but it follows a "document-based" approach: a new configuration is described as an XML document that specifies the values settable parameters should have and this configuration document is uploaded and interpreted by the NetConf agent according to the relevant data model, setting the right attributes or variables. Configurations may be retrieved, edited, deleted and copied/enabled. Locking is also supported to prevent interference between different managers. The default transport mapping of NetConf is over the SSH protocol but all transport mappings support security features in terms of authentication, integrity and confidentiality. At the time of writing the NetConf protocol has been finalized but work on associated data models is only starting, and this is essential given that a management protocol is only a generic access mechanism.

Finally, in the early to mid 2000s, Web Services emerged as a promising XML-based technology for standardizing e-service interfaces and, given the similarity to distributed object technologies, there has been a lot of research and standardization

work targeting their use in network, system and service management. WS interfaces are specified in an XML-based language that supports service inheritance and they can be accessed in various ways, with the default access mechanism being Simple Object Access Protocol (SOAP) over HTTP. One key difference between WS and distributed object approaches such as CORBA is that WS does not have prescribed programming language bindings through standardized APIs, which leaves space for optimized implementations. But given the similarity with distributed object technologies, WS exhibit similar problems when used for management, most notably the support for optimized data retrieval and the fact they constitute a relatively heavyweight technology. We will discuss this aspect further in Section 4.

3 Management Models and Taxonomy of Approaches, Frameworks and Protocols

All the object-oriented approaches that were briefly presented in the previous section fall into two major categories:

- Those that follow the manager–agent model, in which a cluster of managed objects are collectively administered by an agent that provides a unique entry point for accessing them; and
- Those that follow the distributed object or service interface model, in which every managed object or interface is accessed individually.

We will present briefly these two models and we will continue with a taxonomy of the frameworks and protocols presented so far.

3.1 The Manager–Agent Model

The manager–agent model defines the principles of operation for protocol-based management frameworks [2]. Managed resources are modeled through *managed objects (MOs)*, which encapsulate the underlying resource and offer an abstract access interface. Any managed network, system or service element should expose a "cluster" of managed objects modeling its resources across a management interface provided by an *agent*. The interface is formally defined through specification of the available managed object types or classes and the management access service/protocol. *Manager* applications access managed objects through agent interfaces for implementing management policies. A management application may act in both agent and manager roles, and this is the case for peer-to-peer management interactions or for hierarchical management environments.

An agent is a software entity that administers managed objects, responds to management requests and disseminates spontaneous events through the management protocol. It uses an implementation-specific mechanism to access the managed objects and, given its collective view of the relevant cluster, it can provide sophisticated object access facilities for bulk data transfer or selective information retrieval. In addition, it can evaluate event notifications at source and only send

them to interested managers according to predefined, potentially sophisticated, criteria. The management protocol supports access of multiple attributes from multiple objects through one request. The manager–agent model projects a communication framework, in which standardization affects only the way in which management information is modeled and carried across systems, leaving deliberately unspecified aspects of their internal structure. This may result in highly optimized implementations given that there are no standardized internal APIs to be adhered to. The model is depicted in Fig. 1.

OSI-SM, SNMP, COPS-PR, WBEM and NetConf follow the manager–agent model, albeit with important differences regarding the capabilities of the management protocol and the associated information model. It should be also emphasized that NetConf is qualitatively different to all the other approaches given that it follows a "document-based" as opposed to individual managed object access approach.

3.2 The Distributed Object and Service Interface Models

Open Distributed Processing (ODP) [21] is a general ISO/ITU-T framework for specifying and building distributed systems. ODP came as response to the recognition that although ITU-T and IETF protocol-based solutions addressed the problem of heterogeneous system interconnection, the proliferation of application layer standards and distributed applications meant that application inter-communication needed to be addressed as well. This was further fuelled by the convergence of the information technology and communication sectors and the resulting demand for standardized APIs between distributed application components and underlying platforms. Hence, the target for ODP is to facilitate distribution, interoperability but also to achieve software portability.

ODP projects a client-server model, with distributed applications composed of objects interacting solely through accessing each other's *interfaces*. The underlying ODP platform, or Object Request Broker (ORB) in CORBA, provides a number of transparencies, such as access, location, resource and replication. Clients access

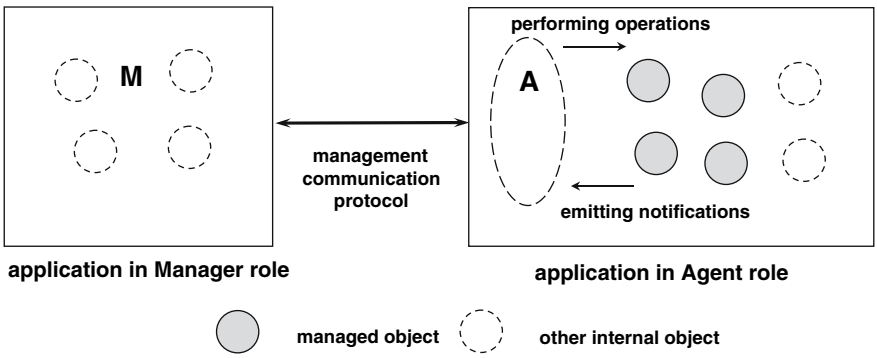

Fig. 1 The manager–agent model

server objects through *interface references*, obtained through access to well-known special servers i.e., name servers. A name server keeps a name space with interface references "advertised" by server objects while clients can resolve names to interface references. Finally, there is an underlying protocol for interoperability but is hidden inside the supporting software platform, with objects "sitting" on the platform through a standardized API. The ODP model is depicted in Fig. 2. CORBA and JRMI follow this model while this is also the case for mobile agent platforms, in fact the latter are in fact distributed object platforms that also support *migration transparency*.

A variation of the distributed object model is the Service Interface Model. In this case, services offer interfaces that are described through service or resource identifiers. The latter can be discovered through special discovery services and then the functionality of the service can be accessed by client applications. This model is in fact the same as the distributed object model but the emphasis is not on software portability, hence there are no standardized APIs. Only the distributed access protocol and the formal service interface description language are standardized, with specific implementations offering their own APIs in a similar fashion to the manager–agent model. We do not provide a pictorial depiction of this model but if we did it would be similar to the distributed object model without the underlying platform. Web Services is the only current framework that follows (or rather has defined) the service interface model.

3.3 A Taxonomy of Management Approaches, Frameworks and Protocols

In Section 2 we looked at all the management approaches, frameworks and protocols from a historical perspective, grouping them into the following categories:

- Procedural approaches;
- Object-oriented protocol-based approaches;
- Distributed object approaches;

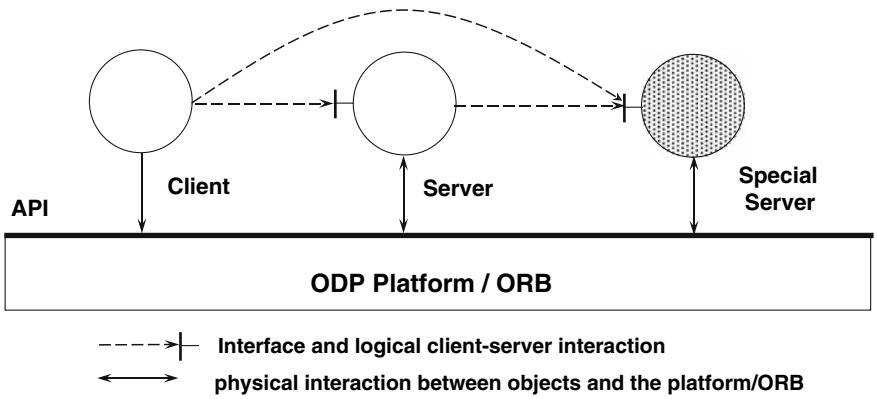

Fig. 2 The distributed object model

- Management by delegation and mobile code approaches; and
- XML and Web-based management approaches.

For the rest of this paper we will ignore procedural approaches which, as we already mentioned, disappeared in favor of object-oriented ones. Trying to categorize the rest, there exist two fundamentally different ways of performing management tasks: by invoking remote operations on managed objects or by sending management logic, i.e., code, to managed devices so that it lies close to the objects it requires in order to perform operations locally. The latter is, as we explained, the management by delegation paradigm and there potentially exist various degrees of freedom that mobile code can have, as we will discuss shortly. So the first categorization of management approaches is:

- Management by remote invocation; and
- Management by delegation.

Considering the former, there exist two different ways of performing remote invocations: performing the invocations on managed objects through an agent, i.e., the manager–agent model; or performing the invocations directly to the managed objects, i.e., the distributed object or service interface model. In the latter case we assume that individual managed objects are modeled as distributed objects or service interfaces, but there also exist variations of this approach as we will discuss in Section 4. We have already mentioned that OSI-SM, SNMP, COPS-PR, WBEM and NetConf follow the manager–agent model while CORBA, JRMI and WS follow the distributed object/service interface model.

Now considering management by delegation, the simplest case is for logic to be uploaded to a managed device in order to operate close to its managed objects. This "one-hop" mobility can be supported by special MIBs in the manager–agent model and relevant standardization work has resulted in the SNMP Script MIB [10] and the OSI-SM Command Sequencer [11]. In the more general case, logic can move from device to device, either through a predefined itinerary and this approach is known as constrained mobility [15], or autonomously by sensing its environment, and this is the full mobile agent approach. So we can categorize management by delegation into one hop manager–agent based approaches and multiple hops mobile code based ones.

The full taxonomy of management approaches, frameworks and protocols is depicted in Fig. 3.

4 Key Technologies and Relevant Issues

In this section we are going to present the salient characteristics of key technologies and discuss the issues behind their success, failure or future potential. The selected technologies to be presented are the following: OSI-SM, because it is a fairly comprehensive technology and is still used in telecommunication environments; SNMP because of its widespread deployment and simplicity; CORBA as a representative distributed object technology, although it has not seen much

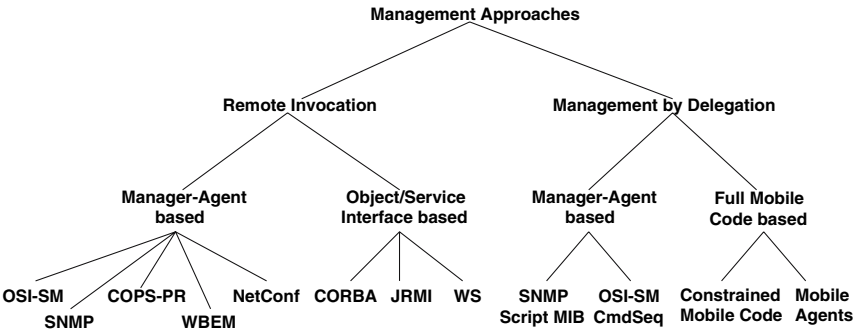

Fig. 3 Taxonomy of management approaches, frameworks and protocols

deployment for network management; and WS because it currently emerges as the potential future technology.

4.1 OSI System Management

OSI-SM [2] was the first management technology to be standardized and, in fact, the manager–agent model was devised in that context. It is a sophisticated and powerful technology but also complicated and expensive to deploy, so it has only found use in telecommunication environments. It is the primary technology behind the TMN Q/X interfaces, although alternative technologies such as CORBA may be also used for the latter. It is accepted that it has been the most powerful technology so far, supporting features that should be essential in any management framework, so it is worth examining them.

OSI-SM uses a fully object-oriented information model supporting inheritance. Managed object classes are specified in the Guidelines for the Definition of Managed Objects (GDMO) abstract language. Key deviations from object-oriented concepts are the runtime specialization of a MO instance through conditional packages and the fact that imperative commands are modeled through a generic method called *action* with a single parameter and result, which may result in awkward parameter modeling. Telecommunication information models were initially specified in GDMO, although there is a push today towards technology-neutral information specification in the Unified Modeling Language (UML), with reverse-engineering of existing models.

The Common Management Information Service/Protocol (CMIS/P) supports Get, Set, Action, Create, Delete and Event primitives. Information access is powerful, supporting both bulk retrieval through *scoping* and selective retrieval through scoping and *filtering*. Since managed objects are organized in a management information tree based on containment relationships, scoping works by selecting a number of subtree objects while filtering eliminates further the selection by applying a predicate on attribute values. Connection-oriented reliable transport is used, which means that arbitrarily large amounts of bulk data may be

retrieved in one go. Concerted configuration changes through a series of Set operations are supported through the OSI Transaction Processing facility, which "brackets" these requests and guarantees atomicity i.e., all to succeed or none to be performed. Finally, the event framework is sophisticated, allowing managers to create their *event discriminators* with filtering on the event type, time, object name of the managed object that emitted the event and the actual notification information.

OSI-SM is a powerful technology but suffers from over-engineering and the fact it is tied to OSI protocols that have gone out o fashion. It provides too many options which make it cumbersome, expensive to implement and relatively difficult to use. For example, conditional packages are a managed object feature that deviates from object-oriented software engineering concepts and could have been realized through subclasses instead. But the OSI-SM concepts have influenced subsequent approaches, for example SNMP get-bulk is influenced from scoping and NetConf filtering is influenced from OSI-SM filtering. In summary, OSI-SM failed to succeed in a large scale despite the significant research and standardization effort that went into it but it has the elements of a complete management technology and may continue influencing future technologies. In addition, there has already been a lot of investment in it in telecommunication environments (SDH/SONET, PSTN, cellular) and a large installed base of equipment with TMN Q interfaces, so it will continue being used there.

4.2 Simple Network Management Protocol

By SNMP [5] we effectively refer to the Internet Management framework. The latter was conceived as a simplification of the OSI-SM one to be used in simpler IP-capable devices. Its simplicity contributed substantially to its success and it is widely supported by IP network devices. While OSI-SM requires managed devices to be relatively complex due to information model complexity, reliable transport and event-based operation, SNMP designers have opted for a very simple information model, unreliable transport and mostly polling-based operation. These decisions leave managed devices simple and have contributed substantially to its success and wide deployment. On the other hand, they shift complexity to Network Management Centers (NMCs) and result in heavy management traffic, limiting the extent to which sophisticated management functionality can be deployed.

The SNMP information model is very simple, with scalar variables (of either integer or string type) used to model managed entities. SNMP objects are formally specified in a language known as Structure of Management Information (SMI). Although we talk of objects in SNMP, they are effectively equivalent to object attributes in other frameworks. There is no inheritance and the only operations allowed on them are read and write. These objects can be either single- or multiple-instanced, the latter modeling tables that consist of a series of rows. Table rows are the only composite objects that can be created and deleted. This very simple variable-based rather than object-oriented approach may result in significant awkwardness and complexity when trying to model complex managed entities.

There is no information reuse through inheritance while complex data types and imperative methods can only be modeled indirectly.

SNMP provides Get, Set and Event (Trap and Inform) operations. Single-instanced objects are accessed through Get while table objects are accessed through two variations of Get, GetNext and GetBulk. SNMP objects in an agent are ordered in a linear fashion given the naming architecture, with table objects having as next the objects of the next row. GetNext and GetBulk exploit this structure, with GetNext requesting the first next of a series of objects (typically a table row) and GetBulk requesting N next objects. With GetNext, a table of R rows can be accessed through R + 1 serial requests. These can be reduced with GetBulk but it is difficult to estimate N properly so as not to "overshoot" over the end of the table and at the same time minimize the required transfers. Given that SNMP uses by default UDP transport—the TCP mapping as in RFC3430 is not widely used, a response to Get, GetNext or GetBulk should typically fit in a single packet to avoid IP-level fragmentation that increases the probability of SNMP packet loss; this reduces effectiveness for bulk transfers. There is also no selective retrieval in a similar fashion to CMIS/P filtering. The event framework allows managers to declare interest on particular event types but without filtering. In addition, for historical reasons management stations do not rely on events, which in SNMPv1 were sent through unreliable *traps*, despite the fact that subsequent versions have included reliable *inform-requests*; as a result, management stations deploy a mostly polling-based regime.

But a fundamental limitation of SNMP is that it does not support coordinated Sets to move from a configuration A to B in a consistent manner through a transaction processing facility. In fact, even simple Set commands can be problematic with SNMP, while an important additional problem is the lack of security. Proper security features appeared only in SNMP version 3 but they have not been deployed because by that time it was clear that SNMP had too many other deficiencies, namely the rudimentary information model and the lack of transaction support. It should be also stated that the lack of strong security discouraged MIB designers from including appropriate configuration features in MIBs, i.e., the full set of required read-write objects. COPS Provisioning (COPS-PR) was conceived as an approach similar to SNMP that could support concerted configuration changes, but it did not maintain compatibility with SNMP while at the same time it did not fix other important shortcomings, e.g., the rudimentary information model, so it did not succeed.

As a result of all these limitations, SNMP has been mostly used for rudimentary monitoring rather than for intrusive management. In fact, given the absence of features for sophisticated monitoring, vendor-specific tools such as Cisco's Netflow are used for the latter. But SNMP has fulfilled an important goal, by providing inexpensive management interfaces to every IP device and showed the potential of an inexpensive management technology. Its simplicity though was also a double-edged sword and it would have been better if it had included desirable features from the beginning: information model inheritance, TCP-based reliable transport, a create operation, transaction support and strong security. Given that these are impossible to retrofit while maintaining backwards compatibility, SNMP is not going to be

developed any further [6] but it will continue being used for simple monitoring, with NetConf seen as the future solution for configuration management.

4.3 Common Object Request Broker Architecture

While there exist distributed object technologies such as Sun's JRMI which is part of J2EE and Microsoft's DCOM, OMG's CORBA [8] is certainly the representative one, being collectively specified so that it constitutes a real-world approach towards an ODP-inspired solution. While manager–agent approaches such as SNMP and OSI-SM are *communication* frameworks, CORBA targets a *programmatic* interface between objects and the underlying Object Request Broker (ORB). Choices made by ITU-T/IETF on the one side and OMG on the other side reflect their different pre-occupations: management communications for the former and distributed software systems for the latter.

CORBA uses a fully object-oriented information model supporting inheritance. Objects are effectively defined through their interfaces, which are specified in the Interface Definition Language (IDL). Methods with any argument and result parameters are possible, providing full flexibility as it befits a distributed systems technology. Attributes with read or read-write properties may be also defined, which effectively results in *<attr>_get* and *<attr>_set* methods being generated. This in fact implies that, in the default case, attributes are accessed individually through a remote method call, which can be expensive.

CORBA specifies a general remote call protocol, the General Inter-Operability Protocol (GIOP), which simply defines request, response and error packets carrying the parameters and results of operations. Its most common mapping is over TCP/IP, known as the Internet IOP (IIOP), providing reliable transport but with connection management hidden by the ORB. Objects are typically accessed through their interface methods, one in each remote call, with no support for bulk transfer or selected retrieval. There is though support for atomic transactions through the transaction service. Clients access server objects through *stub* images in their local address space which hides the complexities of distributed operation and allows non-network programmers to develop distributed applications. Access can be static, through pre-compiled IDL stubs, or dynamic, through the Dynamic Invocation Interface (DII). Inter-Operable references of interfaces (IORs) may be obtained through the naming service. Finally, events are disseminated through the event notification server that allows clients to specify the type of events they want to receive and also filter on their content.

CORBA was initially seen as the all-encompassing unifying management technology but, despite its success in service management, it has never been used for network management despite support by some telecommunication equipment vendors. The key problems with CORBA are the fact that it is relatively heavyweight and expensive technology. Core network devices such as switches and routers many contain more than a hundred thousand managed objects, and making each one of these a separate distribute object with its individual interface is too resource-expensive, even when considering CORBA's Portable Object Adaptor

which is relatively lightweight in comparison to the Basic Object Adaptor (BOA). There are ways round this problem, as we will discuss in Subsection 4.5, but these are proprietary as there has not been an agreed and standardized solution. Another issue with CORBA is the lack of support for bulk data transfer and, again, relevant workarounds are typically proprietary. In summary, CORBA has not been used in network management but it will continue being used in service management given the prior investment in this area, although there is an increased push towards XML/Web-based approaches and WS in particular.

4.4 Web Services

Web Services [22] is an emerging Internet-oriented technology that has strong analogies to CORBA. It is developed and standardized by the WWW Consortium (W3C) so that Web-based e-services expose standard interfaces and are accessed in an open interoperable manner. Web Services aim to put structure in Web content and associated services so that the latter are accessible by distributed applications.

Service interfaces are specified in the Web Services Description Language (WSDL), which constitutes a general XML-based framework for the description of services as communication endpoints, capable of exchanging messages. It describes the service location through a Uniform Resource Identifier (URI), the supported operations and the messages to be exchanged. Service inheritance is also supported. WSDL does not mandate a specific communication protocol but can support different protocol bindings; despite this, the default binding is usually the SOAP. WSDL can be considered as broadly equivalent to CORBA IDL. In this context, URIs are broadly equivalent to CORBA IORs. SOAP is a stateless protocol with XML-based encoding and it is broadly equivalent to CORBA GIOP. The default SOAP mapping is on HTTP/TCP/IP can be considered as equivalent to CORBA IIOP.

Service specification and interface discovery is supported through the Universal Description, Discovery and Integration (UDDI). This provides a mechanism for service providers to advertise, i.e., register, services with it in a standard form so that service consumers query services of interest and discover their location. UDDI is itself implemented as a well-known Web service in terms of interface and location. When used for service specification discovery, it is broadly equivalent to the CORBA Interface Repository, while when used for interface location discovery, it is broadly equivalent to the CORBA Naming service.

It is obvious from the above comparison that WS can be used as a distributed object technology. Some key differences to CORBA are the following. In CORBA, the default client-server coupling is tight, with the client having pre-compiled knowledge of the server's interface, which supports compile time type-checking. Web Service technology was initially conceived as message-oriented, with loose coupling between clients and servers through XML parsing and runtime checking only, in a similar fashion to the CORBA DII. On the other hand, most Web Services platforms support static coupling through stubs, albeit through proprietary mappings. CORBA supports standard language mappings for languages such as

C++, C, Java, etc., while Web Services, being an Internet technology, addresses only the interoperability through WSDL and the protocol e.g., SOAP/HTTP.

Web Services is an emerging technology behind which one can sense the same hype and expectation as that behind CORBA in the mid-1990s. Compared to CORBA, WS is ubiquitous and cheaper technology but suffers to some extent from verbose XML encodings. There have already been supporting software platforms with APIs that are similar to CORBA (stub objects etc.) but easier to use. There has been and there is still ongoing intensive work to provide sophisticated services, so it is likely that WS will become the distributed technology of choice, potentially displacing technologies such as CORBA and JRMI. The Tele Management Forum (TMF) that looks into Next Generation Operation Support Systems (NGOSS) is already considering WS in addition to CORBA and JRMI/J2EE. But the use of WS for network management will largely depend if there will be an IETF initiative for a WS-based next generation Internet management protocol. For this to happen, some of the issues of using distributed object / service interface technologies for network management will need to be resolved, as discussed next.

4.5 Distributed Object and Service Interface Technologies in Management

The use of distributed object technologies for network and service management has been a subject of intense research in the mid to late 1990s. It is now widely accepted that distributed objects are naturally suited for service and application management: service management involves mostly business process re-engineering and automation, for which technologies like CORBA or JRMI/J2EE are well suited; in addition, distributed applications are typically realized using distributed object technologies, so it makes sense to use the same technology to manage them. On the other hand, network and system management have relatively different requirements: large amounts of information need to be accessed, some of it of real-time nature, while concerted configuration changes need to be supported across devices. So the fundamental requirements of network management are: support for flexible information retrieval, both in bulk but also in a selective fashion; support for fine-grained event notifications through selective criteria; and support for transactions that involve many operations to one or more devices.

The popularity of CORBA led the Tele-Management Forum (TMF) and X/Open to setup the Joint Inter-Domain Management (JIDM) taskforce that produced a static mapping, i.e., specification translation between SNMP SMI/OSI-SM GDMO and CORBA IDL, and a dynamic mapping, i.e., interaction translation for supporting generic gateways [23]. Given these developments, CORBA has been considered for network management in telecommunication environments, with ITU-T GDMO specifications translated to IDL. But the use of CORBA, and distributed object technologies in general, presents the following problems. First, there is no support for bulk or selective data transfer; in fact, a remote method invocation per attribute is required in the default case, which can be prohibitively expensive. And second, a scalability problem may arise by making vast amounts of dynamic entities, such as connections, separate objects with their interfaces.

Because of these limitations, common practice for the use of CORBA for network management is to follow a semantic rather than JIDM-like syntactic translation of GDMO to IDL. Commonly accessed attributes of an object class are grouped together through an additional method that returns them, which alleviates the requirement of using separate per-attribute methods. In addition, dynamic entities such as connections are *not* modeled through separate objects/interfaces. Instead, a method that returns all of them as a "list of records" is added to the associated protocol interface. If such entities are static, for example semi-permanent cross-connections created and deleted through management, additional methods to create and delete them, or better to add and remove them from the current list, are added. This modeling approach supports bulk data transfer for multiple-instanced objects in a "predefined manner", relying on the underlying reliable transport. In addition, this approach does not require an excessive amount of objects/interfaces to be present in network devices.

The use of Web Services for network management presents exactly the same problems and issues as the use of CORBA. The problems of bulk data transfer and scalability can be potentially addressed as described above, but an approach for the semantic mapping of existing SNMP-SMI and GDMO specifications to WSDL needs to be standardized. In addition, it remains to consider aspects pertinent to Web Services, such as XML encoding overhead, although initial performance comparisons have been encouraging [24].

5 Related Work

Many researchers have attempted a classification and comparison of management approaches in the past, including the author of this paper. We present briefly related work here without intending to cover completely previous efforts as this would require much more space.

The first seminal work took place in the context of the JIDM activity that was undertaken in common between the TMF and X/Open and it compared the OSI-SM, CORBA and SNMP object models [23]. Subsequent work by the author further compared in depth those three technologies as potential candidates for the unifying telecommunications management technology, examining both protocols and object models and also considering implementation issues [25]. An extensive survey of distributed enterprise network and system management paradigms was presented in [26] and further elaborated in [24]. An examination of active distributed management approaches was subsequently presented in [27]. A list of network management technologies, including their strengths and weaknesses, and the network operator needs regarding the former was presented in [6]. The future of Internet management technologies was considered in [28]. An extended overview of technologies for telecommunication network management was presented in [29]. Finally, the author and his colleagues examined Web Services in comparison to SNMP and CORBA in [30]. It should be noted that the detailed presentation of key technologies in Section 4 of this paper is based on the equivalent presentation in [30].

6 Summary and Outlook

This paper examined the evolution of management technologies from a historical perspective, highlighting important design choices and explaining the how's and why's behind the various frameworks and technologies. After a presentation of all the management aproaches, frameworks technologies from a historical viewpoint, a taxonomy was attepmpted. This considered two fundamentally different management approaches: management by remote invocation and management by delegation. Starting from the latter, relevant approaches range from a. manager–agent based controlled code upload to a device to b. mobile code with a predefined itinerary to c. full mobile agents. Despite the potential power of the management by delegation approaches, there have not been management applications that exploit this power outside the research realm and, as such, these approaches are not used much in practice.

The management by remote invocation approaches fall into two key categories: manager–agent based approaches, in which operations to managed objects are always performed through the agent, and distributed object or service interface approaches, in which management operations are performed directly to the managed objects. The latter approach has seen technologies such as CORBA, JRMI and more recently WS to be considered as potential management technologies, but there has always been the issue of large managed object populations and of bulk data retrieval, as explained. These issues can be circumvented to some extent with careful well-agreed interface design and it would be interesting to see if IETF will consider a next generation Internet management protocol based on Web Services.

The manager–agent approach has resulted in technologies that have seen considerable success, most notably SNMP but also OSI-SM in telecommunication environments to a smaller extent. It would have been nice if SNMP was designed from the beginning with features that would make it a fully capable management technology, i.e., information model inheritance, TCP-based reliable transport, a create operation, transaction support and strong security. But given this is not the case, NetConf has emerged as the technology for configuration management and it should in principle see success and real world deployment when the required data models are also standardized. So SNMP will continue to be used for simple monitoring, NetConf for configuration management and vendor-specific solutions will be used for large scale network monitoring e.g., for data collection for traffic engineering. It would be nice to have a single framework to perform all these tasks and it remains to be seen if there will be a future IETF initiative towards a completely new Internet management framework, based potentially on XML and/or Web Services. It should be finally noted that OSI-SM will continue to be used for network management in telecommunication environments while technologies such as WS, JRMI/J2EE and CORBA will continue to be used for network and service management OSSs.

So in summary, most of the previous technologies will continue to be used and the dream of an end-to-end all-encompassing technology will never be fulfilled. This is not unexpected given the differing requirements of different management domains, the significant investment in current and previous approaches that can not

be neglected and the constant evolution towards new and better technologies. We should simply be prepared to deal with even more co-existing technologies in the future.

Acknowledgments The work presented in this paper was partially supported by the EU EMANICS Network of Excellence project (IST-026854).

References

1. Scott Marcus, J.: Icaros, Alice and the OSF DME, IFIP/IEEE 4th International Symposium on Integrated Network Management (ISINM), pp. 83–92. Chapman & Hall (1995)
2. ITU-T Rec. X.701.: Information Technology—Open Systems Interconnection. Systems Management Overview (1992)
3. ITU-T Rec. M.3010.: Principles for a Telecommunications Management Framework (TMN). Study Group IV (1996)
4. Warrier, U., Besaw, L., LaBarre, L., Handspicker, B.: Common Management information Services and Protocols for the Internet (CMOT & CMIP). IETF RFC 1189 (1990)
5. Case, J., Fedor, M., Schoffstall, M., Davin, J.: A Simple Network Management Protocol (SNMP). IETF RFC 1157 (1990)
6. Schoenwaelder, J.: Overview of the 2002 IAB Network Management Workshop. IETF Informational RFC 3535 (2003)
7. Chan, K., et al.: COPS Usage for Policy Provisioning (COPS-PR). IETF RFC 3084 (2001)
8. Object Management Group.: The Common Object Request Broker: Architecture and Specification (CORBA). Version 2.0 (1995)
9. Yemini, Y., Goldschmidt, G., Yemini, S.: Network Management by Delegation. IFIP/IEEE 2nd International Symposium on Integrated Network Management (ISINM), pp. 95–107. Elsevier (1991)
10. Levi, D., Schoenwaelder, J.: Definitions of Managed Objects for the Delegation of Management Scripts. IETF RFC 3165 (2001)
11. ITU-T Rec. X. 753.: Information Technology—Open Systems Interconnection, Command Sequencer for Systems Management (1998)
12. Schoenwaelder, J., Quittek, J., Kappler, C.: Building distributed management applications with the IETF script MIB. IEEE J. Select. Areas Commun. (JSAC). **18**(5), 702–714 (2000)
13. Vassila, N., Pavlou, G., Knight, G.: Active objects in TMN. IFIP/IEEE 5th Integrated Network Management Symposium (IM), pp. 139–150. Chapman & Hall (1997)
14. Bieszczad, A., Pagurek, B., White, T.: Mobile agents for network management. IEEE Communication Surveys and Tutorials **1**(1), (1998)
15. Bohoris, C., Liotta, A., Pavlou, G.: Evaluation of Constrained Mobility for Programmability in Network Management. IEEE/IFIP 11th International Workshop on Distributed Systems: Operations and Management (DSOM), pp. 243–257. Springer (2000)
16. Distributed Management Task Force.: Web Based Enterprise Management (WBEM), http://www.dmtf.org/standards/wbem/
17. Distributed Management Task Force.: Common Information Model (CIM), http://www.dmtf.org/standards/cim/
18. ITU-T Rec. M.3100.: Generic Network Information Model. Study Group IV, (1996)
19. Wahl, M., Howes, T., Kille, S.: Lightweight Directory Access Protocol v3 (LDAP). IETF RFC 2251 (1997)
20. Enns, R. (ed.): NetConf Configuration Protocol. IETF RFC 4741 (2006)
21. ITU-T Rec. X.900.: Information Technology—Open Distributed Processing. Basic Reference Model of Open Distributed Processing (ODP) (1995)
22. World Wide Web Consortium (W3C).: Web Services Activity Documents, http://www.w3c.org/2002/ws
23. Rutt, T. (ed.).: Comparison of the OSI Systems Management, OMG and Internet Management Object Models. Report of the NMF—X/Open Joint Inter-Domain Management (JIDM) task force (1994)
24. Martin-Flatin, J.P.: Web-based Management of IP Networks and Systems. Wiley (2003)

25. Pavlou, G.: OSI system management, internet SNMP and ODP/OMG CORBA as technologies for telecommunications network management. In: Aidarous, S., Plevyak, T. (eds.) Book chapter in Telecommunications Network Management: Technologies and Implementations, pp. 63–109. IEEE Press (1998)
26. Martin-Flatin, J.P., Znaty, S., Hubaux, J.P.: A Survey of distributed enterprise network and systems management paradigms. J. Network Syst. Manage. (JNSM). **7**(1), 9–26 (1999)
27. Kawamura, R., Stadler, R.: Active distributed management for IP networks. IEEE Commun. **38**(4), 114–120 (2000)
28. Schoenwaelder, J., Pras, A., Martin-Flatin, J.P.: On the future of internet management technologies. IEEE Commun. **41**(10), 90–97 (2003)
29. Boutaba, R., Xiao, J.: Telecommunications network management. In: Bellavista, P. (ed.) Telecommunications Technologies, Systems and Services, Encyclopedia of Life Support Systems, (2007)
30. Pavlou, G., Flegkas, P., Gouveris, S., Liotta, A.: On management technologies and the potential of web services. IEEE Commun. **42**(7), 58–66 (2004)

Author Biography

George Pavlou is a Professor of Communication and Information Systems at the Center for Communication Systems Research, Department of Electronic Engineering, University of Surrey, United Kingdom, where he leads the activities of the Networks Research Group. He holds a Diploma in Engineering from the National Technical University of Athens, Greece, and M.Sc. and Ph.D. degrees in Computer Science from University College London, United Kingdom. In January 2008 he will be joining again the Department of Electrical and Electronic Engineering at University College London, United Kingdom, as a Professor of Communication Networks. His research interests focus on network management, networking, and service engineering, covering aspects such protocol performance evaluation, traffic engineering, quality of service management, policy-based systems, multimedia service control, programmable networks, and communications middleware.