Chapter 2

RELATED WORK

2.1 NETWORK AND SYSTEM MANAGEMENT

Modern day telecommunication industry is facing ever increasing demand for more sophisticated services, higher quality and shorter times to market. Network operators and service providers in the telecommunications industry must meet these demands at a commercially viable cost and should have plans in place for any unforeseen changes, especially, in a complicated environment that distributed, often mobile, data, resources, service access and control [11]. Moreover these networks are growing in size and complexity. In-fact, varied technologies, such as SONET, ATM, Ethernet, DWDM etc., present at different layers of the Access, Metro and Core (long haul) sections of the network, have contributed to the complexity in terms of their own framing and protocol structures. Thus, controlling and managing the traffic in these networks is a challenging task.

Network management (NM) [12] is a fundamental facilitator that typically gathers and analyzes huge amount of data from the network and makes decisions thereof for various functional components of a management system. The primary objective of NM [30] is to maintain network and systems availability and health, aid in configuring the network and systems, guarantee Quality of Service (QoS), enhance performance, provide security, minimise operational overhead (execution of repetitive tasks) and decrease the cost of running the information technology infrastructure.

As a consequence of sharing and interconnecting resources, NM needs to meet the challenges of distribution, heterogeneity and transparency. A number of approaches and architectures that aim at standardising the management process and addressing the heterogeneity problem as well have, therefore, emerged. These approaches specify management architectures, which supply frameworks for standards of relevance to NM. Two of the most widely used architectures are the Internet management architecture
which is also know as SNMP Management [13] and the Open Systems Interconnection Systems Management (OSI-SM) [14] architecture. Other architectures, mainly applied on the telecommunication networks area include the Telecommunications Management Network (TMN) [15] and the Telecommunications Intelligent Network Architecture (TINA) [16]. More recent efforts focus on the definition of emerging management architectures, such as Web-Based Enterprise Management (WBEM) [17], Java-based management [18]. Distributed objects technologies, exemplified by the Common Object Request Broker Architecture (CORBA) [19], represent another interesting approach which gains increasing attention in the management world.

The Internet management architecture has been criticised for exhibiting low degree of scalability, flexibility and re-configurability, mainly attributed to its centralised architecture. The latter two weaknesses also characterise the OSI-SM framework. The emerging management technologies have only partially addressed these problems. The need for distribution of NM functionality has been early recognised by researchers and developers active in this area and several initiatives have been undertaken in this direction. In fact some of them have already led to the specification of standards. However, there is still a long way to go before NM distribution-related problems are satisfactorily addressed.

2.2 MANAGEMENT FUNCTIONAL AREAS

Management activity have been categorised by ISO into five generic functional areas, collectively known as FCAPS from their initials [20, 30], Fault Management, Configuration Management, Accounting Management, Performance Management and Security Management. A brief discussion on these functional areas is given below.

**Fault Management:** the process of collecting information referring to network elements (NE) health. Integrated fault management systems receive reports about malfunctions (alarms), perform alarm correlation and diagnostic tests, identify faults and display various network alarms. This process can be optimised to perform root-cause analysis and suggest/take corrective measures.
**Configuration Management**: controls the configuration state of a system/network and the relationships between components. It also initialises, configures and shuts down network equipment.

**Accounting Management**: defines how network usage, charges and costs are to be identified in the networking environment. It is associated with tariffing schemes that generate charging/billing information.

**Performance Management**: supports the gathering of statistical data, upon which it applies various analysis routines to measure the system performance. That way, it provides an accurate picture of network components and services. This process is capable of proactively pinpointing and forecasting potential problems before they actually occur, based on gathered information. It can predict congestion/bottlenecks and, hence, be used for network future expansions and capacity planning. Performance management represents a central application area for this thesis.

**Security Management**: controls access to network, system service and management components. It can offer authentication, confidentiality, integrity, access control and also handle cryptographic key distribution.

In the telecoms world, management platforms generally support most (if not all) of the OSI functional areas. This is not the case in the IP world, where most platforms support only a fraction of FCAPS. Indeed, management platforms are often simpler in the IP world than their counterparts in the telecom world.

### 2.3 OSI SYSTEM MANAGEMENT

The OSI-SM [20, 30] defines management architecture with well-defined organisational, informational, communication and functional models. The organisational model assigns special roles to the management entities: the manager and the agent. A manager is an entity that controls the management process and makes decisions based on collected information, whereas the agents make available the management information to managers. Abstractions of system/network resources which need to be managed are represented by managed objects (MO). MOs encapsulate the underlying real resources
and enable their manipulation through well-defined operations. The basic architecture of OSI-SM is shown in Figure 2.1.

![Diagram of OSI-SM architecture](image)

**Figure 2.1: Basic OSI-SM architecture.**

An agent administers the MOs on its local device and provides mechanisms for performing management operations upon them, offering an interface to system resources. In essence, the agent acts as name server for the objects (resolves their names to internal handles), object factory since it creates and maintains objects and event server as it disseminates events (notifications are evaluated and forwarded as events to managers according to criteria preset by them) [21].

The communication between the manager and the agents takes place via standardised management protocols. In general, the manager-agent paradigm can be thought of as client/server (CS) relationship, where the manager plays the role of the client and the agent the role of the server. OSI-SM is based on a complex, object-oriented information model. Wherein MO classes are specified by templates and consist of attributes, operations that can be applied to the corresponding objects and the behaviours exhibited by the objects in response to operations, and notifications that can be emitted by the
objects. The functionality of a MO is defined at design time, i.e., it cannot change at runtime. MOs are logically grouped in Management Information Bases (MIB) as shown in Figure 2.2. MIBs are virtual, hierarchical, object-oriented databases including interrelated MOs in a managed environment.

![Figure 2.2: Hierarchical arrangement of MOs in a MIB tree](image)

The exchange of management information between managers and agents is defined by a service, the Common Management Information Service (CMIS), and its protocol, the Common Management Information Protocol (CMIP). The CMIS provides management operation primitives that include M-GET to retrieve data, M-SET to modify data, M-ACTION to request the execution of an action, M-CREATE (M-DELETE) to request the creation (deletion) of an instance of a managed object, and M-CANCEL-GET to cancel an outstanding M-GET request. Agents may report events about managed objects using M-EVENT-REPORT. In addition, CMIS provides multiple-object access through scoping and filtering operations. Scoping allows management operations to be carried out on a selection of one or more managed objects. Filtering consists of boolean expressions with assertions on values of attributes in an object.
OSI-SM also addresses some aspects of management decentralisation through the standardised Systems Management Functions (SMF), which define a rich set of functionality specified in terms of generic object classes. Examples include Metric Objects, which measure resource performance, monitor thresholds and generate notifications and the Summarization function, which provides a framework for the definition, generation, and scheduling of system information summary reports. These functions move intelligence in proximity to the managed resources, reducing the amount of management traffic and providing support for a sophisticated event-driven operation paradigm.

The widespread interest in formal standards has generated considerable interest in CMIP, even though it has not been widely used. One factor contributing to the lack of CMIP’s popularity is the slow evolutionary process of these standards.

2.4 INTERNET MANAGEMENT

Simplicity and small implementation overhead has always been the main objective of Internet management since the early days of its conception [22, 23]. This seems to be the main reason that justifies its popularity and wide use. In this context, Internet management is characterised by a simple information model that lacks the object-orientation (MOs are nothing more than simple variables) and sophistication of its OSI-SM counterpart, but enables easier and faster writing and instrumentation of MIBs. The architecture shown in Figure 2.3 is applied in the Internet management model as well; the difference is that the functionality of the corresponding modules is very much simplified. The Internet management communication model relies on the Simple Network Management Protocol (SNMP) as the communication protocol, which defines connectionless services and primitives for getting and setting variable values and sending notifications. SNMP has been developed with an orientation to TCP/IP networks. As it is the case with most protocols of this kind, its development and implementation occurred with considerable speed. A quick, easy and simple implementation was the first priority of its designers. Hence, the following guidelines have been adhered to:
make it work over very uncomplicated protocols
keep the number of protocol message types small
stick to a unit of information that is a single value, such as an integer or string

Figure 2.3: The SNMP Layering

The User Datagram Protocol (UDP) was chosen as the SNMP transport protocol. That decision was made mainly due to the scalability reasons. Namely, being centralised, SNMP would impose a huge demand on the manager platform system resources to be able to accommodate many open TCP connections to the managed devices. UDP also has small footprint on network resources compared to TCP, while being well suited for short request/response type of operations, which is consistent with the connectionless nature of SNMP. In general, the SNMP framework provides much poorer functionality and expressiveness than CMIP. However, because of the overall complexity and size of CMIP, many claim that this is a case of the cure’s being worse than the disease. These problems and most importantly the domination of Internet over OSI have prevented CMIP from reaching the dominant market position that was originally anticipated. On the other hand, the inherent simplicity of SNMP has been the driving force for its wide acceptance and popularity.
2.4.1 SNMP Protocol Data Units

SNMP uses relatively simple operations and a limited number of Packet Data Units (PDU) to perform its functions. Figure 2.4 shows the types of messages exchanged between the manager and an agent. Five PDUs have been defined in the first version of the standard (SNMPv1):

- **Get Request**: it is used to access the agent and obtain managed objects values. It includes identifiers to distinguish it from multiple requests.
- **Get-Next Request**: it is similar to the Get Request and permits the retrieval of the next logical identifier in a MIB tree.
- **Set Request**: it is used to change the value of a MIB object.
- **Response**: it responds to the Get, Get-Next and Set Request PDUs.
- **Trap**: it allows SNMP agents to report events at their local NE or to change the status of the NE.

![PDU Type Request ID 0 0 Varbind List](a)

![PDU Type Request ID Error status Error index Varbind List](b)

![PDU Type Request ID Non repeaters Max receptions Varbind List](c)

<table>
<thead>
<tr>
<th>Name1</th>
<th>Value1</th>
<th>Name2</th>
<th>Value2</th>
<th>---------</th>
<th>Namen</th>
<th>valuen</th>
</tr>
</thead>
</table>

**Figure 2.4**: SNMP PDU formats: (a) Get request, Get-Next request, Set request, Trap, Inform, (b) Response, (c) GetBulk request, (d) varbind list
Management data are returned in a list structured as a sequence of <object ID: value> pairs, termed the varbind list. As shown in Figure 2.4, SNMP request and response messages have for simplicity reasons the same packet format. Later protocol versions (SNMPv2, SNMPv3), define two additional operations [24]:

- **GetBulk Request**: it has been devised to minimise the number of protocol exchanges required to retrieve large volumes of management data, although there is a maximum PDU size limitation. It includes a field that specifies the number of variables in the varbind list for which a single lexicographic successor is to be returned (non-repeaters) and another field denoting the number of lexicographic successors to be returned for the remaining variables (max-repetitions).

- **Inform Request**: this PDU is used for manager-to-manager communication, i.e. it is sent by an entity playing the manager role on behalf of an application, to another entity playing the manager role, to provide information to an application using the latter entity.

In Figure 2.4, the PDU type for the messages is application data type, which is defined in RFC 1157 as:

<table>
<thead>
<tr>
<th>PDU Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get-request</td>
<td>[0]</td>
</tr>
<tr>
<td>Get-next-request</td>
<td>[1]</td>
</tr>
<tr>
<td>Set-request</td>
<td>[2]</td>
</tr>
<tr>
<td>Get-response</td>
<td>[3]</td>
</tr>
<tr>
<td>Trap</td>
<td>[4]</td>
</tr>
<tr>
<td>Get-bulk-request</td>
<td>[5]</td>
</tr>
<tr>
<td>Inform-request</td>
<td>[6]</td>
</tr>
</tbody>
</table>

RequestID is used to track message with the expected response or indicate loss of the message. A nonzero **ErrorStaus** is used to indicate that an error has occurred. The convention is not to use 0 if no error is detected. **ErrorIndex** is used to provide additional information on the error status. The value is filled with NULL in case where it is not applicable, such as get-request data PDU. A managed object is a scalar variable and is simply called a variable. Associated with the variable is its value. The pairing of the variable and value is called variable binding or **VarBind**. The data PDU in the message contains VarBind pair.
2.4.2 History of SNMPv1, SNMPv2 & SNMPv3 [30]

Considering and comparing with OSI management, SNMPv1 is lightweight in design and it avoids or bypasses the four-year standardisation cycles of the ITU-T [25]. These were the chief reasons for the inaugural success of SNMPv1. Although, in reality, SNMP has evolved at a gradual rate. One of the reasons for its decline has also been the remarkable accomplishment and popularization of the SNMPv1 architecture. It was fine for handling relatively smaller networks however, scaling to vast networks (e.g. geographically spread organizations) was a challenge as it could not appropriately deal with the large volumes of management data. The telecommunications industry had demonstrated that in order to handle this challenge we can allocate the load or use-load balancing through a hierarchical structure of managers. But surprisingly, until late 90s the Internet Engineering Task Force (IETF) did not prioritise this distribution of management load. After SNMPv1 was standardised in 1990, four management architectures have been created and delivered till date: SNMPv2p, SNMPv2u, SNMPv2c, and SNMPv3.

Only central management was supported in SNMPv2p, SNMPv2u and SNMPv2c and then the IETF discarded SNMPv2p as obsolete in 1996 [26]. SNMPv2u was not well adopted as it had “no compelling commercial offerings” and hence is not in use any more. SNMPv2c has 64-bit counters and better error handling than SNMPv1. Hence it is used to manage highly utilized backbone routers in some cases. However it does not offer anything new on account of distribution of load. On the contrary, SNMPv3 is more focused on security rather than the scalability. As a result, IETF spent eight years to deliver a major release called SNMPv3. It took subsequent two years for major vendors to start supporting it. Thus its utilization in production environments is forecasted to be marginal in near future timescales. Another aspect to mark is that it was only in 1999 [27], when the MIBs adding support for one kind of delegation in SNMPv3 were issued, so it might consume some more duration before they are implemented and deployed. In essence, telecommunication vendors working on SNMP management platforms are today compelled to develop proprietary extensions in order to support hierarchies of managers.
2.4.3 Strengths/Limitation of SNMP

According to the main characteristics of SNMP described in the previous section, the main strengths and weaknesses [30] of SNMP-based management are given below:

- Interoperability
- Simplicity
- wide support by IP-equipment vendors
- Small footprint on agents

On the other hand, SNMP also exhibits following weaknesses:

- Scalability
- Unreliability
- Weak Security
- Low level of semantics

A brief discussion on above weaknesses is given below.

i. Scalability This issue can be classified into the following categories:

   **Network overhead:** In the context of network management, network overhead is the proportion of a link capacity used to transfer management data, and thus unavailable for user data. The purpose of a network is to transfer user data, not management data, so an important goal of network management is to keep network overhead low. SNMP is characterised by high network overhead, which is mainly due to the polling-based nature of monitoring and data collection process [28]. The manager repeatedly requests and retrieves specific MIB object values at each poll cycle from remote agents. In many monitoring applications, a considerable portion of network bandwidth is typically wasted to learn nothing other than that the network is operating within acceptable parametrical boundary conditions.
Latency: For polling, latency is the time elapsed between the moment the manager requests the value of a MIB variable and the time it receives it from the agent. It is important to keep latency reasonably low, so as to quickly detect and correct operational problems. End-to-end latency depends on networking conditions (the capacity and error rates of links, the speed of the IP routers traversed between the agent and the manager, etc), the amount of retrieved data and the number of protocol exchanges. When large sets of NEs need to be managed through SNMP, latency can very be high, especially when the ‘control loop’, i.e. the network distance, between the managing and the managed entities is large.

Manager’s processing capacity: The manager’s hardware resources (CPU, memory, etc.) dedicated to management applications cannot be continuously increased, due to cost and hardware constraints, setting a limit on manager processing capability. The centralised structure of SNMP architecture and the lack of data filtering capability characterising SNMP agents result in transferring vast amounts of network management data, subsequently processed at the manager platform. This forces manager’s processing capacity to its limits and intensifies the need to relieve the manager from performing routine data processing tasks.

Capacity of the manager’s local segment: The management data sent by all the agents converge toward a single point, the network segment where the manager is connected to, inevitably creating a bottleneck [29].

Inefficient bulk management transfers: As the amount of data to transfer grows, it makes sense to reduce the overhead by sending the data in bulk, that is, to send unlimited number of MIB variables at a time, while keeping the number of network interactions low. However, SNMP has not been designed for transfers of bulk management data (typically stored in SNMP tables). In SNMPv1, tables are retrieved through successive get-next operations. If the table includes many rows, the manager must perform at least one get-next per row. For tables with hundreds or thousands of rows, an equal number of requests/responses will be
transferred through the network increasing the network overhead and latency. The situation improves with the get-bulk operator offered by SNMPv2c and SNMPv3 frameworks, which allows transferring more data per SNMP message. However, the manager should guess the length of the table to be retrieved and accordingly choose a value for the max-repetitions parameter. Using a low value will cause more PDU exchanges than necessary. Using a high value, however, can result in an overshoot effect [31]: the agent can return data of no interest for the manager.

**SNMP message maximum size:** The large number of protocol exchanges required to complete bulk management data transfers is mainly due to the maximum size limit of SNMP messages. All SNMP agents must accept SNMP messages that are up to 484 bytes in length, but may legally refuse longer messages. Yet, many open-source implementations of SNMP have used the maximum size limit of 1472 bytes (in LAN environments), proposed in. Clearly, when transferring data in the order of Mb, a large number of PDUs will be exchanged; the exact number will depend on the maximum size limit used in the particular SNMP implementation.

**OID naming scheme:** This relates to the information model of SNMP-based management, i.e. the naming conventions for MIB variables. The Object Identifiers (OID) transferred in SNMP messages exhibit a high degree of redundancy. For instance, all objects stored in MIB-II are prefixed with 1.3.6.1.2.1. If this prefix could be omitted, a significant proportion of the space dedicated to the OID name would be saved. Furthermore, the prefixes of the table object OIDs are all identical up to the column number. In this case, more than 90% of the OID name is redundant. All these observations indicate a highly inefficient OID naming scheme.

**No compression of management data:** The first two versions of SNMP (v1 & v2c) did not allow the transparent compression of management data in transit. This unnecessarily increases network overhead and also network latency due to transferring larger volumes of data. As of SNMPv3, it is possible to compress
management data by adding encryption envelopes to SNMP messages. Although this feature was initially intended for encrypting data, it also allows for data compression. When large chunks of data are compressed, the overall latency is also reduced, as the compression time is typically negligible with respect to the time saved to transmit the uncompressed data.

ii. Unreliable transport protocol

Another problem of SNMP-based management is the transport protocol used for transferring SNMP messages, UDP. UDP operates in a connectionless fashion, which saves the three-way-handshake overhead of TCP, and is ideal for exchanging short messages. However, due to the lack of acknowledgements, it is not suitable for communicating critical notifications to the manager. By using an unreliable transport protocol, the management system runs the risk of losing important notifications for trivial reasons such as buffer overflows in IP routers [32].

iii. Security

SNMPv1 and v2c adopt a weak security scheme. Passwords for configuring routers, hubs and servers are passed across the network as clear text, in unprotected packets. Identification, which is based on community strings, is so simplistic that cannot be considered as secure. The main advancement brought by SNMPv3 is security. SNMPv3 supports identification, authentication, encryption, integrity, access control, etc. However, the support of SNMPv3 by major vendors takes place at a slow pace.

iv. Information model: low level of semantics

As far as semantic richness is concerned, the main shortcomings in SNMP are the absence of high-level MIBs, the limited set of SNMP protocol primitives, and the data-oriented nature of the SNMP information model. Due to these limitations, developing high-level management applications is a difficult task, which partly explains why network management applications are often limited to little more than monitoring in the
In particular, due to the way the SNMP market evolved over time, SNMP MIBs offer only low-level Application Programming Interfaces (API), often called instrumentation MIBs. SNMP frameworks provide no support for management applications to dynamically define external data models as part of the MIBs. Although it is possible for applications to retrieve raw MIB data and compute the appropriate data model at the platform host, this is highly inefficient. In addition, SNMP protocols support limited protocol primitives vocabulary, which allows getting/setting atomic variable values and notifying about important events. This is extremely restrictive and is typical of a data-oriented information model, unlike object-oriented models, which are widely used in industry today. The absence of an object-oriented information model in SNMP is generally regarded as one of the main limitations of SNMP. When the Distributed Management Task Force (DMTF) endeavoured to define a new management architecture in the late 90s, it came as no surprise that its first delivery was a new object-oriented information model: the Common Information Model (CIM) [34].

The scalability problems of SNMP are mainly attributed to its centralised model and can be efficiently addressed through MA-based approaches. The latter enable the dynamic delegation of network management functionality to managed elements, where MA objects may filter/correlate management data, adopt an event-driven (instead of polling-based) approach to notify managers about important events and apply data compression, thereby reducing network overhead.

2.5 DISTRIBUTED OBJECT BASED MANAGEMENT

In ’90s, Distributed Objects Technologies (DOT) surfaced as a new programming model. DOTs outlined and explained an object-oriented approach, where objects could interact in spite of not residing on the same system. It became simpler to create integration of DOTs with existing network management architectures because OSI-SM is object-oriented and SNMP managed objects can be mapped onto objects. With this, DOTs facilitated a major swing from the protocol-based styles, embodied by SNMP and OSI-SM, to the distributed object-based styles. Here, two typical and illustrative DOTs are described and their bearing on Network Management Architectures are briefly defined:
2.5.1. CORBA

CORBA [19] is the product of a consortium of over eight hundred companies, known as the Object Management Group (OMG). The OMG has approached the problem of handling the interaction of distributed components by creating interface specifications. Distributed components of the system are able to describe their interfaces using the Interface Definition Language (IDL) and subsequently inter-operate through the underlying Object Request Broker (ORB). Namely, the ORB provides the communication backbone through which distributed components are able to interact. To perform requests or return replies, objects use a generic RPC-like request/response protocol, the General Inter-ORB Protocol (GIOP) or its TCP/IP mapping, the Internet Inter-ORB Protocol (IIOP). Distributed components communicating via an ORB do not need to be aware of the mechanisms used in that communication and are able to discover each other at run time. A number of CORBA services provide basic functions, e.g. the Naming service that allows clients to locate objects based on their names, the Trading service that enables objects location based on their properties and the Event service which allows asynchronous messaging between objects. The applicability of distributed objects on network management has been a subject of intense research in the past few years. In that context, Mazumdar [35] proposed the use of a gateway, which achieves the inter-operation of management applications in CORBA domain and agents in SNMP domain. The main function of the CORBA/SNMP gateway is to dynamically convert method invocations on object references in CORBA domain to SNMP messages for MIB entries at remote agents. A typical architecture deploying CORBA/SNMP gateway is shown in Figure 2.5.

All of the management applications (Fault Management, Configuration Management, Service Management etc.) operate in the CORBA domain as distributed objects and their
interaction with SNMP agents residing on network elements takes place using SNMP protocol. Likewise, Pavlou [36] pointed out the suitability of DOTs in large scale distributed environments and proposed the use of CORBA in TMN open interoperable interfaces, replacing OSI-SM. However, a main direction of the research efforts has been on the seamless integration of legacy systems into emerging distributed object environments.

![Figure 2.5: CORBA/SNMP Gateway](image)

**2.5.2. Java-Based Management**

In the field of Network Management, Java is believed to be a technology instead of a programming language. It is a fresh approach that arose in the 90’s. Since the initial days, Java appeared to be suitable for developing management applications and commercial deployment of the SNMP stack in Java was first announced in 1996 [37]. Post that, many
organizations have made a great headway in this field. Adventnet [38] (now known as ZOHO corporation) is considered as a specialist in the field of Java-based management products has made great progress and provided a comprehensive pack of Internet management tools. This comprises of: a visual builder tool that can be utilized to develop SNMP management Java applets and applications, an Agent Toolkit which automates the activity of constructing SNMP MIBs and instrumenting agents for these MIBs, etc.

In the below segment, summarization of the recently evolving Java-based management technologies is attempted which are all initiatives taken up by Sun Microsystems. They consist of toolkits (JMAPI and JMDK) or standards (JMX) specifically focused on developing management frameworks.

2.5.2.1 JMAPI - JDMK – JMX

Java Management API (JMAPI) [39] was made public by Sun Microsystems in 1996, just following the release of JDK 1.1 which added support for RMI. The Java Management API (JMAPI) is primarily a group of tools and guides that can be utilized to develop management applets supporting RMI. By plotting the managed objects onto Java objects, this API supports MIB-II [23] which is the most widespread SNMP MIB. Not just this, it also simplifies the creation of sophisticated and complex GUIs by providing a splendid graphics library and eventually allowing the visual portrayal of management information. Post this, a Java-based component-oriented management toolkit was created and released by Sun. It was called the Java Dynamic Management Kit (JDMK) [40] and similar to JMAPI; JDMK was also made available to the public. Java Beans is the basis of JDMK and is accompanied with a collection of core management services. In addition, it consists of adapters to facilitate communication via RMI, HTTP and SNMP. While JMAPI only caters to MIB-II, in JDMK, an SNMP-to-Java MIB compiler is also included. This compiler converts the managed objects outlined in any SNMP MIB into JB components which are known as management beans, or MBeans. This toolkit presents a strong and effective framework for creating management applications and supports push and pull from agents.
Then Sun released Java Management eXtensions (JMX) [41] which is a management framework envisioned for object-oriented Web-based management. In 1999, JMX superseded JMAPI. Built on the learning’s of JDMK that Sun had gained, JMX is much more thorough and all inclusive than JMAPI. Unlike JDMK, JMX also provides specifications of the manager part, rather than just focusing on the agent part. However, JDMK and JMX are comparatively newer technologies and hence through study and performance evaluation are required but for all practical purposes, JDMK can be deemed as an integral portion of JMX.

2.6 EMERGENCE OF DISTRIBUTED MANAGEMENT

Management world is faced with a serious challenge today. Protocol-based tactics, demonstrated by the SNMP and OSI-SM have been prevalent. Centralized architecture has a huge blow on the scalability of management because it dumps almost the entire computational load over to the manager platform. However, IETF as well as OSI methodologies exemplify inflexible manager-agent model. Very low-level operations are offered for accessing MIBs. For example, in SNMP, it is possible for the manager to only get and set atomic values in a MIB. Production of intense traffic and processing bottlenecks is the obvious result of this fine grained CS integration which is often known as micro-management. In OSI-SM, only QoS alarms or summarised reports are sent to the higher-level managers, which can be stated as delegation of monitoring tasks to the NEs. Yet, before this generic functionality is eventually deployed to NEs, it should be first researched, standardised and implemented. Even though, this may usually take a lot of time. Whenever any modification is to be done, this cycle of research, then standardisation, then implementation and then deployment should be imitated. As the size of managed network increases, the scalability challenge inherent to centralised architectures becomes more acute. Fundamentally because the managers not just have to communicate with a huge number of elements but also stock and process always incrementing data volumes. As a result, what we get is low performance and requirement of having costlier hardware for the manager platforms. Moreover, the network segments around the manager stations get overloaded with the mixture of data generated by the management platforms and the devices. This challenge of centralised architecture is
further aggrieved and evident when the management intervention is utmost important due to heavy congestion in the network. It creates a vicious circle as during this time: (i) a manager will typically increases its checking/probing with the managed and might as well download changes made in the configuration, thus congestion increases, (ii) it becomes very challenging and slow (at times even unfeasible) to access the devices in the congested network and (iii) because congestion is an abnormal condition, it might generate more notifications to the manager, transmitting all the more traffic on to the network. In summary, to be able to tackle the challenge opposed by centralized architecture, managers should delegate the complex diagnosing and information gathering tasks to the managed devices.

Distributed management has multiple benefits which are well highlighted in many research papers (e.g.[32, 42, 43]). In specific, management distribution:

- Permit applications to effectively utilize the enhanced availability of hardware resources;
- Self-sufficiency and survivability of NMSs is enhanced. This means, whenever the contact with managing process is lost, distributed management elements can continue to work;
- Reduces the need for intensive polling;
- Results in substantial drop of management data, because management communication is majorly achieved at a local level.

Having said this, it is worth highlighting that not necessarily all management applications must be decentralised. On the contrary, centralisation is a suitable model where applications have less inherent requirement of distributed control. Characteristics of these applications are: (a) no requirement for recurrent polling of MIB deltas, i.e. aggregation functions, (b) have better bandwidth links between the manager to managed devices, (c) switch comparatively smaller volumes of data, and (d) no requirement for recurrent, semantically heavy exchanges between manager stations and managed nodes [44]. A brief comparison between centralized vs. distributed model is shown below in table 2.1.
Table 2.1 Management Centralised Vs Distributed Paradigm

<table>
<thead>
<tr>
<th>Serial no.</th>
<th>Centralised Paradigm</th>
<th>Distributed Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (Processing of Data)</td>
<td>Low need for distributed processing, e.g. a small network that can be managed by a centralised control system.</td>
<td>High need for distributed processing, e.g. localised Information processing required either due to data size or for robustness.</td>
</tr>
<tr>
<td>2. (Polling)</td>
<td>Low frequency for required polling, e.g. the NMS does not require constant polling or the network has high bandwidth availability.</td>
<td>High frequency for required polling, e.g. need for constant monitoring of a large number of managed objects.</td>
</tr>
<tr>
<td>3. (Throughput to information ratio)</td>
<td>High ratio of throughput to the amount of management information.</td>
<td>Low ratio of throughput to the amount of management information.</td>
</tr>
<tr>
<td>4. (Need for semantically rich/frequent communication, e.g. networks that support simple services.)</td>
<td>Low need for semantically rich/frequent communication, e.g. networks that support simple services.</td>
<td>High need for semantically rich/frequent communication, e.g. large networks that support complex services.</td>
</tr>
</tbody>
</table>

2.7 DISTRIBUTION ADAPTATION BY INTERNET MANAGEMENT TASK FORCE

The shortcomings or difficulties highlighted in the previous sections were first confessed by the organization which initiated it in the first place. IETF took the below mentioned initiatives around decentralisation with a focus over management of IP based networks. For additional information related to comprehensive surveys on the various approaches of management distribution may refer to [32].
2.7.1 Management Distributed within the SNMP Frameworks

Asynchronous notification mechanism provides a way that can be called as primitive way of decentralisation that SNMPv1 supported. For example, although not as an outcome of a request, but it was however possible for the SNMP agents to shoot traps towards the manager platform whenever some critical incident takes place. Still the limitation was that any management action could not be taken locally by the agents because decisions were taken by the central manager application. In SNMPv2 there was a new introduction which supported the movement from centralised architecture to the hierarchical management architecture. This new introduction was the concept of proxy agent [45], as shown in Figure 2.6.

![Figure 2.6: SNMPv2 Proxy Agent](image)

```
(a) Centralised Management
(b) Hierarchical Management
```

Conventionally, proxy agents have been used as a pass-through function in SNMP, whereby manager requests as well as agent responses are passed through the proxy and that too in a very transparent mode. The manager does not straightaway talk to these
devices but refers the requests to the proxy. And one proxy agent may be accountable for a group of such devices.

SNMPv2 has established that for a hierarchical or tiered management, a proxy may be utilized as an intermediate manager. Though, this architecture does not provide any procedure whereby the managers can allocate tasks to the intermediate managers or can speak with them while these tasks are being executed. Through the theory of inform PDU primitive, promotion of management distribution has also been attempted by SNMPv2.

2.7.2. Remote Monitoring

Remote MONitoring (RMON)[46] is an alternate methodology suggested by the IETF, which presents a greater degree of decentralisation. Here, a concept of monitors or probes is introduced which act as network monitoring systems. These systems can be separate devices that are devoted for link monitoring or can be embedded onto the network devices but RMON pretty much assumes the presence network monitoring systems, as shown in Figure 2.7.

![Figure 2.7: RMON Architecture](image-url)
For exchange of information between the manager and agents working on the probes, SNMP is used. Probes analyse the headers and monitors the packet traffic, thereby providing information regarding the following: connections between the stations, situation of nodes in the network, traffic patterns and the status of the links. In RMON, proper filters are defined and monitoring function is delegated from the managers on to the probes. As such, abnormal behaviours, failure detection and identification of significant events can be done by the probes even though it might not be in touch with the management station. Not only this, semantic compression of information by pre-processing the gathered data can be done by the agent running on probes before it is sent to the management station.

Having said this, there are multiple shortcomings that RMON presents:

- Runtime modifications are not possible because the control operations of a RMON probe can only be set or changed at the time of configuration.
- While managing multiple network segments, the cost is significantly high because typically, for each network segment, a stand-alone RMON compliant device (probe) is essential for monitoring the traffic activities.
- Unlike centralised model that offers device-related data, RMON is only capable of giving traffic-related statistics. Primarily because here, the situation of network is decided by directly inspecting the packets flowing in it as opposed to inspecting the device status.

To conclude, it is not appropriate to use RMON while management operations have to be applicable to system as well as network level together.

2.7.3. Script MIB

In 1999, SNMPv3 architecture was aided with management distribution support and in order to create a framework where a chief manager can assign control to other distributed management stations, a specific DISMAN (DIStributed MANagement) Working Group of the IETF was chartered. DISMAN framework offers methodologies to distribute scripts to remote devices that perform random management tasks. Script MIB [47] helps
in achieving this result where a standard MIB is defined to delegate and invoke management functions (Scripts), and is based on the Internet management framework as shown in Figure 2.8. However, as per the specifications, a script is a very broad reference and is interpreted as some kind of executable program or code that any device implementing MIB can execute.

![Figure 2.8: The Script MIB Approach](image)

Following abilities are provided by Script MIB in particular:

- Management scripts are transferred to the distributed managers;
- Transfer of outcomes obtained by executing management scripts;
- Execution of management scripts are monitored and controlled;
- Management scripts can be initiated, terminated, suspended or resumed;
- Arguments can be transferred for management scripts.
Prior to delegating a script, an administrator should first examine the languages that a particular deployment of Script MIB supports and then appropriately choose a script from the repository. The Script MIB is restricted to the operations executed on six tables which actually compose the MIB. Client pull or server push models are used to upload the Scripts to the devices. Using SNMP management framework, the Script MIB enables various scripts to be terminated, initiated or controlled. For example, a solitary SNMP set request can initiate the script execution. Intermediary or finalised results that are produced or retained at the agents can also be obtained by the manager. Even though a strong management distribution mechanism is provided by the Script MIB, it also displays various shortcomings:

- It is tough and problematic to update prevailing scripts due to the current specifications;
- The approach of Script MIB is particular to Internet management frameworks, just like other standardisation models taken up by IETF or the context of OSI-SM.

In contrast, it is possible for the MAs to deliver a generic methodology for delegation of management function which is independent of a framework.

2.8. RESEARCH APPROACHES ON MANAGEMENT DISTRIBUTION

Many researchers have been inspired by the benefits of hierarchical or distributed management as compared to the traditional, centralised model. A few of them have been illustrated in the below section.

2.8.1. Management by Delegation

Using Management by Delegation (MbD) framework [28], Goldszmidt et al. for the first time exhibited huge potential of vast deployment of distribution across managed devices, which was considered to be a great achievement in the field of network management. By dynamically delegating management tasks to stationary agents (known as “elastic” processes) and thereby enabling execution or implementation of management functions at the end nodes; MbD realises distributed management architecture. The elastic processes
uses a proprietary delegation protocol, known as the Remote Delegation Protocol (RDP) and facilitates a fresh control functionality to upload in the form of scripts. Based on the requirement of functional services, execution of scripts is normally triggered by other processes.

MbD is to believed be pioneer of ideas generated in this field. It has actually helped in promoting network devices to be considered as full-fledged managing entities from their previous perception of being “dumb” data collectors; thereby depicting a robust and potent management distribution model. In MA-based NSNs, distribution of management tasks is attained via MA objects and not via scripts that may be downloaded and also, the execution of MAs is not merely limited to a single device as they may dynamically move from one host to another. This becomes the basic and core differentiation.

2.8.2 Mobile Agents v/s Client/Server

The ability to dynamically move elements of a distributed application across the nodes of a computer network is known as “Code Mobility”. It is not a new idea and the key is to offer an alternate to the traditional Client/Server (CS) architecture of distributed applications and permit an enhanced usage of bandwidth resources with a superior degree of agility and reconfigurability. Various procedures have been designed and implemented in the recent days for the movement of code across the nodes of the network (e.g. remote batch job submission [49]). In distributed operating system researches, a better organised and planned approach has been adopted. In mobile code-based models, local interactions replace CS network communication as shown in Figure 2.9 and saves substantial network load.

A general definition of software agent can be: a computational entity, which can act on other’s behalf; is proactive or reactive; is autonomous; and is capable of learning and co-operating. Classification of software agents can be: static (stationary) or mobile. By introducing the ability of migration into the mobile code, an added degree of autonomy is possible and it brings forward the concept of the Mobile Agent (MA). MA is a software agent which has the capability to dynamically move from one host to another host.
As per the definition available for mobile codes, it becomes obvious that this concept and archetype can be used to develop technology required to accomplish management distribution. Linking mobile codes to network devices in a dynamic fashion can either be done on proactive basis by manager or on reactive basis by the network devices. In this manner, consumption of device resources only happens when it is actually required because the management primitives embedded in mobile code become accessible on the device only when management operations trigger that. Moreover, because they are capable of autonomously move from one host to another and carry out complex management functionalities without manager’s intervention; a new facet is added by MAs in distributed management.

As an output, mobile codes and particularly MAs, correspond to an ever increasing trend in the world of distributed management, which is evident by numerous researches. Just like other application areas, there is a lot of propaganda around MAs in network management and has created high hopes. In spite of all this, till now, industry has not shown great interest to encourage MA-based management and signs for this technology
to take off in near future are not appearing clearly. Many unanswered issues (like security issues) have not been addressed properly by the agent community. This is the main reason why MA has not been commercially adopted as yet and whether it’s use in management application will actually increase performance is still not clear.

2.9. CODE MOBILITY PARADIGM

Mobile code paradigms encompass different technologies, all sharing a single idea: to enhance flexibility by dynamically transferring programs to distributed devices and have these programs executed by the devices. The program transfer and execution can be triggered by the device itself, or by an external entity. In particular, they identified three different types of mobile code paradigms [49]: (a) Code on Demand, (b) Remote Evaluation, and (c) Mobile Agents as shown in Table 2.2.

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
<th>Code</th>
<th>Program stack</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Server</td>
<td>Mobile</td>
<td>Static</td>
<td>Static</td>
<td>WWW, RPC, EJB, Web services</td>
</tr>
<tr>
<td>Code on Demand</td>
<td>Static</td>
<td>Mobile</td>
<td>Static</td>
<td>Java Applets</td>
</tr>
<tr>
<td>Remote Evaluation</td>
<td>Static</td>
<td>Mobile</td>
<td>Static</td>
<td>SQL Commands</td>
</tr>
<tr>
<td>Mobile Agent</td>
<td>Semi Mobile</td>
<td>Mobile</td>
<td>Mobile</td>
<td>MA</td>
</tr>
</tbody>
</table>

2.9.1. Remote Evaluation Paradigm

In the Remote evaluation (REV) paradigm [50], a client (initiator) has the know-how necessary to perform a service but it lacks the required resources, which are located at a remote server (co-operator). Consequently, the client sends the service know-how to the remote site that executes the code using the resources available there. This is a form of push. The information transferred includes the agent code plus a set of parameters (arguments). After executing the operation, the remote site returns the results back to the initiator of the remote evaluation as shown in Figure 2.10.
2.9.2. Code on Demand Paradigm

In Code on demand paradigm, Client is able to access the required resource which is located at same place. However, it lacks the information (code) on how to process such resources. Thus Client interacts with the server, requesting the service know-how. A second interaction takes place when server delivers the know-how to client which can subsequently execute it as shown in Figure 2.11.
2.9.3. Mobile Agent

In the MA paradigm, the service know-how is owned by the client, but some of the required resources and data are located at a remote server. Hence, component A migrates to the server carrying the know-how and possibly some intermediate results. After its arrival, A completes the service using the resources available there. The MA paradigm differs from other mobile code paradigms on that the associated interactions involve the mobility of a computational component. In other words, while in REV and COD the focus is on the transfer of code between components, in the MA paradigm a whole computational component is moved to a remote site, along with its state, the code it needs, and some data required to perform the task. In that sense, MAs can be regarded as a ‘superset’ of REV/COD paradigms, as they can offer all the functionality provided by the latter, with the additional ability of autonomous migration as shown in Figure 2.12.

Figure 2.12: Mobile Agent Paradigm
2.10 LIMITATIONS OF CLIENT/SERVER BASED CENTRALISED NETWORK MANAGEMENT MODELS

A centralized architecture suffers from the lack of scalability and flexibility. Furthermore, the staleness of gathered data (due to network latency involved) and probable error in the selection of management task being carried over (owing to the staleness of data) reduces the reliability of the management applications.

The following are major issues of client server based centralised Network management models.

- **Centralized Management:** In this model, network manager plays a role of a centralized control unit. All the management decisions are taken by a single network node. As the network size grows so the efficiency of network management decreases. One of the drawbacks of centralized management is that if the management node fails, the overall network management would fail.

- **Scalability:** All the management data is transported to management station for management decision. This doesn’t scale as the network grows in complexity and size.

- **Bandwidth wastage:** In Client/Server model, the bandwidth usage associated with management traffic increases as level of hierarchy increases. Thus a large amount of network bandwidth is consumed by network management operations in Client/Server model.

- **Response Time:** Major The response time of a request depends upon the number of hop count between manager and managed device. In Client/Server model, the response time increase as level of hierarchy increases.

- **Fault Tolerance:** The fault tolerance capability of Client-Server NMS is least or zero. This is one of the major draw back of the Client/server based network management models.
2.11 MOBILE CODE – ADVANTAGES

Mobile code technologies represent a powerful programming paradigm, which is useful when designing distributed applications. The commonly agreed benefits of mobile code are following:

- **Enhanced Flexibility.** Clients typically access the resources hosted by a server through a set of services, whose interface is typically predefined and commonly agreed among the client and the server. Mobile code can be used to extend and update dynamically capabilities of applications, thereby enhancing systems flexibility.

- **Exploitation of increased resources availability:** Managed devices resources are characterised by continuously increasing availability in terms of processing power, disk and memory capacity. Mobile code takes advantage of that feature to achieve processing load distribution. In network management field, this advantage can be exploited to perform collection and filtering of management data locally, in a distributed fashion. That way, expensive platforms (managers) dedicated to issuing management requests, collecting, analysing and presenting data are not any longer necessary.

- **Reduction of network traffic:** The transfer of mobile code to the source of data creates less traffic than transferring the data, as mobile code can perform semantic compression of data, delivering pre-processed, high-level information.

- **Asynchronous interaction:** Once downloaded, mobile code can perform distributed tasks, even if the delegating entity does not remain active.

- **Interaction with real-time systems:** Installing mobile code close to a real-time system may prevent delays caused by network congestion. In network management, this problem arises when a number of successive interrelated Management Information Base (MIB) values need to be retrieved to present a snapshot of the system's state, e.g. on Simple Network Management Protocol (SNMP) table retrievals.
- **Support for heterogeneous environments:** Mobile code is separated from the hosts by an environment which is able to receive and instantiate the received code. If the framework is in place, mobile code can target any system, especially when the framework is implemented by a platform-independent language, e.g. Java. The cost of running a JVM on a device is decreasing.

### 2.12. AGENT MOBILITY

The field of MAs and mobile code has lately become a hot research topic covered by many networking and software engineering conferences. As it has been shown throughout the preceding sections, MA technology represents a promising programming paradigm, which can enhance the flexibility and scalability of contemporary NMSs. However, its merits and weaknesses should be carefully evaluated to ensure its effective use in network management applications. This section focuses on aspects related to agent mobility and MA platforms.

#### 2.12.1. Elements of a Mobile Agent Platform

A basic component of a Mobile Agent Platform (MAP) is the MA Server (MAS) that runs on each host where MAs can execute. The main purpose of the MAS is to provide an efficient execution environment able to receive, instantiate and dispatch agents, serve as an interface between incoming MAs and the underlying system resources and offer a set of services required by the MAs to perform their distributed tasks. Focusing on MAs, according to the definition given earlier, they comprise three parts:

- The code part which defines the MAs’ functionality;
- The data part (persistent state), including the values of the variables declared within the MA class;
- The execution thread (with an execution stack).
MA's state is dynamically updated as a result of their visits and interaction with distributed servers where information is collected. MAPs that provide strong mobility enable the transfer of all three parts on every MA migration. Most platforms involve the transfer of only the code and state information (weak mobility).

An agent migration may be initiated either from the MA itself or the hosting MAS server by invoking a move primitive, which allows an MA to move to the next server included into its itinerary, through an agent transfer protocol (ATP). At the time that the move method is called, the MA's state is saved and transferred through the network. At the destination site, the MA state is recovered and the agent instantiated, typically provided with its own thread of execution. ATPs are used to transfer agents between MASs and can be based on several protocols such as sockets, HTTP, Java RMI, etc. In addition, MAPs also provide agent development and deployment facilities, defined in APIs. A set of classes and interfaces are supplied and should be integrated in the agent code in order to enable mobility.

2.12.2. Weak vs. Strong Mobility

Existing mobile code languages provide support for at least one of the following:

- **Strong mobility**: the ability of processes to move their code and execution state to a different site. Processes are suspended, transmitted to the destination site, and resumed there. For instance, Telescript provides mechanisms to implement strong mobility.

- **Weak mobility**: the ability to transfer code across different execution environments; code is accompanied by its persistent state, but no migration of execution state is involved as shown in Figure 2.13. For instance, Java supports only weak mobility.
Figure 2.13: Classification of code mobility & Agent mobility Paradigm

Sun’s JVM does not allow capturing of processes’ execution states and, as a result, very few Java-based MA systems provide strong mobility. Those that do, fall into three categories systems using a modified JVM, a custom JVM and systems using a pre-processor approach. Clearly, the implementation of frameworks supporting strong mobility is not a trivial task, whilst introducing performance penalties in agent transfers. In addition, management tasks of configuration, maintenance and control typically involve the execution of repetitive tasks on every node. That means that the requirements of MA-based network management can be comfortably met by frameworks that only support weak mobility. This statement is also proved by the remarkable precedence of Java over other programming languages that support strong mobility.

A second classification of MAs is based upon their migration plan, i.e. on whether MAs visit one or more hosts. Single-hop agents travel to a target host, start their execution and remain there until they terminate. This type of agents does not need any data when migrating to the target host (except maybe initialisation data) or any methods for itinerary control. Therefore, single-hop MAs compare to downloadable code, i.e. they represent a direct application of REV paradigm.

In contrast, multi-hop or itinerant agents can travel to several sites during their lifetime. Multi-hop MAs are suitable for performing repetitive tasks over a set of devices. They can also perform different tasks, and can adapt their behaviour depending on the tasks
achieved in the previously visited hosts. In multi-hop MAs are further categorised in weak and strong MAs, with the former referring to the migration of an MA without preserving information gathered from previous visits and the latter involving the migration of MAs that preserve their state formed during previous visits. Weak MAs are termed memoryless\(^4\) agents in. To avoid confusion with the well-established definition of weak and strong mobility given in the preceding section, we adopt the term memoryless (multi-hop) MAs to refer to agents that cannot (can) preserve their persistent state when migrating.

### 2.12.3. Commercial Mobile Agent Platforms

The phenomenal popularity of MAs is reflected on several industrial initiatives that led to the development of numerous MAPs. State-of-the-art reports on general-purpose MAPs can be found in, with interesting comparative performance, robustness and scalability tests reported. In this section, we briefly review four representative and popular general-purpose MAPs, all implemented in Java: Aglets, Concordia, Voyager and Grasshopper.

- **Aglets** [51]: The oldest and most well-known platform, developed at the IBM Research Laboratory in Japan. The first version was released in 1996. The migration of Aglets is based on a proprietary Aglets Transfer Protocol. The Aglets Software Development Kit (ASDK) runtime consists of the Aglets server and a visual agent manager, called *Tahiti*. The ASDK provides a modular structure and an easy-to-use API for Aglets programming and also extensive support for security and synchronous/asynchronous agent communication.

- **Concordia** [52]: It has been developed by Mitsubishi Electric. This platform provides a rich set of features, like support for security, reliable transmission of agents, access to legacy applications, inter-agent communication, support for disconnected computing, remote administration and agent debugging.
- **Voyager [53]:** It is probably the most popular MAP, in terms of number of users. It has been developed by ObjectSpace. Voyager is an object request broker with support for MAs. The agent transport and communication is based on a proprietary ORB on top of TCP/IP. Voyager has a comprehensive set of features, including support for agent communication and agent security.

- **Grasshopper [54]:** Grasshopper has been developed by IKV++, with its main power lying on its compliance with FIPA and MASIF standards. MASs in Grasshopper comprises a core agency and a set of one or more places (runtime environments where agents run). The core agency offers a set of services to support agent migration and execution. The communication service that supports agent communication and migration may use a variety of protocols: IIOP, Java RMI and plain sockets.

- **JADE [55]:** (Java Agent DEvelopment Framework) is a software framework to develop agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. It provides an efficient, scaleable, distributed environment that fully implements all aspects of the FIPA(Foundation for Intelligent Physical Agents) communications model. JADE was designed to ease the process of developing agent applications and it uses several mechanisms for accomplishing it. JADE abstracts the communications infrastructure away from the programmer, providing transparent mechanisms for locating both local and remote agents and initiating conversations with them.

The advantages of using one of the MAPs presented above are that (a) they are relatively easy to use and typically well-documented, providing attractive frameworks for the rapid development of MA-based distributed applications; (b) most of them are robust, reliable and well-tested. Experiments, however, have demonstrated that these MAPs do not satisfy all performance requirements, as they either involve increased migration latency (Aglets, Voyager, Grasshopper) or poor scalability and robustness under stressing conditions.
2.12.4. Applications of Mobile Agents

MAs can be useful in several application fields, although none of them necessitates their use; in fact, each application can be designed based on existing technologies. However, the use of MAs can contribute to build these distributed applications in a more simplified and effective way. In the following, we identify some areas in which MA technology can actually give a positive contribution. Network management applications are not mentioned herein as they will be elaborated later.

- **Filtered Data retrieval:** MAs can be an effective tool for retrieving data within a distributed system; in fact, an agent encapsulating the user’s query can migrate to the place(s) where the data is actually stored; therein, the agent can obtain and filter data, and return the user only the useful information. This idea has been used into reduce the latency involved in remote database interactions.

- **Mobile computing:** Users want to access network resources from any position, notwithstanding the band limits of current wireless technologies. Thus, users can submit their requests through an agent, which runs their request within the network and returns the results later (so the user does not need to remain connected, waiting for the results).

- **Distributed Computation:** MAs represent new paradigm for parallel execution of computation-demanding tasks on a distributed network of workstations.

2.13. MOBILE AGENT APPLICATIONS IN NETWORK MANAGEMENT

MA-based distributed management has been a very hot research topic in the past few years, with a large number of proposed applications reported in the literature. This section provides an overview of MA-based approaches in a broad spectrum of applications, including network, systems, fault, configuration and service management. These applications are according to the mobility scheme used. Most of the applications
described below use the MAPs described in the preceding section as underlying platforms, whereas others have chosen general-purpose platforms such as Aglets.

Early work in the field of mobile code, carried over by Goldszmidt et al. [28], introduces the concept of management by delegation. Herein, the management station can extend the capability of the agents at runtime thereby invoking new services and dynamically extending the ones present in the agent on the device. Further work in the field of mobility of code established three major paradigms, a. Code on demand, b. Remote evaluation, c. Mobile agents, introduced by Baldi et al. [29]. Mobile agent based strategies have distinct advantages over the others as it allowed for easy programmability of remote nodes by migrating and transferring functionality wherever it is required.

Landmark work in this direction has been presented in [29] and [57]. A management task carried out on a set of NEs, involving multiple CS interactions with each NE has been examined. Models of the overall traffic and the traffic around the management station are computed for CS, CoD, REV and MAs approaches. The impact of semantic compression is also studied for mobile code approaches. The purpose of these models is to provide the network administrator with an objective quantitative criterion to choose the right approach, given the management task and the managed network topology.

White et al. [56] suggest that a Virtual Managed Component (VMC) resides at each NE, providing incoming MAs an interface to managed resources. The manager side of the NMS requires a similar interface called the Virtual Managed Resource (VMR), which can be viewed as a remote wrapper of the VMC. The VMR can be supplied as a single-hop MA that travels to the remote NE, thereby, seamlessly enabling the management of newly installed NEs. In addition, this allows network component suppliers to transparently use any network management protocol (even a proprietary protocol) for the management of NEs.

While the main focus of MA-related research activity on distributed management has been on developing network management-oriented frameworks and using them on specific applications, some researchers concentrated on evaluating the performance of
MA-based management. The purpose of such performance studies is twofold. First, they suggest ways to efficiently use MAs so as to outperform centralised network management approaches. Second, they allow determining how MAs are best deployed for particular types of network management operations.

Liotta et al. [3] presented an evaluation of MA-based monitoring systems, providing quantifiable metrics based on performance and scalability. MAs are used to perform monitoring tasks in place of traditional centralised polling. Performance is measured using the generated monitoring traffic and the monitoring delay. These parameters are evaluated for a combination of schemes based on monitoring models, MAs organisation and MA deployment patterns. Possible monitoring models consider MAs performing periodic polling. In the first model, each MA polls and analyses a set of managed objects (MO); this MA is subsequently polled by other MAs or the monitoring station. In the second model, each MA periodically polls the MOs, analyses the results, and notifies other MAs or the monitoring station. The third model differs from the previous one that MAs generate data (alarms) only if specific events are detected. The MA organisation can either be flat or hierarchical, and the deployment scheme can either be cloning-based or without cloning. This leads to four possible deployment patterns: flat broadcast with no cloning, flat broadcast with cloning, hierarchical broadcast with no cloning and hierarchical broadcast with cloning.

In [58,59,60], the authors investigated the problem of optimal placement of MAs acting as remote monitoring stations, so as to minimise data collection latency and the network traffic incurred due to (localised) polling. To address this problem, a distributed algorithm is proposed that relies on agents learning about the network topology through standard management interfaces and subsequent deployment of MAs to remote domains through a ‘clone and send’ process. Deployed MAs can possibly adapt to network changes and move again in order to maintain optimality. The latency aspects of MA migrations, with respect to management applications, have been investigated by Lipperts [61]. The results of time measurements indicate that MA migrations can be more time efficient than remote communications, especially when class loading is not necessary (the code of the MA is already at the destination host) and the operations are performed over
slow media. However, the decision concerning the replacement of a remote communication scheme by MA migrations should take into account several parameters, such as the number of remote communications to be replaced, the size of the parameters involved in the remote communication, the size of the agent and the networking environment. Lipperts proposed a solution based on utility theory to aid on deciding whether MAs deployment is justified or not. [62] and [63] included brief quantitative evaluations of their proposed applications demonstrating improved scalability compared to SNMP management.

Bellavista et al.[64] proposed a secure and open mobile agent environment, MAMAS (Mobile Agents for the Management of Applications and Systems) for the management of networks, services and systems. Sahai & Morin [65] introduce the concept of mobile network managers (MNM), which is a location independent network manager and assists the administrator to remotely control his/her managed network, through launching MAs to carry out distributed management tasks. In [66], Oliveira and Lopes propose how the integration of MA-based sub-system could be carried out in the IETF’s DISMAN framework. Manoj Kumar Kona et al. [67] described an SNMP based efficient mobile agent network management structure, in order to cooperate with conventional management system; Pualiafito et al. [68] introduce the Mobile Agent Platform (MAP), used for monitoring the systems state by calculating aggregation functions combining several MIB values; For transferring less network monitoring data and managing devices more effectively, Ghanbari et al.[69] adopted the methods of calculating Health Functions of MIB variables and polling SNMP tables to deploy management agents; Chi-Yu Huang et al. proposed a clustering mobile agent based network management model aiming at large enterprise entrant network [60]. Damianos Gavalas et al.[70] proposed a hierarchical and scalable management model where middle managers are themselves mobile and based on certain policies they dynamically segment the network and deploy other mobile middle managers for data collection. Vipin et. Al. [71] proposed a mobile agent based SNMP table filtering mechanism.

Application of mobile agents in network fault management [72, 73, 75, 75] is another emerging challenging area in the field of distributed network management. In general
there are three steps for the identification of faults- first is fault detection, which involves correlating the alarms to reach up to a common fault such as MIB variables crossing some predefined thresholds. Secondly, fault localization involves identification of objects that might be primary or secondary source of faults and last step is correlating all the results to identify the source of the faults. Mobile agents are most suited for distributed processing. First step is performed by the subordinate agents locally at the object itself and last two steps are performed by manager agent. For a comprehensive review of the work done in distributed fault management area, research paper by Atul Srivastava et. al [92] could be referred.

The main concern comes with using mobile agents is Security. Both hosts that execute the agent and the agent need security mechanisms to protect against threat posed by the malicious hosts and agents. A number of approaches [76, 78, 79] have been developed for protecting mobile agents. Protection of mobile agents includes protecting data and code refers to integrity of mobile agents. These security solutions can be classified into prevention and detection. Mechanisms aimed at prevention use security techniques to prevent the unauthorized access of code and data. On the other hand, mechanisms aimed at detection attempt to detect any unauthorized modification of an agent. The preventive techniques are proactive, aimed at hiding code, data and flow control from hosts that execute them while the detection techniques are reactive, allowing an attack occur in first place. For a comprehensive review of the work done in security management of mobile agents in the field of distributed network management area, research paper by A.K. Sharma et. al [93] could be referred.

2.14. MOBILE AGENTS BASED NETWORK MANAGEMENT MODEL

Damianos Gavalas et al. [70] have discussed number of mobile agent-based network management models in their research work. Few important ones have been discussed and compared against certain design parameters in this section.
1. Flat Bed Model: For a particular management task a single mobile agent is launched from a management station which then traverses the network topology in a sequential manner, visiting each managed device and carrying out the assigned task. Though the model relieves the network from the flood of request/response messages, it introduces the issue of roundtrip delays as the network size grows. This leads to scalability issue if data has to be collected very frequently from managed devices. The model is shown in Figure 2.14.

![Figure 2.14: Flat Bed Model](image)

2. Segmentation Model: Here the scalability issue is addressed by partitioning the network into many administrative or geographical domains as shown in Figure 2.15 and assigning a single MA entity to each one of them.

![Figure 2.15: Segmentation Model](image)
This brings high degree of parallelism in the data collection architecture and brings the response time down by many folds.

3. Hierarchical Static Middle Managers Model: The scalability problem is more adequately addressed by deploying hierarchical models wherein NM tasks are delegated to MAs. They migrate to remote subnetworks/domains where they act as local managers and takes over the responsibility of local devices from the central manager. These models suffer from automatic adaptation of management system to changing network configurations, i.e. mid level manager do not change the location where they execute. The model is shown in fig 2.16.

![Hierarchical Static Middle Managers Model](image)

**Figure 2.16: Hierarchical Static Middle Managers Model**

4. Hierarchical Mobile Middle Managers Model: In search of more flexible solutions, a concept of Mobile Middle Manager (MDM), referring to a management component that operates at an intermediary level between the manager and the management end points, is introduced. The mobility feature of the MDMs allows the management system to adapt dynamically to a changing network conditions. MDMs can be deployed to or removed from a given network segment in response to change in network traffic or move to a least loaded host to optimize local resource usage. The model is shown in Figure 2.17.
Figure 2.17: Hierarchical Mobile Middle Managers Model

2.14.1 Comparative analysis of C/S and various MA-based management models

All the MA-based management models enumerated in the previous section and conventional client/server model (SNMP client/server model) are compared for five typical design parameters, i.e. scalability of management functions as the network grows, load balancing amongst participating entities, fault-tolerance of the management model, dynamic adaptation to the changing needs of network topology & traffic patterns and easy integration of new management services in the management model. Table 2.3 summarizes the comparative study.

Table 2.3: Comparative Analysis of various Network Management Model

<table>
<thead>
<tr>
<th>Design Parameters</th>
<th>C/S Model</th>
<th>FBM</th>
<th>SM</th>
<th>H-SMM</th>
<th>H-MMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>Poor. The network gets flooded with management request/response messages.</td>
<td>Becomes poor with increase in the size of network and if data has to collected very frequently.</td>
<td>Becomes poor with increase in the number of segments.</td>
<td>Better than the C/S, FBM &amp; SM models as Static Manager takes over the management tasks from the central manager.</td>
<td>Better than the C/S, FBM &amp; SM models as Mobile Manager takes over the management tasks from the central manager.</td>
</tr>
</tbody>
</table>
2.15 SUMMARY

The advent of MA technology has signaled many potential benefits in the network management arena and attracted the attention of several researchers working on the field. In particular, MAs promise to overcome the limitations of traditional centralized architectures and address the weaknesses of distributed management approaches reviewed in the previous chapter. As a result, MA-based management applications have largely increased in number. The intensity of research activity on that field is strongly
related with the proliferation of MAPs expressly developed with management applications orientation. These platforms have demonstrated improved security, fault tolerance, MA control, inter-agent communication and interoperability features, while some also aimed at optimizing the performance of MA migrations through sophisticated code distribution mechanisms. All these facts signify that MA community is about to reach the state of maturity.

However, a critical look at the available literature indicate that the following issues need to be addressed towards the design of an efficient mobile agent based distributed network management system for telecommunication networks.

1. Most of the MA-based frameworks suggested in the literature talk about a ‘flat’ network structure for monitoring applications, wherein, a single MA launched from the manager platform sequentially visit all the managed NEs, regardless of the underlying topology. This only partly solves the scalability limitations of centralized architectures as with increase in number of nodes in the network the roundtrip time of the MA increases.

2. Most of the suggested frameworks architecturally deploy static mid-level managers in between the management applications and managed devices. This present flexibility and scalability issues.

3. Models talk about deploying MA as mid-level managers on network devices for managing their own domains or zones. Many of these devices don’t support mobile agent runtime environment. Little attention has been paid on how to manage these devices without mobile agent execution environment like hubs and switches in enterprise networking environment and most of the telecomm devices like terminals, switches and multiplexers. This also includes legacy devices.

4. Not much work is carried out in the telecomm arena. Most of the reported work is in the field of enterprise network management area.

The investigation of these issues comprises the main part of the research work presented in the following chapters, which describe the design and implementation details of a mobile agent based architecture tailored to management applications, with the main focus being on network monitoring and performance management.