Requirements for a P2P platform for dynamic management of large scale networks
Start date of the project:  January 2004
Duration:  48 months
Project Coordinator:  Prof. Dr. math. Friedhelm Meyer auf der Heide
Heinz Nixdorf Institute, University of Paderborn, Germany
Due date of deliverable:  December 2004
Actual submission date:  February 2005
Dissemination level:  PU – public

Work Package 2.3:  Global management of competing transport services over large net domains
Participants:
Telecom Italia Learning Services S.p.A., Rome, Italy
University of Paderborn, Germany
Universitat Polytecnica de Catalunya, Barcelona, Spain
Author of deliverable:
Giovanni Cortese (g.cortese@computer.org)
Antonio Cozzolino (UE008275@guest.teelcomitalia.it)
Federico Morabito (federico.morabito@telecomitalia.it)
Fabrizio Davide (fabrizio.davide@telecomitalia.it)
Abstract

Scalable management and self-organizational capabilities are emerging as central requirements for a generation of large-scale, highly dynamic, distributed applications. We have developed a new distributed information management system called XDM. XDM collects largescale system state, permitting rapid queries and providing on-the-fly attribute indexing. The combination of features makes it possible to solve a wide variety of management and self-configuration problems. This paper describes the design of the system with a focus upon its monitoring facility. XDM is implemented using a peer-to-peer protocol, Pastry. Over the Pastry substrate, some components have been developed and integrated: a data management component, a topic-base Publish/Subscribe communication component and a query processor supporting long running queries functionality.

1 Introduction

Large-Scale Information Systems (such as mobile ad-hoc networks, peer-to-peer overlay networks, grid networks and the WWW) have reached a level that puts them beyond our ability to deploy them, manage them, and keep them functioning correctly through traditional techniques. Reasons for this are their sheer size with millions of users and interconnected devices and their dynamics; they evolve dynamically over time, i.e., components change or are removed or inserted permanently. In the WWW as in peer-to-peer systems and grid networks, sites and resources of different types appear and disappear with relatively high frequency. According to the P2P vision, thousands, millions, or even billions of peers may interact in a seething, transient pattern of communication. Architects of these systems claim this chaos can lead to properties such as “durability”, “anonymity”, “scalability”, and “security” — the list goes on.

In this scenario also the approach to management calls for an innovative point of view.

Traditional network management employs a variety of tools, applications, and devices to assist human network managers in monitoring and maintaining networks. Network management architectures have a rather centralized flavor, based on the manager-agent pattern.

Dynamic IT systems need new management architectures which are decentralised, adaptive and based on autonomous behavior of their components, to address the demand for scalability, to cope with the dynamics nature of the nodes constituting the network, and to achieve resiliency to failure in case of unpredictable events (load variations, external attacks etc).

Effective design of a management architecture, having these characteristics, will require a deep theoretical and empirical understanding of network structure, to monitor the corresponding network configuration, and network dynamics, to characterize optimal routing policies and traffic flows.

Along this path, a number of influential research projects in the field of P2P and autonomic systems have started redefining the concept of management. In addition, basic research on network theory promises to deliver much useful insight to design more scalable and fault tolerant systems.

2 Large Scale P2P systems

P2P technology and its applications seems destined to play a major role in reshaping telecom and enterprise service provisioning infrastructure. On one hand, existing, popular file sharing applications as Gnutella, Kazaa et al. are having a huge impact on the traffic characteristics and demand in ISP networks. From a different perspective, established businesses such as content providers are considering to exploit the P2P approach to launch new Internet services. Finally, there is an overall growing interest even in the enterprise market for self-managing, autonomic applications as P2P technologies promises to deliver ([8],[9]).
Most well-known first generation P2P applications are based on flooding protocols, and do not attempt to provide service level guarantees to their users. Academic research in peer-to-peer (P2P) systems (CAN [2], Chord [6], Pastry [1], Tapestry [13], etc) has concentrated largely on algorithms to improve the efficiency, scalability, robustness, and security of query routing in P2P systems, services such as indexing and search, dissemination, and rendezvous for applications running on top of these systems, or even many of the above.

We review now a number of application areas that have been experimented or are receiving attention as candidate targets for P2P technology.

Routing Problems - All distributed systems need a routing layer to get messages to their intended recipients. Routing takes on P2P characteristics when the scale is large enough (e.g., the Internet) or when centralization is ruled out (e.g., wireless ad hoc networks). Internet and ad-hoc routing are discussed in the related sections.

Backup - The process in which a user replicates his files in different media at different locations to increase data survivability, can benefit greatly from the pooling of otherwise underutilized resources. While Internet backup faces a number of major obstacles, in the corporate environment, when participants enjoy high mutual trust, P2P backup makes sense. In the design of the management architecture we have focused on a data management component enhancing support for data availability in a dynamic environment.

Distributed Monitoring - Monitoring is an important task in any large distributed system. It may have simple needs such as “subscribing” to first-order events and expecting notification when those events are published (e.g., Scribe); it may involve more complicated, on-line manipulation, for instance via SQL queries, of complex distributed data streams such as network packet traces, CPU loads, virus signatures (as in the on-line network monitoring problem motivating PIER); it may be the basis for an on-line, post mortem longitudinal study of many, high-volume data streams. The main requirement we have followed in the design of the management architecture is the monitoring facility.

Data Sharing and dissemination - File sharing is the best understood of P2P applications. In file sharing systems, participants offer their local files to other peers and search collections to find interesting files. Data dissemination is similar to data sharing, with the distinction that the problem is not to store data indefinitely but merely to spread the data for a relatively short amount of time. Often storing is combined with spreading. Usenet, perhaps the oldest and most successful P2P application, is a massively distributed discussion system in which users post messages to “newsgroups”. These articles are then disseminated to other hosts subscribing to the particular newsgroup, and made available to local users.

Networks and Internetworks - Internet Routing Internet routers must communicate to cope with a dynamically changing network topology to determine how to route outbound packets to their destination. They are arranged into “autonomous systems” which “peer” with each other across organizational boundaries, frequently between competitors. Routing protocols have historically assumed that economic incentives and legal contracts are sufficient to discourage misbehavior. At the application layer (e.g., Resilient Overlay Networks (RON) [20]) or at the network layer (e.g., BGP [21]), routers trust information from known peers. They cooperate because the information being exchanged is relevant to all peers and important to their function. This cooperation tends to fail if error, misbehavior or usage patterns cause the data to change too fast. To scale to the size of the Internet, BGP tries to limit the rate of change by aggregating routes instead of having ISPs propagate internal routing updates. Aggregation reduces the ability to detect path outages quickly [22]. RON instead gives up scaling to large numbers of nodes in favor of more fine-grained route information exchanges.
2.1 Distributed Object Location and Routing

The paradigm used in the design of P2P system is based on the new concept of DOLR (Distributed Object Location and Routing) system. DOLR systems provide the ability to route a message to a node without knowing its location. They can tolerate broken routing lists and non-responsive systems through replication of requests. Message routing provides locality properties, ensuring short traversal paths to reach one of a set of data replicas. Privacy and data confidentiality are in some cases a built-in functionality of the DOLR middleware, through cryptographic algorithms used to secure data exchanges.

DOLR address scalability by performing automatic integration of new peers and removal of old ones. They can avoid bottlenecks by distributing load among the peers, and can maintain load balancing related “system invariants” in spite of high rate of high dynamics in the system. DOLRs address availability through redundancy and continuous repair of both data and routing. Introspective techniques are used in some of these systems to move data close to where it is needed, thereby increasing the chance it can be accessed. Denial-of-service resilience is an extreme form of availability, combined with introspection to recognize and suppress attacks.

2.2 Distributed Hash Table

DOLRs are most often based on the DHT concept [11]. A Distributed Hash Table (DHT) is an hash table that is cooperatively maintained by a large number of machines worldwide. The machines may join and leave the system autonomously. The unprecedented scale and dynamism of the system calls for novel design techniques which emphasize decentralization and automatic re-configuration. Briefly, each machine in a DHT is assigned an ID in I = [0; 1). The set of IDs divides I into disjoint partitions, managed by one machine each. As a function of their IDs, the machines set up connections among themselves. These connections are used for routing messages between the machines. The challenge lies in devising efficient decentralized algorithms for ID management and connection maintenance.

A plethora of DHT systems have been designed, based on different algorithms for routing (including hypercubes, randomised, butterflies, small world networks, . . .).

Application oriented overlays

DOLR middleware creates application layer overlays on top of the Internet. In addition to the basic services (the self-managing message routing service), most DOLR systems are coupled with other services which exploit the robustness of the routing overlay, such as scalable application level multicast, and massively distributed, scalable and resilient data management services. Based on this class of global scale, decentralised middleware, researchers are experimenting with designing and building such applications as email infrastructures, collaboration tools, distributed file systems and databases, time stamping services, distributed knowledge management applications.

2.3 Pastry

In the XDM platform we have implemented a DHT based on Pastry, a scalable, distributed object location and routing substrate for wide-area peer-to-peer applications.

A Pastry system is a self-organizing overlay network of nodes, where each node routes client requests and interacts with local instances of one or more applications. Any computer that is connected to the Internet and runs the Pastry node software can act as a Pastry node, subject only to application-specific security policies.

Each node in the Pastry peer-to-peer overlay network is assigned a 128-bit node identifier (nodeId). The nodeId is used to indicate a node’s position in a circular nodeId space, which ranges from 0 to 2128-1. The nodeId is assigned randomly when a node joins the system. It is assumed that nodeId are generated such that the resulting set of nodeId is uniformly distributed in the 128-bit nodeId
The nodeIds are generated by computing a cryptographic hash of the node’s IP address. As a result of this random assignment of nodeIds, with high probability, nodes with adjacent nodeIds are diverse in geography, ownership, jurisdiction, network attachment, etc.

Assuming a network consisting of N nodes, Pastry can route to the numerically closest node to a given key in less than log2bN steps under normal operation (b is a configuration parameter with typical value 4).

For the purpose of routing, nodeIds and keys are thought of as a sequence of digits with base 2^b. Pastry routes messages to the node whose nodeId is numerically closest to the given key. This is accomplished as follows. In each routing step, a node normally forwards the message to a node whose nodeId shares with the key a prefix that is at least one digit (or b bits) longer than the prefix that the key shares with the present node’s id. If no such node is known, the message is forwarded to a node whose nodeId shares a prefix with the key as long as the current node, but is numerically closer to the key than the present node’s id. To support this routing procedure, each node maintains some routing state, which we briefly describe. An example of routing state are shown in Fig. 2.
Each Pastry node maintains a routing table, a neighbourhoods set and a leaf set. A node’s routing table is organized into $\log_2 N$ rows with $2^b - 1$ entries each. The $2^b - 1$ entries at row $n$ of the routing table each refer to a node whose nodeId shares the present node’s nodeId in the first $n$ digits, but whose $n + 1$-th digit has one of the $2^b - 1$ possible values other than the $n + 1$-th digit in the present node’s id. Each entry in the routing table contains the IP address of one of potentially many nodes whose nodeId have the appropriate prefix; in practice, a node is chosen that is close to the present node, according to the proximity metric. If no node is known with a suitable nodeId, then the routing table entry is left empty.

The neighborhood set contains the nodeIds and IP addresses of the $M$ nodes that are closest (according the proximity metric) to the local node. The leaf set $L$ is the set of nodes with the $L/2$ numerically closest larger nodeIds, and the $L/2$ nodes with numerically closest smaller nodeIds, relative to the present node’s nodeId.

3 Management of Large Scale Systems

Over the past two decades, the telecommunication industry has developed and brought to maturity a number of technologies and standards for the management of networks. To some extent, the same technologies can be applied also for managing applications running on top of the network. In this section we review the “classic” network management architecture and technologies, based on SNMP architecture and configuration techniques.

3.1 Concepts

A useful starting point is the classical definition of network management. “Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and QoS requirements at a reasonable cost.”

The key functional areas of network management, as defined by the International Organization of Standardization (ISO) are defined as follows:

- Fault Management.
- Accounting Management.
- Configuration and name management.
- Performance management.
- Security management.

The traditional approach based ISO standards is oriented to the management of the network, giving information about network-level functionalities. To give support also to service providers and networks operators, an approach of network management focused on management of business processes in a cost and time effective way has been developed by TeleManagement Forum (TMF). In this approach, the objectives and the information we want to observe and collect are more focused on the services and their deployment, in order to collect for upper level management tools aggregated information about accounting, billing and provisioning.

In the following sections, we give a basic introduction about the common and traditional techniques and technologies used to address the problem of network management.
3.2 SNMP

We introduce briefly the Simple Network Management Protocol (SNMP), a protocol developed by IETF to standardize network management for TCP/IP-based networks, which rapidly has become the "lingua franca" for network management. SNMP addresses network monitoring, but only at a very limited degree configuration actions on network devices.

A network management system based on SNMP is a collection of tools for network monitoring and control. The architecture is (mostly) centralized and includes the following key concepts:

- **Management station** — a management station include typically a database describing the configuration of all network elements in the managed network, and collected observations related to performance and fault data. A set of applications and GUIs allow a human to view this wealth of information, and act on the network by changing configuration parameters on individual (or groups of) network element(s). Management stations can be programmed to react automatically to notification of important events by executing actions such as operator notification, event logging, system shutdown, and automatic attempts at system repair.

- **Management agent** — each element of the network (such as computer systems and network devices) runs an SNMP agent. Agents respond to enquiries from the management station and send notifications of important events to it.

- **Management information Base (MIB)** — A MIB is an ‘ontology’ describing the attributes and commands related to a specific management functionality or device type, which constrains and provides semantics to the communication between managers and agents.

- **Network management protocol** — the allowed set of message types among agents and management stations.

- **Proxy agent**

In any configuration, at least one manager node (Management station) runs SNMP management software. The Management station execute management applications which monitor and control network elements. The Management station will have, at a minimum:

- A set of management applications for data analysis, fault recovery, and so on.

- An interface by which the network manager may monitor and control the network.

- The capability of translating the network’s requirements into the actual monitoring and control of remote elements in the network.

- A database of information extracted from the MIBs of all managed entities in the network.

Network elements are devices to be managed by Management station, such as bridges, routers, servers, and workstations, are equipped with an agent software module (Management agent). The agent is responsible for providing access to a local MIB of objects that reflects the resources and activity at its node. The agent also responds to manager commands to retrieve values from the MIB and to set values in the MIB.

Resources in the network may be managed by representing these as objects. Each object is a data variable that represents one aspect of the managed agent.

The collection of objects is referred to as a management information base (MIB). The MIB functions as a collection of access points at the agent for the management station. A management station performs the monitoring function by retrieving the value of MIB objects and can cause an action to take place at an agent or can change the configuration settings at an agent by modifying the value of specific variables.
The Simple Network Management Protocol (SNMP) is used to communicate management information between management stations and the agents in the network elements. The protocol includes the following key capabilities:

- **Get**: enables the management station to retrieve the value of objects at the agent.
- **Set**: enables the management station to set the value of objects at the agent.
- **Trap**: enables an agent to notify the management station of significant events.

The message-types exchanged between the nodes and their functions are listed below:

<table>
<thead>
<tr>
<th>Message-type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetRequest</td>
<td>“Mgr-to-agent”: Fetch a value from a specific variable</td>
</tr>
<tr>
<td>GetNextRequest</td>
<td>Fetch a value without knowing its exact name</td>
</tr>
<tr>
<td>GetBulkRequest</td>
<td>Fetch a large volume of data (e.g., a table)</td>
</tr>
<tr>
<td>SetRequest</td>
<td>“Mgr-to-agent”: store a MIB value in a specific variable</td>
</tr>
<tr>
<td>GetResponse</td>
<td>“Agent-to-mgr”: a response to any of the above Request</td>
</tr>
<tr>
<td>InformRequest</td>
<td>“Mgr-to-Mgr”: here’s MIB value, reference to third-part data (e.g., for a proxy)</td>
</tr>
<tr>
<td>Trap</td>
<td>“Agent-to-Mgr”: inform manager of exceptional event</td>
</tr>
</tbody>
</table>

Table 1: Set possible SNMP operations

Operations get-request and set-request provide the basic fetch and store operations; response provides the reply. SNMP specifies that operations must be atomic, meaning that if a single SNMP message specifies operations on multiple variables, the server either performs all operations or none of them. In particular, no assignments will be made if any of them are in error. The trap operation allows managers to program servers to send information when an event occurs. For example, an SNMP server can be programmed to send a manager a trap message whenever one of the attached networks becomes unusable (i.e., an interface goes down).

SNMP was designed to be an application-level protocol that is part of the TCP/IP protocol suite. It is intended to operate over UDP. Each agent must also implement SNMP, UDP, and IP. For an agent device that supports another application, such as FTP, both TCP and UDP are required. Because SNMP relies on UDP, which is a connectionless protocol, SNMP is itself connectionless. The key reason for this is robustness. Network management operations become increasingly important as failures and outages of various sorts are experiences. If SNMP relies on the use of a transport connection, then the loss of that connection could impair the effectiveness of SNMP exchanges. (No provision has been made for the use of SNMP over TCP. However, RFC 1283 prescribes conventions for the use of SNMP over the ISO connection-oriented transport service (COTS)).

The use of SNMP requires that all agents must support a common protocol suite, such as UDP and IP. Unfortunately, some devices do not support any part of the TCP/IP protocol suite. Further, there may be numerous systems (PC, workstations, etc) that support TCP/IP, but for which it is not desirable to add SNMP and its functions. To issue this situation, was develop the proxy agent, that acts as a proxy for one or more other devices.

We include also a brief introduction about SMPv3, the actual version implemented. The importance of SNMPv3 in our work is related to the introduction of security issues in the network management infrastructure. In our design, the security model will be enhanced by cryptographic hash functions ((SHA-1)) used in the overlay networks construction.
3.2.1 SNMPv3

To enhance the basic functionality of SNMP, a new version was introduced in 1993 and revised in 1996, the SNMPv2.

However, neither SNMP nor SNMPv2 offers security features. Specifically, SNMPv1/v2 can neither authenticate the source of a management message nor provide encryption. To correct the security deficiencies of SNMPv1/v2, in 1998 was introduced the SNMPv3. SNMPv3 is designed to secure against the following principal threats:

- **Modification of Information**: the modification threat is the danger that some unauthorized SNMP entity may alter in-transit SNMP messages generated on behalf of an authorized principal in such a way as to affect unauthorized management operations, including falsifying the value of an object.

- **Masquerade**: the masquerade threat is the danger that management operations not authorized for some principal may be attempted by assuming the identity of another principal that has the appropriate authorizations.

- **Message Stream Modification**: the SNMP protocol is typically based upon a connectionless transport service which may operate over any subnetwork service. The reordering, delay or replay of messages can and does occur through the natural operation of many such subnetwork services. The message stream modification threat is the danger that messages may be maliciously re-ordered, delayed or replayed to an extent which is greater than can occur through the natural operation of a subnetwork service, in order to effect unauthorized management operations.

- **Disclosure**: the disclosure threat is the danger of eavesdropping on the exchanges between SNMP engines. Protecting against this threat may be required as a matter of local policy.

SNMPv3 includes three important services: authentication, privacy and access control, described in Figure 4.

There are at least two threats against which SNMPv3 is not protect, that are:

- **Denial of Service**: an attacker may prevent exchanges between managers and agents
Traffic Analysis: an attacker may observe the general pattern of traffic between managers and agents.

### 3.3 Configuration

The traditional Internet protocols aim at providing a best-effort services. For a provider wishing to provide QoS guarantees to its customers, the most common approach is over provisioning of network resources. However, technologies exist and start being deployed which account for providing QoS guarantees per end-to-end flows or prioritized traffic for application types. MPLS is a promising technology that addresses the need to have different class of service by using resource reservation and by integrating ad-hoc mechanisms for QoS guarantees (DiffServ, IntServ). Policy-based architectures have been introduced with the aims to facilitate the network configuration.

#### 3.3.1 MPLS

MPLS (Multi Protocol Label Switching) is a technology growing in popularity with global service providers. MPLS uses label commutation for packets routing, transforming connection-less networks in connection-oriented network.

In many ways, MPLS provides the best of both IP and asynchronous transfer mode (ATM) by combining traffic engineering, subnetwork connections, and different quality of service (QoS) models. IP, on the other hand, provides just a best-effort datagram service. By bringing together these two domains (IP and the ATM-connection-oriented telecoms world) requires an integrated approach to network management.

MPLS provides the possibility of a unified core network for both service providers and enterprises. In this scheme, legacy technologies such as ATM, frame relay, and Ethernet can be pushed out of the core network to the edges. The resulting core network is then packet-based using MPLS and some specified QoS mechanism such as DiffServ, IntServ, and so on. Having a single connection-oriented, QoS-based core technology provides a foundation for standard signaling protocols such as Resource Reservation Protocol with traffic engineering extensions (RSVP-TE) and Label Distribution Protocol (LDP).

We briefly describe how MPLS manages the following functional areas of network management:
• **Configuration.** Modifies the network in some fashion, such as creating a label-switched path (LSP).

• **Accounting** (or billing). Enables an operator to determine usage of network resources.

• **Performance.** Determines whether the network is operating within required limits.

• **Security.** The focus here is ensuring that network resources are protected from unauthorized access.

The network configuration may involve any of the following technologies:

- MPLS interfaces
- IP routing protocols
- MPLS signaling protocols
- Creation of LSPs
- User-defined DiffServ classes of service

Providing accounting/billing support for MPLS network may involve considerations such as calculating network resource usage and apportioning bills or settlement fees. If a single service provider implements the MPLS network, the end users are billed for the service via either a flat-rate or a usage-based model.

Accounting may also be used inside a given service provider to apportion costs to different departments/divisions.

Here are some of the simple accounting metrics that might be included:

- Number of IP packets received and forwarded by an label-edge router
- Number of IP packets forwarded (via MPLS) across an LSP
- Number of dropped IP packets

After the MPLS network is deployed and the service is up and running, it’s important to monitor performance. Nodes can become faulted or congested, which can have a direct impact on the underlying services. Because this in turn can affect the conformance to SLAs by the service provider, resulting in possible financial penalties, performance monitoring is increasingly important. Performance data can be acquired in a number of ways:

- Polling MIB objects
- Receiving threshold crossing events (for example, counters that reach preset levels)
- Real-time collection via trend analysis of selected managed objects

MPLS networks are usually considered to offer at least the same level of security as ATM or frame relay networks. It would probably be advisable for an enterprise user to encrypt any sensitive traffic prior to forwarding it over an MPLS core.

Broadly speaking, the security responsibilities of an network management system consist of protecting the network devices and the associated managed objects, which may include enforcing SNMPv3 usage across the network. The security capability should also verify that the network and its traffic are secure from unauthorized access.
3.3.2 Integrated Services

The Integrated Services architecture was introduced by IETF in 1994 to implement multiservices model over the Internet architecture for the deployment of real-time applications requiring preferential and differentiated treatment for different traffic flows. IntServ architecture introduces some different classes of services (Guaranteed Service, Controlled Load Service and Best Effort) and gives QoS guarantees to micro traffic flows. The bandwidth is assigned to each micro traffic flow according to class of service, traffic profile and requested QoS level. The RSVP (Resource ReSerVation Protocol) is used to manage the specifications of parameters identifying micro-flows, the traffic profile, the class of services and the consequent bandwidth allocation. For large-scale networks, the main weakness of IntServ is the guarantees of QoS at the micro-flows granularity: this will imply to maintain information in the routers interested in the traffic operations about requested bandwidth and profile for each micro-flow. The result is a non efficient memory usage and CPU overload at the routers level, with consequent difficult to follow the growing number of nodes.

3.3.3 Differentiated Services

The Differentiated Services model defines a simple and scalable architecture able to differentiate the treatment of traffic flows without storing information about traffic in the routers.

The scalability of DiffServ architecture is obtained using aggregation of traffic flows (extracting information from IP packet) and using coloration and marking to characterize the strategies and policies to be applied for the routing of the specified packets. To maintain different strategies and policies for each traffic flow, DiffServ creates logical different internets, each one managing part of the traffic according to Better-than-Best-effort service level. The differences with IntServ is that DiffServ doesn’t use information stored in the routers but introduces coloration and marking to manage the class of services differentiation.

Per-Hop-Behaviour is used to manage in a different way each packet, according with its color (Differentiated packets treatment). PHB permits to make resource reservation allocating bandwidth portion to aggregated traffic flows (for some reasonable temporal interval). The resource reservation is applied by configuring Client routers or routers at the edge of DiffServ domain.

This characteristic reflects the core&edge architecture which DiffServ is based on, implementing complex functions in edge of network router. This paradigm will not permit to manage P2P overlay network, where the functionalities have to be included in the core network in order to follow the dynamics of the network.

3.3.4 Policy Based Management

Policy-based architectures have been used in the network management community to provide a standard way for configuring network boxes across different vendors (note that SNMP is weak in the way it supports invocation of actions on agent). Standards defined by the IETF mandate formalisms to specify policies and protocols for exchanging policy information among configuration management applications and network boxes (COPS).

Currently, policy-based infrastructures have been used to provide a limited degree of automation [23]. In general, a policy is defined as a set of rules that are based on the ECA pattern i.e. Event ->if (Condition) ->then (Action). These rules map system states to setting of tunable parameters and invocation of system services [24].

There are multiple approaches for specifying policies. They can be specified as a programming language that is processed and interpreted as a piece of software [25, 26] or in terms of a formal specification language [27, 28] or the simplest approach is to express policies as a sequence of rules.

The IETF has chosen rule-based policy representation in its specifications [29]. Specific protocols are defined (COPS) for exchange of policy information between policy control points and policy.
decision points (i.e. network boxes) — check this. As in the case for SNMP, a language is defined for specifying ‘Policy Information Bases’; again PIB are supposed to be generally agreed ontologies, with a wide support from hardware vendors. In the IETF framework, which represents the accepted standard in the network management community, policies are currently mainly used for configuring IPSec security and QoS policies on routers and other ‘middleboxes’ in the network fabric.

4 XDM, a DHT-based Information Management Infrastructure for management in a large scale scenario

In this section, we describe the functionalities of the proposed infrastructure and we briefly introduce the preliminary prototype, named XDM.

4.1 Infrastructure for network management

A network management system is a collection of tools for network monitoring and control. It is designed to view the entire network as a unified architecture, with addresses and labels assigned to each point and the specific attributes of each element and link known to the system. The active elements of the network provide regular feedback of status information to the network control center. Each network node contains a collection of software devoted to the network management task, referred as a network management entity (NME). At least one host is designated as the network control host, or manager. The manger includes a collection of software called the network management application (NMA). Other nodes, agent nodes, include an NME that responds to requests from a manager system.

In the infrastructure, one manager node acts as managing entity of other nodes, that act as managed devices. The managed devices can be bridges, routers, servers and workstations. The managed devices contain managed objects whose data is gathered into a Management information Base (MIB). The agent is responsible for providing access to local MIB of objects that reflects the resources and activity at its node. The agent also responds to manger commands to retrieve values from the MIB and to set values in the MIB. The nodes in the network communicate using an application-level network management protocol that employs the communications architecture in the same fashion as any other distributed application.

A centralized network management system implies central control, maintaining control over the entire configuration, balancing resources against needs and optimizing the overall utilization of resources.

In a dynamic environment, the centralized configuration is very critical in unpredictable conditions and often a manager station is under utilized while other one is over utilized.

A distributed management system replaces the single network control center with interoperable workstations located on LANs distributed throughout the enterprise.

To prevent anarchy a hierarchical architecture is typically used, with the following elements:

- Distributed management stations are given limited access rights for network monitoring and control, usually defined by resources they serve.
- One central workstation, with a backup, has global access rights and the ability to manage all network resources.

The principal benefits are:

1. Network management traffic overhead is minimized. Much of the traffic is confined to the local environment.

2. Distributed management offers greater scalability. Adding additional management capability is simply a matter of deploying another inexpensive workstation at the desired location.
3. The use of multiple, networked stations eliminates the single point of failure that exists with centralized schemes.

Closest to the users are the management clients. Depending on access privileges, a client workstation may access one or more management servers. The management servers are the heart of the system. Each one supports a set of management applications and a management information base (MIB). They also store common management data models and route management information to application and clients. Those devices to be managed that share the same network management protocol as the management servers contain agent software and are managed directly by one or more management servers. For other devices, management servers can only reach the resources through an element manager, called proxy.

Some nodes in the network are not equipped with common network management software or do not support the network management standards that are being used. To handle such cases, one of the agents serves as a proxy for one or more other nodes. When agent performs in a proxy role, it acts on behalf of one or more other nodes. A network manager that wishes to obtain information from or control the node communicates with the proxy agent. The proxy agent then translates the manager request into a form appropriate to communicate with the target system. Responses from target system back to the proxy are similarly translated and passed on the manager.

4.2 XDM

The model we have followed in the design of network management architecture is inspired on the decentralized infrastructure developed using P2P technologies.

XDM has been designed and developed for resources management in a distributed environment. The central requirement of the architecture is represented by the monitoring facility of large collections of distributed and dynamic resources.

Main task a monitoring facility should address (we do not address here for the moment another set of requirements, which are concerned with mining collected data for information) is collecting and disseminating data representing state, service data usage counters, etc of network resources, devices and PCs distributed in the environment. The monitoring facility should permit to control the load.
Figure 6: Decentralized infrastructure for network management

state at the network level (for example by monitoring in/out packets traffic, active connections), at
the system level (by measuring CPU, storage and memory utilization) and at the application level (by
controlling chosen parameters depending on the particular application). The generated information
should be collected by the manager entity using periodic polling (push) or reporting by queries
(pull).

Many applications rely on availability of information as a knowledge base on which to reason in
order to adapt their services; for example context awareness services use distributed data in sensors
equipped environment or applications based on sensor networks need an infrastructure for collecting
distributed data. A large scale system data collection and monitoring subsystem implemented using a
distributed, p2p computing approach is a more flexible solution for an architecture, able to collect data
from heterogeneous (in terms of computing resources, supported protocols, delivered applications and
services) sources.

We summarize here a number of requirements that a novel information management architecture,
which overcomes limitations in centralised architectures, address or should. This information man-
gegement infrastructure is able to monitor the dynamically evolving state of a (large) collection of
distributed resources, and distribute both raw information and summaries of data to a (possibly
large) number of clients.

It should support directory applications (e.g. building a catalog of resources in the network, map
names to meta-information), as well as performance, fault and usage monitoring applications.

Experiences in network monitoring demonstrated the need for using in a complementary way two
different styles for collecting, at the centralized (for example management station in the SNMP-based
architecture) location, information about the state of devices in the network.

In order to build a consistent model of the resourced and their data (while limiting the amount of
traffic in the network), both a reactive, driven by events sent by agents, and a proactive, based on
periodic polling from the one entity (for example a localization server), have been employed.

Also, a large variety of commercial and academic network monitors exist. Many of these systems
provide tools for collecting monitoring data in a centralized place, and visualizing the data. However,
these systems provide little or no support for dissemination to a large number of interested parties,
aggregation of monitored data, or security. The scale of these systems is often intended for, and limited to, clusters of up to a few hundred machines.

XDM has been designed supporting monitoring facility able to disseminate and collect data of a large number of dynamical resources. To support flexible interaction between components, the architecture (shown in Fig. 7) is composed by an overlay networking substrate (based on Pastry); over the overlay networking substrate, a communication component based on publish/subscribe paradigm and a data management component are built. A Query processor supporting also Long running Queries functionalities has been introduced in the architecture. The following features have been provided by the information management infrastructure:

1. Topic-based Publish/Subscribe, to allow clients to register for information of interest and to collect and distribute information.

2. Distributed Persistent Data Storage, with appropriate support for indexing and complex query mechanisms. To make this service flexible to support a variety of practical deployment scenarios, also appropriate strategies for placement of data in the network should be supported (for example, an agent should be allowed to store the generated data locally or to ask another node for storing it. Similarly, desired levels of replication should be controllable depending on the requirements of the scenario).

3. Query processor (based on XPath language) supporting also Long Running queries, by extending the topic-based publish/subscribe component.

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query processor</td>
</tr>
<tr>
<td>Long running queries</td>
</tr>
<tr>
<td>Subject-based publish/subscribe</td>
</tr>
<tr>
<td>Overlay network basata su DHT</td>
</tr>
</tbody>
</table>

Figure 7: XDM architecture

The scenario we focused on is a dynamical system, characterized by a large number of nodes (PC, devices, sensors)

However, a much larger number of nodes will access to information in an autonomic computing scenario (as well as in traditional, but large scale, environments). To support this decentralised, peer style of access to information, besides group communication features outlined above, desirable properties that the information management architecture include:

1. Flexible discovery of network and application resources and related meta-information

2. Search for devices and agents based on their characteristics, including location, and other parameters (including supported MIBs)

3. Flexible gathering of information (e.g. information of a specific type, source, location of source) from agents
4. Agent programmability — Ability for managers to install software components on agents, to perform data aggregation, information filtering et al.

4.3 Topic-based Publish/subscribe Component

The communication component is based on publish/subscribe paradigm: the communication component implements a multicast tree related to a topic creation. The component is based on Scribe [7], a large-scale event notification infrastructure for topic-based publish-subscribe applications, built over Pastry overlay networking substrate. We briefly describe the working implementation.

Any node may create a topic; other nodes can then register their interest in the topic and become a subscriber to the topic. Any node with the appropriate credentials for the topic can then publish events, and the component disseminates these events to all the topic’s subscribers. Nodes can publish events, create and subscribe to many topics, and topics can have many publishers and subscribers. Each topic has a unique topicId. The node with a nodeId numerically closest to the topicId acts as the rendez-vous point for the associated topic. The rendez-vous point forms the root of a multicast tree created for the topic. The component creates a multicast tree, rooted at the rendez-vous point, to disseminate the events published in the topic. The multicast tree is created using a scheme similar to reverse path forwarding. The tree is formed by joining the routes from each subscriber to the rendez-vous point. Subscriptions to a topic are managed in a decentralized manner to support large and dynamic sets of subscribers. Scribe nodes that are part of a topic’s multicast tree are called forwarders with respect to the topic; they may or may not be subscribers to the topic. Each forwarder (node that is part of a topic’s multicast tree) maintains a children table for the topic containing an entry (IP address and NodeId) for each of its children in the multicast tree.

Exported API (Application Programming Interface) by topic-based Publish/Subscribe component

The information disseminated by the sources are represented using XML language, by introducing a XML schema for “context data” information. The XML representation has been introduced in order to optimize the data management (also the indexing mechanism is based on XML representation) and the query processor. Each node can publish information (context data) by subscribing to a topic of interest and subscribe/unsubscribe to existing topics (and related multicast trees).

The component offers a simple API to its applications:

- **publish(topic, content)**: publishes the specified content on specified topic.
- **publish(content)**: publishes the specified content on topic extracted by the context data (topic is an attribute of context data).
- **subscribe(topic, client)**: causes the client to subscribe to the specified topic. The component makes dispatching of all messages to the subscriber nodes.
- **unsubscribe(topic, client)**: causes the client to unsubscribe from the topic.

4.4 Distributed Data Storage Component

The storage component permits to insert and retrieve files to nodes connected to the network. Files stored are associated with a quasi-unique fileId generated at the time of file’s insertion. The component is based on PAST [14], a large scale peer-to-peer persistent storage utility. The fileId is generated using a secure hashing (SHA, 160 bit), function of filename, user’s public key, randomly chosen string. The component stores the file on the k node whose nodeIds are numerically closest to the 128 most significant bits of the file’s fileId. The replica set nodes is chosen by the following: ReplicaSet(FileId) = Nodes where −NodeId − FileId are among k smallest nodes, k replication factor defined by the user. The invariant that k nodes whose nodeIds numerically closest to the
fileId is maintained over the lifetime of a file, despite the arrival, failure and recovery of nodes. The dynamical events of joining or failure of files imply a files migration, respecting the rules of storage previously defined. Another invariant is that the set of existing fileId values are uniformly distributed in their respective domains. (The same property is verified by the nodeIds) This property follows from the quasi-random assignment of filelds; it ensures that the number of files stored by each node is roughly balanced. This fact provides an initial approximation to balancing the storage utilization among the nodes.

To guarantee privacy, encryption mechanism based on DES with private key and algorithm RSA have been included. Each file have to be validated using DES algorithm with a private key, shared by nodes in the network. To do not permit to unauthorized user to read/write confidential data, the data are encrypted, using algorithm RSA, based on a pseudo digital signature. Using both private and public keys, we obtained a simple mechanism to codify the stored data and to decodify to access information.

The component supports an automatic indexing based on the attributes of the context data (represented in XML): the indexes are used by the query processor, expressing complex queries based on XPath language. The data replication has been introduced with aim to increase the data availability against failure and unpredictable data corruption. An aging mechanism, based on expiration time, has been introduced in order to avoid storage overloading, eventually caused by large sensor data sources. A bulk-transfer functionality has been developed for the management of (large) data feeds.

Exported API (Application Programming Interface) by distributed data storage component

The component API is the following:

- insert(content): causes the local node to insert the content, by indexing the context data using the key type:entityid:sourceid according to the context data representation. The index related to the context data is automatically created.

- insertbulk(content): insert in bulk modality.

4.5 Query processor

The query processor enables complex queries, using the data context representation in XML. The performance of the query processor is optimized by the indexing over the attributed fields in the context data. We report a data context example represented in XML:

```xml
<?xml version="1.0"?>
<xdmdata
  topic="position"
  type="position.location"
  sourceid="agent2.sensor1"
  entityid="position1"
  mtime="1000090909"
  ctime="1000090909"
  etime="1000100000"
  recordid="2"
  logindex="0" >
  -- specific type element, e.g.:
  <xvalue>1.0</xvalue>
  <yvalue>2.0</yvalue>
  --
```
with xmldata as root element and topic/type/sourceid/entityid as attribute fields, used also for the indexing.

The query processor supports complex queries expressed in XPath, based on W3C standard, a syntax for defining parts of an XML document.

XPath defines a library of standard functions for working with strings, numbers and boolean expressions. XML documents can be represented as a tree view of nodes and XPath uses a pattern expression to identify nodes in an XML document. An XPath pattern is a slash-separated list of child element names that describe a path through the XML document.

Example of supported queries are the following:

- query 
  `@entityid="John Doe"`

- query 
  `@type="context.position.gpsposition"`

- query 
  `startswith(@type,"context.position\"")`

- query 
  `startswith(@sourceId,"Agent3\")`

The Long running queries functionality is supported by extending the topic-based publish/subscribe component. A subscriber node can filter published events, using pattern expression over the attribute fields included in the subscribe topic. This functionality permits the subscriber node to receive only information of interest.

An example of Long Running query is the following:

Subscribe 
  
  `startswith(@topic,"Context\"Position\") AND (startswith(@type="gpsposition"),`

in which a node subscribes to the topic “ContextPosition” and receives only filtered information on type “gpsposition”.

Exported API (Application Programming Interface) by Query processor component

The component API is the following:

query(xpathquery): permits to search context data satisfying the XPath query
subscribe(topic, xpathfilter, client): causes local node subscribed to specified topic, to receive data satisfying XPath query.

4.6 Possible Extension: Extensible, Semantic Information Model

We expect that a semantically stronger formalism for expressing ontologies, and related architecture, should emerge (such as semantic web languages, description logics et al). One possible approach to handle much of these challenges is to use Semantic technologies. We aim to exploit two main features of Ontologies and use these features to improve interoperability of the management information architecture. Interoperability is achieved by the flexible and extensible way of modelling the world with Ontologies. Every term which is used has a clearly defined meaning. Through the usage of new standards from the W3C, namely resource description framework (RDF) and web ontology language (OWL), the syntax is well defined. These standards base upon XML, so a broad set of tools to handle them is available. The extension will be object of collaboration among TILS and UPB.

4.7 Related work

A number of systems have been proposed for distributed network monitoring [3,4,5]. Astrolabe is an SQL-like query system focused specifically on aggregation queries for network monitoring [3]. Astrolabe provides the ability to define materialized aggregation views over sub-nets, and to run queries that hierarchically compose these views into coarser aggregates. Astrolabe provides a constrained subset of SQL that sacrifices general query facilities in favor of a family of queries that exploit this hierarchy efficiently. This contrasts with the flat topology and general platform provided by PIER.
PIER, a database-style query engine intended for querying the Internet. PIER is a three-tier system. Applications interact with the PIER Query Processor (QP), which utilizes an underlying DHT [5].

References


[5] R. Huebsch, J. M. Hellerstein, N. Lanhamand alii, Querying the Internet with PIER


[14] A. Rowstron and P. Druschel “Storage management and caching in PAST, a large-scale,persistent peer-to-peer storage utility”, ACM Symposium on Operating Systems Principles (SOSP’01), Banff, Canada, October 2001.


[16] W. Stallings, SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Addison Wesley.


