# Collaborative and Integrated Network and Systems Management: Management using Grid Technologies

Mohammad Hassan
Faculty of Information Technology, Al-Ahliyya Amman University, Jordan

**Abstract**: *Current Internet trends are moving towards decentralization of computation, storage, and resources. Supporting network management for such a vast and a highly complex system has become a challenging issue. A management platform has to sufficiently support decentralization, collaboration, and integration. Grid technologies have the potential to serve as management architecture due to the support of the above features. In this paper, we developed a collaborative network management architecture leveraging the key features of grid technology. Benefiting from this integration, we were able to show that multiple management tasks can be integrated and completed in parallel. This assures the management scalability and efficiency. We also showed that the management information at different networking domains can freely consume the computational resources provided through the grid interface while being executed. Grid interface has guaranteed scalability and reliability for the network management tasks. We have simulated the system prototype and closely studied its efficiency.*

**Keywords:** *Grid, network and systems management, integrated management, virtual organizations.*

*Received September 12, 2011; accepted December 30, 2011; published online August 5, 2012*

## 1. Introduction

Network and system management platforms were based on simple centralized architectures. Centralized architectures have shown serious deficiency in managing current complex networks, such as the Internet. This has led to more complex and distributed architectures for network and system management. Throughout their development, management platforms have passed through intermediate stages such as weakly distributed control systems, strongly distributed control systems, domain based systems, and active distributed management systems [12].

Computing has been recently extended from standalone PCs to Local Area Networks (LAN), and beyond the local network boundary to open networks including the Internet, Wide Area Networks (WAN), and wireless networks. With the assistance of decentralized and high speed networks, computing devices are able to communicate and collaborate with each other regardless of their geographic locations or device types (e.g. PCs, cell phones, handheld devices). Mergers and takeovers, as a result of globalization, very often require computing to be performed in a distributed, heterogeneous, and complex environment. E-services also require service transactions to be conducted between systems over a distributed network such as the Internet. New technologies such as Distributed Object Computing, Web Services, Peer-To-Peer, and Grid Computing [3], are designed to serve these purposes. The Open Grid Services Architecture (OGSA) has recently emerged as a 'second generation' distributed computing approach to grid middleware that is taking grid support forward from an era of Ad-hoc platforms to a more architected approach built on service orientation and web services technologies [10].

Nielsen's law of Internet bandwidth [11], states that the high-end user's connection speed is growing by a rate of 50% annually. On the other hand, Moore's law [9] states that the number of transistors per square inch on integrated circuits doubles every year. However, recently it has doubled every 18 months (i.e., approximately 60% per year), and apparently experts expect it to stay at this level for sometime. Combining these two laws we can determine that the bandwidth is therefore growing at a slower rate than processing power. Therefore, higher performance computing should undergo in to a considerable change since the connection performance of the Internet is not progressing as fast compared to computing speed. Applications that once were tightly coupled and complex are now decentralized with collaborating components spread across diverse computational elements.

Collaborative computing is an emergent trend that requires not only distributed computing capability but also faultless interoperability between different operating systems. Furthermore, interoperability necessitates standard communication protocols and a universal data exchange format. Therefore, we see that

future network and systems management should inherit similar development trends [1].

This paper presents a novel integrated and collaborative network management system using grid technologies. Given *n* number of management platforms on *k* different management domains each with *p* number of different management tasks, we aim to develop an efficient and highly collaborative network management system that satisfies distributed administrative control [8]. Network and system management information is growing in terms of complexity and size. Efficiency is a performance measure to indicate the effective utilization of bandwidth within the network [1, 3]. Collaboration refers to how easily and accurately different distributed management domains share managerial information of each [3].

The new grid-based management system is composed of two dynamic servicing tiers. The user level tier consists of users that have some management tasks. Once the task is verified against the allowed operations stored in the domain virtual organization, the user contacts the service tier to request a completion of the task. The service tier accepts the submitted task and work on behalf of the user to complete the requested management task. A management task may require multiple domains to collaborate and operate their agents in a predefined sequence to obtain a successful user task.

The paper is organized as follows: Previous works and some backgrounds are described in section 2. Section 3 details the system architecture and the proposed system framework components. Section 4 provides the analytical and simulation results performed on the system followed by section 5, where a conclusive discussion is summarized there.

## 2. Background

Network management has been defined as "all measures ensuring the effective and efficient operations of a system within its resource in accordance with corporate goals" [6]. As defined by ISO framework, network management consists of five abstracted areas: performance, configuration, account, fault, and security [6].

Over the past decade, network and system management has increasingly evolved from centralized paradigms, to distributed paradigms [12]. Network management started with a simple centralized platform called SNMP [12]. IP networks have developed dramatically and support many applications that are distributed in nature. RMON, SNMPv2 and SNMPv3 [12], management frameworks are active examples of the early stages of distributed management platforms. As IP networks develop and open new opportunities for new applications, new management paradigms are being proposed. For example, mobile code, distributed objects, collaborative paradigms, enterprise system, active programmable management and domain based management, and recently grid based management [12].

### 2.1. Grid Technology

Depending on many factors, grid is defined in different ways. Fosters [3], defines grid as "The sharing that we are concerned with, is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a virtual organization" [1, 13].

The previous definition highlights four main components of a grid that is of network management interest: First, due to a wide variety of resource and resource types, grid integrates and coordinates resources and users that exist within different management domains [7, 8]. For example, user's desktop vs. central computing, different administrative departments of the same institution vs. those of different institutions. This coordination addresses the complications of security, policy, accountability, and ownership. Second, grid is assembled from standard and open multi-purpose protocols and interfaces that address security, resource discovery, resource allocation, and resource access [2, 3]. Third, grid allows its available resource components to be used in a coordinated manner, such that various QoS can be provided. QoS are responsible for controlling delay, throughput, availability, and security of user's applications [1, 2, 3,]. Forth, grid environments are designed to provide access to the available resources in a faultless manner [2, 3]. Clearly, the above listed grid features are of great support to build a collaborative and integrated network and systems management platform.

Grid computing is an important emerging technology where enterprise distributed management can gain much advantage from. Information has to be passed between the various entities involved in a grid, and Web Services are emerging as the preferred method. The OGSA and the Open Grid Services Infrastructure (OGSI) are two grid computing standards, which specify web services as the method of allocating work to grid providers [2, 4].

Generally, grid components can be categorized based on their purpose as a processing element, a network element, or a storage element. Processing elements very among single processor, multiprocessor,

cluster, and parallel processing systems. Network elements are basically routers, switches, gateways, virtual private network devices and firewalls. Storage elements are network attached storage devices such as automated CD-ROM/DVD, data warehouse, or a dedicated database machine. Literature categorizes grid types into three categories, computational, data and service grids. Each of these will be used to facilitate the functionality of the new proposed management platform [7].

## 2.2. Grid Management Properties

Drawing from grid concepts mentioned in the previous section, we have identified the following properties that we foresee are necessary for constructing our proposed distributed, collaborative and integrated network management platform [5].

1. *Scalability:* Grid concept is proposed such that it has the ability to expand the number of tasks or increase the capabilities of computing, storage, and communication without making major changes to the systems or application software. This implies that complex distributed management tasks can be handled by grid in scalable and efficient manner.
2. *Collaboration:* Grid technologies and infrastructures support the sharing and coordinated use of resource and information in dynamic distributed Virtual Organization (VO). VO that is defined as "the creation from geographically distributed components operated by distinct organization with different policies, or virtual computing system that are sufficiently integrated to deliver the desire QoS" [5]. Enterprise management system should construct a pool of management information that is shard among different management domains. Grid facilitates information sharing thought pre-defined policies. These polices are maintained in a form of virtual organizations.
3. *Standard Interface:* Grid defines Grid Service as "Web Service that provides a set of well-defined interfaces and that follow specific conventions" [4]. This interface addresses discovery, dynamic service creation, lifetime management, notification, addresses naming, and upgradeability. Pre-defined interface simplifies the global interaction with the management system.
4. *Customizability:* Grid defines a basic view of servicing model that is necessitating the usage of factories for service creation. Factories have the ability to dynamically create and manage new service instances. These instances have different pre-designed tasks. Management requests are now performed and controlled through service instantiation from factories. These factories are designed such that it covers certain applicable management tasks.
5. *Service Description:* Grid defines service data structure, which is a required mechanism for discovering available services, determining their characteristics, and configuring grid applications and their requests to match those services. In addition to the service data, grid standard OGSA [3, 12] defines a standard operation, FindServiceData, which retrieves service information from individual service instances, and a standard interface for registering information about grid service instances with services registry. Management applications can query the service description such that it locates the desired management tasks in an accurate manner. Several management services that can be used to perform multiple management tasks in parallel.
6. *Unique Service Addressing:* Grid uses URIs to address the offered grid services. Grid service URI is called the Grid Service Handle, or simply *GSH*. Each *GSH* is unique. There cannot be two grid services (or grid service instances) with the same *GSH*. This unique service addressing simplifies the localization of the management service.
7. *Autonomous Execution:* Tasks or commands are executed in a management domain as an autonomous, management command. This implies that management task does not need to coordinate with a master process running outside the management domain or other management station.
8. *Notifications:* Grid defines task notification services. Notifications are a collection of dynamic, distributed services that are able to notify each other asynchronously of changes to their state. Grid standard defines common abstractions and service interfaces for subscription to and delivery of task notifications, so that services constructed by the composition of simpler services can deal in standard ways with notifications of, for example, errors, task completion, and results. Notification is an essential feature that a management system should consider. Management platform will benefit from this facility by tracking launched tasks, getting task execution state, and task completion.
9. *Service Lifetime Management:* Grid services can be created and destroyed dynamically. They can be destroyed explicitly. They also can be destroyed or become inaccessible through a system failure such as an operating system crash or a network partition. Interfaces are defined for managing a service's lifetime and, in particular, for reclaiming the services and state associated with failed operations. grid standard addresses this requirement by defining a standard SetTerminationTime operation within the required grid service interface for lifetime management of grid service instances. Lifetime management protocols let grid platform eventually discard the state established at a remote location unless a stream of subsequent keepalive

messages that shows an application interest in that particular operation refreshes it. This implies that the management applications have a complete control over the resource and the execution stages.

10. *Efficient Data Transfer:* Grid defines new version of standard FTP called GridFTP [14], which is defined as "a set of extensions to the FTP that provide increased security, reliability and performance to data transfers" [14]. GridFTP provides parallel and third-party control of data transfer, striped data transfer, partial file transfer, and reliable data transfer through fault recovery methods. Management information that requires further analysis and decision making may be huge in size. GridFTP can be used to simplify the transfer or such files.
11. *Availability:* Grid provides the required resources and services that are not provided by any single machine [2]. These resources vary from high throughput parallel machines to distributed collaborative desktops. Also, from small databases that is connected using simple LANs to data warehouses that are connected using fibre optics networks.

The above mentioned grid properties permits us to define our management platform based on grid in such a way that it can be used to facilitate network and systems management tasks in autonomous, scalable, and collaborative manner.

## 3. Network Management using Grid

Current high performance computing applications are decentralized, loosely coupled, and complex. These applications use collaborating components spread across diverse computational elements. Such distributed systems most commonly communicate through different exchange message formats and data structure. As a result of the mentioned growing trend, we propose a management platform that exhibits some emerging attributes such as, delocalization, collaboration, and is component based in nature.

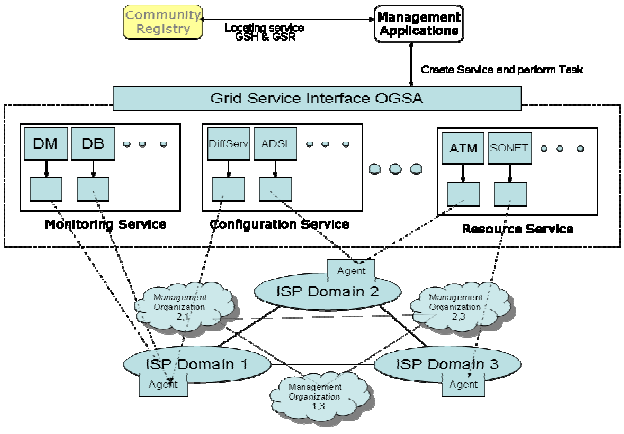


Figure 1. Architecture of collaborative and integrated management using grid.

Figure 1 shows an overview of our proposed collaborative and integrated network management architecture. In each ISP/Management domain there exists a management agent. This agent is a software implementation that is responsible to perform management tasks that are sent from the management applications.

It also consists of the several service implementations that perform designated tasks. Furthermore, it is responsible to monitor, gather, and share the domain management information that the domain agreed to provide based on virtual organization policy that get established. At the grid servicing layer there exist different management services such as monitoring, configuration, fault detection, resource management, etc.

Management applications are launched on a format of grid application. The advantages of this approach consist of performing multiple and different management tasks in an autonomous and parallel manner.

Figure 2 illustrates some aspects of grid that is considered as an advantage to network and systems management operations. This example scenario presents a situation where a grid based management task wants to discover, monitor, and employ remote configuration, such that it assigns more resources to a certain high-bandwidth application. It uses a management information domain from a number of distributed domains based on provided management information from established virtual organizations. From this scenario we observe that this management task requires some service composition associated with some optimization and queuing techniques. For now we will focus on the system interaction to accomplish such a task.

From Figure 2, we illustrate the discovery, monitoring, and employment of a remote configuration of a management service as follows:

1. The management application, which could be a program that acts on behalf of the user first contacts a community registry to obtain the information that is relevant to the required task. Keeping in mind that virtual organization maintains a record such that it identifies the service providers who can provide the required services. In our example the user requires services for monitoring and configuring a segment of a network.
2. The registry returns both grid service handler, which as we said earlier is unique, and grid service reference that identifies the factories for monitoring, configuring, and queuing a management tasks.
3. The user issues requests to monitoring, configuring, and task queuing factories specifying details such as the required monitored information, how to be performed, in what order the tasks are to be performed, and the format of the results.

4. Assuming that this negotiation process proceeds satisfactorily, three new service instances are created with specified initial lifetime stamp.
5. The monitoring service initiates monitoring request against appropriate remote monitoring service that is located in the desired domain.
6. Similar to 5, the configuring service instantiates a configuring service that starts a connection with the designated device.
7. If the task in 5 passes correctly, results are returned from monitoring process to the configuration service.
8. Depending on the started lifetimes, the management application can keep issuing periodic *keepalive* messages to indicate continued interest in such a service. Monitoring or configuring delays may occur for several system reasons such as congestion, queuing. Management application can also subscribe to the task notification mechanism provided by grid standard implementation [14].
9. Once the monitoring process returns the required data, the management application terminates the instance.
10. The application then remains in control of the configuration task until it completes either successfully or an error message gets returned. In either case, new future scenarios might take place.
11. The task queuing service remains active to ensure the configuration commands get executed in a pre-defined and correct manner.

The above illustrated management scenario can be implemented in several ways. The Grid standard facilitates a distributed implementation of the service registry. Furthermore, it allows an integrated service information gathering.

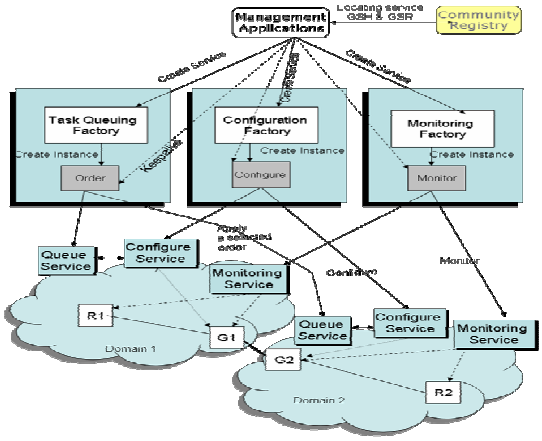


Figure 2. Monitoring and configuration management task scenario.

For example, an intermediate service broker can be involved in service query and assignment. These details are abstracted from the management application and it will just get the relevant service and task information.

## 4. System Construction

Our system consists of few main actors in the management task process. First, the management program that is in a format of a grid application. Second, the management service that is a grid service which conforms to management behaviour. Third, the management agent that is responsible for performing the task. There exist several management services that are formed in order to facilitate the queue of user tasks in certain order of priority. Since management commands are prioritized in nature, we should implement a queuing model for service aggregation and command prioritization to facilitate the implementation of the system.

### 4.1. System Platform

Our proposed management system composes of two dynamic servicing tiers. The first tier consists of management tasks in a format of a grid application that requests management services to perform a management task. The second tier consists of management services that work on behalf of the user to complete the requested management task. Analytically, our system consists of two servicing tiers. Figure 3 illustrates the system architecture.

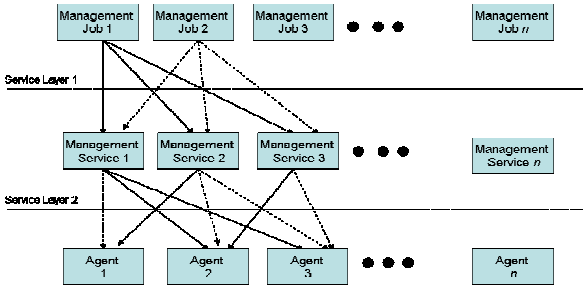


Figure 3. Two levels servicing system.

The management task is an application, which is a program that acts on behalf of the user. It first contacts designated VO records to obtain the information that is relevant to the services which are able to perform the required task. The VO then returns the required services and the method to communicate with them. Furthermore, the VO provides some QoS criteria associated with selected services. The user application constructs different service ordering candidates in a format of a connected graph. The constructed service graphs are compared to the predefined management service sequence provided by the VO.

The optimal graph is selected based on both reliability and performance. A user then issues requests to the services to perform management tasks, specifying the management details, such as the required data, method of performance, in what order the tasks are to be performed, and the format of the results. On behalf of the user, the management service starts to call the agents in the user specified order.

Once the services return the required results, the management application terminates. The user management task remains in control of the performed management job until it completes their application.

### 4.2. Service Aggregation Model

Earlier we defined a management task as a combination of interrelated subtasks that must be executed in certain order before the entire task can be completed. The subtasks are interrelated in a logical sequence in the sense that some can not start until others are completed. Therefore, the system consists of a task and service scheduling that must be addressed. Our system follows a similar analogy. For each management job, we assume that it consists of a list of subtasks that are connected in certain order to represent a task. In abstraction, the system is represented by a set of finite points that represents the management services, some of which are connected to others using lines, which represent the correct service composition order. Each service has an associated cost that is defined as a summation of computation and communication time. The problem is finding the best service order such that we achieve the management task correctly. Our system takes an advantage from the virtual organization concept illustrated earlier. Every management domain provides the correct order of tasks that are required to accomplish the management tasks. Therefore, our system has a reference structure to select the correct service management services from Service Aggregation Component (SAC). It constructs all feasible service compositions and represents them in graph structure. Figure 4 illustrates the service model.

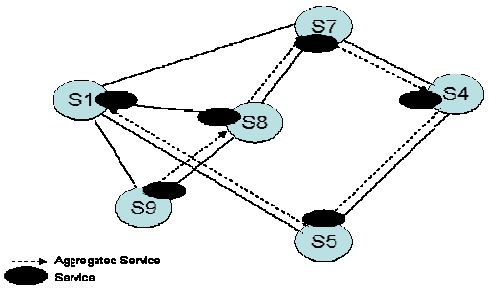


Figure 4. Service aggregation model.

SAC uses the proposed service order provided by virtual organization of the collaborated management domains. The system then selects the most efficient and reliable service composition structure. The selection process is based on the service reliability and performance model that is built either from experience using some AI techniques, such as neural networks, or directly supported by the domain administrators and stored in the virtual organization. In our system, we do rely on the virtual organization to support the required factors. Every management task should be registered in a separate service called Service Tracking Component (STC). STC acts as a management auditing service. It basically registers a management job and all associated jobs to ensure a correct and successful completion of a management task. A role-back mechanism can be triggered once a sub-task has failed to operate successfully. Furthermore, STC saves all successful or failed management tasks for the domain administrator's reference.

## 5. System Performance Evaluation

Experimental results presented in this section have been generated from a simulated architecture of a collaborative and integrated management system using grid services. The framework was simulated using an event-based simulator written in JAVA. The simulator components and event types correspond to the different layers of the system. Each layer has a different execution domain. Virtual organization, management tasks, computing resources, services, domains, and management agents are represented in the system.

From the simulation, the user occurrence inter-arrival time interval remaining until the occurrence of the next user event is totally independent of the time which has elapsed since the occurrence of the last event. Second, the user arrival events follow exponential distribution. Therefore, we have used the Poisson distribution for user generation process. For this purpose, we implemented exponential and random functions to generate the inter-arrival rate ($\lambda$) and grid agent service times ($\mu$). Basically the function implements a continuous random variable following exponential process, $f(x) = \lambda e^{-\lambda x}$.

We have defined four management operations which are the tasks register, read, write, and notify. Each user is randomly assigned a list of management tasks. Since there are four management tasks, there exist also four grid management service factories. Each factory is called by a user based on the generated user tasks. The architecture topology is simulated based on creating a management domain, which consists of management agents and virtual organizations.

Agents are created and assigned domain and unique agent *IDs*. Virtual organizations are created in between two collaborative management domains. The user management jobs are created and assigned to each user randomly. Users first verify the assigned management jobs against the ones allowed by the management domain and stored in the virtual organization of that domain. If it matches a task sequence, the operation will proceed and a service factory will be called to proceed with the operation. Else, there exist two options: first, the virtual organization will return a proposed operation to the user.

The user then verifies the proposed job sequence and either agrees or refuses to continue. Once the job has reached the correct service, it is processed based on calling agents in a correct sequence. Second, the task is

not allowed and the user terminates. Management operations are simulated on one and many management domains. The system simulation has been examined using three different topologies.

Table 1. Simulation topologies.

| Domains | VO Jobs | Management Services | Agents |
|---|---|---|---|
| 3 | 4 | 5 | 20,16,23 |
| 5 | 4 | 5 | 24,21,3,2,10 |
| 7 | 4 | 5 | 4,22,20,3,12,23,15 |

Table 1 shows the simulated topology parameters where the first column represents the number of domains, the second column represents the number of virtual organization job, the third column represents the number of management services required at the grid layer, and finally the last column represents the number of involved management agents. In the simulator, we have varied the number of domains, domain agents, job execution sequences, and virtual organizations to obtain different performance results. Our first experiment was to computing the average job success ratio for a submitted user management tasks that has been validated and allowed into the virtual organization for execution in different topologies. Figure 5 graphs the numerical values, which resulted from computing the average success ratio of user management jobs. From this figure, we found that the increasing number of domains have a little impact on the overall system performance. This is due to the fact that grid interface and grid resources were always able to scale with the large user arrival tasks.

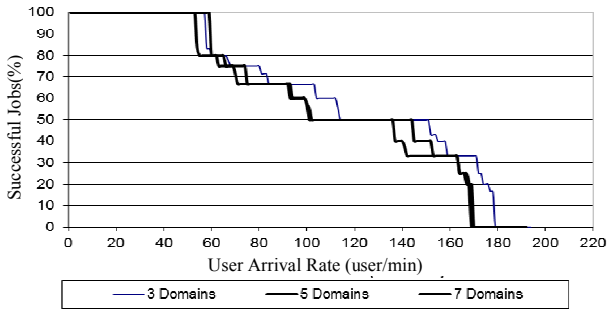


Figure 5. Successful jobs under different user arrival rates.

Our second experiment was done to compute the ratio between the user management requests and the number of agents invoked. Figure 6 illustrates the numerical results. From this experiment, we conclude that the number of launched agents is increasing almost linearly with the number of management jobs at different domain sizes. This is also a supporting and interesting behaviour gained form the integration with the grid technology. Our last experiment was done to examine the ratio between the number of created services and the number of management agents involved. In this experiment we observed that at certain number of users and their jobs (e.g., 100 users) the number of launched services remains constant at around 250 services.

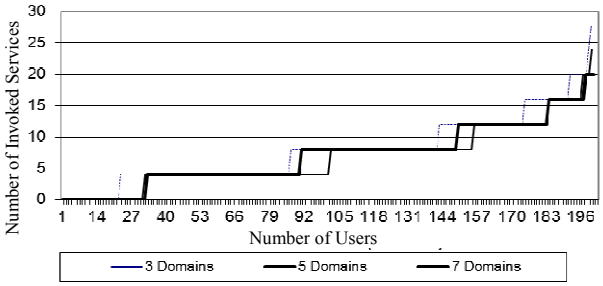


Figure 6. Number of involved services per users jobs.

This is also one of the major gains from employing the grid interface where agents are distributed in a way that it can handle the continuous increase of users without having to increase the agent instances. Figure 7 illustrates the numerical results.

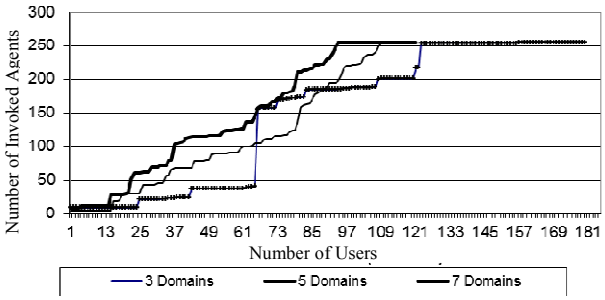


Figure 7. Number of services and involved agents.

The above simulation results have showed that our proposed network management architecture provide far better task handling, scalable with the increase number of users, and adaptive to the increase number of management tasks and management agents. We also observed that the architecture can operate well with the increase number of management domains.

## 6. Conclusions

In this paper we proposed and evaluated new network management architecture using grid technology. The architecture is generic, flexible, and supports different systems and network technologies. The proposed architecture provided several advantages to network and systems management. We showed through an extensive simulation that the proposed architecture is scalable where it supports a wider coverage of different management tasks and the deployment of different management protocols over a wider geographical area. We have also showed that the concept of virtual organizations eliminates management faults that are caused because of incorrect sequence of management job submissions. We also showed that the idea of services and service factories facilitate the adaptability of the system to different customized management tasks. Finally, the anonymous deployment and execution of management jobs simplify the

architecture and keep the management information private to the domain and its registered users.

## References


[1] Czajkowski K., Fitzgerald S., Foster I., and Kesselman C., "Grid Information Services for Distributed Resource Sharing," *in Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, USA, pp. 181-194, 2001.

[2] Foster I., Kesselman C., and Tuecke S., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal on Supercomputer Applications*, vol. 15, no. 3, pp. 200-222, 2001.

[3] Foster I. and Kesselman C., *The Grid: Blueprint for a New Computing*, Morgan Kaufmann, USA, 2004.

[4] Foster I., Kesselman C., Jeffrey M., and Steven T., "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," *Technical Report*, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

[5] Foster I., Kesselman C., Jeffrey M., Steven T., "Grid Services for Distributed System Integration," *IEEE Computer Society*, vol. 35, no. 6, pp. 37-46, 2002.

[6] Hegering H., Abeck S., and Neumair B., *Integrated Management of Network Systems*, Morgan Kaufman Publishers Inc., 1999.

[7] Krauter K., Buyya R., and Maheswaran M., *A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing*, International Journal of Software: Practice and Experience, Wiley Press, USA, 2002.

[8] Lewis L. and Onge D., "Method and Apparatus for Integrated Network Management and Systems Management in Communications Networks," *Technical Document*, US Patent 7,007,104 B1, 2006.

[9] Moore G., "Moore's law of Integrated Circuits," 2004.

[10] Muruganantham S., Srivastha P., and Khana a., "Object Based Middleware for Grid Computing," *Journal of Computer Science*, vol. 6, no. 3, pp. 336-340, 2010

[11] Nielsen J., "Nielsen's Law of Internet Bandwidth," avalibale at: http://www.nngroup.com/ articles/law-of-bandwidth/, last visited 1998.

[12] Rayan S., Pradeep R., and Paramesh N., "Network Management Platform Based on Mobile Agents," *International Journal of Network Management*, vol. 14, no. 1, pp. 59-73, 2004.

[13] Wilkinson B., *Grid Computing: Technologies and Applications*, Chapman & Hall/CRC Taylor & Francis Group, 2009.

[14] Yaylor I., Shields M., Wang I., and Harrison A., "Visual Grid Workflow in Triana," *Journal of Grid Computing*, vol. 3, no. 3-4, pp. 153-169, 2005.


**Mohammed Hassan** assistant professor at Software Engineering Department, Amman Al-Ahlyya University, Jordan. Received his MSc and PhD degrees in information processing systems from Baku State University, Azerbaijan. He has joined the Academia Sector since 2001. His research interests include grid networks, middleware, e-learning, e-government, and software development methodologies.