Architecture of a Network Monitoring Element

Augusto Ciuffoletti[1] and Michalis Polychronakis[2]

[1] CNAF-INFN, Bologna, Italy
[2] FORTH-ICS, Heraklio, Greece

Abstract. A Network Monitoring system is a vital component of a Grid; however, its scalability is a challenge. We propose a network monitoring approach that combines passive monitoring, a domain oriented overlay network, and an attitude for demand driven monitoring sessions. In order to keep into account the demand for extreme scalability, we introduce a solution for two problems that are inherent to the proposed approach: security and group membership maintenance.
Keywords: network monitoring, passive network monitoring, on demand network monitoring, network monitoring element, scalability issues, security issues

1 Introduction

Monitoring the network infrastructure of a Grid has a vital role in the management and the utilization of the Grid itself. The Global Grid Forum (GGF) schema [7], splits this activity into three distinct phases: *production*, *publication*, and *utilization* of measurements. Here we focus on the production and publication, with a special concern for scalability: for measurement production we address the usage of passive network monitoring techniques, while for the publication activity we adopt a domain-oriented overlay network which reduces the complexity of the task.

The challenge comes from the fact that a Grid oriented network monitoring should address network routes, not single links, since this is the kind of information needed to optimize distributed applications. Since each pair of Grid Services should be individually monitored, this makes an $O(n^2)$ complexity for many aspects of network monitoring: from the size of the database containing the results, to the number of pings that probe the system.

We combined a number of ideas in order to limit the complexity of our solution: *i*) monitoring shouldn't address single Grid resources, but pools with similar connectivity; *ii*) monitoring tools shouldn't inject traffic, but observe existing traffic; *iii*) monitoring activity should be tailored on application needs. Only the integration of above ideas can effectively control the problem size, and, in some sense, the first two *open the way* for the application of the third one.

We observe that a Grid *topology* is made of pools of resources reachable through dedicated ingress points: the accessibility of such pools depends on ingress points connectivity, and local administration avoids internal bottlenecks.

Therefore the monitoring topology can be simplified by monitoring *Network Elements* between ingress points of distinct pools.

One Network Monitoring architecture, called GlueDomains [3], has been recently designed and prototyped according to a two levels hierarchical overlay; the purpose of such experiment was mainly the assessment of a number of design principles. A Grid-wide deployment of GlueDomains was carried out during the summer of 2006, as part of the Italian branch of the Large Hadron Collider Computing Grid Project (LCG). Apart from the statistics collected (usual packet loss and roundtrip time, together with an experimental one way jitter measurement tool, published through the GridICE Grid Information Service [2]), the most relevant results from the GlueDomains experiment concern the ease of deployment, as well as the resilience, and stability of the architecture, which were assessed during a one month trial. GlueDomains is included in the current release of the Italian branch of LCG.

GlueDomains architecture centers around a number of specialized units hosting the agents in charge of monitoring the network. Such agents are able to autonomously (re)configure their activity based on a dynamic description of the network monitoring topology, available from a relational database. The monitoring activity was based on a domain partitioning of Grid resources: the target of such monitoring is the *Network Element*, which abstracts the network infrastructure in charge of interconnecting two domains.

One relevant lesson learned from GlueDomains experience is the identification of the role played by the agent that concentrates the network monitoring activity for a domain. This role corresponds to a new resource in the Grid architecture, which is mainly dedicated to network monitoring. In the architecture proposed in this paper we call such agent a *Network Monitoring (NM) Element*: its activity is organized into *Network Monitoring Sessions*.

Another cornerstone concept in our architecture is *passive monitoring*, which is non-intrusive by nature. The internal architecture of NM Elements adopts specific hardware and software solutions to address passive network monitoring.

A third concept that cuts down network monitoring complexity is an *application driven* configuration: this is feasible in a Grid, where applications negotiate computing resources with resource brokers, which can configure *Network Monitoring Sessions* on the fly, providing adequate credentials to NM Elements. The relevant conclusion is that, if network monitoring activity is bound to applications, it will increase linearly with system throughput, not with the square of system size.

A relevant aspect of an *application driven* approach is the interface that a NM Element should offer to the outside. Currently brokers find resource characteristics in the *Grid Information System* (GIS), automatically collected by *preconfigured* network monitoring sessions. This attitude is inappropriate in an *application driven* scenario for its limited scalability, and it would be preferable to connect NM Elements to brokers through a publish-subscribe system. Given that this aspect is still a research topic, we indicate a composite interface, which decouples the input, consisting of monitoring requests, from the output, consisting of observation records.

The scheme described above is based on some knowledge shared by all the NM Elements, which can be assimilated to a *group membership*. Such common knowledge consists of the certificates needed to enforce security, complemented by the composition of Domains. Such data should be readily accessible by any Grid component, although the throughput for access operations can be quite asymmetric: frequent read queries should be performed promptly, while infrequent updates can be treated lazily. We address this problem by replicating this directory on each NM Element, and using an epidemic algorithm in order to maintain consistency of distinct network views.

2 Inside view of a *Network Monitoring Element*

The internal structure of a *Network Monitoring Element* is layered according to the scheme in Figure 1. The upper layer is in charge of implementing the interfaces to the outside, offering a *Network Monitoring Service*.

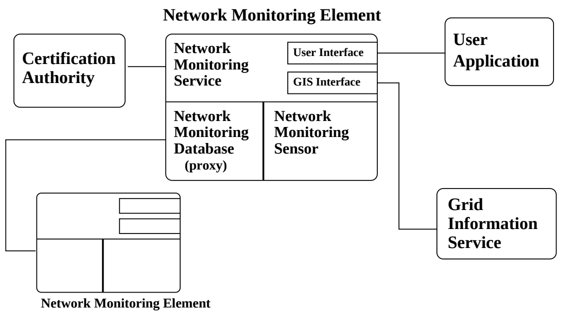
Network Monitoring Element
Certification Authority
Network Monitoring Service
User Interface
GIS Interface
User Application
Network Monitoring Database (proxy)
Network Monitoring Sensor
Grid Information Service
Network Monitoring Element

Fig. 1. Interfaces between the NM Element and other Grid components.

The NM Service offers three distinct interfaces: one for user applications (resource brokers included), another for the *Grid Information Service* (GIS), and one that interacts with the Certification Authority. The *User Application Interface* allows the submission of a request for a specific monitoring session: the NM Service checks broker credentials and verifies local resources availability before accepting a request. In response, the broker receives an acknowledgement. The User Application Interface also provides users with access to the Network Monitoring database. The *GIS interface* allows the publication of observations coming from network monitoring sessions. We do not explore the architecture of the GIS in this paper, but we note that it should enforce certain access limits: for instance, the results of an *on demand* network monitoring activity should be visible, as a general rule, only to trusted users. The GIS should be informed of such limited access by the NM service which received the request.

The lower layer is composed of two distinct modules that do not interact with each other. The *Network Monitoring Sensor* supports monitoring sessions: the implementation of sessions is delegated to specialized modules that take their configuration from the upper layer. The results of the monitoring activity are delivered to the NM Service via dedicated one-way streams from the specific session to the upper layer. We distinguish between *preconfigured* and *on demand* sessions: the former are configured directly by the NM Service module using the Grid topology described by the Network Monitoring Database, while the latter are configured by an outside user application, through the NM Service. The *Network Monitoring Database* describes the domain partition of the Grid, as well as its components: for each element in the Grid (NM elements, Computing, Storage etc.), the database holds a certificate (which contains a reference to a domain) for the element, together with other relevant attributes.

2.1 The Passive Network Monitoring Component

The NM Sensor receives measurement requests from the NM Service: its interface is summarized in Table 1, and Figure 2 describes its internal architecture.

The NM Service creates a new measurement session by sending a `create` request, which specifies the type of the measurement and any measurement-specific parameters, and returns a measurement identifier (`mid`). The creation of a new measurement session does not imply that the measurement will immediately begin upon the receipt of the `create` request; this allows the NM Service to activate or deactivate the measurement, also depending on resources availability and timing requirements.

The measurements are carried out using specialized modules implemented on top of MAPI [18], an API for building passive network monitoring applications.[3]

The basis of MAPI is the *network flow* abstraction, which is generally defined as *a sequence of packets that satisfy a given set of conditions*. These conditions range from simple header-based filters, to sophisticated protocol analysis and content inspection functions.

The back-end of the NM Sensor consists of the basic components of MAPI, namely the monitoring daemon `mapid` and the communication agent `commd`. Packets are captured and processed by `mapid`: a user-level process with exclusive access to the captured packets [18]. The monitoring modules are built as separate applications on top of MAPI. MAPI internally handles all the communication between the modules and the monitoring daemon, making it completely transparent.

The computed results of a measurement are pushed back to the NM Service either on-the-fly, or upon the end of the measurement: the desired behavior is passed in the `create` request.

In the rest of this section we describe the operation of the modules collecting some relevant network metrics.

[3] MAPI is available at `http://mapi.uninett.no`

Function	Parameter	Description
`create`	Type	Measurement type: traffic load, packet loss, or RTT
	Arguments	Measurement-specific parameters
	Return value	Measurement session identifier (`mid`) or error type
`start`	Identifier	The `mid` of the session to be started
	Return value	Acknowledgement or error type
`stop`	Identifier	The `mid` of the session to be stopped
	Return value	Acknowledgement or error type
`close`	Identifier	The `mid` of the session to be terminated
	Return value	Acknowledgement or error type

Table 1. API of the NM Sensor.

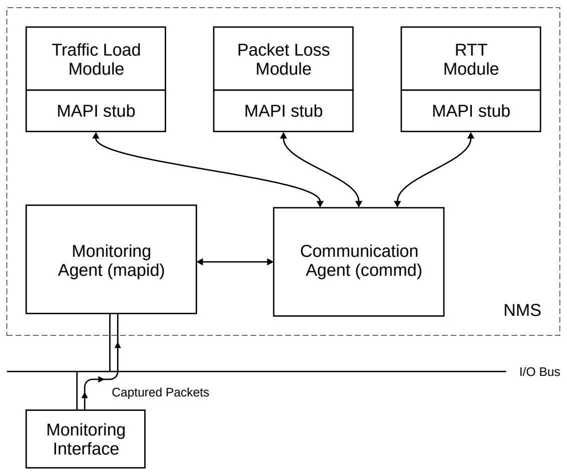

Fig. 2. The architecture of the Network Monitoring Sensor.

The *Network Traffic Load* module provides traffic throughput metrics of varying levels of granularity by passively measuring the number of bytes transferred through the monitored link. Besides aggregate throughput, fine-grained per-flow measurements are available for observing the throughput achieved by specific applications or hosts. MAPI supports generic BPF filters [14], as well as more fine-grained filtering using pattern matching in the packet payloads through string searching or regular expressions.

An estimation of the *Packet Loss Ratio* between two domains is measured by two cooperating observation points, which keep track of packets of specific flows that do not reach the destination within a timeout period. The packet loss module needs traffic information from both ends. This is achieved by creating a network flow in the local sensor, which keeps track of the outgoing packets with a destination IP address that belongs to the remote domain, and a second network flow at the destination domain, specified by the `dstdomain` parameter. The NM Element of the other domain is instructed to create a second network flow, which keeps track of the incoming packets with a source IP address that belongs to the local domain. The packet loss ratio is then estimated by correlating the data from the two network flows.

The *Round-Trip Time* is estimated using the time difference between the `SYN` and `ACK` packets exchanged during the lifetime of existing TCP connections [12]. Each request specifies the destination domain for the end-to-end RTT measurement using the `dstdomain` parameter. The module then creates a network flow that captures the `SYN` and `ACK` packets of existing TCP connections between the two domains, in the unidirectional flow from the local to the remote domain. RTT is estimated from the time interval between the last-`SYN` and the first-`ACK` that is sent from the local to the remote domain. The accuracy of the measurement increases with its duration, since a longer duration allows for more TCP connections to be tracked, which gives a better RTT estimation.

2.2 Outline of a secure group membership scheme

Security issues impose the use of certificates in order to identify the source of configuration inputs to NM elements: this can be assimilated to the secure management of a membership. An efficient and scalable certificates distribution scheme is required [19], conceptually based on the NM Database, where certificates are stored. Access to this database must be secure and scalable, and characterized by a *small* read access latency, a *non bursty* network overhead, and a *predictable* write access latency.

In order to implement such characteristics, the database is replicated: an (almost) complete replica of the whole database is kept at each NM Element. The broadcast of update operations is performed using a number of circulating tokens, each containing a stack of recently issued updates. The number of tokens circulating in the system is tuned automatically, based on a feedback mechanism that enables each NM Database Proxy to inject (or remove) a token when needed.

The peer-to-peer protocol used for token circulation is made secure using the same certificates that are stored in the database itself: upon receiving a token from a neighbor, the NM Database Proxy authenticates it using the public key of the sender retrieved from the local database. In the exceptional case that peer's certificate is not present in the local database, a copy is downloaded from the neighbor.

The protocol is resilient to network and host failures, since it does not follow a preplanned path (or overlay network): tokens wander randomly in the system. Although mostly based on random decisions, the protocol promises an excellent stability and predictability: this conclusion is justified by simulation results reported in [5]. The load is evenly distributed in time and space, while the update latency remains constant.

The *interface* offered by this module to the upper layer consists of the operations outlined in Table 2, and extends the use of the DB to the storage of rather static characteristics of the Network Monitoring topology (like domain partitioning): the `select` function returns the desired data, while the `update` returns an acknowledgement. They take as parameters an SQL-like query and the id of the Element for which the NM Service issues the request.

Function	Parameter	Description
`select`	SQL select query	The SQL-like query that returns the desired data
	Submitter	The id of the Element which submitted the query
	Return value	A data structure containing selected data
`update`	SQL update query	The SQL-like query that modifies the database
	Submitter	The id of the Element which submitted the query
	Return value	A data structure containing the query id
`check`	Query id	The id of the query
	Return value	A data structure containing the status of the query

Table 2. API of the NM Database Module.

While the `select` function is clearly synchronous, the `update` function is not: the acknowledgement reflects the fact that the request has been successfully queued, not necessarily performed. In order to check the (likely) completion of a requested update, the interface offers the `check` function, which takes as parameter the `update id` returned as an acknowledgement, and returns the current status of the update request, derived from the internal queue. The returned status contains a prediction of the completion time.

3 Related work

The network monitoring management has been addressed by a number authors: solutions are differentiated in the way they cope with scalability and security.

NWS [22] is the ancestor of network monitoring services, and it shares the same building blocks with the architecture introduced in this paper: sensors and a directory for available monitoring functions. However, NWS did not consider at all scalability and security issues. Therefore, despite its importance as a proof of concept, its applicability is limited to small, protected networks.

TopoMon [9] can be regarded as an evolution of NWS, in a direction which is somewhat complementary to the approach followed in our work. In fact, TopoMon extends NWS with tools and support for managing link level topology, a knowledge we explicitly exclude from our interests. Although we understand that this information is relevant (for instance, in view of a reservation service that cannot ignore the existence of shared links when allocating end to end communication resources), we prefer to explore scalability and security issues, which are not addressed by TopoMon, instead of insisting on tools for exploring communication infrastructure.

The **JAMM [20]** sensor management system has been implemented at LBNL for purposes which are close to ours, and is able to configure sensors upon request from applications. An LDAP based directory service keeps records of available sensors, and data from sensors flow to the user applications through specialized gateways. The authors suggest to use encrypted communication in order to ensure security.

The architecture we present in this paper addresses security and scalability aspects in a different way. In JAMM, gateways are used to decouple producers from users, in order to limit the fan out from the sensors. Our model is characterized by a more composite approach to scalability: a support for domain partition is provided, which limits the size of the problem, directory management is addressed with explicit reference to its complexity, passive monitoring is explicitly supported to contain communication footprint, and finally the solution of the fan out problem is delegated to a GIS, without introducing a new solution to a problem that must be necessarily solved elsewhere.

An interesting approach to the problem of retrieving monitoring data is offered by **Gigascope** [8], a stream oriented database for storing captured network data in a central repository for further analysis using an SQL-like query language.

A large scale project that focuses on a scalable, secure monitoring infrastructure is **NIMI** [1]. The architecture introduced in this paper shares several aspects with such large scale prototype: mainly, the strict separation of concerns regarding making measurements, requesting measurements, analyzing results, configuring probes is reflected in the internal structure of our NME. In our architecture, which is at the design stage, we introduce a decentralized global view of the overall network monitoring system, which serves as a support also to service discovery. Such aspect is not covered by NIMI, which bases the local knowledge of each probe on the information received by Configuration Point of Contacts, without introducing any form of coordination between them.

We employ passive monitoring as a technology that fits our scalability requirements: likewise, this approach is a cornerstone of the **CoMo** project [11]. One purpose of this project is to allow users to query network data gathered from multiple administrative domains in a secure and reliable way, without interfering with resource availability. The white paper which is available does not address the organization of a registry of available sensors, which is needed to address a large, domain structured network.

Sprint's passive monitoring system [10] also collects data from different monitoring points into a central repository for analysis. The authors observe that the amount of data collected becomes rapidly unmanageable: in our design, this drawback is resolved with the introduction of a domain oriented overlay network, and by offering an interface for on demand monitoring.

Arlos et al. [6] propose a distributed passive measurement infrastructure that supports various monitoring equipment within the same administrative domain.

An approach that makes use of passive monitoring is often based on packet analysis libraries, which extract the desired pieces of information from the monitored traffic. The most widely used library for this purpose is `libpcap` [16], which provides a portable API for user-level packet capture. The `libpcap` interface supports a filtering mechanism based on the BSD Packet Filter [15], which allows for selective packet capture based on packet header fields. **CoralReef** provides a set of tools and supports functions for capturing and analyzing network traces [13]. **Nprobe** [17] is a monitoring tool for network protocol analysis. It is based on commodity hardware, and speeds up network monitoring tasks using filters implemented in a programmable network interface.

The passive monitoring components of our system are based on **MAPI** [18], which shares some functionality with the above monitoring systems, but at the same time provides a more expressive programming interface with significantly extended functionality and, in many cases, increased monitoring performance. Additionally, the distributed version of MAPI [21] enables distributed network monitoring applications through a flexible interface that allows the manipulation of many remote monitoring sensors from a single application.

The overall approach described in this paper is derived from the **GlueDomains** [4] prototype, which has been successfully deployed and used in the Grid infrastructure of the Italian National Nuclear Physics Institute (INFN). However, the existence of a centralized repository for configuration data, together with an extended use of active monitoring techniques, limits the scalability of the GlueDomains prototype to approximately 50 domains, which is reasonable only for a small-scale grid.

4 Conclusions

In this paper we outline the internal architecture and the interface of a network monitoring service, specifically addressing security and scalability issues. The basic building block is the *Network Monitoring Element*, a specialized Grid component. A Grid contains several instances of this component, which is responsible for monitoring *Network Elements* between *Network Monitoring Domains*. A Network Monitoring Element carries out its monitoring activity using passive monitoring, virtually without network overhead. Its activity is described by a number of *Network Monitoring Sessions*. We exclude manual intervention for its configuration, which should be carried out automatically, either using preconfigured sessions, or preferably according to requests from resource brokers.

References

1. A. Adams, J. Mahdavi, M. Mathis, and V. Paxson. Creating a scalable architecture for internet measurement. In *Proceedings of INET98*, Geneva, July.
2. C. Aiftimiei, S. Andreozzi, G. Cuscela, N. D. Bortoli, G. Donvito, S. Fantinel, E. Fattibene, G. Misurelli, A. Pierro, G. Rubini, and G. Tortone. GridICE: Requirements, architecture and experience of a monitoring tool for grid systems. In *Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2006)*, Mumbai - India, February 2006.
3. S. Andreozzi, A. Ciuffoletti, A. Ghiselli, and C. Vistoli. Monitoring the connectivity of a grid. In *2nd Workshop on Middleware for Grid Computing*, pages 47–51, Toronto, Canada 2004.
4. S. Andreozzi, A. Ciuffoletti, A. Ghiselli, and C. Vistoli. Gluedomains: Organization and accessibility of network monitoring data in a grid. Technical Report TR-05-15, Universit di Pisa, Largo Pontecorvo - Pisa -ITALY, May 2005.
5. S. Andreozzi, D.Antoniades, A.Ciuffoletti, A.Ghiselli, E.P.Markatos, M.Polychronakis, and P.Trimintzios. Issues about the integration of passive and active monitoring for grid networks. In *CoreGRID Integration Workshop 2005*, november 2005.

6. P. Arlos, M. Fiedler, and A. A. Nilsson. A distributed passive measurement infrastructure. In *Proceedings of the 6th International Passive and Active Network Measurement Workshop (PAM'05)*, pages 215–227, 2005.
7. R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, B. Tierney, and R. Wolski. A grid monitoring architecture. Recommendation GWD-I (Rev. 16, jan. 2002), Global Grid Forum, 2000.
8. C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: a stream database for network applications. In *Proceedings of the ACM SIGMOD international conference on Management of data*, 2003.
9. M. den Burger, T. Kielmann, and H. E. Bal. TOPOMON: A monitoring tool for grid network topology. In *International Conference on Computational Science (2)*, pages 558–567, 2002.
10. C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, D. Papagiannaki, and F. Tobagi. Design and Deployment of a Passive Monitoring Infrastructure. In *Proceedings of the Passive and Active Measurement Workshop*, Apr. 2001.
11. G. Iannaccone, C. Diot, D. McAuley, A. Moore, I. Pratt, and L. Rizzo. The CoMo white paper. Technical Report IRC-TR-04-17, Intel Research, 2004.
12. H. Jiang and C. Dovrolis. Passive estimation of tcp round-trip times. *SIGCOMM Comput. Commun. Rev.*, 32(3):75–88, 2002.
13. K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and K. Claffy. The architecture of CoralReef: an Internet traffic monitoring software suite. In *Proceedings of the 2nd International Passive and Active Network Measurement Workshop*, Apr. 2001.
14. S. McCanne and V. Jacobson. The BSD packet filter: A new architecture for user-level packet capture. In *Proceedings of the USENIX Winter Conference*, January 1993.
15. S. McCanne and V. Jacobson. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *Proceedings of the Winter 1993 USENIX Conference*, pages 259–270, January 1993.
16. S. McCanne, C. Leres, and V. Jacobson. libpcap. Lawrence Berkeley Laboratory, Berkeley, CA. (software available from http://www.tcpdump.org/).
17. A. Moore, J. Hall, E. Harris, C. Kreibich, and I. Pratt. Architecture of a network monitor. In *Proceedings of the 4th International Passive and Active Network Measurement Workshop*, April 2003.
18. M. Polychronakis, K. G. Anagnostakis, E. P. Markatos, and A. Øslebø. Design of an application programming interface for IP network monitoring. In *Proceedings of the 9^{th} IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 483–496, April 2004.
19. A. S. Tanenbaum. *Computer Networks*, chapter 8.5. Prentice Hall, 4th edition, 2003.
20. B. Tierney, B. Crowley, D. Gunter, J. Lee, and M. Thompson. A monitoring sensor management system for grid environments. *Cluster Computing*, 4(1):19–28, Mar. 2001.
21. P. Trimintzios, M. Polychronakis, A. Papadogiannakis, M. Foukarakis, E. P. Markatos, and A. Øslebø. DiMAPI: An application programming interface for distributed network monitoring. In *Proceedings of the 10^{th} IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2006.
22. R. Wolski. Dinamically forecasting network performance using the network weather service. Technical Report TR-CS96-494, University of California at San Diego, January 1998.