

A Scalable Multi-Agent Architecture for Remote Failure Detection in Web-Sites

Décio Sousa¹, Nuno Rodrigues¹, Luís Silva¹, Artur Andrzejak²

¹*CISUC - Centre for Informatics and Systems of the University of Coimbra
Polo II - Pinhal de Marrocos
3030-290 Coimbra, Portugal
luis@dei.uc.pt*

²*Zuse-Institute Berlin
Takustr. 7, 14195 Berlin, Germany
andrzejak@zib.de*



CoreGRID Technical Report
Number TR-0072

August 30, 2007

Institute on Architectural issues: scalability,
dependability, adaptability (SA)

CoreGRID - Network of Excellence
URL: <http://www.coregrid.net>

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-00426

A Scalable Multi-Agent Architecture for Remote Failure Detection in Web-Sites

Décio Sousa¹, Nuno Rodrigues¹, Luís Silva¹, Artur Andrzejak²

¹*CISUC - Centre for Informatics and Systems of the University of Coimbra
Polo II - Pinhal de Marrocos
3030-290 Coimbra, Portugal
luis@dei.uc.pt*

²*Zuse-Institute Berlin
Takustr. 7, 14195 Berlin, Germany
andrzejak@zib.de*

*CoreGRID TR-0072**
August 30, 2007

Abstract

Despite the huge efforts in developing failure detection tools for web-sites, there are still some particular failures that still become user-visible. These are namely application-level failures that escape detectors implemented by common internal system-level monitors. To successfully detect these failures, several mechanisms have been developed, namely remote failure detectors. In this paper we present GOA-Net: A scalable multi-agent architecture for remote failure detection in web-sites. Its agents implement detection mechanisms that are capable of perceiving, not only web-site availability, but also validate its content. It is capable of performance degradation forecasting which make it unique among similar tools. In this paper we describe all the detection mechanisms, distributed algorithms and design options that constitute this network.

* This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

1 Introduction

Industry have been developing a wide variety of monitoring tools to detect failures and anomalies in IT systems. The most compelling examples include HP OpenView [1], IBM Tivoli [2], Altaworks Panorama [3], Zabbix [4] and Nagios [5]. These tools are excellent to detect failures that occur in the operating system or middleware but some subtle application-level failures escape these system-level monitoring and become user-visible [10, 11]. Typically these failures occur because of operator errors or software bugs (typically non-deterministic) that escaped off-line testing. These failures (Software and Human Operator) still account for 80% of total failures present in Internet Applications [13].

Several mechanisms have been developed to try to successfully detect these failures that still manage to become user-visible. These detectors are log analyzers [24, 25], user behavior monitors [26], application component interaction monitors [17] and remote monitoring frameworks. Despite some of these tools show great detection capabilities of application-level failures (namely [17]) they are only academic proposals with some difficulties of adapting to real world scenario. Nevertheless, remote monitors are starting to be widely used by industry and several mature commercial solutions exist. These monitors are based in monitoring stations disperse over the Internet that simulate real user interactions with a web-site collecting end-users perspectives. This way they can perceive errors in the exact same way a human user would.

In this paper we present the design of an enhanced remote monitoring network that enhances the existing solutions by taking remote monitoring to a next level. Until now, every tool of this kind was based in purely reactive monitoring stations that simply warned system administrators when an erroneous condition was met with their web-sites. The network we are presenting also comprehends this type of reactive behavior but is also pro-active in detecting failures, not waiting for them to actually happen. We believe this to be a major step in further reducing downtime in Internet Sites. This is done with forecasting techniques that can predict performance degradation in a web-site and understand an imminent failure. Furthermore, this is a highly scalable network that is prepared to be easily expanded to monitor other Internet Services. Also, it is the only remote web-site monitoring tool that is capable of analyzing Internet video streaming QoS.

2 Motivation

The motivation for developing a multi-agent network for remote failure detection in web-sites came from the conclusions taken from an experimental study that we previously conducted [27]. In that study we identified the existing mechanism for failure detection in web-sites and drew a comparison between them. We compared a widely used internal system level monitoring system, a log analyzer tool, an application component analyzer inspired by an academic project [17] and an external monitoring agent.

From the results taken in that study we were able to understand the potentials and limitations of each one of these mechanisms. We understood that the internal system-level monitor performed excellent when detecting system-level failures, but lacked in detecting application-level. Log-analyzers showed evidences of being able to detect most system-level failures (although with higher latency than the system-level monitor) and also some application level. Component interaction analyses showed surprising results being capable of detecting almost every kind of failure, both system and application level. But what really caught our eye was the fact that external monitors being able to detect every type of failure and in some particular cases, was even the first tool to detect.

It is certain that component interaction monitors can be of great help in complementing system-level monitors, but these are still in an immature state. They are difficult to deploy and require applications to be instrumented or adapted in order to be possible to collect data needed by these failure detectors. Although there are some attempts to generalize this detection mechanisms (embedding them into application containers [12]) they are still not used by industry and are far from being usable in real-world scenarios. For our study we had to build our own implementation of these detection mechanisms, as for the inexistence of any available implementations.

Because external monitoring showed evidences of being capable of detecting every failure, namely application level, and because these mechanisms exist, are widely used (See section 3), and easily adapt to any web-site, we feel that they can be an excellent alternative to complement internal system-level monitors. Furthermore, we feel that they have a huge monitoring potential and we believe can be used to monitor other internet services and even provide other services not directly related with failure detection.

With this base of evidences, we decided to design GOA-Net (Global Observation Agent Network), a network that is capable of remotely monitoring web-sites but also provide a solid ground for expansion to other services that we feel a well designed network could provide. Malaware detection, security evaluations, dependability and scalability benchmarks of application are just a few examples of services such network can provide.

3 Related Work

As said in the previous section, several different remote monitoring platforms, networks and frameworks exist. Site24x7 [17], Gomez Performance Network [18], Watchmouse [19], Alert Site [20], eXternalTest [21], webAlarm [22], MrAlert [23], AlertBot [18], WatchDog [19], ServerCheck [20] are just some compelling examples of the wide variety of these tools. Most of them are much alike, implementing the same detection mechanisms and architectures. In this section we present 4 that we consider to be representative of the variety that can be found.

Site24x7 [7] from AdventNet is a monitoring service that comprehends a simple architecture based on a single monitoring station providing monitoring only from 1 location. This service has support for simple transactions and content checking, alerts users via e-mail and generates reports with charts on availability and response time of the monitored web-site.

Watchmouse [8] is a commercial online service that can monitor different Internet services for correct functioning and availability. Based on 20 monitoring stations over 3 continents dispersed, this tool is capable of monitoring a large set of Internet protocols (PING, HTTP, HTTPS, SMTP, POP3, IMAP, FTP, TELNET, DNSa, DNSns). Is a complete solution that also provides web-site content checking and validation. Its only drawback is the lack of support for web-site transactions making it only capable of monitoring simple web-page of a configured web-site. Alerts are also generated via e-mail and reports are also available about response time and availability.

Another commercial remote monitoring solution is provided by AlertSite Monitoring Suite [9]. This tool is focused on monitoring web-sites and e-mail servers. It is able to collect very detailed performance metrics for each monitored web-page (DNS resolution time, TCP Connect time, Time to first byte, time for whole page download, content download time, simulation of cached and non-cached users, etc) and capable of generating SLA proof reports on web-sites service level. This platform does not make any content validation.

Gomez Performance Network (GPN) [6] is a network of agents placed all over the world that perform performance tracking and remote web-site monitoring. The GPN agents are deployed in ISP federated servers, client intranets and 10 000+ real-users that provide First-, Middle- and Last-mile perspectives of monitored web-sites. This is, to the best of our knowledge, the only platform that covers all three perspectives. Recently, Gomez has joined a partnership with HP Labs, having integrated GPN with the popular HP OpenView platform, being now the first platform that is capable of external and internal monitoring of web-sites.

GOA-Net wants to implement all these features allying a scalable architecture with good detectors and forecasting techniques that will act according to the data collected. Our network wants to bring new aspects and techniques to help increase the coverage of failure detectors, reduce the latency to detect and enhance with the ability of forecasting potential problems that would lead to a failure.

4 Objectives

In a short list, the objectives for GOA-Net are:

1. Implement remote failure detectors capable of perceiving a web-sites' unavailability
2. Implement web-site content validation mechanisms, both static and dynamic
3. Support not only single page monitoring but also entire web transactions
4. Be capable of correlating monitoring data from first-, middle- and last-miles
5. Implement a highly scalable architecture that is capable of being massively deployed increasing the coverage of failure detectors
6. Forecast performance degradation predicting potential problems that lead to failures
7. Be able to monitor Internet video QoS

We can see (from the previous section) that some of these objectives have already been achieved by the available related tools. 1 and 2 are available in almost every remote monitoring tool, while 3 is implemented only by a smaller

set. 4 is only achieved by GPN and to the best of our knowledge, there is no remote monitoring tool that comprehends any of the last 3 objectives making GOA-Net the first to provide such improvements.

We believe that these three goals represent a major step in improving the capabilities of remote failure detectors. 5 will permit to cover much more sites and potentially extend the concept to other more demanding services. To fulfil this objective we designed a network architecture that can be easily integrated with mass internet applications like instant messaging which permits a huge scalability potential.

6 makes GOA-Net the first tool that is capable of forecasting problems before they actually become failures which marks a significant difference. While every other tool we analysed are purely reactive, GOA-Net is the first to become proactive in the way that it does not simply “wait” for a failure to happen to issue a warning. It is the first that tries to understand the presence of faults that will lead to failures before they actually happen and therefore warn system administrators that their site might fail in a certain interval. Our architecture is built with a special node that provides the required ground to implement such forecasting techniques.

Finally, no remote monitoring tool is capable of perceiving QoS shortages in one of the most demanding Internet applications: video streaming. The agents in GOA-Net have embedded video streaming clients with support for open multicast standards like RTP [15].

In the next section we will deeply describe the architecture of GOA-Net, all its mechanisms and design decisions.

5 Architecture

GOA-Net architecture is composed by three base components: The Master Agent (MA), several Global Observation Agents (GOA) and Forecast Aggregation Nodes (FAN). Besides these three, there is also a web front-end from where system administrators can interact with the network by configuring their monitored web-sites. Figure 1 represents an overview of GOA-Nets' architecture.

GOA is an automaton that acts similarly to a web-crawler, but instead of permanently sweeping the Internet for content indexation, these agents execute, at a polling frequency, some pre-defined transactions in monitored web-sites, as a user would typically do. An important aspect is the fact that when replaying a transaction GOAs always make use of the “no-cache” flag so that web-server caching does not influence the perception it has of the web-site. GOAs are able to perform almost every kind of user interaction in order to simulate real user behaviour. Forms can be filled, products browsed, searches conducted, products added to a shopping cart and business transactions complete. These agents implement error detection mechanisms that range from pure performance metric collections to higher-level content validation. With these mechanisms they are able to understand the correct functioning of a web-site, not only based in pure QoS and performance metrics but also understanding if the content presented by the site is the one it should really be presented.

MA is a central point of coordination of all the GOAs. It has the mission of keeping every configuration of every monitored web-site and manages the assignments of GOAs to each web-site.

In order to start monitoring a web-site, system administrators are required to configure a web-transaction. To do this they are required to download an automatic transaction recorder from the web front-end that is then installed and integrated with their own web-browser enabling it to record the visited pages. Administrators just have to navigate through their web-site, visiting the pages that they want to be included in the transaction. The recorder will record each interaction with the web-site and upload them to the web front-end.

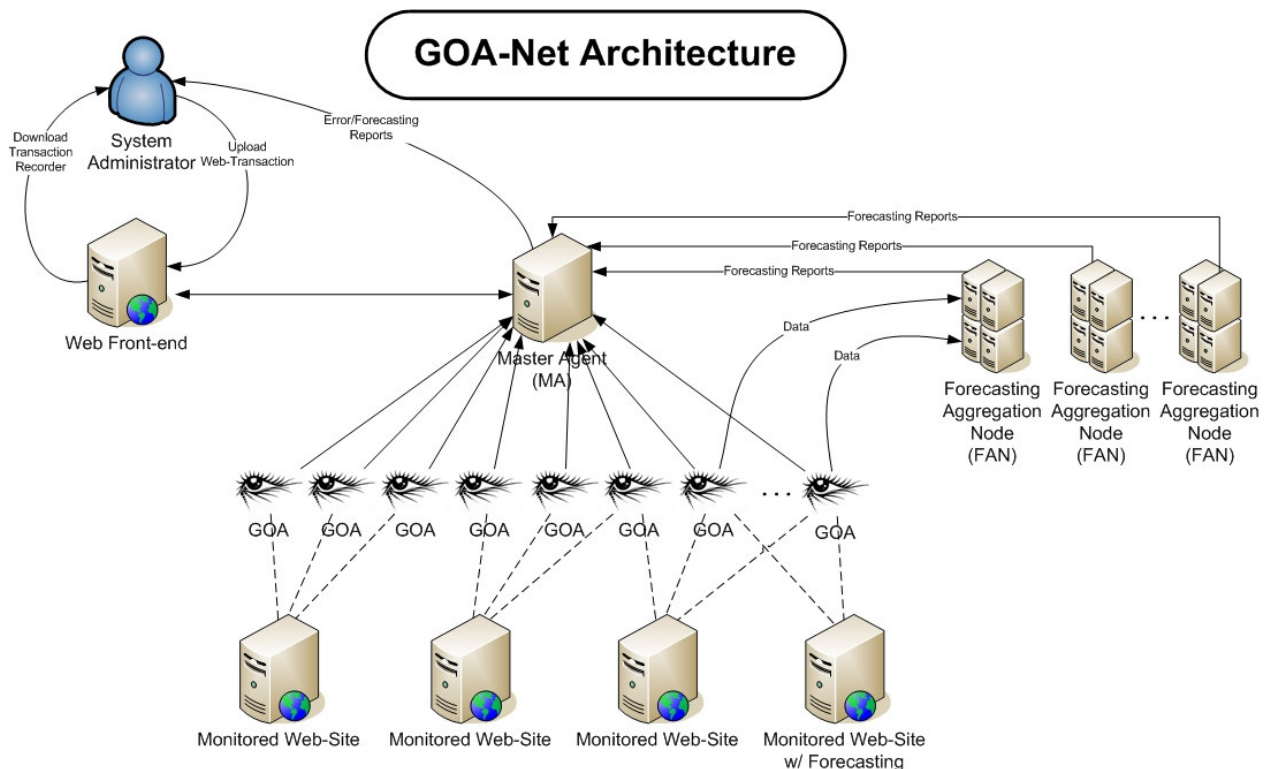


Figure 1 - GOA-Nets' Architecture Overview

Transactions are composed by a set of visited web-pages (≥ 1) and the interactions made with each one of them. Besides creating the transaction itself, administrators are also required to configure a polling frequency, which will be the frequency that the transaction will be replayed by GOAs.

Furthermore, for each page contained in the transaction, administrators are required to configure a set of parameters:

- **Response time threshold** is the maximum response time allowed for that page to fully load. Optionally, this threshold can be separated in DNS resolution time, TCP connect and page download.
- **Matching Expressions** are keywords or expressions that the web-page must or must not contain in order to be considered valid.
- **Dynamic content percentage** is the percentage of the web-page that is expected to vary. In technical terms, is the allowed variation of the page size.

These three parameters are vital because they will be used by GOAs to infer an incorrect behaviour of the web-site. Every transaction configured in the web front-end is then stored in the MA that will keep every configuration of every transaction. For each transaction, the MA will then assign a number of GOAs to monitor that web-site by replaying the recorded transaction and checking its correct functioning. When GOAs find an anomaly with the web-site they issue a notification to the MA that will then issue an error report to the system administrator.

GOAs' anomaly detectors are explained in the next sub-section. Error reports and the number of GOAs assigned to each transaction are issues we leave for discussion in later sections of this paper.

5.1 Detection Mechanisms

GOAs collect several metrics when accessing each page of a transaction. We can distinguish these metrics in two distinct groups: (1) Performance metrics and (2) Content Checking. For the first group, the agents collect data like DNS resolution time, TCP connect time, whole page download time and content download time. Collecting these performance metrics and comparing them with the defined thresholds makes the GOA able to infer an incorrect behaviour from the web-site. Naturally, if the agent fails to resolve the DNS name, fails to open a TCP socket to the server or receives a HTTP error (4xx or 5xx HTTP codes), the agent will immediately infer an incorrect behaviour.

These performance metrics make it possible not only to detect failures but also to perceive their origin (DNS, TCP or HTTP) a bit better.

Although these performance metrics are often good indicators about the behaviour of the web-sites, there are some types of failures (namely application level failures and operator errors) that are visible at the content level. For example, if a database connection is broken between an application server and a DB server, the web-site may still respond properly (in terms of HTTP) but the content presented will appear corrupted. In order to detect these kinds of failures, GOA agents perform some content checking and validation mechanisms. They are able to verify both static and dynamic content in a web-page. By checking the presence of administrator defined expressions in the retrieved web-page, they are able to understand if the content being presented is the one the web-page should really be presenting. This is why the definition of such expressions by system administrators is of vital importance for the correct identification of content related problems in web-pages. Furthermore, by analysing the retrieved web-page size every time the page is accessed, GOAs can check if the size variation is within the system administrator specified percentage and therefore understand if the web-page is/is not showing the correct content.

Besides these detectors, GOA agents have embedded video streaming clients for the most widely used video streaming protocols. This is basically one more content validation mechanism. If a defined transaction includes a web-page that comprehends a video object, GOA will try to stream it analysing QoS metrics like jitter, buffer levels and packet rate. If GOAs notice any service shortage they are prepared to issue a notification to the MA. Furthermore, they include an implementation of RTCP streaming client to be able to monitor multicast video transmission that make use of this protocol.

5.2 Error location

Although in the previous section we have seen that GOAs by themselves incorporate some mechanisms that help identify the source of an identified error (DNS, TCP and HTTP error detection enables the inferring of the error location, DNS server, network or server respectively), more accuracy can be achieved by correlating the information of different GOAs.

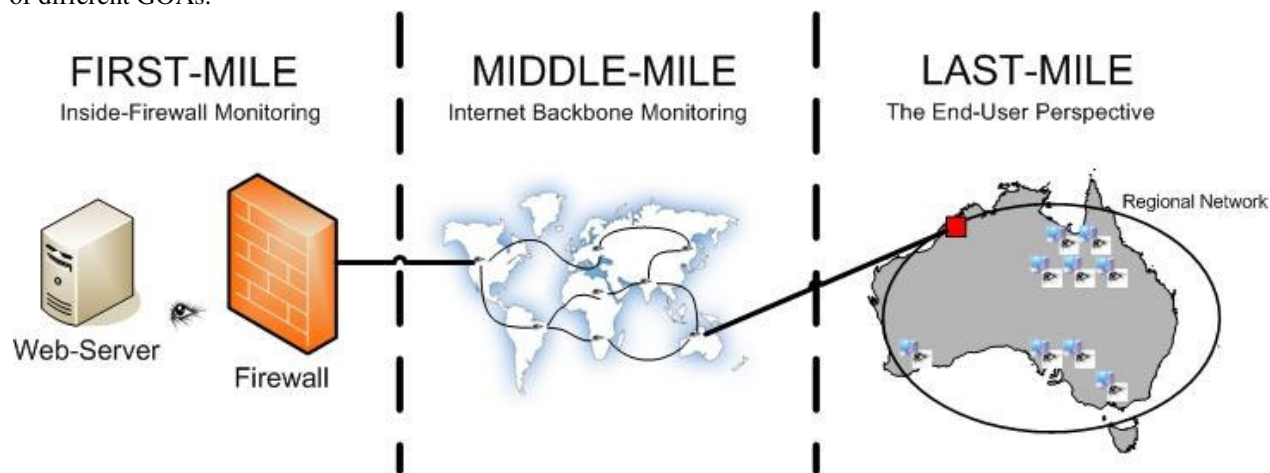


Figure 2 - First-, Middle- and Last-Mile perspectives. “Eyes” represent GOAs.

As GOAs can be massively distributed over the Internet, this multi-agent grid is capable of providing and analysing, the different perspectives of the web-site as perceived in the First and Last miles of the network path. Figure 2 represents these perspectives.

First mile coverage, or inside the firewall coverage, provides the perspective of the web-site behaviour without any network interference. This perspective represents the pure performance of the web-application without the impact of the network. We can accomplish the collection of these metrics by placing an agent running on a machine inside the servers’ local network.

The Middle-Mile perspective is collected from the Internet Backbone from GOAs residing inside the same ISP as the monitored server. The information collected here can be used to diagnose and identify problems that are not visible in the first-mile such as DNS resolution problems, network congestion issues and internal ISP connectivity problems.

Last-Mile gives us the perception of how web-site users are experiencing their interactions with the web-site. Also, this perspective is useful in diagnosing problems that are visible to the end-users, such as performance or reliability problems associated with over-weighted web-pages.

In order for GOA-Net to consider network congestion, server connectivity or DNS resolution problems as the origin of a failure, only agents placed in middle and last mile should report the incident since the first mile is not affected by any of these problems. In this case, differentiation between the three possible origins is made by the type of error perceived by each GOA. Remember from the previous section, that GOAs are able to distinguish between DNS, TCP and HTTP errors. If they report DNS problems, then the cause must reside within the DNS server. If TCP connection reports host unreachable then it is most probably due to the server lacking Internet connectivity. Although, if GOAs timeout connecting to the server, it indicates that there is probably a network congestion between them and the server. Finally, we can infer the cause of the problem to reside inside the server itself if all the agents in the three perspectives report the incident, as this is the only cause of failure that will result in visibility in all three perspectives.

5.3 Scalability

To be a highly scalable network, GOA-Net needs to support the massive distribution of GOAs over the Internet. This can only be achieved if GOAs can be hosted by different hosts available in the Internet like companies, real users and federated ISP servers. This heterogeneity of deployment scenarios raises connectivity problems, since most of the agents will probably be hosted behind firewalls making it very difficult for the MA to reach each agent.

To solve these connectivity issues we assume the following paradigm: Every communication between GOAs and the MA is always initiated by GOAs and never otherwise.

Another important aspect that must be considered is the fault tolerance GOA-Net must have for GOA failures. If GOAs are hosted by normal users, it is obvious that they will not be always available. They will naturally fail because of the most varied reasons (hosts shutdown, hosts disconnect from Internet, hosts shutdown the application, etc.). Considering this aspect and the communication paradigm that we assumed, GOAs are required to periodically send heartbeats to the MA so that it can keep track of the available GOAs and reorganize assignments accordingly.

For GOA-Net to assure a correct monitoring of a web-site it requires to have at least one GOA in each perspective monitoring each web-site. If we consider that GOAs are prone to fail we need to assign redundant GOAs to the same web-site, so that in case one GOA is shutdown, other still captures the desired perspective. The number N of redundant agents is computed based on the empiric observation of the probability of failure that GOAs show. The higher the fail rate, the most redundant agents are assigned.

GOAs were designed to be lightweight autonomous agents that only need to communicate with the master in four different situations.

- At start-up, to be assigned web-sites to monitor.
- Whenever an error condition is met with one of their monitored web-sites.
- Periodically, to send heartbeats to let the MA know they are still alive.
- To retrieve any required updates from master.

After the first interaction with MA, GOAs keep the configurations of each assigned web-site, monitoring accordingly.

MA needs to update the internal GOA assigned web-sites (because of configuration changes, new sites added or removed for monitoring) but cannot directly contact GOAs because our strict communication policy. Instead, it has to wait for a heartbeat and take advantage of the open connection to let the GOA know he has to update his internal configurations. Upon the reception of such indication, GOAs initiate a connection to the master to retrieve the necessary updates.

5.4 Security

Because communication between GOAs and MA is made through public internet, we are forced to consider the existence of sabotage. We cannot allow that third party applications are able to disguise themselves as a GOA agent and send erroneous data to the MA. The results of such actions could result in a complete inoperability of the network.

To avoid this kind of risk, we must protect communication between the GOAs and both the MA and FAN, so that the data source can be verified and the content not tampered. To do this we must have an RSA key pair to identify our MA and FANs. The private keys must be stored in the MA and FAN servers and the public keys as well as the key pair identifications must be embedded into the GOAs code.

When a GOA needs to send some data to the servers, it generates a random session key and uses it with an AES encryption algorithm to encrypt all the data. This session key will then be encrypted with the MA or FAN public key and both the encrypted data and the encrypted session key will be sent to the MA or FAN servers respectively. When it arrives at the server, the encrypted session key will be decrypted and it will be used to decrypt the monitoring data.

With this mechanism, we assure that only the MA or FAN servers will have the ability to decrypt the message and analyze the data and only GOAs are able to generate valid messages to both MA and FANs.

5.5 Forecasting

Performance degradation forecasting requires a lot of processing power and therefore special nodes are needed. Forecast Aggregation Nodes (FAN) are federated servers located in data centres that exist only to aggregate and process data gathered by GOAs. Each of these FANs can be responsible for forecasting performance degradations in a determined number of web-sites. When a FAN reaches its limit of handled web-sites, another must be provided. In sum, these are nodes that will grow on-demand reflecting the present necessities of forecasting GOA-Net experiences. This decision is made upon the assumption that this type of service is not required by the great majority of monitored web-sites, only by those with most demanding availability.

Figure 1 shows a monitored web-site that uses this feature. GOAs assigned to monitor these web-sites, not only send error reports to the master, but also send, in real time, every performance data gathered to a FAN responsible for forecasting performance degradations in the monitored web-site. In case of predicting that a determined web-site will be prone to suffer from performance degradation, or even from a service outage, the FAN will issue a forecasting report to the MA which will forward it to the system administrator. GOAs assigned these forecasting-enable web-sites, are told by the MA to which FAN they are suppose to send their data.

FANs implement forecasting algorithms that are based in time series analysis and floating averages [28, 29].

5.6 Reports

GOA-Net issues two different kinds of reports: Error reports and Forecasting Reports. These last, as the name suggests, are reports generated by FANs when they predict some performance degradation or service outages will occur in a near future. These are essentially warnings to system operators so that they can prevent the happening of such outage or performance degradation. They incorporate the expected failure and the expected time in where this failure will appear together with a confidence interval.

Error reports are generated by the MA upon reception of a notification from a GOA indicating an erroneous condition with a monitored web-site. The generation of such reports is made by the MA which correlates information about all the GOAs monitoring that web-site. To do this, MA keeps an internal table for each monitored web-site that keeps the notifications received from GOAs. Upon reception of such notification the corresponding table is updated reflecting the information received from the GOA. In order to correctly timestamp the reports, MA keeps the mean RTT value for each GOA (calculated on every heartbeat GOAs send). The notification timestamp is then computed as the MA timestamp at the moment notification is received minus the mean RTT (Round Trip Time) of the GOA that sent the notification.

Because GOAs are not expected to issue notifications all at the same time (because they only run their transactions from time to time) it is possible that upon the reception of a notification, other GOAs monitoring the same web-site

may still not notice the anomaly reported. Therefore, the MA has to wait before he can have the perspective of all the GOAs monitoring the web-site. The waiting interval is computed as the defined polling interval for the web-site plus the maximum mean RTT of the agents monitoring that web-site. GOAs that do not issue a notification within this interval are considered not to be “catching” the anomaly.

In order not to delay failure reports to system administrators, issued reports can have different accuracies depending on the number of GOAs on the different perspectives that noticed the anomaly in the moment the report is generated. Three levels of accuracy exist: Low, Medium and High.

Upon the reception of a notification from a GOA, the MA always issues a report. The notification of a problem P implicates that N GOAs notify that same problem. If at least one GOA of each level that is supposed to notice the problem has already issued a notification, then the error report is considered accurate (High accuracy level). If only some of the GOAs have issue the notification, but timestamps indicate that others still have time to notify, the report is considered Medium accurate. In case that none of the expected GOAs notified the MA about the problem but again the waiting period is not yet over, the report is considered to be of Low accuracy.

When the MAs’ waiting period is over and no High accuracy report has been issued, the first notification is considered a false one and a new report is issued disconfirming any previous reports that had been generated.

With these mechanisms GOA-Net is capable of quickly issuing error reports although not compromising the filtering of any possible false positive alerts.

6. Comparison with Related Tools

In Table 1 is presented an exhaustive comparison between the features in GOA-Net and 4 widely used commercial remote monitoring solutions. From this Table we can see that GOA-Net comprehends mostly every feature present in other tools and enhances in three aspects: Is the only one supporting Video QoS monitoring, is highly scalable being designed to comprehend millions of GOAs and is the only tool that is pro-active in forecasting failures before they actually happen.

Nevertheless, other tools are more complete in other aspects such as the type or services supported. While GOA-Net is only capable of monitoring HTTP/S, TCP and DNS, other solutions such as Watchmouse already implement monitoring techniques for a wider variety of Internet Services. However, GOA-Net was designed to allow a simple integration of new services. By simply implementing in GOAs support for other services the whole network becomes capable of monitoring such services.

GPN also comprehends a feature that is not supported by GOA-Net, the integration with an internal monitoring solution. We believe this is a feature that can only be accomplished after GOA-Net is fully implemented and in a mature state. Though, as GOA-Net has a central coordinator agent, the integration can be simplified to the level of integrating the Master Agent of GOA-Net with an internal monitoring solution.

	Site24x7	WatchMouse	AlertSite	Gomez	GOA-Net
Services Supported	HTTP, HTTPS	PING, HTTP; HTTPS, FTP, TELNET, SMTP, POP3, IMAP, DNSa, DNSns	HTTP, HTTPS, TCP, SMTP, POP3, DNS	HTTP, HTTPS, TCP, DNS,	HTTP, HTTPS, TCP, DNS
Number of Monitoring Stations	1	24 geographically distributed across 3 different continents	25 spread worldwide	60 federated and 10 000+ peer clients spread worldwide	Massive Internet distribution can potentially reach millions of peers
Performance Degradation Forecasting	Not Supported	Not Supported	Not Supported	Not Supported	Supported
Internal Monitoring	Not Supported	Not Supported	Not Supported	Complete supported w/ Integration with HP Openview.	Easily Integration with any Internal Monitoring Platform
Performance Metrics	Response Time	Response Time	DNS res. Time, TCP connect, First-Byte, Whole Page, Content Download, First time user, returning customer, TCP traceroutes	DNS res. Time, TCP connect, First-Byte, Whole Page, Content Download, TCP traceroutes	DNS res Time, TCP connect, Whole Page, Vieo QoS
Perspective of Metrics	ISP Backbone (Middle mile)	ISP Backbone (middle mile) and end-user perspective (Last mile)	End-user only (Last mile)	Inside Firewall (First Mile), ISP Backbones (Middle mile), end-user perspective (Last-Mile)	Inside Firewall (First Mile), ISP Backbones (Middle mile), end-user perspective (Last-Mile)
Web-Site Content Checking	Static and Dynamic	Static only	-	-	Static and Dynamic
Support for online transactions	Supported	Not Supported	Supported	Supported	Supported
Distinguish between network and application level problems	Not Supported	Supported	Supported	Supported	Supported

Table 1 - Features comparison table between GOA-Net and other similar tools

6 Conclusions and Future Work

In this paper we presented the design of a highly scalable remote monitoring network. We have enhanced the existing tools in three main aspects:

- High scalability that provides a base ground for further innovative services and widens the coverage of remote monitoring in Internet Sites
- Pro-activity in failure forecasting. For the first time a remote monitoring tool is not a merely reactive warning system that issues warnings when a failure occurs.
- Internet Video QoS monitoring. We integrated in our agents video streaming capabilities so that they can perceive short QoS in Internet Video transmissions

We are now starting a first prototype implementation of this network and will in a near future conduct experimental comparative tests that will help us validate our expectations.

References

- [1] - Hewlett Packard Corporation, HP OpenView Software. <http://www.openview.hp.com>
- [2] - IBM Corporation, IBM Tivoli Software. <http://www.ibm.com/software/tivoli>.
- [3] - Altaworks Panorama. <http://www.altaworks.com/solutionsPanorama.htm>
- [4] - Zabbix. <http://www.zabbix.org/>, January 2007
- [5] - Nagios. <http://nagios.org/>, January 2007
- [6] - Gomez Performance Network. <http://www.gomez.com/products/index.html>
- [7] - Site24x7. <http://www.site24x7.com>
- [8] - Watchmouse. <http://www.watchmouse.com>
- [9] - AlertSite. <http://www.alertsite.com>
- [10] - E. Kiciman and A. Fox, "Detecting Application-Level Failures in Component-based Internet Services", IEEE Transactions on Neural Networks, Vol. 16, Issue 5, 2005
- [11] - A study about online transactions, prepared for TeaLeaf Technology Inc, by Harris Interactive, October 2005
- [12] - George Candea, Emre Kiciman, Steve Zhang, Pedram Keyani, Armando Fox, "JAGR: An Autonomous Self-Recovering Application Server", in Proceedings of the 5th International Workshop on Active Middleware Services, Seattle, WA, June 2003
- [13] - Soila Pertet and Priya Narasimhan, "Causes of Failure in Web Applications", Parallel Data Laboratory, Carnegie Mellon University, CMU-PDL-05-109
- [14] - Berkeley Open Infrastructure for Network Computing - <http://boinc.berkeley.edu/>
- [15] - H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications" RFC3550. Internet Engineering Task Force, July 2003.
- [16] - Randa El-Marakby, David Hutchison, "Scalability Improvement of the Real-Time Control Protocol (RTCP) Leading to Management Facilities in the Internet", in the Proc. Of the Third IEEE Symposium on Computers & Communications (ISCC), 1998
- [17] - E. Kiciman and A. Fox, "Detecting Application-Level Failures in Component-based Internet Services", IEEE Transactions on Neural Networks, Vol. 16, Issue 5, 2005
- [18] - AlertBot - <http://www.alertbot.com/>
- [19] - WatchDog - <http://www.mycomputer.com/>
- [20] - ServersCheck - <http://www.serverscheck.com>
- [21] - eXternalTest - <http://www.externaltest.com/>
- [22] - WebAlarm - <http://www.moniforce.com/>
- [23] - MrAlert - <http://www.mralert.com/>
- [24] - Analog - <http://www.analog.cx>
- [25] - AwStats - <http://awstats.sourceforge.net>
- [26] - Peter Bodík, Greg Friedman, Lukas Biewald, Helen Levine, George Candea, Kayur Patel, Gilman Tolle, Jon Hui, Armando Fox, Michael I. Jordan and David Paterson, "Combining Visualization and Statistical Analysis to Improve Operator Confidence and Efficiency for Failure Detection and Localization", International Conference on Autonomic Computing (ICAC '05) 2005
- [27] - Nuno Rodrigues, Décio Sousa and Luis Silva, "An Evaluation of web-site failure detectors", publication pending.
- [28] - Artur Andrzejak, Patricio Domingues, Luis Silva, "Classifier-Based Capacity Prediction for Desktop Grids", CoreGRID TR-0026, January 2006
- [29] - A. Andrzejak, M. Ceyran, and U. Hermann, "OpenSeries - a Time Series Analysis Library," (unpublished) 2005