Automatic deployment of the Network Weather Service using the Effective Network View

Arnaud Legrand and Martin Quinson
Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon
{arnaud.legrand,martin.quinson}@ens-lyon.fr

Abstract

*The monitoring infrastructure constitutes a key component of any Grid scheduler. The Network Weather Service (*NWS*) is the most commonly used tool to fulfill this need. Unfortunately, users have to deploy the* NWS *manually, which can be very tedious and error-prone. This paper characterizes the* NWS *deployment requirements and introduces a method based on the Effective Network View (*ENV*) network mapper to automatically perform this task. We also present the resulting deployment on our lab's LAN.*

1. Introduction

Grids are a type of parallel and distributed systems that results from the sharing and aggregation of geographically distributed resources between several organizations [7]. Unlike classical parallel machines, Grids present dynamic, heterogeneous and even non-dedicated capacities. Gathering accurate, up to date and relevant informations about them is then a very challenging issue, which has to be addressed before developing schedulers.

Nowadays, the *Network Weather Service* (NWS) constitutes a *de facto* standard in the Grid community as it is used by major *Grid middlewares* like Globus [8] or *Problem Solving Environments* (PSEs) like DIET [2], NETSOLVE [3] or NINF [12] to gather informations about the current state of the platform, as well as about its future evolutions.

Unfortunately, there is no automatic way to deploy the NWS yet, and this deployment has to be done manually. This task is very tedious on such a platform and its proper achievement requires accurate knowledges both about the target network on which the tools are to be deployed and about the NWS internals.

This deployment process can be decomposed in two phases: First, we have to decide what kind of organization the NWS processes should follow in a *deployment planning* phase. Then, this deployment should actually be applied on the platform. The first phase can in turn be decomposed in two steps since we first have to gather the target network topology before computing a deployment plan.

This paper presents a simple method using the *Effective Network View* (ENV) network mapper [16] to collect the needed information and automatically compute a deployment plan which can then be applied using standard tools. The rest of this article is organized as follows: Section 2 presents the NWS, focusing on how network measurements are conducted and characterizing the deployment requirements. Then, we present a rapid network mapping state of the art in Section 3 and detail the ENV tool and its methodology in Section 4. Lastly, we discuss an algorithm to deploy NWS from ENV results in Section 5 and present some results obtained on the LAN of our laboratory.

2. Deploying the Network Weather Service

The NWS (Network Weather Service) [18] is leaded by Prof. Wolski at the University of California, Santa Barbara. This distributed system of sensors and statistical forecasters enables to centralize the current state of the platform.

It enables to monitor the latency and throughput of any TCP/IP link, the CPU load, the available free memory or the free disk space on any host. Concerning CPU load, NWS not only reports the current load, but also the time-slice a new process would get on startup. Concerning link capacities, NWS reports the end-to-end bandwidth, latency and connection time. Given a set of n computers, there is $n \times (n-1)$ paths to test since the network is not symmetric in the general case [14].

In addition to the current state of the platform, the NWS also provides predictions about its future evolutions by applying statistical treatments on the past measurements. Regular measurements in steady state are thus mandatory, even in absence of client requests.

To ensure correct measurements, the network experiments must not interfere with each other. If two measurements were conducted on a given network link at the same time, both of them could be influenced by the bandwidth consumption of the other one, and may therefore report an availability of about the half of the real value.

Wolski et al. [17] handles this problem by introducing the concept of *measurement clique*. A measurement clique is a computer sets in which network measurements are done in a mutually exclusive manner, thanks to a token-ring based algorithm: only the host having the token at a given time is granted to launch network measurements on the links involved in that clique. Unfortunately, these measurement cliques have to be defined manually.

Moreover, token-ring algorithms are known to be not very scalable, and the frequency of the measurements obviously decreases when the number of hosts in a given clique increases. Cliques must then be split in sub-cliques to ensure a sufficient network measurement frequency and increase the responsiveness of the system. These splits have to be done wisely to ensure that the tests in a given clique will never collide on any link with tests from any other cliques.

Of course, clients applications are potentially interested in any end-to-end connexion even if there is no direct measurement between two given hosts because they are in different cliques. The system should then be able to combine the conducted experiments results to deduce the missing values.

For example, given three machines A, B and C, if the machine B is the gateway connecting A and C, it is sufficient to conduct only the experiments on (AB) and on (BC). Latency between A and C can then be roughly estimated by adding the latencies measured on AB and on BC. The minimum of the bandwidths on AB and BC can be used to estimate the one on AC. These values may be less accurate than real tests, but are still interesting when no direct test result is available.

To sum up, the NWS deployment has to satisfy four constraints:

Do not let experiments interfere. All hosts connected by a given physical network must be in the same clique to ensure that the tests conducted on that link are done in a mutually exclusive manner.

Scalability concerns. All cliques should be as small as possible so that measurement frequency is sufficient to ensure a satisfactory reactivity level to the system.

Completeness. If no direct measurement is conducted between two hosts, the system must be able to aggregate the conducted experiments to estimate the network characteristics of their interconnection.

Reduce intrusiveness. In order to reduce the system intrusiveness to its minimum, only the needed tests should be conducted. For example, since the bandwidth is evenly shared by all hosts connected by a hub or a bus, any hosts pair of such a set have the same connectivity. Hence, it is sufficient to measure it for a pair of hosts and use the result for all possible host pairs.

Having a good knowledge about the network topology is clearly fundamental to achieve a good NWS deployment. More precisely, we need to be able to determine which inter-host connexion interfere on which other because they share a physical link. We now present some different ways to gather these informations automatically.

3. Network mapping solution

The OSI Network Model presents seven layers of abstractions, each of them providing a different view of the network. The lower ones are closer to the hardware reality while the higher ones offer a greater abstraction. When considering the network topology, we have therefore to specify the layer considered since it will have a great impact both on the way to discover it and on its possible uses.

The most commonly used topologies are either at the layer 2 or at the layer 3. Layer 2 is the *Medium Access Control* or *Datalink* layer, corresponding to the level of the *Ethernet* protocol. Layer 3 is the *Network* one and involves protocols such as the *Internet Protocol* (IP). The layer 2 is therefore closer to the physical links than the layer 3, and may be used to get useful information about routers and switches that are not directly available from layer 3. On the other hand, layer 3 is closer to the user-level applications view of the network.

3.1. SNMP and BGP

The layer 2 of the OSI model is the one where LAN are defined and configured. It is therefore possible to ask directly to the network components about their configuration as expressed by network administrators using for example the *Simple Network Management Protocol* (SNMP). Albeit, some old or dumb switches and routers do not answer to SNMP requests while other equipments requires the use of proprietary tools and protocols such as the Cisco's Discovery Protocol[1] or Bay Networks' Optivity Enterprise[2].

Completing this view for a WAN can be done by using the *Border Gateway Protocol* (BGP), which is used to exchange routing informations between the autonomous systems composing Internet.

[1] http://www.cisco.com .
[2] http://www.baynetworks.com .

The Remos [5] tool uses the SNMP protocol to construct the local network topology, plus some simple benchmarks to gather informations about WAN [11]. It is then able to reconstruct the part of the network where dumb routers do not answer to the requests in order to get a complete topology.

The advantage of this approach is that such tools directly access to the network configuration as expressed by the network administrator. They are therefore very quick and not intrusive. On the other hand, the use of those protocols is almost always restricted to the authorized users. This is due to two major reasons: security, since it is possible to conduct Deny Of Service attacks by flooding the routers of requests, and privacy, since ISP generally do not like to publicly expose the possible bottlenecks of their networks.

As a matter of fact, this limitation is simply not acceptable in a metacomputing context. Since Grid platforms traditionally involve several well established and large organizations such as universities, obtaining the grant to use level 2 protocols can be very time consuming, and even reveal impossible due to human factors.

But on the other hand, any solution based on level 3 and above are more complicated to design and use because for example of the VLAN technology. It allows to present a logical view of the network differing from the physical reality to the higher layers. It enables network administrators to split the physical network into several logical ones. This is for example used in our lab to separate for security reasons in several networks the machines administrated only by the staff from the laptops and such on which the users may become root. Extra provisions are needed to take such things into account when mapping the network.

3.2. Tomographic approach

Tomography is a methodology used for example in medical imaging to reconstruct a 3D view of an object from several 2D ones taken from different point of view. Likewise, several solutions allow to rebuild a complete view of the network by merging the local views obtained from different hosts. Those methods mainly differ on the methodology used to get the local views to be merged.

The classical `ping` tool reports the round trip time between two hosts. Projects like IDMaps [9] or Global Network Positioning [13] use this method to compute a network topology by clustering the network using this distance metric. This is clearly not sufficient for our needs. We need more information on links, such as their bandwidth (which IDMaps may sometimes provide) and how they would be shared between several streams using them concurrently.

The layer 3 of the OSI model is the one where internetwork connectivity is configured. To avoid infinite loops, all packets are given on creation a given *Time To Live* (TTL). This value is decreased by each router transmitting the packet, and when it becomes zero, the packet is destroyed and an error is signified to the emitter.

This feature is used by the classical `traceroute` tool (and by projects developing wrappers to `traceroute` such as TopoMon [4] or Lumeta [1]). Since most routers indicate their address in the error message generated when the TTL becomes zero, `traceroute` can discover most hops on a given network path by sending several packets with increasing TTLs.

This approach has several drawbacks. First, since routers can return different addresses, combining the paths can be non-trivial. Then `traceroute` gives no information on how concurrent transfers interact when sharing a given link, not even the available bandwidth. In fact, `traceroute` does not report the information relevant in our context: It focuses on the path followed in the network by the packets while we are interested in a more macroscopic view indicating the effects of this data movement at an application level.

Another important issue with `traceroute`-based network mapping solution is that it relies on the fact that routers return an error message when the TTL of a packet becomes zero. Unfortunately, this is not always the case as administrators can disable this feature to avoid Deny Of Service attacks based on flooding. Therefore, more recent work in network tomography focus on the design of new measurement methods relying only on classical packets. In [15], the authors presents a measurement methodology based only on regular packet exchanges.

As this approach does not require any specific privilege, it is very appealing in our context. Unfortunately, the obtained view of the platform does not fulfill our needs. It focuses on the physical interconnection topology and the description of the path followed by the packets while our goal is to identify the interferences between concurrent streams. These two notions are very close, but do not necessary match. If, using this methodology, two paths are reported to be independent (i.e. they do not share any network element), it is clear that data streams using these paths cannot interfere, but the contrary is not necessary true. Indeed, if the shared section is over-dimensioned and able to carry both streams without impacting on their performance, we want our tool to report the paths as independent. This goal's divergence makes this approach unusable directly in our context, even if it could constitute a first guess of the topology.

3.3. Link capacity

`pathchar` is the solution proposed by Jacobson (the `traceroute`'s author) to gather not only the network organization, but also the capacity of each link. As

`traceroute`, packets with different TTLs are transmitted, but the size of these packets also varies. Analyzing the time before the error packet is received, it is possible to infer the latency and bandwidth of each link in the path, the distribution of queue times, and the packet loss probability [6].

The first problem with `pathchar` is that it needs to assume that the sent probes will have negligible queuing delays on all encountered link. Since the probability of this event is rather low on loaded networks, this tool needs to conduct a lot of experiments to be sure that one of them respects this assumption. In current implementations, more than 1500 probes are usually used, so tests can last for hours on highly loaded routes constituted of many hops. Moreover, it only gives the capacity of encountered links and not how their bandwidth will be shared between several competing data streams.

Finally, in order to forge the packets needed for its experiments, `pathchar` needs to be given the super-user privileges on the machines where it runs, which is clearly unacceptable in Grid context.

3.4. Summary

We cannot get the information from where it was configured by network administrators (using SNMP or BGP) because it requires special privileges which cannot always be obtained on a Grid platform. As a consequence, it means that we cannot detect directly the use of technologies like VLAN, commonly used by network administrators to present a logical view different of the physical reality, and special provisions are needed to take this into account.

Existing layer 3 methods are not sufficient for us since they either fail to report which part of the network path are limiting (*i.e.* the ones on which a bandwidth shortage is possible) or need super-user privileges to be used (like `pathchar` does).

We decided to use ENV, presented in next section, because it captures a view of the network well adapted to our needs with no need of special privileges to run.

4. Effective Network View

The *Effective Network View* (ENV [16]) project was developed by Gary Shao at University of California, San Diego. ENV is primarily designed to improve the master/slave paradigm and allows to discover the effective topology of a network from the point of view of a given host. Data acquired are then dependent from the master's choice, as explained in [16].

The main advantage of ENV is that it creates an effective profile of network configuration based only on user-level observations and thus allows to get informations on layer 2 and 3 routers without using low-level tools or protocols which could not be accessible on a Grid platform.

We now detail how ENV collects the data, taking the mapping of the ENS-Lyon network as example. This process can be split in two parts: the first is independent from the choice of the master, while the other is not.

4.1. Structural topology: Master-independent data

This constitutes a first approximation of the topology builded with `traceroute`, and used to guide the active tests of latter phases. Each host involved in the mapping reports the path used to get out of the Grid by executing a `traceroute` to a well known external destination. The part within the mapped network is used to build a tree. Hosts using the same route to get out of the studied network are clustered together as leaves on the same branch.

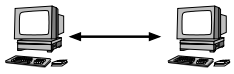
This tree is based on `traceroute` and, as explained in Section 3.2, is therefore not sufficient to detect the interferences between streams. To fulfill our needs, a few more tests, depending on the master's point of view, are conducted.

4.2. Effective topology: Master-dependent data

The second part of the data collection depends on the chosen master. These measurements can be seen as successive refinements of the network structural view. This allow to generate a new tree containing the so-called ENV networks, which contain the critical informations in our context about the layer 2 topology.

Most of these experiments use thresholds to interpret the measurement results. The value of these thresholds may have a great impact on the mapping results, and were determined experimentally and empirically by the ENV authors.

4.2.1 Host to host bandwidth

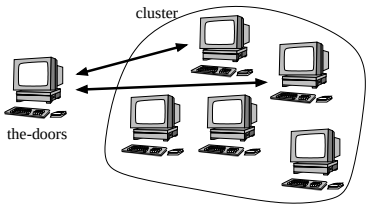


This experiment splits the clusters in order to group the machines having a comparable connexion to the master together. The bandwidth between the master M and any host A is measured separately. Then, clusters previously made are divided into groups with similar bandwidth. If the bandwidth ratio between two hosts exceeds the threshold of 3, their cluster is split to separate them.

4.2.2 Pairwise host bandwidth

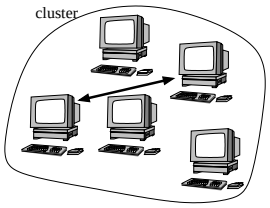
This experiment splits the clusters based on how much the connexion between their members and the master (M) is interfering. For each pair of machines A and B in each clus-

ter, the bandwidths of MA and MB are thus tested whenever the transfers occur concurrently.



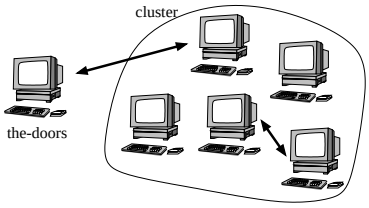
This measurement is then compared to the bandwidth measured in the previous step. If the ratio of the bandwidth (MA) in normal conditions and the one obtained when paired with MB is below a threshold of 1.25, A is declared independent of B and the cluster is split to separate those hosts.

4.2.3 Internal host bandwidth



For each cluster, bandwidth is measured between any pair of machine, within this cluster. This allows to set the local bandwidth parameters for a given cluster. This may be useful for clusters where local bandwidth is different from bandwidth achieved between the cluster and the master. For example in ENS-Lyon, the route from *the-doors* to the *popc* cluster goes trough a 10 Mbps bottleneck, whereas *popc* is on a local 100Mbps hub.

4.2.4 Jammed bandwidth

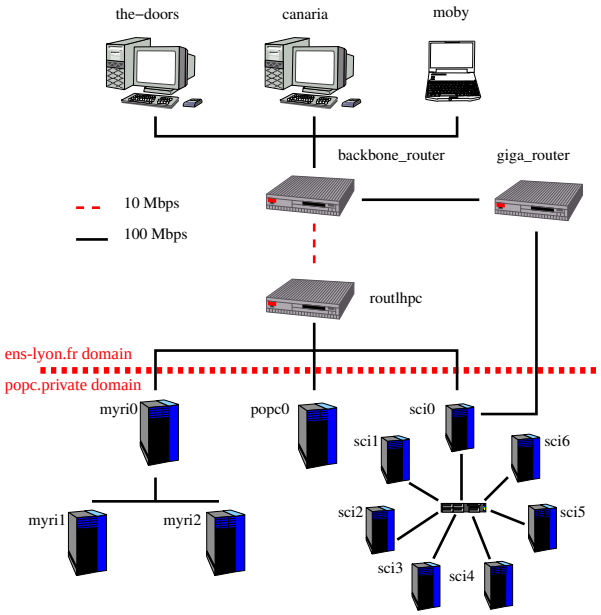


For each cluster, the bandwidth to the master is measured while a transfer between two other hosts of that cluster occurs. This measure is repeated 5 times, and the average of the ratio $Bandwidth/Bandwidth_{jammed}$ is computed.

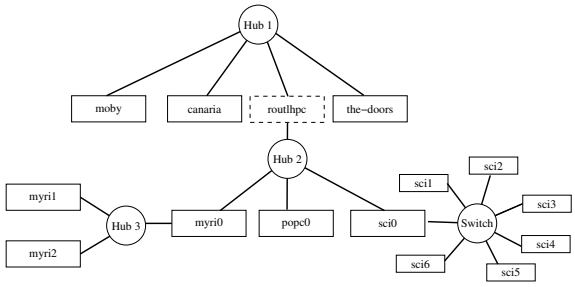
If the average is below the threshold of 0.7, the cluster is reported to be on a shared link. If the average is above the threshold of 0.9, the cluster is reported to be on a switched link. If the average is between those two thresholds, data gathering about this cluster stops since the values are not significant enough.

All these refinements allow ENV to determine whether structural networks are switched or shared based upon bandwidths observed between hosts. This information has a great impact on the NWS deployment, as explained in section 5.

4.3. Execution result



(a) Physical topology (simplified schema).



(b) Effective topology from the-doors's point of view.

Figure 1. Topology of the test network.

Figure 1 presents the results of ENV running on the ENS-Lyon network. Figure 1(a) shows the physical network topology while Figure 1(b) is the result of an ENV run. The presented physical topology is simplified and asymmetric routes as well as used VLANs are not pictured. The pictured effective network view is the one obtained when choosing `the-doors` as master.

We can see that the vision offered by ENV is much more simplified than the physical one since it does report only the routers which cannot be suppressed from the topology without changing its structural characteristics. Besides, ENV reports the fact that `popc0`, `myri0` and `sci0` are on a 100 Mbps hub, whereas links to reach `popc0` and `myri0` from `the-doors` must go through a bottleneck at 10 Mbps.

4.4. Known ENV issues

This section presents several issues we discovered during these experiments and that may impact the applicability of ENV.

Master/Slave paradigm ENV was designed for master/slave computing, and this may lead to important information loss during the mapping. For example, it may prevent this tool to detect direct links between slaves since the packets emitted from the master never use them. This information loss seems to be the price to pay for rapidity, since gathering all information about the network may require a very long time to complete. Indeed, using exactly the same methodology as ENV for a whole mapping would require to first drive $n * (n-1)$ bandwidth tests between each couple of hosts $\{a; b\}$. Then, it would require for each pair of link $\{a; b\}$ and $\{c; d\}$ to conduct experiments to determine whether those network path are dependent or not, *i.e.* whether a transmission on $\{a; b\}$ impacts the bandwidth of $\{c; d\}$ or not. This naive algorithm would not scale at all because of the bandwidth and time consumed. Considering that collecting information about two given links lasts half a minute (which is reasonable since the network needs to stabilize between each experiments), the whole process would last about 50 days for 20 hosts. That is why ENV does not try to completely map the network, but only focuses on a view of the network from a given point of view.

Bandwidth waste Even if focusing on the master/slave paradigm greatly improves the performance of ENV, these tests still introduce a significant bandwidth consumption. It cannot be prevented since active probes are needed to infer layer 2 informations without relying on specific tools which may not be available. Nevertheless, a given platform needs to be mapped with ENV only once, and the results could then be shared between different people. For example, administrators could publish the mapping of their network as reported by ENV, so that any user can use it without recomputing the mapping.

Asymmetric routes The physical network in ENS-Lyon is more complex than depicted on Figure 1(a). Indeed the route between *the-doors* and *popc* goes trough a 10 Mbps link, whereas the other direction uses only 100 Mbps links. According to [14], this situation, even if quite queer, is rather common on Internet. Since ENV bandwidth tests are conducted in only one way, the system cannot detect such issues. Solving this would imply almost a complete rewrite of ENV tests and is still to do.

Likewise, networks are usually packet driven and we cannot guarantee that the path between two given hosts always follows the same route. However, since ENV is mainly used to map local networks, we assume that this is the case for the duration of the experiment, or at least, that global measurements are not too sensitive to this variation.

Reliability and accuracy Even if ENV provides some bandwidth estimations, their accuracy is not crucial in our context since NWS is to be deployed, and provides much more accurate and up-to-date measurements. Qualitative and topological information about the network are therefore much more important than quantitative one.

Nevertheless, the first problem is the possible platform evolution: many inter-dependent tests are to be run. The results given by ENV may then be corrupted if the network load evolves greatly (increasing or decreasing) between tests. There is no solution yet to this problem, except rapidity: the mapping of our platform only last a few minutes, so we can assume that the environment is stable enough to provide meaningful results in those conditions. For bigger platforms. it would be possible to map separately the different parts of the network and then merge the results together.

Moreover, experimental thresholds may be problematic, because they may be specific to platform characteristics like the media type. They may for example be adapted to LAN, but not to Terabit links. Besides, determining if the thresholds are adapted to the current platform is very difficult.

5. Deploying the NWS using ENV

Most common Grid testbeds are constituted of several organizations inter-connected by a wide area network, each of them sharing local resources (like clusters) interconnected by local area network with the others. The resulting platform is a WAN constellation of LAN resources. The most natural solution to satisfy all constraints expressed in Section 2 is then to set up a hierarchical monitoring infrastructure, where intra-site connectivity is tested separately from the inter-site one. If needed, this hierarchy can contain more than two levels, and intra-cluster connectivity would be measured separately from the inter-cluster one.

5.1. Deployment design

This section presents a simple algorithm to determine the NWS deployment using the data gathered by ENV. For each network or subnetwork discovered by ENV, our deployment plan contains at least two cliques:

- If the network is shared, its hosts are supposed to be on the same physical link, so the latency and bandwidth of one couple of hosts is representative for any possible couple. The intra-network connectivity is then

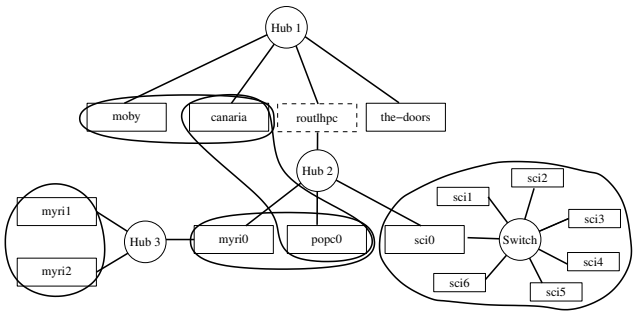


Figure 2. NWS deployment plan in ENS-Lyon.

measured by a clique containing two arbitrary chosen hosts.

Note that even if this is sufficient to gather the needed information, this method reveals a NWS shortcoming: it is currently not possible to inform the system that the connexion between two hosts (AB) is representative of the connexion between two other hosts (CD), and the user has to keep track of this to ask NWS about (AB) when he wants to retrieve the characteristics of (CD).

- If the network is switched, the network characteristics between each host pair are independents and could be measured separately, but each host must be involved in at most one measurement at a given time. That is why we deploy a NWS clique containing all the hosts to make sure that only one measurement will occur at the same time on the given group of hosts.

  Note that using a clique is a bit too restrictive. On a switched network, the tests AB and CD would not collide if they involve different hosts, *i.e.* if $\{AB\} \cap \{CD\} = \emptyset$. On the other hand, the only drawback of forbidding concurrent tests over the group is that the frequency of tests between two given hosts slightly decreases.

The resulting deployment plan for the ENS-Lyon network is depicted on Figure 2. The *sci* cluster is switched, so we pick all its machines to form a new clique, whereas the *myri* cluster is shared, so we pick only two hosts for the local clique (*myri1* and *myri2*). *myri0* and *popc0* were chosen to test the network characteristics on *Hub 2* while *moby* and *canaria* are used to test the *Hub 1*. The connection between *canaria* and *popc0* is used to test the connexion between these hubs.

5.2. Effective deployment

Once the deployment plan has been computed, another challenging issue is to actually apply it, and launch the several parts of NWS on the different hosts with the right options. Indeed, the official version of NWS offers very few support to process management and global configuration. We have to manually ssh to the right host, and pass manually the right option on the command line of each NWS process.

To solve this issue, we designed a simple NWS manager program using a configuration file shared across all involved hosts and applying the local parts on each hosts. The actual deployment of NWS is then as easy as dispatching the configuration file to the hosts (using for example NFS), and running the manager on each machines. This program parses the given configuration file and starts the NWS components with the appropriate options for the machine on which the manager runs. It thus constitutes a convenient way to apply the deployment decisions of the administrator or of an automatic deployment tool.

6. Conclusion

The *Network Weather Service* constitutes a *de facto* standard of Grid platform monitoring tool used by major Grid PSEs. Unfortunately, deploying this tool remains tedious since no automatic tool is provided by default, and since the manual deployment requires insight about both the target network and the NWS internals.

The main difficulty remains to accurately map the target network, and ENV seems to be the most suited tool to our needs among the existing ones. Unlike SNMP-based tools or pathchar, it does not require any specific privileges to run. This is very important on a Grid constituted of several organizations sharing their local resources, since obtaining extra privileges on all parts of the platform can reveal difficult due to human factors. Unlike classical network tomography solutions (based on ping, traceroute, or other measurement methodology) ENV is also able to quantify how each detected network is shared among several concurrent transfers, which reveals to be a crucial information to achieve a proper NWS deployment.

After having characterized the constraints a “good” NWS deployment should satisfy, we proposed a simple algorithm to deduce a deployment plan from the information provided by ENV, and have applied it on the network of the ENS-Lyon laboratory.

This experiment allowed us to identify several shortcomings of the ENV. First, it considers that the routes are symmetric and only test them in one direction. According to [14], this simplification is over-optimistic. Moreover, ENV only provides a tree view of the network to simplify and speed up the mapping process. This is well adapted to a master/slave paradigm, but is a way too limited in the general case since it will overlook some transversal links between leaves of the tree. In order to overcome these issues,

we initiated a new project called ALNEM [10] that constitutes a generalization of the ENV approach with stronger theoretical basements.

Then, the interpretation of the measurements conducted by ENV depends on thresholds whose values were determined empirically. Their values may have a great impact on the results and thus on the quality of the mapping. Yet, an accurate study of this impact is still to be done. Since the quality of those values may well depend on the targeted class of network (LAN vs. WAN), an heuristic to determine them would be necessary to meet our goal of automatic deployment of the NWS.

Other concerns revealed by this experiment involve the NWS itself. First, it does not provide process managing facilities, and we designed a simple NWS manager in charge of applying the deployment plan locally to each machine.

Moreover, the protocol used to ensure that experiments never collide on a given resource (which would lead to wrong measurements) could still be improved. Currently, it ensures that only one pair of hosts from a given group will conduct an experiment at a given time. But on a switched network, more than one experiment may be authorized if the hosts involved in each experiments are different. That is to say that a possibility to lock *hosts* (and not networks) is still needed. Moreover, if the network is shared (using an hub), the connexion capacities between each pair of host will be the same, and testing one pair is enough. Future works would also be necessary to explore the conditions under which network measurements may be combined (when lacking direct experiments) in order to reduce the number of monitored paths and thus improve the system scalability. Active undergoing collaboration with the NWS team should allow us to overcome some of these issues.

Acknowledgment

The authors thank the anonymous reviewers and the Professor Rich Wolski for their insightful suggestions about and beyond this paper.

References

[1] H. Burch, B. Cheswick, and A. Wool. Internet mapping project. http://www.lumeta.com/mapping.html.

[2] E. Caron, F. Desprez, F. Lombard, J.-M. Nicod, M. Quinson, and F. Suter. A Scalable Approach to Network Enabled Servers. In B. Monien and R. Feldmann, editors, *Proceedings of the 8th International EuroPar Conference*, volume 2400 of *Lecture Notes in Computer Science*, pages 907–910, Paderborn, Germany, Aug. 2002. Springer-Verlag.

[3] H. Casanova and J. Dongarra. Using Agent-Based Software for Scientific Computing in the Netsolve System. *Parallel Computing*, 24:1777–1790, 1998.

[4] M. den Burger, T. Kielmann, and H. E. Bal. TOPOMON: A monitoring tool for grid network topology. In *International Conference on Computational Science (ICCS 2002)*, volume 2330, pages 558–567, Amsterdam, April 2002. LNCS.

[5] P. Dinda, T. Gross, R. Karrer, B. Lowekamp, N. Miller, P. Steenkiste, and D. Sutherland. The architecture of the remos system. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, August 2001.

[6] A. B. Downey. Using pathchar to estimate internet link characteristics. In *Measurement and Modeling of Computer Systems*, pages 222–223, 1999.

[7] I. Foster and C. K. (Eds.). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[8] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applications*, 11(2):115–128, 1997.

[9] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. Idmaps: A global internet host distance estimation service. *IEEE/ACM Transactions on Networking*, oct 2001.

[10] A. Legrand, F. Mazoit, and M. Quinson. An application-level network mapper. Technical Report 2002-09, LIP, Feb. 2002.

[11] N. Miller and P. Steenkiste. Collecting network status information for network-aware applications. In *INFOCOM'00*, pages 641–650, 2000.

[12] H. Nakada, M. Sato, and S. Sekiguchi. Design and Implementations of Ninf: towards a Global Computing Infrastructure. *Future Generation Computing Systems, Metacomputing Issue*, 15(5–6):649–658, Oct. 1999.

[13] E. Ng and H. Zhang. Predicting internet network distance with coordiantes-based approaches. In *IEEE INFOCOM'02*, 2002.

[14] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, 1997.

[15] M. Rabbat. Multiple source network tomography. Master's thesis, Rice University, May 2003.

[16] G. Shao, F. Berman, and R. Wolski. Using effective network views to promote distributed application performance. In *International Conference on Parallel and Distributed Processing Techniques and Applications*, June 1999.

[17] R. Wolski, B. Gaidioz, and B. Tourancheau. Synchronizing network probes to avoid measurement intrusiveness with the Network Weather Service. In *9th IEEE High-performance Distributed Computing Conference*, pages 147–154, August 2000.

[18] R. Wolski, N. T. Spring, and J. Hayes. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. *Future Generation Computing Systems, Metacomputing Issue*, 15(5–6):757–768, Oct. 1999.