A First Step Towards Automatically Building Network Representations

Lionel Eyraud Dubois[1] Arnaud Legrand[2] Martin Quinson[3] Frédéric Vivien[1]

[1] LIP, CNRS - ÉNS Lyon - INRIA - UCBL, Lyon, France
Lionel.Eyraud-Dubois@ens-lyon.fr, Frederic.Vivien@ens-lyon.fr
[2] LIG - MESCAL, Grenoble, France
Arnaud.Legrand@imag.fr
[3] Nancy University / LORIA, Nancy, France
Martin.Quinson@loria.fr

Abstract. To fully harness Grids, users or middlewares must have some knowledge on the topology of the platform interconnection network. As such knowledge is usually not available, one must uses tools which automatically build a topological network model through some measurements. In this article, we define a methodology to assess the quality of these network model building tools, and we apply this methodology to representatives of the main classes of model builders and to two new algorithms. We show that none of the main existing techniques build models that enable to accurately predict the running time of simple application kernels for actual platforms. However some of the new algorithms we propose give excellent results in a wide range of situations.

keywords: Network model, topology reconstruction, Grids.

1 Introduction

Grids are parallel and distributed systems that result from the sharing and aggregation of resources distributed between several geographically distant organizations [12]. Unlike classical parallel machines, Grids present heterogeneous and sometimes even non-dedicated capacities. Gathering accurate and relevant information about them is then a challenging issue, but it is also a necessity. Indeed, the efficient use of Grid resources can only be achieved through the use of accurate network information. Qualitative information such as the network topology is crucial to achieve tasks such as running network-aware applications [17], efficiently placing servers [7], or predicting and optimizing collective communications performance [15].

However, the description of the structure and characteristics of the network interconnecting the different Grid resources is usually not available to users. This is mainly due to security (fear of Deny Of Service attacks) and privacy reasons (ISP do not want you to know where their bottlenecks are). Hence a need for

tools which automatically construct models of platform networks. Many tools and projects provide some network information. Some rely on simple ideas while others use very sophisticated measurement techniques. Some of these techniques, though, are sometimes ineffective in Grid environments due to security issues. Anyway, to the best of our knowledge, these different techniques have never been compared rigorously in the context of Grid computing platforms. Our aim is to define a methodology to assess the quality of network model building tools, to apply it to representatives of the main classes of model builders, to identify weaknesses of existing approaches, and to propose new model building algorithms.

The main contributions of this paper is the design of two new reconstruction algorithms, and some evaluations that highlight the weaknesses of classical algorithms and demonstrate the superiority of one of our new algorithms.

The rest of this article is organized as follows. In Section 2, we review the main observation techniques and we identify some that are effective in Grid environments. In Section 3 we review existing reconstruction algorithms and we identify a few representative ones. Based on the analysis of potential weaknesses of these algorithms, we propose two new algorithms. We recall in Section 4 a few quality metrics we proposed in [11] to assess the quality of reconstruction algorithms. In Section 5, we evaluate through simulation the quality of the studied reconstruction algorithms with respect to all the proposed metrics. This evaluation is performed on models of real platforms and on synthetic models.

2 Related Work

Network discovery tools have received a lot of attention in recent years. However, most of them are not suited to Grids. Indeed, much of the previous work (e.g., Remos [9]) rely on low-level network protocols like SNMP or BGP, whose usage is generally restricted for security reasons (it is indeed possible to conduct Deny Of Service attacks by flooding the routers with requests). As a matter of fact, in a Grid, even traceroute or ping-based tools (e.g., TopoMon [8], Lumeta [2], IDmaps [14], Global Network Positioning [20]) are getting less and less effective. For example, ICMP is more and more often disabled, once again to avoid Deny Of Service attacks based on flooding: the Skitter project [4] reports that over 5 years the number of hosts replying to ICMP requests decreased by 2 to 3% per month. Even if recent works have proposed similar or even better functionalities without relying on ICMP, some of them (e.g., pathchar [10]) require specific privileges, which make them unusable in our context. It is mandatory to rely on tools that only use *application-level measurements*, i.e., measurements that can be done by any application running on a computing Grid without any specific privilege. This comprises the common end-to-end measurements, like bandwidth and latency, but also interference measurements (i.e., whether a communication between two machines A et B has non negligible impact on the communications between two machines C et D). Many projects rely on this type of measurements. An example is the NWS (Network Weather Service) [24] software, which constitutes a *de facto* standard in the Grid community as it is used by major

Grid middlewares like Globus [13], or *Problem Solving Environments* (PSEs) like DIET [5], to gather information about the current state of a platform and to predict its evolutions. NWS is able to report the end-to-end bandwidth, latency, and connection time, which are typical application-level measurements. However, the NWS project focuses on quantitative information and does not provide any kind of topological information. It is however natural to address this issue by aggregating all NWS information in a single clique graph and use this labeled graph as a network model. In another example, interference measurements have been used in ENV [22] and enabled to detect, to some extent, whether some machines are connected by a switch or a hub. A last example is ECO [18], a collective communication library, that uses plain bandwidth and latency measurements to propose optimized collective communications (e.g., broadcast, reduce, etc.). These approaches have proved to be very effective in practice, but they are generally very specific to a single problem and we are looking for a general approach.

3 Studied Reconstruction Algorithms

We are thus looking for a tool based on application-level measurements that would enable any network-aware application to benefit from reasonably accurate information on the network topology. In most previous works, the underlying network topology is either a clique [24,18] or a tree [3,22]. Our reference reconstruction algorithms are thus clique, minimal spanning tree on latencies, and maximal spanning tree on bandwidths. As our experiments show (Section 5), these methods produce very simple graphs, and often fail to provide a realistic view of platforms. We thus designed two new reconstruction algorithms, as a first step towards a better reconstruction. The first algorithm aims at improving an already built topology and is meant to be used to improve an existing spanning tree. The second one reconstructs a platform model from scratch, by growing a set of connected nodes. Both algorithms keep track of the routing while building their model, to be able to correct a route connecting two nodes whose latency was previously inaccurately predicted. We focus on latency rather than on bandwidth as bandwidths are less discriminant.

Algorithm Improving. Algorithm Improving is based on the observation that if the latency between two nodes is badly over-predicted by the current route connecting them, an extra edge should be inserted to connect them through an alternate and more accurate route. Among all pairs of "badly connected" nodes, we pick the two nodes with the smallest possible measured latency, and we add a direct edge between them. Each time Improving adds an edge, for each pair of nodes whose latency is over-predicted, we check whether that pair cannot be better connected through the just introduced edge, and we update the routing if needed. This edge addition procedure is repeated until all predictions are considered sufficiently accurate. The accuracy of predictions is necessarily

arbitrary. In our implementation, it corresponds to a deviation of less than 10% from actual measurements.

Algorithm AGGREGATE. Algorithm AGGREGATE uses a more local view of the platform. It expands a set of already connected nodes, starting with the two closest nodes in terms of latency. At each step, AGGREGATE connects a new *selected* node to the already connected ones. The selected node is the one closest to the connected set in terms of latency. AGGREGATE iteratively adds edges so that each route from the selected node to a connected node is sufficiently accurate. Added edges are greedily chosen starting from the edge yielding a sufficiently accurate prediction for the largest number of routes from the selected node to a connected node. We slightly modified this scheme to avoid adding edges that will later become redundant. A new edge is added only if its latency is not significantly larger (meaning less than 50% larger) than that of the first edge added to connect the selected node. Because of this change, we may move to a new selected node while not all the routes of the previous one are considered accurate enough. We thus keep a list of *inaccurate* routes. For each edge addition we check whether the new edge defines a route rendering accurate an inaccurate route. When all nodes are connected, we add edges to correct all remaining inaccurate routes, starting with the route of lowest latency.

4 Assessing the Quality of Reconstructions

We want to thoroughly assess the quality of reconstruction algorithms. To fairly compare various topology mapping algorithms, we have developed ALNEM (Application Level Network Mapper). ALNEM is developed with GRAS [21] that provides a complete API to implement distributed application on top of heterogeneous platforms. Thanks to two different implementations of GRAS, ALNEM can work seamlessly on real platforms as well as, with SIMGRID [16], on simulated platforms. ALNEM is made of three main parts: 1) a measurement repository; 2) a distributed collection of sensors performing bandwidth, latency, and interference measurements; 3) a topology builder which uses the repository.

To assess the quality of model builders, we use two different and complementary approaches. For both approaches, we consider a series of original platforms; and for each platform we compare the original platform and the models built from it. The two approaches can be seen as different points of view on models: a communication-level one and an application-level one.

4.1 End-to-End Metric

A platform model is "good" if it allows to accurately predict the running time of applications. The prediction accuracy depends on the model capacity to render different aspects and characteristics. Often, researchers only focus on bandwidth prediction. However, latencies and interferences can also greatly impact an application performance. Therefore, we consider the three following characteristics:

Bandwidth. We need to know the available bandwidths as soon as the applications send messages of different sizes.

Latencies. Latencies are often overlooked in Grid computing, but Casanova presented an example [6] where one third of the time needed to transfer a 1 GByte of data would be due to latencies. Therefore, we must model them.

Interferences. Many distributed applications use collective communications (e.g., broadcasts or all-to-all) or independent communications between disjoint pairs of processors. The only knowledge of latencies and bandwidths does not allow to predict the time needed to realize two communications between two disjoint pairs of processors: this depends on whether the two communications use a same physical link[4]. Also, knowing the network topology and being able to predict communication interferences enables to derive more efficient algorithms [17].

Methodology. Our evaluation methodology is based on simulations. Given one original platform, we measure the end-to-end latencies and bandwidths between any two processors. We also measure the end-to-end bandwidths obtained when any two pairs of processors simultaneously communicate. We then perform the same measurement on the reconstructed models. To compare the results, we build an accuracy index for each reconstruction algorithm, each graph, and each studied network characteristic. For latencies and bandwidths, following [23], we define *accuracy* as the maximum of the two ratios x_R/x_M and x_M/x_R, where x_R is the reconstructed value and x_M is the original measured one. We compute the accuracy of each pair of nodes, and then the geometric mean of all accuracies.

4.2 Application-Level Measurements

To simultaneously analyze a combination of the characteristics studied with end-to-end measurements, we compare, through simulations, the performance of several classical distributed routines when run on the original graph and on each of the reconstructed ones. This evaluates the predictive power of the reconstruction algorithms for applications with more complex but realistic communication patterns. We study the following simple distributed algorithms (listed from the simplest communication pattern to the most complicated one):

- **Token ring:** a token circulates three times along a randomly built ring (the ring structure is not correlated to that of the interconnection network).
- **Broadcast:** a randomly picked node sends a message to all the other nodes.
- **All-to-all:** all the nodes simultaneously perform a broadcast.
- **Parallel matrix multiplication:** a matrix multiplication is realized using ScaLAPACK outer product algorithm [1].

[4] It also depends whether the shared communication link is a backbone [6], etc.

This evaluation must be done through simulations. Indeed, the measurements on the reconstructed models can obviously not be done experimentally. Furthermore, the comparison of experimental (original platform) and simulated (reconstructed models) measurements would introduce a serious bias in the evaluation framework, due to the differences between the actual world and the simulator.

5 Experimental Results

We present two types of experiments: the first one is based on a modeling of a real network architecture, while for the second one we generated synthetic platforms using GridG [19]. As stated in section 3, we evaluate several reconstruction algorithms. In addition to our three reference reconstruction methods (Clique, minimal spanning tree on latencies (TreeLat), and maximal spanning tree on bandwidths(TreeBW)), we analyze the performance of the Aggregate algorithm, and of the Improving procedure applied to both spanning trees: ImpTreeLat and ImpTreeBW.

5.1 Renater

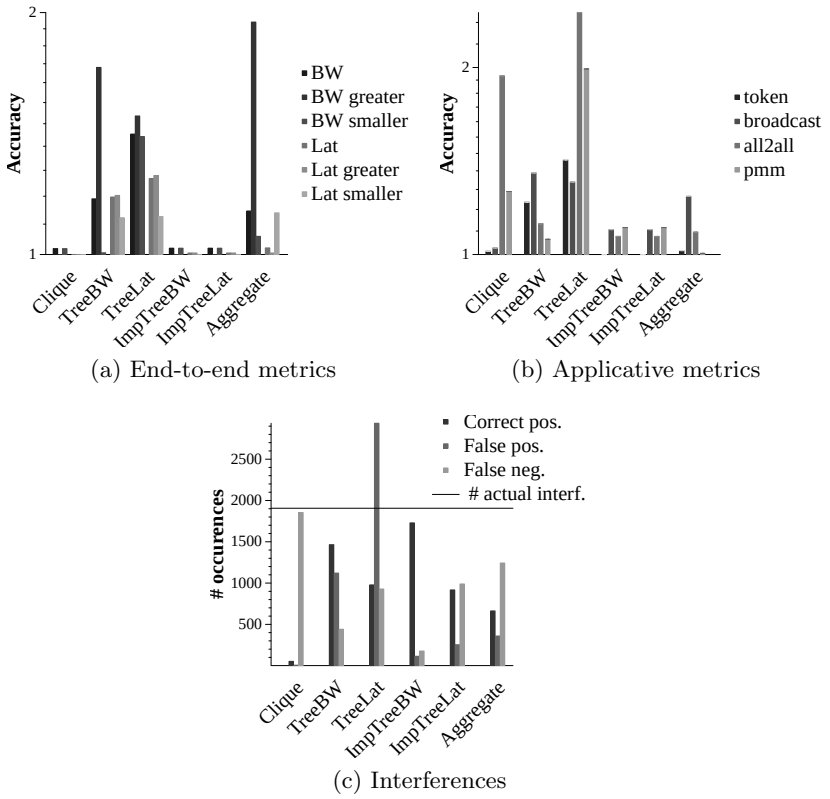
Renater[5] is the French public network infrastructure that connects major universities. Thanks to a collection of accounts in several universities, we were able to measure latencies and bandwidths between the corresponding hosts. For security reasons, these measurements were performed using the most basic tools, namely `ping` for latency and `scp` of bandwidths. Thanks to the topology information available on the Renater website we created a model of this network, that we annotated with the bandwidths and latencies we measured. We then executed our reconstruction algorithms on the obtained model.

Figure 1 shows the evaluation of the reconstructed topologies. For end-to-end metrics, we plotted the average accuracy for both latency and bandwidth, and we also detailed the average accuracy for over- and under-predicted values. Unsurprisingly, Clique has excellent end-to-end performances whereas TreeLat and TreeBW have poor ones. Aggregate over-estimates bandwidth for a few couples, but both ImpTreeLat and ImpTreeBW have excellent end-to-end performances.

Regarding applicative performance, Clique is unsurprisingly good for token and broadcast where there is always at most one communication at a time and very bad for all2all and pmm. ImpTreeLat and ImpTreeBW are once again equivalent and now clearly have much better results than any other heuristics. They are actually within 10% of the optimal solution for all applicative performances. Last, the interference evaluation (Figure 1c) enables us to distinguish ImpTreeBW and ImpTreeLat. ImpTreeBW accurately predicts more than 95% of interferences whereas ImpTreeLat overestimates 50% of interferences!

[5] http://www.renater.fr

This experiment shows that our reconstruction algorithms are able to yield platforms with good predictive power. It also suggests that our ImpTreeBW algorithm can provide very good reconstructions. The good performance of ImpTreeBW may be explained by the fact that this is the only algorithm which builds a non trivial graph (i.e.,, not a clique) while using both the information on latencies and bandwidths. However, these encouraging results obtained on a realistic platform must be confirmed by a more comprehensive set of experiments, using a large number of different platforms, which we do in the next section.



(a) End-to-end metrics

(b) Applicative metrics

(c) Interferences

Fig. 1: Simulated tests on the Renater platform.

5.2 GridG

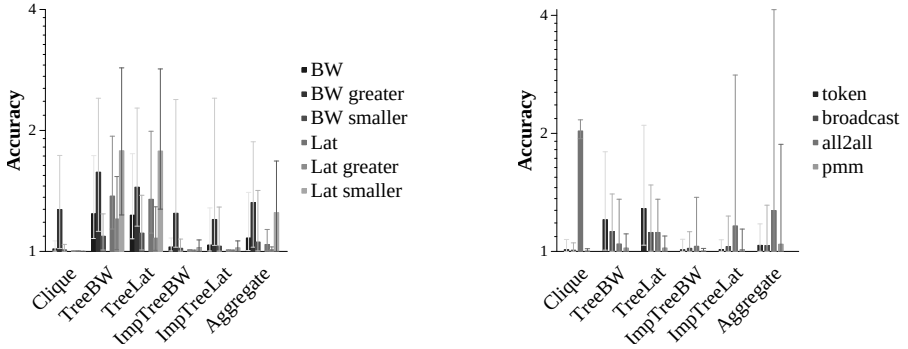
For a thorough validation of our algorithms, we used the GridG platform generator [19] to study realistic Internet-like platforms, which may be different from the very few platforms we can access and thus test directly. In this experiment,

we generated 2 different kinds of platforms: in the first group, all of the hosts are known to the measurement procedure, which means that it is possible to deploy a process on all internal routers of the platform. In the second group, only the external hosts are known to the algorithms. For each group, we generated 40 different platforms, each of them containing about 60 hosts.

The results are shown on Figures 2 and 3. For end-to-end metrics, we plotted the average accuracy for both latency and bandwidth, and we also detailed the average accuracy for over- and under-predicted values. We have also indicated the minimum and maximum values obtained over all 40 platforms.

Figure 2 confirms the results of the previous section: the improved trees have very good predictive power, especially ImpTreeBW, with an average error of 3% on the most difficult application, namely All2All. The results of Clique would be very good too. But as it fails to take interferences into account, it fails to accurately predict the running time of all2all. (Note that the fact that Clique over-estimates the bandwidth for a few pairs of hosts is due to routing asymmetry in the original platform.) We can also see that the basic spanning trees have better results than in the previous experiment. This is due to the fact that GridG platforms contain parts that are very tree-like, which these algorithms are able to reconstruct easily.

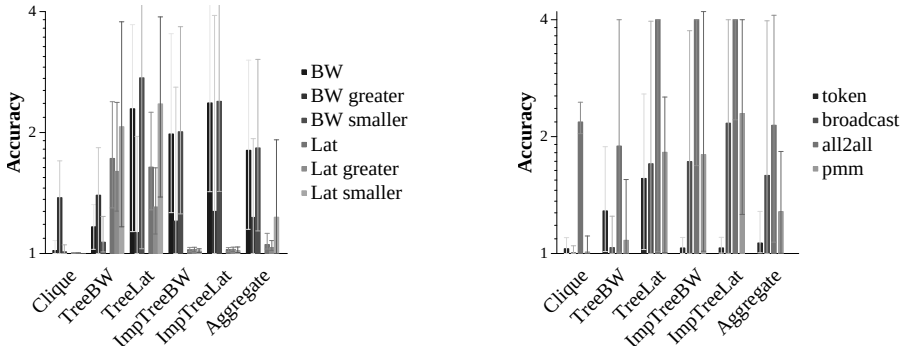
However, Figure 3 shows that platforms with hidden routers are much more difficult to reconstruct. The performance of the clique platform remains the same as before, but all other algorithms suffer from a severe degradation. It is not clear yet whether this degradation comes from a wrong view of the topology of the platform, or from the wrong bandwidth predictions which we can see on Figure 3a.



(a) End-to-end metrics

(b) Applicative metrics

Fig. 2: Simulated tests on the GridG platforms, with processes on every host.



(a) End-to-end metrics (b) Applicative metrics

Fig. 3: Simulated tests on the GridG platforms, with hidden routers

6 Conclusion

In this work, we proposed two new reconstruction algorithms and we compared them with classical reconstruction algorithms (namely spanning trees and cliques) through a thorough evaluation framework. This evaluation framework and the evaluated algorithms are part of the ALNeM project, an application-level measurement and reconstruction infrastructure, which is freely available[6].

We showed that our Improving procedure, when applied to the maximal spanning tree on bandwidth, performs very well on instances without internal routers. The particular efficiency of this algorithm may be explained by the fact that this is the only algorithm which builds a non trivial graph (i.e., not a clique) while using both the information on latencies and bandwidths. As a future work, we should design an algorithm which uses the two types of information simultaneously when building a model, rather than using one type of information after the other, as is done to obtain our ImpTreeBW models.

None of the studied algorithms is fully satisfying in a Grid context, with hidden internal routing nodes. Our future work is thus to extend the algorithms to enable them to cope with such a situation. So far, no algorithm is using any information on interferences. This should also be addressed as this information should enable us to design even more efficient network model building tools.

References

1. L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, 1997.
2. H. Burch, B. Cheswick, and A. Wool. Internet mapping project. http://www.lumeta.com/mapping.html.

[6] http://gforge.inria.fr/plugins/scmcvs/cvsweb.php/contrib/ALNeM/?cvsroot=simgrid

3. J. W. Byers, A. Bestavros, and K. Harfoush. Inference and labeling of metric-induced network topologies. *IEEE TPDS*, 16(11):1053 – 1065, 2005.
4. The cooperative association for internet data analysis. http://www.caida.org/.
5. E. Caron and F. Desprez. DIET: A scalable toolbox to build network enabled servers on the grid. *IJHPCA*, 20(3):335–352, 2006.
6. H. Casanova. Modeling large-scale platforms for the analysis and the simulation of scheduling strategies. In *IPDPS*, Apr. 2004.
7. P.-K. Chouhan, H. Dail, E. Caron, and F. Vivien. Automatic middleware deployment planning on clusters. *IJHPCA*, 20(4):517–530, Nov. 2006.
8. M. den Burger, T. Kielmann, and H. E. Bal. TOPOMON: A monitoring tool for grid network topology. In *ICCS 2002*, volume 2330 of *LNCS*, pages 558–567, 2002.
9. P. Dinda, T. Gross, R. Karrer, B. Lowekamp, N. Miller, P. Steenkiste, and D. Sutherland. The architecture of the remos system. In *HPDC-10*, 2001.
10. A. B. Downey. Using pathchar to estimate internet link characteristics. In *Measurement and Modeling of Computer Systems*, pages 222–223, 1999.
11. L. Eyraud and M. Quinson. Assessing the quality of automatically built network representations. In *1st Workshop on Programming Models for Grid Computing (CCGrid 2007)*, 2007. To appear.
12. I. Foster. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.
13. I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applications*, 11(2):115–128, 1997.
14. 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.
15. T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang. MagPIe: MPI's collective communication operations for clustered wide area systems. *ACM SIGPLAN Notices*, 34(8):131–140, 1999.
16. A. Legrand, M. Quinson, K. Fujiwara, and H. Casanova. The SimGrid project - simulation and deployment of distributed applications. In *HPDC-15*. IEEE Computer Society Press, 2006.
17. A. Legrand, H. Renard, Y. Robert, and F. Vivien. Mapping and load-balancing iterative computations on heterogeneous clusters with shared links. *IEEE Trans. Parallel Distributed Systems*, 15(6):546–558, 2004.
18. B. Lowekamp and A. Beguelin. ECO: Efficient collective operations for communication on heterogeneous networks. In *IPDPS'96*, 1999.
19. D. Lu and P. Dinda. Synthesizing realistic computational grids. In *Proceedings of ACM/IEEE Supercomputing 2003 (SC 2003)*, Nov. 2003.
20. T. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM*, volume 1, pages 170– 179, 2002.
21. M. Quinson. GRAS: A research & development framework for grid and P2P infrastructures. In *PDCS 2006*, 2006.
22. G. Shao, F. Berman, and R. Wolski. Using effective network views to promote distributed application performance. In *PDPTA*, June 1999.
23. D. Tsafrir, Y. Etsion, and D. G. Feitelson. Backfilling using system-generated predictions rather than user runtime estimates. In *IEEE TPDS*. to appear.
24. 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.