Monitoring, Aggregation and Filtering for Efficient Management of Virtual Networks

Stuart Clayman, Richard Clegg, Lefteris Mamatas[†], George Pavlou and Alex Galis
Dept of Electronic Engineering, University College London, London, UK
Email: sclayman@ee.ucl.ac.uk, richard@richardclegg.org, gpavlou@ee.ucl.ac.uk, agalis@ee.ucl.ac.uk
[†] Space Internetworking Center, Dept. of Electrical and Computer Engineering,
Democritus University of Thrace, Xanthi, Greece
Email: emamatas@spice-center.org

***Abstract*—Virtual Networks are characterised as highly dynamic network environments, where topologies and nodes adapt rapidly to changes in user and service demands, user location and context changes, or resource constraints. This paper presents a rigorous assessment of an Information Management Overlay (IMO) suitable for monitoring virtual networks. The IMO system allows complex monitoring tasks to be performed in a scalable manner in order that the network manager can monitor the desired network properties in a way which is accurate, scalable, and low-bandwidth. The monitoring architecture is decentralised, not resource intensive (in terms of memory or CPU), and adaptable to a wide range of virtual topologies.**

The IMO architecture uses Information Aggregation Points and Information Collection Points to scalably aggregate, filter, and collect data. The IMO is tested for scalability in both simulation (with over 35,000 thousand nodes) and on a custom-made virtual network testbed (with over 700 virtual routers), where we show that it performs well at a variety of monitoring tasks. This paper shows that the system is scalable and that intelligent configuration of the monitoring system greatly improves its efficiency. Furthermore, filtering can help reduce the monitoring load, however, the improvement brought in by filtering is greatly dependent on the nature of the monitoring task.

I. Introduction

In order to manage the challenging and dynamic infrastructures of virtual networks there needs to be a monitoring system which can collect and report on the behaviour of the resources, combined with a management system that can use the monitoring information in order to make decisions regarding network behaviour. In this paper we present an information management platform, called the Information Management Overlay (IMO), which is an infrastructure that regulates information flow based on the current state and topology of the network environment. In particular, we address the problem of implementing and testing an Information Management Overlay (IMO) for dynamic virtual networks (or any other dynamic types of networks)

The IMO consists is a decentralised router-level monitoring system which collects data from nodes in a network, a system of aggregation points used to gather such data, and an IMO controller which controls data collection, data aggregation, and aggregation node placement. The testing of this infrastructure takes place both on a dedicated testbed for virtual networks (where the monitoring system runs on a network with over 700 virtual routers running on eleven physical machines) and in simulation (where topologies with over 35,000 simulated nodes have been tested). The virtual routers were designed and built for this research. Extensive evaluation of the monitoring system shows that (i) the a specially devised node placement algorithm is efficient, (ii) the monitoring system scales very well as the network grows and (iii) the filtering system for aggregation points can significantly reduce monitoring traffic, however, the reduction depends on the nature of the metric to be monitored.

The IMO is an important step towards a unified information management infrastructure for virtual networks. The work presented here is a rigorous and detailed test of the Information Management Overlay introduced in [1]. The actual monitoring software used on each virtual router is known as Lattice and was introduced in [2] and [3]. This paper reports on a successful large scale testbed deployment of the monitoring software with IMO control algorithms and a large-scale simulation test of this control.

In [1], a node placement algorithm known as *HotSpot* was tested on a virtual network testbed using the Xen hypervisor [4]. It used a static topology, with 33 virtual machines over three physical machines. This paper takes a different approach by using a lightweight virtual network testbed based on Java Virtual Machines, which can run either virtual routers or virtual service elements. This is complemented with a much more lightweight simulation. This allows for much larger scale tests – in the testbed, the 33 routers from [1] was improved to over 700 routers and in simulation 35,000 nodes. The *HotSpot* placement algorithm from [1] is replaced by a novel placement algorithm for dynamic networks, called *Pressure* which greatly improves upon *HotSpot*. In contrast to [1] this paper considers dynamic topologies and this is an important improvement over the static tests. A real life virtual network is highly dynamic.

The structure of the paper is as follows. Section II describes the research background regarding the problem addressed. Section III describes the proposed monitoring framework. Section IV describes how the framework was implemented and tested in simulation and in a real-life virtual network testbed. Section V describes tests of node placement algorithms. Section V describes scalability tests. Section V describes tests of information filtering. Finally, section VI concludes this paper.

II. Background

Virtual networks aim at better utilisation of the underlying infrastructure [5] in terms of (i) reusing a single physical or logical resource for multiple other network instances, or (ii) to aggregate multiples of these resources, in order to obtain more *functionality*, such as providing a pool of resources that can be utilised on demand. As an example, virtual networks can be aggregated (or federated) together. Such an approach requires aggregation and dissolution of control, data, and information planes, which is a challenging problem. *Manageability* is considered to be the biggest concern for network virtualisation [6].

Management applications need to be adaptive to a rapidly changing environment with respect to specific network properties, and to service or user requirements, as examples. This implies that management applications should be supported by a platform that collects, processes, and disseminates information characterising the virtual network. Monitoring is a main management problem in the network virtualization environment [6].

Key design requirements of an information management infrastructure are: (i) information collection from the sources (e.g., virtual routers), (ii) information processing that produces different information abstractions, and (iii) information dissemination to the management entities exploiting that information. It is common that such design approaches aggregate information using aggregate functions. In [7], the authors employ a QoS data aggregation/refinement technique in order to ensure end-to-end service quality stated in service level agreements (SLAs). Other approaches addressing the fundamental trade-off between information accuracy and management overhead using information aggregation include [8] and [9]. Consequently, real-time monitoring of network parameters may introduce significant communication overhead, especially for the root-level nodes of the aggregation trees. Information flow should adapt to both the information management requirements and the constraints of the network environment, one example being: changes in the information collection configuration.

Existing proposals on monitoring virtual networks primarily cover experimental infrastructure focused on monitoring or explore particular monitoring aspects. For example, Planetlab [10] uses resource monitoring through a resource broker, realizing admission control. In [11], the authors propose monitoring over virtual networks as an enabler technology for accountability. Their focus is mainly on secure information probing mechanisms. In Cabo [12], topology discovery is realized through a discovery plane that uses flooding for dissemination of reachability information. In [13], the authors describe a solution for monitoring information representation through MIBlets, i.e., logical structures providing abstract and selective views of the physical network resources allocated to virtual networks. Virtual Internet infrastructure [14] supports dynamic discovery, deployment, and monitoring. Multiple overlays coexist, supporting control and network recursion. In [15], the authors introduced a common abstraction layer for all the management software in order to be able to aggregate information from diverse, even conflicting management paradigms, followed by different infrastructure providers. Other proposals target at specific virtual network environments, e.g., VNRM [16] for ATM networks.

In [17], the authors introduce performance metrics and an evaluation methodology for assessing performance of network and service monitoring frameworks. They studied the effect of load and number of managed nodes on scalability as well as the impact of management activities on the user-perceived performance of the framework. The same authors provide a queuing theoretical model of commonly used monitoring frameworks that explores scalability limits of such environments [18]. In this paper, we also consider scalability as a critical aspect. In [19], the authors evaluate different placement algorithms in the context of Content Distribution Networks (CDNs). In [20], the authors investigate the performance of topology-informed replicated server placement.

Existing monitoring systems such as Ganglia [21], Nagios [22], MonaLisa [23], R-GMA [24] and GridICE [25] have addressed the monitoring of large distributed systems. They are designed for fixed, and relatively slowly changing physical infrastructure that includes servers, services on those servers, routers and switches. However, they have not addressed or assumed a rapidly changing and dynamic infrastructure as present in virtual environments. In the physical world, new nodes/routers do not appear or disappear very often. Sometimes some new routers and servers are purchased and added to a rack, and a router or server may fail. Also, it is rare that a server will move from one location to another. In the virtual world, the opposite is the case. Many new virtual routers and virtual hosts can appear and disappear rapidly, often within minutes. Furthermore, the virtual hosts can be migrated from one network to another while still retaining their capabilities. Approaches, such as [26], perform monitoring over virtual networks, at the service level, without addressing the constraints of the virtual network environment.

III. Information Management Overlay

The Information Management Overlay (IMO) described here is a fundamental component of an Autonomic Network architecture [27]. The IMO can be seen as an implementation of a Knowledge Plane (such as the Knowledge Plane proposed in [28]) for a network which provides an interface for management applications to gain information about a network. The aim of the IMO is to allow the efficient and scalable collection of data about a running network. In this paper, we present an IMO which is suitable for virtual networks. In a virtual network, the topology may change dynamically as new nodes may be added to the network at any time.

The IMO is a management infrastructure that collects, processes, and disseminates information from/to the network entities and management applications, acting as an enabler for autonomic self-management functionality. It consists of the IMO Controller and a number of IMO-nodes placed at different points in the network, forming a hierarchy. A Management Application can interact with the IMO Controller (see Figure 1) by specifying its requirements in terms of the

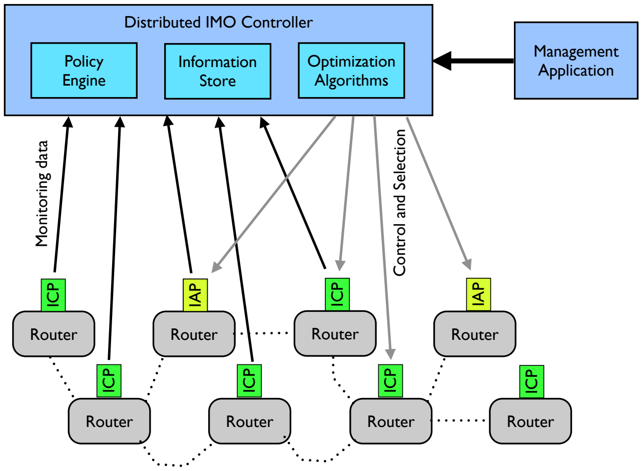

Fig. 1. Information Management Overlay (IMO) Controller and Routers

information sources, the type of information, the monitoring rate, the aggregation function, etc. Furthermore, it can specify performance optimisation requirements with respect to network overhead or processing cost. The IMO Controller performs overlay-wide control of the platform and enforces decisions by communicating with the appropriate IMO-nodes, in order to satisfy the above requirements.

The key factors that allow the IMO to be scalable, efficient, and robust are the quantity and the placement of the IMO-nodes. Since the latter are a subset of the network nodes, each topology should be carefully analysed, based on algorithms that enable the optimum deployment of the IMO-nodes in an automated manner.

The IMO-nodes themselves may have one of the following roles: (i) to collect information, acting as Information Collection Points (ICPs), (ii) to aggregate information, acting as Information Aggregation Points (IAPs), or (iii) both, acting as Information Collection & Aggregation Points (ICAPs). Our infrastructure supports both optimum placement of IMO-nodes and information filtering based on accuracy objectives, in order to adjust the performance-related trade-offs.

Information Collection Points (ICPs) originate the data to be monitored. They are either the machines where that data is generated or monitoring points associated with a network of machines and collecting the data for that network. The ICPs are built as a Lattice Data Source and are controlled by the IMO. The IMO may select the nature of the data to be collected, the frequency of data collection, and various filtering options.

Information Aggregation Points (IAPs) aggregate the data from the ICPs. The IMO configures each ICP to send its data periodically to one IAP (an IAP may also be an ICP, which is an instance of an ICAP, in which case it keeps the data). The data sent to the IAPs may be filtered or combined using various statistical functions. The IAPs are built as a Lattice Data Consumer.

The IMO Controller is responsible for the setup and optimisation of the overlay. It takes input from a management application regarding the optimisation requirements, such as the percentage of nodes that are to be IAPs or the filtering that is required at ICPs. It then configures the ICPs and the IAPs via their respective controllers. The IMO Controller has many components, but its 3 main components are: the optimisation algorithms, the information store, and the policy engine. These are the core part of the IMO management and decision making machinery. These components and well as the ICP and IAP are described in more detail in [1].

IV. Testbed framework

This section describes the testbed and methods used to test the IMO system. The experimental framework described here uses a common experimental harness to drive both testbed run results and simulation results. The test runs are replicated several times to ensure replicability of results.

The testbed consists of a large number of software routers running inside Java Virtual Machines (JVMs) across a smaller number of physical host machines. The routers are logically independent software entities, which communicate with each other via network interfaces. The virtual router was developed by the authors to implement much of the IP stack in an efficient way. Each physical machine can run as many as 70 of these virtual routers. The testbed executes on eleven physical machines running the elements of the IMO.

The underlying monitoring framework used by the testbed on the routers, known as Lattice, is described in [2] and [3]. The Lattice monitoring system has been used successfully to provide data on all of the virtual elements and the running services of a cloud computing service environment [2] as well as for virtual networks [3]. The measurements supplied have been used for service and network management.

The virtual routers used in the testbed are relatively complex software entities. They can be started and stopped in less than a second. They can be linked or unlinked to create quickly changing network topologies. The routers use distance vector routing with the split-horizon hack and poisoned reverse attempting to mitigate the count-to-infinity problem. In addition, routed packets have a time-to-live (to ensure they do not get stuck in routing loops). The routers, therefore, pass data hop-by-hop over the virtual network.

Each virtual router also implements virtual ports and networked applications can be run on a datagram socket interface over these virtual ports. The Lattice monitoring software is run over these virtual ports. In this system, every virtual router is an ICP and a selected subset are IAP (and the ICP and IAP code is run as a Lattice application on every virtual router). Each ICP is informed which IAP it should pass data to and the monitoring data is routed hop-by-hop like any other data.

The results given in this paper, whether from simulation or from the testbed, have a common structure to aid comparability. Each “experiment” lasts one hour, results are collected every ten seconds, and the last fifty of these are averaged. Each point on the graph is then averaged over five experiment replications. The mean is plotted and error bars of plus and minus one standard deviation. Each point on every testbed graph therefore represents five hours of testbed time (simulations are, naturally, not done in real time). For simulation the traffic on the network is estimated using equation (1) whereas in the testbed setting the generated traffic is measured.

Because the final deployed nature of virtual networks remains quite uncertain, the testbed and simulation here rely on the input of arbitrary probability distributions to model node arrival rates and how nodes link together. In the experiments performed here, nodes arrive as a Poisson process (exponential distribution of inter-arrival times) as this has been shown to be a realistic distribution for a number of real traffic arrival processes on the current Internet [29, table 3]. Links are added between nodes as a random process, with every new node having one link plus a number of extra links, with a Poisson distribution. The nodes are linked at random (so older nodes will tend to acquire more links).

In the testbed framework the traffic generated by monitoring can be *directly* measured. This is the traffic between all the ICPs and their associated IAP generated by the monitoring data. For the simulation, this traffic level must be estimated, and is defined here.

Let $N(t)$ be the set of nodes in the network at time t and let $D_j(t)$ be the distance (number of hops) from node j to its nearest aggregation point at time t. An estimate for a normalised traffic level $T(t)$ is then given by

$$T(t) = \sum_{j \in N(t)} D_j(t). \tag{1}$$

The *Pressure* algorithm is now defined. Define the "pressure score" for a node j, $P_j(t)$ as

$$P_j(t) = \sum_{i \in N(t)} \max(0, D_i(t) - d_{i,j}),$$

where $d_{i,j}$ is the distance (in hops) from node i to node j. This can be calculated in a decentralised way by a node asking its neighbours for their distance and value of $D_i(t)$ and this query being passed on recursively. It can be shown that choosing the node with the maximum pressure score $P_j(t)$ is a locally-optimal, greedy algorithm for reducing the traffic as defined in (1). That is to say, the node with the highest $P_j(t)$ is the best possible choice of a single node to become an IAP if the goal is reduction of traffic reduction. (The choice is only locally optimal as it takes no account of future network evolution or future choices of IAP).

For comparison, an algorithm from our previous work [1] called HotSpot is used. The HotSpot score for a node is

$$H_j = d(1)_j^3 + d(2)_j^2 + d(3)_j,$$

where $d(i)_j$ is the number of neighbours at i hops node j has. This algorithm is used for comparison in addition to the baseline *Random* algorithm which simply picks any non-IAP node with equal likelihood.

In every experiment, a minimum proportion of nodes to be IAPs is selected. This is monitored every ten seconds. If the proportion of IAPs is below this, then a new IAP is selected according to either *Pressure*, *HotSpot*, or *Random*. ICPs update their preferred IAP when new IAPs are added or when topologies change.

In a large network it is envisaged that the IMO controller selecting the IAP would be decentralised. The calculation of *HotSpot* and *Pressure* scores do not require global information and could be achieved by the nodes successively querying their neighbours. The selection of the node with the highest score to become an IAP can then be achieved with standard decentralised election algorithms.

V. Results

The first part of this section describes how the traffic changes as the proportion of IAPs changes. The word "nodes" is used here and for the rest of the paper as a generic term incorporating routers in the testbed or simulation nodes. IAP nodes are a subset of these nodes chosen as IAPs. Each experiment run (whether in simulation or on the real testbed) takes a similar form. The network of nodes is grown from an empty network, using parameters as described in section IV. In simulation, new nodes arrive in the network at an average rate of one per second, so that after a one hour long run the network will average 3,600 nodes. On the testbed, new nodes arrive at the slower rate of one per twenty seconds, so that the average size is 180 nodes at the end of an hour.

Figure 2 (top) shows the estimated traffic per node versus the proportion of nodes which are IAPs for the simulation network. Three different node placement algorithms are tried. The traffic is in arbitrary units as calculated from equation (1) and is scaled by the number of nodes in the network. The proportion of nodes which are IAPs is tested with 1%, 2%, 5%, 10% and 20%. As would be expected, the higher the proportion of IAPs the less the estimated traffic per node. Roughly speaking, doubling the proportion of IAPs removes around 0.4 units of traffic per node for each scheme. The effect of changing the assignment scheme remains considerable. As it is expected, the *Pressure* scheme is best, the *HotSpot* scheme comes second, and the *Random* scheme is worst of all. The effect of moving from Random (the least efficient scheme) to *Pressure* (the most efficient) is approximately the same as doubling the proportion of IAPs in the network, which shows that an intelligent choice of IAP selection algorithm is certainly important.

Figure 2 (bottom) shows the same experiment on the testbed with an average of 180 nodes per run. Node proportions of 2% ,5% and 10% are tested (less than that would mean only two IAPs on the testbed one of which is always the first node create). The results are very similar to those from simulation although, due to the smaller network size, the error bars are proportionally larger. The *Pressure* algorithm clearly outperforms *Random* and seems to outperform *HotSpot* (although this could be argued since the results a re within one standard deviation). The advantages are more clear over *HotSpot* as the proportion of IAPs grows (this would be consistent with the idea of the *Pressure* algorithm being better at assigning monitoring nodes since it is used more often when there are more IAPs).

The conclusion of this section, therefore, is that the placement algorithm chosen for placement of IAP nodes for the IMO can make an important difference to the level of monitoring traffic observed. This finding is consistent in both simulation and on the testbed. While the simulation results are clearer (more nodes can be used for testing) the testbed results verify that the benefits do occur on a real system.

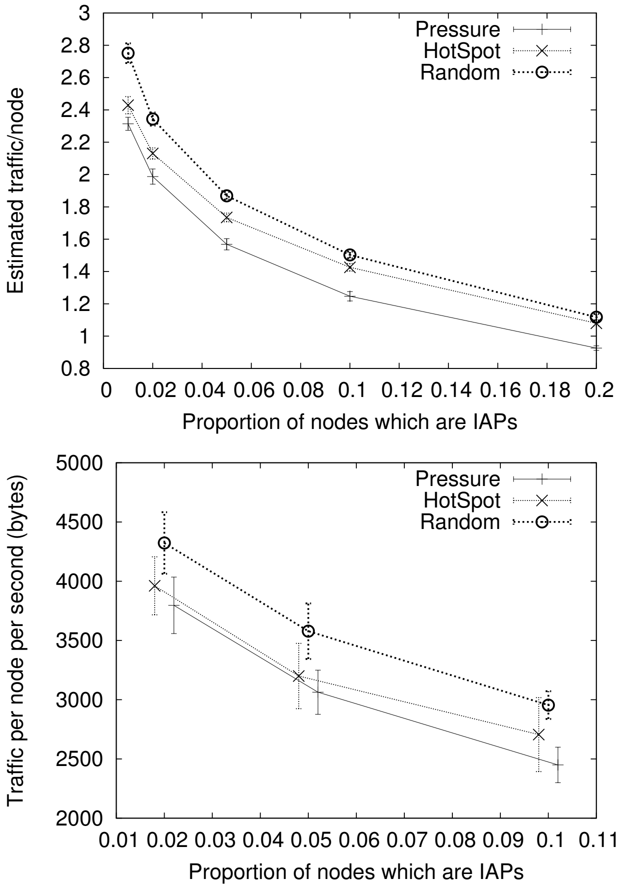

Fig. 2. Proportion of nodes as IAPs vs estimated traffic: Top (simulation). Bottom (testbed).

The second part of these results describe how the monitoring framework scales as the number of nodes increases. Figure 3 (top) shows the estimated traffic per node versus the number of nodes for the three placement algorithms in simulation. Node arrival rates are varied and five runs of each size are tried. The network size varies from an average of 36 nodes per run to an average of 36,000 nodes per run. It is important to note that the x-axis is plotted on a logscale. The number of nodes varies from around forty to around seven thousand. As can be seen, in common with the results from the previous section, the Pressure algorithm performs best, the *HotSpot* algorithm second best and the Random algorithm worst of all. More importantly, the scaling behaviour of all algorithms is acceptable and of the Pressure algorithm, very much so. For smaller experiments, the error bars (one standard deviation above or below the mean for five experimental runs) are large. This is to be expected since with only 40 nodes the placement of four IAPs could greatly vary the results. Also, if a particular experiment happened, by chance, to have 41 nodes (and hence five IAPs) the difference in traffic levels would be large. The most important conclusion is that the increase in traffic as the network increases in size is small (especially for the Pressure algorithm).

Figure 3 (bottom) shows the scaling results from the testbed using 10% of nodes as IAPs and using the same three placement algorithms. This confirms the good scaling behaviour of both the *Pressure* and *HotSpot* algorithms, and shows that the traffic does not grow greatly as the network size increases. Obviously, the testbed cannot run as many nodes as the simulation, the mean arrival rate is varied up to one router every five simulated seconds, giving the mean size of the final network as 720 nodes. As expected, *Pressure* is better than *HotSpot*, which is in turn better than *Random* (although, again, this conclusion should be treated with caution due to the size of the error bars in smaller network sizes). Note that the variation in number of nodes between the algorithms is due to statistical variation between runs.

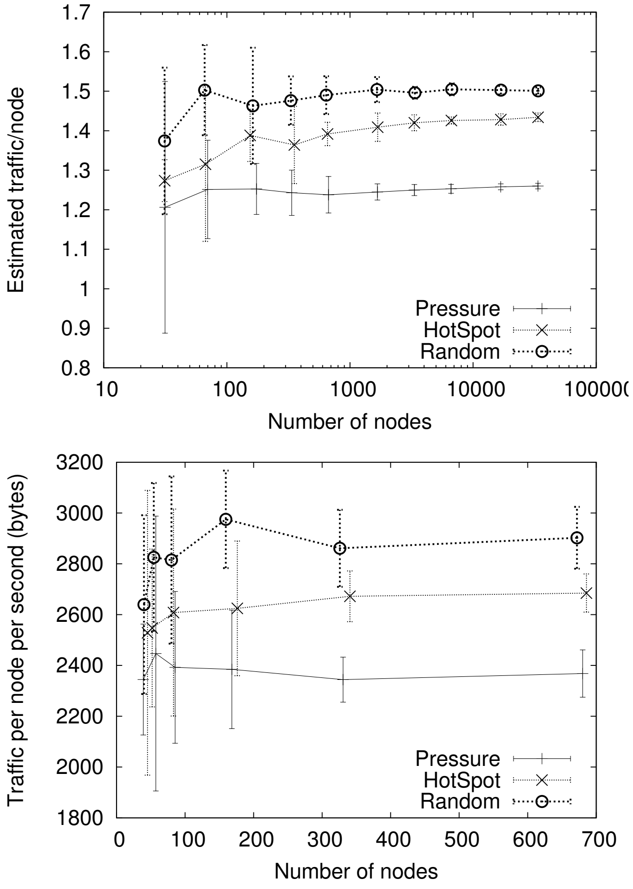

Fig. 3. Traffic per node vs no. of nodes: Top (simulation). Bottom (testbed).

The conclusions of this section are two fold. Firstly, the *Pressure* algorithm shows the best behaviour regardless of the proportion of IAPs as the number of nodes in the network grows. Secondly, the scaling behaviour of the monitoring system is excellent. The amount of monitoring traffic per node does not increase greatly as the network grows in size (indeed given error bars there's little reason to suppose it increases at all). These are encouraging results in simulation and on the testbed for implementation of this monitoring system in a large network.

The third and final part of these results show the effects of filtering monitoring data at the ICP. Testbed results are shown with parameters as described in section IV and with a mean arrival rate of one node per twenty seconds, giving an average final network size of 180 nodes. As usual, each point on the graph is the average of five hour long experiments.

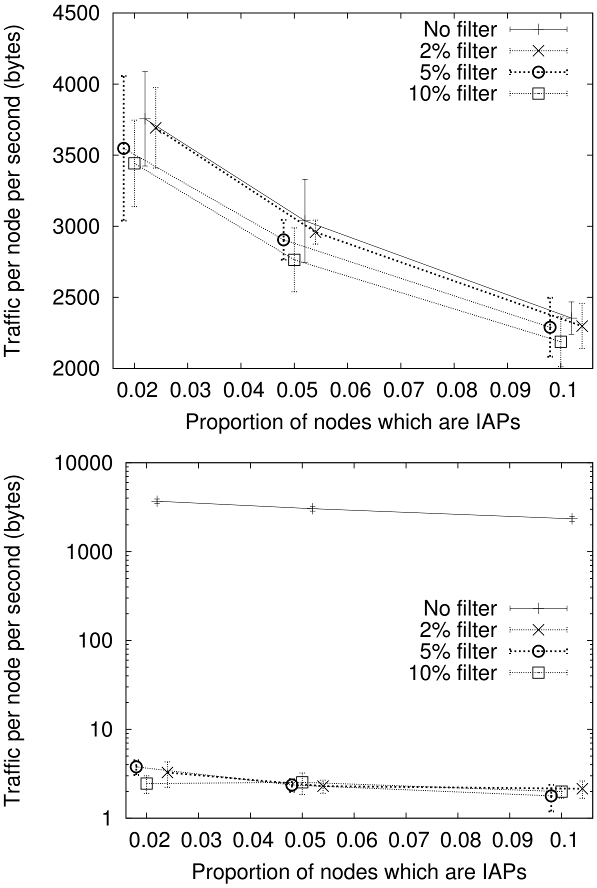

Fig. 4. Effectiveness of filtering: Top (CPU usage). Bottom (memory usage).

Figure 4 (top) shows the results of filtering the traffic so that no monitoring data is sent unless the recorded level varies by 2%, 5% or 10% compared with the last value sent. The CPU utilisation on the host machine is monitored. The proportion of IAPs is varied between 2%, 5% or 10% but, for clarity so that error bars do not overlap the points are shifted slightly to the left and right of their true positions. Only the *Pressure* placement algorithm is tested. As can be seen, the filtering in this case is not greatly effective at reducing traffic levels. Although traffic is reduced, the reduction is relatively minor even with the 10% filter in place. It is likely that this is because the CPU utilisation varies greatly from moment to moment.

Figure 4 (bottom) shows the same results but for monitoring memory usage instead of CPU utilisation. Note the logscale on the x-axis. Due to the nature of the testbed, the memory usage rises throughout the period of the experiment. However, the rise is slow and over short time periods the memory usage remains relatively constant. In this case, therefore, the filtering is extraordinarily effective in reducing monitoring traffic. For example, in the scenario where 2% of nodes are IAPs the traffic is reduced from 3,700 bytes per node per second to only 3.3 bytes per node per second. In the no filter case, monitoring data is sent every second. In the 2% filter case, monitoring data is sent, on average only every 1,100 seconds. There seems to be no significant variability in the results depending on filter level. The degree of difference in filtering effectiveness between the this and the CPU monitoring scenario could not be more marked although the examples were not selected with this in mind. It is clear that this type of filtering can be ineffective or extremely effective at reducing monitoring traffic depending on the nature of the quantity monitored. While this was expected the authors were greatly surprised by the extent to which this was the case.

Clearly, in choosing the filtering level there is a tradeoff between accuracy and traffic reduction. In the case of monitoring CPU utilisation filtering did not greatly reduce traffic and no filtering would be a sensible choice. In the case of monitoring memory load 1% filtering is a sensible choice and 10% filtering loses accuracy without making further traffic reduction. This highlights a need for such filters to be automatically tuned to the nature of the traffic being monitored.

VI. Conclusions

This paper has described the implementation and testing of an Information Management Overlay (IMO) for virtual networks (and other highly dynamic network environments). The system described uses both Information Collection Points (ICPs) and Information Aggregation Points (IAPs) in conjunction with an IMO Controller to provide a scalable solution to monitoring and managing networks where topologies may change quickly.

The tests were carried out in simulation (for over 35,000 nodes) and in a real testbed of virtual routers (for over 700 routers). The results have shown that the proposed monitoring system performed extremely well, and that the choice of placement algorithm is important as intelligent placement of aggregation points can reduce monitoring traffic greatly. The system presented scales extremely well as the amount of monitoring traffic per node on the network did not significantly increase with the size of the network, by several orders of magnitude. The *Pressure* algorithm works better than the older *HotSpot* algorithm at reducing traffic and both are better than *Random* as would b expected. Finally, filtering the monitored traffic was shown to be a potentially useful way to reduce the level of monitoring traffic. The success of the filtering in reducing traffic depends greatly on the nature of the entity to be filtered and what a management system, which is to filter intelligently, must do with regard to the nature of the entity being monitored.

A future direction for this work would be automating the filtering process. It is clear that for the memory monitoring, 2% filtering is good for reducing traffic but for the CPU monitoring the reduction in accuracy is unlikely to be worth the trade-off. An intelligent automatic system could detect the appropriate level of filtering based upon a management input describing the relative importance of information accuracy objective [30] and traffic reduction.

Acknowledgements

This work is partially supported by the European Union UniverSELF [31] project of the 7th Framework Program.

References

[1] L. Mamatas, S. Clayman, M. Charalambides, A. Galis, and G. Pavlou, “Towards an information management overlay for emerging networks,” in *NOMS 2010*, 2010.

[2] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero-Merino, L. Vaquero, K. Nagin, and B. Rochwerger, “Monitoring service clouds in the future internet,” in *Towards the Future Internet - Emerging Trends from European Research*. IOS Press, April 2010.

[3] S. Clayman, A. Galis, and L. Mamatas, “Monitoring virtual networks with lattice,” in *Management of Future Internet - ManFI 2010*, 2010. [Online]. Available: http://www.manfi.org/2010/

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand *et al.*, “Xen and the art of virtualization,” *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.

[5] T. Anderson, L. Peterson, S. Shenker, and J. Turner, “Overcoming the internet impasse through virtualization,” *Computer*, vol. 38, pp. 34–41, April 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1058219.1058273

[6] N. M. K. Chowdhury and R. Boutaba, “A survey of network virtualization,” *Comput. Netw.*, vol. 54, pp. 862–876, April 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.10.017

[7] Y. Lin and M. Chan, “A scalable monitoring approach based on aggregation and refinement,” *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 4, pp. 677–690, 2002.

[8] A. Prieto and R. Stadler, “A-gap: An adaptive protocol for continuous network monitoring with accuracy objectives,” *IEEE Transactions on Network and Service Management (TNSM)*, vol. 4, no. 1, pp. 2–12, 2007.

[9] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi, “Holistic aggregates in a networked world: Distributed tracking of approximate quantiles,” in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 25–36.

[10] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, “PlanetLab: an overlay testbed for broad-coverage services,” *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[11] E. Keller, R. B. Lee, and J. Rexford, “Accountability in hosted virtual networks,” in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, ser. VISA ’09. New York, NY, USA: ACM, 2009, pp. 29–36. [Online]. Available: http://doi.acm.org/10.1145/1592648.1592654

[12] N. Feamster, L. Gao, and J. Rexford, “How to lease the internet in your spare time,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 61–64, January 2007. [Online]. Available: http://doi.acm.org/10.1145/1198255.1198265

[13] W. Ng, D. Jun, H. Chow, R. Boutaba, and A. Leon-Garcia, “MIBlets: A practical approach to virtual network management,” in *Integrated Network Management, 1999. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on*. IEEE, 1999, pp. 201–215.

[14] J. Touch, Y. Wang, L. Eggert, and G. Finn, “A virtual internet architecture,” *ISI Technical Report ISI-TR-2003-570*, 2003.

[15] M. Feridan, M. Moser, and A. Tanner, “Building an abstraction layer for management systems integration,” in *Proceedings of the 1st IEEE/IFIP International Workshop on End-to-End Virtualization and Grid Management (EVGM2007)*, 2007, pp. 57–60.

[16] W. Ng, R. Boutaba, and A. Leon-Garcia, “Provision and customization of ATM virtual networks for supporting IP services,” in *ATM Workshop, 1999. IEEE Proceedings*. IEEE, 1999, pp. 205–210.

[17] A. Lahmadi, L. Andrey, and O. Festorh, “Performance of network and service monitoring frameworks,” *11th IFIP/IEEE International Symposium on Integrated Network Management*, 2009.

[18] A. Lahmadi, L. Andrey, and O. Festor, “Design and validation of an analytical model to evaluate monitoring frameworks limits,” *Eighth International Conference on Networks*, 2009.

[19] L. Qiu, V. Padmanabhan, and G. Voelker, “On the placement of web server replicas,” in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2001, pp. 1587–1596.

[20] P. Radoslavov, R. Govindan, and D. Estrin, “Topology-informed internet replica placement* 1,” *Computer Communications*, vol. 25, no. 4, pp. 384–392, 2002.

[21] M. L. Massie, B. N. Chun, and D. E. Culler, “The ganglia distributed monitoring system: Design, implementation and experience,” *Parallel Computing*, vol. 30, p. 2004, 2003.

[22] “Nagios,” http://www.nagios.org/.

[23] H. Newman, I. Legrand, P. Galvez, R. Voicu, and C. Cirstoiu, “MonALISA : A distributed monitoring service architecture,” in *Proceedings of CHEP03, La Jolla, California*, 2003.

[24] A. Cooke, A. J. G. Gray, L. Ma, W. Nutt *et al.*, “R-GMA: An information integration system for grid monitoring,” in *Proceedings of the 11th International Conference on Cooperative Information Systems*, 2003, pp. 462–481.

[25] S. Andreozzi, N. De Bortoli, S. Fantinel, A. Ghiselli, G. L. Rubini, G. Tortone, and M. C. Vistoli, “GridICE: A monitoring service for grid systems,” *Future Gener. Comput. Syst.*, vol. 21, no. 4, pp. 559–571, 2005.

[26] J. Liang, X. Gu, and K. Nahrstedt, “Self-configuring information management for large-scale service overlays,” in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007, pp. 472–480.

[27] A. Galis, S. Denazis, A. Bassi, P. Giacomin *et al.*, *Management Architecture and Systems for Future Internet Networks*. IOS Press, http://www.iospress.nl, ISBN 978-1-60750-007-0, April 2009.

[28] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, “A knowledge plane for the internet,” in *SIGCOMM ’03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 3–10.

[29] R. Clegg, C. Di Cairano-Gilfedder, and S. Zhou, “A critical look at power law modelling of the internet,” *Computer Communications*, vol. 33, no. 3, pp. 259–268, 2010.

[30] A. Prieto and R. Stadler, “Adaptive distributed monitoring with accuracy objectives,” in *Proceedings of the 2006 SIGCOMM workshop on Internet network management*. ACM, 2006, pp. 65–70.

[31] Univerself consortium, “Univerself project,” http://www.univerself-project.eu/.