Towards an Information Management Overlay for the Future Internet

L. Mamatas, S. Clayman, M. Charalambides, A. Galis and G. Pavlou
Department of Electronic & Electrical Engineering
University College London, Torrington Place
London WC1E 7JE, U.K.
{l.mamatas, s.clayman, m.charalambides, a.galis, g.pavlou}@ee.ucl.ac.uk

***Abstract*—There has been recently an increasing research interest in network management infrastructures that autonomously adapt to the dynamics of environment. In this paper, we present an information management platform, called Information Management Overlay (IMO) as an infrastructure that regulates information flow based on the properties and configuration of the network environment. We also discuss the design, implementation and optimization issues of this platform. Furthermore, we explore experimentally how IMO can adapt to different configurations and optimization requirements**

I. Introduction

A key issue in the network management for the Future Internet is the development of a common control space, which has autonomic characteristics and enables heterogeneous network technologies, applications and network elements to interoperate efficiently. The management applications need to be adaptable to the rapidly changing environment, such as the specific network properties, the service and user requirements. This implies that management applications and network entities should be supported by a platform that collects, processes, and disseminates information characterizing the network environment. An increased awareness for the generic and specific properties of the network may bridge the gap between the specific technical details of the network entities and the high-level management goals. Here, we consider an infrastructure that manages information flow and processing within the network as an important stepping-stone towards this vision.

Key design requirements of an information management infrastructure are: (i) the information collection from the information sources (e.g., the network devices), (ii) the information processing that produces different information abstractions, (iii) the information dissemination to the entities that exploit that information, (iv) the information flow optimization with respect to the specific requirements of the underlying network (e.g., topology properties) and the high-level management objectives in the system. The design of such a system is challenging because it should be able to:

- *self-adapt to different configurations and network properties*, e.g., to regulate information flow, information accuracy or monitoring configuration, accordingly.
- *support self-management functionalities*, i.e., to receive decisions autonomously that can be harmonized towards common goals.
- *scale efficiently*, at least, up-to the level of the administrative domains that are exploiting such information.

The above characteristics should be exhibited without human intervention and with minimum management overhead.

It is common that such design approaches (e.g., [1], [2]) have a hierarchical structure and aggregate information using aggregate functions, such as SUM, AVERAGE, MIN or MAX. Consequently, the real-time monitoring of network parameters may introduce significant communication overhead, especially for the root-level nodes of the aggregation trees. We argue that information flow should adapt to the information management requirements and the constraints of the network environment. For example, the location of the aggregation points (APs) significantly impacts the associated overhead. Consequently, the IMO-nodes may dynamically change position in order to adapt to changes in the network environment, e.g., changes in the information collection configuration. Here, we explore the self-adaptation behavior of the IMO infrastructure in different conditions. A special component, the IMO Controller, is responsible for performing adjustments of the information flow whenever the network environment requires it.

In this paper, we pay particular attention to the control requirements of the IMO infrastructure, highlighting that the IMO incorporates policy-based management functionalities in the IMO Controller. We give examples of performance requirements and how local optimization algorithms can be harmonized towards them. For example, a requirement for minimum network overhead may lead to a different aggregation graph from a requirement to minimize CPU utilization. We discuss algorithms that are able to: (i) self-adapt to the environment, and (ii) be harmonized towards common goals. In this context, we evaluate different optimum IMO-node placement algorithms with respect to the optimization requirements.

We have implemented and evaluated the proposed IMO infrastructure in different scenarios using both real experiments and simulations. Our experiments cover different topology scales, different information collection / aggregation configurations, and information flow adjustments. We explore how different performance trade-offs (e.g., processing cost vs communication overhead) can be related to IMO configuration. We show results associating information flow optimization algorithms with example performance requirements. The results presented are very promising and call for further investiga-

tions.

This paper is structured as follows: section II presents the related work; section III details our proposal; section IV presents our experimental results; section V discusses important aspects of our work and presents several open issues; and section VI concludes the paper.

II. Related Work

The Information Management Overlay is a basic component of the autonomic management architectural model proposed in the Autonomic Internet (AutoI) project [3], [4]. The AutoI project [3], [4] aims to develop a self-managing resource overlay that spans across heterogeneous networks and supports service mobility, security, and quality of service. The IMO realizes the core functionalities of the Knowledge Plane in an architecture such as AutoI [4].

The IMO is a focused-functionality Knowledge Plane that supports the management applications with the necessary information towards the realization of self-management capabilities. It is different from the Knowledge Plane proposed in [5], which is a unified solution that includes cognitive techniques and knowledge management. The IMO decouples information management from the other network management functionalities. These other functions are considered to be part of either the Management plane or the Orchestration plane, especially within the AutoI architecture [4].

In [6] the authors present specific implementation details of a knowledge plane architecture that consists of a network knowledge plane (NetKP) in the network layer, and of multiple specialized KPs (spec-KPs) on top of it. Their architecture is built using static agents. In our case, the IMO uses dynamic nodes that may change position dynamically for the purpose of information optimization.

Most autonomic computing platforms are targeted to stable systems with sufficient resources [7], [8]. There are a limited number of approaches that are focused on dynamic environments, such as MANETS [9]. In [9], the authors propose a context-aware system driven by policies that can be tailored towards high-level goals through policy modification. The reconfigurable context-sensitive middleware (RCSM) [10] deals with context awareness in mobile devices assuming reliable underlying ad hoc transport protocols. In [11] the authors introduced a collaborative context determination approach optimized for MANETs. In this approach, a mobile node gains its context information from its neighboring peers. These studies are focusing on specific environments (i.e., infrastructureless).

The Aura project [12] associates accuracy and confidence values to context information and is optimized for ubiquitous environments. Other approaches that address the fundamental trade-off between information accuracy and management overhead include GAP [1], A-GAP [2], and [13], [14]. The information collected is usually aggregated using functions such as SUM, AVERAGE, MIN, and MAX. The GAP [1] and A-GAP [2] protocols are generic aggregation protocols with controllable accuracy. These approaches adjust the level of accuracy based on the constraints of the network environment. For example, resource-limited sensors may trade information accuracy for extended network lifetime. In our proposal, information accuracy is complemented with other parameters that adapt the information flow to the network environment, i.e., the number of APs and the structure of the overlay topology.

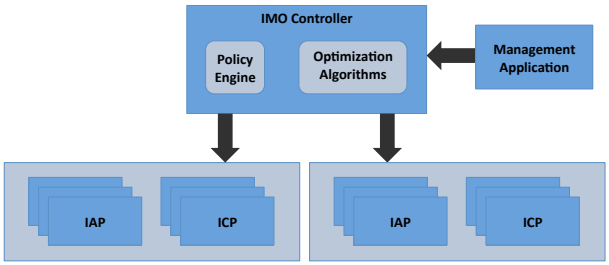

Fig. 1. The IMO Architecture

The design of an autonomic management architecture for the Future Internet needs to overcome many challenging problems that include an efficient representation and derivation of knowledge, the incorporation of cognitive techniques, security issues, information flow optimization, compatibility with legacy hardware and software, resolution of knowledge conflicts, etc. The detailed characteristics of an infrastructure that supports similar architectures with information, with particular focus on information flow optimization, are presented in the next section. Key requirements are the optimization of the information flow and the dynamic placement of IMO-nodes with respect to management overhead and resource utilization.

III. The Information Management Overlay

The Information Management Overlay consists of the IMO Controller and a number of IMO-nodes placed in different points in the network, forming a hierarchy. The Management Application interacts with the IMO Controller and specifies its information requirements (see Fig. 1), e.g., the information sources, the type of information, the monitoring rate, the aggregation function etc. Furthermore, it specifies a perfomance requirement, e.g., to optimize network overhead or processing cost. The IMO Controller performs overlay-wide control of the platform, takes and enforces decisions through communicating with the appropriate IMO-nodes, in order to satisfy the above requirements.

The IMO-nodes 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.

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 IMO-nodes are a subset of the physical nodes, each overlay topology should be carefully created, based on algorithms that enable the optimum deployment of the IMO-nodes in an autonomic way. One key requirement of

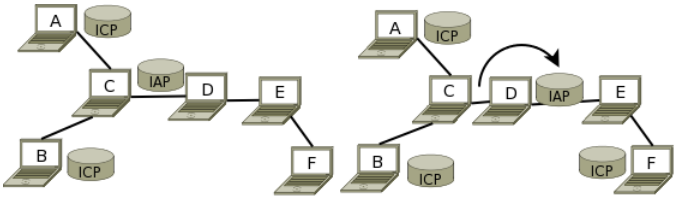

Fig. 2. Autonomic IMO structure adaptation

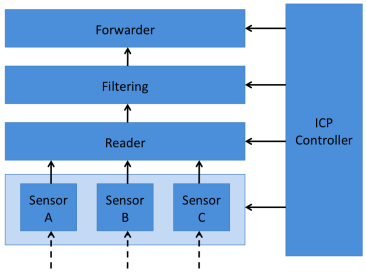

Fig. 3. Structure of Information Collection Points

the architecture is that the overlay topologies should be self-adaptive to the changes in the network environment. As an example of this, an Information Aggregation Point (IAP) may change location if the set of the Information Collection Points (ICP) changes. Figure 2 shows how this self-adpation works. In the left of Fig. 2 we see 6 nodes (A-F) together with 2 ICPs (at node A and at node B) and an IAP at node C. In the right of Fig. 2 a new ICP is deployed to node F. This change to the environment triggers the re-deployment of the IAP to a new position, namely from node C to node D. The traffic generated from the three ICPs will now be balanced.

We detail below the different IMO components, i.e., the Information Collection Points, the Information Aggregation Points and the IMO Controller.

A. The Information Collection Points (ICPs)

The Information Collection Points act as sources of information: they monitor hardware and software for their state, collect configuration parameters, or present capabilities. The IMO supports three types of monitoring queries to an ICP, they are: (i) the 1-time queries which collect information that can be considered static, e.g., the number of CPUs, (ii) the N-time queries which collect information periodically, and (iii) the continuous queries that monitor information in an on-going manner.

The location of the ICPs should be as close as possible to the corresponding real sources of information in order to reduce management overhead. Filtering rules and accuracy objectives should be applied at the ICPs, especially for the N-time and continuous queries, for the same reason. Furthermore, the ICPs should not be many hops away from the corresponding IAPs.

In Fig. 3 we see the structure of an Information Collection Point which we have designed and built for our own Information Management Overlay platform. The ICP consists of 5 mains components, i.e., the *sensors*, a *reader*, a *filter*, a *forwarder* and an *ICP controller*, which are described below.

The *sensors* can retrieve any information required. This can include common operations such as getting the state of a server with its CPU or memory usage, getting the state of a network interface by collecting the number of packets and number of bytes coming in and out, or getting the state of discs on a sytem presenting the total volume, free space, and used space. In our implementation each sensor runs in its own thread allowing each one to collect data at different rates and also having the ability to turn them on and off if they are not needed.

The *reader* collects the raw measurement data from all of the sensors of an ICP. The collection can be done at a regular interval or as an event from the sensor itself. The reader collects data from many sensors and converts the raw data into a common measurement object used in the IMO framework. The format contains meta-data about the sensor and the time of day, and it contains the retrieved data from the sensor.

The *filter* takes measurements from the reader and can filter them out before they are sent on to the forwarder. Using this mechanism it is possible to reduce the volume of measurements from the ICP by only sending values that are significantly different from previous measurements. For example, if a 5% filter is set, then only measurements that differ from the previous measurement by $\pm 5\%$ will be passed on. By using filtering [15] in this way, the ICP produces less load on the network. In our case, the filtering percentage matches the accuracy objective of the management application requesting the information.

The *forwarder* sends the measurements onto the network. The common measurement object is encoded into a network amenable measurement format. The measurements are encoded using XDR [16] as a way to minimize the size of the transmitted data. The XDR format is commonly used in monitoring systems [17] in order to reduce the network loading.

The *ICP Controller* controls and manages the other ICP components. It controls (i) the lifecycle of the sensors, being able to turn them on and off, and to set the rate at which they collect data; (ii) the filtering process, by changing the filter or adapting an existing filter; (iii) the forwarder, by changing the attributes of the network (such as IP address and port) that the ICP is connected to.

For the IMO platform we have developed various sensors which can measure attributes from CPU, memory, and network components of a server host. We can also measure the same attributes of virtualized hosts by interacting with a hypervisor to collect these values. Finally, we have sensors that can send emulated measurements. These are useful for testing and evaluation purposes, with one example being an emulated response time, which we use in our experiments.

B. The Information Aggregation Points (IAPs)

The Information Aggregation Points apply aggregation functions to the collected measurement information. This aggregation process increases the level of information abstraction, thereby transforming the data into a structured form, but at the same time reducing the load on the network. Aggregation works in situations where information consumers

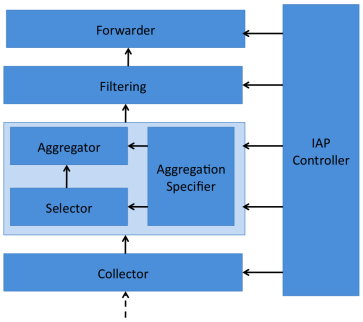

Fig. 4. Structure of Information Aggregation Points

do not need a continuous stream of data from an ICP, but can get by with an approximation of the data. For example, getting an occasional measurement with the average of the volume of traffic on a network link may be enough for some applications. Some common aggregation functions include SUM, AVERAGE, STDDEV, MIN and MAX.

In Fig. 4 we see the structure of an Information Aggregation Point which we have designed and built for our own Information Management Overlay platform. The IAP consists of 7 main components, which include a *collector*, an *aggregation specifier*, a *selector*, an *aggregator*, a *filter*, a *forwarder* and an *IAP Controller*. All of these components are described below.

The *collector* collects measurement data from the network and converts the XDR encoded measurement format into a measurement object. After this the measurement objects are saved in a data store for later aggregation processing. The data store used in the IMO is the Timeindexing Framework [18], [19] which allows any kind of data to be stored and retrieved using timestamps or time intervals.

The Timeindexing Framework provides the mechanism by which arbitrary sequences of measurements can be selected and aggregated. It creates an index into the data, called a timeindex, and provides an API for accessing the data. For example, if the aggregator needs to aggregate the last 30 seconds worth of data it can create a time interval from *now* back 30 seconds, and then ask the timeindex to get that data. This time interval provides an abstract representation of time (including representations for now and back 30 seconds), and does not need to be specified with concrete times.

The *aggregation specifier*, the *selector*, and the *aggregator* are actually combined into an aggregation engine. The *aggregation specifier* specifies *when* the aggregator executes, *what* it aggregates, and *how* it aggregates. These three specifications are similar to those used in SLA compliance systems [20], because the process of analysing the data is similar.

The *when* specification is of the form: wake up every N seconds, which will cause the aggregation engine to wake up regularly to provide an aggregation. The *what* specification takes the form of a time interval, such as “from now, back 30 seconds”. The *how* specification is the name of a function to aggregate the data, such as AVERAGE, SUM, etc.

The *selector* selects the required measurements to aggregate using the *what* specification. It determines what data is eventually chosen by applying the time interval, such as “from now, back 30 seconds”, to the timeindex and selecting the relevant measurements. In this case it will cause the selector to select the most recent 30 seconds worth of data. As the data store uses timeindexing the time interval can be changed arbitrarily. Once the selection is complete the selected data is passed to the aggregator.

The *aggregator* aggregates the selected measurements presented by the *selector*. It uses the how specification to aggregate data. Although it is most common to use aggregation functions, such as SUM, AVERAGE, STDEV, MIN and MAX, the IAP Controller can pass in an arbitrary function into the aggregator in order to do the aggregation. This gives considerable power and flexibility when determining aggregations. Once the aggregation is calculated the aggregated measurement data is passed to the *filter*.

The *filter* takes measurements from the aggregator and can filter them out before they are sent on to the forwarder. Again this reduces the volume of measurements by only sending values that are significantly different from previous measurements. Using filtering in this way in the IAP, like filtering in the the ICP, produces less load on the network.

The *forwarder* sends the aggregated measurements onto the network. The common measurement object is encoded into the same network amenable format as in the ICP using XDR. By having the same network format, the consumers of the measurement data do not need to know if data has come directly from and ICP or has come from an IAP. This allows hierarchies of elements to be composed as an IAP can further aggregate data from other IAPs if this is required.

The *IAP Controller* controls and manages the other IAP components. It controls (i) the *collector*, by changing the attributes of the network (such as IP address and port) that the IAP listens to, (ii) the aggregation process, by managing the *aggregation engine* and by passing in the *aggregation specifier*, (iii) the filtering process, by changing the *filter* or adapting an existing *filter*, (iv) the *forwarder*, by changing the attributes of the network (such as IP address and port) that the IAP sends to.

The aggregation engine itself is flexibile enough to be given different aggregation specifications by the IAP Controller in order to process the data in varying way. For example, it can be told to wake up once an hour and select data for the last day, and then apply an aggregation function. This is achieved using a mechanism that relys on plugins. These plugins represent code blocks which can be pre-defined, such as an average aggregator, or can be defined to suit the need.

Information dissemination is done through the Information Aggregation Points. As an application may request a specific piece of information from an IAP, the deployment location of the IAPs should also consider the locations and the traffic requirements of the nodes retrieving information, the Information Consumers. As well as requesting information, an application has the option to subscribe to an event-based notification service by setting an appropriate threshold to a specific type of information. Whenever this threshold is exceeded, the application is notified.

C. IMO Controller

The IMO Controller has centralized functionality and is responsible for the setup and optimization of the overlay. As shown in Fig. 1, it takes input from a management application regarding optimization requirements (e.g. cpu/memory, network resources, or response time) and then configures the ICPs and the IAPs via their respective controllers. The functionality of the IMO Controller is realized by two components: (a) the policy engine, and (b) the optimization algorithms. The optimization algorithms can potentially implement a variety of optimization tasks that involve performance related tradeoffs, but in its current form it focuses on algorithms for the placement of aggregation points. The policy engine provides the necessary logic for the optimal configuration of the overlay according to the optimization requirements and the properties of the environment. These two components are discussed in more detail in the following sections.

1) Optimization Algorithms: As mentioned above, the IMO Controller can implement various optimization algorithms. Examples include optimizing with respect to the network protocol deployed (e.g. multicast for a large number of receivers), or, trading processing for communication cost by using compression techniques. The optimization considered in this work however, concerns the placement of the aggregation points in the network. This is a process which is carried out when the overlay is initially deployed, but can also be triggered at run-time to react to changes in the network (as shown in Fig. 2) and to react to emerging requirements from the management application.

We present below different node placement algorithms, namely the: Random, Hotspot and two variations of the Greedy Algorithm. Similar algorithms have been proposed in the area of Content Distribution Networks [21].

Random Algorithm

The Random algorithm randomly chooses k IMO-nodes from the set of N available nodes, based on a uniform distribution. These k nodes become the IMO-nodes. It is a very simple algorithm and thus produces insignificant processing overhead. It is suitable for performance requirements to optimize resource consumption in terms of CPU or Memory utilization..

Hotspot Algorithm

The Hotspot algorithm places the k IMO-nodes near the nodes that have more chances to produce significant communication overhead. A node is considered as a "hotspot" when it is connected to a high number of nodes. In practice, node density is the important factor. For each potential IMO-node, a cost function F, that quantifies node density is calculated. The cost function is elaborated as follows:

$$F(N_i) = a^3 + b^2 + c^1 \quad (1)$$

where a, b, c are the number of nodes that have one hop, two hops and three hops distance from the node N_i, respectively. We note that in certain environments (e.g., MANETs) the number of hops may not be the only parameter that expresses the network distance between two nodes. For example, a similar cost function may also include parameters such as network latency, available energy, free memory, CPU power, etc. The hotspot algorithm is suitable for performance requirements that focus on the network overhead optimization.

Greedy Algorithm

The Greedy algorithm places IAPs one at a time. Every new iteration considers all previous selections in place, choosing the best node using a cost function that is based on the number of hops. The selected cost function should ideally reflect the performance requirements. For example, a requirement to maximize network lifetime may change this cost function from the number of hops to one that is a function of the available energy. An adaptive cost function that reflects the diverse node requirements in terms of resource availability and the performance requirements is a subject of a future work.

The greedy A algorithm assumes that every node N_i is a potential source of data. However, in reality only a subset of N nodes are data sources for the IMO. We call this subset S. Based on the assumption that the IMO is aware of the locations of the information sources, the Greedy B algorithm considers only the S nodes as information sources rather than every potential node (i.e., N). In our case, this information is passed from the Management Application to the IMO through the IMO Controller. In environments that all nodes are potential information sources, the Greedy A algorithm may be used instead.

We note that the Random and Greedy algorithms are suitable for both the initial and the dynamic placement of IAPs because they can produce a result from a partial input. The Hotspot algorithm may be used only for the initial placement, because it requires a cost function calculation for each node in the network.

2) Policy Engine: It is evident that there are a number of parameters to be considered when configuring the information management overlay, such as the size of the network, the number of IAPs to be used, and the tolerance level of information accuracy. Furthermore, some of these parameters may change during the course of the systems operation and reconfiguration may be required to maintain optimized collection of information. For this reason, our approach incorporates policy-based management [22] as the mechanism to achieve adaptation in a flexible and simplified manner. We have used policies to drive the functional behavior of the overlay, based on the requirements of the management application and the properties of the environment. The policies are executed by the IMO Controller and have an overlay-wide scope. Our implementation is based on the Ponder2 framework [23] and considers two policy types that: (a) deploy the appropriate IAP placement algorithm, and, (b) set the accuracy level at which the information is collected. The decision on the placement algorithm in the first policy type is based on a set of constraints that represent network properties and the optimization requirement from the management application. An instance of such a policy is specified below in Ponder2 format, where the topology size (medium - "Med"), its type (dynamic

- ”Dyn”), and the optimization requirement (response time - ”Resp”) are encoded in the conditional part. The action (execGreedy) executes a script that implements the Greedy algorithm, passing the number of IAPs to be deployed (3 in this case) as a parameter.

```
policy := root/factory/ecapolicy create.
policy event: root/event/deplIMO.
policy condition: [ :topolSize :topolType :optzType |
          (topolSize=="Med") & (topolType=="Dyn") &
          (optzType=="Resp") ].
policy action: [ root/plAlg execGreedy: 3 ].
root/policy at: "greedyAlg" put: policy.
policy active: true.
```

As mentioned in Section IIIB, the accuracy with which information is transmitted has an impact on the communication overhead. The IMO Controller can regulate this parameter on both ICPs and IAPs by configuring their filter. The following policy is enforced only if the management application requires the optimization of network resources, and sets the accuracy as a deviation percentage (5%).

```
policy := root/factory/ecapolicy create.
policy event: root/event/deplIMO.
policy condition: [ :optzType | optzType=="Net" ].
policy action: [ root/fltr setAcc: 5 ].
root/policy at: "filter" put: policy.
policy active: true.
```

The advantage of introducing interpreted logic in the form of policies is that the behavior of the managed overlay can be dynamically changed without modifying the underlying implementation. The filtering value can, for example, be adjusted without interrupting the information collection process. In the current implementation, and for evaluation purposes, policies are manually triggered and the constraint parameter values are represented by static variables. Part of the future work in this area involves acquiring dynamic environment properties and optimization requirements, and then executing the policies at run-time. Furthermore, policy logic will also be introduced within ICPs and IAPs controllers so that decisions can be taken locally in a distributed fashion.

IV. Experimental Analysis

In this section we evaluate different aspects of the IMO infrastructure using both real experiments and simulations. We implemented all basic functionalities of the IMO infrastructure including Information Collection Points that monitor network and system resources, Information Aggregation Points that perform dynamic placement, some sample Information Consumers, and the policy-driven IMO Controller.

We performed the following tests:

- *Impact of filtering mechanism*, where we show how the information accuracy can be traded for communication cost.
- *Impact of information source*, where we show the impact of different sources of information, including CPU and memory sensors, and an emulated source (i.e., the response-time emulator).
- *Impact of AP placement algorithms*, where we show the impact on the communication overhead and their processing cost in small-scale and large-scale networks.

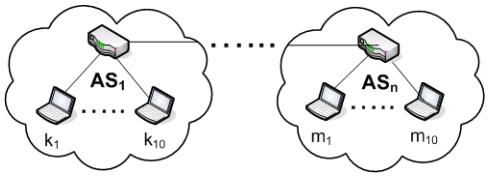

Fig. 5. Simulation topology

We complemented our real experiments with simulations in order to explore the impact of the placement algorithms in large-scale topologies.

A. Experimental Setup

We deployed the IMO infrastructure on a test-bed that consists of two SUN servers with 4 CPU cores (2.5 GHz each) and 8GB of memory and one with 8 CPU cores (1.9 GHz each) and 32GB of memory. Each server hosts 10 virtual machines using the XEN virtualization platform [24]. With such a setup, we emulated a network that consists of 33 machines. In our experiments, each physical machine acts as an emulated Autonomous System (AS).

In the real scenarios, we gradually increase the number of physical machines from one to three, in order to show the impact of our solution in different topology scales. In these experiments, all the virtual machines act as information sources. One Aggregation Point is deployed dynamically using the AP placement algorithms and the Information Consumer is hosted on the last physical machine of each experimental run. Greedy B is the default placement algorithm used.

Here, we measure *throughput* at the Aggregation Points and the Information Consumers and *transmission time* from the information collectors to the aggregation points. The Information Collectors and Aggregation Points transmit a measurement every 30 seconds, unless it is otherwise stated. We note that the experimental scenarios were defined as appropriate inputs to the IMO Controller and the corresponding policies.

In the simulation scenarios, we evaluated the proposed IMO-node placement algorithms using the linear topology shown in Fig. 5, matching the emulated topology in the test-bed. We implemented the simulator using TCL and executed the experiments on an Acer TravelMate 3022WTMi laptop with 1GB of memory. In our implementation, we used the Dijkstra’s shortest path algorithm for any node distance calculation in hops.

For the simulations, we measure the average data transmitted per hop (measured in Bytes / sec / hop) and the average processing time for the IMO-node placement algorithm. The former metric quantifies the communication and the latter the processing cost.

In all experiments, we assume that the IMO is aware of the graph of the network topology. This assumption is not unreasonable, since topology is one of the most important information aspects that need to be handled from such infrastructures.

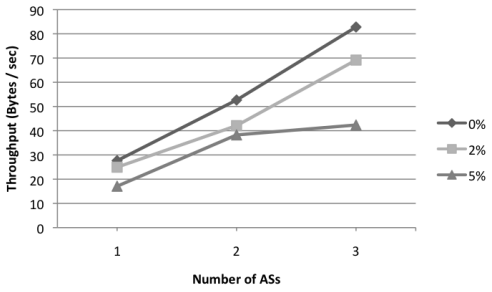

Fig. 6. Impact of Filtering on the Aggregation Point

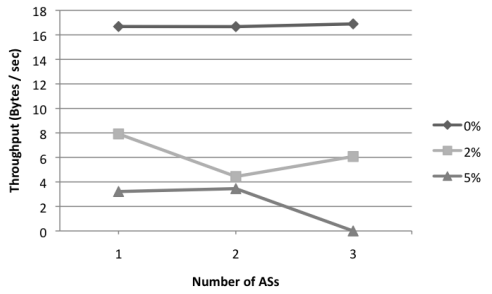

Fig. 7. Impact of Filtering on the Information Consumer (rate 5 sec)

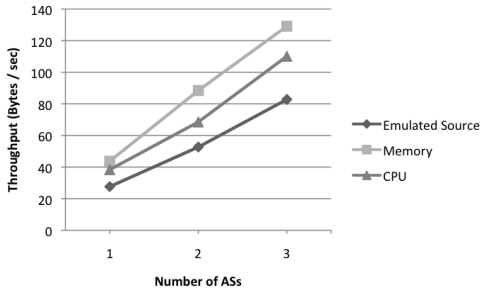

Fig. 8. Impact of Information Source on the Aggregation Point

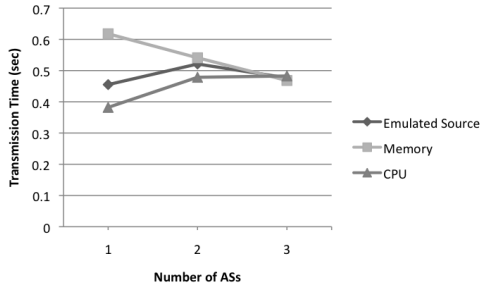

Fig. 9. Impact of Information Source on the Aggregation Point

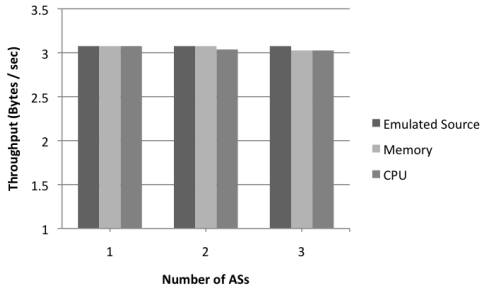

Fig. 10. Impact of Information Source on the Information Consumer

B. Real measurements

In the following scenarios, we tested the IMO infrastructure in different conditions, focusing on different aspects each time. We evaluated the statistical accuracy of our results by performing 10 runs and calculating the standard deviation. We did not observe any large deviations. For example, in a scenario with 2 ASs and 20 VMs acting as information sources, the standard deviation in the *throughput* measurements was just 3.4% of the average *throughput* value. Now we present our results.

1) Impact of Information Filtering: As we can see in Fig. 6, the *throughput* at the AP is reduced significantly due to the filtering algorithm. So, filtering can adjust the trade-off between information accuracy and communication overhead. The impact grows with more information sources and larger topologies. In Fig. 7, we show the impact of filtering on the *throughput* that reaches the information consumer, in case the aggregation point transmits one measurement every 5 sec. Again, filtering improves communication overhead. We note that in certain cases, the communication overhead may be zero (e.g., for 5% filtering and 3 ASs - Fig. 7), in the situation where the deviation of the monitoring value never exceeds the filtering specification.

2) Impact of Information Sources: In this scenario, we show the impact of different information sources on the *throughput* and *transmission time*. We use different information sensors that monitor CPU utilization and Memory. Additionally, we use a sensor that collects measurements from a response-time emulator. As we can see in Fig. 8, the type of information impacts on the communication cost. In our case, the emulated source associates with much less *throughput* than the CPU and memory sources. In the case of the *transmission time* (see Fig. 9), the difference is cancelled for larger topologies.

3) Impact of AP placement algorithms: In this scenario, we evaluate the impact of the placement algorithms on the communication cost. Here, we used the second version of the Greedy algorithm. In our case (i.e., see Fig. 11), the random placement of the AP often results to a position many hops away from the information sources, which increases significantly the communication overhead. On the other hand, an aggressive placement algorithm, such as the hotspot, can find the best location for the AP. In this scenario, the greedy algorithm is a good choice too, because performs well and is associated with less processing cost than the hotspot, especially if the number of sources are less than the available nodes. Larger topologies and numbers of APs are evaluated using simulations. These results can be found below.

C. Simulation Results

We ran two series of simulations, consisting of 5 and 10 networks in the backbone line (i.e., Autonomous Systems - ASs), respectively. As illustrated in Figure 5, 10 nodes were connected to each AS. All of our simulations have

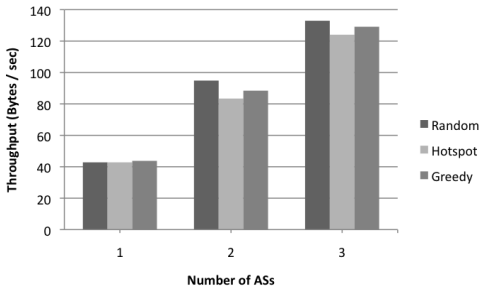

Fig. 11. Impact of AP Placement Algorithm

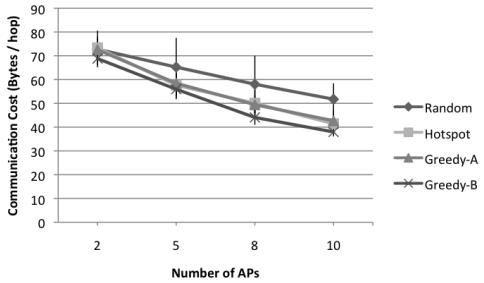

Fig. 12. Communication Cost (5 ASs)

a non-deterministic nature. Consequently, we executed each one 20 times using different initiation values. Each of our figures depicts the average values of the particular metric. In the figures, we indicate the standard deviation of our measurements, every time it is not insignificant.

We randomly placed 10 ICPs in the topology, each of which periodically transmit data (i.e., with a period of 1 sec) to the nearest IAP (in number of hops). The data is collected from the IAP, processed, and transmitted again to a node that exploits this information (i.e., an Information Consumer). The node that resolves information is also placed randomly in the topology. In our scenario, we assume that each data packet has the size of 100 bytes. The data transmission between the AP and the Information Consumer has a period of 10 seconds, assuming that the aggregated information requires lower data transmission rates. In both scenarios, we ranged the number of APs from 2 to 10.

The Fig. 13 and 15 show that the Random placement algorithm is associated with the lower processing cost. This result is consistent with the complexity of the algorithm that is O (MN) [21]. However, the lower complexity comes with a high communication cost (as in Fig. 12 and 14). The Hotspot and Greedy B algorithms have similar complexity (see Fig. 13 and 15) but the Greedy B algorithm performs better in terms of communication cost (as in Fig. 12 and 14). The Hotspot and Greedy A algorithms produce almost the same communication cost. However, the associated processing cost is significantly higher for the Greedy A algorithm.

In general, the Greedy B algorithm performs better in terms of communication and processing cost. In this example, the Greedy B algorithm produces almost the same communication cost with the Greedy A algorithm (see Fig. 14). However, the latter does use 3 less APs.

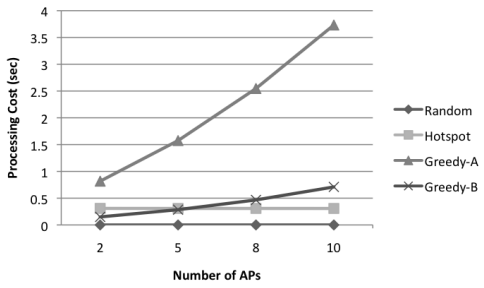

Fig. 13. Processing Cost (5 ASs)

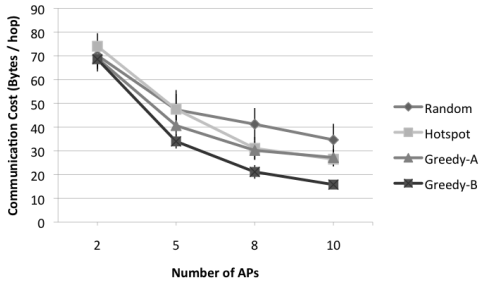

Fig. 14. Communication Cost (10 ASs)

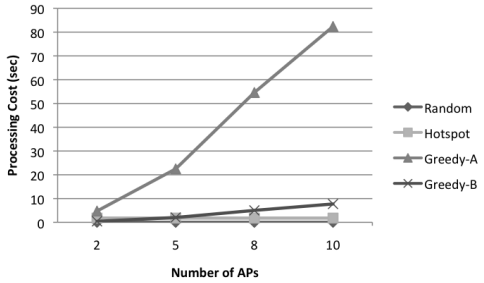

Fig. 15. Processing Cost (10 ASs)

V. Discussion & Open Issues

We consider that future work can be out in the following areas:

- to evaluate IMO robustness with dynamic scenarios, e.g., dynamic topologies or information flow requirements.
- to enrich the IMO system with additional policy-driven functionality, e.g., consider additional programmable parameters of individual placement algorithms.
- to add new optimization algorithms that focus on other aspects of the information management, e.g., information distribution, network protocol used etc.

Finally, to aid researchers in this area it is a goal to release the IMO platform as an open-source software.

VI. Conclusions

This paper presents the design, the implementation, and the optimization issues of an Information Management Overlay. This IMO is an inner component of the Autonomic Management Architecture [4] proposed in the Autonomic Internet project [3]. The IMO supports network management

applications with information, thus acting as an enabler for self-management functionality. We discuss different strategies to regulate the information flow with respect to the existing conditions and performance requirements. We validated our proposal with real experiments and simulations.

ACKNOWLEDGMENT

This work is partially supported by the European Union through the Autonomic Internet (AutoI) project [3] and the RESERVOIR project [25] of the 7th Framework Program. We would like to thank the members of the AutoI consortium for their helpful comments.

REFERENCES

[1] M. Dam and R. Stadler, "A generic protocol for network state aggregation," in *In Proceedings of Radiovetenskap och Kommunikation (RVK)*, 2005.

[2] R. A. Gonzalez Prieto, "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, June 2007.

[3] "Autonomic Internet (AutoI) Project," http://www.ist-autoi.eu/, 2008-2010.

[4] 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.

[5] 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.

[6] J. Li, "Agent organization in the knowledge plane," Massachusetts Institute of Technology, Tech. Rep. MIT-CSAIL-TR-2008-034, June 2008.

[7] C. H. Crawford and A. Dan, "emodel: Addressing the need for a flexible modeling framework in autonomic computing," *International Symposium on Modeling, Analysis, and Simulation of Computer Systems*, 2002.

[8] X. Dong, S. Hariri, L. Xue, H. Chen *et al.*, "Autonomia: an autonomic computing environment," 2003, pp. 61–68. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1203684

[9] A. Malatras, A. M. Hadjiantonis, and G. Pavlou, "Exploiting context-awareness for the autonomic management of mobile ad hoc networks," *Journal of Network and Systems Management*, vol. 15, no. 1, pp. 29–55, 2007.

[10] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S. K. S. Gupta, "Reconfigurable context-sensitive middleware for pervasive computing," *IEEE Pervasive Computing*, vol. 1, no. 3, pp. 33–40, 2002.

[11] J. Mantyjarvi, P. Huuskonen, and J. Himberg, "Collaborative context determination to support mobile terminal applications," *IEEE Wireless Communications*, vol. 9, pp. 39–45, 2002.

[12] G. Judd and P. Steenkiste, "Providing contextual information to pervasive computing applications," in *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2003, p. 133.

[13] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi, "Holistic aggregates in a networked world: distributed tracking of approximate quantiles," in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2005, pp. 25–36.

[14] A. Sharaf, J. Beaver, A. Labrinidis, and K. Chrysanthis, "Balancing energy efficiency and quality of aggregate data in sensor networks," *The VLDB Journal*, vol. 13, no. 4, pp. 384–403, 2004.

[15] 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.

[16] R. Srinivasan, "Xdr: External data representation standard," 1995.

[17] 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.

[18] S. Clayman, "Time indexing - an introduction," May 2003, `http://www.timeindexing.com/documentation/papers.html`.

[19] ——, "Timeindexing repository," `http://xircles.codehaus.org/projects/timeindexing`.

[20] A. Sahai, A. Sahai, S. Graupner, S. Graupner *et al.*, "Specifying and monitoring guarantees in commercial grids through sla," in *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'03)*, 2002, pp. 292–299.

[21] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *In Proceedings of IEEE INFOCOM*, 2001, pp. 1587–1596.

[22] M. Sloman, "Policy driven management for distributed systems," *Journal of Network and Systems Management*, vol. 2, pp. 333–360, 1994.

[23] "Ponder2 policy framework," `http://www.ponder2.net/`.

[24] 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.

[25] B. Rochwerger, D. Breitgand, E. Levy, A. Galis *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, 2009. [Online]. Available: http://www.research.ibm.com/journal/rd/534/rochwerger.pdf