Monitoring Flow Aggregates with Controllable Accuracy

Alberto Gonzalez Prieto and Rolf Stadler

KTH Royal Institute of Technology
Stockholm, Sweden
{gonzalez, stadler}@ee.kth.se

Abstract. In this paper, we show the feasibility of real-time flow monitoring with controllable accuracy in today's IP networks. Our approach is based on Netflow and A-GAP. A-GAP is a protocol for continuous monitoring of network state variables, which are computed from device metrics using aggregation functions, such as SUM, AVERAGE and MAX. A-GAP is designed to achieve a given monitoring accuracy with minimal overhead. A-GAP is decentralized and asynchronous to achieve robustness and scalability. The protocol incrementally computes aggregation functions inside the network and, based on a stochastic model, it dynamically configures local filters that control the overhead and accuracy. We evaluate a prototype in a testbed of 16 commercial routers and provide measurements from a scenario where the protocol continuously estimates the total number of FTP flows in the network. Local flow metrics are read out from Netflow buffers and aggregated in real-time. We evaluate the prototype for the following criteria. First, the ability to effectively control the trade off between monitoring accuracy and processing overhead; second, the ability to accurately predict the distribution of the estimation error ; third, the impact of a sudden change in topology on the performance of the protocol. The testbed measurements are consistent with simulation studies we performed for different topologies and network sizes, which proves the feasibility of the protocol design, and, more generally, the feasibility of effective and efficient real-time flow monitoring in large network environments.

1 Introduction

Several key management tasks, such as SLA verification, accounting and intrusion detection depend on monitoring state variables in the network. For many such tasks, the IP flow has emerged as the appropriate level of abstraction and granularity for monitoring. This has made flow monitoring an active research topic [14][17][18]. Its high relevance in practical scenarios has led the IETF to create the *IP Flow Information Export* (IPFIX) working group, focused on standardizing different aspects of flow monitoring, such as information models and information exchange protocols [15].

One of the key challenges in flow monitoring is controlling the trade off between the costs (e.g., processing resources, memory requirements, management traffic) and the accuracy of the monitored metrics. Examples of research efforts in this area are [16][17][18].

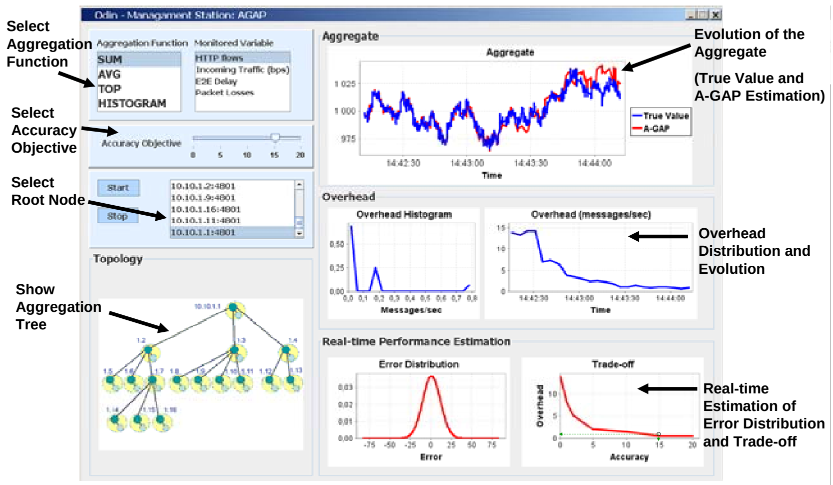

Figure 1. Interface on the management station for evaluating A-GAP on the testbed. It shows the effect of changing the accuracy objective from 0 to 15 flows. at time 14:42:30 in a testbed scenario. As a consequence, A-GAP reduces the overhead at the cost of an increased error in estimating the aggregate. The interface provides also real-time estimation of the error distribution and of the trade-off curve accuracy vs overhead.

A relevant technique in this context is aggregation. It consists on computing network-wide metrics from device-level metrics across a network. Examples of aggregation functions are SUM, AVERAGE, MIN, MAX, and HISTOGRAM. Sample flow aggregates are the total number of VoIP flows, the most popular flow destination, or a histogram of flow sizes in a network domain. Monitoring flow aggregates enables administrators to learn the volume of traffic different applications generate and infer the performance requirements of end users. Monitoring flow aggregates also permits identifying elephant flows, a key task in traffic engineering [20].

While it is often crucial to know how accurate aggregate estimates are, network management solutions deployed today usually provide only qualitative control of the accuracy and do not support the setting of an accuracy objective [4].

The focus of this paper is on providing continuous estimates of flow aggregates with controllable accuracy in today's IP networks.

Our solution is based on Netflow [14] and A-GAP [11], a generic aggregation protocol with controllable accuracy. Router-level flow metrics are read from Netflow buffers. A-GAP continuously aggregates these router-level metrics into network-wide metrics by (i) creating and maintaining a self-stabilizing spanning tree and (ii) incrementally aggregating the metrics along the tree. A-GAP is push-based in the sense that changes in monitored metrics are sent towards the management station along the aggregation tree. The protocol controls the management overhead by filtering updates that are sent from monitoring nodes to the management station. The filters periodically adapt to the dynamics of the monitored variables and the network environment. All operations in A-GAP, including computing the aggregation function and filter configuration, are executed in a decentralized and asynchronous fashion to ensure robustness and achieve scalability. [11] contains the description of the stochastic model A-GAP uses for filter computation and performance prediction.

This paper reports on our prototype for flow monitoring with controllable accuracy and its evaluation on a testbed of commercial routers. At the cost of introducing an overlay of monitoring nodes, no changes to the routers are required. The results presented validate the protocol design and suggests the feasibility of real-time flow monitoring in large-scale dynamic network environments.

The paper is organized as follows. Section 2 provides an overview of A-GAP. Section 3 discusses the implementation of A-GAP. Section 4 contains the evaluation scenarios and the testbed results. Section 5 discusses related work. Section 6 concludes the paper.

2 Overview of A-GAP

2.1 Problem Statement

We consider a dynamically changing network graph G(t) = (V(t), E(t)) in which nodes $n \in$ V(t) and edges/links $e \in$ E(t) $\subseteq$ V(t) x V(t) may appear and disappear over time. Each node *n* has an associated local variable $w_n(t)$. The term *local variable* is used to represent a local state variable or device counter that is being subjected to monitoring. Local variables are updated asynchronously with a given sampling rate.

The objective is to engineer a protocol on this network graph that provides a management station with a continuous estimate of $\Sigma_n w_n(t)$ for a given accuracy. The protocol should execute with minimal overhead in the sense that it minimizes the (maximum) processing load over all nodes. The load is expressed as the number of updates per second a node has to process. The accuracy is expressed as the *average error* of the estimate over time.

Throughout the paper we use SUM as aggregation function. Other functions can be supported as well, as discussed in [11].

2.2 A-GAP

A-GAP is based on GAP (Generic Aggregation Protocol), an asynchronous distributed protocol that builds and maintains a BFS (Breadth First Search) spanning tree on an overlay network [1]. The tree is maintained in a similar way as the algorithm that underlies the 802.1d Spanning Tree Protocol (STP) [5]. In GAP, each node holds information about its children in the BFS tree, in order to compute the partial aggregate, i.e., the aggregate value of the local management variables from all nodes of the subtree where this node is the root. GAP is event-driven in the sense that messages are exchanged as results of events, such as the detection of a new neighbor on the overlay, the failure of a neighbor, an update to an aggregate or a change in the local management variable.

A drawback of such an approach is that it can cause a high load on the root node or on nodes close to the root, specifically in large networks. In order to reduce this overhead, A-GAP introduces filters in the nodes. When the partial aggregate (or the local variable in the case of a leaf node) of a node *n* changes, then *n* sends an update to its parent if the difference between the value reported in its last update and the current value exceeds the local filter width F^n.

Minimizing the Protocol Overhead. Estimating the network variable at the root node with minimal overhead for a given accuracy can be formalized as an optimization problem. Let n be a node in the network graph, ω^n the rate of updates received by node n from its children, E^{root} the distribution of the estimation error at the root node, and ε the accuracy objective. We formulate the problem as

$$\text{Minimize } \underset{n}{Max}\{\omega^n\} \quad \text{s.t.} \quad E(|E^{root}|) \leq \varepsilon \tag{1}$$

whereby ω^n and E^{root} depend on the filter widths $(F^n)_n$, which are the decision variables.

We have developed a stochastic model for the monitoring process. The model is based on discrete-time Markov chains and describes individual nodes in their steady state. For each node n, it relates the error of the partial aggregate of n, the step sizes that indicate changes in the partial aggregate, the rate of updates n sends and the width of the local filter. The model is described in detail in [11]. The model permits us to compute the distribution of the estimation error at the root node and the rate of updates processed by each node.

A-GAP continuously estimates the evolution of the management variables that the protocol aggregates, one of the variables in our model. Based on these estimates, all others model variables, such as the error distributions and incurred overhead, are dynamically computed. Such an approach lets A-GAP adapt quickly, compared to an approach whereby all model variables are estimated.

A Local Heuristic. An optimal solution to (eq. 1) can be computed using a (centralized) grid search algorithm, a well-known optimization technique, where the model variables for all nodes in the aggregation tree are computed bottom-up. Such an approach, however, is not feasible for large networks, since the computational cost of this algorithm grows exponentially with the number of nodes. A-GAP realizes a distributed heuristic, which attempts to minimize the maximum processing load on all nodes by minimizing the load within each node's neighborhood. A-GAP maps (eq. 1) onto a local problem for each node n as follows:

$$\text{Minimize } \underset{\pi}{Max}\{\omega^\pi\} \quad \text{s.t.} \quad E(|E^n_{out}|) \leq \varepsilon^n, \tag{2}$$

where π is the set composed by the node n and its children. This means that node n attempts to minimize the maximum load in a neighborhood for a given accuracy objective ε^n of its partial aggregate.

The node attempts to solve (eq. 2) by periodically re-computing the filters and accuracy objectives of its children, based on the stochastic model. Re-computing the filters $(F^c)_c$ allows node n to influence its own load ω^n, while re-computing the accuracy objective ε^c of a child c allows the node to influence the load ω^c on c.

A-GAP computes the local filters and accuracy objectives in a decentralized and asynchronous fashion, as described in detail in [11].

The two keys configuration parameters of A-GAP are (i) the maximum number of children whose filters and accuracy objectives are recomputed during a control cycle $|\Omega|$, and (ii) the period of the control cycle τ. As discussed in [11], both parameters influence the adaptability and computational cost of A-GAP.

3 Implementation

A-GAP executes on a distributed management architecture, whereby each network device participates in the monitoring task by running a *management process*, either internally on the network element or on an external associated device. In our testbed, the management processes execute on Linux PCs, or alternatively, on low-cost mini-computers. Each computer, which we also call a *monitoring node*, runs the management process associated with one of the routers. Monitoring nodes communicate with each other via overlay links.

Figure 2 shows the design of a monitoring node. The *node manager* is responsible for executing the commands from overlay peers and the management station. These include the invocation of services and protocols. Local services a node supports are overlay maintenance, node/link failure detector, reliable communication and local device access. The overlay maintenance service constructs and maintains the overlay that interconnects the monitoring nodes. The failure detector detects the failing of a neighboring node. The reliable communication service provides reliable and secure message passing across overlay links. The device access service provides access to local variables on the network device through SNMP, CLI, Netflow, etc.

A monitoring node is implemented in Java. A-GAP alone is in the order of 2500 lines of code. The heuristic used for solving the problem shown in (eq. 2) is implemented using JSci (v0.94) for solving systems of linear equations [2]. Message exchange between monitoring nodes is implemented using XML. All protocol invocations and services run as threads in a single JVM.

The interface on the management station shown in figure 1 facilitates the evaluation of A-GAP. It allows setting configuration parameters of the protocol, including the aggregation function, the accuracy objective, and the root node of the aggregation tree. Once the protocol has set up the aggregation tree on the overlay, the tree topology is displayed in the lower left corner. On the right side, the interface provides real-time information on A-GAP's performance. First and foremost, the estimate of the aggregate and its evolution over time (top right). Second, the distribution of the estimation error for the current networking conditions and objective (bottom center). Third, the estimated trade-off curve between the protocol overhead and the error objective for the current networking conditions (bottom

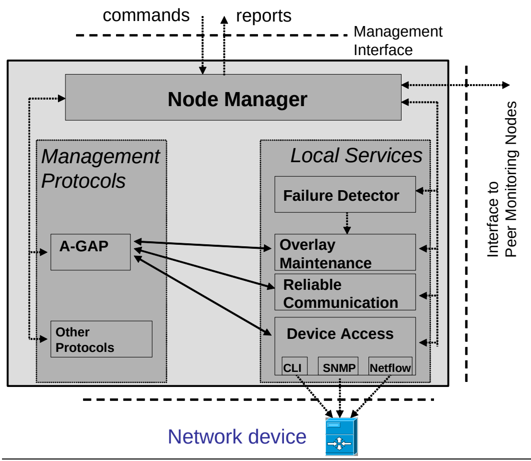

Figure 2. Design of a monitoring node

right). The current operating point on the curve is displayed as well. Furthermore, the interface provides data from an application that monitors the execution of A-GAP, namely, the true aggregate over time, as well as the distribution and evolution of the management overhead (blue curves in figure 1). These metrics are pushed by the management processes to a collecting node. The interface is built with JFreeChart (v.1.0.2) [3] that draws the graphs in real-time.

4 Evaluation

4.1 Testbed Setup

Figure 3 gives the setup of our testbed for the evaluation and shows an aggregation tree on the overlay created by A-GAP. The testbed includes 16 Cisco 2600 Series routers and 16 rack-mounted PCs running the monitoring nodes. Routers and PCs are connected through four 100Mbps Ethernet switches (a Netgear FSM750S and three Netgear FSM726S). An NTP server synchronizes the clocks on the PCs for the purpose of estimating the “true” aggregate. A Spirent Smartbits 6000 programmable traffic generator injects flows into the testbed.

4.2 Measured Metrics

During the experiments, we collect the following metrics. First, we trace A-GAP’s *estimation of the aggregate* by logging all updates of the aggregate at the root node. Second, we trace the value of the *local variable* of each node (obtained by reading the router’s Netflow cache) by logging all updates to this variable on the node. The first and the second metrics are used to compute the *estimation error*, which we define as the average difference between the sum of all the local variables (called the *true value* throughout the paper) and the *estimation of the aggregate* by A-GAP. Third, we trace the *management overhead* of each node by counting the number of updates each node receives during a control cycle. This data is used to compute the *maximum load* over all

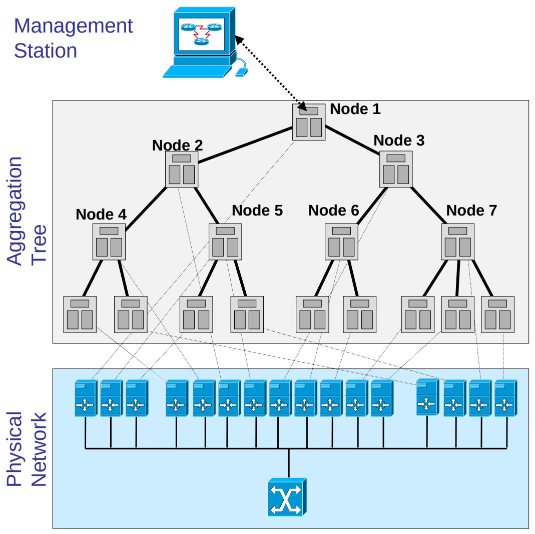

Figure 3. Testbed configuration. The area at the bottom represents the physical network: 16 commercial routers. Each router is associated with a monitoring node. Monitoring nodes communicate via an overlay (middle area). The management station on top interacts with the root node of the aggregation tree.

nodes in the testbed.

4.3 Scenarios Description

The local management variable in the experiments is the number of FTP flows entering the network through that node. Therefore, the monitored aggregate is the number of FTP flows in the network. The traffic generator injects flows into the testbed following a random walk process with barriers. For the experiments in this paper, the aggregate takes values between 0 and 450 flows.

The local variables are sampled asynchronously, once every second, and are read from the Netflow caches of the Cisco routers through CLI.

During all experiments, the overlay topology does not change, and the aggregation tree set up by the protocol has the structure given in figure 3. The control cycle of A-GAP is set to 5 seconds, and the number of children whose filters are recomputed during a control cycle is set to 2.

All experiments start with an initialization phase of some 30 seconds, in which the aggregation tree is set up and the model variables are estimated or computed. This is followed by a transient period of up to 60 seconds. After that, the measurement period starts, which is 350 seconds for all experiments. The accuracy objective is set at the beginning of the experiment and it is not changed during a run.

4.4 Measurement Results

Estimation accuracy versus protocol overhead. We have run a set of experiments with different accuracy objectives, and we have measured the protocol overhead in function of the experienced error. Every point in figure 4 corresponds to a run on the testbed

We observe that the overhead decreases as the estimation error increases. As the error grows larger, the decrease becomes smaller. Estimation errors above 10 flows do not significantly reduce the overhead anymore.

This observation is consistent with simulation results of A-GAP, where we see the same qualitative behavior for different overlay topologies and network sizes ranging from tens of nodes to several hundreds [11].

Meeting the accuracy objective. A further analysis of the measurement data shows that the difference between the accuracy objective and the experienced estimation error is small. For all the experiments in this evaluation, which include dozens of runs, it has been less than one flow (considering that typical aggregate values in our experiments are around 200 flows).

The difference between accuracy objective and the experienced estimation error has two main causes. First, updates from different nodes in the network experience different delays in reaching the root, which distorts the evolution of the estimate at the root node. (This distortion is not captured by our stochastic model [11], since, for reasons of simplicity, it does not consider networking and processing delays). A second cause is the inaccuracy in the stochastic model variables used for filter computation, for instance, as a result of errors in the estimation of the evolution of the local variables.

These measurement results demonstrate that we can effectively control the accuracy of the estimation that A-GAP provides. Second, we can control the trade off between the accuracy of the estimation and the protocol overhead. Specifically, the larger the error A-GAP is allowed to make, the smaller the overhead it incurs.

Robustness: a router gets disconnected. In this experiment, we assess the adaptability of A-GAP to the disconnection of a router (and the computer running its associated management process). The disconnection happens instantly and the failure detectors in the neighbors of the disconnected node detect the failure in a sub-second.

When a failure is detected, A-GAP reconstructs the spanning tree. At the same time, the partial aggregates in some nodes are recomputed. The local mechanism for filter re-computation assures that the filters in the nodes adapt to the new tree structure.

In this particular experiment, node 6 (shown in figure 3) is disconnected from the network at time ≅ 101 seconds. The spanning tree is reconstructed on the overlay (not shown in the figure). Figure 5 shows traces of the experiment. In the upper graph, which shows the maximum load across all nodes, we see no apparent transient period. In the lower graph, which shows the estimation of the aggregate provided by A-GAP and the true value, we observe a spike during a transient period of a sub-second. For this experiment, the accuracy objective is 4 flows, and we see that the objective is achieved both before and after the failure.

This observation is consistent with the properties of protocols that use aggregation trees [9][1] and with our results from simulating A-GAP [11], where we observe brief spikes in estimation errors. During tree reconstruction, some local variables may be considered more than once, or not at all, in the estimation of the global aggregate, which explains the spike in the estimation error at the root node. Specifically, in this experiment, the local variables of the children of node 4 are not considered at the root until updates from their new parents in the tree reach the root.

When simulating A-GAP for large networks, we have seen a significant peak in the overhead for cases where tree reconstruction involves a large number of nodes–a phenomena that we did not expect to see in our small testbed.

Distribution of the estimation error. In this experiment, we evaluate the capability of A-GAP for providing performance estimation. Based on our stochastic model, A-GAP can provide, for a given error objective, the distribution of the estimation error and the expected overhead at the nodes.

Figure 6 shows the error distribution for a run where A-GAP constructs a tree as shown in figure 3. The accuracy objective is 4 flows. One curve shows the error distribution estimated by A-GAP, the other gives the result from measuring the errors on the testbed.

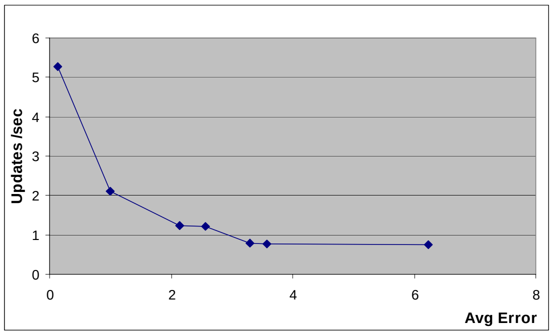

Figure 4. Testbed measurements: management overhead incurred by A-GAP as a function of the accuracy of the estimation.

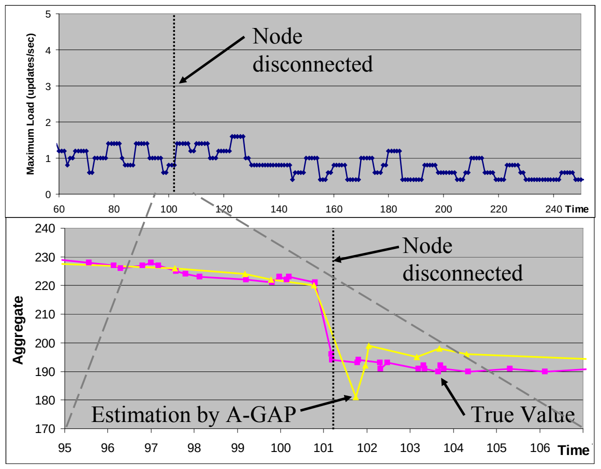

Figure 5. Estimation error at the root node and management overhead caused by a node disconnection.

We observe that both curves are close to each other, and the estimation by A-GAP is accurate in this sense. We see also that both distributions have long tails. The maximum possible error (i.e., the sum of all filter widths) in this run is 26. The estimated probability of having such a large error, though, is very small, in the order of 10-13. The actual maximum error during the experiment is 14. (The observation that the maximum error is a rare event is also made by other authors [9][12]. This confirms our choice of the average error as control parameter for the protocol, rather than the maximum error, which other authors advocate [10].)

Other measurements from our testbed also show that A-GAP accurately estimates the expected protocol overhead. Results can be found in [13].

Real-time Monitoring with Netflow. While we have developed A-GAP as a protocol for estimating aggregates of local variables in real-time, the accuracy of this estimation depends on the accuracy of the local variables, which capture device counters, local MIB objects, etc. In the measurements presented in this paper, the local variable is the number of entries in a Netflow cache. Netflow keeps an entry of a flow traversing a router as long as its latest packet has traversed the router within the interval [now()-Δt, now()]. (In our setup Δt is 10 seconds.). As Netflow defines flows in terms of packet inter-arrival times, counting the flows in a Netflow cache generally overestimates the number of flows currently traversing a router. In this sense, the figures in this paper overestimate the number of flows on the testbed. By taking into account flow statistics, such as flow duration, a more accurate estimation of the number of flows that currently traverse a router can be computed, and we are implementing such an algorithm on the monitoring nodes. Note though that local algorithms for obtaining the local variables are needed by but independent of A-GAP.

4.5 Comparative Evaluation on a Simulator

In order to compare the results from the testbed with those from the simulation-based evaluation of A-GAP [11], we have run testbed experiments where the local variables are based on the packet traces used in the simulations. The traces were captured on two 1 Gbit/s links that connect University of Twente to a research network. We have compared testbed results with simulation experiments for the same A-GAP configuration and traces. For the simulation runs, the link speeds in the overlay are set to 100 Mbps. The communication delay is set to 4 ms, and the time to process a

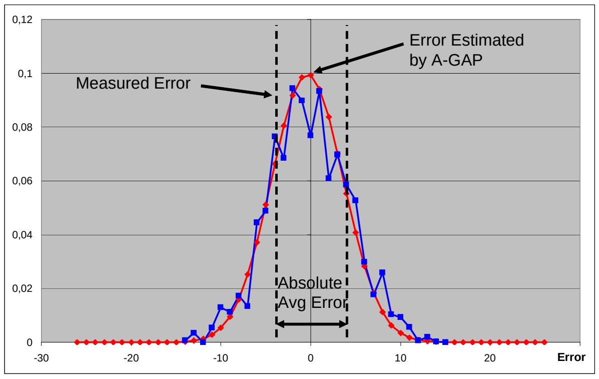

Figure 6. Distribution of the error predicted by A-GAP and the actual error at the root node

message at a node is set to 1 ms. For a more detailed description of the simulation framework and set up, see [11].

This comparative evaluation is one way to strengthen our prediction on how A-GAP would perform in large networks that we have simulated [11]. This would be case, if the simulation results for the testbed configuration turn out to be very similar to the measurement results from the actual testbed. Alternatively, this comparative evaluation will give us some insight into potential limitations of our simulation-based studies.

Our results [13] (not included in this paper due to space restrictions) show that the trade-off curves for a simulation run and testbed measurements are very close. The difference in overhead is below 3,5%.

When considering the difference between the accuracy objective and the estimation error, we observe that the simulation gives results that are closer to the objective, but the differences between testbed and simulation results are very small. We explain this difference with the fact that the simulation model is simplified compared to the reality of the testbed.

5 Related Work

Recently, there has been significant research in real-time monitoring of network aggregates with the goal of achieving accuracy at low cost. For an overview, see [10]. Most of the proposed approaches have been evaluated using simulation. An example of a scheme that has been evaluated in a prototype implementation is described in [6]. The scheme in [6] differs from A-GAP in two ways. First, it is centralized in the sense that the management station computes the filter widths for all nodes, and all nodes communicate directly with the management station. Like A-GAP, [6] requires an execution environment on the nodes. Second, the accuracy objective in [6] is the maximum error, while A-GAP uses the average error as objective (which we argue is more significant for practical applications). [6] reports on an evaluation in a similar testbed setting. The authors show that they can effectively control the trade off between accuracy and overhead and provide a trade off curve that is qualitatively similar to figure 4.

[7] and [8] report on implementations of real-time monitoring of aggregates in the context of sensor networks. Similar to A-GAP, the above two works are based on in-network aggregation along spanning trees. They aim at providing periodically a snapshot of the aggregate, while A-GAP gives a continuous estimation of the

aggregate. The focus of [7] and [8] is on studying the impact of lossy links on the accuracy of the estimated aggregate.

6 Conclusions

In this paper we have presented testbed results from our prototype for flow monitoring with controllable accuracy on a testbed with 16 commercial routers. Our prototype is based on Netflow and A-GAP. A-GAP is decentralized and asynchronous, two key properties for achieving robustness and scalability. At the cost of introducing an overlay of monitoring nodes, no changes to the routers have been required.

The experimental results show that we can effectively control the *trade off between estimation accuracy and protocol overhead* for A-GAP on a testbed. For the scenarios considered in this paper, A-GAP reduces the overhead by one order of magnitude, when allowed an error of 8 flows (figure 4), which is a relative error of less than 1%. The results also show that the protocol adapts quickly to a node failure on the testbed, in a manner that is consistent with what we expect from simulation results.

We also demonstrate the capability of A-GAP for *providing accurate performance estimation in real-time*. The management station can obtain, in real-time, an accurate view of (i) the distribution of the estimation error for the aggregate, and (ii) the expected overhead for each node in the system.

All the above results are consistent with simulation results that have been obtained for different topologies and much larger network sizes (up to some 700 nodes) [11]. Furthermore, the experimental results discussed in section 4.5 show that the behavior of our A-GAP implementation is very similar to that of the protocol running in a simulation environment. This validates our simulation model, proving that its assumptions and simplifications are reasonable. As a consequence, we are much more confident in the simulation results reported in [11], which have been obtained for different topologies and much larger network sizes, and the overall understanding of the behavior of A-GAP.

Together with [11], the results in this paper validate the protocol design and suggest the feasibility of real-time monitoring in large-scale dynamic network environments, in an efficient and effective manner.

Both simulation and testbed experiments [13] have shown that the choice of the *overlay topology can have a significant impact on A-GAPs performance*, and we plan to study this aspect in more detail A second issue that we plan to focus on is reducing the cost of computing the filters in A-GAP. We currently solve the local problem in (eq 2) through exhaustive search, and are searching for an efficient heuristic that yields close to optimal results. Future work also includes the design of algorithms for flow identification. In this paper, the identification of FTP flows is based on the transport port of the flow. This simple algorithm might not be valid for all applications. For instance, a popular VoIP application as Skype can use randomly chosen ports [19].

Acknowledgments. This paper describes work undertaken in the context of the Ambient Networks –an FP6 IST project that is partially funded by the Commission of the European Union.

References

1. M. Dam, R. Stadler, “A Generic Protocol for Network State Aggregation”, Radiovetenskap och Kommunication (RVK), Linkoping, Sweden, June 2005.
2. JSci, http://jsci.sourceforge.net/, December 2006.
3. JFreeChart, http://www.jfree.org/jfreechart/, December 2006.
4. C. Olston, et al., “Adaptive Precision Setting for Cached Approximate Values”, ACM SIGMOD 2001, Santa Barbara, USA, May 2001.
5. IEEE. ANSI/IEEE Std 802.1D, 1998 Edition. IEEE, 1998.
6. C. Olston, J. Jiang and J. Widom, “Adaptive Filters for Continuous Queries over Distributed Data Streams”, ACM SIGMOD 2003, San Diego, USA, June 2003.
7. S.R. Madden et al., “TAG: a tiny aggregation service for ad-hoc sensor networks”, 5th Symposium on Operating Systems Design and Implementation, Boston, USA, December 2002.
8. J. Zhao et al., “Computing aggregates for monitoring wireless sensor networks”, 1st IEEE International Workshop on Sensor Network Protocols and Applications, Anchorage, USA, May 2003.
9. A. Boulis, S. Ganeriwal, and M. B. Srivastava, “Aggregation in sensor networks: an energy - accuracy tradeoff”, Elsevier Ad-hoc Networks Journal (s.i. on sensor network protocols and applications), 2003.
10. A. Gonzalez Prieto, "Adaptive Management for Networked Systems", Licentiate thesis, KTH Royal Institute of Technology, Sweden, June 2006. Available at: http://www.ee.kth.se/~gonzalez
11. A. Gonzalez Prieto, R.Stadler "A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives", IEEE Transactions on Network and Service Management, vol. 4, no. 1, June 2007.
12. M. A. Sharaf, et al., “Balancing energy efficiency and quality of aggregate data in sensor networks”, ACM International Journal on Very Large Data Bases, 13(4):384–403, December 2004.
13. A. Gonzalez Prieto and R.Stadler "Implementation and Evaluation of A-GAP: Adaptive Monitoring with Controllable Accuracy", KTH Technical Report, January 2007, Available at: http://www.ee.kth.se/~gonzalez
14. Cisco Netflow, http://www.cisco.com/warp/public/732/netflow/index.html
15. IETF IP Flow Information Export working group, http://www.ietf.org
16. K. Keys, D. Moore, and C. Estan, “A robust system for accurate realtime summaries of internet traffic”, SIGMETRICS Perform. Eval. Rev., vol. 33, no. 1, pp. 85--96, 2005.
17. M. Molina, A. Chiosi, S. D'Antonio and G. Ventre, “Design principles and algorithms for effective high-speed IP flow monitoring “,Computer Communications Volume 29, Issue 10, 19 June 2006, Pages 1653-1664.
18. L.Yang, G. Michailidis, “Sampled based estimation of network traffic flow characteristics”, IEEE Infocom 2007, Anchorage, USA, May 2007.
19. K. Suh, D. R. Figueiredo, J. Kurose and D. Towsley, “Characterizing and detecting skype-relayed traffic”, IEEE Infocom 2006, Barcelona, Spain, April 2006.
20. T. Mori et al., “Identifying elephant flows through periodically sampled packets”, 4th ACM SIGCOMM conference on Internet measurement, Taormina, Italy, October 2004.