Adaptive Distributed Monitoring with Accuracy Objectives

Alberto Gonzalez Prieto and Rolf Stadler

KTH Royal Institute of Technology

{gonzalez, stadler}@ee.kth.se

ABSTRACT

We present A-GAP, a novel protocol for continuous monitoring of network state variables, which aims at achieving a given monitoring accuracy with minimal overhead. Network state variables are computed from device counters using aggregation functions, such as SUM, AVERAGE and MAX. The accuracy objective is expressed as the average estimation error. A-GAP is decentralized and asynchronous to achieve robustness and scalability. It executes on an overlay that interconnects management processes on the devices. On this overlay, the protocol maintains a spanning tree and updates the network state variables through incremental aggregation. It dynamically configures local filters that control whether an update is sent towards the root of the tree. It reduces the overhead by attempting to minimize the maximum processing load over all management processes. We evaluate A-GAP through simulation using an ISP topology and real traces. The results show that we can effectively control the trade-off between accuracy and protocol overhead, that the overhead can be reduced significantly by allowing small errors, and that an accurate estimation of the error distribution can be provided in real-time.

Keywords

distributed management, real-time monitoring, large-scale distributed systems, adaptive systems

1. INTRODUCTION

The ability to provide continuous estimates of management variables is vital for many management tasks, including network supervision, quality assurance, and proactive fault management. Often, management variables that are monitored in these tasks are aggregates that are computed from device variables across the network using functions such as SUM, AVERAGE, MIN, and MAX. Sample aggregates are the total number of VoIP flows, the maximum link utilization, or a histogram of the current load across routers in a network domain. While it is often crucial to know how accurate such 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.

Engineering continuous monitoring solutions for network management involves addressing the fundamental trade-off

between accurate estimation of a variable and the management overhead in terms of traffic and processing load [6].

In this paper, we address the problem of continuous monitoring with accuracy objectives for large-scale network environments. Our goal is to achieve an efficient solution that allows us to control the accuracy of the estimation.

The paper introduces A-GAP, a generic aggregation protocol with controllable accuracy. A-GAP allows for continuously computing aggregates of local variables by (i) creating and maintaining a self-stabilizing spanning tree and (ii) incrementally aggregating the variables along the tree (fig 1). It is push-based in the sense that changes in monitored variables 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 the filter widths, are executed in a decentralized and asynchronous fashion to ensure robustness and achieve scalability.

We developed a stochastic model of the monitoring process, which allows us to compute the filter widths as the solution to the problem of minimizing the management overhead for a given estimation error. A distributed heuristic solution to this problem is implemented in A-GAP. A distinctive feature of this approach is that it provides us with an estimation of the error distribution at the management station in real-time.

There are recent results available on monitoring network variables with accuracy objectives [6][8][13][14]. All these works, however, express the accuracy in terms of the *maximum estimation error*, which is often at least an order of magnitude larger than the experienced error, and its use in practical scenarios therefore questionable. For this reason, we advocate controlling the *average estimation error*. Furthermore, our protocol can provide an estimation of the maximum error if needed, while the other schemes cannot give any estimation of the experienced error. Regarding other differences to related research, the scheme proposed by Olston et al. [6] is centralized, while ours is decentralized. Several schemes have been recently proposed in the context of wireless sensor networks [8][13][14] and are not directly applicable to network management systems. The main objective of those works is on extending the life time of the network, mainly through minimizing the amount of information exchanged among nodes.

The paper is organized as follows. Section 2 defines the problem of real-time monitoring with accuracy objectives. Section 3 describes our protocol for continuous monitoring with accuracy objectives, A-GAP. Section 4 provides evaluation results. Section 5 concludes the paper.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGCOMM'06 Workshops, September 11-15, 2006, Pisa, Italy.

Copyright 2006 ACM 1-59593-417-0/06/0009...\$5.00.

Fig. 1. A spanning tree on the management overlay for aggregating local management variables. The aggregation function is SUM.

2. The Problem: Real-time Monitoring with Accuracy

System Architecture. This work assumes a distributed management architecture, whereby each network device participates in the computation by running management processes, either internally or on an external, associated device. These management processes communicate via a *management overlay network* for the purpose of monitoring. We also refer to this overlay as the *network graph*. A node in this graph represents a management process.

In the example shown in fig. 1, each management process runs a leaf node and an aggregating node of a spanning tree that spans the management overlay.

While the topology of this overlay can be chosen independently from the topology of the underlying physical network, we assume in this paper, for simplicity, that both topologies are the same, i.e., that the management overlay has the same topology as the underlying physical network.

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 $e \in E(t) \subseteq V(t) \times V(t)$ may appear and disappear over time. Each leaf 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 $\sum_n w_n(t)$ 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

Fig. 2. Each node has a local filter. It sends an update to its parent whenever the partial aggregate (or local variable) exceeds the filter width.

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 section 5.

3. A-GAP: A Distributed Solution

A-GAP, the protocol presented in this paper, is based on GAP (Generic Aggregation Protocol), which we developed in our earlier work [1]. GAP is an asynchronous distributed protocol that builds and maintains a BFS (Breadth First Search) spanning tree on an overlay network. The tree is maintained in a similar way as the algorithm that underlies the 802.1d Spanning Tree Protocol (STP) [9].

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 aggregate update 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, one can either apply a rate limitation scheme, which imposes an upper bound on message rates on each link, or one can introduce a filter scheme, whereby a node drops updates when small variations of its partial aggregate occur. Note that, for both schemes, the overhead is reduced at the cost of introducing an error in estimating the aggregate.

For A-GAP, we choose a filter scheme. When a partial aggregate (or local variable) 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 . Fig. 2 illustrates the behavior of a local filter. The solid line represents the partial aggregate (or local variable), the dashed line gives the

value of the last update from n to its parent, and the dotted line shows the filter width. The vertical line indicates the time at which n sends an update.

3.1 The Optimization Problem

Estimating the aggregate 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, F^n the filter width of node n , E^{root} the distribution of the estimation error at the root node, and ε the accuracy objective. We formulate the problem as:

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

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. It is described in detail in a long version of this paper [10]. For each node n , the model relates the error E^n of the partial aggregate of n , the step sizes that indicate changes in the partial aggregate, the rate ω^n of updates n sends and the width F^n of the local filter. In a leaf node, the step size represents the changes of the local variable, which is modeled as a random walk. The model permits us to compute the distribution E^{root} of the estimation error at the root node and the rate of updates ω^n processed by each node.

An optimal solution to (1) can be computed using a (centralized) grid search algorithm, a well-known optimization technique. 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 on each node individually. A-GAP maps (1) onto a local problem for each node n :

$$\text{Minimize } \omega^n \quad \text{s.t.} \quad E(E^n) \leq \varepsilon^n \quad (2)$$

Each node n attempts to minimize its load ω^n for a given accuracy objective ε^n of its partial aggregate and dynamically adapts the solution to changes in the model variables (the determination of ε^n is discussed below). This is achieved by each node periodically recomputing the filters of its children, based on the model in [10], with the constraint that the filter of a child can not be wider than the filter of a parent.

3.2 The Realization of A-GAP

A-GAP computes the local filters in a decentralized and asynchronous fashion.

Initialization. A-GAP initializes by constructing a spanning tree on the management overlay in the same way as GAP does [1]. The accuracy objectives are set to zero and filters are set to the minimal width on all nodes, so that changes to all local variables

are reported up the aggregation tree to the root node. During a warm-up period, each leaf node starts estimating the evolution of its local variable. After this period, all model variables are initialized on all nodes by running an Echo algorithm [3] on the spanning tree.

Estimating the evolution of the local variables. As stated above, the evolution of a local variable is modeled as a random walk. A-GAP continuously estimates the step sizes for the random walk model on each node, using a maximum likelihood estimator and a moving window. See [10] for details. These step sizes are the only variables that the protocol estimates. All other variables are continuously computed by A-GAP based on these estimates.

Re-computing local filters. Each node n independently runs two processes F1 and F2 in parallel. F1 executes a control cycle with period τ . The choice of τ affects the adaptability and computational cost of A-GAP, as discussed later.

Each node asynchronously runs a control cycle (see pseudo-code below) that starts with (2) the node polling each child c for the (distribution of) model variables [10]. Then, (3) the node selects a subset Ω of its children γ^n for re-computing their filters. The node computes the set of filters $(F^c)_c$ with $c \in \Omega$ by solving the local optimization problem (eq. 2) through a grid search in which the search space is limited to $[F^c-1, F^c+1]$ for all $c \in \Omega$, while observing the constraint that the filter of a child can not be wider than its parent's. Next, (5) the new accuracy objectives ε^c are computed for all the children. Finally, (7) the node computes its own model variables.

F1:

```

1 every  $\tau$  seconds
2   request model variables from all children  $\gamma^n$ 
3   select  $\Omega \subseteq \gamma^n$  children
4   compute new filters for  $\Omega$ 
5   compute new accuracy objectives for  $\gamma^n$ 
6   send new filters to  $\Omega$  and objectives to  $\gamma^n$ 
7   compute local variables

```

F2:

```

8 on message received from parent
9   if message type is new configuration
10      set new filter and accuracy objective
11   if message type is variables request
12      reply with local variables

```

A node n computes the accuracy objective ε^c for a child c as follows:

$$\varepsilon^c = \alpha \varepsilon^n \frac{E(E_{out}^c)}{E(E_{out}^n)}, \quad (3)$$

where $\alpha \in (0, 1]$ is a protocol parameter.

Handling topology changes and failures. When a new node joins or leaves the overlay, or a node fails, A-GAP follows the GAP design [1]. In such cases, GAP reconstructs the BFS

Fig. 4. Management overhead incurred by A-GAP (and ARC) as a function of the accuracy of the estimation

spanning tree and re-computes the partial aggregates. Note that, during changes, the root node continues to provide an estimate of the network variable. However, the accuracy of the estimation may be different from the objective during the transient period, until the model variables, which are dependent on a node's position in the tree, are refreshed on all nodes.

The two planes of A-GAP. A-GAP can be understood as operating in two different planes. On the data plane, updates of the local variables are propagated towards the root and filtering occurs. In this plane, information flows bottom-up, from the leaves towards the root.

In the control plane the filters are computed. In this plane, information flows in both directions, bottom-up and top-down. The model variables (step sizes and errors) flow from the leaves towards the root and are incrementally aggregated. The filters are distributed top-down, from the parents to their respective children.

4. Evaluation through Simulation

Simulation setup and scenario. We have evaluated A-GAP through extensive simulations using the SIMPSON simulator [4]. The results presented in the paper are based on simulating A-GAP for the topology of Abovenet [5], consisting of 654 nodes and 1332 links. The overlay topology is chosen to follow the physical Abovenet topology. Link speeds are set to 100 Mbps. The time to process an update at a node is 1 ms, and the communication delay is 4 ms. A-GAP is configured with $\tau=1$ seconds (local control cycle), $|\mathcal{Q}|=4$ (number of children receiving updated filters during a control cycle).

A local management variable in the simulation represents the number of HTTP flows entering the domain at a given router. The monitored aggregate thus corresponds to the number of HTTP flows in the domain. For these simulation runs, the aggregate is on average in the order of 20,000 flows. Local variables are asynchronously updated every second. The simulation traces that describe the behavior of the local variables have been produced using packet traces from two 1 Gbit/s links that connect University of Twente to a research network [11].

The results reported below correspond to a series of simulation runs with a measurement period of 200 seconds simulation time of the protocol in steady state. Before that, the simulation starts

Fig. 5. Error of the estimation as a function of the accuracy objective

with an initialization phase of A-GAP, which takes some 30 seconds, followed by the setting of the accuracy objective ε and a transient period of about 25 seconds.

The trade-off between estimation accuracy and protocol overhead. Fig. 4 shows the protocol overhead (i.e., the maximum number of processed updates across all nodes) as a function of the average error. Every measurement point in fig. 4 corresponds to a simulation run. As can be seen, the overhead decreases monotonically, as the error objective is increased. For small errors, the load decreases faster than for larger errors.

As expected, the overhead can be reduced by allowing a larger average estimation error. For example, compared to an error objective of 0 (which results in an achieved error of 4.5, see below), allowing an error of 3 flows (achieved error 5.4) reduces the load by 32%. An error of 10 flows (achieved error 10.7) reduces the load by 80%.

Let us mention in this context that the overhead incurred by A-GAP does not depend on the value of the aggregate, but on the changes of the aggregate as modeled by the step size variable.

Meeting the accuracy objective. From the simulation runs above, we evaluated the difference between the accuracy objective ε and the actual error e of the estimate at the root node for a period T . The estimation error e is computed as:

$$e = \frac{1}{T} \min_d \left\{ \int_0^T |A(t) - \tilde{A}(t+d)| dt \right\} \quad (4)$$

where A stands for the actual aggregate and $\tilde{A}$ for the estimation provided by A-GAP at the root node.

For all measurement points (fig. 5), we find that the measured error is at most about 4.5 above the error objective. We explain this by the fact that updates from different nodes in the network experience different delays for reaching the root. Since the model upon which filter re-computation is based does not consider networking and processing delays, the error of the estimate exceeds the error objective ε by a (small) margin $\delta=e-\varepsilon$. It is feasible to give a bound for δ in real-time, based on the evolution of local variables and the processing/communication delays, for the purpose of tuning the protocol.

Load distribution. Fig. 6 shows the distribution of the load (number of processed updates) over all nodes for a single run with

Fig. 6. Distribution of the management load over all nodes

an error objective of 10 flows. We observe that most nodes have a load below 10 updates per second. These are nodes with a small number of nodes in their subtree. At the same time, we find a few nodes with high loads around 100 updates per second. The most loaded node is generally the root of the tree.

The figure indicates that the load is unbalanced and that the root node experiences a high load compared to the median, although our protocol attempts to minimize the maximum load. This observation motivates further investigation.

Distribution of the estimation error. Fig. 7 shows the predicted error distribution computed by A-GAP and the actual error measured in a simulation run, for an error objective of 8. The vertical bars indicate the average actual error.

As we can see, the predicted error distribution is close to the actual distribution. More importantly, the distributions have long tails. While the average error in this measurement period is 9.7, the maximum error during the simulation run is 39 and the maximum possible error (that would occur in an infinite measurement period) is 104. Based on this observation, we argue that an average error objective is more significant for practical scenarios than a maximum error objective, as suggested by other authors (see [6][7][8]).

Benchmarking A-GAP against a rate-control scheme. Fig. 4 also compares the performance of A-GAP against an asynchronous rate-control scheme (ARC). As A-GAP, ARC uses a spanning tree and incremental aggregation to continuously estimate the aggregate of local variables. The control parameter for ARC is the minimum time interval $T_{\min}$ for a node to send an update to its parent.

In these and other experiments we performed [10], A-GAP incurred a lower overhead than ARC. As expected, both approaches perform similarly for very small estimation errors.

A rate-control approach that would permit to set update intervals individually for each node would probably perform better than ARC. In this context, it would be interesting to engineer a rate-control scheme that supports error objectives in the same way as A-GAP does, and then perform a comparative evaluation.

The cost of computing filters in A-GAP. The computational cost of computing filters in A-GAP depends on two parameters: the number of children $|\mathcal{Q}|$ whose filters are recomputed in each

Fig. 7. Distribution of the error predicted by A-GAP and the actual error at the root node

control cycle, and the control cycle period τ . We have evaluated the computational cost of A-GAP in function of $|\mathcal{Q}|$ for several of the above simulation runs. The evaluation was performed on an Intel Pentium 1.6 Ghz with 512 Mb of RAM. A-GAP has been written in Visual C++.

The result has been that, for all nodes, the computational costs for $|\mathcal{Q}| = 2, 3$, and 4 are below 40, 75 and 180 milliseconds, respectively. The computational cost increases exponentially with $|\mathcal{Q}|$ and, for this reason, the filters for only a limited number of children are re-computed during each control cycle. However, constraining $|\mathcal{Q}|$ limits, at the same time, the adaptability of A-GAP to a sudden change in network conditions.

The computational cost of computing filters is inversely proportional to τ . While an increase of τ reduces the computational cost, it reduces the adaptability of the system.

5. Discussion

In this paper, we introduced A-GAP, a protocol for continuous monitoring with accuracy objectives. A-GAP follows the push approach to monitoring and uses the concept of incremental aggregation on a spanning tree. A-GAP is decentralized and asynchronous, two key properties for achieving robustness and scalability.

The simulation results show that we can effectively control the trade-off between accuracy and overhead. A-GAP can reduce the overhead significantly (some 80% in our scenarios) when allowed a modest error in estimating an aggregate.

In A-GAP, accuracy is expressed in terms of the average error, which we argued to be more significant in practice than the maximum error, as suggested in the recent literature.

The protocol dynamically adapts to changes in network topology and to node failures, as well as to changes in the statistics that are computed from the local management variables. We consider management protocols that exhibit this kind of autonomic behavior essential for future communication systems.

A-GAP enables performance prediction at run-time. Based on our stochastic model, a manager can be provided with the distribution of the estimation error for the current network conditions and a given accuracy objective. The protocol can further provide

answers to what-if questions, as “What would be the expected overhead for an allowed error of ϵ , under the current network conditions?”.

Although we have used SUM as the only aggregation function throughout this paper, other aggregation functions, such as AVERAGE, MIN, and MAX can be supported with straightforward modifications. In general, aggregation functions which are composed of functions that are both commutative and associative, can be supported in A-GAP.

A-GAP assumes access to local variables. This can be realized through a variety of mechanisms including reading MIB variables and CLI commands for retrieving, e.g., Netflow caches as in our testbed [12]. The specific mechanism used is transparent to A-GAP.

How does A-GAP handle a sudden change in the global aggregate, as the consequence, e.g., of a malfunction in a key network component? In the worst case, the root node receives an update to the global aggregate whenever the change exceeds the current maximum estimation error. (The maximum estimation error is the sum of all filter widths.) This worst case relates to a scenario in which the sudden change is the result of all local variables changing at the same time so that, on all nodes, the local state reaches the local filter width. In the general case, the smaller the set of local variables causing the change, the smaller the change needs to be for the root to receive a new estimate of the global aggregate.

Our current work includes evaluating the scalability of A-GAP and implementing A-GAP on our lab testbed described in [12] to assess its effectiveness in the context of IP flow monitoring.

As for future work, we plan to evaluate various triggers for re-computing filter widths, which are periodically re-computed in the current design. This may permit A-GAP to adapt faster to changes in the networking conditions and to reduce its overhead. There is also room for improving the algorithm we currently use to solve the local optimization problem (eq. 2). Further, we will investigate which overlay topologies provide good performance properties for A-GAP.

Also, we plan to develop a version of A-GAP that does not use local filters but limits on a node’s update rates, in order to control the protocol overhead. Such an approach might enable us to better balance the load among the management processes.

Other open questions relate to the stochastic model that builds the bases for filter computation. First, we have not yet solved the global problem (eq. 1), which would give us a benchmark against which A-GAP could run. Second, the model assumes, e.g., independence among the local variables that are aggregated. Such an assumption does not hold in many networking scenarios. We would like to know how good the error estimations of A-GAP are for practical scenarios in which management variables are correlated.

Acknowledgments. The authors would like to thank György Dán and Mads Dam at KTH for fruitful discussions around the stochastic model and the design of A-GAP. This work uses network traces collected by the DACS group of the University of Twente for the protocol evaluation. This paper was supported in part by the EC IST-EMANICS Network of Excellence (#26854). This paper describes work undertaken in the context of the Ambient Networks –IST project, which is partially funded by the Commission of the European Union.

6. REFERENCES

- [1] M. Dam, R. Stadler, “A Generic Protocol for Network State Aggregation”, Radioteknisk och Kommunikation (RVK), Linköping, Sweden, 14-16 June, 2005.
- [2] S. Dolev, A. Israeli, and S. Moran, “Self-Stabilization of Dynamic Systems Assuming only read/write Atomicity”, *Distributed Computing*, 7:3-16, 1993.
- [3] K. Lim, R. Stadler, “A Navigation Pattern for Scalable Internet Management”, *IEEE/IFIP IM 2001*, Seattle, USA, 14-18 May, 2001.
- [4] K. Lim and R. Stadler. SIMPSON — a SIMPLE Pattern Simulator fOr Networks. <http://www.s3.kth.se/lcn/software/simpson.shtml>, 2006.
- [5] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP topologies with Rocketfuel”, *ACM/SIGCOMM*, 2002, Pittsburgh, USA, August 2002.
- [6] C. Olston, B. T. Loo and J. Widom, “Adaptive Precision Setting for Cached Approximate Values”, *ACM SIGMOD 2001*, Santa Barbara, USA, May 2001.
- [7] C. Olston and J. Widom, “Efficient Monitoring and Querying of Distributed, Dynamic Data via Approximate Replication”, *IEEE Data Engineering Bulletin*, March 2005.
- [8] Antonios Deligiannakis, Yannis Kotidis, and Nick Roussopoulos, “Hierarchical In-Network Data Aggregation with Quality Guarantees”, *EDBT 2004*, Crete, Greece, March 2004.
- [9] IEEE. ANSI/IEEE Std 802.1D, 1998 Edition. IEEE, 1998.
- [10] A. Gonzalez Prieto and R. Stadler, “Distributed Real-time Monitoring with Accuracy Objectives”, *KTH Technical Report*, May 2006. Available at: <http://www.ee.kth.se/~gonzalez>
- [11] R. van de Meent, A. Pras, Traffic Measurement Data Repository, University of Twente, <http://traffic-repository.ewi.utwente.nl/>, May 2006.
- [12] K. S. Lim and R. Stadler, “Weaver - realizing a scalable management para-digm on commodity routers”, In *IFIP/IEEE IM 2003*, Colorado Springs, USA, March, 2003.
- [13] G. Cormode et al., “Holistic aggregates in a networked world: distributed tracking of approximate quantiles”, In *ACM SIGMOD International Conference on Management of Data*, Baltimore, USA, June 2005.
- [14] 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.