Decentralized Real-time Monitoring of Network-wide Aggregates

Rolf Stadler, Mads Dam,
Alberto Gonzalez, Fetahi Wuhib
ACCESS Linnaeus Center
KTH Royal Institute of Technology
Stockholm, Sweden

stadler@ee.kth.se

ABSTRACT

The traditional monitoring paradigm of network and systems management, characterized by a central entity polling individual devices, is not adequate for today's large-scale networked systems whose states and configurations are highly dynamic. We outline principles for monitoring such new systems and stress the need for protocols that continuously monitor network-wide aggregates. To keep the overhead at acceptable levels, such protocols must be tunable, e.g., allow controlling the trade-off between accuracy and overhead. We describe and compare two of our efforts in developing protocols for decentralized monitoring of aggregates; one is based on spanning trees, the other on gossiping.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations – *Network Monitoring.*

General Terms

Algorithms, Management, Performance, Experimentation.

Keywords

Monitoring, Aggregation, tree-based protocols, gossip protocols

1. INTRODUCTION

Monitoring, i.e., the process of acquiring state information from a network or networked system, is fundamental to system operation. In traditional network and systems management, monitoring is performed on a per-device basis, whereby a monitoring station periodically polls devices in its domain for the values of local variables, such as device counters or performance parameters. These variables are then processed on the management station to compute an estimate of a network-wide state, which is analyzed and acted upon by other management programs. SNMP is probably the best-known protocol that supports this monitoring paradigm.

Over the past 20 years, this paradigm has proved fairly successful for networks of moderate size, whose configurations rarely

change and whose states evolve slowly and thus do not require intervention within seconds by an outside system. These assumptions however do not hold for many of today's networked systems. In the following, we outline our thoughts on a monitoring system for networks that are very large, whose configuration changes frequently, and whose state is highly dynamic and thus must be available at control points with short delay. While we address the issues from a networking perspective, we believe that the concepts put forward are applicable to a range of networked systems, including data centers where the devices of interest are not network elements but processor cards, for instance.

1. A self-organizing monitoring layer inside the managed system. To ensure scalability and fast reaction times, the processing associated with monitoring should be carried out inside the network if possible. We thus advocate research towards a light-weight, distributed management layer inside the network that offers end-to-end monitoring primitives to management applications and end systems outside the network.

2. Monitoring network-wide aggregates. The monitoring layer must provide estimates of aggregates of local variables in real-time. Such aggregates may be computed across nodes in a neighborhood, a network domain or the entire network. Simple examples of aggregates include sums, averages, extremal values, percentiles and histograms of device counters.

Aggregates contain information about the state of an entire system, as opposed to that of a single device, and many management applications depend on such data. For the purpose of quality assurance, for instance, it may be required to continuously track the number of VoIP flows in a network domain or the distribution of traffic composition across all links. Similarly, to achieve a given level of availability, it may be necessary to know, at all times, the percentage of links that operate above 50% utilization and to identify the 10 most loaded links.

3. Polling, continuous monitoring, threshold detection. Three basic primitives a monitoring layer must provide are distributed polling, continuous estimation, and threshold detection for network-wide aggregates. Polling, also referred as 1-time queries, gives a snapshot of the aggregate. Often, the evolution of the aggregate over time is of interest, which requires a different set of protocols that perform continuous monitoring. Detecting that an aggregate crosses a configured threshold is a basic means for anomaly detection. While such a primitive can be realized in a straightforward way through using a protocol for continuous monitoring, such an approach is hopelessly inefficient in large networks and an alternative method must be devised [12]. In order

to keep the complexity of the management layer low and to enable efficient, effective and scalable operation, all these protocols must be self-configuring, robust, and tunable at runtime.

4. Controlling the performance trade-offs in monitoring. In large-scale networks, continuous monitoring with maximum achievable accuracy of even a single aggregate can become unfeasible due to high traffic and processing overhead. In addition, modern routers contain hundreds of counters that are locally available for monitoring, many of which are needed in aggregated form to support autonomic management. Consequently, when designing monitoring protocols, the engineering trade-offs must be controllable at invocation or even at run-time. Monitoring protocols can be optimized towards providing estimates with low overhead, small delay, high accuracy, or high degree of robustness. As jointly optimizing these metrics is generally not possible, the right operating point in the parameter space created by these metrics must be chosen. For instance, recent results suggest that allowing for modest errors in estimating an aggregate can reduce the protocol overhead by an order of magnitude in a realistic setting [4]. Taking into account that different management applications have different requirements regarding the quality of the estimates (delay, accuracy, etc.), the operating point must be a control parameter of a protocol. By allowing a management application to change the operating point at run-time, monitoring functions can be built that adapt their operation to the required quality of monitoring data, which may change over time.

The best-known approach to computing aggregates in a distributed fashion involves creating and maintaining a spanning tree (in the monitoring layer) and aggregating state information along that tree, bottom-up from the leaves towards the root (e.g., [2][23][17][4]). Such a tree can be built in a decentralized, self-stabilizing manner, which provides the monitoring protocol with robustness properties. A second, less-studied approach involves the use of gossip protocols, which typically rely on randomized communication to disseminate and process state information in a network (e.g., [14][19][10][11]).

While both types of protocols execute on a network graph, which can be realized as an overlay in the monitoring layer described above, there are strong differences between tree-based and gossip-based aggregation. First, gossip-based aggregation protocols tend to be simpler as they do not maintain a distributed tree. Second, in tree-based aggregation, the result of an aggregation operation is available at the root node. In gossip-based aggregation however, estimates of the aggregate are available on all nodes, and they generally converge towards the true aggregate, for many network topologies, with an upper bound that is logarithmic in time and system size. Third, failure handling is very different for tree-based aggregation than for gossip-based aggregation. If a node fails, a tree-based aggregation protocol needs to reconstruct the aggregation tree. With gossip protocols, a node failure can produce so-called mass loss, which causes a bias in the aggregation process and needs to be corrected.

In the following, we report on our recent and ongoing work in continuous monitoring of aggregates. In the context of the above four principles, we have previously developed a protocol for distributed polling that is based on an echo algorithm [15] and are currently engineering protocols for threshold detection, both tree-based [12] and gossip-based [13].
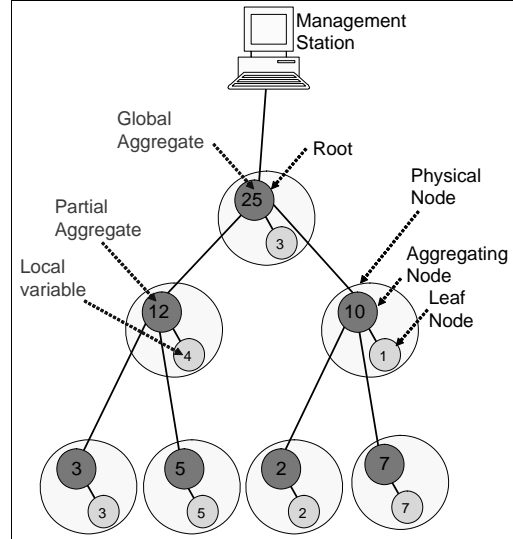

Figure 1. Aggregation tree with aggregation function

2. Monitoring using Tree-based Aggregation

In this section, we summarize some of our recent work on continuous monitoring with accuracy objectives, which aims at achieving an efficient and scalable solution. Specifically, we outline A-GAP, a generic aggregation protocol with controllable accuracy. It allows for continuously estimating aggregates of local variables by (i) creating and maintaining a self-stabilizing spanning tree and (ii) incrementally aggregating the variables along that tree. 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 on all nodes. 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. As a distinctive feature, the protocol provides an estimation of the error distribution at the management station in real-time. Furthermore, it supports a variety of control objectives, including percentiles and average (absolute) error, the latter of which is used in this paper. More details than given here can be found in [4][3].

2.1 System Architecture

The protocol assumes a distributed management architecture, whereby each network device participates in the monitoring task by running a management process, 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*. The topology of the overlay can be chosen independently from the topology of the underlying physical network. The protocol creates a spanning tree on this overlay, interconnecting all physical devices, such that each node on this spanning tree contains a leaf node and an aggregating node of the aggregation tree (see figure 1).

2.2 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

$e \in E(t) \subseteq V(t) \times V(t)$ appear and disappear over time. Each leaf n has an associated local variable $w_n(t)$, which is an integer-valued quantity. 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 (Σ denotes an aggregation function). 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 absolute error* of the estimate over time. We use here SUM as aggregation function. Other functions can be supported as well, as discussed [4].

2.3 Protocol Description

A-GAP, the protocol discussed in this section, is based on GAP (Generic Aggregation Protocol), which we developed in our earlier work [18]. 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). 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 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 a local variable. A-GAP inherits from GAP the functions of creating and maintaining the aggregation tree (specifically, handling node arrivals, departures and failures) and that of incremental aggregation.

A tree-based protocol like GAP can cause a high load on the root node and on nodes close to the root, specifically in large networks, since each change in a local variable creates a sequence of update messages along the path from a leaf node to the root. 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 ignores updates when only small changes to its partial aggregate are reported. To control the protocol overhead, A-GAP employs a filter scheme. One could develop a protocol similar to A-GAP, which employs a rate limitation scheme, by adapting the approach described in Section 2.4 from filter computation to rate computation. Note that, for both of these schemes, the overhead is reduced at the cost of introducing an error in estimating the aggregate.

2.4 Filter computation

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. The problem can then be stated as: Minimize $_{Max_n}\{\omega^n\}$ s.t. $E[|E^{root}|] \leq \varepsilon$, whereby ω^n and

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

We developed a stochastic model for the monitoring process, which is based on discrete-time Markov chains and describes individual nodes in their steady state [4]. 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 of updates n sends and the filter width F^n. At a leaf node, the change of the local variable over time is modeled as a random walk. The stochastic 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.

To solve the optimization problem, A-GAP employs a distributed heuristic, which maps the global problem into a local problem that each node solves in an asynchronous fashion. This way, each node periodically computes the local filters and (local) accuracy objectives for its children. A-GAP continuously estimates the step sizes in the leaf nodes for the random-walk model using a maximum likelihood estimator (MLE). Note that these step sizes are the only variables that the protocol estimates. All other variables are dynamically computed based on these estimates.

2.5 Evaluation Results

We evaluated A-GAP through extensive simulations and present here results from only two scenarios, related to a) controlling the trade-off between protocol overhead and estimation error and b) estimating the error distribution in real-time. Both scenarios share the following settings. The management overlay follows the physical topology of Abovenet, an ISP, with 654 nodes and 1332 links. Link speeds in the overlay are 100 Mbps. The communication delay is 4 ms, and the time to process a message at a node is 1 ms. The local management variable represents the number of HTTP flows entering the network at a given node, and thus the monitored aggregate is the current number of HTTP flows in the network. (In the Abovenet scenarios, the aggregate is in the order of 20.000 flows.) The local variables are updated asynchronously, once every second. The evolution of the local variables is simulated based on packet traces that were captured by the University of Twente at two of their network access points and then processed by us to obtain traces for all nodes in the simulation [4].

Figure 2 shows a result from the first scenario and shows the protocol overhead (i.e., the maximum number of processed updates across all nodes) in function of the experienced error. Every point in the graph corresponds to a simulation run. We observe that the overhead decreases monotonically as the estimation error increases. Consequently, the overhead can be reduced by allowing a larger estimation error, and the error objective is an effective control parameter. For example, compared to an error objective of 0 (which results in an experienced error of 4.5), an error objective of 2 flows (experienced error 5) reduces the load by 30%; an error objective of 20 flows (experienced error 21) leads to an 85% reduction in load.

Figure 3 relates to the second scenario and 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 one can see, the predicted error distribution is close to the actual distribution.
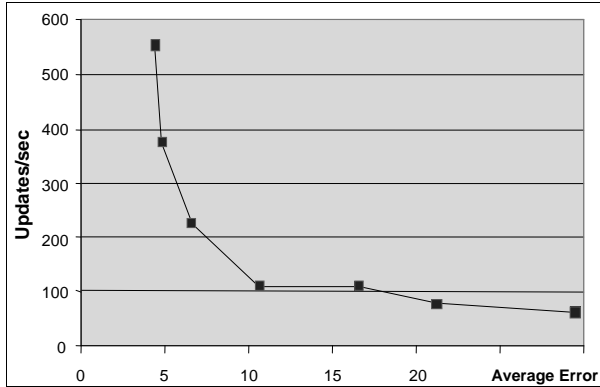
Figure 2. Protocol overhead incurred by A-GAP as a function of the experienced error e.

Figure 3. Distribution of the error predicted by A-GAP and the actual error at the root node

More importantly, the distributions have long tails. While the average error in this measurement period is 8.76, the maximum error during the simulation run is 44 and the maximum possible error (that would occur in an infinite measurement period) is 70.

We have implemented A-GAP and deployed it on a testbed of 16 commercial routers where it is used for monitoring IP flows [3]. The testbed measurements are consistent with the 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.

2.6 Related Work

Most current research in monitoring aggregates is carried out in the context of wireless sensor networks, where energy constraints are paramount and the objective is to maximize the lifetime of the network. Further, many recent works on monitoring the evolution of aggregates over time focus on n-time queries that estimate the aggregate at discrete times and are realized as periodic snapshots (e.g., [2][16]).

The trade-off between accuracy and overhead for continuous monitoring of aggregates has been studied first by Olston et al. who proposed a centralized monitoring protocol to control the trade-off [6][7].

The main differentiator between A-GAP and related protocols is its stochastic model of the monitoring process. This model allows for a prediction of the protocol performance, in terms of overhead and error, and for the support of flexible error objectives. In fact, all protocols known to us that allow controlling the trade-off between accuracy and overhead support only the maximum error as accuracy objective; in practical applications, however, the average error or a percentile error are often more useful metrics.

3. Monitoring using Gossip protocols for Aggregation

This section summarizes one of our recent results in the context of robust and scalable monitoring. Specifically, we report on a gossip protocol, G-GAP, which enables continuous monitoring of network-wide aggregates. Further, we provide an initial comparative assessment of G-GAP against GAP, a tree-based aggregation protocol (see Section 2.3), using simulation. Surprisingly, we find that the tree-based aggregation protocol
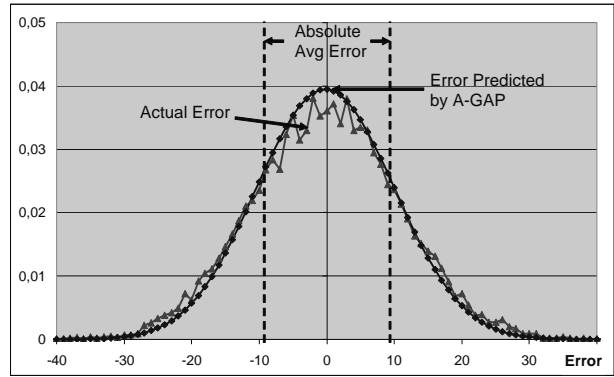
consistently outperforms the gossip protocol for comparative overhead, both in terms of accuracy and robustness. For further details, see [11].

3.1 Elements of the protocol design

G-GAP is based on push-synopsis, a gossip protocol for computing aggregates proposed by Kempe et al. [10]. The main extension to the push-synopses protocol is a scheme that renders the protocol robust to crash failures (except for cases where neighbors fail within short time of each other). The basic idea is that each node distributes recovery shares of its state to its neighbors and keeps track of its recently sent messages. These shares are then used to restore the protocol invariants (i.e., conserve the mass of the system) in case a node is deemed to have failed.

The convergence analysis for G-GAP reduces to that of push-synopses, as we can establish time bounds after which, in stable state, the behavior of G-GAP and push-synopses is identical.

G-GAP is assumed to execute on a network graph in the same architectural setting as described in Section 2.

3.2 Related work

Several approaches have been pursued in addressing node failures, and the associated problem of mass loss, with gossip-based aggregation. A straightforward approach is to restart the gossip protocol periodically [19]. While this does not prevent mass loss from occurring, it reduces the extent to which mass loss accumulates and thus mitigates estimation errors.

Mehyar et al. [20] recently presented a solution for computing the average of local values on a dynamically changing network graph. While there are similarities in the use of recovery information between Mehyar's protocol and G-GAP, there are major differences in the protocol design. First, failure recovery is an integral part of Mehyar's protocol, while it is a well-defined modular part of G-GAP that can be left out or replaced. Second, our protocol supports a range of aggregation functions, whereas the protocol in [20] has been designed for the computation of averages, and it is unclear to which extent it generalizes to other functions, such as SUM or other synopses.

An early example of a system addressing node failures is the Astrolabe monitoring system [21]. There, a tree-based scheme is used for aggregation, while a gossip protocol disseminates partial
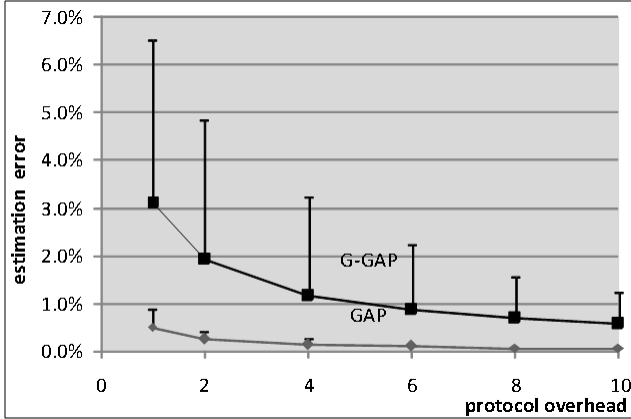
Figure 4: Estimation error vs. protocol overhead

Figure 5: Estimation error vs. failure rate

aggregates among peers in the tree hierarchy. Gossiping is used to achieve robustness, by allowing a pre-selected child node to instantly replace a parent node in case the latter fails. In this case the mass loss problem does not arise, since gossiping is not used for aggregation, but only for dissemination of information.

3.3 Comparative Evaluation to tree-based aggregation

We give simulation results for two scenarios, one focusing on controlling the trade-off between protocol overhead and estimation accuracy, the other scenario assessing the influence of the node failure rate on the estimation accuracy. We provide measurement results for both the gossip protocol G-GAP and the tree-based protocol GAP, which are compared for similar overhead. The overhead is measured in round rates. For G-GAP a round rate of 1 per sec means that the protocol executes one round per second, resulting in 1msg/sec per overlay link. For GAP a round rate of 1 per sec results in a maximum of 1msg/sec per overlay link.

In both scenarios, the network topology, the traces of the local variables and the simulation parameters are the same as or very similar to those given in Section 2. In contrast to the setup in Section 2 though, the overlay topology is generated by GoCast [8] with target connectivity of 10, which produces an average distance of 3.1 hops and a diameter of 4 hops in the overlay. Furthermore, the aggregation function is AVERAGE.

In the first scenario, we measure the estimation accuracy of G-GAP and GAP as a function of the protocol overhead. We run simulations for round rates of 1 to 10 messages per sec.

Figure 4 gives the results. Each measurement point corresponds to a simulation run. The top of the bars indicate the 90th percentile of the estimation error. As expected, for both protocols, increasing the round rate results in decreasing the estimation error. Therefore, the round rate controls the trade-off between estimation accuracy and protocol overhead. In addition, for comparable overhead (i.e. the same round rates), the average error in G-GAP is around 8 times that of GAP.

In the second scenario, we compare the estimation accuracy of G-GAP with that of GAP by measuring the estimation error as a function of the failure rate for a comparable overhead. We assume a failure de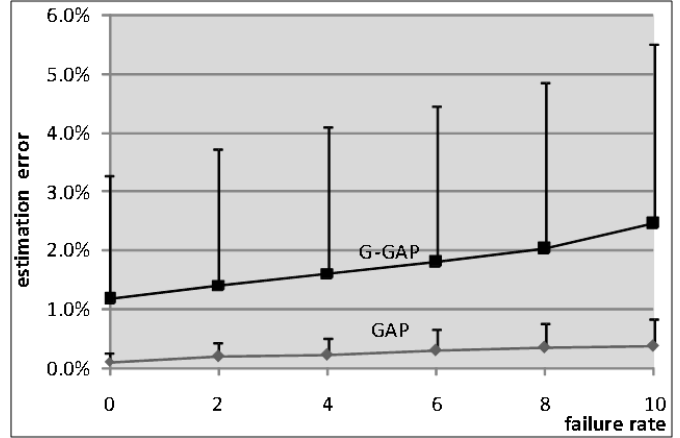tection service in the system that allows a node to detect the failure of a neighbor within 1 sec. We vary the failure rate from 0 to 10 node failures/sec. Failure arrivals follow a Poisson process, and they are uniformly distributed over all running nodes. A failed node recovers after 10sec.

Figure 5 shows the results. As can be seen from the figure, the error for both GAP and G-GAP increases with the failure rate. We also see that the slope is steeper and the spread is wider for G-GAP than for GAP. This result is surprising to us. We would have expected a gossip protocol to perform better, compared to a tree-based protocol, under high node failure rates.

4. Discussion

Research into efficient state aggregation under constraints has recently been proposed in different contexts (e.g., [1][22]). What makes the problem unique and interesting for the field at hand–network operations–are the constraints that are specific to a distributed monitoring layer, the rich functionality of network management operations, and the potentially large number of concurrently executing monitoring operations within such a layer.

The work presented in this paper is part of our effort towards a comparative assessment of tree-based vs. gossip-based approaches to distributed real-time monitoring, which we believe to be of significant practical relevance. We are interested in how typical protocol representatives from both approaches compare performance-wise when certain defining parameters in network scenarios are varied, such as the sampling frequency of the local variables, the network size, the rate of node failures, etc. Such comparisons are challenging, as it is difficult to generalize simulation results beyond particular scenario setups and measured parameter ranges.

An aspect that merits in-depth study in our opinion is the impact of the network graph (i.e., the overlay topology) on the performance of a monitoring protocol. Results are available on how network graphs determine convergence properties of gossip protocols (e.g., [10][5]). From an engineering perspective, it is of interest to identify topological parameters, such as the node degree, as control parameters for controlling protocol trade-offs. This would make it possible, for instance, to initialize a protocol, together with its network graph, with the desired operating point.

The example of A-GAP demonstrates the advantages of having a model of the monitoring process and being able to dynamically

compute model variables during protocol execution, e.g., to provide, in real-time, estimates of the protocol's performance (overhead and estimation error in the case of A-GAP). For the problem to become tractable in practice and solvable with feasible overhead, simplifying assumptions must be made. In the case of A-GAP such assumptions include the independence of local variables and the omission of communication and computing delays. We showed that such simplifications can be justified within the parameter ranges of our evaluation.

5. References

[1] A Giridhar, PR Kumar: "Towards a Theory of In-Network Computation in Wireless Sensor Networks," IEEE Communication Magazine, April 2006.

[2] A. Deligiannakis, Y. Kotidis and N. Roussopoulos, "Hierarchical in-Network Data Aggregation with Quality Guarantees,", In Proc. 9th International Conference on Extending Database Technology (EDBT'04), Heraklion – Crete, Greece, March 14-18, 2004.

[3] A. Gonzalez Prieto and R. Stadler: "Monitoring Flow Aggregates with Controllable Accuracy," 10th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS 2007), San José, California, USA, Oct 31 - Nov 2, 2007.

[4] A. Gonzalez Prieto, R. Stadler: "A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives", IEEE Transactions on Network and Service Management (TNSM), Vol. 4, No. 1, June 2007.

[5] A. Olshevsky, J.N. Tsitsiklis. "Convergence rates in distributed consensus averaging,"45th IEEE Conference on Decision and Control (CDC 06), San Diego, CA, Dec 08.

[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, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams", ACM SIGMOD 2003, San Diego, USA, June 2003.

[8]   C. Tang, and C. Ward, "GoCast: Gossip-Enhanced Overlay Multicast for Fast and Dependable Group Communication," In Proc. International Conference on Dependable Systems and Networks (DSN'05), Yokohama, Japan, June 28 - July 1,2005.

[9] D. Jurca, R. Stadler: "Computing Histograms of Local Variables for Real-Time Monitoring using Aggregation Trees," 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), Long Island, NY, June 1-5, 2009.

[10] D. Kempe, A. Dobra and J. Gehrke, "Gossip-Based Computation of Aggregate Information," In Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03), Cambridge, MA, USA, October 11-14, 2003.

[11] F. Wuhib, M. Dam, R. Stadler, A. Clemm: "Robust Monitoring of Network-wide Aggregates through Gossiping," 10th IFIP/IEEE International Symposium on Integrated Management (IM 2007), Munich, Germany, May 21-25, 2007.

[12] F. Wuhib, M. Dam, R. Stadler: "Decentralized Detection of Global Threshold Crossings Using Aggregation Trees," Computer Networks, Vol. 52, No. 9, pp 1745-1761, 2008.

[13] F. Wuhib, M. Dam, R. Stadler: "Gossiping for Threshold Detection," 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), Long Island, NY, June 1-5, 2009.

[14] K. Birman, "The promise, and limitations, of gossip protocols, "ACM SIGOPS Operating Systems Review archive, Volume 41, Issue 5, October 2007.

[15] K.S. Lim and R. Stadler, "Real-time Views of Network Traffic Using Decentralized Management," 9th IFIP/IEEE International Symposium on Integrated Network Management (IM'2005), Nice, France, May 16-19, 2005.

[16] 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.

[17] M. A. Sharaf, J. Beaver, A. Labrinidis and P. K. Chrysanthis, "Balancing energy efficiency and quality of aggregate data in sensor networks," The International Journal on Very Large Data Bases, vol. 13, issue 4, pp. 384-403, December 2004.

[18] M. Dam and R. Stadler: "A generic protocol for network state aggregation," RVK 05, Linköping, Sweden, June 14-16, 2005.

[19] M. Jelasity, A. Montresor and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," ACM Transactions on Computer Systems, vol. 23, Issue 3, pp. 219-252, August 2005.

[20] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, R. Murray, "Asynchronous Distributed Averaging on Communication Networks," IEEE/ACM Transactions on Networking, August 2007.

[21] R. van Renesse, K. Birman, and W. Vogels, "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring," ACM Transactions on Computer Systems, Vol. 21, Issue 2, pp.164-206, May 2003.

[22] S. Keshav: "Efficient and Decentralized Computation of Approximate Global State," ACM SIGCOMM CCR, Jan 2006.

[23] S. Madden and M. Franklin and J. Hellerstein and W. Hong, "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks," Fifth Symposium on Operating Systems Design and Implementation (USENIX - OSDI '02), Boston, MA, USA, December 9-12, 2002.