# Gossiping for Threshold Detection

Fetahi Wuhib, Rolf Stadler
School of Electrical Engineering
KTH Royal Institute of Technology
Stockholm, Sweden
{*fetahi,stadler*}*@kth.se*

Mads Dam
School of Computer Science and Communication
KTH Royal Institute of Technology
Stockholm, Sweden
*mfd@kth.se*

***Abstract*—We investigate the use of gossip protocols to detect threshold crossings of network-wide aggregates. Aggregates are computed from local device variables using functions such as SUM, AVERAGE, COUNT, MAX and MIN. The process of aggregation and detection is performed using a standard gossiping scheme. A key design element is to let nodes dynamically adjust their neighbor interaction rates according to the distance between the nodes' local estimate of the global aggregate and the threshold itself. We show that this allows considerable savings in communication overhead. In particular, the overhead becomes negligible when the aggregate is sufficiently far above or far below the threshold. We present evaluation results from simulation studies regarding protocol efficiency, quality of threshold detection, scalability, and controllability.**

## I. Introduction

Threshold crossing alerts (TCAs) indicate to a management system that a monitored management variable, for instance a MIB object, has crossed a preconfigured value—the threshold. Variables that are monitored for TCAs typically contain performance-related data, such as link utilization or packet drop rates. In order to avoid repeated TCAs in case the monitored variable oscillates, a threshold $T^{g+}$ is typically accompanied by a second threshold $T^{g-}$ called the hysteresis threshold, set to a lower value. The hysteresis threshold must be crossed, in order to clear the TCA and allow a new TCA to be triggered when the threshold is crossed again (see fig. 1).

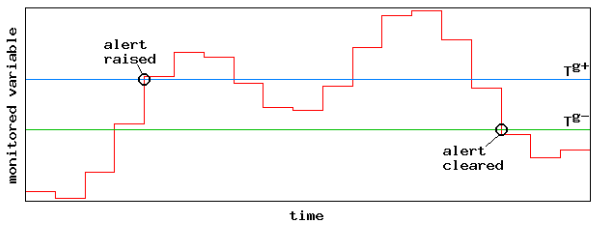


Fig. 1. Threshold Crossing Alerts: an alert is raised when a monitored variable crosses a given threshold $T^{g+}$ from below. The alert is cleared when the variable crosses a lower threshold $T^{g-}$ from above.

In this paper, the management variables monitored for TCAs are network-wide aggregates. They are computed from local device variables across the network (or across a network domain). Examples of such aggregates include the average link utilization, the current resource consumption by p2p traffic, or the total number of VoIP flows in the network.

Traditionally, aggregating local variables from individual devices has been performed in a centralized way, whereby an application, running on a management station, first retrieves local variables from agents in network elements and then aggregates the values on the management station. To avoid the well-known drawbacks of this approach—poor scalability and fault tolerance—this work follows a distributed approach to detecting threshold crossings. We assume that each network device participates in the monitoring activity by running a management process, either internally or on an external, associated device. These management processes communicate via an overlay network for the purpose of monitoring for threshold crossings. Throughout the paper, we refer to this overlay as the network graph. A node in this graph represents a management process together with its associated network device.

In order to monitor an aggregate for threshold crossings, a mechanism is needed that estimates the current value of the aggregate. Such a mechanism is usually realized in two ways: (1) using a tree-based protocol whereby the aggregate is computed with the help of a spanning tree whose nodes are the above-mentioned management processes (e.g.,[1], [2]), and (2) using a gossip protocol whereby each node on the network graph maintains an estimate of the global aggregate by means of periodically exchanging state information with its neighbors (e.g., [3], [4], [5]).

This paper focuses on gossip protocols for detecting threshold crossings of aggregates. While some tree-based protocols have been proposed for this purpose (e.g., [6], [7]), no work on gossip protocols has been reported to date, and this paper thus presents the first results.

A key idea is to dynamically throttle neighborhood interaction rates for nodes whose local estimate of the aggregate is far from the threshold. This allows a threshold detection protocol to run with a very low overhead in typical situations where the global aggregate is far from the threshold. On the other hand, when the global aggregate approaches the monitored threshold, the protocol temporarily adjusts to higher rates needed to obtain good accuracy in TCA generation.

For many network management applications such a behavior is highly beneficial. For example, when the monitored quantity reflects some kind of critical network property, focusing management traffic to the point of entering a critical state allows

This work has been part of 4WARD, a $7^{th}$ Framework ISP project funded by the EC. It has further been supported by the ACCESS Linnaeus Center at KTH.

to save management overhead both under normal operation (when the aggregate is far from the threshold), as well as once the critical state has been entered—i.e. when the threshold has been crossed. Also, when threshold crossing is a rare event, it allows a management application to statistically multiplex the monitoring of multiple uncorrelated TCAs.

The difficulty is to engineer such message rate adaptation schemes while obtaining good performance, i.e. short detection times, and low probabilities for false positives and false negatives. In this paper we examine the design space for gossip-based protocols augmented with a rate adaptation scheme using push-synopses, a gossip protocol for computing aggregates introduced by Kempe et al. [3], as the baseline.

This research is part of our agenda of performing a comparative assessment of tree-based vs. gossip-based approaches to threshold detection (and, more generally, to real-time monitoring). We have recently developed a tree-based protocol for threshold detection with the same design goals as the protocols given in this paper [7]. While we briefly describe our current understanding on how the tree-based and gossip protocols compare in terms of performance, a thorough comparison is planned for future work.

The paper is organized as follows. Section II presents related work. Section III presents the objective for and the design space of a gossip-based protocol for detecting threshold crossing. Section IV presents protocols selected from this design space. Section V presents the results of the experimental evaluation of some of the protocols. Finally, section VI discusses the results and concludes the work.

## II. Related Work

The traditional approach to the detection of threshold crossings of network-wide aggregates is using an aggregation protocol to continuously compute the aggregate on a node and to evaluate on that node the threshold conditions every time the aggregate is updated. Several results, both centralized ([8], [9], [10], [11]) and decentralized ([7], [6]), that improve on this approach have been published recently. The common goal is achieving efficiency by reducing protocol overhead, compared to the traditional approach, when the aggregate is far from the threshold. In general, local thresholds define conditions under which nodes refrain from sending updates of their local states, thereby reducing the protocol overhead.

In a recent work [7], we proposed the protocol TCA-GAP which augments a spanning tree-based aggregation protocol with a rate adaptation scheme for TCA generation. The key idea is to recursively assign local thresholds to subtrees, and to introduce a scheme for reassignment of these local thresholds once the threshold conditions are violated.

In this paper, we use *gossip protocols* for detecting threshold crossings of network-wide aggregates. Gossip protocols, also known as epidemic protocols, are protocols that generally execute in periodic rounds. They can be characterized by asynchronous and often randomized communication among nodes in a network ([12], [3]). Originally, they have been proposed for disseminating information in large dynamic environments [12], and, more recently, they have been applied to various other tasks, including the construction of robust overlays ([13], [14]), estimation of network size [15], and computation of network-wide aggregates ([3], [5], [4], [16], [17]).

## III. Objective and Protocol Design

We are considering a network graph $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$ with nodes $i \in \mathcal{V}(t)$ and edges/links $e \in \mathcal{E}(t) \subseteq \mathcal{V}(t) \times \mathcal{V}(t)$. To each node $i$ is associated a local state variable (or local variable) $x_i(t) \geq 0$ that represents the quantity whose aggregate is being subjected to threshold monitoring. The local variables are time-invariant if each $x_i(t)$ is a constant function. We assume that local variables are *non-negative real-valued* quantities.

### A. Design Goals

The objective is to engineer a protocol that raises an alert on a distinguished node, referred to as the *root node*, whenever the aggregate $\mathcal{F}_i x_i(t)$ crosses a given global threshold $T^{g+}$ from below, and to clear the alert when the aggregate crosses a corresponding lower threshold $T^{g-}$ from above (see fig. 1). $\mathcal{F}$ denotes a generic aggregation function. Aggregation functions we consider in this context include AVERAGE, SUM, COUNT, MAX and MIN.

The main design criteria are:

- *Efficiency*: The communication and processing overhead of the protocol should be small, specifically during periods where the aggregate is far (above or below) the threshold.
- *Quality of detection*: The protocol should achieve short delays for detecting threshold crossings, and false positives and false negatives should be rare.
- *Scalability*: The protocol should allow for efficient operation with high quality of detection in large networks with at least thousands of nodes.
- *Controllability*: The protocol should allow for controlling the tradeoff between quality of detection and protocol overhead through management parameters that can be adjusted at runtime.

### B. The Design Space

The key idea in adapting a gossip-based aggregation protocol to threshold detection is to dynamically adjust the message rate of a node (the rate at which a node communicates updates of its local state) according to the distance of its local estimate of the aggregate from the current threshold. To develop this idea three largely orthogonal problems need to be resolved:

1) What is the underlying mechanism for dynamic rate adjustment?
   - N) No rate adjustment. This is used as a reference point to which the refined protocols are compared.
   - R) Rate reduction. Rates for nodes with aggregates far from the threshold are reduced, but not completely eliminated. This ensures that the estimate of the aggregate on all nodes is updated, at least with the reduced rate.

S) Rate suppression. Nodes with aggregates far from the threshold completely refrain from sending updates to their neighbors. Care must be taken to ensure that nodes, that have stopped interaction with their neighbors in this way, would not cause estimation errors in the aggregate that could result in false positives or false negatives.

2) How are TCA's triggered and cleared? Triggering TCA's on the basis of the local aggregates only is likely to need some amount of filtering to obtain acceptable levels of false positives and false negatives. The problem is to eliminate local bias: For example, a sudden jump in the local variable can produce, for a short amount of time, a significant error in the local estimate of the aggregate. This problem is common to gossip protocols used for continuous monitoring. We consider three options:
   N) No filtering. This is again used as a reference point.
   L) Local filtering. The local aggregates are low-pass filtered before being used for TCA generation. In this paper we use a simple policy of requiring the local aggregate to have crossed the threshold for a minimum number $wait_{max}$ rounds before a TCA is generated. A large value of $wait_{max}$ has the effect of reducing the probability of false positives while causing threshold crossings of short duration to remain undetected and resulting in an increase in the detection delay.
   G) Global snapshot. The gossip mechanism is used as a trigger for a global snapshot computation. That is, the local detection of a threshold crossing at the root node triggers a computation of the snapshot of the monitored aggregate using a distributed polling algorithm that gives an accurate estimate of the aggregate in a short period. An Echo algorithm [18], [19] can be used for this purpose. Alarms are raised or cleared based on the accurate estimate that the snapshot provides, hence eliminating false positives. The local detection of a threshold crossing can be done though local filtering discussed above. In this case, a smaller value of $wait_{max}$, compared to the simple local filtering, can be chosen since false positive is not an issue any more. Note that the per-round overhead of the baseline protocols is typically similar to that of Echo: in either case two messages traverse each link, one in either direction.

3) How is the duality of upper and lower threshold crossings exploited? The assumption of a distinguished root node allows a relatively simple solution to this problem, using TCA's detected at the root node to trigger mode switches, and using the gossiping infrastructure to propagate these mode switches throughout the network. We consider two alternatives:
   N) No mode switching. Only threshold crossings in one direction is considered. We include this option mainly for presentation purposes.
   M) Mode switching. The main concern is to ensure global agreement of modes (i.e., which hysteresis threshold to monitor) at all times since inconsistent mode assignment in the presence of rate adaptation can cause local estimates to diverge arbitrarily from the true aggregates.

We use the following naming convention for the protocols. A protocol name is composed of three characters. The first character represents the rate adjustment mechanism used (N,R or S), the second the triggering mechanism (N,L or G) and the third mode switching (N or M). The character 'x' is used as a wildcard.

Altogether we identify a design space of 18 protocols, ranging from a baseline design NNN using a baseline protocol for detecting threshold crossings in one direction without filtering, to the most complex design SGM, implementing a rate suppression scheme using global snapshots for TCA generation, with mode switching. Table I lists all protocols with mode switching.

| | | Rate adjustment | | |
|---|---|---|---|---|
| | | None | rate Reduction | rate Suppression |
| TCA triggering | No filtering | NNM | RNM | SNM |
| | Local filtering | NLM | RLM | SLM |
| | Global snapshot | NGM | RGM | SGM |

TABLE I
LIST OF PROTOCOLS WITH MODE SWITCHING.

In the following section we explore some of this design space, concentrating on three points in this space which we have found most interesting, namely the protocols NNM, RxM and SxM.

## IV. PROTOCOL SCHEMES

In this section we detail the protocol design space that was presented in the previous section, in terms of pseudocode. The protocols are presented for the aggregation function AVERAGE. An adaptation of the aggregation function to SUM, COUNT and linear synopses is straightforward [3]. For time-invariant input, MAX/MIN is straightforward, and for time-varying input MAX/MIN can be approximated by a log-sum-exp approximation (e.g., MAX$x_i \log\left(\sum_{i=1}^{n} e^{x_i}\right)$), which then can be computed in a straightforward way using the protocols.

We give the protocols for a partially asynchronous network model whereby nodes communicate with each other via message passing and where communication and processing delays are bounded. All nodes execute the protocol in rounds using local, non-synchronized clocks. We assume that there is one *root* node in the network (possibly elected by some *(*leader election) algorithm [18]), and that all nodes know $T^{g+}$ and $T^{g-}$.

### A. The Baseline Protocol

The baseline protocol shown in fig. 2 is a straightforward extension of 'push-synopses' [3]. The push-synopses protocol is an instance of a general iterative update scheme which has been examined by Tsistiklis et al. (cf. [20], [21]). Push-synopses results from the baseline protocol by dropping the initialization of the `sign` variable and dropping step 3 in each iteration. The choice of push-synopses is not critical: other baseline protocols can be used without substantially affecting the conclusions of this paper.

In [21], it is shown that for a time-invariant matrix $A = (\alpha_{i,j})$ (see step 4 of fig. 2), mild assumptions on $A$ suffice to ensure polynomial-time convergence of push-synopses to the average, and [3] shows logarithmic convergence of push-synopses for the special case of complete graphs and uniform gossip (i.e., $\alpha_{i,i} = \alpha_{i,j} = 0.5$ for $j \neq i$ randomly chosen).

```
round 0 {
  1) s_i=x_{i,0}; w_i=1; sign=1
  2) send (s_i, w_i) to self
}
round r > 0 {
  1) let {(s_l^*, w_l^*)} be all pairs sent to i
     during round r
  2) s_i=(x_{i,r} − x_{i,r−1}) + Σ_l s_l^*; w_i=Σ_l w_l^*
  3) if root() && sign * s_i/w_i > sign*th(sign)
     { raise_alert(sign); sign *= −1 }
  4) choose shares α_{i,j} ≥ 0 for all nodes j
     such that Σ_j α_{i,j} = 1
  5) for all j|α_{i,j} ≠ 0 send (α_{i,j} * s_i, α_{i,j} * w_i)
     to j
}
```

Fig. 2. The baseline protocols NNM: pseudocode for node $i$

The protocol works as follows. In round 0, nodes initialize their local state variables as shown in fig. 2. Then, for all later rounds nodes aggregate shares they have received in the previous round to update their state. Step 3 is executed only by the root node which raises an alert and switches mode if a local threshold crossing event is determined to have taken place. Finally, in steps 4 and 5, a node partitions its local state into shares and transmits these shares to its neighbors and itself. The choice of coefficients $\alpha_{i,j}$ is subject to the network constraint that $\alpha_{i,j} \neq 0$ only if there is a link between nodes $i$ and $j$. Note that we generally assume the network adjacency matrix to be reflexive, i.e. all nodes are linked to themselves.

The variable `sign` determines the current mode: 1 means that there is no alert and the protocol would raise alert when the aggregate grows above the threshold while -1 means that there is an alert and the protocol clears the alert when the aggregate falls below the threshold. The function `th(`*sign*`)` returns $T^{g+}$ or $T^{g-}$ according to the value of *sign*. The function `root()` returns true on the distinguished node and false everywhere else.

### B. Rate Reduction

The efficiency of the baseline protocol of fig. 2 can be improved by reducing the message rate for nodes whose local estimate of the aggregate is far from the threshold. This can be done in a variety of ways. Here we propose, as an example, a binary approach: when the local estimate of the aggregate (of the node itself, or one of its neighbors) is sufficiently close to the threshold, messages are transmitted at the normal rate (i.e. each round). When the local estimates are far from the threshold, the message rate is reduced by a constant factor $delay_{max}$. Whether the aggregate is far from the threshold $T$ or not (the predicate `close` in fig. 3 below) is determined by a factor $k \in [0, 1]$ such that, for positive values of *sign*, the local estimate $a$ is close to $T$, if $a \geq k * T$, and for negative values of *sign*, if $a \leq T/k$.

```
round 0 {
  1) s_i = x_{i,0}; w_i = 1; delay = delay_max; sign =
     1
  2) send (s_i, w_i) to self
}
round r > 0 {
  1) let {(s_l^*, w_l^*)} be all pairs sent to i
     during round r
  2) s_i = (x_{i,r} − x_{i,r−1}) + Σ_l s_l^*; w_i = Σ_l w_l^*
  3) if root() && tc(s_i/w_i, th(sign), sign)
     raise_alert(sign)
  4) if ∃l.close(s_l^*/w_l^*, th(sign), sign)
     delay = 0 else delay−−
  5) choose shares α_{i,j} ≥ 0 for all nodes
     j such that Σ_j α_{i,j} = 1
  6) if delay==0 then {for all j|α_{i,j} ≠ 0
     send (α_{i,j} * s_i, α_{i,j} * w_i) to j;
     delay=delay_max}
     else send (s_i, w_i) to self.
}
```

Fig. 3. RxN: Rate reduction, positive mode. Pseudocode for node $i$

Fig 3 presents the pseudocode for this protocol. The function `tc` implements an alert triggering mechanism (e.g., unfiltered local (N), filtered local (L), or global snapshot (G): see section III-B) for the given estimate of the aggregate ($\frac{s_i}{w_i}$), threshold (`th(`*sign*`)`) and sign (*sign*).

In step 4) of fig 3, operation at the high rate is determined not by the ratio $s_i/w_i$ but by $s_l^*/w_l^*$ 'closest' to the threshold. The consequence of this is that all neighbors of a node with an estimate close to the aggregate operate at high rates, allowing a faster convergence of the aggregate.

Note that the protocol of fig. 3 is given for the case of positive signs only. In the next section we consider the problem of propagating sign changes to allow dual mode operation.

### C. Dual Mode Operation

Global agreement about the current mode must exist among all nodes in the network for dual mode operation to produce good performance. For instance, for the rate reduction protocol

fig. 3, if the root node has positive sign, an increase in the local estimate of some other node $i$ should cause the node to run at a higher rate so that this increase would be propagated faster to the root. However, if the sign of node $i$ is negative, it causes $i$ to operate at the slower rate, delaying the detection of a possible threshold crossing. Several strategies are possible for mode switching. The design choice made in this paper is to let threshold crossings detected at the root node control mode switching throughout the network. In fig. 4 we highlight the changes needed to be made to the rate-reduction protocol in fig. 3 to implement mode switching. ... represents part of the pseudocode of fig. 3 that does not change.

```
round 0 {
  1) ...; switched = 0
  2) ... (s_i, w_i, sign) ...
}
round r > 0 {
  1) ... (s*_l, w*_l, sign*_l) ... triples ...
  2) ...; tsgn = sgn(sign*_l, sign)
  3) if root() && tc(s_i/w_i, th(sign), sign)
     { raise_alert(sign); sign *= -1;
     switched = 1 }
     if not(root()) && tsgn ≠ sign
     { sign *= -1; switched = 1 }
4), 5) ...
  6) if delay == 0 or switched then ...
     (α_{i,j} * s_i, α_{i,j} * w_i, sign) ...
     ... (s_i, w_i, sign) ...
  7) switched = 0
}
```

Fig. 4. RxM: Rate reduction, dual mode. Pseudocode for node $i$

The protocol uses the auxiliary function sgn which either returns 1 or $-1$ based on the current sign and the sign variables received from neighbors. If a sign received from neighbors is different from the current sign then it returns the sign from neighbors. Otherwise it keeps the current sign unchanged. Note that a necessary and sufficient condition for this solution to work correctly under all operating conditions is that all mode switches initiated by the root are separated by a minimum of $d$ rounds where $d$ is the diameter of the network. A simple way of enforcing this condition is by using local filtering (L) for triggering of the TCAs and using $wait_{max} > d$.

### D. Rate Suppression

As discussed in section III-B, a more efficient alternative to rate reduction is to completely suppress message exchange for nodes with aggregates that are far from the threshold. Such a protocol replaces lines 4)–6) in fig. 4 by the corresponding lines in fig. 5. The main idea is that nodes fully participate in message exchanges (i.e., become *active*) only if either their local estimate of the aggregate is close to the threshold, or the mode has just switched. Otherwise, they exchange messages only with those neighbors that are close to the threshold. Note

```
4) active = close(s*_i/w*_i, th(sign), sign)
5) if active or switched
   { choose α_{i,j} ≥ 0 for all nodes j
   such that Σ_j α_{i,j} = 1 }
   else
   { choose α_{i,j} ≥ 0 whenever j = self
   or close(s*_j/w*_j, th(sign), sign), and
   let α_{i,j} = 0 otherwise }
6) for all j|α_{i,j} ≠ 0 send
   (α_{i,j} s_i, α_{i,j} w_i, sign) to j
```

Fig. 5. SxM: The rate suppression protocol with dual mode. Pseudocode for node $i$

that, similar to the RxM protocol, this allows activeness (i.e., the property that nodes do not suppress update messages) to spread faster to nodes that are not active, allowing them to participate in the computation of the aggregate, hence allowing for a faster convergence of the aggregate.

## V. Evaluation

We have evaluated key points in the design space outlined above through simulation using SIMPSON [22], a discrete event simulator that allows us to simulate message exchanges over large network topologies and message processing on the network nodes. (The key reason for choosing SIMPSON in this study over one of the popular network simulators like NS2 has been its suitability for simulating large networks.) Here, we present simulation results from various scenarios where we evaluate the efficiency, the quality of threshold detection, scalability with respect to the number of nodes and the controllability of the protocols. All simulation studies for this paper have been performed using real traces or derivatives of real traces to simulate the local variables.

### A. Simulation setup and evaluation scenarios

*1) Evaluation metrics:* We evaluate the protocols using the following metrics. First, we measure the protocol overhead as the average number of messages processed/sec/node (Note that this value is fixed for baseline NNx protocols and depends only on the overlay graph and the protocol round rate). Second, we evaluate the quality of threshold crossing detection by measuring the detection delay and accuracy of detection. The detection delay is measured as the difference between the time the protocol reports a crossing and the time the actual crossing occurs, as explained below. The accuracy of detection is measured by computing the fractions of false positives (alerts raised by the protocol without no corresponding actual alerts occurring) and false negatives (cases where the protocols fail to raise alerts when actual crossings have occurred). In graphs illustrating the measurement results, 95% confidence intervals are given wherever appropriate.

*2) Local variables:* In all scenarios, a local variable represents the number of HTTP flows that enter the network at a specific router, and the aggregate of those variables represents the average number of such flows in the network. We simulate the behavior of the local variables using packet

traces captured at the University of Twente [23]. The first trace, which we call the UT trace, has been created as follows. We sampled every second the number of HTTP flows from those original traces, which produced traces that give the evolution of the number of HTTP flows over time. Then, we divided the new traces into segments of 150sec each. From those segments, we constructed traces of 1500sec for each node in the simulation, by randomly selecting and concatenating ten of those segments. Across all traces, the average value of the local variables is about 45 flows, and the standard deviation of the change between two consecutive samples is about 3.4 flows.

The second trace, which we call Periodic UT trace, is obtained by adding a sinusoidal bias to the UT trace $w_i(t)$ on a node $i$ as $w_i(t)^* = int\left(w_i(t) + 23 * \left(1 + \sin(\frac{2\pi t}{30} - \frac{\pi}{2})\right)\right)$ where $int()$ returns the integer component of its argument. In our simulations, we use the UT trace to study the behavior of the protocols in scenarios where no threshold crossing occurs, while we use the Periodic UT trace to induce multiple, clearly defined threshold crossings.

*3) Overlay topology:* The topologies used for the network graphs in our simulations are generated by GoCast [14], a gossip protocol that builds topologies with bidirectional edges and small diameters. The protocol allows setting the (target) connectivity of the graph. Unless stated otherwise, the topology used in the simulations has 654 nodes.(This number is chosen so that our results here are directly comparable with our earlier work on threshold detection [7]). All topologies are generated with a target connectivity of 5, which, for the 654 node topology we use, produces an average internode distance of 4.3 hops and a diameter of 7 hops in the graph.

*4) Other Simulation Parameters:* We run the simulations with the following default parameters unless stated otherwise:

- Aggregation function: AVERAGE
- Maximum message rate: 4 msg/sec per link.
- Protocol parameter: k=0.9
- Processing overhead: 1ms/message
- Network delay across links of the graph: 5ms
- Length of a simulation run: 1500sec, with an initialization period of 10sec
- Threshold values: $T^{g+}$ is set at 1.05 times the average value of the aggregate and $T^{g-}$ at the average value of the aggregate.
- $wait_{max}$=5 for triggering through local filtering and $wait_{max}$=2 for triggering through global snapshot.

The above values are set based on experience with our testbed [24], internet measurements, and the need for a sufficient number of measurement events to obtain statistically significant simulation results.

### B. Protocol efficiency

In this scenario, we assess the efficiency of the SGM and NNM protocols by measuring the protocol overhead in a scenario where several threshold crossings occur. We run the protocols on the 654-node network graph where the local variables change according to Periodic UT trace. The simulation is run for 45 seconds and fig. 6 shows the trace of the simulation.

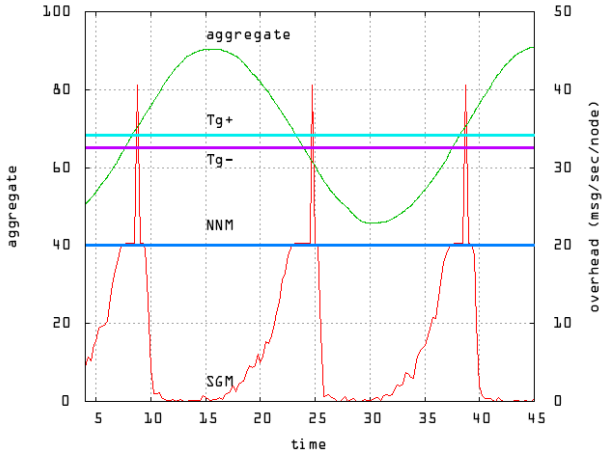


Fig. 6. The protocol overhead over time for the SGM and NNM protocols

Fig. 6 shows the change of the aggregate and the protocol overhead over time. During the simulation run, three threshold crossings occur: at around t=8.3sec (upper threshold crossing), t=24sec (lower threshold crossing) and t=38.2sec (upper threshold crossing). For the baseline NNM protocol, since no message throttling is employed, the protocol overhead is constant (at around 20msg/sec/node). For the SGM protocol, around the time of each threshold crossing, (e.g., between t=7sec to t=10sec) we observe a peak in protocol overhead, as the number of nodes sending messages increases.

We also observe spikes in the SGM protocol when the aggregate crosses the threshold, which are attributed to the overhead of the Echo algorithm. The heights of the spikes from the peaks are approximately equal to that of the peaks themselves, confirming our assessment that the overhead of Echo and the baseline gossip protocol is comparable.

From the above observations, we conclude that the protocol overhead is low whenever the aggregate is far from the threshold while it is highest during the period shortly before a threshold is crossed and for short period after. Third, since a baseline protocol would always send out messages at the highest rate, the protocol overhead of the SGM protocol during peak periods is equal to the overhead of a baseline protocol.

We have achieved a very similar result for the SLM protocol. The main difference is that SLM does not exhibit the spikes since it does not use global snapshots.

### C. Quality of detection: Latency and accuracy

In this scenario we study the delays for detecting threshold crossings by SGM, SLM and the baseline NNM protocol. We simulate the protocols on the 654-nodes topology with the periodic UT trace, resulting in 100 (50 upward and 50 downward) threshold crossings. The resulting delay distributions are shown in fig. 7.

The figure shows that, for this particular scenario, NNM detects the threshold crossings in between -1 and 0.7secs, SGM in between 0.3 and 1.1 seconds and SLM in between

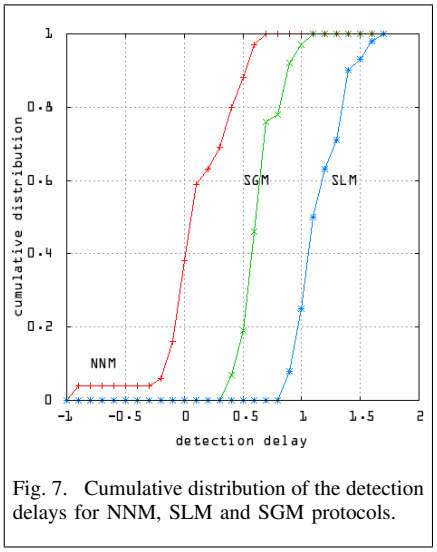


Fig. 7. Cumulative distribution of the detection delays for NNM, SLM and SGM protocols.

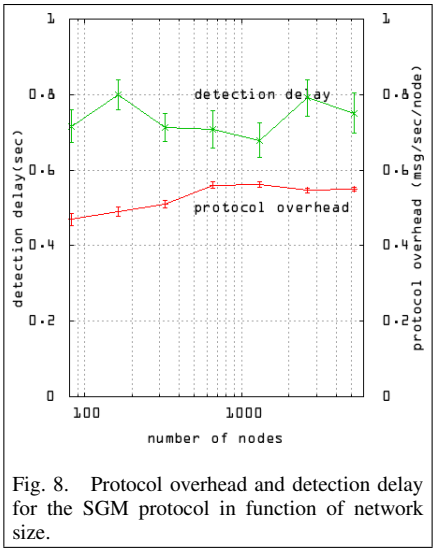


Fig. 8. Protocol overhead and detection delay for the SGM protocol in function of network size.

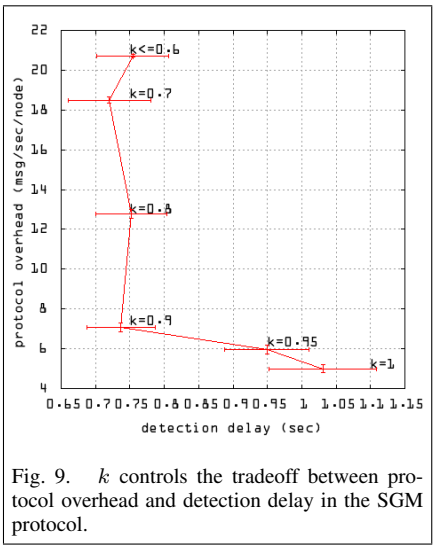


Fig. 9. $k$ controls the tradeoff between protocol overhead and detection delay in the SGM protocol.

0.8 and 1.7secs. The shape of this distribution depends on the dynamics of the local variables, the network size and the topology of the network graph.

For this and other experiments in this paper, we are able to unambiguously associate detected threshold crossings to actual threshold crossings (and hence measure the delay) by considering only TCA's occurring within a short (i.e., 2sec) interval of the actual threshold crossing. The negative detection delays thus obtained for the case of the NNM protocol reflects a propensity to false positives for this protocol. In this scenario, of all alerts raised, 4% were false positives. This number was as high as 14% for other scenarios in our experiments. On the other hand, neither SGM nor SLM exhibited false positives in our experiments. (Indeed, the use of global snapshots makes it highly unlikely for SGM to raise false positives.) None of the protocols exhibited false negatives in all scenarios of our experiments.

### D. Scalability

We study the protocols in two scenarios, where we measure the protocol overhead and the detection delay of threshold crossings as a function of the network size for the SGM protocol.

For both scenarios, GoCast is used to generate graphs with target connectivity of 5 for networks of size 82, 164, 327, 654, 1308, 2626 and 5232. The diameter of the graphs range from 5 (for the 82 node network) to 8 hops (for the 5232 network).

For the first scenario, we use the UT trace to simulate the behavior of the local variables. For each topology, the threshold is set at twice the average value of the aggregate during a run. The result is shown in fig. 8. Each point on the graph is the outcome of 10 simulation runs of 150sec on different graphs (of same size and connectivity).

The figure suggests that the protocol overhead, measured in msg/sec/node, for the specific settings of this scenario, is largely independent of the network size. Note also that the observed overhead is about two orders of magnitude lower than that of the baseline protocol.

In the second scenario, we measure the average detection delay as a function of the network size. The local variables are simulated using the periodic UT trace. The result is also shown in fig. 8.

The figure suggests that the average detection delay of threshold crossings is largely independent of the system size. In the general case, for synthetic traces generated by the same (random) process, if we ignore the time it takes to complete an Echo, we would expect such a result. (For this scenario, the completion time of Echo increases by 50ms between the smallest and largest network.)

From the two scenarios above, we conclude that for the parameters space investigated, SGM is highly scalable in the sense that the detection delay and the protocol overhead is largely independent of system size.

### E. Controllability

We next examine the controllability of SGM. As control parameter, we use $k$. The average protocol overhead and the average TCA detection delay is measured as functions of $k$. For the scenario, we use the default parameters for the simulation. The periodic UT trace is used to generate multiple threshold crossings. The scenario is run for values of $k$ equal to 0, 0.6, 0.7, 0.8, 0.9, 0.95, and 1. The result is shown in fig. 9.

We observe that up to a certain value of $k$ (around $k = 0.9$), the overhead reduces without a major increase in detection delay. We speculate that the specific value of $k$ for which this holds depends on the topology and the dynamics of the local variables. Beyond $k = 0.9$, the protocol overhead decreases with an increase in the detection delay until $k$ reaches 1.

## VI. Discussion and Conclusion

We have explored the use of gossip-type aggregation protocols for distributed detection of threshold crossings of aggregates. Gossip protocols iteratively refine a local estimate of a global aggregate by nearest neighbor interactions. The key idea in our protocols is to let nodes dynamically adjust

the protocol rate according to how far their local estimate of the aggregate is from the threshold. We have identified a family of protocols, organized according to the rate adjustment mechanism, the mechanism for triggering threshold crossing alerts, and whether or not the protocol exploits the symmetry in TCA detection by implementing a hysteresis-like functionality (dual modes). Key points in the design space have been evaluated, by simulation, for efficiency, quality of detection, scalability, and controllability. The results, at least for the choice of aggregation function and local variables in our simulations, are promising: when the aggregate is far from the threshold the protocol overhead is negligible, and when the aggregate is close to the threshold the overhead is comparable with that of the underlying aggregation protocol. We obtained small detection delays and, for the scenarios considered in this paper, absence of false positives and false negatives. Regarding scalability, at least for the scenarios considered in this paper, we did not observe any significant dependence of detection delay on system size. Finally, we have identified a protocol parameter that allows to control the tradeoff between overhead and detection delay.

In [7], we have performed a similar evaluation of the performance of TCA-GAP, a tree-based protocol for detecting threshold crossings. Preliminary comparison of the gossip-based protocols presented here with TCA-GAP suggests that tree-based protocols are more efficient (with up to an order of magnitude lesser overhead) while gossip-based protocols have smaller detection delays (up to 50% of TCA-GAP) that are less dependent on the system size. We plan to investigate this further.

A formal analysis of the convergence properties of our protocols is currently underway and will be reported at a later stage.

This paper considers static networks only. Adaptations of gossip-based aggregation to dynamic networks have been considered by several authors [16], [5], [4]. The modifications needed to handle dynamic networks appear largely orthogonal to the ideas put forward here. Whether that is really so remains, however, to be seen.

## REFERENCES


[1] A. G. Prieto and R. Stadler, "A-GAP: An adaptive protocol for continuous network monitoring with accuracy objectives," *Network and Service Management, IEEE Transactions on*, vol. 4, pp. 2–12, June 2007.

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

[3] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *FOCS '03: Proceedings of the $44^{th}$ Annual IEEE Symposium on Foundations of Computer Science*, p. 482, IEEE Computer Society, 2003.

[4] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Trans. Comput. Syst.*, vol. 23, no. 3, pp. 219–252, 2005.

[5] F. Wuhib, M. Dam, R. Stadler, and A. Clemm, "Robust monitoring of network-wide aggregates through gossiping," in *IM '07: $10^{th}$ IFIP/IEEE International Symposium on Integrated Network Management*, pp. 226–235, May 2007.

[6] D. Breitgand, D. Dolev, and D. Raz, "Accounting mechanism for membership size-dependent pricing of multicast traffic.," in *NGC '03: Networked Group Communication*, pp. 276–286, 2003.

[7] F. Wuhib, M. Dam, and R. Stadler, "Decentralized detection of global threshold crossings using aggregation trees," *Computer Networks*, vol. 52, no. 9, pp. 1745–1761, 2008.

[8] M. Dilman and D. Raz, "Efficient reactive monitoring," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, no. 4, 2002.

[9] R. Keralapura, G. Cormode, and J. Ramamirtham, "Communication-efficient distributed monitoring of thresholded counts," in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pp. 289–300, ACM Press, 2006.

[10] L. Huang, M. Garofalakis, J. Hellerstein, A. Joseph, and N. Taft, "Toward sophisticated detection with distributed triggers," in *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pp. 311–316, ACM Press, 2006.

[11] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pp. 301–312, ACM Press, 2006.

[12] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, (New York, NY, USA), pp. 1–12, ACM Press, 1987.

[13] D. Ernst, A. Hamel, and T. Austin, "Cyclone: a broadcast-free dynamic instruction scheduler with selective replay," in *ISCA '03: Proceedings of the $30^{th}$ annual international symposium on Computer architecture*, (New York, NY, USA), pp. 253–263, ACM Press, 2003.

[14] C. Tang and C. Ward, "Gocast: Gossip-enhanced overlay multicast for fast and dependable group communication," in *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, (Washington, DC, USA), pp. 140–149, IEEE Computer Society, 2005.

[15] D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers, "Decentralized schemes for size estimation in large and dynamic groups," in *NCA '05: Proceedings of the Fourth IEEE International Symposium on Network Computing and Applications*, pp. 41–48, IEEE Computer Society, 2005.

[16] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, "Asynchronous distributed averaging on communication networks," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 512–520, Aug. 2007.

[17] D. Mosk-Aoyama and D. Shah, "Computing separable functions via gossip," in *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, (New York, NY, USA), pp. 113–122, ACM Press, 2006.

[18] G. Tel, *Introduction to Distributed Algorithms*. New York, NY, USA: Cambridge University Press, 2001.

[19] K. S. Lim and R. Stadler, "A navigation pattern for scalable internet management," in *IM '01: $7^{th}$ IFIP/IEEE Int. Symp. on Integrated Network Management*, 2001.

[20] J. N. Tsitsiklis, *Problems in Decentralized Decision Making and Computation*. Cambridge, MA, USA: Dept. of Electrical Enginering and Computer Science, Mass. Institute of Technology, 1984. Ph.D. Dissertation.

[21] A. Olshevsky and J. N. Tsitsiklis, "Convergence rates in distributed consensus averaging," in *CDC '06: Proceedings of the $45^{th}$ IEEE Conference on Decision and Control*, (Washington, DC, USA), pp. 3387–3392, IEEE Computer Society, 2006.

[22] K. S. Lim and R. Stadler, "SIMPSON — a SIMple Pattern Simulator fOr Networks." http://www.s3.kth.se/lcn/software/, 2009.

[23] U. of Twente, "Traffic measurement data repository." http://m2c-a.cs.utwente.nl/repository/, 2006.

[24] K.-S. Lim and R. Stadler, "Real-time views of network traffic using decentralized management.," in *IM '05: $10^{th}$ IFIP/IEEE International Symposium on Integrated Network Management*, pp. 119–132, IEEE, 2005.