Scalable Real-Time Monitoring for Distributed Applications

C.-H. Philip Yuen and S.-H. Gary Chan, *Senior Member*, *IEEE*

Abstract—In order to assess service quality of a networked application (such as a streaming session), distributed monitoring servers need to continuously collect application-specific performance metrics in real time. Much of the previous work to address this is to use distributed aggregation tree (DAT) rooted at each monitor. However, this approach often leads to high monitoring delay and network stress. In this paper, we study a highly scalable monitoring network for distributed applications. In the network, there are distributed monitors collecting application performance in two steps: first, client applications report their performance to some proxies by means of a client overlay, and then the proxies report the performance to the distributed monitors using another proxy overlay. We first formulate the problem to construct overlays minimizing monitoring delay. The problem is shown to be NP-hard. Then, we present a simple, efficient, and scalable monitoring algorithm called SMon, which continuously reduces network diameter in real time in a distributed manner. Through simulations and actual experimental measurements with implementation, we show that SMon achieves low monitoring delay, network stress, and protocol overhead for distributed applications.

Index Terms—Distributed protocol, real-time network monitoring, peer-to-peer network, proxies

1 Introduction

In recent years we have witnessed the deployment of many large-scale distributed applications for file sharing, video-on-demand (VoD), Internet TV, voice over IP (VoIP), etc. For these applications, knowing their overall performance in real time provides important insight into user experience and network conditions. With such monitoring performance, the administrators can respond in an appropriate and timely manner to offer good service level or lower operating cost.

For example, if an unusually high packet delay, jitter, or loss is detected in a certain Internet domain, some routers or links may have failed. The local administrators can then respond by checking and fixing that. Another example is collecting real-time user loads at application servers. If the load of a server is getting high, the administrator may turn on some other servers to offer better quality of service. On the other hand, if its load is low, the server may be turned off, diverting the user traffic to some other servers to save operational cost. Yet another example is to estimate the number of concurrent users accessing a file or streaming session. Knowing such popularity in real time leads to better resource allocation (e.g., by tuning caching scheme and allocating bandwidth). Furthermore, if a high user churn rate is detected for an application in a certain Internet domain, the administrator may turn on some proxies to enhance network stability.

Given the importance of real-time monitoring, we consider in this paper how to efficiently collect performance statistics for large-scale applications (in terms of distribution, central tendency, statistical dispersion, etc.). A simple approach to achieve this is to use a centralized architecture, where all clients continuously feed back their performance to some monitoring or log servers, the so-called *monitors*. By pooling these feedbacks, the monitors can then summarize the overall performance. This approach, however, suffers from scalability problem because the monitor has to maintain large number of connections and may be overwhelmed by the volume of feedback traffic. Such "fan-in" approach to the monitors also leads to much network stress.

To scale to large group, recent work has used a peer-to-peer approach to aggregate the performance metrics. The clients form a monitoring tree rooted at a monitor. Each client sends its performance metrics upstream to its parents for aggregation until the root is reached (i.e., each client is an aggregation point of its children toward the root).[1] Though these schemes are scalable, they often have not considered client fan out and monitoring delay in tree construction. Furthermore, a large-scale application may have multiple distributed monitors. Building a global aggregation tree for each of the monitors leads to problems in scalability and network stress. Because clients may churn (join, leave, or fail) at any time, forming independent global trees rooted at these monitors also make it difficult to isolate churns in one region from the others.

We propose a highly scalable and highly efficient monitoring network with *multiple* distributed monitors. To achieve scalability and to better isolate churn effects, the network consists of *two* tiers, the proxy tier and client tier. Performance is aggregated in two steps: first, clients feed

- *C.-H.P. Yuen is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: chyuen@cse.ust.hk.*
- *S.-H.G. Chan is with the Department of Computer Science and Engineering, Sino Software Research Institute, Risk Management and Business Intelligence Program, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China. E-mail: gchan@cse.ust.hk.*

Manuscript received 26 Jan. 2011; revised 30 Oct. 2011; accepted 24 Jan. 2012; published online 3 Feb. 2012.
Recommended for acceptance by S. Rangarajan.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-01-0044.
Digital Object Identifier no. 10.1109/TPDS.2012.60.

1. The aggregate function may be *MAX*, *MIN*, *SUM*, *AVG*, *COUNT*, etc.

back their measured statistics to the proxies via an overlay tree, then the proxies forward the statistics to the monitors using yet another overlay tree among themselves. Both clients and proxies have their own fan-out constraints due to their processing capability and bandwidth.

We consider the important problem of jointly optimizing the two tiers to minimize monitoring delay. We address this design issue through the followings:

- *Problem formulation and its complexity analysis:* After presenting the aggregation mechanism in this real-time multimonitor proxy-assisted monitoring network, we formulate our research problem, which is to minimize the network diameter (i.e., the worst case overlay delay from clients to monitors). We prove that the problem is NP-hard.
- *SMon: A distributed and adaptive algorithm for real-time monitoring*: We propose a simple, distributed, and adaptive algorithm called SMon, which builds a low-delay monitoring network in the presence of peer churns.[2] SMon continuously reduces monitoring delay by adjusting the peers into better positions in the overlay. It has constant runtime complexity due to a randomized algorithm. The protocol is guaranteed to converge, and adaptive to peer churns.
- *Simulation and experimental studies*: We conduct extensive simulation on SMon. We show that SMon achieves substantially better performance as compared with recent approaches (in terms of delay, stress, etc.). Furthermore, we have implemented SMon. Experimental measurements on real Internet further confirm its adaptivity, effectiveness and low overhead.

We briefly discuss previous work as follows: Research on monitoring has been focusing on constructing Distributed Aggregation Tree (DAT) (e.g., [1], [2], [3], [4], [5], [6]). However, it has not considered the fan-out constraint of peers and how to minimize monitoring delay. Though some strong heuristics and approximation algorithms have been proposed to reduce the network diameter [7], [8], their centralized nature is not applicable to a large-scale dynamic network we consider here. There has been much work on distributed algorithms which may be used for network monitoring [8], [9], [10], [11]. However, they consider either a single-source or a single-tier network. The work of SplitStream constructs multiple multicast trees with structured peer-to-peer overlay, rather than a single overlay on an unstructured two-tier network as we considered here [12]. Given that the two tiers of our monitoring network are coupled, the work cannot be directly applied to solve our problem. Our work differs from network *inference* in that we are interested in the construction of overlay trees to study the performance of *overlay* nodes while network inference is to deduce *underlay* network performance from pairwise overlay path measurement [13], [14].

The rest of this paper is organized as follows: We first present the monitoring network under consideration and its problem formulation in Section 2. In Section 3, we discuss how peer churns are handled in SMon. We describe in detail how SMon adaptively builds an overlay to reduce delay in Section 4. Illustrative simulation results is discussed in Section 5. We conclude in Section 6. The supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.60, contains more detailed literature survey, illustrations, and pseudocodes of the operations of SMon algorithms, NP-hardness proof of our problem, and more complete simulation and experimental results.

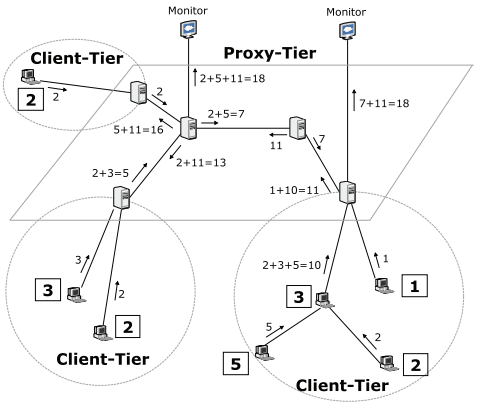


Fig. 1. A multimonitor proxy-assisted monitoring network.

2 System Description and Problem Formulation

2.1 System Description

We show in Fig. 1 the two-tier monitoring network under consideration. Multiple monitors distributed in the network are to collect global performance statistics. The proxy-tier consists of distributed proxies to aggregate statistics, while the client-tier consists of clients whose performance is to be monitored. The clients form an aggregation tree rooted at the proxies. The proxies form an efficient spanning tree among themselves to exchange their aggregated data from their clients. They forward the aggregated data in real time to the monitors attached to some proxies. The problem is to construct efficient overlays in both client and proxy tiers such that the monitoring delay (i.e., the maximum delay from the clients to monitors) is minimized. To address this, we need to answer two inter-related questions: 1) What should the optimal aggregation trees be for the client and proxy tiers? 2) Which clients are associated with which proxies, and which monitors are connected with which proxies?

As in DAT, SMon makes use of data aggregation for efficient monitoring. We show an aggregation example in Fig. 1. The number inside the rectangle denotes the performance metric of the client. For concreteness, let's take the summation as our example of aggregate function (i.e., the monitors are interested in the sum of the metric in the whole network).[3] A client first sends its measured metric upstream to its parent for aggregation until a proxy is reached. For ease of route maintenance, the proxies form a spanning tree among themselves to further aggregate their metrics. This is done by sending to an outgoing link the

2. In this paper, a "peer" refers to either a monitor, proxy, or client.

3. Note that some other more complex aggregation functions may be used, such as grouping, filtering, etc.

aggregated data from all the other links in the tree. Finally, the proxy with attached monitor(s) further sends the aggregated metric to the monitors.

Clearly, the monitoring delay is the sum of two parts: 1) the delay from a client to its proxy through the aggregation tree in the client-tier; and 2) the delay from that proxy to the monitors (in the proxy-tier) through the spanning tree. We study how to minimize the worst case delay (i.e., diameter) subject to degree bounds of the clients and proxies (the maximum connections it can establish with other peers). This is not a trivial problem as the two tiers are coupled; simply considering the two tiers independently would not lead to optimal solution.

2.2 Problem Formulation and Its Complexity

We now formulate the research problem of minimizing the diameter of the monitoring overlay. Let $\mathcal{M}$, $\mathcal{P}$, and $\mathcal{C}$ the disjoint sets of all the monitors, proxies, and clients, respectively. We model the two-tier overlay network as an undirected graph $G = (V, E)$, where V is the set of vertex representing all the participating nodes given by

$$V = \mathcal{M} \cup \mathcal{P} \cup \mathcal{C}, \tag{1}$$

and E is the set of overlay edges. Let $t \in \{\mathbb{P}, \mathbb{C}\}$ be either the proxy-tier ($\mathbb{P}$) or client-tier ($\mathbb{C}$). For any node $v \in V$, let its degree in t be $d^t(v)$ and its maximum degree (degree bound) in t be $d^t_{max}(v)$, i.e.,

$$d^t(v) \leq d^t_{max}(v),\ \forall v \in V,\ t \in \{\mathbb{P}, \mathbb{C}\}. \tag{2}$$

Let l_{ij} be the latency of the edge $\langle i, j\rangle \in E$. Let $\mathbb{R}(u, v)$ be the route from node u to node v in the two-tier overlay tree. Denote the delay from node u to node v along $\mathbb{R}(u, v)$ as $D(u, v)$ given by

$$D(u, v) = \sum_{\langle i,j\rangle \in \mathbb{R}(u,v)} l_{ij}. \tag{3}$$

Define $\delta(v)$ the worst case delay from node v to *any* monitor m given by

$$\delta(v) = \max_{m \in \mathcal{M}} D(v, m). \tag{4}$$

The Degree-Bounded Minimum-Delay Two-Tier Overlay (DBMDTTO) Problem: The DBMDTTO problem is to build a two-tier overlay tree T of G, which minimizes the worse case client-to-monitor delay, i.e.,

$$\min \max_{c \in \mathcal{C}} \delta(c), \tag{5}$$

subject to (2).

The DBMDTTO problem is NP-hard. Please see the supplementary file, available online, for the proof.

3 Handling Peer Dynamics in SMon

Given the problem is NP-hard, we propose a simple, distributed and effective algorithm called SMon for real-time monitoring. In this section, we first present the notations and terminologies used in the discussion of SMon. Then, we discuss how SMon handles peer churns, i.e., joins, leaves, or failures. Illustrative examples and pseudocodes of the schemes can be found in the supplementary file, available online.

3.1 Notations and Terminology

Let $T(V, E_T)$ be the entire overlay tree constructed out of $G(V, E)$, where $E_T \subseteq E$. Denote $T_{\langle i,j\rangle}$ (or $T_{\langle j,i\rangle}$) the partial tree of T that contains proxy j (or i) after the removal of the overlay link $\langle i, j\rangle$. Clearly,

$$T_{\langle i,j,\rangle} \cup \langle i, j\rangle \cup T_{\langle j,i\rangle} = T(V, E_T). \tag{6}$$

Further, let $\mathcal{M}_{\langle i,j\rangle}$, $\mathcal{P}_{\langle i,j\rangle}$, and $\mathcal{C}_{\langle i,j\rangle}$ be the sets of all the monitors, proxies, and clients, respectively, in $T_{\langle i,j\rangle}$.

Let $\Delta(p)$ be the worst case delay from proxy p to all the monitor(s) directly attached to it, which is given by

$$\Delta(p) = \max_{m:\ \forall m \in \mathcal{M},\ \langle p,m\rangle \in E_T} l_{pm}. \tag{7}$$

Clearly, from (4), $\Delta(p) \leq \delta(p), \forall p \in \mathcal{P}$. Further, define $\delta_{\langle i,j\rangle}(v)$ the worst case delay from node v to *any* monitor in $T_{\langle i,j\rangle}$ given by

$$\delta_{\langle i,j\rangle}(v) = \begin{cases} \max_{m \in \mathcal{M}_{\langle i,j\rangle}} D(v, m), & \text{if } \mathcal{M}_{\langle i,j\rangle} \neq \emptyset, \\ -\infty, & \text{otherwise.} \end{cases} \tag{8}$$

Let T_v be the partial tree of T in client-tier rooted at v. Let $\Gamma(v)$ be the worst case delay from *any* client c in T_v to node v, as given by

$$\Gamma(v) = \max_{c \in T_v} D(c, v). \tag{9}$$

Define $\gamma(p)$ the worst case delay from *any* client in T to proxy p, which can be written as

$$\gamma(p) = \max_{c \in \mathcal{C}} D(c, p). \tag{10}$$

Clearly, $\Gamma(p) \leq \gamma(p), \forall p \in \mathcal{P}$. Further define $\gamma_{\langle i,j\rangle}(p)$ the worst case delay from *any* client in $T_{\langle i,j\rangle}$ to proxy p:

$$\gamma_{\langle i,j\rangle}(p) = \begin{cases} \max_{c \in \mathcal{C}_{\langle i,j\rangle}} D(c, p), & \text{if } \mathcal{C}_{\langle i,j\rangle} \neq \emptyset, \\ -\infty, & \text{otherwise.} \end{cases} \tag{11}$$

Consider all the client-to-monitor aggregation paths that pass through proxy p. There can be two exclusive cases to consider: 1) the client is in T_p; and 2) the client is *not* in T_p. Clearly, the worst case delay of the paths in Case 1 is given by

$$\delta(p) + \Gamma(p), \tag{12}$$

while the worst case delay of the paths in Case 2 is given by

$$\max_{q:\ \forall q \in \mathcal{P},\ \langle p,q\rangle \in E_T} \big(\delta_{\langle q,p\rangle}(p) + \gamma_{\langle p,q\rangle}(p)\big). \tag{13}$$

We define $\Pi(p)$ the worst case delay of all the client-to-monitor aggregation paths that pass through proxy p, which is given by

$$\Pi(p) = \max\begin{pmatrix} \delta(p) + \Gamma(p), \\ \max_{q:\ \forall q \in \mathcal{P},\ \langle p,q\rangle \in E_T} \big(\delta_{\langle q,p\rangle}(p) + \gamma_{\langle p,q\rangle}(p)\big) \end{pmatrix}. \tag{14}$$

We show in Fig. 2 an example of the above symbols (see Table 1 for nomenclature). In Fig. 2, the delay on all

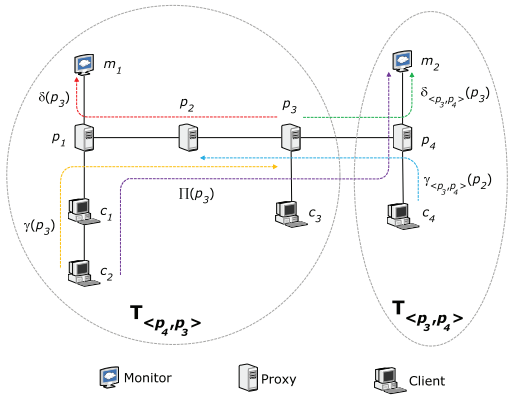


Fig. 2. Notations.

the overlay links are the same. The path for $\delta(p_3)$ is $(p_3 - p_2 - p_1 - m_1)$; the path for $\delta_{\langle p_3,p_4\rangle}(p_3)$ is $(p_3 - p_4 - m_2)$; the path for $\gamma(p_3)$ is $(c_2 - c_1 - p_1 - p_2 - p_3)$; the path for $\gamma_{\langle p_3,p_4\rangle}(p_2)$ is $(c_4 - p_4 - p_3 - p_2)$; and the path for $\Pi(p_3)$ is $(c_2 - c_1 - p_1 - p_2 - p_3 - p_4 - m_2)$.

3.2 Monitor Churns

A joining monitor should connect to a proxy neighbor that leads to low delay from all the clients. As in many distributed systems such as peer-to-peer networks, there is a *Rendezvous Point* (RP) in the network, which is essential to bootstrap the joining process. The RPs in SMon are well-known servers performing different functions as suitable for monitors, proxies, and clients. All peers should know the RPs. Note that the RP load is low as it is only connected at the join times of requests.

A new monitor m receives from RP a random list of proxies as the potential proxy neighbors. It may seek more potential neighbors through gossip. After receiving the candidates, the monitor requests $\gamma(p)$ of each of the potential proxy neighbor p and evaluates $(\gamma(p) + l_{pm})$. After computing the delays, m connects to the proxy neighbor p with available degree and with the lowest $(\gamma(p) + l_{pm})$.

When a monitor is about to leave, it informs its attached proxy. Upon detecting a monitor has left, the proxy it attached to updates its set of neighbors. The failure of a monitor is detected by its absence of heart-beat. In this case, the proxy it attached to also updates its set of neighbors.

3.3 Client Churns

An arriving client should connect to a parent that leads to low delay to all the monitors. A new client c receives from RP a random list of proxies and clients as its potential parents. Client c then requests $\delta(v)$ from each of the potential parent v and evaluates $(\delta(v) + l_{cv})$ with respect to each of them. The client may request more peers in a gossip manner. After computing the delays, c connects to the parent v with available degree of the lowest $(\delta(v) + l_{cv})$.

When a client is about to leave, it initiates a leave message to its parent, asking the parent to update its set of children. It also sends a message to all its children, asking them to look for new parents. The repair process of the children is two-phase. First, they will try to connect to their grandparent. If the first phase fails (due to, for example, their grandparent does not have available degree), they will start the join process.

TABLE 1
Table of Nomenclature

Symbols	Meanings
$T(V, E_T)$	Entire overlay tree of $G(V, E)$
$T_{\langle i,j\rangle}$	Partial tree of T that contains node j after the removal of the overlay link $\langle i, j\rangle$
T_v	Partial tree of T in client-tier with node v being the root
$\mathbb{R}(u, v)$	Route from node u to node v
$D(u, v)$	Delay from node u to node v along $\mathbb{R}(u, v)$
t	Label for either the proxy-tier ($\mathbb{P}$) or client-tier ($\mathbb{C}$)
$d^t(v)$	Degree of node v in t
$d^t_{max}(v)$	Degree bound of node v in t
$\Delta(p)$	Worst-case delay from proxy p to the monitor directly attached to it
$\delta(v)$	Worst-case delay from node v to any monitor in T
$\delta_{\langle i,j\rangle}(v)$	Worst-case delay from node v to any monitor in $T_{\langle i,j\rangle}$
$\Gamma(v)$	Worst-case delay from any client c in T_v to node v
$\gamma(p)$	Worst-case delay from proxy p to any client in T
$\gamma_{\langle i,j\rangle}(p)$	Worst-case delay from proxy p to any client in $T_{\langle i,j\rangle}$
$\Pi(p)$	Worst-case delay of all client-to-monitor aggregation paths that pass through proxy p

In order to ensure connectivity in the join process, each of the clients maintains a *Root-Path* that begins from its root proxy toward itself.[4] Keeping the *Root-Path* is important because in the rejoin process, a peer in the potential-parent list returned by RP may be a descendant of the repairing client. A loop will exist if a client takes its descendant as its parent. The repairing client eliminates looping (hence overlay disconnection) in the rejoin process by examining whether the *Root-Path* of the potential parents contains its own identity or not.

Note that the length of the *Root-Path* of a node is the same as its depth in the monitoring tree. The tree in steady state is very close to a complete tree as SMon effectively utilizes the available degrees of nodes to form a low-delay overlay. As a result, the tree depth, and hence the maximum length of *Root-Path*, is $O(\log n)$. It is clear that the memory and traffic overhead to keep the *Root-Path* updated is low.

A client may fail at anytime. To handle this, each client regularly sends its heartbeat to its parent and children. When a client finds its parent fails, it looks for a new parent using the repair process. On the other hand, when it finds some of its children fail, it updates its set of children.

3.4 Proxy Churns

A newly arrived proxy should look for a proxy neighbor that leads to low delay to all the monitors. A new proxy p receives a random list of potential proxy neighbors from RP. It then requests $\delta(q)$ of each of the potential proxy neighbor q and evaluates $(\delta(q) + l_{pq})$ with respect to each of them. It may get more candidates through gossip. After computing

4. Such root path can be easily maintained as follows: A root proxy sends the *Root-Path* that contains its identity to its children. The child concatenates its identity to the *Root-Path* and sends to its children, and so on.

the delays, p connects to proxy neighbor q with available degree of the lowest $(\delta(q) + l_{pq})$.

When a proxy is about to leave, it initiates a leave message to all of its monitor neighbors, proxy neighbors, and client children, asking them to rejoin the network. To guarantee connectivity in the spanning tree after the rejoin process, we put a special node in the proxy-tier network, termed *virtual root*, which acts as the root of proxy-tier (note that the virtual root may be colocated with the RP). The virtual root connects to exactly one of the proxies and will not leave the network (like RP). The only responsibility of the virtual root is to send the *Root-Path* that contains its identity to its proxy neighbor p. Proxy p will then concatenate its identity to the *Root-Path* and send to its proxy neighbors. By doing so, we can maintain an unidirectional path in the proxy-tier, and thus the repairing proxy can eliminate looping (hence overlay disconnection) in the rejoin process. The operation of how proxies handle neighbor failures is similar to that of clients.

4 Adaptation to Reduce Delay in SMon

The monitoring network should keep evolving over time to accommodate peer churns by continuously moving peers into better position through *adaptation* to reduce delay. Each peer in SMon periodically runs an adaptation algorithm. An adaptation is performed if and only if it reduces the worst case source-to-end delay. As will be clear in the following, the cost of dynamic adaptation is bounded by the degree of each peer. As the maximum degree of each peer is a constant, our adaptation algorithms achieve constant-time complexity (i.e., $O(1)$).

It worths pointing out here that our focus is on overlay to *monitor* performance (i.e., the control plane). The plane is independent of and separated from data distribution (i.e., the data plane). The planes may treat each other as background traffic and handle errors independently. For performance monitoring, the control plane does not have to be 100 percent reliable, i.e., occasional packet losses due to adaptation may be tolerated or recovered through retransmissions with best effort. In the following, we discuss how the monitors, proxies, and clients adapt to achieve low monitoring delay. We prove the convergence of the algorithms in the supplementary file, available online.

4.1 Monitor Adaptation

A monitor m, attached to a proxy p, periodically requests $\gamma(q)$, where q is the proxy neighbors of p. It evaluates $(\gamma(q) + l_{mq})$. The monitor replaces the proxy with the lowest delay.

4.2 Proxy Adaptation

A proxy p periodically broadcasts a *Migration Request Message* (MRM) to its proxy neighbors with a time to live (TTL) value, which is decremented by 1 each time it is forwarded until it hits 0. When a proxy q with available degree receives MRM, it replies p with a GRANT message, which is a tuple of $\langle \delta_{\langle p,r\rangle}(q), \gamma_{\langle p,r\rangle}(q), l_{pq} \rangle$, where r is p's one-hop proxy neighbor.

Proxy p may receive a number of GRANT messages. For each replier q, proxy p considers a new two-tier overlay tree constructed by replacing the overlay link $\langle p, r\rangle$ with the overlay link $\langle p, q\rangle$, and calculates $\Pi(p)$ for the overlay tree according to (14) using the tuple. The proxy neighbor that yields the smallest $\Pi(p)$ is chosen as the new neighbor.

4.3 Client Adaptation

There are two adaptation cases for clients: Grandparent Migration where a client chooses its grandparent as its better parent, and Proxy Migration where a client chooses a better proxy to associate with. Both cases aim at pushing clients closer to monitors to reduce delay subject to the degree bounds. They are discussed below:

- *Grandparent Migration*:
  A client c sends a migration request to its grandparent d. The grandparent d with available degree adopts client c as its child if this reduces the delay from c toward monitors, i.e., client c leaves its current parent and connects to d if

$$d^{\mathbb{C}}(d) < d^{\mathbb{C}}_{max}(d), \tag{15}$$

  and

$$l_{cd} + \delta(d) < \delta(c). \tag{16}$$

  If d does not have available degree to perform the above step, it "triggers" one of its children to seek for a new parent so as to free itself up for client c. The clients being affected in this case are c and the triggered client. Clearly, the worst-case client-to-monitor delay of the affected clients before adaptation is

$$\max(\delta(c) + \Gamma(c), \delta(e) + \Gamma(e)), \tag{17}$$

  where e is the triggered client. Client d will trigger its child e that has the smallest $(l_{ed} + \Gamma(e))$ to seek for a new parent to achieve a much balanced overlay tree. Note that if e is the parent of c, c will leave e in the adaptation. Therefore, in this case $\Gamma(e)$ becomes

$$\max_{v \in \{T_e \setminus T_c\}} (\delta(v) - \delta(e)). \tag{18}$$

  Child e will seek for a parent that yields the smallest client-to-monitor delay from a set consisting of d's children and c. This triggering will be performed if and only if the worse case client-to-monitor delay of the affected clients after adaptation is lower than (17).
- *Proxy Migration*:
  A client c, given its parent is a proxy p, requests from each of p's proxy neighbors with available degree labeled as $q\delta(q)$ and evaluates $(l_{cq} + \delta(q))$. Client c leaves its current parent and connects to proxy q if this reduces the delay toward monitors, i.e., it switches if

$$d^{\mathbb{C}}(q) < d^{\mathbb{C}}_{max}(q), \tag{19}$$

  and

$$l_{cq} + \delta(q) < \delta(c). \tag{20}$$

  If q does not have available degree to perform the above step, it may trigger one of its children to seek for a new parent so as to free itself up for client c. The clients being influenced in this case are c and the

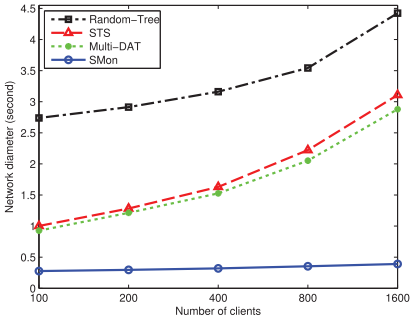


Fig. 3. Maximum monitoring delay versus number of clients.

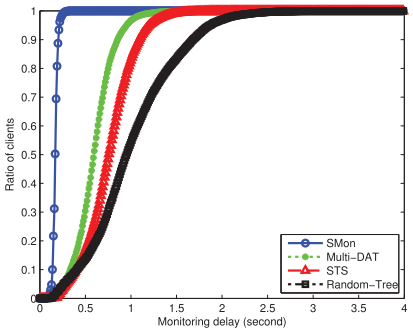


Fig. 4. Monitoring delay distribution.

triggered client and their descendants. The worst case client-to-monitor delay of the influenced clients before adaptation is (17). The selection of the triggered client and its new parent are the same as that of the case of *Grandparent Migration*. This trigger case will be performed if and only if the worst case client-to-monitor delay of the influenced clients after adaptation is lower than (17).

Clients only migrate to their grandparent (found in their Root-Path) to reduce the monitoring delay and the maintenance overhead on the list of potential parents. Note that it is possible for a client to connect to a parent that is not in its current path to the root monitor. This is the case when the client is a direct child of a proxy and it finds a better parent in one of the neighbors of the proxy.

5 Illustrative Simulation Results

We have conducted extensive simulation and experimental studies on SMon. This section presents our simulation results. The experimental results on our implementation of SMon are shown in the supplementary file, available online.

5.1 Simulation Environment and Metrics

We have implemented an event-driven simulation on SMon using C++. We use Brite [15] to generate a two-level top-down hierarchical topology consisting of eight autonomous systems each of which has 625 routers (yielding a total of 5,000 routers and about 20,000 links). Brite also provides us link latency in millisecond. Peers are attached to the routers randomly. Proxies and clients arrive according to a Poisson process with rate λ_p (request/minute) and λ_c (request/minute), respectively. The sojourn time of the proxies and clients are exponentially distributed with mean $1/\mu_p$ (minutes) and $1/\mu_c$ (minutes), respectively. Clearly, the number of proxies and clients in the system is Poisson with mean λ_p/μ_p and λ_c/μ_c, respectively [16]. Unless otherwise stated, we use the following baseline parameters: $\lambda_p = 0.2$ request/minute, $1/\mu_p = 100$ minutes, $\lambda_c = 20$ request/minute, $1/\mu_c = 40$ minutes, proxy-tier degree bound $= 10$, client-tier degree bound $= 5$, number of monitors $= 3$, adaptation interval $= 1$ minute, TTL $= 4$, and processing delay per hop $= 30$ msec.

We use the following evaluation metrics in our study:

- *Monitoring delay:* Monitoring delay is the path delay from a client to a monitor. We are interested in its maximum (i.e., diameter), average, and distribution among all the clients.
- *Network stress:* Network stress is defined as the average number of connections established in an used underlay link.
- *Network overhead:* Network overhead is defined as the average number of request messages in adaptation sent in the network per unit time (including those flooded ones).

We compare SMon performance with the following traditional and recent schemes:

- *Random-tree:* In this scheme, the clients form a tree rooted at a proxy by connecting to a random parent subject to their degree bounds. The proxies form another spanning tree among themselves by connecting to a random neighbor subject to the degree bounds. The scheme is simple to implement, and may approximate many DAT approaches making use of DHT.
- *Multi-DAT:* In this scheme, the proxies in the proxy-tier form multiple DATs, each of which rooted at a monitor. We use closest parent scheme for our overlay construction. Newly arrived peers choose parents that are closest to them, subject to the degree bound. This scheme is simple, and captures locality of the peers.
- *STS:* In this scheme, the clients form a DAT rooted at the proxies using closest parent scheme. The scheme builds the proxy-tier network using Shared Tree Streaming (STS) protocol [10]. STS has been shown to achieve low network diameter.

5.2 Illustrative Results

We plot in Fig. 3 the maximum monitoring delay (i.e., diameter) against the number of clients. We increase the number of clients by increasing $1/\mu_c$ (as the expected number of clients in the system is λ_c/μ_c). The delay in all schemes increases with the number of clients. This is expected due to a larger overlay. SMon outperforms the other three schemes because its peers continuously move into better positions in the overlay so as to reduce the monitoring delay. Both Multi-DAT and STS do not perform so well as SMon, because they have not considered joint optimization of proxy tier and client tier as SMon does. Random-Tree performs the worst because it has many long aggregation paths.

The cumulative distribution function (CDF) of the monitoring delay of the four schemes is shown in Fig. 4. Clearly, the delay of SMon is lower than the other three schemes. This shows the effectiveness of the adaptation mechanism in

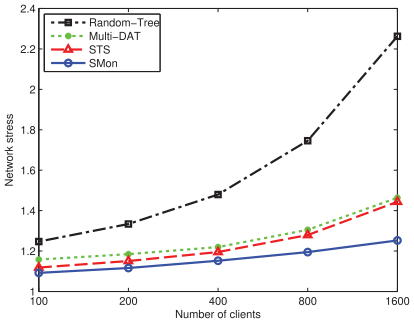


Fig. 5. Network stress versus number of clients.

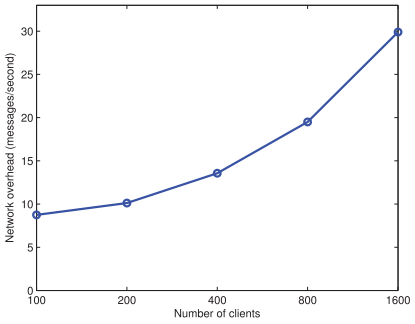


Fig. 7. Network overhead versus number of clients.

reducing delay of SMon. Moreover, the variation of the delay of SMon is lower than the other three schemes as it has lower spread in delay.

We plot in Fig. 5 the network stress against the number of clients. The stress in all schemes increases as the number of clients increases. This is because the number of connections increases when the network size increases. SMon outperforms the other three schemes as the number of clients increases. This is because in SMon, peers continuously move into better positions in the overlay. This eliminates many long connections and saves much bandwidth in the network. More efficient routing leads to lower network stress. Multi-DAT builds multiple DATs each for a monitor overlapping each other, hence has a higher stress than STS. The stress in Random Tree is the highest, because the connections are made randomly with some physical links may be used multiple times by different peers.

We plot in Fig. 6 the network diameter against the degree bound of the client-tier. The delay in all schemes decreases as the degree bound increases. This is because a larger bound yields a flatter overlay, therefore lowering the delay. The delay of SMon is substantially lower than the others as SMon better utilizes the available degrees of peers to reduce delay.

We plot in Fig. 7 network overhead against number of clients. The overhead increases with the number of clients, because the more the clients, the more the messages are introduced into the network. The overhead of SMon is low, which can be illustrated by picking a point in the graph. Take 800 clients, which means a total of 19.5 message/second. Assume the control message is of size 8 kb (a reasonable estimate). Then each client generates on average $(19.5 \text{ message/second}/800 \times 8 \text{ kb}) = 0.195$ kb/second overhead.

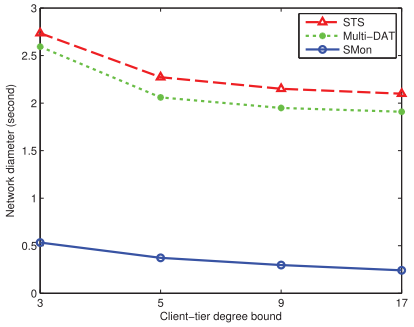


Fig. 6. Maximum monitoring delay versus client-tier degree bound.

Because the control traffic shares the same paths on the overlay tree as monitoring traffic, it has a stress of 1.2 (see Fig. 5); each used underlay link hence has only on average $(0.195 \text{ kb/second} \times 1.2) = 0.234$ kb/second control traffic overhead. This is negligible as compared with the data rate and monitoring traffic of the application.

6 Conclusions

In this paper, we have studied a highly scalable real-time monitoring network with multiple monitors for large-scale applications. Our monitoring network uses data aggregation and consists of two tiers, the proxy tier and client tier. In the proxy tier, multiple proxies are set up to isolate peer churns. In the client tier, distributed clients form an overlay tree to aggregate their performance data.

We have studied minimizing the monitoring delay of this network. We first formulate the problem and prove that it is NP-hard. We then propose and study a simple, efficient, and distributed algorithm called SMon, which continuously reduces the monitoring delay in the presence of peer churns. Our simulation results show that SMon indeed achieves substantially lower delay and network stress than existing and state-of-the-art approaches. It also has low-communication overhead. We have implemented SMon and our experimental measurements further confirm its adaptivity and effectiveness.

Acknowledgments

This work was supported, in part, by the General Research Fund from the Research Grant Council of the Hong Kong Special Administrative Region, China (611209), and Google Mobile 2014 and Faculty Research Awards.

References

[1] B. Yu, J. Li, and Y. Li, "Distributed Data Aggregation Scheduling in Wireless Sensor Networks," *Proc. IEEE INFOCOM*, 2009.

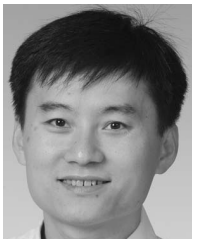
[2] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks," *ACM SIGOPS Operating Systems Rev.*, vol. 36, no. SI, pp. 131-146, 2002.

[3] R.V. Renesse, K.P. Birman, and W. Vogels, "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining," *ACM Trans. Computer Systems*, vol. 21, no. 2, pp. 164-206, 2003.

[4] P. Yalagandula and M. Dahlin, "A Scalable Distributed Information Management System," *Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm.*, pp. 379-390, 2004.

[5] I.A. Dahlia, I. Abraham, D. Malkhi, and O. Dobzinski, "LAND: Locality Aware Networks for Distributed Hash Tables," Technical Report 2003-75, Leibnitz Center of the School of Computer Science and Eng., the Hebrew Univ. of Jerusalem, 2003.
[6] W.-P.K. Yiu, X. Jin, and S.-H.G. Chan, "VMesh: Distributed Segment Storage for Peer-to-Peer Interactive Video Streaming," *IEEE J. Selected Areas in Comm.*, special issue on advances in peer-to-peer streaming systems, vol. 25, no. 9, pp. 1717-1731, Dec. 2007.
[7] K.H. Vik, C. Griwodz, and P. Halvorsen, "Constructing Low-Latency Overlay Networks: Tree Vs. Mesh Algorithms," *Proc. IEEE 33rd Conf. Local Computer Networks (LCN '08)*, pp. 36-43, 2008.
[8] E. Brosh, A. Levin, and Y. Shavitt, "Approximation and Heuristic Algorithms for Minimum-Delay Application-Layer Multicast Trees," *IEEE/ACM Trans. Networking*, vol. 15, no. 2, pp. 473-484, Apr. 2007.
[9] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "OMNI: An Efficient Overlay Multicast Infrastructure for Real-Time Applications," *Computer Networks*, vol. 50, no. 6, pp. 826-841, 2006.
[10] T.M. Baduge, A. Hiromori, H. Yamaguchi, and T. Higashino, "A Distributed Algorithm for Constructing Minimum Delay Spanning Trees under Bandwidth Constraints on Overlay Networks," *Systems and Computers in Japan*, vol. 37, no. 14, pp. 15-24, 2006.
[11] S.-H. G. Chan and F. Tobagi, "Distributed Servers Architecture for Networked Video Services," *IEEE/ACM Trans. Networking*, vol. 9, no. 2, pp. 125-136, Apr. 2001.
[12] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-Bandwidth Multicast in Cooperative Environments," *SIGOPS Operating Systems Rev.*, vol. 37, pp. 298-313, Oct. 2003.
[13] Y. Chen, D. Bindel, H.H. Song, and R.H. Katz, "Algebra-Based Scalable Overlay Network Monitoring: Algorithms, Evaluation, and Applications," *IEEE/ACM Trans. Networking*, vol. 15, no. 5, pp. 1084-1097, Oct. 2007.
[14] M. Coates, Y. Pointurier, and M. Rabbat, "Compressed Network Monitoring for IP and All-Optical Networks," *Proc. Seventh ACM SIGCOMM Conf. Internet Measurement (IMC '07)*, pp. 241-252, 2007.
[15] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Universal Topology Generation from A User's Perspective," *Proc. MASCOTS '01*, Jan. 2001.
[16] L. Kleinrock, *Queueing Systems: Theory*, vol. 1. John Wiley & Sons, 1976.

C.-H. Philip Yuen received the BEng degree in computer science and the MPhil degree in computer science and engineering from the Hong Kong University of Science and Technology (HKUST), Kowloon, in 2008 and 2010, respectively. His research interests include computer networks, multimedia networking, and peer-to-peer systems.



S.-H. Gary Chan (S'89-M'98-SM'03) received the BSE degree (highest honor) in electrical engineering from Princeton University, Princeton, NJ, in 1993, with certificates in applied and computational mathematics, engineering physics, and engineering and management systems and the MSE and PhD degrees in electrical engineering from Stanford University, Stanford, CA, in 1994 and 1999, respectively, with a minor in business administration. He is currently an associate professor of the Department of Computer Science and Engineering, director of Sino Software Research Institute, and co-director of Risk Management and Business Intelligence program, The Hong Kong University of Science and Technology (HKUST), Hong Kong. His research interest includes multimedia networking, peer-to-peer streaming and technologies, and wireless communication networks. He has been an associate editor of *IEEE Transactions on Multimedia* (2006-2011), and is a vice-chair of Peer-to-Peer Networking and Communications Technical subcommittee of IEEE Comsoc Emerging Technologies Committee. He has been guest editors of *IEEE Transactions on Multimedia* (2011), *IEEE Signal Processing Magazine* (2011), *IEEE Communication Magazine* (2007), and *Springer Multimedia Tools and Applications* (2007). He was the TPC chair of IEEE Consumer Communications and Networking Conference (CCNC) 2010, Multimedia symposium in IEEE Globecom (2007 and 2006), and IEEE ICC (2007 and 2005), and Workshop on Advances in Peer-to-Peer Multimedia Streaming in ACM Multimedia Conference (2005). He is the recipient of Google Mobile 2014 award in 2010 and 2011, and is a member of honor societies Tau Beta Pi, Sigma Xi, and Phi Beta Kappa. He has been a visiting professor/adjunct researcher in Microsoft Research Asia (2000-11), research collaborator at Princeton University (09), visiting associate professor at Stanford University (2008-2009), director of Computer Engineering Program at the HKUST (2006-2008), visiting assistant professor in Networking at University of California at Davis (1998-1999), and research intern at the NEC Research Institute, Princeton, NJ (1992-1993). He was a William and Leila Fellow at Stanford University (1993-94). At Princeton, he was the 1993 recipient of the Charles Ira Young Memorial Tablet and Medal and the POEM Newport Award of Excellence. He is a senior member of the IEEE.