Hierarchical Notification Dissemination for IMS Presence using Network Coordinates

Dylan Andrew Harper Smith, Changcheng Huang, and James Yan
Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
dylan.ah.smith@gmail.com, huang@sce.carleton.ca, jim.yan@sympatico.ca

Abstract—**The presence service provides event-based notifications of the status of contacts. Currently duplicate notifications must be sent directly to a set of watchers in unicast messages, putting unneeded strain on outgoing network links of the source node. In this article, we propose an extension to the Session Initiation Protocol (SIP) to distribute notifications through a tree of watchers using network coordinates to build the tree. Our simulation results show traffic costs halved by using the extension without significantly adding delay.**

Keywords-IP Multimedia Subsystem; Network Coordinates; Presence; SIP; Tree Network

I. INTRODUCTION

Presence is a service supported by the IP Multimedia Subsystem (IMS) to provide a service subscriber updates on the status of the subscriber's contacts before the subscriber attempts to initiate a multimedia session with any of them. These updates are normally sent out directly to a set of destinations that have subscribed to receive these updates; however, this method causes a number of identical messages to be sent across network links close to the source of the notifying node. IP Multicast could reduce message replication but to date it is not widely deployed [1]. Instead, this paper presents an application-layer multicast approach that uses a tree-based distribution of notifications, with network coordinates guiding the set up of the distribution tree.

II. BACKGROUND ON IMS PRESENCE

The presence service provides a feature that has existed for years in instant messaging services to display a status indicator for each contact in a contact list. The user providing presence information is called a presentity. Presence conveys to watchers (nodes viewing presence status) the presentity's ability or willingness to communicate. This can allow a watcher to quickly identify who they can communicate with from their contact list before trying to initiate a conversation.

IMS is an architecture framework, developed for next generation networks, to provide IP (Internet Protocol) multimedia services. Presence is one of the services provided by IMS. Presence in IMS is built on top of the SIP (Session Initiation Protocol) event notification framework [2] through the presence event package [3]. Presence is used through devices with software to act as Presence User Agents (PUAs). Notifications are used to keep track of the presence status of all the presentities on a user's contact list.

We appreciate the financial support from the NSERC grant #CRDPJ 354729-07 and the OCE grant #CA-ST-150764-8.

A Resource List Server (RLS), which is defined in [4], can store the user's contact list as a resource list. The PUA can then subscribe to the resource list by sending a SUBSCRIBE message to their RLS, which will then send a SUBSCRIBE request to the presence URI (Uniform Resource Identifier) of each presentity on their contact list. After a successful subscription, the RLS becomes a watcher for the duration of the subscription, and an initial NOTIFY message is sent with the presentity's presence status. Subscriptions expire after a time specified in the SUBSCRIBE message's *Expires* header, which may be 0 for explicitly unsubscribing. Such a message will be referred to in this paper as an unsubscribe message.

A PUA updates the presentity's presence status through sending a PUBLISH message to the RLS. The RLS then sends a NOTIFY message to all the presentity's watchers with the presentity's current presence status. Notifications received by an RLS are throttled and aggregated before being sent back to the PUA subscribed to the resource list.

A PUA could directly subscribe to presentities without using an RLS, but then redundant data would be sent through the PUA's access network, which would be even less suitable for a mobile device. A PUA not using an RLS can be modeled as an RLS used by a single PUA because of the similar traffic behavior. In either case notifications are sent over the same link, so a type of multicast would be beneficial.

A. Presence Data

The messages for IMS presence are sent using the SIP protocol [5], and the presence data are stored in an XML (eXtensible Markup Language) based format called the Presence Information Data Format (PIDF). PIDF is a minimal specification that only defines a minimal amount of elements, but was designed to be extendible. PIDF only defines in [6] presence data to include an online/offline status, a contact URI (e.g. phone number or SIP address), a human readable note, and a timestamp for when the presence data were last changed.

PIDF alone is not sufficient for commercial applications since it would not even allow a user to specify that they are online yet busy. Four extensions to PIDF that are used for IMS presence are Rich Presence Information Data Format, Contact Information in Presence Information Data Format, the Timed Presence extension, and the User Agent Capability Extension [7]. These extensions are backwards compatible, so extra information provided using these extensions is ignored by PUAs not implementing them, and new extensions can be added without breaking existing functionality.

Standard extensions enable devices to properly interpret, display, and update the rich presence data automatically. Presence data may be automatically updated by the device based on user activity, sensor data, or by the presentity manually entering data. For example, a device could update presence status to at work, in a meeting, and in a video conference based on the presentity's Global Positioning System coordinates, calendar data, and device usage respectively.

Sending the whole presence document wastes bandwidth since only small portions of the presence data will be changed each time they are published. Partial notifications save bandwidth by only sending the data that have changed, using a different XML format with additional elements to specify what was removed, added, or modified. SIP provides a mechanism, known as SigComp (Signal Compression), which can further reduce the size of the transmitted data [7].

Partial notifications and compression reduce the size of each notification messages without reducing the number of messages. Frequent automatic presence status updates may originate from devices using PIDF extensions, and their notification will be duplicated over an outgoing link for all watchers. Section IV proposes a hierarchical notification protocol that uses network coordinates to build a multicast tree out of watcher nodes, which reduces duplicate notifications on network links without network-layer multicast support.

III. Related Work

IP Multicast for IMS has been proposed in [8]. IP Multicast would be an ideal solution to the problem of sending homogeneous notifications to multiple destinations, but IP Multicast has not yet been widely adopted [1]. IMS presence requires communication between RLSs in home networks controlled by multiple service providers, so wide support for IP Multicast would be required in routers between these RLSs. Instead, application-layer multicast only needs a subset of RLSs supporting a new protocol to provide similar benefits.

The idea of using tree-based dissemination of presence notifications has been previously explored in [9]. Their approach was to set up a distribution tree at a presentity's presence server by dividing the watchers into clusters of at most K nodes where K is the maximum degree in the tree. The clusters are formed by sorting the watchers based on whether the nodes use a radio or wired interface before segmenting them into clusters with at most K nodes each. The clusters are further segmented at each child until a tree is formed.

A problem with the above method of setting up the tree is that there will be a good chance that child nodes will be far away from parent nodes. This can cause long delays as notification messages are passed down the tree, with intermediate nodes being at random places on the Internet. This may cause a notification message to travel between opposite sides of the Internet to reach a leaf node that is close to the root node. This will result in greater delay and more stress on the network since notifications will remain in the network for a longer time, either in transit or queued by routers.

Related work has been done using network coordinates for peer-to-peer file sharing [10], but differs greatly since large quantities of data continue being transferred after the source node leaves the peer-to-peer network, so a tree structure would not be appropriate due to reliability concerns. Network coordinates have also been used for Minimum Spanning Tree (MST) construction [11]; however, presence is a soft real-time application where delay from the source node needs to be considered in addition to traffic cost.

The presence service also differs from MST construction in that multiple trees will need to be constructed from different source nodes, and will only use a subset of the nodes for each tree. The overhead for tree construction must be low since there are not large amounts of data being transferred.

IV. Hierarchical Notification Dissemination

Application-layer multicast can be performed by organizing nodes into a tree rooted at the source node. The source node sends out a notification message to only its child nodes, and then each non-leaf node in this tree that receives a notification message forwards the message to each of its children. This still requires the same number of notification messages to be sent; however, the messages are not all sent out from the same node, so the extra load and queuing delay near the source node is avoided. Additionally, the distance that each message needs to travel can be shortened by making sure child nodes are as close to their parent node as possible. Therefore, strain on the network is reduced by minimizing the total number of hops used to notify all the watchers.

Network coordinates have been chosen, for its low traffic cost, as the method to gather network location information for use in trying to set up an optimal distribution tree.

A. Network Coordinates

Network coordinates (NCs) are multi-dimensional virtual coordinates calculated for each node in the network to approximate the distance between two nodes in a network without measuring it directly. The distance between two nodes is typically measured using latency of a request-response pair, which is used to update the NCs. The approximate distance between two nodes with known NCs can then be calculated by finding the distance between the virtual positions of the nodes.

Vivaldi [12] was chosen as the NC algorithm for our tree-based presence notifications for its simplicity and wide use as a basis for NC research [10][11]. Another NC algorithm, such as Pharos NCs [11], could be used if it were found to be better.

In the Vivaldi NC algorithm, all nodes start off at the origin. Nodes make measurements between themselves and other nodes to modify their NCs. The algorithm is conceptually a spring-based system; where one could think of there being springs placed between nodes with a rest length equal to the measured round-trip time (RTT). Forces are calculated for each measurement between two nodes, and those nodes push away or pull towards each other with a force proportional to the difference between the measured RTT and the one calculated from the NCs. When nodes have the same NC, such as at the start when all nodes are at the origin, a random direction for the force is used, otherwise the direction is parallel to the path between the NCs.

Equation (1) is used to calculate the force exerted between two nodes where L_{ij} is the measured RTT between node i and node j; x_i and x_j are the NCs for node i and j respectively. The magnitude of the force is composed of the measured distance subtracted by the NC distance. The direction of the force, $u(x_i-x_j)$, is parallel to the path between both NCs.

$$F_{ij} = (L_{ij} - \|x_i - x_j\|)u(x_i - x_j) \tag{1}$$

The force will displace node i by Δx_i as shown in (2β). The force between the two nodes is weighed based on a weight w from (2α), which is calculated using the NC error e of each node compared to the total NC error for both nodes. The constant c_c is used to control how quickly the algorithm converges in order to avoid oscillation of NCs.

$$w = e_i / (e_i + e_j) \tag{2α}$$

$$\Delta x_i = F_{ij} c_c w \tag{2β}$$

Equation (2) requires that each node keep track of their local error e_i, which (3β) shows to be a rolling average of the error for each sample e_s, found using (3α).

$$e_s = \left| \|x_i - x_j\| - L_{ij} \right| / L_{ij} \tag{3α}$$

$$e_i = e_s c_e w + e_i (1 - c_e w) \tag{3β}$$

After enough measurements are made between the nodes, their NCs start to converge to values that can approximate the round-trip time between nodes without direct measurements between them. At the same time the NCs will adapt as the network changes.

Vivaldi NCs can work with normal Euclidean coordinates; however, it was found in [12] that these do not map very well to the structure of the Internet. For instance, most nodes will have a certain network access delay, but the backbone of the Internet can be approximated well using 2-dimensional Euclidean coordinates. Therefore, the Vivaldi algorithm was found in [12] to work best with 2-dimensional coordinates and a height vector to model the network access delay. The distance between two nodes is their Euclidean distance plus their height vectors x_h and y_h, as seen in (4).

$$\|x - y\| + x_h + y_h \tag{4}$$

When NCs get pushed from different sides and have nowhere to go in the Euclidian space they are pushed up by increasing their height vector.

NCs have the ability to be reused between distribution trees. Presence would require many different distribution trees, each rooted at a different presentity's RLS since every presentity may be a source of notifications. The process of building a new distribution tree is made simpler by reusing NCs globally within the presence system.

B. *Building the Distribution Tree*

In order to build the distribution tree, we propose a heuristic approach to reduce the traffic cost and average notification delay. Traffic cost is measured by taking the sum of each packet's end-to-end transmission time since during that time the packet will either be in transit or queued by a router. In either case it will use network resources during this time.

The tree will be constructed in a lazy manner to reduce traffic. Nodes will initially subscribe to the root node to be a child of that node. Notification messages will be used to reassign nodes by specifying a set of new children for the node receiving the message. This allows a new node to be delegated all the way down the tree in a single notification using no extra messages, only a slightly larger message size. The reassigned node will receive the notification from its new parent, letting the node know of the new parent assigned to it.

Before a node notifies its children, it will check whether it will reassign new nodes to its other children. For each of these new nodes, its parent will find the shortest distance between it and its siblings or parent, and assign the new node to the node that it is closest to. If the node is closest to its parent, then it will be notified directly, otherwise, it will be reassigned using a notification to one of the other child nodes. The effect of this approach is that nodes are effectively grouped into child clusters (clusters of nodes rooted at a child node), and the diameter of the clusters (which is the maximum delay between nodes in the cluster) should get smaller further down the tree.

NCs will be used to determine the distance between two nodes. Nodes with a large NC error will not be reassigned until the NC error drops below an error threshold. The error threshold, currently set at 0.5, prevents a random tree from developing early on since it results in large notification delays.

When a watcher wants to stop receiving notifications, they will send an unsubscribe message to their parent node rather than to the root of the tree. If the node that is leaving has children, then one of those children will need to be chosen to take the leaving nodes place. This is done by sending an empty notify message to promote a child node, which will specify the node's new parent and may include a list of new children for that node. Empty notifications are partial notifications that do not specify any presence changes. These will not need to be forwarded to notify children, but will need to be used to let new children know of their new parents using an extra header. The unsubscribe message will need to specify the promoted node as taking the leaving node's place using an extra header. These extra headers are explained in section C.

When a node is being promoted to take its parent's place, a value called the family delay is calculated for all the child nodes. A family refers to a parent node and all of its children. The family delay is calculated in (5), where x_i, x_j, and x_p are NCs for the current child being considered, a sibling, and its grandparent respectively. N is the number of candidates for becoming the parent node. The node with the lowest family delay is selected as the new parent of the family.

$$family\_delay(j) = N \cdot \left\| x_j - x_p \right\| + \sum_{i}^{N} \left\| x_i - x_j \right\| \quad (5)$$

The weight of N for the delay between the grandparent and the candidate is based on the fact that each child would need to wait for that delay before a message is sent to them. This formula favors reducing average delays in order to choose parent nodes with lower network access delays that are close to the other nodes in the family.

C. Extending the SIP Event Notification Framework

To perform tree-based notification dissemination, a SIP extension is proposed to modify the behavior of the SIP event notification framework in a standard backwards compatible way. This way the extension would be usable for other event notification packages in addition to the one for presence.

The SIP extension will be identified by an option tag. Support for the extension is advertised by the watcher when they send a SUBSCRIBE message to a presence server with the extension's option tag included in SIP's *Supported* header. The response will include the option tag in the *Require* header if the presence server decides to use this extension. After this point the SIP extension can be used, so semantics of the protocol described in section B should be followed and extra headers could be used.

The SIP extension will allow messages to include NC information in a *NCInfo* header, which would store floating point values for the position (x, y, height), and the NC error. This will allow NCs to be updated and sent to other nodes to use. The SIP protocol uses request-response pairs so that it can be used over an unreliable network protocol like the User Datagram Protocol (UDP). As a result, nodes sending requests can store the time the request was sent, then calculate the delay when the response is received in order to update the node's NCs. An ACK request could also follow a request-response pair to allow both sides to update their NCs.

In order to build a multicast tree to distribute notifications, extra headers will be needed to move nodes in the tree. The two headers for this purpose would be *NewChildren* to assign children to a node and *NewParent* to notify a node that it has been moved to a new parent.

Section VI describes additional aspects to the protocol that have not been implemented and tested in the simulation, but first the results of simulations are discussed in section V.

V. Simulation

The distributed notification protocol described in this paper is being tested using a simulation. The new protocol was implemented for the ns-2 network simulator.

The simulation uses median round-trip times available from a 72-hour PlanetLab ping trace, described in [13], to add delays for all the messages sent based on the source and destination of the message. The ns-2 simulation uses a star topology, with the center node acting as a router that adds delays to messages and emulates the delays on the Internet.

All the outer nodes in the star topology act as RLSs, sending and receiving notifications on behalf of the users. The relationships between users are based on their social relationships, which impact presence traffic patterns. These social relationships are similar to friend relationships on a social network like Orkut, so real world data are used that were collected from public data available from the Orkut social network. The dataset that is being used for the simulation is from the research presented in [14]. Presence status changes are assumed to have a Poisson distribution in the simulation, with the frequency of these events being proportional to the number of users. When a status change event occurs, a presentity is chosen uniformly at random for a status change and their watchers are notified by the RLS. The social network data are used to determine a user's contact list.

When the simulation is being initialized, users are mapped to RLSs, so there may be multiple users mapped to a single RLS. The traffic between the PUAs and their RLSs are not part of this simulation, but a presentity's status change affects their RLS's behavior. When a presentity's status changes from offline, their RLS makes sure they are subscribed to all the presentities on this user's contact list. When a presentity's status changes to offline, then the RLS unsubscribes from all the presentities that are no longer being watched by their users. Initially all presentities are offline and NCs are reset to the origin with a full error of 1.0.

The simulation measures the notification delay whenever a node receives a non-empty notification. It also measures traffic cost in the router node by counting the number of packets sent and keeping a sum of the transmission time of all the packets. Per second averages are taken over all the nodes for notification delay and over all end-to-end messages for traffic cost. Each second is considered individually since the initialization period has different traffic behavior. The simulation was run with 500 users assigned uniformly at random to 60 RLSs, and status changes occur according to a Poisson distribution with a mean time of 20 seconds between status changes for each presentity. The results of running the simulation can be seen in Fig. 1 and Fig. 2.

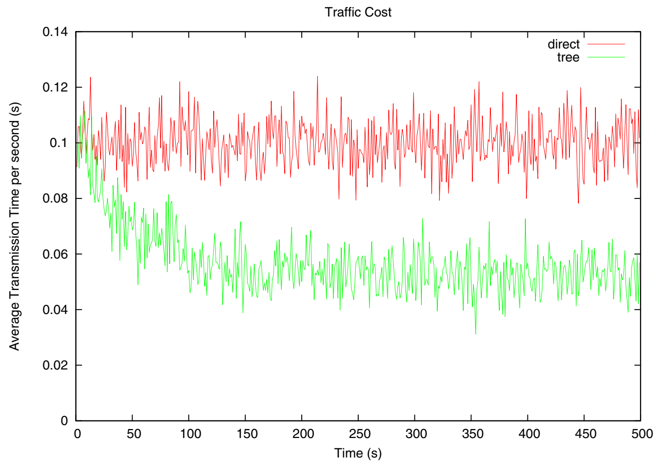

Figure 1. Per second average traffic cost

Fig. 1 shows that tree-based distributed notifications can cut the network traffic in half compared to direct notifications. The traffic cost is higher at the start of the simulation since NCs have a high error to start with and accurate NCs are needed to create the distribution trees.

Fig. 2 shows a small additional average delay for tree-based notifications compared to direct notifications. However, without the error threshold the notification delay would spike up and drop down at the start of the simulation because of trees being created with inaccurate NCs.

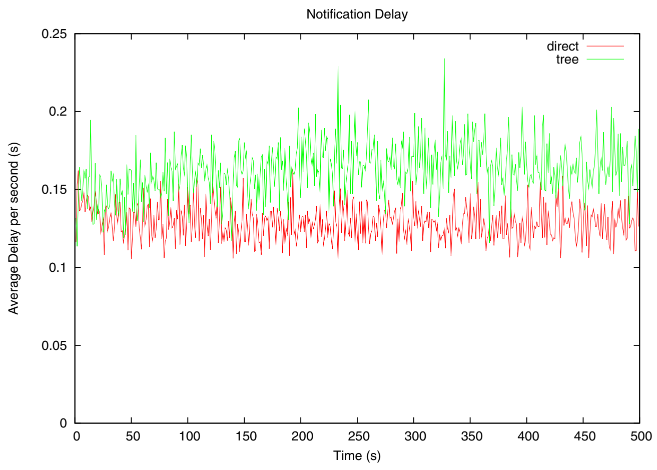

Figure 2. Per second average notification delay

Currently the simulation prevents packets loss, so no SIP retransmissions are used. Reliability will be a topic of further research along with the topics covered in the next section.

VI. Future Work

SIP retransmission still needs to be implemented in the simulation to test the impact of packet loss and node failure. Subscriptions are soft-state, because of their *Expires* header, so parent node failure could be detected. Orphaned nodes could subscribe to the root node to reconnect themselves and their descendants to the tree.

The current proposed protocol piggy-backs NC updates on existing traffic, which is mostly between neighbors. This increases prediction error [12], but a *neighbor decay* technique from [10] could improve these results without extra overhead.

Another improvement to the protocol could be to use (5) to replace non-root parent nodes with a new child if this could lower the family delay. The parent node could be represented as a child node for the sake of these calculations. If the parent node were replaced, then it could become a new child of the promoted node so that it finds its place in the tree.

VII. Conclusions

This paper has shown some early results from using tree-based notification dissemination in presence based on NCs. The results show improvement on standard IMS presence. The traffic cost was reduced without significantly affecting the average delay. We attribute this to our low overhead tree building protocol that uses NCs to place watchers in notification trees. The lack of overhead makes our protocol useful even for transferring relatively small quantities of data. Further work aims to improve these results.

IMS presence traffic was simulated using some real world data. The similarity between social networks and presence relationships were used to model the behavior of users, and ping traces were used to simulate network delay. There are still limitations for the simulation to address. Network errors are not simulated, so their effect is not included in the results.

Our proposed SIP extensions apply to the SIP event notification framework in general such that it could directly be used for other SIP event packages besides presence. Therefore, the benefits of using NCs for tree-based notification dissemination are worth pursuing further. The heuristics that are used for building a notification tree could also be used for other systems that do not use SIP.

References

[1] S. Islam and J. W. Atwood, "Sender access control in IP multicast," in Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on, 2007, pp. 79-86.

[2] A. B. Roach, "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[3] J. Rosenberg, "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.

[4] A. B. Roach, B. Campbell, and J. Rosenberg, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", RFC 4662, August 2006.

[5] J. Rosenberg, et al., "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[6] H. Sugano, et al., "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

[7] G. Camarillo and M. A. Garcia-Martin, The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds. Wiley, 2004, pp. 303-330.

[8] A. Ikram, M. Zafar, N. Baker, and R. Chiang, "IMS-MBMS convergence for next generation mobile networks," in NGMAST '07: Proceedings of the 2007 International Conference on Next Generation Mobile Applications, Services and Technologies, 2007, pp. 49-56.

[9] R. Lin, H. Zou, Y. Zhao, and F. Yang, "A novel approach to optimize information dissemination in IMS presence system," in NEW2AN '08 / ruSMART '08: Proceedings of the 8th International Conference, NEW2AN and 1st Russian Conference on Smart Spaces, ruSMART on Next Generation Teletraffic and Wired/Wireless Advanced Networking, 2008, pp. 187-198.

[10] J. Ledlie, P. Gardner, and M. Seltzer. “Network Coordinates in the Wild,” in NSDI'07: Proceedings of USSENIX Network Systems Design and Implementation, 2007, pp. 299-311.

[11] Y. Chen, Y. Xiong, X. Shi, J. Zhu, B. Deng, and X. Li, "Pharos: accurate and decentralised network coordinate system," Communications, IET, vol. 3, pp. 539-548, 2009.

[12] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in SIGCOMM '04: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2004, pp. 15-26.

[13] P. Pietzuch, J. Ledlie, and M. Seltzer, "Supporting network coordinates on PlanetLab," in WORLDS'05: Proceedings of the 2nd Conference on Real, Large Distributed Systems, 2005, pp. 19-24.

[14] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, 2007, pp. 29-42.