2008 IEEE Region 10 Colloquium and the Third International Conference on Industrial and Information Systems, Kharagpur, INDIA
December 8 -10, 2008
Paper: 412

# A Bandwidth Aware Topology Generation Mechanism for Peer-to-Peer based Publish-Subscribe Systems

Abhigyan, Joydeep Chandra, Niloy Ganguly
Department of Computer Science & Engineering, Indian Institute of Technology, Kharagpur
abhigyan.sharma@gmail.com, joydeep@cse.iitkgp.ernet.in, niloy@cse.iitkgp.ernet.in

***Abstract*—A publisher subscriber system is an event notification service where events generated by the publishers are routed to subscribers with matching subscriptions. Our work on publisher-subscriber systems in peer-to-peer (p2p) networks is motivated by the question : "What if the nodes in the network do not have the bandwidth to route all the events generated?" In such a case nodes may not receive all notifications because of lack of bandwidth, and even those with high bandwidth may starve for notifications if its neighbours do not have the bandwidth to send event notifications. In this paper, we consider *Sub-2-Sub* architecture for content based publisher subscriber systems and show with the help of simulations that the event dissemination rate will actually be much slower than expected when the publishing rate is very high. We propose a topology design and event dissemination mechanism which removes nodes which become bottleneck in dissemination. Side by side, it allows a node to receive more events if it has more bandwidth. Our system is built on top of Sub-2-Sub publisher subscriber system, and provides much improved performance over it in terms of percent of event received by nodes, and ratio of uploads to downloads (which shows the removal of bottleneck nodes in dissemination).**

## I. Introduction

PUBLISHER Subscriber system is an event notification service where the publisher of the event is ignorant of the subscribers for that event. The goal of the system is to send the event notification to the subscribers interested in that message. An example of a publisher-subscriber system is Google Alerts [1]. In this case users register the search queries for which they wish to receive notifications. As new content is found on the internet by Google crawlers, it is matched with the registered search queries. The most relevant new content (e.g. top 10 results) matching the query is sent to the user. In this case the publisher (website owner) is unaware of the subscribers (users interested in that content), but the publisher-subscriber system of Google alerts makes it possible for the subscribers to receive notifications for new events.

Implementing scalable publisher-subscriber systems is a challenge. Each incoming event would have to be matched against registered subscriptions to find the matching subscriptions and then sent out each second to all matching subscribers. If we consider an event rate of thousands per second and each event being routed to thousands of subscribers, the resources (computation cycles and number of messages sent) could run into millions. This motivates the need for a distributed system which is scalable to the size of the internet today. The requirements of a publisher-subscriber system make Peer-to-Peer networks an attractive model for scalable solutions. Using a p2p model, the event notifications can be sent to matching subscribers by multicasting, which distributes the load of sending enormous number of messages from few servers and links to all nodes spread throughout the network. Moreover, a p2p routing scheme that distributes the computational load among all nodes in the network evenly would cause minimal computational cost at each node.

So far many solutions have been proposed which have used DHTs such as Chord [8] and Pastry [3], R-trees [4] and hypercube [5]. Nearly all of these solutions have focused on improving the scalability, minimizing dissemination time (hop counts to disseminate to all users), supporting complex subscriptions (such as range queries), and reducing the number of false positives (subscriber receiving events which do not match their subscriptions) and false negatives (an event notification which was not received by the subscriber for that event). Many of these solutions achieve acceptable bounds, such as very small number of false positives; no false negatives and dissemination hop count which is logarithmic in the number of users.

But, one important component missing in all of these publisher-subscriber models is the bandwidth constraint of the network. One basic assumption throughout is that any number of messages can be sent from one node to another with zero latency. On challenging this assumption, the performance metrics above change. For example,

- What if the subscriber does not have sufficient bandwidth to receive all the matching events? How can any system ensure no false negatives in such a case?
- What if there is a limit on upload bandwidths? In such a case there would be buffer delays at each node. A large rate of new events and small upload bandwidths would mean increased buffer delays at every node. Even though the number of hops to disseminate is logarithmic, the actual time taken to disseminate may be much greater.
- What if a node has the bandwidth necessary to receive all events, but its neighbours do not have the sufficient bandwidth to route event notifications to them? The node would be unfairly starved for event notifications in this case.

We now examine some of the systems with regards to questions above. The solutions based on DHTs [3], [8] and some of the other solutions [10] propose a rendezvous node for a topic/range of values. In such a case, the dissemination capabilities within the range become limited by bandwidth of the rendezvous node, and it could limit the rate of notifications sent to other subscribers. Dissemination using trees such as R-trees [4] faces the problem that every event routed to child nodes must pass through parent. If the parent node does not have sufficient bandwidth then its child nodes are deprived of notifications. Meghdoot [2] proposes an overlay where new node, irrespective of its own subscription, is assigned the range of events where more new events are being generated. As a result, the node has a large number of false positives, since it keeps receiving notifications outside of its subscription range.

These systems are unsuitable for deployment on internet, where nodes have different bandwidths and the event rates could potentially be large. We propose that an optimal solution should try and achieve the following objectives:

1) Maximize the number of events received by each node
2) Ensure that a node should receive as many messages as it can send out. This strategy removes bottlenecks in dissemination, ensures fairness and acts as incentive to the nodes that spend more bandwidth.

978-1-4244-2806-9/08/$25.00©2008 IEEE

We propose a topology and event dissemination mechanism that fulfills the above stated objectives. Our solution is built on top of Sub-2-Sub, a state-of-the-art publish-subscribe system by Voulgaris et al [7], which has an unstructured overlay in contrast with other structured overlays such as DHTs. Sub-2-Sub performs very well on some of the metrics discussed above. But, on our experiments where rates of new events were high, we found that Sub-2-Sub had a high percentage of false negatives for a major fraction of the nodes. This happened because the nodes who received a large fraction of new events did not disseminate them to other nodes due of lack of bandwidth.

We keep the basic mechanism of event dissemination the same as in Sub-2-Sub. But we redesign the topology so that event dissemination started from nodes having more bandwidth towards the nodes which have less bandwidth. We designed a mechanism by which nodes with smaller bandwidth can limit the rate of messages being sent to them. Finally we allow nodes to change their position in the overlay. Thus nodes having high bandwidth links that are starving because of low bandwidth neighbors can reorganize the overlay and find suitable neighbors so that it receives adequate events.

The results show that these modifications reduce the percent of false negatives significantly for majority of the nodes. It also ensure fairness by cutting down messages to those nodes which were receiving a large number of events but were unable to send out messages.

The paper is organized as follows: Section II defines exactly our publisher-subscriber problem and our assumptions; Section III describes Sub-2-Sub, the problems with its topology and the motivation behind our design; Section IV details our system architecture; Section V contains simulation details and results; Section VI concludes the paper along with a critique of some of the other approaches.

## II. Publisher-Subscriber Model

Both our system and Sub-2-Sub fall under the paradigm of Content-based Publish-Subscribe model (which is different from topic-based publisher-subscriber system where every event falls under a predefined topic). In content-based systems, an event is a value over a set of $k$-attributes represented as $A_1 = a_1, A_2 = a_2, \ldots A_k = a_k$. A subscription is specified over a set of attributes $A_1, A_2, \ldots A_k$ such that $A_i = [submin_i, submax_i]$.

Ours is a p2p-model where the nodes form an overlay network efficiently disseminating new events. Every node can either be a publisher, or a subscriber or both. Whenever a node publishes an event, it sends it to the pub-sub system which disseminates the data to the interested subscriber nodes (if any). The pub/sub system can be considered as a daemon process running in each node that accepts an incoming event and runs a thread to handle the dissemination of that event to other nodes.

We have built the system for one attribute only. But, ideas in this paper can be extended to build a k-attribute system also. Like several of the existing systems (including Sub-2-Sub), our system is designed for numerical attributes only. For such systems, character strings have to be mapped to a numerical value. In our system, events are integer values and every subscription is an integer range $[a_1, a_2]$.

In the next section we describe the working of a p2p content-based publisher subscriber system proposed by *Voulgaris. et. al.* named Sub-2-Sub [7].

## III. Sub-2-Sub

The Sub-2-Sub is built using two protocols, CYCLON [6] and VICINITY [9]. The CYCLON protocol creates a distributed random graph. The VICINITY protocol clusters nodes together based on a similarity metric. In both the protocols, neighbours gossip with each other. They periodically exchange a subset of their neighbours, and after exchange update their neighbour set by including some of the nodes received.

The topology of Sub-2-Sub consists of three types of links: random links, overlapping-interest links and ring links. Random links at each node are links to a random subset of nodes in the network. These links are maintained by protocol CYCLON. Overlapping-interest links are a subset of those nodes which have subscriptions with values closest to (or overlapping with) this node's subscription. VICINITY Protocol manages these links. Ring links are nodes which have subscriptions overlapping with current node and have IDs closest to this node. For a range $(x_i, x_j)$ of a node's subscription, the ring link is established with that node whose subscription covers this range and whose ID is closest to this node than any other ID. There are two sets of ring links, upper ring links consisting of nodes which have ID greater than this node and lower ring links consisting of nodes having IDs smaller than this node. Another instance of VICINTY protocol finds the ring links for each node. The topology is not a static one, but

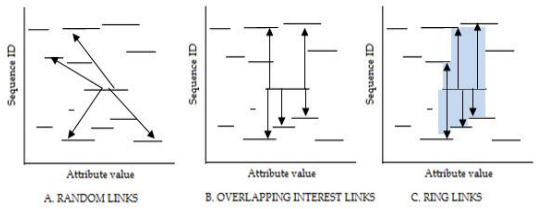


Fig. 1. The figure shows the three kinds of links formed in Sub-2-Sub as stated by Voulgaris et. al. From the left, they are random links, overlapping interest links and ring links.

dynamic. The random links keep changing after message exchanges with neighbours and ring links and overlapping-interest links getting updated in face of node-failures. The event dissemination is done in two steps:

1) *Finding first matching node*: A greedy algorithm is followed where each node sends the event to that neighbour node whose subscription is closest to the event. Simulations show that within very small number of steps a matching node is found.
2) *Disseminating to other nodes*: For every event, all the matching nodes form a *ring* due to presence of upper and lower ring links. When a node receives a new event, it sends it to its upper and lower ring links. It also picks a random neighbour from its overlapping interest links whose subscription matches this event and sends it the message. This message effectively shortcuts the *ring* and experiments indicate that even for long ring sizes, dissemination can occur in logarithmic number of hops.

### A. Discussion of Sub-2-Sub Topology

The topology of Sub-2-Sub because of being random presents cases which can prevent satisfactory service to all nodes. We discuss some of the major problems here:

1) *Large number of ring neighbours for long range subscriptions*: Long range subscriptions can have potentially many ring neighbours. For $2^m$ length subscription there can possibly be $2^{m-1}$ ring neighbours of length 2. This would consume bandwidth due to vicinity exchanges, which leaves less bandwidth for event notifications.
2) *Subscriptions with insufficient bandwidth*: If a node is not able to send out the messages it receives, then such a node becomes

a bottleneck in event dissemination. Apart from increasing dissemination time, it may prevent other nodes from receiving events.

3) *Starvation faced by high bandwidth nodes*: All the nodes in Sub-2-Sub would receive events at the same rates irrespective of their bandwidth. This pace would be set by the average bandwidths of the nodes and hence if the rate of new events is more than the average rate, the higher bandwidth nodes have no way of receiving them.

*The common thread in the above problems is the mismatch between the neighbouring nodes in the overlay, either in terms of bandwidth or the length of subscription.* We sought to end this difference by grouping nodes having similar bandwidth per unit length of subscription. This ensures that the rate of dissemination of events is same at every node within a group. In this way groups having low to high rates of event dissemination are formed. The dissemination starts from the node having highest bandwidth and then events are forwarded to nodes having lower bandwidths. The nodes with lower rates of dissemination can limit the number of events that is being sent to them to match their rate of dissemination. Further, a node is not fixed to a group; it can move to groups with higher and lower bandwidths to maximize the events it is receiving.

## IV. System Architecture

We present the details of our system architecture that consists of the topology generation mechanism, the event dissemination mechanism, followed by the flow control and cell switching mechanism.

### A. Topology

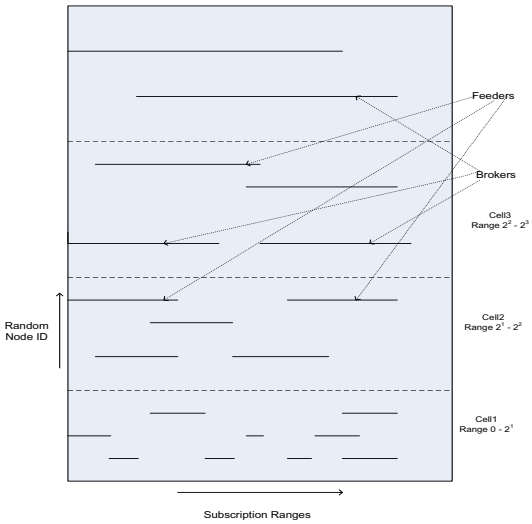


Fig. 2. The cells along with their subscription ranges. In each cell the nodes are ordered along the y axis according to the increasing node id, whereas x axis shows the subscription range of the node.

1) Each node now has a $\langle ID, CellNo \rangle$ where $Cellno = 1, 2, 3, \ldots$. The nodes are ordered based on their cell number. Nodes within the same cell are ordered by their IDs. (Fig. 3). The nodes with smaller cell numbers have more bandwidth available per unit.
2) Initially a node with subscription length $L$ is placed in cell $M$ such that $2^{M-1} \leq L < 2^M$. The objective is to place the nodes with low range subscriptions at a lower cell as compared to the high range subscriptions. Fig. 2 represents the position of the nodes in various cells.
3) The ring links are formed by VICINITY protocol using the new ordering of nodes. This ensures that the ring links are formed between nodes in the cell and also between nodes across the cell boundaries. For upper ring links, a node will choose the node in the next higher cell with the least id. For lower ring links, a node will choose the node in the next lower cell with the largest id. (Fig. 3) The random links and overlapping-interest links are independent of ID and remain the same.
4) Nodes having ring links to nodes in the lower cell will be termed as *brokers*, whereas the nodes in the lower cells that have ring links to nodes in the higher cells will be termed as *feeders*. The feeder directs all messages coming to it in its range to the broker nodes.(Fig. 3)

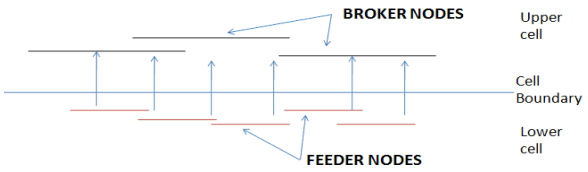


Fig. 3. The figure shows the brokers and feeders across a cell boundary

### B. Event Dissemination

The steps of the event dissemination mechanism is discussed next.

1) A new event is first sent to any matching node using the method as in Sub-2-Sub
2) *Routing to lowest cell*: The event is then sent to the lowest cell which contains a node with matching subscriptions. For this, each node receiving a new event picks the neighbour among its vicinity and ring neighbours which has a matching subscription and is smallest in the ordering of nodes. This process stops once the event reaches the lowest possible cell (cell 1) or no node which is smaller in the ordering is found. Once the node reaches the lowest cell, dissemination starts.
3) *Dissemination*: The dissemination of the data within each cell occurs as in Sub-2-Sub. It then disseminates from lower to higher cells. Across the cell boundary, events are sent through the feeders from lower cell to brokers at higher cell.

By default, no bandwidth or flow control mechanisms are imposed by any node. However if the number of events per unit cycle received at a node increases beyond a global threshold (say M per cycle), it initiates a flow control mechanism as discussed in the following subsection.

### C. Flow Control Mechanism

The goal of this mechanism is to limit the rate of events entering a cell. Our defined limit for cell $C$ is: A length of $2^{C-1}$ shall receive not more than $M$ events per cycle ($M = 20$ for our system).
By default no flow control is imposed at feeders. Flow control starts by generating a flow control event. When a node finds that *number of events in buffer* $\leq \frac{1}{4}$(*Max Buffer size*), it checks the events rates it is receiving. If the event rates exceed the defined limit for this cell, it generates a flow control event, for the range where event rates have exceeded. This event is disseminated down the cell preferentially over other new events. Hence it reaches any one broker in a few cycles. The first broker receiving the event becomes the controller for flow control.

978-1-4244-2806-9/08/$25.00©2008 IEEE

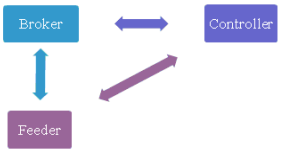


Fig. 4. Tripartite flow between broker, feeder and controller

*1) Controller, Broker and Feeder: Tri-partite flow control:* The flow control steps are as follows:

1) First, a controller ascertains if the events rates have actually exceeded within the flow control range. The controller (which itself is a broker) sends messages to other brokers (its neighbours) within the flow control range. (⟨*Check event rate, Range, Controller ID*⟩)
2) Broker replies back to controller with their event rates (⟨*Event rate, Broker ID*⟩) and forwards this message to their neighbouring brokers who are in the flow control range. Controller, on receiving brokers' replies, is able to decide whether event rate exceeds and flow control is needed.
3) To implement flow control, controller sends out a similar message to its neighbouring brokers (⟨*Flow control, Range, Controller ID*⟩). The brokers forward it to neighbouring brokers and to their feeders. item The coordinator waits for replies from feeders about the event rates at their site. Once all feeders in the range have responded (⟨*Event rate, Feeder ID*⟩), the controller calculates the sum of event rates. If event rate exceeds, it calculates limits for each feeder proportional to their event rates. It sends out the limit imposed on that feeder (⟨ *Max event rate limit, Controller ID*⟩). The feeders, while sending messages to brokers, stick to these limits.
4) After every 50 cycles, the steps 3 and 4 are repeated to redefine the limits based on new event rates at feeders or to cancel the imposed limit (if event rates have reduced below threshold).

### D. Cell Switching of Nodes

As stated earlier, depending on event rates, some of the nodes in the higher cells may need to switch to a lower cell. This is because, when a flow control mechanism is implemented in a higher cell, messages are sent up at a controlled rate by the feeders in the lower cells. Hence node that wants more messages to arrive and has adequate bandwidth can move down to a lower cell. This is done as follows:

1) *Information about higher event rates*: An initial problem is to find a mechanism by which nodes in higher cells can be informed that there are more events available at lower cell. For this, feeders whose message buffers exceed a threshold size (500 for our system) set a reserved bit in the message frames that are being forwarded to brokers above. The brokers also propagate the same messages with the bit set to the nodes within the cells for the matching subscription range. Thus nodes in the subscription range on receiving the message will get informed that messages are being dropped at a feeder in some cell below.
2) *Decide about switch*: If nodes receive $K$ dropped messages within the past $K$ cycles ($K = 10$ for our system), then they check if they have the sufficient bandwidth for a switch. To give an estimate, they check if their bandwidth can support twice the event rates which they are receiving now. If yes, the node decides to switch to lower node.
3) *Making the switch*: To switch from one cell to another, a node $M$ tries to find a ring neighbour for itself, in the new cell. For this, it sends out a *CELL SWITCHING* message to a neighbouring node (⟨*CELL SWITCHING, CELL NUM, ID*⟩). This message is forwarded from node to node until it reaches a node which can form a ring link. Each node selects it to forward to that node among its neighbours which it thinks is the best candidate for being a ring link of node $M$ in the new cell. Once a ring link is found, node informs its neighbours about the switch and switches to a new cell.
4) Nodes wishing to move to a new cell will have to move one cell at a time. If after moving to a new cell below, it finds its buffer fill up to the threshold value, it goes back up one cell following the same procedure as stated above.

## V. Simulation and Results

Sub-2-Sub can find the first matching subscription quite quickly as has been shown in [7]. Our goal was to test its event dissemination capabilities under different rates of events.

The efficacy of dissemination of events under different rates of new events can be tested with rings of various sizes, having nodes with different length of subscriptions. Keeping this objective, we took the following parameters for our experiments. In our experiments 1000 nodes are considered with subscription range $[a, b]$, $0 \leq a < b \leq 100$. The points $a, b$ are randomly chosen, which generated subscriptions of length from 1 to 100. The size of the rings in this case is found to be 30 to 300. We varied the rate of events from 50 events per cycle to 200 events per cycle and took our measurements. The events are uniformly distributed in the range $[0, 100]$. In our system, the download bandwidth is considered unlimited, but the upload bandwidth is limited per cycle. This matches with the behaviour of Peer-to-Peer file sharing clients in Bit Torrent where users usually set upload limits but download limits are unlimited to allow quickest possible downloads. Using a simple accounting scheme for calculating the bandwidth units used per cycle (e.g. *CYCLON* message costs 10 units and an event message costs 2 units), we could calculate the cost (in units) for sending all messages in any cycle. A node buffers all event messages it received during a cycle and disseminates only at the end of the cycle. It sends out at most as many messages which utilize the upload bandwidth fully or the buffer is emptied. This ensures that the upload bandwidth limitations are adhered to. We present our results for these three representative simulations which have been run for 1000 cycles each:

*Simulation A*: Every node has a bandwidth of 100 units and the event rates are 50 events per cycle. (Moderate rate of events)

*Simulation B*: Every node has a bandwidth of 200 units and the event rates are 200 events per cycle. (High rate of events)

*Simulation C*: Nodes have bandwidth of 100/200/300 units with equal probability and event rates are 50 events per cycle. (Different bandwidth nodes to demonstrate the efficacy of nodes switching cells)

### A. Results

We present the simulation results comparing the performance of our proposed architecture with Sub-2-Sub for various parameters like accuracy, message dissemination ratio, and latency and effectiveness of dissemination. Each of these parameters are defined next. *Accuracy and messages received*: Accuracy is defined as the percentage of events received out of the events that a particular node is supposed to receive. We find that in Sub-2-Sub, all nodes, irrespective of the

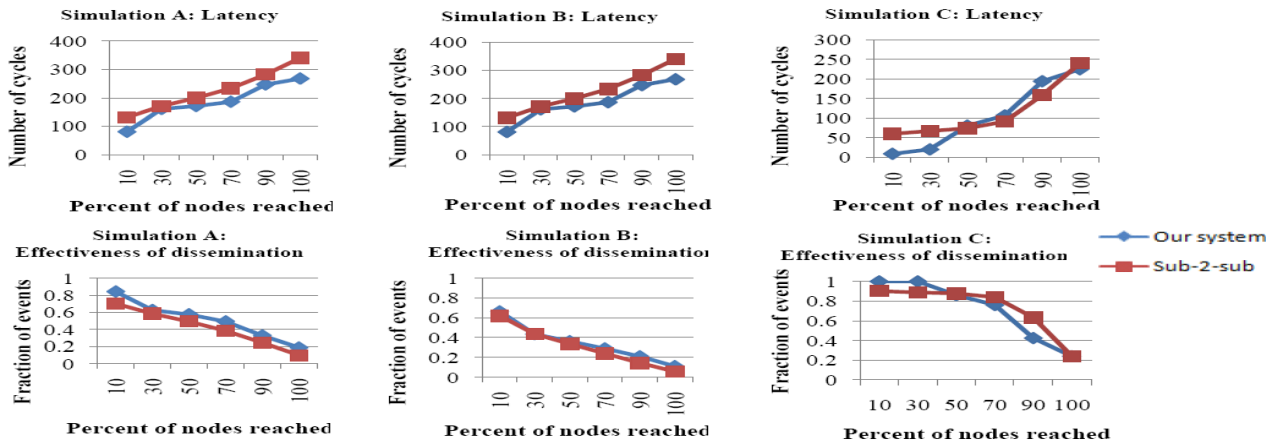


Fig. 5. Simulation results showing the latency and the effectiveness of event dissemination. The square dots in the graphs represent the results for the Sub-2-Sub and the diamond dots represent the results for our proposed architecture.Latency is determined by the number of cycles required for events to reach a certain fraction of nodes. Effectiveness of dissemination is determined by the fraction of events that reaches certain percentage of nodes.

length of subscriptions have similar accuracy figures. In our system, nodes having smaller subscription length have much improved accuracy. This difference in accuracy is more marked for a higher event rates (Simulation B). The accuracy for longer subscriptions goes down.

The number of messages received by a node during dissemination shows similar trends. The number of messages for very long subscriptions is cut down.

| Simulation A: Accuracy % | | | Simulation A: # of Messages | | |
|---|---|---|---|---|---|
| Subscription length | Sub-2-Sub | Our System | Subscription length | Sub-2-Sub | Our System |
| 1 to 4 | 3 | 90 | 1 to 4 | 1089 | 933 |
| 4 to 16 | 52 | 97 | 4 to 16 | 3771 | 6049 |
| 16 to 64 | 53 | 66 | 16 to 64 | 14336 | 15546 |
| 64 to 256 | 53 | 38 | 64 to 256 | 29094 | 17399 |

TABLE I

| Simulation B: Accuracy % | | | Simulation B: # of Messages | | |
|---|---|---|---|---|---|
| Subscription length | Sub-2-Sub | Our System | Subscription length | Sub-2-Sub | Our System |
| 1 to 4 | 44 | 82 | 1 to 4 | 2986 | 2387 |
| 4 to 16 | 45 | 94 | 4 to 16 | 11779 | 20180 |
| 16 to 64 | 42 | 41 | 16 to 64 | 41835 | 37881 |
| 64 to 256 | 39 | 26 | 64 to 256 | 78861 | 49486 |

TABLE II

| Simulation C: Accuracy % | | | Simulation C: # of Messages | | |
|---|---|---|---|---|---|
| Subscription length | Sub-2-Sub | Our System | Subscription length | Sub-2-Sub | Our System |
| 1 to 4 | 84 | 92 | 1 to 4 | 1880 | 1333 |
| 4 to 16 | 80 | 98 | 4 to 16 | 6964 | 5730 |
| 16 to 64 | 82 | 87 | 16 to 64 | 22215 | 20142 |
| 64 to 256 | 82 | 53 | 64 to 256 | 46590 | 24267 |

TABLE III

*Message Dissemination Ratio (MDR)*: The MDR is the ratio of messages sent by a node to the number of messages received. We found the longer range nodes, send close to half of the messages (50-60%) they received (Simulations A and B). These nodes are a bottleneck in dissemination and are responsible for low accuracy of several long range subscriptions. Our system has good MDR (close to 90%) for all the subscriptions lengths except very small length subscriptions. It shows that nodes send out nearly all events that they receive. This also validates the penalty we impose on long range subscriptions since they receive huge number of messages but send only half of them.

| Simulation A: MDR | | | Simulation B: MDR | | |
|---|---|---|---|---|---|
| Subscription length | Sub-2-Sub | Our System | Subscription length | Sub-2-Sub | Our System |
| 1 to 4 | 1.104 | 0.797 | 1 to 4 | 0.904 | 0.764 |
| 4 to 16 | 1.093 | 0.900 | 4 to 16 | 1.051 | 0.8901 |
| 16 to 64 | 1.000 | 1.026 | 16 to 64 | 0.984 | 1.008 |
| 64 to 256 | 0.597 | 0.963 | 64 to 256 | 0.546 | 0.973 |

TABLE IV

| Simulation C: MDR | | |
|---|---|---|
| Subscription Length | Sub-2-Sub | Our System |
| 1 to 4 | 1.026 | 0.749 |
| 4 to 16 | 1.099 | 0.913 |
| 16 to 64 | 1.024 | 1.018 |
| 64 to 256 | 0.747 | 0.969 |

TABLE V

Looking back at our goals — the improved accuracy (maximizing the number of events received) and good MDR (deliver as many events as one can send out) — show that we have made significant progress in finding optimal topology. We now show that our system does not compromise on other metrics and compares favourably with Sub-2-Sub.

*Latency and effectiveness of dissemination*: If we consider the latency i.e. the number of cycles needed for an event to reach a node and effectiveness of dissemination i.e. the percentage of nodes receiving an event, then our system performs comparably or better with that of Sub-2-Sub. The average number of cycles needed to disseminate information to nodes is not hugely different for our system and Sub-2-Sub. Our dissemination process takes more hops, due to first routing

to lower level cells and the presence of brokers at each level. These results show that the delays at the nodes are determining factor for latency, instead of the number of hops. It also implies that the queuing delay per message at each node is higher for Sub-2-Sub than our system. The effectiveness of dissemination graphs below show the fraction of events that reaches a given percentage of nodes. The metrics are similar for Sub-2-Sub as for our system.

## VI. Conclusion

In most of the previous works on publish-subscribe system, hardly any attention have been paid to the event arrival rate, heterogeneity in bandwidth of nodes and node processing capabilities of each node. We found that, by design, many of the existing systems are liable to poor performance on the internet where nodes have different bandwidths and rates of new events can potentially be large. Considering these problems, we designed a topology which aims to provide fair service to maximum number of nodes.

Our basic ideas behind this topology formation are: dissemination should proceed from bandwidth-rich zones to lesser bandwidth zones in the network; a flow control mechanism to limit the messages sent to low bandwidth nodes. To implement this, the algorithm gives node freedom to change its position in the topology to maximize the events it is receiving. Through experimentation, we have shown that the proposed method is effective in terms of the accuracy of the delivery of events and also produces better message delivery ratio. It outperforms and enhances the hitherto existing Sub-2-Sub system.

## References

[1] http://en.wikipedia.org/wiki/Google_Alerts.

[2] D. Agarwal A. Gupta, O.D. Sahin and A.E. Abbadi. Meghdoot: Content-based publish/subscribe over p2p networks. In *5th International Middleware Conference*. ACM/IFIP/USENIX, October 2004.

[3] M. Castro A. Rowston, A. Kermarrec and P. Druschel. Scribe: The design of a large-scale notification infrastructure. *In 3rd International Workshop on Networked Group Communication (NGC2001),*, 2001.

[4] Silvia Bianchi, Pascal Felber, and Maria Gradinariu. Content-based publish/subscribe using distributed r-trees. In *Euro-Par*, pages 537–548, 2007.

[5] Y. Choi and D. Park. Mirinae: A peer-to-peer overlay network for content-based publish/subscribe systems. *IEICE Trans. Commun.*, E89-B(6):1755–1765, June 2006.

[6] D. Gavidia S. Voulgaris and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2), June 2005. Special issue on Self-Managing Systems and Networks.

[7] Anne-Marie Kermarrec Spyros Voulgaris, Etienne Rivire and Maarten van Steen. Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks. *The fifth international Workshop on Peer-to-Peer Systems, Santa Barbara, CA, USA (IPTPS 2006)*, February 2006.

[8] P. Triantafillou and I. Aekaterinidis. Content-based publish subscribe over structured p2p networks. In *Third International Workshop on Distributed Event-based Systems (DEBS)*, pages 104–109, Edinburg, Scotland, U.K., May 2004.

[9] S. Voulgaris and M. van Steen. *Euro-Par 2005 Parallel Processing*, volume 3648/2005 of *Lecture Notes in Computer Science*, chapter Epidemic-Style Management of Semantic Overlays for Content-Based Searching. Springer Berlin / Heidelberg, 2005.

[10] Y. Zhu and Y. Hu. Ferry: An architecture for content-based publish/subscribe services on p2p networks. In *International Conference on Parallel Processing (ICPP)*, 2005.