# Locality-Awareness in a Peer-to-Peer Publish/Subscribe Network

Fatemeh Rahimian, Thinh Le Nguyen Huu, Sarunas Girdzijauskas

# Locality Awareness in a Peer-to-Peer Publish/Subscribe Network

Fatemeh Rahimian, Thinh Le Nguyen Huu , Sarunas Girdzijauskas

Swedish Institute of Computer Science
{fatemeh,thinh, sarunas}@sics.se

**Abstract.** Peer-to-peer publish/subscribe systems are promising solutions to provide distributed content distribution services at Internet-scale with low cost. One of the potential problems with peer-to-peer overlays, however, is the inefficient traffic and large delays, due to the mismatch between the physical network and the overlay topology. This paper introduces a locality-aware extension to a peer-to-peer publish/subscribe system, named Vitis. The ultimate purpose is to avoid communications over long-distance links, instead, nodes send data over short-distance and low-cost links, when possible, while maintaining an acceptable quality of service. We show, through simulations, that the average data delivery time is up to 40% improved. The cost to pay is at most 10% more relaying in the peer-to-peer overlay.

**Keywords:** Publish/Subscribe, Peer-to-Peer, Locality-awareness

## 1 Introduction

Distributed applications and services are getting significantly popular over the Internet. Social networks, news syndications, and multi-player on-line games are just a few such services one may use on a daily basis. One of the dominant communication protocols, used in many of these applications, is the publish/subscribe protocol. Work on the distributed publish/subscribe models, therefore, has attracted many researches in the field and several interesting solutions have been introduced [5, 6, 20, 22, 4, 11, 21, 24]. In particular, peer-to-peer publish/subscribe systems, such as [5, 6, 4, 11, 21], exploit the increasingly available resources at the edges of the network, in order to provide a scalable and reliable, yet cheap service to millions of users over the Internet. Although a few of these solutions are proven to be effective, a problem that exists in many peer-to-peer systems in general, is the mismatch between the overlay topology and the topology of the underlying network. Neglecting this mismatch in data-intensive applications could render these systems very inefficient. Since publish/subscribe systems are all about content distribution, and this content could be potentially massive, it is important to design proximity-aware publish/subscribe overlays.

In this paper, we propose a solution for incorporating locality-awareness into an existing topic-based publish/subscribe system, *Vitis* [21]. Currently in Vitis, nodes build up clusters by taking advantage of the subscription correlations.

More precisely, a node selects its *neighbors*, i.e., the nodes it directly connect to, based on the similarity of subscriptions. If two nodes have more topics in common, they have a better chance to become neighbors. Vitis nodes, however, are oblivious to node geographical positions, which may result in large traffic in the physical network.

The core of our solution is to get nodes gain more information about the location of the other nodes, thus, enabling them make a better decision on where to forward the data. As a result, most of the traffic is likely to be confined among nodes that are co-located in the same ISP or the same geographical area, and there is less communication between nodes lying on different continents or countries. Average delay to get some data delivered is also reduced, since a large portion of data forwarding is on the short links.

Through simulation and by using synthetically generated subscription, as well as real-world subscriptions from Twitter, we show that the average delivery time of the published events is improved by up to 40%. This improvement, however, comes at the cost of (at most) 10% more relay traffic overhead in the overlay. Since the total relaying traffic is only 4% of the whole overlay traffic, we conclude that our solution pays off, as it brings along up to 40% improved delivery time, which translates into more efficient communications in the physical network.

In the next section we briefly go through some of the related work for locality awareness in peer-to-peer networks. In Section 3, we first explain how Vitis works currently, and then elaborate on our technique to make the overlay locality-aware. In Section 4 we report the results of our experiments and in Section 5 we conclude the paper.

## 2 Related Work

In [26], X. Y. Zhang and Q. Zhang suggested *mOverlay*, an efficient overlay network, with locality of network hosts, that reduces the communication delay and cost. In order to gain information about the position of nodes, mOverlay utilizes a set of dynamic landmarks to measure the distance of nodes and group the close ones together. Having the estimated distances, nodes are assigned to the closest group they could find, thus, locality-awareness in the overlay is improved. Although mOverlay has shown improvements in gaining locality-awareness, it does have a few limitations and weaknesses. The group that nodes fall into, is dependent on the initial nodes that are introduced to it. Since the group choosing process terminates after a number of rounds, it may leave the new node in a group far from the optimal one. Furthermore, mOverlay does not mention anything about upper bound of a group, thus, some groups may get over crowded, while others are quite empty. Skewed distribution of nodes makes the cost of topology maintenance and cache exchanging within a group high. Lastly, the solution uses timestamps for selecting new group leader, and for this, it assumes nodes have synchronized clocks, which is not the case in reality.

Yuhao Lin et al. in [18] presented another solution for this problem. The solution, which is called *location-aware topology matching (LTM)*, continuously reconstructs the overlay by disconnecting slow connections and choosing physically closer nodes as logical neighbors. Similarly to previous solutions, nodes in LTM detect their distance to others by measuring roundtrip message delay. Based on this information, LTM nodes can detect and cut most of the expensive and redundant links, thus makes the topology more efficient. The problem with LTM is when nodes do the optimization, a node A may remove the link between itself and another node B without knowing that this link is essential to node B. This could leave node B disconnected for a while until the next round when node B tries to reestablish the connection to A. The process of removing and creating link can go on for a long time until there are significant changes in connections that result in a different topology. Thus, it creates an overhead in the cost to maintain the overlay and the overlay changes all the time without making the topology better. Also, LTM does not help the new nodes to initially acquire an appropriate position in the overlay. Optimization process happens mainly on constructed overlay, in which nodes are randomly connected. Thus, it may take a long time before the overlay stabilizes. Furthermore, LTM only works on a small scale and does not fully optimize the whole peer-to-peer network, resulting in a network that is partially optimized here and there but not on the whole.

Using an approach similar to mOverlay [26], the authors of [23] and [13] proposed a solution, named *distributed binning* to arrange nodes into areas that will exploit the geographical information to gain locality-awareness. The idea behind this is that nodes are arranged into bins, with unique identifiers. Nodes within the same bin are connected using, so called, short links, and communications over short links are rather cheap and fast while links connecting bins together (long links) are more expensive. Ideally, there should be only one single link between two bins, however redundancy could be used to make the system more robust against failure. As before, network latency is calculated by counting the average time of round communication, for instance the average of 10 pings. Unlike [26], distributed binning uses a set of static landmarks to determine the relative position of nodes. Any static systems such as DNS root servers or a set of well-known, widely distributed servers could be used for this purpose. On the scale of the Internet, it is estimated that 8 to 12 landmarks is enough for the system. This solution experiences the similar trade-offs to those that use dynamic landmarks. First of all, since the landmarks are static and the latency level division is predefined, bins are rather static and not very flexible. This leads to the situation that some bins are over-crowded than the others and maintaining the overlay within those bins becomes too expensive.

Another approach is to embed geographical position into node identifiers as proposed in [27, 8, 9]. In those solutions, a node ID can have hierarchical prefixes representing regions, sub-regions, countries, ISPs and actual node ID. Nodes in the same area will share some similar bits. This is rather a static solution and it requires a policy to distribute node IDs among regions based on the population of nodes so that there are enough node ids for a crowded area, while not allocating

too many for sparse ones. Yet another approach is to use an indexing system in order to enable nodes to autonomously acquire their localization information and exchange it with other nodes in the peer-to-peer overlay [7, 19]. There are also solutions that collaborate with the ISPs in order to provide users with accurate localization informations [1, 25]. However, these solution require that ISPs deploy and maintain the equipments for ordering the list of possible services.

## 3 Locality-aware Publish/Subscribe

Previous studies to improve locality-awareness have many flaws and limitations. Some constructs an overlay that is too complicated and too costly to maintain. Some are inefficient and only improve locality-awareness of some areas in the overlay rather than a complete solution on a large scale. In this paper, locality-awareness improvement is studied based on Vitis [21], a gossip-based [14] hybrid overlay for Internet-scale publish/subscribe developed by Rahimian et al.

### 3.1 Vitis - a topic-based publish/subscribe system

Vitis [21] is a topic-based peer-to-peer publish/subscribe solution that requires only a bounded node degree, but generates very low traffic overhead, compares to its counterparts. It uses a novel technique for overlay construction, in which nodes exploit the subscription similarities and select as neighbors, nodes with whom they share the most topics. It denotes a *cluster* for a topic as a maximal connected subgraph of the overlay, which includes a set of nodes that are all interested in that topic.

Due to the bounded node degree requirement, there is no guarantee that all the nodes, which are interested in a topic, connect together. In fact, any number of clusters for the same topic can emerge in different parts of the overlay. Although nodes inside a cluster are reachable from one another, in order to make sure a published event for a topic is delivered to all the subscribers, all the clusters of that topic must be also linked together. The path that connects different clusters of the same topic is called *relay path*. Such a path includes nodes that are not interested in the topic themselves. We refer to these nodes as *relay nodes*, hereafter. The challenge is to decrease the number of required relay nodes, while making sure that all the clusters associated with a topic (and therefore, all the nodes interested in that topic) are linked together. To enable relaying between the clusters, Vitis uses *rendezvous routing* [3] on top of an unstructured overlay. The node that has the closest identifier to a topic identifier is designated as the *rendezvous node* for that topic.

Moreover, Vitis nodes utilize a novel algorithm to select a number of representative nodes, as *gateways*, in each cluster. The number of gateways for a cluster is proportional to the diameter of the subgraph that represents the cluster. Gateway nodes are responsible for employing the navigable small-world overlay to connect to other clusters for the same topic. They perform a greedy

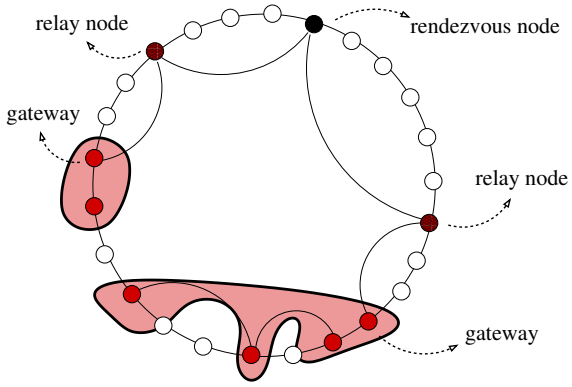



**Fig. 1.** Vitis Overlay Structure: Multiple clusters of a topic are connected together via gateway, relay, and rendezvous nodes.

lookup for the topic identifier, and all meet at the same node, i.e., rendezvous node. The structure of Vitis is depicted in Figure 3.1.

Whenever a node publishes an event on a topic, it sends a notification to those neighbors in its routing table, which are interested in that topic, or act as a relay node for the topic. A node that receives a notification, pulls the event from the sender and forwards the notification to all its own interested neighbors. As a result, the notification propagates inside the cluster of the publisher node. When the notification is received by the gateway node, it is forwarded along the relay path. The notification goes up to the rendezvous node and again down the other existing relay paths, if any other cluster for that topic exists. It, then, reaches the gateway node(s) of those clusters, and will be flooded inside those clusters, accordingly.

To enable an efficient clustering, Vitis nodes use a utility function to rank any arbitrary nodes that they come across. The ranking is based on the similarity of interests between the two nodes, which in turn, defines the benefits of establish a connection between the two nodes. Every node ranks the nodes it learns about, and selects the highest ranked nodes as its neighbors in the overlay. By doing so, a node ensures its neighbors will have the most similar topics to itself and an interesting event is more likely to be interesting for its neighbors too, thus keeping relay traffic overhead very low. Every node also selects a few small-world connections [15] for the sake of correctness and efficiency of routing in the overlay.

### 3.2 Locality-awareness improvement in Vitis

Vitis can further be improved to deliver a better performance in terms of the actual network traffic. Currently, Vitis puts the nodes with similar interests into clusters and most of the data dissemination happens inside these clusters. Although this optimizes the amount of traffic in the overlay network, it is not

optimal for the actual network traffic, as nodes far apart in the physical network may be positioned in the same cluster and exchange huge amount of data, in which they have a common interest. Communication between nodes, thus, becomes costly, high-delayed and bandwidth-limited. On the other hand, connecting nodes to the nodes with close-by geographical coordinates, does not necessarily result in a lower communication cost at the end, because the traffic is not anymore confined to efficient clusters, and many uninterested nodes may have to relay data to ensure that it reaches all the corresponding subscribers. Hence, we will have more relay traffic and a longer delivery time.

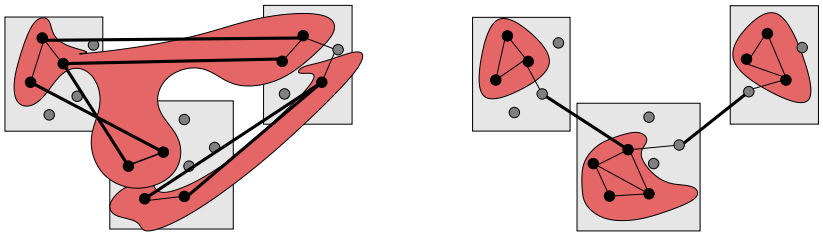
Hence, to improve the performance of Vitis, it is necessary to maintain the clusters of nodes with similar interests, while reducing the geographical divergence inside these clusters. Putting differently, we would like to enhance locality-awareness, while keeping the relay overhead trade-off at an acceptable level. As mentioned before, a number of connections in Vitis are chosen based on a utility value, and the utility value is calculated purely based on the similarity in node subscriptions. A node has no clues about where the other nodes are, thus its view does not reflect its actual physical connections. In order to make the overlay more similar to the physical world, nodes need to also consider the notion of distance when selecting their connections. Note that, small-world connections are selected just as before and remain untouched by this improvement.

We introduce a utility function that could dramatically change the behavior of the overlay and help enriching nodes with proximity knowledge. The new utility function is shown in 1.

$$utility(i,j) = (\frac{1}{d(i,j)})^n \cdot \frac{\sum\limits_{t \in subs(i) \bigcap subs(j)} rate(t)}{\sum\limits_{t \in subs(i) \bigcup subs(j)} rate(t)} \tag{1}$$

where $i$ and $j$ are two nodes in the system and $sub(i)$ and $sub(j)$ are the sets of subscriptions of the two nodes, respectively. Note, a subscription is a set of topics that a node is interested in. In reality, data/event about some topics are published more frequently than the other topics. The term $rate(t)$ indicates the publication rate of events on topic $t$. If a topic is hot and has many publications, the overlay builds more efficient clusters for it. Without losing the generality, we assume the publication rate of all topics are equal. The term $d(i,j)$ denotes the *distance* between two nodes $i$ and $j$. This distance is measured as the round-trip latency. As a result, nodes with a long communication delay, which are very likely in different ISPs or distantly located, are less preferable. Now, the question is how to tune the effect of locality versus subscription similarity. In other words, we can have different weights (denoted as $n$ in the utility function) for distance impact relative to common interests impact, and we do not know yet what is the optimal configuration. In Section 4.1 we carried a set of simulations to select an appropriate value for $n$.

Nevertheless, the utility value is not only reflecting how similar the two nodes are, but also how close they are in the physical network. Link selection, thus,



(a) Connections in the overlay network are oblivious to locality. Clusters are span across different ISPs, thus, many overlay links cross the boundaries of these ISPs.

(b) The overlay topology is locality-aware. Clusters are confined to the ISPs and few links cross the ISPs in order to connect the clusters together.

**Fig. 2.** The impact of locality-awareness on the overlay topology. Nodes, colored in black, are all interested in the same topic. ISPs are shown in light gray boxes, and the red (dark gray in the printed version) regions represent the clusters in the overlay network.

becomes more locality-aware and nodes make better decision in term of limiting the cost of communication. The closer two nodes node are, the better chance they have to establish a connection, if they have some common interests. On contrary, although two nodes may have many common interests, if they are so far away, there is very little chance that they get directly connected and stay in the same cluster.

Having a closer look at the topology of the overlay, we observe that, compared to the original Vitis, the clusters are now bigger in numbers, but smaller in size. Figure 2(a) shows a part of an original Vitis overlay, which does not take locality into account. Clusters are span across different ISPs, thus, many overlay links cross the boundaries of these ISPs. In the locality-aware version, depicted in figure 2(b), however, overlay connections are not anymore oblivious to the geographical distance of the nodes, thus, clusters contain nodes that are close-by in both physical space and subscription space. Therefore, data dissemination mostly happens inside the clusters and very few links connect the clusters that reside in different ISPs. This not only generates far less traffic across ISPs, but also improves the delivery latency. However, since the number of clusters is increased, in order to connect all the clusters of a topic together, more relay nodes (gray nodes in figure 2(b)) might be involved. Putting differently, the price of improvements in the delivery time has to be paid by the increased overhead in relay traffic. Nevertheless, the traffic in the underlying network is forwarded more efficiently. To better understand this trade-off, we run a series of experiments, which we will describe in details in Section 4.

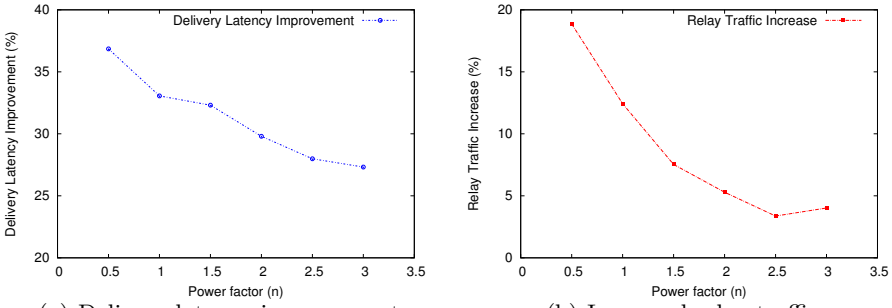
## 4 Evaluations

We verify the correctness of our solution by simulating it on Kompics [2], a message-passing component model for building distributed systems. Simulations are done with several configurations and different input data. The configuration parameters are tweaked to find out the optimal settings that delivers the most improvement as well as the best trade-offs.

**Experimental setup.** Simulations are ran with a peer-to-peer network that consists of up to 4000 nodes. To model node subscriptions, the following patterns are considered: (i) random, (ii) low-correlated, (iii) high-correlated, and (iv) Twitter subscription data. For the first three patterns, a pool of 1000 topics is generated randomly and nodes pick the same amount of topics as their interests. In random pattern, each node selects 50 topics randomly from the pool, thus, nodes subscriptions are quite diverging and they have almost no correlation. However, in low-correlated pattern, the topic pool is divided into 20 buckets, each node then picks five random buckets and selects ten topics from each bucket. For high-correlated pattern, the settings are two buckets and 25 topics from each. Differently from random patterns, nodes have more probability to share common topics and the subscriptions are more correlated. In the other words, two arbitrary nodes may share many topics since they are picked from the same bucket.

The simulation scenario features nodes joining with a uniform distribution in a deterministic way, inter-arrival time is 100 milliseconds. Even though there are many randomizations involved, the scenario can completely be reproduced again to guarantee identical set of inputs. After joining, node are given a period of time, warm-up period, to make connections, exchange views, set up relay paths as well as to gather into clusters. When the warm-up period is over, the topology is in a stable state and then 1000 events are published. Publishing is done by random nodes selecting a random topic in their interests and generating an event on that topic. Finally, when all events are published, there is a cool-down period until all events get delivered. Statistics of the run are gathered and logged for further analyzing and then the simulation terminates. Each experiment is run with and without locality-awareness and performances are put on the scale for comparison. We measure (i) the average delivery time of the published events and (ii) the average percentage of relay traffic load on each node.

**Estimating the distance.** There are many ways to model distance between nodes. In our implementation, distances between nodes are modeled based on real-world measurements using a technique, called King [12]. Kompics provides us with a full latency matrix driven from King measurements. This latency matrix provides a node with the estimated distance to each and every of its neighbors. King works by approximating the latency between two end points by measuring the latency between nearby authoritative DNS name servers, using carefully constructed recursive queries. Compared to its counterparts, King is




(a) Delivery latency improvement (b) Increased relay traffic

**Fig. 3.** The impact of $n$ in the utility function 1 on the performance

more accurate as it is based on direct, on-line measurements rather than off-line extrapolations. The King nodes are more geographically diverse than PlanetLab nodes, and the median round trip time of the King data set is 159ms [12]. Nevertheless, how the latency is measured is orthogonal to our work, and the resulting estimation, whatsoever, will act as an input to our algorithm.

### 4.1 Constructing the optimal utility function

The first set of experiments is to determine the optimal weight for the distance metric in our utility function. Power factor $n$ in formula 4.2 is tweaked with different values sampling from the range 0.5 to 3.0. The experiments are run with 500 nodes that subscribe to 50 topics randomly. In figure 3(a) delivery time improvements at different power factor are recorded and plotted on the graph. Trade-offs of the system are also plotted up in figure 3(b) Using power factor of 0.5 gives the best improvement in delivery time, roughly 37%. When power factor gets larger, distance seems to have less effect on the latency improvement. This is because a typical round-trip delay is around $100ms$, and when taking the reverse, it results in a rather small value. When the power factor goes up, the effect of round-trip delay (distance) gets small rapidly and the overlay is moving towards the case with no locality-awareness. Figure 3(b) shows a similar trend with traffic trade-offs. Power factor of 0.5 generates 18% more traffic overhead, the highest in all, while overhead drops down to only 4% at power factor 3. Choosing the power factor is then just a matter of determining whether the improvement is good enough to go over the trade-off. In the rest of our experiments we assumed we can tolerate around 12% increase in relay traffic is return for 33% improvement in delivery latency, thus, we use the power factor 1, hereafter.

### 4.2 Evaluation of different subscription patterns

In this experiment, the network size is increased from 500 nodes to 4000 nodes, to study how the system behaves when the network gets more populated. We

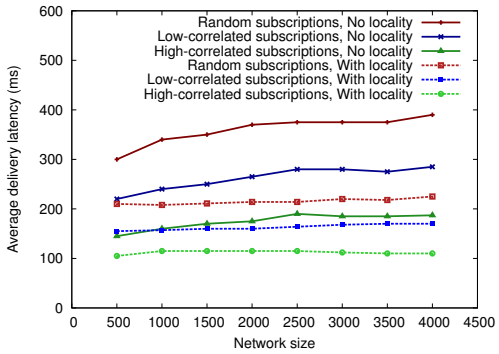



**Fig. 4.** The impact of size and subscription pattern on the delivery latency

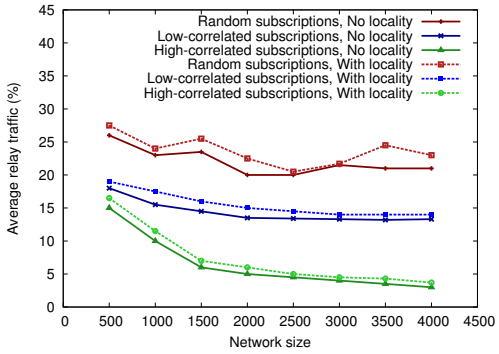



**Fig. 5.** The impact of size and subscription pattern on relay traffic

also observe the impact of several subscription patterns on the performance of the system.

Firstly, the impact of size and subscription pattern on the delivery latency with and without locality is shown in figures 4. The general trend for delivery latency, i.e., figure 4, indicates that the overlay routing is efficient, as it grows logarithmically with the network size. Without locality-awareness, random pattern has the largest delivery time, followed by the low-correlated pattern, and the high-correlated pattern has events delivered the fastest. Differences between the patterns at each point are roughly from $50ms$ to $100ms$, which is due to the more effective clustering in the presence of subscription correlation. When locality-awareness is added, the delivery latency in all subscription patterns reduces significantly. For example, latency in a network of 4000 nodes with random subscriptions drops from nearly $400ms$ to below $250ms$. Note, more correlated subscriptions still get a better performance, which means that the impact of clustering is preserved. As it is shown, locality-awareness improves the delivery latency between 25% and 40%, and the improvement increases with the network size.

We also measured the performance in terms of the relay traffic. As shown in figure 5, the generated relay traffic is more affected by the subscription pattern and the network size, whereas locality-awareness has a very little impact. Nevertheless, the relay traffic is increased by 10% on average. Note that, the relay traffic is decreased in larger networks, which is promising for scalability. It is also important to note, the increased relay traffic is in the overlay network, and not the physical network. In fact, the reduced delivery latency suggests that the routing in the physical network is performed more efficiently.

### 4.3 Evaluation on real world data

Previous experiments have proven that the proposed solution works efficiently on synthetically generated subscription patterns. However, in real world scenarios, subscription patterns are usually more complex: nodes subscribe to different number of topics and topics usually have a skewed popularity [16]. To evaluate the performance of our system with a real-world dataset, we use the Twitter subscription set [10, 16]. In Twitter any user can follow other people in order to receive their tweets (published events). Any user can also be followed by other users. That is, a user plays a dual role: a node and a topic. Twitter subscriptions have various sizes with a complex Poisson distribution with a very heavy tail. As there exist 2,484,449 records in the available data set and the simulation infrastructure cannot handle this many, we had to take a sample from this huge dataset. The sample, however, must represent the whole data set, i.e., it has to preserve the correlation between user subscriptions and maintain a comparable statistics. To make sure that we get an unbiased sample, we borrowed ideas from [17], and in a few steps collected the sampled nodes. Initially, we collected a number of Twitter nodes randomly from the whole dataset. we call this initial set the candidate set. Scanning through the whole dataset, we appended the set with all the nodes that are followed by the candidates, i.e, the subscription of the initial candidates were appended. We repeat the scanning and appending process until we collect the required number of sampled nodes. We then refine the set by removing all the nodes that either do not follow anyone in the current sample set or those that are followed by no one.

To make it even more realistic, one of the experiments is ran with churn. Churn is created by letting half of the nodes join and publish half of the events; the system then waits for some time before the rest of nodes join and the other half of events are published. The experiment is to verify if the system is robust against churn and if it is able to maintain full hit ratio. Unlike the previous simulations, this time, network size is fixed at 1000 nodes, while the number of events varies from 250 to 1000 events, with the step of 250.

Figures 6 and 7 show the performance of the system with Twitter subscription data. As expected, the performance is insensitive to the number of events. Delivery latency improvement, shown in figures 6, is very good both with or without churn (35% improvement on average). It is also interesting to see that when there is churn, delivery time is shorter than without churn. This is because nodes join in two batches and when the first batch joins, the network size

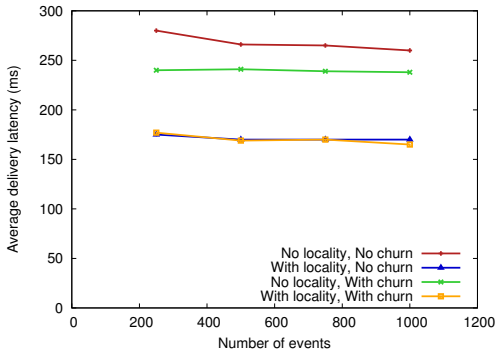



**Fig. 6.** The impact of Twitter subscription pattern and churn on delivery latency

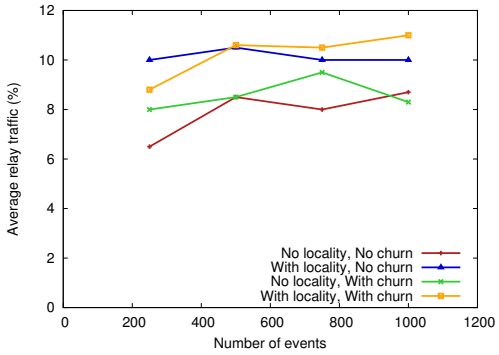



**Fig. 7.** The impact of Twitter subscription pattern and churn on relay traffic

is smaller (half size), and therefore, the first set of events are delivered much faster. Figures 7 shows the relay traffic overhead, with and without locality-awareness. Relay traffic is less than 12% in all the cases and there is no big difference between data points.

## 5 Conclusion

Optimizing and reducing the costs in publish/subscribe systems is an interesting topic among many other challenging problems. This paper solidly focuses on making a publish/subscribe system, Vitis, ISP-friendly, in order to save network bandwidth and resources. Communications and event propagation favors short and cheap links rather than those long-delay, expensive connections. This is not only cost-wise but is also an improvement in system performance, avoiding bottle necks and utilizing local resources more efficiently.

The paper proposes a solution to embed locality information into the overlay topology, that is, to make the connections between nodes to better reflect the physical network connections. It introduces a notion of distance in the neighbor

selection mechanism of Vitis and studies how we can exploit locality-awareness in-line with the subscription correlations. In other words, we investigate the trade-off between delivery latency and relay traffic with various subscription patterns, derived from real-world subscription, as well as, synthetically generated.

The simulation results showed that the system with locality-awareness outperformed the original system, in terms of delivery time, regardless of the subscription pattern. Simulations with synthetic patterns, i.e., random, low-correlated and high-correlated resulted in improvements ranging from 25% to 40%. With real-world Internet-scale data from Twitter application, the system still performs nicely with averagely 35% improvement. However, faster delivery has to be paid off by a slight increase in the amount of uninteresting traffic, i.e., relay traffic, in the overlay network. However, relay traffic over-head is negligible. Even when the subscriptions are more complex, such as the scenario with Twitter data and churn, increments of relay traffic are kept very low, at 2% to 4%. This traffic, however, is in the application layer, while the reduced delivery latency shows that the actual routing in the physical network is done more efficiently.

## References

1. Aggarwal, V. and Feldmann, A. and Scheideler, C.: Can ISPs and P2P users cooperate for improved performance?, ACM SIGCOMM Computer Communication Review, 2007.
2. Arad, C., Dowling, J., and Haridi, S.: Developing, simulating, and deploying peer-to-peer systems using the kompics component model, Proceedings of the Fourth International ICST Conference on Communication system software and middleware, 16, 2009.
3. Baldoni, R., and Virgillito, A.: Distributed event routing in publish/subscribe communication systems: a survey, DIS, Universita di Roma La Sapienza, Tech. Rep, Citeseer, 2005.
4. Baldoni, R., Beraldi, R., Quema, V., Querzoni, L., and Tucci-Piergiovanni, S.: TERA: topic-based event routing for peer-to-peer architectures, Proceedings of the international conference on Distributed event-based systems, 2007.
5. Castro, M., Druschel, P., Kermarrec, A.M., and Rowstron, A.I.T.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure, IEEE Journal on Selected Areas in communications, Citeseer, 2002.
6. Chockler, G., Melamed, R., Tock, Y., and Vitenberg, R.: Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication, Proceedings of the 2007 inaugural international conference on Distributed event-based systems, 2007.
7. Choffnes, D.R. and Bustamante, F.E.: Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems, ACM SIGCOMM Computer Communication Review, 2008.
8. Freedman, M.J., Freudenthal, E., and Mazieres, D.: Democratizing content publication with Coral, Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation, 2004.
9. Freedman, M.J., Vutukuru, M., Feamster, N., and Balakrishnan, H.: Geographic locality of IP prefixes, Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, 1313, 2005.

10. Galuba, W., Aberer, K., Chakraborty, D., Despotovic, Z., and Kellerer, W.: Outtweeting the Twitterers-Predicting Information Cascades in Microblogs, 3rd Workshop on Online Social Networks (WOSN10), 2010.
11. Girdzijauskas, S., Chockler, G., Vigfusson, Y., Tock, Y., and Melamed, R.: Magnet: practical subscription clustering for Internet-scale publish/subscribe, Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, 2010.
12. Gummadi K.P., Saroiu S. and Gribble S.D.: King: Estimating Latency between Arbitrary Internet End Hosts, SIGCOMM Internet Measurement Workshop, 2002.
13. He, Y., Zhao, Q., Zhang, J., and Wu, G.: Topology-aware multi-cluster architecture based on efficient index techniques, Network and Parallel Computing, Springer, 2005.
14. Jelasity, M., and Babaoglu, O.: T-Man: Gossip-based overlay topology management, Lecture Notes in Computer Science, Springer, 2006.
15. Kleinberg, J.: The small-world phenomenon: an algorithm perspective, Proceedings of the thirty-second annual ACM symposium on Theory of computing, 2000.
16. Krishnamurthy, B., Gill, P., and Arlitt, M.: A few chirps about twitter, Proceedings of the first workshop on Online social networks, 2008.
17. Kurant, M., Markopoulou, A., and Thiran, P.: On the bias of BFS, Arxiv preprint arXiv:1004.1729, 2010.
18. Liu, Y., Xiao, L., Liu, X., Ni, L.M., and Zhang, X.: Location awareness in unstructured peer-to-peer systems, Parallel and Distributed Systems, IEEE Transactions, 2005.
19. Papa Manzillo, M. and Ciminiera, L. and Marchetto, G. and Risso, F.: CLOSER: A Collaborative Locality-Aware Overlay SERvice, Parallel and Distributed Systems, IEEE Transactions on, 2011.
20. Pietzuch, P.R., and Bacon, J.M.: Hermes: A distributed event-based middleware architecture, 22nd International Conference on Distributed Computing Systems Workshops, IEEE Computer Society, 2002.
21. Rahimian, F., Girdzijauskas, S., Payberah, A.H., and Haridi, S.: Vitis: A gossip-based hybrid overlay for Internet-scale publish-subscribe, 2011 IEEE International Parallel and Distributed Processing Symposium, 2011.
22. Ramasubramanian, V., Peterson, R., and Sirer, E.G.: Corona: A high performance publish-subscribe system for the world wide web, Proceedings of Networked System Design and Implementation (NSDI), 2006.
23. Ratnasamy, S., Handley, M., Karp, R., and Shenker, S.: Topologically-aware overlay construction and server selection, INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, 2002.
24. Strom, R., Banavar, G., Chandra, T., Kaplan, M., Miller, K., Mukherjee, B., Sturman, D., and Ward, M.: Gryphon: An information flow based approach to message brokering, International Symposium on Software Reliability Engineering, 1998.
25. Xie, H. and Yang, Y.R. and Krishnamurthy, A. and Liu, Y.G. and Silberschatz, A.: P4p: provider portal for applications, ACM SIGCOMM Computer Communication Review, 2008.
26. Zhang, X.Y., Zhang, Q., Zhang, Z., Song, G., and Zhu, W.: A construction of locality-aware overlay network: mOverlay and its performance, Selected Areas in Communications, IEEE, 2004.
27. Zhou, S., Ganger, G.R., and Steenkiste, P.A.: Balancing locality and randomness in DHTs, School of Computer Science, Carnegie Mellon University, 2003.