TERA: Topic-based Event Routing for peer-to-peer Architectures

R. BALDONI[1] R. BERALDI[1] V. QUÉMA[2] L. QUERZONI[1] S. TUCCI-PIERGIOVANNI[1]
[1] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italia
{baldoni,beraldi,querzoni,tucci}@dis.uniroma1.it

[2] LSR-IMAG Laboratory, Sardes project
INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Saint-Ismier
vivien.quema@inrialpes.fr

Abstract

The completely decoupled interaction model offered by the publish/subscribe communication paradigm perfectly suits the interoperability needs of todays large-scale, dynamic, peer-to-peer applications. The unmanaged environments, where these applications are expected to work, pose a series of problems (potentially wide number of partipants, low-reliability of nodes, absence of a centralized authority, etc.) that severely limit the scalability of existing approaches which were originally thought for supporting distributed applications built on the top of static and managed environments. In this paper we propose an architecture for implementing the topic-based publish/subscribe paradigm in large scale peer-to-peer systems. The architecture is based on clustering peers subscribed to the same topic. The major novelty of this architecture lies in the mechanism employed to bring events from the publisher to the cluster (namely outer-cluster routing). The evaluation shows that this mechanism for outer-cluster routing has a probability to bring events to the destination cluster very close to 1 while keeping small the involved number of out-of-cluster peers. Finally, the overall architecture is shown to be scalable along several fundamental dimensions like number of participants, subscriptions, and to exhibit a fair load distribution (load distribution closely follows the distribution of subscriptions on nodes).

Publish-Subscribe, Gossip-based algorithms

1 Introduction

Publish/subscribe is a communication paradigm of growing popularity for information dissemination in large scale distributed systems. Participants to the communication can act both as producers (*publishers*) and consumers (*subscribers*) of information. Publishers inject information in the system in the form of *events*, while subscribers declare their interest in receiving some of the published events, issuing *subscriptions*. Subscriptions express conditions on the content of events (*content-based* model) or just on a category they should belong to (*topic-based* model). The paradigm states that once an event is published, for each subscription whose conditions are satisfied by the event (we say that the event matches the subscription), the corresponding subscriber must be notified. The basic building block of systems implementing the publish/subscribe paradigm is a distributed *event dissemination* mechanism able to bring any published event from the publisher to the set of matched subscribers, while completely decoupling their interaction[10].

While publish/subscribe for managed systems has been widely studied and various solutions exist in the literature [18, 6, 3], publish/subscribe for unmanaged systems, like peer-to-peer systems, is today an active field of research [8, 2, 24].

In peer-to-peer systems, the event dissemination mechanism is usually implemented on top of an *overlay network* connecting all user nodes (both publishers and subscribers). Overlay networks [1, 23] are specifically designed to support information diffusion characterized by a high-level of reliability in large scale and unreliable environments.

Event dissemination in such systems can be trivially implemented flooding each event in the overlay and then filtering out events that do not match local subscriptions at each single node. However, the semantics of the publish/subscribe paradigm can be leveraged to confine the dissemination of each event only in the set of matched subscribers, without affecting the whole network (*traffic confinement*) [20, 2, 24]. This is particularly important when the event matches very specific interests which have a small number of subscribers with respect to the total number of nodes.

Even though traffic confinement brings obvious advantages (as it potentially saves traffic in the network), its implementation poses non-trivial problems. Basically, traffic confinement should be realized with the following objec-

tives:

1. *Interest Clustering.* Subscribers should be arranged trying to cluster those that have common interests. In this way, once the event reaches one member of the cluster, its dissemination can be limited to the cluster itself. Ideally, each cluster should contain *all* subscribers interested in a given event in order to avoid a loss of reliability (i.e. the capacity of the system to notify each event to the set of matched subscribers).

2. *Inner-Cluster Dissemination.* Once the event reaches one member of the interested cluster, the dissemination inside the cluster can follow a simple flooding scheme or more sophisticated routing techniques can be used to save more traffic [24].

3. *Outer-Cluster Routing.* An event can be published on any node, therefore it must be routed from it, to one node belonging to the target cluster. Note that, traffic confinement is fully satisfied when non-interested subscribers do not receive the event. Then, the goal of outer-clustering routing is to reach the target cluster while involving a number of non-interested subscribers as small as possible.

To the best of our knowledge, current solutions [20, 2, 24], mainly deal with the first two points, i.e. the problem of clustering all subscribers with common interests together and the problem of how to efficiently disseminate the event inside the cluster. Less attention has been devoted to the routing from the publisher to the target cluster. In this paper we propose a novel architecture, namely TERA (Topic-based Event Routing for p2p Architectures), for the implementation of topic-based publish/subscribe systems in large-scale, unmanaged peer-to-peer (p2p) environments and we specifically evaluate the impact of the TERA outer-cluster routing in getting traffic confinement.

More specifically, TERA realizes interest clustering by instantiating a dedicated overlay network for each topic (*topic overlay*) and including in it all subscribers subscribed to that topic. Then, in order to realize outer-cluster routing, each node is equipped with an access point lookup table (of limited size) containing a set of pairs $< topic, subscriber >$, in which the subscriber represents an access point for the cluster it belongs to. Thanks to the mechanisms employed by TERA to update these tables, topics are uniformly represented and, given a specific topic, each subscriber of that topic has the same probability to appear as an access point. These properties hold even with uneven interest distributions among topics, i.e. some topics more popular than others. Outer-cluster routing of an event follows a random walk in order to find an access point for the target topic. We show how, thanks to a uniform distribution of topics and subscribers in lookup tables, routing of each event from the source node to the target overlay has a probability of successes close to 1 with random walks involving a small number of outer cluster nodes and a reasonable access point lookup table size. As regards inner-cluster dissemination, we evaluate the performance of a simple flooding inside the topic overlay network we use. However, more sophisticated mechanisms could be embedded in TERA as the ones proposed in [20, 24] to reduce traffic of the inner-cluster dissemination.

Our evaluation also shows that mechanisms employed in TERA have a cost of dissemination per-event that scales with respect to the number of nodes constituting the system, the number of subscriptions/topics issued, and the event publication rate. Finally, TERA is shown to fairly distribute the system load according to the number of subscription currently issued by each participant.

The paper is organized as follows: Section 2 gives an overview on TERA's infrastructure, while Section 3 details its internal architecture. Section 4 evaluates, with both analytical and experimental methods, TERA's characteristics with respect to the event dissemination mechanism by traffic confinement and the overall system's scalability. Section 5 offers an overview on related works, and, finally, section 6 concludes the paper.

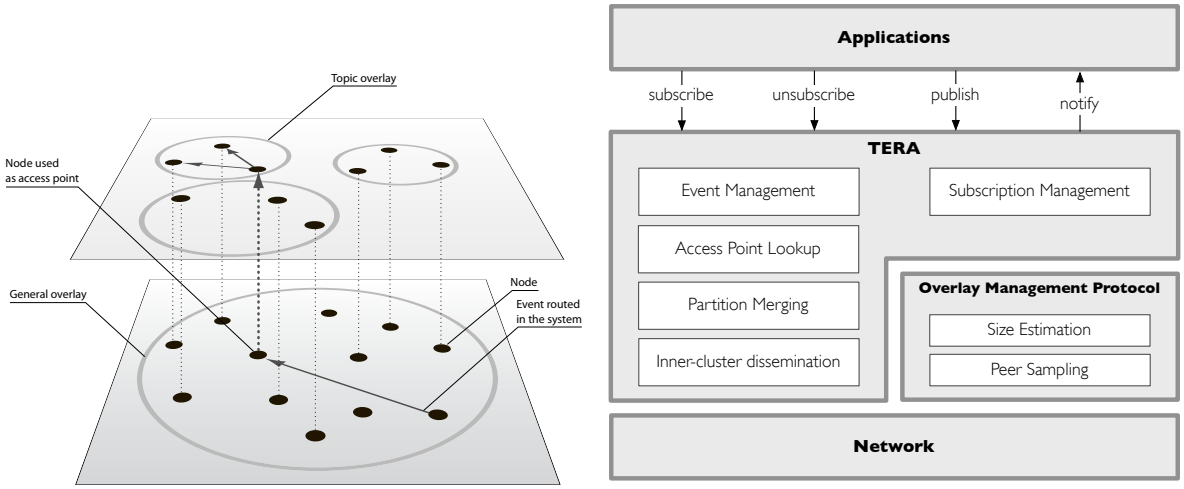
2 An Overview of TERA

TERA is a topic-based publish/subscribe system designed to offer an event dissemination service for very large scale peer-to-peer systems. Each published event is "tagged" with a *topic* and is delivered to all the subscribers that expressed their interest in the corresponding topic by issuing a *subscription* for it. The set of available topics is not fixed, nor predefined: applications using TERA can dynamically create or delete them.

2.1 Architecture

Nodes participating to TERA are organized in a two-layers infrastructure (ref. Figure 1(a)). At the lower layer, a *global overlay* network connects all nodes, while at the upper layer various *topic overlay* networks connect subsets of all the nodes; each topic overlay contains nodes subscribed to the same topic. All these overlay networks are separated and are maintained through an overlay management protocol(detailed in the next Section).

Subscription management and event dissemination in TERA are based on two simple ideas: nodes that want to subscribe to a topic t are required to join the corresponding topic overlay that connects at the upper layer all nodes subscribed to t (interest clustering). When an event e, tagged with topic t, is published by a node (not necessarily subscribed to t) it is first routed at the lower layer to an *access*



(a) System overview

(b) Node architecture

Figure 1. The TERA publish/subscribe system.

point for topic t, i.e. one of the nodes subscribed to t (outer-cluster routing); this node then diffuses e at the upper layer in the topic overlay associated with topic t, in order to deliver it to all the other subscribers of t (inner-cluster dissemination). In this way the traffic generated for event dissemination remains confined within the target topic overlay.

Figure 1(b) depicts a high level overview of a node's internal architecture. TERA is a software layer that offers to applications running on the same node an interface to *subscribe*/*unsubscribe* topics, *publish* information and be *notified* of incoming events.

TERA's internal components (*Event Management*, *Subscription Management*, *Access Point Lookup*, *Partition Merging*, *Inner-Cluster Dissemination*), detailed in Section 3, working on distinct nodes interact through an existing network infrastructure, that is usually represented by the Internet, and leverages services provided by an overlay management protocol (*Size estimation*, *Peer sampling*) detailed in the following Section.

2.2 The Overlay Management Protocol

An overlay network is a logical network built on top of a physical one (usually the Internet), by connecting a set of nodes through some links. A distributed algorithm running on nodes, known as the Overlay Maintenance Protocol (OMP), takes care of managing these logical links. Each node usually maintains a limited set of links (called *view*) to other nodes in the system. The construction and maintenance of the views must be such that the graph obtained by interpreting nodes as vertices and links as arcs is connected. Indeed, this is a necessary condition to enable communication from each node to all the others.

TERA requires the overlay management protocol to implement (1) a *peer sampling service*, able to provide *uniform* node samples, and (2) a *size estimation service*. Numerous protocols exist today that can be employed to maintain a peer-to-peer overlay network. However, protocols best suited to provide uniform samples of nodes are those based on the *view exchange* technique [1, 23]. These protocols periodically update views maintained at each node by swapping random view entries between randomly chosen nodes. The view exchange technique lets the protocol build and maintain overlay topologies that closely resemble random graphs. Consequently, these overlays exhibit high connectivity and low diameter, which make them resilient to massive node failures, and adequate topologies for implementing efficient broadcast primitives. Concerning the size estimation service, many protocols working on view exchange-based overlay management protocols have been proposed [13, 15, 16, 17, 19]. Note that TERA also requires the OMP to expose a primitive that can be used to force a view exchange with some other node.

3 Implementation details

TERA's internal structure is made of five main components (Figure 2): *Event Management*, *Subscription Management*, *Access Point Lookup*, *Partition Merging* and *Inner-Cluster Dissemination*. In this section we describe the details of their implementation with the exception of the Inner-Cluster Dissemination component. In this paper we will adopt a simple flooding for this component to

evaluate TERA (Section 4). Note that, many existing solutions [9, 11, 4, 24] can be adopted accordingly to the specific application requirements. Appendix 6 reports a pseudo-code description of each detailed component.

Subscription Management The Subscription Management component handles new subscriptions and unsubscriptions, updating the *Subscription Table* — a data structure containing a list of couples $< t, i >$, where t is a topic the node is subscribed to, and i is the corresponding topic overlay identifier[1] — and instructing the overlay management protocol to join/leave topic overlay networks associated to subscribed/unsubscribed topics.

A new subscription for a topic, causes the Subscription Management component (i) to add and entry for the topic to the *Subscription Table*, and (ii) to ask the overlay management protocol to join the corresponding topic overlay. To fulfill the latter point, the overlay management protocol needs at least one identifier of a node already part of the topic overlay. This identifier can be obtained through a lookup executed on the Access Point Lookup component. Note that, if no identifier is returned, the node instantiates a new topic overlay[2].

Unsubscriptions are handled by removing an entry in the Subscription Table and asking the overlay management protocol to leave the corresponding topic overlay.

The Subscription Management component is also responsible for periodically advertising the list of currently subscribed topics to a set of nodes randomly chosen in the general overlay. For each topic, the advertised list contains the corresponding topic overlay identifier and an estimation of the topic popularity. The topic popularity is estimated by the size estimation service provided by the overlay management protocol running in the corresponding topic overlay.

The list is advertised to D nodes whose identifiers are obtained from the peer sampling service provided by the overlay management protocol; this guarantees that the list will be advertised to a set of nodes randomly chosen from the whole system population. The received advertisements are used to update data structures in the Access Point Lookup and Partition Merging components (more details will be given in their corresponding sections).

Event Management The Event Management component implements the main logic required for publishing and diffusing events, as well as for notifying subscribers. An event dissemination starts as soon as an application *publishes* some data in a topic. It is done in two steps: the event is first routed to a node subscribed to the topic (this node acts as an access point for it); then, the access point diffuses the event in the overlay associated to the topic. The first step is realized through a lookup executed on the Access Point Lookup component: if the lookup returns an empty list of node identifiers, the node discards the event.

When a node subscribed to the topic receives an event for which it must act as an access point, it uses the broadcast primitive provided by the Inner-Cluster Dissemination service to forward the event to all nodes belonging to the corresponding topic overlay. When a node subscribed to the topic receives a broadcasted event, it notifies the application.

Access Points Lookup The Access Point Lookup component plays a central role in TERA's architecture as it is used by both the Event Management and Subscription Management components to obtain lists of access points identifiers for specific topics. Its functioning is based on a local data structure, called *Access Point Table* (APT), and a distributed search algorithm based on random walks.

Each APT is a cache, containing a limited number of entries, each with the form $< t, n >$, where t is a topic and n the identifier of a node that can act as an access point for t. APTs are continuously updated following a simple strategy: each time a node receives a subscription advertisement for topic t from a node n, it substitutes the access point identifier for t if an entry $< t, n' >$ exists in the APT, otherwise it adds a new entry $< t, n >$ with probability $1/P_t$, where P_t is the popularity of topic t estimated by n and attached to the subscription advertisement. When an APT exceeds a predefined size, randomly chosen entries are removed.

As a consequence of this update strategy, APTs have the following properties:

1. APT entries tend to contain non-stale access points,
2. inactive topics (i.e. topics that are no longer subscribed by any node) tend to disappear from APTs,
3. each access point is a uniform random sample of the population of nodes subscribed to that topic,
4. the content of each APT is a uniform random sample of the set of active topics (i.e. topics subscribed by at least one node),
5. the size of each APT is limited.

The first property is a consequence of the way new entries are added to APTs; suppose, in fact, that there is only one topic t in the system subscribed by two nodes, n_a and n_b; suppose, moreover, that, at certain point of time, n_b *unsubscribes* t. Starting from that moment, only n_a will advertise t, therefore nodes containing an entry $< t, n_b >$ will eventually substitute it with entry $< t, n_a >$, as the

[1] The identifier is generated by the node that instantiated the topic overlay.

[2] Different overlays might thus be created for the same topic, which is handled by the merging mechanism described later in the paper.

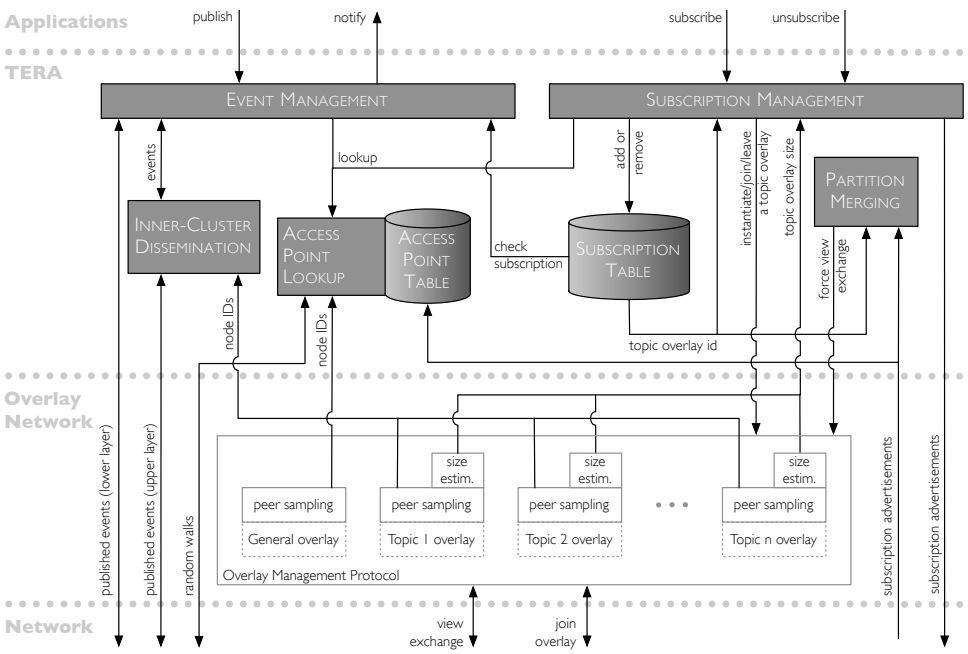


Figure 2. A detailed view of the architecture of TERA.

uniformity of node samples provided by the peer sampling service guarantees that n_a will eventually advertise t to all the system population. The second property comes from the fact that inactive topics are no longer advertised. They are, thus, eventually replaced by active topics in APTs (assuming that the set of active topics is larger than the maximum APT size). The third property is a consequence of the fact that subscription advertisements are sent to nodes returned by the peer sampling service that provides uniform random samples, and that each node advertises its subscriptions with the same period. The fourth property is also a consequence of this fact, and of the fact that the APT update mechanism uses estimations of topic popularities[3] to normalize APT updates. Let us remark that in the proposed update mechanism, all subscribers periodically advertise their subscriptions, and nodes drop these advertisements in inverse proportion to the topic popularity. One might think that this strategy induces the exchange of unnecessary messages. Nevertheless, trying to reduce the number of sent messages by having subscribers advertise their subscriptions in inverse proportion to the topic popularity would increase the probability of APTs containing references to stale access points[4].

[3]A highly popular topic (i.e. a topic subscribed by many nodes) will be advertised more often than a less popular one.

[4]This comes from the fact that advertisements are also used to update access point for topics that are yet contained in APTs, regardless of their popularities.

Note that the APT update algorithm described above has been designed with the goal of maintaining fresh APTs content. Other strategies can be employed as well. For instance, the algorithm could use some form of knowledge about the stability (i.e. uptime) of a node, or its distance from the updated node to choose whether to keep or replace an access point.

Given the APTs limited size, nodes may only have a limited knowledge of the set of active topics. To solve this problem, the Access Point Lookup component *searches* for access points in APTs stored at other nodes. This search is implemented as a random walk in the global overlay. The rationale behind this search mechanism is that, given the uniform randomness of APTs' content and of node identifiers returned by the peer sampling service, it is possible to set the lifetime of the walks and the APT table size such that, given a topic, with a certain probability either (i) an access point for it will be found, or (ii) it will safely be considered as inactive.

Note that the reliability of event dissemination in TERA strongly depends on the behaviour Access Point Lookup component. Section 4 reports a detailed evaluation of this aspect.

Partition Merging The Partition Merging component implements mechanisms used to maintain topic overlay networks. It is motivated by the fact that if two nodes concur-

rently subscribe to a same topic for which no access point exists, the system may end up with two disconnected topic overlay networks for the topic. It is thus necessary to define a mechanism to detect the presence of partitioned topic overlays and merge them.

Partitioning detection is performed each time a subscription advertisement, sent by a node n, is received by a node n'. n' checks for each advertised topic it is also subscribed to, if the local topic overlay identifier corresponds to the one contained in n's advertisement. A mismatch between the two identifiers shows that two distinct partitions exist for the same topic overlay. In order to merge these two partitions, the merging mechanism on n' forces the overlay management protocol to execute a view exchange for the partitioned topic overlay with node n. The aim of this view exchange is to mix nodes belonging to partitioned overlays in the views of both n and n'. From this time on the topic overlay is no more partitioned (therefore, an event can be successfully broadcasted reaching all the subscriber) even if two different overlay identifiers can still exist in the system. Resynchronizing different overlay identifiers is needed anyway to prevent further useless forced view exchanges. The Partition Merging component must thus resynchronize identifiers of nodes belonging to the same overlay[5].

Note that the partition merging mechanism is fundamental to limit the influence of our traffic confinement strategy on global event dissemination reliability. Section 4 reports a detailed evaluation of this aspect.

4 Evaluation

4.1 Experimental setup

We implemented a prototype of TERA using Peersim [12], an open source Java simulation framework for peer-to-peer protocols. Peersim allowed us to test TERA on large simulated networks, modeling with sufficient precision the environment where TERA is supposed to work. The overlay management protocol employed in our prototype is Cyclon [23], which provides every node with a view representing a uniform random sample of the system. Cyclon is a *cycle-based* protocol: at each cycle a node executes a view exchange phase. Phases among nodes are supposed to have the same duration, but are not synchronized. A peer sampling service is built upon Cyclon just picking up random node identifiers from the view. These samples are then used to feed a size estimation service built through the algorithm introduced in [16]. We assume cycles as the reference time unit in the rest of this section. During a cycle, a process can handle the messages that were sent to it in the previous cycles.

[5]This can be simply accomplished by exchanging identifiers during each view exchange, and by deterministically choosing one of them.

Concerning the workload model, there is currently no publicly available data traces of real pub/sub applications. Consequently, we tested our algorithm on various synthetic scenarios, following the approach used in other studies [21, 5, 7]. In particular we characterized the set of events and subscription used in our tests as follows.

The set of subscriptions is characterized by the following four properties: *number of topics*, *number of subscriptions*, *topic popularity distribution* (i.e. how subscriptions are distributed on topics), and *subscription distribution on nodes*. Subscriptions are distributed on nodes following a uniform distribution[6]. Concerning topic popularity, we consider two distributions:

- Uniform: each subscription can be issued with the same probability on any topic.
- Power-law (also called Zipf): topic popularity distribution follows a zipf curve, leading to systems where few topics are highly popular, while a lot of topics are not popular.

The set of events is characterized by the following four properties: *number of topics*, *number of events*, *event distribution on topics*, and *event distribution on nodes*. In our tests, we consider uniform distributions for event distribution on both topics and nodes.

4.2 Outer-cluster routing assessment

In this section we show, through an evaluation of both the Access Point Lookup and the Partition Merging components, how outer-cluster routing realized through TERA performs.

4.2.1 Topic distribution in APTs

We start by presenting an experiment showing that the method used in TERA to update APTs content ensures a uniform distribution of topics in every APT. This is a fundamental property for APTs as it allows TERA to use their content as a uniform random sample of the active topic population and build on it the access point lookup mechanism. We ran tests over a system with 10^4 nodes, each advertising its subscriptions every 5 cycles to 5 neighbors out of 20 (the overlay management protocol view size). APT size was limited to 10 entries. We issued 5000 subscriptions distributed in various ways on 1000 distinct topics, and we measured, for each topic, the number of APTs containing an entry for it. The expected outcome of these tests is to find a constant value for such measure, regardless of the initial topic popularity distribution.

[6]Subscriptions regionalism is not considered as it would be "destroyed" by the continuous exchange of views in the general overlay.

Figure 3(a) shows the results for an initial uniform distribution of topic popularity. The X axis represents the topic population (each topic is mapped to a number). Each black dot represents the number of times a specific topic appears in APTs, while the grey dot represents its popularity. The plot shows that each topic is present, on average, in the same number of APTs, with a very small error that is randomly distributed around the mean. This confirms that the topic distribution in APTs can be considered uniform.

Figures 3(b) and 3(c) show the results for an initial zipf distribution of topic popularity. The two graphs report the results for differently skewed popularity distributions (distribution parameter $a = 0.7$ and $a = 2.0$). As these graphs show, TERA is always able to balance APT updates, and delivers an almost uniform distribution. Even in an extreme case ($a = 2.0$), the APT update mechanism is able to balance the updates coming from the small number of active topics (in this scenario only 79 topics share the whole 5000 subscriptions), maintaining their presence in APTs around the same average value with a small standard deviation (always below 5%). In the next evaluations, we only report results for zipf popularity distribution with $a = 0.7$, as results for other values of a did not exhibit significant differences.

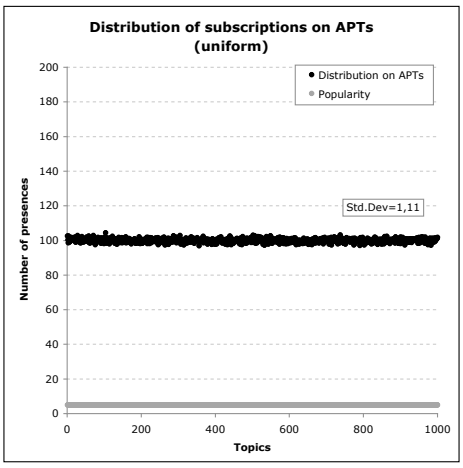
4.2.2 Access Point Lookup

In this section, we evaluate the probability for the access point lookup mechanism to successfully return a node identifier for a lookup operation (in the case such node exists). We denote by K the lifetime of the random walk (the maximum number of visited nodes), by $|APT|$ the size of APT tables, and by $|T|$ the number of topics[7]. The probability p to find an access point for a specific topic in an APT is $p = \frac{|APT|}{|T|}$. Assuming that every APT contains the maximum allowed number of entries, the probability that an access point cannot be found within K steps is $Pr\{fail\} = (1-p)^K$. Thus, the probability to find the access point visiting at most K nodes is $Pr\{success\} = 1-(1-p)^K = 1-\left(1-\frac{|APT|}{|T|}\right)^K$. Therefore, to ensure with probability P that an access point for a given topic will be found, it is necessary that sizes K or $|APT|$ be such that:

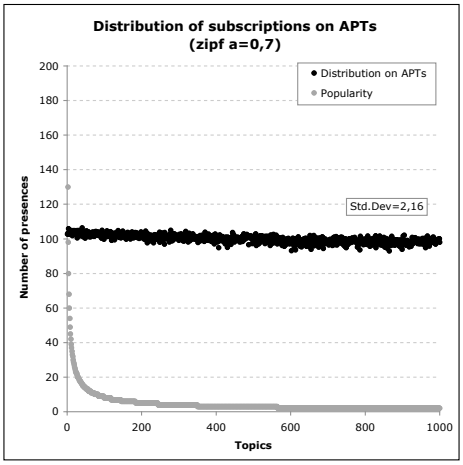
$$K = \frac{ln(1-P)}{ln\left(1-\frac{|APT|}{|T|}\right)} \quad \text{or} \quad |APT| = |T|\left(1-\sqrt[K]{1-P}\right)$$

Note that, given K and P, $|APT|$ linearly depends on $|T|$. In order to reduce APT size, it would be necessary to increase random walks length (i.e. using a large value for

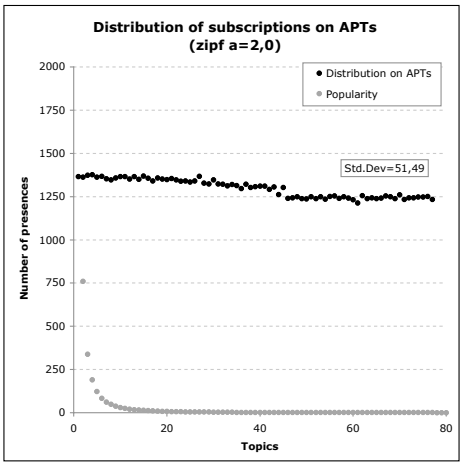
[7]Thanks to the fact that APTs can be considered as uniform random samples of the set of active topics, each node can estimate at runtime the value of $|T|$ [16].



(a)



(b)



(c)

Figure 3. The plot shows how topics are distributed among APTs (black dots) when the topic popularity distribution (grey dots) is (a) uniform and (b-c) skewed (zipf with parameter a).

K) negatively affecting the time it takes to find an access point. To mitigate this problem, it is advisable to launch r multiple concurrent random walks, each having a lifetime $\lceil \frac{K}{r} \rceil$. Indeed, the fact that topics are uniformly distributed among APTs guarantees that launching multiple concurrent random walks does not impact the lookup success rate. In this way, access point lookup responsiveness is improved at the cost of a slightly larger overhead due to the independency of each random walk lifetime.

We ran experiments to check that TERA's behavior is close to the one predicted by the analytical study. Tests were run on a system with 1000 nodes, each having Cyclon views holding 20 nodes. At the beginning, 5000 subscriptions were issued uniformly distributed on 1000 distinct topics. Lookups were started after 1000 cycles. Each lookup was conducted starting four concurrent random walks ($r = 4$).

Figure 4(a) shows how the access point lookup success ratio changes when varying the lifetime of each random walk (K) for different values of $|APT|$. For each line, we plotted both simulation results (solid line) and values calculated using the analytical study (dashed line). The plot confirms that TERA's lookup mechanism is able to probabilistically guarantee that an access point for an active topic will be found with probability P. Note that this plot also shows that the actual memory size required by APTs is limited. Indeed, consider the biggest APT size plotted on the graph: 400 entries. Assuming that each entry in an APT is a string containing 256 characters, the memory size occupied by an APT containing 400 entries is about 104kB.

4.2.3 Partition Merging

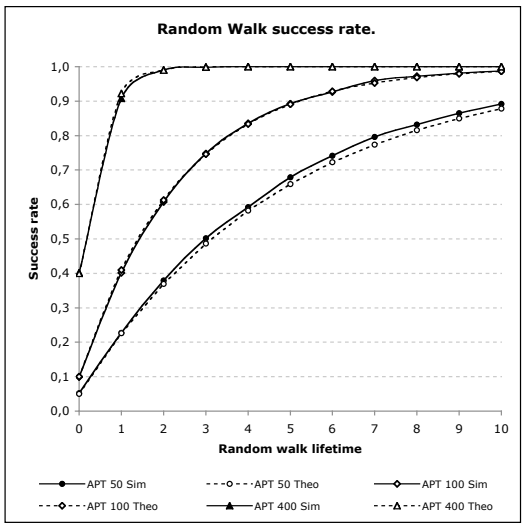
In this section, we analyze the probability for the partition merging mechanism to detect a very small overlay partition, and the time it takes for this to happen. Suppose that there is a topic represented by an overlay network partitioned in two clusters containing $|G|$ and 1 nodes, respectively[8]. Let us call n this single node. The probability p to detect the partition in a cycle can be expressed as $p = 1 - (p_a \cdot p_b)$, where p_a is the probability that none of the nodes in G advertise its subscriptions to n, and p_b is the probability that n does not advertise its subscriptions to any of the nodes in G.

Probability p_a can be expressed as

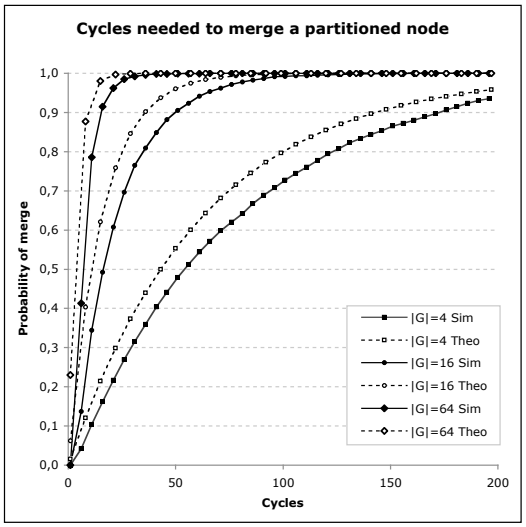
$$p_a = (1 - Pr\{a\ node\ advertises\ to\ n\})^{|G|}$$

Every node in G advertises its subscription to n only if n is contained in its view for the general overlay, and if n is one of the D nodes selected for the advertisement. Let

[8]Note that the case where a partition is constituted by a single node is the most difficult to solve as the probability for nodes belonging to distinct partitions to meet is the lowest possible one.

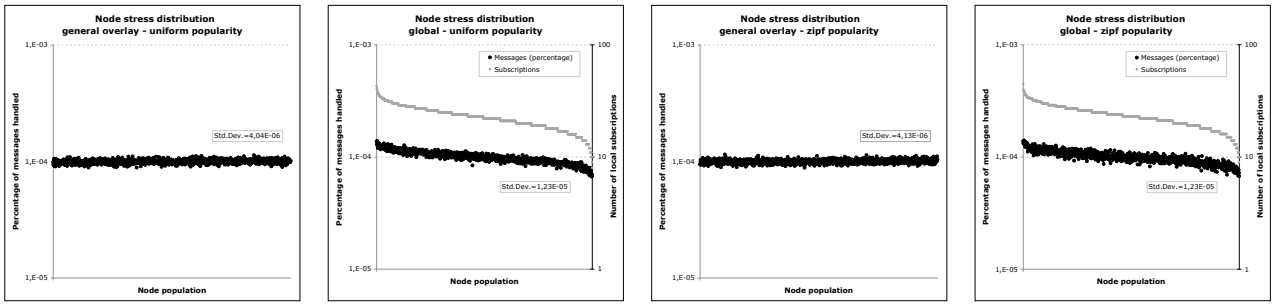


(a)



(b)

Figure 4. (a) The plot shows how the success rate for access point lookups changes when varying the maximum APT size and the random walk lifetime. Solid lines represent results from the simulator, while dashed lines plot values from the formula. (b) The plot shows how the probability to detect a topic overlay partition increases with time (cycles). Solid lines represent results from the simulator, while dashed lines plot values from the formula. The tests were run varying the number $|G|$ of nodes subscribed to the topic.



(a) (b)

Figure 5. The plots show how the load generated by TERA is distributed among nodes when the distribution of topic popularity is either uniform (a) or zipf (b). For both popularities, the figure shows in the left graph the load distribution in the general overlay and, in the right graph, the global load distribution (black points), together with the subscription distribution on nodes (grey points).

us suppose, for the sake of simplicity, that D is equal to the view size. In this case $Pr\{a\ node\ advertises\ to\ n\} = \frac{|View|}{(N-1)}$, where N is the total number of nodes. Consequently, $p_a = \left(1 - \frac{|View|}{N-1}\right)^{|G|}$.

Probability p_b is equal to the ratio between the number of views a node n can have that do not contain nodes subscribed to t (i.e. nodes in G), and all the possible views. Therefore, $p_b = \frac{C(N-1-|G|,|View|)}{C(N-1,|View|)}$, where $C(n,k)$ is the number of k-combinations of a set with n elements. Note that, correctness of p_b formula is guaranteed by the fact that, thanks to the uniform randomness of the peer sampling service provided by the OMP, every view has the same probability to appear on a node.

It follows that the overall probability is

$$p = 1 - \left(\left(1 - \frac{|View|}{N-1}\right)^{|G|} \cdot \frac{C(N-1-|G|,|View|)}{C(N-1,|View|)}\right)$$

From the expression of p, we can derive the probability that a merger will happen in H cycles:

$$Pr\{merger\ within\ H\ cycles\} = 1 - (1-p)^H =$$
$$1 - \left(\left(1 - \frac{|View|}{N-1}\right)^{|G|} \cdot \frac{C(N-1-|G|,|View|)}{C(N-1,|View|)}\right)^H$$

This formula shows that the merger probability tends to 1 as cycles pass by, regardless of the topic popularity. Moreover, not surprisingly, the amount of cycles needed to observe a merger is conversely proportional to the popularity $|G|$ of the topic.

To confirm this result, we tested the partition merging mechanism in networks made up of 1000 nodes, with a single topic. In these tests, G subscriptions for the topic are initially issued on various nodes, that quickly form a topic overlay. Then, a new subscription is issued on a node not yet subscribed, and a failed lookup is simulated, in order to create a second topic overlay. We observed the time it took to the partition merging mechanism to detect the partition. Note that considering one single topic does not impact this time. Indeed, this assumption has only for consequence that this topic will be present in every APT. Nevertheless, APTs are not used in the merging mechanism (received subscriptions are only checked against the node subscriptions and not against entries in its APT).

Figure 4(b) reports the results for tests conducted varying $|G|$. We plot both simulation results (solid line) and expected values calculated with the formula (dashed line). The results confirm the analytical study[9]: as cycles pass by every topic partition is detected. Moreover, it is harder to detect partitions for less popular topics (i.e. lower values for $|G|$), with respect to highly popular topics.

4.3 Scalability assessment

4.3.1 Node stress distribution

A very important aspect that must be taken into account is node stress distribution, i.e. the fraction of the whole overhead generated by TERA experienced by each single node. In particular, the burden imposed on nodes should be fairly

[9]The differences shown in the figure, between theoretical and simulation values are a consequence of simplifying assumptions done in the analytical study.

subdivided among all participants to avoid the appearance of *hot spots*.

To test node stress under various possible workloads, we ran tests with both uniform and zipf topic popularity distributions. Tests were run on a system with 10^4 nodes. We issued $2 \cdot 10^5$ subscriptions distributed on 1000 distinct topics, and then diffused one event by cycle during the whole simulation duration. Events were uniformly distributed over topics. In order to evaluate how the load is distributed among nodes, we measured the fraction of messages handled by each node during the tests, separating figures for messages exchanged in the general overlay and for those exchanged in topic overlays.

Figures 5(a) show the results for a test with uniform topic popularity, while figures 5(b) show the same results for an initial zipf distribution with parameter $a = 0.7$. Pictures on the left show how load is distributed in the general overlay. As shown by the graphs, TERA is able to uniformly distribute load among nodes, avoiding the appearance of hot spots. This result is obtained regardless of the distribution of topic popularities. Pictures on the right show the global load experienced by nodes; in these graphs, nodes on the X axis are ordered in decreasing local subscriptions count (i.e. points on the left refer to nodes subscribed to more topics), in order to show how the global load is affected by the number of subscriptions maintained at each node. The number of subscriptions per node is also plotted with grey dots. The graphs show how load distribution closely follows the distribution of subscription on nodes, actually implementing the pragmatic rule *"the more you ask, the more you pay"*, then fairly distributing the load among participants.

4.3.2 Message cost per notification

The traffic confinement strategy implemented by TERA induces some overhead. In order to assess the global impact of this overhead, we evaluated the average cost incurred by TERA to notify a *single* event to a subscriber, namely the total number of generated messages divided by the number of actual notifications[10]. This cost includes both messages generated to disseminate the event, and messages generated for TERA's maintenance. To offer a reference figure, we also evaluated the cost incurred by a simple *event flooding*-based approach [11] in the same settings.

Figure 6(a) reports the results when the total number of subscriptions varies between 10^2 and 10^6. The number of topics is fixed and equal to 100. The network considered in this test was constituted by 10^4 nodes, while the event publication rate was maintained constant at 1 event per topic in each cycle. For the evaluation to be meaningful, we required each topic to be subscribed by at least one subscriber; therefore, each curve is limited on its left end by the number of available topics. Moreover, we required each node to subscribe each topic at most once; therefore, each curve is limited on its right end by the number of nodes in the system times the number of available topics (e.g. the curves start from 100 subscriptions and end at $10^2 \cdot 10^4 = 10^6$ subscriptions).

The reference cost expressed by the simple event flooding algorithm decreases as the number of subscriptions increases. This behaviour is justified by the fact that the total cost incurred by the algorithm for each event dissemination is constant, regardless of the number of subscriptions (as it only depends on the popularity of each topic). Consequently, increasing the number of subscriptions has a positive impact on the algorithm efficiency: each event broadcast in the overlay network will generate a higher number of notifications.

TERA's behaviour is more complicated, as various factors have an impact on its global cost. This global cost is the sum of two contributions: a constant amount and a variable one. The former does not depend on the total number of subscriptions: it corresponds (i) to the cost induced by the overlay management protocol's view exchange mechanism for the general overlay, and (ii) to the cost induced by the access point lookup mechanism. The latter is proportional to the total number of subscriptions per topic issued in the system, and includes the cost (i) of subscription advertisements, (ii) of the view exchange mechanism for topic overlays, and (iii) of the broadcast service used to implement inner-cluster dissemination.

When the number of subscriptions per topic is close to one (on the left end of the curve), the constant part of the total cost is dominant. Therefore, the average notification cost decreases as for the simple event flooding algorithm. On the contrary, when the number of subscriptions per topic increases, the variable part of the cost becomes dominant. Consequently, the average notification cost quickly reaches a lower bound that is defined by the out degree used in the broadcast service (in our experiments we considered an out degree equal to the view size, i.e. 20). Note that, the employment of smarter broadcast mechanisms could, in principle, further reduce the asymptotic notification cost.

As expected, TERA and the event flooding protocol have a comparable behavior when the number of subscribers per topic is close to the total number of nodes. Indeed, in such case, each node is subscribed to every topic; therefore, it is interested in every event published in the system, making differences between the two approaches negligible.

Figure 6(b) reports the same test, ran varying the amount of topics and maintaining a fixed number of subscriptions

[10] In our tests this number always corresponded to the expected number of notifications, i.e. no notifications were missed.

[11] Each event is broadcast in an overlay network containing all participants. The overlay is built and maintained through the same overlay management protocol employed by TERA (Cyclon). Also the mechanism is the same considered for TERA.

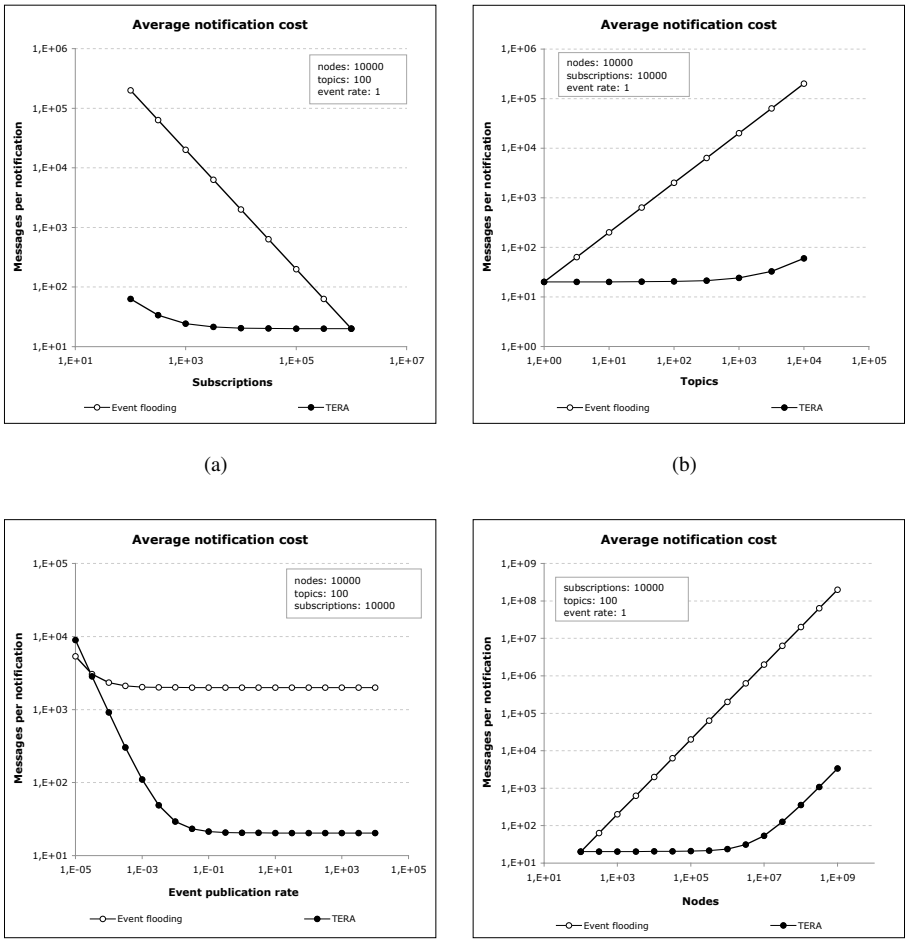


Figure 6. The plots show the average number of messages needed by TERA to notify an event when the number of subscriptions (a), of topics (b), the event publication rate (c) and the total number of nodes in the system (d) varies. For each figure, results from a simple event flooding algorithm are reported for comparison.

(10^4). In this case, the algorithm's behavior is dual with respect to the previous figure: a higher number of topics increases the load for simple event-flooding (because it causes each generated event to be matched by a smaller number of subscribers), while TERA's performance remain almost unchanged.

Figure 6(c) reports the same test when the number of subscriptions and topics is kept constant (100 topics and 10^4 subscriptions), while the event publication rate per topic varies between 10^{-5} and 10^5. The plots show a clear tradeoff: when the event publication rate is very low, the higher overhead caused by TERA is not compensated by the advantages induced by traffic confinement. Nevertheless, these advantages comes into play as soon as the event publication rate raises. This result confirms TERA's ability to better scale in high load settings.

Finally, figure 6(d) reports how TERA scales with respect to the number of nodes in the system. This test has been run in a scenario where 10^4 subscriptions are uniformly distributed over 100 topics, and events are published with a rate of 1 event per topic at each cycle. The number of nodes varies between 100 and 10^9. The curves show that TERA gracefully scales as the number of nodes increases, up to a point after which the overhead due to view exchanges in the general overlay becomes dominant and is no longer compensated by event notifications (that only depends from the constant amount of subscriptions).

5 Related Work

Publish/subscribe systems based on peer-to-peer architectures have been introduced a few years ago with the development of topic-based systems built on top of Distributed Hash Tables (DHTs). SCRIBE [8] and Bayeux [26] are two pub/sub systems built on top of two DHT overlays (namely Pastry [22] and Tapestry [25]), which leverage their scalability, efficiency and self-organization capabilities. Systems like SCRIBE use the decoupled key/node mapping provided by the DHT to efficiently designate a *rendez-vous* node for each topic. This node is responsible for collecting each event published for that topic and diffusing it toward subscribed nodes. The main drawbacks of this approach are the presence of a single node responsible for the management of each topic (that can quickly become a hot spot for very popular topics) and the usage of the standard DHT routing protocol to disseminate each event (thus involving in the dissemination nodes that are not interested in the event).

An interesting variant of this technique was proposed by CAN [20]: members of the system subscribed to the same topic are clustered in a *separate* overlay where events belonging to the corresponding topic are simply flooded. From this point of view the architecture of [20] implements a mechanism for traffic confinement that is quite similar to TERA's one. However, in [20] a single access point exists for each topic overlay. Contrarily, TERA's outer-cluster routing does not impose a single access point for each topic overlay thus avoiding issues related to traffic hot spots and single point of failures, but rather makes every node subscribed to a topic a possible access point.

Note that problems linked to the presence of hot spots and single points of failure in DHT-based systems can be mitigated introducing replication and load balancing techniques. However, these techniques introduce other problems (e.g. consistency of topic-related data structures) and, moreover, force a larger number of nodes to handle all the load for a specific topic, even if they are not interested in that topic.

Unstructured peer-to-peer systems were introduced as a substrate for topic-based event dissemination in [2]. The system proposed in that work maintains, through the widespread use of probabilistic algorithms, a hierarchy of groups that directly maps a topic hierarchy. Each group contains nodes subscribed to a specific topic and is maintained through a probabilistic membership protocol [14]. The lack in [2] of a general overlay network, not related to any specific topic, means that every publisher, before publishing an event, must become part of the group corresponding to the topic it wants to publish in. This also means that nodes playing the role of simple publishers receive events they are not subscribed to. Publishers in TERA are not required to join any topic overlay before publishing events; they are part of the general overlay, and the outer-cluster routing mechanism leverages it to disseminate events they produce.

Recently, an interesting work by Voulgaris *et al.* [24], proposes Sub-2-Sub, a solution to implement a content-based publish/subscribe system. In Sub-2-Sub subscribers sharing the same interests are clustered in ring-shaped overlay networks through a self-organizing algorithm that continuously analyzes overlapping intervals of interests. The work mainly focuses on the evaluation of this novel method for interest clustering and the related inner-cluster dissemination. Nevertheless, the outer-cluster routing issue is also addressed, even if it is not fully evaluated. In particular, a general overlay connects at lower level all nodes of the system and the outer-cluster routing is based on a gossip-based protocol which exploits the proximity of interests to speed up the outer-cluster dissemination and to involve as few non-interested nodes as possible. However, the outer-cluster routing mechanism proposed is deeply related to the content-based semantics, since it exploits partial overlapping of interests. Its direct application to a topic based systems, in which partial overlapping does not exist, reduces to a simple gossip.

6 Conclusions

This paper introduced TERA, a novel scalable architecture for topic-based event dissemination in unmanaged, large-scale peer-to-peer environments. Scalability of the proposed architecture has been assessed along several dimensions: number of nodes, subscriptions, topics and event publication rate. The paper presented, through both analytical and experimental studies, different aspects of the event dissemination mechanism paying most of the attention to the outer-cluster routing assessment. Results showed how TERA supports event dissemination reliably while confining traffic and achieving a fair load distribution.

After the encouraging results shown in this work, we are currently working on several aspects of the TERA's infrastructure to asses their performance. The first fundamental aspect that must be evaluated is TERA's behaviour in dynamic scenarios where nodes can subscribe/ unsubscribe topics and join/leave the system at any time. TERA's internal components and algorithms have been designed to be resilient to the dynamic behaviour of nodes, but the actual impact of such behaviours on system performance must be carefully evaluated. Preliminary tests confirm our intuition: the widespread adoption of randomized algorithms and data structures renders TERA resistant and adaptable to both subscriptions and nodes churn. A second aspect we are currently focusing on is the development of new algorithms for building and maintaining more sophisticated topic overlay networks. The main target is to improve inner-cluster dissemination by reducing generated traffic while achieving a high level of reliability. The improvements provided by such a solution, coupled with the outer-cluster routing mechanism introduced in this work, would constitute the basis for a highly reliable event diffusion infrastructure able to effectively confine traffic. Finally, we would like to point out that one of the most interesting properties of TERA is its ability to uniformly spread load among participants. However, while this characteristics is desirable in many settings, there can be scenarios where nodes heterogeneity can be smartly leveraged to ease the burden on less powerful nodes. In this respect we plan, as a future work, to study the impact of heterogeneous node capabilities (in terms of memory, bandwidth, or computation resources) on current TERA's algorithms, and possibly modify them accordingly.

References

[1] A. Allavena, A. Demers, and J. E. Hopcroft, *Correctness of a Gossip Based Membership Protocol*, Proceedings of the ACM annual symposium on Principles of Distributed Computing (PODC), 2005, pp. 292–301.

[2] S. Baehni, P. Th. Eugster, and R. Guerraoui, *Data-aware multicast.*, Proceedings of the International Conference on Dependable Systems and Networks (DSN), 2004, pp. 233–242.

[3] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom, and D.C. Sturman, *An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems*, Proceedings of International Conference on Distributed Computing Systems (ICDCS '99), 1999.

[4] Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky, *Bimodal multicast*, ACM Transactions on Computer Systems (TOCS) **17** (1999), no. 2, 41–88.

[5] Fengyun Cao and J. Pal Singh, *Efficient event routing in content-based publish-subscribe service networks*, Proceedings of the 23rd IEEE Conference on Computer Communications (INFOCOM) (Hong Kong, China), vol. 2, IEEE, Washington, 7-11 March 2004, pp. 929 – 940.

[6] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, *Design and evaluation of a wide-area notification service*, ACM Transactions on Computer Systems **3** (2001), no. 19, 332–383.

[7] A. Carzaniga and A.L. Wolf, *A benchmark suite for distributed publish/subscribe systems*, Tech. Report CU-CS-927-02, Software Engineering Research Laboratory, Department of Computer Science, University of Colorado at Boulder, 2002.

[8] M. Castro, P. Druschel, A. Kermarrec, and A. Rowston, *Scribe: A large-scale and decentralized application-level multicast infrastructure*, IEEE Journal on Selected Areas in Communications **20** (October 2002), no. 8.

[9] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec, *Lightweight Probabilistic Broadcast*, ACM Transanctions on Computer Systems **21** (2003), no. 4, 341–374.

[10] P.T. Eugster, P.A. Felber, R. Guerraoui, and A.-M. Kermarrec, *The many faces of publish/subscribe*, ACM Computing Surveys **35** (2003), no. 2, 114–131.

[11] I. Gupta, K. Birman, and R. van Renesse, *Fighting fire with fire: using randomized gossip to combat stochastic scalability limits*, Journal of Quality and Reliability Engineering International (2002).

[12] Márk Jelasity, Gian Paolo Jesi, Alberto Montresor, and Spyros Voulgaris, *Peersim*, `http://peersim.sourceforge.net/`.

[13] Márk Jelasity and Alberto Montresor, *Epidemic-style proactive aggregation in large overlay networks*, Proceedings of The 24th International Conference on Distributed Computing Systems (ICDCS), 2004, pp. 102–109.

[14] A.-M. Kermarrec, L. Massoulié, and A.J. Ganesh, *Probabilistic Reliable Dissemination in Large-Scale Systems*, IEEE Transactions on Parallel and Distributed Systems **14** (2003), no. 3.

[15] D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers, *Decentralized schemes for size estimation in large and dynamic groups*, Proceedings of the 4th IEEE International Symposium Network Computing and Applications (NCA), 2005.

[16] Laurent Massoulié, Erwan Le Merrer, Anne-Marie Kermarrec, and Ayalvadi Ganesh, *Peer counting and sampling in overlay networks: Random walk methods*, Proceedings of the 25th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), 2006.

[17] E. Le Merrer, A-M. Kermarrec, and L. Massoulie, *Peer to peer size estimation in large and dynamic networks: A comparative study*, Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing, 2006, pp. 7–17.

[18] B. Oki, M. Pfluegel, A. Siegel, and D. Skeen, *The information bus - an architecture for extensive distributed systems*, Proceedings of the 14th ACM Symposium on Operating Systems Principles (SOSP), 1993, pp. 58–68.

[19] D. Psaltoulis, D. Kostoulas, I. Gupta, K. Birman, and A. Demers, *Practical algorithms for size estimation in large and dynamic groups*, Proceedings of the 23rd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), 2005.

[20] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker, *Application-level multicast using content-addressable networks*, Lecture Notes in Computer Science **2233** (2001), 14–34.

[21] A. Riabov, Z. Liu, J.L. Wolf, P.S. Yu, and L. Zhang, *Clustering algorithms for content-based publication-subscription systems*, Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS), 2-5 July 2002, pp. 133–42.

[22] A. Rowstron and P. Druschel, *Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems*, Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 12-16 November 2001, pp. 329–350.

[23] S. Voulgaris, D. Gavidia, and M. van Steen, *CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays*, Journal of Network and Systems Management **13** (2005), no. 2.

[24] Spyros Voulgaris, Etienne Rivière, Anne-Marie Kermarrec, and Maarten van Steen, *Sub-2-sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks*, Research Report RR5772, INRIA, Rennes, France, December 2005.

[25] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, *Tapestry: A Resilient Global-scale Overlay for Service Deployment*, IEEE Journal on Selected Areas in Communications **22** (2003), no. 1, 41–53.

[26] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz, *Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination*, Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video, 25-26 June 2001, pp. 11–20.

Appendix A: Pseudo-code description

ST :	(Subscription Table) a set of tuples with the form $< topicID, overlayID >$ where $topicID$ is a topic identifier, and $overlayID$ is an overlay identifier.
APT :	(Access Point Table) a set of tuples with the form $< topicID, nodeID >$ where $topicID$ is a topic identifier, and $nodeID$ is a node identifier.

Table 1. Data structures

Algorithm: 1 - Subscription Management

On application subscribing topic t **do**
  if $\nexists\, i\ :< t, i >\in ST$ **then**
    $L \leftarrow lookup(t)$
    if $L = \emptyset$ **then**
      $i \leftarrow Instantiate()$
    else
      $i \leftarrow Join(t, n), n \in L$
    $ST \leftarrow < t, i >$
On application unsubscribing topic t **do**
  if $\exists\, i\ :< t, i >\in ST$ **then**
    $ST \rightarrow < t, i >$
    $Leave(i)$
Every T **time units do**
  $L \leftarrow GetSamples(D)$
  for all $n \in L$ **do**
    for all $< t, i >\in ST$ **do**
      $s \leftarrow GetSizeEstimation(i)$
      send subscription advertisement containing $< t, i, s >$ to n

Algorithm: 2 - Event Management

On publish or receive event e for topic t **do**
  if $\exists\, i\ :< t, i >\in ST$ **then**
    notify e to applications subscribed to t
    diffuse e in t's topic overlay
  else
    $L \leftarrow$ lookup access points for t
    if not $L = \emptyset$ **then**
      send event e for topic t to node n, where $n \in L$

Algorithm: 3 - Access Point Lookup

On lookup for topic t **do**
  if $< t, n >\in APT$ **then**
    $A \leftarrow n$
  else
    $L \leftarrow GetSamples(r)$
    for all $m \in L$ **do**
      $A \leftarrow$ starts a random walk through m and collects the result
  returns A
On receive subscription advertisement $< t, i, s >$ **from** n **do**
  if $< t, x >\in APT$ **then**
    $x \leftarrow n$
  else
    $APT \leftarrow < t, n >$ with probability $1/s$
    removes random entries from APT to match its maximum size

Algorithm: 4 - Partition merging

On receive subscription advertisement $< t, i, s >$ **from** n **do**
  if $< t, j >\in ST$ **and** $i \neq j$ **then**
    $j \leftarrow ForceViewExchange(t, j, n)$

$Instantiate() \rightarrow overlayID$: instantiate a new topic overlay and returns the corresponding overlay identifier.

$Join(topic, node) \rightarrow overlayID$: joins the topic overlay associated to *topic* using *node* as the bootstrap node, then returns the corresponding overlay identifier.

$Leave(overlayID)$: leaves the topic overlay identified by $overlayID$.

$GetSamples(num, overlayID) \rightarrow list$: returns a $list$ of num node identifiers sampled from the overlay identified by $overlayID$; if this parameter is omitted the samples are drawn from the general overlay.

$GetSizeEstimation(overlayID) \rightarrow num$: returns an estimated size for the topic overlay identified by $overlayID$.

$ForceViewExchange(topic, overlayID, node) \rightarrow overlayID$: execute a view exchange process for the overlay associated to *topic*, and locally identified by $overlayID$, with *node*; returns an overlay identifier that is deterministically (e.g. with a min/max function if identifiers are numerical) chosen between the local and remote ones.

Table 2. Functions provided by the Overlay Management Protocol.