Scientific Report: Resource friendly and large scale group communication on support for mobile middleware

Boris Koldehofe
Chalmers University of Technology

27th November 2004

The research groups of EPFL and Chalmers are both activly participating in the Minema network. Previous meetings have shown that there is a good match in research interests in particular in the area of large scale group communication. The purpose of this visit was to strengthen the coorperation between the research groups on a project in the context of Minema. During the time of the reserach visit (from 1st of September until 31st of October 2004), we have worked on Publish/Subscribe systems using P2P dissemination.

Publish/Subscribe systems allow processes to express their interest by performing a subscription. After a subscription a process will be asynchronously notified on all events corresponding to its interest. P2P dissemination allows to scale to a large number of processes and provide a failure resilient management of membership. Commonly one distinguishes between structured and unstructured P2P systems.

In our work we have identified two fundamental dissemination schemes on support for publish/subscribe: gossiping and application level multicast. The goal was to evaluate the differences with respect to fault tolerance, fair sharing, and message complexity depending on the structure of the network. In order to support fair sharing we have developed a topic aware membership algorithm which can be used in combination with gossiping and application level multicast. During the dissemination of events only those processes participate in the dissemintation which share an interest in the respected topic. At the same time the proposed algorithm provides the same message complexity as approaches in the litereature which do not provide topic awareness. Our work also gives evidence that a structured lookup service allows an efficient maintenance of data structures commonly used in unstructured P2P dissemination systems.

The details and the current status of our work are included in this report (cf Current Research Document). Our plan is to extend this paper in order to publish it to a major conference related to midlleware and publish/subscribe systems. As a next step in this direction Boris Koldehofe will visit EPFL again from 3rd ofJanuary until 8th of January 2005.

Current Reserach Document:

Sébastien Baehni, Rachid Guerraoui, Sidath B. Handurukande, Oana Jurca, Boris Koldehofe
EPFL Lausane
Chalmers University of Technology

Abstract

Research on probabilistic reliable P2P communication has shown how to provide lightweight scalable and decentralised membership which is highly resilient to failures. For topic based publish/subscribe several solutions building on structured and unstructured P2P networks have been proposed. This work identifi es two fundamental ways of achieving publish/subscribe in P2P systems, gossiping and application level multicast trees, and analyses differences with respect to fault tolerance, fair sharing, and message complexity depending on the provided structure and the dissemination scheme. We also propose a DHT based algorithm providing topic awareness, which allows to eliminates the use of parasite messages, i.e. messages propagated by a process which does not share an interest in the content of the message. The algorithm can be used in combination with the discussed generic dissemination schemes and the provided topic awareness does not add an overhead in message complexity compared to other DHT based solution. The algorithm can easily be integrated in existing DHT based solutions by generalisation of the underlying hash function. Although gossiping usually is associated with unstructured P2P communication, this work gives evidence that a structured lookup service allows an effi cient maintenance of data structures commonly used in unstructured P2P dissemination algorithms.

Keywords: *publish-subscribe, structured and unstructured P2P dissemination, group communication*

1 Introduction

Publish/subscribe is a frequently applied communication paradigm in distributed computing. A process can subscribe to events which correspond to its interest and will in return be asynchronously notified on events corresponding to information published by processes sharing the same interest. Publish/subscribe has been implemented using various ideas structuring the dissemination process including different paradigms like topic, content and type based publish/subscribe.

Recent work on building large scale publish/subscribe systems use lightweight decentralised P2P multicast dissemination. Besides scalability a common goal is to provide reliable event delivery in a highly dynamic environment. In this work we evaluate with respect to fair sharing, message complexity, fault tolerance, and reliability the use of lightweight P2P communication. Hereby we consider structured and unstructured P2P communication in combination with two fundamental dissemination schemes, gossiping and application level multicast trees. Structured and unstructured P2P dissemination need to maintain membership as well as maintain a view of suitable communication partners. While unstructured dissemination algorithms focus only on event dissemination by using for example gossiping, structured P2P systems also provide a lookup service which allows to route efficiently for a given key to its closest location. Such a structure is also called a *distributed hash table* (DHT).

In this work we evaluate *fairness* in topic based publish/subscribe by considering the distribution of the load induced by the dissemination of events. Fairness considers the interest of the subscriber, i.e. the amount of work performed by a process should correspond to its interest. In many exiting solutions a significant amount of work in forwarding messages is done by processes which are not interested in the message. In this case the messages are also called *parasite* messages. It may be still of advantage to accept parasite messages when this helps in reducing the overall load of messages forwarded by a process.

The results of this work, however, suggest that for the discussed dissemination systems there is no real trade-off between the use of parasite messages and overall work performed by processes. We propose a *topic aware* DHT, which can be used to disseminate efficiently messages in a publish/subscribe system without using during the dissemination any parasite messages. Moreover, the algorithm can support dissemination schemes in establishing topic hierarchies, which is an important property for many applications.

Topic awareness provides that a process, interested in a topic, can find in its immediate neighbourhood at least one process sharing the same interest. Topic awareness can be achieved by clustering processes with the same interest around close hash values such that routing to a process sharing the same interest involves only processes sharing the same interest. While usually DHTs distribute the location of processes uniformly in the hash domain, the topic aware hash table proposed in this work distributes topics uniformly in the hash domain. Processes that subscribe to a set of topics are physically mapped close to at least one topic of interest. The proposed algorithm can be integrated in existing structured P2P systems by generalising the underlying hash function.

We evaluate the impact of different dissemination schemes in combination with the proposed algorithm. While typically structured P2P systems rely on application level multicast when used for implementing publish/subscribe, we show that many variations of lightweight gossiping can be implemented in combination with a DHT. We compare the differences in reliability, fair sharing, and message complexity of the proposed dissemination schemes.

2 Background

The Publish/Subscribe paradigm allows processes to express their interest in events by performing an operation subscribe. After subscription, a process will receive events which correspond to the expressed interest. Publish/Subscribe systems may differ in the way they express interest such as topic based, content based, and type based publish/subscribe schemes (cf. [7]). Large scale publish/subscribe systems like [2, 15, 17] usually use topic based publish/subscribe systems. In topic based publish/subscribe each process subscribes to a list of topics. Any published event is associated with a topic and needs to be delivered to all processes interested in the respective topic. Often publish/subscribe systems consider hierarchies of topics and subtopics, i.e. a process subscribing to a topic will receive all events corresponding to the topic and all subtopics.

The implementation of scalable decentralised topic based publish/subscribe system commonly use group communication to handle interest management in form of group membership and uses the underlying multicast primitive to disseminate events to all group members. While in early research the focus was on reliable fault tolerant approaches, recent P2P based solutions trade the strong reliability guarantee with reliability expressed with probabilistic guarantee in order to scale to a large number of processes and provide a highly dynamic membership scheme.

One can distinguish among P2P solutions for large scale event dissemination between structured and unstructured P2P-systems. Structured P2P systems like [16, 13, 14, 1] handle dynamic membership and provide a lookup service which efficiently routes for a given key to its closest location. Such a structure is also referred to as a *distributed hash table* (DHT). Unstructured P2P solutions providing probabilistic reliability use the gossiping paradigm (cf. [6, 3, 8, 9, 10, 12]). In distributed computing, gossiping was introduced in the context of data replication, whereas with pbcast [3] gossiping became an attractive communication paradigm for large scale event dissemination. In gossiping protocols a process is assumed to maintain a set of processes forming its view. When disseminating an event processes which have received this event forward the event to a subset of their neighbours. Similar to an infectious disease under certain condition one can guarantee w.h.p. that every process of the system receives the respective event.

In publish/subscribe structured and unstructured P2P solutions have been proposed. In structured networks, approaches like [15] form an application level multicast tree. With help of the P2P system the application multicast tree can be maintained in as self-stabilising way. However, there is a difference in the involvement of processes depending whether they are leave or forwarder in the tree. Moreover, the root of the tree is not necessarily interested in the topic. When building topic hierarchies the root of multicast tree has to perform the largest amount of work. Distributing the load among the peers fairly is an issue in SplitStream [4] and Bullet [11] where the data is divided in smaller items and distributed using several streams. For SplitStream a peer may depending on its interest subscribe to a number of streams corresponding to its inbound requirement. Efficient ways of splitting data streams by providing fairness and a low amount of overall work has been analysed in [5].

Splitting data in smaller streams is an attractive property in the context of multimedia streams. In the context of publish/subscribe systems the events are often too small so that splitting data may be used only to provide redundancy.

Data aware multicast [2] is based on the gossiping paradigm and involves processes corresponding to the topics they have subscribed. Moreover, it considers the use of topic hierarchies. In the dissemination of events only processes are involved which have subscribed to the topic.

Application level multicast trees involve significantly fewer messages compared to gossiping. However, both schemes address also different reliability properties. Where for gossiping the issue is to inform with high probability all processes in the presence of faults, for application level multicast trees a large set of processes may be temporarily disconnected.

3 Notation and measures of interest

Let $T = \{t_1, \ldots, t_m\}$ denote a set of topics (dynamically changing over time). A *topic hierarchy* is a partial order relation $H(T) \subset T \times T$ where $(t_i, t_j) \in T$ if all published events corresponding to t_j correspond also to t_i. The topic hierarchy can be represented by a directed acyclic graph $D(T, E(T))$ where $(t_i, t_j) \in E(T)$ if and only if $(t_i, t_j) \in H(T)$ and $\forall (t_l, t_j) \in H(T) \Rightarrow (t_l, t_i) \in H(T)$

Let $P(G, H(T))$ denote a publish/subscribe system and $G = \{p_1, \ldots, p_n\}$ a group of ongoing joining and leaving processes. A process joining $P(G, H(T))$ becomes member of G and can subscribe to any topics in $H(T)$. For process $p \in G$ T_p denotes the set of topics p subscribed to. For all $t \in T_p$, p may also publish events. Let e denote an event corresponding to a topic t_i. The event dissemination system of P ensures with reliability parameter α to inform all processes which have subscribed to topics $\in \{t_j with (t_j, t_i) \in H(T)\}$ will receive e. Each process in G subscribing to a topic t_i will receive all published events corresponding to t_i with reliability α. The reliability parameter α may vary from best effort reliability, over probabilistic delivery guarantee, to strong reliability in the presence of failures.

We distinguish with respect to a publish/subscribe system $P(G, H(T))$ among following operations:

- *subscribe*$(t_1, \ldots, t_l)$: a process which is member of $P(H(T), G)$, will after performing this operation also be interested to know about topics $t_1, \ldots, t_l$.
- *unsubscribe*$(t_1, \ldots, t_l)$: after performing this operation, a process will not receive any events corresponding to topics $t_1, \ldots, t_l$.

We assume that each process knows a unique identifier associated with its physically location (e.g. a network address). For process p we write p_{id} when referring to the identifier of p_{id}. Moreover, a process knowing p_{id} can use point to point communication in order to send a message to p. We also assume the names of topics $\in T$ to be unique.

For evaluation of $P(G, H(T))$ we introduce the following complexity measures:

- *Message Complexity* counts the overall amount of messages which needs to be send in order to perform a publish operation of an event.
- *Work* considers the product of message size and distance. In this paper the events are assumed to be small, so we assume the message size to be constant as it is typical for many publish/subscribe systems.
- *Fair sharing* considers the ratio between events forwarded and events received. Ideally this is the same for all processes of the P2P system. We assume events to be of constant size.
- *Parasite messages* are the number of messages during the event dissemination which were created by a process not sharing an interest in the content of the event.

Let *hash* $\in \{0, \ldots, b-1\}^* \rightarrow \mathbb{Z}_M$ be a function known to all processes mapping a key of arbitrary length (e.g. a group or process identifier) to a domain of fixed length in a way that for two different keys a collision is unlikely to happen and the values appear uniformly distributed among the domain. Let *hash*$_1(x) = m_1 \in \{0, \ldots, b-1\}^{l_1}$ and *hash*$_2(y) = m_2 \in \{0, \ldots, b-1\}^{l_2}$. We define *hash*$_1(x) \circ$*hash*$_2(y)$ to be the concatenation of the strings m_1 and $m_2 \in \{0, \ldots, b-1\}^{l_1+l_2}$.

A distributed hash table (DHT) allows to map keys to a location maintained by a process. In particular, as described in detail in Section 4.1 we allow a process to be responsible for multiple locations. The size of the hash table is determined by the number of locations denoted by N. When referring to the number of processes which subscribed to a specific topic $t \in T$, we write N_t.

4 A topic aware DHT

This section describes an algorithm suitable to maintain dynamic subscriptions and unsubscriptions for topic based publish/subscribe considering also topic hierarchies. If there exists multiple processes interested in the same topic, the scheme guarantees to find a process performing a single routing step.

Moreover, for any topic $t \in T$ a failure free execution can inform all processes interested in topic t using $N_t - 1$ routing steps. Hence, the algorithm can be combined with dissemination schemes discussed in Section 5 such that the dissemination only involves processes which share an interest in the disseminated event.

The membership algorithm can be combined with various routing schemes for DHTs mentioned in the literature. However, we require a generalisation which allows processes to act using multiple roles and use routing depending on a role, as well as lookup information depending on a role. We outline the required generalisations in Section 4.1. In Section 4.2 we present a publish/subscribe membership scheme building on the generalised DHT which provides topic awareness. In Section 4.5 we show how to implement topic hierarchies.

4.1 Generalised DHT

Most commonly a join operation of process p with respect to a DHT considers the identifier p_{id}. A successful join operation maps p_{id} to a virtual address in the domain of the DHT, e.g. by using consistent hashing. In the following we describe a generalisation which allows a processes to maintain multiple roles within a DHT. Besides using p_{id}, process p can select a role denoted by r. The mapping of a process to an address of the DHT is achieved by using a generalised hash function *hash*(p_{id}, r). This way a process can join a DHT multiple times using different roles and use different virtual addresses. Any lookup of *hash*(p_{id}, r) is guaranteed to end in p. If q forwards a lookup operation to p, q will add to its request the virtual location *hash*(p_{id}, r). p will serve the lookup by using a routing table corresponding to r. For maintenance of its roles p uses a set L_p, called the *location view* of p. L_p is defined by

$$L_p := \{\textit{hash}(p_{id}, r) \mid p \text{ joined the DHT using role } r\}.$$

For each $v \in L_p$, p maintains a routing table $R_p(v)$. The implementation of $R_p(v)$ can be done using existing DHT protocols of the literature. We assume $R_p(v)$ provides a set of neighbours $Q_p(v)$ which have closest common prefix for values $x > v$ as well as for $x < v$. After performing a lookup of w using $R_p(v)$, $R_p(v)$ will point to a forwarder with value u whose common prefix is larger or equal to the common prefix of w and v.

4.2 Achieving topic awareness

The generalised scheme of Section 4.1 can be used to achieve a topic aware membership algorithm for publish/subscribe by maintaining only a small location view. The role of a process p can be any topic $t \in T_p$. Assuming there exists for a topic t more than one processes which have subscribed to t. *Topic awareness* for a publish/subscribe membership algorithm guarantees that any process p with $t \in T_p$ knows at least one other process, say q, with $t \in T_q$ such that q can serve a routing request without using any *parasite messages*.

As a first step we propose a hash function which can be used to implement topic awareness. From Lemma 4.1 we can obtain *hash*(p_{id}, t) such that for any pair of processes q and r with *hash*$(q_{id}, t) \in L_q$ and *hash*$(r_{id}, t) \in L_r$, q and r can route to each other without using any parasite messages.

Lemma 4.1 *Assume that for any pair of topics $t, t' \in T$* hash *is collision free, i.e.* $\text{hash}(t) \neq \text{hash}(t')$. *Let p denote a process in $P(H(T), G)$ interested in $t \in T$. If* $\text{hash}(p_{id}, t) := \text{hash}(t) \circ \text{hash}(p_{id})$ *and* $\text{hash}(p_{id}, t) \in L_p$ *then p can lookup any other process q with* $\text{hash}(q_{id}, t) \in L_q$ *by using no parasite messages.*

Corollary 4.1 *If $\forall p \in P(H(T), G), \forall t \in T_p$* $\text{hash}(p_{id}, t) \in L_p$ *then the generalised DHT provides topic awareness.*

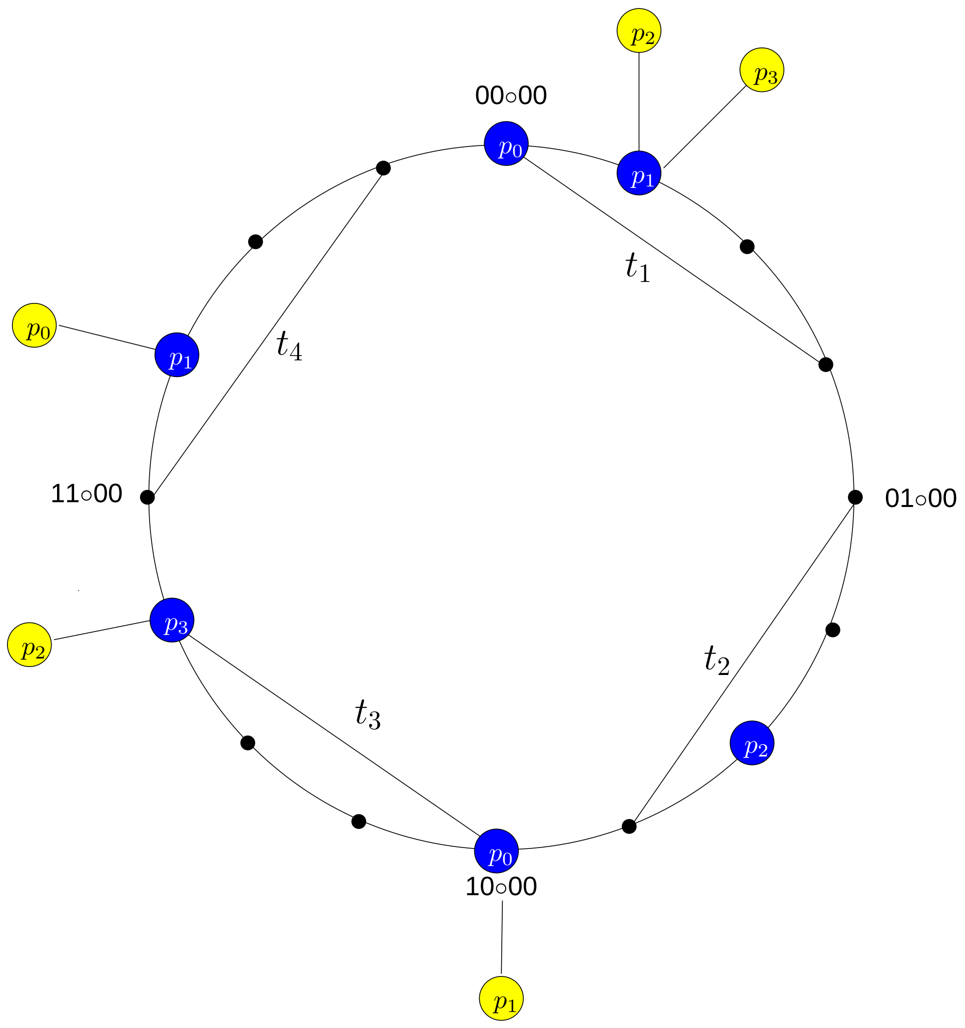

Figure 1: Forwarders and children in a topic aware DHT allowing a maximum of four topics and four processes.

Corollary 4.1 follows immediately from Lemma 4.1 and gives a first implementation to achieve topic awareness: for every subscription to topic t, p keeps an entry in its location view, i.e. $\mathit{hash}(p_{id}, t)$ exists in L_p.

We show in the following how to reduce the average size of the location view, i.e. a process uses in its location view only a subset $S_p \subset T_p$ such that $L_p = \{\mathit{hash}(t, p_{id}) \mid t \in S_p\}$. We achieve this by making a distinction between forwarders and children (cf. Figure 1). In this case case the average size of a location view is expected to be reduced proportional to the number of children accepted by a forwarder.

A process is called a *forwarder* with respect to a topic $t \in T_p$ if $\mathit{hash}(t, p_{id}) \in L_p$, otherwise it is said to be a *child*. For each value $v = \mathit{hash}(p_{id}, t) \in L_p$ a forwarder p maintains a set of at most k children $C_p(v)$ interested in t. For any child in $C_p(v)$, p is also called the *parent*.

If q is a child with $q \in C_p(\mathit{hash}(p_{id}, t))$, then p is the closest forwarder to $\mathit{hash}(q_{id}, t)$. This means any routing request to a child will reach its parent, which then can forward the routing request. A child q maintains a set of forwarders denoted by $F_q(v)$ which also includes its parent. $F_q(v)$ is typically initialised by its parent. Process q can use $F_q(v)$ to route to any process interested.

In the following we show how processes, based on subscriptions and unsubscriptions manage their location view. The algorithm uses following operations to maintain the parent child relation as well as its location view:

- *join*(p, q, t): Assumes q is the closest forwarder to $v_p = \mathit{hash}(p_{id}, t)$. After performing this operation, p will be a child of q, i.e. $p \in C_q(v_q := \mathit{hash}(q_{id}, t)$ and $F_p(v_p)$ contains q and neighbours of q in $R_q(v_q)$.

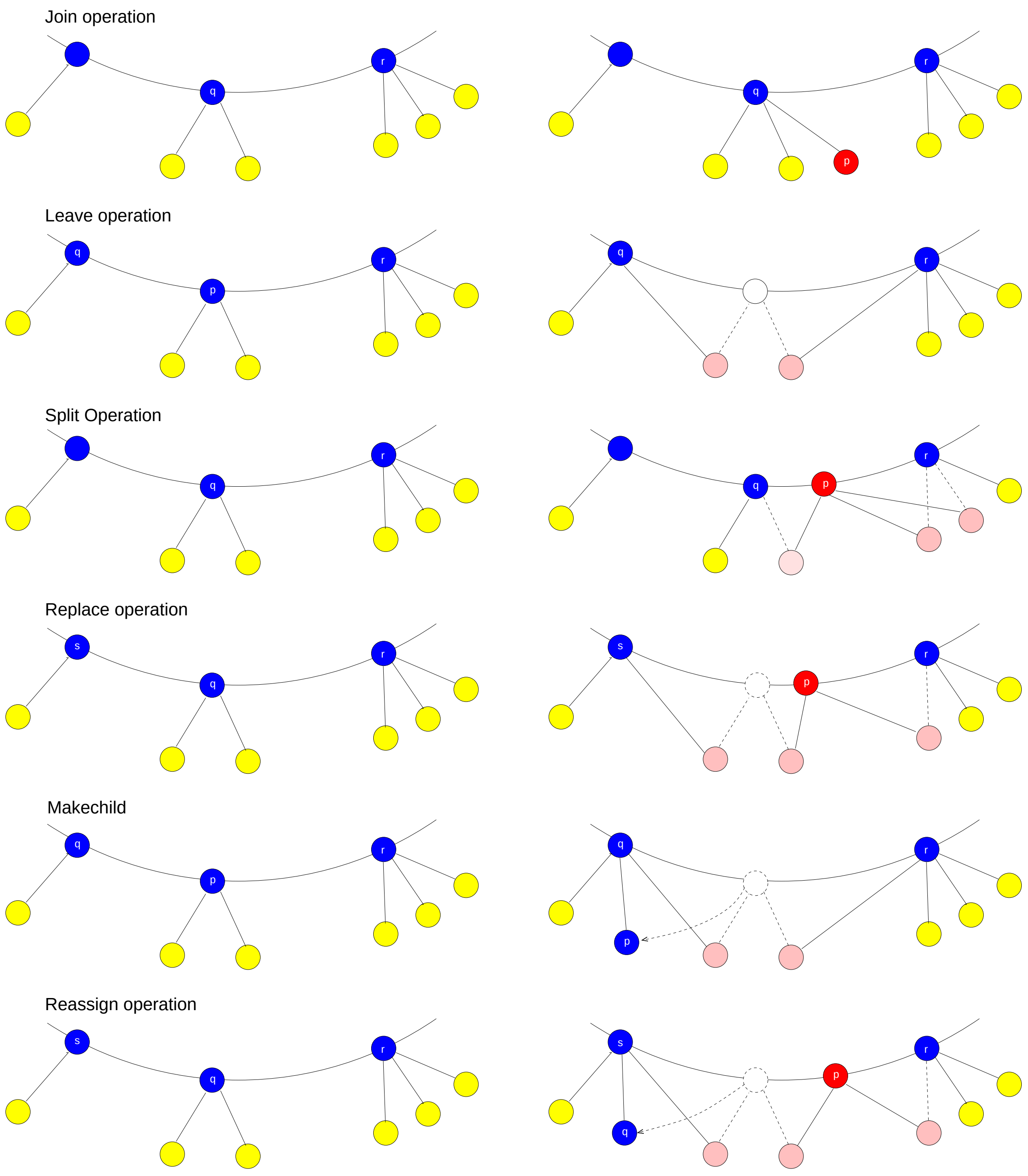

Figure 2: Algorithms operations when performing subscriptions and unsubscriptions

- *leave*(p,t): Assumes p is a forwarder of *hash*(p_{id},t) and there exists at least one neighbour interested in t. After performing this operation, p is neither forwarder nor child with respect of t. The children of p become children of the closest forwarder.
- *split*(p,q,r,t): Assumes q or r is a forwarder to *hash*(p_{id},t) with q being the closest forwarder to *hash*(p_{id},t) with *hash*(q_{id},t) $<$ *hash*(p_{id},t) and r being the closest forwarder with *hash*(r_{id},t) $>$ *hash*(p_{id},t). After performing this operation p, q and r are forwarders and the children of q and r become children at their closest forwarders.
- *split*(p,q,t): Assumes q is the closest forwarder to *hash*(p_{id},t). If *hash*(q_{id},t) $<$ *hash*(p_{id}) then q finds r which is closest forwarder to *hash*(p_{id},t) $>$ *hash*(p_{id},t) and performs *split*(p,q,r,t).

Otherwise q finds r which is closest forwarder to *hash*(p_{id}, t) with *hash*$(p_{id}, t) <$ *hash*(p_{id}, t) and performs *split*(p, r, q, t).

- *replace*(p, q, t): Assumes q is a forwarder to *hash*(p_{id}, t) with closest neighbours x, y. After performing this operation, q is neither forwarder nor child with respect of t, while p joins as a new forwarder. All children of q, x, y become children at their respective closest forwarder.
- *makechild*(p, q, t): Assumes p and q are forwarders for t where q is the closest forwarder to p. After performing this operation p becomes a child of q and all children of p are reassigned to their respective closest forwarders.
- *reassign*(p, q, t): Assumes q is the closest forwarder to *hash*(p_{id}, t). After performing this operation, p will be a forwarder with respect to topic t by performing first *split*(p, q, t). Thereafter also q becomes a child *makechild*(q, p, t).

Subscriptions. Let p be a process subscribing to a set of topics $t_1, \ldots t_l$. Initially routes to $v_{p_i} :=$ *hash*(t_i, p_{id}). Let $q_1, \ldots, q_l$ be processes where process q_i maintains $v_{q_i} :=$ *hash*(t_i, p_{id}). For any topic t_i which is maintained by q_i, but q_i is not interested in topic t_i, p becomes a forwarder by adding *hash*(t_i, p_{id}) to L_p. If L_p is still empty, p chooses q_i with *hash*(t_i, p_{id}) of q_i with maximum set $C_{q_i}(v_{q_i})$ and initiate operation *split*(p, q_i, t_i). After p operates as forwarder for at least one topic it may initiate operation *replace*(p, q_i, t) for forwarders q_i with

$$\frac{|L_{q_i}| - 1}{|T_{q_i}|} > \frac{|L_p| + 1}{|T_p|}.$$

Besides considering the size of the location view one may consider other metrics such as the forwarded and received traffic of a process.

Finally, p initiates for all i with $v_{p_i} \notin L_p$ operation *join*(p, q_i, t_i).

Unsubscriptions. If process p unsubscribes from topics $t_1, \ldots t_l$, it contacts all processes q_i which function as a forwarder for p to remove p from its set of known children. For all topics t_i for which p functions as a forwarder and maintains any children, p checks whether there exists another forwarder interested in the same topic. In this case p initiates operation *leave*(p, t_i). Otherwise p selects a suitable process q_i in $C_p($*hash*$(p, t_i))$ and initiates operation *replace*(q_i, p, t_i). If p is forwarder for topic t_i, but does not maintain any children for t_i, p simply removes *hash*(p, t_i) from its view.

Maintaining at most k children. Subscription and unsubscription may overload certain processes, i.e. forwarder p of t may become responsible for more than k children. In this case p selects a suitable child, say q and initiates operation *split*(q, p, t).

4.3 Correctness in the absence of failures

At first let us consider the case where operations do not interleave, i.e. every process is involved in at most one operation at a time.

Lemma 4.2 *If all operations during unsubscription and subscription perform correctly and do not interleave, the subscription and unsubscription algorithm guarantees, that a process will be forwarder or child for topic t if and only if it is interested in topic t. Moreover for every child p there exists a parent and the parent is closest to* $\text{hash}(pid, t)$.

Proof. Since initially a process is interested in no topics, we need to show that after p performs subscriptions or unsubscriptions, T_p changes accordingly and for any process q, T_q remains unaffected by the operations of p. Moreover, when inserting or removing a forwarder, all children need to be relocated to the closest parent.

When performing subscribe, p becomes forwarder for all topics where there is no other process. In this case there are no possible conflicts. If a subscribing process performs a *split*(p, q, t) operation, children of q become either children of q or of p. Otherwise p joins as a child to the closest forwarder and no other processes are affected.
When unsubscribing, we distinguish between the case where p unsubscribes as a child or as a forwarder. For topics for which p is a child unsubscribing by contacting the parent does not affect any other processes. If p is a forwarder and has no children p can leave the DHT with respect to the topic safely without affecting any children. Otherwise replace and leave ensures that all children of p and of p's closest neighbours are relocated to the closest forwarders.
Hence, if operations do not interfere, neither subscriptions and unsubscriptions affect any other processes and correctly maintain the forwarder child relationship. □

In order to deal with interleaving operations we introduce the possibility to abort operations which then need to be retried by the initiator. Every operation is of the form *op*(p, q, t) where p initiates the operation whereas q can abort the operation. q serves an operation once all previous operations initiated or requested have been completed. However, there are two exceptions: first when q detects the assumption for an operation is not satisfied any longer then aborts the operation immediately, second whenever it receives a join operation it either accepts or aborts it immediately.

Lemma 4.3 *Every join operation succeeds after a finite number of retries.*

Proof. *join*(p, q, t) will only be aborted after q has left the DHT, i.e. by performing operations leave or replace. In this case the number of forwarders which can be found in a DHT is strictly decreasing. After rerouting a finite number of steps p will either be routed to a forwarder q' accepting join or to a forwarder which is not interested in the topic. In this case p performs *join*$(p, \perp, t)$ and will be forwarder with respect to the topic. □

Lemma 4.4 *After initiation, a leave operation will succeed in a finite number of steps.*

Proof. When performing leave, a process p first leaves the DHT. Afterwards there will be no further operations which can interleave with p. p requests from all children to perform a join operation. Note that these children are not involved in any other operation and thus perform join immediately. Since according to Lemma 4.3 join operations are guaranteed to finish in a finite number of step, p will receive from all children acknowledgements after a finite time. □

Lemma 4.5 *Any sequence of interleaving operations maintains subscriptions and unsubscriptions correctly.*

4.4 Providing Failure resilience

In the situation of link or process failures it becomes costly to establish correctness as illustrated in the case of no failures. Instead we assume that in the non faulty case the execution behaves equivalent as described in the previous section. However, when failures occur we allow communication to timeout. In this case a process can still locally complete the operation. As a consequence some children may not be connected to a forwarder or to a forwarder which is not closest to them. Therfore every child performs a little protocol periodically which self-stabilises to a correct state.

Algorithm 1 illustrates how children become reassigned to their closest parent. Every child p routes to *hash*(p_id, t) and its parent by contacting a node in $F_p($*hash*$(p_id, t))$ which forwards the routing request. Let q be the process be closest to *hash*(p_id, t) receiving the routing request of p. If p is not a child of q, q adds p to its set of children. In any case q acknowledges to be the parent and forwards a recent view

Algorithm 1 Self-Stabilising Protocol finding for a child the closest parent

```
VAR
  $C_p$:           the parent known $p$
  $F_p$:           Alternative forwarders known to $p$
  $R_p$:           Routing table of $p$
  $Q_p$:           Neighbour set of the routing table of $p$
  $parent_p$:      Parent of $p$
  $recupdate$:     Truth value to check whether $p$ received an update
  $l$:             maximum number of alternative forwarders
Repeat periodically
  if $recupdate = true$ then
    Choose $q \in F_p(hash(p_i d, t))$
    Send to $q$ $\langle ROUTEPARENT, p_{id}, t\rangle$
  else
    Using $R_p(hash(p_i d), t')$ find $q = lookup(hash(p_i d, t))$
    Send to $q$ $\langle ISPARENT, p_{id}, t\rangle$
  end if
On $q$ receives $\langle ROUTEPARENT, p_{id}, t\rangle$
  Using $R_q(hash(q_i d), t)$ find $r = lookup(hash(p_i d, t))$
  Send to $r$ $\langle ISPARENT, p_{id}, t\rangle$
On $q$ receives $\langle ISPARENT, p_{id}, t\rangle$
  $C_q(hash(q_{id}, t)) = C_q(hash(q_{id}, t)) \cup p_i d$
  for $i = 1$ to $l$ do
    Choose $r_{id} \in Q_q(hash(q_{id}, t))$ uniformly at random
    $F'_p(hash(q_{id}, t)) = F'_p(hash(q_{id}, t)) \cup r_{id}$
  end for
  Send to $p$ $\langle ISCHILD, q_{id}, t, F'_p(hash(q_{id}, t))\rangle$
On $p$ receives $\langle ISCHILD, q_{id}, t, F'_p(hash(q_{id}, t))\rangle$
  recupdate = true
  $parent_p(hash(p_{id}, t)) = q_{id}$
  $F_p(hash(p_{id}, t)) = F'_p(hash(q_{id}, t)) \cup q_{id}$
```

of its neighbours to p. If p does not receive an acknowledgement, it can retry using an arbitrary role for routing.

Lemma 4.6 *If no failures occur after a finite number of steps any child will be located to the closest forwarder.*

4.5 Topic hierarchies

Lemma 4.7 *Let $G_t = \{p_1, \ldots, p_l\}$ denote a non empty set of processes that subscribed to topic t. The topic aware DHT algorithm of Section 4.2 allows to route to a process in G_t chosen uniformly at random.*

Proof. A process selects a random value *rand* and requests to route to *hash*$(rand, t)$. Since G_t is non empty, the topic aware algorithm guarantees that there exists at least one process being forwarder of t. The routing of the DHT guarantees to find the forwarder closest to *hash*$(rand, t)$. Since each forwarder maintains in its view the closest processes in G_t, a route will locate the process in G_t closest to *hash*$(rand, t)$. By the uniformness of *hash* this corresponds to selecting uniformly at random a process in G_t. □

Basic idea: If a process subscribes to a topic t of level i, is assumed to know the names of topic at lower level. Hence, it can find a lower level process by performing random routing.

5 Dissemination using topic awareness

Section 4 has shown how to achieve topic awareness in a DHT. This section focuses on dissemination schemes which remove parasite messages and aim to provide fair load sharing. We focus on two ways of achieving dissemination, gossiping and application level multicast. Gossiping is commonly used by unstructured P2P systems, however we show that one can maintain at low cost a suitable membership to implement gossiping by exploiting the uniform structure of DHTs. Application level multicast is the common way to achieve dissemination and can easily constructed using the routing table of the DHT. Differences between gossip based and application level multicast dissemination are based on reliability and message load given by the schemes. Gossip based dissemination aims at reliability expressed with high probability in the occurrence of failures. Application multicast in structured P2P systems, ensures self maintenance of the multicast tree, i.e. a failure of a process will cause restructuring of the application multicast. Reliability achieved by gossip based dissemination use high redundancy. Therefore the message complexity, i.e. $O(n \log n)$ push based unstructured protocols compared to $\Theta(n \log \log n)$ using combined push and pull is higher than by using application level multicast, which can be achieved by sending $\Theta(n)$ messages.

5.1 Gossip based dissemination

Gossip based protocols maintain a view of communication partners which may change over time. We consider a framework of gossiping in the spirit of pbcast. In every round a process communicates with k partners (k denotes the fanout) from its view chosen uniformly at random and informs them about recently informed events. Reliability depends on the way how the dissemination scheme terminates (cf. [10]), the fanout, and the size of the local view. Ideally the membership scheme allows to choose communication partners uniformly at random among all group members. Lpbcast addresses this issue by maintaining partial views and exchanging randomly selected communication partners, such that a random choice of a process from the local view, appears as choosing the destination uniformly at random among all processes. In Scamp randomness is used to initialise the view of a process. The view size needs to be in the $O(\log n + C)$ and processes have to gossip in every round with approximately $O(\log n)$ communication partners in order to achieve reliability w.h.p. (specify exactly).

In the following we show that DHTs can be used to efficiently maintain partial views before proposing how to combine gossiping in a fair manner with structured DHT of section 4. Algorithm 2 outlines a generic lightweight membership algorithm. A process maintains a view containing a maximum of l other group members. Depending on the fanout and the frequency the membership algorithm refreshes the view of a process. Refreshing the view happens by each process sending with probability f its own address to K processes of the system. The destinations of the processes are determined by routing to K values chosen uniformly at random within the domain of the DHT's hash function.

Lemma 5.1 *Assume hash provides a uniform mapping of process identifiers to their locations in a DHT. An algorithm using the membership of Algorithm 1 for $l = K$ and $f = 1$ corresponds to select neighbours uniformly at random from a full view. The membership algorithm provides an overhead of $O(K \log n)$ additional routing messages performed by each process.*

Remark 1 *pbcast implemented in an unstructured P2P network requires to maintain a full view. Lemma 5.1 shows that in a structured P2P network can reduce the view size to k by using an overhead of $O(k \log n)$ additional routing messages performed by each process. Algorithm 2 can also initialise the membership provided used in the dissemination scheme of Scamp. However, here we do not experience any overhead since the local view remains static.*

Remark 2 *Algorithm 2 can also implement lightweight gossiping by exploiting proximity of the membership information.*

Topic aware gossiping. In order to combine the DHT providing topic awareness of Section 4 with Algorithm 2, every process p needs to maintain for every topic $t \in T_p$ corresponding data structures e.g. gossip view, update frequency, etc. According to Lemma 4.7 routing to $r = \textit{hash}(\text{rand}, t)$ finds group member chosen uniformly at random among the processes which subscribed to topic t. Hence, we can generalise Algorithm 2 to maintain for each topic a gossip view. Dissemination of events with respect to topic t happens by using the corresponding view to t.

Algorithm 2 Generic Lightweight Membership Protocol

```
VAR
  $V_p$:   known group members of $p$
  $K$:     fanout
  $f$:     frequency the view gets updated
  $l$:     maximum size of a view
Initialisation
  for $i = 1$ to $l$ do
    $r =$ random value $\in hash(\{0, \ldots, b-1\}^*)$
    route $\langle REQUEST, p_{id} \rangle$ to $r$.
  end for
Do in every round with probability $f$
  for $i = 1$ to $K$ do
    $r =$ random value $\in hash(\{0, \ldots, b-1\}^*)$
    route $\langle REQUEST, p_{id} \rangle$ to $r$.
  end for
On $p$ receives $\langle MYADDRESS, q_{id} \rangle$
  $V_p = V_p \cup q_{id}$
  if $|V_p| > l$ then
    remove oldest identifi er s.t. $|V_p| = l$
  end if
On $p$ receives $\langle REQUEST, q_{id} \rangle$
  Send to $q$ $\langle MYADDRESS, p_{id} \rangle$
```

Remark 3 *As an alternative it may suffice to apply the gossiping protocol only for processes with* $\text{hash}(p_{id}, t) \in L_p$ *at the cost of lower reliability for processes which are children for* t *trading reliability with space constraints.*

5.2 Application Level Multicast forest

Using a topic aware DHT allows a space efficient construction of a forest of application level multicast trees which can be used to achieve a fair and efficient distribution of the work without using any parasite messages. Achieving a fair data distribution is coupled to all processes, interested in the same topic, performing equal amount of work when forwarding a message. We ensure this by constructing a forest in which every forwarder/process is a root of an application level multicast tree. By using the random routing technique, introduced in Section 4.5, the load of performing a multicast becomes evenly balanced and according to Lemma 4.7 only processes interested in the topic becomes involved. In the following we outline the construction and maintenance of the forest. In our description for simplicity we consider only forwarders. However note that it is straight forward to generalise this scheme to include all processes interested in the topic.

Let us consider a topic $t \in T$. Each forwarder of t is required to maintain $O(\log N_t)$ links. Similar to [5] we distinguish between level $0, \ldots, \lceil \log N_t \rceil - 1$ links. During the dissemination of an event the root of the tree will use level 0 links to forward the event. Accordingly a process will forward an event, once it has received the event via a level i link, to its level $i+1$ links.

Let p be a forwarder and let $m = |\mathit{hash}(\{0, \ldots, b\}^*|$. We propose a construction of an application level multicast where a level i link corresponds to the forwarders reachable when routing to

$$\mathit{hash}(t) \circ (\mathit{hash}(p_{id}) + m/2^i \bmod m)$$

and

$$\mathit{hash}(t) \circ (\mathit{hash}(p_{id}) - m/2^i \bmod m).$$

In this construction the highest level links are the closest neighbours in Q_p. Hence, a process can determine the number of levels needed locally by using its neighbourhood set. The maintenance of level links can happen in two ways: i) bounded to incoming links, i.e. from time to time a process reinitialises its outgoing level i links, or ii) bounded to outgoing links, i.e. processes route to their level i incoming links (using the same function).

Maintenance bounded to outgoing links is of advantage for quickly adapting to changes in the structure, e.g. in a construction which is bounded to incoming links a new process subscribing may need to wait some time before receiving disseminated events, while a construction which is bounded to outgoing links ensures that processes receive events immediately after a successful update of all incoming links.

In order to deal with failing processes the root of a multicast tree needs to send periodically heartbeat messages. Processes which did not receive any heartbeat messages via a level i link perform a lookup of the respective incoming link.

Further redundancy can be achieved by disseminating via multiple application level multicast tress. This ensures that in spite of a fixed number of failures a disseminated event reach all interested and failure free processes. Redundancy can be also used to save on heartbeat messages. When disseminating an event the identifiers of all root processes of used application multicast trees are attached to the event. A process which received the event only from a subset of these processes can determine which multicast tree did not perform correct and fix the respective level i link.

5.3 Dissemination using topic hierarchies

Both dissemination schemes of Section 5.1 and Section 5.2 can be used to consider topic hierarchies. The gossiping schemes of Section 5.1 can be combined with data aware multicast [2]. By modification of Algorithm 2 we can maintain a set of random vertices of the closest higher level topic for which there is at least one interested process. When gossiping, a part of a processes gossip message will go to higher level processes.

When using application level multicast the root of the tree is also responsible to forward the published event to a process of the next level. Again one can modify Algorithm 2 for processes to maintain a set of higher level processes which can be used to initiate a redundant application level multicast of the next higher level.

6 Evaluation and Analysis

We evaluate the dissemination schemes of Section 5 with respect to fair sharing, space complexity, time complexity, reliability. We distinguish between four protocols:

- *FullGossip*: The dissemination scheme uses gossiping and makes no distinction between forwarders and children.
- *ForwardGossip*: Only forwarders are used in a gossip view. A forwarder receiving an event for the first time forwards the event its set of children.
- *FullTreeCast*: Every process is part of an application level multicast for all topics subscribed.
- *ForwardTreeCast*: Only forwarders are used to built a forest of application level multicast trees. A forwarder also propagates an event to its children when receiving the event for the first time.

Fair sharing. In a gossip based protocol a process communicates with destinations chosen uniformly at random from its view. In a dynamic view initialised by Algorithm 2 every process/forwarder has the same probability of receiving an event and being involved in gossiping the event in the next round. Hence, every process/forwarder is expected to perform the same amount of work with respect to a topic. For the *FullGossip* protocol this implies that every process performs fairly. In the *ForwardGossip* protocol fairness depends on the distribution of forwarders and children. A process being forwarder of a hot topic has potentially to do more work than a forwarder of a low interest topic. If the distribution of traffic with respect to topics is known in advance we can use this information during subscription to achieve fairness (cf. Section 4.2). Otherwise the use of heuristics can give a fair distribution of the load among the processes.

The construction of application level multicast trees in Section 5.2 ensures that every process interested in a topic t is root of exactly one application level multicast tree. Choosing the root for disseminating uniformly at random guarantees that every process is expected to perform $2^i/N_t$ messages via a level i link. e.g. in half of the cases a process is expected to propagate messages while in the other half a process is a leave node in an application level multicast tree. Similar to gossip based protocols *FullGossip* provides full fairness, while *ForwardGossip* requires also a fair assignment of forwarders and children.

Space complexity. In combination with the topic aware DHT the main space cost is the maintenance of routing tables. For each value $v = \mathit{hash}(p,t) \in L_p$ a process keeps a routing table of size of $O(\log N)$. Note that the size of a view or the number of links is bounded by $O(\log N_t)$. Hence, the full view dissemination schemes needs $O(|T_p| \log N)$ while forwarder based dissemination schemes require space $O(|L_p| \log N)$. Assuming a fair distribution of children and forwarders the average space by a process depends on the overall number of topics maintained in a publish subscribe system and how densely each topic is populated. Each active topic needs at least one forwarder. Let k be the maximum number of children a forwarder excepts before performing a split operation. If for every $t \in T$ there exists at least k processes sharing an interest then $|L_p|$ can be bounded by $1 + |T_p|/(k+1)$, i.e. achieving a space reduction assumes the topic space is densely populated.

A more significant space reduction can be achieved for densely populated topics if one associates with children a tree like structure. Each vertice of the tree corresponds to a DHT using $O(\log N_t)$ vertices. Hence, the depth of the tree is in the order of $O(\log N_t / \log\log N_t)$ and each childs routing table is bounded to $O(\log\log N_t)$. For determining the correct position we take a hash function which maps p_{id} similar to CAN [13] to a cartesian product of $O(\log\log N_t)$ dimensions and additional a level chosen uniformly at random. In this case the value of each dimension determines the path to follow while the level determines at which level to join a DHT. Within a DHT a process always routes to the process closest to $\mathit{hash}(p_i d)$. By using this scheme in a densely populated topic space a processes needs only to be forwarder of $O(\log |T_p|)$ topics and maintain a routing table of size $O(\log N)$ wheras for the remaining topics a process maintains a routing table of size $O(\log\log N)$, resulting in an overall space complexity of $O(\log |T_p| \log N + |T_p| \log\log N)$. Although the scheme is more space efficient when topics are densely populated and one could use a similar protocol like Algorithm 1 to self-stabilise children to be closest to their forwarder, the maintenance cost and overhead are significantly higher.

Time complexity. For a topic t gossip based dissemination and aplication level multicast terminate in $O(\log N_t)$ rounds. When using topic hierachies, in each round a topic will reduce the distance to processes which have subscribed to higher level topics. In this case the worst time complexity is $O(\log N_t \log |T|)$. Distinguishing between forwarders and children adds only a constant cost.

Reliability. Application level multicats protocol can tolerate up to f failures when using $f+1$ different root vertices. When distinguishing between forwarders and children then a failure of a forwarder affects at most k children. Hence, f failures will affect kf processes not receiving the message.

Gossip based dissemination informs w.h.p., i.e. for a constant c with probaility $O(1 - n^{-c})$ all processes assuming for a constant $1 > \epsilon > 0$, there exists at least $(1-\epsilon)$ non faulty processes. In case when distinguishing between forwarders and children also every failing process will affect k children.

7 Conclusion

This work has proposed a topic aware membership algorithm suitable for large scale topic based publish subscribe systems. In combination with fundamental dissemination schemes, topic awareness ensures a fair and efficient distribution of events by maintaining also the benifits of recent structured P2P membership schemes in providing a failure resilient maintenance of membership. Topic awareness allows to deploy the technique of random lookups which find for a given topic a process chosen uniformly at random which has subscribed to the topic. This technique gives benifits when establishing topic hierachies, but also supports failure redundant event dissemination.

The results of this work suggest that the distinction between structured and unstructured P2P dissemination systems is not a matter of the used dissemination system, but the structure of the P2P system matters in how to bootstrap data structures used for dissemenating events.

In the context of gossiping this work has shown that one can use a structured P2P system in order to establish in a space efficient way a dynamic set of randomly chosen communication partners. The cost are $O(\log N)$ routing messages per round compared to $O(1)$ when maintaining a static membership. This corresponds to the higher space complexity of dynamic views over static views in unstructured networks.

References

[1] Luc Onana Alima, Ali Ghodsi, Per Brand, and Seif Haridi. Multicast in DKS(N; k; f) overlay networks. In *In Proceedings of the 7th International Conference on Principles of Distributed Systems (OPODIS '03)*. LNCS Springer-Verlag, 2003.

[2] Sébastien Baehni, Patrick Th. Eugster, and Rachid Guerraoui. Data-aware multicast. In *Proceedings of the 5th IEEE International Conference on Dependable Systems and Networks (DSN 2004)*, 2004.

[3] Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, may 1999.

[4] Miguel Castro, Peter Druschel, Kermarrec Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, volume 37, 5 of *Operating Systems Review*, pages 298–313, New York, October 19–22 2003. ACM Press.

[5] Dahlia Malkhi Danny Bickson and David Rabinowitz. Efficient large scale content distribution. In *Proceedings of the 6th Workshop on Distributed Data and Structures (WDAS'2004)*, 2004.

[6] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, pages 1–12, Vancouver, BC, Canada, 1987. ACM Press.

[7] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[8] Patrick Th. Eugster, Rachid Guerraoui, Sidath B. Handurukande, Anne-Marie Kermarrec, and Petr Kouznetsov. Lightweight probabilistic broadcast. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2001)*, pages 443–452, Gothenburg, Sweden, July 2001.

[9] Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié. Scamp: Peer-to-peer lightweight membership service for large-scale group communication. In *Proceedings of the Third International COST264 Workshop, LNCS 2233*, pages 44–55, Berlin Heidelberg, 2001. Springer-Verlag.

[10] Boris Koldehofe. Buffer management in probabilistic peer-to-peer communication protocols. In *Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS '03)*, pages 76–85. IEEE, October 2003.

[11] Dejan Kostić, Adolfo Rodriguez, Jeannie Albrecht, and Amin Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, volume 37, 5 of *Operating Systems Review*, pages 282–297, New York, October 19–22 2003. ACM Press.

[12] J. Pereira, L. Rodrigues, M.J. Monteiro, and A.-M. Kermarrec. NEEM: Network-friendly epidemic multicast. In *Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS '03)*, pages 15–24. IEEE, October 2003.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 161–172, 2001.

[14] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[15] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Proceedings of the Third International COST264 Workshop on Networked Group Communication*, pages 30–43. Springer-Verlag, 2001.

[16] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM 2001 Conference*, pages 149–160, New York, August 2001. ACM Press.

[17] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20. ACM Press, 2001.