Efficient application-level multicast on a network-aware self-organizing overlay

Laurent Massoulié, Anne-Marie Kermarrec, Ayalvadi J. Ganesh

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Efficient application-level multicast on a network-aware self-organizing overlay

Anne-Marie Kermarrec , Laurent Massoulié and Ayalvadi J. Ganesh

N°5502

Novembre 2004

Systèmes numériques

ISSN 0249-6399 ISRN INRIA/RR--5502--FR+ENG

Efficient application-level multicast on a network-aware self-organizing overlay

Anne-Marie Kermarrec *, Laurent Massoulié † and Ayalvadi J. Ganesh ‡

Systèmes numériques
Projet Paris

Rapport de recherche n°5502 — Novembre 2004 — 20 pages

Abstract: The growth of peer-to-peer applications on the Internet motivates interest in general purpose overlay networks. The construction of overlays connecting a large population of transient nodes poses several challenges. First, connections in the overlay should reflect the underlying network topology, in order to avoid overloading the network and to allow good application performance. Second, connectivity among active nodes of the overlay should be maintained, even in the presence of high failure rates or when a large proportion of nodes is not active. Finally, the cost of using the overlay should be spread evenly among peer nodes for fairness reasons as well as for the sake of application performance. To preserve scalability, we seek solutions to these issues that can be implemented in a fully decentralized manner and rely only on local knowledge from each node.

In this paper, we propose an algorithm called the *Localiser* which addresses these three key challenges. We then describe the design of an efficient tree-based multicast system which builds multicast trees on top of the overlay produced by the localiser, and evaluate its performance in supporting application-layer multicast.

Key-words: Unstructured overlay networks, peer-to-peer, application-level multicast, random graphs, fault tolerance, network locality, load balancing.

(Résumé : tsvp)

* Anne-Marie.Kermarrec@irisa.fr
† MSR Cambridge, UK, lmassoul@microsoft.com
‡ MSR Cambridge, UK, ajg@microsoft.com

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Protocole de diffusion efficace dans un système pair à pair optimisé

Résumé : L'essor des applications pair à pair sur Internet génère un intérêt croissant pour les systèmes pair à pair génériques. La construction de tels réseaux permettant de connecter un grand nombre de pairs dynamiques pose de nombreux défis. Tout d'abord, il est important que les connexions virtuelles du système pair à pair reflètent la topologie sous jacente, et ce, afin d'éviter de surcharger le réseau et pour assurer des performances raisonnables aux applications. Ensuite, la connectivité dans le réseau doit être assurée en présence de taux de défaillance élevés ou lorsque seulement une portion faible des participants est active. Enfin, la charge de maintenance du réseau doit être partagée équitablement entre les pairs. Afin de préserver les propriétés de passage à l'échelle des systèmes considérés, nous nous concentrons sur des solutions totalement décentralisées où chaque participant ne dispose que d'une connaissance locale du système.

Dans ce papier, nous proposons un algorithme adressant les trois objectifs mentionnés ci-dessus. Nous décrivons ensuite un algorithme de construction d'arbre permettant de tirer profit des propriétés du réseau obtenu afin de fournir une fonctionnalité de diffusion applicative performante. Une évaluation de performances par simulation, atteste de ces bons résultats.

Mots-clé : Réseau pair à pair non structurés, diffusion applicative, graphes aléatoires, tolérance aux fautes, proximité réseau, équilibrage de charge.

1 Introduction

Peer-to-peer applications currently constitute one of the fastest growing uses of the Internet. A challenging research problem is to develop overlay architectures that can support such applications without overloading network resources. Features of peer-to-peer systems such as the transience of users and the absence of powerful central servers motivate fully decentralized schemes that do not require global knowledge of network topology or membership, and that are robust to node failures or disconnections. Scalability is crucial in this context, and overlays must be built ensuring that, as the overlay size increases, the potential load on each node grows slowly and the global load is evenly balanced among the nodes. Other objectives include low latency, fault tolerance, and ease of reconfiguration when failures occur.

Several recent approaches to building structured overlay networks [22, 21, 23] employ the abstraction of a distributed hash table (DHT). DHT schemes typically combine a global naming scheme based on hashing with the maintenance of hierarchically structured routing tables using the assigned names. This rich hierarchical structure of the name space can be leveraged in several ways by applications. Since node names typically don't reflect location in the network topology, additional mechanisms are needed to ensure geographically efficient routing in these systems. In an extension to the original CAN protocol, nodes measure their distance to a set of landmark nodes. They use these distances to compute their relative position in the overlay, thus partially reflecting the Internet topology. In Pastry and Tapestry [22, 26], nodes choose local nodes as much as possible to populate their routing tables.

The work described here differs in considering unstructured overlays, which are currently used in Internet-wide deployed systems such as Gnutella [15]. Several schemes have been proposed for building such overlays, which do not rely on global naming or any additional hierarchical structure [11, 13, 7]. In this paper, we work with the Scamp protocol [13], in which the only information maintained at each node is the list of addresses of a small subset of other nodes called its neighbours. Scamp provides probabilistic guarantees on global connectivity in the presence of failures by ensuring that each node has a sufficiently large degree (number of neighbours); the required degree is logarithmic in the group size [16], so the memory requirements on each node are scalable. Moreover, Scamp maintains this relationship between node degree and group size even as nodes join and leave the group.

A drawback that the Scamp protocol shares with many of the other schemes mentioned above is that it is oblivious to the underlying network topology. Hence, communication between peers tends to impose a high load on the network, thus limiting its applicability in wide-area settings. Moreover, the degrees of individual nodes are not tightly constrained, so some nodes may be responsible for communicating with many more peers than others. If these nodes sit behind low-bandwidth connections, application performance may be compromised.

Our first contribution in this paper is to propose a mechanism to optimize unstructured overlays according to a proximity criterion in order to reduce network load [19]. The same mechanism can be used, at virtually no extra computation or communication cost, to balance node degrees. This not only helps to balance the load, but also improves the resilience of the system to link and node failures. (If all nodes are not equal in their capabilities, either in terms of the load they can handle or

in terms of their reliability, then we may wish to have unbalanced degrees in a pre-specified fashion. Our mechanism can be adapted to this end, with no increase in complexity.) All these features are integrated into a single algorithm called the *localiser*, which functions in a fully decentralized fashion and requires no global knowledge.

Next, we turn our attention to peer-to-peer applications, focusing in particular on application-level multicast. The lack of deployment of IP multicast has led to considerable interest in application-layer multicast schemes [1, 3, 6, 11, 7]. Although these do not generally offer as good performance as network-layer approaches [8, 9], they can be deployed by end systems without support from the network layer. Directional Gossip [18] creates an overlay targeted at wide-area networks. It takes account of the network topology by computing a weight for each neighbour reflecting the connectivity and the network topology. However, this overlay construction leads to a static hierarchy sensitive to failures. The schemes described in Narada [7] and Nice [1, 2] rely on optimized mesh networks dedicated to application-layer multicast. The NICE infrastructure [1, 2] relies on a hierarchical structure which is updated to reflect the network topology as nodes join. The root of the hierarchy is in charge of dealing with every single join request and propagating it down the hierarchy in such a way that new nodes join the right low level cluster to reflect the network topology. In Narada, [7], an unstructured optimized overlay is built in a decentralized way. This mesh is then optimized according to several network metrics (load, distance, end node capacity) by computing a utility function that is continually updated. Narada requires each node to maintain the global membership information and to update it as nodes leaves and join the system. It is clear that these schemes have limited scalability. Since they are targeted at medium-sized groups, scalability is less of an issue. Nevertheless, it does motivate interest in more scalable solutions, and those that can be supported by general-purpose overlay networks. In fact, most of the above-mentioned schemes for constructing overlay networks can be used to support multicast. One such protocol is described in [5], which constructs multicast trees on top of the structured overlay described in [22].

Our second contribution in this paper is to propose and evaluate an algorithm for supporting application-layer multicast on top of an unstructured overlay. Specifically, we propose an algorithm to construct and maintain multicast trees on top of any overlay network. We compare the performance of this multicast scheme implemented on the Scamp overlay with that implemented on the optimized overlay produced by the *localiser*. We also compare it with IP multicast. We evaluate performance along several metrics including network load, delay and node overhead.

The rest of the paper is organized as follows. In Section 2, we give some background on the design requirements for overlay networks and give an overview of the Scamp protocol. In Section 3, we present the *localiser* algorithm for optimizing unstructured overlays along with a few performance results. In Section 4, we present an algorithm for constructing multicast trees on an unstructured overlay, and discuss how the properties of the overlay affect the multicast tree performance. We also present mechanisms to deal with node and link failures. In Section 5, we present performance results obtained through simulations on realistic network topologies. Section 5.5 contains a discussion of related work and conclusions.

2 Unstructured overlay networks

The basic property required of an overlay network is to be connected, so that any two peers can communicate over it. Moreover, connectivity should be maintained in the presence of failures or temporary disconnections of some fraction of member nodes, and the communication latency between any two members should not be too large. While a fully connected overlay would meet these requirements, it requires global knowledge of membership, which is obviously impractical in large and highly dynamic systems. We would like to achieve the same goals with a partial knowledge of membership at each member that scales well in the group size.

We shall model the overlay as a graph with nodes i representing members, and a directed edge (i, j) representing that i knows j. If the "who knows who" relation is symmetric, the edges are taken to be undirected. In the rest of the paper, we call the nodes to which a node is connected in the overlay its neighbours. The degree of a node is the number of neighbours it has. If the graph is directed, the in-degree of node i is the number of nodes that know i, while its out-degree is the number of nodes that i knows.

Erdös and Renyi [10] introduced the following classical model of a random graph: there are N nodes and an edge is present between each pair of nodes with probability p (which may depend on N), independent of other edges. If the graph is undirected, the mean degree of a node is $(N-1)p$, whereas if it is directed, $(N-1)p$ is the mean in-degree or out-degree. In the undirected case, they showed that there is a threshold for connectivity at a mean degree of $\log(N)$: if the mean degree is $\log N + x$, then the probability that the graph is connected goes to $e^{-e^{-x}}$ as $N \to \infty$.

These analytical results suggest that the membership protocol should be designed so as to provide each member with a set of neighbours whose size is of order $\log N$. In that case, the memory requirements on each member grow slowly in the overlay size, but the overlay network nevertheless has the desired properties. In addition, it is desirable that the set of neighbours at a node consist mainly of nearby nodes, in an appropriate network proximity metric. This is needed to ensure that communication on the overlay doesn't impose a high network load. Finally, we shall require the degree to be approximately the same at all nodes, both in order to balance load and to improve the resilience to failures.

Scamp The Scamp protocol, briefly described bellow, introduced in [12], achieves degrees of the desired size on average without maintaining any global information about the overlay size. This overview is included here for completeness as our localiser algorithm and multicast protocol are evaluated on overlays constructed by Scamp. However, it should be noted that the localiser algorithm could be used in other settings.

The Scamp joining algorithm works as follows:

1. New nodes join the system by contacting an arbitrary pre-existing member. The edge between the new member and the contact is used to connect the new member to the overlay.

2. When a node receives a join request, it forwards the new node ID to all its neighbours in the overlay. It forwards $c-1$ additional copies of the join request to randomly chosen neighbours.

3. When a node receives a forwarded join request, it accepts the new member as one of its neighbours with a probability p which depends on its degree in such a way that nodes with a low degree are more likely to connect to the new member. If it rejects the new connection, it forwards the request to a randomly chosen neighbour.

Note that the joining algorithm is fully decentralized: nodes can implement it using only local knowledge. It is shown in [12] that, when N nodes have joined the system, the overlay constructed by this protocol is connected, and each node is connected to $c\log(N)$ other nodes on average. The Scamp protocol also includes a lease mechanism to deal with leaving nodes and a heartbeat mechanism to improve fault tolerance. A detailed description and performance evaluation can be found in [13].

3 Network awareness and load balancing

All connection requests in Scamp are treated identically, and so the resulting overlay does not reflect the underlying network topology. As it would be more efficient if each node were to have local neighbours rather than remote ones, our first objective is to favour low cost connections, where cost refers to any specified network-centric criterion such as delay or spare capacity. Second, we note that Scamp targets an average node degree of $c\log(N)$ but does not tightly constrain individual node degrees. Our second objective is to reduce variability in node degrees. In addition to balancing the load, this can significantly increase resilience to failures.

We propose a Metropolis-type algorithm [20] (like simulated annealing at fixed temperature) to achieve both these goals. This is a simple iterative scheme for minimizing a function f on a domain D. Starting from any point $x \in D$, a move to a nearby point y is proposed, for example by sampling the step $y-x$ from a specified distribution. The move is then accepted with some probability which is decreasing in $f(y)-f(x)$. The reason for allowing moves that may increase f is to avoid becoming trapped at a local minimum. If the move is rejected, the new point is x. The process is repeated from the new point.

We adapt this basic idea to our problem by choosing an energy function incorporating our objectives. The energy function is

$$E(G) = w \sum_{i \in \mathcal{V}(G)} d_i^2 + \sum_{(i,j) \in \mathcal{E}(G)} c(i,j),$$

where the sum is taken over all vertices i and edges (i,j) in G, w is a weight parameter and $c(i,j)$ is a measure of the cost of communication between i and j. For example, $c(i,j)$ could be the ping time or a more complex measure incorporating bandwidth availability on the route between i and j. Note that this is a global function incorporating the degrees of all nodes and the cost of all links. The essential feature that enables us to obtain a decentralized protocol is that the difference in energy between “neighbouring" configurations can be computed locally, as shown below. We now give a detailed description of the protocol.

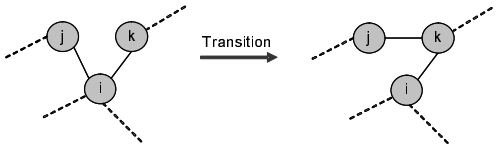

Figure 1: Effect of a local transition

3.1 Localiser algorithm

Each node maintains the IP address of its neighbours in the overlay in a data structure called `Degree`. We consider symmetric connections here.[1] Periodically, and at unit rate, each node i performs the following steps:

1. Choose two neighbours, node j and node k, uniformly at random from `Degree`, and measure the link costs $c(i,j)$ and $c(i,k)$;
2. Send messages to node j and node k, which send back respectively $d(j)$ and $d(k)$, their degrees. In addition, node j sends back its estimate of $c(j,k)$;
3. Evaluate locally the cost of replacing link (i,j) with link (j,k) as depicted on Figure 1. The cost is the energy difference between these configurations, which is $\Delta E = 2w(d_k - d_i + 1) + c(j,k) - c(i,j)$.
4. Perform the transition, as depicted on Figure 1, with probability $p = \min\left(\left(e^{-\Delta E/T}\frac{d_i(d_i-1)}{d_k(d_k+1)}\right),1\right)$. To this end, i sends messages to j and k asking them to establish connections with k and j respectively. In addition i and j get disconnected (after receiving an acknowledgment that the connection between j and k has been established).

Observe that the algorithm has a positive probability of making changes which would increase the energy; the reason is to avoid getting stuck at local minima of the energy function. Also note that the number of edges, and hence the mean degree, is conserved since the algorithm does not add or remove edges, but only moves them.

Denote by $\mathcal{G}(N,M)$ the space of labeled graphs on N nodes with M arcs. The localiser algorithm defines a reversible Markov chain on $\mathcal{G}(N,M)$. It is readily verified (see [4], for example) that the stationary distribution of this Markov chain is given by the Gibbs distribution on $\mathcal{G}(N,M)$ defined by

$$\pi(G) = \frac{1}{Z_T} e^{-E(G)/T} \mathbf{1}_{G \text{ is connected}}, \tag{1}$$

[1] Edges were oriented in the original Scamp protocol, but it is straightforward to modify it to construct an undirected graph.

where Z_T is a normalization constant. In other words, if the algorithm is stopped after a sufficiently long time, the resulting overlay is a random graph G whose probability distribution is close to that given in (1). If we take $T = \infty$, then $\pi(G)$ is the uniform distribution on all connected graphs with N nodes and M edges. This is the same as the distribution in the classical Erdös-Renyi model, conditioned on having M edges and on being connected; it takes no account of locality or of degree balance. As T decreases, the distribution $\pi(G)$ increasingly favours low energy configurations, namely those with good locality and with balanced degrees.

The localiser algorithm is parameterized by w and T:

1. The parameter w specifies the emphasis placed on degree balancing relative to locality. If $w = 0$, the graph is optimized for network proximity only. Evaluation has shown that the resulting maximum degree is large compared to the mean degree. As $w \to \infty$, balancing node degrees takes precedence over improving locality.

2. The parameter T (used to compute the probability of performing a transition) is called temperature and specifies a trade-off between accuracy - how close we get to the optimal configuration - and speed of convergence. Small values of T ensure that the system is more likely to concentrate on low energy configurations but can lead to slower convergence.

It is easy to modify the algorithm to account for different inherent capabilities of the nodes. Each node i chooses a weight η_i within some pre-specified range to reflect its capability. The energy function is specified as

$$E(G) = w \sum_{i \in \mathcal{V}(G)} (d_i/\eta_i)^2 + \sum_{(i,j) \in \mathcal{E}(G)} c(i,j).$$

We now use the same algorithm as above, with the obvious modification that

$$\Delta E = 2w\Big(\frac{d_k}{\eta_k} - \frac{d_i}{\eta_i} + \frac{1}{2\eta_k^2} + \frac{1}{2\eta_i^2}\Big) + c(j,k) - c(i,j).$$

The effect of the modification is that the algorithm seeks to equalize d_i/η_i instead of d_i.

For the sake of clarity, we have presented the localiser algorithm as acting on a overlay that has previously been constructed using Scamp or some other protocol. In practice, it would obviously be more efficient to integrate the localiser into the overlay construction protocol.

3.2 Resilience to failures

The Scamp protocol creates connected graphs in failure-free operation, and connectivity is preserved by the localiser. However, the graph could become disconnected due to link or node failures. The connectivity analysis of random graphs can easily be extended to account for link and node failures [16]. For example, if links fail with probability ε independently of each other, then the graph remains connected after failures if the initial mean degree is taken to be $(\log(N))/(1-\varepsilon)$. Thus, the overlay constructed by the Scamp protocol, with mean degree $c \log N$, can tolerate failure rates up to $(c-1)/c$ before losing connectivity.

It turns out that, as the failure rate increases, connectivity is initially lost because of the isolation of a small number of nodes rather than of a large connected component. Nodes which start out with a small degree (due to randomness in the Scamp protocol) are more likely to become isolated due to link failures. Thus, we may intuitively expect that balancing node degrees improves resilience to failures. This is shown to be the case in [14], where it is established that the overlay modified by the localiser retains connectivity even when a fraction up to $\exp(-\frac{1}{c})$ of nodes have failed. This is an improvement on the basic Scamp overlay, which, as noted above, can only tolerate failure rates up to $1-1/c$ before losing connectivity.

3.3 Localiser performance

The localiser algorithm was introduced in [19], which contains a detailed evaluation of the improvements it offers over a generic unstructured overlay such as the one built by Scamp. We just recall a few performance results for completness in this paper.

3.3.1 Experimental setting

We use a simple packet-level discrete event simulator and run the simulations on network topologies generated randomly according to the Georgia Tech (GT) [25] transit-stub model, as well as on a subgraph of a real-world corporate topology. We use two GT topologies in this paper: TOP-600 and TOP-5050 are composed respectively of 600 and 5050 core nodes. A LAN is attached to each core node. Each LAN has a star topology and is composed of 100 nodes on average. We consider groups composed of 50,000 nodes, selected at random from the 60,000 and 505,000 nodes of TOP-600 and TOP-5050 respectively. Link delays are modelled by simply assigning a propagation delay of 1ms to each LAN link and 40.5ms to each core link. We also use the Corpnet network topology model composed of 298 routers, generated using measurements of the world-wide Microsoft Corporation network. This topology is designated as *Corp* in the rest of the paper. Simulation results presented are the average over 10 simulations for each configuration. In this section, we only present results related to the TOP-5050. More results can be found in [19]. The other topologies will be considered in the application-multicast evaluation.

3.3.2 Simulation results

We evaluate the localiser along three metrics *(i)* to evaluate the impact of the degree balancing mechanism, we measure the mean and maximum degree, varying the weight w and the number of iterations; *(ii)* to assess to what extent the localiser captures the underlying network topology, we measure the mean edge length, namely the mean distance between a node and its neighbours in the overlay; *(iii)* finally, to assess the impact of the localiser on the resilience to failures, we measure the number of disconnected nodes as the number of faulty nodes increases. We compare the performance of the basic overlay produced by Scamp (using $c = 2$) with that of the refined overlay obtained by using Scamp and then applying the localiser algorithm.

(W, T, NbIt)	(0,0,0)	(10,1,100)	(10,1,1000)	(10,1,5000)
Max degree	61.2	38	27.8	27

Figure 2: Max value of the out degree in a 50,000 node system on Top-5050, mean value is approximately 20

(W, T, NbIt)	(0,0,0)	(10,1,100)	(10,1,1000)	(10,1,5000)
Mean distance	483	408	290	196
Stdv	472	396	274	182

Figure 3: Mean and standard deviation of distances to neighbours in a 50,000 nodes system on Top-5050

Degree balancing To evaluate the degree balancing mechanism, we ran experiments varying the value of w and the number of iterations of the localiser algorithm per node. In the rest of this section, the parameter values are given in the format (w, T, number of iterations per node), with (0,0,0) referring to a standard overlay built using Scamp. Figure 2 shows the maximum degree (maximum number of neighbours) in a 50,000 node system on TOP-5050. Results show that the localiser is indeed very effective in homogenizing node degrees quickly, and the maximum degree gets very close to the mean value as the number of iterations increases. The results show that the maximum degree gets close to the mean value after a reasonable number of iterations and so the localiser is indeed effective in balancing the load. We observed (see [19]) that 1000 iterations suffice in these examples to achieve perfect degree balancing, and 100 iterations suffice when w is large, i.e., the emphasis is on balancing degrees. In practice, we don't require perfect degree balancing - adequate balancing is achieved after 100 iterations even for $w = 10$, where the emphasis is more on improving locality.

Network awareness The second objective of the localiser is to enable the overlay to reflect the network topology. To evaluate its performance on this goal, we measure the distance from each node to its neighbours in the overlay. Figure 3 presents the mean distance and the standard deviation in a 50,000 node system on TOP-5050.

Resilience to failures In order to assess the resilience to failures, we measure the number of nodes that become disconnected as the number of faulty nodes increases from 0 to 25,000 (50% of the nodes) in steps of 250. In order to evaluate the number of disconnected nodes, we use a flooding message initiated at a random live node and compute the number of live nodes receiving the flooded message.

In Figure 4, we plot the number of disconnected nodes against the number of faulty nodes in TOP-5050. The first point to note from the figures is the good connectivity of the basic overlay produced by Scamp; we have chosen $c = 2$, so the Scamp overlay should be able to tolerate up

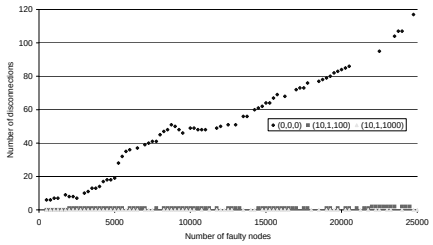

Figure 4: Resilience to failures in a 50,000 node overlay on Top-5050

to 50% failures. Indeed, the fraction of live nodes disconnected stays below 0.5% when up to 50% of the nodes fail. Secondly, both figures demonstrate that the localiser algorithm improves the resilience to failures significantly, after just a small number of iterations. The impact of the localiser increases with the number of faulty nodes. For example, in TOP-5050 when 50% of the nodes are faulty, 0.40% and 0.07% of live nodes are disconnected in, respectively, the standard overlay and the localiser (10,1,100). There were no disconnections observed in the localiser (10,1,1000) and (10,1,5000) configurations. For this reason, we have not shown results for the (10,1,5000) configuration. Disconnections (at a rate of 0.01%) were observed only when more than 60% of nodes are faulty.

4 Balzac: A tree-based application-level multicast protocol

We now describe a simple algorithm, called Balzac, for building multicast trees on top of a generic unstructured overlay. We will then evaluate it in detail on the overlays described in the last section.

4.1 Tree building

This tree is composed of shortest paths from the root to all other nodes using only links present in the overlay, as depicted in Figure 5. Any node may be the root of the multicast tree. The spanning tree is built as follows:

1. The source of the multicast initializes its own depth to 0 and floods a `createTree` message over the overlay. This message contains a node depth field, which is initialized to 1 by the source.

2. Each node, upon receiving its first `createTree` message, performs the following actions:

   - it takes as its parent in the tree that node from which it got this message.[2]

[2]This simple heuristic ensures that the tree contains paths which are shortest in terms of total delay. If it is desired to optimize some other network metric, the fboded `createTree` message should carry information about this metric and be suitably updated when it passes through each node.

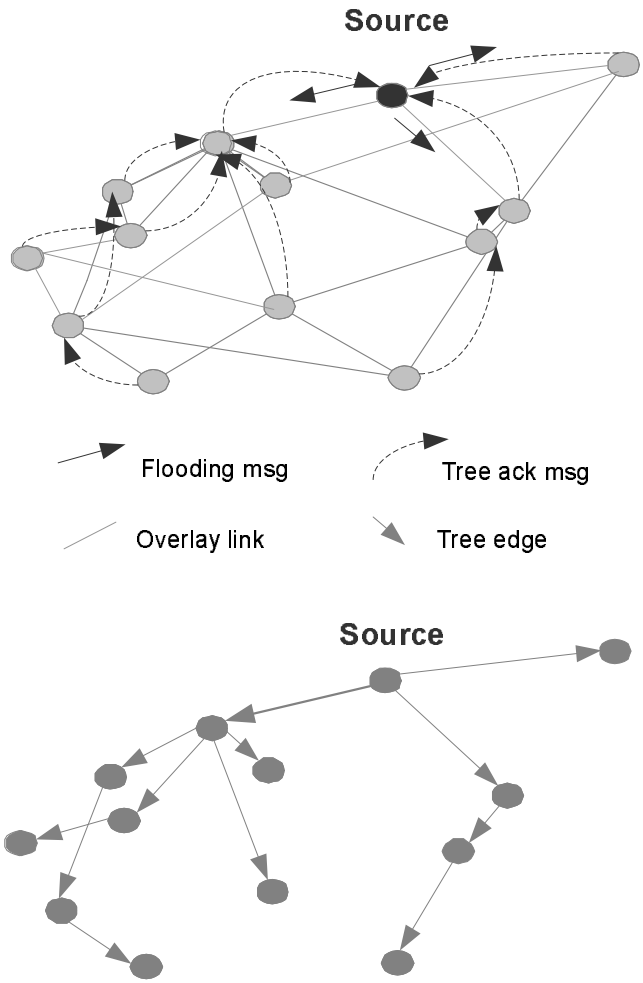

Figure 5: SPANNING TREE

- it sets its own depth to the value of the node depth field in this message. It then increments the node depth and forwards the `createTree`message to all its neighbours.
- It sends back a `parent` message to the node it chose as parent.

3. Upon receipt of a `parent` message, a node takes the sender as a child and updates its children table for the multicast tree accordingly.

Note that the degree of a node in the multicast tree is no bigger than its degree in the overlay, which is logarithmic in the group size. Thus, the multicast is scalable.

Unlike the overlay, the multicast tree is not fault tolerant. The failure of a single node or link disconnects all its descendants. Hence, we need a repair mechanism; we now describe one which only uses the underlying overlay.

4.2 Tree repair

Each node periodically sends a `still alive` message to its children in the tree. Should a node notice that such a message is not received as expected, it assumes that its parent has failed and attempts to reconnect, at either the same or a higher level in the tree. Say node i wants to reconnect. It then sends a `repair` message containing `nodedepth`(i) to all its neighbours in the overlay (except its own children, because they do not fulfill the depth constraint). Upon receipt of such a message, a node j replies if it has depth smaller than that of i and otherwise forwards the `repair` message to its parent in the tree. The node i then chooses as a parent the first node j which replied to its message. The reason for seeking to reconnect only to nodes at smaller depth is to ensure that a cycle is never created.

The repair mechanism introduces edges into the multicast tree that may not have been present in the overlay, and that may not be efficient in terms of network cost. It also increases variability in node degrees and may cause some nodes to become overloaded. In order to restore network efficiency and to ensure that the number of children per node is not much larger than the average node degree in the overlay, the trees are periodically refreshed. To this end, the source of the tree periodically multicasts a message using the flooding mechanism instead of the spanning tree. This gives nodes the opportunity to reconnect to the optimal parent in case repairs have provided them with sub-optimal parents. A complementary solution might be to use additional random links, as in [2]. Overlay links, other than those to parents or children, can be used for this purpose.

Suppose that the root never fails. Under this assumption, if there are no “simultaneous" failures (by which we mean two successive failures separated by less than the time needed to repair the first one), then the repair mechanism is guaranteed to work unless all the neighbours of node i are descendants of the same failed node. This is because a `repair` message from i to one of its neighbours j will, in the worst case, be forwarded until it gets to the root, which will then take on node i as its child. The event that all of node i’s neighbours are descendants of its parent is extremely rare if neighbours are chosen uniformly at random. It may be less rare when the overlay is highly clustered (optimised for network proximity). In either case, the repair mechanism needs to deal with this eventuality. One solution is as follows. If node i fails to get a response to its `repair` message within a specified timeout, then it can contact the root (source of the multicast) requesting it to use the tree refresh mechanism described above. Such a request initiates the construction of a fresh multicast tree, with connectivity of the overlay being sufficient to ensure that all active nodes will be incorporated into the multicast tree.

Iterations per node	0 (no opt.)	100	1000	5000
Avg. no. of hops from source	4.9	5.29	6.94	8.20
Depth	12	11.4	18.3	20

Table 1: Tree depth in a 50,000 node system on TOP-5050

5 Application-level multicast performance

The localiser evaluation (Section 3.3 and [19]) is along the metrics of maximum node degree, mean distance to neighbours (in terms of network delay) and resilience to failures. While they capture general features of the overlay network, they do not say very much about performance on specific applications. Here, we concentrate on evaluating the performance of application-layer multicast, using the multicast tree built by the Balzac algorithm. The performance evaluation is carried out through simulations on realistic network topologies, using the same setting as the one described in Section 3.3.

We compare the application-level multicast implemented in Balzac with IP multicast using a shortest path tree formed by the union of the unicast routes from the source of the tree to all nodes [8, 9]. The routing policy weights generated by the Georgia Tech random graph generator are used for this purpose.

The experiments are conducted as follows: *(i)* An initial overlay is produced using the Scamp joining protocol. All nodes join the overlay during this phase; *(ii)* During the second phase, we reshape the Scamp overlay using *Localiser* with parameters $w = 10$ and $T = 1$. (iii) We compare a Balzac tree built on each of these overlays, starting from a random node. The metrics considered are the delay to deliver multicast messages, the load imposed on the network and the load imposed on each node.

5.1 Tree Depth

The *Localiser* could potentially create an overlay with a large diameter in hop count, due to aggressive localisation of links. This would result in the depth of the multicast tree being large, and give rise to high latency if the forwarding delay at the nodes is significant. To see whether this is a problem in practice, we evaluated the depth and average distance from a node to the root on the Balzac tree built on the localised overlay. The results depicted in Tables 1 and 2 for TOP-5050 and *Corp*, show that the tree depth increases by a factor of two in the worst case, which we believe is an acceptable penalty for reducing network load.

5.2 Node overhead

In Balzac, as in any application-level multicast scheme, each node is in charge of forwarding multicast messages to its children in the tree. Hence the overhead, defined as the number of messages

Iterations per node	0 (no opt.)	100	1000	5000
Avg. no. of hops from source	7.37	9.01	8.01	9.17
Depth	19.8	19.6	17.8	20.4

Table 2: Tree depth in a 50,000 node system on Corp

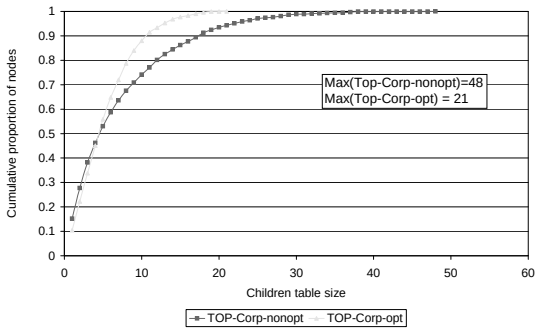

Figure 6: Cumulative distribution of number of children in a 50,000 node overlay on Corp

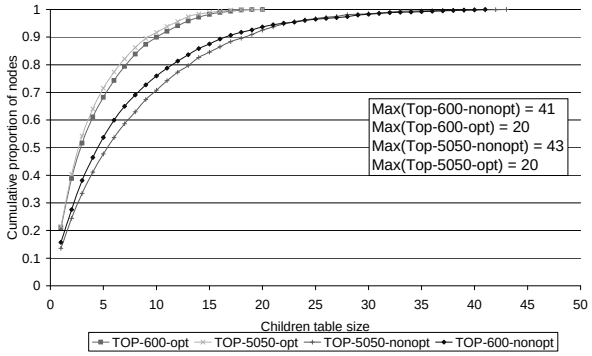

Figure 7: Cumulative distribution of number of children in a 50,000 node overlay on TOP-600 and TOP-5050

each node has to forward, is simply the number of children it has. Figures 6 and 7 display the cumulative distribution of the number of children per node for *Corp* and TOP-600 topologies. In these figures, the optimized overlay refers to the use of *Localiser* with 1000 iterations per node, while non-optimized overlay refers to the one produced by Scamp. We observe that the load is well-balanced, especially after optimization by the localiser, showing the scalability of the multicast.

Iterations per node	0 (no opt.)	100	1000	5000	10000
Mean Link stress	3	2	2	1	1
Max link stress	3800	1927	1277	636	265
% of links wo messages	32.68	30.25	27.6	35.62	36.89

Table 3: Link Stress in a 50,000 node system on TOP-600

Iterations per node	0 (no opt.)	100	1000	5000
Mean Link stress	3	2	2	1
Max link stress	1534	1105	934	144
% of links wo messages	39.37	38.87	31.14	36.91

Table 4: Link Stress in a 50,000 nodes system on TOP-5050

Iterations per node	0 (no opt.)	100	1000	5000
Mean Link stress	2	2	1	1
Max link stress	12849	13351	7899	3532
% of links wo messages	33.8	34	35	36.8

Table 5: Link Stress in a 50,000 nodes system on TOP-Corp

5.3 Network Load

IP multicast ensures that a message does not traverse a particular physical link more than once. For most network topologies, application level multicast is bound to send multiple message copies along some network links. To evaluate the network load, we measure the link stress, defined as the number of copies of each message carried by a physical link, upon transmission of a multicast message using the Balzac tree. Results, depicted in Tables 3, 4 and 5 show that the mean link stress is not large, but the maximum can be very large in the Scamp overlay. It is progressively reduced by *Localiser* to a substantial extent. The mean value of the link stress includes the non utilized links, or links which are not traversed by any message during the multicast. The percentage of such links is indicated in the Tables.

5.4 Relative delay penalty

The final set of evaluations compares delays in Balzac and IP multicast using the following metrics. We measure the relative delay penalty (RDP) for each node, defined as the ratio of the delay using Balzac to the IP delay. We measure two additional metrics: the Relative Maximum Delay (RMD) is the ratio of the maximum delay using Balzac to the maximum delay using IP multicast, while the Relative Average Delay (RMD) is the ratio of the mean delay using Balzac to the mean delay using IP multicast. RAD and RMD avoid the anomalies associated with simply averaging the RDP over all

Iterations per node	0 (no opt.)	100	1000	5000
RAD	2.15	1.64	1.30	0.82
RMD	4.33	1.57	1.37	0.86
Maximum RDP	450	242	172	2

Table 6: Delay in a 50,000 node system on TOP-600

Iterations per node	0 (no opt.)	100	1000	5000
RAD	2.93	2.38	1.67	1.16
RMD	4.46	2.14	1.24	0.97
Maximum RDP	679	485	206	55

Table 7: Delay in a 50,000 node system on Top-5050

Iterations per node	0 (no opt.)	100	1000	5000
RAD	1.72	1.23	1.14	1.1
RMD	2.12	1.62	1.19	1.19
Maximum RDP	34.85	14.5	3.74	3.16

Table 8: Delay in a 50,000 node system on Corp

nodes, because this average is dominated by those nodes with very small IP delay. For such nodes, the Balzac delay may be a very large multiple of the IP delay but they nevertheless see a small delay.

Figure 8 shows the cumulative distribution of the RDP metric in a 50,000 node group on TOP-600. The *y*-value of a point represents the proportion of nodes with a RDP lower than or equal to the *x*-value. For the sake of clarity we represented only the RDP up to value 10. In all configurations, more than 99% of nodes fall into that category; the maximum RDP values are indicated in Table 6. Tables 6 and 7 indicate the figures for TOP-600 and TOP-5050 respectivey. The results clearly show the improvement in RDP as the number of iterations of the *Localiser* algorithm increases. Table 8 indicates the figures for the *Corp* topology, we observe that the delays are better than on Georgia Tech topologies without any optimization, however the improvement after using the *Localiser* is still significant.

We observe from Table 6 that the RAD is 2.15 and the RMD is 4.33 for Balzac built on the Scamp overlay. These results are already good, and the reason is that the tree in Balzac is built using the shortest paths (according to delay). Furthermore, the delay decreases substantially after only 100 iterations of the *Localiser*.[3] Finally, the table shows that the maximum value of RDP is also substantially reduced by using the *Localiser*.

[3]Eventually, the delay using Balzac is better than the delay using IP. This is because IP routing does not always produce minimum delay routes as it is performed using the routing policy weights from the GeorgiaTech model [25].

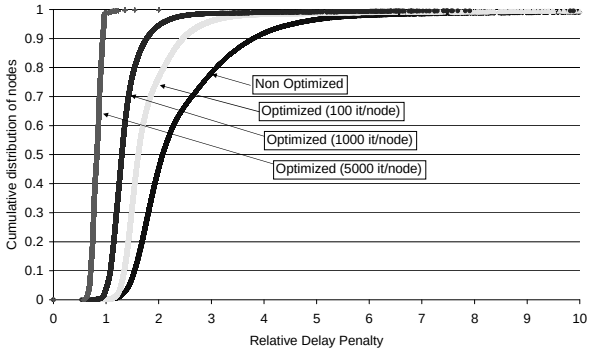

Figure 8: Relative delay penalty in a 50,000 node overlay on TOP-600, w=10, T=1

5.5 Conclusion

In this paper, we have proposed a localiser algorithm which optimizes large-scale unstructured overlay networks. The proposed algorithm serves multiple purposes: *(i)* it reflects the network topology by reshaping the overlay via a Metropolis scheme; *(ii)* it achieves load balancing by sharing the responsibility for connections evenly among the system nodes and finally *(iii)* it significantly improves the resilience to failures. The localiser is fully decentralized and only relies on local information available at each node.

We then illustrated the effectiveness of the optimized overlay on application-level multicast. We proposed a lightweight algorithm for constructing and maintaining a multicast tree on top of this overlay, and evaluated the performance of application-layer multicast in this setting. Performance evaluation shows that the scheme provides very good results as far as delay is concerned, with the delay being quite close to that achievable by IP multicast. This is due to the optimization of the underlying overlay combined with the fact that the tree is built while optimizing on the delay metric. The stress on network links imposed by our scheme remains much higher than for IP multicast; however, it is comparable to other application-level multicast protocols.

References

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, USA, August 2002.

[2] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. In *Proc. ACM SIGMETRICS*, San Diego, CA, USA, June 2003.

[3] K.P. Birman, M. Hayden, O.Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.

[4] P. Brémaud. *Markov chains*. Springer, 1999.

[5] M. Castro, P. Druschel, A-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8), October 2002.

[6] M. Castro, M. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. An evaluation of scalable application-level multicast built using peer-to-peer overlay networks. In *Infocom'2003*, San Francisco, CA, USA., March 2003.

[7] Yang hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8), October 2002.

[8] S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2), May 1990.

[9] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The PIM Architecture for Wide-Area Multicast Routing. *IEEE/ACM Transactions on Networking*, 4(2), April 1996.

[10] P. Erdös and A. Renyi. On the evolution of random graphs. *Mat Kutato Int. Közl*, 5(17):17–60, 1960.

[11] P.T. Eugster, R. Guerraoui, S.B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, To appear, 2003.

[12] A. Ganesh, A.-M. Kermarrec, and L. Massoulié. SCAMP: Peer-to-peer lightweight membership service for large scale group communication. In *Proc. NGC (Networked Group Communication) 2001*, J. Crowcroft and M. Hofmann eds., Springer, 2001.

[13] A. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2), February 2003.

[14] A. Ganesh and L. Massoulié. Failure resilience in balanced overlay networks. In *Proc. Forty-first Allerton Conference on Communication, Control and Computing*, October 2003.

[15] The Gnutella protocol specification, 2000. http://dss.clip2.com/GnutellaProtocol04.pdf.

[16] A.-M. Kermarrec, L. Massoulié, and A.J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3), March 2003.

[17] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vadhat. Bullet: High bandwidth data dissemination using an overlay mesh. In *SOSP'2003*, Bolton Landing, NY , USA., October 2003.

[18] M.-J. Lin and K. Marzullo. Directional gossip: Gossip in a wide-area network. Technical Report CS1999-0622, University of California, San Diego, Computer Science and Engineering, June 1999.

[19] L. Massoulié, A.-M. Kermarrec, and A.J. Ganesh. Network awareness and failure resilience in self-organizing overlay networks. In *Symposium on Reliable and Distributed Systems (SRDS '03)*, Florence, Italy, 6-8 October 2003.

[20] N. Metropolis, M. N. Rosenbluth, A. W. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[21] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. ACM SIGCOMM*, San Diego, CA, USA, August 2001.

[22] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, November 2001.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. ACM SIGCOMM*, San Diego, CA, USA, August 2001.

[24] R. van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed systems monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(3), 2003.

[25] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proc. IEEE INFOCOM*, San Francisco, CA, USA, 1996.

[26] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley, April 2001.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399