T-Man: Fast Gossip-based Construction of Large-Scale Overlay Topologies

Márk Jelasity Ozalp Babaoglu

Technical Report UBLCS-2004-7

May 2004

Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna (Italy)

T-Man: Fast Gossip-based Construction of Large-Scale Overlay Topologies[1]

Márk Jelasity[2] **Ozalp Babaoglu**[2]

Technical Report UBLCS-2004-7

May 2004

Abstract

Topology, as defined by the "who knows whom" relation, has many applications that are of crucial importance in distributed systems. For example, characteristics of the communication topology play a key role in determining the performance of many functions including search, routing, monitoring, control and information dissemination. Other essential properties of functions such as scalability and robustness to failures are also heavily dependant on the communication topology. The "who knows whom" relation can also be used to define solutions for problems such as sorting a set of numbers, clustering nodes based on similarity or pairing search requests with the locations of the information. In large-scale, dynamic, fully distributed systems constructing and maintaining a topology is a highly non-trivial problem. In this paper we identify topology management as an abstract service, independent of application or function, and propose a gossip-based scheme called T-MAN *for the construction of a large class of different topologies. The topology is defined by a ranking function which can be based on any appropriate property of components, including geographical location, semantic proximity, bandwidth, or simply on an abstract, pre-defined topology over an ID space like a ring or a torus. In the present work, we examine three specific topologies: ring, torus and binary tree. We give theoretical arguments and experimental results to demonstrate that the proposed protocol is lightweight and fast in that it converges quickly, approximately in logarithmic time, and that this speed is largely independent of the actual topology being constructed. We also demonstrate how to apply* T-MAN *for sorting a set of numbers. Finally, we address some practical issues and propose techniques to improve convergence during the end-phase and the bootstrapping problem of the protocol.*

1. This work was partially supported by the Future and Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923).
2. Department of Computer Science, University of Bologna, Italy

1 Introduction

In large, fully distributed systems, topology—the "who is connected to whom", or "who knows whom" relation—forms the basis of, or has a major impact on practically all functions.

Perhaps the most important example is communication topology which is a key concern in large and dynamic distributed systems consisting of thousands or millions of nodes. Firstly, topology has to support the application that is being implemented. For example, in the case of routing or search, the routing tables have to define a low diameter topology that also includes some informed hints on how to get closer to the destination. For most applications, including gossip-based dissemination and routing, it is also important that geographically long range links be used less often than short range links, meaning that topology also has to reflect physical proximity. In addition, topology has to be robust, that is, contain enough redundancy so as to prevent partitioning due to failures. Secondly, the costs of constructing and maintaining a desired topology have to be acceptably low. For example, no matter how desirable, it is not feasible to maintain a full view of a dynamic system at each node since the cost of maintenance would be unacceptable. Meeting both the effectiveness and efficiency requirements is challenging yet decisive for the success of an application.

But the "who knows whom" relation can have interpretations and uses other than simply communication. For example, it can define *sorting*, where each node seeks to know those nodes that preceed it and those that follow it according to some pre-defined total order relation. Achieving this in large, fully distributed systems is typically a difficult task. Similarly, the "who knows whom" relation can express search results, where each node learns about other nodes that hold a desired piece of information. and finally, it can express clustering, where nodes seek other nodes that are in some sense similar to them, share some interests or have the same goals as they do.

Motivated by these observations, we can think of topology management as a general purpose function that is desirable in large distributed systems. Quite remarkably, the above examples—communication topology, sorting, search, clustering—all share some requirements: the topology management algorithm has to be fast, scalable and accurate. To introduce a common framework, we formally define the *topology construction problem* as a cornerstone of topology management, and propose a gossip-based probabilistic solution to this problem called T-MAN. The topology is defined by a single ranking function that ranks nodes according to increasing distance from any given fixed node. Using only local gossip messages, T-MAN gradually evolves the topology to match it to the one defined by the ranking function. We show that the protocol is scalable and fast, with convergence times that grow as the logarithm of the network size. These properties allow it to be used even when several different topologies have to be created on demand, and also in dynamic systems where the set of nodes or the properties of nodes change rapidly. Additionally, the general formulation of the ranking function allows us to deal with a wide range of different topologies.

Related work Gossip-based protocols have recently gained notable popularity. Apart from traditional applications for database replication [1], gossiping algorithms have been applied to solve numerous other practical problems including failure detection [18], resource monitoring [17] and data aggregation [6, 7]. A recent survey by Eugster et al. provides an excellent introduction to the field [3]. In this paper we suggest a novel application of the gossip communication model to solve the topology construction problem.

Issues related to topology management have also received considerable attention in an effort to support functions like lookup and routing and to achieve robustness and scalability. In particular, distributed hashtables (DHTs) [11, 15, 16, 21] maintain structured topologies to support routing and key lookup, and unstructured overlays [2, 5, 9, 13] are constructed to achieve scalability, robustness and to support gossip-based information dissemination and data aggregation. In other contexts, superpeer topologies are used, mainly in peer-to-peer (P2P) filesharing networks [4, 20].

Our approach is different from the above protocols in that its scope is more general: a wide range of topologies can be constructed and maintained, in a rapid, scalable manner with an *extremely simple* and intuitive protocol. Unlike the above protocols, T-MAN can be used in large distributed systems to construct very different topologies or jumpstart other protocols (e.g., DHTs) rapidly *on demand* in a flexible and adaptive manner. We are aware of related work to achieve some, but not all of these features. For instance, Massoulié and Kermarrec [10] propose a protocol to evolve a topology that reflects proximity, in a manner not unlike our approach, yet the scope of the application is limited to proximity, while in our case this is only a special case. Voulgaris and van Steen [19] propose a method to jumpstart Pastry, a DHT based on a ring topology. However, the protocol they propose is tailored to that specific topology and it is unclear if it can be generalized beyond this application.

Outline In Section 2 we present the framework of our approach. We define the topology construction problem, and we present the T-MAN protocol for solving it. In Section 3 we specify the experimental settings, defining the problem instances we examine, along with the chosen measures of performance. Section 4 is concerned with characterizing the convergence of our solution. We present an approximate model that proves to be useful in predicting the logarithmic convergence of the protocol. Practical issues, like dynamism, synchronization and bootstrapping are touched on in Section 5. We also briefly sketch our solution to the sorting problem, an important application of the protocol. Section 6 presents empirical results with different parameter settings to verify our claims on convergence speed. Section 7 concludes the paper.

2 Basic Ideas

2.1 System model

We consider a set of nodes connected through a routed network. A node has an address that is necessary and sufficient for sending it a message. Each node maintains addresses of other nodes through a *partial view* (*view* for short), which is a set of c *node descriptors*. The parameter c is uniform for all nodes. In addition to an address, a node descriptor contains a *profile*, as we explain later. The addresses in the views of the nodes define the links of the *overlay network topology*, or simply the *topology*.

We assume that nodes have access to local clocks that measure the passage of real time. In the present paper we will make the simplifying assumption that there exists a single synchronization point when the protocol is started at all nodes. Furthermore, we assume that the communication channels and the nodes are reliable. These assumptions allow us to focus on the counter-intuitive, and therefore, more interesting properties of our protocol. We revisit these issues in Section 5 where we sketch how the simplifying assumptions can be relaxed in more practical settings.

2.2 The problem

Assuming the model above, we define the *topology construction problem*. The input of the problem is a set of N nodes, the view size c and a *ranking function* R that can order a list of nodes in increasing distance from a given node. The ranking function R takes as parameters a base node x and a set of nodes $\{y_1, \ldots, y_m\}$ and outputs the set of all possible orderings of these m nodes. We will say that $R(x, \{y_1, \ldots, y_m\})$ ranks y_i *strictly lower* than y_j if y_i precedes y_j in all of the possible rankings. This relation defines a partial ordering over the nodes $\{y_1, \ldots, y_m\}$.

In the first version of the problem, the task is to construct the views of the nodes such that for a node x the view of x view_x contains exactly the first c elements of a possible ranking of the entire node set, that is, $R(x, \{\text{all nodes except } x\})$ contains a ranking that starts with the elements of view_x. In other words, there exists no node y outside view_x such that y ranks strictly lower than any element from view_x. In a more general version of the problem, which we call the *topology embedding problem*, we are given another parameter $k < c$ and we require that view_x contains k elements that are the first k elements of some ranking from $R(x, \{\text{all nodes except } x\})$. In this case the remaining $c - k$ elements of the view can be arbitrary.

One way of generating ranking functions is through the function $d(x, y)$ that gives the distance between nodes x and y. When d defines a metric space on the set of nodes, the ranking function can simply order the given set according to the distance from the base node. In this case another version of the embedding problem can be defined, that is, we can require the view of a node to contain all nodes closer than a given distance to it, or a subset of such nodes if there are more than c of them. Note that this *distance based embedding* problem is not equivalent to embedding based purely on ranking.

In this paper we will examine three topologies, all are defined through a distance function. In the following we define these topologies. First let us recall that all nodes have a profile as part of their descriptor. It is this profile that the ranking function uses to calculate the possible rankings of nodes, so the definition of the data type and range of the profiles is always part of the definition of the ranking function.

```
view ← initialView()
do at a random time once in each
consecutive interval of T time units
  $p$ ← selectPeer()
  myDescriptor ← (myAddress,myProfile)
  buffer ← merge(view,{myDescriptor})
  send buffer to $p$
  receive view$_p$ from $p$
  buffer ← merge(view$_p$,view)
  view ← selectView(buffer)
```

(a) active thread

```
do forever
  ($q$, view$_q$) ← waitMessage()
  myDescriptor ← (myAddress,myprofile)
  buffer ← merge(view,{myDescriptor})
  send buffer to $q$
  buffer ← merge(view$_q$,view)
  view ← selectView(buffer)
```

(b) passive thread

Figure 1. The T-MAN protocol.

Line and ring Here, the profile of a node is a real number. The distance function for the line is $d(a, b) = |a - b|$. In the case of a ring, profiles are from an interval $[0, N]$ and distance is defined by $d(a, b) = \min(N - |a - b|, |a - b|)$ Ranking is defined through this distance function.

Mesh and torus The profiles are two-dimensional real vectors. The distance for the mesh is the Manhattan distance. It is given by calculating the one dimensional distance described above along the two coordinates and returning the sum of these distances. Applying the periodic boundary condition (as for the ring) results in a tube for one coordinate and a three dimensional torus for both coordinates. Again, ranking is defined through this distance function.

Binary tree The profiles are binary strings of length m, excluding the all zero string. Ranking is defined through the notion of distance as the shortest path length between the two nodes in the following undirected rooted binary tree. The string $0 \ldots 01$ is the root. Any string $0a_1 \ldots a_{m-1}$ has two child nodes $a_1 \ldots a_{m-1}0$ and $a_1 \ldots a_{m-1}1$. Strings starting with 1 are leafs. This topology is of interest because (unlike the previous ones) it has a very short (logarithmic) diameter of $2m$.

Sorting problems Distance functions are not the only way to define meaningful ranking functions. This is the reason we do not call the ranking function ordering because it is possible that no partial ordering is consistent with it. In other words, for a fixed base node, the ranking function is allowed to rank a strictly lower than b in a set of input nodes and strictly higher than b in another set. To illustrate this, consider the ranking function that can be used to construct a topology defined by the ordering of a set of numbers, as described in section 5.1.

2.3 The proposed solution

The topology construction problem becomes interesting when c is small and the number of nodes is very large. Randomized, gossip-based approaches in similar settings, but for other problem domains like information dissemination or data aggregation, have proven to be successful [3, 6]. Our solution to topology construction is also based on a gossip communication scheme.

We assume that each node executes the same protocol shown in Figure 1. The protocol consists of two threads: an active thread initiating communication with other nodes, and a passive thread waiting for incoming messages.

A view is a set of node descriptors. A call to MERGE(view$_1$,view$_2$) returns the union of view$_1$ and view$_2$. Method INITIALVIEW() is used to set the initial view of the node and is part of bootstrapping. It is desirable that this initial view be as close as possible to a random sample of the whole set of nodes. Fortunately there are a large number of protocols that provide us with such a service. We discuss this issue further in Section 5.

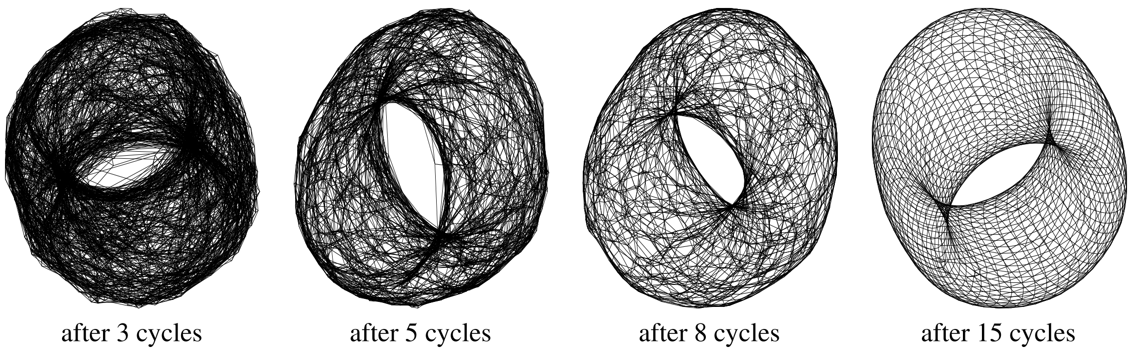


Figure 2. Illustrative example of constructing a torus over $50 \times 50 = 2500$ nodes, starting from a uniform random topology with $c = 20$. For clarity, only the nearest 4 neighbors (out of 20) of each node are displayed.

The two key methods are SELECTPEER and SELECTVIEW. Method SELECTPEER uses the current view to return an address. First, it applies the ranking function to rank the elements in the view. Next, it returns a random sample from the *first half* of the view according to ranking order. This choice of implementation will be motivated in Section 4 where we analyze the convergence of the protocol. Method SELECTVIEW(BUFFER) also applies the ranking function to rank the elements in the buffer. Afterwards, it returns the *first c elements* of the buffer according to ranking order.

The underlying idea is that this way nodes can optimize their own views using the views of their close neighbors, and since all nodes do the same, close neighbors become gradually closer and closer and so the views of the close neighbors will keep serving as a useful source of additional, even better links. Still, the behavior of the protocol is not easy to grasp since this highly interdependent dynamics of the views is rather non-trivial. In Section 4 we offer a fairly simple and intuitive way of modeling and understanding the protocol.

Although the protocol is not synchronous, it is often convenient to refer to *cycles* of the protocol. We define a cycle to be the time interval during which N view updates happen. Since each single contact results in two updates at the two involved nodes, and during $T/2$ time units $N/2$ contacts are initiated on average, we define $T/2$ units as one cycle. This means that during T time units *two* cycles are completed on average.

Figure 2 illustrates the results of the protocol when used to construct a small torus (visualizations were done using [8]). For this example, it is clear that 15 cycles are sufficient for convergence, and the target topology is already evident even after very few cycles. As we will see, T-MAN proves to be scalable so the time complexity of the protocol remains in this order of magnitude even for a million nodes. In the rest of the paper we take a closer look at the dynamics of the protocol on different topologies and parameter settings.

3 Experimental Setup

All the simulation results presented in this paper were produced using PEERSIM, an open source simulator developed at the University of Bologna [14].

The instances of the topology construction problem we examine are defined as follows. We apply the three distance-based ranking functions that define the ring, torus and binary tree topologies, as defined in Section 2.2. The motivation of this choice is that the ring is a large diameter topology and it is relevant for the sorting application (Section 5.1), the binary tree is of a logarithmic diameter and the torus is relevant in proximity problems being based on a 2-dimensional grid. The network sizes (N) examined are 2^{14}, 2^{17} and 2^{20}. We initialize the profiles of the nodes in a regular manner, that is, in the case of the ring topology, we assign the numbers $1, 2, \ldots, N$ to the nodes, and likewise for the torus $((1,1),(1,2),\ldots,(\sqrt{N},\sqrt{N}))$ and the binary tree (all binary strings of length $\log_2 N$).

This regularity is not critical for the success of the protocol. On the contrary, one of the most important applications is sorting an arbitrary set of numbers, as we argue in Section 5.1. However, this controlled setting allows us to monitor the dynamics of the protocol in a more informed manner as the distance function becomes equivalent to the hop count in the *target topology* defined by the links that connect nodes at distance 1. The measures of performance were chosen with the distance-based embedding problem in mind: we are interested in studying how efficiently the protocol finds links that connect nodes at distance 1. We will call those links *target links*. We will therefore focus on the dynamics of the number of target links as a function of cycles.

4 Analysis and Refinements

4.1 Rapid Convergence Phase

The key idea for understanding the dynamics of the system rests in the very first communication step of a node. In that step, two views are merged, both containing c random samples of the entire population. According to the protocol, the new view is the closest c elements according to a possible ranking. That is, the new view can be described as the closest c elements of $2c$ random samples. Note that this observation is independent of the ranking function, so it holds for all applications.

Following an inductive heuristic approach, we suggest a simple approximate model in which the view after cycle i (that is, after the ith communication) is the closest c elements out of $2^i c$ random samples. There are two main sources of error that account for deviations from this ideal model. The first one is what we call the unbalanced contact distribution. During T time units, a node itself initiates only one communication but can be contacted several times, the actual number being a random variable. In particular, if a node communicates too many times, it will converge faster than the others towards closer nodes and therefore it will provide less useful nodes to others when contacted, having a relatively more biased view already. On the other hand, if a node communicates too few times, it will converge slower than others and so the nodes it contacts will prove less useful since they will be typically far away and their views will be already more biased. The second source of error is related to the fact that even if assuming that the views at all nodes follow the model in some cycle i, the closest c nodes the model refers to are closest to the node holding the view. This means that when two nodes exchange their views after cycle i then—unlike during the first contact—the two sample sets held by the two views are not from exactly the same distribution.

To elaborate on this model and its implications, we first introduce the notion of the *maximal rank* $r_{a,b}$ of a node b with respect to a base node a. This value is the largest rank that node b can possibly be assigned among any subset of the entire population when ranked with a as a base node. For example, in the ring topology and the population of elements $1, \ldots, N$ we have $r_{j,j+1} = 2$ because $j+1$ is either the first or the second (which can happen when $j-1$ is present) irrespective of the other values. As another example, consider the random ranking function on the same population that returns a random ordering. This ranking is induced by the distance function according to which all pairs are equidistant. The resulting target topology is effectively a random topology, indeed, with this ranking function T-MAN becomes equivalent to LPBCAST [2]. In this case all maximal ranks are N for all nodes for all base nodes.

Now, consider those nodes that have maximal rank smaller than c. In particular, in the case of the three topologies we are considering, the nodes that are at distance 1 from a base node have a very small maximal rank. For a ring it is 2, for the torus it is 4 and for the binary tree it is 2, 3 or 1 if the base node is the root, an internal node or a leaf, respectively. Now, if we assume that our model is accurate, we can characterize the probability $p_{a,b}(i)$ that b is present in the view of a after cycle i. First of all, $p_{a,b}(0) = c/(N-1)$. Second, if $r_{a,b} < c$ then

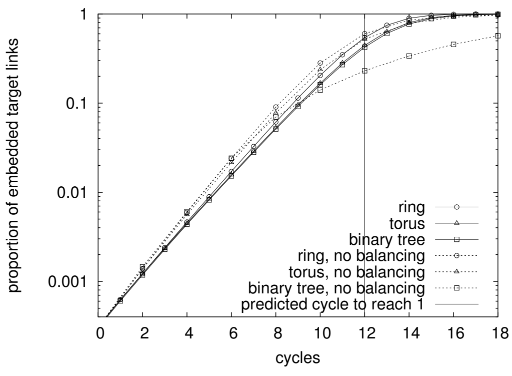
$$p_{a,b}(i) = 2p_{a,b}(i-1) = 2^i p_{a,b}(0) = \frac{c2^i}{N-1} \qquad (1)$$

as long as this value is smaller than one. We can naturally interpret the point when it reaches one as the predicted end of convergence. Expressing i from $c2^i/(N-1) < 1$ gives us

$$i < \log_2(N-1) - \log_2 c \qquad (2)$$

which is a logarithmic bound on the convergence time.

To test this model, we performed experiments with $N = 2^{17}$ and $c = 40$ (Figure 3). (Other network and view sizes are also tested in Section 6). Note that for these values of N and c, (2) results in a bound of 12. The experiments marked as "no balancing" represent the algorithm as shown in Figure 1. We



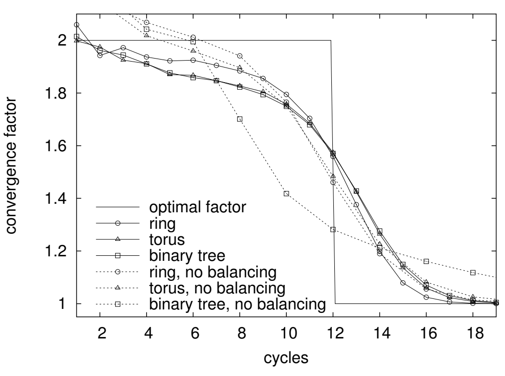


Figure 3. Comparison of convergence speed for network sizes $N = 2^{17}$ and $c = 40$, with and without balancing of the number of contacts per cycle per node. Averages from 20 runs are shown.

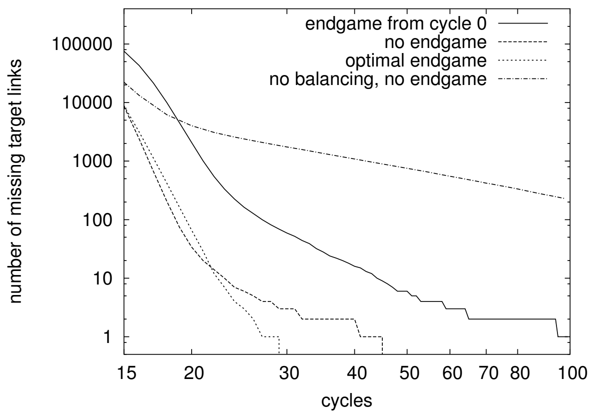


Figure 4. Comparison of convergence speed for the ring topology in the end phase for network sizes $N = 2^{17}$ and $c = 40$ for different versions of T-MAN. Averages of 20 runs are shown.

can clearly see that the proportion of embedded target links increases exponentially in the initial phase as predicted by (1). To illustrate how fast this increase really is, we also plot the *convergence factor*, which is the factor by which the proportion of embedded target links increases in a cycle. We can see that the factor reasonably approximates the predicted factor (which is 2 until the cycle where convergence is predicted). The case of the binary tree is an exception. This problem will be solved due to a modification we propose in Section 4.2.

4.2 The Endgame

The initial phase of convergence is exponential with a good approximation, and after this phase the vast majority of the target links are already embedded. For some applications it might be sufficient but in other cases it might be important to find *all* target links.

Due to the sources of error described above, a few nodes are left behind in convergence and find themselves at the wrong place when most of the others are properly linked. In this phase, unlike in the exponential initial phase, the convergence time heavily depends on the actual topology, because these late comers have to use the already converged structure to climb gradually to their position. For this reason, the worst case topology is the ring, which has a linearly growing diameter. Results for the ring topology with the same settings as above ($N = 2^{17}$, $c = 40$), are shown in Figure 4. (see Section 6 for more results with other ranking functions). It is clear that the basic, "no balancing" protocol performs poorly. We suggest two modifications to the original protocol to handle the end phase.

Balancing the number of contacts The relatively poor performance during the end phase is partly due to the unbalanced number of contacts the nodes make, as described above. To solve this problem, we allow nodes to count the total number of contacts (either passive or active) they have had and refuse connection if this number is larger or equal to the current cycle number, as derived locally from the elapsed time (recall that the length of a cycle is $T/2$ time units). We also allow nodes to search for neighbors from the view until one accepts a connection. This search mechanism is inexpensive as it involves only one bit ping messages. In Figure 3 we can see that this balancing mechanism solves the performance problem of the binary tree topology as well in the fast convergence phase. Also, in Figure 4 we can see that adding balancing improves performance significantly (see the curve marked as "no endgame").

Optimized endgame The other improvement comes from the observation that during the end phase, the situation is radically different than in the fast convergence phase. Here the remaining misplaced nodes climb on the already converged structure to their position. In this setting, it pays to aggressively favor those nodes from the view that are the closest, just like in a routing application. The endgame version of the protocol therefore changes method SELECTPEER so that it assigns exponentially decreasing sampling probabilities to the neighbors according to increasing rank. The predicted start of the end phase is given by (2). In the "optimal endgame" version we execute this modified protocol from this predicted point.

The optimal endgame combined with balancing clearly outperforms all other versions. To verify that this is not due to the fact that this version of SELECTPEER is better in general, we tested the protocol applying endgame since the beginning. As expected, this version is clearly inferior to the alternatives, so the endgame variant of SELECTPEER is indeed suitable only for the endgame.

Note that for other settings of N and c we get similar results, which are not presented due to space restrictions. From this point onwards, all of the remaining experiments are performed using both the contact balancing and optimized endgame extensions of the protocol.

5 Practical Considerations

5.1 Clustering and Sorting

So far we have looked at regular initial distributions of node profiles. It is easy to see that a clustered, irregular distribution can result in a clustered topology when the ranking function is defined by a distance metric over the space of profiles. Indeed, the topology construction problem as defined in Section 2.2 implies that the solution to the problem is a clustered network when nodes form clusters in which to a node from the cluster all nodes from the same cluster are closer than all the nodes outside it, *and* the size of these clusters is smaller than the view size parameter c. In applications when the goal is exactly to find those clusters, this behavior is desirable. On the other hand, when applied to sort the nodes, that is, to construct a one dimensional topology in which the nodes are ordered according to some ordering defined over their profile, we would like to ensure connectivity.

This can be achieved by the following direction dependent ranking. First, separate the set of nodes to be ranked into two groups: one that is to the left, and another that is to the right of the base node. Order these two sets according to the underlying desired ordering. Merge the ordered sets so that a node that had index i in any of the sets is assigned index $2i$ or $2i+1$ in the final ranking, choosing randomly between these two possibilities. The effect of direction dependent ranking is illustrated in the small example in Figure 5. Observe the clustering with the distance-based ranking and the perfect ordering with direction dependent ranking. Note that this direction dependent ranking can be easily extended to other problems, for example, creating a *connected* topology in two dimensions that reflects geographical proximity.

5.2 Bootstrapping

As mentioned before, the most important aspect of bootstrapping is the initialization of views with sufficiently random samples from the set of nodes. For this purpose, we propose using a protocol such as NEWSCAST, which is a gossip-based protocol for maintaining a connected, dynamically changing, random topology [5]. It takes care of membership management (joins and leaves) and it can serve as an underlying source of random nodes for any protocol that needs such nodes, like T-MAN. The basic idea of NEWSCAST is very similar to that of T-MAN, only ranking is based on the freshness of the node descriptors. In this

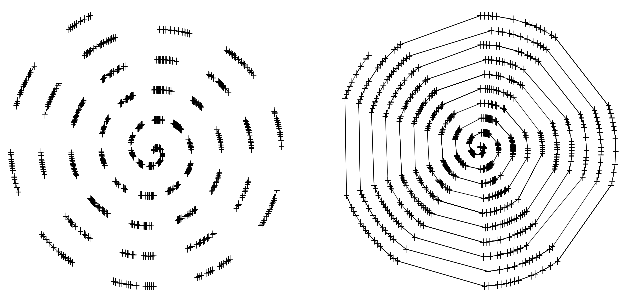

(a) distance based (b) direction dependent

Figure 5. Illustrative example of converged topologies obtained with distance-based and direction dependent ranking, with $N = 1000$, $c = 20$. Line is displayed as spiral for convenience. Only the closest 2 links are shown from each node.

manner, old information gets continuously removed from the system, and since there is no sense of convergence, no synchronization is necessary whatsoever. All experiments reported in Section 6 were performed using NEWSCAST for the initialization of the views, instead of a truly random sample.

5.3 Synchronization and Dynamism

In the context of data aggregation with gossip protocols, we have proposed solutions to handle synchronization and dynamism in [12]. In short, we believe that these solutions might work also for T-MAN, but this intuition remains to be validated.

Here, the key idea to handle dynamism is periodic automatic restarting, organizing the runs in so called *epochs*, which serve also as synchronization units. The start of the new epoch effectively propagates through the system similarly to an epidemic broadcast, only much faster because most of the nodes initiate it at the same time. The epoch has a fixed length after which a new one starts automatically.

Note also that several instances of the protocol can run at the same time which increases reliability and in our case also performance, in particular, the endgame can be shortened significantly because in different instances different nodes will find their place late so the union of more topologies will very likely be of much higher quality. Last but not least, union can be performed also in time if periodic restarting is applied, since during two consecutive epochs it is unlikely that the new set of nodes in the system is totally non-overlapping with the previous set due to dynamism.

6 Simulation Experiments

We have performed simulation experiments to examine the speed of convergence of the protocol both in the initial phase, which was predicted to be exponential, and in the end phase.

The experimental setup described in Section 3 applies. The version of T-MAN we have run utilizes both extensions introduced in Section 4.2: the contact balancing and the endgame peer selection techniques. Furthermore, the views were initialized using NEWSCAST, as described in Section 5.2. This is to ensure that the convergence results are not due to the uniform random sampling in the initial view (which is a rather strong assumption), but can also be obtained using realistic starting conditions.

The results are shown in Figure 6. To interpret the results on the convergence factor, let us give the predicted end of the initial exponential convergence phase for the parameter settings examined, using (2):

	$c = 20$	$c = 40$	$c = 80$
$N = 2^{14}$	10	9	8
$N = 2^{17}$	13	12	11
$N = 2^{20}$	16	15	14

In Figure 6 we can clearly observe exactly this trend of one cycle of shift with halving the view size and three cycles of shift when increasing the network size by a factor of 2^3. The only exceptions is $c = 20$ which is shifted slightly more than predicted. This is due to the fact that $c = 20$ is small enough so that

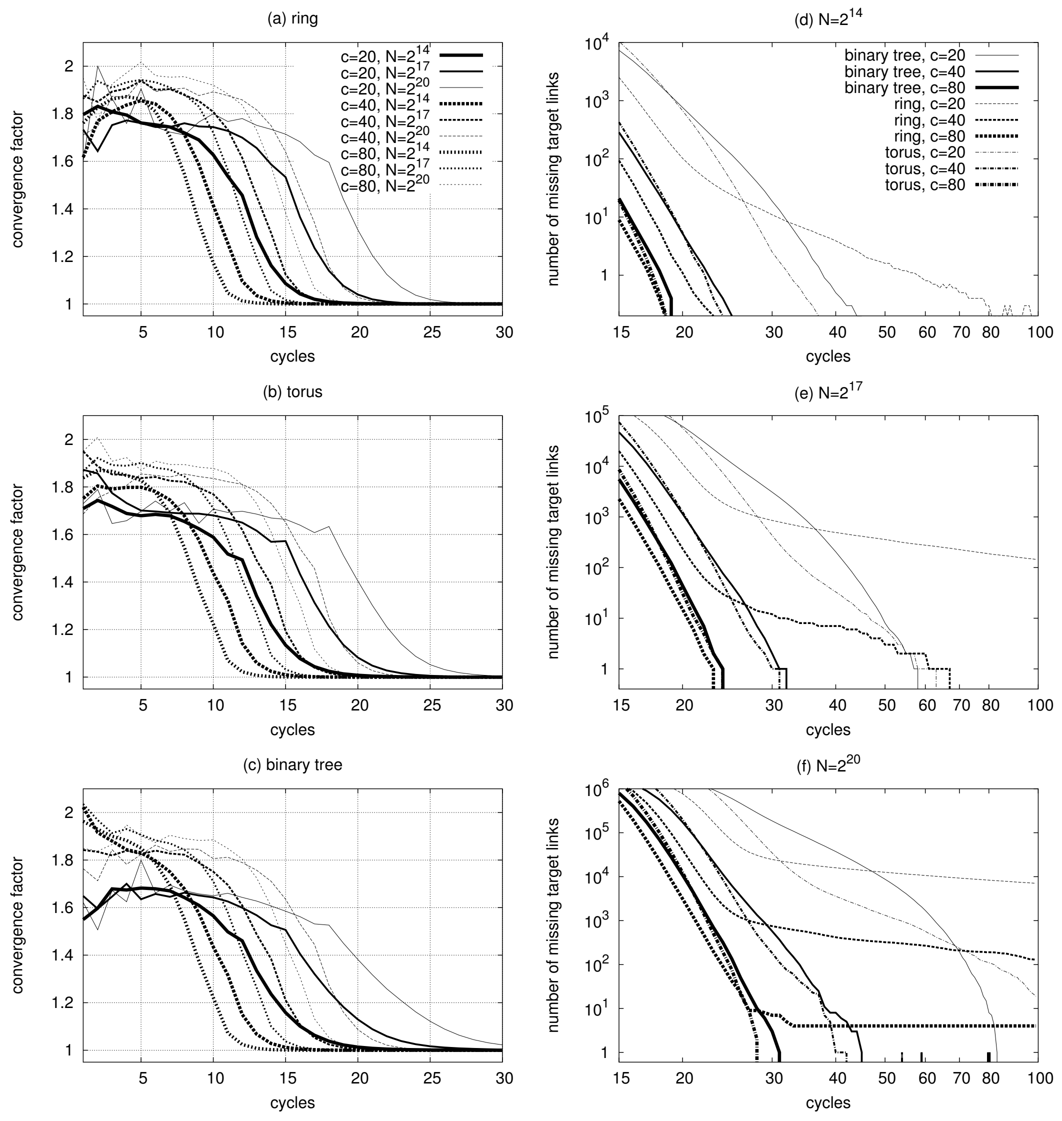


Figure 6. Comparison of convergence speed in the initial exponential phase and in the end phase for network sizes $N = 2^{14}, 2^{17}, 2^{20}$ and $c = 20, 40, 80$ for the ring, torus and binary tree topologies. The results displayed are averages of 50 and 10 runs for $N = 2^{14}$ and $N = 2^{17}$, respectively, and show a single run for the case $N = 2^{20}$.

sampling error can play a more important role in undermining the validity of the prediction. On the other hand, along the dimension of size with a fixed $c = 20$ the scaling is still according the the three cycle (and so logarithmic) shift as predicted. Remarkably, the above observations hold for all three of the topologies.

This is not so in the end phase, as expected, when the remaining small number of nodes use the already converged structure to find their position. Accordingly, in the binary tree topology, the end phase is also rapid, because the evolved structure allows for efficient routing, being low diameter. In fact, the convergence in the end phase of any misplaced node can be expected within a logarithmically increasing number of cycles, which is confirmed by the results. Similar arguments hold for the torus, only the convergence time there is not logarithmic but grows with the square root of the network size in the worst case. In both cases, we can observe fast convergence even for the smallest view size.

The case of the ring is different, because the target topology is of a large diameter, so the remaining few misplaced nodes can have a hard time reaching their destination. Until up to $N = 2^{17}$ the view size $c = 40$ is sufficient to reach full convergence within 70 cycles. In the largest network even with $c = 80$ we have not reached full convergence in the single run we performed within 100 cycles. Note however that at cycle 30 the number of missing links is already less then 10, out of the $2 \cdot 2^{20} = 2,097,152$ target links, and even for $c = 40$ we miss only 100 links in cycle 100. This level of precision will satisfy most applications. Still, it is true that the worst case waiting time in the end phase for perfection is linear. As mentioned before, the most promising techniques for future exploration to boost end phase performance are the concurrent execution of more instances of the protocol and to use past information when the restarting technique described in Section 5.3 is used.

7 Conclusions

In this paper we have introduced T-MAN, a gossip based protocol for the construction of a general class of topologies. We have shown experimentally and supported by an approximate analytical model that during an initial exponential convergence phase the protocol finds the vast majority of the desired links in logarithmic time and during the end phase the remaining links are also found, but here the convergence time depends on the target topology itself. Our results suggesting logarithmic convergence time independently of the target topology are promising, since they suggest scalability and a potentially wide range of applications.

We focused mainly on establishing the basic convergence properties of the protocol in a rather abstract manner. Although we have touched on many practical aspects suggesting solutions to problems like synchronization and dynamism, the ideas we mentioned need a thorough evaluation in the context of T-MAN. Obviously, specific applications like sorting or search also form a promising research direction we are currently pursuing.

References

[1] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database management. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, Aug. 1987. ACM.

[2] P. T. Eugster, R. Guerraoui, S. B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003.

[3] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. From epidemics to distributed computing. *IEEE Computer*. to appear.

[4] FastTrack: Wikipedia page. `http://en.wikipedia.org/wiki/FastTrack`.

[5] M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, Nov. 2003.

[6] M. Jelasity and A. Montresor. Epidemic-style proactive aggregation in large overlay networks. In *Proceedings of The 24th International Conference on Distributed Computing Systems (ICDCS 2004)*, pages 102–109, Tokyo, Japan, 2004. IEEE Computer Society.

[7] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 482–491. IEEE Computer Society, 2003.

[8] Y. Koren. Embedder. `http://www.research.att.com/~yehuda/index_programs.html`.

[9] C. Law and K.-Y. Siu. Distributed construction of random expander graphs. In *Proceedings of The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2003)*, San Francisco, California, USA, Apr. 2003.

[10] L. Massoulié, A.-M. Kermarrec, and A. J. Ganesh. Network awareness and failure resilience in self-organising overlays networks. In *Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS 2003)*, pages 47–55, Florence, Italy, 2003.

[11] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, 2001.

[12] A. Montresor, M. Jelasity, and O. Babaoglu. Robust aggregation protocols for large-scale overlay networks. Technical Report UBLCS-2003-16, University of Bologna, Department of Computer Science, Bologna, Italy, Dec. 2003. to appear in the Proceedings of DSN 2004.

[13] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter peer-to-peer networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 21(6):995–1002, Aug. 2003.

[14] PeerSim. http://peersim.sourceforge.net/.

[15] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In R. Guerraoui, editor, *Middleware 2001*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer-Verlag, 2001.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 149–160, San Diego, CA, 2001. ACM, ACM Press.

[17] R. Van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2), May 2003.

[18] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In N. Davies, K. Raymond, and J. Seitz, editors, *Middleware '98*, pages 55–70. Springer, 1998.

[19] S. Voulgaris and M. van Steen. An epidemic protocol for managing routing tables in very large peer-to-peer networks. In *Proceedings of the 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, (DSOM 2003)*, number 2867 in Lecture Notes in Computer Science. Springer, 2003.

[20] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering (ICDE 2003)*, Los Alamitos, CA, Mar. 2003. IEEE Computer Society Press.

[21] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, University of California, Berkeley, Apr. 2001.