User-Perceived Performance of the NICE Application Layer Multicast Protocol in Large and Highly Dynamic Groups*

Christian Hübsch, Christoph P. Mayer, and Oliver P. Waldhorst

Institute of Telematics, Karlsruhe Institute of Technology (KIT),
76128 Karlsruhe, Germany
{huebsch,mayer,waldhorst}@kit.edu

Abstract. The presentation of a landmark paper by Chu et al. at SIGMETRICS 2000 introduced application layer multicast (ALM) as completely new area of network research. Many researchers have since proposed ALM protocols, and have shown that these protocols only put a small burden on the network in terms of link-stress and -stretch. However, since the network is typically not a bottleneck, user acceptance remains the limiting factor for the deployment of ALM. In this paper we present an in-depth study of the user-perceived performance of the NICE ALM protocol. We use the OverSim simulation framework to evaluate delay experienced by a user and bandwidth consumption on the user's access link in large multicast groups and under aggressive churn models. Our major results are (1) latencies grow moderate with increasing number of nodes as clusters get optimized, (2) join delays get optimized over time, and (3) despite being a tree-dissemination protocol NICE handles churn surprisingly well when adjusting heartbeat intervals accordingly. We conclude that NICE comes up to the user's expectations even for large groups and under high churn.

1 Introduction

IP multicast is a technology with significant maturity that has been developed for several decades. Nevertheless it lacks global deployment for reasons that are manifold. They include, e. g., that IP multicast was designed without a specific commercial use-case in mind, leading to insufficient support for address allocation, group management, authorization, protection against attacks, and network management [7]. Nevertheless, many applications demand for multicast communication, and the demand is constantly growing in face of current trends towards

* This work was partially funded as part of the *Spontaneous Virtual Networks* (*SpoVNet*) project by the Landesstiftung Baden-Württemberg within the BW-FIT program and as part of the Young Investigator Group *Controlling Heterogeneous and Dynamic Mobile Grid and Peer-to-Peer Systems* (*CoMoGriP*) by the *Concept for the Future* of Karlsruhe Institute of Technology (*KIT*) within the framework of the German Excellence Initiative.

B. Müller-Clostermann et al. (Eds.): MMB & DFT 2010, LNCS 5987, pp. 62–77, 2010.

video and TV transmission. A solution to this hassle was presented by [6], which proposed to implement multicast distribution inside the end systems rather than in the network itself, creating an completely new area of research. Many protocols following this paradigm, denoted as end-system or *application layer multicast* (*ALM*), were developed. At the same time, it has been shown that the burden that ALM puts on the network core is small. In particular, *link stress*, i. e., transmitting the same multicast message across a physical link, and *link stretch*, i. e., extending the length of a delivery path compared to a unicast transmission, are bounded and can be easily handled by todays over-provisioned provider networks. Thus, from a provider's point of view ALM is an appealing alternative to IP multicast.

However, pushing multicast away from the network core towards the end systems also shifts the acceptance problem from the providers towards the users. To this end, users must experience a sufficient performance in order to utilize an ALM solution on a large scale. The user-perceived performance is determined by three major factors: First, the user must be able to join an ALM multicast group sufficiently fast, which is important, e. g., when zapping around between multiple TV channels. Second, he should experience only a minimum delay, which is important, e. g., for following a live broadcast or even more for interactive applications like attending a video conference. Third, the bandwidth consumed by the ALM protocol should be low, which is important, e. g., when using a computer for other things than following a multicast transmission. Furthermore, ALM must show the flexibility to be employed in different application scenarios, where two scenarios are of particular interest: The first is broadcasting popular events, which will on the one hand lead to multicast groups of significant size but on the other hand of high stability, since many people will follow the transmission for the entire duration. The second scenario is receiving a multicast transmission along the way, which will lead to fewer, but rather instable users. Examples for the second scenario include, again, zapping between TV stations.

In this paper, we present an in-depth evaluation study of the NICE application layer multicast protocol that significantly extends the results presented in [1,2,3]. In particular, we focus on the *user-perceived* performance of the protocol instead of the performance from the network perspective. The main work on NICE [1,2,3] focused on underlay behavior in terms of stretch and stress. Therewith, the authors have taken a network-view, whereas our work evaluates NICE from the end-user perspective. Similarly to the NICE studies, the work of Tang et al. focused on hop-counts in NICE using different extensions for underlay-awareness [11]. In terms of network dynamics the authors of NICE evaluated the behavior of NICE using a bulk-churn model [3]. Our work uses realistic—but aggressive—churn models to analyze NICE under real-world churn models. We base our churn models on recent work by Stutzbach et al. that analyzed churn-behavior of real-world P2P systems [10]. We do not cover extensions developed to explicitly provide robustness like [5], but rather analyze the robustness of the original NICE towards churn. In summary, we answer questions like: How long does it take until I can view a multicast transmission? What is the transmission

delay I have to expect? How much traffic is generated on my dial-up link? How do other people constantly joining and leaving the system affect my transmission quality?

To answer these questions, we conduct experiments using the overlay simulation framework *OverSim* [4] that allows for large-scale simulations and use of different churn models.

Our main findings are: (1) increasing the number of nodes by a factor of 16 increases latencies by only 31%, (2) join delays get optimized over time, and (3) under a heavy churn configuration data success rates of 98.8% can be achieved. From these results, we conclude that NICE is well suited to provide user-perceived performance in large-scale and dynamic environments.

The remainder of this paper is structured as follows: To make the paper self-contained, we give a description of the NICE protocol in Section 2. We also state design decisions in our implementation concerning issues that are not clear in the original proposal [2]. In Section 3 we describe our simulation methodology that is used within the following Sections 4 and 5. In Section 4 we look at NICE's performance in scenarios with higher numbers of nodes. We evaluate the protocol's performance in face of churn in Section 5. Finally, concluding remarks are given in Section 6.

2 NICE

The NICE protocol [1,2,3] is an early ALM approach that implements an unstructured overlay (i. e. a node's position in the overlay is not fixed). It explicitly aims at scalability by establishing a cluster hierarchy among participating member nodes. We first give a general protocol description in Section 2.1 before detailing on specific implementational aspects in Section 2.2.

2.1 Basic Protocol

NICE divides all participating nodes into a set of clusters. In each cluster, a cluster-leader is determined that is responsible for maintenance and refinement in that cluster. Furthermore, all cluster-leaders themselves form a new set of logical clusters in a higher layer, exchanging protocol data. Respective cluster-leaders are determined from one layer for the next higher layer. This process is iteratively repeated until a single cluster-leader in the topmost cluster is left, resulting in a layered hierarchy of clusters (compare Figure 1). Protocol traffic is mainly exchanged between nodes residing in the same cluster, leading to good scalability.

Each cluster holds between k and $(\alpha k - 1)$ nodes, α and k being protocol parameters. In case the number of nodes in a cluster exceeds the upper bound the cluster is split into two clusters of equal size. If the lower bound is undercut, the cluster is merged with a nearby cluster. Clusters are formed on the basis of a 'distance' evaluation between nodes, where distance is basically given by network latency. Cluster-leader election is accomplished by determining the

graph-theoretic center of the cluster and choosing the node closest to that point. Nodes in the same cluster periodically exchange heartbeat messages to indicate their liveliness and report measurements of mutual distance to other nodes in that cluster. Cluster-leaders decide on splitting and merging of clusters as they are aware of the current cluster size and all distances between nodes inside their cluster.

The objective of NICE is to scalably maintain the hierarchy as new nodes join and existing nodes depart. Therefore, the following invariants are maintained:

1. At every layer, nodes are partitioned into clusters of size between k and $(\alpha k - 1)$.
2. All nodes belong to a L_0 cluster and each node belongs only to one single cluster at any layer.
3. Cluster-leaders are the center of their respective cluster and form the immediate higher layer.

For bootstrapping, the joining node queries a *Rendezvous Point* (RP) for the set of nodes that reside in the highest cluster. The node then queries the nearest of these nodes for the set of the next lower layer, iteratively repeating this process until the lowest layer L_0 of the hierarchy is reached. As soon as this nearest cluster is determined, the node requests from the L_0 cluster's leader to join and finally becomes part of the cluster. Graceful or ungraceful leaving of nodes is either detected by explicit protocol messages or through missing heartbeat messages.

A node intending to send out multicast data sends its data to all nodes in all clusters it currently resides in. A node receiving data from inside its cluster forwards the packet to clusters it is part of except the cluster it received the data from. This leads to each participant implicitly employing a dissemination tree to all other nodes in the structure.

An exemplary NICE structure, consisting of three hierarchical layers (L_0–L_2) is shown in Figure 1. Here, Layer L_0 holds five clusters containing five member nodes each. The four phases (a) to (d) demonstrate the steps in data dissemination for a given initial sender. Members forwarding the data in each

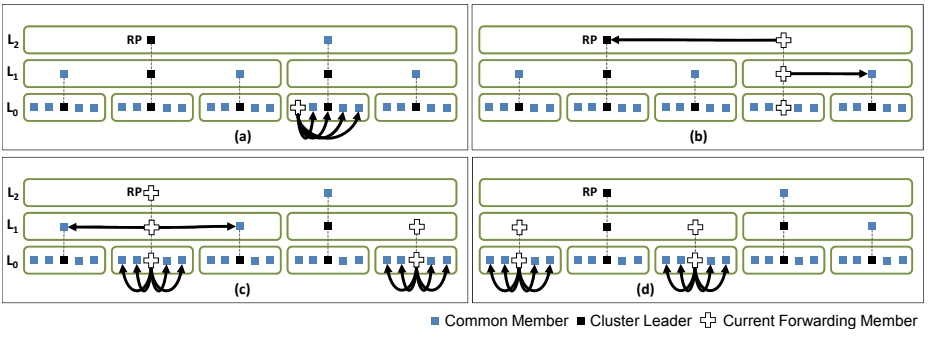

Fig. 1. Layered hierarchical NICE structure

step are symbolized by crosses. Also in this example, the cluster-leader of the highest cluster in the hierarchy constitutes the RP.

2.2 Design Decisions

We implemented NICE in the open-source overlay simulation framework OverSim [4] based on the technical descriptions given in [3]. We will now detail on relevant design decisions that are not specified by the protocol itself to ease the understanding of the evaluations given in this paper.

Heartbeats and Distance Evaluation. Heartbeat messages are sent to direct cluster neighbors participating in any cluster the sender resides in. The heartbeat interval HBI triggers periodic heartbeat messages. We use heartbeats for protocol information exchange and simultaneously for the evaluation of mutual distances between nodes (cf. Figure 2a). Distances in NICE are evaluated through round-trip time measurements between heartbeat-exchanging nodes. As those evaluations are prone to variance we use an exponentially weighted moving average to smooth distance measurements over time. Also, to avoid intersecting heartbeats that would tamper with measurements, we use dedicated heartbeat sequencing (cf. Figure 2b), helping to avoid error-prone distance evaluations when out-of-order packet receptions occur.

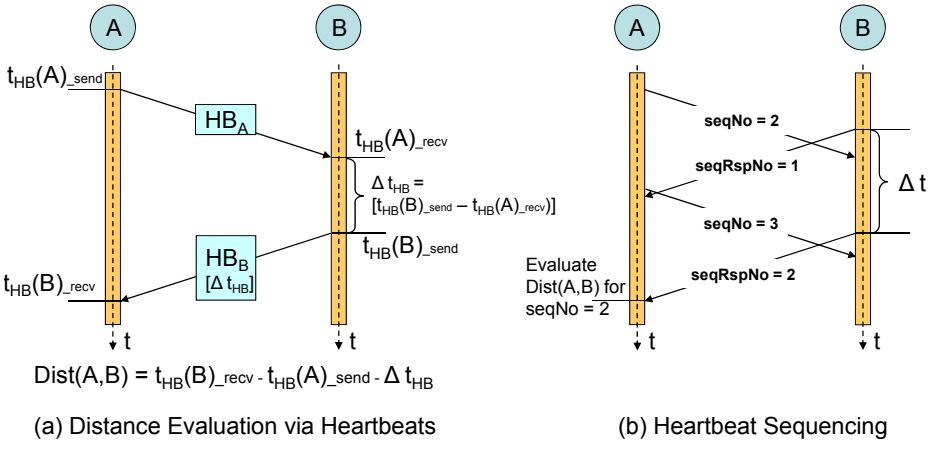

Fig. 2. Distance Evaluation

Bootstrap and Join Phase. A joining node queries the RP for all nodes of the highest cluster and starts sending probe packets for distance estimation. The joining node descends towards its nearest node, iteratively repeating this process until layer L_0 is reached. In our implementation, we use an in-band RP, always being the cluster-leader residing in the highest current hierarchy cluster.

Maintenance and Refinement. After a successful join phase nodes start maintaining the overlay structure with respect to their point of view. This is

realized by periodic heartbeats to all cluster neighbors to validate liveliness and ensure protocol invariants (cf. Section 2.1).

Heartbeat messages are used for propagation of distance measurements. This way nodes are able to make decisions autonomously. A node being cluster-leader in cluster C_i in layer L_i emits a special leader-heartbeat in this cluster, holding contents of a normal heartbeat as well as information about members in the direct super-cluster C_{i+1} in layer L_{i+1}. This information is used by neighbors in C_i to find better cluster-leaders in C_{i+1}. A node in the super-cluster is considered better if the distance to this node is at least min_SC_Dist percent smaller than towards the current cluster-leader (min_SC_Dist being a protocol parameter). Should a closer cluster-leader B in C_{i+1} be found for a querying node A, the latter will change its cluster membership to the cluster in layer L_i that B is leader of. Furthermore, neighbors in C_i use the neighbor information in the leader-heartbeat to update their view of the current cluster memberships, i. e. they add new nodes or delete nodes that are no longer part of the cluster.

If a cluster-leader detects a violation of the cluster size upper bound it determines the resulting two new clusters as follows: Let C_i be the cluster to split, C_j, C_k the resulting sub-clusters. For all possible combinations $\{C_j, C_k | C_j \subseteq C_i \backslash C_k \wedge C_k \subseteq C_i \backslash C_j\}$ determine the particular resulting cluster-leader and check the maximum distance from each such leader inside the respective cluster. The number of combinations is bounded by the clustersize parameter k. If C_i holds n nodes, $n \leq (\alpha k)$, this leads to $\frac{n!}{2\left(\frac{n}{2}!\right)^2}$ combinations to test. As soon as an appropriate cluster-split-set has been determined, the cluster-leader signals the change throughout the specific cluster and all involved higher-layer clusters.

Should a cluster hold less than k nodes, it will be merged with one of its neighboring clusters on the same layer. The leader of the specific cluster C_i in layer L_i is also part of cluster C_{i+1} in layer L_{i+1}, knowing its distances to the nodes residing in the latter. With this information, the leader is able to determine the nearest node of cluster C_{i+1}, being the candidate to merge cluster C_i with, and initiate the merge operation.

As part of the periodic refinement process, cluster nodes decide if the current cluster-leader remains optimal. This is accomplished by finding the node with the smallest maximum distance to all other members in this cluster, based on each nodes local distance knowledge. To avoid fluctuations we use a lower bound min_CL_Dist which has to be exceeded in order to trigger a cluster-leader change.

Protocol Recovery. Changes in the network or the NICE hierarchy can lead to temporary soft-state inconsistencies between nodes. In case of severe hierarchy inconsistencies like partitioning or failure, nodes can decide to reconnect to the structure. It can further occur that more than one cluster-leader becomes responsible for one node. This can happen due to temporary duplicate leaderships in clusters, packet loss during leader transfer, or other inaccurate negotiation procedures. In our implementation duplicate leaderships are detected by heartbeat messages. If a node A after reception of heartbeat H_1 at time t_1 from leader B receives a heartbeat H_2 from leader C at time t_2 with $t_2 - t_1 <$ HBI it checks

if the predecessing heartbeat before H_1 was sent from leader B. This will with high probability indicate a child relationship to both B and C. Node A will then resolve the situation with a proactive cluster leave sent to leader C.

In addition to nodes detecting duplicate leaders, cluster-leaders also have to be able to detect mutual duplicate leaderships inside a cluster. Duplicate cluster-leaderships appear if one node decides to be new cluster-leader while the old leader did not take this decision—or in some cases never will due to different distance knowledge. Such situations are detected if a cluster-leader receives a leader-heartbeat message in the same cluster he is leader of.

3 Evaluation Methodology

In this Section, we detail on the simulation environment and the setup of our simulations. Furthermore, we discuss relevant performance measures that are used in the remainder of this paper.

3.1 Simulation Environment

Our experiments are conducted using the peer-to-peer simulation framework *OverSim* [4]. OverSim provides a flexible environment for simulation of structured and unstructured overlay networks with focus on scalability of the simulation models with respect to the number of simulated nodes as well as re-use of modules implementing overlay functionality. The core part of OverSim comprises various network models, that each model the underlying network with a different level of detail, and thus, complexity of the simulation model and simulation runtime. The network model of our choice is OverSim's *SimpleUnderlay* that is frequently employed for performance evaluation from the end-system perspective. This network model abstracts from network and transport mechanisms and arranges nodes in Euclidean space. The Euclidean distance of two nodes determines the basic network latency between them. A random jitter between 0%–5% is added to this latency for each packet transmission. Note that this network model does not capture packet losses, i. e., every packet that is sent t is received by the destination. This behavior is consistent with the behavior in other ALM simulation studies, e. g. [2]. We provide details on the setup of our OverSim simulations in the next section.

3.2 Simulation Setup

We have implemented the NICE protocol as described in Section 2 as an OverSim application that is executed by OverSim's simulation models. In our simulations we analyze up to 8 000 NICE nodes, where the number of nodes differs with the experiments. Nodes are arranged randomly in a two dimensional field of size [150,150], i. e., the maximum delay experienced for a transmission between two nodes is $\approx 212\,ms$ with addition of the random jitter. Based on the parameter terms introduced in Section 2.2 we use the protocol parameter values given in

Table 1 (unless stated different in the specific experiments). Furthermore, we use a simulation setup that is given by the simulation parameters also shown in Table 1.

Our simulation experiment can be subdivided into two phases. In the *initial phase* after the start of the simulation, the NICE hierarchy is incrementally constructed. That is, one new node joins the network every second, until the anticipated number of nodes is reached. We choose this approach to avoid difficult effects that could appear in the initial phase and that are not subject to our evaluations. After the last node has joined we employ a backoff time of 60 seconds to stabilize the hierarchy. The initial phase is followed by the *data exchange phase*. In this phase, a given node, fixed but chosen uniformly at random from the set of all nodes, sends a multicast packet every 5 seconds for evaluation of scalability in Section 4, and every 1 seconds for evaluation of churn in Section 5. Note that although the resulting data rate is very low, this is sufficient to compute the performance measures of interest as described in Section 3.4. After 10 minutes of data exchange, we again employ a backoff of 60 seconds before finishing the simulation run.

Note that depending on the considered application scenario nodes may be either stable during the data exchange phase or join/leave the NICE hierarchy at arbitrary times due to churn. Since we consider both types of application scenarios, we describe our model of dynamic node behavior in the next section.

Table 1. Protocol and simulation parameters

NICE-specific		Simulation-specific	
Parameter	Value	Parameter	Value
α, k	3	Number of nodes	500–8 000
HBI	{1,5,10} s	Offset after last join	60 s
Maintenance Interval	3.3 s	Measurement phase	300 s
Peer Timeout	2 HBI	Joins	∼every 1 s
Query Timeout	2 s	Data Interval	1 s (churn), 5 s (scalability)
Structure Timeout	3 HBI	Field Size	[150,150]
min_CL_Dist	30%	Simulation Time	2 000 s
min_SC_Dist	30%		

3.3 Churn Model

Churn is the process of nodes joining and leaving the overlay structure. As joins and leaves trigger adaptation and therewith restructuring of the overlay, they can cause packet loss due to inconsistencies, or partitioning. Resilience to churn is conventionally achieved through redundant links in the overlay structure, resulting in higher cost [8]. Furthermore, dedicated mechanisms for overlay robustness have been developed to cope with high churn [9].

To study the performance of NICE under heavy churn we have to define appropriate churn models for our simulations. Several churn models have been described in the literature, which use either Poisson, Random, Exponential, or

Weibull distributions to model a node's session length, i. e., its dwell time in the hierarchy. A recent study of Stutzbach et al. [10] analyzed different real-world networks (Gnutella, Kad, BitTorrent) and identified that (1) session length distribution is quite similar over different networks, and (2) that the session length distributed is best modeled through a Weibull distribution. Prior work on NICE evaluated the protocol under bulk churn where groups of nodes collectively join and leave the overlay simultaneously [2]. Opposed to [2], we use individual churn following the Weibull distribution in our simulations according to the Weibull PDF defined as follows:

$$f(x;\lambda,\mu) = \begin{cases} \frac{\mu}{\lambda}\left(\frac{x}{\lambda}\right)^{\mu-1} e^{-(x/\lambda)^{\mu}} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{1}$$

We use shape parameter $\mu = 0.5$, as a compromise of shape values identified in the work of Stutzbach. As the scale parameter λ varies greatly depending on the observed system, we perform simulations with different λ values to achieve different mean lifetimes of the nodes, i. e., different degrees of churn. All parameter values of the churn model together with the corresponding mean (in minutes and seconds) and variance of the session length are shown in Table 2. As our goal is to find the limits of robustness for NICE, our scaling values result in much smaller mean lifetimes than the values presented by Stutzbach et al., which are also shown in the table.

For churn simulations we use a mean of 128 nodes and single-source multicast. The simulation time is 3 600 s, subdivided as follows: Again, we have an initial phase where 128 are created in the first 128 s, one node per second. We start the churn model together with the data transmission face at simulation time 200 s and end it at simulation time 3 540 s. The source node of the transmission is selected as described above and is not subject to churn. Finally, the simulation ends at 3600 s. We evaluate different churn rates as detailed in Table 2. Finally we perform one simulation with no churn as reference model.

3.4 Performance Measures

In our simulations, we consider four measures to capture user-perceived performance of the ALM protocol:

Join Delay: This is the delay for integrating a new node into the NICE hierarchy. Since it is the time a user has to wait until he can receive a multicast

Table 2. Weibull parameters and properties used for churn simulation in our work and real-world observations from Stutzbach et al.

	Our Work							Stutzbach et al.		
μ	0.50							0.34	0.38	0.59
λ	0.83	2.50	5.00	7.50	10.00	12.5	15	21.30	42.40	41.90
mean [minutes]	1.66	5	10	15	20	25	30	117.25	163.38	64.46
mean [seconds]	100	300	600	900	1 200	1 500	1 800	7 035	9 802	3 867
variance [minutes]	14	125	500	1 125	2 000	3 125	4 500	241 986	313 390	13 395

transmission, we consider this measure of particular interest from the user's perspective. For a given node, we measure this delay by the time between contacting the rendezvous point and finally integrating the node in a cluster at layer L_0 following the procedure described in Section 2.

Data and Heartbeat Latency: Data latency is the time required to transmit a data packet from the multicast source to a given destination node. This latency is of particular interest for users following a real-time transmission. It can be measured by setting timestamps when sending and receiving a multicast packet, respectively. Furthermore, the *hopcount*, i. e., the number of hops in the NICE hierarchy that must be traversed to deliver a multicast packet, can be computed using a field in the packet header. We will show that data latency depends to some extent on the latency for transmissions inside a cluster. Thus, we will also measure heartbeat latency, which is the delay experienced by heartbeat messages as described in Section 2.2.

Overhead: Maintaining the NICE hierarchy comes at a cost, which is quantified by the overhead for sending control messages. Overhead is relevant from the user's perspective since it must be transmitted over the user's access link. We measure the overhead for a given node by summing up the sizes of all control messages it generates according to the protocol description in Section 2. We assume addresses to be 32 bit and that e. g. heartbeat messages hold all known cluster members together with their related distance evaluations, each stored also in a 32 bit value.

Successfully Delivered Packets: Although OverSim's simple underlay does not consider packet losses, a multicast data packets may be lost due to structural problems in the NICE hierarchy, in particular under heavy churn. Since data packet losses directly affect the transmission quality, we consider them of particular interest from the user's perspective. To compute the fraction of successfully delivered packets is non trivial, since it is not clear how to count a node that is part of the hierarchy when a data packet is send by the source, but leaves the hierarchy before the packet is able to reach it. Thus, we measure successfully delivered packets only for those nodes that are a part of the hierarchy when a packet is sent and do not leave the hierarchy until transmission of the next packet.

In the following sections, we use the performance measures defined here to evaluate the performance of the NICE protocol.

4 Protocol Scalability

This section analyzes user-perceived performance of the NICE protocol from a scalability perspective. Consistent with [2], it focuses on the performance during initial construction and during stable operation of the multicast structure, while the performance under churn is considered in Section 5. Opposed to [2], we analyze large scenarios (i. e., up to 8000 nodes) and focus on user-perceived rather then network-centric performance.

In a first experiment, we analyze the join delay as defined in Section 3.4. We plot join delay as a function of time in Figure 3 for different sizes N of the multicast group. Recall that in the experiment one node per second joins the hierarchy. In the figure, one line is drawn for each node's join delay. Join delay depends on the number of layers in the hierarchy, since a node starts to join at the rendezvous point in the highest layer L_k and descends through the hierarchy. In order to relate join delay and hierarchy depth, the figure also depicts the current number of layers in the structure.

The figure shows that for $N < 1500$ the resulting hierarchy has four layers, while it has five layers for larger group sizes. Indeed, we find that the join latency—as expected—depends on the current number of layers with the most significant increase when raising the hierarchy depth from four to five layers. As an interesting fact, the figure indicates that for a hierarchy with Layers $L_0, \ldots, L_k$ the join delay is highest directly after Layer L_k has been established. Afterwards, it decreases constantly until Layer L_{k+1} is added to the hierarchy. This is due to the fact that a new established layer leads to few nodes in the higher layer clusters, as illustrated in Figure 4. This figure plots the mean number of nodes in clusters of each layer computed once a second. The figure shows that the cluster size in the lower layers stays quite constant over time. In contrast higher layer clusters are incrementally filled with nodes, confirming the claim made above. For understanding the decrease in join latency depicted in Figure 3, recall that a node must perform a distance estimation for one cluster on each layer $L_k, \ldots L_1$ until it reaches the lowest layer L_0. That means basically waiting for a response by the cluster member that is closest in each cluster with respect to latency. Since fewer nodes within a cluster reduces the probability for having a nearby node—as we will illustrate later—this increases join delay due to an higher delay for distance probing in the higher layers. However, the join delays are significantly below two seconds with an average of 0.51 seconds. We conclude from Figures 3 and 4 that even for large multicast groups NICE provides a reasonable join delay for the users.

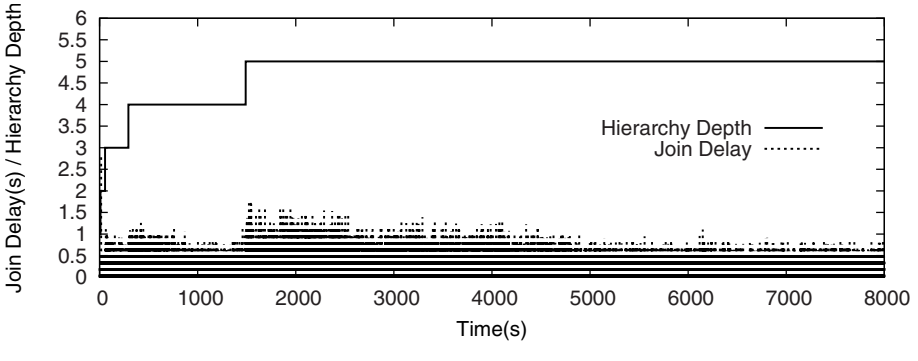

Fig. 3. Per-node delays for hierarchy joins and hierarchy depth, 8000 Nodes

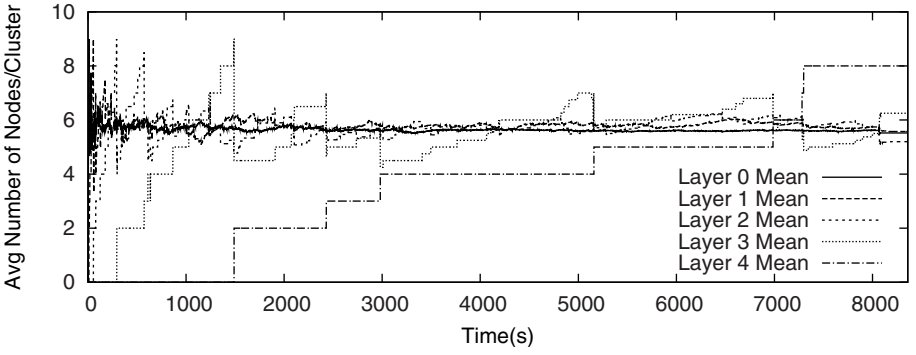

Fig. 4. Average Cluster Sizes per Layer, 8000 Nodes

In the next experiment, we focus on data latency as defined in Section 3.4. We will restrict us to the latency achieved when the structure has stabilized, as results for a structure with nodes entering and leaving are be shown in Section 5. Figure 5 plots the cumulative distribution function (CDF) of the data latency. One would expect data latency to increase with an increasing size N of the multicast group. However, we find that it is quite stable regardless of group size, e. g. a growth in number of nodes from 500 to 8 000 increases mean latencies by 31% from 173.6 ms to 228 ms. To gain deeper insight into this behavior, we plot the CDF of one hop heartbeat latency measured inside each cluster as well as the hopcount distribution for multicast packets in Figures 6 and 7, respectively. Confirming our claim made earlier, Figure 6 indicates that intra cluster latency decreases significantly with an increasing number of nodes, since the probability for clustering nearby nodes increases. Thus, every hop a data packet must traverse for delivery takes less time. This fact compensates the moderate increase in path length that results from increasing the depth of the hierarchy, which is illustrated in Figure 7. We conclude from Figures 6 and 7 that NICE is highly scalable with respect to data latency.

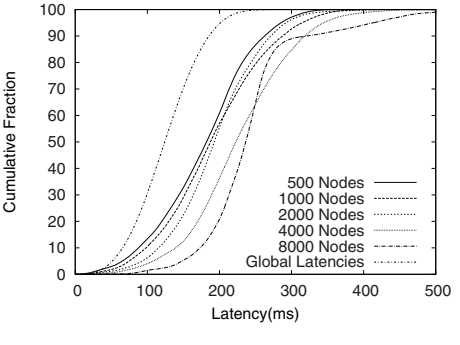

Fig. 5. Latencies for data dissemination

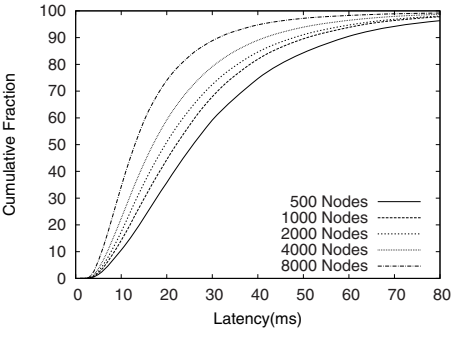

Fig. 6. Intra-cluster Latencies

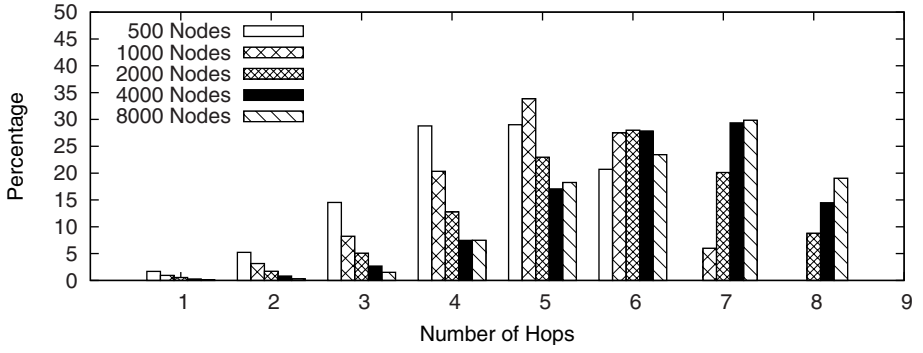

Fig. 7. Distribution of hopcount for delivering multicast packets

The last performance measure we consider is the overhead as defined in Section 3.4. We plot overhead as a function of time in Figure 8. Since control overhead may depend on the position of a node in the cluster hierarchy, we subdivide the overhead by the layers. That is, we plot mean overhead for all nodes that have the highest layer cluster membership on the same layer. The traffic is both aggregated by 1 s and 10 s respectively. As expected, the figure shows that the overhead for nodes in the highest layer cluster and especially for the rendezvous point is significantly higher that for nodes that are only members of lower layer clusters, since heartbeat messages etc. must be send for each cluster. As soon as the depth of the hierarchy is increased from k to $k+1$, the traffic for the nodes that are member of cluster k but not of cluster $k+1$ stabilizes. The fact that each additional cluster membership adds a constant overhead for cluster maintenance leads to a linear grow in mean overheads with the number of clusters. For the considered protocol configuration, each additional cluster membership increases overhead by about 1–3 kbit/s. Given a logarithmic growth of hierarchy depth, we conclude from Figure 8 that NICE is scalable with respect to control overhead. Nevertheless, more powerful nodes with high bandwidth connections

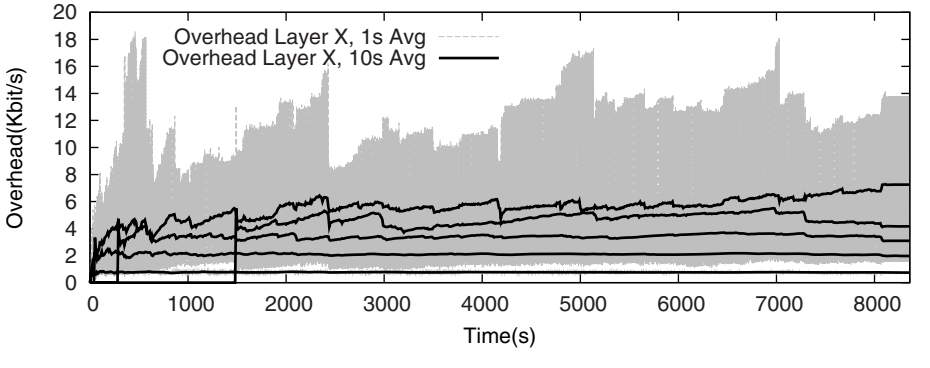

Fig. 8. Average control overhead per node, 8000 nodes

seem more suited for membership in higher layer clusters, leading to the conclusion that the choice of cluster leaders should not only be determined by average latency. While we focus on user-perceived results in this paper, the experiments also pose questions concerning reciprocal effects between the cluster parameters. We are currently investigation issues like efficient parameter selection in specific scenarios in the context of another work.

The experiments shown in this section indicate that the NICE protocol from a user's perspective performs well even for large but stable groups. However, depending on the application scenario, coping with large group sizes might be less important than providing stability of the multicast structure under high node churn. We will enlighten this aspect in the next section.

5 Churn

In this section, we evaluate the user-perceived performance of the NICE protocol under the realistic churn model introduced in Section 3.3. Recall that as our goal is to find the limits of robustness for NICE, our scaling values result in much smaller mean lifetimes than the values presented by Stutzbach. Furthermore, a mean number of 128 nodes is considered in this scenario, being rather instable as a result of churn. We believe this node number is sufficient to get an impression of NICE's abilities to handle fluctuations during multicast transmissions.

In a first experiment, we analyze the impact of churn on the structure of the NICE hierarchy. Similar to Figure 3, we plot both the join delay and the hierarchy depth as a function of time in Figure 9. The figure additionally shows the number of group members. We find that during the data transmission phase the hierarchy depth may alternate between two and three layers due to churn. However, adding or removing layers does only delay the join operation of a few nodes. In fact, the join delay of most nodes is with an average of 470 ms almost unaffected by churn. In an experiment not shown here due to space limitations we find that data latency is not affected by churn, too, although multicast packets may be lost as we show below. We conclude from Figure 9 that NICE performs well under churn from the perspective of user-perceived latencies.

Since packet loss certainly affects user-perceived performance, we analyze the fraction of successfully delivered multicast packets as defined in Section 3.4 in a last experiment. Using variations in heartbeat intervals (HBI) defined in Section 2 and using the churn rates given in Section 3.3 we evaluate the packet loss ratio to find the robustness limits of the NICE structure. Figure 10 plots the probability for successful delivery of a multicast packet as a function of mean node lifetime together with the involved standard deviations. We alternate the heartbeat-interval HBI between values 1s, 5s, and 10s. Note that without churn (not shown in the figure), NICE delivers close almost 100% of the multicast packets successfully, since the NICE hierarchy is stable and packet losses on lower layers are not considered by the underlay model. However, even under moderate churn, a value of HBI larger than 1s implies significant packet loss, such that 10% and more of the packets are not delivered. Only an aggressive HBI value

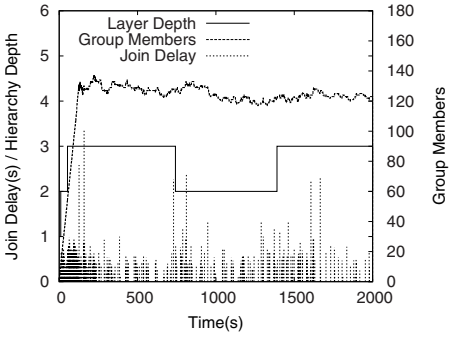

Fig. 9. Structure under churn

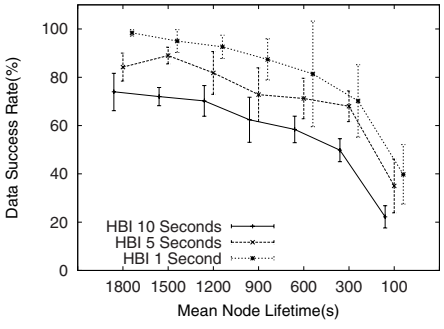

Fig. 10. Packet success rate under churn

of 1s can compensate the churn up to a certain extend. Nevertheless, for node lifetimes smaller that 900s even such aggressive parametrization of NICE fails to successfully deliver more than 90% of the packets. We conclude from Figure 10 that high churn either requires aggressive efforts to maintain the NICE hierarchy or some resilience mechanisms as introduced, e. g.in [5].

6 Conclusions

In this work we evaluated the NICE application layer multicast protocol from a user perspective. Our focus is on scalability—using large groups of up to 8 000 nodes—and the behavior under churn—using aggressive versions of realistic churn-models. By extensive simulations we achieve the following insights: (1) increasing the number of nodes by a factor of 16 increases latencies by only 31% in the considered scenario, (2) join delays get optimized over time, and (3) under a heavy churn configuration data success rates of 98.8% can be achieved. Despite not being designed for high churn, NICE can cope with such situations by using smaller heartbeat intervals. Furthermore, NICE also achieves good scalability for large groups.

References

1. Banerjee, S., Bhattacharjee, B.: Analysis of the NICE Application Layer Multicast Protocol. Technical Report UMIACS TR 2002-60 and CS-TR 4380, Department of Computer Science, University of Maryland, College Park, MD 20742, USA (June 2002)
2. Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable Application Layer Multicast. In: Proceedings of SIGCOMM, Pittsburgh, Pennsylvania, USA, August 2002, pp. 205–217 (2002)
3. Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable Application Layer Multicast. Technical Report UMIACS TR-2002-53 and CS-TR 4373, Department of Computer Science, University of Maryland, College Park, MD, USA (2002)

4. Baumgart, I., Heep, B., Krause, S.: OverSim: A Flexible Overlay Network Simulation Framework. In: Proceedings of 10th IEEE Global Internet Symposium (GI 2007) in conjunction with IEEE INFOCOM, Anchorage, Alaska, USA, May 2007, pp. 79–84 (2007)
5. Birrer, S., Bustamante, F.E.: Resilience in Overlay Multicast Protocols. In: Proceedings of the IEEE Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Monterey, California, USA, September 2006, pp. 363–372 (2006)
6. Chu, Y., Rao, S., Zhang, H.: A Case For End System Multicast. In: Proceedings of ACM SIGMETRICS, Santa Clara, CA, USA, June 2000, pp. 1–12 (2000)
7. Diot, C., Levine, B.N., Lyles, B., Kassem, H., Balensiefen, D.: Deployment Issues for the IP Multicast Service and Architecture. IEEE Network Magazine Special Issue on Multicasting 14(1), 78–88 (2000)
8. Li, J., Stribling, J., Morris, R., Kaashoek, M.F., Gil, T.M.: A Performance vs. Cost Framework for Evaluating DHT Design Tradeoffs under Churn. In: Proceedings of INFOCOM, Miami, Florida, USA, March 2005, pp. 225–236 (2005)
9. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling Churn in a DHT. In: Proceedings of the USENIX, Boston, MA, USA, June 2004, pp. 1–14 (2004)
10. Stutzbach, D., Rejaie, R.: Understanding Churn in Peer-to-peer Networks. In: Proceedings of Conference on Internet Measurement, Rio de Janeiro, Brazil, October 2006, pp. 189–202 (2006)
11. Tang, H., Janic, M., Zhou, X.: Hopcount in the NICE Application Layer Multicast Protocol. In: IEEE/SMC Multiconference on Computational Engineering in Systems Applications, Beijing, China, October 2006, pp. 1020–1026 (2006)