On Runtime Adaptation of Application-Layer Multicast Protocol Parameters

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

On Runtime Adaptation of Application-Layer Multicast Protocol Parameters

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

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

Reasonable choice of protocol parameters is crucial for the successful deployment of overlay networks fulfilling given service quality requirements in next generation networks. Unfortunately, changing network conditions, as well as changing application and user requirements may invalidate an initial parameter choice during the lifetime of an overlay. To this end, runtime adaption of protocol parameters seems to be a promising solution—however, it is not clear if protocol parameters can be adjusted dynamically at runtime in a distributed setting. In this paper, we show—using the NICE application layer multicast protocol as an example—that runtime adaptation of protocol parameters is indeed feasible. We propose an algorithm for adapting the NICE clustersize parameter k dynamically at runtime and discuss the impact on service quality. Our simulations show that runtime adaptation of NICE protocol parameters is promising for service improvement and that data latencies can be optimized by up to 25% without increasing the overhead significantly for most of the nodes.

1 Introduction

During the evolution towards next generation networks, the Internet architecture faces a number of limitations due to upcoming requirements like mobility, multi-homing, or multicast communication. E. g., a clean implementation of multicast communication within the Internet architecture has been proposed for years but never experienced widespread end-to-end deployment. One reason for the rare deployment of IP-based multicast is its inflexibility and requirement for provider trust. Therefore, deployment of overlay networks attracted great interest throughout the scientific community as a feasible approach for adding new services like multicast communication on top of the Internet architecture in an

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.

unintrusive and flexible way. A large number of protocols have since been proposed that enable the overlay-based deployment of new services like multicast, or distributed hash tables [4].

As services implemented by application-layer overlay networks typically provide worse service quality than native implementations in lower layers, careful parameterization and fine-tuning of the overlay protocol becomes essential. Unfortunately, runtime changes of network conditions, as well as application and user requirements can make a good parameterization turn worse during the lifetime of an overlay. This requires re-adjustment of overlay parameters during runtime to prevent degradation of service quality—at best without any service downtime.

In this paper we show by the example of the NICE application-layer multicast (ALM) protocol that runtime adaptation of parameters is indeed feasible. For this purpose, we conduct in-depth simulation experiments using a NICE implementation in the P2P simulator OverSim [2]. As service quality measures, we use message latency and resulting protocol overhead. We identify the size of the node clusters maintained by NICE, the thresholds for cluster refinement and the interval of the heartbeats exchanged by members of a cluster as relevant parameters for tuning the service quality. Subsequently, we show that these parameters can be adapted at runtime by providing an algorithm for dynamically setting the cluster size as an example. We discuss the (positive and negative) impact of an adaptation on service quality measures. Our simulation experiments indicate that runtime adaptation of the NICE protocol parameters is promising for service quality as it may improve latency up to 25%.

The remainder of this paper is structured as follows: Related work is presented in Section 2. To make the paper self contained we introduce the NICE ALM protocol in Section 3. We then present an in-depth analysis of the NICE behavior with respect to parameters that determine service quality in Section 4. In Section 5 we discuss how parameters can be adapted at runtime and illustrate the impact of runtime adaptation on service quality measures. Finally, concluding remarks are given in Section 6.

2 Related Work

Work in overlay adaptation and optimization can be roughly divided into two groups: *structure optimization* and *underlay optimization*. Structure optimization focuses on performance of the overlay structure in itself, whereas underlay optimization tries to adapt the overlay structure to the underlay, aiming to overcome performance disadvantages of overlay-based networks. Our work can be categorized as structure optimization. Work that has been performed under aspects of overlay robustness and churn [7] is explicitly excluded in our work.

Our work is most closely related to [6]: The authors present the DHT protocol Accordion that adjusts itself to different network sizes by adaptation of the routing table size. While their focus is on performance and robustness under churn in a DHT protocol, we work on adaptation in ALM and focus on service

quality. Furthermore, we concentrate on feasibility using the NICE protocol as an example. Earlier work by the same authors [7] presented an analytical parameter-space evaluation that focuses on systems under churn. The authors identify the routing table size for DHT protocols as the most important parameter. This is similar to the cluster size parameter of the NICE protocol that we use in this work for dynamic adaptation.

Fan and Ammar describe reconfiguration policies for adaptation of overlay topologies [3]. The authors consider design problems for static and dynamic overlay networks, the dynamic being based on occupancy cost and reconfiguration cost. Their work provides insight into general reconfiguration and the question of when to perform reconfiguration, whereas our work focuses on the concrete effects of parameter adaptation.

Our work is based on a defined overlay structure with adaptation within the parameter-space. Jelasity and Babaoglu [5] in contrast use a topology-space to develop a protocol for topology structure adaptation, called T-MAN. The T-MAN protocol allows for runtime variation of overlay topologies.

The authors of the NICE protocol performed analysis on static parametrization of the protocol, with focus on the k parameter [1]. In their work they look at stretch and stress and how they are affected by selecting a constant parameter k at design time. The ZIGZAG protocol [9] for application-layer multicast is similar to NICE in its layered clustering structure, but outperforms NICE with respect to node degree, and failure recovery. ZIGZAG, too, defines a constant parameter k, similar to NICE. Therefore, we see applicability of our work to ZIGZAG.

Interesting work has been published by Mao et al. on the MOSAIC system for dynamic overlay composition at runtime [8]. The MOSAIC system can dynamically compose a set of overlay protocols to include/exclude specific properties—like mobility, or performance features—provided by the respective overlay. In contrast, our work tries to optimize behavior inside a single overlay protocol.

3 NICE Protocol

The NICE protocol [1] is an early approach for ALM that implements an overlay aiming at scalability by establishing a cluster hierarchy among participating member nodes. In the following, we give a short description of the protocol.

3.1 Basic Protocol

NICE divides all participating nodes into a set of clusters. Protocol traffic is mainly exchanged between nodes residing in the same cluster, leading to good scalability. 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. Each cluster holds between k and $(\alpha k - 1)$ nodes, α and k being protocol parameters. In case of size bound violations, a cluster is split, or 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. NICE aims at combining "near" nodes in the same cluster. Cluster-leader election is accomplished by determining the node nearest to the graph-theoretic center of that cluster. 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. Conformance tests and rearrangements are performed by NICE periodically. Based on the hierarchical clustering structure, paths for data dissemination in NICE are defined implicitly. 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 one it received it from. This leads to each participant implicitly employing a dissemination tree to all other nodes in the structure. To analyze the effects of parameter adjustment, we implemented NICE in the open-source overlay simulation framework OverSim [2] based on the technical descriptions given in [1].

4 Static Parameter Selection

In this section we analyze the protocol parameters of NICE with focus on their impact on service quality.

4.1 Protocol Parameters and Service Quality Measures

The protocol behavior of NICE is adjustable by a variety of parameters. For completeness, we mention them here shortly in order to focus on the most relevant in the following. In our implementation of NICE we find the parameters α, k, HBI, min_CL_Dist and min_SC_Dist, triggering cluster sizes, interval length between heartbeat messages, and decision bounds for clusterleader estimations, respectively. Furthermore, the protocol employs several timers to detect failures in communication or structure. The *Maintenance Interval* determines the interval in which a node checks for protocol invariants. *Peer Timout* is defined to be the period of time after which a node assumes another node has failed or gone. It is a configurable multiplicity of HBI. *Structure Timeout* is the period of time after which a node assumes to be partitioned from the structure and attempts to reconnect. The *Query Timeout* detects lost queries in NICE for initiation of retransmissions (while NICE is typically soft-state, some messages have to be assured to be received in order to work properly).

NICE-specific		Simulation-specific	
Parameter	Value	Parameter	Value
α, k	3	Number of nodes	512
HBI	5s	Offset after last join	60 s
Maintenance Interval	3.3 s	Measurement phase	600 s
Peer Timeout	2 HBI	Joins	∼every 3 s
Query Timeout	2 s	Data Interval	6 s
Structure Timeout	3 HBI		
min_CL_Dist	30%		
min_SC_Dist	30%		

Table 1: Protocol and simulation parameters

As service quality measures, we consider data latency and protocol overhead. Data latency is given by the average time that elapses between sending a data packet via multicast and receiving it. Protocol overhead is measured by the average bandwidth used by a node for sending control messages.

Several pre-evaluations have shown that the major impact is limited to 3 relevant parameters in NICE:

- the clustersize parameter k
- the refinement of node cluster memberships, in the following referred to as *inter-cluster refinement* (especially by adjusting min_SC_Dist)
- the rate of protocol heartbeat messages (HBI)

Due to space limitations we only focus on k in the remainder of this paper.

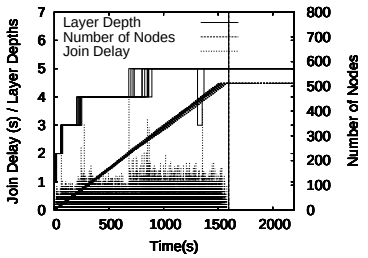
4.2 Experimental Setup

Our experiments are conducted using the peer-to-peer simulation framework OverSim [2]. As network model we chose OverSim's *SimpleUnderlay* and use the protocol-specific and simulation-specific parameters given in Table 1.

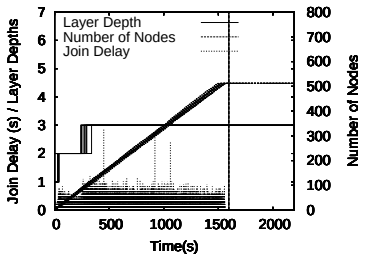
In our simulations we analyze a total of 512 nodes in NICE. From simulation start, a new node joins the network every 3 s. After the last node joined we employ a backoff time of 60 s to stabilize the hierarchy. Then, every node starts sending a data packet using the multicast structure every 6 s. After 10 min of data exchange, we again employ a backoff of 60 s before finishing the simulation run. All simulation settings have been conducted with 30 different seeds of the random number generator and mean values have been calculated for all performance measures.

4.3 Clustersize Parameter k

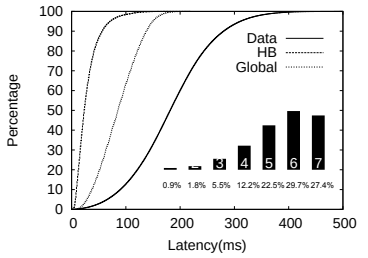
The clustersize parameter k determines the thresholds of cluster sizes that trigger splitting and merging a cluster with neighboring clusters. As all nodes in a cluster directly exchange protocol messages, increasing k will intuitively increase per-node overhead. In contrast, larger clustersizes also lead to fewer layers in the



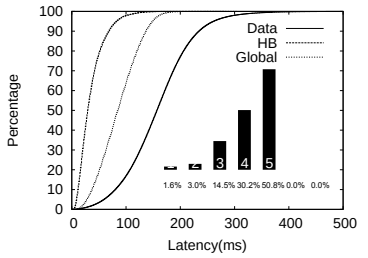
(a) Struct Building Process, k=2



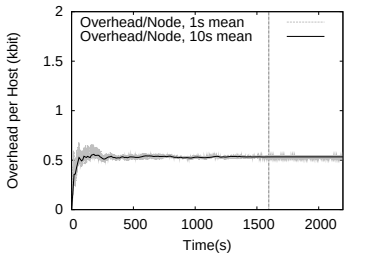
(b) Struct Building Process, k=4



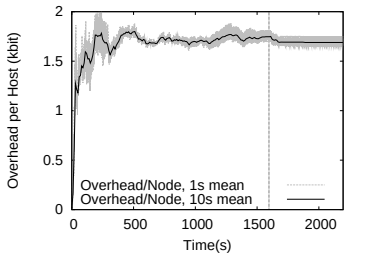
(c) Latencies and Hopcount, k=2



(d) Latencies and Hopcount, k=4



(e) Mean Overhead per Node, k=2



(f) Mean Overhead per Node, k=4

Fig. 1: Sensitivity to Clustersize Parameter k

hierarchy. Therefore, data packets have to traverse less overlay hops, leading to lower overall data latencies. Thus, adjustment of k trades off protocol overhead against data packet latencies.

Figure 1 gives an overview of the impact of different choices of k. Here, the rows reflect the resulting NICE structure, latencies and overlay hopcounts, and overhead per node, respectively. For the individual columns we set k to be 2 and 4, respectively. Concerning the hierarchy structure (Figures 1a and 1b) we find that small changes of k already have mentionable impact on protocol properties. The figures show the number of nodes in the structure and the resulting number of layers as a function of simulation time. For each joining node it also shows the *join delay*, i. e., the time that has passed between first contacting the RP

and finally joining a cluster in layer L_0. Each figure visualizes the values for 10 out of 30 different seeds each.

In general, incrementing k leads to a decrementation in the hierarchy depth. While the final structure with $k = 2$ converges to five layers, it converges to three with $k = 4$. Each additional layer also increases the mean join delays that joining nodes take to become part of the structure. Figures 1c and 1d show the resulting latencies and hopcount distributions for each k, acquired after the structure finished its building process. The figures show three aspects of latencies: (1) global network latencies between all nodes as they have been placed randomly in the simulation field, (2) latencies inside the clusters built (i. e. the heartbeat round-trip times), and (3) data latencies that the data packets experience when being routed through the overlay. We also show the distribution of hopcounts for the data packets, meaning how many overlay nodes they pass before reaching all nodes in NICE. It is clearly visible that the NICE clustering process combines nodes with low network latencies in the same cluster. Decreasing hopcounts lead to lower latencies. Finally, Figures 1e and 1f compare the resulting mean overhead per node for each k. As the number of neighbors in a cluster increases with k, the overhead due to heartbeat and other signaling also grows. Note that adjusting k also influences the impact on the underlay by trading off between resulting stress and stretch [1]. Due to space limitations, we will not discuss these aspects in this paper.

We conclude from these experiments that adjusting the clustersize parameter k enables trading off data latency and control overhead. Thus, it is promising to adapt k in a scheme for runtime parameter adaption. However, it is not clear whether k can be changed for an existing NICE overlay at runtime. We will discuss this issue in more detail in the following Section 5.

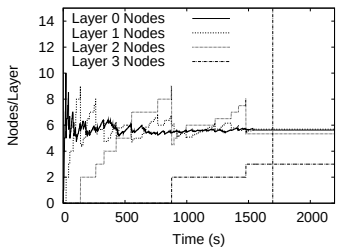
5 Runtime Parameter Adaptation

Instead of choosing parameters for overlays at design time, we propose to enable the protocol to adjust them during runtime. In the following, we provide an algorithm for choosing the clustersize parameter k dynamically and illustrate its impact on the service quality measures.

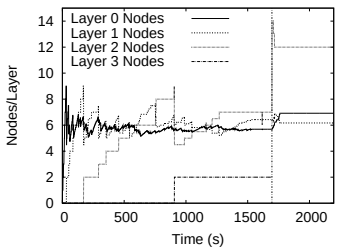
5.1 Adaptive Selection of the Clustersize Parameter k

In Section 4.3 we stated how k implicitly trades off overhead against data latencies in NICE. Assuming the protocol has knowledge about the desired latency constraints on the one hand and the tolerable resulting overhead on the other hand, it may adaptively re-adjust k during runtime to provide desired data latencies without exceeding its overhead bounds. As data packets in NICE traverse the whole structure through its hierarchy layers, latencies will increase with the depth of the hierarchy.

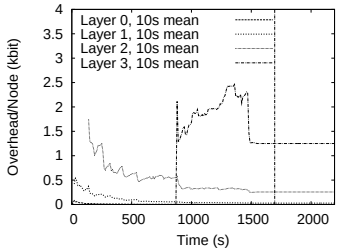
Given a NICE structure of depth d (d being the number of hierarchy layers), to decrease the overall depth by one, k has to be chosen such that all nodes in



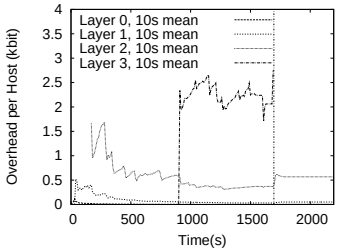
(a) Members per layer, static k=3



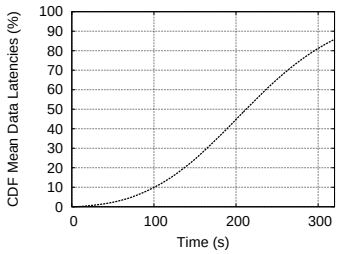
(b) Members per layer, opt. k



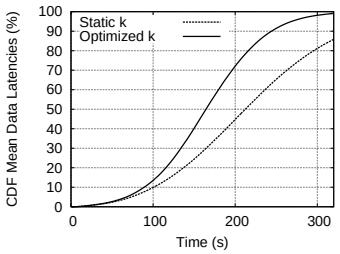
(c) Overhead per node, static k=3



(d) Overhead per node, opt. k



(e) Distribution of data latency, static k=3



(f) Distribution of data latency, optimized k

Fig. 2: Clustersize k Runtime Adaptation Schemes (512 nodes)

layer $d-1$ become part of one bigger cluster. In that case, there will be only one cluster leader left on layer d, i. e. this cluster is eliminated. To determine a suitable k under a worst-case assumption, the cluster-leader of the highest cluster in layer d must assume that every cluster in layer $d-1$ holds its maximum node number of $\alpha k - 1$ nodes. As the highest cluster-leader knows the number x of nodes in layer d (i. e. its direct neighbors in the single highest cluster), it may determine the worst case number of nodes in layer $d-1$ to be $x(\alpha k - 1)$. Based on this information the cluster-leader calculates a new value k_{new} as follows:

$$k_{new} = (x * (\alpha * k - 1) + 1)/\alpha$$

This ensures that all nodes in layer $d-1$ will fit in a single cluster, resulting in a decrease of one layer in the hierarchy structure. After calculating k_{new}, the cluster-leader instructs all nodes in layer d to merge their $d-1$-clusters with him, so that he stays the last node in layer d which is equivalent to eliminating the highest layer cluster. Furthermore, it propagates k_{new} to its new cluster members by including the new value in its periodic heartbeat messages. A node receiving a changed value k will update its own cluster it is leader of, but with a randomized bounded backoff to prevent all nodes from refining their structural part at the same time. Note that the worst case estimation may raise k to a value that potentially induces much more overhead to the participating nodes than really necessary. Thus, the protocol may be optimized by the following addition: If the highest cluster-leader knows the exact number of nodes in layer $d-1$ it can choose k to be just big enough to hold all such nodes in one cluster. To gain this knowledge, all nodes in cluster layer d tell their specific current number of $d-1$ cluster nodes to the leader of the highest cluster by including this information in their periodic heartbeat messages. At the time of changing k, the highest leader may sum up these numbers to find a value num that satisfies the following:

$$(\alpha * k) \leq num \leq x * (\alpha * k - 1)$$

In general this value will be smaller than the one computed using the worst case assumption, leading to a choice for k that is significantly smaller. We will illustrate this fact in the following Section.

5.2 Performance Results

Figure 2 gives insights in the dynamic adjustment of k during runtime. Each column covers a different case, being (1) no runtime adaptation of k at all, (2) optimized adaptation of k to decrease the hierarchy depth. In case (2) the adjustment of k is triggered actively 300 seconds after the structure has stabilized, i. e., the new value of k is propagated and the merge of all clusters in $d-1$ is triggered. The first column compares the development of the average number of members in clusters of a specific layer in the two cases (1) – (2), respectively. Figure 2a shows that with a static value of k, the clusters in each layer show a comparable size, with more layers being created with growing number of participants. When adapting k at runtime (Figure 2b), the highest layer is eliminated, while the next lower layer grows notably.

Looking at the overhead per node, Figures 2c and 2d show that the overhead per node naturally grows with the highest layer the node resides in. In case of adapting k, all nodes in the highest layer after the adaptation have higher overhead due to the higher number of cluster participants they have to exchange protocol messages with. While the overhead grows, it still remains manageable.

Comparing data latencies, we see that latency is significantly reduced as shown in Figure 2e and 2f. To illustrate the gain of the adaptation more clearly, Figure 2f compares the latencies before and after the runtime adaptation of k. The figure shows a decrease in mean data latency by 25%. We conclude that

the clustersize parameter k can be adapted during runtime in order to trade off latency against protocol overhead. How to determine an optimal new value for k is subject to future research.

6 Conclusion

Careful selection of overlay parameters is crucial for the successful deployment of overlay-based services in next generation networks. As conditions may change due to dynamics in the network or overlay structure, we propose the self-adaptation of parameters at runtime to adapt to service- and user-requirements. As a first step towards this autonomous behavior we identified parameters with high impact and showed the feasibility of adapting the clustersize parameter k during runtime using the exemplary NICE ALM protocol. By extensive simulation we presented the impact of protocol parameters and behavior during parameter changes. We have shown that runtime adaptation of k is feasible and can reduce data latencies by up to 25%.

References

1. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'02)*, volume 32, pages 205–217, Oct. 2002.
2. I. Baumgart, B. Heep, and S. Krause. OverSim: A Flexible Overlay Network Simulation Framework. In *Proc. 10th IEEE Global Internet Symp. (GI '07) in conjunction with IEEE INFOCOM*, pages 79–84, May 2007.
3. J. Fan and M. H. Ammar. Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies. In *Proc. 25th IEEE Int. Conf. on Computer Communications (INFOCOM'06)*, pages 1–12, Apr. 2006.
4. M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas. A Survey of Application-Layer Multicast Protocols. *Communications Surveys & Tutorials, IEEE*, 9(3):58–74, July 2007.
5. M. Jelasity and O. Babaoglu. T-Man: Gossip-based Overlay Topology Management. In *Proc. 4th Int. Workshop on Engineering Self-Organizing Applications (ESOA'06)*, volume 3910, pages 1–15, May 2006.
6. J. Li, J. Stribling, R. Morris, and F. M. Kaashoek. Bandwidth-efficient Management of DHT Routing Tables. In *Proc. 2nd conference on Symp. on Networked Systems Design and Implementation (NDSI'05)*, volume 2, pages 99–114, May 2005.
7. J. Li, J. Stribling, R. Morris, F. M. Kaashoek, and T. M. Gil. A Performance vs. Cost Framework for Evaluating DHT Design Tradeoffs under Churn. In *Proc. 24th IEEE Int. Conf. on Computer Communications (INFOCOM'04)*, volume 1, pages 225–236, Aug. 2005.
8. Y. Mao, B. T. Loo, Z. Ives, and J. M. Smith. Mosaic: Unified declarative platform for dynamic overlay composition. In *Proc. Int. Conf. On Emerging Networking Experiments And Technologies (CoNEXT'08)*, pages 883–895, Dec. 2008.
9. D. A. Tran, K. Hua, and T. Do. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In *Proc. 22th IEEE Int. Conf. on Computer Communications (INFOCOM'03)*, volume 2, pages 1283–1292, Mar. 2003.