Quality of Service for Multicasting using NICE

Marc Brogle, Sebastian Barthlomé, Torsten Braun
Institute for Computer Science and Applied Mathematics
University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland
brogle|barthlom|braun@iam.unibe.ch

ABSTRACT

To distribute data from one sender to multiple receivers efficiently and concurrently, multicasting is one of the most appropriate mechanisms. Application Layer Multicast (ALM), often also referred to as Overlay Multicast, has been introduced to overcome the limitations of IP Multicast. The OM-QoS (Quality of Service for Overlay Multicast) framework aims to enable QoS for different ALM protocols. We applied the OM-QoS mechanisms to the Overlay Multicast protocol NICE and performed evaluations in QoS environments using resource reservations and measurement based QoS. Our evaluations show that we can support the QoS requirements of all paths in the multicast tree, while introducing an acceptable overhead in terms of delay.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Application layer multicast, Overlay networks, Peer-to-peer

1. INTRODUCTION

Multicasting is an efficient mechanism for a sender to distribute multimedia data to many receivers concurrently. IP Multicast is an implementation of the multicast paradigm for the Internet. Unfortunately, IP Multicast is not widely available for end users in the Internet today. This is due to multiple reasons, such as complex provider billing agreements, security concerns, configuration complexity and others. Peer-to-Peer (P2P) and Application Layer Multicast (ALM) networks offer a viable solution to overcome the limitation of IP Multicast availability. To give the best service to end-users, Quality of Service (QoS) mechanisms for P2P/ALM networks would be desirable. This would enhance user experience for multimedia applications such as IP-TV, multiplayer online games, real-time A/V conferencing and streaming, etc.

A description and evaluation of how Scribe/Pastry could be made QoS aware, by changing Pastry's ID assignment mechanism, was presented in [6]. The evaluation of the approach was performed using Freepastry's integrated simulator. Freepastry is a freely available implementation of Scribe / Pastry. A self-managing Quality of Service for Overlay Multicast (OM-QoS) framework as a protocol independent approach on how to enable QoS for different P2P/ALM protocols was proposed in [8]. Different P2P protocols were analyzed and solutions on how to make those protocols QoS aware were presented but not evaluated. The solution to make Content Addressable Networks (CAN) [16, 15] QoS aware using OM-QoS as presented in [8] was evaluated in [5] using the OMNet++ [19] network simulator. In this paper, we discuss the application of OM-QoS to NICE (NICE is the Internet Cooperative Environment) [1] and evaluate this also using OMNet++ [19]. We introduce mechanisms to support certain guarantees regarding delay constraints for the path from a node to the multicast root.

The remainder of this paper is structured as follows. Section 2 gives a very short overview of P2P/ALM and NICE. In Section 3 we describe how we enhanced NICE with QoS awareness/capabilities. The evaluation of our approach is described in Section 4. Finally, in Section 5 we present our conclusions.

2. RELATED WORK

2.1 Application Layer Multicast

Application Layer Multicast (ALM) [10] and Peer-to-Peer (P2P) [18, 13] networking has become very popular. ALM helps to overcome the limited availability of IP Multicast to end users in the Internet today. In order to improve the P2P/ALM concepts, Quality of Service (QoS) aspects have to be taken into account. Different approaches such as QRON [12] and Pathaware Multicast [11] have been introduced to enable QoS functionality in P2P/ALM.

In order to offer a flexible and general approach to support QoS for ALM, the OM-QoS (Quality of Service for Overlay Multicast) framework [6, 7, 8, 5] has been introduced. It aims to enable different P2P/ALM protocols to support QoS. The first analysis in [6] focused on how to make the Scribe ALM [9], which runs on-top of Pastry P2P [17], QoS aware. This approach could work also with other Plaxton routing [14] based ALM/P2P infrastructures, such as Bayeux and others. In order to support more P2P/ALM protocols, mechanisms have been analyzed regarding how to enable QoS for NICE [1] and Content Addressable Networks (CAN) [16, 15]. The corresponding solutions have been described in [8].

SAC'10 March 22-26, 2010, Sierre, Switzerland.

2.2 NICE Overlay Multicast

In this paper, we focus on the implementation and evaluation of OM-QoS for NICE. Nodes in NICE are arranged in clusters and

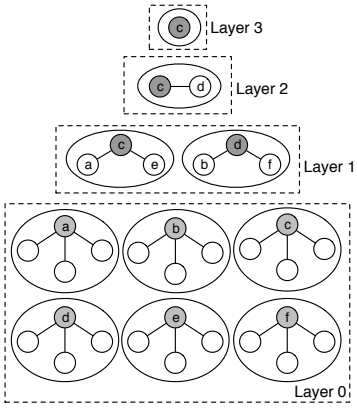


Figure 1: Example of NICE

layers and are structured hierarchically as shown in Fig. 1. Nodes within the same cluster are called cluster mates. Each cluster mate knows its cluster leader and some or all other cluster mates in the same cluster. The size of a cluster varies between the lower bound k and the upper bound $3k-1$, with k being a predefined cluster constant larger than zero. These boundaries help to avoid conflicting maintenance operations, which will be discussed later. One of the cluster mates within a cluster is determined to be the cluster leader. This cluster leader is then a member of a cluster of the next higher layer. Cluster leaders are determined by choosing randomly a node from the so-called graph-theoretic center of clusters. This graph-theoretic center is calculated by calculating the eccentricity for each node. The eccentricity is the maximum distance from the node to any other node in the cluster. Nodes with minimum eccentricity then build the center of the cluster. Each layer consists of one or more clusters. They are are ordered from the bottom layer zero to the top layer n. The top layer consists only of a single cluster with one cluster member. This is the root of the NICE network. The structure of NICE is specified with five invariants. They have to be fulfilled at any time. 1) A node belongs to only a single cluster on each layer. 2) A node located at layer L is also also located at layers $L-1, ..., 0$. 3) A node not present in layer L can not be present in any higher layer ($L+i, i \geq 1$). 4) The size of a cluster is between k and $3k-1$, where k is a constant with $k > 0$. 5) There is a maximum of $\log_k N$ layers and the highest layer only contains one node (the root node).

2.2.1 Joining a NICE Network

A node joining a NICE network has to contact first the root node for that specific NICE network. It then receives a list of all cluster leaders on the next lower layer. In the next step, it contacts every reported node and identifies the closest node to itself in terms of round trip time (RTT). This helps to select the most appropriate cluster to join. Therefore, nodes that are physically close are grouped together in a cluster. Afterwards, the joining node sends a new request to the determined closest node. This process is iteratively repeated until layer 0 is reached. There, the new node joins the most appropriate cluster.

2.2.2 Leaving a NICE Network

Nodes can leave a NICE network in two ways, gracefully and ungracefully. In the graceful case, the leaving node announces its departure to its cluster mates before it really disappears. Other nodes can now react to this situation appropriately and perform any handovers from the leaving node that might be necessary. If a node suddenly disappears, then it leaves ungracefully. The cluster leader and the cluster mates do not recognize the departure of the node directly but only by timeouts and missing heartbeat messages. In such cases, parts of the NICE network might not work properly due to invalid states or missing handovers.

2.2.3 Maintenance of a NICE Network

Maintenance of NICE contains several refinement operations to handle the nodes and the tree structure. Heartbeat messages are used to periodically exchange information such as the view of a cluster by a node and its RTT to the other cluster members. The refinement operations might consist of splitting a cluster, merging two clusters, or to determine a new cluster leader. The refinement operations are only invoked by a cluster leader when it detects an invalid or not optimized state in its cluster. Leave and join operations could require the cluster leaders to be changed.

3. QUALITY OF SERVICE FOR NICE

3.1 QoS Aware Multicast Trees

In order to enable QoS for Overlay Multicast, the multicast distribution tree has to hold certain properties. As described in [6, 8], the multicast tree has to be built such that the QoS requirements or capabilities are monotonically decreasing from the root to the leaf nodes. This guarantees that nodes are only connected to a parent node having the same or higher QoS requirements than the connecting node itself. Therefore, the parent node will be able to serve the connecting node with multicast data that supports the connecting nodes' QoS requirements. Such a QoS aware tree is depicted in Fig. 2, where thicker lines between nodes correspond to higher QoS requirements.

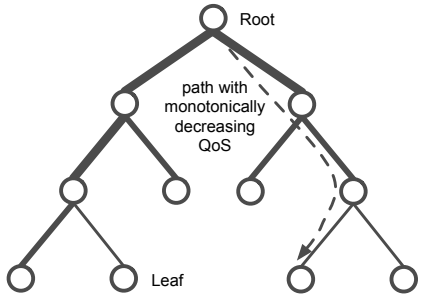


Figure 2: QoS Supporting Multicast Tree

3.2 QoS Classes

To manage different QoS parameters, we introduce the concept of QoS classes. We can combine multiple QoS parameters into a QoS class, which is then represented as a single integer value. To maintain the structure as shown in Fig. 2, QoS classes have to follow certain rules: 1) A total order relation for all QoS classes exists. 2) QoS class parameters are independent of link length and number of hops in the network. 3) The number of QoS classes is finite. This means that a QoS class can be a combination of bandwidth, hop-by-hop jitter and CPU requirements. Parameters that are accumulated over many hops, e.g., end-to-end delay, can not be combined into a QoS class. Our solution for the delay problem will be explained later in this Section.

3.3 QoS Trees for NICE

As a first option, the creation of QoS aware trees can be achieved by a simple modification of the NICE protocol. Instead of determining the cluster leader from the graph theoretic center of the cluster, it is determined by the nodes having the highest QoS class in the cluster. This approach is explained in more detail in [8].

A second approach to build such QoS aware trees is the Quality of Service for Overlay Multicast (OM-QoS) Framework layered (protocol independent) approach presented in [8]. The basic idea is to use for each active QoS class a dedicated P2P/ALM instance (called slice). Using NICE, we build a separate NICE slice for every QoS class as shown in Fig. 3. The different slices are interconnected using gateway nodes, connecting adjacent slices (in terms of QoS) with each other. This approach guarantees that the resulting multicast tree in such a collective of NICE networks follows the previously described property of monotonically decreasing QoS requirements (from root to leaf nodes) as shown in Fig. 2.

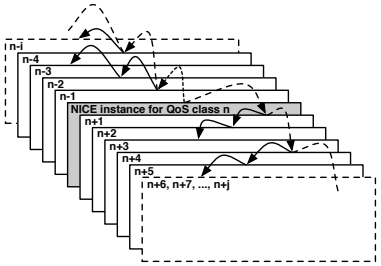


Figure 3: QoS for NICE using multiple NICE Slices

To reduce the overall hop-count and latency from the root to a leaf node, the different NICE slices can not only be interconnected by single steps through adjacent slices. Instead, additional links can be established jumping over multiple slices as shown in Fig. 3. This mechanism helps to improve the reliability and latency of forwarding multicast data between slices. To further optimize the reliability, we introduced backup links between the slices as proposed in [2]. If a gateway node fails, the adjacent gateway nodes from a lower or higher slice (in terms of QoS) already have backup links to alternative gateway nodes prepared and ready. These backup links to alternative gateway nodes then can takeover the task of forwarding multicast data between slices from the failing node on the fly.

3.4 Node to Root RTT Constraints for NICE

To support also end-to-end delay constraints, we introduce end-to-end delay as an additional parameter. This is independent from the QoS class construct and allows a peer to request a QoS class (for example a certain bandwidth) for its connection and also ask for an upper bound of the round trip time (RTT) to the root of the multicast tree. We call this upper bound *node to root RTT constraints*.

In this paper, we only take node to root RTT constraints ranging from 25ms–50ms into account when joining the NICE network. When a node joins a NICE network, it automatically participates in multicast distribution tree for that NICE network, and starts receiving and possibly also forwarding multicast data. We implemented this mechanism to support node to root RTT constraints as follows. When a newly joining node measures the RTT to the potential cluster leaders on each layer, it also asks the potential cluster leaders on all layers for their current node to root RTT value. The cluster leader that fulfills the node to root RTT constraint as closely as possible is then selected for the next iteration. That means the node offering the maximum node to root RTT that is still below the node to root RTT constraints (with a certain safety margin) is selected for the next step in the next lower layer. This is continued until we reach layer 0, where the node joins a cluster that fulfills the node to root RTT constraints.

As an example, we can assume that we have a NICE network with 4 layers. In the top layer 3, only the root is present in its own cluster. A newly joining node asks the cluster leader in layer 3 (the root node) for its cluster mates in layer 2. These cluster mates are cluster leaders of clusters in layer 1 and also at layer 0. The joining node then measures the RTT to these reported nodes and asks them for their node to root RTT. Now, the joining node can determine its own node to root RTT if it would join a cluster of the previously reported nodes. It selects and contacts the node that offers the maximum node to root RTT below its own node to root RTT constraints to ask for that nodes' cluster mates in layer 1. These reported nodes are cluster leaders of clusters in layer 0. Then, determining the node to root RTT matching the node to root RTT constraints are performed again as described before. Finally, the node joins the cluster in the layer that offers the maximum node to root RTT just below its own node to root RTT constraints.

4. EVALUATION

4.1 Overview

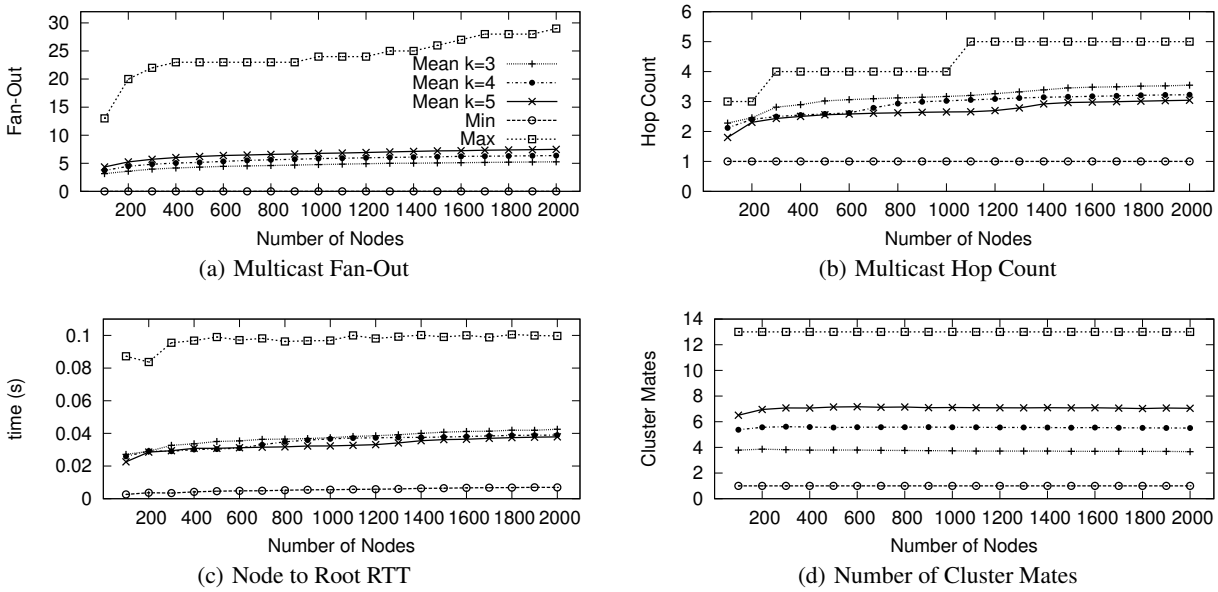
Our evaluation of QoS for NICE has been performed using the OMNet++ [19] simulator. We implemented the basic NICE protocol and made some enhancements to support further reliability, such as handshakes for cluster-leader transfers, split and merge operations, leaving and root transfers. Our implementation is fully decentralized, the overlay structure was setup and maintained using only message transmission among the NICE nodes.

We investigated different scenarios, starting comparing the influence of the k parameter on certain issues of a normal NICE network without QoS support. Furthermore, we compared normal NICE and QoS aware NICE (protocol dependent OM-QoS approach). The QoS aware NICE has been evaluated using static hard QoS guarantees offered by the underlying network and dynamic soft QoS. Hard QoS could be achieved using e.g. DiffServ or EuQoS [3]. Soft QoS is a measurement based best-effort QoS. Nodes measure the QoS provided by potential parents and then select the one supporting the required QoS. But, QoS capabilities of a node can change over time. This means that some links between nodes might not support the required QoS anymore after a certain time. Then, a new parent has to be found, which again on the link between the node and the parent supports the initially required QoS. The protocol independent (layered) OM-QoS approach was also investigated. Finally, we also looked at the node to root RTT constraints explained in Section 3.4 that a node might request. We modified the join process to the NICE network in such a way that nodes should have their node to root RTT below their requested upper bound for that value directly after they entered the network.

4.2 Simulation Scenarios and Parameters

The latencies between all the nodes for the simulation scenarios are determined using distance matrices defining the latencies for each possible node pair connection. We use for that matter distance matrices, which we built using topologies generated by Brite [4]. As model, we used "Router Waxman" with $\alpha = 0.15$, $\beta = 0.2$, and with 5000 squares for the main plane and inner planes. Nodes were placed randomly with incremental growth type and have two neighboring nodes.

Brite creates a topology file that contains nodes and edges with distances between nodes specified as a delay value in ms. For every



(a) Multicast Fan-Out

(b) Multicast Hop Count

(c) Node to Root RTT

(d) Number of Cluster Mates

Figure 4: Comparing normal NICE with k=3, k=4, and k=5

Table 1: Delay Properties of Distance Matrices in ms

Matrix	**min RTT** (ms)	**mean RTT** (ms)	**max RTT** (ms)
Matrix 0	0.08	22.47	48.44
Matrix 1	0.09	30.35	90.48
Matrix 2	0.05	30.56	94.58
Matrix 3	0.05	29.76	90.23
Matrix 4	0.07	23.26	57.52
Matrix 5	0.09	22.78	51.82
Matrix 6	0.04	22.77	49.24
Matrix 7	0.08	23.27	52.30
Matrix 8	0.05	22.91	53.92
Matrix 9	0.05	23.27	50.83
Matrix 10	0.08	22.47	48.44
Matrix 11	0.05	22.91	54.00
Matrix 12	0.01	23.13	54.17

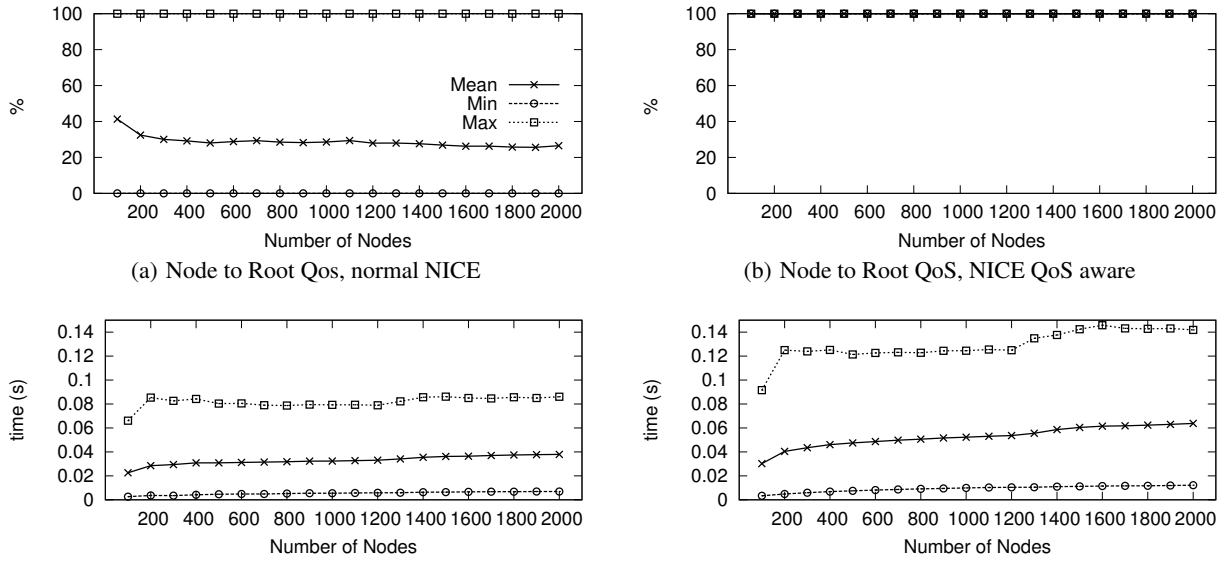
Table 2: Values Evaluated in the Simulation Scenarios

Multicast fan-out	describes the number of cluster mates a cluster leader has to serve with multicast data.
Multicast hop count	depicts the number of hops required to reach the root of the multicast tree.
Node to root RTT	denotes the round trip time (RTT) from a node to the root of the multicast tree.
Number of cluster mates	measures the number of cluster mates per cluster leader.
Node to root QoS	shows the percentage of paths, which fulfill the QoS requirements.
Percentage of multicast received	describes the percentage of multicast messages received per node.
Rejoin duration	is the time a node takes to join a new cluster, if its QoS requirements are not fulfilled in its current cluster.
Node to root RTT constraints fulfilled	is the percentage of nodes for which NICE satisfies the given E2E RTT constraint during the initial join process.
Node to root RTT after join	is the RTT from a node to the multicast root directly after joining.
Node to root RTT failed difference	is the difference between effectively achieved node to root RTT and the node to root RTT constraint after join.

pair of nodes, we calculated the shortest path in terms of delay. Then we stored these values in the distance matrices as the delay for a certain edge between two nodes. Table 1 depicts the minimum, maximum, and mean delay values of the matrices.
We performed our simulations using various network sizes with a node count from 100 to 2000 in steps of 100. Each of these steps was evaluated using 13 different distance matrices, which we evaluated each with three different random seeds. The seeds are influencing the arrival time, departure decisions and many other random based values. Therefore, each scenario consists of 780 simulation runs. For all scenarios, we removed 1% of the outliers (0.5% of the min. and max. values each). We evaluated different values in each of the scenarios as presented in Table 2.

4.3 Impact of k Value on Normal NICE

Figure 4 compares the impact of the k value to normal NICE on multicast fan-out, hop count, node to root RTT, and number of cluster mates. We evaluated the maxima and minima for the different values of k individually. But, we only show the highest maximum and lowest minimum encountered for any value of k.
In Fig. 4(a), we can see that the average fan-out is only slightly depending on k. The presented maximum is encountered using $k = 5$ and is caused by the root node. Leaf nodes do not forward any multicast data, hence we have a minimum fan-out of 0. This is where



(a) Node to Root Qos, normal NICE

(b) Node to Root QoS, NICE QoS aware

(c) Node to Root RTT, normal NICE

(d) Node to Root RTT, NICE QoS aware

Figure 5: Comparing normal NICE with with QoS aware NICE, both with 256 QoS Classes and k=5

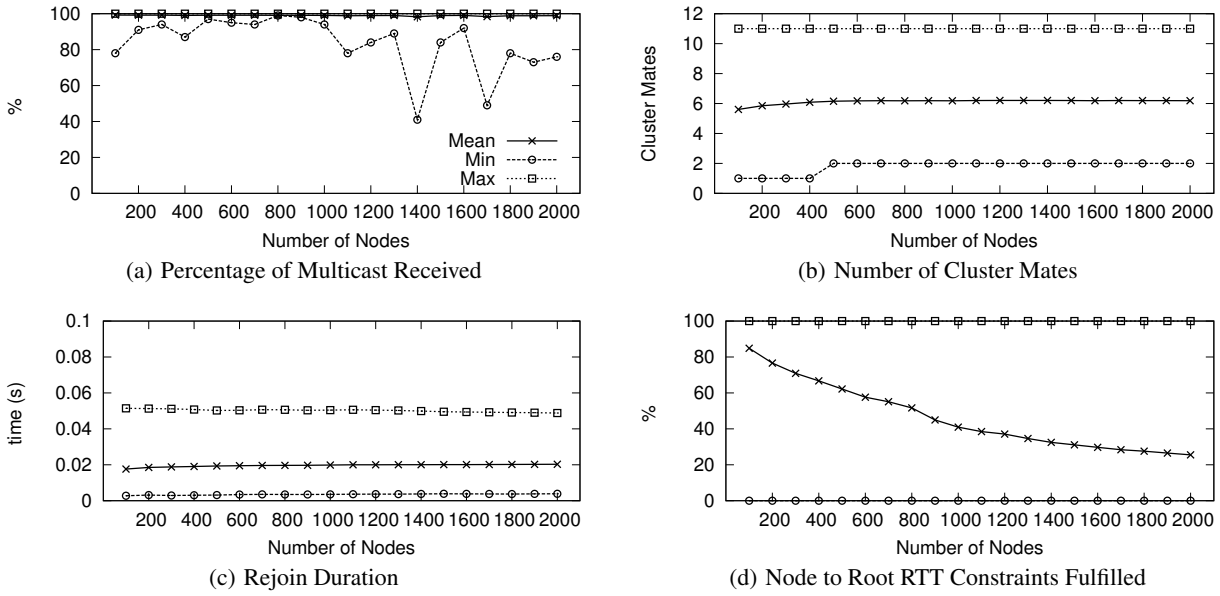
we have the highest number of cluster mates per cluster on average, and where this particular root node has to serve every layer with multicast messages. With lower k, the hop count increases as shown in Fig. 4(b). When we have smaller clusters, we will get more layers, which leads to a higher hop count. The maximum is though still acceptable, only 1–2 hops higher than the average. Since NICE always tries to optimize clusters according the RTT between the cluster mates, the mean node to root RTT values as shown in Fig. 4(c) are only slightly increasing with lower k values. The maximum is slightly above twice the worst average. Finally, the number of cluster mates presented in Fig. 4(d) is again heavily depending on the k value and behaving as expected, being somewhat above the effective k value. The maximum is again roughly twice the highest average value encountered for $k = 5$. For all these aspects analyzed, the minima values are as expected, considering that the outliers have been removed.

4.4 Normal NICE vs. QoS Enabled NICE

In Fig. 5 we compare normal NICE (QoS-unaware) with QoS enabled NICE using the protocol dependent OM-QoS approach, both having $k = 5$. Choosing $k = 5$ reduces the number of layers in a NICE network, and therefore also reduces the hop count and node to root RTT while only having a slightly increased fanout. Each node gets a QoS class assigned from the range 0–255. We assume that the underlying network will provide the requested QoS for the QoS aware NICE scenario. Therefore, the QoS will remain static. We call this hard QoS. In normal NICE, the cluster leaders are determined using delay measurements and calculating the graph-theoretic center as described in Section 3.3. Therefore, QoS classes are not taken into account when joining or when cluster leaders have to be determined in normal NICE. But we still check how many paths would satisfy the property of monotonically decreasing QoS requirements as described before. In Fig. 5(a) we can see that on average, only 30% of the paths hold the property, whereas in Fig. 5(b) with QoS aware NICE, 100% of the paths fulfill the QoS requirements. Since the QoS aware NICE has a different cluster leader determination mechanism, we also compare the node to root RTT in Figures 5(c) and 5(d). The normal NICE mode is optimized and has an average node to root RTT between 20ms–40ms depending on the number of nodes in a NICE network. The QoS aware NICE has an average node to root root RTT of 30ms–60ms. Using the QoS aware mechanism adds another 50% of extra delay to the node to root RTT. This is expected due to the fact that cluster leaders are not anymore determined and optimized using delay measurements. Instead, QoS classes are used to determine cluster leaders. The maximum of the node to root RTT value on the other hand almost doubles when using QoS mechanisms. Still, introducing QoS to NICE using our mechanism allows us to guarantee that all paths from the root node to the leaf nodes in the multicast tree fulfill the QoS requirements while adding an acceptable overhead in terms of delay.

4.5 NICE with Dynamic Soft QoS

In Figure 6, we look at the scenario with dynamic soft QoS. This means that the QoS guarantees can change over time. If QoS cannot further be supported along a path, the node has to look for a new parent (or cluster leader in NICE) that actually supports again its QoS requirements. In this evaluation, we assume that QoS guarantees can fail up to 5 times per node while it is in a NICE network. The time interval during which this can happen depends on the join and leave times of a node which are determined randomly. It is also determined randomly how often QoS guarantees fail per node . Again, we use 256 QoS classes and also $k = 5$ for this scenario with the protocol dependent OM-QoS approach to support QoS for NICE. As shown in Fig. 6(a), the dynamic behavior of the QoS guarantees does not have a negative impact on the percentage of multicast messages received by nodes on average. Almost all multicast messages are delivered as intended. Some nodes though might have a higher loss rate due to multiple rejoins, merging and split operations, cluster leader and root transfers, as visible from the minimum value. During these times, there might be some moments where a node is not part of a cluster, and therefore might not receive multicast messages from a cluster leader. This is especially the case for nodes that are only participating in a NICE network for a short time. If during this time the previously mentioned operations (rejoin, split, merge, transfers) occur frequently,



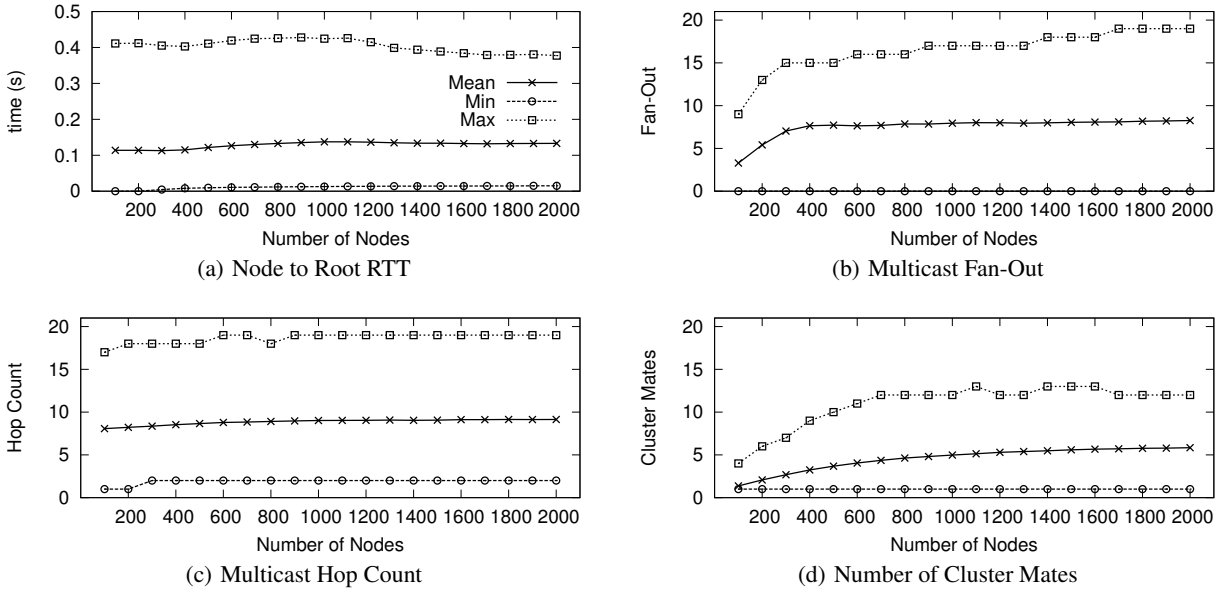
(a) Percentage of Multicast Received

(b) Number of Cluster Mates

(c) Rejoin Duration

(d) Node to Root RTT Constraints Fulfilled

Figure 6: QoS aware NICE in a Dynamic Soft QoS Environment with 256 QoS Classes and k=5



(a) Node to Root RTT

(b) Multicast Fan-Out

(c) Multicast Hop Count

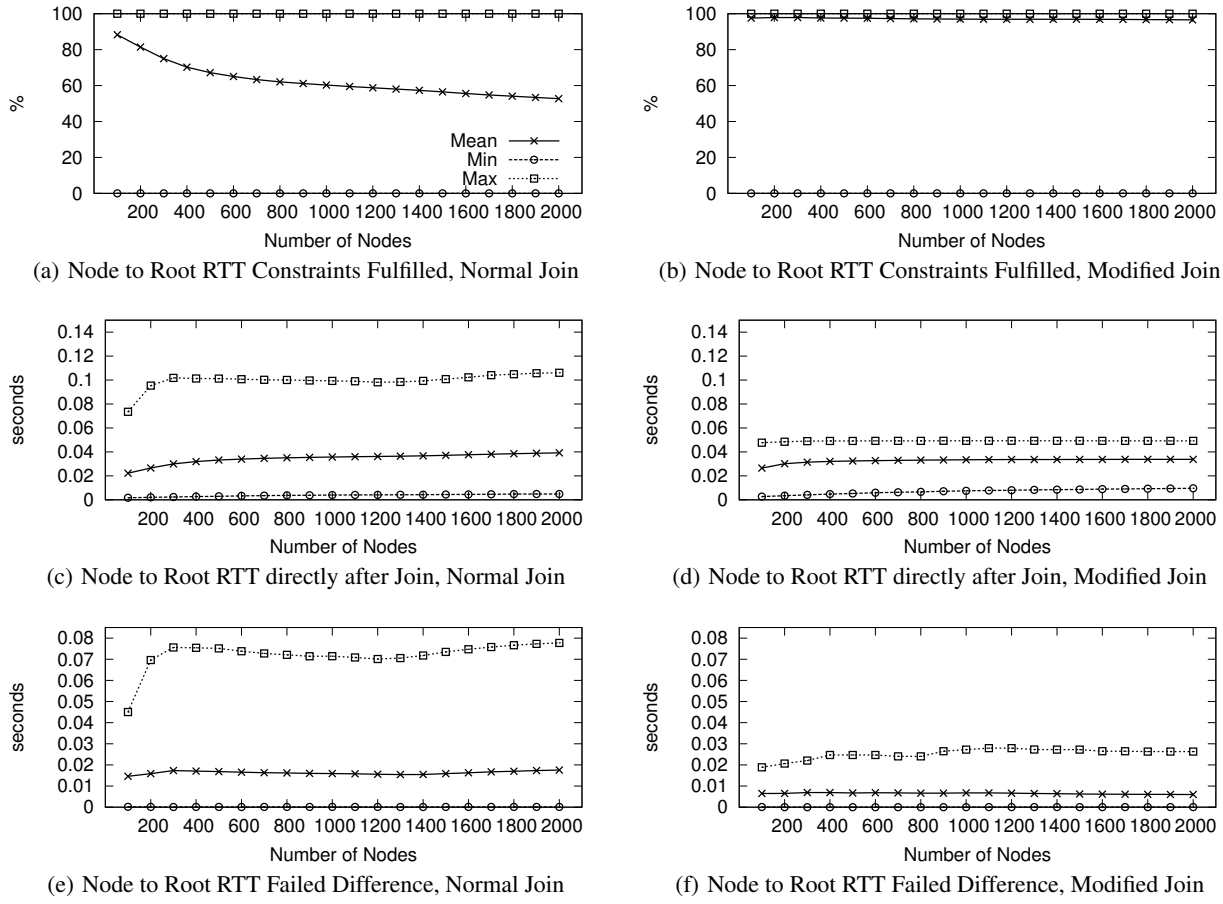
(d) Number of Cluster Mates

Figure 7: QoS aware NICE using the OM-QoS Framework Layered Approach with 32 QoS Classes and k=5

it can happen that such nodes only receive a few multicast messages. This then leads to a high loss rate. The number of cluster mates presented in Fig. 6(b) is lower (by one node) than in the normal NICE network previously shown in Fig. 4(d) for $k = 5$. This is due to the fact that nodes actually leave their cluster and rejoin other clusters, which contributes to a more equal distribution of nodes among the layers and clusters. The rejoin duration is presented in Fig. 6(c), where the average rejoin duration is around 20ms and the maximum is around 50ms, which are both acceptable values. Figure 6(d) presents how many paths actually fulfill certain node to root RTT constraints for a NICE network that does not take these RTT constraints into account. Each node selects a node to root RTT constraint in the range of 25ms–50ms. We then check how many paths actually fulfill these constraints during the time a node participates in the NICE network.

4.6 NICE with Protocol Independent OM-QoS

The results for a QoS aware NICE network using the protocol independent (layered) OM-QoS approach with 32 QoS classes and $k = 5$ are presented in Fig. 7. Nodes are now distributed over 32 different NICE networks (slices). There is one dedicated NICE network for each of the 32 QoS classes. Before, there was only one NICE network for all the nodes independent of their QoS class. Nodes joining the same network results in more layers and clusters in the network compared to distributing them among several slices, which then have less layers and clusters. We now used 32 QoS



(a) Node to Root RTT Constraints Fulfilled, Normal Join

(b) Node to Root RTT Constraints Fulfilled, Modified Join

(c) Node to Root RTT directly after Join, Normal Join

(d) Node to Root RTT directly after Join, Modified Join

(e) Node to Root RTT Failed Difference, Normal Join

(f) Node to Root RTT Failed Difference, Modified Join

Figure 8: Comparing QoS aware NICE with Normal Join and Modified Join taking Node to Root RTT Constraints into Account

classes, which is still a reasonable value, to have an acceptable hop count compared to normal NICE, because of the additional hops introduced by forwarding data between slices. As already mentioned, we now have much smaller NICE networks in terms of participants, with less clusters and less layers. Nodes that join a specific NICE network slice might be distributed in such a way, that only small clusters would be formed in those NICE network slices. This fact leads to a lower fan-out per node, less hops to the root of the multicast tree and also a lower node to root RTT inside the individual NICE network slices compared to having one big NICE network holding all nodes. As shown in Fig. 7(a), using multiple slices of NICE in parallel has an impact on the average node to root RTT, which is now 110ms–120ms. Also the maximum raises up to 400ms. This is due to the fact that the 32 NICE network slices have to be interconnected as described in Section 3.3, causing additional hops resulting in a higher node to root RTT. The average fan-out presented in Fig. 7(b) is between 3–8 hops. Although we have additional fan-out caused by the gateway links, the average fan-out starts lower than for the normal NICE with $k = 5$ as shown in Fig 4(a). It also has the same peak for the average for more hosts in the scenario. The maximum is though raising almost up to 20, which is due to the multiple optimized gateway links as well as the backup links (when they are in use). The hop count presented in Fig. 7(c) suffers from the same penalty of having multiple instances and is now on average between 8–9 hops, with a maximum raising almost up to 20 hops. Having multiple parallel instances has again an impact on the number of average cluster mates as presented in Fig. 7(d). Again, the nodes are distributed over parallel instances of NICE. Therefore, the average number of cluster mates is reduced compared to the normal NICE network with $k = 5$ as presented in Fig. 4(d). Still, since we have now 32 NICE instances in parallel and have optimizing gateway links and backup links between the different NICE instances, the overall resulting values from that scenario remain acceptable. In some cases they even have been improved, e.g. average number of cluster mates and maximal fan-out.

4.7 NICE with Node to Root RTT Constraints

Figure 8 presents the evaluation results comparing QoS aware NICE using the normal join process of NICE and a modified join process that takes the node to root RTT constraints into account. When nodes join the NICE network, they select randomly a node to root RTT constraint in the range of 25ms–50ms. The number of fulfilled node to root RTT constraints for a QoS aware NICE network that does not offer a join process taking those constraints into account is presented in Fig. 8(a). Almost half of the nodes do not have a node to root RTT, which is below their required constraints (directly after joining the NICE network). In a NICE network with the modified join process taking these constraints into account, almost all nodes have their RTT to root constraints satisfied (directly after joining the NICE network) as shown in Fig. 8(b). The resulting node to root RTT after a node has joined a NICE network using the normal NICE join process is presented in Fig. 8(c). The average starts to go quite close to the upper end (50ms) of the node to root RTT constraint's range with more nodes in the network. The

maximum is around 100ms, which is the double of the allowed range. The results for a NICE network using the modified join process, which tries to fulfill the node to root RTT constraints (between 25ms–50ms) of a node are shown in Fig. 8(d). The average for the node to root RTT is close to the middle (37.5ms) of the RTT constraint's range. The maximum is at the end of the constraint's range (50ms). In Figures 8(e) and 8(f), the deviation from the required node to root RTT constraint directly after joining is presented. As expected, using the normal NICE join process results in a higher deviation to the node's RTT constraints compared to the modified join process. The node to root RTT values achieved using the modified join process would be normally still tolerable. This is due to the fact that the deviations are small and the achieved values for the node to root RTT are generally only slightly above constraint. Keep in mind that using the modified join process results in only a few nodes that do not have fulfilled their requirements regarding the node to root RTT constraint (directly after joining the NICE network) as shown in Fig. 8(b).

5. CONCLUSION

In this paper, we have presented OM-QoS (Quality of Service for Overlay Multicast) applied to NICE, which allows NICE to offer Quality of Service (QoS) enabled multicasting. We evaluated the different approaches of OM-QoS, the protocol dependent as well as the protocol independent (layered) approach.

Using OM-QoS in the protocol dependent approach introduces a slight additional delay compared to normal NICE. This is due to the modified cluster leader determination mechanism, but does not change other basic aspects of NICE significantly. All the paths from the node to the root of the multicast tree in the hard QoS and soft QoS scenarios support QoS when OM-QoS was used. The soft QoS results show that in dynamic QoS environments, nodes quickly can find new clusters (which means parents in the multicast tree) in case QoS is not supported anymore by its current cluster. In the hard QoS scenarios, QoS is statically guaranteed by resource reservation performed in the underlaying network. Hence, joining a cluster that supports the QoS requirements of a node means that the cluster leader / parent will always support the requested QoS as long as the node remains in the NICE network.

The protocol independent (layered) approach introduces additional hops and has higher delays compared to the protocol dependent approach. But, using the protocol independent approach also reduces other aspects, such as the average number of cluster mates and maximum fan-out. This is due to the fact that there are now smaller NICE networks existing and cooperating in parallel slices. Finally, our presented approach to take also certain node to root RTT constraints into account seems to be promising. When a node joins a NICE network, it tried to find a cluster that supports its node to root RTT requirements. A joining mechanism that takes node to root RTT constraints into account enables that the majority of the nodes have those constraints fulfilled directly after joining.

6. REFERENCES

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 32, New York, NY, USA, 2002. ACM.

[2] T. Braun, V. Arya, and T. Turletti. Explicit routing in multicast overlay networks. *Computer Communications*, 29(12), August 2006.

[3] T. Braun, M. Diaz, J. Enrquez-Gabeiras, and T. Staub. *End-to-End Quality of Service Over Heterogeneous Networks*. Springer, 2008.

[4] Brite - Boston University representative internet topology generator, online: http://www.cs.bu.edu/brite/, 2009.

[5] M. Brogle, L. Bettosini, and T. Braun. Quality of service for multicasting in content addressable networks. In *12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS 09)*. Springer LNCS, October 2009.

[6] M. Brogle, D. Milic, and T. Braun. QoS enabled multicast for structured P2P networks. In *Workshop on Peer-to-Peer Multicasting at the 4th IEEE Consumer Communications and Networking Conference*. IEEE, January 2007.

[7] M. Brogle, D. Milic, and T. Braun. Supporting IP multicast streaming using overlay networks. In *QShine: International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*. ACM Press, Aug. 2007.

[8] M. Brogle, D. Milic, and T. Braun. Quality of service for peer-to-peer based networked virtual environments. In *P2P-NVE 2008 Workshop at the 14th IEEE International Conference on Parallel and Distributed Systems*, Melbourne, Victoria, Australia, December 2008. IEEE.

[9] M. Castro, P. Druschel, A. M. Kermarrec, and A. I. T. Rowstron. Scribe: a large-scale and decentralized application-level multicast infrastructure. *Selected Areas in Communications, IEEE Journal*, 20(8):1489–1499, 2002.

[10] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas. A survey of application-layer multicast protocols. *Communications Surveys & Tutorials, IEEE*, 9(3), 2007.

[11] M. Kwon and S. Fahmy. Path-aware overlay multicast. *Computer Networks*, 47(1):23–45, January 2005.

[12] Z. Li and P. Mohapatra. Qron: Qos-aware routing in overlay networks. *Selected Areas in Communications*, 22(1), 2004.

[13] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, 2005.

[14] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *SPAA '97: Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, New York, NY, USA, 1997. ACM.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 31. ACM Press, 2001.

[16] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, London, UK, 2001. Springer.

[17] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, London, UK, 2001.

[18] S. A. Theotokis and D. Spinellisa. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, December 2004.

[19] Website. OMNET++, online: http://www.omnetpp.org, 2009.