Quality of Service for Overlay Multicast in Chord

Marc Brogle, Andreas Rüttimann, Torsten Braun
Institute for Computer Science and Applied Mathematics
University of Bern, Switzerland
brogle|ruettima|braun@iam.unibe.ch

***Abstract*—This paper describes how Quality of Service (QoS) support can be introduced to Overlay Multicast in a Chord Peer-to-Peer network. We support the concept of QoS classes (to support various hop-by-hop QoS parameters such as bandwidth requirements) as well as node to root Round Trip Time (RTT) constraints. Our evaluations show that we can guarantee QoS without changing the basic properties of Chord significantly.**

I. Quality of Service for Chord Multicast

In order to enable QoS for Overlay Multicast, the multicast distribution tree has to hold certain QoS path properties. The QoS requirements/capabilities (e.g. bandwidth) of nodes have to monotonically decrease on the paths from the root to the leaf nodes in the tree. This is described in more detail in [1], [2] where we also introduce the concept of QoS classes. Using QoS classes, multiple QoS parameters can be combined into one discrete parameter. QoS parameters, which would be accumulated over many hops, can not be combined into a QoS class, e.g. end-to-end delay or end-to-end jitter. We will show a different solution for end-to-end delay QoS support.

Only a simple modification of Chord is necessary to enable the creation of QoS aware trees. Nodes in Chord will have to be ordered by their QoS classes. This means that on the Chord ring, we will have clockwise monotonically decreasing QoS classes. The higher the ID of a node the lower is the QoS class of that node (low QoS class = low QoS). We use a core based tree with the root of the multicast tree as the node with the smallest Chord ID. All multicast messages will then be routed using the forwarder driven multicast approach. A multicast message will always be forwarded in clockwise direction on the Chord ring. Therefore, it will always be sent to a node having a higher ID, hence having the same or a lower QoS class. An example with 3 QoS classes is shown

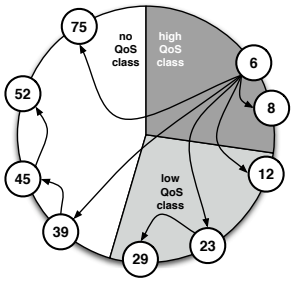


Fig. 1. QoS Support for Chord Multicast

in Fig. 1. The Chord ID space is split into 3 partitions. The partition holding the lowest IDs of Chord is reserved for nodes that require high QoS. The next partition holds nodes with low QoS requirements. Finally, nodes that do not require any QoS are in the partition holding the highest part of Chord IDs. Since the multicast root is the node with the lowest ID, multicast data dissemination holds the QoS path properties described before. All paths from the root to the leaf nodes have monotonically decreasing QoS requirements. Further details about this approach are explained in [2].

In order to support QoS using Chord, we had to make some adjustments and enhancements of the original Chord protocol. By default, Chord is a well functioning, structured P2P network that is highly scalable. But, when fast changes occur in a Chord network (e.g., multiple leaves and joins), multicast reliability becomes a problem. Since some nodes might not yet have updated their finger tables and successor lists, some nodes might not be served with multicast messages. Hence, we made optimizations for robustness and reliability:

- **Improved stabilization:** Modification of pointers (fingers or successors) causes execution of stabilization directly instead of only periodically. Hence, errors in finger and successor tables (invalid pointers) are corrected earlier.
- **Predecessor self discovery:** Nodes are able to search and find predecessors by themselves. Stabilization can now rely on nodes having up-to-date correct predecessors.
- **Self error correction:** Nodes can add fingers as successors if they do not find successors (repairs a broken ring).
- **Improved finger table:** Fingers can change their position in the finger table. Fingers are only deleted if their IDs exceed the nodes' current Chord ID + $2^{(fingerindex)}$.
- **No duplicate fingers:** More distinct fingers allow to have a more balanced distribution of the multicast tree.
- **Backward fingers:** Each node maintains a list of nodes, that have an entry in their finger table pointing to it. This helps to eliminate multicast errors.
- **Multicast optimizations:** We limited the multicast fan-out of nodes to have an upper bound for the maximum fan-out and to avoid that too many nodes directly connect to the root for end-to-end delay QoS support.
- **Improved Multicast routing:** Multicast fan-out limitations cause nodes to select receivers from their finger list (leads to better tree distribution and decreased hop count).

We required those optimizations in order to support QoS by having a more robust/reliable Chord version. Besides offering QoS using the class construct as described in [1], [2], we also wanted to support guarantees for the RTT between a

multicast receiving node and the multicast root node. Nodes can have a certain constraint regarding this so called *node to root RTT* for multicast messages. Therefore, they would only connect to a parent node that would support those constraints. When using Chord's default multicast mechanism (*forwarder driven multicast*), a node is not able to select its own parent for multicast delivery that would match its node to root RTT constraint. Therefore, supporting node to root RTT only works with the *receiver driven multicast* approach, where children can explicitly chose their multicast parents. In order to support receiver driven multicast, nodes need to know a few nodes that could act as their multicast parents and then select one of those as their actual parent. This can be supported by introducing backward fingers. A backward finger of node X points to node Y, which has X in its finger or successor table.

To reduce the overall multicast fan-out of nodes, we limited the maximum fan-out of a node to 7. This also helps to avoid that all nodes try to select the root as their parent, which is often the best candidate to fulfill node to root RTT constraints of nodes. Therefore, a potential parent can reject a child's request if it exceeds a certain number of children. As a consequence, it may take some time to find a parent that satisfies the node to root RTT and that can still accept new children. During this time, a node is not able to receive multicast messages.

II. Evaluation

A. Simulation Setup and Parameters

To evaluate our approach, we implemented the optimized Chord protocol in the OMNeT++ [3] simulator. We look at different scenarios. First, we compare forwarder driven multicast performance using our enhanced Chord protocol with and without QoS class support. We are assuming to have static hard QoS guarantees offered by the underlying network. This can be achieved using e.g. DiffServ or QoS provided by an approach proposed by EuQoS [4]. Then, we analyze the receiver driven multicast approach, where we can support also delay guarantees.

We used 13 distance matrices, which define the latencies between each possible pair of nodes. The generation and properties of those matrices are presented in [5].

The scenarios were evaluated using various network sizes. The different networks had a node count from 100 to 2000 in steps of 100. Each node step was evaluated using the 13 different distance matrices. Additionally, we used three different random seeds for each matrix and node count combination. Those random seeds influence the arrival time, departure time and other random based decisions and values. This then leads to a total of 780 simulation runs per scenario. We removed 1% of the outliers (0.5% of the min. and max. values each) for all runs. We interpreted different values presented in Table I.

TABLE I
Values Evaluated in the Simulation Scenarios

Multicast Hop Count	number of hops required to reach the root of the multicast tree.
Node to Root RTT	RTT from a node to the root of the multicast tree.
Multicast Fan-Out	number of nodes that a multicast parent has to serve with multicast data.
Percentage of Node to Root QoS Fulfilled	percentage of paths that fulfill QoS class requirements. Hence, these paths hold the QoS path properties.
Percentage of Node to root RTT Constraints Fulfilled	percentage of nodes for which Chord satisfies given node to root RTT.

B. Forwarder Driven Multicast with and without QoS Support

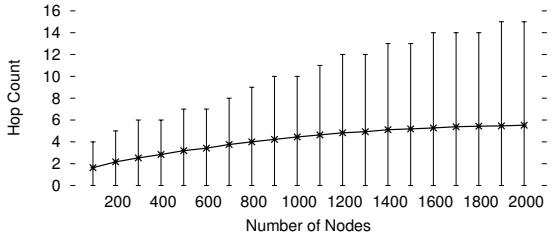
We first compare the forwarder driven multicast approach of our enhanced Chord protocol without QoS and with QoS support enabled. Figure 2 presents the multicast hop count and node to root RTT. There is not a significant difference between the multicast hop count for Chord without QoS as in Fig. 2(a) and for Chord with QoS enabled as shown in Fig. 2(b). The average of hops is between 2 to 6 and the maximum raises from 4 to 15. The node to root RTT correlates with the multicast hop count. In Figures 2(c) and 2(d), the average is between 50 to 150 ms. The maximum starts at 150 ms and increases up to 450 ms. As a conclusion, modifying the ID assignment method to enable QoS does not change Chord regarding multicast hop count and node to root RTT.

Finally, we discuss the percentage of node to root QoS fulfilled and percentage of node to root RTT fulfilled in Fig. 3. Nodes assign themselves a QoS class from the range 0–255. We check how many paths from the root to all nodes hold the previously described QoS path properties. Figure 3(a) presents the results for normal Chord using random ID assignment, i.e., QoS classes are not taken into account when node IDs are assigned. Here, only 15% to 40% of the paths hold the QoS path properties. Since at some times only a few nodes could remain in a Chord network, there can be moments where all paths or no paths at all fulfill the QoS path properties. On the other hand, with QoS enabled Chord, 100% of the paths fulfill the QoS path properties. This is shown in Fig. 3(b).

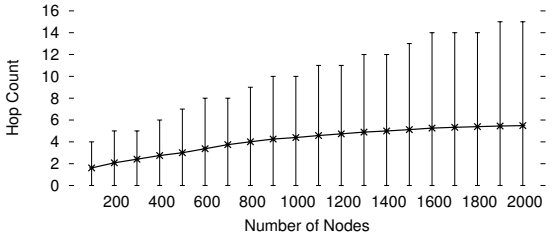
Nodes also assign themselves a node to root RTT constraint from the range of 100–200ms. The forwarder driven multicast approach does not offer a mechanism to fulfill those constraints. Therefore, as shown in Figures 3(c) and 3(d), paths from the root to leaf nodes do not always satisfy the constraints. In small networks, the constraints are easily met, but with larger networks and increasing hop count, the average value goes down below 60%. There is no significant difference between the QoS aware Chord and the Chord without any QoS support. Since sometimes only a few nodes could remain in a Chord network, all paths or no paths at all fulfill the node to root RTT constraints.

C. Receiver Driven Multicast in Chord

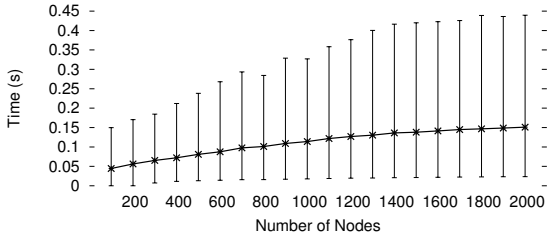
In this section, we evaluate the receiver driven multicast approach for Chord. Using this approach, we take not only the QoS class mechanism into account when joining Chord, but also the node to root RTT constraints when looking for a multicast parent. These node to root RTT constraints of nodes range from 100–200ms. We determined this range by analyzing the average overall hop count ($\sim$ 4 hops) presented



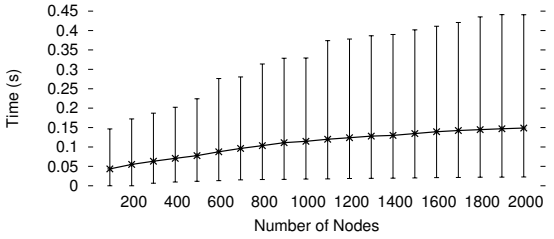
(a) Average Multicast Hop Count, no QoS



(b) Average Multicast Hop Count, with QoS

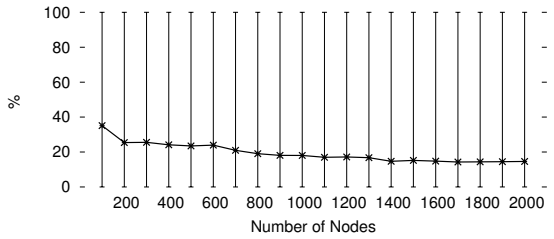


(c) Average Node to Root RTT, no QoS

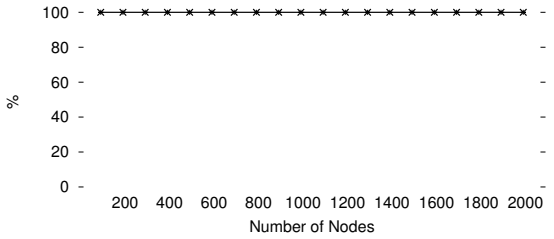


(d) Average Node to Root RTT, with QoS

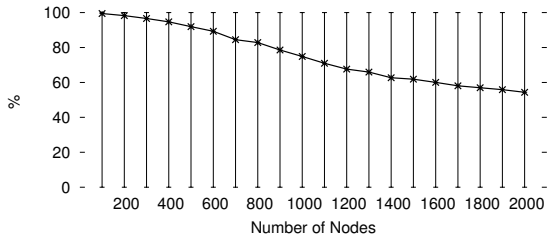
Fig. 2. Comparing Chord with and without QoS regarding Multicast Hop Count and Node to Root RTT



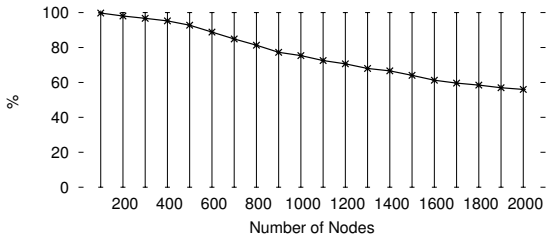
(a) Average Percentage of Node to Root QoS Fulfilled, no QoS



(b) Average Percentage of Node to Root QoS Fulfilled, with QoS



(c) Average Percentage of Node to Root RTT Fulfilled, no QoS



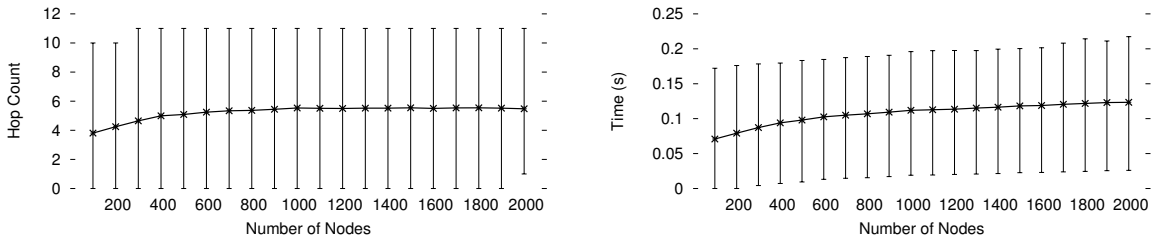
(d) Average Percentage of Node to Root RTT Fulfilled, with QoS

Fig. 3. Comparing Chord with and without QoS regarding Percentage of Node to Root QoS and Node to Root RTT Fulfilled

in Fig. 2(a) in relation to the average RTT ($\sim$ 25ms) between nodes resulting from the distance matrices.

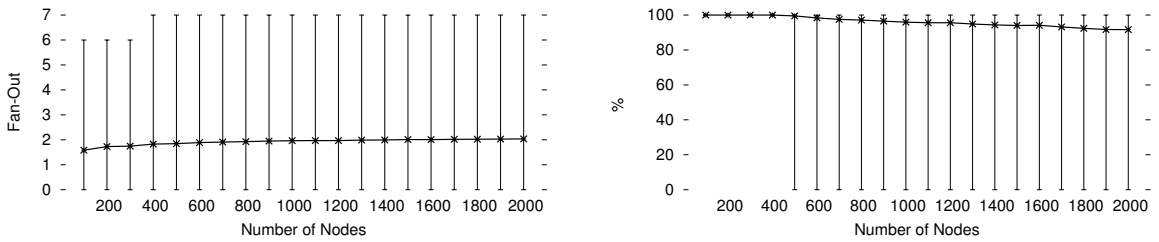
Figure 4 shows the multicast hop count and node to root RTT for receiver driven multicast in Chord. The average multicast hop count presented in Fig. 4(a) is between 4 and 5. This is slightly higher than for forwarder driven multicast as presented in Figures 2(a) and 2(b). In the forwarder driven multicast approach, the root or a node with a low ID can already reach nodes at the upper end of the ID space via one hop. This is due to the multicast forwarding mechanism using the finger table to determine children nodes for a parent. But, this is not the case for receiver driven multicast, where the child determines and selects its parent. Here, a child node tries to find a potential parent in its ID neighborhood, from the range $[\frac{nodeID}{2}, nodeID]$. Therefore, the hops from one node to another are normally smaller in terms of ID space coverage. On the other hand, this has a positive impact on the maximum multicast hop count value. In Figures 2(a) and 2(b) compared to Fig. 4(a), the hop count starts at a larger value but grows more slowly and only up to 11 hops. This means that the receiver driven multicast approach scales better in terms of multicast hop count than forwarder driven multicast. The minimum multicast hop count value of 1 with 2000 nodes is solely due to the outlier removal.

The node to root RTT results for receiver driven multicast are shown in Fig. 4(b). The forwarder driven multicast approach results were shown in Figures 2(c) and 2(d). Comparing them with the receiver driven multicast approach shows that the



(a) Average Multicast Hop Count (b) Average Node to Root RTT

Fig. 4. Multicast Hop Count and Node to Root RTT in Chord with Receiver Driven Multicast



(a) Average Multicast Fan-Out (b) Average Percentage of Node to Root RTT Fulfilled

Fig. 5. Multicast Fan-Out and Node to Root RTT Fulfilled in Chord with Receiver Driven Multicast

average node to root RTT starts at a higher value but grows more slowly. It is only in the range of 70–125ms compared to 50–150ms for the forwarder driven multicast approach. Also the maximum node to root RTT is lower and grows very slowly. This behavior is of course due to the fact that we take the node to root RTT constraints of nodes into account when they look for a multicast parent. The maximum node to root RTT value starts below the upper boundary of the constraints range, which is between 100–200ms. With more nodes in the network, the maximum though exceeds the upper boundary of 200ms. But, the receiver driven multicast approach is still more scalable in terms of node to root RTT value than the forwarder driven multicast approach.

The multicast fan-out and percentage of node to root RTT fulfilled for the receiver driven multicast approach of Chord are shown in Fig. 5. The average multicast fan-out for receiver driven multicast is almost constant at 2 as shown in Fig. 5(a). The maximum multicast fan-out is constant at 7 for networks with more than 400 nodes. There is no significant difference to the behavior of the multicast fan-out for forwarder driven multicast with and without QoS. Therefore, those results are not presented additionally.

The percentage of node to root RTT fulfilled for receiver driven multicast is presented in Fig. 5(b). The receiver driven multicast approach performs much better than the forwarder driven multicast approach presented in Figures 3(c) and 3(d). For network sizes up to 1200 nodes, the average percentage is above 95% for the receiver driven multicast approach. Afterwards, it falls down just slightly below 92% for up to 2000 nodes. This is significantly better than using the forwarder driven multicast approach, where the average percentage falls below 68% for 1200 nodes and below 55% for 2000 nodes.

III. Conclusion

In this paper, we presented mechanisms to enable Quality of Service (QoS) support for multicast in Chord. All our optimizations considered, the forwarder driven multicast approach in Chord can provide a very well distributed multicast tree. The modified Chord is very reliable (100% received multicast messages). It is also very scalable by putting a multicast fan-out limit of 7. Generally, it behaves very robust, even for constant rejoins. The overall latencies are quite good due to a reduced hop count caused by a well balanced tree. Multicast trees built using the receiver driven multicast approach can provide node to root RTT guarantees. Almost all paths fulfill the node to root RTT constraints of nodes for small and medium sized networks. For large networks up to 2000 nodes, a very high percentage (92%–100%) of the paths still fulfill the constraints. This receiver driven multicast approach also reacts to RTT changes of nodes over time and adapts and rearranges the multicast tree to again fulfill the constraints. Generally, the receiver driven multicast approach performs and scales even better than the forwarder driven multicast approach.

References

[1] M. Brogle, D. Milic, and T. Braun, "QoS enabled multicast for structured P2P networks," in *Workshop on P2P Multicasting, 4th IEEE Consumer Communications and Networking Conference*. IEEE, January 2007.

[2] ——, "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: IEEE, December 2008.

[3] Website, "OMNeT++, online: http://www.omnetpp.org," 2009.

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

[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.