Decentralized topology aggregation for QoS estimation in large overlay networks

Stefan Wieser
Institute of Information Technology (ITEC)
Klagenfurt University
Klagenfurt, Austria
swieser@itec.aau.at

Laszlo Böszörmenyi
Institute of Information Technology (ITEC)
Klagenfurt University
Klagenfurt, Austria
laszlo@itec.aau.at

***Abstract*—This paper introduces a scalable approach for efficient, low-cost multi-level Quality of Service (QoS) estimation in large overlay networks (ON). We modify an existing distributed partitioning algorithm [1], and use it to create "QoS maps". QoS maps empower applications to quickly predict several QoS metrics for any given route, and to obtain multiple alternative routes to any target node in the ON. We show that our modifications of the partitioning algorithm permit the aggregation of large hubs, but still preserve the sub-linear runtime of the original heuristic. Simulations with large ONs are performed to evaluate the proposed approach and demonstrate its scalability. Finally, we outline our estimation algorithm that we use to predict QoS and perform QoS aware routing in any given ON.**

***Keywords*-overlay networks;QoS estimation;hierarchical aggregation**

I. Introduction

While ONs have received a lot of attention both in practice and in research, routing and QoS prediction in large ONs remain difficult problems [2]. Our goal is to provide applications in existing large ONs with so-called "QoS maps" that allow low-cost estimates of the QoS experienced on different paths, and that provide multiple alternative paths to any target node.

To perform efficient, QoS-aware routing in ONs that are too large to have a global view, we simplify their topologies using hierarchical aggregation. This aggregation, together with the aggregated QoS information stored throughout the network is what we refer to as a "QoS map". In contrast to other overlays that create a new hierarchical topology to disseminate data, our QoS maps preserve the *existing* topology of the underlying network.

The main contributions of this paper are as follows: First, it explains how an existing distributed partitioning algorithm should be changed, so that it partitions any ON into sub-networks ("clusters") that can be used for distributed hierarchical aggregation. Second, it introduces three different methods to deal with hubs - nodes with a lot of neighbors - during partitioning, and evaluates their effects on the aggregated topology. It shows that with our modifications, large ONs can be aggregated into low-depth hierarchies with clusters of a small, bounded size. As a result, nodes only need to store little state about the ON to allow QoS-aware routing. Finally, it outlines our algorithm to perform multi-level QoS estimation using the aggregated information.

II. Related Work

A number of approaches attempt to solve the problem of scalability in large networks. Hybrid ONs and ONs that build distribution trees are popular choices for multicast data dissemination. They differ from our approach in that they either cannot provide alternative routes and QoS estimates to applications, or rely on the underlying network for routing entirely. NICE [3] also builds a hierarchy with bounded cluster size; however, it is optimized against a certain metric, such as delay, and builds a *new* ON based on it. In contrast, our approach preserves the structure of the underlying network, permits scalable routing and multi-level QoS estimates based on any QoS metric known to the network. Pheromones-based approaches mimic the behavior of ants foraging food, however scalability of traffic caused by simulated ants is not trivial and requires careful optimization [4]. Furthermore, despite these optimizations, they offer no direct support for QoS estimation.

To increase scalability in large ONs, topology aggregation (TA) is commonly used [5]. By aggregating a topology consisting of several nodes into a new, more compact topology, less data is required to get a high level overview of the whole. As the complexity of the topology is reduced, more expensive routing algorithms become feasible. QoS epitomes are used to approximate the QoS of all paths through such an aggregated topology. The various approaches are summarized in [6]. While we assume that QoS epitomes are used to approximate the QoS of the clusters we aggregate, we do not limit ourselves to a specific method.

To aggregate an ON, it has to be split into distinct partitions first. Heuristics are used, as graph partitioning is known to be an NP-complete problem [7]. However, many approaches, assume a global view for at least the initial partition step [8], use central components [9], or do not bound the number of nodes in a cluster [10], [11], and thus do not scale for a large dynamic environment. In addition, common k-way partitions are not practical for creating an aggregation, as the number of clusters is fixed to k, and cluster size - a variable - may turn out to be very large.

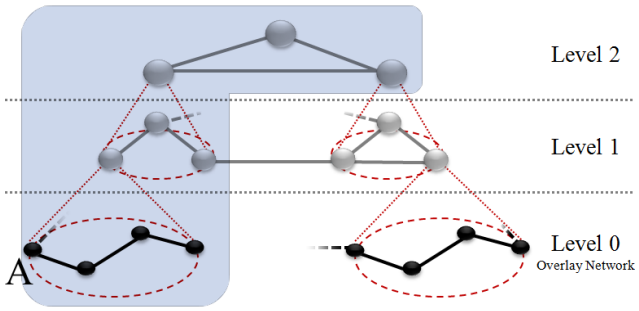

Figure 1. A three level hierarchy with $s = 4$. Parts of the ON and the aggregated topology have been omitted for clarity, and links to them are represented as dashed lines. Level 0 contains the original ON, and is partitioned into several clusters. Each cluster is represented by a virtual node at level 1. This process is repeated, and at level 2, the entire network is represented by only three clusters, making the structure of the network easy to grasp. The shaded area illustrates node A's view of the network.

Finally, as the number of nodes in an ON may not be known in advance, choosing a fitting k can be difficult. We therefore follow a non-sequential and fully distributed variant of the Basic Partition algorithm proposed by [1], which produces partitions with *bounded radius*. The heuristic provided in [1] has low runtime and message complexity, however, naïvely applying it repeatedly to create a hierarchy results in topologies with only trivial (single-node) partitions that cannot be aggregated any further. We make several modifications, which are described in this paper.

III. Hierarchy Construction

Consider an existing large ON of n nodes with unique identifiers. For scalability, we aggregate this network into a hierarchy of h levels: The lowest hierarchy level, level 0, is the original ON and therefore contains all n nodes with no aggregation. This level is then partitioned into several clusters based on the nodes' locality in the original ON, while bounding the number of nodes per cluster with the constant s. The identifier of a cluster is the largest identifier of the nodes it contains. Each cluster on level l is represented as a single (virtual) node on level $l + 1$. A virtual node uses the identifier of the cluster it represents as its own identifier. Connections between nodes of different clusters on level l also exist on level $l+1$ between the virtual nodes of both clusters. We continue building hierarchy levels until the uppermost hierarchy level consists of at most s clusters. Figure 1 shows this concept with a hierarchy of three levels.

A. Simulation Setup

We evaluate our changes through simulations on large random graphs with varying density. During the simulation, nodes exchange messages to partition the network and aggregate the topology. The simulation advances in discrete steps. At each step, every node may read all messages it has received, and may send messages to its *directly connected neighbors*. This is reasonable, as the average time to parse and respond to a message is negligible [12]. Note that the simulation steps directly relate to the expected runtime on "real" networks if multiplied by the average packet propagation and queuing delay on the network. For example, aggregating a dense network with $3,000$ nodes takes $2,503 \pm 58.3$ simulation steps. If we assume an average propagation and queuing delay of $50ms$, the aggregation takes an average of 125.15 ± 2.92 sec.

B. Distributed Partitioning

To aggregate the topology of the ON, we first partition it: For this, we build up on the partitioning algorithm described in [1], which partitions the network by building clusters. Each node (a potential "cluster head") attempts to add surrounding nodes with increasing hop-count ("layers") to its own cluster. The i^{th} layer consists of all nodes with a distance of i hops from the cluster head that do not already belong to a "finished" cluster. Whenever one node attempts to add another, the node with the highest identifier prevails. A stopping condition is used to decide when to stop adding layers: The original heuristic stops growing a cluster if the ratio between cluster members and neighbors exceeds a threshold that is determined by a configurable constant and the total number of nodes in the network. If this stopping condition is fulfilled after adding a new layer, that layer is dropped again, and the cluster is declared to be "finished". A cluster is also finished if no more nodes are found that could be added. Finished clusters no longer participate in the algorithm. Once all clusters are finished and the number of clusters in the uppermost hierarchy level is not larger than s, the simulation is considered complete.

C. Modifications to the Heuristic

Several changes to the partitioning algorithm were needed, to make it suitable for our QoS maps:

1) Stopping Condition: The original algorithm [1] bounds the number of inter-cluster edges, dependent on the network size. This requires knowledge of the total number of nodes in the network graph, which is costly to derive accurately for large ONs. We evaluated several stopping conditions in [12], and found that bounding the number of inter-cluster edges e in addition to the cluster size s works well. In our experiments, we set s to 10, as it offers a good compromise between runtime overhead and hierarchy depth, and e to $s - 1$, to decreases the likelihood of aggregating nodes into hubs with more than $s - 1$ neighbor clusters. Because imposing a hard limit on inter-cluster edges made it impossible to aggregate highly connected networks, we always allow cluster heads to add their first layer, even if the number of inter-cluster edges exceeds e.

2) Multiple Hierarchy Levels: As the algorithm runs in parallel, without explicit synchronization, no global knowledge of when the current hierarchy level is fully partitioned

into clusters exists[1]. Any finished cluster immediately begins running the partitioning algorithm in the next highest hierarchy level. Therefore, we must consider that different parts of the ON may perform aggregation on different hierarchy levels at the same time. We include each node's current hierarchy level in every message it sends. A node will not execute the partitioning algorithm until all its neighbors are at least the same hierarchy level as itself. Any node with a neighbor with a higher hierarchy level considers it to be part of a finished cluster. These changes do not affect the correctness of the algorithm in [1].

3) Aggregation of Large Hubs: Hubs with more than $s-1$ neighbors pose a problem for the original heuristic, as it partitions the network by adding layers of surrounding nodes to the partition of a potential cluster head. If the first layer exceeds the partition size bound, it is dropped again, which results in a partition with only a single node. Once the entire network consists of such hubs, no further aggregation is possible. Hubs are very common in ONs, and frequently occur in higher hierarchy levels during aggregation. We considered several approaches to deal with them. First, we permitted adding at least one layer, even if it exceeded the desired cluster size s. We refer to this as Naïve Aggregation (NA). Figures 2 and 3 show this simple approach resulted in the lowest runtime and hierarchy depth, however, clusters are no longer bounded. On our ONs with $5,000$ nodes and a target cluster size of 10, the average size of the clusters on the topmost hierarchy level is 170.17 ± 20.624 nodes. As excessively large cluster sizes defeat the purpose of aggregation, we do not consider NA suitable for QoS maps.

We approach this new problem in two steps. The original algorithm either adds all nodes of a layer, or none. We change the algorithm to perform "Partial Explorations" (PE): If a node is surrounded by more than $s-1$ neighbors, the cluster head randomly accepts $s-1$ nodes into the cluster. The remaining nodes are dropped and the cluster is switched to a finished state. Figure 2 shows that the runtime of the algorithm increases. This is expected, as excess nodes are dropped and continue to participate in the algorithm. NA would have accepted these nodes, creating very large clusters that are unbounded. In the worst case, NA aggregates the entire network into a single cluster with n nodes. In contrast, PE bounds the size of each cluster at s nodes. Furthermore, despite the increase, PE retains the sub-linear runtime of the original algorithm. As an upper bound on the cluster size is crucial for scalability, we argue that this trade-off is worthwhile. The main problem of PE is seen in Figure 3: The hierarchy depth increases sharply, which is again caused by large hubs: As the number of a hub's neighbors is only reduced by up to $s-1$ nodes at each hierarchy level, hubs with many neighbors cause a significant increase in depth.

[1]An exception to this is the uppermost hierarchy level. As all nodes are in the same cluster, they observe that their cluster has no inter-cluster edges, which indicates that the hierarchy has fully been built.

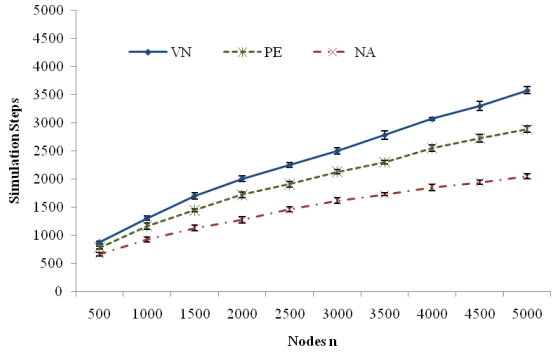

Figure 2. Runtime on random dense ONs; different numbers of nodes.

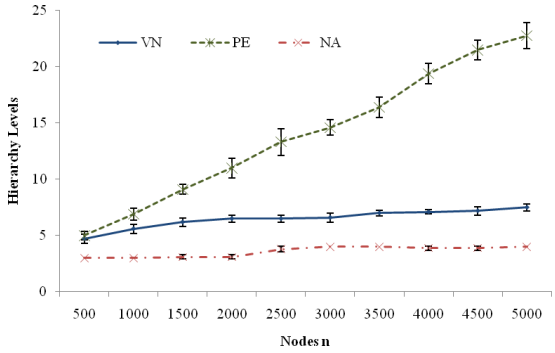

Figure 3. Hierarchy depth on random dense ON; different numbers of nodes.

We therefore introduce Virtual Nodes (VNs): Once a cluster completes a partial exploration, the cluster head splits into the finished cluster, and a virtual node. The virtual node is connected to the cluster, as well as all remaining neighbors, and continues to participate in the algorithm. With VNs, hubs are broken up effectively within a single hierarchy level. While this causes additional load on the hub, we argue that if such large hubs exist in the network or the hierarchy, they are likely sufficiently powerful to handle the additional load. Figure 2 shows the runtime increases over PE and NA. This is again expected, as a hub splitting into two nodes effectively introduces new (albeit virtual) nodes to the network. On the other hand, Figure 3 shows that VNs decrease the hierarchy depth once more. While the runtime increased over the original heuristic, it remains sub-linear. In addition, all large clusters have been removed, and the hierarchy depths are only one to two levels deeper than the theoretical optimum. We therefore consider the runtime increase of VN a reasonable trade-off for our QoS maps.

IV. QoS Estimation

Once the hierarchy is created, it can be used for efficient routing and QoS estimation in the original ON. In this section we outline our approach for QoS estimation. The detailed algorithm with a complexity of $O(\log^2 n)$, and its evaluation are omitted due to space limitations, and

can be found in our technical report [12]. Whenever a cluster is finalized, QoS epitomes of any desired QoS metric are created. They approximate the QoS experienced for traversing that cluster from any entry node to any exit node, which reduces the amount of information that needs to be advertised to other clusters. Note that any cluster in our QoS map may be the target cluster of a route, in which case they do not have an exit node. We therefore compute the worst QoS metrics from each entry point to any node in the cluster and include it in the epitome. Obtaining this information has little overhead as the cluster size is bounded.

A node that wishes to predict the QoS experienced on a path to a target node first locates the highest hierarchy level that contains both its own cluster and the cluster of the target. Recall that every node has a local view of the topology, which contains the topology of its own cluster, and all its parent clusters. For example, Figure 1 shows the local view of node "A". Therefore, using its local view, it can now find possible routes to the cluster of the target node on that hierarchy level, utilizing any common routing algorithm.

Each of these routes contains up to three types of clusters: a source cluster, a target cluster and zero or more transit clusters. Note that for the trivial case, in which both nodes share the same cluster at level 0, source and target clusters may be identical. These cluster types differ in the amount of local information the source node has, and are treated differently for QoS prediction purposes: *Transit clusters* are traversed on route to the target node. As QoS epitomes for each cluster contain the aggregated QoS from each entry to each exit point, they can be used directly to get an estimate of the cost of traversing them. *The target cluster* is the least accurate cluster type. As they do not have an exit point, it would be necessary to have a view of the target cluster in order to provide a precise estimation. Without said view, the worst case QoS from its entry node to any node in the cluster has to be assumed. *The source cluster* can be broken down to the next lowest hierarchy level to use less aggregated QoS epitomes and so increase the quality of the prediction. We set the new target node to any exit point we consider and apply the algorithm recursively, until we reach the unaggregated ON of our own cluster at level 0. After the QoS epitomes for each cluster are combined, the source node now has a coarse estimate of the QoS for each path without consuming any network resources. The accuracy of this estimate can be improved in two ways: A cooperative target node can estimate the QoS from each entry point of its cluster to itself as described before, and transmit that information to the source node. In addition, with the cooperation from nodes of any transit cluster, its less aggregated next-lowest hierarchy level can be considered in place of the QoS epitome.

Note that the topology of the ON is aggregated without considering QoS. Therefore, a cluster may contains paths and nodes with vastly different quality. This is intended: QoS maps aim at giving a view of the landscape of the given ON. We do *not* intend to change the landscape itself. By comparing the range of QoS metrics in the QoS epitomes, they can be used to identify poor topology-awareness in the underlying ON and thus help to find a better one. In any case, our QoS maps allow QoS estimation regardless of how well or poorly the underlay is organized.

V. Conclusion and Future Work

This paper introduced QoS maps, a decentralized hierarchical approach to perform scalable QoS estimation in large ONs. We have shown that by modifying a decentralized version of the Basic Partition algorithm [1], we are able to aggregate the network topology into a hierarchy with small cluster sizes and low depth. Finally, we outlined a multi-level QoS estimation algorithm that uses QoS maps to predict QoS on any path between two nodes starting using only local information. Future work will address churn in the ON: In a first experiment, we follow a similar approach to [3] and perform small, localized changes to the hierarchy, merging sparse and splitting large clusters as necessary. However, a detailed evaluation of this approach for our QoS maps is subject of future research.

References

[1] B. Derbel, M. Mosbah, and A. Zemmari, "Sublinear fully distributed partition with applications," *Theory Comput. Syst.*, vol. 47, no. 2, pp. 368–404, 2010.

[2] R. Hou, K.-S. Lui, K.-C. Leung, and F. Baker, "Routing with QoS information aggregation in hierarchical networks," in *17th Int. Workshop on Quality of Service*, July 2009, pp. 1–9.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *SIGCOMM Computer Communication Review*, vol. 32, pp. 205–217, August 2002.

[4] S. S. Aman, M. R. Akbarzadeh-Totonchi, and M. Naghib zadeh, "A novel approach to distributed routing by Super-AntNet," in *Proc. IEEE CEC 2008*, 2008, pp. 2151–2157.

[5] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster, "Analysis of topology aggregation techniques for QoS routing," *ACM Comput. Surveys*, vol. 39, no. 3, 2007.

[6] W. Y. Tam, K. S. Lui, S. Uludag, and K. Nahrstedt, "Quality-of-service routing with path information aggregation," *Comput. Networks*, vol. 51, no. 12, pp. 3574–3594, 2007.

[7] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified NP-complete problems," in *Proc. 6th Annual ACM Symposium STOC '74*. New York, NY, USA: ACM, 1974, pp. 47–63.

[8] F. Pellegrini and J.-H. Her, "Efficient and scalable parallel graph partitioning," in *5th Int. Workshop PMAA'08*, Neuchâtel Suisse, 2008.

[9] D. G. Thaler and C. V. Ravishankar, "Distributed top-down hierarchy construction," in *Proc. IEEE INFOCOM '98*, 1998, pp. 693–701.

[10] S. Basagni, "Distributed clustering for ad hoc networks," in *Proc. 4th Int. Symposium on Parallel Architectures, Algorithms, and Networks, I-SPAN '99*, 1999, pp. 310–315.

[11] Y. Wan, S. Roy, A. Saberi, and B. Lesieutre, "A stochastic automaton-based algorithm for flexible and distributed network partitioning," in *Proc. SIS 2005*, 2005, pp. 273–280.

[12] S. Wieser and L. Boeszoermenyi, "Self-organizing topology aggregation for QoS maps and routing," Klagenfurt University, Tech. Rep. TR/ITEC/01/2.11, March 2011.