

QoS-Guaranteed Path Selection Algorithm for Service Composition

Manish Jain
Georgia Tech
jain@cc.gatech.edu

Puneet Sharma
HP Labs, Palo Alto
puneet.sharma@hp.com

Sujata Banerjee
HP Labs, Palo Alto
sujata.banerjee@hp.com

Overlay networks have been employed as a popular solution to work around rigid and often suboptimal IP routing to provide better performance. Examples of services/applications that can use overlay networking include multicasting, end-to-end QoS and secure overlay services [1], [2]. Service overlay networking (SON) is an emerging approach that builds on overlay networks to deliver advanced services to user. The main idea in the SON approach is that each overlay node can provide one or more basic services, referred to as *component-services*, in addition to application data forwarding service. These *component-services* act as building blocks for more advanced services, which can be created by combining these *component-services* in series or in parallel. Figure 1 illustrates the use of SON to provide media-content at user-specific rates by joining two component-services, *content* and *transcoder*, without requiring the content to be stored in multiple formats.

One of the main challenges in SON is to, for a given user request, find a path between a source and a destination passing through a specific set of overlay nodes such that the path will satisfy one or more QoS requirement specified by user. This set of nodes depend on the user request which determines the sequence of services (and therefore nodes) that must be selected to satisfy user request. During the last few years, there have been several proposals for service path selection in SON [3], [4], [5]. The main limitation in these approaches is that they aggregate multiple constraints using a linear function. Jaffe [6] showed that the use of a linear function to aggregate multiple constraints can not satisfy all the QoS constraints. In this work, we propose a heuristic algorithm, *K-Closest Pruning (KCP)*, to solve the problem of multi-constraint service path selection for SON in polynomial time. The main feature of this algorithm is that the path selected by this algorithm meets all the QoS requirements specified by the user/application.

We model SON as a set of N overlay nodes O_1, \dots, O_N and N^2 overlay links connecting the N nodes. Each overlay node is characterized by the maximum resource capacity and average resource utilization. Similarly, the overlay link between any node O_i and O_j is characterized by three metrics: loss rate, delay and available bandwidth. The distinct type of component-services available from N overlay nodes is denoted by S_1, \dots, S_M , where $M < N$, such that each node supports at most one component-service. An end-user request has two components: advanced service \mathcal{A} that can be mapped to a *service template* $S_J^1, S_K^2, \dots, S_T^n$, which specifies composition order of n services, and QoS constraints \mathcal{Q} in terms of end-to-

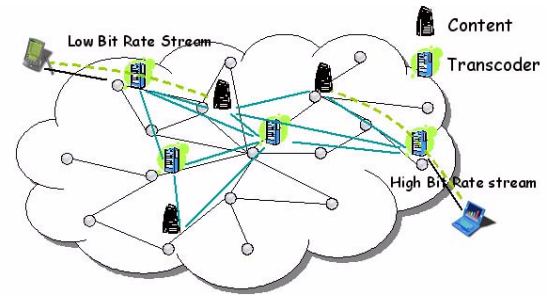


Fig. 1. Service Overlay Network (SON)

end loss, delay and available bandwidth. Then, our objective is to find a *service-path*, which is defined as sequence of nodes providing such that the request QoS constraints \mathcal{Q} are satisfied.

A service path is said to satisfy a user QoS constraint only if the aggregated(end-to-end) node/link properties is better than the QoS constraint. This aggregation could be additive such as link delays or max-min such as service path capacity. If any link or node property violates the corresponding QoS requirement, then the service path containing such link or node would certainly fail to meet the QoS requirement. Therefore, the basic step is to examine all the link and node properties and eliminate any links/nodes that do not meet all user specified QoS requirements.

Next, we describe our algorithm, referred to as *K-Closest Pruning (KCP)* algorithm, to find the service path. The main idea in our approach is to leverage network proximity information to reduce search space, i.e. to reduce the number of qualified overlay nodes/links, for service path selection for a given request. Suppose a user request $S_J^1, S_K^2, \dots, S_T^n$, with QoS constraints \mathcal{Q} , arrives at an overlay node O_r . The significance of the node O_r is that it must be the last node in any service path that is selected for this user request. There are two main steps in this algorithm: first is the *search space reduction* and second is the depth first search. The intuition in the *search space reduction* is that overlay nodes which are close to node O_r are more likely to be able to meet the delay constraints whereas the nodes that are distant from O_r are less likely to meet the delay constraints of user request.

More specifically, in the first step, we set O_r as the root, or level 0, of the tree. Then, we select K nodes, which are closest to O_r in terms of delay and can provide last service S_T^n from the service template. These K nodes form the level 1 of the

tree rooted at O_r . We, then, iteratively choose K -closest nodes providing service $S_X^{(n-i)}$, the $(n-i)^{th}$ service from service-template, for each node providing n^{th} service and put them at $n - (n-i) + 1$ level in the tree. This iterative process results in K -ary tree rooted at O_r . The leaf nodes, at level n , in the resulting tree provide the first service S_j^1 from the template. Any potential service path must include one node from each level in this tree.

As part of the second step, we use depth first search to find all potential service paths. The potential service paths must meet all the QoS requirements specified by user. This is ensured by performing the following steps. We start the tree traversal at root node O_r and proceed with tree traversal in depth first manner. Suppose that we are at an intermediate node O_i . We examine the path from O_i to O_r and compare the link/node properties with the user specified QoS constraints \mathcal{Q} . If the path from root till the node O_i meets all the QoS requirements, then we continue tree traversal by selecting the left first child of O_i . If O_i does not have any child or all the children have been visited, then go to the adjacent node of O_i . If the path from the root till O_i does not meet all of the QoS requirements, we cutoff the sub-tree below O_i and proceed to the adjacent node in the same level. When we reach the leaf node O_l in level n , we record the path from O_l to O_r if the path meets QoS requirement.

At the end of depth first search, the user is *admitted* if we have found at least one path. Otherwise, the user request is *rejected*. If more than one path is found, then the selection of service path is based on whether our objective is *load balancing* or *reuse*. If the objective is to have high degree of load balancing, then we select the path that has the minimum sum of node utilization among the candidate path. On the other hand, if our goal is to maximize reuse, then we select a path that has maximum number of nodes with active instances of services in service-template.

To evaluate the KCP algorithm, we have implemented a flow-level discrete-event simulator. We have used both a random overlay topology and Planet-lab traces [7] to construct overlay topology in our evaluation experiments. In our experiments, SON provides a total of 5 distinct services and each overlay node provides 1 out of these 5 services. End-user requests, with random duration and QoS requirements, are generated from a Poisson process with an average rate of 166 requests per sec.

We have also implemented the QSC algorithm, as described in [8], in our simulator to compare its performance with KCP. To evaluate the performance of KCP variants and QSC approach, we use the following five performance measures: *reject ratio*, *QoS violation rate*, *QoS violation degree*, *idle node ratio* and *node utilization*. First three metrics are more relevant to the performance perceived by user, and the last two metrics are indicators of service provider performance.

Figure 2 shows both the *reject ratio* and *QoS violation ratio* for KCP variants and QSC algorithms for increasing user request arrival rate. The main observation is that while KCP variants have non-zero reject ratio, the violation ratio is zero

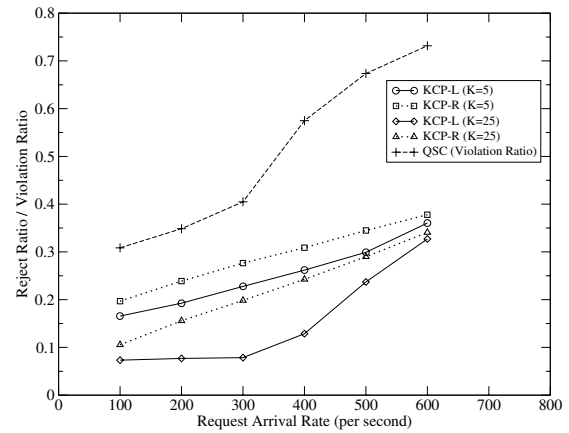


Fig. 2. Reject/Violation Ratio (-L: load-balancing variant, -R: reuse variant)

for them whereas the trend is reversed for QSC. This implies that KCP algorithm is able to guarantee QoS requirements of all admitted requests while keeping a relatively low reject ratio. This property of admission control in the algorithm may be useful to the network operators if the relative penalty incurred from failure to meet user QoS requirements outweighs the gains from keeping reject ratio equal to zero.

To summarize, we have presented a novel algorithm for path selection in service overlay network. Initial results based on simulation indicate that the algorithm achieves the following: first, performance of algorithm does not vary significantly for changing values of K ; second, the algorithm is able to guarantee multiple QoS requirements of a given user requests; third, algorithm rejects certain number of requests, however the reject ratio does not exceed 10% of the user request load. We have introduced two distinct objectives *reuse* and *load balancing* and demonstrated their use in modifying the path selection of KCP algorithm to achieve these objectives with affecting performance of KCP algorithm. In future, we plan to focus on more performance evaluation based on both current simulation based approach and experiments on Internet. Another open question that we plan to investigate is how to dynamically determine the optimal value of K for a network with arbitrary number of overlay nodes.

REFERENCES

- [1] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQoS: Offering QoS using Overlays," in *First Workshop on Hop Topics in Networks (HotNets-I)*, 2002.
- [2] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services," in *Proceedings ACM SIGCOMM*, 2002, pp. 61–72.
- [3] X. Gu, K. Nahrstedt, and B. Yu, "SpiderNet: An Integrated Peer-to-Peer Service Composition Framework," in *Proceedings of HPDC*, May 2004.
- [4] J. Liang and K. Nahrstedt, "Service composition for advanced multimedia applications," in *Proceedings of MMCN*, 2005.
- [5] B. Raman and R. Katz, "Load balancing and stability issues in algorithms for service composition," in *Proceedings of INFOCOM*, 2003.
- [6] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, pp. 95–116, 1984.
- [7] "S³: Scalable Sensing Service," <http://networking.hpl.hp.com/s-cube/>, Apr. 2006.
- [8] X. Gu, K. Nahrstedt, R. N. Chang, and C. Ward, "QoS-Assured Service Composition in Managed Service Overlay Network," in *Proceedings of ICDCS*, may 2003.