QoS-Guaranteed Path Selection Algorithm for Service Composition

Manish Jain
Telchemy Inc.
Duluth, USA
manish.jain@telchemy.com

Puneet Sharma
Hewlett-Packard Labs
Palo Alto, USA
puneet.sharma@hp.com

Sujata Banerjee
Hewlett-Packard Labs
Palo Alto, USA
sujata.banerjee@hp.com

ABSTRACT

Service overlay networking is an emerging approach, which employs overlay nodes to provide advanced services by dynamically *composing* it from basic services available on overlay nodes. Advanced service request from users can have different and multiple quality-of-service (QoS) requirements and finding a *service path* that meets these multiple requirements is an open problem. Also, network operators have operating requirements such as load-balancing to minimize hotspots and/or minimizing the overall utilization of resources in their network. In this work, we describe a novel algorithm K-Closest Pruning (KCP), based on proximity based tree pruning, to efficiently determine a *service path meeting all the QoS requirements*. An additional novel feature in this algorithm is that it incorporates the minimal resource utilization or load-balancing constraints into the path selection process. KCP algorithm achieves a polynomial running time and is the first, in our knowledge, to take both the QoS requirements (user/application perspective) and resource utilization (operator perspective) into account. We show that the KCP algorithm performs significantly better than previous solutions in terms of meeting the QoS requirements of user requests.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network communications

General Terms

Performance

Keywords

Overlay network, service composition

1. INTRODUCTION

The Internet continues to evolve without any mechanism to provide performance guarantees or explicit feedback about network performance to the applications. Overlay networks have been proposed as a work around to the limitations of the Internet architecture for enabling new applications and services. Examples of such proposals include multicasting [4], end-to-end QoS [19] and secure overlay services [12]. The primary usage of overlay networks, initially, had been limited to providing *data forwarding* for the applications/services. Consider the example of a video delivery service to a set of customers with diverse video playing capabilities and QoS requirements. Traditional approaches tackled the needs of these customers by storing the video in different formats at a content server and utilizing the overlay network for transmitting video along a path meeting the QoS requirements. An alternative architecture, which reduces storage requirement of stored video, involves using the overlay nodes for *transcoding*. The server would store the video only in one format and use the transcoding service, in real-time, to deliver it in the users' desired format. Overlay nodes with such additional capabilities form the basis of *Service Overlay Network* (SON) [2, 8].

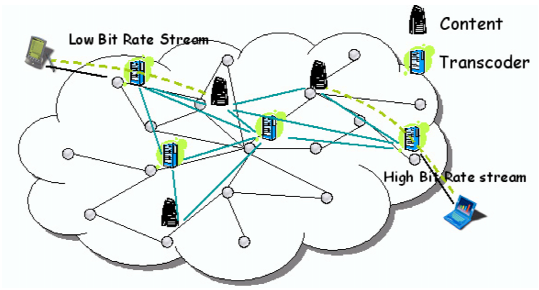

Figure 1: Service Overlay Network

Each overlay node in the SON provides one or more basic services, referred to as *component-services*, that act as a building block for the *advanced-service*. This process of combining multiple component-services to provide a advanced-service to the user is referred to as *service composition*. Figure 1 illustrates the use of SON to provide a advanced-service to the users by joining several component-services. Recently, service composition is becoming more and more prevalent with the growing popularity of the service mash-ups using web service 2.0 framework for web service composition. Such service composition has applications in e-commerce, science, education etc. *mappr.com*, *allconsuming.net* and *yahoo! pipes* are some examples of service mash-ups that string together web services from different providers.

QShine 2008, July 28-31, 2008, Hong Kong, Hong Kong.
 ISBN 978-963-9799-26-4
DOI 10.4108/ICST.QSHINE2008.3959

The efficient delivery of advanced service using service composition involves finding sequence of overlay nodes such that QoS requirements for the advanced service are met. This problem can be formulated, as described in § 3, as a path selection problem in overlay network with an additional constraint: the path between source and destination must pass through a specific set of overlay nodes. This set of nodes depends on the user request which determines the sequence of services (and therefore nodes) that must be selected to satisfy user request. Given the exponential complexity of path selection in large overlay networks, several heuristic based algorithms for path selection, using both single constraint [13] and multiple constraints [15], have been proposed. However, the existing solutions for overlay path selection are not suitable for path selection in SON due to the additional constraint. Furthermore, there is a need to incorporate the service providers requirements such as load-balancing and reuse of overlay nodes/component-service in the path selection process.

In this work, we propose a heuristic based algorithm, *K-Closest Pruning (KCP)*, to solve the problem of multi-constrained service path selection for SON in polynomial time by reducing the search space. The main feature of this algorithm is that the selected path meets all QoS requirements specified by the user/application. The path search algorithm gives precedence to the delay QoS parameter and constrains the search to the nodes in close proximity to the current node under consideration in the end-to-end path. In the KCP algorithm, the distance of the node from the user/application is used as an initial criteria to select nodes. As we show later, the proximity based pruning of the search space achieves the goal of reducing the running time of the algorithm without sacrificing the quality of the results. Our algorithm not only selects QoS-guaranteed path to meet user requirements but also enables service providers to control the load-balancing of resource usage across multiple replicas or greater service reuse.

The rest of the paper is organized as follows. § 2 presents the issues in the area of SON and previous approaches. § 3 describes the overlay network model, path selection constraints and formulates the path selection problem. § 4 presents our heuristic algorithm, KCP, for multi-constrained service path selection. § 5 presents the evaluation methodology, and performance results of KCP algorithm. Finally, we discuss open issues and summarize in § 6.

2. RELATED WORK

There are three main issues in the area of service composition using overlay networks and several researchers have looked at these problems. The first issue deals with the problem of *service/node placement*. Specifically, previous work focuses on how to dynamically choose the number and location of service/servers that will be able to satisfy the QoS requirements and/or provide better end-to-end performance than native routing [18, 5]. The second issue is that of *end-to-end network measurements* i.e. inferring performance on network paths using active or passive measurements from end hosts. Several tools and techniques have been developed for accurate networks measurements [10, 17].

The third question, which is the focus of this work, is the *path selection* problem in Service Overlay Network. It has been known for some time that path selection with multiple QoS constraints is an NP-complete problem [6, 21]. Hence, only heuristics algorithm with polynomial time complexity are considered feasible. This point of view has resulted in a large number of proposed heuristic based algorithm for path selection in SON during the last few years [7, 14, 16, 11].

Gu et al. [7] propose an algorithm, QoS-assured Service Composition (QSC) algorithm, similar to Dijkstra algorithm to find the best path to service user request. The authors propose a single metric to represent "cost" on the edge in the graph used by Dijkstra's algorithm. For each user request, the algorithm generates a candidate graph that is comprised of all the overlay nodes that provide any component-service from service template corresponding to user request. The weight of each edge in candidate graph is derived by aggregating the properties of corresponding edge and nodes in SON network. The path selection problem then involves finding the shortest path in the candidate graph. Choi et al. [3] have proposed a graph mapping approach where the actually topology is mapped to layered candidate graph. Each layer corresponds to a QoS constraint or a component service. Dijkstra's algorithm is then used to find the appropriate path in the candidate graph. Though, our approach also layers the overlay nodes according to the service request, we use proximity based pruning to reduce the search space.

Liang et al. [14] propose algorithms for path selection in SON, which also considers the case of "multicast" i.e. path selection in the case of multiple destinations. Raman et al. [16] focus on server load balancing and proposes a metric, least inverse available capacity, for path selection. The main limitation in these work is that they aggregate multiple constraints using linear function. Jaffe [9] showed that the use linear function to aggregate multiple constraints can not satisfy all the QoS constraints. Furthermore, there are no simple shortest path algorithms based on non-linear functions.

3. SERVICE OVERLAY ROUTING

In this section, we first describe the service overlay network model and introduce the terminology. Next, we present the various cost functions and QoS constraints involved in path selection and classify QoS constraints in to two classes, user based and service provider based. Finally, we formulate the service path selection problem in terms both user and service provider based QoS constraints.

3.1 Service Overlay Network Model

The traditional IP network comprises of routers and links, which form the network core, and end-systems, which are connected to the network core. End-systems are the sources and sinks of *data* on the network, and the network core provides best-effort datagram delivery service for this data. The *Service Overlay Network (SON)* is formed by a subset of end-systems, which are interconnected by overlay links. Overlay links are virtual links and are formed using IP tunnels comprising of multiple physical links over the IP network.

We model SON as a set of N overlay nodes $O_1, \ldots, O_N$ and N^2 overlay links. Each overlay node O_i has some resources including memory and processing capacity. We aggregate all these resources into a single metric and represent the maximum available resource (or node capacity) of overlay node O_i by R_i and its utilization by U_i. We characterize the overlay link between any node O_i and O_j by three metrics: loss rate $\rho(O_i, O_j)$, delay $D(O_i, O_j)$ and available

bandwidth $A(O_i, O_j)$. These three metrics depend on the state of the underlying IP links that constitute the virtual overlay link. We assume that the SON can infer or measure these network properties by using the tools that are available for e2e network measurements.

The N nodes in SON supports a set of M component-services $S_1, \ldots, S_M$, where $M < N$. Each service has multiple *replicas* i.e., each component-service is supported by several overlay nodes. We denote a replica of service S_i on overlay node O_p by O_p^i. End-user requests can arrive at any overlay node in the network. Each user request has two components: desired advanced service $\mathcal{A}$ and desired user QoS constraints $\mathcal{Q}$. The end-user request $\mathcal{A}$ is serviced as follows. First, the user request is mapped to a *service template*, denoted as $S_i, S_j, \ldots, S_t$. The service template specifies the sequence in which the component-services must be connected to compose the advanced service[1]. The second step is to map the service template corresponding to a sequence of replicas. The sequence of replicas that will be selected to service user request constitute the *service path* for the user request. Then, for service template $S_i, S_j, \ldots, S_t$, we represent service path as $O_p^i, O_q^j, \ldots, O_x^t$.

3.2 Path Selection Constraints

Selection of overlay nodes for the *service path* is subject to two constraints - user QoS constraints and service provider constraints. $\mathcal{Q}$ includes any constraints that may be specified directly by the user or be inherent to the desired services. We consider loss rate (L^{QOS}), available bandwidth (A^{QOS}), delay (D^{QOS}) and node utilization (U^{QOS}) as QoS constraints for the selection of nodes. Service provider constraints, on the other hand, enable the SON operator to tune the path selection by taking certain operational factors in to account. We focus on two such operational factors, which we deem important for most network operators: *load balancing* and second is *reuse*.

3.2.1 User QoS Constraints:

Delay and loss rate can be formulated as an additive constraint. $d_{q,z}$ denotes the e2e delay of a service path from node O_q to O_z.

$$d_{q,z} = D(O_q, O_r) + \cdots + D(O_y, O_z) \leq D^{QOS} \quad (1)$$

If $\rho_{q,z}$ denotes the e2e loss rate of a service path from node O_q to O_z, then it can be expressed as,

$$\rho_{q,z} = 1 - ((1 - \rho(O_q, O_r)) \times \cdots \times (1 - \rho(O_y, O_z))) \leq L^{QOS} \quad (2)$$

Available bandwidth and node utilization constraints can be expressed as follows:

$$A_{q,z} = min\{A(O_q, O_r), \ldots, A(O_y, O_z)\} \geq A^{QOS} \quad (3)$$

$$\forall O_i \text{ in the service path } : U(O_i) \leq U^{QOS} \quad (4)$$

3.2.2 Service Provider Constraints:

As discussed earlier, to service a user request, a sequence of overlay nodes must provide a component-service. Now, consider the case when two user requests of same type arrive. The two user requests can then be serviced in two ways: creating two service paths with mutually exclusive set of overlay nodes or using one service path to service both requests. These two approaches of servicing user requests illustrate two mutually exclusive objectives that path selection algorithm must satisfy: first is *load balancing* and second is *reuse*. The goal of load balancing would be to distribute load for a service S_i equally among its replicas. On the other hand, the goal of *reuse* is to maximize the utility of each

To model these two objectives, we must consider the cost of providing the component-service on an overlay node $C(O_p^i)$. Two possibilities must considered while taking the cost of component-service in to account: either the overlay node is currently not providing the component-service or it is. The *service cost* $C(O_p^i)$ in the former case is equal to $C(S_i)$ and in the latter case equals $\beta \times C(S_i)$. β is referred to as *service reuse factor*, which determines whether the service on the overlay node is being reused. $\beta = 0$ implies that an existing active instance of service can be used without increasing the node utilization. On the other hand, $\beta = 1$ implies that a new service instance needs to be created for each incoming request. In general, reusing an existing replica on overlay node to satisfy user request would imply the value of β between 0 and 1.

We present two metrics, *service path cost (SPC)* and *service path utilization (SPU)*, to select a path satisfying either reuse or load balancing constraint, respectively. The SPC for a potential service path (PSP), $\mathcal{M}_{SPC}$, is defined as the sum of all the service costs, for creating active replica instances, on the overlay nodes in the corresponding path.

$$\mathcal{M}_{SPC}(i) = C(O_f^i) + C(O_r^j) + \cdots + C(O_y^t) \quad (5)$$

$\mathcal{M}_{SPC}$ measures the effect of reuse along each node in the PSP. According to reuse objective, the most suitable service path P is the one with the maximum number of nodes doing reuse. Such a path would have minimal service path cost,

$$P = k^{th} \text{ PSP such that } k = \arg\min\{\mathcal{M}_{SPC}(k)\} \quad (6)$$

Similarly, we define SPU for i^{th} PSP, ($\mathcal{M}_{SPU}$), as the sum of node utilization of all overlay nodes in the corresponding path. This node utilization includes the increase in utilization due to creation of active replica.

$$\mathcal{M}_{SPU}(i) = U(O_f) + U(O_r) + \ldots U(O_y) \quad (7)$$

SPU quantifies the aggregate load in the overlay nodes of a service path and can be used to compare the distribution of load in the PSPs. For two paths P_a and P_b, a lower value of SPU for P_a relative to P_b indicates that the aggregate load is higher in P_b than in P_a. Therefore, P_a is more suitable according to load balancing constraint. In general, a path with minimal service path utilization among PSPs is the best candidate for service path,

$$P = k^{th} \text{ PSP such that } k = \arg\min\{\mathcal{M}_{SPU}(k)\} \quad (8)$$

3.3 Problem Statement

Given a set of N overlay nodes $O_1, \ldots, O_N$ and N^2 overlay links, where each node O_i is annotated with its utilization U_i and each overlay link between O_i and O_j is annotated with 3 metrics: loss rate $\rho(O_i, O_j)$, delay $D(O_i, O_j)$ and available bandwidth $A(O_i, O_j)$. Then, for a given user request and its associated QoS constraints, the goal is to find a service path that satisfies the constraints specified in equation 1, 2, 3, 4 and lastly, either 6 or 8.

[1]A one-to-one mapping from the user request to the service template is assumed.

4. PROPOSED PATH SELECTION ALGORITHMS

In this section, we describe a novel algorithm with proximity based tree pruning, K-Closest Pruning (KCP), to solve path selection problem described in the previous section. The key feature of our algorithm is that, unlike previous approaches mentioned in § 2, it does not aggregate multiple QoS constraints into single constraints. Additionally, KCP algorithm provides a *control knob* to either do load balancing or reuse. We also propose a simple modification to QSC algorithm, called QSC-modification or QSC-M.

4.1 K-Closest Pruning(KCP): Proximity based algorithm

The main idea in our approach is to reduce search space, i.e. to reduce the number of qualified overlay nodes/links, for service path selection for a given request. A service path will satisfy a user QoS constraint only if the aggregated 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 exceeds the corresponding QoS requirement, then the service path containing such link or node would certainly fail to meet the QoS requirement. Therefore, the key to reducing search space 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 do multi-constrained path selection. Suppose a user request $S_j, S_k \ldots S_t$, 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 *search space reduction* and second is *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^u 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 $u - 1^{th}$ for each node providing service u^{th} and put them at $u - (u - 1) + 1$ level in the tree. This iterative process results in *K-ary* tree rooted at O_r. The leaf nodes, at level u, 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_n. We examine the path from O_n 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_n meets all the QoS requirements, then we continue tree traversal by selecting the left first child of O_n. If O_n does not have any child or all the children have been visited, then go the the adjacent node of O_n. If the path from the root till O_n does not meet all of the QoS requirements, we cutoff the subtree below O_n and proceed to the adjacent node in the same level. When we reach the leaf node O_l in level u, 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*. Our algorithm also provides a new control knob aimed at service providers to bias the path selection for load balancing or service reuse. If more than one path is found, then the selection of service path is based on whether the service provider objective is *load balancing* or *reuse*. If the objective is to have high degree of load balancing, then we select the path with minimum SPU metric, otherwise, we select the path with minimum SPC metric.

4.2 QSC-Modification (QSC-M)

Next, we propose QSC-M algorithm that builds on top of QSC algorithm [7]. Both the approaches use the same path selection algorithm to find the shortest path; the difference between QSC-M and QSC is that QSC-M has an additional step after the shortest path is found. QSC-M checks whether the shortest path meets the QoS requirements or not. User request is admitted if the shortest path satisfies all the QoS requirements. Otherwise the user request is rejected. QSC, on the other hand, does the path selection based only on finding the shortest path and not on explicitly satisfying the QoS requirements, and therefore a service request is never rejected in QSC. As we later show in the evaluation section QSC-M performs significantly better that QSC. This is because QSC-M does not hold resources for the sessions where QoS constraints are violated.

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of KCP algorithm using simulations. We show the trade-offs between the reuse and load balancing approach, and compare their performance with QSC approach proposed in [7].

5.1 Evaluation Methodology

To evaluate KCP and QSC-M algorithm, we have implemented a flow-level discrete-event simulator. We use two methods to construct the Service Overlay Network topology. In the first method, we use gt-itm [1] to generate native network topology based on transit-stub model. The key parameters for the topology generated from gt-itm are: 3 transit network, 8 stub network per transmit network and 20 nodes per network. The total number of nodes in this topology are 2000. In our simulation, the delay for the native links is uniformly distributed from 4 to 20 msec, loss rate from 2% to 8% and the capacity from 10 to 100 Mbps. The native layer routing uses Dijkstra's shortest path (with delay as the cost metric) algorithm.

The overlay network consists of 200 SON nodes which are randomly selected from 2000 native network nodes. The overlay nodes are connected in a full-mesh topology and the capacity of the overlay link is equal to the *narrow link* capacity. SON provides a total of 5 distinct services and each overlay node provides 1 out of these 5 services.

In the second method, we use the S^3 Planet-lab measurements [22] to construct SON topology. These Planet-lab measurements provide the delay and capacity between over-

lay nodes and therefore the resulting topology has a realistic delay and capacity distribution. Additionally, the topology is not a full-meshed topology and the connectivity is determined based on the availability of data in the Planet-lab data.

In addition to SON topology, the end-user workload is another important parameter in evaluation of these algorithms. Tang et. al [20] developed an open source synthetic workload generator for streaming media applications (called MediSyn). Medisyn generates realistic and reproducible synthetic work load, which models a number of characteristics unique to streaming media services, including file duration, session duration and non-stationary popularity of media accesses. We have also evaluated the performance of these algorithms with end-user requests which are generated from a Poisson process i.e. the request inter-arrival follows an exponential distribution. In both the end-user request generation model, each user request arrives randomly at one of the overlay nodes in SON. A user request is *admitted* into the system if a service path meeting the QoS requirements exists. Otherwise, the user request is *rejected*.

We have also implemented QSC algorithm, as described in [7], in our simulator to compare its performance with KCP and QSC-M. The QSC algorithm is based on Dijkstra's shortest path algorithm. For each user request, the algorithm generates a candidate graph that is comprised of all the overlay nodes that provide any component-service from service template corresponding to user request. The weight of each edge in candidate graph is derived by aggregating the properties of corresponding edges and nodes in the SON network. The cost of overlay link between O_i and O_j is given as

$$\frac{D(O_i,O_j)}{D^{QoS}} + \frac{\rho(O_i,O_j)}{L^{QoS}} + \frac{C(S_i)}{U(O_i)} + \frac{A^{QOS}}{A(O_i,O_j)}$$

At this time, we do not simulate any cross traffic in our experiments. Consequently, any load fluctuations in a network link can happen only due the assignment of a user request to that link. The use of cross-traffic to introduce random and independent load fluctuations in native links and therefore in overlay links will be examined in future work.

5.2 Performance Metrics

We evaluate path selection based on five performance metrics: *reject ratio, QoS violation rate, QoS violation degree, idle node ratio and node utilization*. First three metrics, which are more relevant to the performance perceived by user, examine algorithm performance by quantifying the fraction of admitted requests to total requests, and the rate and the extent of violations of various QoS constraints. Last two metrics, which are indicators of network utilization, are a measure of the number of idle nodes and distribution of load on the nodes in the network.

Specifically, *reject ratio* is a measure of how many user requests are not accepted (i.e. rejected) due to incapability of SON to meet their QoS requirements. It is defined as the fraction of number of requests rejected to the total number of request that arrive in the system. Smaller reject ratio would mean that larger number of user requests were admitted into the system. This is desirable as long as the user requests that are admitted have their QoS satisfied. On the other hand, network operators may not prefer a smaller reject ratio that is is achieved by admitting user requests without satisfying their QoS requirement because it will violate the service level agreements with users. We use *QoS violation ratio* to quantify the the number of user request whose QoS requirements were violated even once over their duration. It is defined as the fraction of admitted user requests that experience at least one QoS violation. *QoS violation degree* measures the magnitude of QoS violation and is defined as the difference between the measured QoS value and the desired QoS value relative to desired QoS value. *Idle node ratio* is measures the average number of of idle nodes (i.e. the nodes with zero utilization). *Node utilization* measures the averages node utilization over all the nodes in SON.

5.3 Evaluation using Medisyn based workload and Planet-lab based topology

We next examine the performance of two variants of KCP algorithm for different values of K and different load conditions, and compare its performance with QSC algorithm. The user requests are generated from Medisyn. For the results in Figure 2 to 4, the request arrival rate is varied from 200 to 700 requests. The average duration of all the requests is generated from a Zipf distribution in Medisyn with one of the following four mean and standard deviation values, (20, 30, 63, 98) and (60, 40, 60, 122), respectively. The QoS requirement associated with each user request is distributed as follows: delay QoS from 30 to 150 msec, bandwidth QoS from.3 to 65 Mbps and loss rate QoS from 8% to 25%. We have used planet-lab traces to construct the SON topology. Therefore, the overlay link delay and capacities in the trace driven topology correspond to delay and capacities in planet-lab and allows us to evaluate algorithms in a more realistic scenario. Note that we have set the loss rate to 0 for all the links because the loss rate information was not available in the planet-lab traces. In the following figures, KCP-R refers to KCP algorithm using reuse as objective for path selection, while KCP-L refers to using load balancing as objective.

Figure 2 shows both the *reject ratio* and *QoS violation ratio* for KCP variants, QSC and QSC-M algorithms for increasing user request arrival rate. First observation is that only QSC algorithm has non-zero QoS violation ratio. The main reason for a non-zero QoS violation ratio is that QSC algorithms admits a user request as long as shortest path is found and does not check whether the QoS requirements have been met. Since QSC algorithm only looks for shortest path, it is able to admit all the user requests and has a reject ratio equal to 0. KCP algorithm, on the other hand, admits a request only if a service path that meets all the QoS requirement is found. If such a service path is not found, then the user request is rejected. Similarly, QSC-M approach also rejects the request if the shortest path does not meet all QoS requirement.

Second observation is that both the reject ratio and QoS violation ratio increase with the user request arrival rate. This is not unexpected because higher request arrival rate leads to higher network/node resource utilization and consequently lower chances of finding service paths that match the QoS requirements of user request.

Figure 3 and 4 show the load variation and idle node ratio for all algorithm. We show the standard deviation of node utilization as a metric for load variation. The standard deviation is an important metric to quantify how much load balancing does a given algorithm achieve. The goal of load

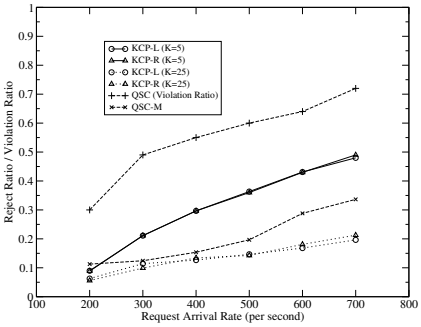

Figure 2: Reject/Violation Ratio (Medisyn Workload and Planet-lab Topology)

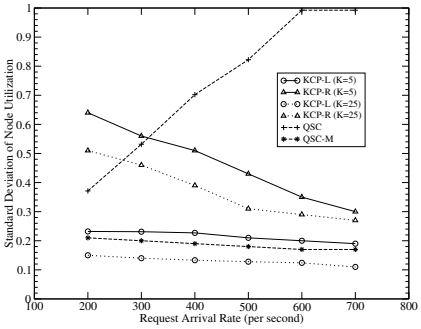

Figure 3: Node Utilization Variability (Medisyn Workload and Planet-lab Topology)

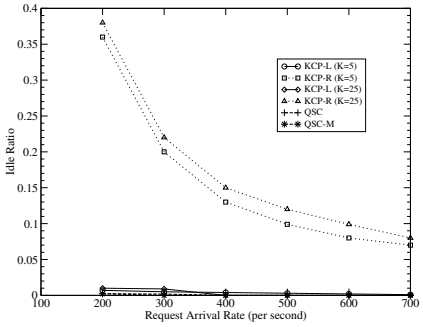

Figure 4: Idle node ratio (Medisyn Workload and Planet-lab Topology)

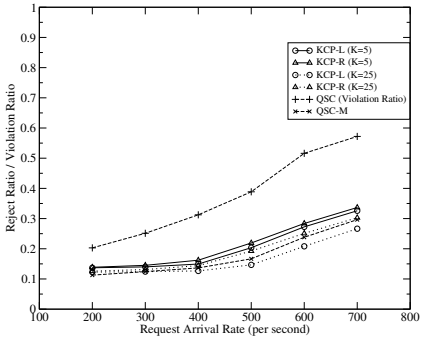

Figure 5: Reject/Violation Ratio (Random Workload and Planet-lab Topology)

balancing is to distribute the load from user requests among all the nodes in the network i.e. reduce the variation of load among overlay nodes. A higher value of standard deviation, therefore, would imply poor load balancing. Figure 3 clearly shows that KCP-L outperforms KCP-R in terms of load balancing objective. Also note that the standard deviation of node utilization shows a non-monotonic increase for all algorithms except QSC. As the node utilization approaches 100%, the node utilization is "clamped" by node capacity. This clamping effect causes the reduction in load variation at high loads.

KCP-R, on the other hand, has more number of idle nodes than KCP-L, as shown in figure 4. This implies that KCP-R achieves better reuse performance than KCP-L for a given K. In other words, KCP-R has low fragmentation of node resources and always has resources to admit user requests that may require 100% of node resources. Thus it is evident that both KCP-L and KCP-R achieve their individual goals and the service providers can control the load distribution on various service nodes by choosing either the load-balancing objective or the service reuse objective.

We have also evaluated the performance of these algorithms when the average duration of request is considerably longer. Specifically, the average duration of all the requests has one of the following four mean and standard deviation values, (4000, 7000, 9300, 11821) and (600, 400, 600, 1223), respectively. The results were similar to the case when the average duration of user request is shorter.

5.4 Evaluation using Random Workload and Planet-lab based topology

Next, we show results comparing the performance of proposed algorithms against QSC approach when the user requests are generated from Poisson distribution, as described in §5.1. Specifically, the user request interarrivals follow an exponential distribution with their arrival rate in the range [200,700]. The request duration is uniformly distributed between 15 and 45 secs, delay QoS from 200 to.30 msec and bandwidth QoS from 7 to 45 Mbps.

Figure 5 shows reject ratios for KCP and QSC-M algorithms, and violation ratio for QSC algorithm. The main observation is that KCP variants and QSC-M algorithms are able to meet all the QoS requirements by rejecting 10-25% requests under a range of load conditions. Additionally, the fraction of rejected request is less than 50% of the requests that experienced QoS violation in QSC.

Figure 6 and 7 show the load variation and idle node ratio for all algorithms. Similar to the result from previous section, we observe that KCP-L algorithm has the minimum load variation and therefore achieves the highest degree of load balancing. The advantage of KCP-R approach is evident in figure 7 which clearly shows that KCP-R has maximum fraction of completely idle nodes under all load conditions.

6. CONCLUSION

In this work, we have presented a novel algorithm for path selection in service overlay network. Our simulation results validate that the algorithm achieves the following: first, the value of K can be tuned to balance the performance requirements and the computation complexity; second, the algorithm is able to guarantee multiple QoS requirements of a

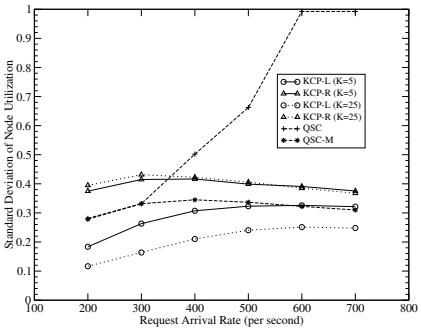

Figure 6: Node Utilization Variability (Random Workload and Planet-lab Topology)

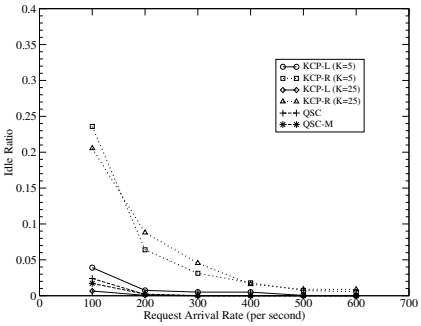

Figure 7: Idle Node Ratio (Random Workload and Planet-lab Topology)

given service composition request; third, the algorithm keeps the request reject ratio low by better utilization of resources. We have also introduced two distinct service provider objectives of *reuse* and *load balancing* and show that using simpler metrics such as SPC or SPU can help in modifying the path selection of KCP algorithm to achieve these objectives without affecting performance of KCP algorithm. The use of these two objectives coupled with algorithm's ability to guarantee each QoS constraint essentially provides simple tunable knobs for customizing this algorithm for different deployment scenarios.

Several important problems remain open for future work. What should be the value of K for a network with arbitrary number of nodes? Initial investigation shows that K depends on the user request arrival rate and resource availability. Other factors that K would depend on include size of SON, user request load/request-duration distribution and total number of services provided by SON. It would be useful to dynamically determine the value of K based on SON state. We have identified that *Reuse* and *Load balancing* have their own advantages for doing path selection. An open question is how to dynamically choose one of the two objectives according to network conditions or service provider utility functions.

7. REFERENCES

[1] Modelling Topology of Large InternetWorks. http://www.cc.gatech.edu/project, Nov. 2005.

[2] S. Banerjee, Z. Xu, S. Lee, and C. Tang. Service adaptive multicast for media distribution networks. In *IEEE WIAPP*, 2003.

[3] S. Choi, J. S. Turner, and T. Wolf. Configuring sessions in programmable networks. In *INFOCOM*, pages 60–66, 2001.

[4] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In *Proceedings of ACM SIGCOMM*, Aug. 2001.

[5] E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. In *Proceedings of ACM SIGCOMM*, Aug 2002.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[7] X. Gu, K. Nahrstedt, R. N. Chang, and C. Ward. QoS-Assured Service Composition in Managed Service Overlay Network. In *ICDCS*, 2003.

[8] M. Harville, M. Covell, and S. Wee. An architecture for componentized, network-based media services. In *IEEE ICME*, July 2003.

[9] J. M. Jaffe. Algorithms for Finding Paths with Multiple Constraints. *Networks*, 14:95–116, 1984.

[10] M. Jain and C. Dovrolis. End-to-end Estimation of Available Bandwidth Variation Range. In *Proceedings of SIGMETRICS*, jun 2005.

[11] J. Jin and K. Nahrstedt. On exploring performance optimizations in web service compositions. In *Proceedings of Middleware*, 2004.

[12] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings ACM SIGCOMM*, pages 61–72, 2002.

[13] Z. Li and P. Mohapatra. QRON: QoS-Aware Routing in Overlay Networks. *IEEE JSAC*, 22(1):29–40, 2004.

[14] J. Liang and K. Nahrstedt. Service composition for advanced multimedia applications. In *Proceedings of MMCN*, 2005.

[15] P. V. Mieghem and F. A. Kuipers. Concepts of Exact QoS Routing Algorithms. *Transactions on Networking*, 12(5), 2004.

[16] B. Raman and R. Katz. Load balancing and stability issues in algorithms for service composition. In *Proceedings of INFOCOM*, 2003.

[17] P. Sharma, Z. Xu, S. Lee, and S. Banerjee. Estimating network proximity and latency. *ACM SIGCOMM CCR*, 36(3), 2006.

[18] S. Shi and J. Turner. Placing Servers in Overlay Networks, 2002.

[19] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: Offering QoS using Overlays. In *HotNets-I*, 2002.

[20] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat. Medisyn: a synthetic streaming media service workload generator. In *ACM NOSSDAV*, 2003.

[21] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE JSAC*, 14(7), 1996.

[22] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee. S3: A scalable sensing service for monitoring large networked systems. In *INM: 2006 SIGCOMM Workshop*, 2006.