Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 873–885
computer communications
www.elsevier.com/locate/comcom

State aggregation of large network domains

Yong Tang [a], Shigang Chen [a,*], Yibei Ling [b]

[a] *Department of Computer and Information Science and Engineering, University of Florida, E460 CSE Building, PO Box 116120, Gainesville, FL 32611-6120, USA*
[b] *Applied Research Laboratories, Telcordia Technologies, NJ, USA*

Received 2 November 2005; received in revised form 20 October 2006; accepted 23 October 2006
Available online 14 November 2006

Abstract

Many important network functions (e.g., QoS provision, admission control, traffic engineering, resource management) rely on the availability and the accuracy of network state information. It is impractical to maintain the complete state information of a large internetwork at a single location. Large networks are often hierarchically structured, with each domain advertising its aggregated state. To achieve scalability, a delicate tradeoff has to be made between minimizing the size and maximizing the accuracy of the aggregated state. Given certain space limitation, inaccuracy introduced by different aggregation methods varies greatly. This paper gives a unified account of state aggregation based on approximation curves. The existing aggregation methods are special cases in the solution space under this model. New aggregation methods based on polynomial curves, cubic splines, and polylines are proposed, and their accuracy/space tradeoffs are studied. Extensive simulations show that these new methods approximate the network state far more accurately than the existing ones. In particular, the polylines achieve the best accuracy/space tradeoff.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Network state aggregation; Service approximation curve; Quality of service

1. Introduction

Many important network functions (e.g., QoS provision, admission control, traffic engineering, resource management) rely on the availability and the accuracy of network state information. Take a few examples. To provide high-quality IP television, information about the available network resources is needed to find routing paths with sufficient bandwidth and bounded delay between two nodes in the network. The statistical QoS provision in a DiffServ domain requires the network state information to implement the admission control in order to regulate the total traffic volume of high-priority classes. The construction of virtual circuits in ATM networks, the layout of overlay networks in MPLS, and the implementation of load balancing over multiple paths can all benefit from the knowledge of the network state information. However, it is impractical to maintain the complete state of a large network. To solve the scalability problem, the current internetworks are structured hierarchically as a collection of domains, which may be further decomposed into subdomains recursively. At each hierarchical level, the state information can be aggregated and only the aggregated state is advertised externally.

Although the aggregated information is imprecise, it can be very useful [1]. For instance, the soft QoS provision in a DiffServ domain can accommodate certain degree of imprecision without violating the service contract. In QoS routing, multiple candidate paths may be generated from the imprecise aggregated state. A signaling process will then be used to reject the unqualified paths [2]. The chance of producing unqualified paths is directly related to how accurate the aggregated state is. Improving the accuracy of aggregated state is the subject of this paper.

Any state aggregation algorithm must make a tradeoff between two conflicting objectives. First, the aggregated

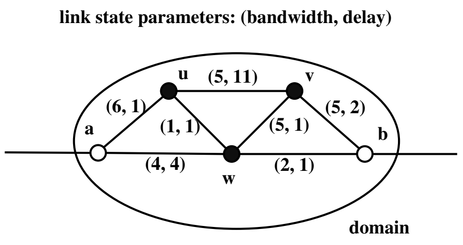
* Corresponding author. Tel.: +1 352 392 2713; fax: +1 352 392 1220.
E-mail address: sgchen@cise.ufl.edu (S. Chen).

information about a domain should be as accurate as possible. The accuracy directly impacts the performance of any network function that uses the information to make decision. Second, the size of the aggregated state should be reduced as much as possible. After all, the whole purpose of aggregation is to make the state information more compact so that scalability can be achieved. It is normally true that reducing information introduces imprecision. Hence maximizing the accuracy and minimizing the size of the aggregated state are two conflicting objectives, and a trade-off has to be made. In addition to the above issues, the network aggregation often deals with multi-dimensional state. In this paper, we focus on two dimensions, *delay* and *bandwidth*.

The aggregated state of a domain does not contain detailed information about the internal structure of the domain, but only contains information describing end-to-end properties between border nodes [2–5]. Fig. 1 shows a domain with two border nodes, a and b. The domain state is collected, aggregated, and advertised periodically. The delay and the bandwidth of each physical path from a to b represent a point on the bandwidth–delay plane, and the points of all physical paths from a to b define a *service staircase*, which outlines the area of supported services (the shaded area). There are up to $|E|$ corner points on the staircase [2], where $|E|$ is the number of physical links in the domain. Therefore, the worst-case space complexity for precisely storing a staircase is $O(|E|)$. The paths in the shaded area (such as $P7$) are not explicitly captured by the corner points. That does not mean those paths will never be utilized. The staircase is updated periodically. Once the current best paths (represented by the corner points) are overly used, other paths will emerge as the new best paths.

A router keeps detailed information about the domain it belongs to. The information includes the domain topology and the link state. An example is given in Fig. 1(i). But a router cannot afford to keep information in such details for all other domains in a large network. One solution is to store aggregated information about other domains. Fig. 2(i) shows the complete topology of a network, consisting of four domains, A–D. The state information stored at router a is illustrated in (ii), which consists of detailed state of domain A and aggregated state of domains B, C, and D. The aggregated state of an external domain consists of the border routers of the domain and the state between each pair of border routers. To disseminate aggregated state information, the border routers of the domains may calculate the aggregated states of their domains periodically and exchange the aggregated states by an OSPF-like protocol.

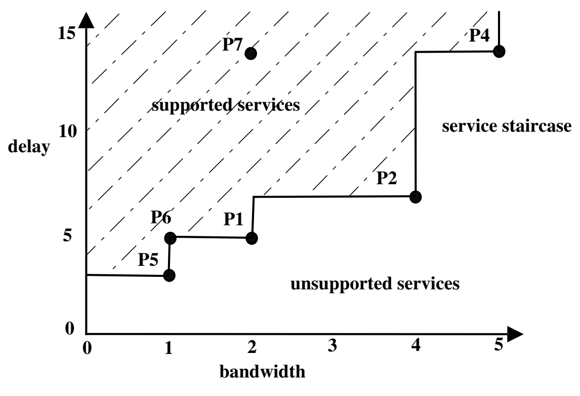
We use a QoS routing example from [2] to demonstrate how to use the state information in Fig. 2(ii). Suppose router a receives a request from a local host to find a QoS-constrained path to a remote host in domain D. Based its view of the network, a computes a feasible routing path, consisting of an intra-domain path and an inter-domain path. The intra-domain path specifies a local route from the local host to a border router. The inter-domain path specifies a sequence of border routers of transit domains; it ends at a border router of the destination domain. After that, a



(i)

paths		(bandwidth, delay)
P1:	a => w => b	(2, 5)*
P2:	a => w => v => b	(4, 7)*
P3:	a => w => u => v => b	(1, 18)
P4:	a => u => v => b	(5, 14)*
P5:	a => u => w => b	(1, 3)*
P6:	a => u => w => v => b	(1, 5)
P7:	a => u => v => w => b	(2, 14)

(ii)



(iii)

Fig. 1. Service staircase and supported services (shaded area).

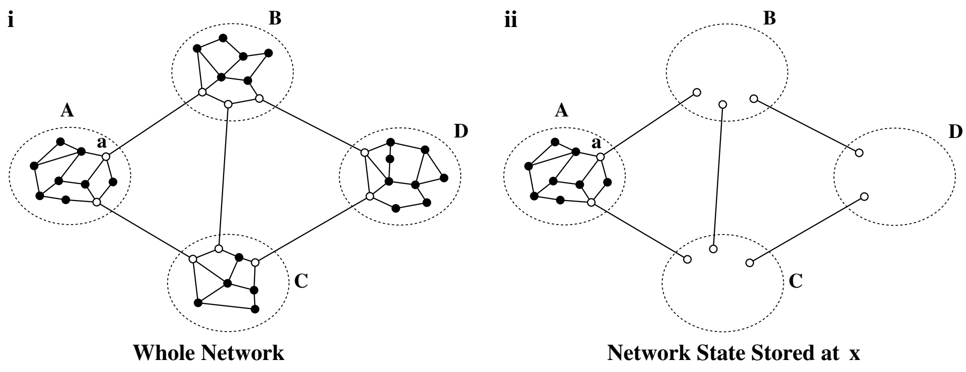


Fig. 2. A router stores detailed local state but aggregated remate state.

routing message will be sent to traverse the border routers of the inter-domain path. When a border router receives the routing message, it computes the path segment within its domain to the next border router. If such a path segment cannot be found without violating the QoS constraints, *crankback* [6] is performed to choose an alternative path. For more details, readers are referred to [2,7].

Now the question is how to represent the aggregate state of a domain in a compact form. More specifically, how to represent the state between any pair of border routers of the domain? If we directly use the staircase shown in Fig. 1(iii), the worst-case storage overhead will be too high [2,5]. The past research mainly aimed at reducing the service staircase in order to fit it in a fixed amount of memory, whose size is independent of the internal complexity of the domain. The known approaches approximate the staircase by a single point [3,4], a line segment [2], or based on the deviation from a reference point [5], which are all insufficient for achieving high accuracy.

In this paper, we study how to aggregate the state information of large network domains by using service approximation curves. The existing approaches are special cases in the solution space under our model. We propose three new non-linear approximation curves, *polynomial curves*, *piecewise cubic splines*, and *polylines*, which approximate a service staircase with far better accuracy. With a tunable parameter, these approximation curves allow the system designers to make tradeoff between aggregation accuracy and space overhead. We present the algorithms for calculating the new curves and analyze the complexities of the algorithms. We describe how to aggregate a large domain topology to a logical mesh, star, or star-by-pass topology when the proposed approximation curves are used to represent the network state between border routers. We compare our new aggregation methods with the existing ones by extensive simulations, which demonstrate that the new methods significantly improve the accuracy of aggregated state. In particular, given the same amount of memory space, polylines produce the most accurate aggregated state among all approaches that we compare.

It should be noted that there are two sources of inaccuracy in the network state information stored at a router. One source comes from periodic state advertisement. The larger the advertisement period, the larger the inaccuracy. The other source comes from state aggregation, which is what this paper studies. These two sources of inaccuracy are orthogonal. This paper exclusively focuses on developing better aggregation methods, which approximate any given network state with certain accuracy. It does not consider the issue of advertisement period.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 gives the network model. Section 4 proposes the new approximation curves. Section 5 discusses how to aggregate a large domain topology. Section 6 presents the simulation results. Section 7 draws the conclusion.

2. Related work

The problem of aggregating network state with one dimension (e.g., delay or bandwidth) is well studied [8,9]. The focus of this paper is the aggregation of multi-dimensional state (delay and bandwidth), which is a much harder problem [10–12].

The research of aggregating the state of a bandwidth–delay sensitive domain can be traced back to [3], where Lee transforms the domain to a spanning tree among border nodes. The spanning tree provides a distortion-free aggregation for bandwidth, but not for delay, for which significant errors may be introduced. Iwata et al. [4] propose an aggregation approach for a network with six state parameters. Linear programming is used to minimize the distortion. Based on a specific preference order among the state parameters, one path can be selected as the ‘‘best’’ among all paths between a pair of border routers. In [4], the state of the ‘‘best’’ path represents the state of all paths between the two border routers. Essentially one point on the bandwidth–delay plane is used to represent the service staircase.

Korkmaz and Krunz [5] approximate the bandwidth–delay staircase by three values: the minimum delay *min_d*, the maximum bandwidth *max_b*, and the smallest stretch

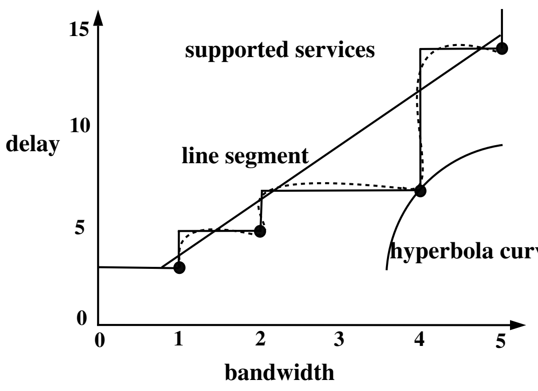


Fig. 3. Approximate the service staircase.

factor *min_s* of all paths between two border nodes. The stretch factor of a path measures how much the delay (bandwidth) of the path deviates from the best delay (bandwidth) of all paths. It defines a hyperbola curve that resides below the staircase (Fig. 3). The supported services consist of all points whose delay values are larger than *min_d*, whose bandwidth values are smaller than *max_b*, and whose stretch factors are smaller than *min_s*. Although this approach achieves much better approximation than a point, the distortion can still be high [2], especially when the general shape of the staircase is convex.

Lui et al. [2] choose a line segment to approximate the staircase. The line segment representation is integrated with Dijkstra's algorithm (DA) and the centralized bandwidth–delay routing algorithm (CBDRA) [13,14]. The line segment is determined by the least square method that minimizes the summation of the squares of the distances from the corner points on the staircase to the line. Simulations show that line segments work better than stretch factors on average. However, it is obvious that a straight line does not approximate a staircase well in general (Fig. 3).

One might think that the approximation curve should completely lie within the supported-services area in order to avoid accepting unsupportable requests. This is in fact not necessary. On one hand, such a conservative curve may reject many supportable requests. On the other hand, for soft QoS provision, a request close to the supported area is acceptable; for hard QoS provision, the admission control may use an additional signaling process to verify whether a chosen path can satisfy the requirements or not [2].

3. Network model

A network is modeled as a set of domains that connect with each other through border routers. A domain is modeled as $\langle V, B, E\rangle$, where V is a set of routers, B ($\subset V$) is a set of border routers, and E is a set of links connecting the routers. Let $\kappa_{a,b}$ denote the service staircase for a pair of border routers, a and b. $\kappa_{a,b}$ is a function of bandwidth, specifying the minimum delay that can be achieved between the routers for any given bandwidth value. When the discussion involves no other router pairs, we will drop the subscript and simply use κ. The convex corner points of the staircase κ are called *representatives*, which uniquely define the shape of κ.

The most accurate way of keeping the state information between two border nodes is to store the set of representatives. However, the number of representatives can be as large as $|E|$ [2]. Approximation is necessary to reduce the overhead of advertising the state information and storing it in routers of other domains. The task is to identify a *service approximation curve* κ^* that is close to the staircase κ but take less space to store. There can be many different types of approximation curves. Each will have a different symbol replacing the superscript "*". Improving accuracy and reducing space are two conflicting objectives in the selection of approximation curve. Better accuracy normally requires more space. Regardless how large the domain is, our goal is to limit the space requirement for storing κ^* to $O(1)$, while minimizing the difference between the approximation curve and the service staircase.

Before presenting our approximation methods, we use an example to show how the approximation curves may be used. Refer back to Fig. 2(ii). Suppose router a knows the detailed state information of its own domain and the aggregated state of external domains. The aggregated state is computed by and propagated from the border routers of external domains. The aggregated state of an external domain is defined as follow. It contains only the border routers of the domain and one logical link for each pair of border routers. Other logical topologies for the aggregated state will be discussed in Section 5. A service approximation curve κ^* is associated with each logical link. It is a non-decreasing function on the bandwidth–delay plane. For an arbitrary bandwidth value w, $\kappa^*(w)$ gives the minimum delay that can be achieved by a path with bandwidth w between the two border routers. Now suppose router a receives a request to find a routing path from a local host to a remote host in domain D, such that the bandwidth of the path is no less than w_{req} and the delay of the path is no more than d_{req}. Based on its knowledge about the network, router a finds a candidate path by the following algorithm.

Step 1. Assign a delay value $\kappa^*(w_{req})$ to each logical link, where κ^* is the approximation curve of the link.

Step 2. Remove all physical links whose bandwidths are smaller than w_{req} and all logical links whose assigned delays are infinite.

Step 3. Find the minimum-delay path in the residual graph by Dijkstra's algorithm and send a routing message along the path. As we have discussed in Section 1, when receiving the routing message, the border router of an external domain will attempt to fill in the path segment in that domain. As it travels, the routing message will verify if the bandwidth and delay requirements can be met along the path.

4. Service approximation curves

We first define the metric of *area distance*, which is used to evaluate the accuracy achieved by a service approximation curve. We then present three new approximation curves.

4.1. Area distance

We use Fig. 4 to illustrate the distortion caused by an approximation curve κ^*. Any service request (w_{req}, d_{req}) can be supported if it is above the staircase κ. Now suppose we make decision based on κ^* instead of κ. A request (w_{req}, d_{req}) above the approximation curve κ^* but below the staircase in region II will be accepted. It is however not supported by any physical path because it is not in the region of the supported services above the staircase. On the other hand, a request (w_{req}, d_{req}) above the staircase but below the approximation curve κ^* in region III can be supported in reality, but is rejected. Region II consists of all *over-estimated* services, and region III consists of all *under-estimated* services. Therefore, It is natural to use the sizes of these regions to judge the quality of approximation by κ^*.

Given a service curve κ and an approximation curve κ^*, the *positive area distance*, $\Delta_+(\kappa,\kappa^*)$, measures the total size of regions of over-estimated services. The *negative area distance*, $\Delta_-(\kappa,\kappa^*)$, measures the total size of regions of under-estimated services.

$$\Delta_+(\kappa,\kappa^*) = \int_{w_l}^{w_h} \max(\kappa(w) - \kappa^*(w), 0)\ \mathrm{d}w$$

$$\Delta_-(\kappa,\kappa^*) = \int_{w_l}^{w_h} \max(\kappa^*(w) - \kappa(w), 0)\ \mathrm{d}w$$

$[w_l \ldots w_h]$ is the bandwidth range where κ and κ^* differ. The *area distance*, $\Delta(\kappa,\kappa^*)$, measures the total size of regions of both over-estimated services and under-estimated services.

$$\begin{aligned}\Delta(\kappa,\kappa^*) &= \Delta_+(\kappa,\kappa^*) + \Delta_-(\kappa,\kappa^*)\\ &= \int_{w_l}^{w_h} |\kappa(w) - \kappa^*(w)|\mathrm{d}w\end{aligned}$$

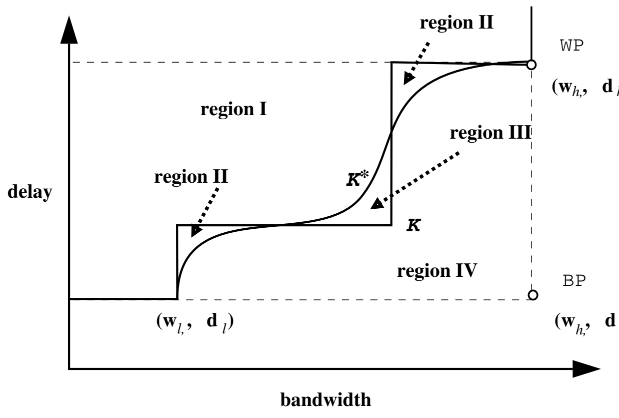


Fig. 4. Area distance.

$\Delta(\kappa,\kappa^*)$ is a good indicator of the overall distortion introduced by an approximation curve. On the other hand, if a conservative system design requires that most accepted requests must receive the desired QoS, then κ^* should be selected to minimize $\Delta_+(\kappa,\kappa^*)$. If an aggressive system design requires most supportable services must be accepted, then κ^* should be selected to minimize $\Delta_-(\kappa,\kappa^*)$.

Our goal is to find the appropriate approximation curves so that $\Delta_+(\kappa,\kappa^*)$, $\Delta_-(\kappa,\kappa^*)$, and/or $\Delta(\kappa,\kappa^*)$ are minimized without incurring too much overhead. In the following, *polynomial curves*, *cubic splines*, and *polylines*, are introduced.

4.2. Approximation by polynomial curve

Referring to Fig. 4, let the first representative of the staircase κ be (w_l, d_l), which is the convex corner with the smallest bandwidth. Let the last representative be (w_h, d_h), which is the convex corner with the largest bandwidth. Our first approximation method uses a polynomial function

$$\kappa^{pc}(w) = \begin{cases} d_l & ; w \leqslant w_l \\ \sum_{i=0}^{n} \alpha_i w^i & ; w_l < w \leqslant w_h \\ \infty & ; w > w_h \end{cases} \tag{1}$$

where, α_0, $\alpha_1, \ldots, \alpha_n$ are coefficients, which determine the shape of the polynomial curve. The *line segment* in [2] is a special case of polynomial curves with $n = 1$. The value of n determines the quality of the approximation. A larger n means a better resolution and additional space overhead as well. On the other hand, n does not have to be too large. In our simulations, $n = 4$ works considerably better than $n = 1$. As n increases further, the rate of accuracy improvement slows down.

For a general staircase function, it is difficult to determine the optimal values for α_0, $\alpha_1, \ldots, \alpha_n$ such that $\Delta(\kappa,\kappa^{pc})$ is minimized. The intuition is that we want the approximation curve to be close to the staircase, i.e., the distance between their corresponding points (with the same bandwidth value) should be minimized. Based on this observation, we use the least square method to find the coefficients α_0, $\alpha_1, \ldots, \alpha_n$. Given m data points $(w_1, d_1), \ldots, (w_m, d_m)$ on the staircase, the least square method tries to minimize the following cost function

$$\begin{aligned}\phi(\alpha_0, \alpha_1, \ldots, \alpha_n) &= \sum_{i=1}^{m} (d_i - \kappa^{pc}(w_i))^2\\ &= \sum_{i=1}^{m} (d_i - \sum_{j=0}^{n} \alpha_j w_i^j)^2\end{aligned}$$

To minimize $\phi(\alpha_0, \alpha_1, \ldots, \alpha_n)$, the coefficients α_0, $\alpha_1, \ldots, \alpha_n$ must yield zero first derivatives. Hence, we have the following linear equations.

$$\frac{\partial\phi}{\partial\alpha_0} = -2\sum_{i=1}^{m} w_i^0 (d_i - \sum_{j=0}^{n}\alpha_j w_i^j) = 0$$
$$\frac{\partial\phi}{\partial\alpha_1} = -2\sum_{i=1}^{m} w_i^1 (d_i - \sum_{j=0}^{n}\alpha_j w_i^j) = 0$$
$$\cdots\cdots\cdots$$
$$\frac{\partial\phi}{\partial\alpha_n} = -2\sum_{i=1}^{m} w_i^n (d_i - \sum_{j=0}^{n}\alpha_j w_i^j) = 0$$

The values of $\alpha_0, \alpha_1, \ldots, \alpha_n$ can be calculated by solving the above linear equations.

The m data points we used in the least square method do not have to be the corner points of the staircase. They should be selected from the service staircase evenly between w_l and w_h. When m is chosen to be n, the resulting polynomial will pass every selected data point. For a larger m, the polynomial cannot pass every data point but will minimize the summation of the squares of the distances from the data points to the polynomial curve. In general, a larger m allows the polynomial curve to fit the staircase better as more information about the shape of the staircase is provided in the calculation. But a larger m also means more computation overhead.

To approximate a different service curve other than a staircase, we simply apply the same algorithm on the m data points taken from that curve.

The space needed for storing κ^{pc} is $n+5$ float numbers, with $n+1$ floats for the coefficients and four floats for w_l, d_l, w_h, and d_h. The time complexity for solving the linear equations is $O(nm+n^3)$. It is higher than $O(m)$ for generating the line segment [2] or calculating the stretch factor [5]. Nevertheless, it is not significant overhead because n does not have to be large. In addition, it takes far more time, $O(|E|^2+|E||V|\log|V|)$ [2], to compute the representatives before approximation can be performed. The overhead for calculating representatives is common to all approaches and far larger than the approximation overhead.

4.3. Approximation by cubic spline

Our second approximation method uses a cubic spline κ^{cs}, which approximates the service staircase κ segment by segment with piecewise polynomial functions. We make a brief description of cubic spline below. Details about curve fitting by spline functions can be found in [15].

Let $w_l = \lambda_0 < \lambda_1 < \ldots < \lambda_n = w_h$ be an even partition of the interval $[w_l, w_h]$. Take two additional values with the same spacing on both sides of the interval: λ_{-2} and λ_{-1} smaller than w_l; λ_{n+1} and λ_{n+2} greater than w_h. A series of cubic spline basis functions are defined. For $0 \leqslant j \leqslant n$,

$$B_j(w) = (\lambda_{j+2} - \lambda_{j-2}) \sum_{s=-2}^{2} \frac{(\lambda_{j+s} - w)_+^3}{\prod_{k=-2,k\neq s}^{2} (\lambda_{j+s} - \lambda_{j+k})} \qquad (2)$$

where $(x)_+$ equals x if $x \geqslant 0$, and 0 otherwise.

The cubic spline $\kappa^{cs}(w)$ is a linear combination of $B_j(w)$.

$$\kappa^{cs}(w) = \begin{cases} d_l & ; w \leqslant w_l \\ \sum_{j=0}^{n} \alpha_j B_j(w) & ; w_l < w \leqslant w_h \\ \infty & ; w > w_h \end{cases}$$

where $\alpha_0, \alpha_1, \ldots, \alpha_n$ are coefficients, which determine the shape of the cubic spline. The cubic spline consists of piecewise polynomials that capture the local subtlety of the service staircase. The more the number of coefficients, the better the approximation.

Similar to the previous subsection, we use the least square method to find the coefficients that minimizes the square error defined below.

$$\phi(\alpha_0, \alpha_1, \ldots, \alpha_n) = \sum_{i=0}^{m} (d_i - \kappa^{cs}(w_i))^2$$

where $(w_1, d_1), \ldots, (w_m, d_m)$ are m data points selected from the service staircase evenly between w_l and w_h. $m \geqslant n+1$. To minimize $\phi(\alpha_0, \alpha_1, \ldots, \alpha_n)$, the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_n$ must yield zero first derivatives, which leads to $(n+1)$ linear equations.

$$\frac{\partial\phi}{\partial\alpha_0} = -2\sum_{i=1}^{m} B_0(w_i)(d_i - \sum_{j=0}^{n}\alpha_j B_j(w_i)) = 0$$
$$\frac{\partial\phi}{\partial\alpha_1} = -2\sum_{i=1}^{m} B_1(w_i)(d_i - \sum_{j=0}^{n}\alpha_j B_j(w_i)) = 0$$
$$\cdots\cdots\cdots$$
$$\frac{\partial\phi}{\partial\alpha_n} = -2\sum_{i=1}^{m} B_n(w_i)(d_i - \sum_{j=0}^{n}\alpha_j B_j(w_i)) = 0$$

The values of $\alpha_0, \alpha_1, \ldots, \alpha_n$ can be calculated by solving the linear equations.

The space needed for storing k^{cs} is $n+5$ float numbers, with $n+1$ floats for the coefficients and four floats for w_l, d_l, w_h, and d_h. The minimal value for n is one. The time complexity for solving the linear equations is $O(nm+n^3)$, which is the same as that of polynomial-curve approximation. The above method can also approximate any service curve other than staircase simply by taking the m data points from that curve.

4.4. Approximation by polyline

Although a cubic spline approximates the local trend of a staircase very well, it has a weakness. Like a polynomial curve, a cubic spline incurs heavier distortion near the corner points as it uses a *continuous* curve to approximate the *discrete* steps (Fig. 4).

The corner points contain more information than the other points on the staircase because they determine the shape of the staircase. Furthermore, some corner points are more important than others. For example, there are eight corner points in Fig. 5. Points P_1, P_5, and P_8 are more important than the other five because they control the

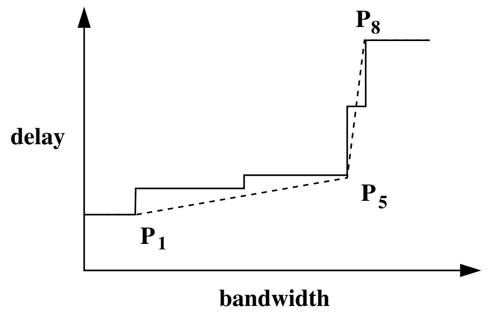


Fig. 5. Polyline approximation.

global shape of the staircase. Hence, we can approximate the staircase by only keeping the most important corner points. In Fig. 5, we may keep only P_1, P_5, and P_8, and use a polyline κ^{pl} connecting them to approximate the service staircase.

Now the problem is to determine which corner points are more important than the others. Suppose we are allowed to choose only n' corner points due to a space limitation. We would like to choose such n' points that minimize $\Delta(\kappa,\kappa^{pl})$. Suppose there are q corner points in total. The number of different combinations of n' points is $C_q^{n'}$. It takes $O(q)$ time to compute $\Delta(\kappa,\kappa^{pl})$ for each combination. Therefore, a brute-force algorithm will take $O(qC_q^{n'})$ time to find the best n' points. In the following, we describe a heuristic algorithm with a time complexity of $O(n'q)$.

Given a set of q corner points on the staircase, $\{P_1,\ldots,P_q\}$. Initially, the approximation polyline κ^{pl} is a line segment connecting P_1 and P_q, denoted as P_1–P_q. Compute the distances from all other corner points to the polyline, and find the corner point P_x with the maximum distance to P_1–P_q. Then insert P_x into the polyline, which becomes P_1–P_x–P_q. Repeat the above process until the polyline contains n' corner points.

The space needed for storing κ^{pl} is $2n'$ float numbers, two for each selected corner point. According to our simulations, a small value for n' is sufficient to achieve a good approximation. To approximate an arbitrary service curve, the next point to be inserted into the polyline does not have to be a corner point. It is the point that has the maximum distance from the polyline.

5. Aggregating domain state

Traditionally the aggregation process [2,5] is to transform the domain topology to a mesh topology and, if necessary, further transform the mesh to a star and then to a star-with-bypass topology, as illustrated in Fig. 6. Different aggregation methods do not vary on the overall process but instead on how the topology transformation is performed for a given type of approximation curves, such as stretch factor in [5] or line segment in [2]. This section describes the transformation process when the approximation curves in Section 4 are used.

5.1. Mesh topology

The first step is to transform the domain topology to a mesh topology, consisting of all border routers and a service approximation curve for each pair of border routers. Note that the pair is ordered. (a,b) and (b,a) are two different pairs. For the clarity, we only draw two internal routers in Fig. 6(i). Aggregation however should only be done on large domains with many internal routers and complex intra-domain topologies. The transformation of a physical topology takes two steps.

- The first step is to find the service staircase for each pair of border routers, (a,b). We must find all convex corner points (called representatives) of the staircase, which can be achieved by iteratively executing Dijkstra's algorithm for every link bandwidth value. We first sort the links in the domain by descending bandwidths. Let the largest bandwidth be w_{max}. We find the smallest delay d_{max_w} from a to b in a graph that consists of only links of bandwidth w_{max}. (w_{max}, d_{max_w}) is a representative that has the largest bandwidth. We then insert the links of the next largest bandwidth value (w') and find the smallest delay d'. If $d' < d_{max\_w}$, (d', w') is another representative. The process ends after the smallest delays for all

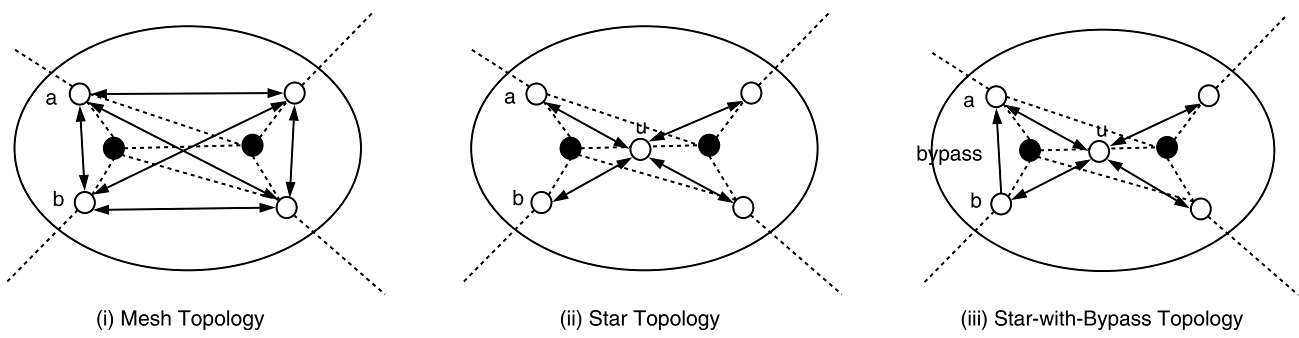


Fig. 6. The physical topology is first transformed to a mesh topology among the border routers, then to a star topology with a logical nucleus, and finally to a star-with-bypass topology. A physical link is represented by a dotted line, a logical link is represented by a solid line, a border router is represented by a white circle, and an internal router is represented by a black circle. A double-arrow link means two logical links in opposite directions.

bandwidth values are identified. In fact, the representatives from a to all other border routers can be calculated together by the nature of Dijkstra's algorithm. Since there are at most $|E|$ different bandwidth values, Dijkstra's algorithm will be executed at most $|E|$ times. Then, the total running time of finding the representatives from one border router to all other border routers is $O(|E|^2 + |E||V|\log|V|)$. Because there are $|B|$ border nodes, the total running time for finding all representatives is $O(|B||E|^2 + |B||E||V|\log|V|)$.

- The second step is to compute an approximation curve for each staircase, which has been discussed in Section 4. The aggregated domain state has one logical link per border pair and the state of the logical link is represented by the approximation curve. This aggregated domain state will be advertised by the border routers to the rest of the network.

The mesh topology has $|B|(|B|-1)$ logical links. The state of a link (a,b), $\forall a,b \in B, a\neq b$, is a service approximation curve, denoted as $\kappa^*_{a,b}$, where $*$ is *pc*, *cs*, or *pl* (for polynomial curve, cubic spline, or polyline), depending on which type of approximation curves are used.

5.2. Star topology

The mesh topology has $O(|B|^2)$ logical links. For a domain with many border routers, further aggregation may be needed [16]. By transforming it to a star topology, we can reduce the number of logical links to $2|B|$, as illustrated in Fig. 6(ii). A logical nucleus node, denoted as u, is introduced as the center of the start topology. For any border router $a \in B$, there are two logical links, (a,u) and (u,a), whose service curves are defined as follows:

$$\kappa_{a,u} = \frac{1}{2(|B|-1)} \sum_{b\in B,\ b\neq a} \kappa^*_{a,b}$$
$$\kappa_{u,a} = \frac{1}{2(|B|-1)} \sum_{b\in B,\ b\neq a} \kappa^*_{b,a} \tag{3}$$

Use $\kappa_{a,u}$ as an example. It is a function representing a curve on the bandwidth–delay plane. We approximate $\kappa_{a,u}$ by another function $\kappa^*_{a,u}$ so that it can stored in a fixed amount of space. The approximation is done by using the same techniques developed in Section 4. Note that $*$ can be *pc*, *cs*, or *pl*. We discuss these cases separately.

- *Case 1:* $* = pc$. First (3) becomes

$$\kappa_{a,u} = \frac{1}{2(|B|-1)} \sum_{b\in B,\ b\neq a} \kappa^{pc}_{a,b}$$

$\kappa^{pc}_{a,b}$ is defined by (1) and is associated with two data points (w_l,d_l) and (w_h,d_h). Let $min\_w_l = \min_{b\in B,\ b\neq a}\{w_l\}$ and $max\_w_h = max_{b\in B,\ b\neq a}\{w_h\}$. From (1), we can determine the value of $\kappa_{a,u}(w)$ for $w \leqslant min\_w_l$ and $w > max\_w_h$

$$\kappa_{a,u}(w) = \begin{cases} \frac{1}{2(|B|-1)} \sum_{b\in B,\ b\neq a} \kappa^{pc}_{a,b}(min\_w_l) & ; w \leqslant min\_w_l \\ \sum_{b\in B,\ b\neq a} \kappa^{pc}_{a,b}(w) & ; min\_w_l < w \leqslant max\_w_h \\ \infty & ; w > max\_w_h \end{cases} \tag{4}$$

Define the approximation curve as follows:

$$\kappa^{pc}_{a,u}(w) = \begin{cases} \frac{1}{2(|B|-1)} \sum_{b\in B,\ b\neq a} \kappa^{pc}_{a,b}(min\_w_l) & ; w \leqslant min\_w_l \\ \sum_{i=0}^{n} \alpha_i w^i & ; min\_w_l < w \leqslant max\_w_h \\ \infty & ; w > max\_w_h \end{cases}$$

To determine the coefficients, $\alpha_0, \alpha_1, \ldots, \alpha_n$, we take a number of data points from $\kappa_{a,u}$, which are (w_1,d_1), $(w_2,d_2),\ldots,(w_m,d_m)$, calculated from (4) with $min\_w_l = w_1 < w_2 < \ldots < w_m = max\_w_h$. After that, the process of computing the coefficients based on the data points is identical to that of Section 4.2.

- *Case 2:* $* = cs$. The process of calculating $\kappa^{cs}_{a,u}$ is very similar to the above approach of calculating $\kappa^{pc}_{a,u}$, except that cubic spline is used in place of polynomial curve.
- *Case 3:* $* = pl$. The process of finding a polyline $\kappa^{pl}_{a,u}$ is identical to that of Section 4.4, except that the points that are inserted into the polyline do not have to be selected from the corner points. At each iteration, the point that has the greatest distance to the polyline is selected.

5.3. Star-with-bypass topology

We want the star topology to carry similar state information as the mesh topology. Namely, for a mesh link (a,b), $\kappa^*_{a,b} \approx \kappa^*_{a,u} + \kappa^*_{u,b}$ which means the area distance between the left side and the right side should not be too big. When the area distance is big for some mesh links, we add those mesh links (with their approximation curves) as bypasses to the star topology. An example is given in Fig. 6(iii). Following [2,5], in our simulations we always add $|B|$ bypass links between border-router pairs that have the largest distortion between the mesh topology and the star topology.

6. Simulation results

We use simulations to evaluate the accuracy of aggregated domain state when different approaches are used to approximate the state between border routers. We implement the new approaches of *polynomial curve* (PC), *cubic spline* (CS), and *polyline* (PL), together with the existing approaches of *line segment* (LS) [2], Korkmaz–Krunz (KK) [5], *best point* (BP) and *worst point* (WP). The domain topology will either be aggregated to a mesh or a star-with-bypass.

Refer to Fig. 4. The *best point* (BP) approach uses one point, composed of the best (largest) bandwidth w_h and the best (smallest) delay d_l of all representatives of a staircase. The *worst point* (WP) approach also uses one point, composed of the best bandwidth w_h and the worst delay d_h of all representatives. Their approximation curves are defined as follows:

$$\kappa^{bp}(w) = \begin{cases} d_l & ; w \leqslant w_h \\ \infty & ; w > w_h \end{cases}$$

$$\kappa^{wp}(w) = \begin{cases} d_h & ; w \leqslant w_h \\ \infty & ; w > w_h \end{cases}$$

What's unique about BP is that it is the most aggressive approach in state aggregation: its positive area distance $\Delta_+(\kappa, \kappa^{bp})$ is the largest among all approaches under comparison, and its negative area distance $\Delta_-(\kappa, \kappa^{bp})$ is the smallest (zero). On the other hand, WP is the most conservative approach: its positive area distance $\Delta_+(\kappa, \kappa^{wp})$ is zero, and its negative area distance $\Delta_-(\kappa, \kappa^{wp})$ is the largest among all approaches mentioned above. Therefore, BP and WP are suitable as benchmarks in comparison.

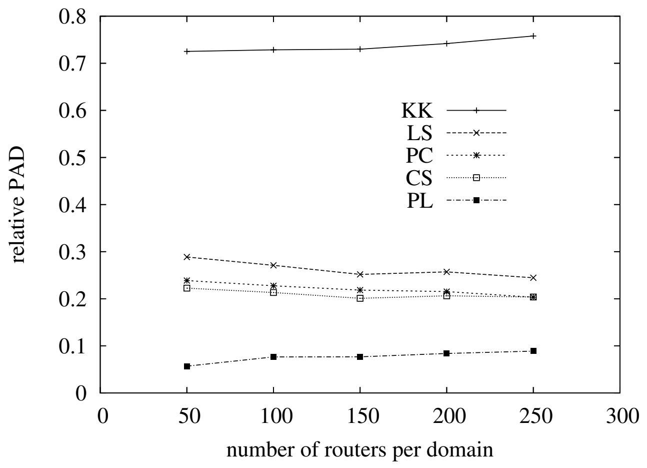
The approximation curves of PC, CS, PL, LS, and KK are denoted as κ^{pc}, κ^{cs}, κ^{pl}, κ^{ls}, and κ^{kk}, respectively. We define three performance metrics for comparison in our simulations. For $\kappa^* \in \{\kappa^{pc}, \kappa^{cs}, \kappa^{pl}, \kappa^{ls}, \kappa^{kk}\}$,

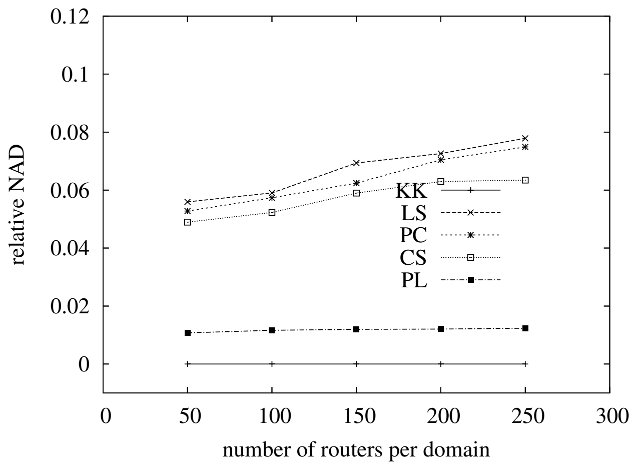
$$\text{relative PAD of } \kappa^* = \frac{\Delta_+(\kappa, \kappa^*)}{\Delta_+(\kappa, \kappa^{bp})}$$

$$\text{relative NAD of } \kappa^* = \frac{\Delta_-(\kappa, \kappa^*)}{\Delta_-(\kappa, \kappa^{wp})}$$

$$\text{relative AD of } \kappa^* = \frac{\Delta(\kappa, \kappa^*)}{\Delta(\kappa, \kappa^{bp})}$$

where PAD, NAD, and AD are abbreviations for positive area distance, negative area distance, and area distance, respectively (Section 4.1). These performance metrics measure the aggregation accuracy that an approximation approach can achieve after a domain topology is aggregated to a mesh or star-with-bypass topology. For example, if the relative PAD of an approximation approach is 0.5, it means that the size of mistakenly included regions of unsupported services is 50% of the size caused by BP. If the relative NAD is 0.5, it means that the size of mistakenly excluded regions of supported services is 50% of the size caused by WP. If the relative AD is 0.5, it means that the combined size of mistakenly classified service regions is 50% of the size caused by BP. In short, an approximation





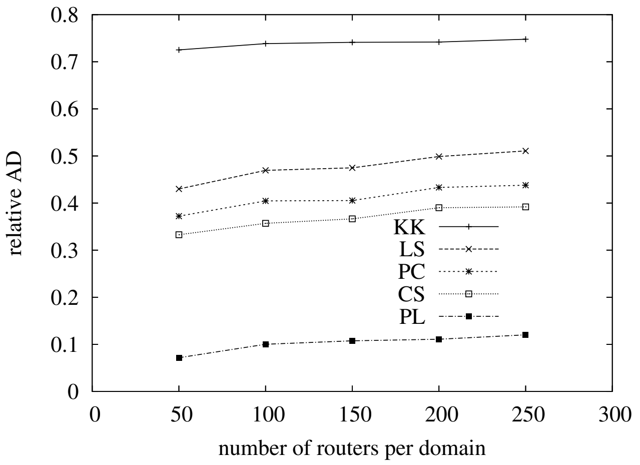


Fig. 7. Comparing aggregation accuracy of different approximation approaches after a domain topology is transformed to a mesh topology. PC, CS, and PL outperform LS and KK. PL is the best.

approach achieves better accuracy if its relative PAD/NAD/AD are smaller.

The domain topologies are randomly generated based on the Waxman model [17], which has been widely used to model intranet topologies. More specifically, the topology of each domain is created as follows: the routers are randomly placed in a one-by-one square, and the probability of creating a link between router u and router v is

$$p(u, v) \propto e^{-d(u,v)/\beta L}$$

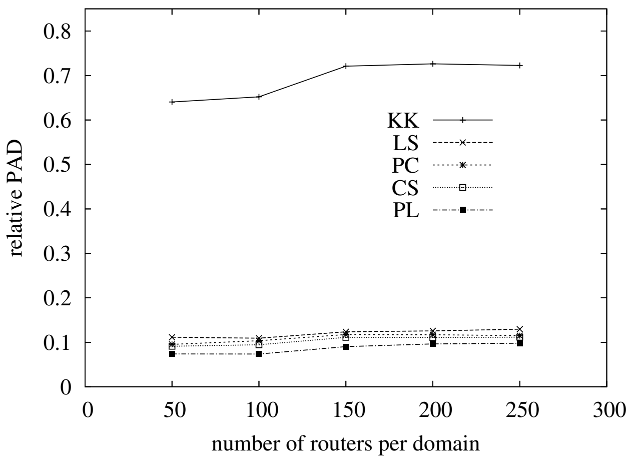
where $d(u, v)$ is the distance between u and v, $\beta = 0.6$, and L is the maximum distance between any two routers. The average node degree is 4. The other simulation parameters are: The number of routers in each domain is selected between 50 and 250. There are five border routers. The delays (bandwidths) of the physical links are randomly generated, following an exponential distribution with an average of 100 units. For delay, a unit may represent certain number of millisecond; for bandwidth, a unit may represent certain number of kilobytes (per second).

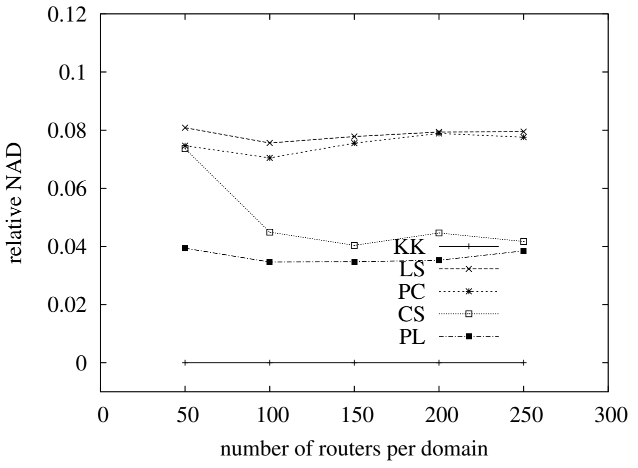
Each data point is the average of 100 simulation runs, with independently generated domain topologies and link states. More specifically, after a domain is generated, for each pair of border routers, we compute the service staircase. Then different approximation curves are used to approximate the staircase, and their relative PAD, NAD, and AD are calculated. After the above process is repeated on 100 different domains, we take the average results.

6.1. Comparing different approximation approaches

We compare the aggregation accuracy achieved by different approximation approaches after a domain topology is transformed to a mesh or a star-with-bypass topology. The simulation results show that the proposed PC, CS, and PL work better than the existing KK and LS. PL performs best.

Fig. 7 shows the aggregation accuracy after a domain topology is transformed to a mesh topology among border routers. The horizontal axis is the domain size. The top plots present the relative PAD values and the relative NAD values of various approximation approaches. The bottom plot presents the relative AD values. The simulation results show that PC, CS, and PL outperform LS





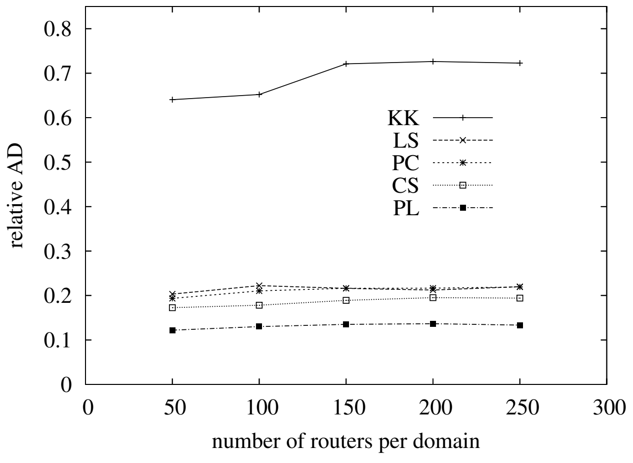


Fig. 8. Comparing aggregation accuracy of different approximation approaches after a domain topology is transformed to a star-with-bypass topology. PC, CS, and PL outperform LS and KK. PL is the best.

and KK. The performance of PC and CS is comparable. PL is the best among all. Its size of misclassified service regions is roughly one fourth that of LS. KK has significant PAD (positive area distance), about 70% that of BP. By the nature of its design, the NAD (negative area distance) of KK is always zero (Fig. 3).

Fig. 8 shows the aggregation accuracy after a domain topology is transformed to a star-with-bypass topology. Similar results are observed although the improvement is less significant than that for the mesh topology. The reason is that the aggregation process of transforming a mesh to a star takes the average of the approximation curves of different router pairs. This process introduces extra distortion, which has relatively more impact on the approximation approaches that achieve better accuracy in the mesh topology. PL remains to be the best, considerably outperforming other approaches.

The space requirements of KK and LS are fixed, three float numbers for KK and 4 for LS. The minimum space requirements for PC, CS, and PL are 6, 7, and 4, respectively. In this simulation they all use eight float numbers. One might argue that, although PL outperforms LS, it is an "unfair" comparison because PL uses more space. LS is a special case of PL. If PL also uses four floats, it becomes LS. The design of PL, as well as PC and CS, provides the flexibility of making tradeoff between accuracy and space. The point is that better accuracy is not achieved for free. The cost is more space. Our new approximation approaches provide the system designers with more choices. If accuracy is considered to be very important and some additional space can be offered, we show that trading space for accuracy is possible. Furthermore, Not all higher-order approximation curves work equally well. PL is the best choice.

6.2. Space/accuracy tradeoff by PC, CS, and PL

Fig. 9 compares PC, CS, and PL for accuracy/space tradeoff when a domain topology is aggregated to a mesh topology. The horizontal axis is the space (number of floats) used to store an approximation curve. The top plots compare the relative PAD values and the relative NAD values of different approximation approaches. The bottom plot compares the relative AD values. Because the minimal space requirement of CS is 7, there is no data point for CS when the number of floats is 6. PC, CS, and PL all achieve better approximation accuracy when the space increases. PC and CS are comparable. PL achieves the best

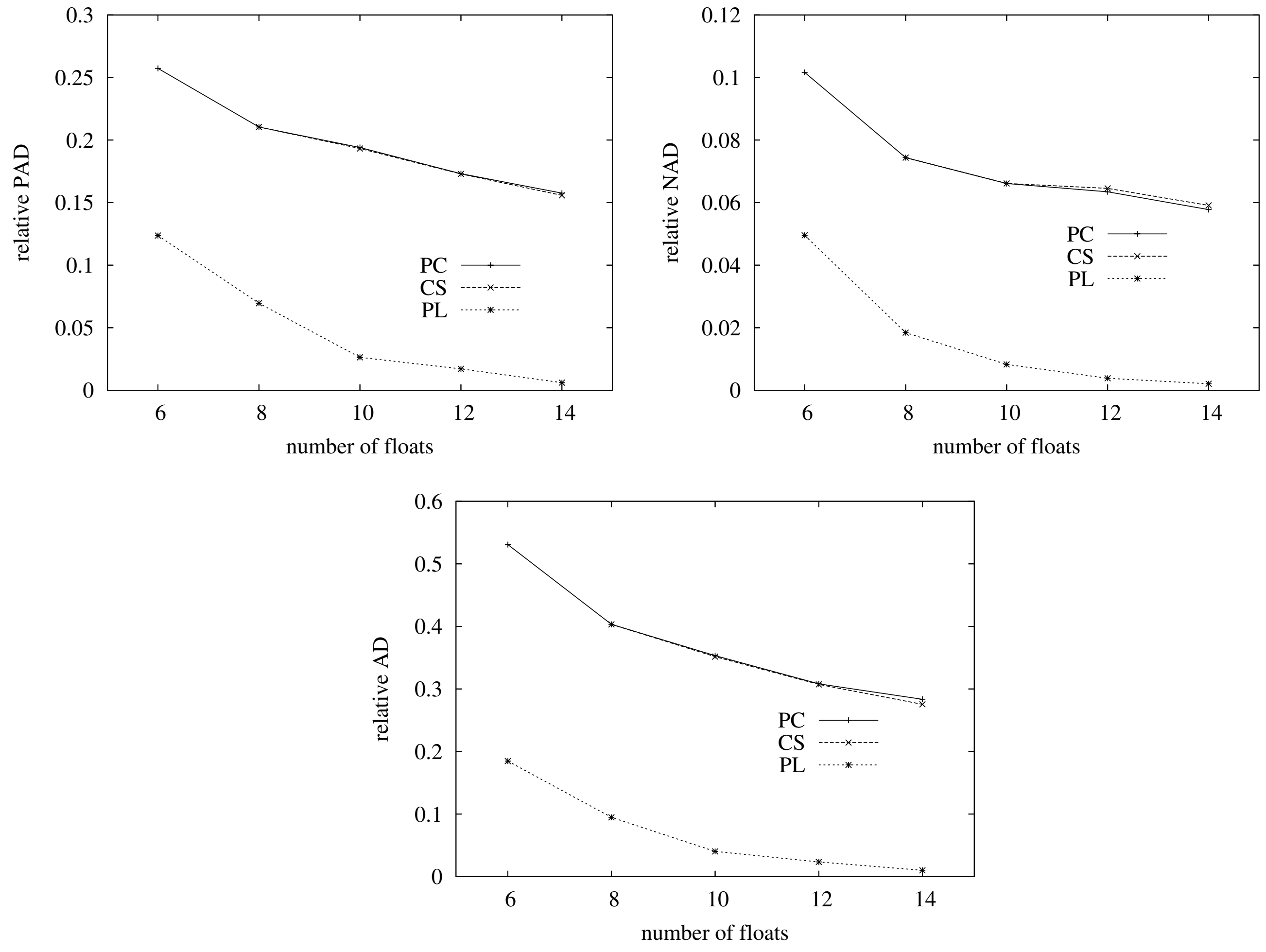


Fig. 9. Accuracy/space tradeoff in mesh topology.

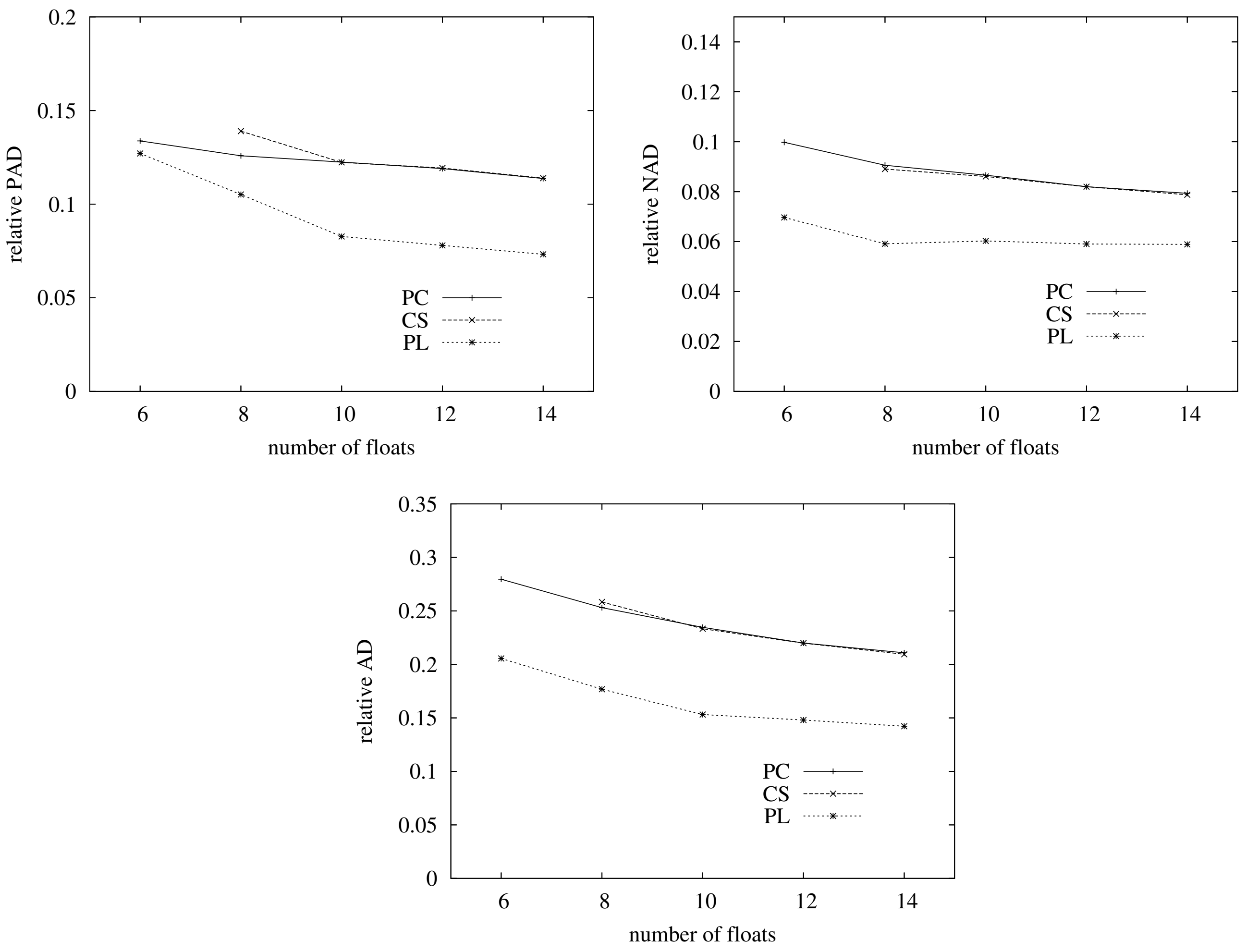


Fig. 10. Accuracy/space tradeoff in star-with-bypass topology.

accuracy/space tradeoff. When the space is relatively small (6–10), the rate of improvement by PL is large. When the space increases further, the rate of improvement becomes less and less significant. Therefore, the ideal space requirement for PL is around 10 floats.

Fig. 10 compares the accuracy/space tradeoff when a domain topology is aggregated to a star-with-bypass topology. Again, PC and CS are comparable. PL performs the best.

7. Conclusion

This paper studies how to aggregate the state information of large network domains by using service approximation curves. We propose new approximation curves that outperform the existing ones. We describe the algorithms for calculating these curves and analyze the time/space complexities of the algorithms. The new curves capture the global shape of the staircase, as well as the local subtlety, depending on the space used to store the curves. Better accuracy can be achieved with larger space. We discuss how to aggregate a domain topology to a logical mesh, star, or star-with-bypass topology when the new approximation curves are used. We compare the proposed approximation approaches with the existing ones by extensive simulations. The results show that polynomial curves, cubic splines, and polylines significantly outperform Korkmaz–Krunz curves and line segments. Polylines are the best among all. In the future work, we will extend the proposed methods to aggregate network state of higher dimensions with other parameters than delay and bandwidth. Examples are delay jitter and cost.

References

[1] F. Hao, E.W. Zegura, On Scalable QoS Routing: Performance Evaluation of Topology Aggregation, in: IEEE INFOCOM'00, 2000, pp. 147–156.

[2] K.-S. Lui, K. Nahrstedt, S. Chen, Routing with topology aggregation in delay–bandwidth sensitive networks, IEEE Transactions on Networking 12 (1) (2004).

[3] W.C. Lee, Spanning Tree Method for Link State Aggregation in Large Communication Networks, in: IEEE INFOCOM'95, vol. 1, Boston, MA, USA, Apr. 1995, pp. 297–302.

[4] A. Iwata, H. Suzuki, R. Izmailov, B. Sengupta, QOS aggregation algorithms in hierarchical ATM networks, in: 1998 IEEE International Conference on Communications. ICC 1998, vol. 1, Atlanta, GA, USA, June 1998, pp. 243–248.

[5] T. Korkmaz, M. Krunz, Source-oriented topology aggregation with multiple QoS parameters in hierarchical networks, ACM Transactions on Modeling and Computer Simulation (TOMACS) 10 (4) (2000) 295–325.

[6] ATM Forum, Private network–network interface specification version 1.0," f-pnni-0055.000, Mar. 1996.

[7] W. Lee, Topology Aggregation for Hierarchical Routing in ATM Networks, in: ACM SIGCOMM'95, 1995, pp. 82–92.

[8] W.C. Lee, Minimum Equivalent Subspanner Algorithms for Topology Aggregation in ATM Networks, in: 2nd International Conference on ATM (ICATM), vol. 1, Atlanta, GA, USA, June 1999, pp. 351–359.

[9] B. Awerbuch, Y. Shavitt, Topology aggregation for directed graphs, IEEE/ACM Transactions on Networking (TON) 9 (1) (2001) 82–90.

[10] S. Chen, K. Nahrstedt, An overview of quality of service routing for next-generation highspeed networks: problems and solutions, IEEE Network 12 (6) (1998) 64–79.

[11] D. Bauer, J.N. Daigle, I. Iliadis, P. Scotton, Efficient frontier formulation for additive and restrictive metrics in hierarchical routing, in: 2000 IEEE International Conference on Communications. ICC'2000, vol. 3, New Orleans, LA, USA, June 2000, pp. 1353–1359.

[12] V. Sarangan, D. Ghosh, R. Acharya, Performance Analysis of Capacity-aware State aggregation for Inter-domain QoS routing, in: IEEE Globecom'2004, Dallas, TX, Dec. 2004, pp. 1458–1463.

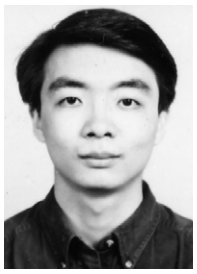
[13] Z. Wang, J. Crowcroft, Bandwidth–Delay Based Routing Algorithms, in: IEEE Proceedings of Globecom'1995, vol. 3, 1995, pp. 2129–2133.

[14] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, IEEE Journal on Selected Areas in Communications, vol. 14, no. 7, ,1996, pp. 1228–1234.

[15] P. Dierckx, Curve and Surface Fitting with Splines, Oxford University Press, Inc., Oxford, England, 1993.

[16] L. Guo, I. Matta, On state aggregation for scalable QoS routing, in: IEEE Proceedings of the ATM Workshop'98, may 1998, pp. 306–314.

[17] B.M. Waxman, Routing of multipoint connections, IEEE Journal of Selected Areas in Communications (1988) 1617–1622.



Yong Tang (yongt@sunbelt-software.com) received his BS degree from Peking University, Beijing, China, in 1999. He graduated with his Ph.D. degree in Computer Science from University of Florida in 2006. He is currently with Sunbelt Software Inc. His primary research field is network security, especially on mitigating distributed denial-of-service attacks and Internet worm attacks.

Shigang Chen (sgchen@cise.ufl.edu) received his B.S. degree in computer science from University of Science and Technology of China in 1993. He received M.S. and Ph.D. degrees in computer science from University of Illinois at Urbana-Champaign in 1996 and 1999, respectively. After graduation, he had worked with Cisco Systems for three years before joining University of Florida as an assistant professor in the Department of Computer & Information Science & Engineering. His research interests include wireless networks, Internet security, and overlay networks. He received IEEE Communications Society Best Tutorial Paper Award in 1999. He was a guest editor for ACM/Baltzer Journal of Wireless Networks (WINET) and IEEE Transactions on Vehicle Technologies. He served as a TPC co-chair for the Computer and Network Security Symposium of IEEE IWCCC 2006, a vice TPC chair for IEEE MASS 2005, a vice general chair for QShine 2005, a TPC co-chair for QShine 2004, and a TPC member for many conferences including IEEE ICNP, IEEE INFOCOM, IEEE SANS, IEEE ISCC, IEEE Globecom, etc. His email address is sgchen@ cise.ufl.edu.

Dr. Yibei Ling is research scientist at applied research, Telcordia technologies (formerly Bellcore). His research interests include distributed computing, query optimization in database management system, scheduling, checkpointing, system performance, fault localization and self-healing in mobile ad hoc network and power-aware routing in mobile ad hoc network. He has published several papers in IEEE Transactions on computers, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Biomedical Engineering, SIGMOD, ICDE, PODC, Information system. He is the architect, as well as developer, of the voice subsystem of Telcordia Notification System. Yibei received his BS in EE from Zhejiang University in 1982, his MS in statistics from Shanghai medical university (now Fuda University) in 1988, and his Ph.D in CS from Florida State University at Miami in 1995. He is a member of the IEEE.