A Distributed Routing Algorithm for Networks with Data-Path Services

Xin Huang, Sivakumar Ganapathy and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts Amherst, MA, USA
Email: {xhuang,sganapat,wolf}@ecs.umass.edu

***Abstract*—Many next-generation Internet architectures propose advanced packet processing functions in the data path of the network. Such "services" are typically performed on some nodes along the path of a packet. We present a novel decentralized algorithm that can determine an allocation of services to network nodes. The algorithm can obtain globally optimal solutions for a single service and approximate solutions for two or more services. In our simulation results, we validate the correctness of the algorithm, quantify the quality of the approximations, and compare the results to those of a centralized algorithm. Our results show that the proposed algorithm presents an effective solution to the service placement problem that can be implemented in a realistic network.**

I. Introduction

The Internet has evolved from a simple store-and-forward network to a communication platform that provides a variety of advanced data transmission features. Complex packet processing functions are implemented in the data paths of routers where packets can be inspected and modified as they traverse the network. Such functionality can target a variety of goals ranging from security (e.g., firewalls, intrusion detection) to performance (e.g., quality-of-service scheduling, protocol offloading and SSL termination) to application-layer support (e.g., content distribution and caching, peer-to-peer support).

In the current Internet, these features are shimmed into the existing network architecture, which does not support functional extensions to the network data path. To overcome this ad-hoc deployment of point solutions, new network architectures have been proposed where so-called "network services" are an inherent feature of the data path. New protocols and network functions can be deployed by extending the set of services offered by network nodes. Deployment of these experimental network architectures is currently explored by the networking community in the context of next-generation Internet research.

Routing has been widely studied in conventional networks and several centralized and distributed algorithms are widely deployed. In the context of networks where services are provided in the data path, routing becomes a much more complex problem that goes beyond simply determining the shortest path between two nodes. Services that are required for a end-to-end connection may be available only on some nodes that are not located along the shortest end-to-end path. In such a case, it is necessary to determine through which nodes traffic should flow. When more services are necessary, the problem becomes increasing difficult.

In this paper, we address this routing problem in networks that provide data-path services. In prior work, only centralized algorithms have been proposed to solve the routing problem. However, in large deployments, centralized algorithms suffer from scalability problems. In this paper, we present a distributed algorithm, which uses heuristics for solving the routing problem, and thus can achieve the necessary scalability. We describe how to represent the necessary information that nodes need to exchange and how per-flow routing decisions are made. We also compare our algorithm experimentally to the centralized algorithm to evaluate how well our heuristics perform.

Specifically, the contributions of this paper are:

- A formalization of the routing problem that is encountered in networks that provide data-path services.
- A novel distributed algorithm, called Distributed Service Matrix Routing (DSMR), that can find the optimal path for up to one service and a near-optimal path for two or more services.
- An extensive evaluation of this routing algorithm to quantify the quality of approximated routes provided by our heuristics.

The remainder of the paper is organized as follows. Section II discusses related work. Section III presents the formal problem statement and existing centralized solution. In Section IV, we introduce our distributed routing algorithm. Section V presents experimental results from our prototype implementation. Section VI concludes this paper.

II. Related Work

Extensions to the feature set of the original Internet architecture [1] have been proposed in many forms. Such features have either been deployed to solve specific problems (e.g., network address translation to allow IP address reuse [2], firewalls to improve security [3]) or as general concepts for data path flexibility (e.g., active networks [4] or programmable routers [5] or configurable protocols configurable protocol stacks [6] an protocol heaps [7]).

To manage the complexity of these new features, an IETF working group has attempted to define Open Pluggable Edge Services (OPES) [8]. In such an architecture, end-systems can specify a set of data flow operations that are implemented

on nodes throughout the network. TCP tunnels are used to create a multi-hop end-to-end stream. In the context of connection establishment, Guha and Francis address the problem of traversing multiple "middle-boxes" in the current Internet [9]. While their work focuses specifically on traversing firewalls and network address translators, it can be seen as a step towards managing connections involving general services.

For next-generation networks, where we can consider a deviation from the current Internet architecture [10], our prior work has described the concepts of network services as first-class networking functions [11]. A more specific architecture for the implementation of such services was described recently [12]. We envision that these services could be implemented on a variety of platforms ranging from workstations routers [13] to programmable routers [14] and virtualized router platforms [15]. For next-generation networks, it is important to consider network services that range from application-layer end-system service (e.g., services in a grid computing environment [16]) to packet processing services on routers (e.g., data path services for the next-generation Internet [11]) to ensure flexibility to adapt to novel network uses that we cannot yet predict.

To address the routing problem in networks with service nodes, Choi et al. have developed a centralized algorithm algorithm to solve the service placement problem [17], which was later expanded to consider capacity constraints [18]. We use this algorithm as a comparison for our distributed algorithm. The distributed algorithm presented in our work uses a "service matrix," which 2-dimensional extension to the distance vector used in existing routing protocols (e.g., RIP [19]).

III. Network Services and Service Placement Problem

Before presenting the Distributed Service Matrix Routing algorithm in Section IV, we first discuss network services in more detail, formalize the routing problem, and present the state of the art centralized solution.

A. Network Services

Network services encompass all operations that occur in the data path of routers. Note that occasionally, the term "network service" is used in the context of application layer services that can be accessed via the network (e.g., access to data, storage, or computation). We mainly focus on functionality that is located in the data link, network, and transport layers. Examples of such services include firewalls, intrusion detection, QoS scheduling, VPN or SSL tunneling, content transcoding, reliable transmission, forward error correction in wireless networks, content distribution, etc.

We envision a future network architecture, where end-to-end connections are established with a specification of the sequence of services that need to be performed on data that is transmitted. This allows end-system applications to specify an ideal configuration of network support and allows the network to provide this support due to the explicit communication of application needs. More details about this architecture can be found in [11], and a concrete system design is described in [12].

The implementation of the actual services is accomplished via programmable routers that use network processors or other packet processing systems to handle data plane operations. Such programmable routers have been designed, and high-performance embedded network processors are commercially available. In this work, we focus on the control plane of such network service architecture and address the question of how to manage these data path operations in network with a large number of processing nodes.

When considering the control plane, it is important to note that there are two dimensions of protocol and service composition:

- Functional Dimension: The functional dimension determines *which* components and functions are combined to create a novel protocol stack. This leads to a sequence of services that need to be performed along the end-to-end communication path. Examples of such functional composition have been shown in the context of transport layer protocols [20] and mobile and ad-hoc routing protocols [21].
- Spatial Dimension: The spatial dimension determines *where* the processing services that are related to protocol processing are placed. The distributed nature of network services requires that suitable processing nodes and communication paths between them can be found.

In this paper, we focus on the second dimension, i.e. where to place service processing tasks, and assume that a suitable functional composition is given. In particular, we present a distributed algorithm that can exchange the necessary routing information to allow each node to determine which services of a connection request to process and in which direction to forward traffic.

B. Routing Problem in Network Service Context

Routing traffic that requires various services through a network can be divided into two subproblems: (1) determining the placement of services (i.e., which node should perform the processing of packets), and (2) determining the shortest path between these service nodes. The latter – finding the shortest path – is a well-understood problem that had been solved. Determining which nodes should be used for services is more challenging. We state this problem, which we call the *service placement problem*, more formally in this section.

A service placement problem is defined as follows. The *network* is represented by a weighted graph, $G = (V, E)$, where nodes V correspond to routers and end systems and edges E correspond to links. Each edge $e_{i,j}$ that connects nodes v_i and v_j is labeled with a weight $w_{i,j}$ that represents the communication cost (e.g., delay). Each node v_m is labeled with the set of services that it can perform $u_m = \{S_j | \text{service } S_j \text{ is available on } v_m\}$ and the processing cost $c_{m,j}$ (e.g., processing delay) of each service. A *connection request* is represented as $R = (s, t, (S_{j_1}, \ldots, S_{j_k}))$, where s is the source node, t is the destination node, and $S_{j_1}, \ldots, S_{j_k}$ is an ordered list of services that are required for this connection.

Given a network G and a request R, find a path for the connection such that source and destination are connected and all services can be processed along the path. The path is defined as $P = (E^P, M^P)$ with a sequence of edges, E^P, and services mapped to processing nodes, M^P: $P = ((e_{s,v_{i_1}}, \ldots, e_{v_{i_h},t}), (S_{j_1} \rightarrow v_{m_1}, \ldots, S_{j_k} \rightarrow v_{m_k}))$. To determine the quality of a path, we define the total cost $C(P)$ of accommodating connection request R as the sum of link cost and processing cost: $C(P) = \left(\sum_{\{(x,y)|e_{x,y} \in E^P\}} w_{x,y}\right) + \left(\sum_{\{(j_i,m_i)|S_{j_i} \rightarrow v_{m_i} \in M^P\}} c_{m_i,j_i}\right)$.

In the context of this problem statement, we make several assumptions. We assume routes through the network can be different for every connection (e.g., similar to source routing), but are fixed for the lifetime of each connection. We also assume that the type and order of services is known at connection setup time and does not change over the lifetime of the connection. In many cases, it is desirable to find the *optimal* connection setup. We view this optimality in terms of the least cost allocation of a single connection request.

C. Centralized Solution

To solve the problem state above, a solution has been proposed by Choi et al. in prior work [17]. This solution is a "centralized" solution in the sense that it requires a complete view of the network in order to optimally allocate a connection request. The main idea is to represent the network as a graph with multiple layers, where each layer indicates which service of the request has already been processed. A single cost metric for both communication and processing is used. The optimal path can be found by simply employing Dijkstra's shortest path algorithm on the multi-layer graph and mapping it back to the original network.

The construction of this so-called "layered graph" is done as follows: To route a request $R = (s, t, (S_{j_1}, \ldots, S_{j_k}))$, a total of $k + 1$ instances of the original network are used to represent the graph layers. The top layer (layer 0) is used for communication before service S_{j_1} is performed. The next layer (layer 1) represents communication that is performed after service S_{j_1} (and before service S_{j_2}) is completed. Layers $i-1$ and i are connected with vertical edges on all nodes v_m, where service S_{j_i} processing is possible (i.e., $S_{j_i} \in u_m$). The costs of these vertical edges are c_{m,j_i}, which correspond to the cost of processing service S_{j_i} on node v_m. Vertical edges are added between all $k + 1$ layers. Then, using Dijkstra's algorithm, the shortest path is calculated from the layer 0 instance of source node s to the layer k instance of destination node t. The resulting path provides the least cost connection from the source to the destination while ensuring that all services are performed in sequence (due to vertical edges). To obtain the final path in the original network, all nodes and edges are projected onto a single instance of the network. Vertical edges in the path correspond to service placements and horizonal edges are used to connect the path. An example of a layered graph solution for the problem shown in Figure 1 is shown in Figure 2.

As has been shown in [17], the layered graph approach provides an optimal solution when services are requested in a given order and a single cost metric for communication and processing is used. Also, link capacities and processing capacities are considered to be unconstrained. In case of capacity constraints, the routing problem becomes NP-complete and heuristics can be applied [17]. The computational complexity of the layered graph algorithm is simply that of Dijkstra's algorithm on the layered graph. Since it contains $\mathcal{O}(k)$ times as many nodes and edges as the original graph, the complexity is $\mathcal{O}(k(|E| + |V|) \log(k|V|))$.

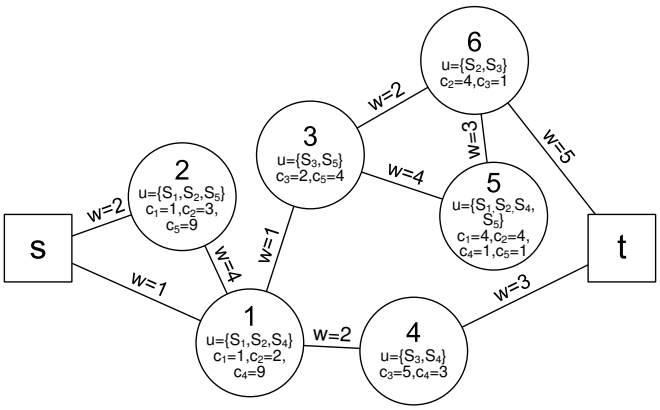


Fig. 1. Example Topology for Service Placement Problem.

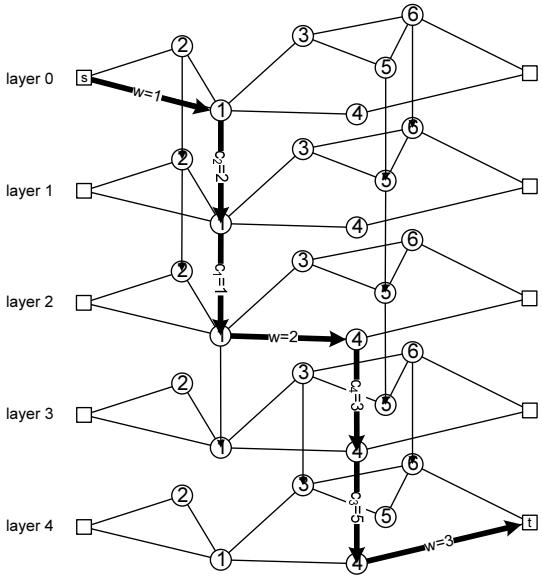


Fig. 2. Layered Graph Algorithm Solution for $R = (s, t, (S_2, S_1, S_4, S_3))$. The total cost of the path is 17.

There are several fundamental limitation of the layered graph algorithm that limit its scalability to large network deployment:

- All nodes require a complete view of the network including all links and nodes.
- For each routing request a different layered graph may need to be constructed and a shortest path algorithm needs to be calculated.

To address these issues, we present a distributed algorithm that can solve the service placement problem.

IV. Distributed Service Matrix Routing Algorithms

A centralized solution in the form of the layered graph algorithm requires global knowledge about all links and nodes in the network. This approach is not scalable and thus is difficult to be implemented on a large network topology. Furthermore, the centralized solution is not compatible with the concept of "hierarchical topologies," where groups of nodes are organized into Autonomous Systems (ASs) and detailed topology information from inside an AS is not visible to other ASs. Instead, we show that a decentralized solution, in form of the Distributed Service Matrix Routing (DSMR) algorithm presented in this section, is more suitable for large and hierarchical networks.

We structure our discussion of DSMR, by distinguishing between routing exchange operations, where information pertinent to routing and service placement is exchanged between routers, and request routing operations, where connection requests (and thus traffic) are routed. First, we describe the general concepts of our algorithm.

A. Service Placement as Dynamic Programming Problem

To illustrate the challenges of routing in a service network, consider a very simple scenario: Two end-systems are connected through a sequence of routers such that there is only a single path connecting them. A connection setup request that asks for a sequence of services to be performed along the way is issued. As the connection request traverses the path, how can a router determine if it should perform a service or if it is better to pass the service request to a downstream router that may be able to perform it at lower cost? In particular, how can this question be answered without having a complete view of the network?

We have discovered that a solution to this problem can be obtained by using a dynamic programming approach similar to what Bellman proposed for shortest path routing [22]. Let $c_v^{j_1,\dots,j_k}(t)$ denote the cost of the shortest path from node v to node t where services $S_{j_1},\dots,S_{j_k}$ are performed along the way. For shortest path computation (i.e., no services), we use the notation $c_v^-(t)$. Thus, a node v can determine the least cost path by considering to process i ($0 \le i \le k$) services and forwarding the request to any neighboring node n_v ($n_v \in \{x \in V | e_{v,x} \in E\}$):

$$c_v^{j_1,\dots,j_k}(t) = \min_{0\le i\le k}\left(\sum_{l=1}^{i} c_{v,j_l} + \min_{n_v}\left(w_{v,n_v} + c_{n_v}^{j_{i+1},\dots,j_k}(t)\right)\right). \quad (1)$$

The argument i on the right side determines how many of the k services that need to be performed should be processed on node v. Note that if $i = 0$, no service is processed. The argument n_v determines to which neighbor of v the remaining request should be sent.

B. DSMR Implementation

Using the above solution, we can design a distributed dynamic programming algorithm to solve the service placement problem.

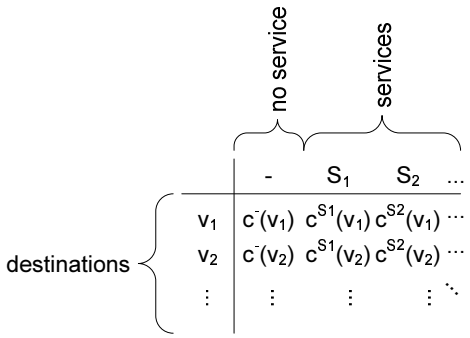


Fig. 3. Service Matrix Data Structure.

1) Routing Exchange: Clearly, a distributed solution to this problem requires that nodes exchange their $c_v^*(*)$ values ('*' represents any service sequence or any destination) to neighboring nodes (similar to how it is done in RIP for distance vector routing [19]). To capture this information, we define a control plane data structure called "service matrix" (see Figure 3). This matrix is a two-dimensional extension of a distance vector, where the second dimension is an enumeration of all possible services. As in distance vector routing, the first dimension is a list of destinations (or destination prefixes).

For the routing exchange, each node advertises its service matrix to its neighbors. Thus, all nodes can collect all $c_n^*(*)$ values from their neighbors and thus solve Equation 1 to populate their own service matrices.

2) Request Routing: When handling connection routing and setup, a request $R = (s, t, (S_{j_1},\dots,S_{j_k}))$ is propagated from s to t. A node v along the path needs to determine from its service matrix which services $S_{j_1},\dots,S_{j_i}$ should be processed locally. This can be done simply by looking up $c_v^{S_{j_1},\dots,S_{j_k}}(t)$. Rather than having to compute Equation 1 for each connection request, we assume that the control plan precomputes $c_v^*(*)$ and the corresponding number of local services i and next hop n_v. Additionally, the list of nodes where services are allocated, $v_{m_1},\dots,v_{m_k}$, can also be stored (and exchanged with neighbors).

Using a simple lookup into the service matrix, a node can determine that it should allocate i services to itself and pass a request for the remaining services along to its neighbor n_v (i.e., $R' = (s, t, (S_{j_{i+1}},\dots,S_{j_k}))$). By the time the request reaches t, all services (except for those that may be optimally placed onto t) are allocated to nodes along the path.

3) Practical Constraints: DSMR provides the globally optimal path and service allocation under the following assumptions:

- The information exchange between routers has converged to stable service matrices.
- All possible service sequences $S_{j_1},\dots,S_{j_k}$ are listed in these service matrices.

The first point has been sufficiently addressed in the context of shortest path routing and, for the purpose of this paper, we assume that all routes are stable. The second point is more critical for a practical implementation. If there are $|S|$ different services available in a network and all of them could be combined arbitrarily to service request of maximum length

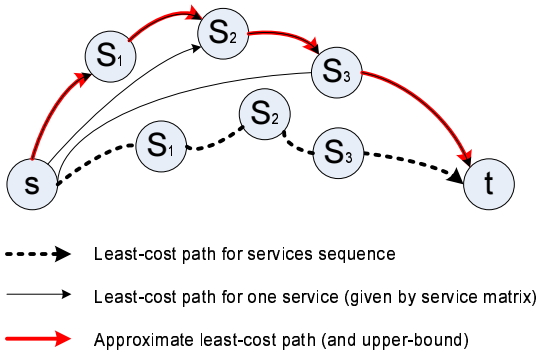


Fig. 4. DSMR Approximation and Path Bounds.

k_{max}, a total of of $\mathcal{O}\left(|S|^{k_{max}}\right)$ columns would be required in the service matrix. Clearly, that does not provide a level of scalability necessary for large networks.

To address this problem, we discuss an approximate solution to the placement problem, where only the routing information for no services or single service (i.e., $|S|+1$ columns in the service matrix) are necessary.

C. Approximate DSMR

To reduce the size of the service matrix, we can use an approximation that requires only information about the optimal placement of single services, no matter what sequence of service are requested. The idea is illustrated in Figure 4 for a sequence of three services.

The approximate DSMR algorithm processes the sequence of services from request $R = (s, t, (S_{j_1}, \ldots, S_{j_k}))$, from back to front. First, the optimal path between s and t for S_{j_k} is determined. Since S_{j_k} is a single service, it's optimal path and allocation to node v_{m_k} can be determined from the reduced service matrix, which contains only $|S|+1$ columns. This step corresponds to placing S_3 in Figure 4. With S_{j_k} placed, the same service matrix can now be used to determine where to place $S_{j_{k-1}}$ along a path from s to v_{m_k}. This step corresponds to placing S_2 in Figure 4. The process is repeated until all services have been placed.

Using this approximate placement algorithm, a node can compute an upper bound for the optimal path cost $C(P^{opt})$: The upper bound path, P^+, is the path that is determined by the approximate DSMR algorithm. Its cost is the sum of all paths for individually placed services minus the redundant portions of the path. Thus, $C(P^+) = \sum_{i=1}^{k-1} \left(c_s^{S_{j_i}}(v_{m_{i+1}}) - c_s^-(v_{m_{i+1}})\right) + c_s^{S_{j_k}}(t)$. Since $c_s^{S_{j_i}}(v_{m_{i+1}}) > c_s^-(v_{m_{i+1}})$, the sum within the expression yields a positive cost.

An important question is how tight the upper bound is, since it determines the quality of the approximate DSMR algorithm compared to the optimal solution (provided by either DSMR or the layered graph algorithm).

V. Evaluation of DSMR

We evaluate DSMR and approximate DSMR in comparison the to the centralized layered graph algorithm to quantify their effectiveness in solving the service placement problem. For the entire evaluation, we assume that approximate DSMR maintains a service matrix for single services (i.e., $k+1$ columns).

TABLE I
Time and Space Complexity of Algorithms

Algorithm	Time (route lookup)	Space
Layered graph	$\mathcal{O}(k(\|E\|+\|V\|)\log(k\|V\|))$	$\mathcal{O}(k\cdot(\|E\|+\|V\|))$
DSMR	$\mathcal{O}(1)$	$\mathcal{O}\left(\|S\|^{k_{max}}\cdot\|V\|\right)$
Approximate DSMR	$\mathcal{O}(k)$	$\mathcal{O}(\|S\|\cdot\|V\|)$

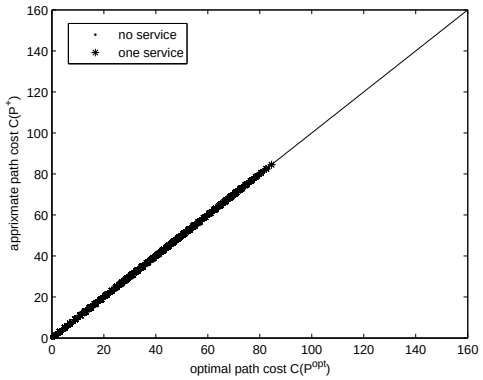
A. Algorithmic Complexity

The time and space complexity of DSMR is compared to the layered graph algorithm in Table I. The centralized layered graph algorithm requires a significant amount of time to perform a route lookup since it needs to create the layered graph on demand and compute the shortest path. In contrast, DSMR requires only constant time to perform a lookup since the service matrix contains the optimal path for all possible destinations and service combinations. However, the data structure size for DSMR is prohibitively large. Approximate DSMR strikes a good balance between lookup time and space requirement. It's lookup time is proportional to the number of services. The service matrix data structure is also small and proportional in size to the number of services and destination nodes. This comparison, again illustrates the benefits of using a distributed algorithm over a centralized algorithm, but also highlights the need for using approximate DSMR over the original DSMR to limit space requirements.

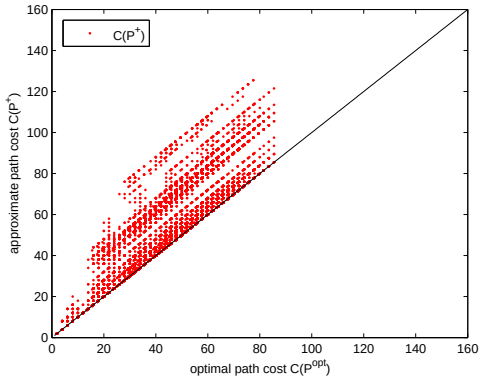
B. Simulation Results

To quantify the quality of approximate DSMR compared to the optimal DSMR or layered graph, we built a prototype of a network with 14 ASs (56 nodes) on Emulab and implemented the approximate DSMR for routing between these ASs. These results are compared with results from a corresponding simulation of the same topology which uses layered graph for routing. Each of the nodes is capable of performing between zero and four different types of services. After the service matrix exchange has stabilized in the emulab prototype, we evaluate approximate DSMR by iterating through connection requests from all possible sources to all possible destination using all possible service combinations. The same requests are issued to the simulation of the layered graph algorithm. This process allows us to obtain the cost of the upper bound, $C(P^+)$ (i.e., the cost of the path that approximate DSMR calculates), and the cost of the optimal path, $C(P^{opt})$ (i.e., the cost of the layered graph result).

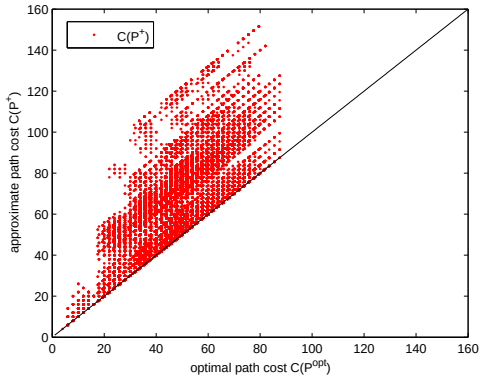
1) Correctness of DSMR: Figure 5 shows the optimal path cost $C(P^{opt})$ on the x-axis and the approximate DSMR path cost $C(P^+)$ on the y-axis. Each data point represents one connection request. In Figure 5(a), we can see that for $k = 0$ and $k = 1$, $C(P^+) = C(P^{opt})$ as all points fall on the diagonal. Thus, for zero or one service, approximate DSMR indeed provides the optimal solution as we expect. This results



(a) No Service ($k = 0$) and One Service ($k = 1$).



(b) Two Services ($k = 2$).



(c) Four Services ($k = 4$).

Fig. 5. Quality of Approximate DSMR Compared to Optimal Path for Requests with Different Numbers of Services.

validates that the service matrix exchange (i.e., DSMR) is a correct substitute of the layered graph program.

2) Approximation at Source: Figure 5(b) and 5(c) show the quality of approximate DSMR routes for two or more services ($k = 2$ and $k = 4$) compared to the optimal centralized algorithm. As expected, we observe that $C(P^+) \geq C(P^{opt})$. With more services ($k = 4$), the distance from the diagonal increases for some solutions, indicating the approximation decreases in quality. However, numerous solutions remain close to the optimal path.

We can quantify how many requests are served how well by considering the cumulative distribution function (CDF) shown

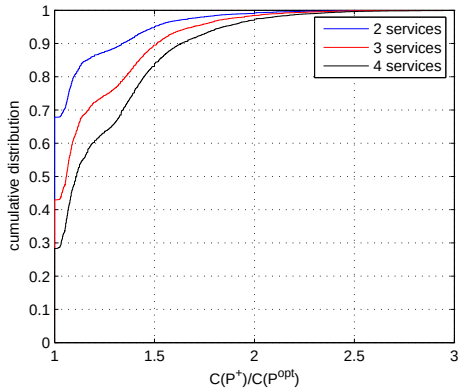


Fig. 6. CDF of Approximate DSMR Path Cost to Optimal Path Cost.

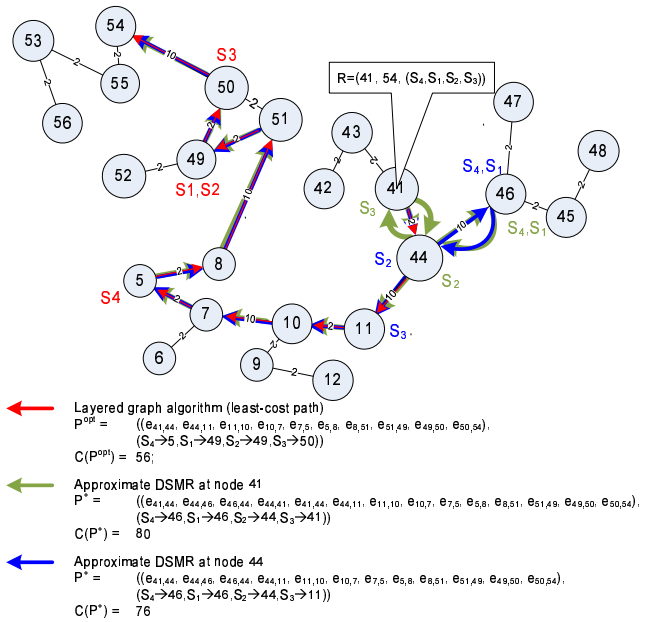


Fig. 7. Example Path For Layered Graph and Approximate DSMR Algorithm. Approximate DSMR may improve the path quality as request traverses the network. Computation cost is 1 for all services.

in Figure 6. The x-axis shows the relative cost the approximate DSMR solution relative to the optimal path cost (i.e., $C(P^+)/C(P^{opt})$). We can see that for two services, around 68% of requests are routed with identical cost as the optimal solution. More than 95% of the requests are less than 50% more costly than the optimal path (i.e., $C(P^+)/C(P^{opt}) \leq 1.5$). For four services, these percentages drop as it becomes more difficult to find the optimal path by approximation. Still, nearly 30% of requests are routed optimally, and 95% are less than double the optimal cost.

3) Approximation Along Path: The above results consider the quality of the path when comparing the view of the source node with the overall optimum. However, connection requests are passed from node to node. Along this process, the path quality may be improved. This effect is illustrated in Figure 7, where the optimal path for a connection request is compared to two approximate DSMR solutions. The first DSMR approximation is the path determined by the source

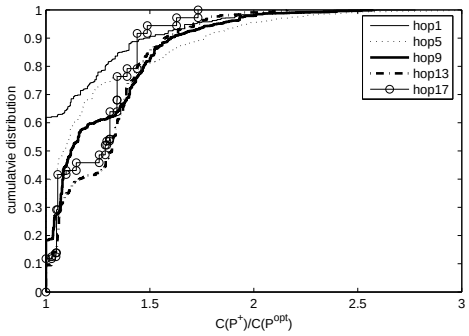


Fig. 8. CDF of Approximate DSMR Path Cost to Optimal Path Cost for Different Path Lengths and four Services ($k = 4$).

(i.e., node 41), which has a cost of $C(P^+) = 80$. When the connection request is passed to the next node (i.e., node 44), the new approximate DSMR computation yields a better path with $C(P^+) = 76$. Thus, the solution provided by approximate DSMR at the source really is a bound and may improve as requests traverse the network.

To further illustrate the improvement of routing as a request gets closer to the destination, Figure 8 shows the CDF for four services for different path lengths (counted in hops). We observe that short paths are more frequently close to the optimal path, whereas longer paths are less likely to be optimal. However, in the vast majority of all cases, the relative cost is less than 2 (i.e., $(C(P^+)/C(P^{opt}) \leq 2)$.

It is important to note that an approximation of an optimal path that yields nearly $2\times$ longer paths would be unacceptable in the current Internet. Since only shortest path routing (without any services) is currently used, optimality in finding the path is expected. However, when multiple services are required for a connection, longer paths and less tight approximations are acceptable. It may be acceptable to route a connection along a longer path in order to avoid the heavy computational overhead that finding the optimal solution with the layered graph algorithm would entail.

VI. Conclusions

In next-generation network architectures, services in the data path will be used to provide a diverse set of packet processing features. Network will need to be able to route connection requests such that nodes along the path can provide the requested service combinations. We present a novel distributed algorithm to solve the problem of finding optimal and approximate routes. Distributed Service Matrix Routing can find the optimal routes with a constant-time lookup in a data structure that may be very large depending on the number of services. Our approximate Distributed Service Matrix Routing algorithm can provide near-optimal routing information with significantly less space requirements and only a small increase in processing complexity. Our evaluation results show that the quality of the approximation is such that the majority of connection requests (for two services) still is routed along the optimal path. We believe that this algorithm provides an important step towards making network architectures with data path services a practical reality.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0626690.

References

[1] D. D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proc. of ACM SIGCOMM 88*, Stanford, CA, Aug. 1988, pp. 106–114.
[2] K. B. Egevang and P. Francis, "The IP network address translator (NAT)," Network Working Group, RFC 1631, May 1994.
[3] J. C. Mogul, "Simple and flexible datagram access controls for UNIX-based gateways," in *USENIX Conference Proceedings*, Baltimore, MD, Jun. 1989, pp. 203–221.
[4] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 2, pp. 5–18, Apr. 1996.
[5] T. Wolf, "Challenges and applications for network-processor-based programmable routers," in *Proc. of IEEE Sarnoff Symposium*, Princeton, NJ, Mar. 2006.
[6] N. T. Bhatti and R. D. Schlichting, "A system for constructing configurable high-level protocols," in *SIGCOMM '95: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, Cambridge, MA, Aug. 1995, pp. 138–150.
[7] R. Braden, T. Faber, and M. Handley, "From protocol stack to protocol heap: role-based architecture," *SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 17–22, Jan. 2003.
[8] A. Barbir, P. Reinaldo, R. Chen, M. Hofmann, and O. Hilarie, "An architecture for open pluggable edge services (OPES)," Network Working Group, RFC 3835, Aug. 2004.
[9] S. Guha and P. Francis, "An end-middle-end approach to connection establishment," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, Kyoto, Japan, Aug. 2007, pp. 193–204.
[10] A. Feldmann, "Internet clean-slate design: what and why?" *SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, Jul. 2007.
[11] T. Wolf, "Service-centric end-to-end abstractions in next-generation networks," in *Proc. of Fifteenth IEEE International Conference on Computer Communications and Networks (ICCCN)*, Arlington, VA, Oct. 2006, pp. 79–86.
[12] S. Ganapathy and T. Wolf, "Design of a network service architecture," in *Proc. of Sixteenth IEEE International Conference on Computer Communications and Networks (ICCCN)*, Honolulu, HI, Aug. 2007.
[13] N. C. Hutchinson and L. L. Peterson, "The x-kernel: An architecture for implementing network protocols," *IEEE Transactions on Software Engineering*, vol. 17, no. 1, pp. 64–76, Jan. 1991.
[14] L. Ruf, K. Farkas, H. Hug, and B. Plattner, "Network services on service extensible routers," in *Proc. of Seventh Annual International Working Conference on Active Networking (IWAN 2005)*, Sophia Antipolis, France, Nov. 2005.
[15] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.
[16] I. Foster and C. Kesselman, Eds., *The Grid – Blueprint for a New Computing Infrastructure*, 2nd ed. Morgan Kaufmann, 2004.
[17] S. Y. Choi, J. S. Turner, and T. Wolf, "Configuring sessions in programmable networks," *Computer Networks*, vol. 41, no. 2, pp. 269–284, Feb. 2003.
[18] S. Y. Choi and J. S. Turner, "Configuring sessions in programmable networks with capacity constraints," in *Proc. of IEEE International Conference on Communications (ICC)*, Anchorage, AK, May 2003.
[19] C. Hedrick, "Routing information protocol," Network Working Group, RFC 1058, Jun. 1988.
[20] G. T. Wong, M. A. Hiltunen, and R. D. Schlichting, "A configurable and extensible transport protocol," in *Proc. of the Twentieth IEEE Conference on Computer Communications (INFOCOM)*, Anchorage, AK, Apr. 2001, pp. 319–328.
[21] F. Bai, G. Bhaskara, and A. Helmy, "Building the blocks of protocol design and analysis: challenges and lessons learned from case studies on mobile ad hoc routing and micro-mobility protocols," *SIGCOMM Computer Communication Review*, vol. 34, no. 3, pp. 57–70, Jul. 2004.
[22] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, Jan. 1958.