Hierarchical Max-Flow Routing

Chansook Lim
Dept. of Computer Science
Univ. of Southern California, Los Angeles
chansool@usc.edu

Stephan Bohacek
Dept. Electrical & Computer Eng.
Univ. of Delaware, Newark
bohacek@eecis.udel.edu

João P. Hespanha
Dept. Electrical & Computer Eng.
Univ. of California, Santa Barbara
hespanha@ece.ucsb.edu

Katia Obraczka
Computer Eng. Dept.
Univ. of California, Santa Cruz
katia@cse.ucsc.edu

***Abstract*— This paper describes a technique to reduce the computational complexity of max-flow routing, based on a hierarchical decomposition of the network. The computational complexity of this hierarchical max-flow routing is comparable to that of Dijkstra's and Bellman-Ford's algorithms. It is shown that in many of today's networks, this hierarchical approach provides nearly the same performance as flat (i.e., non-hierarchical) routing, with significantly less computation.**

I. Introduction

At any given point in time, traditional Internet routing uses a single path to send packets from a source to a destination. However, there are many advantages in deviating from this principle. In multi-path routing, packets are spread over multiple distinct paths in their way from a source to a destination. The use of multiple paths have been proposed to increase data throughput [1], reduce traffic congestion [2], improve network utilization [3, 4], increase network security [5–8], and improve robustness [8–10]. Alternate path routing (APR) is often used to rapidly find alternate paths between source and destination [11], but sometimes it is also used to switch some load to longer paths when the shortest paths are overloaded [12]. Although end-to-end flows are generally not spread over multiple paths, APR still provides some degree of load balancing and route failure protection.

The computation of multi-path routing tables is a challenging problem. Equal-Cost Multi-Path (ECMP) and OSPF Optimized Multi-Path (OSPF-OMP) are multi-path routing algorithms that provide alternate paths of equal cost [13, 14]. However, by only using distinct paths that have the same cost, these algorithms do not make full use of the redundancy available in the network. Game Theoretic Stochastic Routing (GTSR) is a particular approach to multipath routing that provides a rigorous way to determine which routes should be used and with what frequency [5, 8]. GTSR finds all paths between a source-destination pair and computes *next-hop probabilities*, i.e., the probabilities that a packet takes a particular next-hop. This contrasts with single-path algorithms that simply determine the next-hop.

This material is based upon work supported by the National Science Foundation under Grant No. ANI-0322476.

GTSR computes the next-hop probabilities using a max-flow computation. This computation has a game theoretical interpretation because GTSR's routing policies are saddle solutions to a zero-sum game, in which we regard routing as one player that attempts to defeat worst-case link/node faults. In this game, one associates to each link ℓ a percentage p_ℓ of packets that will not be forwarded in the event of a fault. It was shown in [15] that optimal saddle routing policies for this game can be computed by solving a max-flow optimization [16] over a graph with arc capacities given by $1/p_\ell$. When GTSR is utilized to improve robustness, the designer typically selects low values for p_ℓ in links that are perceived to be more robust. This favors sending more traffic through these links. We thus call $1/p_\ell$ the *level-of-robustness (LOR)* of link ℓ. As mentioned above, GTSR solves max-flow optimizations with link capacities given by the LOR. With some abuse of terminology, we use the expression *LOR of a source-destination pair* to denote the maximum flow between two nodes. Routing schemes that compute next-hop probabilities by solving max-flow problems are called *max-flow routing*.

Perhaps the main practical obstacle to using max-flow routing is the fact that the max-flow optimizations needed to compute the next-hop probabilities require significant computation. The main contribution of this paper is a technique (described in Section II) to reduce the computational complexity of max-flow routing based on a hierarchical decomposition of the network. Hierarchical decompositions to reduce routing computation is commonly used in minimum-cost routing. However, to the best of our knowledge, this is the first paper that proposes a method for hierarchical decomposition for max-flow routing. We show in Section III that the computational complexity of this *hierarchical max-flow routing* is comparable to that of Dijkstra's and Bellman-Ford's algorithms. This is in spite of the fact that we cannot use the very efficient Goldberg-Tarjan max-flow algorithm [17] in all the computations of our hierarchical approach.

Hierarchical approaches ignore topological information in order to reduce computational complexity. The computed flow may therefore be smaller than the maximum one allowed by

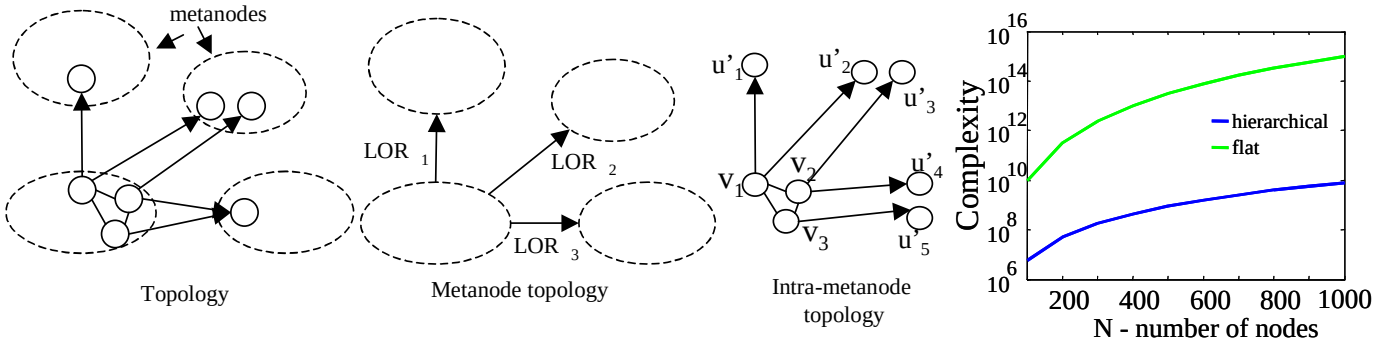


Fig. 1. The inter-metanode routing determines LOR_i between metanodes. The intra-metanode topology constructs fictitious nodes u'_j. The constraints are $LOR_2 f_{v_1 u'_1} = 2 \times LOR_1 f_{v_1 u'_2}$, $LOR_3 f_{v_1 u'_1} = 2 \times LOR_1 f_{v_3 u'_5}$, $f_{v_1 u'_2} = f_{v_2 u'_3}$, and $f_{v_2 u'_4} = f_{v_3 u'_5}$. The right-hand-side figure compares the computational complexity of the hierarchical and flat routing.

the links' LOR and consequently the robustness provided by multi-path routing may also be reduced. Sections IV and V explore this performance/computation trade-off. In Section IV it is shown that if the connectivity within the subdomains (or "metanodes") is higher than the inter-metanode connectivity, then hierarchical max-flow routing performs as well as flat (i.e., non-hierarchical) routing. In Section V we examine several topologies, both simulated and real, and we show that the decrease in performance due to the hierarchical approach is generally small.

In the past, an important deterrent to the use of multipath routing was the fact that TCP is known to exhibit significant performance degradation when subject to persistent packet reordering, which will likely happen under multipath routing. A number of TCP variants have recently been proposed to obviate this problem [18–20]. Among these, TCP-PR [20] has been specifically designed for stochastic multipath routing.

II. Hierarchical Max-Flow Routing

In this section we discuss how the next-hop probabilities are computed for hierarchical max-flow routing. We begin by partitioning the directed graph $G = (V, E)$. The partition used impacts significantly the performance and computational complexity of hierarchical max-flow routing. For high performance, partitioning with high-connectivity within each component is desirable whereas partitioning into optimally-sized components enables low computational complexity. Since these two objectives are often conflicting, optimal partitioning is a challenging problem. While there are many possible techniques to partition a graph, we have found that the method described in the [21] works well. We call each component in the partition a *metanode* and denote the set of metanodes by V_{meta}. In addition, a *border link* $\overrightarrow{vw}$ is a link connecting two nodes $v \in P \in V_{meta}$ and $w \in Q \in V_{meta}$ in distinct metanodes $P \neq Q$. Let F_{PQ} be the set of border links that span from nodes in P to nodes in Q. These links are collapsed into a single link $\overrightarrow{PQ}$, with LOR equal to the sum of all the LOR provided by all links in F_{PQ}. The metanode-level graph is denoted by $G_{meta} := (V_{meta}, E_{meta})$, where

$$E_{meta} = \left\{ \overrightarrow{PQ} : \exists\ v \in P \text{ and } w \in Q \text{ such that } \overrightarrow{vw} \in E \right\}.$$

This graph provides the top level of the hierarchy.

The routing table computation is performed in two steps: *inter-metanode routing* and *intra-metanode routing*. The inter-metanode routing is computed using the graph G_{meta} and is a straightforward application of max-flow routing. Specifically, for each metanode source and destination, a max-flow problem is solved on G_{meta} with the capacity of each meta-link defined as above.

Once the inter-metanode routing is known, the intra-metanode routing can be computed. Depending on whether the source and destination are within the metanode or if data merely transits through the metanode, the intra-metanode routing computation is performed slightly differently. We first examine the intra-metanode routing for a metanode P when the source s is in P but the destination is not in P. In this case, the intra-metanode routing is determined by solving a constrained type of max-flow problem over the graph defined by the metanode P, along with some extra nodes and links that represent egress links. The flow over the egress links are constrained for two purposes: first, in order to satisfy the inter-metanode routing, and second, for load-splitting. Figure 1 depicts how the extra nodes and links are constructed and how the constraints are found. Formally, a constrained max-flow problem is solved over the graph G_P with source s where the graph, destinations, and constraints are defined as follows.

Let w be an edge node of P and define $\mathcal{N}_{\overline{P}}(w) := \{u \notin P : \overrightarrow{wu} \in E\}$, i.e., the neighbors of w that are not in P. For each node in $\mathcal{N}_{\overline{P}}(w)$, we construct the fictitious node u' and link $\overrightarrow{wu'}$. We say that u' is *derived* from the link $\overrightarrow{wu}$. Then $G_P = (V_P, E_P)$ where

$$V_P = \{v \in P\} \cup \{u' : u' \text{ is derived from } \overrightarrow{wu} \text{ with } w \in P,\ u \notin P, \text{ and } \overrightarrow{wu} \in E\},$$

$$E_P := \{vw \in E : v, w \in P\} \cup \left\{ \overrightarrow{wu'} : u' \text{ is from } \overrightarrow{wu} \text{ with } w \in P,\ u \notin P, \text{ and } \overrightarrow{wu} \in E \right\}.$$

The set of derived u' are the destinations.

Two sets of constraints are imposed on the flows across the egress links. The first set of constraints imposes that the ratio of flows from one metanode to another one must coincide with ratio of the LOR that results from the inter-metanode routing. Suppose that the inter-metanode routing yields a LOR between metanodes P and Q_i equal to $LOR\,(PQ_i)$. Then the following constraints are imposed

$$LOR\,(PQ_j) \sum_{\substack{v\in P \\ u' \text{derived from } \overrightarrow{vu} \text{ with } u\in Q_i}} f_{vu'} = LOR\,(PQ_i) \sum_{\substack{v\in P \\ u' \text{derived from } \overrightarrow{vu} \text{ with } u\in Q_j}} f_{vu'}.$$

The second set of constraints imposes that all the links that "go to the same metanode" must have the same flow. That is, if $\overrightarrow{vu_1} \in E$ and $\overrightarrow{wu_2} \in E$, with u_1 and u_2 both in the same metanode, $Q \neq P$, then the constraint

$$f_{\overrightarrow{vu_1}} = f_{\overrightarrow{wu_2}}$$

is imposed, where $f_{\overrightarrow{vu_i}}$ is the flow from v to u_i' and u_i' is derived from $\overrightarrow{vu_i}$ for $i = 1, 2$. In addition to these two sets of constraints, the flow in every link is limited by the link's LOR.

For the case where the destination is in P but the source is not in P, the approach is nearly the same, but the border links are ingress links instead of egress links. If neither the source nor destination are in P, then there is flow over ingress and egress links. If the source and destination are within P, then no flow is carried out of P and the intra-metanode routing is computed by solving the max-flow problem over the subgraph of nodes in P and links that have both ends in P.

III. Computational Complexity

The main motivation behind hierarchical max-flow routing is to reduce the computational complexity. In this section, the complexity of the flat and hierarchical approaches are compared. In this analysis we do not account for the computation needed for graph partitioning, because most changes in network topology do not require a re-partitioning of the graph and therefore the computation needed for partitioning can be amortized over many routing table computations.

Let N denote the total number of nodes in the network G, i.e., $N = |V|$. In addition, let M denote the number of metanodes, i.e., $M = |V_{meta}|$. For simplicity, we assume that the network G is partitioned into M metanodes with the equal number of nodes and denote by $n := N/M$ the number of nodes in each metanode.

For regular max-flow problems (i.e., where there are no constraints of the relative flows across different links), Goldberg and Tarjan's algorithm [17] is known to be the fastest. For flat routing and for inter-metanode routing, this algorithm can be used. Given a sparse network, such as most Internet networks, with the average node degree much smaller than the total number of nodes, Goldberg and Tarjan's algorithm has a computational complexity of $O\left(k^2 \log k\right)$ for k nodes. Thus, in the case of flat routing, we must solve $O\left(N^2\right)$ problems, each of complexity $O\left(N^2 \log N\right)$, resulting in a total complexity of $O\left(N^4 \log N\right)$.

For intra-metanode routing, if the source and the destination are in the same metanode, then the problem reduces to a standard max-flow problem and Goldberg and Tarjan's algorithm can be used. However, when the source and the destination are in distinct metanodes, a constrained max-flow problem must be solved. The constraints make it impossible to use Goldberg and Tarjan's algorithm, hence we employ a fast linear programming (LP) algorithm. The fastest LP algorithms using interior point methods are at least as fast as $O(\frac{n^3 L}{\log n})$, where, in this case, $L = nd + \log(n^2 d)$ and d is the average node degree [22]. Note that $O(\frac{n^3 L}{\log n}) \approx O(\frac{n^4 d}{\log n})$ and $d \ll n$ for typical Internet networks. Thus the inter-metanode routing requires $O\left(M^2\right)$ regular max-flow problems to be solved, each with M nodes, yielding a complexity of $O\left(M^4 \log M\right)$. For the intra-metanode routing, there are $O\left(M^2\right)$ cases where neither the source and destination are in the metanode, and $O\left(nM\right)$ cases where either the source or destination are within the metanode, and $O\left(n^2\right)$ cases where both the source and destination are within the same metanode. In the first two cases, linear programming must be used, hence the complexity for those problems is $O\left(M(nM + M^2)\right) O(\frac{n^4}{\log n})$. In the case where both source and destination are within the metanode, the total complexity is $O\left(Mn^2\left(n^2 \log n\right)\right)$. Thus, the total complexity for the hierarchical approach is $O\left(M(nM + M^2)\frac{n^4}{\log n} + Mn^4 \log n + M^4 \log M\right)$. This bound can be minimized by selecting an optimal partition size $n := N/M$.

Figure 1 shows a comparison of the computational complexity of flat routing compared to hierarchical where the hierarchical routing uses the optimal value of n. Curve fitting shows that the resulting complexity of the hierarchical approach is $O\left(N^{3.2}\right)$. Recall that the time to calculate the shortest paths for all source/destination pairs is $O(N^2 \log N)$ for sparse networks when Dijkstra's algorithm is used. In the case of Bellman-Ford, the complexity is $O\left(N^3\right)$ [16]. Hence, the hierarchical approach yields substantial savings over the flat approach and results in a complexity that is comparable to today's algorithms.

IV. Analysis of Flat versus Hierarchical Max-Flow Routing

Flat routing uses complete topology information whereas, in order to reduce the computation, hierarchical routing uses only limited information. This has the potential to result in sub-optimal routing. However, there are some cases for which hierarchical routing is as good as flat routing. To gain intuition into this issue, we note the LOR of a route is equal to the LOR of the bottleneck. Thus, if the inter-metanode routing provides a LOR equal to μ and each metanode can support a LOR that meets or exceeds μ, then the LOR provided by

the hierarchical routing would be μ. Therefore, if the nodes within the metanodes are better connected than G_{meta}, then the hierarchical approach performs as well as the flat approach. This idea can be formalized through the following series of propositions.

Proposition 1: Hierarchical routing provides a LOR that is no higher than the LOR provided by flat routing.

Proof: Flat routing is an unconstrained maximization, hence the set of next-hop probabilities that are considered in the maximization includes the set of next-hop probabilities that are considered by the hierarchical routing. Hence, the LOR provided by hierarchical routing cannot be higher than that provided by the flat routing. ■

Proposition 2: The LOR provided by the inter-metanode routing (i.e., routing between metanodes and neglecting the topology and LOR provided within the metanodes) is no less than the LOR provided by flat routing.

Proof: The LOR increases with the LOR provided by each link. Since the inter-metanode routing assumes that the links within the metanode provide infinite LOR, the LOR provided by the inter-metanode routing is no less than the LOR provided by the flat routing, which uses the correct topology and LOR within the metanodes. ■

Proposition 3: If, for a particular source-destination pair $\{S, D\}$ that are in distinct metanodes, the inter-metanode routing demands a LOR S_i from each metanode M_i, and, for the source and destination, each metanode is able to provide a LOR greater or equal to S_i, then the hierarchical routing provides the same LOR as flat routing.

Proof: As shown in [8], finding the routing that maximizes the LOR is equivalent to solving a max-flow problem over the graph with each router a vertex and each link an arc with capacity equal to the LOR provided by the link. The resulting LOR is the maximum flow. The LOR provided by the inter-metanode routing is equivalent to the max-flow over G_{meta}, with capacities of each link equal to the LOR of the links between metanodes. Suppose that for a particular source-destination pair, the max-flow problem over G_{meta} results in a flow S_i transiting metanode M_i. If the flow S_i is less than amount of flow that can be accommodated by metanode M_i, then the hierarchical max-flow problem yields a feasible solution. Here the solution of the hierarchical max-flow problem is the flow given by the inter-metanode routing and the resulting flows that cross the links within the metanodes. If S_i surpasses the max-flow that can transit the metanode, then hierarchical flow problem results in a flow that is not feasible, that is, the flows computed either do not satisfy conservation of flow (i.e., more flow enters a node than exits) or the flow across a link exceeds the capacity of the link. On the other hand, if the transit flow can be accommodated by all metanodes, then the hierarchical flow problem results in a maximum flow that is feasible, hence the LOR provided by the hierarchical routing matches the max-flow over G_{meta}. By Proposition 2, the LOR possible over G_{meta} is not surpassed by the LOR provided by flat routing, and by Proposition 1 the LOR provided by the flat routing is not surpassed by the LOR provided by the hierarchical routing. Hence, the LOR provided by the flat routing is equal to the LOR provided by the hierarchical routing. ■

This proposition implies that it is possible to check if the hierarchical routing has a negative impact on the LOR by simply determining if the metanode's LOR meets or exceeds the LOR required by the inter-metanode routing.

Definition 1: A worst-case connection is the source-destination pair that has the least LOR.

Corollary 1: Suppose that G_{meta} is k-connected and each metanode is j-connected with $j \geq k$. The worst-case connections of the hierarchical approach that have source and destination in distinct metanodes are provided the same LOR as they are by flat routing.

Proof: Since G_{meta} is k-connected, the LOR required by each metanode must be greater than k, which can be accommodated by the metanodes. Hence Proposition 3 applies. ■

In summary, if the metanodes are better connected internally than the graph connecting the metanodes and the source and destination are in distinct metanodes, then hierarchical routing will yield the same LOR as the flat routing. Considering a graph such as the graph of routers, one expects a natural partition of the graph into subnetworks. For example, the entire Internet can be partitioned into ASs. If the subnetworks are better connected than the network connecting the subnetworks, then hierarchical routing provides a LOR that is the same as the flat routing. As will be seen in the next section, such good connectivity of subnetworks appears to be common in the Internet.

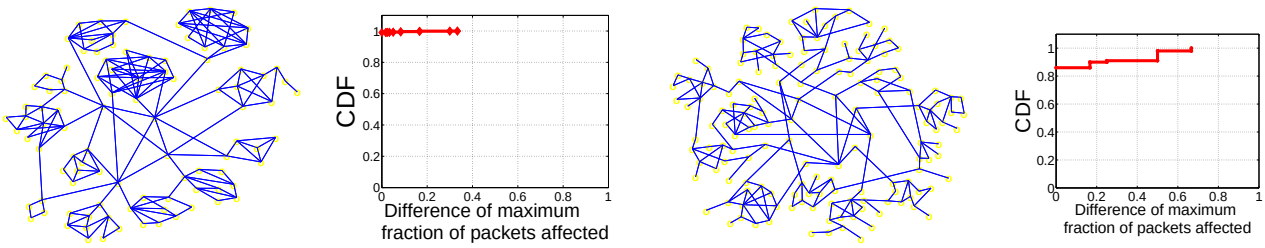
V. Numerical Comparisons between Flat and Hierarchical Max-Flow Routing

A. Methodology

To compare flat vs. hierarchical max-flow routing, we investigated the robustness of the two routing policies. To this effect, we measured the fraction of packets that would be affected by a worst-case single-link failure. For simplicity, we assumed that all links have the same LOR. Flat max-flow routing leads to the least possible fraction of packets, which is exactly the reciprocal of the min-cut for a given source-destination pair.

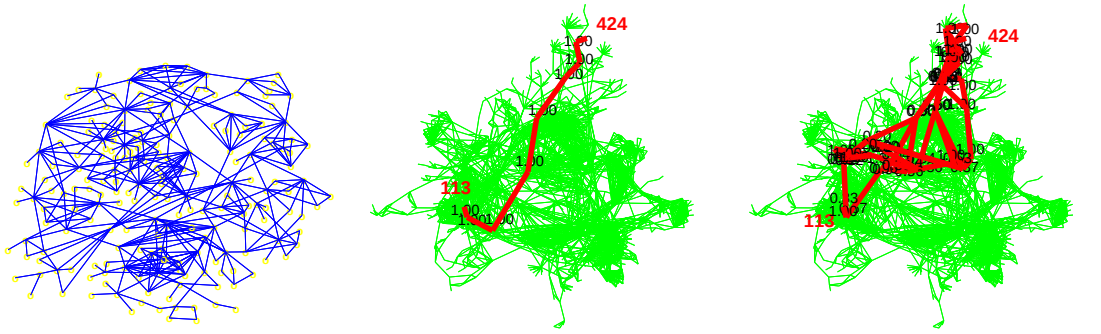
We considered two types of networks: generated by topologies generators and real topologies observed in the Internet. For the generated topologies, we used the popular Transit-Stub topology generator of GT-ITM, which generates graphs based on a three-level hierarchy consisting of transit domains, stub domains, and LANs attached stub nodes. The number of nodes or domains and the connection probability between two nodes in a domain are parameterized. For redundant paths, transit-to-stub edges and stub-to-stub edges are added.

For a comparison on more realistic topologies, we used two ISP topologies recovered from the Rocketfuel datasets [23]:



(a) (b) (c) (d)

Fig. 2. Transit-Stub topology. In (a), the nodes within each metanode are better connected than in (c). The plots (b) and (d) show the difference between the fraction of packets that would be affected by a single link failure for the two types of routing (horizontal axis) vs. the percentage of source/destination pairs (vertical axis). For both topologies, the figures show the difference is small for a very large percentage of pairs. However, hierarchical routing performs better in the topology (a) because of the better intra-metanode connectivity.



(a) Exodus topology (b) Flat max-flow routing (c) Hierarchical max-flow routing

Fig. 3. ISP topologies. (c) shows how multiple routes are found by the hierarchical routing.

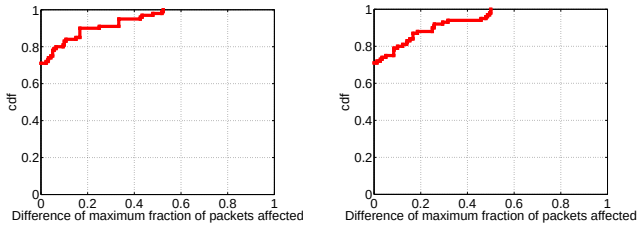
Exodus, Verio[1].

B. Results

We start with the topologies generated by the Transit-Stub topology generator. These topologies have one transit domain which contains 12 transit nodes. One stub domain is attached to each transit node. The probabilities of the connectivity between the transit nodes and between the nodes in a stub domain are 0.6 and 0.42, respectively. This topology is shown in Figure 2(a). The whole transit domain was considered as one metanode and each stub domain was regarded as a metanode. Figure 2(b) shows the difference between the flat and hierarchical routing schemes. We see that the schemes perform nearly identical. When one link fails, over 99% of source/destination pairs have the same degree of robustness under the hierarchical routing as they do under flat routing.

Figure 2(c) shows another Transit-Stub topology where the connectivity between the transit nodes and between the nodes in a stub domain are 0.2 and 0.2, respectively. Figure 2(d) shows that flat routing performs slightly better than the hierarchical routing. The key difference between the two topologies is that the intra-metanode topologies in Figure 2(a) were better connected than those in Figure 2(c). The results obtained are consistent with the analysis in Section IV. Nonetheless, even in the second topology, the difference between hierarchical and flat routing is small with 86% of source/destination pairs seeing no difference in the number of packets affected by a worst-case link failure.

[1]The dataset contains three kinds of data files for each ISP. Among them, we chose the files with the suffix .r0.cch, which include only routers that are believed to be part of the ISP by the name.



(a) Exodus (b) Verio

Fig. 4. Difference between the fraction of packets that would be affected by a single link failure for the two types of routing (horizontal axis) vs. the percentage of source/destination pairs (vertical axis).

Figure 3(a) shows the topology of the ISP Exodus and Figures 3(b) and (c) show the topology of Verio. The topology for Exodus contains 200 nodes and was partitioned into 18 components. The Verio topology used for the simulation contains 960 nodes, many of which have high node degree. For Verio, the partition consisted of 41 components. Figure 4 compares the performance of hierarchical routing and flat routing for these topologies. The overall difference between the two approaches is larger than the one observed for the two Transit-Stub topologies in Figure 2. However, over 70% of the source/destination pairs still see no difference in the number of packets affected by a worst-case link failure. In

addition, for essentially all the source/destination pairs, this difference is less than 50%.

We just saw that the hierarchical approach does not suffer from a significant reduction in performance when compared to flat routing. It turns out that, in some cases it outperforms flat routing. This can be seen in Figure 3(b)-(c), where flat routing uses a single path because the min-cut is one, but hierarchical routing makes use of multiple paths in areas of the network where redundancy exists. This difference between flat and hierarchical routing is significant. In the case of Verio, the flat routing produced an average of 15.4 links at maximum capacity (LOR), while the hierarchical routing resulted in only 4.8 links with the maximum capacity. This shows that hierarchical routing was able to spread packets more evenly across the network, avoiding the creating of hot spots, which generally improves robustness.

VI. Conclusions

Max-flow routing forwards packets to minimize the impact of failures. However, the computational complexity of max-flow routing is quite high, making it impractical for moderate-size networks. Hierarchical max-flow routing provides an approach that is far less computationally intensive. Since hierarchical max-flow routing ignores some topology information, it may not provide the same level of robustness as flat routing. However, we showed that, by enlarge, hierarchical max-flow routing can perform just as well or better than flat max-flow routing. The reason for this is that typical topologies can be partitioned into components, for which the nodes within each component are well connected. This paper only considered two levels of hierarchy. Future work will examine the performance when there are more hierarchical levels.

References

[1] J. Chen, S.Chan, and V.Li, "Multipath routing for video delivery over bandwidth-limited networks," *IEEE JSAC*, 2004.

[2] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *Proc. of INFOCOM*, 2001.

[3] V. Mirrokni, M. Thottan, H. Uzunalioglu, and S. Paul, "Simple polynomial time frameworks for reduced-path decomposition in multi-path routing," in *Proceedings of IEEE INFOCOM*, 2004.

[4] S. J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *ICC*, 2001.

[5] S. Bohacek, J. Hespanha, J. Lee, K. Obraczka, and C. Lim, "Enhancing security via stochastic routing," in *Proc. of the 11th IEEE Int. Conf. On Comput. Communications and Networks*, 2002.

[6] P. Papadimitratos and Z. Haas, "Secure message transmission in mobile ad hoc networks," *Elsevier Ad Hoc Networks*, vol. 1, 2003.

[7] P. Lee, V. Misra, and D. Rubenstein, "Distributed algorithms for secure multipath routing," in *Proc. of INFOCOM*, Mar. 2005.

[8] S. Bohacek, J. Hespanha, J. Lee, C. Lim, and K. Obraczka, "Game theoretic stochastic routing," *Submitted*, 2005.

[9] M. Marina and S. Das, "On-demand multipath distance vector routing in ad hoc networks," in *Proc. of ICNP*, 2001.

[10] C. Tang and P. K. McKinley, "A distributed multipath computation framework for overlay network applications," tech. rep., Michigan State University, 2004.

[11] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," in *ACM SIGMOBILE Mobile Computing and Communications Review*, 2001.

[12] M. Pearlman, Z. Haas, P. Sholander, and S. Tabrizi, "On the impact of alternate path routing for load balancing in mobile ad hoc networks," in *Proc. of the ACM MobiHoc*, 2000.

[13] C. Hopps, "Analysis of an equal-cost multi-path agorithmn," *RFC 2992*, Nov. 2000.

[14] C. Villamizar, "Ospf optimized multipath (ospf-omp)," *draft-ietf-ospf-omp-03*, p. 46, June 1999.

[15] S. Bohacek, J. Hespanha, and K. Obraczka, "Saddle policies for secure routing in communication networks," in *Proc. of the 41st Conf. On Decision and Contr*, pp. 1416–1421, 2002.

[16] T. Cormen, C. Leiserson, and R. Rivest., *Introduction to Algorithms*. Boston, MA: MIT Press, 2001.

[17] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM*, vol. 35, pp. 921–940, 1988.

[18] E. Blanton and M. Allman, "On making TCP more robust to packet reordering," *ACM Computer Communications Review*, vol. 32, 2002.

[19] N. Zhang, B. Karp, S. Floyd, and L. Peterson, "RR-TCP: A reordering-robust TCP with DSACK," in *Proceedings of IEEE INCP*, 2003.

[20] S. Bohacek, J. P. Hespanha, J. Lee, C. Lim, and K. Obraczka, "TCP-PR: TCP for persistent packet reordering," in *ICDCS03*, pp. 222–231, May 2003.

[21] J. P. Hespanha, "An efficient Matlab algorithm for graph partitioning," tech. rep., University of California, Santa Barbara, Oct. 2004. Available at `http://www.ece.ucsb.edu/~hespanha/techreps.html`.

[22] J. E. Beasley, *Advances in Linear and Integer Programming*. Oxford University Press, 1996.

[23] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, 2004.