Avoiding Ties in Shortest Path First Routing

Mikkel Thorup Matthew Roughan
AT&T Labs-Research
Shannon Laboratory
Florham Park, NJ 07932
USA
(mthorup,roughan)@research.att.com

Abstract—**First we discuss problems associated with ties and flow splitting with shortest path first protocols such as OSPF and IS-IS. The problems relate to uncertainty in the splitting when there are multiple shortest path from a source to a destination. Even if routers are configured for even splitting, there can easily be unpredicted biases that can overload links and thereby affect quality of service guarantees for virtual leased lines.**

Second we show how one can set the OSPF/IS-IS weights so as to avoid ties and yet minimize congestion. On real and synthetic networks we demonstrate experimentally that load balancing typically can be done nearly as well without ties as with ties assuming exact even splitting. In fact we get close to the optimum for general routing, including the possibilities with MPLS.

The contribution of the second author is an appendix with measurements from a real network showing how even spltting can be off by 20%. Such bias can replicate if the traffic meets multiple ties from source to destination.

Keywords— **SPF, OSPF, IS-IS, traffic engineering, traffic management, local search, combinatorial optimization.**

I. Introduction

SHORTEST Shortest Path First (SPF) protocols such as Open Shortest Path First (OSPF) [1] or Intermediate System-Intermediate System (IS-IS) [2] are the most commonly used intra-domain internet routing protocols today. The domain is here the backbone of an internet service provider (ISP). Traffic is routed along shortest paths. The weights of the links, and thereby the shortest path routes, can be changed by the network operator. In cases of *ties* where several outgoing links are on shortest paths to the destination, the flow is split roughly evenly.

We assume that we have access to a demand matrix telling how much traffic is routed between different source-destination pairs. Here, by source-destination pair, we mean the points at which a packet enters and exits the ISP backbone. The final destination of a packet, determined by its full IP address, is somewhere outside the network. However, in this paper, we only consider the traffic as it moves within the ISP backbone.

A demand matrix could be estimated from concrete measurements, as described in [3] (see also [4], [5], [6]). The demand matrix could also represent service level agreements (SLAs) on virtual leased lines or virtual private networks where the ISP has committed to support certain amounts of traffic between different source-destination pairs.

Our general traffic engineering objective is to set the link weights so as to (1) avoid ties and (2) route the demands without congestion in terms of link loads exceeding capacities with resulting packet loss and back-off in TCP.

Without the constraint of avoiding ties, this problem has already been studied intensively [7], [8], [9], [10], [11], [12]. A description of the general infrastructure behind this kind of OSPF/IS-IS traffic engineering is given in [13].

A. Splitting considered harmful

Even splitting is a nice tool for balancing the flows in order to avoid congestion in network, like the one in Figure 2, but here we consider ties and splitting harmful. The basic problem is that the splitting may be biased in unpredictable ways, and this makes it difficult to predict the loads on the links. A different kind of problem is that troubleshooting in a network is more difficult if we do not know the path a packet takes from source to destination.

We do consider splitting over parallel links a nec-

essary evil. In the networks considered here, we assume that parallel links have been replaced by a single logical link of appropriate capacity.

Below we first describe why unpredictability is a problem, next we describe how it arises. Finally, we have some remarks on how splitting affects troubleshooting.

A.1 The pain of unpredictable biases

To see that unpredictable biases can be a problem, consider the case where the demand matrix represents SLAs on virtual leased lines or virtual private networks where the ISP has committed to support certain amounts of traffic between different source-destination pairs. If there is a high quality of service (QoS) guarantee, the ISP needs to ascertain that the demands are routed within the link capacities.

If there is a risk of a substantial unpredicted bias, we have to worry about all the ways that traffic can get distributed in the network, and this may require significantly higher network capacity than if we knew how it would actually be routed.

The above being said, there may be cases, such as parallel links, where a diciplined use of ties is justified. Another relavant case will be mentioned in Appendix B. Our warning here is against an uncritical use of ties, just assuming exact even splitting.

A.2 Sources of unpredictable biases

To appreciate the problem, we have to consider how the splitting is actually done. Cisco routers allow (pseudo) random splitting on either a per-packet or a per-destination basis [14]. The per packet choice gives the most even split. However, if packets from the same flow follow different routes, they are likely to arrive out of order, degrading the performance of TCP. This problem is avoided by the per-destination splitting, which is therefore the default.

We note here that per-destination refers to the full IP address that the packet is going to, not the destination within the ISP's backbone. The destination within the backbone is just the place where the packet leaves the backbone. A customer is typically given a whole block of IP addresses leading to the same backbone destination, and if the customer is biased in the use of these IP addresses, this causes a bias in the splitting that is unpredictable from the view-point of the ISP.

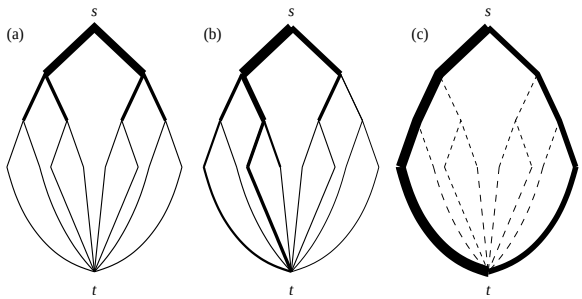

Fig. 1. Biased splitting. A tree is used to spread the traffic from s to t. In (a) we have exact even splitting, and each leaf in the tree receives the same traffic. In (b) all splittings are $2:3$, so the bias at the leaf level is $2^3:3^3=8:27$. In (c) the same hash function is used in all routers, so traffic is only split the first time, and most leaves get no traffic.

Digging a bit deeper, the per-destination splitting is based on a hash function mapping the IP addresses into next-hop links. In Cisco routers, the number of possible next-hops per entry in the routing table is limited to 6. If there are more outgoing links on shortest paths to the destination, only some of these can be represented in the table. Furthermore, the output of the hash function may be based on a small power of two for simple implementation. If the number of outgoing links (the "bins") does not divide evenly into the number of hash outputs, the splitting gets uneven. For example, consider a hash function with $2^4 = 16$ outputs and a routing-table entry with 6 next hops. Ultimately, the split is going to be $2:2:3:3:3:3$.

It may seem that a bias $2:3$ is not that bad, but assume that a shortest paths traverse 3 ties. Then the bias can get as bad as $2^3:3^3=8:27$. In fact, something much worse can happen, for some router vendors have used the same hash function for routers of the same model. This means that when first traffic has been split once, it will stay together in subsequent ties of same outdegree. The different scenarios are illustrated in Figure 1.

We note that with parallel links, we cannot have such replication, for at the end, the split traffic has merged back again. This is one reason for considering them a less malicious special case.

Like the traffic based biases, the above router in-

duced biases are viewed as unpredictable for the ISPF. The OSPF protocol [1] does not specify how ties are to be resolved, so these details of the splitting depends on concrete routers and their configuration. Moreover, the exact mechanism may be inaccessible either as a vendor trade secret or because it is based on some unknown random seeds. Finally, the mapping from hash values to next hops can be history dependent, based on when the next hops where last identified for the destination.

In particular, we have argued that for generic optimization of OSPF weights, as in [7], [8], even splitting can only be taken as a rough approximation, which cannot be trusted if we need a more precise understanding of how traffic spreads in a network.

A.3 Splitting troubles troubleshooting

Having described the splitting mechanisms above, we point out here how they can make troubleshooting more difficult. Per-destination splitting makes troubleshooting more difficult because programs such as traceroute will typically only report on one route, and so problems on alternate routes may go unseen. Per-packet splitting does not help trouble shooting, because then traceroute might report inconsistent routes, as alternate packets from the traceroute take different paths.

B. Result: splitting is not necessary to avoid congestion

In this paper we demonstrate experimentally that for real and synthetic networks, one can typically find a weight settings with no ties that is competitive, within few percent, of the best possible routings using ties and exactly even splitting. In fact, we get within few percent of the optimum for general routing, including protocols such as MPLS [15].

In all our examples our optimized tie free weight setting gains at least 40% over default weight settings such as the one suggested by Cisco [16] of making the weight of a link inversely proportional to its capacity. In our networks these defaults gave rise to hundreds of ties, which we, being nice to the defaults, assumed were split exactly evenly.

Note that it is trivial to construct examples, like the one in Figure 2, in which splitting is useful. Likewise, it is easy to construct examples in which OSPF/IS-IS is worse than MPLS [17], [7]. However,

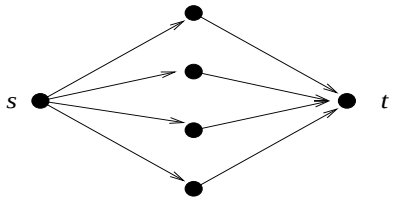

Fig. 2. Constructed example where splitting helps, but what about real networks?

as demonstrated by our experiments, as well as those in [7], [9], [10], and [18], such constructed examples do not tell anything about real networks.

The only previous work we have found on optimizing OSPF/IS-IS weights avoiding ties is [18], but the largest network they consider has 13 links, whereas we deal with up to 360 links. Also, in their general approach to avoiding ties, they use at least as many bits in the weights as there are edges in the network. This is prohibitive since weights in real networks may have only 16 bits. In our experiments, we only need weights less than 1000, hence representable by 10 bits.

B.1 Summing-up

We claim that one can find good weight settings that simultaneously minimize congestion and avoid ties. We then have unique shortest path routing, and we can use this to determine exactly how much capacity is needed on the links to satisfy QoS constraints for virtual leased lines or virtual private networks. With ties, we would have to over-engineer the network with extra capacity to deal with unpredicted biases in the routing.

The uniqueness may also be useful in trouble shooting since the path of a packet is determined from its source and destination.

Finally, without ties, we do not have to worry about the quality of the splitting mechanisms in the routers, e.g., if the number of next-hops per destination is limited. Without ties, we only need one next-hop per destination.

In this paper, we will mostly consider a single fixed network and demand matrix. However, the techniques have been integrated in the traffic management system from [8] for dealing with changes

in the network and demand matrix as well as multiple traffic classes.

C. Contents

First, in §II, we define our exact model and objectives. Next, in §III, we describe our approach to do load balancing without ties. Our experimental set-up is described in §IV and the results are discussed in §V. Finally, we have some concluding remarks in §VI. Also, we have two appendices: in §A we show that optimizing OSPF without ties is NP-hard, and §B we present some concrete measurements on how biased the splitting can be.

II. The basic model

We are going to use essentially the same model as in [7], [8], except that we are going to penalize ties in the weight setting.

A. The general routing problem

Optimizing the use of existing network resources can be seen as a general routing problem defined as follows. We are given a directed graph $G = (N, A, c), A \subseteq N \times N$ with arc capacities $c = (c_a)_{a \in A}$. The nodes and arcs represent routers and the capacitated links between them. The graph is simple in the sense of having no parallel links. As mentioned, we assume that parallel links have been replaced by a single logical link of appropriate capacity. Also, we are given a demand matrix D that, for each pair (s, t) of nodes, tells us how much traffic flow we need to send from s to t. We refer to s and t as the source and the destination of the demand. Many of the entries of D may be zero, and in particular, $D[s, t]$ should be zero if there is no path from s to t in G. A routing solution specifies for each source-destination pair how the demanded traffic should flow in the network. The load ℓ_a on an arc a is then the total traffic flow through the arc, including the contributions from each source-destination pair.

For real instances of the problem, additional complicating constraints such as nodes forbidden for transit traffic or point-to-multi-point demands arise [3]. These kind of constraints can be integrated by modifying the graph including artificial links, but these constraints do not affect the methods and results presented here and are left out for the sake of clarity.

So far, we have been rather vague about our objective of "avoiding overloaded arcs", and we will now define some more exact objectives. The utilization of an arc a is the load divided by the capacity, i.e. ℓ_a/c_a, and a link is overloaded if the utilization exceeds 100%. The max-utilization is the maximum utilization over all links.

Minimizing the max-utilization as in [10] is a natural and intuitive objective for routing. We note that there may be some links for which we are more concerned about high utilization for than others, but we can just view such links as having a reduced capacity. We will consider the max-utilization in this paper, but it suffers from allowing a single bottle-neck, e.g. an ingress link from another domain over which we have no control, to dominate the whole picture. Also, it doesn't penalize using very long detours. To get a measurement considering the whole network, we consider cost functions of the form

$$\Phi = \sum_{a \in A} \phi(\ell_a, c_a)$$

summing a cost $\phi(\ell_a, c_a)$ from each arc a depending on the relation between the load ℓ_a and the capacity c_a. More precisely, we define $\phi(\ell_a, c_a)$ as the continuous function with $\phi(0, c_a) = 0$ and derivative in the load ℓ_a of

$$\phi'(\ell_a, c_a) = \begin{cases} 1 & \text{for} \quad 0 \le x/c_a < 1/3, \\ 3 & \text{for} \quad 1/3 \le x/c_a < 2/3, \\ 10 & \text{for} \quad 2/3 \le x/c_a < 9/10, \\ 70 & \text{for} \quad 9/10 \le x/c_a < 1, \\ 500 & \text{for} \quad 1 \le x/c_a < 11/10, \\ 5000 & \text{for} \quad 11/10 \le x/c_a < \infty. \end{cases} \tag{1}$$

The arc cost function $\phi(\cdot, 1)$ is illustrated in Figure 3. Generally it is cheap to send flow over an arc with a small utilization ℓ_a/c_a. The cost increases progressively as the utilization approaches 100%, and explodes when we go above 110%.

Because of the explosive increase in cost as loads exceed capacities, our objective typically implies that we keep the max-utilization below 1, or at least below 1.1, if at all possible.

The objective function was chosen on the basis of discussions on costs with people close to the AT&T IP backbone. The exact coefficients are not important. We tried many variations and found that this did

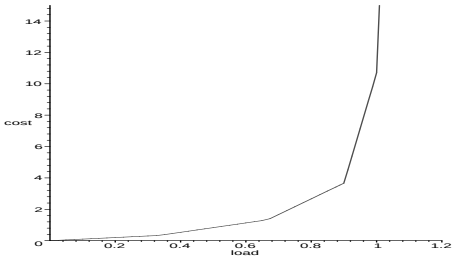

Fig. 3. Arc cost $\phi(\ell_a, 1)$ as a function of load ℓ_a with capacity $c_a = 1$.

not change the quality of our results. Also, it had no substantial impact to use a smoother objective function with smaller segments. More importantly, the routing solutions found were very robust to changes in the objective function. In particular, when optimizing routings for Φ, our solutions tended to also do very well with respect to max-utilization.

The piece-wise linearity of our cost function has the advantage that using a Linear Programming (LP) solver, we can find the optimal solution to the general routing problem with no limitations to how we can distribute the flow between the paths. We can then compare ourselves against this unrealistic ideal to see how competitive we are with *any* other approach, including MPLS.

A problem in the current formulation of Φ is that it does not provide a universal measure of congestion. With the max-utilization, it is a problem for any network if it exceeds 1, and we would like a similar universal cut-off for our summed link-costs. To achieve this, we use a normalized cost function[1]

$$\Phi^* = \Phi / \Psi$$

where Ψ is the cost we would have had if all demands were sent along hop-count shortest paths and all links on these paths had utilization 1. Then we pay $\phi(1,1)$ per unit load on a link, so if $\Delta(s,t)$ is the hop-count distance between s and t, $\Psi = \sum_{(s,t)\in V^2}(D[s,t] \cdot \Delta(s,t) \cdot \phi(1,1))$. Note that for a given network and demand matrix, the division by Ψ doesn't affect which routings are considered good. However, $\Phi^* \geq 1$ implies that we are performing as badly as if all flows where along hop-count shortest paths with utilization 1 utilization. The same cost can, of course, also stem from some loads going above capacity and others going below, or by flows following longer detours via less utilized arcs. Nevertheless, it is natural to say that a routing *congests* a network if $\Phi^* > 1$.

[1] The normalization from [7] was defined differently so that it was $\phi(1,1) = 10\frac{2}{3}$ times bigger.

A.1 Gold customers

As mentioned, our cost function Φ from (1) is also good for keeping the max-utilization below 100%. Now, suppose we are dealing with gold customers that we promise to route over links with utilization below 60%. We can incorporate this in (1), simply replacing c_a with $c'_a = 0.6c_a$. In the concluding remarks, we shall briefly discuss how we can integrate guarantees to the gold customers with good best-effort service for regular customers.

B. OSPF/IS-IS routing

As mentioned, this paper focuses on shortest path first routing such as OSPF [1] and IS-IS [2], which are the most commonly used intra-domain internet routing protocols today. The network operator assigns a weight to each link, and shortest paths from each router to each destination are computed using these weights as lengths of the links. In each router, the next link on all shortest paths to all possible destinations is stored in a table, and a flow arriving at the router is sent to its destination by splitting the flow between the links that are on the shortest paths to the destination. In practice, at least with OSPF, the splitting is roughly even.

In this paper, our goal is to avoid ties, so that we do not have any issues with the splitting. Here we assume that the input has no parallel links. If the real input has parallel links, these should be encoded as single links in a preprocessing step. The capacity is calculated depending on whether they are to be used in parallel with the same weight, or with one as back-up for the other, the back-up link getting a bigger weight.

We will compare our performance against standard defaults, such as that of setting link weights inversely proportional to capacity. These create lots of ties, and we will evaluate them with respect to two types of splitting:

Even splitting where the flow is divided evenly between all shortest path links to destination.

Penalized splitting where, in case of a tie, we first

increase the flow by a factor 1.2 and then split it evenly, e.g. sending 60% out each link in connection with a 2-way tie.

The penalized splitting is modeling that we allow splits to be somewhat uneven, but to give a safe upper-bounds on the loads, we prepare for the higher loads in all directions. The factor 1.2 is somewhat arbitrary, but seemed reasonable given that we measured bias up to 17% in an operational network (cf. Appendix B). Also, note that we have to be on the conservative side if we want to provide QoS performance guarantees. Yet the penalized splitting is too negative in that we don't get credit for the lightly loaded links in the precense of bias. We view penalized splitting as our pessimistic estimator of what happens with bias, and equal splitting as our optimistic estimator. The truth is then sandwiched somewhere between these two estimators.

III. Weight setting with local search

As shown in Appendix A, it is NP-hard even to get an approximately optimal weight setting avoiding ties, so we resorted to a local search heuristics for the weight setting.

¿From [7], we already have a highly tuned local search that for a given network and demand matrix optimizes the weights so as to minimize Φ from (1), *assuming even splitting*. As a local search heuristic [19], it starts with an arbitrary weight setting. Then repeatedly it tries to improve the current weight setting by changing one or a few weights.

To get a local search avoiding ties, we essentially just changed the internal flow computation to use penalized splitting instead of even splitting. Generally, one can always exclude ties by increasing the penalty factor, say from 1.2 to 10, and by increasing the range of weights. A probabilistic argument shows that if we have a network with n nodes and m links with random weights in the range $\{1, .., n \times m\}$, the network is unlikely to have nay ties. However, even for our largest networks with 100 nodes and 360 links, our tie penalizing local search found integer weights in the range $\{1, ..., 1000\}$ giving no ties and good load balancing.

IV. Experiments

Our basic experimental networks and demand matrices are the same as in [7]. We have a proposed AT&T IP backbone with 90 nodes, 274 arcs with projected demands. Also, we have synthetic 2-level networks produced using the generator GT-ITM [20], based on a model of Calvert, Bhattacharjee, Daor, and Zegura [21], [22]. This model places nodes in a unit square, thus getting a distance $\delta(x, y)$ between each pair of nodes. These distances lead to random distribution of 2-level graphs, with arcs divided in two classes: *local access* arcs and *long distance* arcs. Arc capacities were set equal to 200 for local access arcs and to 1000 for long distance arcs. Inspired by classical entropy models for urban traffic [23], demands were modeled as follows. For each node x, we pick two random numbers $o_x, d_y \in [0, 1]$. Further, for each pair (x, y) of nodes we pick a random number $c_{(x,y)} \in [0, 1]$. Now, if the Euclidean distance (L_2) between x and y is $\delta(x, y)$, the demand between x and y is

$$\alpha o_x d_y c_{(x,y)} e^{-\delta(x,y)/2\Delta} \tag{2}$$

Here α is a parameter and Δ is the largest Euclidean distance between any pair of nodes. Above, the o_x and d_x model that different nodes can be more or less active senders and receivers, thus modeling hot spots on the net. Because we are multiplying three random variables, we have a quite large variation in the demands. The factor $e^{-\delta(x,y)/2\Delta}$ implies that we have relatively more demand between close pairs of nodes, yet the distance on its own never has an impact bigger than a factor $\sqrt{e} = 1.648...$. In our experiments, we also tried not using the distance factor $e^{-\delta(x,y)/2\Delta}$, and the results were essentially unchanged. In fact, we should mention that the model without the distance factor also has been used in [24] for Internet traffic. In [24] they also consider voice and transaction data but with a large distance factor.

We then compared a variety of different routing schemes:

NoTiesOSPF Our new optimized weight setting avoiding ties.

WithTiesOSPF The optimized weight setting from [7] using ties evaluated with even splitting.

InvCapOSPF(*) Weights sat inversely proportional to capacity evaluated with even splitting (penalized splitting).

UnitOSPF(*) All weights sat to one evaluated with even splitting (penalized splitting).

L2OSPF(*) Weights sat Euclidean distance (L_2) be-

tween end-points, evaluated with even splitting (penalized splitting).

OPT The optimal general routing, covering the possibility with MPLS.

Above, WithTiesOSPF, InvCapOSPF, UnitOSPF, L2OSPF, and OPT are all taken from [7] while NoTiesOSPF, InvCapOSPF*, UnitOSPF*, and L2OSPF* are new.

The results or our experiments are presented in Figures 4–9 with different scaling of the demand matrices. Figure 9 is special because we scaled each capacity with 0.6 as suggested in Section II-A.1.

On the left sides, we have the normalized cost function, and on the right sides we have the corresponding max-utilization. For all the OSPF/IS-IS schemes, the normalized cost and max-utilization are calculated for the same weight setting and routing. However, for OPT, the optimal normalized cost and the optimal max-utilization are computed independently with different routing. We do not expect any general routing to be able to get the optimal normalized cost and max-utilization simultaneously.

V. Discussion

Below, we ignore Figure 9 till Section V-A. First, as in [7], we note that all curves start off pretty flat, and then, quite suddenly, start increasing rapidly. This behavior follows our cost function that explodes when the load of a link reaches its capacity (cf. (1) and Figure 3). The most interesting comparison between the different schemes is the amount of demand they can cope with before the network gets congested in the sense that its normalized cost exceeds 1.

First we note for each of the default weight settings InvCapOSPF, UnitOSPF, and L2OSPF, the gap from the pessimistic penalized splitting up to the optimistic even splitting is never more than about 15%, the maximum being for InvCapOSPF in Figure 6. Thus the effects of biased splitting is more limited than one could have feared.

Second we see that NoTiesOSPF improves with 40%-75% over the default weight settings InvCapOSPF, UnitOSPF, and L2OSPF optimistically evaluated with even splitting. In particular, for AT&T's proposed backbone in Figure 4 the improvement is 70%.

Finally, if we ignore Figure 6, we see that the maximal gap from NoTiesOSPF to OPT is at most 5%, with WithTiesOSPF lying somewhere in between. Figure 6 is an out-lier with a gap of 30%, but we note that this is one of the smaller networks with only 50 nodes. For contrast, in the larger networks with 100 nodes, the gap is at most 3%. One possible explanation is that the law of large numbers tends to smooth out the traffic for larger networks.

The max-utilization for WithTiesOSPF and NoTiesOSPF has a step-wise pattern which pretty much follows the steps in our cost function Φ from (1). In particular, we see that they both tend to do very well when the max-utilization gets to around 1, whereas they do not worry so much about the max-utilization when it is lower.

A. Gold customers

In Figure 9 we have used exactly the same network and demand matrix as in Figure 4. However, in our object function (1), we used the gold customer trick from Section II-A.1 of multiplying each capacity with 0.6. Our target was to get the max-utilization below 60% if possible. On the right hand side, we see that our target was achieved in the sense that the optimal solution could only support 2% more demands with a max-utilization of 62%.

Mathematically, getting from Figure 4 to Figure 9, we first scaled the capacities by 0.6 in (1). For the same weight setting and routing, we correspondingly scale the demands by 0.6 and then we get the same normalized cost function. When it comes to the max-utilization, we need to scale both demands and max-utilization by 0.6. Thus, whereas we in Figure 4 did a good job in keeping the max-utilization around 100%, in Figure 9, we do a good job in keeping the max-utilization round 60%.

VI. Concluding remarks

We have pointed out that ties is a problem in shortest path routing such as OSPF and IS-IS. First, the exact details of the splitting depends on the individual router, and may not be released by the vendor. Hence we cannot predict packet routes from source to destination. Second, the splitting is typically not exactly even. The splitting may be based on some random-like hash function, but sometimes, the same hash function is used in all routers, and as a result, uneven splitting may be repeated all over the net-

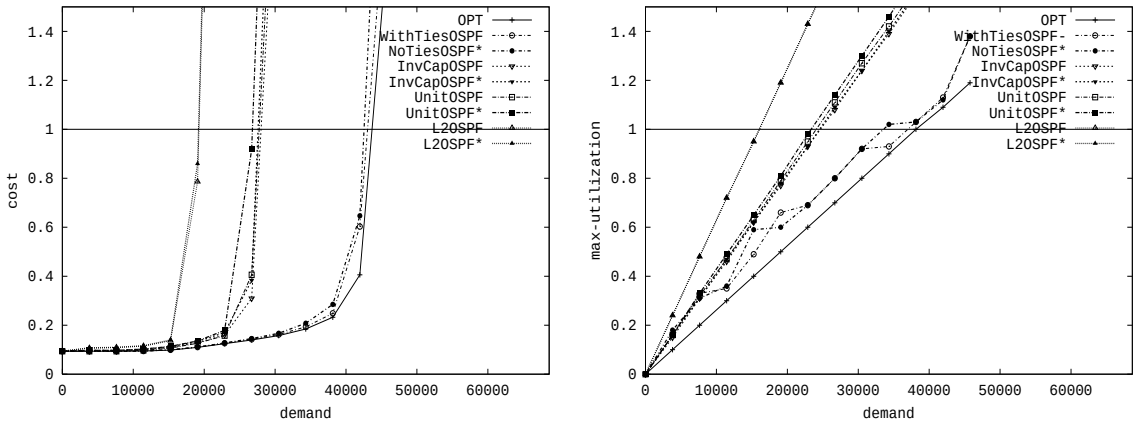

Fig. 4. AT&T's proposed backbone with 90 nodes and 274 arcs and scaled projected demands.

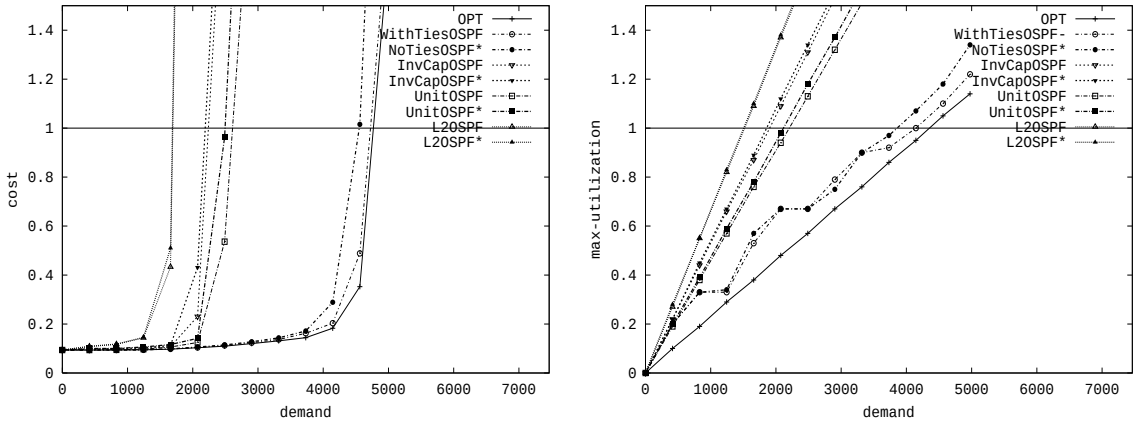

Fig. 5. 2-level graph with 50 nodes and 148 arcs.

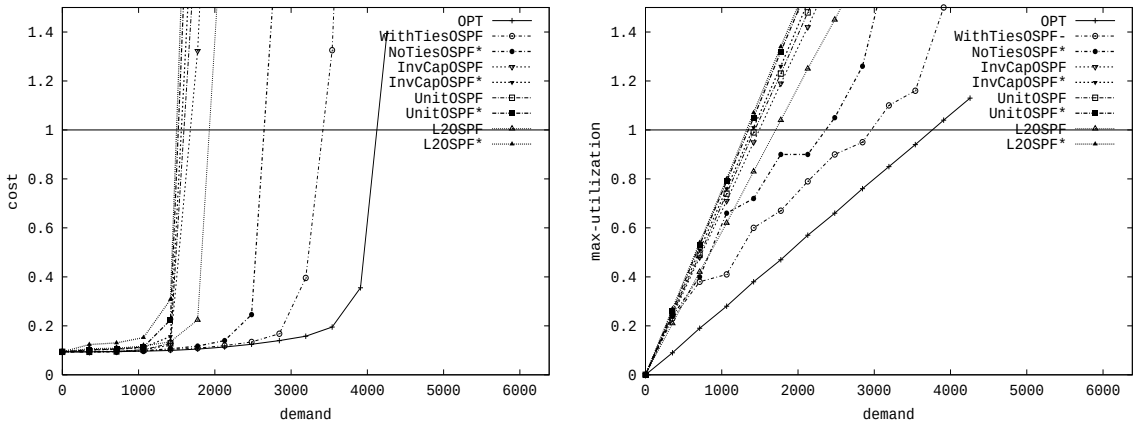

Fig. 6. 2-level graph with 50 nodes and 212 arcs.

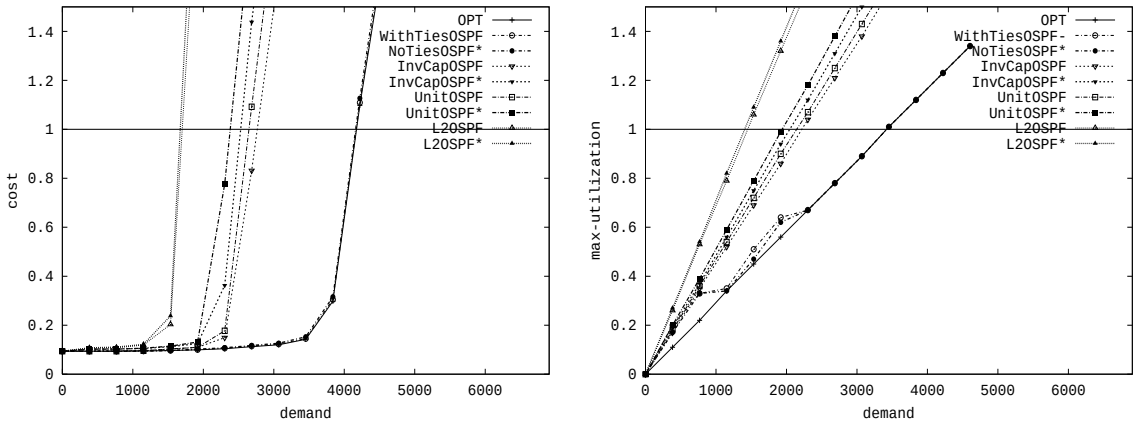

Fig. 7. 2-level graph with 100 nodes and 280 arcs.

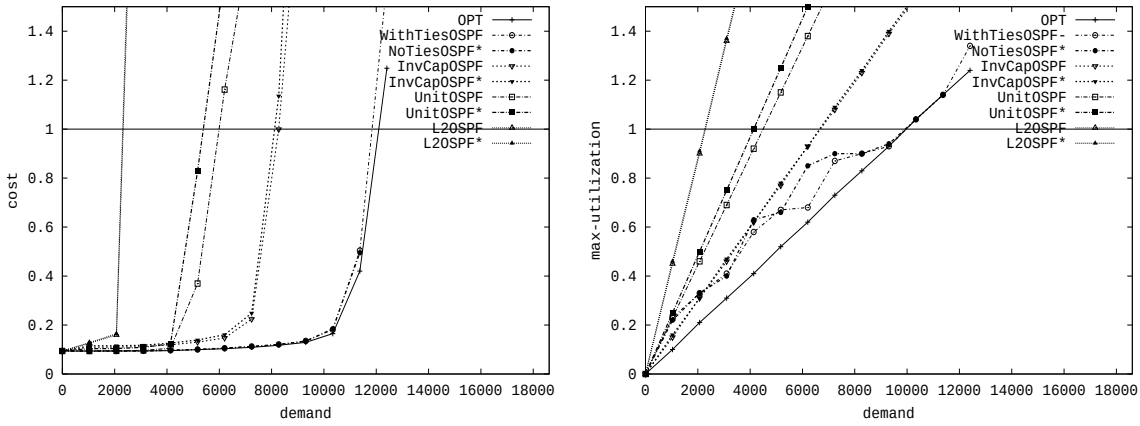

Fig. 8. 2-level graph with 100 nodes and 360 arcs.

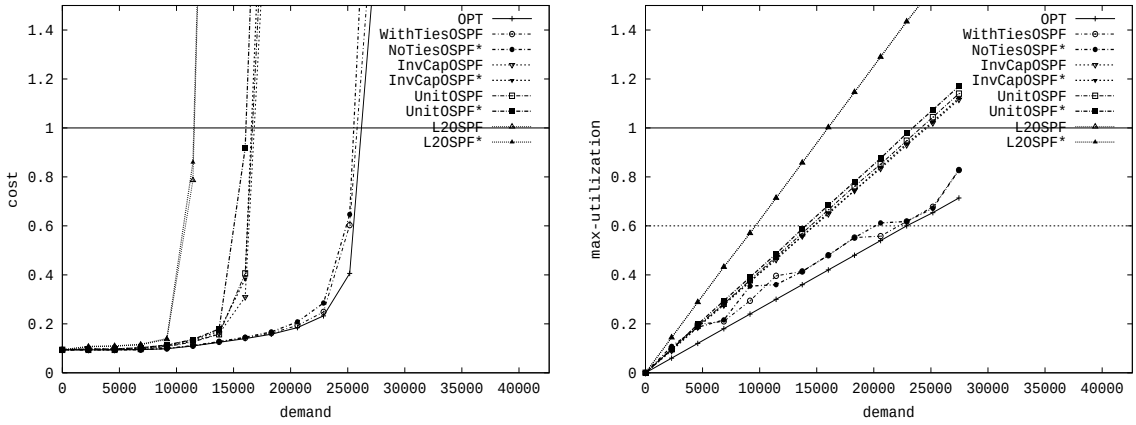

Fig. 9. Gold version of AT&T's proposed backbone, aiming at 60% utilization.

work. The possibility of uneven splitting makes it difficult to predict link loads even with a given demand matrix. This gives problems in connection with virtual leased lines, where we want to guarantee customers that certain demands can be satisfied within the capacity of our network.

Our experimental work on real and synthetic networks indicate that the problem of predicting the link loads because of bad splitting around ties may not be as serious for default weight settings such as inverse capacity. However, for exact predictions of flows for given demands, we need to avoid ties, and then we are very happy to report that optimized weight without ties can do nearly as well as the optimized weights with ties and even splitting from [7], improving with 40%-75% over the default weight settings, and typically getting within few percent of the general routing optimum, including MPLS.

In all of the above work we have assumed a single fixed network and demand matrix. However, the techniques has been integrated in the traffic management system from [8] for dealing with changes in the network and demand matrix as well as multiple traffic classes. That is, the system from [8] actually supports penalized splitting as discussed in this paper.

ACKNOWLEDGMENT

The author would like to thank Jennifer Rexford and Aman Shaikh for many helpful comments.

REFERENCES

[1] J. T. Moy, "OSPF version 2," Network Working Group, Request for Comments: 2328, `http://search.ietf.org/rfc/rfc2328.txt`, April 1998.

[2] R. Callon, "Use of OSI IS-IS for routing in TCP/IP and dual environments," Network Working Group, Request for Comments: 1195, `http://search.ietf.org/rfc/rfc1195.txt`, December 1990.

[3] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 265–279, 2001.

[4] J. Cao, D. David, S.V. Weil, and B. Yu, "Time-varying network tomography: Router link data," *Journal of the American Statistical Association*, vol. 95, pp. 1063–1075, 2000.

[5] N.G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 280—292, 2001.

[6] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast, accurate computation of traffic matrices from link loads," in submission.

[7] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. 19th IEEE Conf. on Computer Communications (INFOCOM)*, 2000, pp. 519–528.

[8] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *Journal of Selected Areas in Communications (Special Issue on Recent Advances on Fundamentals of Network Management)*, vol. 20, no. 4, pp. 756–767, 2002.

[9] K.G. Ramakrishnan and M.A. Rodrigues, "Optimal routing in shortest-path data networks," *Lucent Bell Labs Technical Journal*, vol. 6, no. 1, pp. 117–138, 2001.

[10] F.Y.S. Lin and J.L. Wang, "Minimax open shortest path first routing algorithms in networks supporing the SMDS services," in *Proc. IEEE International Conference on Communications (ICC)*, 1993, vol. 2, pp. 666–670.

[11] M. Ericsson, M.G.C Resende, and P.M. Pardalos, "A genetic algorithm for the weight setting problem in OSPF routing," *J. Combinatorial Optimization*, vol. 6, no. 3, pp. 299–333, 2002.

[12] L. S. Buriol, M. G. C. Resende, Celso C. Ribeiro, and M. Thorup, "A memetic algorithms for OSPF routing," in *Proc. 6th INFORMS Telecom*, 2002, pp. 187–188.

[13] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, Oct. 2002.

[14] Cisco, "How does load-balancing work?," 2002, Document ID 5212. Available at `http://www.cisco.com/warp/public/105/46.html`.

[15] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Network Working Group, Request for Comments, `http://search.ietf.org/rfc/rfc3031.txt`, 2001.

[16] Cisco, "Configuring OSPF," 2001, Documentation at `http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/ip_c/ipcprt2/1cdospf.htm`.

[17] D. O. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Communications Magazine*, vol. 37, no. 12, pp. 42–47, Dec. 1999.

[18] A. Bley, M. Grötchel, and R. Wessäly, "Design of broadband virtual private networks: Model and heuristics for the B-WiN," in *Proc. DIMACS Workshop on Robust Communication Networks and Survivability, AMS-DIMACS Series 53*, 1998, pp. 1–16.

[19] E. H. L. Aarts and J. K. Lenstra, Eds., *Local Search in Combinatorial Optimization*, Discrete Mathematics and Optimization. Wiley-Interscience, Chichester, England, 1997.

[20] E. W. Zegura, "GT-ITM: Georgia tech internetwork topology models (software)," `http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz`, 1996.

[21] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. 15th IEEE Conf. on*

Computer Communications (INFOCOM), 1996, pp. 594–602.

[22] K. Calvert, M. Doar, and E. W. Zegura, "Modeling Internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, 1997.

[23] P. L. Toint, "Transportation modelling and operations research: A fruitful connection," in *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*, M. Labbé, G. Laporte, K. Tanczos, and P. L. Toint, Eds. 1998, vol. 166 of *NATO ASI series: Ser. F, Computer and systems sciences*, pp. 1–27, Springer.

[24] A. Dwivedi and R.E. Wagner, "Traffic model for USA long-distance optimal network," in *Proc. Optical Fiber Communication Conference (OFC)*, 2000, pp. 156–158, TuK1.

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

[26] J. Håstad, "Some optimal inapproximability results," *Journal of the ACM*, vol. 48, no. 4, pp. 798–859, 2001.

APPENDIX

I. HARDNESS WITHOUT TIES

In this section, we show that it is NP-hard to find even an approximately good weight setting in general. The proof follows the standard pattern from [25]. First we have

Theorem 1: There is an infinite family of networks so that the optimal tie-free weight setting routes all demands utilization at most 1, but where it is NP-hard to find a tie-free weight setting avoiding that some link has a utilization of 2.

Proof: Our starting point is an instance of 3-SAT, that is, each clause has exactly 3 literals.

For each variable x we have a source s_x and a sink t_x with a demand of one in between. Further, we have two disjoint paths T_x and F_x from s_x to t_x, $T_x \cap F_x = \{s_x, t_x\}$. In T_x, we have an edge t_x^i for each positive occurrence of x in the clauses. Similarly, in F_x, we have an edge f_x^i for each negative occurrence of x in the clauses. All links have capacity 1, and we have a demand of 1 from s_x to t_x. If x is true, the demand will be routed along F_x, leaving T_x free, and if x is false, the demand will be routed along T_x, leaving F_x free. In terms of weights, the choice can be made by making the link weights on the chosen path $1/2$ whereas those on the other path are 1.

Now, take an arbitrary clause $C = x \vee \bar{y} \vee z$. Suppose this is the ith positive occurrence of x, the jth negative occurrence of y, and kth positive occurrence of z. We then have a new source vertex s_C connected with an edge to the starting points of the three edges t_x^i, f_y^j, and t_z^k, and edges from their finish points to a new destination. The demand from s_C to T_c is 1, and so is the capacity of the new links. The demand from s_c to t_C can be routed if and only if one of the occurrence links t_x^i, f_y^j, and t_z^k are free; otherwise, one occurrence link gets load and utilization 2. We note that we can choose an occurrence link t_x^i, f_y^j, or t_z^k by setting its incident edges to $1/2$ whereas the other edges incident to s_C and t_C have weight 1. ■

Theorem 2: There is an infinite family of networks so that approximating the optimal cost within a factor 11.

Proof: We use exactly the same construction as before. We use a deep result of Håstad [26] that it even if each variable has the same number of negative and positive occurrences, it is NP-hard approximate the number of satisfiable clauses within a factor $7/8 + \varepsilon$ for any ε. Let n be the total number of variable occurrences.

First, we consider the loads corresponding to a satisfying assignment. Then each link gets load 0 or 1. We wish to count how many links get loaded. Because the number of positive and negative occurrences are the same, on the average, we get a half loaded link per occurrence of x from the demand from s_x to t_x. Also, from clause C, we get 3 loaded links on one of the three paths from s_C to t_C, which is 1 per occurrence, so in total the number of loaded links is $1.5n$, for a total cost of $1.5\phi(1,1)n$.

Now, any assignment we can find will have at least $1/8 - \varepsilon$ clauses not satisfied, and for each of these, one link with load 2. This gives at least $1/3 \cdot (1/8 - \varepsilon)n = (1/24 - \varepsilon')n$ such overloaded links. The total link load is unchanged, so we are saving a corresponding number of links with load 1. Hence the increase in load is $(1/24 - \varepsilon')n(\phi(2,1) - 2\phi(1,1))n$. Thus, we are going to get a cost which is at least

$$\begin{aligned} & 1 + \frac{(1/24 - \varepsilon')(\phi(2,1) - 2\phi(1,1))}{1.5\phi(1.1)} \\ = \; & 1 + \frac{(1/24 - \varepsilon')(4560\frac{2}{3} - 2 \cdot 10\frac{2}{3})}{1.5 \cdot 10\frac{2}{3}} \\ > \; & 11 \end{aligned}$$

times as big as that of a satisfying assignment. ■

II. Biases measured in a real network

This section (the contribution of the second author) provides motivation for the work above by presenting examples the bias of splitting in a large operational backbone. The measurements are based on Simple Network Management Protocol (SNMP) data taken over one month in January 2002. The available SNMP data provides the five minute average link utilizations over for each link in the network. Thus, to measure splitting, we needed to find a pair of links in the network topology where ideal load balancing would split the traffic evenly. Then, comparing the loads on this pair of links gives us the bias of the actual load balancing.

It is not generally hard to find such link pairs in most networks. A common example is the *double star* in Figure 10. All traffic from the nodes on the left to the destination on the right should be split evenly over the links to and from the two center nodes – for example, the traffic from the top left node would be evenly split across the two bold links. The reason such topologies commonly occur is to provide redundancy, so that if a link or node fails the traffic can use the alternate route. We note that this, bias or not, is an example of a desirable use of ties and splitting. Our general point in this paper is just that ties shouldn't be used uncritically, and that they are not in general needed to avoid congestion.

We examined a number pairs of links configured in double stars. Figure 11 shows three examples from the same double star configured precisely as in Figure 10. Considering examples from a single double star makes for a more interesting comparison in some respects. Each point in a plot shows one measurement, with the x- and y-axis showing the link utilizations on the pair of measured links, and the dashed line showing equality. The graphs are divided vertically into three different cases (corresponding to three different leftmost nodes in Figure 10). The titles, in and out, refer to the direction of the traffic into the double star: "in" corresponds to traffic going from left-to-right, and "out" to traffic in the reverse direction. One can see in these examples a range of behaviors – from very close to balanced splitting to noisy variation away from balanced splitting, to systematic bias away from an even split.

Figure 12 shows summary statistics of the observed biases in 9 double star configured nodes. We

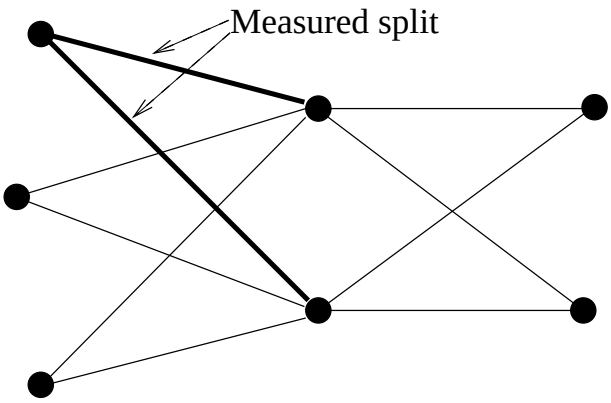

Fig. 10. The double star topology. By default, all links have equal weight. Traffic is evenly split between the pairs of links connecting the leftmost nodes to the middle nodes, for example across the pair of bold links.

restrict our attention to those setups carrying appreciable traffic so as to prevent undue bias from traffic from a single source. The plot shows the Mean Relative Deviation (MRD) of the traffic on the two links, defined by

$$MRD = E\left[\frac{|x-y|}{|x+y|/2}\right],$$

where x and y are the traffic on the two links across which traffic is split. The MRD measures the relative size of the deviation from $x = y$, and is computed using a sample mean over the month of data used here.

The figure shows that the majority of biases fall below 7% but that in at least one case range as far as 17%. Also of interest is the "out" traffic, the traffic from right-to-left in the double start shows more bias than the "in" traffic (which also appears to be the case in Figure 11). While this range of mean bias appears reasonable, we found that the maximum bias over the month of data could easily be of the order of 100% (though this figure must be treated more carefully, because a rerouting event, or topology change during the experimental interval might result in such a bias) and the maximum bias was typically greater than 20%.

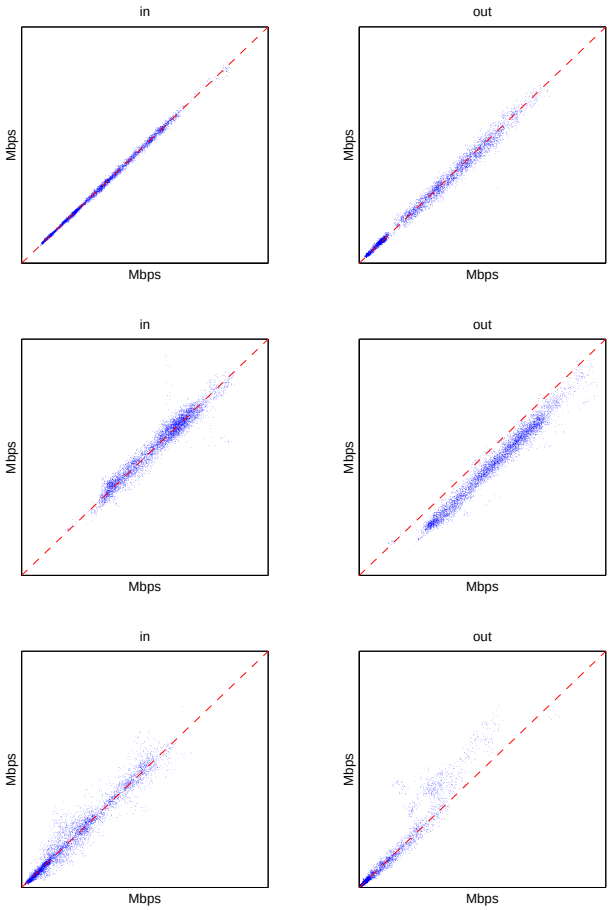

Fig. 11. Observed traffic in a single double star.

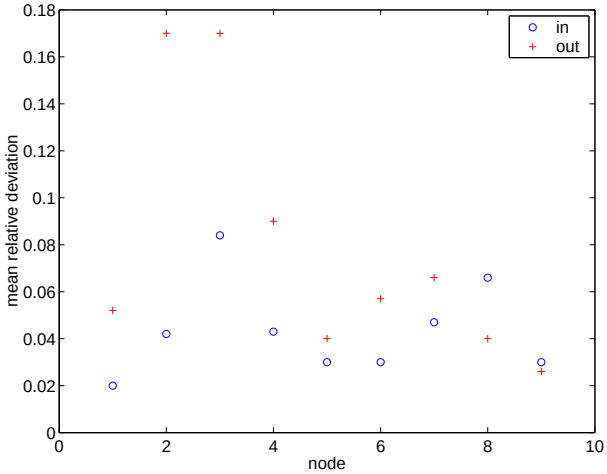

Fig. 12. Observed biases in double star configurations.