PAPER

Drouting Architecture: Improvement of Failure Avoidance Capability Using Multipath Routing

Yasuhiro OHARA[†*a)], **Hiroyuki KUSUMOTO**[††b)], **Osamu NAKAMURA**[††c)], ***Nonmembers***, ***and*** **Jun MURAI**[††d)], ***Member***

SUMMARY Failure avoidance capability is a desired feature for telecommunication networks, such as the Internet. However, not all failures can be promptly bypassed on the Internet because routing systems that are responsible for detecting and avoiding failures cannot detect all failures. Consequently, failures can interrupt internet communications for a long time, such as a few hours. This paper proposes a novel routing architecture called *Drouting* that enables flexible failure avoidance. In Drouting, routers calculate multipaths from a source to a destination by constructing Directed Acyclic Graphs (DAGs) that include all links in the intra-domain network graph. IP packets carry packet tags that are set by the end host. The packet tags are used to select a network path from the multipath routes. In this paper, the failure avoidance property of Drouting architecture is evaluated through comparison with another proposal, *Deflection*, using simulations. Simulations were performed on inferred and synthetic topologies. Drouting exhibits similar performance with Deflection in terms of the number of nexthops, the number of paths and the length of paths, while Drouting outperforms Deflection in the probability of success of failure avoidance.
key words: *multipath routing architecture, network reliability, failure avoidance*

1. Introduction

Internet communications are disrupted by failures of routers, circuit failures, fibre optic cable cuts, misconfigured routers, software bugs in routers, and many other causes. In general, communication reachability is recovered through the recomputation of alternative routes. Routing systems have the responsibility to detect failures and to recompute alternative routes.

An important problem arises here that there are many cases where the routing system cannot detect the failures, due to its unlimited variety of the causes. For example, malfunctions caused by either hardware or software bugs in the forwarding plane cannot be detected through OSPF Hellos [1] or BGP Keepalives [2]. In case of a failure that cannot be detected by the routing system, it may take a long time (e.g. a few hours) for the network to recover, as the network administrator informed of the failure by users examines and recovers it manually. Although the length of the failure duration depends on the individual network, past studies showed that a small fraction of failures can last more than an hour. In a study at an ISP [3], approximately 20% of failures lasted more than two hours. It is shown in [4], [5] that about 10% of real failures lasted more than 20 minutes, and approximately 5% lasted more than one hour, in the Sprint backbone.

Many methods have been proposed to minimize service downtime, such as multipath routing, IP restoration and path protection (see Sect. 2 for related work). Recently a method called Deflection has been proposed [6]. Deflection constructs a multipath routing on the existing shortest-path machinery and utilizes those roundabout paths when the packet tag is changed by the source host, as explained in Sect. 3.

In this paper we propose a novel routing architecture called Drouting. Drouting simplifies and generalizes Deflection and improves the ability to route around failures. The Drouting architecture consists of two components. First is the multipath route calculation component. The route calculation component computes multipath routes by constructing Directed Acyclic Graphs (DAGs) that include all links in the network. DAGs that include all links in the network have not been studied for the purpose of Internet routing. The second component of Drouting is the tag forwarding framework. The IP packet forwarding procedure is slightly changed to select the actual nexthop from the precomputed multipath routes based on the packet tag. A packet tag corresponds to a network path deterministically without any state management on routers. The tag forwarding component enables end hosts to change a route dynamically, which can be based solely on user preferences.

We evaluated Drouting by comparing it to Deflection using simulations. The comparisons were on: (1) the number of nexthops, (2) the number of paths, (3) the length of paths and (4) the probability of avoiding failures by changing the packet tag. Drouting exhibits similar results with Deflection in the first three points, while in terms of the failure avoidance probability, Drouting outperforms Deflection on most topologies.

The rest of this paper is organized as follows. Related works are given in Sect. 2. In Sect. 3 the Deflection is reviewed as a competitor. In Sect. 4 we describe our architecture and how it works. In Sect. 5 we present the evaluation results and its analysis. This paper is concluded in Sect. 6.

Manuscript received June 4, 2007.
Manuscript revised October 17, 2007.
[†]The author is with the Graduate School of Media and Governance, Keio University, Fujisawa-shi, 252-8520 Japan.
[††]The authors are with the Faculty of Environmental Information, Keio University, Fujisawa-shi, 252-8520 Japan.
[*]Presently, with the graduate school of Japan Advanced Institute of Science and Technology.
a) E-mail: yasu@sfc.wide.ad.jp
b) E-mail: kusumoto@wide.ad.jp
c) E-mail: osamu@wide.ad.jp
d) E-mail: jun@wide.ad.jp
DOI: 10.1093/ietcom/e91–b.5.1403

2. Related Work

RON [7], Detour [8] and PlanetLab [9] utilize overlay networks for the purpose of failure avoidance. MPDA [10], MDVA [11] and MPATH [12] are multipath routing algorithms that calculate non-shortest multipaths. They use a condition for the routing metric called the Loop Free Invariant (LFI). FIR [13] computes routing tables per network interface using the previous hop information in the routing calculation in order to improve network availability when transient failures occur. IPFRR [14] explores calculating loop-free alternate nexthops using a routing metric condition similar to LFI. Deflection [6] extends LFI (which is equivalent with what they call Rule-1) and calculates more multipaths depending on the previous hop. It also proposed a packet tag system for end-systems to "deflect" the path upon failures.

Until Deflection, little has been discussed regarding an actual architecture to utilize multipaths for the purpose of failure avoidance.

3. Deflection Architecture

In this section we review the Deflection architecture [6] to understand for later comparison with our architecture. Deflection is akin to multipath routing schemes. It calculates multipaths using the specific conditions of routing metrics and the identity of previous hop in the forwarding of the packet. The nexthop set that forms a loop-free path is called the "deflection set." There are three conditions of routing metrics for neighbor to decide whether or not the neighbor can be put in the deflection set. The first, called Rule-1, is equivalent to LFI [10]–[12]. Rule-1 calculates only a small deflection set for a router, and the use of a Rule-1 deflection set does not achieve sufficient failure avoidance capability. Hence, the second condition, Rule-2, is made to extend Rule-1 by using the identity of the previous hop. Rule-2 calculates many members in the deflection set, and provide sufficient failure avoidance capability, but it can result in a backtracking path that transit the same node twice. For this reason the path of Rule-2 tends to become longer. To avoid this problem, Rule-3 is introduced with more complex conditions to prevent immediate backtracks in a path.

Among the three rules, Rule-3 is the best solution and is the major competitor with our method, since Rule-1 calculates too few nexthops and Rule-2 calculates unrealistic network paths, which backtrack and transit the same node twice. For this reason, Rule-2 is not considered as a viable competitor in this paper.

Deflection allows the end host to voluntarily try a different path through the network. In Deflection, packet tags are used to switch paths within the multipath set. When the end host detects (or suspects) a failure in the network that the routing system has not yet corrected, it can switch the packet tag and possibly bypass the failure until the network heals itself. If, after changing the packet tag, packets still do not reach their destination, the end host can try another packet tag. Up to ten packet tags are tested; the first five tags to be tested are 1 through 5, then the next five are chosen randomly from the range $[6, 1023]$.

In order for routers not to synchronize with other routers in terms of the forwarding direction for the same packet tag, Deflection proposes to prepare for each router a prime number p from the first primes (e.g. the first 10) greater than or equal to k (where k is the size of the deflection set). Then the router uses p to form $n = (t \bmod p) \bmod k$, where t is the packet tag, and the packet is forwarded to the n-th member in the deflection set. We obey these rules in later simulations.

There are paths that cannot be used in the network, because of the following two reasons. First, two or more routers may choose the same prime number for p. If this condition occurs, the routers will synchronize in terms of selection of the n-th nexthop in the deflection set, resulting in the possibility of non-used combinations of nexthops. Second, because the number of possible tags is 1024, at most 1024 paths can be used for a source-destination pair. If the network has more than 1024 paths in a source-destination pair, those paths greater than the number 1024 cannot be used.

Assuming m and n are the number of links and nodes in the graph respectively, and k is the number of neighbors for a router, complexity for Deflection Rule-3 in a router is that of $3k$ times[†] Dijkstra's SPF [15], hence $O(km + kn \log n)$. Calculation of deflection sets for all routers in the network can be done by one Dijkstra for each router plus one Dijkstra for each case of one link removed in the network graph. Hence, the complexity is $O((m + n)(m + n \log n))$.

4. Drouting architecture

4.1 Overview

Drouting is similar to Deflection in that both calculate multipath routes and switch a path using a packet tag. Compared to Deflection, Drouting generalizes the specification of the packet tag. In Drouting the packet tag is a random value, while in Deflection choosing specific packet tags is proposed. Another difference is the method of calculating the multipath. Deflection calculates multipath routes by using the identity of the previous hop (incoming hop) of a packet and relations between routing metrics. Drouting does not depend on routing metrics and calculates simple hop-by-hop multipath routes without using the identity of the previous hop.

The route calculation component of Drouting computes multipath routes by constructing Directed Acyclic Graphs (DAGs) that include all links in the network. DAGs that include all links in the network have not been studied for

[†]Deflection calculates the shortest paths on the graph without link to the nexthop rooting from the nexthop, on the graph without the link from previous hop rooting from the router itself, and on the graph rooting from previous hop.

the purpose of Internet routing.

The tag forwarding component enables end hosts to dynamically change a path based on user preferences. A packet tag is assigned to a network path deterministically without having to maintain any states on routers. The packet tag is randomly chosen. A source host changes its packet tag only when it desires to use another network path. In order to avoid packet reordering and degradation of TCP performance, source hosts are assumed to assign the same packet tag for all the packets in one TCP session.

In the beginning, the source host initiates a communication using packet tag 0. The packet tag 0 is treated as special packet tag that instructs the network to use the default shortest path. The source host is assumued to detect problems on the communication path in some way such as a fixed timer for packet losses or dynamic bandwidth estimation (for methods to detect the bandwidth problems, refer to [16]. This problem is beyond the scope of this paper). Once the source host detects a problem, it randomly chooses a new packet tag, such as `0x159bf`. The new packet tag is expected to be assigned to a new communication path, which may stochastically avoid the problem.

Introducing a special packet tag 0 provides the separation of default routing plane and backup routing plane. Although Drouting can be used also for the default routing plane, this paper focuses on the use only in the backup routing plane, in order to compare the failure avoidance property with Deflection fairly. When Drouting is used also for default routing plane, a failure case (a combination of source-destination pair and a failure node, upon which comparisons are based) for Drouting and Deflection would be different, because Drouting's default path may traverse the failure node depending on the value of the first packet tag. (See Sect. 5.5 for the details of the failure avoidance simulation.) Thus the use of Drouting also in default plane makes the evaluation of failure avoidance property by comparison harder, and studying the use of Drouting in default routing plane is left as future work.

In our architecture, a source node can neither predict nor specify in advance which network path will be assigned for its packets. A network path is only randomly assigned to a packet as a result of forwarding the packet with the particular packet tag. However, a source node can specify the same network path for multiple communication sessions to the same destination, by using the same packet tag.

Changing the packet tag enables avoidance of a long failure even if the routing system fails to detect the network failure. If a failure occurs in the network and the routing system can detect the failure, the routing system will automatically recompute network routes, hence altering network paths to avoid the failure, the same as in the existing Internet. A source host can use an alternative path by changing the packet tag, regardless of whether or not the routing system detects (and hence will route around) the failure. By changing the packet tag, the network path which the packet will take may or may not be changed, depending on the randomly generated new packet tag.

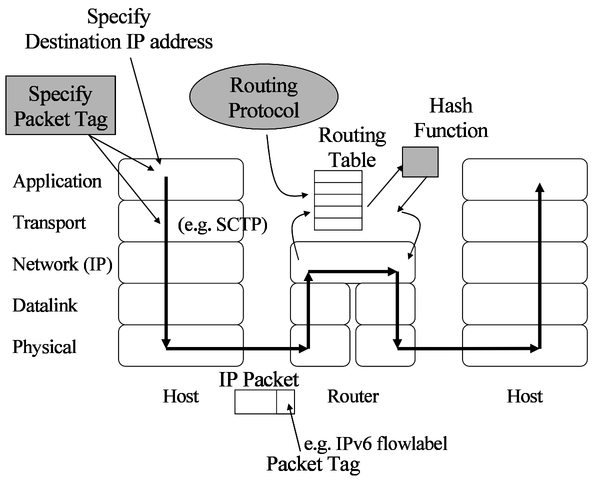


Fig. 1 Drouting architecture.

Our architecture can be used to minimize perceived network downtime as end hosts do not have to wait for the routing system to finish route recomputations. Furthermore, end hosts can possibly address QoS performance problems by changing the packet tag when the performance of the current path deteriorates. For example, congested links and/or overloaded routers can be avoided to improve QoS performance.

Figure 1 shows an overview of the Drouting architecture. Components modified from the existing routing architecture are shaded. Notice that Drouting is an intuitive extension of the basic multipath routing. Only the routing protocols and packet forwarding engine are slightly changed without adding a significant new complication. The packet tag is assumed to be stored in the IPv6 flowlabel [17], hence no packet format modification is needed. Because the length of the IPv6 flowlabel field is 20 bits, the packet tag has the range [`1-0xfffff`] in this paper. Changing the packet tag is the task of either application software or a transport protocol such as SCTP [18]. The Routing algorithm in the routing protocol is discussed in Sect. 4.2. The actual design of the routing protocol is beyond the scope of this paper. The packet forwarding framework is discussed in Sect. 4.3.

4.2 Multipath Route Calculation

To guarantee loop-free packet forwarding and that each packet will eventually reach its destination, routes must be calculated such that routes for a given destination construct a DAG that sinks to the destination. Each route corresponds to a directed link in the DAG. By following the nexthops of the routes hop-by-hop in the DAG, a packet will eventually reach its destination without any routing loop.

In the Drouting architecture we propose a method to construct DAGs that include all of the links in the network graph, in order to provide the maximum number of alternative paths in hop-by-hop routing. The difference between the DAGs utilized by the existing routing systems and the DAGs that include all links in the network graph is illus-

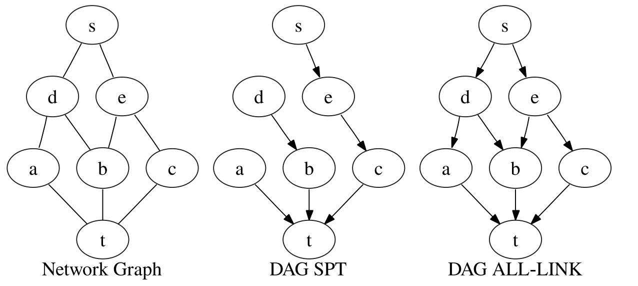


Fig. 2 Existing routing DAG and DAG that includes all links.

trated in Fig. 2. Shortest Path Tree (SPT), as shown in the middle of the figure, has been used for routing on the example network graph shown in the left of the figure. We propose to use the DAG that include all links (ALL-LINK), as shown in the right of the figure, to improve the failure avoidance capability.

A DAG that includes all links can be constructed by topological ordering [19] in which nodes are labeled from 1 to n and the links are directed from the higher-labeled node to the lower-labeled node. To construct a DAG that sinks to the destination, labeling is started from the destination node.

Ohara and Imahori propose [20] constructing DAGs using Maximum Adjacency (MA) ordering [21] to make DAGs that have the maximum number of paths to the destination. Drouting employs this method, and the failure avoidance property is investigated through simulations for the first time in this paper. The formal problem definition and proof for the multipath calculation method are given in [20].

The complexity of the MA ordering to calculate a DAG is $O(m + n \log n)$. Each router must calculate all DAGs for each destination. Hence, the complexity of Drouting for both a router and the overall network is $O(mn + n^2 \log n)$.

4.3 Tag Forwarding

An end host randomly chooses the ten tags for Drouting in the range [1,`0xfffff`]. The network path to be used by a traffic flow will be selected stochastically from the combination of the nexthops in all intermediate routers.

In Drouting, packet tags are random values. Based on this random value, and through the random switching of a nexthop in the nexthop set in the routers, a stochastic selection of a network path for a packet tag is achieved.

Each intermediate router performs a routing table lookup by using the IP destination address as the key (as in the existing Internet). The resulted routing table entry contains multiple nexthops to the destination. The router selects the nexthop randomly by the packet tag. The assignment of a packet tag to a nexthop is performed deterministically, such that: (1) intermediate routers do not need to maintain the state of each traffic flow (i.e., TCP session or IP source-destination pair), and (2) randomly chosen packet tags yield a network path randomly.

We employ a perfect hash function $F_x : U \mapsto U$ (U is the set represents the tag space) which is different for each router v_x. The nexthop selection is $n = F_x(t) \bmod k$ (where t is the packet tag, k is the number of nexthops in the routing table entry, and the selected nexthop is the n-th nexthop in the routing table entry). The F_x can be implemented as follows. First, prepare an table tab[$|U|$] and initialize tab[t] $= t$ for each tag $t \in U$. Next, swap tab[t] with tab[i] of a random index i for each tag t.

4.4 Differences from Deflection

A summary of the differences of Drouting from Deflection is described here.

- Deflection limits the number of tags to 1024, hence the number of paths Deflection provides are up to 1024. Drouting limits the number of paths to `0xfffff`−1 (subtracting special default tag 0).
- The prime number chosen by each router in Deflection may synchronize. This leads to unused combinations of nexthops, i.e., unused paths. Drouting does not have this problem.
- The methods used to calculate multipath routes are different. The method of Deflection considers relations between routing metrics to calculate multipaths, while Drouting calculates the multipaths with maximum connectivity to the destination. Complexity for a router is $O(km + kn \log n)$ in Deflection, and $O(mn + n^2 \log n)$ in Drouting. This difference comes from the fact that, if a Deflection router has k neighbors, it calculates the routing tree k times, once assuming each neighbor is the previous hop (removed from the graph to avoid immediate backtracking). Hence the complexity of Deflection is parameterized by k (the number of neighbors). In contrast, Drouting decides the routing graph (basically not tree, because it includes all links), for each destination. Hence the complexity of Drouting is parameterized by n (the number of destination nodes). The difference of complexity is hence that of k (more precisely, $3k$) and n, and can be deemed to be negligible. Moreover, Deflection requires the memory space to hold the results of each of $3k$ Dijkstra runs to finish the calculation process for a router. In Drouting it is simpler because the result for each destination is independent and the memory used for the calculation of different destination can be released. In summary, the complexities for a router to execute routing calculation in a network in both methods are equivalent to a constant multiple of Dijkstra, and hence they are both realistic and reasonable.

5. Evaluation

5.1 Simulation Environment

We evaluated Drouting architecture by comparing it against Deflection on several topologies using simulations. The topologies were obtained from Rocketfuel [22] and BRITE

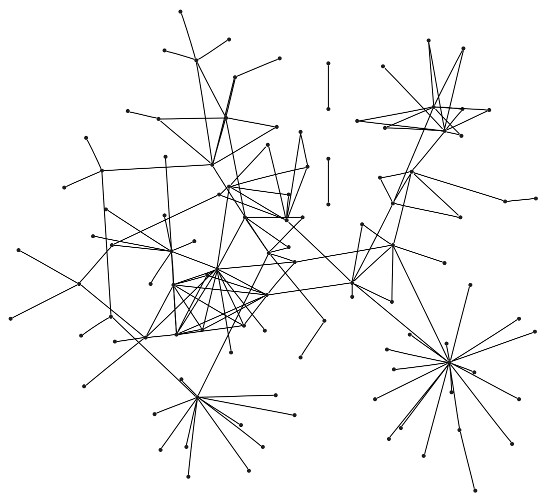


Fig. 3 Network topology of Telstra (AS1221).

Table 1 The network graphs.

Name	#nodes	#links	description
Telstra	108	153	AS1221
Sprint	315	972	AS1239
Ebone	87	166	AS1755
Tiscali	161	328	AS3257
Exodus	79	147	AS3967
Abovenet	141	374	AS6461
BA-100-invcap	100	390	by BRITE
BA-100-minhop	100	390	by BRITE

[23].

The comparisons were done in four aspects: (1) the number of nexthop, (2) the number of paths, (3) the length of paths, and (4) the probability of success in avoiding failures. We compare four cases: Deflect-1, Deflect-2, Deflect-3 and Drouting. The notations Deflect-1, Deflect-2, Deflect-3 indicate Deflection routing Rule-1, Rule-2, Rule-3 in [6] respectively. Deflect-3 is the major competitor, as mentioned in Sect. 3.

Although the results are completely network graph dependent, as for the number of nexthops, the number of paths and the length of paths, we present here the comparison results on the Telstra (AS1221) topology from Rocketfuel. The descriptions and explanations of this section related to the number of nexthops and paths and the length of paths apply only to the Telstra network topology. The results for other topologies are given in Appendix A. The network topology of Telstra is shown in Fig. 3. For the probability of failure avoidance, the results on all topologies are shown. All topologies consist of Telstra, Sprint, Ebone, Tiscali, Exodus, Abovenet, BA-100-invcap and BA-100-minhop. The sizes of the network graphs are shown in Table 1. Telstra, Sprint, Ebone, Tiscali, Exodus and Abovenet are ISP topologies inferred by Rocketfuel. BA-100-invcap and BA-100-minhop are synthetic topologies generated by BRITE. The routing metric of BA-100-invcap is inversely proportional to the bandwidth. The routing metric of BA-100-minhop is 1

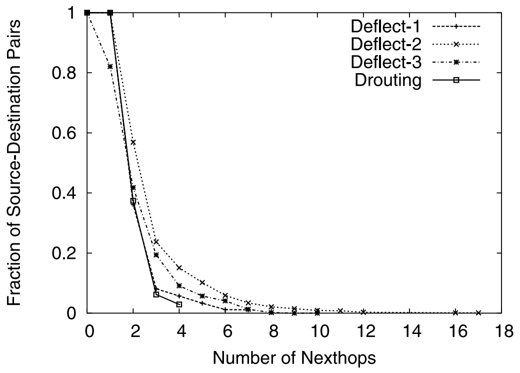


Fig. 4 The distribution of number of nexthops in a routing table entry in a simulation of the Telstra topology (AS1221).

for all links (i.e., minimum hop routing).

5.2 Number of Nexthops

We present the comparison based on the number of nexthops in routes for each source-destination pair in all routers. First, a smaller number of routers that have only one nexthop to the destination is preferred. This is because routers with only one nexthop may become a single point of failure. Next, the distribution where the majority of routers have a small number of nexthops equally is preferred over the distribution where a small number of routers have huge nexthops and others have a few nexthops. This is because it is desired to provide multipaths for greater number of source-destination pairs.

The Complementary Cumulative Distribution Function (CCDF) of the source-destination pair is plotted against the number of nexthops in a routing table entry in Telstra topology in Fig. 4. In the figure, Deflect-2 has the largest number of nexthops. Then, briefly, Deflect-3, Deflect-1 and Drouting follow in the order. Deflect-1 and Drouting have largely similar numbers of nexthop distribution. In Deflect-1, 36.1% of the source-destination pairs have 2 or more nexthop entries. In Drouting, 37.4% have.

The result seems to show that Drouting does not calculate enough nexthops to provide sufficient multipaths. However, as shown later in Sect. 5.3, Drouting provides enough multipaths. The fact that Drouting calculates the least nexthops for similar or better performance means that Drouting is more memory efficient than Deflection.

Deflect-3 fails to calculate a non-empty deflection set for 17.9% of the source-destination pairs. This is due to the stub nodes in the Telstra topology and the fact that Deflect-3 does not calculate a backtracking path.

5.3 Number of Paths

This section compares the number of alternative paths between a source and a destination. The number of paths in Deflection was counted by examining tags from 0 to 1024 and counting the unique paths, as was done in [6]. The number of paths in Drouting is counted by enumerating all possible paths. The numbers of paths counted here represent the actual paths that can be used in each routing method.

Figure 5 shows the numbers of paths for each routing method in Telstra topology with the focused version on the range below 100 paths. Deflect-1 provides only a small number of paths where almost all source destination pairs have only less than 10 paths. Although Deflect-2 has the largest number of paths, Deflect-2 is not a viable competitor as mentioned in Sect. 3. Deflect-3 and Drouting provide largely similar numbers of paths. It is surprising that even though the numbers of nexthops between Deflect-1 and Drouting is largely the same (Fig. 4), the numbers of paths they provide are completely different. We observe that the route calculation method of Drouting provides a significant number of paths by using a small number of nexthops efficiently.

In Fig. 5, Drouting provides a small number of paths (less than 20) to more source-destination pairs than Deflect-3. This is preferred, because all nodes having a similar number of alternative paths are preferred over the case where only small number of nodes have huge alternative paths. Hence, among viable competitors, Drouting was the best in this case.

5.4 Path Length

Here we compare the length of paths that each routing methods provide. The shorter the length of the path becomes, the better the routing we obtain, since generally a longer path means more communication delay. The average and maximum path lengths in nodes over multipaths for each source-destination pair in Telstra are shown in Fig. 6. For both average and maximum path lengths, the relations between each routing methods are largely the same. Deflect-2 tends to provide longer paths, and the maximum path length is 28 nodes. Drouting provides paths which are significantly shorter than Deflect-2 but yet slightly longer than Deflect-3. For example, the fraction of source-destination pairs that has an average path length of more than 10 nodes is 21.0% in Deflect-3, while in Drouting it is 31.7%. Deflect-1 provides shortest paths in which the length of almost all paths are less than 10 nodes.

In summary, Drouting provides slightly longer paths in Telstra. The length of paths is completely the graph dependent, and it can be very long when Drouting is employed. This is a trade-off for improved failure avoidance capability (see Sect. 5.5).

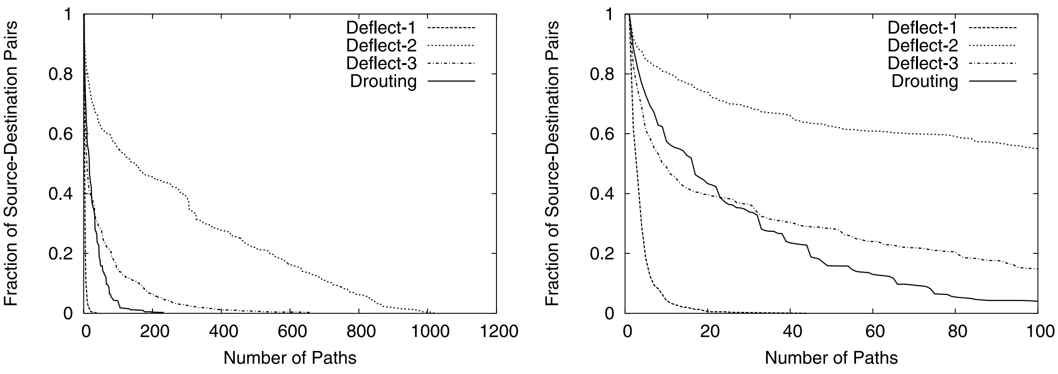


Fig. 5 The number of paths per source-destination node pair in Telstra (left) and the focused version on the range below 100 paths (right).

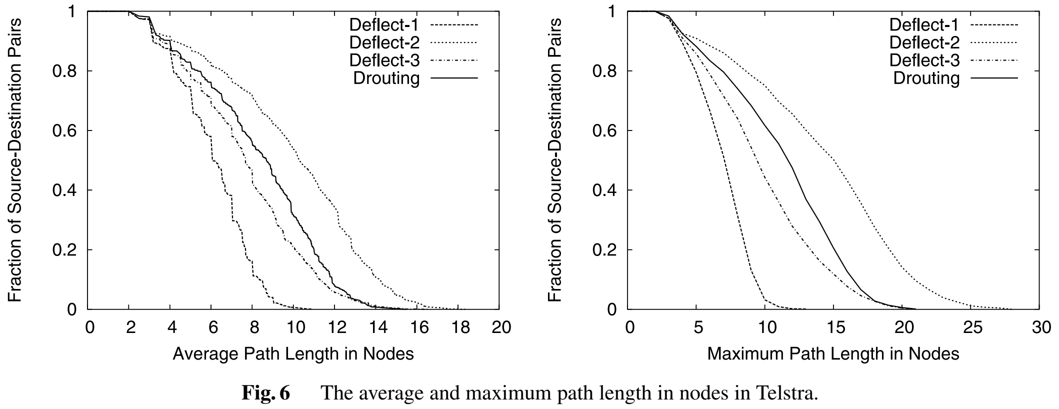
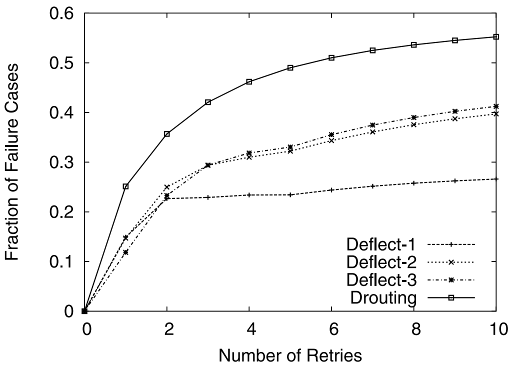


Fig. 6 The average and maximum path length in nodes in Telstra.



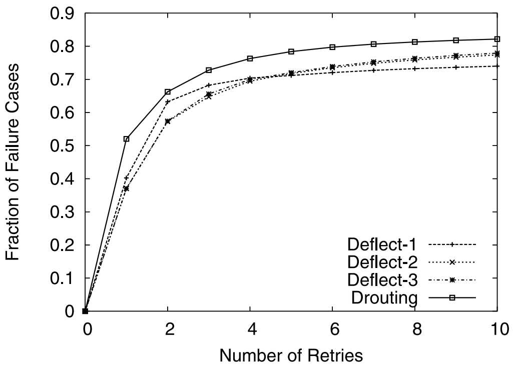
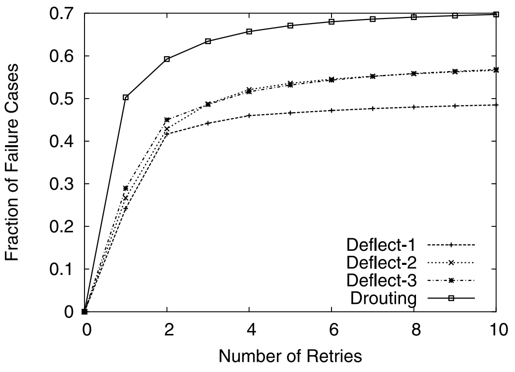


Fig. 7 The failure avoidance for a failure in Telstra (left) and Sprint (right).



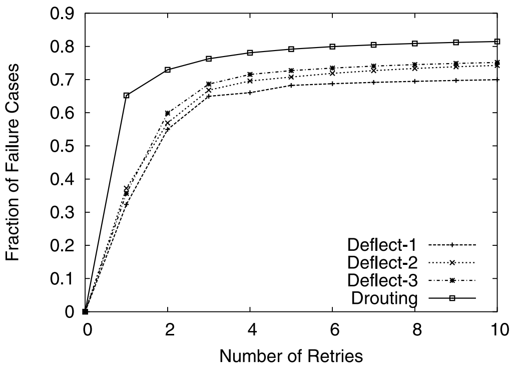


Fig. 8 The failure avoidance for a failure in Ebone (left) and Tiscali (right).

5.5 Failure Avoidance Simulation

The failure avoidance simulations were done as follows. First, we randomly chose a node to fail. (Additional simulation results for 10 failed nodes are given in Appendix B). Then, for every source-destination pair, we check if the failure node is on the default shortest path calculated by the Dijkstra calculation. For each source-destination node pair that includes the failure node in the default shortest path (this is called “a failure case”), a tag was chosen to see if the retry using the new tag avoids the failed node in forwarding the packet. If the retry did not avoid the failed node, another new tag was tested. At most ten tags were tested. For each failure case, we recorded the number of tags tested before the failed node was successfully avoided. This procedure was executed for each method, Deflection and Drouting, with the same source-destination pair and failure node. We checked 1000 random failure cases for both Deflection and Drouting, and recorded the fraction of failure cases in which the failure was successfully avoided within x tag changes (retries).

Notice that the assumption here is that Drouting is used as a backup routing plane, where the routing plane is constructed separately from the default routing plane. This is just like Deflection, and is done to compare with Deflection fairly. This assumption makes Fig. 1 incorrect in a precise sense, because after the assumption a router has two routing tables, one for default routing and the other for Drouting.

The result of failure avoidance simulations for Telstra and Sprint, Ebone and Tiscali, Exodus and Abovenet, and BA-100-invcap and BA-100-minhop are shown in Figs. 7, 8, 9, 10, respectively. Notice that although actual value increased depends on the individual network, the failure avoidance probability is increased on all topologies. The Drouting improves not only the probability of recovery after ten retries, but also the probability on the first retries.

As for Drouting, the simulation results shown here utilize the retry packet tag range of [0-`0xfffff`] instead of [1-`0xfffff`], which is not precisely the same as the expected deployment as backup routing plane in real network. However, the difference is negligible; it is expected that just 1 smaller range of packet tag does not influence the major results of the failure avoidance property.

5.6 Observations

The improvement of the failure avoidance property of Drouting stems from the way MA ordering calculates the multipath on the network graph. MA ordering chooses the node with the maximum connectivity to the node set that has already decided the routes to the destination, in each of its steps. The decision of routes to destination on a node with smaller connectivity is postponed, so that the node can utilize the routes on other nodes with more connectivity later when the node is decided on its routes. This way MA or-

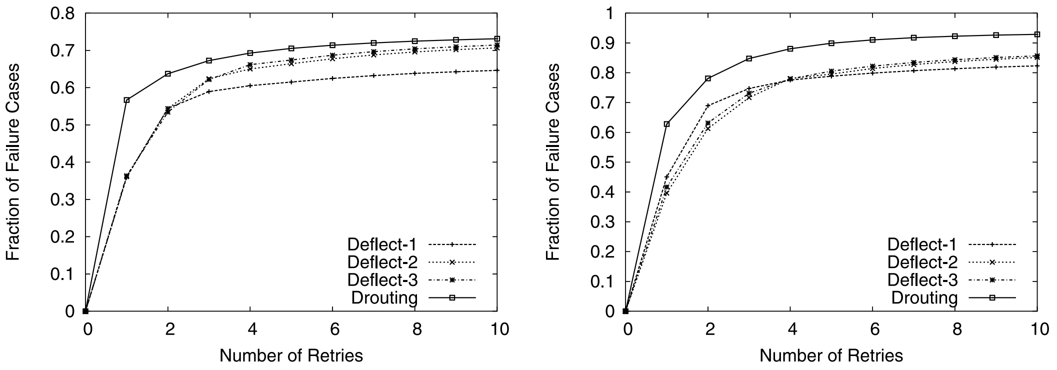


Fig. 9 The failure avoidance for a failure in Exodus (left) and Abovenet (right).

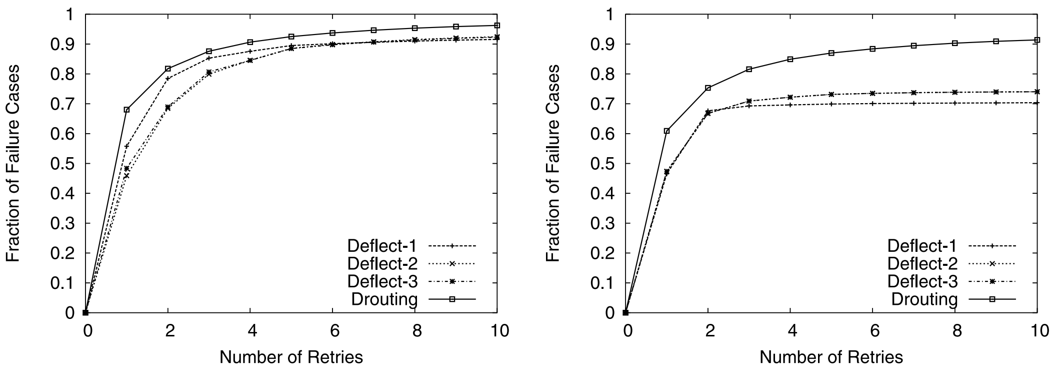


Fig. 10 The failure avoidance for a failure in BA-100-invcap (left) and BA-100-minhop (right).

dering maximizes the minimum connectivity among all the nodes to the destination [20]. This enables a smaller number of nexthops to result in a larger number of paths to the destination using efficient utilization of combinations of the nodes with the small number of nexthops.

The number of paths in Drouting concentrate in lower numbers among all nodes, compared to Deflect-3 (Fig. 5). Deflect-3 distributes the number of paths uniformly among nodes, namely the number of nodes with more paths and with lesser paths are both increased compared to Drouting. This negatively affects the failure avoidance property, since the more nodes with lesser paths, the smaller the possibility to avoid failure at these points.

Deflection methods are based on shortest path routing, considering a smaller routing metric to be better. Hence except Deflect-2 which backtracks, Deflection chooses shorter paths and prunes longer paths. This leads to a relatively smaller number of hops in the paths. On the other hand, Drouting which uses MA ordering, does not consider routing metrics and shortness of paths, and calculates also longer, roundabout paths. The existence of longer roundabout paths appears both in average and maximum path length (Fig. 6).

6. Conclusion

We have presented the Drouting architecture, which enables user-driven change of traffic paths. Users or end hosts can avoid failures even when the routing system fails to detect the failures. Hence Drouting contributes to improving the robustness and reliability of the network.

Simulations showed improved failure avoidance probability compared to the previous work called Deflection. We studied the properties of Deflection and Drouting in four aspects, namely the number of nexthops, the number of paths, the length of paths and the failure avoidance probability. The results showed significant improvement in failure avoidance probability while the other properties are reasonable.

There are a number of beneficial characteristics in the Drouting architecture. First, many multipath routes can be held in a Drouting network and utilized effectively. Many multipath routes are further expected to be used for the purpose of load-balancing and QoS-oriented routing. Second, calculating routes by constructing DAGs that include all links in the network can increase the probability to recover reachability. The probability increases as the network is more redundant. Third, the Drouting architecture is a simple and intuitive extension of the current routing architecture. The simplicity of the Drouting architecture preserves the most beneficial properties of the Internet such as extendability. Last, network administrators in internet service providers can enforce their routing policies by using route filters in constructing DAGs.

The future work is as follows. This paper calculates DAGs in a centralized way. But in practice, the DAGs and multipath routes should be calculated in a distributed fashion, where each router calculates them independently and autonomously. The distributed algorithm and the protocol for the distributed computation of DAGs are left as future work. Implementation of traffic engineering and QoS routing on top of Drouting have not been addressed. Further simulations employing more complicated user traffic models have to be studied, where many users located across the network change their paths independently.

It is anticipated that the Drouting architecture can be applied also at the inter-domain routing level. Because the Drouting architecture abstracts networks in general graph structures, it can be applied to both the intra-domain level (the nodes are routers) and the inter-domain level (the nodes are Autonomous Systems (ASes)). Many issues, such as designing the new inter-domain routing protocol, are left as future work for the inter-domain routing level. In particular, the multipath routing calculation method used in this paper, MA ordering, is not considered appropriate because it requires the synchronization of the entire topology among all nodes (in other words, synchronization of the entire Internet topology among all ASes). This is not realistic, and hence we will need a new multipath routing calculation algorithm that enables distributed computation and enforcement of AS policies.

References

[1] J. Moy, "OSPF version 2," RFC 2328, IETF, April 1998.

[2] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)," RFC 4271, IETF, Jan. 2006.

[3] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of internet stability and backbone failures," FTCS'99: Proc. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing, p.278, IEEE Computer Society, Washington, DC, USA, 1999.

[4] G. Iannaccone, C. nee Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," IMW'02: Proc. 2nd ACM SIGCOMM Workshop on Internet Measurment, pp.237–242, ACM Press, New York, NY, USA, 2002.

[5] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.N. Chuah, and C. Diot, "Characterization of failures in an IP backbone network," INFOCOM, 2004.

[6] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," SIGCOMM'06: Proc. 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp.159–170, ACM Press, New York, NY, USA, 2006.

[7] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," SOSP'01: Proc. Eighteenth ACM Symposium on Operating Systems Principles, pp.131–145, ACM Press, New York, NY, USA, 2001.

[8] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of internet path selection," SIGCOMM'99: Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp.289–299, ACM Press, New York, NY, USA, 1999.

[9] A. Nakao, L. Peterson, and A. Bavier, "Scalable routing overlay networks," SIGOPS Oper. Syst. Rev., vol.40, no.1, pp.49–61, 2006.

[10] S. Vutukury and J.J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," SIGCOMM'99: Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp.227–238, ACM Press, New York, NY, USA, 1999.

[11] S. Vutukury and J.J. Garcia-Luna-Aceves, "MDVA: A distance-vector multipath routing protocol," INFOCOM, pp.557–564, 2001.

[12] S. Vutukury and J.J. Garcia-Luna-Aceves, "An algorithm for multipath computation using distance-vectors with predecessor information," Proc. Eight International Conference on IEEE Computer Communications and Networks, pp.534–539, Oct. 1999.

[13] S. Lee, Y. Yu, S. Nelakuditi, Z.L. Zhang, and C.N. Chuah, "Proactive vs reactive approaches to failure resilient routing," INFOCOM, 2004.

[14] A. Atlas and A. Zinin, "Basic specification for IP fast-reroute: Loop-free alternates," Internet-Draft, IETF, March 2007. draft-ietf-rtgwg-ipfrr-spec-base-06.txt

[15] E.W. Dijkstra, "A note on two problems in connection with graphs," Numerische Mathematik, vol.1, pp.269–271, 1959.

[16] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls on end-to-end available bandwidth estimation," IMC'04: Proc. 4th ACM SIGCOMM Conference on Internet Measurement, pp.272–277, ACM Press, New York, NY, USA, 2004.

[17] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering, "IPv6 flow label specification," RFC 3697, IETF, March 2004.

[18] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream control transmission protocol," RFC 2960, IETF, Oct. 2000.

[19] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows: Theory, algorithms, and applications, Prentice-Hall, Upper Saddle River, NJ, USA, 1993.

[20] Y. Ohara and S. Imahori, "Computing a DAG using MA ordering for the all-to-one maximum flow routing problem," Mathematical Engineering Technical Report METR2007-09, Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 2007.

[21] H. Nagamochi and T. Ibaraki, "Graph connectivity and its augmentation: Applications of MA orderings," Discrete Appl. Math., vol.123, no.1-3, pp.447–472, 2002.

[22] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," IEEE/ACM Trans. Netw., vol.12, no.1, pp.2–16, 2004.

[23] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," MASCOTS, vol.00, p.0346, Aug. 2001.

Appendix A: Comparison of Nexthops and Paths

In this section, comparisons of nexthops and paths in some topologies other than Telstra are given. Comparisons of nexthops and paths in Ebone, Tiscali, Exodus and Abovenet are given in Figs. A· 1, A· 2, A· 3 and A· 4, respectively. The comparisons in large networks such as Sprint are hard, because calculating the number of path in Drouting involves the enumeration of all paths (see Sect. 5.3), which does not end in polynomial time. Hence, we do not give the comparison on Sprint here.

Notice that in Ebone, Drouting calculates the largest numbers of paths to nodes which exceeds even that of Deflect-2. For average and in maximum path length, Drouting is the longest due to the existence of many long, roundabout paths. In Tiscali and Exodus, we observe that the tag forwarding method in Deflection limits the number of paths to 1024, even though there seems to be more possible paths. In Exodus, Drouting exhibits lesser paths despite of similar or better failure avoidance property (Fig. 9), which is simi-

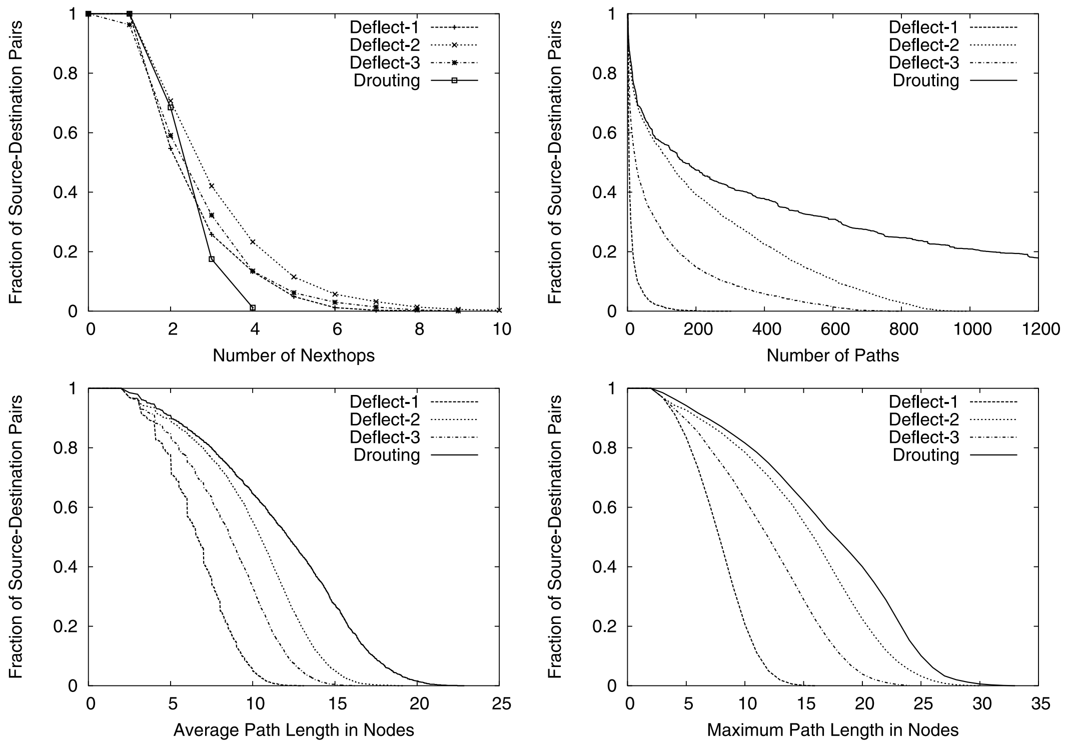


Fig. A· 1 The number of nexthops, paths, average and maximum path length in Ebone.

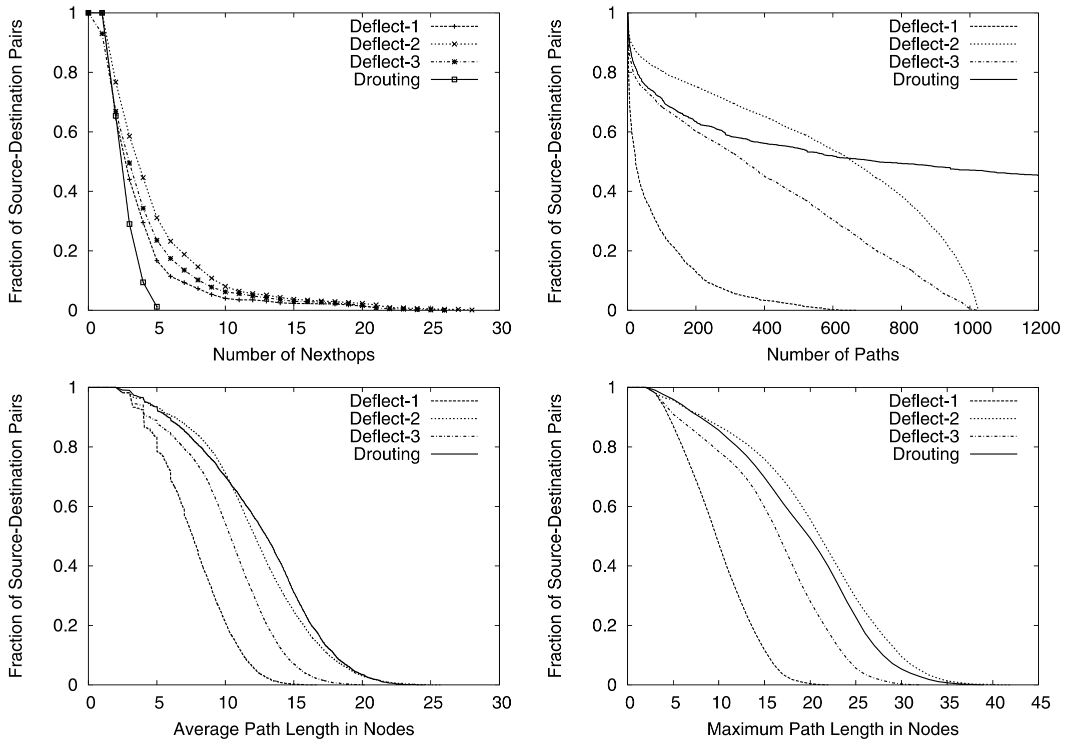


Fig. A· 2 The number of nexthops, paths, average and maximum path length in Tiscali.

lar to the case in Telstra. Futhermore, both the average and maximum length of paths are shorter than Deflect-3.

We attribute these results to the characteristics of each routing methods and the graph structures (including the relations of routing metrics).

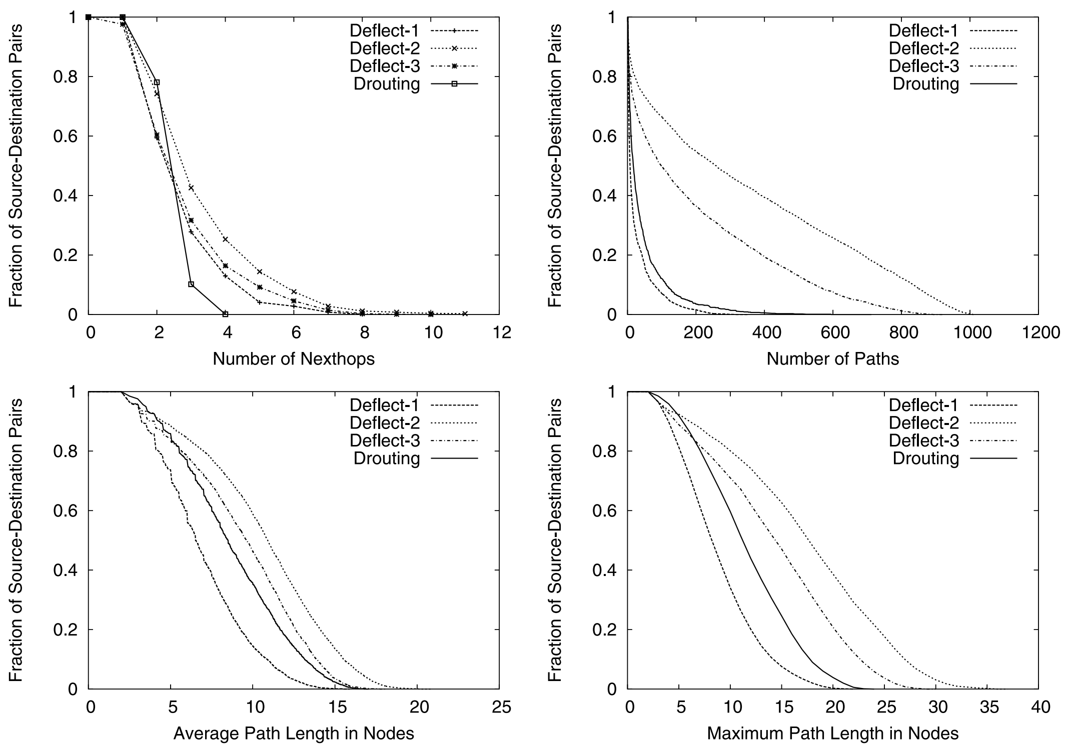


Fig. A· 3 The number of nexthops, paths, average and maximum path lenght in Exodus.

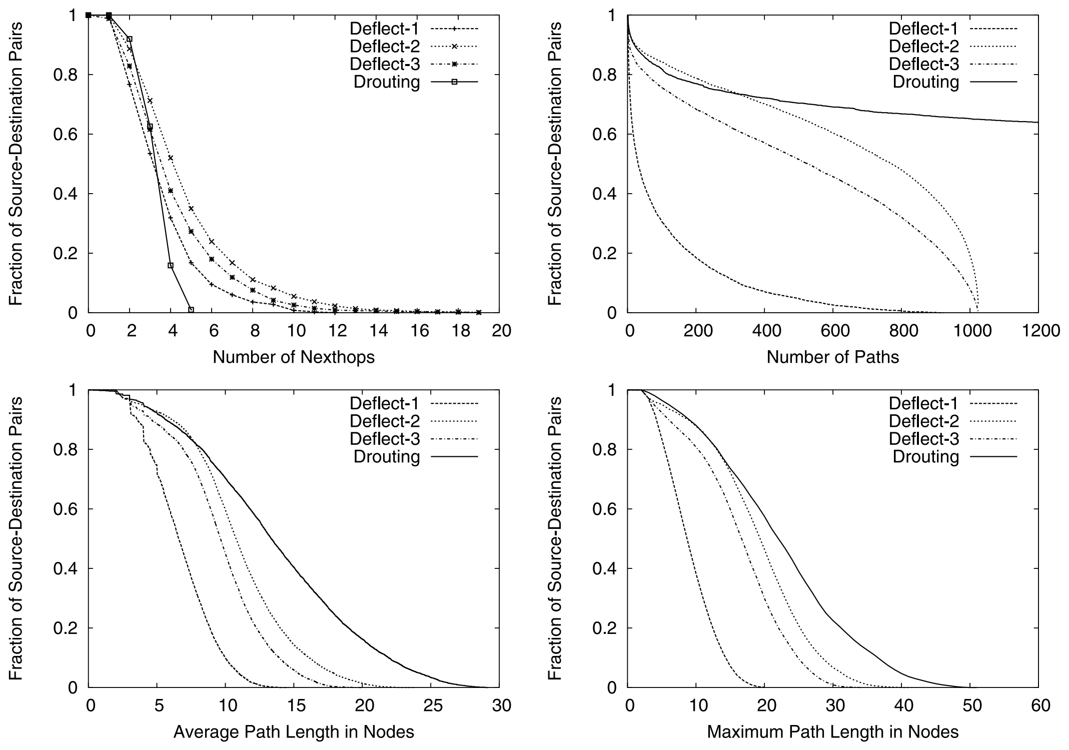


Fig. A· 4 The number of nexthops, paths, average and maximum path length in Abovenet.

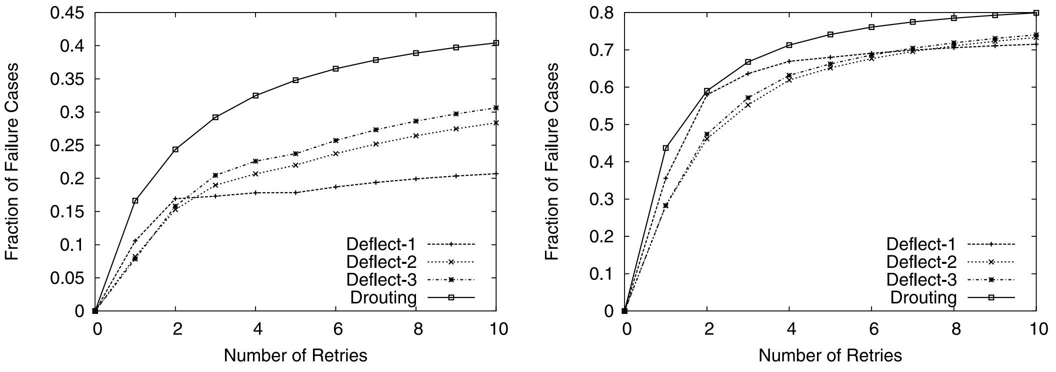


Fig. A· 5 The failure avoidance for 10 failures in Telstra (left) and Sprint (right).

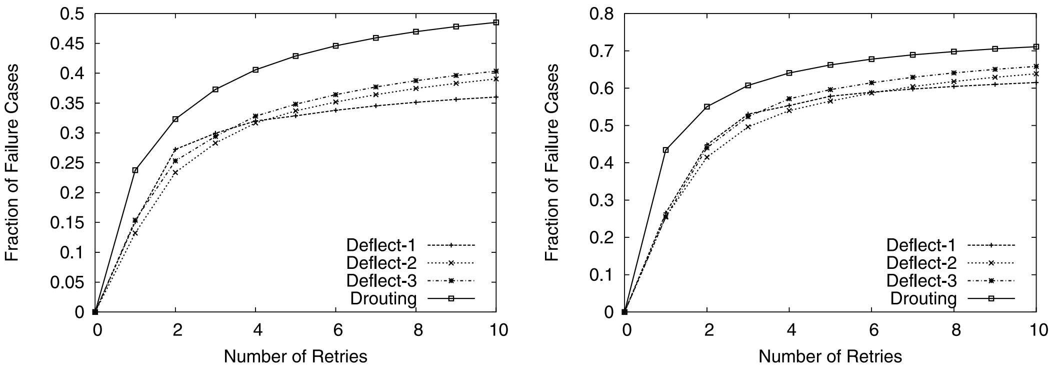


Fig. A· 6 The failure avoidance for 10 failures in Ebone (left) and Tiscali (right).

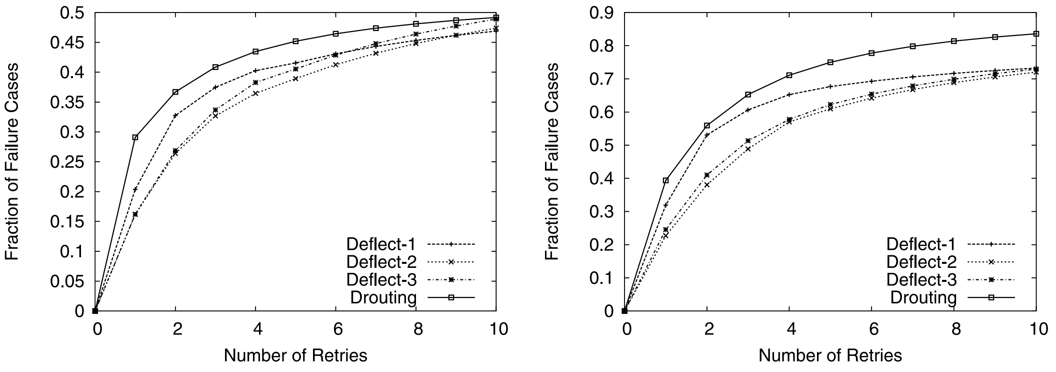


Fig. A· 7 The failure avoidance for 10 failures in Exodus (left) and Abovenet (right).

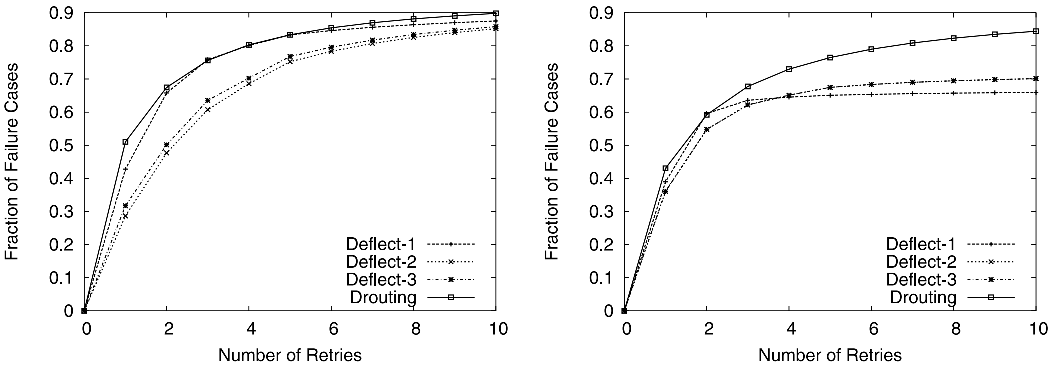


Fig. A· 8 The failure avoidance for 10 failures in BA-100-invcap (left) and BA-100-minhop (right).

Appendix B: Failure Avoidance Simulation Results for 10 Failed Nodes

Additional failure avoidance simulation results for 10 failed nodes for Telstra and Sprint, Ebone and Tiscali, Exodus and Abovenet, and BA-100-invcap and BA-100-minhop are shown in Figs. A· 5, A· 6, A· 7 and A· 8, respectively.

Yasuhiro Ohara graduated from Keio University in 1999, the Faculty of Environmental Information, from Master's Program in 2001, Graduate School of Media and Governance. He is now an assistant professor at the graduate school of Japan Advanced Institute of Science and Technology, specializing in Internet Routing.

Hiroyuki Kusumoto graduated from the Master's Program of Graduate School of Science Osaka University in 1985. He received his Ph.D. from Keio University in 1995. He is now an Associate Professor at the Faculty of Environmental Information, Keio University.

Osamu Nakamura graduated from Keio University in 1982, Department of Mathematics, Faculty of Science and Technology, MS for Computer Science from Keio University in 1984, received Ph.D. in Computer Science, Keio University, 1993. He is currently a Professor, Faculty of Environmental Information, Keio University since April 2006.

Jun Murai graduated from Keio University in 1979, Department of Mathematics, Faculty of Science and Technology, MS for Computer Science from Keio University in 1981, received Ph.D. in Computer Science, Keio University, 1987. He is currently a Professor, Faculty of Environmental Information, Keio University since April 1997, and a Vice-President of Keio University since May 2005. His research interests include computer science, computer networks and computer communication.