2-25-2011

On the Feasibility and Efficacy of Protection Routing in IP Networks

Kin Wah (Eric) Kwong
University of Pennsylvania

Lixin Gao
University of Massachusetts - Amherst

Roch A. Guérin
University of Pennsylvania, roch@guerinfamily.org

Zhi-Li Zhang
University of Minnesota - Twin Cities

Suggested Citation:
K.W. Kwong, L. Gao, R.A. Guérin and Z. Zhang. (2011). "On the Feasibility and Efficacy of Protection Routing in IP Networks." *Transactions on Networking.*

Postprint Version

On the Feasibility and Efficacy of Protection Routing in IP Networks

Abstract

With network components increasingly reliable, routing is playing an ever greater role in determining network reliability. This has spurred much activity in improving routing stability and reaction to failures, and rekindled interest in centralized routing solutions, at least within a single routing domain. Centralizing decisions eliminates uncertainty and many inconsistencies, and offers added flexibility in computing routes that meet different criteria. However, it also introduces new challenges; especially in reacting to failures where centralization can increase latency. This paper leverages the flexibility afforded by centralized routing to address these challenges. Specifically, we explore when and how standby backup forwarding options can be activated, while waiting for an update from the centralized server after the failure of an individual component (link or node). We provide analytical insight into the feasibility of such backups as a function of network structure, and quantify their computational complexity. We also develop an efficient heuristic reconciling protectability and performance, and demonstrate its effectiveness in a broad range of scenarios. The results should facilitate deployments of centralized routing solutions.

Keywords

network, routing, protection, standby, reliability

Disciplines

Digital Communications and Networking

Comments

Suggested Citation:
K.W. Kwong, L. Gao, R.A. Guérin and Z. Zhang. (2011). "On the Feasibility and Efficacy of Protection Routing in IP Networks." *Transactions on Networking.*

Postprint Version

On the Feasibility and Efficacy of Protection Routing in IP Networks

Kin-Wah Kwong, Lixin Gao, *Fellow, IEEE*, Roch Guérin, *Fellow, IEEE*, and Zhi-Li Zhang

***Abstract*—With network components increasingly reliable, routing is playing an ever greater role in determining network reliability. This has spurred much activity in improving routing stability and reaction to failures, and rekindled interest in centralized routing solutions, at least within a single routing domain. Centralizing decisions eliminates uncertainty and many inconsistencies, and offers added flexibility in computing routes that meet different criteria. However, it also introduces new challenges; especially in reacting to failures where centralization can increase latency. This paper leverages the flexibility afforded by centralized routing to address these challenges. Specifically, we explore when and how standby backup forwarding options can be activated, while waiting for an update from the centralized server after the failure of an individual component (link or node). We provide analytical insight into the feasibility of such backups as a function of network structure, and quantify their computational complexity. We also develop an efficient heuristic reconciling protectability and performance, and demonstrate its effectiveness in a broad range of scenarios. The results should facilitate deployments of centralized routing solutions.**

***Index Terms*—Network, routing, protection, standby.**

I. Introduction

Intra-domain routing in IP networks has traditionally relied on distributed computations among routers, with the concatenation of individual forwarding decisions eventually resulting in packet delivery. In spite of their inherent adaptability and scalability, distributed computations can make troubleshooting harder, because of the many sources of inconsistencies they allow. This has renewed interest in centralized routing solutions [7], [9], [27] for IP networks, at least in settings where scalability is less of a concern, *e.g.*, intra-domain routing. Centralizing decisions not only guarantees full visibility into the forwarding state of individual routers (now essentially cheap *forwarding engines* or FEs), it also affords added flexibility in computing paths that meet different requirements.

In spite of its advantages and even when scalability is not an issue, centralizing decisions has disadvantages. Of particular concern for reliability is latency in reacting to failures, *i.e.*, the central server needs to be notified, react to the failure, and communicate updated forwarding information to all affected FEs. This can result in non-negligible "gaps" after failures, during which FEs have no valid forwarding states for some destinations, and translate into substantial packet losses. Given the sub-50ms target for failure recovery (including routing failures) that many modern applications mandate and the fact that several distributed routing solutions are able to meet this target, it is imperative for a centralized routing system to demonstrate similar capabilities. Developing solutions that can achieve this goal is one of the paper's motivations. A natural approach to the problem is through preventive mechanisms, *e.g.*, by having the central server pre-compute forwarding decisions for common (most) failure scenarios, *and* pre-load those in the FEs so that updated forwarding state is locally available. However, such solutions have their limitations. For one, the sheer volume of alternate forwarding states across failure scenarios will likely require that it be stored in "slow" memory to keep costs low. As a result, updating data path forwarding tables could take time. More importantly, even if the central server does not have to download updated forwarding state, it remains responsible for coordinating when and which FEs switch-over to the new state. As discussed in [19], failure to do so can introduce forwarding loops, whose effect can be worse than failures.

Ensuring uninterrupted (or minimally interrupted) packet forwarding in the presence of failures remains, therefore, a significant challenge in centralized routing systems. Our goal in this paper is to explore a possible solution to this problem, and in the process take centralized routing one step closer to offering an effective alternative for intra-domain routing. Furthermore, because a corollary of centralized routing is simplified FEs, we seek to realize this goal with no or minimal impact on data plane complexity. In particular, we want to avoid either encapsulation-based solutions that require additional packet manipulations, as well as packet marking and interface-specific forwarding solutions that often call for significant expansion to the size (and therefore cost) of forwarding tables. Instead, our goal is to allow all (most) FEs to have, for each destination present in their forwarding tables, a pre-configured next-hop to which packets for that destination can be forwarded in case of failure of the primary next-hop(s). The trigger to switch to backup forwarding is entirely local (*i.e.*, unavailability of the primary next-hop(s)), and forwarding loops should be precluded.

In other words, we consider an IP network where (intra-domain) routing is under the responsibility of a central server, so that routers (FEs) are only responsible for (destination-based) packet forwarding. Because of the use of a central server, path computation is not restricted to shortest paths based on a common set of link weights. Instead, each desti-

Kin-Wah Kwong and Roch Guérin are with the Department of Electrical and Systems Engineering, University of Pennsylvania. Emails: kkw@seas.upenn.edu, guerin@ee.upenn.edu

Lixin Gao is with the Department of Electrical and Computer Engineering, University of Massachusetts. Email: lgao@ecs.umass.edu

Zhi-Li Zhang is with the Department of Computer Science and Engineering, University of Minnesota. Email: zhzhang@cs.umn.edu

This work was supported by NSF grants CNS-0627004, CNS-0626617, and CNS-0626808. Part of the results were presented at IEEE INFOCOM 2010.

nation prefix is associated with an "independently" computed (primary) forwarding tree (more generally a directed acyclic graph, or DAG), rooted at the egress node associated with the destination. Our goal is then to compute a set of primary routing trees (or DAGs), one for each destination, so that all nodes in the tree, or when not feasible[1] as many nodes as possible, have a standby alternate next-hop available when the primary next-hop becomes unreachable[2]. We term such a routing, *protection routing*, and introduce it more formally in Section III. Protection routing is readily realized when each node in the DAG has two or more independent next-hops towards the destination, *e.g.*, as sought in [24], [20]. Its simplicity not withstanding, this is easily shown not to be simultaneously feasible for all nodes (at least one node is limited to only one next-hop). Furthermore, it ignores the option for two nodes to mutually protect each other, and exploring the benefits this affords is one of the motivations for this paper. In addition, while standby protection to failures is desirable, its impact on operational performance should also be accounted for. Incorporating this aspect when computing protection routing is another goal of the paper.

The concept of protection routing as just defined bears similarities with a number of related concepts, and we expand on this in Section II. The paper nevertheless makes a number of novel contributions and in particular:

1) It offers new insight into network topological properties that ensure the feasibility of protection routing;
2) It establishes that computing a protection routing is an NP-hard problem;
3) It develops a heuristic for computing a routing that reconciles the often conflicting goals of protectability and performance;
4) It demonstrates the heuristic's ability to realize an effective trade-off between protectability and performance across a range of network topologies, and for several variations of the notion of protectability (see Section III-A for details).

In addition, although the work is primarily motivated by its application to centralized routing systems, the concept of protection routing and the paper's analysis also apply to a distributed setting. However, computing a protection routing in a distributed manner while avoiding loops requires not only that all routers have full knowledge of network topology (*i.e.*, as in link state protocols), but also that they rely on a common and consistent algorithm for computing (standard and protection) routes. This is challenging with the randomized heuristics often required by the intractable nature of many routing problems (all randomization steps need to be identical).

The rest of the paper is structured as follows. Section II reviews related works and contrasts the approach and findings of the paper against them. Section III introduces protectability more formally and defines several possible variants. Section IV presents an analytical investigation of protection routing, while Section V leverages insight from this analysis to develop a computationally efficient heuristic. The heuristic favors protectability, while seeking to minimize its impact on performance. This trade-off is further investigated in Section VI, which develops a modified heuristic that relaxes protectability in exchange for improved performance. Section VII evaluates the efficacy of the heuristics in several different scenarios and for different definitions of protectability. Section VIII summarizes the paper's findings.

II. Related Works

This paper considers a centralized routing system similar to that proposed in [7], [9], [27]. In those works, the primary motivation for centralizing path computation was manageability. In [19], an efficient message-dissemination solution was proposed to minimize signaling overhead and avoid the formation of transient loops in such an environment. This paper builds on these earlier works by assuming a centralized routing solution, but differs in its focus. Its aim is to overcome problems associated with the potential for increased latency after failures, because of the system's reliance on a central server responsible for coordinating updates to the forwarding states of FEs. Our motivations and general approach for handling this issue are similar in principle to those behind many of the IP fast re-routing (IPFRR) schemes that have been proposed (see [25] for a generic introduction to IPFRR and its goals). We expand below on specific differences between our solution and individual IPFRR mechanisms, but an important contributor to those differences comes from our ability to exploit the flexibility afforded by centralized path computations to produce routing solutions that are difficult, if not impossible, to realize in the traditional, distributed environment assumed by most IPFRR solutions.

IPFRR's main goal is to ensure fast (sub-50ms) convergence of intra-domain routing protocols, as soon as failures have been detected. Current proposals fall in either one of two categories: those that can operate with an unmodified IP forwarding plane; and those that involve the use of a different (usually more complex) forwarding paradigm. The former category is the more relevant to this paper, which also seeks to offer protection to failure while preserving the simplicity and scalability of IP forwarding. In particular, one of our goals is to maximize the fast re-routing "coverage" achievable in any network by taking advantage of the flexibility of centralized routing in computing paths and controlling local forwarding decisions at each FE.

Examples of IPFRR mechanisms belonging to the first category include loop-free alternate (LFA) [1], O2 [24], [22], [21], DIV-R [20] and MARA [18]. The LFA proposal of [1] is aimed primarily at IP networks that run distributed, shortest-path-based routing algorithms. Furthermore, it relies on a criterion for ensuring loop freedom when selecting next-hop alternates (backups) (see [1, Inequality 1]) similar to the invariant of [20]. As alluded to earlier, imposing such a requirement prevents neighboring nodes from backing each other up (the criterion enforces an ordering among nodes, so that only one is eligible as a backup for the other). Both factors limit the coverage that the scheme is able to provide. This limitation is not present

[1] It is easy to construct network graphs for which no matter what routing is chosen, one or more nodes have no alternate next-hop.

[2] Forwarding reverts to the primary next-hop when the failure is fixed or an updated forwarding state is received.

in O2 [24], which is not restricted to using shortest paths and that introduces the concept of "joker" links specifically for the purpose of allowing mutual backups. These similarities make the O2 body of work [24], [22], [21] the most relevant to this paper, and it is, therefore, important to articulate differences in both scope and contributions.

O2 shares with this paper its applicability to (or more precisely, need for) a centralized routing system, and the goal of maximizing the number of nodes that are protected against any single link or node failure. In O2, this is realized by ensuring that every node has an "out-degree" (number of next-hops) of two - hence the name O2 - with one of them available as a backup in case of failures. This is similar to our goal as stated in Section I, with the difference that we do not seek to impose a limit of two on the out-degree, and will often allow more, especially when trying to reconcile the need for load-balancing with protectability. As a matter of fact, exploring the trade-off that exists between protectability and performance is an important difference between our work and O2. This difference is further reflected in the path computation algorithm we propose to jointly optimize protectability and performance. We demonstrate in Section VII the benefits of our algorithm in terms of both performance and protectability, when compared to O2 algorithms [21]. Another difference between our work and the O2 contributions is our focus on identifying specific conditions for the feasibility of a protection routing, and conversely the complexity of finding one when it exists. In particular, we formally establish in Section IV that the problem of computing a protection routing is NP-hard, and provide several characterizations of network topology that affect the feasibility of protection routing.

The DIV-R algorithm of [20] and the several MARA algorithms of [18] have similar goals as O2 and this paper, but differ in their approaches. DIV-R proposes a distributed algorithm to maximize a metric that reflects the number of next-hops available to each node. This may be effective against link failures, but as shown in Section VII, less so when considering node failures. The MARA algorithms consider several path computation problems aimed at improving minimum connectivity and fully utilizing all available links; hence affording greater resilience to failures (MARA's all-to-one maximum connectivity problem is the most relevant, and similar in spirit to DIV-R). As with DIV-R, protection against node failures is not explicitly taken into account and neither is the trade-off between performance and protectability.

The second category of IPFRR works includes [28], [10], [5], [26], [16], [14], which seek to deliver protectability irrespective of network topological limitations at the cost of possible changes to packet forwarding. In contrast, we want to avoid adding complexity to the data plane. Interface-specific forwarding tables that handle packet re-routing after failures while preventing loops are considered in [28]. Multiple "topologies" are used in [10], [5]; each covering different failures, with routers switching from one to another upon detecting a given failure and marking packets according to the topology to be used. In [26], protection is achieved by using tunnels to detour packets around failures; hence requiring packet encapsulation and decapsulation. [16] investigates the combination of LFA and tunneling to handle different failure scenarios. Finally, [14] proposes carrying root-cause failure information in packets to allow routers to diagnose problems and select alternate paths.

III. Model and Problem Formulation

We model the network as a directed graph $G = (V, E)$, with V the node set, E the link set, and $|V| = n$. A directed link from node i to node j is denoted by (i, j). $N_G(i) = \{j \in V | (i, j) \in E\}$ is the neighbor set of node i in G. As discussed in Section I, we assume that information such as network topology and link bandwidth is available to a central server for the purpose of path computation. We further assume that packet forwarding is destination-based without reliance on packet marking or encapsulation even in the presence of failures, *i.e.*, the standard IP forwarding paradigm.

For a destination[3] $d \in V$, let $R_d = (V, E_d)$ be a routing for traffic destined to d, where $E_d \subseteq E$. R_d is a directed acyclic graph (DAG) rooted at d and defines a destination-based routing. In R_d, every node $i \in V \setminus \{d\}$ has at least one outgoing link. A node j is called a primary next-hop (PNH) of node i if $(i, j) \in E_d$, and the link (i, j) is called a primary link of node i. If a node has multiple PNHs, traffic is split evenly across them. One advantage of centralized routing is that R_d's can be computed independently of each other. In contrast, a standard IGP such as OSPF computes routings that are coupled by a common set of link weights. Thus, without loss of generality, we focus on a single destination d.

When computing R_d, the goal is to maintain uninterrupted packet forwarding in the presence of any single "component" (link or node) failure, except for d itself. As discussed in Section I, this is vital in a centralized routing system to avoid extended forwarding interruptions after failures. Note that a two-node connected network is implicitly assumed, so that paths to d still exist after a failure[4]. Protection routing seeks to find them. Handling more severe failures may be feasible at the cost of additional complexity and a network connectivity high enough to avoid partitions.

Definition III.1 *After a single component failure f, the resulting network and routing for destination d are denoted by G^f and R_d^f, respectively. G^f and R_d^f are constructed by removing the failed component (node and/or link(s)) associated with f from G and R_d respectively.*

Definition III.2 *Node i is said to be upstream of node j in a routing R_d if there exists a path from node i to node j in R_d. Conversely, node j is then downstream of node i.*

Definition III.3 *In a routing R_d, node $i \neq d$ is said to be protected (with respect to d), if after any single component failure f that affects node i's PNH(s), there exists a node $k \in N_{G^f}(i)$ such that the following two conditions are satisfied:*

1) *Node k is not upstream of node i in R_d^f.*
2) *Node k and all its downstream nodes (except d) have at least one PNH in R_d^f.*

[3] For simplicity, we associate each node with a single destination, while in practice this would encompass all prefixes for which a node is the egress.

[4] As discussed in [19], this also ensures that the central server eventually learns about the failure, and can compute and download new forwarding states to FEs without creating routing loops.

Node k is called a secondary next-hop (SNH) of node i for f and d. By convention, destination d is always protected.

Definition III.3 is inspired by LFA but does not mandate the use of shortest paths, nor does it require [1, Inequality 1] to prevent loops. The two conditions of Definition III.3 imply that when the PNH of node i fails and packets are rerouted to node k: (i) routing loops never form [condition (1)]; and (ii) packets are delivered to d through node k and its downstream nodes in R_d^f [condition (2)]. Examples illustrating the feasibility or infeasibility of these conditions are provided in Section III-B.

A. Protection routing

Definition III.3 formalizes the notion of protectability for an *individual* node, against *single* failures that locally (at that node) affect forwarding decisions towards a *given destination*. This basic concept can be extended in a number of ways. It can be extended to all nodes for a given destination, and more generally to all nodes and all destinations. Specifically, we define the concepts of protection routing and protectable graphs as follows:

Definition III.4 *R_d is said to be a protection routing if every node $i \in V$ is protected in R_d.*

Definition III.5 *A graph $G = (V, E)$ is said to be protectable if a protection routing exists $\forall\, d \in V$.*

By Definition III.4, if R_d is a protection routing, packet forwarding (and delivery) can proceed uninterrupted from any node in the network to d in the presence of any single component failure (besides that of d itself). Similarly, if a graph is protectable according to Definition III.5, this guarantee extends to all nodes and all destinations. However, as we shall see in Section IV, such strong guarantees need not always be feasible and only a subset of nodes and/or destinations may be protectable. In such cases and given the traditional duplex nature of communications, it is useful to introduce the concept of duplex protection routing as follows:

Definition III.6 *A duplex protection routing with respect to destination d exists if the following conditions are satisfied:*

1) *There exists a protection routing R_d.*
2) *For every node $i \in V \setminus \{d\}$, there exists a routing R_i such that node d and all its downstream nodes toward i are protected.*

Definition III.6 states that if a duplex protection routing exists for d, then even in the presence of a single failure, packets originating from any node will be delivered to d [condition (1)] and conversely responses from d will be able to reach any node [condition (2)]. Note that the requirement of duplex protection is stronger than that of (simple) protection routing specified in Definition III.4, *i.e.*, it requires the existence of a protection routing for a given destination d, as well as "partial" protection routings for all nodes (the branch from d back to each individual node now serving as the destination). This more stringent requirement limits the number of routing solutions capable of realizing it. As we shall see in Section VII-C, this can affect the trade-off between protection and performance. However, it should be noted that when a graph is protectable according to Definitions III.5, protection routing and duplex protection routing are equivalent. In other words, in a protectable graph all nodes are protected and duplex protected for all destinations, and vice-versa.

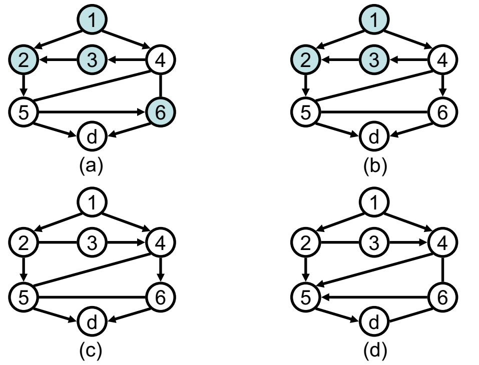


Fig. 1. Different routing choices (denoted by arrows) for destination d in a sample network. Shadowed nodes are unprotected.

In general, the main challenges are in *identifying* when such routings are feasible, and in *computing* them, or when not feasible, computing routings that maximize protectability (duplex or not). In particular, when a protectable graph is not feasible, the goal will be finding routings that for each destination maximize the number of (duplex) protected nodes[5]. We explore these computational challenges in Section IV, and in Sections V and VI we develop a heuristic to compute routings that satisfy different protectability and performance goals. Before proceeding with these investigations, we provide some illustrative examples that help anchor the discussion.

B. Discussion

Fig. 1 illustrates on a simple network topology how different routing choices affect protectability for a destination d. The routing of Fig. 1(a) does not protect nodes 1, 2, 3 and 6 under Definition III.3. For example, although node 1 has two PNHs, it is not protected against a failure of node 2. This is because its other PNH, node 4, is itself upstream of the failed node 2 (this violates condition (2) of Definition III.3). Similarly, node 6 is not protected against a failure of link $(6, d)$, as its two neighbors, nodes 4 and 5, are both upstream of itself (this violates condition (1) of Definition III.3). The routing of Fig. 1(b) succeeds in protecting node 6 against the failure of link $(6, d)$, because node 5 is now a valid SNH. However, according to condition (2) of Definition III.3, node 1 is still not protected against a failure of node 2, as even if node 4 now has a PNH (*i.e.*, node 6) that does not rely on node 2, it will still forward some packets destined to d towards node 3 (node 4 is unaware of the failure of node 2 and load-balances across its two PNHs) that remains unprotected. This last issue is resolved in Fig. 1(c), where all nodes are now protected. Note that to ensure protectability, more links are left unused during normal operations, so that they are available for mutual backups after failures. This illustrates the tension that exists between performance and protectability, and is one of the issues we explore further in Sections VI and VII.

[5] Definition III.6 can be readily generalized to cases where duplex protectability is available for only a subset of nodes, *i.e.*, an individual node i is duplex protected for a given destination d, if all its downstream neighbors towards d are protected, and if d and all its downstream neighbors towards i are protected.

Fig. 1(d) illustrates a subtle issue that arises from the choice of conditions in Definition III.3, and in particular condition (2) that calls for backup paths to only use PNHs. Fig. 1(d) gives an example of a routing for which node 2 is not protected (against a failure of node 5) according to Definition III.3, but that still ensures delivery of packets to d after a failure of node 5. This is because, when node 5 fails, node 2 forwards packets to its SNH, node 3, which passes them to its PNH, node 4. Node 4's PNH, however, was also node 5, so that node 4 knows that it must forward packets to its own SNH, node 6, which finds itself in a similar situation and forwards packets to its own SNH, namely, d. This ensures delivery of packets to d, but violates condition (2) of Definition III.3. The intuitive "fix" of relaxing condition (2) to allow packet forwarding using both PNH and SNH does unfortunately not work, as such a relaxation can introduce loops. In general, instances where backup paths such those of Fig. 1(d) improve protectability appear to be rare. Furthermore, systematically exploring them can add significant computational complexity, as all possible combinations of PNHs and SNHs need to be considered. Section V-D introduces a compromise based on an algorithm that iterates over possible SNH assignments once a choice of PNHs has been finalized, and allows the discovery of paths such as those of Fig. 1(d).

IV. Analysis

The goal of this section is to explore topological properties that are sufficient to ensure protectability, *i.e.*, determine if a graph is protectable, and characterize the algorithmic complexity of computing routings that realize it. We model a network as an undirected graph $G = (V, E)$, so that finding a protection routing is equivalent to identifying an orientation for a subset of links such that every node is protected, *i.e.*, an ordering among nodes that makes re-routing possible without creating loops. Graph terminology not defined in the paper can be found in [6]. All proofs are provided in appendices at the end of the paper, except for the proof of Theorem IV.5 that is only partially provided and available in full in [12].

In the next two sub-sections, we explore sufficient conditions for a graph to be protectable. Because, as mentioned earlier, protection routing and duplex protection routing are equivalent when a graph is protectable, we do not distinguish between them in those two sections. The third sub-section is devoted to the topic of computational complexity for which computing a protection routing and a duplex protection routing are considered separately.

A. Does a simple sufficient condition exist?

A necessary condition for a graph to be protectable is for every node to have two neighbors. Thus, it is natural to ask if the node degree of a graph can be used to characterize protectability. A simple sufficient condition for a graph to be protectable is as follows.

Theorem IV.1 *Every graph with $n \geq 5$ nodes and minimum degree at least $\lceil n/2 \rceil$ is protectable.*

Theorem IV.1 cannot be improved in that we cannot replace the bound of $\lceil n/2 \rceil$ with $\lfloor n/2 \rfloor$, as there exists a 1-node-connected graph[6] with minimum degree $\lfloor n/2 \rfloor$, which is obviously not protectable.

Theorem IV.1 implies that in the absence of any global graph property, a high minimum degree is needed to guarantee protectability. A natural next step is to explore if introducing global graph properties such as link- and node-connectedness can yield less stringent sufficient conditions for protectability. Intuitively, k-link-connectedness, for k large enough, would seem sufficient to ensure protectability. Surprisingly, this is not true in general, no matter how large k is. The result is summarized as follows.

Theorem IV.2 *For any given $k \in \mathbb{Z}^+$, there exists a k-link-connected graph that is unprotectable.*

Theorem IV.2 establishes that even with arbitrarily many link-disjoint paths, a protection routing is not guaranteed to exist. A similar question can be asked using the stronger condition of node-connectedness. In this setting, we only have the weaker result of Theorem IV.3 and a conjecture as follows.

Theorem IV.3 *For $k = 2, 3$, there exists a k-node-connected graph that is unprotectable.*

Conjecture IV.1 *For any given $k \geq 4$, there exists a k-node-connected graph that is unprotectable.*

The reason why the relatively strong properties of Theorems IV.2 and IV.3 fail to ensure protectability, is because destination-based routing induces an *ordering* among nodes; something that is not present when, for example, computing node-disjoint paths. Hence, even if each node *individually* has several disjoint paths to a destination, this need not hold when coupling them through a common destination-based routing.

B. On random graphs

The previous results showed that even graphs with very rich connectivity, *e.g.*, large degree or connectedness, were not guaranteed to be protectable. However, the proofs of these results involved graphs with very specific structure. A natural question is whether such graphs are the norm or the exception. To explore this question, we rely on a family of graphs with little or no special structure, *i.e.*, random graphs, and investigate what can be said about their protectability. We use Erdös-Rényi random graphs $G(n, p)$, where n denotes the number of nodes and p is the link probability (*e.g.*, see [4, Chapter 2]), and analyze under what conditions such graphs are protectable as n becomes large. This calls for finding routings such that for each destination all nodes have a suitable SNH to reroute traffic after failures. The random and relatively homogeneous structure of random graphs makes it possible to establish the following result.

Theorem IV.4 *Let $G \in G(n,p)$ and $p = (4 + \varepsilon)\log n/n$ where $\varepsilon > 0$ is any constant and $\log$ is the natural logarithm. Then asymptotically almost surely G is protectable.*

Theorem IV.4 implies that a mean degree that grows like $O(\log n)$ is sufficient to ensure that a random graph is protectable with probability tending to 1 as $n \to \infty$. In other words, in the absence of structure explicitly aimed at defeating it, the level of connectivity required to ensure protectability

[6] If n is odd, such a graph can be constructed by taking the union of two copies of complete graph $K^{\lceil n/2 \rceil}$ connected at one node.

is significantly lower than that required by Theorem IV.1. Although random graphs are not representative of all network topologies, this provides some hope that protectability is feasible in many practical networks with reasonable connectivity. The next section is devoted to assessing how difficult the task of computing such protection routings is.

C. NP-completeness of protection routing problems

This section analyzes the algorithmic complexity of computing protection and duplex protection routings for a given destination d. Note that while when a graph is protectable, protection routing and duplex protection routing are equivalent, simply comparing Definitions III.4 and III.6 clearly indicates that this is not so when focusing on a single destination d. As a result, both need to be investigated individually. For that purpose, we formulate the **PR problem** and the **DPR problem** as follows.

PR problem

- Instance: Given an undirected graph $G = (V, E)$ and a destination node $d \in V$.
- Question: Does a protection routing destined to d exist?

DPR problem

- Instance: Given an undirected graph $G = (V, E)$ and a destination node $d \in V$.
- Question: Does a duplex protection routing with respect to d exist?

Theorem IV.5 *The PR and the DPR problems are NP-complete.*

Theorem IV.5 indicates that in an arbitrary graph there is no known polynomial-time algorithm to solve the PR and the DPR problems unless P=NP. The proof is based on a reduction from the 3SAT problem, and the PR-specific version of the proof is in Appendix D while its DPR complement can be found in [12]. Given the computational complexity of the PR and DPR problems, heuristics are therefore required to compute protection routings.

V. Heuristic Design

Our goal is to compute n routings (one for each destination) to maximize protectability while realizing good network performance (*e.g.*, congestion) in normal (failure-free) situations. Computing a protection routing (PR) or duplex protection routing (DPR) for a destination is NP-hard because the decision versions of the problems are NP-complete. Because all n routings contribute to link loads, adding the dimension of performance introduces a coupling that only makes the problem harder. Practical solutions must, therefore, rely on heuristics. This section describes a heuristic for computing solutions to the PR and DPR problems introduced in the previous sections. Solutions that satisfy different definitions of protectability could also be computed by the heuristic, provided that their requirements can be mapped to an "appropriate" cost function (see below for additional details on the characteristics of appropriate cost functions).

A. Heuristic outline

The heuristic seeks to compute routings $R_d, \forall\, d \in V$, which achieve protectability while minimizing the resulting performance degradations in the absence of failures. In order to realize this goal while keeping computational complexity low, the heuristic is structured in two distinct phases.

Phase 1 proceeds to compute protection routings independently for each destination d. Its goal is to minimize a cost function Ω_d that measures "unprotectability" for each individual destination. Ω_d can in turn be chosen to reflect different protectability targets. For example, if a PR solution is needed, Ω_d can be set to measure the number of unprotected nodes, or a "weighted" version in case some nodes are deemed more important than others. If instead a DPR solution is the target, Ω_d can be set to measure the number of nodes that are either unprotected themselves or with at least one unprotected node in their downstream path to the destination[7]. More generally, pretty much any function can be used for Ω_d based on the specific protectability goals being targeted.

Phase 2 complements Phase 1 by jointly considering the routings it produced, *i.e.*, how together they affect network performance, and attempts to modify them to optimize performance without hurting protectability. Performance is measured through a cost function Φ, which for example can reflect network congestion. For illustration purposes, we select the function $\Phi = \sum_{l \in E} \Phi_l$ of [8], where Φ_l denotes the congestion cost of link l as a function of its load. Other expressions for Φ can be readily used. Pseudocode specifications of all the heuristics can be found in [12].

B. Phase 1 - Greedy search

Phase 1 uses a greedy search with a cost function $F_d, d \in V$, that focuses on Ω_d but remains congestion aware. Congestion is not explicitly accounted for in F_d to preserve independent computations across destinations. It is used to influence the greedy exploration of the solution space.

Specifically, prior to Phase 1, a standard traffic optimization routine, *e.g.*, [8], is run to assess the best network congestion cost Φ^{opt} in the absence of protectability considerations. This provides routings, $R_d^{opt}, d \in V$, that achieve Φ^{opt}, as well as a benchmark against which to compare network congestion costs under protection routing. Each routing R_d^{opt} can be computed using a shortest path algorithm with appropriate link weights. These link weights are used in Phase 1 to compute a deviation $||R_d - R_d^{opt}||$ between a proposed protection routing R_d and R_d^{opt}. This deviation is measured[8] using $\Gamma_d = \sum_{i \in V} \Gamma_{i,d}$, where $\Gamma_{i,d}$ denotes the distance from node i to destination d under R_d, with distances computed using the link weights of R_d^{opt}. The smaller Γ_d, the "closer" R_d is to R_d^{opt}. Thus, Φ achieved by R_d is also expectedly closer to Φ^{opt}. This metric guides the selection of solutions during Phase 1 as follows.

The cost function F_d is defined as $F_d = \langle \Omega_d, \Gamma_d \rangle$ where $\langle a_1, b_1 \rangle > \langle a_2, b_2 \rangle$ if and only if $a_1 > a_2$, or $a_1 = a_2$ and $b_1 > b_2$. This gives precedence to protectability, while favoring solutions with lower congestion costs (as measured through Γ_d) when it does not affect protectability. The optimization

[7] This accounts for the fact that when not all nodes are protectable, leaving a node close to the root (destination) unprotected affects duplex protectability of a greater number of nodes.

[8] Other measures can easily be accommodated.

carried out in Phase 1 is then of the form:

$$\forall\, d \in V \quad \underset{R_d}{\text{minimize}} \;\; F_d = \langle \Omega_d, \Gamma_d \rangle \;\; . \tag{1}$$

Note that although Γ_d in F_d accounts for congestion, computations for different destinations are still decoupled. This is because Γ_d is computed based on a fixed reference point (*i.e.*, the link weights that produced R_d^{opt}). This also avoids evaluating the cost function Φ for each candidate routing, an operation that in itself has a significant computational cost.

A "Greedy-search" heuristic is used to minimize Eq. (1). It was inspired by approximation algorithms for the 3SAT problem from which the NPC of the PR problem and the DPR problem is reduced, and operates on routings limited to trees (*i.e.*, each node except the destination has only one PNH). There are two motivations for the latter. First, assigning multiple PNHs to a node may affect the protectability of other nodes as discussed in Section III-B. Second, the sheer number of possible combinations involving multiple PNHs makes it computationally impractical to consider them all. Allowing multiple PNHs can reduce congestion through better load-balancing. This aspect is considered separately in Phase 2.

The heuristic starts with an initial routing $R_d = T\left(R_d^{opt}\right)$ obtained by extracting a tree from R_d^{opt} (when multiple next-hops are available, one is randomly selected). The main loop uses local feasibility checks to explore improvements in F_d when swapping the PNH of node $i \in V \setminus \{d\}$. This process repeats until F_d shows no improvement for all nodes. A diversification step is then executed, and generates a new random shortest path tree rooted at d. Unlike the first tree based on R_d^{opt}, the new tree is generated using random link weights uniformly selected in $[1, 1000]$. This ensures that after exploring the neighborhood of R_d^{opt}, the search restarts at a different point of the solution space[9]. The heuristic stops after P diversifications without improvement to F_d.

C. Phase 2 - Load-balancing

The routing trees R_d^*, $\forall d \in V$, of Phase 1 are used as inputs to Phase 2. Phase 2 seeks to assign multiple PNHs to nodes to better distribute traffic (load-balance), subject to the constraint that $\Omega_d, \forall d \in V$, cannot increase. Its procedure examines each node i in decreasing order[10] of its congestion[11], and tries to assign it multiple PNHs to better load-balance traffic and reduce its congestion cost. Note that Phase 2 involves evaluating Φ for each candidate routing, and this is where the bulk of its computational cost lies. The heuristic stops when Φ cannot be further reduced through new PNH assignments.

D. SNH assignment

The first two phases of the heuristic produce a set of routings that maximize protectability while minimizing congestion cost by load-balancing across multiple PNHs, as long as it does not affect protectability. By definition of protectability, all protected nodes have at least one SNH they can use in case of failure to forward packets on an alternate path that delivers packets to the destination solely through PNH forwarding. As discussed earlier, the restriction to PNH forwarding imposed by Definition III.3 precludes backup paths involving multiple SNHs, which could improve protectability. Allowing such paths, however, requires some care to avoid loops. Note that this step is important in practice, as unless packet marking is used, *i.e.*, to indicate prior SNH forwarding, independent forwarding decisions across nodes can readily result in multiple SNHs being used in the presence of single failures, *e.g.*, the case of Fig. 1(d). In this section, we describe an algorithm that assigns SNH (when a choice is available) to allow backup paths involving multiple SNHs, while ensuring the absence of loops. This is outlined for a given d with details in [12].

$R_d = (V, E_d)$ denotes the routing for d produced by Phases 1 and 2, where $E_d \subseteq E$ is the set of primary links. After failure f, $R_d^f = (V^f, E_d^f)$ denotes the residual routing after removing the failed component(s). Let $S_d^f : V \to V \cup \{\emptyset\}$ denote the SNH assignment mapping for failure f, with $S_d^f(i) \in V \cup \{\emptyset\}$ the SNH assigned to node i. An empty assignment, *i.e.*, $S_d^f(i) = \emptyset$, implies that there is either no need to assign an SNH to node i because its PNH is not affected by f, or no suitable SNH can be found. Our goal is to explore SNH assignments that maximize protectability when allowing backup paths that involve multiple SNHs.

Let $H_d^f = \left(V^f, E_d^f \bigcup_{i \in V, S_d^f(i) \neq \emptyset} \left(i, S_d^f(i)\right)\right)$ be a routing under failure f. Note that H_d^f combines R_d^f and S_d^f, and hence permits the use of multiple SNHs. This calls for additional precautions when assigning SNHs. Specifically, assume that node k is a candidate SNH for node i after failure f. Node k can be selected if the following two conditions are satisfied: (H1) H_d^f remains a DAG after the addition of link (i, k); (H2) Node k and all its downstream nodes (except d) have an out-degree of at least one in H_d^f. Condition (H1) ensures that loops are avoided, while Conditions (H1) and (H2) together guarantee packets delivery to d. Using these two conditions, SNHs can be assigned to improve protectability of nodes affected by failure f and with initially (after Phases 1 and 2) no feasible SNH, *i.e.*, $S_d^f(i) = \emptyset$. This continues until no SNH assignment satisfying conditions (H1) and (H2) is found. Note that the fact that PNHs remain fixed is in part what keeps computational complexity manageable.

VI. Trading Protectability for Performance

The cost function F_d gives strict precedence to protectability. A natural question is whether this can be relaxed to trade-off protectability for performance. Such a trade-off can be formulated using the following optimization:

$$\underset{R_d, d \in V}{\text{minimize}} \;\; \Phi \tag{2}$$

subject to

$$\Omega_d \leq (1 + \varepsilon_d)\, \Omega_d^* \qquad \forall d \in V \tag{3}$$

where Ω_d^* denotes the smallest possible value of Ω_d, and $\varepsilon_d \geq 0$ controls how much protectability can be traded-off for performance.

[9] Other diversification methods were tried, *e.g.*, shuffling a subset of PNHs in the tree, generating increasingly perturbed versions of R_d^{opt}, etc. The more diverse starting points of a random diversification consistently resulted in a better exploration of the solution space.

[10] Other orders, *e.g.*, random, fixed, were tried and found to perform worse.

[11] Defined by the sum of link costs at the node.

In realizing such a trade-off, computational complexity is again the main concern. Our proposed solution is based on two observations: (i) computing Ω_d^*, $\forall d \in V$, as required by Eq. (3), calls for performing Phase 1; and (ii) a large number of routings are examined during Phase 1. A natural option is to take advantage of the availability of those routings. Specifically, we keep *all* routings examined during Phase 1, and at the end of Phase 1 we identify those that satisfy Eq. (3). We then select for each destination d, the routing that minimizes Γ_d. Those routings can subsequently be further improved by invoking Phase 2.

This approach leverages the computational tractability of the previous heuristic (it has the same computational complexity, and avoids most expensive computations of the cost function Φ), and the additional memory it requires to store the routings examined during Phase 1 is relatively small. Intelligently discarding routings whenever they fail to satisfy Eq. (3) based on the current estimate of Ω_d^* can further reduce this memory.

VII. Evaluation

This section offers numerical evidences of the efficacy in computing protection routings and trading protectability for performance of the heuristic introduced in Sections V and VI.

The evaluation proceeds in three steps. The first step (Section VII-B) compares the proposed heuristic to earlier works with a similar goal of protectability. Because those works relied on a definition of protectability that is essentially that of the PR problem, the evaluation is carried out in that context rather than that of the DPR problem. Evaluating the efficacy of the heuristic for the DPR problem is the focus of the second step of the evaluation (Section VII-C). Finally, the third step (Section VII-D) evaluates the heuristic's ability to compute solutions that realize a desired trade-off between protectability and performance. Before presenting the results, we review the environment in which the evaluation is conducted.

A. Evaluation settings

1) Network topologies: A range of real and synthesized topologies are used to evaluate the heuristic as key metrics of network structure (*e.g.*, node degree, network size, etc.) vary.

- *RN*: Random topology of given average node degree.
- *PL*: Power-law topology based on the preferential attachment model [2].
- *AS*: Real topologies from the Rocketfuel project [23] and labeled by their AS numbers[12].

Link capacities are all set equal to unity with traffic demand (see below) used to generate heterogeneous load levels.

2) Traffic matrix: The traffic matrix $M = [r(s,t)]_{|V|\times|V|}$ is generated using a gravity model [11], [15], [3], [17] as follows: Traffic volume from node s to node t is defined as $r(s,t) = b_s \frac{e^{a_t}}{\sum_{i\in V\setminus\{s\}} e^{a_i}}$ where b_s is the total traffic originating at node s, and is uniformly distributed in [10, 50], [80, 130] and [150, 200] with probabilities 0.6, 0.35 and 0.05 respectively, a_t is the "mass" of node t which is proportional to the number of links it has and $\sum_{i\in V} a_i = 1$. The larger a node's mass, the more traffic it attracts. Using b_s, the model generates three different load levels. Finally, M is scaled to produce a reasonable link utilization in the network.

3) Heuristic setting: The heuristic involves only one parameter, P, used as the stopping criterion of Phase 1. We set $P = 10$, so that Phase 1 is stopped if there is no improvement after 10 diversification rounds. This value balances solution quality and computational time in our experiments.

4) Comparison: We use the proposed two-phase heuristic and the SNH assignment algorithm to compute protection routing solutions, and the results are denoted by PR or DPR. The solutions produced by the heuristic are then compared to the following previous approaches:

- *SP*: Routings computed by the OSPF optimization in [8].
- *DIVR*: Routings computed by DIV-R from [20].
- *O2*: Routings computed by the pattern-based algorithm[13] of [21].

We believe this provides a reasonable coverage of both the heuristic's performance across different networks, and its comparison to other alternatives. SP is commonly used for intra-domain routing in large ISP's, *e.g.*, [17], and focuses solely on performance. DIVR, like [18], seeks to maximize the number of PNHs at each node but without considering the use of SNHs after failures. O2 optimizes for protectability, but is oblivious to performance.

B. On the efficacy of computing protection routing solutions

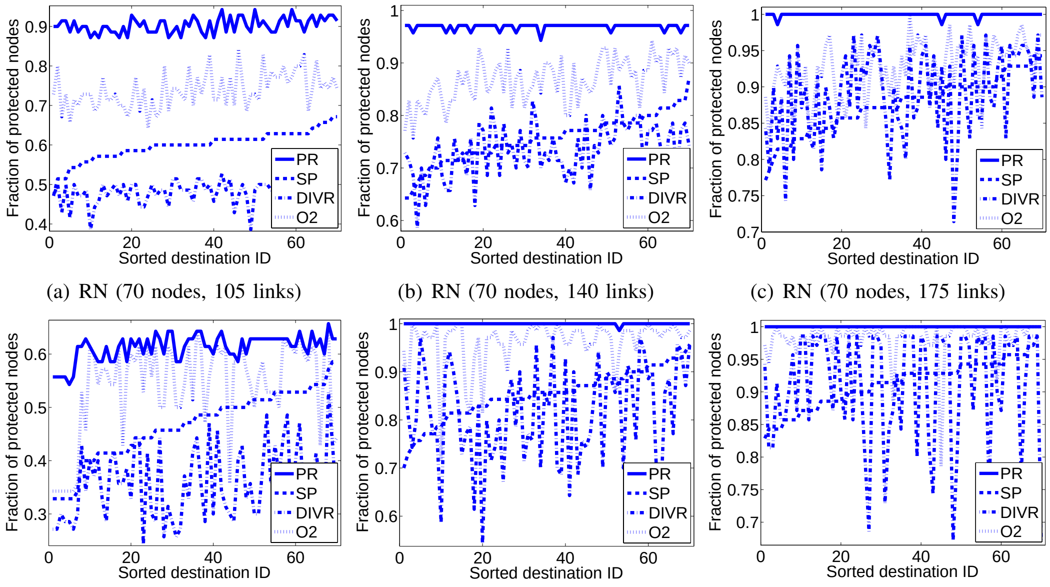
This section investigates the efficacy of the heuristic in computing protection routing solutions by comparing it to earlier alternatives with similar goals. To facilitate this comparison, we set Ω_d in the heuristic to be the number of unprotected nodes for destination d, which corresponds to the primary metric of interest in earlier approaches. Evaluations are carried out on both synthesized and real topologies (Figs. 2 and 3), with results first reported for synthesized topologies with mean degrees varying from three to five. The x-axis of Figs. 2 and 3 shows destination IDs sorted in ascending order of the number of protected nodes under SP, while the y-axis reports the number of protected nodes for each destination. The results illustrate that PR significantly improves protectability when compared to other solutions. Moreover, the results show that the gap is still present even in richly connected topologies for which, as indicated by Theorems IV.1 and IV.4, a protection routing is more likely to exist. Hence, even in those topologies, protection routings remain difficult to find unless an efficient heuristic is used.

It should be noted that the relatively poor performance of DIVR can, as mentioned earlier, be partly attributed to its focus on link failures that makes it more susceptible to node failures. Another finding from the figure is that a mean degree of 4 (*i.e.*, 70 nodes and 140 links) appears sufficient to realize near 100% protectability with PR. This indicates that protectability should be feasible in practice under reasonable connectivity.

The results of another set of evaluations carried out on real ISP topologies are shown in Fig. 3, where the mean degrees

[12] Nodes isolated from the giant component are removed.

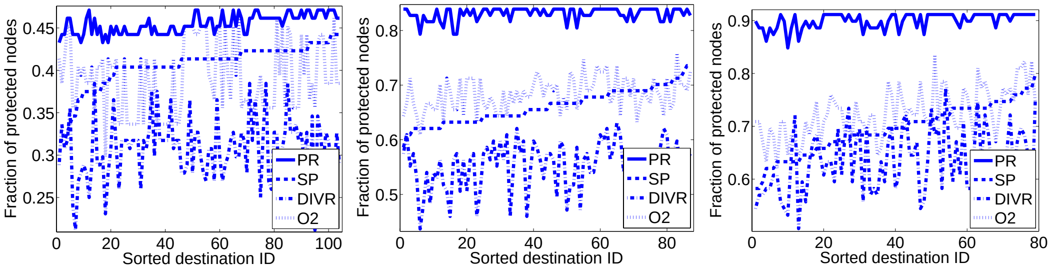
[13] This algorithm is chosen among several O2 heuristics, because, as reported in [21], it provides better protectability.



(a) RN (70 nodes, 105 links) (b) RN (70 nodes, 140 links) (c) RN (70 nodes, 175 links)

(d) PL (70 nodes, 105 links) (e) PL (70 nodes, 140 links) (f) PL (70 nodes, 175 links)

Fig. 2. Protectability across synthesized topologies.



(a) AS 1221 (104 nodes, 151 links) (b) AS 1755 (87 nodes, 161 links) (c) AS 3967 (79 nodes, 147 links)

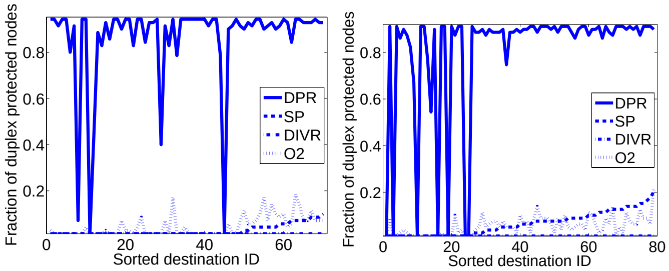
Fig. 3. Protectability across real topologies.

TABLE I
NETWORK PERFORMANCE ACROSS TOPOLOGIES.

Topology [# nodes, # links]	RN [70,105]	RN [70,140]	RN [70,175]	PL [70,105]	PL [70,140]	PL [70,175]	AS 1221	AS 1755	AS 3967
Average link load (PR)	0.20	0.31	0.22	0.24	0.21	0.29	0.11	0.26	0.15
Average link load (SP)	0.18	0.28	0.20	0.21	0.19	0.28	0.10	0.23	0.13
Average link load (DIVR)	0.22	0.44	0.36	0.25	0.28	0.43	0.12	0.31	0.18
Average link load (O2)	0.18	0.32	0.25	0.22	0.23	0.35	0.11	0.24	0.14
Maximum link load (PR)	0.86	0.66	0.55	0.66	0.53	0.74	0.93	0.98	0.91
Maximum link load (SP)	0.66	0.66	0.55	0.67	0.62	0.84	0.93	0.90	0.89
Maximum link load (DIVR)	0.90	1.36	1.23	1.00	1.56	2.92	1.13	1.57	1.17
Maximum link load (O2)	1.12	1.18	1.29	1.33	1.53	2.60	1.30	1.23	1.18
Increase in Φ under PR (%)	48.06	11.79	0.12	15.12	10.60	-3.98	3.73	19.72	32.51
Increase in Φ under DIVR (%)	55.05	4690	3284	91.86	13203	56690	492	3619	889
Increase in Φ under O2 (%)	495	978	1358	4455	11112	44917	3792	1246	1667

of AS 1221, AS 1755 and AS 3967 are 2.90, 3.70 and 3.72, respectively. The figure offers similar conclusions, namely, PR is effective in computing protection routings, and its advantage over other solutions remains even in richly connected ASes such as AS 3967.

Table I displays several performance metrics across topologies, where comparisons with SP reflect the cost of protectability. In the case of Φ, increases relative to SP are reported for PR, DIVR and O2, where a positive (negative) value denotes degradation (improvement). Under PR, network performance typically degrades slightly. This is expected because PR leaves some links unused under normal conditions to ensure they are available for protection after failures. This cost is, however, small, especially in comparison to that incurred by DIVR and O2, which often result in very high levels of congestion. This is in part because both are oblivious to performance when computing routings, and demonstrates that the heuristic is successful at reconciling performance and protectability. For completeness, we also report the increase in average path length (hop count) over SP across all experiments. The increases are 10.81%, 42.75%, and 14.10% under PR, DIVR, and O2, respectively. This illustrates that under PR, protectability has minimal impact on path length. Details about average path length in each experiment can be found in [12].



(a) RN (70 nodes, 140 links)

(b) AS 3967 (79 nodes, 147 links)

Fig. 4. Duplex protectability across synthesized and real topologies.

C. *Realizing duplex protection routing*

In this section, we extend the investigation to the computation of routing solutions that offer duplex protection. As mentioned earlier, most communications are duplex in nature, so that duplex protection is a natural metric. Furthermore, duplex protectability is important for locations such as data centers, server farms, and Internet gateways which attract a high volume of traffic and require high availability to other nodes in the network.

In order to compute a routing that offers duplex protectability to destination d, we set Ω_d in the heuristic to measure the number of nodes that are either unprotected or have at least one unprotected node in their downstream paths towards d. Note that this choice, as opposed to simply the number of unprotected nodes used in the previous section, gives preference to routings where unprotected nodes are towards the "leaves" of the routing DAG rooted at d. This captures a key aspect of duplex protectability, namely, the notion of end-to-end connectivity as opposed to only local protectability.

As before, the performance of the heuristic is compared to that of several other alternatives. For illustration purposes, we report the results for a random topology (70 nodes and 140 links) and AS 3967 in Fig. 4. The x-axis is again destination IDs sorted in ascending order of the corresponding number of nodes with duplex protectability under SP. Results obtained on other topologies were consistent with those of Fig. 4, which again illustrates the efficacy of the heuristic. As discussed in Section III-A, because of the more stringent requirement that duplex protectability imposes, the number of routing solutions that satisfy it is more limited. The impact this has is clear from comparing Fig. 4 with Fig. 2(b) and Fig. 3(c). Although the heuristic still vastly outperforms other approaches, the number of nodes for which duplex protectability can be guaranteed is lower than that for which (simple) protectability can be ensured. In particular, Fig. 4 identifies a handful of nodes (destinations) for which the heuristic performs poorly. These correspond to node locations for which ensuring duplex connectivity in the presence of (single) failures is essentially impossible. This has implications for service location, *i.e.*, it would be unwise to locate a data center there or even to make one of these nodes a major gateway to the Internet, and the output of the heuristic offers guidelines for such choices.

Another dimension in which the more stringent requirements of duplex protectability manifest themselves is in their impact on routing performance. Specifically, the smaller number of routings that satisfy duplex protectability means that optimizing for protection may force the choice of a poor performing routing. This is again illustrated by comparing the performance of (duplex) protection routing solutions of Fig. 4 with their counterparts of Section VII-B that were computed under the lesser requirements of (simple) protection routing. When compared to the performance of SP, the solutions of Fig. 4 delivered increases of 20.06% and 48.82% in the network cost Φ. This is to be compared to increases of only 11.79% and 32.51% for the solutions of Section VII-B. Nevertheless, we see that the heuristic still achieves much better performance than either DIVR or O2 (see Table I) that both target the lesser requirements of (simple) protectability. In the next section, we explore further the heuristic's ability to sacrifice some protectability to improve performance.

TABLE II
TRADE-OFF BETWEEN PROTECTABILITY AND PERFORMANCE.

ε	0	0.2	0.5	1	1.5	2
AS 3967 (79 nodes, 147 links)						
Decrease in Φ (%)	0	12.68	19.32	22.27	24.34	26.06
Increase in NDup (%)	0	3.33	9.34	54.08	82.30	120.36
NDup (in nodes)	15.16	15.67	16.58	23.36	27.64	33.41
Random topology (70 nodes, 140 links)						
Decrease in Φ (%)	0	4.53	6.55	9.21	9.68	9.86
Increase in NDup (%)	0	6.28	38.01	85.32	119.15	144.60
NDup (in nodes)	9.54	10.14	13.17	17.68	20.91	23.34

D. *Trading protectability for performance*

The previous section demonstrated that duplex protectability has a cost when it comes to network performance. Following the discussion of Section VI, we evaluate the heuristic's ability to produce solutions that trade some degradation in (duplex) protectability to improve performance. For simplicity, we assume that in Eq. (3), $\varepsilon_d = \varepsilon$, $\forall d \in V$.

Table II illustrates the trade-off between duplex protectability and performance for two topologies. The table uses results for $\varepsilon = 0$ (*i.e.*, no trade-off) as a benchmark for the decrease[14] in Φ realized by an increase in the number of non-duplex-protected nodes (denoted by "NDup" in the table) allowed by different values of ε's and averaged over all destinations.

The main observation from the results of Table II is that the proposed heuristic successfully trades-off a lower level of protectability for better performance. In addition, it is generally the case that the maximum gains are obtained "early on," *i.e.*, a slight initial decrease in protectability provides the biggest relative gains in performance. This demonstrates that the heuristic can provide network operators with a tunable solution for selecting a routing that provides the desired balance between protectability and performance. Furthermore, since the solution has essentially the same computational complexity as the base heuristic, it can be readily used in practice. Similar conclusions hold when seeking the same trade-off for (simple) protection routing. Results for this setting can be found in [13].

E. *Computational complexity*

Finally, to support our claim of computational efficiency, we report computational times under duplex protection routing for several large topologies. For the RN [70,175], PL [70,175],

[14] The relative decrease in Φ is at most 100%, which corresponds to $\Phi = 0$.

and AS 1221 topologies, run times were 1.99 hours, 1.64 hours, and 1.82 hours, respectively. None of the experiments required more than 300MB of memory. These results are obtained with a Pentium Xeon 2.66 GHz machine. Note that the computational times are realized without trying to explicitly take advantage of the inherent parallelism of the computations (the n routings of Phase 1 are independent and can be computed in parallel).

VIII. Conclusion

This paper has investigated the feasibility of protection routing in a centralized routing system, which displays heightened sensitivity to failures due to latency in responses from the central server. The paper identified topological properties that affect the feasibility of protection routing and established that computing protection routings is NP-hard. It developed an efficient heuristic to compute routings that not only optimize protectability, but also minimize its performance cost. The heuristic was shown to outperform earlier proposals, and its efficacy demonstrated for a range of topologies.

There are many directions in which this work can be extended. The first is to demonstrate the feasibility of protectability in a centralized routing system through an implementation. Another direction is to more systematically investigate the aspect of network design for protectability. Yet another area is to develop update mechanisms at the central server that are aware of which nodes have protection and which do not, and select update orderings based on this information and the need to avoid loops when updating forwarding states.

References

[1] A. Atlas and A. Zinin, “Basic specification for IP fast reroute: Loop-free alternates,” IETF RFC 5286, September 2008.

[2] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 509–512, October 1999.

[3] S. Bhattacharyya, N. Taft, J. Jetcheva, and C. Diot, “POP-level and access-link-level traffic dynamics in a Tier-1 POP,” in *Proc. ACM IMW*, 2001.

[4] B. Bollobas, *Random Graphs*, 2nd ed. Cambridge U. Press, 2001.

[5] T. Cicic, A. Hansen, A. Kvalbein, M. Hartmann, R. Martin, M. Menth, S. Gjessing, and O. Lysne, “Relaxed multiple routing configurations: IP fast reroute for single and correlated failures,” *IEEE Transactions on Network and Service Management*, vol. 6, no. 1, pp. 1–14, March 2009.

[6] R. Diestel, *Graph Theory*, 3rd ed. Springer, 2005.

[7] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, “The case for separating routing from routers,” in *Proc. ACM SIGCOMM FDNA workshop*, 2004.

[8] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing OSPF weights,” in *Proc. IEEE INFOCOM*, 2000.

[9] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, “A clean slate 4D approach to network control and management,” *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, October 2005.

[10] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, “Fast IP network recovery using multiple routing configurations,” in *Proc. IEEE INFOCOM*, 2006.

[11] K.-W. Kwong, R. Guérin, A. Shaikh, and S. Tao, “Improving service differentiation in IP networks through dual topology routing,” in *Proc. ACM CoNEXT*, 2007.

[12] K.-W. Kwong, L. Gao, R. Guérin, and Z.-L. Zhang, “On the feasibility and efficacy of protection routing in IP networks,” University of Pennsylvania, Tech. Rep., July 2009.

[13] ——, “On the feasibility and efficacy of protection routing in IP networks,” in *Proc. IEEE INFOCOM*, 2010.

[14] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, “Achieving convergence-free routing using failure-carrying packets,” in *Proc. ACM SIGCOMM*, 2007.

[15] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, “Traffic matrix estimation: Existing techniques and new directions,” in *Proc. ACM SIGCOMM*, 2002.

[16] M. Menth, M. Hartmann, R. Martin, T. Cicic, and A. Kvalbein, “Loop-free alternates and not-via addresses: A proper combination for IP fast reroute?” *Computer Networks*, vol. 54, no. 8, pp. 1300–1315, 2010.

[17] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot, “IGP link weight assignment for operational Tier-1 backbones,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 789–802, August 2007.

[18] Y. Ohara, S. Imahori, and R. V. Meter, “MARA: Maximum alternative routing algorithm,” in *Proc. IEEE INFOCOM*, 2009.

[19] H. Peterson, S. Sen, J. Chandrashekar, L. Gao, R. Guérin, and Z.-L. Zhang, “Message-efficient dissemination for loop-free centralized routing,” *ACM SIGCOMM Comp. Comm. Rev.*, vol. 38, no. 3, July 2008.

[20] S. Ray, R. Guérin, K.-W. Kwong, and R. Sofia, “Always acyclic distributed path computation,” *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 307–319, February 2009.

[21] C. Reichert, Y. Glickmann, and T. Magedanz, “Two routing algorithms for failure protection in IP networks,” in *Proc. ISCC*, 2005.

[22] C. Reichert and T. Magedanz, “Topology requirements for resilient IP networks,” in *Proc. 12th GI/ITG Conf. on Meas., Mod. & Eval. of Comp. & Comm. Sys*, 2004.

[23] Rocketfuel project. [Online]. Available: www.cs.washington.edu/research/networking/rocketfuel

[24] G. Schollmeier, J. Charzinski, A. Kirstädter, C. Reichert, K. Schrodi, Y. Glickman, and C. Winkler, “Improving the resilience in IP networks,” in *Proc. HPSR*, 2003.

[25] M. Shand and S. Bryant, “IP fast reroute framework,” Internet Draft, June 2009, (work in progress).

[26] M. Shand, S. Bryant, and S. Previdi, “IP fast reroute using Not-Via addresses,” Internet draft, July 2009, (work in progress).

[27] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai, “Tesseract: A 4D network control plane,” in *Proc. NSDI*, 2007.

[28] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, “Failure inferencing based fast rerouting for handling transient link and node failures,” in *Proc. IEEE Global Internet*, 2005.

Appendix A
Proof of Theorem IV.1

Let $n \geq 5$ be the number of nodes in a graph. First consider how to build a protection routing for a destination d. Since the graph has minimum degree at least $\lceil n/2 \rceil$, it has a Hamilton cycle [6, Theorem 10.1.1, p. 276]. For a given Hamilton cycle, the nodes are labeled clockwise, using $\{1, 2, \ldots, n-1\}$, starting from the node besides destination d. Note that destination d must have at least $\lceil n/2 \rceil - 2$ link(s) connected to some nodes other than nodes 1 and $n-1$. Suppose that one of the links from destination d lands on node $j \in [2, n-2]$. If $j \leq \lceil (n-1)/2 \rceil$, we build a 3-branch spanning tree rooted at node d as follows where the arrows indicate the orientation of the routing: (1) $j-1 \rightarrow j-2 \rightarrow \cdots \rightarrow 1 \rightarrow d$, (2) $\lceil (n-1)/2 \rceil \rightarrow \lceil (n-1)/2 \rceil - 1 \rightarrow \cdots \rightarrow j \rightarrow d$, and (3) $\lceil (n-1)/2 \rceil + 1 \rightarrow \lceil (n-1)/2 \rceil + 2 \rightarrow \cdots \rightarrow n-1 \rightarrow d$. If $j \geq \lceil (n-1)/2 \rceil + 1$, we build a 3-branch spanning tree rooted at node d as follows: (1) $j+1 \rightarrow j+2 \rightarrow \cdots \rightarrow n-1 \rightarrow d$, (2) $\lceil (n-1)/2 \rceil \rightarrow \lceil (n-1)/2 \rceil + 1 \rightarrow \cdots \rightarrow j \rightarrow d$, and (3) $\lceil (n-1)/2 \rceil - 1 \rightarrow \lceil (n-1)/2 \rceil - 2 \rightarrow \cdots \rightarrow 1 \rightarrow d$. Since each node has degree at least $\lceil n/2 \rceil$, it is easy to see that every node has at least one link connected to another tree branch based on the above construction, and hence the routing is protected. We can apply the same technique to find a protection routing for every destination, and as a result, the graph is protectable.

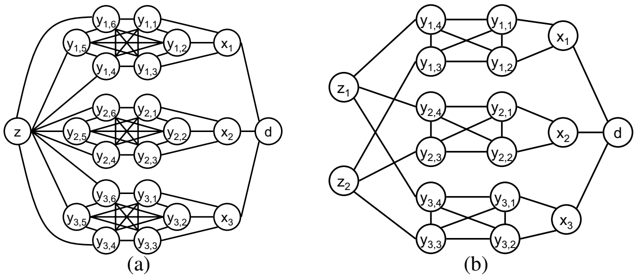


Fig. 5. Unprotected graphs: (a) 3-link-connected, (b) 3-node-connected.

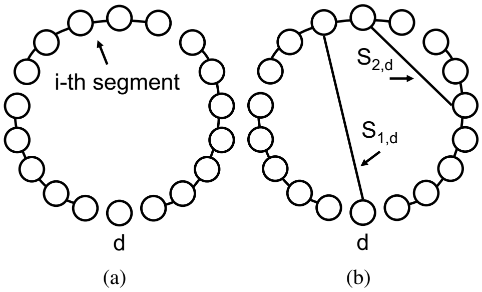


Fig. 6. Construction of a protection routing in a random graph.

Appendix B
Proof of Theorems IV.2 and IV.3

We first show that for any given $k \in \mathbb{Z}^+$, there exists a k-link-connected graph that is unprotectable. Let d be the destination. Let x_i, $i = 1, 2, \ldots, k$, be nodes directly connected to destination d. Let $G_i = (V_i, E_i)$ be a fully connected subgraph where $V_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,2k}\}$. k links are established from node x_i to nodes $y_{i,1}, y_{i,2}, \ldots, y_{i,k}$ for $i = 1, 2, \ldots, k$. Then node z is created so that it connects to nodes $y_{i,k+1}, y_{i,k+2}, \ldots, y_{i,2k}$ for $i = 1, 2, \ldots, k$. We use H_k to denote the final graph. The graph H_3 is shown as an example in Fig. 5(a). It is easy to see that H_k is k-link-connected and 2-node-connected. We can argue that H_k is not protectable as follows. Note that node z has to select at least one PNH. Without loss of generality, suppose that its PNH is a node in G_1. Since the graph is 2-node-connected and when the link (x_1, d) is failed, the traffic originated at node x_1 has to use one of the nodes $x_2, x_3, \ldots, x_k$ to reach the destination, and the traffic needs to go through node z. As a result, a loop is formed under the destination-based routing.

To prove that there exists a 3-node-connected graph that is unprotectable, we give a counter-example as shown in Fig. 5(b) where the destination is again denoted by d. Suppose that a protection routing for destination d exists. First, note that if link (x_1, d) is failed, a protection routing needs either node $y_{1,3}$'s PNH to be z_2 or node $y_{1,4}$'s PNH to be z_1. Similar constraints arise from the failure of links (x_2, d) and (x_3, d). Therefore, a protection routing requires 3 PNHs incoming to nodes z_1 and z_2. Additionally, nodes z_1 and z_2 totally need 2 outgoing PNHs and 2 SNHs in order to protect themselves. As a result, $3 + 4 = 7$ links are required from z_1 and z_2, but they only have 6 links. Hence, a contradiction.

Appendix C
Proof of Theorem IV.4

In the proof, we assume that nodes in a random graph in $G(n, p)$ are labeled $\{1, 2, \ldots, n\}$, and that the notation $a_n \sim b_n$ denotes $\lim_{n\to\infty} a_n/b_n = 1$. A property A of graphs in $G(n, p)$ is said to exist asymptotically almost surely (a.a.s.) if $\mathbb{P}(A) \to 1$ as $n \to \infty$. Unless otherwise specified, $\log$ denotes the natural logarithm.

The proof is based on a construction to find a protection routing for every destination in a random graph. The main difficulty in the proof is in constructing routings simultaneously for every destination such that each node is protected in the sense that they have a suitable SNH to reroute traffic when their PNHs are failed. Specifically, we need to ensure that the requirements imposed to ensure the existence of a protection routing to a given destination d in a graph in $G(n, p)$, do not overly constrain the structure of the graph so as to prevent the existence of protection routings to other destinations. To overcome this difficulty, we use a two-round exposure (*e.g.*, see [4]) to construct a protection routing for every destination. The two-round exposure means that if $G_1(n, p_1)$ and $G_2(n, p_2)$ are generated independently on the same node set, then $G_1(n, p_1) \cup G_2(n, p_2)$ is distributed as $G(n, p)$ where $p = p_1 + p_2 - p_1 p_2$, with any instance of multiple links between nodes in graphs in $G_1(n, p_1) \cup G_2(n, p_2)$ replaced by a single link. Based on this fact, our approach is to generate two independent, "suitable" random graphs in $G_1(n, p_1)$ and $G_2(n, p_2)$, such that a protection routing can be constructed.

Let $G_1 \in G_1(n, p_1)$ and $p_1 = (1 + \varepsilon_1) \log n/n$ where $\varepsilon_1 > 0$ is any constant. By a well-known result (*e.g.*, see [4]), a.a.s. G_1 has a Hamilton cycle. In the following, we let $p_2 = a \log n/n$ where $a > 0$ is some constant determined later, and generate an independent graph $G_2 \in G_2(n, p_2)$ such that a protection routing can be constructed based on a given Hamilton cycle in G_1.

Suppose that a Hamilton cycle exists in G_1. For a destination d, we divide the Hamilton cycle into $m \geq 2$ segments (excluding destination d) where the value of $m \in \mathbb{N}$ is determined later, and each i-th segment, $i = 1, 2, \ldots, m$, has g_i nodes where $[(n-1)/m] - 1 \leq g_i \leq [(n-1)/m] + 1$. An example is shown on Fig. 6(a). Next, we introduce two additional structures from G_2, which if present will allow the construction of a protection routing to destination d:

- $S_{1,d}$: At least one link connects destination d to a node in each segment.
- $S_{2,d}$: Each node in each segment has at least one link connected to a node in another segment.

Sample illustrations of $S_{1,d}$ and $S_{2,d}$ are shown in Fig. 6(b), and our goal is to demonstrate the existence of these two structures for all destinations.

If $S_{1,d}$ exists, we can construct an m-branching routing tree rooted at destination d. Furthermore, when $S_{2,d}$ exists, each node has a neighbor in another segment, which provides protection against failure. As a result, the existence of $S_{1,d}$ and $S_{2,d}$ implies that of a protection routing for destination d. In the remainder of the proof, we establish that a.a.s. this is the case.

Let $X_{i,d} = 1$ if destination d has no link connected to the previously constructed i-th segment of the Hamilton cycle, and $X_{i,d}=0$ otherwise. Let $X_d=\sum_{i=1}^{m} X_{i,d}$, and note that if $X_d=0$, then $S_{1,d}$ exists. Because $\mathbb{E}\left(X_{i,d}\right)=\mathbb{P}\left(X_{i,d}=1\right)$, hence

$$\begin{aligned}\mathbb{E}\left(X_d\right) &= \sum\nolimits_{i=1}^{m} \mathbb{E}\left(X_{i,d}\right)=\sum\nolimits_{i=1}^{m}\left(1-p_2\right)^{g_i} \\ &\sim m(1-p_2)^{n/m} \le m\exp\left(-p_2 n/m\right). \qquad (4)\end{aligned}$$

Let $Y_d=\sum_{i=1,i\neq d}^{n} Y_{i,d}$ where $Y_{i,d}=1$ if node $i\neq d$ has no link connected to a node in another segment, and $Y_{i,d}=0$ otherwise. If $Y_d=0$, then $S_{2,d}$ exists. First, note that the total number of nodes in any $m-1$ segments is at least $\{(n-1)/m-1\}\{m-1\}$, so the probability that a node $i\neq d$ has no link connected to a node in another segment is at most $(1-p_2)^{\{(n-1)/m-1\}\{m-1\}}$, and hence

$$\begin{aligned}\mathbb{E}\left(Y_d\right) &= \sum\nolimits_{\substack{i=1\\ i\neq d}}^{n} \mathbb{E}\left(Y_{i,d}\right) \le n(1-p_2)^{\{(n-1)/m-1\}\{m-1\}} \\ &\le n\exp\left\{-p_2\left(n-1\right)\left(1-1/m\right)+p_2 m\right\} \\ &\sim n\exp\left\{-p_2 n\left(1-1/m\right)\right\}. \qquad (5)\end{aligned}$$

Next, we investigate the existence of the constructions $S_{1,d}$ and $S_{2,d}$ to all destinations $d\in V$. In G_2, let $Z_d=1$ if either $S_{1,d}$ or $S_{2,d}$ or both does not hold, and $Z_d=0$ otherwise. Define $Z=\sum_{d=1}^{n} Z_d$, which counts the number of destinations without a protection routing based on the above construction. Note that by Markov inequality,

$$\mathbb{P}\left(Z_d=1\right)=\mathbb{P}\left(X_d+Y_d\geq 1\right)\leq \mathbb{E}\left(X_d\right)+\mathbb{E}\left(Y_d\right). \qquad (6)$$

Then, using Markov inequality with Eqs. (4) and (5), we have

$$\begin{aligned}\mathbb{P}\left(Z=0\right) &= 1-\mathbb{P}\left(Z\geq 1\right)\geq 1-\mathbb{E}\left(Z\right) \\ &= 1-n\mathbb{P}\left(Z_d=1\right)\geq 1-n\mathbb{E}\left(X_d\right)-n\mathbb{E}\left(Y_d\right) \\ &\geq 1-nm\exp\left(-p_2 n/m\right) \\ &\quad -n^2\exp\left\{-p_2 n\left(1-1/m\right)\right\}. \qquad (7)\end{aligned}$$

To ensure that every destination has a protection routing, *i.e.*, $\mathbb{P}\left(Z=0\right)\rightarrow 1$ as $n\rightarrow\infty$, Eq. (7) needs the following two conditions as $n\rightarrow\infty$:

$$\log n-p_2 n/m\rightarrow -\infty \quad\Rightarrow\quad a>m \qquad (8)$$

$$2\log n-p_2 n\left(1-1/m\right)\rightarrow -\infty \quad\Rightarrow\quad a>\frac{2}{1-1/m} \qquad (9)$$

Because the value of a should be kept as small as possible, we pick $m=3$ and hence $a=3+\varepsilon_2$, where $\varepsilon_2>0$ is any constant. As a result, G_2 should be generated using $p_2=(3+\varepsilon_2)\log n/n$.

Thus, using the two-round exposure, a.a.s. $G\left(n,p\right)=G_1\left(n,p_1\right)\cup G_2\left(n,p_2\right)$, where $p=(4+\varepsilon)\log n/n$ and $\varepsilon>0$ is any constant, has a protection routing for all n destinations.

Appendix D
Proof of Theorem IV.5 for the PR problem

This appendix establishes that the PR problem is NP-complete using a reduction from the 3SAT problem. As before, we use d to denote the destination. First, note that one can verify whether a given routing destined to node d is protected in a polynomial time. Thus, the PR problem is in NP.

Assume the 3SAT problem consists of n boolean variables, $u_1,u_2,\ldots,u_n$, whose complements are denoted by $\bar{u}_1,\bar{u}_2,\ldots,\bar{u}_n$ respectively. Let $U=\{u_1,\bar{u}_1,\ldots,u_n,\bar{u}_n\}$. The boolean equation of the 3SAT problem consists of m clauses, *i.e.*, $B=C_1C_2\cdots C_m$ where $C_j=x_j+y_j+z_j$ and $x_j,y_j,z_j\in U$ for all $j=1,2,\ldots,m$. We build a graph G' to represent B, and show that the 3SAT problem has a solution if and only if G' has a protection routing destined to node d. The construction of G' involves a combination of *squares, switches* and *connectors*, which are shown in Fig. 7(a) and detailed below.

Each clause $C_j, j=1,2,\ldots,m$, is associated with what we term a C_j-square that consists of a four-node mesh topology (see Fig. 7(a)), where each node has one outgoing link. One of those links is connected to destination d, and the other three links are associated with one of the three boolean variables x_j, y_j and z_j of clause C_j.

Observation D.1 *A C_j-square is protected if it has at least two outgoing PNHs.*

The function of a C_j-square is to capture whether clause C_j is true or not. When clause C_j is true, at least one of its boolean variables must be true. This is represented by having the corresponding link, *i.e.*, the link with associated variable x_j, y_j or z_j, assigned to be an outgoing PNH. In other words, the boolean variable is true if and only if the link it has been assigned to is an outgoing PNH.

The next building block is a graph component called a u_i-switch, $i=1,2,\ldots,n$. The structure of a u_i-switch is shown in Fig. 7(a), where links a and b are outgoing. The function of the u_i-switch is to represent the boolean variable u_i.

Observation D.2 *To make a u_i-switch protected, the routing must result in a configuration where at least one of links a or b is an outgoing PNH.*

The key feature of a u_i-switch, as highlighted in Observation D.2, is that it can be used to couple the boolean assignment of u_i and the routing configuration.

Next we introduce the connector building block, structured as shown in Fig. 7(a). It has three outgoing links e, f and g. The connector will be used to constrain the direction of a protection routing, which will then be used in the NPC reduction. The approach for realizing this goal is based on the following two observations.

Observation D.3 *If link e is an incoming PNH into the connector, then any protection routing for the connector cannot also have link f as another incoming PNH into the connector and must have link g as an outgoing PNH from the connector.*

Observation D.4 *If link f is an incoming PNH into the connector, then we can find a routing to protect the connector such that link e is an outgoing PNH from the connector and link g is not necessarily used.*

From Observations D.3 and D.4, we see that links e and f can be used to control the direction of a protection routing (PNH assignment into and from the connector) for the connector.

We are now ready to proceed with the construction of G' using squares, switches and connectors so as to realize the desired NPC reduction. The idea of the construction is to impose some constraints on the assignments of the outgoing PNHs of the squares, and hence the reduction can be done by examining those resulting assignments. Each C_j clause is associated with a C_j-square with three links corresponding to the variables x_j, y_j and z_j, as discussed earlier, and each of these links is connected to a connector. Fig. 7(b) shows how

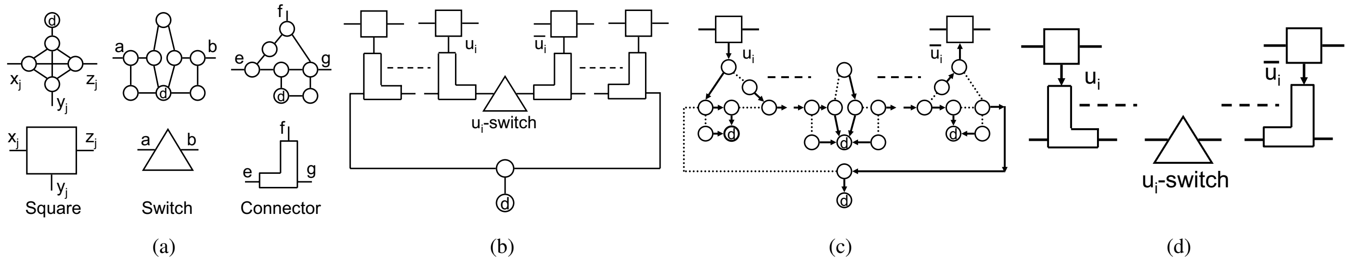


Fig. 7. (a) Structures of square, switch and connector, and their schematic representations. (b) An example of a graph formed by squares, switch and connectors for boolean variables u_i and $\bar{u}_i$. (c) An example of a routing configuration if u_i is true. (d) An impossible configuration under u_i-switch.

to connect the building blocks together for boolean variables u_i and $\bar{u}_i$. In the figure, all the connectors on the left (resp. right) hand side are connected to the squares with boolean variable u_i (resp. $\bar{u}_i$), and the connectors themselves are connected in serial. The u_i-switch is in the middle to connect the two arrays of connectors on both sides. This construction repeats for all boolean variables. Therefore, given the 3SAT problem, the corresponding G' can be constructed accordingly in polynomial time.

Suppose that a 3SAT solution exists, *i.e.*, each clause must have at least one boolean variable with a value of true. Next, we find a protection routing using the following method. In each C_j-square, we assign the link corresponding to the true boolean variable as an outgoing PNH. If the clause only has one true boolean variable, we can assign the link connected to destination d as an outgoing PNH. As a result, each C_j-square has at least two outgoing PNHs. An example is given below.

Consider u_i-switch, $i = 1, 2, \ldots, n$, and suppose that u_i is true. We assign the routing as shown in Fig. 7(c) (by Observations D.2, D.3, and D.4) where the arrows indicate PNHs and the dotted lines denote SNHs. A similar construction can be used if $\bar{u}_i$ is true (*i.e.*, u_i is false). In this case, the orientation of the routing is from right to left as opposed to that of Fig. 7(c). Hence, a protection routing can be found.

Now suppose that a protection routing exists on G'. First, we have the following observation.

Observation D.5 *If a protection routing exists, the situation shown in Fig. 7(d) can not happen. In other words, it is impossible that links u_i and $\bar{u}_i$ from their squares are incoming PNHs to their connectors.*

Observation D.5 can be explained as follows. If a protection routing exists, the switch must have at least one outgoing PNH (by Observation D.2) which points to link e of a connector. By Observation D.3, link f of that connector can not be an incoming PNH from a square, and link g of that connector must be an outgoing PNH. Since the connectors are connected in serial, the situation shown in Fig. 7(d) can not happen.

Given a protection routing, we use the following method to find a 3SAT solution. We look at each C_j-square, $j = 1, 2, \ldots, m$, and note that there must exist an outgoing PNH via link x_j, y_j or z_j due to Observation D.1. For example, if link x_j is an outgoing PNH, then we set x_j to be true. By Observation D.5, x_j can be assigned true or false consistently. After looking at all C_j-squares, if there are any remaining, unassigned boolean variables, they can be assigned true or false arbitrarily. As a result, each clause is satisfied and hence $B = C_1 C_2 \cdots C_m$ is satisfied.

Kin-Wah Kwong received the B.E. degree (First Class Honors) and the M.Phil. degree in Electronic Engineering from the Hong Kong University of Science and Technology, in 2003 and 2005, respectively. He also received the Ph.D. degree in Electrical and Systems Engineering from the University of Pennsylvania, Philadelphia, in 2010. His current research interests are Internet routing, network economics, social networks, and peer-to-peer systems. He was co-recipient of the IEEE INFOCOM 2010 Best Paper Award.

Lixin Gao (F'10) is a professor of Electrical and Computer Engineering at the University of Massachusetts at Amherst. She received her Ph.D. degree in computer science from the University of Massachusetts at Amherst in 1996. Her research interests include multimedia networking, and Internet routing and security. Between May 1999 and January 2000, she was a visiting researcher at AT&T Research Labs and DIMACS. She is an Alfred P. Sloan Fellow and received an NSF CAREER Award in 1999. She has served on number of technical program committees including SIGCOMM2006, SIGCOMM2004, SIGMETRICS2003, and INFOCOM2004, and was on the Editorial Board of the IEEE/ACM Transactions on Networking.

Roch Guerin (F IEEE '01 /F ACM '06) received an engineer degree from ENST, Paris, France, and M.S. and Ph.D. degrees in EE from Caltech. He joined the Electrical and Systems Engineering department of the University of Pennsylvania in 1998, as the Alfred Fitler Moore Professor of Telecommunications Networks. Prior to that he was with the IBM T. J. Watson Research Center. From 2001 to 2004 he was on partial leave from Penn, starting Ipsum Networks, a company that pioneered the concept of protocol participation in managing IP networks. His research has been in the broad area of networked systems.

Dr. Guerin has been an editor for many ACM and IEEE publications, and is currently the Editor-in-Chief for the IEEE/ACM Transactions on Networking. He served as General Chair of the IEEE INFOCOM '98 conference, as Technical Program co-Chair of the ACM SIGCOMM '01 conference, as General Chair of the ACM SIGCOMM '05 conference, and as Technical Program co-Chair of the ACM CoNEXT '07 conference. In 1994 he received an IBM Outstanding Innovation Award for his work on traffic management and the concept of equivalent bandwidth. In 2010 he received the IEEE INFOCOM Achievement Award for "Pioneering Contributions to the Theory and Practice of QoS in Networks." He was on the Technical Advisory Board of France Telecom from 2001 to 2006, and on that of Samsung Electronics in 2003-2004. He has been on Simula's Scientific Advisory Board since 2010.

Zhi-Li Zhang received a B.S. degree in Computer Science from Nanjing University, China, in 1986 and his M.S. and Ph.D. degrees in computer science from the University of Massachusetts in 1992 and 1997. In 1997 he joined the Computer Science and Engineering faculty at the University of Minnesota, where he is currently a professor. From 1987 to 1990, he conducted research in Computer Science Department at Arhus University, Denmark, under a fellowship from the Chinese National Committee for Education. He has held visiting positions at Sprint Advanced Technology Labs, IBM T.J. Watson Research Center, Fujitsu Labs of America, Microsoft Research China, and INRIA, Sophia-Antipolis, France.