

MIT Open Access Articles

Providing protection in multi-hop wireless networks

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: Kuperman, Greg, and Eytan Modiano. "Providing protection in multi-hop wireless networks." In 2013 Proceedings IEEE INFOCOM, 926-934. Institute of Electrical and Electronics Engineers, 2013.

As Published: <http://dx.doi.org/10.1109/INFOCOM.2013.6566881>

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Persistent URL: <http://hdl.handle.net/1721.1/81449>

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike 3.0



Providing Protection in Multi-Hop Wireless Networks

Greg Kuperman
MIT LIDS
Cambridge, MA 02139
gregk@mit.edu

Eytan Modiano
MIT LIDS
Cambridge, MA 02139
modiano@mit.edu

Abstract—We consider the problem of providing protection against failures in wireless networks subject to interference constraints. Typically, protection in wired networks is provided through the provisioning of backup paths. This approach has not been previously considered in the wireless setting due to the prohibitive cost of backup capacity. However, we show that in the presence of interference, protection can often be provided with no loss in throughput. This is due to the fact that after a failure, links that previously interfered with the failed link can be activated, thus leading to a “recapturing” of some of the lost capacity.

We provide both an ILP formulation for the optimal solution, as well as algorithms that perform close to optimal. More importantly, we show that providing protection in a wireless network uses as much as 72% less protection resources as compared to similar protection schemes designed for wired networks, and that in many cases, no additional resources for protection are needed.

I. INTRODUCTION

Multi-hop wireless mesh networks have become increasingly ubiquitous, with wide-ranging applications from military to sensor networks. As these networks continue gaining in prominence, there is an increasing need to provide protection against node and link failures. In particular, wireless mesh networks have recently emerged as a promising solution for providing Internet access. Since these networks will be tightly coupled with the wired Internet to provide Internet services to end-users, they must be equally reliable. Wired networks have long provided pre-planned backup paths, which offer rapid and guaranteed recovery from failures. These protection techniques cannot be directly applied to wireless networks due to interference constraints. As opposed to wired networks, two wireless nodes in close proximity will interfere with one another if they transmit simultaneously in the same frequency channel. So, in addition to finding a backup route, a schedule of link transmissions needs to be specified. In this work, we consider the problem of providing guaranteed protection in wireless networks with interference constraints via pre-planned backup routes, as well as their corresponding link transmission schedules.

Guaranteed protection schemes for wired networks have been studied extensively [1–5], with the most common scheme being

This work was supported by NSF grants CNS-1116209 and CNS-0830961, by DTRA grant HDTRA-09-1-005, and by the Department of the Air Force under Air Force contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

1 + 1 guaranteed path protection [5]. The 1 + 1 protection scheme provides an edge-disjoint backup path for each working path, and guarantees the full demand to be available at all times after any single link failure. Protection schemes optimized for wireless networks with interference constraints have not yet been considered. Typically, an approach for resiliency in wireless networks (in particular sensor networks) is to ensure that there exists “coverage” for all nodes given some set of link failures [6, 7]. This approach to resiliency does not consider routing and scheduling with respect to interference constraints, and assumes that there exists some mechanism to find a route and schedule at any given point in time. Furthermore, there is no guarantee that sufficient capacity will be available to protect against a failure. The idea of applying 1 + 1 protection in wireless networks is briefly mentioned in [8]. However, [8] does not study the specific technical details of such an approach to wireless protection. The goal of this chapter is to study protection mechanisms for wireless networks with a particular focus on the impact of wireless interference and the need for scheduling.

The addition of interference constraints makes the protection problem in a wireless setting fundamentally different from the ones found in a wired context. After a failure in a wireless network, links that could not have been used due to interference with the failed link become available, and can be used to recover from the failure. In fact, it is often possible to add protection in a wireless setting without using any additional resources.

Consider allocating a protection route for the following example, shown in Fig. 1. The wireless network operates in a time-slotted fashion, with equal length time slots available for transmission. Any two nodes within transmission range have a link between them, and each link’s time slot assignment is shown in the figures. We assume a 1-hop interference model where any two links that have a node in common cannot be active at the same time. Additionally, we assume unit capacity links. Before any failure, the maximum flow from s to d is 1, and can be achieved using a two time slot schedule, as shown in Fig. 1a. At any given point in time, only one outgoing link from s can be active, and similarly, only one incoming link to d can be active. Wireless links $\{s, c\}$, and $\{c, d\}$ cannot be used prior to the failure of $\{s, b\}$, but become available after $\{s, b\}$ fails. After the failure of $\{s, b\}$, flow can be routed from s to c during

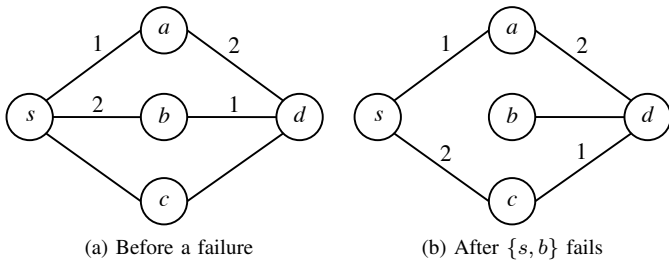


Fig. 1: Time slot assignment for protection in a wireless network

time slot 2, and from c to d during slot 1, as shown in Fig. 1b. Similar schedules can be found for failures of the other links. The maximum flow from s to d is 1 for both before *and* after a failure; i.e., there is no reduction in maximum throughput when allocating resources for a protection route on $\{s, c\}$ and $\{c, d\}$: protection can be assigned for “free”. This is in contrast to a wired network where the maximum throughput without protection from s to d is 3, and the maximum throughput when assigning a protection route on $\{s, c\}$ and $\{c, d\}$ is 2, which amounts to a $\frac{1}{3}$ loss in throughput due to protection.

The novel contributions of this chapter is introducing the Wireless Guaranteed Protection (WGP) problem in multi-hop networks with interference constraints. In Section II, the model for WGP is presented. In Section III, properties of an optimal solution are examined for a single demand with 1-hop interference constraints, which are then used to motivate the development of a time-efficient algorithm. In Section IV, an optimal solution is developed via a mixed integer linear program for general interference constraints. In Section V, time-efficient algorithms are developed that perform within 4.5% of the optimal solution.

II. MODEL AND PROBLEM DESCRIPTION

In this chapter, solutions to the guaranteed protection problem for multi-hop wireless networks subject to interference constraints are developed and analyzed. Our goal is to provide protection in a manner similar to what has been done in the wired setting. Namely, after the failure of some network element, all connections must maintain the same level of flow that they had before the failure. In order to do so, resources are allocated and scheduled in advance on alternate (backup) routes to protect against failures.

In wired networks, two adjacent nodes can transmit simultaneously because they do not interfere with one another; if capacity exists on a set of links, a path can be routed using that capacity. Wireless networks are different; interference constraints must be considered. A set of links in close proximity cannot transmit simultaneously on the same frequency channel; only one link from that set can be active at a time, or else they will interfere with one another. Not only must a path between the source and destination be found with available capacity, but also a schedule of link transmissions needs to be determined. This is known as the routing and scheduling problem [8–16],

which is known to be NP-Hard [9].

The addition of interference constraints adds complexity to the traditional wired protection problem, but also presents an opportunity to gain protection from failures with minimal loss of throughput. After a failure in a wireless network, links that could not have been used due to interference with the failed link become available, and can be used to recover from the failure. In fact, it is often possible to add protection in a wireless setting without any loss in throughput.

The following network model is used for the remainder of the chapter. A graph G has a set of vertices V and edges E . An interference matrix \mathcal{I} is given, where $I_{ij}^{kl} \in \mathcal{I}$ is 1 if links $\{i, j\}$ and $\{k, l\}$ can be activated simultaneously (do not interfere with each other), and 0 otherwise. The interference matrix is agnostic to the interference model used (i.e., it can be used to represent nearly any type of link interferences). For the remainder of this work, we focus on the 1-hop interference model (any two links that share a node cannot be activated simultaneously), but our schemes can be adapted to the K -hop [17] interference model as well. Our goal in this chapter is to develop a framework for routing and scheduling with protection under interference constraints. We assume nodes are fixed, links are bidirectional, and that the network uses a synchronous time slotted system, with equal length time slots; the set of time slots used is \mathcal{T} . Only link failures are considered, and a single-link failure model is assumed; it is straightforward to apply the solutions developed in this chapter to node failures as well. For now, we assume centralized control; the algorithms presented can be modified to work in a distributed fashion, as done in [18]. Additionally, we only consider a single frequency channel.

III. EFFICIENT ALGORITHM FOR A SINGLE DEMAND

In this section, we aim to achieve insight into providing protection for wireless networks with interference constraints by examining the solution for a single demand under a basic set of network parameters: 1-hop interference constraints and unit capacity links. In Section III-A, properties of an optimal solution are examined for routing and scheduling with and without protection. In Section III-B, a time efficient algorithm is developed that finds a maximum throughput guaranteed to be within 1.5 of the optimal solution. In Appendix B, a polynomial time algorithm is presented that tightens this bound and finds a solution guaranteed to be within $\frac{6}{5}$ of the optimal solution.

A. Solution Properties

In this section, properties of an optimal solution for WGP for a single demand are examined. First, we look at routing and scheduling without protection, and then those results are extended to the protection setting.

Lemma 1. *The maximum flow that can be routed and scheduled between the source s and destination d under 1-hop interference constraints without protection is 1.*

Proof: Under 1-hop interference constraints, only one link exiting the source node can be active at time. Since each link

has unit capacity, the maximum flow that can leave the source (or enter the destination) is 1. ■

While Lemma 1 indicates that a flow of 1 is possible, it does not necessarily mean that a flow of 1 can be achieved. We note that if the source s and destination d are adjacent, then a maximum flow of 1 can always be achieved by using one edge between the two. We assume for the remainder of this section that s and d are at least two hops apart. We now give the properties of maximum flows for a single demand in a unit-capacity wireless network under 1-hop interference constraints.

Lemma 2. *To achieve the maximum flow of 1, there must exist at least two node-disjoint paths from s to d .*

Proof: Assume otherwise: there are no node-disjoint paths, and there is a maximum flow of 1 possible from s to d . If there are no node-disjoint paths between s and d , then by Menger's theorem there exists a single node j whose removal will separate s and d [19]; hence, all paths from s to d must pass through j . In order for a flow of 1 to exist between s and d , node j must have a total of 1 unit of flow coming in, and 1 unit of flow going out. This is only possible if node j is both receiving and transmitting the entire time, which is not possible under the 1-hop interference model since node j cannot be both receiving and transmitting simultaneously. ■

Corollary 1. *If two node-disjoint paths from s to d do not exist, then the maximum flow is $\frac{1}{2}$.*

Proof: In the proof for Lemma 2, it was shown that if no node-disjoint paths exist between s and d , all paths must cross some node j . Node j cannot be receiving and transmitting at the same time under the 1-hop interference model. The maximum flow that j can support is to be receiving half of the time, and transmitting the other half. Increasing the amount of time j is transmitting will reduce the amount of time it can receive, which reduces the overall flow. The same is seen if the amount of time j is receiving is increased. Hence, j will have incoming flow half of the time, and outgoing flow for the other half. Since each edge has unit capacity, the maximum flow possible through node j is $\frac{1}{2}$. ■

Any loop-free path from s to d (when s and d are greater than one hop apart from one another) can have an interference-free schedule by alternating between time slots 1 and 2 for each edge of the path; hence, any edge of the path will only be active for half of the time, and the path will support a flow of $\frac{1}{2}$. If two or more node-disjoint paths do exist, then a maximum flow is dependent on the total number of edges in the disjoint paths.

Lemma 3. *If there exists two node-disjoint paths between s and d with an even total number of edges over both paths, then the maximum flow of 1 is achievable.*

Proof: When both paths have an even total number of edges, an interference-free schedule using two time slots is possible by alternating time slot assignments on each path. If each path has an even number of edges, then path 1 will begin with time slot 1 and end with time slot 2, and path 2 will begin

with time slot 2 and end with time slot 1. Each unit-capacity edge will be active for half of the time; hence, each path carries a total of $\frac{1}{2}$ unit of flow, giving the maximum flow of 1 using both paths. A similar schedule can be shown for the case when each path has an odd number of edges. ■

To help see Lemma 3, two examples are shown in Fig. 2, with the time slot assignments for the links labeled in the figures. In Fig. 2a and 2b, there are two node-disjoint paths from the source s to destination d that have an even total number of edges. In Fig. 2a, each path has an even number of edges, and in Fig. 2b, each path has an odd number of edges. An interference-free schedule for the two paths can be found using two time slots. Each link is active for $\frac{1}{2}$ of the time; hence, each path can support a flow of $\frac{1}{2}$, giving a total flow of 1 from s to d .

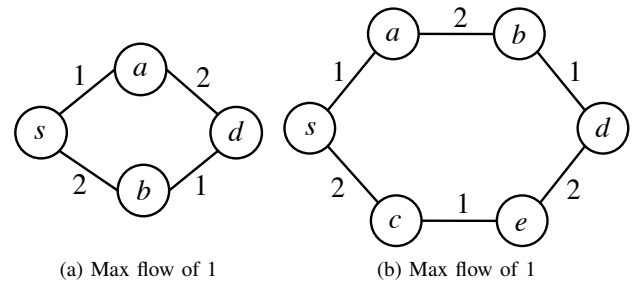


Fig. 2: Node-disjoint paths with an even total number of edges

Corollary 2. *If there exists more than two node-disjoint paths between s and d , a maximum flow of 1 is always achievable.*

Proof: If there are more than two node-disjoint paths, then there always exists a pair of node-disjoint paths that has an even total number of edges. ■

If the total number of edges in the two node-disjoint paths is odd, then the two-time slot schedule used in Fig. 2 to achieve the maximum flow of 1 over the two paths is not possible; additional time slots are needed. While a maximum flow of 1 may not be possible, a minimum flow of $\frac{2}{3}$ is always feasible over two node-disjoint paths if a third time slot is used.

Lemma 4. *For a pair of node-disjoint paths that have an odd total number of edges, a flow of $\frac{2}{3}$ is always possible between s and d using three time slots.*

Proof: Remove one of the edges from one of the paths; there is now an even number of edges in the two paths. Schedule the two paths using two time slots as if they were a pair of node-disjoint paths with an even number of edges. After this step, all of the scheduled edges can operate without interference. Now reintroduce the removed edge. This reintroduced edge is adjacent to two edges that each have a time slot assignment of 1 and 2, respectively. Clearly, time slot 1 or 2 cannot be assigned to the reintroduced edge, so assign it time slot 3. With three time slots, each link is active for $\frac{1}{3}$ of the time. Since each link has unit capacity, each path carries $\frac{1}{3}$ flow, and the total flow over both paths is $\frac{2}{3}$. ■

An example demonstrating Lemma 4 is shown in Fig. 3. The two node-disjoint paths have an odd total number of edges and it is not possible to schedule the two paths using only two time slots. A third time slot is added, and a feasible schedule is now possible. Using these three time slots, each link is active for $\frac{1}{3}$ of the time, and each path can support a flow of $\frac{1}{3}$, which gives a total flow of $\frac{2}{3}$.

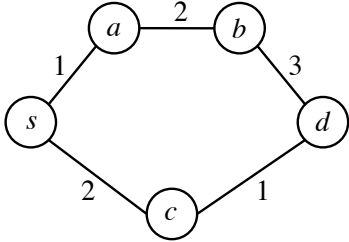


Fig. 3: Node-disjoint paths with an odd total number of edges supporting a flow of $\frac{2}{3}$

We note that it is in fact possible to construct schedules using more than three time slots to achieve higher throughput on node-disjoint paths that have an odd number of edges. These results are given in Appendix B.

These results can be extended to the case where protection is required. For protection against any single link failure in a graph $G = (V, E)$, consider each subgraph after a link failure: $G^e = (V, E \setminus e)$, $e \in E$; all the previous results still apply to each of these new subgraphs. To find the maximum possible protected flow, the maximum flow is found after each edge is individually removed (each possible edge failure). The *minimum* of these flows is the maximum protected flow.

B. Time Efficient Algorithm

Using the different properties of a solution for a single demand under 1-hop interference constraints, we develop an algorithm to solve the problem efficiently. If there exists two node-disjoint paths with an even total number of edges, then the maximum flow is 1 between the source and destination. If there are no node-disjoint paths, then the maximum flow is $\frac{1}{2}$. If there exists only a pair of disjoint paths that has an odd total number of edges, then a flow of $\frac{2}{3}$ can be guaranteed. To find the maximum protected flow between nodes s and d in a graph $G = (V, E)$, the maximum flow is found for each link failure by using a subgraph with each link e removed: $G^e = (V, E \setminus e)$, $e \in E$. The minimum of these maximum flows is the maximum protected flow possible for the demand.

The key to finding the maximum protected flow is to be able to identify node-disjoint paths between s and d with either an even or odd total number of edges. If there are at most two node-disjoint paths, then the maximum flow can only be found if it is possible to find a pair of paths with an even total number of edges. Hence, we focus on trying to find a pair of node-disjoint paths that have an even total number of edges over both of the paths. There has been limited work on trying to identify shortest paths with an even number of edges [20], but no work looking at such an algorithm for disjoint paths.

Development of the optimal algorithm is as follows: we first find the shortest pair of *edge*-disjoint paths with an even number of total edges, and then we extend this algorithm to find the shortest pair of *node*-disjoint paths with an even number of total edges.

1) *Shortest pair of edge-disjoint paths with an even number of total edges*: To find the shortest pair of edge-disjoint paths with an even number of edges, we begin by considering the more general case without the even-edge restriction (the paths can have any number of edges), which was previously considered in [21]. We use a different formulation for the problem by using *minimum-cost flows*, which are defined as finding a flow of minimum cost between a source and destination in a network that has both edge costs and edge capacities [19]. Minimum-cost flows have the property that when given all integer inputs (for edge costs and capacities), they will have all integer solutions (integer flows). We solve the shortest disjoint pair of paths problem by solving the following optimization problem: find a flow of minimum cost to route two units from s to d in a graph with unit capacity and unit cost edges. This will find the shortest pair of disjoint paths since two units of flow need to be routed, no edge can have more than a single unit of flow, and with integer inputs, the solution will be integer, which will be two edge-disjoint paths of unit flow and minimum cost.

One algorithm to solve the minimum-cost flow problem is the successive shortest paths (SSP) algorithm [19]. SSP finds the shortest path, and routes the maximum flow possible onto that path. This repeats until the desired flow between the source and destination is routed. SSP runs in polynomial time to solve the minimum-cost flow formulation for the shortest pair of disjoint paths; further details of SSP can be found in [19].

Using SSP to solve for a minimum-cost flow requires the use of some shortest path algorithm. Assume there exists a shortest path algorithm that is capable of finding a path with an even or odd number of edges; label these algorithms Even and Odd Shortest Path (ESP and OSP, respectively). Using SSP to solve for the shortest pair of disjoint paths with either ESP or OSP as the shortest path function will always yield a pair of disjoint paths with an even total number of edges (if they exist). We call this the Even Shortest Pair of Edge-Disjoint Paths algorithm.

Lemma 5. *The Even Shortest Pair of Edge-Disjoint Paths algorithm will find, if it exists, the shortest pair of disjoint paths with an even total number of edges.*

Proof: First, we provide more detail for the shortest successive paths (SSP) algorithm. For each iteration of SSP, flow is routed on the residual graph, which allows new flows to cancel existing flows; flows in residual graph are known as “augmenting paths”. The residual graph is defined as follows: if edge $\{i, j\}$ has a capacity and cost of (u_{ij}, c_{ij}) with a flow of $f_{ij} \leq u_{ij}$ on it, the residual graph will have two edges $\{i, j\}$ and $\{j, i\}$ with respective costs and capacities $(u_{ij} - f_{ij}, c_{ij})$, and $(f_{ij}, -c_{ij})$ [19]. Finding augmenting paths on the residual graph maintains node conservation constraints; after each iteration of SSP, the residual graph is updated.

To find the shortest pair of disjoint paths, two iterations of

SSP on a unit capacity graph are needed. The first pass will find a path with m_1 number of edges, which depending on if ESP or OSP was used, will be even or odd. The second pass, which is done on the residual graph, will find a path with m_2 edges. If the second path uses any residual flow from the first path, its flow will completely cancel the first path's flow, effectively canceling the usage of the edge in the final set of disjoint paths. Call the number of edges that are cancelled m_x . Since each path used a cancelled edge, when that edge is removed, both paths will no longer traverse the cancelled edge. The total number of edges in the final set of disjoint paths is $m_1 + m_2 - 2m_x$, which is always even. ■

In order to use SSP to find the shortest pair of disjoint paths with an even number of edges, a shortest path algorithm is needed that can find a path with an even or odd number of edges. The algorithm in [20] that finds the shortest path with an even number of edges cannot be easily extended to find the shortest pair of disjoint paths with an even number of edges. Hence, we first focus on developing an algorithm to find the Even Shortest Path (ESP), and then extend ESP to find the Odd Shortest Path (OSP).

We modify the standard Bellman-Ford recursion [19] to search for only paths with an even number of edges, which is shown in Equation 1. We label $S_z(s, k)$ to be the minimum-cost path from node s to k using at most $2z$ edges. The cost of edge $\{i, j\}$ is c_{ij} ; in our case $c_{ij} = 1, \forall \{i, j\} \in E$. Instead of checking if a path from s to j plus edge $\{j, k\}$ is of lower cost than the existing path from s to k , we check to see if the path from s to i plus two edges $\{i, j\}$ and $\{j, k\}$ are of lower cost than the existing path from s to k .

$$S_z(s, k) = \min \left[\min_{\substack{\{i, j\} \in E \\ \{j, k\} \in E \\ \{i, j\} \neq \{j, k\}}} (S_{z-1}(s, j) + c_{ij} + c_{jk}), S_{z-1}(s, k) \right], \\ \forall z = 1..|V|, \forall k \in V \quad (1)$$

To find the shortest path from the source s with an odd number of edges, we run ESP from all neighboring nodes of s (nodes that are one hop from s). The lowest cost path leading back to the source is the solution to OSP.

2) *Shortest pair of node-disjoint paths with an even number of total edges*: In order to optimally solve for routing and scheduling under 1-hop interference constraints, a pair of *node-disjoint* paths with an even number of edges must be found. The Even Shortest Pair of Edge-Disjoint Paths algorithm finds the shortest pair of *edge-disjoint* paths with an even number of edges. To use the *edge-disjoint* algorithm to solve the *node-disjoint* case, each node is transformed into two separate nodes with an edge of zero cost between them: one node has all incoming edges, and the other all outgoing (as shown in Fig. 4). If there existed multiple *edge-disjoint* paths that intersected at node v , they would no longer be able to be *edge-disjoint* in the transformed network, because then they would all have to share the edge $\{v_{in}, v_{out}\}$.

Running the Even Shortest Pair of Edge-Disjoint Paths algorithm on the transformed network will find *node-disjoint*

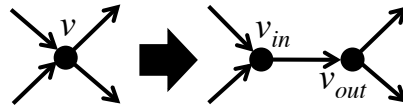


Fig. 4: Node splitting to find node-disjoint paths

paths, but not necessarily achieve the desired result of a pair of disjoint paths with an even number of edges. With the addition of zero-cost edges to the transformed network, finding a pair of disjoint paths with an even number of edges in the transformed network may not yield paths with an even number of edges in the original network. A modification to the algorithm must be made to account for the new edges: in the transformed network, when choosing between an existing path from s to k , or some new path s to i plus a segment i to k , consider only segments that have an even number of “original” edges. This will ensure that a final path in the original network will have an even number of edges. The algorithm now begins to more closely resemble the Floyd-Warshall algorithm [19], which considers joining segments to find a shortest path. This new algorithm is called Even Shortest Pair of Node-Disjoint Paths.

These results can be extended to solve Wireless Guaranteed Protection problem with a single demand under 1-hop interference constraints. The maximum flow is found after every possible edge failure for each subgraph $G^e = (V, E \setminus e)$, $\forall e \in E$. The minimum of these maximum flows is the maximum protected flow. For each instance, we first see if there exists a pair of node-disjoint paths with an even total number of edges. If this exists, then a maximum flow of 1 is possible. If not, we check to see if there exist node-disjoint paths with an odd total number of edges (by running the standard edge-disjoint path routing algorithm on the transformed graph). If this exists, then a flow of $\frac{2}{3}$ is possible using three time slots. If no node-disjoint paths exist, then find some path from s to d , which can support a flow of $\frac{1}{2}$.

IV. AN OPTIMAL FORMULATION FOR WIRELESS GUARANTEED PROTECTION

In the previous section, an optimal solution for routing and scheduling with protection for a single demand was presented. While this provides insight, typical networks will need to simultaneously handle multiple connections. Additionally, many networks have interference constraints other than the 1-hop model. This section provides a mathematical formulation to the optimal solution for the Wireless Guaranteed Protection (WGP) problem with general interference constraints. In particular, for a set of demands, a route and schedule needs to be found such that after any link failure, all end-to-end connections maintain their same level of flow. For general interference constraints, the routing and scheduling problem was demonstrated to be NP-Hard [9]. We conjecture that adding protection constraints preserves NP-hardness; hence, a mixed integer linear program (MILP) is formulated to find an optimal solution to WGP.

In wired networks, a typical objective function for protection is to minimize the total allocated capacity needed to satisfy all demands. A similar objective cannot be clearly defined for

wireless networks since the concept of capacity changes in the presence of interference constraints. Consider some active link $\{i, j\}$. An adjacent link $\{j, k\}$ cannot be used simultaneously with $\{i, j\}$ because of interference; hence, simply adding additional link capacity (in a wired sense) will not allow its use. Another time slot must be allocated to allow a connection to use $\{j, k\}$ such that it does not interfere with $\{i, j\}$. Adding an additional time slot will reduce the time that each individual time slot in the schedule is active, which reduces the overall throughput of the network [8, 9, 13]. For example, consider a network with two time slots and a connection that supports a flow of 1 using these two time slots. If a third time slot is added to the schedule, then the original two time slots are only active for $\frac{2}{3}$ of the total time, and that flow's scheduled throughput is reduced from 1 to $\frac{2}{3}$. Thus, the objective we consider is to use a minimum number of time slots to route and schedule each demand with protection.

Finding a protection route and schedule using the minimum number of time slots allows for a simple comparison to existing wired and wireless protection schemes. The difference between the number of time slots necessary to route and schedule a set of demands before and after adding protection will be considered the reduction of the maximum throughput. To be consistent with the wireless protection scheme mentioned in [8], wireless flows are restricted to single paths (no flow splitting allowed). For ease of exposition, the MILP assigns the same throughput to all demands; see Appendix Section A for the formulation with different throughput requirements.

For the MILP, the following values are given:

- $G = (V, E)$ is the graph with a set of vertices and edges
- D is the set of flow requirements
- u_{ij} is the capacity of link $\{i, j\}$
- \mathcal{I} is the interference matrix, where $I_{ij}^{kl} \in \mathcal{I}$ is 1 if links $\{i, j\}$ and $\{k, l\}$ can be activated simultaneously, 0 otherwise
- \mathcal{T} is the set of time slots in the system, $\mathcal{T} \subset \mathbb{Z}^+$

The MILP solves for the following variables:

- x_{ij}^{sd} is a routing variable and is 1 if primary flow is assigned for demand (s, d) on link $\{i, j\}$, 0 otherwise
- $y_{ij,kl}^{sd}$ is a routing variable and is 1 if protection flow is assigned on link $\{i, j\}$ for the demand (s, d) after the failure of link $\{k, l\}$, 0 otherwise
- $\lambda_{ij}^{sd,t}$ is a scheduling variable and is 1 if link $\{i, j\}$ can be activated in time slot t for the demand (s, d) , 0 otherwise
- $\delta_{ij,kl}^{sd,t}$ is a scheduling variable and is 1 if link $\{i, j\}$ can be activated in time slot t after failure of link $\{k, l\}$ for the demand (s, d) , 0 otherwise
- s_t is 1 if time slot t is used by any demand, and 0 otherwise

The objective function is to minimize the number of time slots (the length of the schedule) needed to route all demands with protection:

$$\text{Objective: } \min \sum_{t \in \mathcal{T}} s_t \quad (2)$$

The following constraints are imposed to find a feasible routing and scheduling with protection.

Before a link failure:

- Flow conservation constraints for the primary flow: route primary traffic before a failure for each demand.

$$\sum_{\{i,j\} \in E} x_{ij}^{sd} - \sum_{\{j,i\} \in E} x_{ji}^{sd} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in V, \forall (s, d) \in D \quad (3)$$

- In any given time slot, for a given demand, only links that do not interfere with one another can be activated simultaneously.

$$\sum_{(s,d) \in D} \lambda_{ij}^{sd,t} + \sum_{(s,d) \in D} \lambda_{kl}^{sd,t} \leq 1 + I_{kl}^{ij}, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall t \in \mathcal{T} \quad (4)$$

- Only one demand can use a given link at a time.

$$\sum_{(s,d) \in D} \lambda_{ij}^{sd,t} \leq 1, \quad \forall \{i,j\} \in E, \forall t \in \mathcal{T} \quad (5)$$

- Ensure enough capacity exists to support the necessary flow for demand (s, d) on edge $\{i, j\}$ for the length of time that the link is active.

$$x_{ij}^{sd} \leq \sum_{t \in \mathcal{T}} \lambda_{ij}^{sd,t} u_{ij}, \quad \forall \{i,j\} \in E, \forall (s,d) \in D \quad (6)$$

- Mark if slot t is used to schedule a demand before a failure.

$$\lambda_{ij}^{sd,t} \leq s^t, \quad \forall \{i,j\} \in E, \forall t \in \mathcal{T}, \forall (s,d) \in D$$

After a link failure:

- Flow conservation constraints for protection flow: route protection traffic after each link failure $\{k, l\} \in E$.

$$\sum_{\{i,j\} \in E} y_{ij,kl}^{sd} - \sum_{\substack{\{j,i\} \in E \\ \{k,l\} \neq \{i,j\}}} y_{ji,kl}^{sd} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in V, \forall \{k,l\} \in E, \forall (s,d) \in D \quad (7)$$

- In any given time slot after the failure of link $\{k, l\}$, only links that do not interfere with one another can be activated simultaneously.

$$\sum_{(s,d) \in D} \delta_{ij,kl}^{sd,t} + \sum_{(s,d) \in D} \delta_{uv,kl}^{sd,t} \leq 1 + I_{uv}^{ij}, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall \{u,v\} \in E, \forall t \in \mathcal{T}, \forall \{i,j\} \neq \{k,l\} \neq \{u,v\} \quad (8)$$

- Only one demand can use a given link at a time after the failure of link $\{k, l\}$.

$$\sum_{(s,d) \in D} \delta_{ij,kl}^{sd,t} \leq 1, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall t \in \mathcal{T} \quad (9)$$

- Ensure enough capacity exists after the failure of link $\{k, l\}$ to support the necessary flow on edge $\{i, j\}$ for the length of time that the link is active.

$$y_{ij,kl}^{sd} \leq \sum_{t \in \mathcal{T}} \delta_{ij,kl}^{sd,t} u_{ij}, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall (s,d) \in D \quad (10)$$

- Mark if time slot t is used to schedule a demand after the failure of link $\{k, l\}$.

$$\delta_{ij,kl}^{sd,t} \leq s^t, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall t \in \mathcal{T}, \forall (s,d) \in D$$

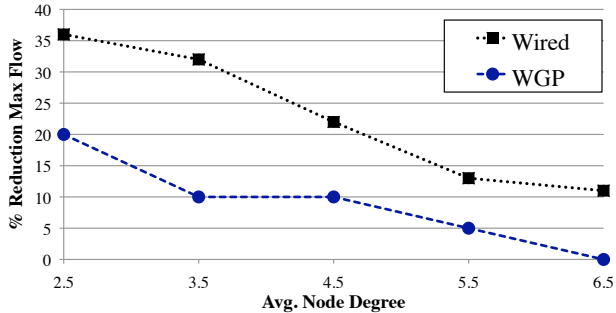


Fig. 5: Reduction of throughput when adding protection

To demonstrate how protection can be added to wireless networks with minimal reduction of throughput, WGP is compared to both the wired (without interference) and wireless protection (with interference) schemes. One hundred random graphs were generated with 25 nodes each. Nodes that are physically within a certain transmission range of one another are considered to have a link, and the transmission range is varied to give different desired average node degrees. The node degree is varied from 2.5 to 6.5, and for each graph, ten source/destination pairs are randomly chosen to be routed concurrently. All links have unit capacity; 1-hop interference constraints were used for the wireless networks. The simulation results are found in Fig. 5.

For comparison to wired protection, we use the same network topologies, however, in the wired case we do not enforce the interference constraints (i.e., all links can be activated simultaneously). For wired protection, we compute the reduction in throughput as the reduction in maximum flow after protection is added. As compared to the wired protection scheme, WGP has a lower reduction in throughput for all node degrees examined. For node degree 2.5, both the WGP and the wired protection schemes have larger reductions in throughput: 20% for WGP and 37% for wired. This is because at lower node degrees, there are fewer available end-to-end paths, and therefore after a failure, there are fewer routing options available. As the node degree increases, and there are more available end-to-end paths, the reduction in throughput decreases when adding protection. In fact, it is often possible for WGP to have no reduction in the throughput between the protected and unprotected setting. For an average node degree of 3.5, WGP only loses about 10% of throughput when adding protection, while the wired scheme loses 32%. For 20% of the simulations at node degree 3.5, there was no loss in throughput for WGP. When the node degree goes to 6.5, WGP no longer has any loss in flow, while the wired setting still has a loss of 11%.

We compare WGP to a wireless 1+1 protection scheme. In particular, wireless 1+1 protection applies the wired 1+1 protection scheme to wireless networks (as mentioned in [8]): i.e., find a schedule for the shortest pair of disjoint paths in the network between the source and destination, with the primary flow before a failure routed onto one path, and the backup flow routed onto the other. To compare WGP to wireless 1+1, the number of time slots needed beyond the non-protection routing

Avg. Node Degree	% Reduction of Protection Time Slots
2.5	72
3.5	63
4.5	60
5.5	52
6.5	46

TABLE I: WGP vs. Wireless 1+1

are compared; these are the time slots needed to meet the protection requirements. Table I shows the percent reduction in number of time slots needed to provide protection using WGP over wireless 1+1. When the average node degree is 2.5, WGP has up to a 72% reduction of time slots needed to meet protection requirements. The reason for this is that wireless 1+1 is scheduling two paths for each demand, a primary and a backup, and not trying to recapture any capacity after a failure; this in turn causes a significant increase in interference between connections. As the node degree increases, there is increased path diversity and more opportunities to find interference-free routings; hence, wireless 1+1 has better performance. But at all times, wireless 1+1 needs significantly more time slots to provide protection for all of the demands than WGP does, which is able to recapture capacity after a failure.

V. ALGORITHMS FOR PROVIDING WIRELESS PROTECTION

In the previous section, an MILP was presented to find an optimal solution to Wireless Guaranteed Protection (WGP), which is not a computationally efficient method of finding a solution. In this section, two time-efficient algorithms are presented to solve the Wireless Guaranteed Protection problem for a set of demands. Similar to the previous section, primary and backup flows are restricted to single paths, and the objective is to minimize the length of the schedule to route all demands with protection. We first show that this problem is NP-Hard under 1-hop interference constraints. Next, algorithms are developed assuming unit demands, unit capacity edges, and a single link failure model; the algorithms can be modified to reflect other values of demand and capacity. The algorithms are developed for dynamic (one-at-a-time) arrivals: an incoming demand needs to be routed and scheduled over an existing set of connections; the existing set cannot have their routings or schedules changed. A 1-hop interference model is used, but the algorithms can be extended to a generic K -hop interference model, with the extensions detailed in the end of Section V-B. We find that when compared to the optimal batch case (all connections are routed and scheduled simultaneously), the dynamic routing performs within a few percentage points of optimal.

First, in Section V-A, we demonstrate WGP to be NP-Hard under 1-hop interference constraints when flows are restricted to a single path. Next, in Section V-B, an algorithm to find a shortest 1-hop interference-free path using a minimal number of time slots is presented. This serves as the building block for the next two algorithms that are developed. In Section V-C,

an algorithm for finding a minimal length schedule for WGP is presented, where a backup route and schedule is found for each possible failure. This approach has drawbacks in that after any failure, a new route is found; hence, a route and schedule for each failure event needs to be stored. To overcome this, an algorithm is developed in Section V-D using disjoint paths such that only two paths are needed: a primary and a backup. In Section V-E, the performance of the two algorithms are compared to the optimal MILP formulation.

A. Complexity Results under 1-hop Interference Constraints

Without protection, the routing and scheduling problem is NP-Hard under general interference constraints [9]. But if flows for each demand are allowed to be split, a polynomial timed algorithm is possible for 1-hop interference constraints [11]. We demonstrate that when flows cannot be split, the routing and scheduling problem becomes NP-Hard under 1-hop interference constraints.

Theorem 1. *Finding the minimum length schedule to route a set of demands under 1-hop interference constraints when flow splitting is not allowed is NP-Hard.*

We first consider the following necessary and sufficient condition for routing a set node-disjoint¹ pairs, $(s_1, d_1), \dots, (s_N, d_N)$, in only two time slots without flow splitting.

Lemma 6. *Under 1-hop interference constraints, a set of demands that are node-disjoint can be routed and scheduled using two time slots without flow splitting if and only if there exists node-disjoint paths between each of the node pairs.*

Proof: If there exists a node-disjoint path between every node pair in the set of demands, then a schedule using two slots is possible. A proof can be accomplished by construction. Any individual path can be scheduled in two time slots by using alternating time slots. If all paths are node-disjoint, then there exists no conflicts between paths under 1-hop interference constraints. Therefore, all paths can be scheduled using the same two-time slots.

For the other direction, assume otherwise: a two slot schedule is possible without there being a node-disjoint path between every pair of nodes in the set of demands. There exists a node v that has $m \geq 2$ paths crossing it. There are m paths coming into v and m paths out that need to be scheduled. Under 1-hop interference constraints, this will require at least $2m$ time slots to produce an interference free schedule. ■

Using Lemma 6, Theorem 1 can be quickly demonstrated.

Proof of Theorem 1: We reduce the Disjoint Connecting Paths Problem (DCPP) [22] to ours. DCPP asks the following question: given a graph $G = (V, E)$ and a collection of N node-disjoint pairs $(s_1, d_1), \dots, (s_N, d_N)$, does G contain N mutually node-disjoint paths, one connecting s_i and d_i for each i , $1 \leq i \leq N$? We can ask an equivalent question for our routing and scheduling problem: can a set of N node-disjoint pairs be routed and scheduled using the minimal number of time slots (two) under 1-hop interference constraints without

flow splitting? If yes, then by Lemma 6 that means we have found N mutually node-disjoint paths, one connecting s_i and d_i for each i , $1 \leq i \leq N$, which solves DCPP. An answer of no means a solution to DCPP does not exist. ■

Next, we extend this complexity result to the case when protection is required.

Theorem 2. *Finding the minimum length schedule to route a set of demands with protection under 1-hop interference constraints without flow splitting is NP-Hard.*

The proof can be found in Appendix Section C.

B. Minimum Schedule for an Interference Free Path

We begin by developing an algorithm to find a shortest interference-free path using the minimum number of time slots under the 1-hop interference model. This algorithm will be a building block for the two protection algorithms that will be discussed in the upcoming sections. We consider an incoming demand for a connection between nodes s and d . Connections already exist in the network, with the set of \mathcal{T} time slots already in use. Based on how the current connections are routed and scheduled, a set of edge interferences \mathbf{I} can be constructed, where for every edge $\{i, j\}$, $I_{ij} \in \mathbf{I}$ is the set of time slots that cannot be used on that edge because either that time slot is already used by $\{i, j\}$, or using that time slot on $\{i, j\}$ will interfere with another edge using it at that time. The set of edge interferences \mathbf{I} can be constructed in polynomial time, and will be given as an input to the algorithm.

First, we wish to determine the shortest interference free path without using any additional time slots beyond the set \mathcal{T} , and without rescheduling or rerouting existing connections. Each edge $\{i, j\}$ has a set of free time slots during which it can be used: $\tau_{ij} = \mathcal{T} \setminus I_{ij}$. Let P be the set of edges used in a path. If each edge of a loop-free path P has at least two free time slots, then that path can be scheduled without interference using the existing time slot allocation \mathcal{T} .

Lemma 7. *For 1-hop interference, a loop-free path P can be scheduled without interference if $|\tau_{ij}| \geq 2, \forall \{i, j\} \in P$.*

Proof: If $|\tau_{ij}| \geq 2, \forall \{i, j\} \in P$, then each edge in P has an available time slot that does not interfere with its adjacent set of edges. Since the path is loop-free, any two edges that use the same time slot will never be less than one hop apart from one another, and therefore never interfere with each other. ■

Using the result from Lemma 7, the following algorithm is constructed to find a 1-hop interference-free path using only the set of time slots \mathcal{T} : remove all edges in G that have $|\tau_{ij}| \leq 1$, find the shortest path P_{sd} between s and d , and assign time slots to the edges in P_{sd} such that it has an interference-free schedule.

An improvement can be made to the algorithm by attempting to maximize the number of free time slots on any edge, so that future connections will be less likely to require additional time slots to find an interference-free path. Currently, edges that have many free time slots are not given any preference. If an edge has only the minimal number of free time slots, it may be selected

¹A node is a source or destination for at most one demand.

for use in a path. This may hurt finding interference-free paths for future connections by limiting the number of available time slots on an edge, thus necessitating new time slots. We assign a cost for each edge to be equal to the number of time slots that that edge interferes with: $c_{ij} = |I_{ij}|$. With respect to these new edge costs, a minimum-cost interference-free path is found. The more time slots an edge is in conflict with, the more expensive that edge will be, and the less likely it will be used in a route. We refer to this algorithm as `int_free_path`, which will return the edges and schedule of a path between s and d .

To find an interference free path that tries to minimize future conflicts, and using minimum additional time slots, we first find a minimum-cost interference free path for the current set of time slots assigned in the network, \mathcal{T} . If such a path does not exist, increase the set of available time slots by 1, and repeat. We note that the set of time slots will never increase by more than two since a feasible schedule can be found for any path with two free time slots. We call this algorithm `find_path`.

C. Minimum Length Schedule for Wireless Protection

In this section, an algorithm is developed that tries to find the minimum length schedule for the Wireless Guaranteed Protection problem, with an approach that is similar to the optimal solution found by the MILP in Section IV. The problem is broken up into $|E| + 1$ subproblems. First, the minimum length schedule is found to route the set of demands before a failure. Then, for each possible failure, the minimum length schedule is found to route the set of demands on a failure graph $G^{kl} = (V, E \setminus \{k, l\})$, $\forall \{k, l\} \in E$ (i.e., the graph that remains after the failure of edge $\{k, l\}$). Each of the solutions to these subproblems represents the route and schedule necessary to meet the protection requirements for the set of demands before and after any link failure.

The *maximum* of any of these minimum length schedules will be the length of the schedule needed to add protection to set of demands in a wireless network. The algorithm is called `minimum_protect`; it will return the set of paths and schedules for each demand, indicating which path and schedule to use after any link failure.

D. Disjoint Path Wireless Guaranteed Protection

In Section V-C, an algorithm was described to find the minimum number of time slots to route and schedule a set of demands with protection. After any failure, a new route is found; hence, many possible routing configurations exist, and a route and schedule for each failure event needs to be saved. A more desirable approach may be to limit the number of paths needed to only two: a primary and a backup. Before continuing with the development of the algorithm, a complexity result is presented regarding using disjoint paths to provide protection in a wireless network with 1-hop interference constraints. For a set of time slots \mathcal{T} , simply determining if any solution exists to WGP using disjoint paths is NP-Complete.

Theorem 3. *For an incoming connection between s and d , using disjoint paths to provide protection in a wireless network*

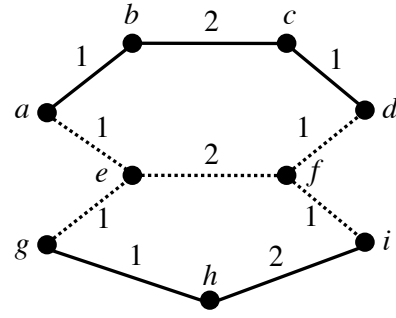


Fig. 6: Disjoint path routing and scheduling with protection

with 1-hop interference constraints for the set of time slots \mathcal{T} is NP-Complete.

A reduction is performed from the Dynamic Shared-Path Protected Lightpath-Provisioning (DSPLP) [2]. The proof can be found in Appendix Section D.

Our approach for developing an algorithm to solve WGP using disjoint paths is similar to the wireless 1 + 1 protection scheme described earlier; however, we take advantage of the time slot reuse that is possible before and after a failure, as well as the opportunity to share protection resources between failure disjoint demands. If an edge in a primary path P uses time slot t , then for 1-hop interference, all edges adjacent to that edge also cannot use t . After the failure of an edge in the primary path, the time slots used to route that path are no longer needed (since they are not being used). The time slots on the edges of the primary path that did not fail now can be reused for protection; furthermore, the time slots on the edges that interfered with the failed primary path also become free to use for protection.

Protection resource sharing can also allow for time slot reuse. If two primary paths are failure disjoint under a single link failure model, only one will fail at a time. Hence, a time slot t on adjacent edges can be shared for protection between the two failure disjoint connections, since the two adjacent edges will never be activated simultaneously.

An example is shown in Fig. 6. Two demands need to be routed under 1-hop interference constraints: one from a to d , and another from g to i . Each edge is assigned a time slot, with the time slot labeling shown in the figure. The edges used for primary flow are indicated by solid lines, and the edges used for protection are dotted lines. After the failure of edge $\{a, b\}$, the entire primary path between a and d is no longer active, and its time slots will no longer be in use; hence, edges $\{a, e\}$ and $\{f, d\}$ can use time slot 1, even though they would have conflicted with $\{a, b\}$ and $\{c, d\}$ before the failure. Similarly, $\{g, e\}$ is assigned time slot 1, even though primary edge $\{g, h\}$ is assigned the same time slot. Since both primary paths are failure disjoint, time slot 2 on $\{e, f\}$ is shared between the two connections for protection. Additionally, because at most one backup path will be used at a time, protection edges $\{g, e\}$ and $\{a, e\}$ can both be assigned time slot 1; they will never interfere with one another. Similarly, $\{f, i\}$ and $\{f, d\}$ can be

both assigned time slot 1.

This idea of time slot reuse after a failure forms the basis for the disjoint path wireless protection algorithm, which we label `disjoint_protect`. We consider an incoming demand requesting a connection between nodes s and d . Connections already exist in the network, with the set of \mathcal{T} time slots already in use. A interference-free primary path between s and d , P_{sd} , is found using `find_path`. Once a primary path fails, none of the time slots needed for that path, or on the edges that interfered with that path, are needed, and they become available to be used for protection. Next, a backup path B_{sd} is found that is disjoint to P_{sd} , and does not interfere with any of the other connections that did not fail. Additionally, the backup path B_{sd} will not interfere with the protection routings for the different existing demands that would fail if an edge in P_{sd} fails (i.e., B_{sd} will not interfere with the protection paths for demands whose primary paths are not disjoint with P_{sd}). The algorithm is detailed in Algorithm 1.

Algorithm 1

$(\mathbf{P}_{sd}, \mathcal{I}, \mathcal{T}) = \text{disjoint_protect}(G, \mathcal{I}, \mathcal{T}, s, d)$

Find route and schedule before a failure:

$(P_{sd}, \mathcal{T}, \mathbf{I}') = \text{find_path}(G, \mathbf{I}, \mathcal{T}, s, d)$

Construct new network without the edges in path P_{sd} :

$G^F = (V, E \setminus P_{sd})$

Once an edge for that demand fails, none of the slots needed to support it are used and become available. Construct a failure interference set using the interference set for the primary routes before that demand was routed, and the failure interference sets for each edge $\{k, l\}$ in P_{sd}

$\mathbf{I}^F = (\cup_{\{k, l\} \in P_{sd}} \mathbf{I}^{kl}) \cup \mathbf{I}$

Find a disjoint path, and schedule it:

$(P_{sd}^F, \mathcal{T}, \mathbf{I}'') = \text{find_path}(G^F, \mathbf{I}^F, \mathcal{T}, s, d)$

Update interference sets:

Before a failure: $\mathbf{I} = \mathbf{I}'$

After each primary path failure: $\mathbf{I}^{kl} = \mathbf{I}''$, $\forall \{k, l\} \in P_{sd}$

$\mathcal{I} =$ Set of \mathbf{I} and all \mathbf{I}^{kl}

$\mathbf{P}_{sd} =$ Set of P_{sd} and all P_{sd}^{kl}

Return $(\mathbf{P}_{sd}, \mathcal{I}, \mathcal{T})$

E. WGP Algorithm Simulations

The algorithms `minimum_protect` and `disjoint_protect` are compared to the optimal solution found by the MILP in Section IV. A similar simulation setup is used as that in Section IV. One hundred random graphs were generated with 25 nodes each. The node degree is varied from 2.5 to 6.5, and for each random graph, ten source/destination pairs are randomly chosen to be routed concurrently, each with a unit demand. All links have unit capacity, and 1-hop interference constraints were used. The algorithms route and schedule demands one-at-a-time, while the MILP optimizes the route and schedule for all demands together (in batch). To compare the two, the algorithms randomly order the set of

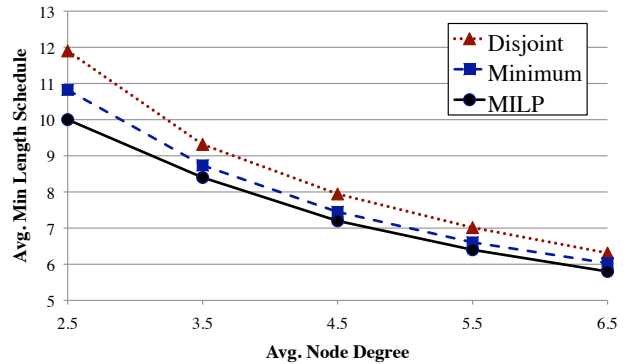


Fig. 7: Avg. time slots needed for WGP

demands, and then solves for each demand one-at-a-time. The simulation results are found in Fig. 7.

Similar to the previous simulation, as node degree increased, the average minimum length schedule decreased. This is because of the increased diversity in possible number of end-to-end path, which leads to a greater opportunity of finding interference free paths. On average, `minimum_protect` needed only 4.5% more time slots to meet all requirements than the optimal MILP needed, and `disjoint_protect` needed 10.1% more time slots than the MILP.

VI. CONCLUSION

In this chapter, the problem of guaranteed protection in a multi-hop wireless network is introduced. Because of link interference, resources that were unavailable prior to a failure can be used for protection after the failure. In fact, protection can often be provided using no additional resources. For the case of a single demand with 1-hop interference constraints, properties of an optimal solution are presented, and a time-efficient algorithm is developed that solves the problem of wireless routing and scheduling with and without protection, guaranteeing a maximum throughput that is within 1.5 of optimal. For general interference constraints and multiple concurrent demands, an optimal solution is developed for the protection problem via a mixed integer linear program. When compared to using traditional wired protection schemes on a wireless network, our Wireless Guaranteed Protection (WGP) scheme uses as much as 72% less protection resources to achieve the same level of resiliency. Two low-complexity algorithms to solve WGP are developed, and on average, these algorithms perform close to the optimal solution. A future direction for our work is to adapt the schemes developed in this chapter to a distributed setting.

APPENDIX

A. MILP for WGP with Different Throughputs

Some demand between nodes s and d has its own throughput requirement f^{sd} . The objective of the MILP is to minimize the number of times slots needed to schedule every demand. Since each demand has a throughput requirement, finding a minimum length schedule will be with respect to keeping the ratio of the different demands' throughputs constant. We assume f^{sd}

is integer $\forall (s, d) \in (V, V)$. If necessary, the demands and link capacities can be scaled by the smallest integer that makes all demand values integer (hence, f^{sd} is assumed to be at the very least rational, $\forall (s, d) \in (V, V)$).

For the MILP, the following values are given:

- $G = (V, E)$ is the graph with a set of vertices and edges
- D is the set of flow requirements
- f^{sd} is the flow required between nodes (s, d) ; $f^{sd} \in \mathbb{Z}$
- u_{ij} is the capacity of link $\{i, j\}$
- \mathcal{I} is the interference matrix, where $I_{ij}^{kl} \in \mathcal{I}$ is 1 if links $\{i, j\}$ and $\{k, l\}$ can be activated simultaneously, 0 otherwise
- \mathcal{T} is the set of time slots in the system, $\mathcal{T} \subset \mathbb{Z}^+$

The MILP solves for the following variables:

- x_{ij}^{sd} is a routing variable and is 1 if primary flow is assigned for demand (s, d) on link $\{i, j\}$, 0 otherwise
- $y_{ij,kl}^{sd}$ is a routing variable and is 1 if protection flow is assigned on link $\{i, j\}$ for the demand (s, d) after the failure of link $\{k, l\}$, 0 otherwise
- $\lambda_{ij}^{sd,t}$ is a scheduling variable and is 1 if link $\{i, j\}$ can be activated in time slot t for the demand (s, d) , 0 otherwise
- $\delta_{ij,kl}^{sd,t}$ is a scheduling variable and is 1 if link $\{i, j\}$ can be activated in time slot t after failure of link $\{k, l\}$ for the demand (s, d) , 0 otherwise
- s_t is 1 if time slot t is used by any demand, and 0 otherwise

The objective function is to minimize the number of time slots (the length of the schedule) needed to route all demands with protection:

$$\textbf{Objective:} \quad \min \sum_{t \in \mathcal{T}} s_t \quad (11)$$

The following constraints are imposed to find a feasible routing and scheduling with protection.

Before a link failure:

- Flow conservation constraints for the primary flow: route primary traffic before a failure for each demand.

$$\sum_{\{i,j\} \in E} x_{ij}^{sd} - \sum_{\{j,i\} \in E} x_{ji}^{sd} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in V, \forall (s, d) \in D \quad (12)$$

- In any given time slot, for a given demand, only links that do not interfere with one another can be activated simultaneously.

$$\sum_{(s,d) \in D} \lambda_{ij}^{sd,t} + \sum_{(s,d) \in D} \lambda_{kl}^{sd,t} \leq 1 + I_{kl}^{ij}, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \{i,j\} \neq \{k,l\}, \forall t \in \mathcal{T} \quad (13)$$

- Only one demand can use a given link at a time.

$$\sum_{(s,d) \in D} \lambda_{ij}^{sd,t} \leq 1, \quad \forall \{i,j\} \in E, \forall t \in \mathcal{T} \quad (14)$$

- Ensure enough capacity exists to support the necessary flow f^{sd} for demand (s, d) on edge $\{i, j\}$ for the length

of time that the link is active.

$$f^{sd} x_{ij}^{sd} \leq \sum_{t \in \mathcal{T}} \lambda_{ij}^{sd,t} u_{ij}, \quad \forall \{i,j\} \in E, \forall (s,d) \in D \quad (15)$$

- Mark if slot t is used to schedule a demand before a failure.

$$\lambda_{ij}^{sd,t} \leq s^t, \quad \forall \{i,j\} \in E, \forall t \in \mathcal{T}, \forall (s,d) \in D$$

After a link failure:

- Flow conservation constraints for protection flow: route protection traffic after each link failure $\{k, l\} \in E$.

$$\sum_{\substack{\{i,j\} \in E \\ \{k,l\} \neq \{i,j\}}} y_{ij,kl}^{sd} - \sum_{\substack{\{j,i\} \in E \\ \{k,l\} \neq \{j,i\}}} y_{ji,kl}^{sd} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in V, \forall \{k,l\} \in E, \forall (s,d) \in D \quad (16)$$

- In any given time slot after the failure of link $\{k, l\}$, only links that do not interfere with one another can be activated simultaneously.

$$\sum_{(s,d) \in D} \delta_{ij,kl}^{sd,t} + \sum_{(s,d) \in D} \delta_{uv,kl}^{sd,t} \leq 1 + I_{uv}^{ij}, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall \{u,v\} \in E, \forall t \in \mathcal{T}, \{i,j\} \neq \{k,l\} \neq \{u,v\} \quad (17)$$

- Only one demand can use a given link at a time after the failure of link $\{k, l\}$.

$$\sum_{(s,d) \in D} \delta_{ij,kl}^{sd,t} \leq 1, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall t \in \mathcal{T} \quad (18)$$

- Ensure enough capacity exists after the failure of link $\{k, l\}$ to support the necessary flow f^{sd} on edge $\{i, j\}$ for the length of time that the link is active.

$$f^{sd} y_{ij,kl}^{sd} \leq \sum_{t \in \mathcal{T}} \delta_{ij,kl}^{sd,t} u_{ij}, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall (s,d) \in D \quad (19)$$

- Mark if time slot t is used to schedule a demand after the failure of link $\{k, l\}$.

$$\delta_{ij,kl}^{sd,t} \leq s^t, \quad \forall \{i,j\} \in E, \forall \{k,l\} \in E, \forall t \in \mathcal{T}, \forall (s,d) \in D$$

B. Schedules for Higher Throughput on Node-Disjoint Paths with an Odd Number of Edges

In Section III-A, for a pair of node-disjoint paths whose total number of edges is odd, a schedule using three time slots was used to achieve a flow of $\frac{2}{3}$ between the source and the destination. Each link is assigned one of three time slots, and since each link is active for only $\frac{1}{3}$ of the time, each path supports a flow of $\frac{1}{3}$, and the total end-to-end flow is $\frac{2}{3}$. An example is shown on the five edge network in Figure 8a.

By using additional time slots, it is in fact possible to increase the end-to-end throughput. For the same five edge network, a maximum flow of $\frac{5}{6}$ is possible using six time slots. A schedule that achieves this flow is shown in Figure 8b. On the shorter path, three time slots are assigned to each link. Since there are a total of six time slots in use, each link is active for half of the total time, and is supporting a flow of $\frac{1}{2}$. On the longer path, each link is assigned two time slots. These two time slots

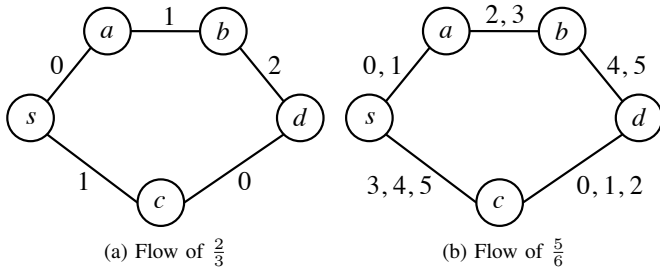


Fig. 8: Node-disjoint paths with an odd total number of edges supporting a flow of $\frac{2}{3}$ and $\frac{5}{6}$.

represent $\frac{1}{3}$ of the total time; hence the longer path supports a flow of $\frac{1}{3}$. The total flow on both paths is $\frac{1}{2} + \frac{2}{3} = \frac{5}{6}$.

If the longer of the two node-disjoint paths has K edges, then it is in fact possible to always achieve a throughput over the two paths of $\frac{2K-1}{2K}$ by employing the following scheduling scheme that uses $2K$ time slots. On the shorter path, we assign half of the time slots to each edge: K to $(2K-1)$ on the first edge, 0 to $(K-1)$ on the second edge, and alternate between those two assignments for each subsequent edge for the remainder of the path. Since each edge of the shorter path uses half of the time slots, each edge is active $\frac{1}{2}$ of the time, and the shorter path carries a flow of $\frac{1}{2}$.

On the longer path (having K edges), assign $(K-1)$ time slots to each edge in the following fashion: For the j^{th} edge, where edge 0 leaves the source and edge $K-1$ enters the destination, assign time slots $\text{mod}[j(K-1), 2K]$ through $\text{mod}[(j+1)(K-1)-1, 2K]$. The notation $\text{mod}[a, b]$ represents the modulo function whose value is the integer remainder when a is divided by b . Each edge has $(K-1)$ time slots assigned to it and is active for $\frac{K-1}{2K}$ of the time, allowing the longer path to support a flow of $\frac{K-1}{2K}$. The total flow across both paths is $\frac{K}{2K} + \frac{K-1}{2K} = \frac{2K-1}{2K}$. This scheduling scheme can always achieve a throughput of $\frac{2K-1}{2K}$, which is demonstrated in Lemma 8.

Two examples are shown in Figure 9. In the first network, shown in Figure 9a, the longer path has five edges ($K=5$), and the shorter has four. Ten time slots are used in total, with the shorter path supporting a flow of $\frac{1}{2}$, and the longer path supporting a flow of $\frac{4}{5}$, resulting in an end-to-end flow of $\frac{9}{10}$. It is straightforward to see that if the shorter path had two edges instead of four, the same throughput would have been achievable using the same ten time slots. In the second network, shown in Figure 9b, the longer path has four edges ($K=4$), and the shorter has three; eight time slots are used. The shorter path supports a flow of $\frac{1}{2}$, the longer path supports a flow of $\frac{3}{8}$, and the total end-to-end flow is $\frac{7}{8}$.

Lemma 8. *For a pair of node-disjoint paths with an odd number of edges, where the longer path has K edges, a schedule exists that achieves a throughput of $\frac{2K-1}{2K}$ over the two paths.*

Proof: We demonstrate the scheduling scheme presented

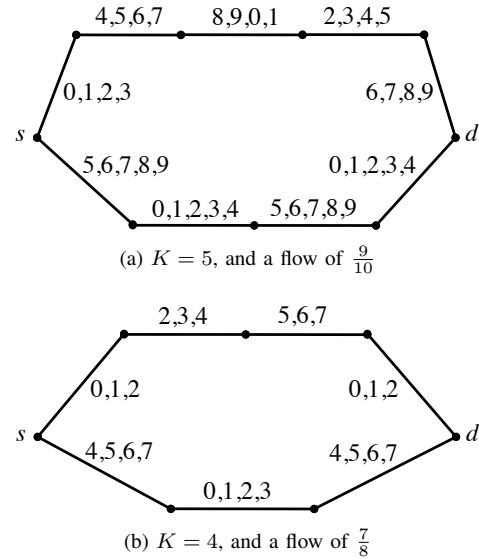


Fig. 9: Node-disjoint paths with an odd number of edges supporting flows of $\frac{2K-1}{2K}$

in this section always achieves the desired rate of $\frac{2K-1}{2K}$. We consider two cases: K is odd, and K is even.

We first examine the case where K is odd; an example was shown in Figure 9a. Since the longer path has an odd number of edges, the shorter path must have an even number. On the shorter path, half the time slots are assigned to each edge, alternating between time slots K to $2K-1$ on the first edge, time slots 0 to $K-1$ on the second edge, and so forth until the final edge. Since there is an even number of edges, the final edge of the shorter path entering the destination will be assigned time slots 0 to $K-1$. On the longer path, time slots $\text{mod}[j(K-1), 2K]$ through $\text{mod}[(j+1)(K-1)-1, 2K]$ are assigned to the j^{th} edge, with edge 0 leaving the source and edge $K-1$ entering the destination. This results in $K-1$ time slots assigned to each edge. By construction, the edges leaving the source for each path do not interfere with one another. We need to verify that the final edges entering the destination also do not interfere. The final edge of the path entering the destination is numbered $j=K-1$; the time slot assignment for that edge is $\text{mod}[(K-1)(K-1), 2K]$ through $\text{mod}[K(K-1)-1, 2K]$. The value of $\text{mod}[a, b]$ is equal to $a - b \lfloor \frac{a}{b} \rfloor$ [23], where $\lfloor q \rfloor$ is the integer floor of some value q . The final time slot assigned to edge $K-1$ is:

$$\begin{aligned} & \text{mod}[K(K-1)-1, 2K] \\ &= K(K-1)-1 - 2K \left\lfloor \frac{K(K-1)-1}{2K} \right\rfloor \\ &= K(K-1)-1 - 2K \left\lfloor \frac{K-1}{2} - \frac{1}{2K} \right\rfloor \end{aligned}$$

Since K is odd, $\frac{K-1}{2}$ is integer; hence, we get:

$$\begin{aligned}
& \text{mod}[K(K-1) - 1, 2K] \\
&= K(K-1) - 1 - 2K \left(\frac{K-1}{2} + \left\lfloor -\frac{1}{2K} \right\rfloor \right) \\
&= K(K-1) - 1 - 2K \left(\frac{K-1}{2} - 1 \right) \\
&= K(K-1) - 1 - K(K-1) + 2K \\
&= 2K - 1
\end{aligned}$$

The final edge ($j = K - 1$) of the longer path is assigned time slots: $(K + 1)$ through $(2K - 1)$. The final edge of the shorter path was assigned time slots 0 to $K - 1$. Hence, when K is odd, this scheduling scheme will not cause interference between adjacent edges, and will achieve an end-to-end flow of $\frac{2K-1}{2K}$.

We next demonstrate a similar result for when K is even; an example network was shown in Figure 9b. The longer path has an even number of edges, and the shorter path has an odd number. Again, on the shorter path, half the time slots are assigned to each edge, alternating between time slots K to $2K - 1$ on the first edge, time slots 0 to $K - 1$ on the second edge, and so forth until the final edge. Since there is an odd number of edges, the final edge of the shorter path entering the destination will be assigned time slots K to $2K - 1$. We now consider the final edge entering the destination of the longer path, which will be assigned time slots $\text{mod}[(K-1)(K-1), 2K]$ through $\text{mod}[K(K-1) - 1, 2K]$. The final time slot for this edge will have the value $\text{mod}[K(K-1) - 1, 2K] = K(K-1) - 1 - 2K \lfloor \frac{K-1}{2} - \frac{1}{2K} \rfloor$. Since K is even, $\frac{K-1}{2}$ is *not* integer, but $\frac{K}{2}$ is; hence, we get:

$$\begin{aligned}
& \text{mod}[K(K-1) - 1, 2K] \\
&= K(K-1) - 1 - 2K \left\lfloor \frac{K-1}{2} - \frac{1}{2K} \right\rfloor \\
&= K(K-1) - 1 - 2K \left(\frac{K}{2} + \left\lfloor -\frac{1}{2} - \frac{1}{2K} \right\rfloor \right) \\
&= K(K-1) - 1 - 2K \left(\frac{K}{2} - 1 \right) \\
&= K^2 - K - 1 - K^2 + 2K \\
&= K - 1
\end{aligned}$$

The final edge of the longer path is assigned time slots 1 through $(K-1)$. The final edge of the shorter path was assigned time slots K to $2K - 1$. Therefore, when K is even, this scheduling scheme will not cause interference between adjacent edges at the destination, and will achieve an end-to-end flow of $\frac{2K-1}{2K}$. ■

Using the scheme described above, the minimum throughput that can be guaranteed on a pair of node-disjoint paths with an odd number of edges is $\frac{5}{6}$, which is greater than the $\frac{2}{3}$ flow described in Section III-A. The minimum guaranteed throughput of $\frac{5}{6}$ is independent of K .

Lemma 9. *For a pair of node-disjoint paths, a schedule can always be found that guarantees a flow of at least $\frac{5}{6}$ from the*

source to the destination.

Proof: We consider only the case when the source and destination are more than one hop apart; otherwise, only one edge needs to be activated between the two nodes, carrying the maximum flow of 1 without the use of node disjoint paths.

When there are an even number of edges over the two node-disjoint paths, then the maximum flow of 1 can be achieved using two time slots, as was shown in Lemma 3.

When there are an odd number of edges over the two node-disjoint paths, where the longer path has K edges, then a flow of $\frac{2K-1}{2K}$ can always be achieved, which was demonstrated in Lemma 8. We now show that the minimum K is 3, hence the minimum flow is $\frac{5}{6}$. Since the source and destination are more than one hop apart, the minimum number of edges over both paths is 4. With 4 total edges, the two paths have an even number of edges, and a maximum flow of 1 is achievable using two time slots. The next smallest number of edges for both paths is 5. Since the source and destination cannot be one hop apart, this means the longer path has 3 edges, and the shorter has 2. Hence, the smallest value of K possible is 3, which gives an achievable throughput of $\frac{2K-1}{2K} = \frac{5}{6}$. Any value of K that is greater than 3 will result in a higher achievable throughput. ■

If only a pair of node-disjoint paths exist with an odd number of edges between the source and destination, a flow of $\frac{2K-1}{2K}$ can be found using $2K$ time slots. But this does preclude the possibility of higher feasible throughputs existing that use additional edges. Consider the example in Figure 10.

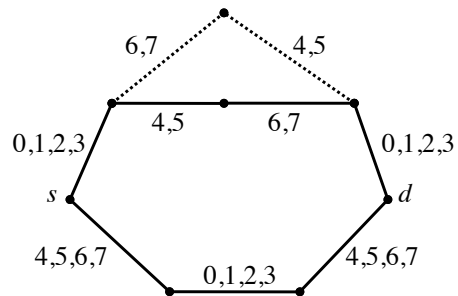


Fig. 10: Node-disjoint paths with additional edges supporting a flow of 1

The pair of node-disjoint paths between nodes s and d are shown using the solid edges, and additional edges that connect with one of the node-disjoint paths are shown using the dotted edges. In this network, all possible pairs node-disjoint paths have an odd number of edges. The longer path has 4 edges; if we scheduled according the scheme described earlier, a flow of $\frac{7}{8}$ can be achieved between s and d . But by using the dotted edges, in addition to the solid edges, a schedule can be found that achieves the maximum flow of 1, as shown in Figure 10. This shows that the flow $\frac{2K-1}{2K}$ is strictly a lower bound on the maximum flow that can be found when only a pair of node-disjoint paths exist that have an odd number of edges. We do not currently consider the problem of determining whether or

not a maximum flow of 1 exists in this case, and we leave it as future work.

C. Proof for Theorem 2

Theorem 2: Finding the minimum length schedule to route a set of demands with protection under 1-hop interference constraints without flow splitting is NP-Hard.

Proof: We prove the protection version of the problem to be NP-hard by reducing the non-protection version to it. We begin by taking a graph $G = (V, E)$ and transforming it to some other graph $G' = (V', E')$. Graph G' will be constructed such that any feasible solution in G using two time slots for node-disjoint paths for a set of node-disjoint demands (i.e., a solution for the non-protection problem in G) will have an immediate solution that includes protection using two time slots for the same set of node-disjoint demands.

Consider an edge $\{i, j\}$ of some path between s and d in G , which is node-disjoint from any other path and is scheduled to use time slot 1. Three edges, $\{i, k\}$, $\{k, l\}$, and $\{l, j\}$, can be added to protect $\{i, j\}$ without needing any additional time slots, with time slot assignments as shown in Fig. 11. Edge $\{i, k\}$ and $\{l, j\}$ are assigned time slot 1, and since they will only be activated after the failure of edge $\{i, j\}$, they do not conflict with the time slot assignment for $\{i, j\}$. Furthermore, the edge on the path in G directly preceding and directly following edge $\{i, j\}$ will be time slot 2 (because the schedule only consisted of two time slots); so, after $\{i, j\}$ fails, the protection routing of $\{i, k\}$, $\{k, l\}$, and $\{l, j\}$ will not interfere with the existing scheduled edges of the path. Additionally, since we are currently considering a feasible solution of node-disjoint paths for the set of node-disjoint demands, no node will have more than a single path crossing it, so the protection path of $\{i, k\}$, $\{k, l\}$, and $\{l, j\}$ will not interfere with any other demands.

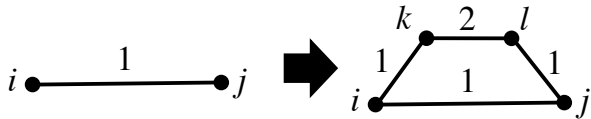


Fig. 11: Edge transformation for NP-hardness proof

To begin, it is clear that any solution for a non-protection routing in two time slots for the set of node-disjoint demands in G will immediately give a protection routing and schedule using two time slots in G' for the same set of demands. We now consider the other direction: for a set of node-disjoint demands, does a solution that uses two time slots for the protection problem on G' give a solution for the same set of demands using two time slots for the non-protection problem in G ? If the protection problem returns a solution for a routing and scheduling using two slots, then that means that before any failure, and after any failure, the set of node-disjoint demands can be routed in two time slots. So, if a routing and schedule is found, then we take the route and schedule from before a failure (which is in two time slots), and transfer any flow that may have been routed onto $\{i, k\}$, $\{k, l\}$, and $\{l, j\}$ to edge $\{i, j\}$, with

$\{i, j\}$ having the same time slot assignment as $\{i, k\}$. Because of how our transformation was performed, this will always yield a feasible solution for the set of node-disjoint demands in G . In general, it is not necessarily the case that no solution to the protection problem indicates no solution to the non-protection problem. But for our particular graph transformation, this is the case; we know that if a two time slot solution exists in G , then a protection routing must exist that uses two time slots. Hence, if the minimum schedule to the protection problem for the set of node-disjoint demands in G' uses more than two time slots, then the schedule for the set of demands in G must use more than two time slots.

We complete the proof by noting that our problem is clearly in NP, and that the graph transformation of G to G' can be accomplished in polynomial time. ■

D. Proof for Theorem 3

Theorem 3: For an incoming connection between s and d , using disjoint paths to provide protection in a wireless network with 1-hop interference constraints for the set of time slots \mathcal{T} is NP-Complete.

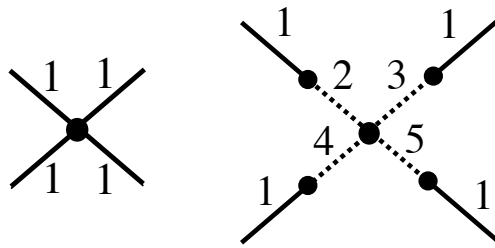
Proof: We reduce the Dynamic Shared-Path Protected Lightpath-Provisioning (DSPLP) [2] to our problem. We first begin by giving details of DSPLP.

DSPLP has the following set of parameters: W is the set of possible wavelengths on any link. $L - 1$ paths are routed, where (w_i, b_i) is the i^{th} working and backup path, respectively. The question DSPLP asks is: does there exist a (w_L, b_L) from s to d that satisfies shared path protection constraints? Those constraints being: (1) w_L and b_L are link disjoint; (2) w_L and w_i , $1 \leq i < L$, do not utilize same wavelength on any common link; (3) w_L and b_i , $1 \leq i < L$, do not utilize same wavelength on any common link; (4) b_L and b_i , $1 \leq i < L$, can share a wavelength on a common link if w_L and w_i are link disjoint. The following is provided by DSPLP: graph G with vertices and edges (V, E) ; W is the set of possible wavelengths on any link. λ_{ij} is the set of wavelengths used on edge $\{i, j\}$ for primary paths; λ_{ij}^{kl} is the set of wavelengths used on edge $\{i, j\}$ to protect against the failure of $\{k, l\}$. $T = |W|$. More simply put, some new incoming demand that needs a disjoint primary and protection path, can share backup resources with some other demand if the two primary paths are failure disjoint.

There is a clear parallel between the wavelength multiplexing scheme that DSPLP is based on and our wireless protection scheme that uses time slots: time slots used for routing/scheduling on a link are similar to wavelengths used on a link for routing and protection. In [2], NP-Completeness is shown for a network with $T = 1$; hence, it is sufficient for us to demonstrate that if our problem can solve an instance of DSPLP with only one wavelength, our problem is also NP-Complete.

If wavelengths are considered as timeslots, they will interfere with one another. Clearly more than one time slot must exist in order to find a feasible routing and schedule in a wireless network with 1-hop inference constraints. So, we “extend” a “new” link from each node, and we increase T (the number of time

slots in the wireless network) such that there exists sufficient time slots to change existing paths that use one wavelength in the original network into interference free schedules using the “new” edges in G^W . It is easy to see that the number of new time slots that need to be added will be the maximum node degree of the network. An example is shown in Fig. 12. The node degree is 4, and each edge has a path routed on it using the existing wavelength. We extend “new” edges out of the node, and increase the number of time slots to any value above 5. Now an interference free schedule can be assigned to allow the edges that used wavelength 1 (now time slot 1) to continue routing those paths using time slot 1.



(a) Node in DSPLP network (b) Node with “new” edges

Fig. 12: Time slot assignment for extended “new” edges

We modify the network G to G^W in the following manner. We “extend” a “new” link from each node, and increase T^W (the number of time slots in the wireless network) by some “large enough” value, such that the “new” edges in G^W will never interfere with one another, or with existing demands using wavelength/time slot 1. “Old” links in G^W that used wavelength 1 to support a lightpath in G will have no available time slots in G^W . Since the primary links using wavelength 1 in G will no have no free time slots in G^W , every future incoming demand in G^W will be edge-disjoint from the existing primary demand that use wavelength/time slot 1, and hence can share backup capacity with existing demands. All links in G that use wavelength 1 for protection will have only one free time slot (time slot 1) available for use in G^W , and only as the backup path for a future demand in G^W . All other “old” links in G^W will have only one time slot available, time slot 1, available for use for the primary or protection path.

Consider some new incoming demand in G^W that needs to be routed and scheduled with a disjoint primary and protection path. The way the network G^W was constructed will ensure that if a solution exists for the new demand, then a solution exists for DSPLP in G . The new demand in G^W will only be able to use time slot 1 on the “old” links, which is wavelength 1. Any solution for wireless protection using disjoint paths in G^W can be converted to a solution for DSPLP in G by removing the “new” edges. If no solution exists for wireless protection using disjoint paths in G^W , then it is clear that no solution exists for DSPLP in G . It is also clear that any solution for DSPLP in G will solve wireless protection using disjoint paths in G^W (with the trivial addition of the “new” edges).

To complete the proof, we note that our problem is clearly in NP. ■

REFERENCES

- [1] S. Ramamurthy, L. Sahasrabudde, and B. Mukherjee, “Survivable WDM Mesh Networks,” *Journal of Lightwave Technology*, vol. 21, no. 4, p. 870, 2003.
- [2] C. Ou, J. Zhang, H. Zang, L. Sahasrabudde, and B. Mukherjee, “New and Improved Approaches for Shared-Path Protection in WDM Mesh Networks,” *Journal of Lightwave Technology*, vol. 22, no. 5, 2004.
- [3] B. Mukherjee, “WDM Optical Communication Networks: Progress and challenges,” *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1810–1824, 2000.
- [4] W. Yao and B. Ramamurthy, “Survivable Traffic Grooming with Path Protection at the Connection Level in WDM Mesh Networks,” *Journal of Lightwave Technology*, vol. 23, no. 10, p. 2846, 2005.
- [5] A. Saleh and J. Simmons, “Evolution Toward the Next-Generation Core Optical Network,” *Journal of Lightwave Technology*, vol. 24, no. 9, p. 3303, 2006.
- [6] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, “Highly-resilient, energy-efficient multipath routing in wireless sensor networks,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 4, pp. 11–25, 2001.
- [7] N. Ahmed, S. Kanhere, and S. Jha, “The holes problem in wireless sensor networks: a survey,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 2, pp. 4–18, 2005.
- [8] M. Kodialam and T. Nandagopal, “Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem,” in *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM, 2003, pp. 42–54.
- [9] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, “Impact of interference on multi-hop wireless network performance,” *Wireless networks*, vol. 11, no. 4, pp. 471–487, 2005.
- [10] M. Alicherry, R. Bhatia, and L. Li, “Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks,” in *Proceedings of the 11th annual international conference on Mobile computing and networking*. ACM, 2005, pp. 58–72.
- [11] B. Hajek and G. Sasaki, “Link scheduling in polynomial time,” *Information Theory, IEEE Transactions on*, vol. 34, no. 5, pp. 910–917, 1988.
- [12] W. Wang, Y. Wang, X. Li, W. Song, and O. Frieder, “Efficient interference-aware tdma link scheduling for static wireless networks,” in *Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006, pp. 262–273.
- [13] J. Zhang, H. Wu, Q. Zhang, and B. Li, “Joint routing and scheduling in multi-radio multi-channel multi-hop wireless networks,” in *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on*. IEEE, 2005, pp. 631–640.
- [14] R. Gupta, J. Musacchio, and J. Walrand, “Sufficient rate constraints for qos flows in ad-hoc networks,” *Ad Hoc Networks*, vol. 5, no. 4, pp. 429–443, 2007.
- [15] X. Lin and S. Rasool, “A distributed joint channel-assignment, scheduling and routing algorithm for multi-channel ad-hoc wireless networks,” in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 1118–1126.
- [16] V. Kumar, M. Marathe, S. Parthasarathy, and A. Srinivasan, “Algorithmic aspects of capacity in wireless networks,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 133–144.
- [17] G. Sharma, R. Mazumdar, and N. Shroff, “On the complexity of scheduling in wireless networks,” in *Proceedings of the 12th annual int’l conf. on Mobile computing and networking*. ACM, 2006, pp. 227–238.
- [18] G. Sharma, C. Joo, N. Shroff, and R. Mazumdar, “Joint congestion control and distributed scheduling for throughput guarantees in wireless networks,” *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 21, no. 1, p. 5, 2010.
- [19] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, New Jersey, 1993.
- [20] U. Derigs, “An efficient dijkstra-like labeling method for computing shortest odd/even paths,” *Information processing letters*, vol. 21, no. 5, pp. 253–258, 1985.
- [21] J. Suurballe and R. Tarjan, “A quick method for finding shortest pairs of disjoint paths,” *Networks*, vol. 14, no. 2, 1984.
- [22] M. Garey and D. Johnson, *Computers and intractability: A Guide to the Theory of NP-Completeness*. Freeman San Francisco, CA, 1979.
- [23] D. E. Knuth, *Art of Computer Programming Volume 1: Fundamentals Algorithms*. Addison-Wesley Publishing Company, 1972.