# Distributed Partial Information Management (DPIM) Schemes for Survivable Networks - Part I

Chunming Qiao Dahai Xu
Department of Computer Science and Engineering
State University of New York at Buffalo
{qiao, dahaixu}@cse.buffalo.edu

***Abstract*— This paper describes a novel framework, called Distributed Partial Information Management (or *DPIM*). It addresses several major challenges in achieving efficient shared path protection under distributed control with only *partial* information, including (1) how much partial information about existing active and backup paths (or APs and BPs respectively) is maintained and exchanged; (2) how to obtain a good estimate of the bandwidth needed by a candidate BP, called BBW, and subsequently select a pair of AP and BP for a connection establishment request so as to minimize total bandwidth consumption and/or maximize revenues; (3) how to distributively allocate minimal BBW (and deallocate maximal BBW) via distributed signaling; and (4) how to update and subsequently exchange the partial information.**

**A DPIM-based scheme using Integer Linear Programming is described to illustrate our approach. In addition, an ultra-fast and efficient heuristic scheme is described. With about the same amount of partial information, such a heuristic-based DPIM scheme can achieve almost as a good performance as the ILP-based DPIM scheme, and a much better performance than another ILP-based scheme described in [1]. The paper also presents an elegant method to support dynamic requests for protected, unprotected, and pre-emptable connections in the unified DPIM framework.**

## I. Introduction

It has long been recognized that connection-oriented services are useful for providing Quality of Services (QoS) to mission-critical applications. In this work, we focus on an important traffic engineering problem, which is dynamic establishment and release of bandwidth guaranteed connections from ingress nodes to egress nodes in a Generalized Multi-Protocol Label Switched network.

Another crucial, and closely related traffic engineering problem is efficient allocation of spare capacity to provide survivability. This is because as one relies more and more on information exchanged through networks, avoiding prolonged disruptions to information exchange due to unexpected failures of network equipment (e.g., link or node) becomes increasingly important.

A common approach to protecting a (bandwidth guaranteed) connection carrying critical information from a single link failure[1], termed *path protection*, is to use a link-disjoint pair of active path (AP) and backup path (BP) from an ingress node to an egress node. This work focuses on path protection as it can achieve fast restoration and be bandwidth efficient in a mesh network by exploiting *shared protection*, whose concept is illustrated using the following example. Assume that two connections, requiring $w_1$ and $w_2$ units of bandwidth, respectively, are established using two link disjoint APs. Since the two APs cannot be broken at the same time due to a single link failure, their corresponding BPs need not be "activated" at the same time either. Hence, if the two corresponding BPs use the same link $e$, they can share the backup bandwidth (BBW) without affecting the survivability of either connection. More specifically, with shared protection, the total BBW that needs be allocated on link $e$ (for the two BPs) is $\max\{w_1, w_2\}$, instead of $w_1 + w_2$ without BBW sharing as in the so-called *No-Sharing* (NS) scheme [1]. Hereafter, the term "connection" refers to protected as well as bandwidth guaranteed connection unless otherwise specified.

In this work, we will study an *on-line* case where not all requests for connection establishment and release arrive at the same time, and a decision as to how to satisfy a request (if possible at all) has to be made without knowing about any future requests and, for the sake of guaranteed QoS, *without* being able to rearrange the way existing connections are established.

In such an on-line case, a common objective is to allocate (or deallocate) a minimal (or maximal) total bandwidth or TBW, which is the sum of BBW allocated on all BPs and the bandwidth allocated on all APs, hereafter called ABW, when satisfying each request for connection establishment (or release). Although on each link used by an AP, a fixed amount of ABW (say $w$ units) is to be allocated (or deallocated), the amount of BBW to be allocated (or deallocated) in any shared path protection schemes depends on many factors including which links are used by the corresponding AP. This is why *shortest pair of path* (or SPP) algorithms such as the one in [2] can no longer guarantee minimum TBW allocation for a given connection establishment request, let alone those based on the so-called *active path first* or APF heuristic [3]).

In [1], an approach which uses Integer Linear Programming (ILP) to determine a pair of AP and BP, called *Sharing with Complete Information* (or SCI), was proposed. While SCI leads to maximal BBW sharing and maximal improvement (about 37%) over the NS scheme in terms of TBW consumption, a controller needs to maintain $O(L \cdot Q)$ *complete per-flow* information, where $L$ is the average path length (in terms of the number of links) and $Q$ the average number of existing con-

[1]The problem of protection against a single node failure can be transformed to that of protection against a single link failure by splitting each node into two halves with a "virtual" directed link in between.

nections, which could be thousands or more. Accordingly, SCI is not suitable for implementation under distributed control as the signaling overhead involved in exchanging that amount of information among distributed controllers is just too much. Implementing SCI under centralized control is also a challenge, especially if requests for connection establishment or release arrive frequently as driven by some emerging applications. This is because a centralized controller can easily become a performance bottleneck that also limits the network's scalability. In addition, if the centralized controller fails, the entire network will be down.

To address the above deficiencies of SCI, [1] proposed another scheme based on an ILP formulation called *Sharing with Partial Information* (or SPI). SPI requires each controller (at an edge node) to maintain only $O(E)$ partial (and aggregated) information, where $E$ is the number of links in a network, but results in a much lower improvement (namely about 16%) over NS. A different approach, which uses the APF heuristic to determine a pair of AP and BP, called *Survivable Routing* or SR, was proposed in [3]. SR requires each edge node to maintain $O(E^2)$ complete *but* aggregated information to achieve a near-optimal improvement (about 36%) over NS, and thus its scalability is still limited,

In this paper, we propose two new distributed control schemes under what we call the *distributed partial information management* or DPIM framework, with one using an ILP formulation (mainly for the purpose of comparison), and the other using the APF heuristic. In a nutshell, the novelty of the DPIM framework is that the $O(E^2)$ complete and aggregated information maintained by each node in SR is now partitioned among all the nodes in a network, and thus each node only maintains (and uses) $O(E)$ partial information.

A salient feature of the proposed schemes is their ability to allocate (and deallocate) *minimal* (and *maximal*) BBW allocation on a chosen BP even though only partial information is available at each node, which is impossible in SPI. As a result, they can achieve remarkable improvement over SPI (their improvements over NS are 28% and 26% respectively).

The rest of the paper is organized as follows. Section II contains the notation to be used throughout the paper. It also describes closely related prior work. Section III describe the two DPIM schemes including what partial information is needed and how distributed routing is performed. Section IV addresses another important integral part, which is how distributed signaling for connection establishment and release is performed, and in particular, how partial information is updated and exchanged. Section V presents the performance evaluation model used as well as numerical results of the comparison between the proposed DPIM schemes and a few existing approaches. Section VI discusses how to support unprotected and pre-emptable connections. Section VII distinguish additional related work from the proposed DPIM schemes. Section VIII summarizes our contributions.

## II. Notation and Prior Work

In this section, we first present the notation to be used in the paper and then discuss closely related prior work.

### A. Notation

We consider a network $G$ with $E$ directed links (represented by set $\mathcal{E}$) and $V$ nodes, which can be classified into two categories: *edge* nodes (ingress or egress), to which users or terminal devices are connected, and *core* nodes (which are nodes other than an edge node).

To facilitate our presentation, we will use a tuple $(s \rightarrow d, w)$ to represent a new request for connection establishment (or release), where $s$ and $d$ are the source (or ingress) and destination (or egress) of the connection, respectively, and $w$ is the bandwidth (in units) requested by the connection.

The following additional notation will be used, where a calligraphic font style (e.g., $\mathcal{A}$ is used to denote a set or a vector while a non-calligraphic style (e.g., $A$) is used to denote a scalar value:

- $\mathcal{OUT}(n), \mathcal{IN}(n) \subset \mathcal{E}$: Set of links going from and coming into node $n \in \mathcal{V}$, respectively.
- $\mathcal{AP}$ and $\mathcal{BP}$: Set of links along an AP and BP, respectively.
- $\mathcal{A}_e$: Set of connections whose APs traverse link *e*.
- $A_e = \sum_{k \in \mathcal{A}_e} w_k$: Total (i.e., aggregated) ABW on link *e* dedicated to the connections in $\mathcal{A}_e$.
- $\mathcal{B}_e$: Set of connections whose BPs traverse link *e*.
- $B_e$: Total BBW allocated on link *e* for $\mathcal{B}_e$. Due to BBW sharing, $B_e \leq \sum_{k \in \mathcal{B}_e} w_k$.
- $R_e$: Residue bandwidth of link $e$. Its initial value is equal to the capacity of link $e$, $C_e$. $R_e = C_e - A_e - B_e$ (with only protected connections).
- $\mathcal{S}_a^b = \mathcal{A}_a \bigcap \mathcal{B}_b$: Set of connections whose APs traverse link *a* and whose BPs traverse link *b*, where $a, b \in \mathcal{E}$.
- $S_a^b = \sum_{k \in \mathcal{S}_a^b} w_k$: Total amount of bandwidth required by the connections in $\mathcal{S}_a^b$. It is a fraction of $\mathcal{A}_a$ as well as $\mathcal{B}_b$ that is used by the APs and BPs, respectively, of the connections in $\mathcal{S}_a^b$.
- $BC_a^b$: *Additional BBW* needed on link $b$ in order to use it as a part of a BP for a new connection whose AP traverses link *a*. Its value depends on which BBW estimation method is used.

While most of the above notation are similar to those used in [1], the following notation are *specific* to the proposed DPIM schemes:

- $BC_e$: Estimated BBW needed on link $e$ along a new BP. Assuming that its corresponding AP is known, $BC_e = \max_{\forall a \in \mathcal{AP}} B_a^e$. Whether this value is the minimum BBW needed on link $b$ or not depends on which BBW estimation method is used to derive $B_a^e$. In addition, this is equal to the actual BBW allocated on link $b$ in all the schemes mentioned so far except the proposed DPIM schemes (which may result in an over-estimation but always allocates the minimal BBW).
- $\mathcal{P}_{\mathcal{B}}(e) = \{S_a^e | a \in \mathcal{E}\}$: *Profile* of BBW on a given link $e$. This is a *vector* consisting of a list of $S_a^e$ values, one for each link $a$. Basically, it specifies the amount of BBW on link $e$ that is used to protect against the failure of every other link (e.g., $a_1, a_2 \cdots a_E \in \mathcal{E}$) in the network.

- $P_{Be} = \max_{\forall a} S_a^e$: This is the maximum value over all the components in $\mathcal{P_B}(e)$. It is also the minimum (or *necessary*) amount of BBW needed on link *e* to backup all active paths. If a BBW allocation scheme (such as the DPIM schemes to be described) always allocates minimum BBW on link $e$, then $B_e = P_{Be}$.
- $\mathcal{P_A}(e) = \{S_e^b | b \in \mathcal{E}\}$: *Profile* of ABW on a given link $e$. This is a *vector* consisting of a list (or set) of $S_e^b$ values, one for each link $b$. It complements $\mathcal{P_B}(e)$, and specifies the amount of ABW on link $e$ that is protected by every link (e.g., $b_1$, $b_2 \cdots b_E \in \mathcal{E}$) in the network.
- $P_{Ae} = \max_{\forall b} S_e^b$: This is the maximum value over all the components in $\mathcal{P_A}(e)$. It is also the *sufficient* amount of bandwidth that needs be reserved on any link in the network in order to protect against the failure of link $e$.

### B. Prior Solutions

In this subsection, we summarize three closely related schemes for shared path protection, namely, SCI, SPI and SR (mentioned in Sec. I). Some other related work will be described in Sec. VII.

Since *no prior work* has provided any detail on distributed signaling for establishing connections (let alone releasing connections), the following description of SCI, SPI and SR will be limited to mainly how routing is performed in each scheme.

*1) SCI: Sharing with Complete Information:* In this scheme, the centralized controller maintains complete *per-flow* information on all existing APs and BPs in a network. More specifically, for every link $e \in \mathcal{E}$, both $\mathcal{A}_e$ and $\mathcal{B}_e$ are maintained in addition to $R_e$. Note that, based on such information, all other parameters described in Sec. II-A and in particular $S_a^b$ for every link $a$ and link $b$ (i.e., all combinations) can be derived. It turns out that such complete but aggregated information is what SCI really needs.

In SCI, the problem of minimizing TBW needed (i.e., jointly optimizing the selection of an AP and an BP) to satisfy a new connection establishment request is solved based on the following Integer Linear Programming (ILP) formulation (which is slightly modified from [1] for easier understanding).

Assume that the AP and BP for a new connection establishment request $(s \rightarrow d, w)$ will traverse links $a$ and $b$, respectively. Since the total BBW that needs be reserved on link $b$ is $S_a^b + w$, and the BBW already reserved on link $b$ is $B_b$ (which is sharable), the additional BBW needed on link $b$ to protect against the failure of link $a$ is basically $\max\{S_a^b + w - B_e, 0\}$. Specifically, we have

$$BC_a^b = \begin{cases} \infty & \text{if } a = b \text{ or } R_a < w \text{ or} \\ & \quad R_b < S_a^b + w - B_b \quad \text{(i)} \\ 0 & \text{else if } S_a^b + w \le B_b \quad \text{(ii)} \\ S_a^b + w - B_b & \text{else if } S_a^b + w > B_b \text{ and} \\ & \quad S_a^b + w - B_b \le R_b \quad \text{(iii)} \end{cases} \tag{1}$$

To facilitate the ILP formulation, assume that $\mathcal{AP}$ and $\mathcal{BP}$ are implemented with two bit-maps with $E$ components each. Let $\mathcal{AP}_e$ be set to 1 if link $e$ is used in the AP and 0 otherwise. Clearly, on link $e$ whose $\mathcal{AP}_e = 1$ in a solution obtained by ILP, $w$ units of additional ABW need be dedicated. Similarly, let $\mathcal{BP}_e$ be 1 if link *e* is used on the BP and 0 otherwise. Accordingly, for any link $e$,

$$BC_e = \max_{\forall a} BC_a^e(\mathcal{AP}_a + \mathcal{BP}_e - 1) \tag{2}$$

Note that $BC_e \ge 0$ is the minimum additional BBW needed on link $e$. It is also the actual BBW to be allocated on link $e$.

The objective of the ILP formulation is to determine a pair of AP and BP (or equivalently, $\mathcal{AP}$ and $\mathcal{BP}$) such that the following cost function is minimized:

$$w \cdot \sum_{e \in \mathcal{E}} \mathcal{AP}_e + \sum_{e \in \mathcal{E}} BC_e \tag{3}$$

subject to the following constraints which are self-explaining:

$$\sum_{e \in \mathcal{OUT}(n)} \mathcal{AP}_e - \sum_{e \in \mathcal{IN}(n)} \mathcal{AP}_e = \begin{cases} 1 & n = s \quad \text{(i)} \\ -1 & n = d \quad \text{(ii)} \\ 0 & n \ne s, d \, \text{(iii)} \end{cases} \tag{4}$$

$$\sum_{e \in \mathcal{OUT}(n)} \mathcal{BP}_e - \sum_{e \in \mathcal{IN}(n)} \mathcal{BP}_e = \begin{cases} 1 & n = s \quad \text{(i)} \\ -1 & n = d \quad \text{(ii)} \\ 0 & n \ne s, d \, \text{(iii)} \end{cases} \tag{5}$$

$$\mathcal{AP}_e + \mathcal{BP}_e \le 1 \quad \textit{and} \quad \mathcal{AP}_e, \mathcal{BP}_e \in \{0,1\} \tag{6}$$

As mentioned earlier, such a scheme allows the new BP to share maximum BBW with some existing BPs but has two major drawbacks which make it non-scalable. One is the huge amount of information to be maintained, and the other is the extremely high computational overhead involved in solving the ILP formulation, which makes it especially unsuitable for the on-line situation.

*2) SPI: Sharing with Partial Information:* In this scheme, only the values of $A_e$ and $B_e$ (in addition to $R_e$) for every link $e$ are maintained by the central controller. An ILP formulation similar to the one described above can be used. More specifically, it was suggested in [1] that $S_a^b$ be replaced by $A_a$ in Eq. 1 used to determine $BC_a^b$. In essence, if an AP were already chosen (as in the case where APF heuristic is used), the estimated backup cost on link $b$ becomes

$$BC_b = \max\{\max_{\forall a \in \mathcal{AP}} (A_a + w - B_b), 0\} \tag{7}$$

Note that, the additional BBW so estimated, which, in SPI, is also the same as the actual BBW (or backup cost) to be allocated for link $b$, is usually larger than the minimum needed because $A_a \ge S_a^b, \forall b$.

While the ILP formulation takes as much time to solve as in SCI, a quicker method which obtains a near-optimal solution (with respect to the solution obtained by the ILP formulation for SPI) in about 1 second per request in a 70-node network was suggested in [1]. The main deficiency of SPI, as mentioned earlier, is that it achieves a lower improvement over NS when compared to SCI as a price paid for maintaining only partial information.

*3) Survivable Routing (SR):* In the so-called SR scheme [3], instead of maintaining complete per-flow (or equivalently per-path) information as in SCI, complete aggregate (or equivalently per-link) information is maintained. More specifically, in SR, every node essentially maintains a matrix of $S_a^b$ for all links $a$ and $b$, as well as other necessary information. Also, SR uses the active path first (APF) heuristic instead of ILP formulation to determine a pair of paths. More specifically, for every connection establishment request, an AP (with a minimal number of links whose $R_e > w$) is found first using a shortest path algorithm. Then, the links used by the AP is removed, and each remaining link $e$ is assigned a cost of $BC_e = \max_{\forall a \in \mathcal{AP}} (S_a^e + w - B_e)$ (which is similar to Eq. 2), and those whose $R_e < BC_e$ are then removed as well. Thereafter, a *cheapest* BP is chosen, and each link $b$ on the BP is allocated an amount of additional BBW equal to $BC_b$ (the estimated backup cost).

The main deficiency of SR is the $O(E^2)$ information needed, which limits its scalability. In fact, in a wavelength-routed Wavelength Division Multiplexed (WDM) network where each connection (called lightpath or wavelength path) occupies an entire wavelength channel on a link it spans, maintaining $O(E^2)$ information is not much better than maintaining complete per-flow information (i.e., $\mathcal{A}_e$ and $\mathcal{B}_e$) as in SCI. This is because the expected maximum number of lightpaths in a WDM network with $K$ wavelengths on each of its E directed links is about $Q = \frac{K \cdot E}{L}$ (where $L$ is the average path length). Hence, the amount of information that needs be maintained in SCI is $O(Q \cdot L) = O(KE)$, which is the same as $O(E^2)$ given that $K$ is on the same order as E (e.g., a few hundreds).

A variation of SR, called Successive SR or SSR can achieve a better BBW sharing than SR. The main difference between SR and SSR is that, in the latter, some existing BPs may change, not only in the way they are routed but also the amount of additional BBW reserved for them, after the matrix $S_a^b$ is updated as a result of setting up a new connection. Such changes may in turn trigger changes to other existing BPs until an equilibrium state is reached. This iterative process involving changes in existing BPs introduces a high signaling and control overhead, especially under distributed control. This is the main reason that we will focus on schemes that *do not* require existing BPs to change (but the proposed DPIM-SAM schemes can also be extended to allow BPs to change in order to make the tradeoffs between a high complexity and an improved BBW sharing).

## III. The Proposed DPIM Schemes

In this section, we describe the proposed DPIM schemes, and in particular what $O(E)$ partial information is maintained and exchanged, and how a pair of AP and BP is determined. Distributed signaling in the DPIM schemes to support minimal BBW allocation and maximal BBW deallocation will be described in the next section.

### A. Distributed Partial Information Management

We will call the first scheme DPIM-SAM (based on ILP) since it performs both *Sufficient* and *Aggressive* estimation (mentioned in Sec. III-C) of the additional BBW needed for each candidate link on a BP (in addition to *Minimal BBW allocation* (mentioned in Sec. IV-A)). The second scheme will be simply called DPIM-M-A as it does not perform any (sophisticated) BBW estimations but is especially suitable for implementation using existing IP routing functions.

A distinct feature of DPIM-SAM, which makes it a truly distributed approach, is that, every node $n$ (i.e., edge or core) maintains *some* information. But only the information on every *local* (and outgoing) link $e$, i.e., $e \in \mathcal{OUT}(n)$, are maintained at each node. More specifically, each node $n$ maintains the following four scalars for each local link $e$: $B_e$, $R_e$, $P_{Ae}$, and $P_{Be}$. Note that the last two scalars can actually be derived from the additional information to be maintained (see below) and thus are for convenience only.

In addition, each node $n$ also maintains the following vectors (profiles of ABW and BBW) for each *local* link $e$: $\mathcal{P}_{\mathcal{A}}(e)$ and $\mathcal{P}_{\mathcal{B}}(e)$. These two profile vectors, (each containing up to $E$ components of $S_e^b$ and $S_a^e$, respectively), contain additional information that is not utilized by SPI. The first profile vector is used to determine an up-to-date value of $P_{Ae}$ (which is useful for an edge node to obtain a good estimate of $BC_e$ on the remote link $e$ as to be described in the next subsection). The second profile is a key to achieving minimal BBW allocation and maximal BBW deallocation. Even with such additional information, the total amount of information maintained at each node is practically limited to $O(E)$ (as the number of local links at each node is typically bounded by a small constant).

At each edge node, the *only* non-local information to be maintained (and exchanged) is three scalars, $B_e$, $R_e$, and $P_{Ae}$, for each *remote* link $e$. Accordingly, the amount of local and remote information that needs be maintained is still limited to $O(E)$.

DPIM-M-A requires even less information and in fact, only minimum remote information. More specifically, no local or none-local information on $P_{Ae}$ (or $\mathcal{P}_{Ae}$) is needed at any node. In addition, the only non-local information an edge node needs to maintain (and exchange) is the scalar, $R_e$, for each remote link $e$. Therefore, DPIM-M-A can be easily implemented using existing OSPF routing protocols.

### B. Distributed Routing

In the discussion below on the proposed DPIM schemes, it is assumed that each request to establish a connection arrives at its ingress node. The ingress node then acts as a decentralized controller, and performs explicit routing to determine a pair of paths (i.e., specifies the entire AP and BP) for the request.

Note that, we assume that each edge node maintains the topology of the entire network by, e.g., exchanging link state advertisements (LSAs) among all nodes (edge and core nodes) as in OSPF. Note that, non-local information may also be either multicast to all edge nodes using dedicated signaling protocols, or broadcast to all nodes with extended LSAs.

Note also that in a heavily-loaded network with a limited link capacity, when an ingress node fails to find a suitable pair of paths because of insufficient residual bandwidth for example, the connection establishment request will be rejected (such a request, if submitted after some existing connections have been released, may be satisfied).

### C. Path Determination

We first describe an ILP formulation to determine a pair of AP and BP for a connection establishment request using DPIM-SAM, which is similar to that described earlier for SPI in Section II-B.2, but with several improvements.

First, we observe that with the information on $P_{A_a}$, the additional BBW needed on link $b$ when link $a$ is used by AP is at most $P_{A_a} + w - B_b$ (called *Aggressive* estimation of the additional BBW), which is smaller than $A_a + w - B_b$ as estimated in SPI. Secondly, the additional BBW is also bounded by $w$ (called *Sufficient* estimation), and hence, one can replace $S_a^b + w - B_b$ in Eq. 1 with $\min\{P_{A_a} + w - B_b, w\}$ . In other words, assuming the AP is known, the backup cost on link $b$ can be estimated as

$$BC_b = \min\{\max_{\forall a \in \mathcal{AP}} (P_{A_a} + w - B_b), w\} \tag{8}$$

which is more accurate than the estimation given in Eq. 7.

One can also improve the objective function in the ILP formulation by using the following instead:

$$w \cdot \sum_{e \in \mathcal{E}} \mathcal{AP}_e + \epsilon \cdot \sum_{e \in \mathcal{E}} BC_e \tag{9}$$

where $\epsilon(< 1)$ may be set to 0.9999 for example. This $\epsilon$ helps an ILP solver to choose a shorter path to use as an AP which will lead to better BBW sharing. More specifically, whenever there are two pairs of paths with the same TBW (with or without BBW sharing), and the AP in the first pair is longer than the AP in the second pair (in which case, the BBW required on the BP in the first pair must be smaller than that in the second pair for the two pairs to have the same TBW), the second pair with the shorter AP will be chosen by the ILP solver for using this $\epsilon$. On the other hand, the first pair with the longer AP may be chosen by the ILP solver if the $\epsilon$ is not used. While our results show that $\epsilon$ yields only slightly better BBW sharing, such a method of assigning less weight to a unit of BBW (which is similar to a collateral payment) than to a unit of ABW (which is similar to a cash payment) are also useful for other applications. Note that, the quick method suggested for SPI [1] can also be adapted in order to speed up the path determination process.

Path determination in DPIM-M-A is much simpler. Just like in SR, an AP (with a minimal number of links whose $R_e > w$) is found first using a shortest path algorithm. The difference between the two is that in DPIM-M-A, after removing all the links along the AP, every remaining link will be assigned a cost of $w$ (instead of the calculated value of the backup cost as in SR). Those links with $R_e < w$ will then be removed, and a shortest path is found for use as the BP. This ultra-fast algorithm takes less than 0.05 seconds for each request in a 70-node network (approximately 20 times faster than the quick method [1]). One alternative to DPIM-M-A is to remove all links with $R_e < w$, and find a shortest pair of paths using the SPP algorithm in [2]. Then, the shorter of the two can be used as the AP and the other as the BP.

Note that, once the BP is chosen, however, minimal BBW ($\leq w$) will be allocated on each link along the BP in the proposed DPIM schemes (as to be described next). This implies that some of the links removed earlier because of their insufficient $R_e < w$ could have used by the BP. Hence, another alternative to DPIM-M-A is to just remove the links along the AP, and find a shortest path for use as the BP. It is possible, however, that a link on the BP so found may not have sufficient $R_e$ (even with minimum BBW allocation). Such a link will have to be removed after the signaling packet to reserving BBW on the BP fails (see the section below and also Sec. IV-D for more discussion), and a new BP will have to be selected. This alternative may be useful to improve the selection of a BP at the expense of a higher signaling complexity and longer BP set-up latency.

## IV. Distributed Signaling

Signaling is an important integral part of any distributed control scheme as it directly affects how the distributively maintained information is updated and exchanged, and consequently the signaling overhead and feasibility/scalability of the distributed control scheme. In this subsection, we describe how distributed signaling is done in DPIM-SAM. Distributed signaling in DPIM-M-A is only slightly different (and in fact, simpler).

The basic idea of distributed signaling is as follows. Once the two paths are determined, the ingress node sends signaling packets to the nodes along the two paths to allocate bandwidth to the two paths. More specifically, let $\mathcal{AP} = \{a_i | i = 1, 2, \cdots p\}$ and $\mathcal{BP} = \{b_j | j = 1, 2, \cdot \cdot \cdot q\}$ be the set of links along the chosen AP and BP, respectively (whose lengths are $p$ and $q$, respectively). Then, an "AP Set-up" packet will be sent to the nodes along the AP to establish the requested connection, which contains a unique connection identifier, the explicit route (e.g., $\mathcal{AP}$), and the bandwidth requested (i.e. $w$) among other information. The connection set-up process may be carried out in any reasonable distributed manner by reserving $w$ unit of bandwidth on each link $a_i \in AP$, creating an switching/routing entry with an appropriate connection identifier (e.g., a label), and configuring the switching fabric (e.g., a cross-connect) at each node along the active path, until the egress node is reached. The egress node then sends back an acknowledgment packet (or ACK). Actions to be taken at each node that are specific to the proposed DPIM schemes will be described in the following subsections.

In addition, a "BBW Allocation" packet will be sent to the nodes along the chosen BP. This packet will contain information similar to that carried by the AP Set-up packet. At each node along the BP, similar actions will also be taken except that the switching fabric will *not* be configured. In addition, the amount of bandwidth to be reserved on each link $b_j \in BP$ may be less than $w$ due to potential BBW sharing, and in fact, it will be a minimal value which may even be less than that estimated by Eq. 8.

To facilitate connection release (to be discussed in Sec. IV-C), the ingress node creates and maintains a record for the connection, which includes the connection identifier, $\mathcal{AP}$ and $\mathcal{BP}$, as well as the requested bandwidth $w$.

### A. BBW Allocation

A naive approach to BBW allocation, which we call *estimation-based* allocation or the E approach, is to include estimated BBW needed on each link $b_j \in BP$ (given by Eq. 8 for example) in the BBW Allocation packet.

Here, we describe a different approach called *minimum bandwidth allocation* or simply the M approach, used by both DPIM-SAM and DPIM-M-A. The basic idea is to let the BBW Allocation packet contain the information on AP (i.e., a linked list $\mathcal{AP}$). Upon receiving this information, each node $n$ that has an outgoing link $e \in BP$ updates the locally maintained profile vector $\mathcal{P}_{\mathcal{B}}(e)$ and $P_{Be}$. More specifically, only $p$ components in $\mathcal{P}_{\mathcal{B}}(e)$ that correspond to links $a_i \in \mathcal{AP}$ (where $p$ is the length of AP), need to increase their values by $w$ (i.e., $S^e_{a_i} = S^e_{a_i} + w$). Also, if the old value of $P_{Be}$ is maintained, the updated value of $P_{Be}$ can be easily obtained as the largest among its old value and the values of the newly updated $p$ components. Hence, only a marginal computing overhead is involved.

Thereafter, the amount of BBW to be allocated on link $e$ is $bw = P_{Be} - B_e \geq 0$. If $bw > 0$, then $B_e$ and $R_e$ are increased and reduced by $bw$, respectively, and the updated values of $B_e$ and $R_e$ are multicast to all ingress nodes.

Since $P_{Be}$ is the necessary (i.e., minimum) BBW needed on link $e$, the M approach will allocate minimal additional BBW on link $e$ each time a BBW Allocation packet is processed. Fig. 1 shows an example in which the M approach outperforms the E approach. In this example, it is assumed that two connections $(1 \rightarrow 4, 2)$ and $(3 \rightarrow 4, 3)$ have been established in sequence as shown in Fig. 1 (a) and (b). In all figures, a numerical number displayed along a link with (and without) a quotation mark denotes the BBW (and ABW), respectively, reserved on that link.

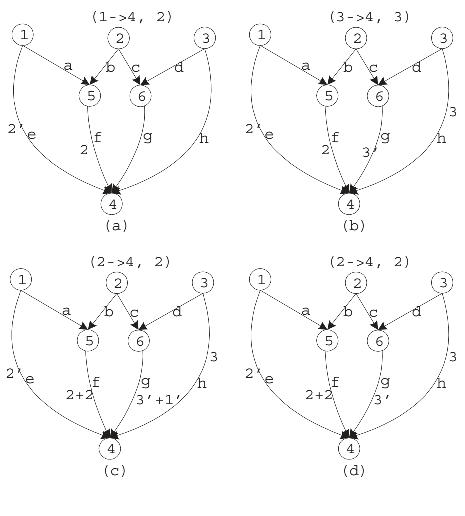

Fig. 1. An illustration of minimal BBW allocation

Now consider a new connection $(2 \rightarrow 4, 2)$ which will use links $f$ and $g$ on the AP and BP, respectively. Since $P_{Af} = 2$ and $B_g = 3$ (prior to the establishment of the connection), using the E approach, one still needs to allocate one (1) additional unit of BBW on link $g$ as shown in Fig. 1(c). However, using the M approach, $P_{Bg}$ is still 3 after establishing the connection, so no additional BBW on link $g$ is allocated as shown in Fig. 1(d).

Note that in the DPIM schemes, a pair of paths is determined by an ingress node based on only partial information. So even with minimal BBW allocation, it is reasonable that they will underperform SCI or SR which has access to complete information, and thus can select a better pair of paths.

### B. Maintaining Partial Information on AP

ABW allocation is straight-forward (as it involves a fixed amount, $w$). However, in order to allow ingress nodes to maintain updated information on $P_{Ae}$ for every link $e$ as in DPIM-SAM for the purpose of path determination, the following additional actions need to be taken as a part of the connection set-up process in DPIM-SAM.

First, the ingress node will send an “AP Set-up” packet carrying the information on the chosen BP to the nodes along AP. Upon receiving such information, each node $n$ that has an outgoing link $e \in AP$ updates the profile vector $\mathcal{P}_{\mathcal{A}}(e)$ as well as $P_{Ae}$ (in much the same way that $\mathcal{P}_{\mathcal{B}}(e)$ as well as $P_{Be}$ are updated on the BP). Thereafter, $R_e$ is reduced by $w$ and the updated values of $R_e$ and $P_{Ae}$ are multicast to all ingress nodes.

In DPIM-M-A, no information on BP needs be carried by the AP Set-up packet, and only the updated $R_e$ along the AP needs be multicast to all ingress nodes.

### C. Connection Release

As in performing distributed signaling to establish a connection, an important issue in dealing with a connection release request is how to update the locally maintained partial information and distribute the updated information among the edge nodes.

In both DPIM schemes, when a connection release request arrives at an ingress node, the ingress node will send an “AP Tear-Down” packet and a “BBW Deallocation” packet to the nodes along the AP and BP, respectively. These two packets will carry the connection identifier, and sent to the first intermediate node along the AP and BP, respectively. Similar to an AP Set-up packet, an AP Tear-down packet will carry $\mathcal{BP}$[2] in DPIM-SAM, but not in DPIM-M-A. Also, similar to a BBW Allocation packet, a BBW Deallocation packet will also carry $\mathcal{AP}$ in both DPIM schemes.

A node along the BP processes the BBW Deallocation packet in much the same way that it processes the BBW Allocation packet, except that here, the $p$ components in $\mathcal{P}_{\mathcal{B}e}$ will be adjusted down (instead of up) by $w$. After $P_{Be}$ is updated, the amount of BBW to be deallocated on link $e$ is $bw = B_e - P_{Be} \geq 0$. If $bw = 0$, no actions other than forwarding the BBW Deallocation packet to the next node along the BP needs be taken. Otherwise, $B_e$ and $R_e$ decreases and increases, respectively, by $bw$. These two updated values are multicast to all edge nodes in DPIM-SAM but only one ($R_e$) in DPIM-M-A.

Similarly, a node along the AP processes the AP Tear-down packet in much the same way that it processes the AP Set-up packet, except that when there was an increase in a value, it should be an decrease instead, and vice versa.

[2]But there is no need for it to carry $\mathcal{AP}$ as the connection-identifier can be used for hop-by-hop forwarding of the AP Tear-Down packet.

### D. *Aborted Connection Establishment*

With distributed signaling, each ingress node may have the most up to date information on e.g., on $R_e$ when it computed the paths for a connection establishment request it received. However, since two ingress nodes will choose their corresponding pair of paths *independently* of each other, they may send out two AP Set-up packets, for example, requesting for $w_1$ and $w_2$ units of ABW on the same link $e$, where $w_1 < R_e$ and $w_2 < R_e$, but $w_1 + w_2 > R_e$. In such a case, one of the AP Set-up packet will fail to proceed (and be dropped), and an negative acknowledgment (or NAK) will be sent back to the its originating ingress node. In addition, any portion of the corresponding AP already established prior to link $e$ will be released. The ingress node, upon receiving the NAK, will also need to deallocate BBW along the corresponding BP. The ingress node may then choose to reject the connection establishment request, or wait until it receives updated information (if any) before trying a possibly different AP (and/or BP).

Similar situations may also occur where a BBW Allocation packet cannot proceed because of insufficient $R_e$ on an outgoing link $e$. In such cases, similar backtracking actions will take place.

## V. Performance Evaluation

We compare the performance of various schemes including SCI, SPI, SR, DPIM-SAM (ILP without using $\epsilon$) and DPIM-M-A. Processing and signaling overheads are ignored in this quantitative comparison study. The performance of the two alternatives to DPIM-M-A discussed earlier are not evaluated here. However, for completeness, we also evaluate the performance of SPI-A, where A is short for APF. SPI-A works the same way as DPIM-M-A except that after an AP is found, each remaining link will be assigned an (actual) cost given by Eq. 7 (instead of an estimated cost of $w$ as in DPIM-M-A) before a cheapest path will be found for use as the BP.

In the rest of the section, we describe the network topology assumed, traffic types considered, and performance metrics used before presenting the results.

### A. *Network Topology*

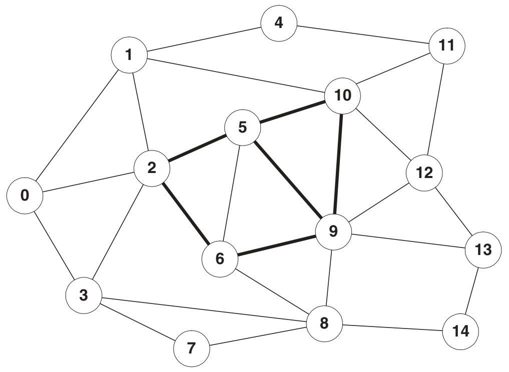

Fig. 2. A 15-node network

To facilitate a fair comparison between our approaches and prior ones, we consider the topology shown in Fig. 2, which is the same as that used in [1] and has 15 nodes and 28 bi-directed edges (for a total of 56 links). The capacity of each link is assumed to be either infinite or limited as to be discussed in the next subsection. Another network with 70 nodes (264 links) is also considered, and it is found that the two networks generate relatively consistent performance results.

### B. *Traffic Types*

We consider two types of traffic, one in which an established connection lasts forever (i.e., incremental traffic) as in [1, 3], and the other in which it may terminate after a certain duration (i.e., dynamic traffic).

In both cases, the ingress and egress of a connection establishment request is evenly distributed among all nodes, and requests arrive in an on-line fashion. For the case with incremental traffic, the bandwidth required by the connections is uniformly distributed between 1 and 10 units as in [1]. Note that, any request arrival process may be assumed.

For the case with dynamic traffic, the bandwidth required varies from 1, 2, 3, 4, 6 and 12 units with probability being 20%, 10%, 30%, 10%, 10%, 20%, respectively. In addition, requests are assumed to arrive according to a Poisson process, and the connection duration has a Pareto distribution. This is just an attempt to model realistic traffic (which may be self-similar and whose bandwidth requirements range from OC-1 to OC-12). Other possibilities, including uniformly distributed bandwidth requirements and exponentially distributed connection durations, have also been examined, and we have found that they have no significant impact on the performance of various schemes studied in this paper.

### C. *Performance Metrics*

The following two performance metrics are used, one for each traffic type considered.

*1) Bandwidth Saving (Ratio):* To obtain this metric, it is assumed that the capacity of each link is infinite (and hence all requests will be satisfied), and the traffic is incremental. After an appreciable number of requests have been satisfied, TBW consumed (i.e., sum of ABW and BBW on APs and BPs, respectively) for each of the schemes is evaluated and consequently, bandwidth saving, in terms of TBW consumption ratio over the NS scheme, is determined as similarly done in [1]. Note that, for a given request, the BBW needed will be no less than the ABW needed in NS. Hence, even if an ideal scheme that achieves maximum BBW sharing is used, the bandwidth saving ratio will be upper-bounded by 50% (achievable only if no BBW is needed at all).

*2) Total Earning (Ratio):* The bandwidth saving measure may not mean much since in a practical case, all links have a finite capacity and thus not all requests can be satisfied.

Accordingly, in this set of experiments (simulation), we assume that each link has a finite capacity and dynamic traffic is considered. For example, in the Fig. 2 above, each dark (bold) link (consisting of two unidirectional links) is assumed to have a capacity of 192 units in each direction (to model an OC-192 link), and each of the other links has a capacity of 48 units in each direction (to model an OC-48 link). As a result, some requests will be rejected under a heavy traffic load.

The total number of rejected connection establishment requests (after an initial set of requests are satisfied) using each scheme has been used as a performance measure (e.g., in [1]). However, comparison between different schemes based on such a measure (or equivalently blocking probability) may not be fair. Since different schemes will accommodate different requests, but this measure does not differentiate one request from another. So, for example, if one scheme can satisfy one request for a connection from Alaska, USA to NY, USA, while the other can satisfy two requests for connections between Buffalo, NY and New York, NY instead, it would not be fair enough to say that the second scheme is better than the first.

This motivates us to use the total earning (or revenue) as a metric. To this end, a scheme-independent *Earning Rate* matrix for the entire network is used. An entry at $(i, j)$ represents earnings per bandwidth unit and time unit by a connection from ingress $i$ to egress $j$. The earnings from a connection from $i$ to $j$ is thus the product of the earning rate, requested units of bandwidth, and the connection duration.

In this study, for lack of a better alternative, the earning rate is based on the cost of using the cheapest (or shortest) pair of AP and BP in the network from $i$ to $j$ (assuming there were infinite capacity in the network), and hence is independent of the current load in the network [3]. An important and desirable consequence of using the assumed earning rate (along with the earnings from a connection) is that it tends to discourage an algorithm that tries to maximize earnings from choosing an unnecessarily expensive (or long) path to establish the connection. Because choosing an expensive/long path under such a model may prevent other (future) connections from being established and thus resulting in lost revenues.

We compare the total earnings of each scheme and in particular, the improvement ratio over the NS scheme.

## D. Simulation Results

Table I shows the average bandwidth saving ratio (vs. NS) (over the 10 experiments) in the 15-node network and a 70-node network. For each network, the first row is for schemes using an ILP formulation, and the second row is for the corresponding schemes using the APF heuristic.

TABLE I
AVERAGE BANDWIDTH SAVING RATIO

| 15-node network | | |
|---|---|---|
| 37.2%(SCI) | 15.6%(SPI) | 28.0%(DPIM-SAM) |
| 36.4%(SR) | 8.2%(SPI-A) | 25.5%(DPIM-M-A) |
| **70-node network** | | |
| 35.5%(SCI) | 9.0%(SPI) | 26.4%(DPIM-SAM) |
| 34.3%(SR) | -19.0%(SPI-A) | 24.0%(DPIM-M-A) |

An interesting observation is that the bandwidth saving ratio of DPIM-M-A, which is 25.5% and 24% in the two networks, respectively, is quite impressive, while that of SPI-A is not and in fact, extremely disappointing in the 70-node network. The reason for this (and in particular, the negative improvement ratio in the 70-node network) is that the $BC_b$ given in Eq. 7 is so over-estimated that it could even be higher than $w$ (which is used by NS), and thus results in excessive BBW allocation. This problem of SPI can be easily fixed by using $w$ as an upper bound for $BC_b$ as in the DPIM schemes, but having this fix alone does not help improve the performance of SPI much according to our results (not shown here).

[3]If the earning rate is load-dependent, it will become scheme-dependent also.

Table II shows the average total earning ratio (vs. NS) over 10 experiments (that are different from those mentioned earlier). In these experiments, each network is loaded with heavy (dynamic) traffic. These results also show that the DPIM schemes consistently outperform SPI with a wide margin. Interestingly, due to the dynamics in the large 70-node network, SR and DPIM-M-A performs slightly better than their ILP counterparts. Finally, the results indicate that the over-estimation problem in SPI has more severe effects in networks with limited capacity and dynamic traffic because requests that could have been accommodated even by NS will now have to be rejected, which results in lost earnings.

TABLE II
TOTAL EARNING RATIO

| 15-node network | | |
|---|---|---|
| 28.7%(SCI) | -1.6%(SPI) | 19.3%(DPIM-SAM) |
| 27.6%(SR) | -5.9%(SPI-A) | 13.8%(DPIM-M-A) |
| **70-node network** | | |
| 30.1%(SCI) | -14.6%(SPI) | 12.3%(DPIM-SAM) |
| 31.4%(SR) | -30.9%(SPI-A) | 12.5%(DPIM-M-A) |

## VI. SUPPORT MULTIPLE CLASSES OF CONNECTIONS

In this section, we describe how to accommodate two additional classes of connections which differ from the protected class of connections discussed so far in their tolerance to faults: namely, *unprotected* and *pre-emptable*.

An unprotected connection, denoted by $\mu$, does not have a BP so if (and only if) its path (similar to an AP of a protected connection) is broken due to a failure, traffic carried by $\mu$ will be lost. A pre-emptable connection, denoted by $\rho$, is unprotected, and in addition, carries low-priority traffic such that even if a failure on some link $a$ does not break $\rho$ itself, the traffic carried by $\rho$ may be pre-empted because its bandwidth on some link $b$ will be taken away by a BP traversing link $b$, whose corresponding AP (for a protected connection) is now broken as it uses link $a$.

### A. Transformation

To accommodate all three classes of connections in one unified framework, we transform a request for the establishment of $\mu$ or $\rho$ into a request for the establishment of a protected connection as follows.

The definition of $\mu$ above implies that $\mu$ needs a dedicated amount of bandwidth on a path (just as an AP) but no BBW.

Hence, a request for the establishment of $\mu$ can be treated as a request for the establishment of a protected connection with its AP used to establish $\mu$ and without having to deal with BP selection and BBW allocation.

The case for an pre-emptable connection establishment request is more complex because we need to make sure that not only a pre-emptable connection will share BBW with any other BPs, but also it will be prevented from sharing any bandwidth with other pre-emptable connections.

Accordingly, we treat a request for the establishment of $\rho$ as a request for the establishment of a protected connection, but will use its BP to establish $\rho$ (and thus carries traffic), and a "phantom AP" which will use a "virtual path". Such a virtual path is guaranteed to be link disjoint with any real/physical paths in the network for use by $\rho$. In addition, all such phantom APs for pre-emptable connections will use some common portion of this virtual path (which has an unlimited and zero-cost bandwidth). Accordingly, their corresponding BPs used to establish pre-emptable connections cannot share any bandwidth (BBW) among each other, but can share BBW with any other BPs.

### B. Extension to DPIM

The DPIM schemes and in particular, the one with the ultra-fast APF heuristic, can be extended to support these two additional classes of connections as follows. Let $\mu_e$ and $\rho_e$ denote the sum of the bandwidth required by unprotected and pre-emptable connections, respectively, on link $e$. Since $\mu_e$ cannot be shared, it does not need be distinguished from $A_e$ and hence, we may define $A'_e = A_e + \mu_e$. Also, based on the transformation from a preemptable connection request into a protected connection request discussed above, we further define $B'_e = \max\{B_e, \rho_e\}$ and $R'_e = C_e - A'_e - B'_e$, where $C_e$ is the capacity of link $e$ and also the initial value of $R'_e$. Just like before, there is no need to maintain $A'_e$ (or $\mu_e$). But each node $n$ (edge or core) does need to maintain, locally, $R'_e$ (instead of $R_e$), and $\rho_e$ for link $e \in \mathcal{OUT}(n)$. Optionally, $B'_e$, which can be obtained from $B_e$ and $\rho_e$, can also be maintained locally for convenience. An edge node will also maintain, for every remote link $e$, $R'_e$ (instead of $R_e$), and $\rho_e$.

When processing a request related to a protected connection (whether for its establishment or for its release), one may follow the same procedure outlined earlier by replacing $R_e$ with $R'_e$ and $B_e$ with $B'_e$ (note that $B_e$ still needs be updated and maintained as before).

One can process a request related to an unprotected connection in much the same way that one processes a request related to a protected connection with the exception that there is no corresponding BP. Accordingly, APF works perfectly for a request to establish $\mu$.

Finally, one can process a request to establish $\rho$ using a variation of the APF heuristic as follows. First, for every link $e \in \mathcal{E}$, one calculates $bw = \rho_e + w - B'_e$. It then assigns $\max\{bw, 0\}$ as the cost of link $e$, and finds a cheapest path (just as a BP for a protected connection is found using DPIM-M-A). $\rho$ can now be established in much the same way that BBW is allocated along the path with the following three modifications: (1) the switching fabric at each node along the path should be configured; (2) $bw$ is the actual amount of additional bandwidth allocated on link $e$; and (3) $\rho_e$ and possibly $B'_e$ should be updated and their values should be multicast to all edge nodes). A request to release $\rho$ can be processed similarly.

## VII. More on Related Work

In addition to the work in [1, 3], there is a large body of related work on shared path protection with foci on and applications to SONET, ATM and WDM networks. For example, [4] described a distributed scheme where each node maintains a so-called fault management table (FMT) for each local link $e$, which lists every flow whose active path (AP) or backup path (BP) uses link $e$, but routing of APs and BPs is fixed, that is, independently of such information[4]. [5] described, among others, a distributed scheme where a node maintains a table similar to FMT for each local link, but routing of a BP is done hop-by-hop by forwarding a signaling packet that follows the breadth-first-search (BFS) order. Another distributed scheme, where no information other than $R_e$ about remote links is needed by each edge node, was only briefly mentioned in [6] without any detail on how signaling is done and what information is maintained locally at each node. It seems that this scheme can only achieve a low degree of BBW sharing without trying several possible BPs.

Another distributed scheme, which is mainly for link-based restoration, and requires each node to maintain $\mathcal{P}_{\mathcal{B}}(e)$ for each local link $e$, was described in [7]. The main difference between this scheme and our proposed DPIM schemes (as well as the above three schemes) is that such information needs to be sent to all edge nodes, and thus the scheme is more like SR.

Finally, several schemes similar to SCI-I, but mainly for an off-line case, were described in [5, 8, 9]. Many other ILP or heuristic based approaches have also been proposed and the readers are referred to a brief survey of these work in [3, 5] and the references contained therein.

## VIII. Conclusion

In this paper, we have developed a novel and comprehensive distributed control framework, called DPIM, for maintaining partial and aggregated (local and/or non-local) information needed to achieve efficient share path protection of bandwidth guaranteed connections. We have specified the information to be maintained as well as how it is updated and exchanged through distributed signaling. A simple but elegant solution, which is the first of the kind to our best knowledge, has also been proposed to support unprotected and pre-emptable connections under the DPIM framework.

The proposed DPIM scheme can, also for the first time, allocate minimal backup bandwidth (BBW), as well as deallocate maximal BBW, with only $O(E)$ partial information and under distributed control. We have compared the performance of various schemes assuming networks with both finite and infinite link capacity, and and in the later case, used a performance metric based on a fair scheme-independent earning rate. It has been shown that the proposed ultra-fast heuristic-based DPIM

[4]In addition, the amount of information in FMT is $O(\frac{LQ}{E} \cdot L)$, which is definitely larger than $O(E)$ and could be as much as $O(E^2)$.

scheme can achieve almost the same performance as the ILP-based DPIM scheme and in fact, both perform remarkably better than some existing approaches with partial information.

## References

[1] Murali Kodialam and T V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *INFOCOM'00*, 2000, pp. 902–911.

[2] J.W. Suurballe and R.E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, 1984.

[3] Yu Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," in *INFOCOM'01*, 2001, pp. 699–708.

[4] C. Dovrolis and P. Ramanathan, "Resource aggregation for fault tolerance in integrated service networks," in *ACM Computer Communication Review, Vol. 28, No. 2*, 1998, pp. 39–53.

[5] B. Doshi and et al., "optical network design and restoration," *Bell Labs Technical Journal*, pp. 58–84, Jan.-Mar. 1999.

[6] Ramu Ramamurthy, Sudipta Sengupta, and Sid Chaudhuri, "Comparison of centralized and distributed provisioning of lightpaths in optical networks," in *OFC'01*, 2001, pp. MH4–1.

[7] Ching-Fong Su and Xun Su, "An online distributed protection algorithm in WDM networks," in *ICC'01*, 2001.

[8] Yijun Xiong and Lorne G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks," in *IEEE/ACM Trans. on Networking, Vol. 7, No. 1*, 1999, pp. 98–110.

[9] Ramu Ramamurthy et al., "Capacity performance of dynamic provisioning in optical networks," *Journal of Lightwave Technology*, vol. 19, no. 1, pp. 40–48, 2001.