Performance of Protected Working Capacity Envelopes based on *p*-Cycles: Fast, Simple, and Scalable Dynamic Service Provisioning of Survivable Services

Gangxiang Shen[#], Wayne D. Grover[*]
TRLabs Edmonton, 7th Floor, 9107-116 Street N. W., Edmonton, Alberta, Canada T6G 2V4
and Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada

ABSTRACT

As an alternative of the Shared Backup Path Protection (SBPP) method, we develop a framework for dynamic provisioning of survivable services based on the use of *p*-cycles to form a Protected Working Capacity Envelope (PWCE) within which dynamic provisioning of protected services is greatly simplified. Based on *p*-cycles, the restoration speed of rings is obtained, but with the capacity efficiency of shared-mesh networks. In addition, with PWCE, arbitrarily fast dynamic service demands can be handled with much less complexity (in terms of database dependency and state update dissemination) than under SBPP. Only a simple OSPF-topology view of non-exhausted spans in the envelope is required. If a new path can be routed through the envelope, it is protected by virtue of being routable. This is in contrast to needing a full database of network state so that the end-user can set up a shared backup protection path under SBPP. In addition, dissemination of state updates occurs only on the time-scale of the non-stationary evolution of the demand statistics, not on the time-scale of individual connections. During statistically stationary periods, there is no dissemination of state updates whatsoever with an envelope that is well matched to its load. The PWCE concept thus offers some new tradeoffs between operational simplicity and spare capacity efficiency. The main contribution of this work is the detailed implementation and simulation of test networks operating under PWCE and designed with novel envelope volume maximizing formulations.

Keywords: *p*-cycles, protected working capacity envelope, dynamic service provisioning, optical networks, survivable services

1. INTRODUCTION

GMPLS-based Shared Backup Path Protection (SBPP)[1] is so far the most widely known approach for dynamic survivable service provisioning and has the full impetus of the IETF driving it. Not surprisingly, it inherits the main aspects of the conventional Internet routing paradigm: global databases of network state, local execution of centralized routing algorithms assuming global state information, and link state advertisements (LSAs) to update all nodes. SBPP also adheres to the strong Internet tradition of strictly end-to-end control, even though this places significant burden and responsibility for an end-user or end-application to track overall network state and work out their own protection arrangements. We think that the requirement for explicit establishment of a shared-capacity protection path, in dependency of the state information on all other paths in the network, for every working path limits the scalability and practical robustness of the current approach. The main concern is with the total signaling volumes and total database synchronization requirements either as network scale or as the pace of dynamic provisioning increases. In particular, maintaining network-wide coherence of the complicated spare capacity sharing relationships is a challenge which may hinder the method from provisioning dynamic service fast enough or robustly enough for long periods of operating time. In addition, not all end-users of lightpath services will want to be bothered with the internals of the network. A user should be able to establish and tear down paths at their own volition while simply indicating whether the path needs to be included in network protection or not. These considerations, as well as the call by some industry observers and research funding organizations[2] to seek radical new paradigms for future networking, and to strive to make the complexity of communications invisible to users[3], mandate that we should look for novel alternative approaches. Another concern about SBPP is specific to the current state of optical networking technology—basically, until all

---

[#] gshen@trlabs.ca; phone 1780 441-3856; fax 1 780-3600; www.trlabs.ca
[*] grover@trlabs.ca; phone 1 780 492-2532; fax 1 780 441-3600; www.trlabs.ca

Network Architectures, Management, and Applications II, edited by S. J. Ben Yoo,
Gee-Kung Chang, Guangcheng Li, Kwok-wai Cheung, Proc. of SPIE Vol. 5626
(SPIE, Bellingham, WA, 2005) · 0277-786X/05/$15 · doi: 10.1117/12.579373

individual wavelength channels along the links of a preplanned backup route are actually cross-connected following activation of an SBPP backup path (which only happens at failure time in SBPP), one doesn't know if the particular concatenation of optical links and channels actually *works* end-to-end from the standpoint of transmission integrity. This is a major reason to consider more localized protection methods and especially methods such as *p*-cycles in which the protection paths are in a fully pre-connected, tested, and monitored state prior to their use in real time for service survivability.

The Protected Working Capacity Envelope (PWCE) concept is a novel and recently proposed alternative for dynamic survivable service provisioning intended to address all these concerns. The concept was initially developed in[4] and then proposed in summary form to a wider audience in[5]. On the face of it, the PWCE proposal seems to have major advantages over SBPP in the aspects of control simplicity and potentially superior performance. However, while the framework and basic concepts were given in[4,5], implementation details and quantitative simulation-based comparisons were left for further work. This included the development of theory to design a PWCE envelope given a certain capacity budget, and although not difficult, the details of a practical control system for PWCE using protocols such as GMPLS was also left for follow-up work. In particular, the performance of PWCE has not yet been quantitatively evaluated in comparison with the SBPP provisioning technique. It needs to be confirmed if PWCE can outperform or at least achieve a comparable blocking performance to that of SBPP in equivalent capacity. The present work is limited in scope, however, to considering so-called "static" PWCE designs in the presence of stationary statistical load patterns. The prospect of *adaptive* PWCE, outlined in[4,5], has recently also been studied and is expected to appear in[6].

The objective here is to implement and study PWCE in the context of a network based on span-protecting *p*-cycles for survivability. The description in[5] was given in the context of span-restorable mesh networks in general, of which *p*-cycles are an interesting special case. *p*-Cycles are chosen for their ring-like switching speed and mesh-like capacity efficiency[7] coupled with the fact that individual unit-capacity or grouped capacity *p*-cycles can be easily established and changed as desired by the same optical cross-connects through which lightpath services are also established and released. In addition, as fully pre-connected protection structures, *p*-cycles have an inherent predictability of transmission performance in an optical network in that schemes which have to cross-connect optical channels in real time upon failure can only hope to match with extensive pre-failure monitoring of optical levels, dispersion, cross-talk, noise and polarization impairments coupled with transmission quality prediction tools.[§] The performance of PWCE is evaluated in terms of control overhead and blocking probability in comparison with SBBP-based provisioning. The outcome essentially verifies all the expected advantages of PWCE in terms of operational simplicity, comparable or even better blocking performance, and ring-like restoration speed over localized pre-connected optical protection paths which can be verified from a transmission integrity standpoint ahead of failure time.

Section 2 recaps the PWCE concept. Section 3 presents the PWCE envelope design models. Two new concepts that take form under the PWCE context—volume maximization and envelope structuring—are explained and their effects considered. Section 4 addresses test cases and simulation methods. In Section 5 we evaluate the performance of the PWCE scheme in comparison to SBPP including control overhead and blocking probability. Section 6 offers a concluding discussion.

2. CONCEPT OF A PROTECTED WORKING CAPACITY ENVELOPE (PWCE)

The main concepts behind PWCE are best appreciated by first considering a conventional span-restorable network design. Given a demand matrix, and a routing of demands over the graph, we obtain a set of working channel requirements on each span and can design a set of corresponding spare capacity allocations that guarantee restorability for any span failure at a time. This results in a division of capacity into working and spare. The working capacity serves the demands and the spare capacity provides protection for the working capacity. At the first glance, this design method seems limited to static demand problems and not helpful to dynamic service provisioning. However, even if designed for a specific static demand matrix, the set of working channels on each span can actually support many different demand patterns, not only the one exemplar to which it was designed. More generally, every configuration of a network of total

---

[§] Of course with time, as standards develop for DWDM systems, it should eventually be possible to cross-connect any series of lightwave channels (optically) to form a path dynamically without doubt about the output signal quality. This is not the case today, however. Only if payloads are electronically accessed, regenerated, and applied to new optical carriers is this kind of "first-time, every-time" certainty about the performance of a dynamically assembled optical path assured. This is one of the main advantages of pre-connected protection schemes for translucent or transparent optical networks for the foreseeable future.

capacities, can be divided into a set of working channels and a corresponding reserve network to protect those channels. Any demand matrix, which is fully routable under the working capacities present, is thus inherently also survivable under the particular partitioning of total installed capacity into working and spare. Moreover, the set of working channels defines a protected operational *envelope* within which any number of demand patterns can come and go as long as the resultant instantaneous demand combination lies within the envelope. Thus, PWCE-based provisioning involves *provisioning over inherently protected capacity, as opposed to explicitly provisioning protection for every service*. If one can route the service through the available channels of a PWCE then the service *is* inherently protected, simply by routing it. Provisioning protected services looks the same as point-to-point routing over a non-protected network. One does not have to make any explicit arrangements for protection of every individual path or frequently globally update network state for every individual path setup (or takedown). Fig. 1 illustrates a specific one of the many possible partitionings of total capacity into working and spare channel sets. What all such partitionings have in common is that the working channel count $\left\langle w_i^0 \right\rangle$ on each span is fully restorable within the corresponding graph of spare channel capacities $<s_i>$ on all other spans. Few arbitrary partitionings of capacity on each span will satisfy this property, but the theory for such fully-restorable partitionings is easily based on existing knowledge about span-restorable mesh network design (c.f. Ref. 4, Chapter 5) and even while satisfying the restorability property, there are a very large number of different $\left\langle w_i^0 \right\rangle$ configurations, i.e., protected working envelopes, that are feasible under an initial distribution of total capacity.

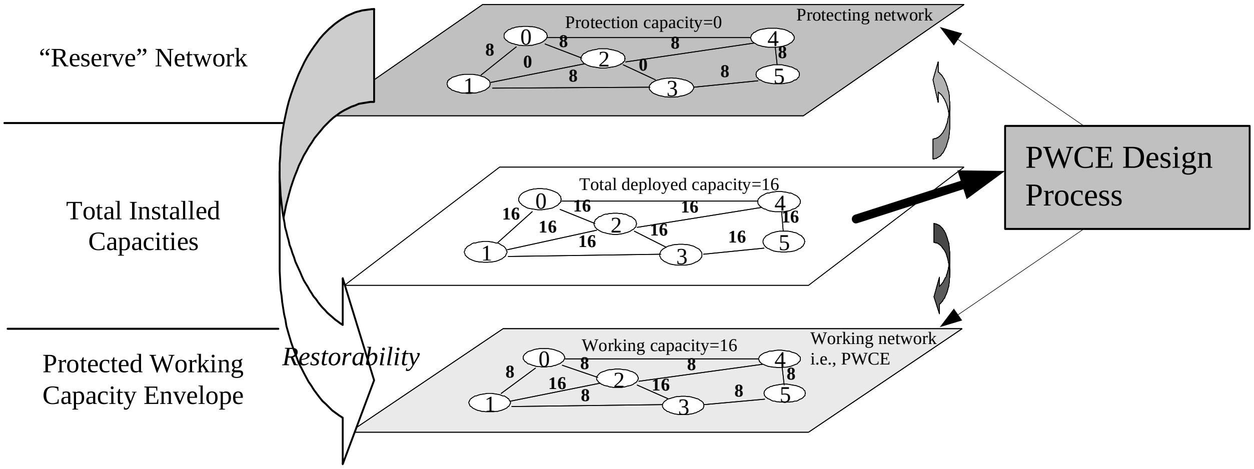


Fig. 1. Partitioning of total installed capacity into working and spare to define one possible protected working capacity envelope.

In the top of Fig.1, a set of spare channels on each span defines a *reserve network* of spare capacities $<s_i>$. Under span restoration, any distribution of spare channels provides for a certain corresponding number of protected working channels $\left\langle w_i^0 \right\rangle$ on each span below. To understand $\left\langle w_i^0 \right\rangle$, it is in effect the answer to the question: "If span *i* fails, then by rerouting through the spare capacity of the surviving graph between the end nodes of *i*, what is the *maximum* number of replacement path segments that we can create?" Any capable restoration algorithm will achieve $\left\langle w_i^0 \right\rangle$ equal to the capacity of the minimum-cut between end-nodes of the failed span through the reserve network $<s_i>$. Once the partitioning is defined any number or combination of working paths can be routed through the envelope, up to the point where all $\left\langle w_i^0 \right\rangle$ channels are used on any one span, without any attention to protection arrangements because the channels used for provisioning in the working layer are themselves protected by the reserve network (and some embedded restoration or protection mechanism). As long as the quantities $\left\langle w_i^0 \right\rangle$ on spans support routing of the demand, it is inherently protected end-to-end with no further action. Once a connection is served, local marking on each span indicates to subsequent path setup processes that the individual channel is no longer available. No other nodes need an accounting of individual channel states as they do in SBPP where sharing relationships are defined between individual paths and individual backup channels. Under PWCE other nodes need only know that the span continues having one or more provisionable channels available. This is the default case requires no signaling for state update dissemination.

Moreover it can be appreciated that as long as span occupancies remain under $\left\langle w_i^0 \right\rangle$ (i.e., within the envelope) then path setups and tear-downs can be going on arbitrarily frequently and it makes no difference—there is no signaling needed to arrange protection per-path and no global state update dissemination whatever. The only signaling is for the source-routed establishment and tear down of each path by its originator node, and does not involve any nodes other than those on the paths themselves. Only if the *pattern* of random dynamic demand evolves—in a way such that a span approaches the envelope—is *any* updated network state dissemination required. A single LSA then either withdraws the highly utilized span from further routing or issues an updated cost for OSPF-type routing over that span.

Nodes operating under PWCE need only participate in simple OSPF-type of topology orientation (not OSPF-*TE*) to support distributed end-node provisioning via a constrained source-routing protocol (such as RSVP-TE or CR-LDP). And at this level of transport basic topology is almost never changing, so this is an almost one-time learning of the basic graph topology. Full-blown OSPF-TE dissemination of detailed changes in actual capacity and shareability state on each span is not needed because every edge of the graph will remain available for routing as long as its current in-use channel count is below the maximum number of working channels $\left\langle w_i^0 \right\rangle$ that can be protected on that span. Nodes can be told via a centralized NMS what the dimension of the current PWCE is (i.e., the $\left\langle w_i^0 \right\rangle$ value on each of its incident spans). This is a database of one number per span to be maintained within each node and does not need global dissemination. For now, let us consider a network in which the spare capacity of each span is a pre-assigned and fixed. This defines a "static" PWCE.

An important property is that actions of any type related to ensuring protection occur only on the time-scale of the statistical evolution of the network load pattern itself, *not* on the time-scale of individual connections. Thus, any need for network management actions or state change dissemination is *far* less dynamic than the traffic itself. It takes a shift in the *statistics* of the demand pattern to require a logical change in the working envelope. Importantly, such adjustment actions also occur on a time scale where traffic behavior exhibits correlated observable trends that can be taken into account in capacity-configuration planning. Variations in total demand and the pattern of demand, have strong correlations day-over-day which would allow the advance planning of several envelope configurations within the installed total capacities, each of which is known to suit the characteristic time of day to minimize any blocking. In contrast SBPP works at the call-by-call time-scale where individual departures and arrivals always appear essentially random, and routing is individually controlled by end-users. This is an environment of inherently incremental local reaction to the next arrival, not involving any opportunity for optimizing capacity use or routing strategies at overall network level to enhance performance. The protected envelope requirement is, however, very slowly changing or static over long periods of time, even in the most frenetically dynamic network. No matter how rapidly individual lightpath demands come and go at random, the *envelope* requirement will not change at all if the demand process is at statistical equilibrium. The envelope is only sensitive to non-stationary drift in the underlying pattern of random arrival/departure processes.

2.1. *p*-Cycle-based PWCE

p-Cycles, introduced in 1998[7], are somewhat like BLSR rings, but with support for the protection of *straddling* span failures as well as the usual protection of spans on the ring itself. An important property of *p*-cycles is that the cycles are fully pre-configured with pre-planned spare capacity and when a span failure happens, only the two end node of the span do any real time switching, but no switching actions are required at any intermediate nodes of the cycles. This property therefore greatly improves the *p*-cycles restoration speed to be essentially the same as BLSR rings. (In fact where propagation distances factor into BLSR speed, *p*-cycles will beat BLSR restoration times for straddling span failures because the protection path length averages only half the cycle circumference, not its full circumference as in rings.) This is an important speed advantage over restoration schemes and over other schemes such as SBPP where the protection routes are pre-planned but all the switches on the intermediate nodes of these routes need to seize and cross-connect spare capacity in real-time upon a failure. In addition, the advantage of optical-path predictability due to preconfiguration has already been pointed out. At the same time, however, protecting straddling span failures enables networks to be designed with essentially the same capacity-efficiency as a span-restorable mesh network. This is why the discovery of *p*-cycle based networking is so remarkable—we truly get the best of both ring and mesh worlds: "the speed of rings and the efficiency of mesh"[7]. The literature on *p*-cycles is now fairly extensive, so we need not say too much more. For a comprehensive treatment of *p*-cycles, Chapter 10 in[4] and/or the *p*-cycles home page on the web[8].

The application of the PWCE concept to a *p*-cycle network is similar to that for a span-restorable mesh network. The difference is that instead of the spare capacity being assembled on demand into a required path set for restoration of a specific failure that arises, a set of *p*-cycle structures will always be pre-established within the set of spare channels $<s_i>$. The concepts above still apply directly in that given a distribution of total capacities, and specific reserve of $<s_i>$ spare channels implies a PWCE $\left\langle w_i^0 \right\rangle$ except that in everything that follows from here, the basic mapping $<s_i> \rightarrow \left\langle w_i^0 \right\rangle$ has the additional intermediately of a defined set of *p*-cycles built within the $<s_i>$ (i.e., $<s_i>\rightarrow$*p*-cycles$\rightarrow \left\langle w_i^0 \right\rangle$) so that when the envelope needs it, protection is available not through access to "raw" $<s_i>$ but through preconfigured, optically tested, ready, and fast-acting *p*-cycles.

3. PWCE CAPACITY DESIGN USING *p*-CYCLES

Above we observed that given a set of total capacities, we can partition them many ways to define different operating envelopes. Notionally these envelopes can have different total "volumes" (the sheer total number of working channels provided) and also different "shapes." The "shape" of an envelope refers to its specific $\left\langle w_i^0 \right\rangle$ vector structure in terms of which node-pairs inherently see more or less potential connectivity than others. So how do we define the best operational envelope? It will involve these two general notions of volume and shape. For given total capacities, we obviously want the operating envelope to be as *voluminous* as possible. But we may also want the envelope to be in some sense structured or *shaped* well to support the characteristic pattern of random demand intensities (the *load*) that is expected.

There are various principles through which the design of PWCEs can be approached. One way to start with is to use the conventional *p*-cycle design method, based on a forecast or other model demand pattern and then adjust its spare capacity distribution for maximum PWCE volume based on forcer principles. In a conventional design there are often many non-forcer spans[9], so the envelope is not optimal in a volume-maximized sense. The point is that although optimal in terms of minimum spare capacity for the model demand pattern given, a conventional design does not fully exploit the spare capacity from a PWCE volume standpoint. We therefore employ the forcer-structure-exploitation model from [10] to construct volume-maximized PWCEs. This brings the $\left\langle w_i^0 \right\rangle$ on each span up to where it enters a co-forcer relationship with the initial forcer spans. In[10] we developed three ILP models to exploit this form of extra PWCE capacity under the forcer structure of the conventional designs. Here, we extend these models to construct PWCEs under various budget constraints. We also consider the structuring effect in the designs. This creates a total of eight possible design strategies, summarized in Fig. 2. For ease of reference we call these models A, B,…, H. Space does not permit formal statement of every model in full form (these can be found elsewhere [10, 11, 12]), but nor is that necessary to convey the key principles that leads to construction of all cases. Instead, let us present the "Structuring" Model B to show how it works and then just comment on how the other models vary to reflect their specific other.

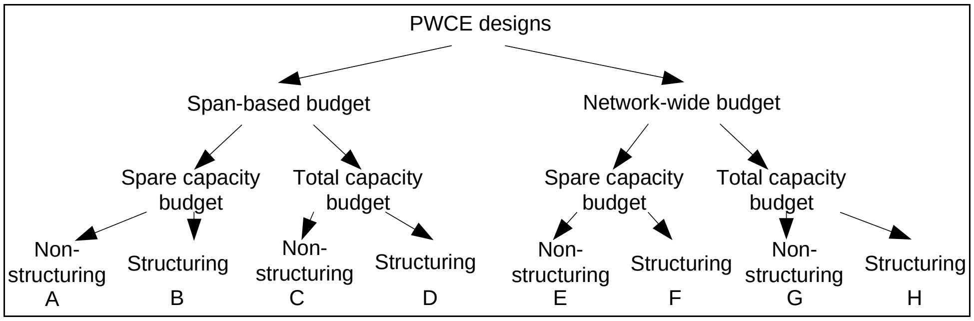


Fig. 2. Taxonomy of various PWCE design models

Some parameters and variables common to all models are as follows:

Sets and Parameters:

S is the set of spans of a network.

P is the set of all simple cycles of the network graph that are eligible to be used for *p*-cycles. Note that the number of finally selected *p*-cycles is normally much smaller than that of eligible cycles for the design problem.

X_i^j encodes *p*-cycle to span protection relationships. It is two if span *i* is a straddler on cycle *j*, one if span *i* is an on-cycle span of cycle *j*, and zero otherwise.

P_k^j encodes the layout of cycles on the graph. It is one, if cycle *j* uses span *k*, zero otherwise.

l_k is a vector of one value per span used in different ways relating to the strategies of "shaping" the PWCE. For example l_k may reflect target (not necessarily required) capacity levels "shaped" to match a given demand-matrix undergoing shortest path based routing. It is a feature through which we can influence or promote certain general structural properties, but without pinning the design down to optimality for only one single demand matrix.

α is a weighting factor to mediate the trade-off between shape-conformance structuring and volume maximization of a PWCE.

s_k is the number of spare channels assigned on span *k* in a PWCE partitioning. Depending on the design strategy, it may also appear as a decision variable, not always an input parameter.

Variables:

w_k^0 is the number of protected working channels on span *k*. $\langle w_k^0 \rangle$ defines a PWCE design result.

n_j is the number of unit (i.e., single-channel) copies of cycle *j* to be preconfigured as *p*-cycles.

λ is a scalar variable associated with the extent of asserting the shape information in l_k on the solution.

Now we can introduce the example design model for discussion.

Model B: Combined Structuring and Volume Maximization under a Span-wise Spare Capacity Budget

Objective: maximize $\left\{\lambda + \alpha \cdot \sum_{k \in \boldsymbol{S}} w_k^0\right\}$

Constraints:

$$\sum_{j \in \boldsymbol{P}} X_i^j \cdot n_j \geq w_i \quad \forall i \in \boldsymbol{S} \tag{1}$$

$$s_k \geq \sum_{j \in \boldsymbol{P}} P_k^j \cdot n_j \quad \forall k \in \boldsymbol{S} \tag{2}$$

$$w_k^0 \geq \lambda \cdot l_k \quad \forall k \in \boldsymbol{S} \tag{3}$$

Constraints (1) and (2) are standard "restorability asserting" and "spare capacity generating" constraints, respectively, adopted directly from standard *p*-cycle spare capacity design[7]. For this model to have a constrained solution, spare channel counts on each span are input parameters (not variables). Within this spare capacity environment the problem is to form a set of *p*-cycles that protects the largest possible total number of working channels on the spans (the second term of the objective), balanced against a target shape-matching effect represented by the structuring target $\langle l_k \rangle$. The shaping aspect reflects that pure volume maximization may place envelope capacity on spans that are not really needed that much for routing of most of the plausible instantaneous demand patterns. If designing for random but statistically stationary dynamic demand, a natural strategy is to set $\langle l_k \rangle$ to match the accumulated capacities on span that is the expected Erlang intensities of the random arrival/departure stream on each node pair, which is shortest-path routed over the graph. The idea of *shape structuring* is thus to beneficially influence the distribution of envelop capacity to reflect plausible relative load intensities, but without greatly (or even at all) reducing the total envelope channel volume. Thus it can be seen how constraint (3) works: it ties in with the bi-criterion objective function to try to shape the envelope to be similar to the structuring pattern by bringing up the shape compliance factor λ. In results where the α is set to a very small value this promotes maximization of the shape compliance scalar λ until the spare capacity cannot guarantee restorability of the envelope if λ is any higher. Then (the hypothesis is), that the secondary objective of maximizing envelope becomes the predominant effect enhancing the PWCEs ability to face random demand.

The other ILP models in Fig. 2 can now be discussed. Model A is a non-structuring version of model B, which simply considers volume maximization as the unique objective. Thus, model A is effectively a special case for test purposes which is just the limiting case of model B with $\alpha \sim \infty$. Model D is also a shape conforming model but instead of spare capacity being given, it assumes that the total deployed capacity on each span is a given parameter and the spare capacity on each span becomes a variable determined by the design model. Model D thus determines the split of the total deployed capacity on each span into working and spare capacities as illustrated in Fig. 1. Model C is the volume-only version of model D. (C is to D as A is to B). Models E and F follow the same shaping, and volume-only pattern, but in each of their cases it is a network-wide total spare capacity given as a parameter that constrains the problem. Models G and H are the corresponding cases where finally it is only a budget on total network capacity, the sum of all working and spare capacities (or costs), which is given as a constraining parameter. As before, G purely maximizes the volume of PWCE, while H considers the structure shaping effort as well.

4. TEST CASES AND SIMULATION METHODS

In this section we detail the experimental methods for implementing and comparing SBPP and PWCE schemes in terms of control overhead and blocking. Under dynamic service provisioning, a network can be regarded as a discrete-event driven system with two types of random event, *service connection arrival* and *service connection release*. For comparative studies of blocking performance, we follow widely-used practice and assume that arrivals follow a Poisson process with an arrival rate of ρ per second, and each established service have a mean holding time of $1/\mu$ and a negative-exponential distribution. We normalize time measurements using $1/\mu = 1$ so that the service traffic load between each node pair can be considered in units of Erlangs as ρ. The arrival and release event sequences run independently on each node-pair concurrently. The blocking probability is defined as the ratio of the network total blocked lightpath requests to the number of total arriving requests over a simulation period.

Under the PWCE scheme, once a PWCE is defined, provisioning within it is the same as service provisioning with no protection, and we employ a hop-based shortest path algorithm to search for a feasible route for each newly arriving service request. If the search is successful, then the path is established and the status of the available network resources is updated at each span with the consumed resources set as *unavailable*. This is a centrally computed result but exactly emulates the operation in practice with constrained source routing based on a simple OSPF-type view of the topology of currently non-exhausted spans. Only when there is no free capacity left on a span, is the span an LSA issued (in the emulation) and the exhausted span is effectively removed from the graph seen by the routing algorithms for new arrivals.

The basic simulation steps for SBPP provisioning are as follows. For a service request (an arrival) event, we search for a pair of first-fit working and disjoint protection routes. The two routes are link-disjoint and the protection path is ensured to maximally share already committed spare channels on the backup routes. Once the working and protection paths are established successfully, the network state is updated to record used working capacity, spare capacity, and update the spare capacity sharing relationships on each span. The information on the working and protection route pair is also recorded in the connection database. If the search process cannot find such a pair of working and protection routes, the service request is blocked and discarded. Upon service release, all resources consumed by the working path and any spare channels solely used by its backup path are returned to the available or unused status. Here “solely” means that the released backup path is the only one currently assuming use of that spare channel if needed. A spare channel is only truly released back to an available state when all sharing relationships on it are removed. Full details of the First Fit (FF) SBPP route searching algorithm, can be found in[11, 12].

To set the budgets or capacity constraints for different PWCE test case design, we employed the conventional *p*-cycle survivable network design as a baseline to determine the required working and protection capacities. Shortest path routing of a representative demand or Erlang load matrix was used to design the initial working capacity in this preliminary design of the test cases. If there was only a single shortest route between a node pair, then all the demand units between the node pair were carried by the route; otherwise, if there was more than one equal shortest route between the node pair, then the demand units were evenly allocated onto each of the routes. The required spare capacity was computed by the optimal *p*-cycle Spare Capacity Assignment (SCA) design model[7] to fully protect the working capacity. We then use the working and spare capacity total of this preliminary design to set baseline budgets for various volume-maximized PWCE designs. The spare capacities were used in the ILP models which take a spare capacity budget per span, and the sums of working and spare capacities were used for the models that take total capacity budgets, either per-span or network-wide, as detailed above. To compare the performances of the PWCE and SBPP fairly, we ran simulations for both with equivalent network capacity. For example, for the survivable network based on the conventional *p*-cycle design, we used the resulting protected working capacity to function as a PWCE to provision dynamic survivable services. Corresponding to this, we used the same *total* capacity (which is sum of the envelope capacity and the corresponding protection capacity) on each span as being available to provision services with SBPP. As is the nature of SBPP it therefore sees the total capacity of the network as being available, without any a priori designation of channels as working or (shared) protection. The identical time-and-space sequence of arrivals and departures on the network are presented for the PWCE provisioning model, but the latter provisioning process only "sees" the $\left\langle w_k^0 \right\rangle$ capacities of each span under each different strategy for PWCE definition as being available for service provisioning.

We evaluated performance on a 14-node 21-span NSFNET, a 10-node 22-span SmallNet, and an 11-node 26-span COST239 network[10] models. For initial capacity planning in PWCE designs we assumed that in NSFNET and SmallNet each node pair had three units of demand and in COST239 each node pair had two units of demand. These were the demand matrices to generate capacity budgets for the conventional design and to generate the $\left\langle l_k \right\rangle$ shaping vector for PWCE test case designs. A total of 10^5 arrival events were simulated for each measurement of blocking probability.

5. RESULTS AND DISCUSSION

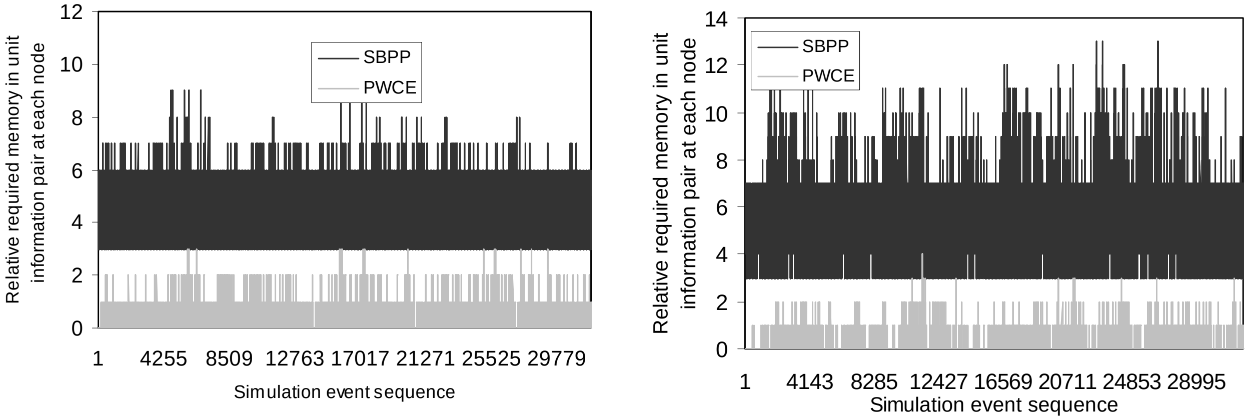
In this section, we present and discuss our experimental results. We first compare the required control overhead and memory for network state database storage between PWCE and SBPP. This is followed by the blocking performance comparison between the two provisioning methods. In addition, we also consider the effects of more frequent updating of available capacities via additional LSA flooding in the PWCE strategy. The effect of the number of usage-levels on PWCE's blocking performance and control overhead is evaluated.

5.1. Control overhead and network state memory

To verify the operational simplicity of PWCE we conducted experiments to collect the statistical information on the network control overhead and the average memory required at each node to store the network state information. Assumptions were made as follows. First, for PWCE we assumed that only when the capacity on a span is used up or becomes available again, is an LSA message flooded to update the link status at all other nodes. Depending on the span initial status, the field of "Status" can be "available" or "unavailable." As long as the capacity on a span is never exhausted, the span status is always conceived as "available" by default. In contrast, for SBPP we count a new round of LSA message flooding whenever a new connection is established or an old connection is released. The source node of each connection is responsible for flooding the information on the change of connection status. Each of the LSA messages should contains information as follows: (1) working route, which is represented by a sequence of span IDs, (2) protection route, which is represented by a sequence of span IDs as well, and (3) information of spare capacity sharing on each span transit by the protection route. If we consider the *information pair* (Span ID, Status) as a basic unit of control overhead, for a survivable service with a *W*-hop working route and a *P*-hop protection route, to flood such a message the network needs at least a control overhead of *W*+*P* units of basic information pair. Note that here we assume that the information on spare capacity sharing has been piggybacked on the information of working and protection routes, which in fact has underestimated the required control overhead of SBPP.

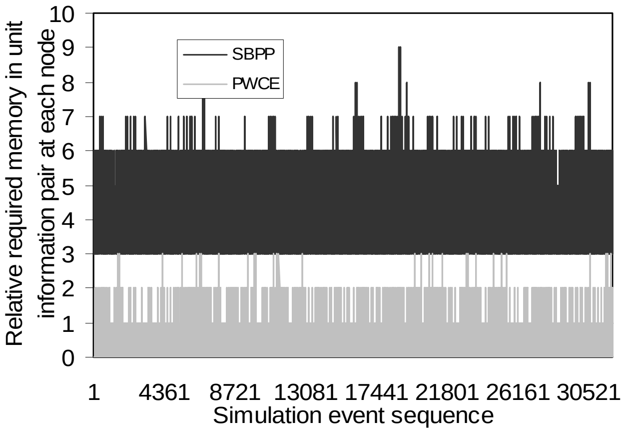
Compared to SBPP, PWCE is expected to have much less control overhead owing to its lower LSA flooding frequency and smaller LSA message size. The lower LSA flooding frequency of PWCE is attributed to the feature that only when the capacity of a span is used up or available again should an LSA be flooded. And the smaller message size is because each time PWCE needs to flood only *one* information pair when needed, while SBPP always needs to flood *a sequence of* information pairs (i.e., *W*+*P* units) for each state change of a connection. As a quantitative measure to reflect the relative amount of control overheads, we counted the signaling in terms of the basic overhead unit of an information pair above.

The three test networks were investigated in the context of design model B with $\alpha = 10^{-6}$. Results are shown in Figs. 3 (a), (b), and (c) respectively. The x-axis shows the simulation event sequence and the y-axis shows the amount of control overhead in unit information pairs. The simulation event sequence corresponds to all the arrival or departure requests. For each event, the y-axis records the directly-resulting amount of control signaling for network state updating. The data of SmallNet was collected when the traffic load on each node pair is 2.0 Erlangs, NSFNET was studied when the traffic load is 2.4 Erlangs, and COST239's traffic load is 1.2 Erlangs. From the figures, it is easy to see that SBPP requires a much higher control overhead than PWCE. For example, in SmallNet, the control overhead of PWCE is around 0.1 information pairs disseminated per event on average, while the corresponding value of SBPP is around 3.9, a factor of ~40 higher. For NSFNET the difference is a factor of ~100 times and ~20 times in the COST239 test case. From the above results, it seems that under a sparser network, PWCE can save more control overhead. This is reasonable since in a sparser network, the working and protection routes are normally longer than those in a denser network, which therefore requires a relatively larger value of *W+P* for each LSA message under SBPP. Therefore, from control-overhead-saving point of view, a sparser network seems to show more advantages over a denser network.



(a) SmallNet, uniform traffic load between each node pair 2.0 Erlangs

(b) NSFNET, uniform traffic load between each node pair 2.4 Erlangs



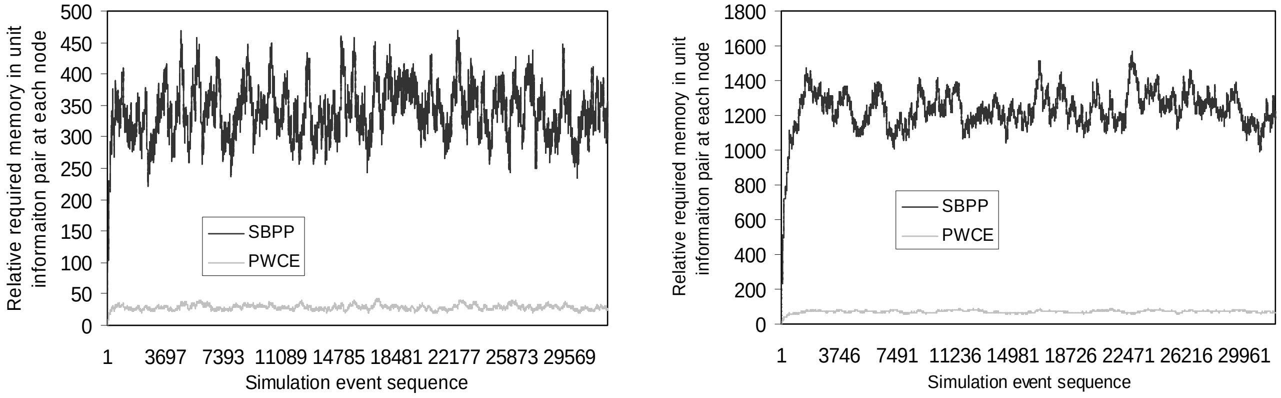
(c) COST239, uniform traffic load between each node pair 1.2 Erlangs

Fig. 3. Control overhead comparison between SBPP and PWCE

In interpreting these data it should also be noted that, low as it is, the PWCE signaling volume also depends on the blocking probability of the simulation, which in turn depends on the total Erlang load of random demand relative to the physical capacity of the test case networks. In fact as blocking becomes negligible, PWCE signaling (for state update) vanishes in the limit. To first order, however, SBPP update signaling load is not responsive at all to blocking level (hence to relative offered load to capacity ratio). Only at very high blocking levels would the effect be noticed—more blocked connections means less state update because fewer connections are actually established relative to offered requests. It follows, therefore, that while these results experimentally demonstrate and validate claims of PWCE signaling reduction relative to SBPP (20 to 100 times), the reader should appreciate that in fact any desired ratio between the two could actually be obtained experimentally because as the relative load of the test case scenarios lowers, to very low blocking, PWCE update signaling will in fact vanish.

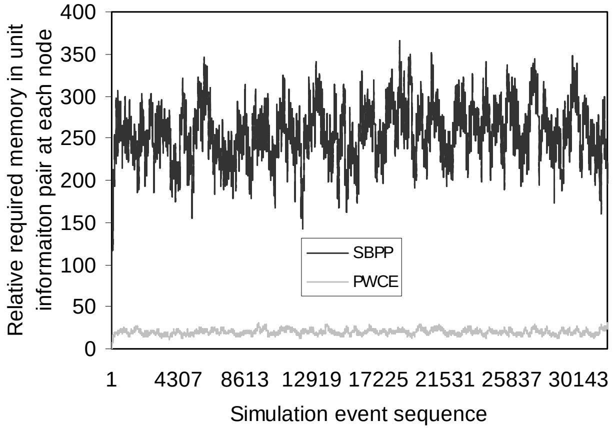
In the network control system such as GMPLS, all the network status information is stored at each local node, so whenever a new connection request arrives, the node can make an independent decision to determine whether a new connection can be established or not. For SBPP, the network status information includes the states of all the connections in the network, of which each consists of a pair of working and protection routes and the corresponding spare capacity sharing relationship. In addition, the state database also needs to maintain the information on the network topology, such as which span capacity has been used up, and which one is still available. In contrast, under PWCE each node needs to maintain the information on the connections started or terminated at the node itself only and the OSPF-like span availability information, which does not need to store any information related to the protection routes.

As a measure of network-wide state memory, we continue using the information pair (Span ID, Status), as our basic unit of required network-wide state memory. From the same experiments as above the results in Fig. 4 were obtained. The x-axis is as a before but now the y-axis records the average memory required at each node to store network state information. Under SBPP, a connection with *W*-hop working route and *P*-hop protection route can be measured to require a memory of *W+P* information pairs. Similarly, under PWCE, a connection with *W*-hop working route (no need to store any information related to protection routes in PWCE) is estimated to require *W* information pairs in memory. Observing the experimental results, it is easy to see that SBPP requires a much higher memory at each node that supports source-routed provisioning actions to store the network status information. On average the differences are around 10 times for SmallNet and COST239. On the sparser NSFNET, the difference is more than 18 times. The same reason can be ascribed to the fact that a sparser network normally has a larger value of *W+P*.



(a) SmallNet, traffic load between each node pair 2.0 Erlangs

(b) NSFNET, traffic load between each node pair 2.4 Erlangs

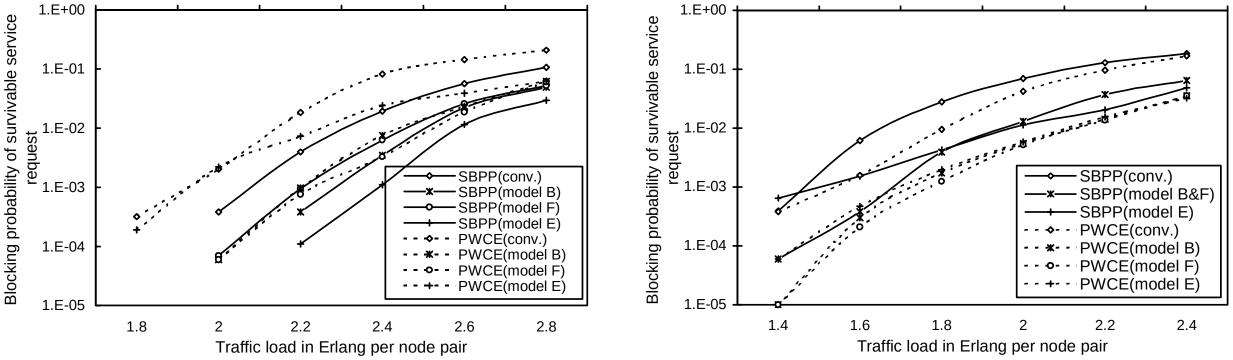


(c) COST239, uniform traffic load between each node pair 1.2 Erlangs

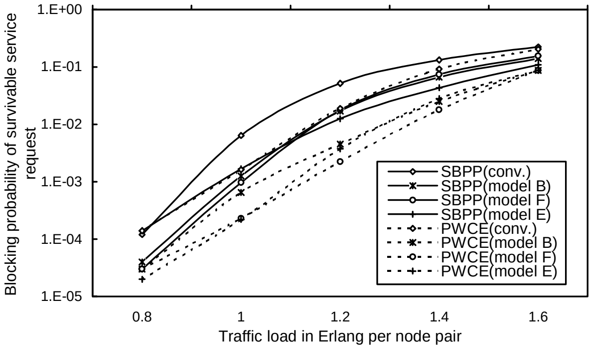
Fig. 4. Required network status memory at each node: an approximate comparison between PWCE and SBPP

5.2. Blocking performance

In this section, we compare the blocking performances of PWCE and SBPP in networks of identical capacity and loads according to the methods detailed above. Fig. 5 shows the simulation results for the three test networks under various design cases. In the legend, the “conv.” denotes the network based on conventional capacity design for he nominal demand matrix. Legend entries “model B, E, F” denote the networks designed by models B, E, and F respectively. Fig. 5(a) shows the blocking results on NSFNET. Here SBPP has slightly better blocking than PWCE for each of the design cases. However, the results in Figs. 5 (b) and (c) on the more highly connected SmallNet and COST239 networks (average nodal degree are 4.4 and 4.7 respectively) are the other way around. Here, PWCE outperforms SBPP in all the design cases. The performance difference is most pronounced in Fig. 5(c) where the curves of the PWCE-based schemes (i.e., PWCE models B, E and F) are clearly grouped together, all below the SBPP-based design cases. We can explain these results as follows:



(a) NSFNET (b) SmallNet



(c) COST239

Fig. 5. Blocking performance of PWCE-based and SBPP-based provisioning schemes based on hop-based shortest and the FF algorithms in the NSFNET (a), SmallNet (b), and COST239 (c) networks. All simulation rounds=10^5 arrival events

Two differences between SBPP and PWCE provisioning methods seem to lie at the heart of the experimental behaviors seen. One is the protection mechanism that they employ, and the other is whether the protection capacity needs to be directly considered in service provisioning. First, SBPP is a path-oriented protection mechanism, while *p*-cycles are a span-oriented protection mechanism, so SBPP should generally have slightly higher intrinsic capacity efficiency in terms of spare to working ration required for survivability. Therefore, in this aspect the SBPP provisioning method can more widely share spare capacity during provisioning process than the PWCE provisioning method. On the other hand, the PWCE method requires a more global optimal relationship between the spare capacity and protected working capacity even during the provisioning process. This tends to work against SBPP because it handled all events incrementally, whereas a well-dimensioned PWCE has the prospect of greater initial and ongoing global optimality. After a PWCE is constructed, the protection capacity is never involved in any direct operation (or change) related to the dynamic survivable service provisioning and so cannot evolve incrementally into a poorer global configuration under random demand—but SBPP can. With SBPP, the optimal protection relationship can only be ensured locally on a connection basis, instead of statically network-wide. For each incoming survivable service, the SBPP provisioning process finds the first shortest route to establish a working path and second shortest route, which is link-disjoint from the first route, to establish a backup path. During setting up the backup path, although it is allowed to maximally share spare capacity enroute, such sharing can only locally ensure the optimality of the current connection. It is impossible to reconfigure other existing connections or even foresee the future connections to make a network-wide optimization. After several connections are established, it can be expected that the consequential spare capacity sharing can be less efficient than that of static PWCE protection.

In summary, there are two effects, intrinsic protection sharing efficiency and network-wide optimality that affect the overall network performance. SBPP shows advantage in the spare capacity sharing on the sparsest of the networks (NSFNET), where it is harder for a span-restoration technology to be as efficient. On the other hand in networks where the nodal degree is higher, intrinsic protection capacity efficiency is less of an issue and this enable PWCEs greater sense of global optimality of structure to let it win out in terms of blocking. Thus, all the results for the test networks can be explained as the consequences of the interaction between these two effects. In the COST239 and SmallNet networks, where the benefit of wider sharing scope of SBPP is weakened due to the high connectivity of the network, the effect of network-wide optimality therefore overwhelms the effect of sharing scope, which causes the *p*-cycle-based PWCE provisioning method to display a lower blocking probability. In contrast with NSFNET, which has comparatively low connectivity, the greater intrinsic efficiency of path protection matters more in terms of available capacity left for working paths and so SBPP exhibits better blocking.

To further validate this reasoning and to study how the respective blocking performances of SBPP and PWCE are affected by the graph connectivity, we designed a series of artificial network topologies based on a 10-node ring network as an initial topology. Step by step we added spans to the network to gradually increase the network nodal degree. Because the number of nodes in the network is ten, any addition of a span will increase the network nodal degree by 0.2. Starting from the ring network with nodal degree 2.0, we added ten spans to the network until the network nodal degree is 4.0 and generated a total of eleven network topologies. We assume two units of uniform forecasted demand on each node pair. We designed PWCEs for these networks based on the previous volume-maximized models and ran simulations for them. It was found that, indeed, with the increase of the nodal degree, the blocking performance of PWCE relative to SBPP improves progressively and outperforms SBPP when the network nodal degree is equal to 3.2 or higher. The result is quite in line with the results that we have obtained for the above three specific test networks and verifies the interpretation above. NSFNET has average nodal degrees not greater than 3.0, in which the performance of SBPP is better than PWCE; SmallNet and COST239 both have average nodal degrees larger than 4.0, in which the performances of SBPP are better than SBPP. Practically speaking, however, what is of most significance is simply that in no case is PWCE blocking very different from SBPP. From a practical operating standpoint, if the blocking is essentially on the same order of magnitude, then the differences in blocking hardly matter—it means that all the other advantages such as operational simplicity of PWCE can be accessed for their own purposes and benefits.

5.3. Impact of multiple span-usage update thresholds on control overhead and blocking performance

In the results above for PWCE, LSAs are emitted only when the envelope capacity of a span was strictly exhausted or returned to being available again. In addition the hop-based shortest path routing algorithm was used to search working routes through the PWCE. The advantage of this is that we can save much control overhead by assuming that each span is always available on the topology without flooding any LSA message to update the current capacity usage level on each span. Nevertheless, searching for the working route simply based on hops within non-exhausted spans may not achieve the best blocking performance because there is no sense in which that form of routing tries to "hedge" against using relatively heavily-loaded spans. The idea in this sub-study is therefore to use more fine-grained information on relative capacity usage on each span to compute the routing cost. In other words, the "shortest" route will now appear to be the least-loaded path between source and destination, otherwise known as the Least Load (LL) algorithm[13]. The cost of this is, however, more control overhead to update the status of network capacity usage more frequently. This therefore forms a tradeoff situation between control overhead and network blocking performance. With a lower control overhead, the blocking performance may not be the best, but to improve the blocking performance by using the LL algorithm, more control overhead is required.

Corresponding to the above tradeoff, a more general LSA flooding strategy is what we call the "usage-based flooding strategy." Under this strategy the total envelope capacity on a span is divided into several levels. Whenever a change to a new usage level occurs on a span, an LSA is emitted to update other nodes of the change. In this light, the prior LSA policy is just the extreme case of usage-based LSA updates where a single usage level is defined, i.e., either zero or more than zero. Because there is almost no LSA flooding under that policy, we can now refer to it as the *hibernating* strategy. At the other extreme, every change in a single channel usage status, generates an updating LSA, and is referred to as the *real-time* strategy. Under the later scheme, LSA messages are flooded almost as frequently as with SBPP, aside for a factor of about two fewer for PWCE due to the average relative number of spans affected by a provisioning or release action. PWCE also only needs to flood information related to working capacity, while SBPP is flooding information about both working and spare capacities and share capacity sharing relationships.

It is easy to see that a more active flooding strategy requires a larger volume control overhead, but yields more accurate and current information on capacity usage of each spans. And this is information we can use with the LL routing algorithm to try to do more congestion-avoidance in routing. To implement and study effects of multiple usage-level reporting we define the number of usage levels as a parameter *l*. Based on the remaining (available) envelope capacity on a span we then determine the usage level as:

$$Usage \quad level = \left\lfloor {(c-1)\cdot l} \big/ {w_i^0} \right\rfloor + 1 \tag{4}$$

where c is the current unused envelope capacity on the span and w_i^0 is the total envelope capacity on the span. This divides the envelope capacity into nearly evenly spaced “trip points” at which the utilization state will be updated by an LSA. For example, if w_i^0 =9 then with four usage levels an update will occur every time channel usage rises or falls by two. If the remaining capacity is one, two or three channels, it belongs to usage level one; four or five channels belong to level two, etc. Note that this is biased towards declaring the lowest usage level somewhat early. This can help to out an extra emphasis on avoiding outright congestion when using the LL algorithm to search routes. The LL routing algorithm searches working routes based on the following usage-threshold link cost criterion:

$$Span \quad cost = 1 - \left({Usage \quad level} \big/ {l} \right) \tag{5}$$

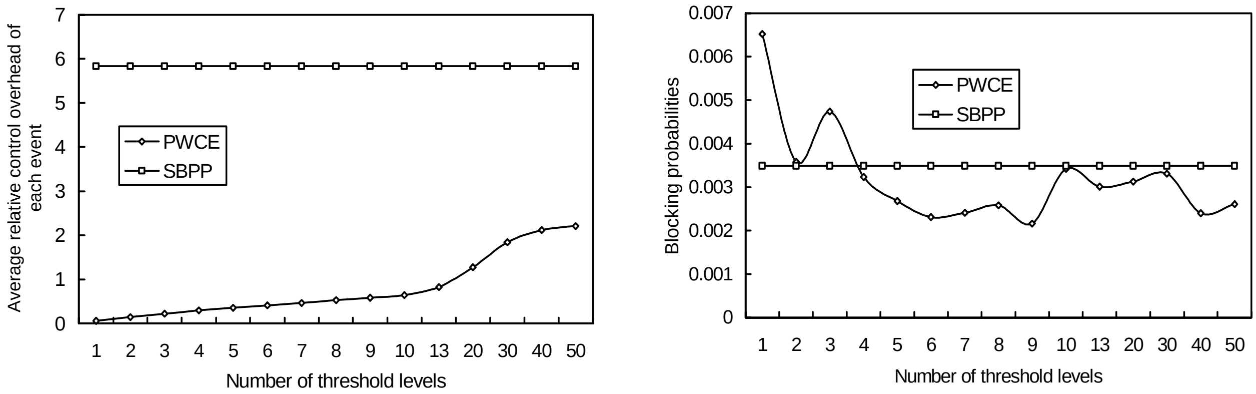
Eq. (5) thus charges a routing cost, which is proportional to the current relative utilization of total envelope capacity on each span. Assuming a route is at or near shortest hop as well, a route with the minimum sum of span cost means that the route has the lowest capacity utilization or is the least congested. Strictly it is possible for the same route cost to be seen between say a short moderately loaded route and a long lightly loaded route because usage level are simply summed over all spans traversed. With LL nothing is directly enforcing shortest hop properties any longer.

In the following results we study how the frequency of LSA updating trades off against the performance of PWCE in terms of control overhead and blocking performances. We want to see if there is any clear thresholding effect where a perhaps very marginal increase in control overhead can generate a significant improvement in blocking performance.

Simulations were conducted for NSFNET at 2.4 Erlang traffic load per node pair and the required control overheads and blocking were measured as the number of usage levels for reporting was increased. The results are compared with those for SBPP scheme, which was assumed to follow the real-time LSA flooding strategy (any of service arriving or release events will result in an LSA flooding). This seems justified and necessary because PWCE can function correctly with any number of usage levels—any effects involved are only matters of performance, not correct functionality. But by its nature SBPP does not have the same luxury. For error-free operation, it must update the global network state databases after every provisioning change (except, arguably, releases could be batched-up and released in summary form) with only a performance loss, not a failure to function correctly. Thus, SBPP has a constant amount of control overhead for different numbers of the threshold levels.

Fig. 6 shows the results for the required control overheads and corresponding blocking performances under different numbers of usage-reporting levels. Fig. 6(a) shows the average of required control overhead in unit of “information pair” units for each operational event (arrival or release). Again we see that in general, SBPP requires a much higher control overhead than PWCE. The difference ranges between 2.5 times and about one hundred times when the network is under two extreme cases. Under PWCE we see that the larger the number of LSA flooding threshold levels, the more the control overhead is required. Interestingly, however, for PWCE we also see that in moving from one usage-reporting threshold to six usage levels, the control overhead increases rather lazily in a linear fashion, but the blocking performance improvement, in Fig. 6(b), varies more dramatically. When the number of usage levels is greater than four, the PWCE blocking always outperforms SBPP. This was never the case in the prior NSFNET results no matter which design model or experimental condition. We ascribe this to the usage-sensitive LSA reporting strategy and the additional effectiveness this information gives to the LL route computation method. The net effect is to enable PWCE to outperform SBPP at the cost of a small increase of control overhead. Ultimately, it is interesting to see that increasing the number of usage levels above six brings no more blocking improvements. In Fig. 6(b) we even see that the blocking probabilities increase when the number of threshold levels lies between ten and forty. This indicates (as would reasonably be expected) that under usage-based updating there is a saturation effect at which a limited resolution of usage levels is good enough to achieve blocking performance as good as full real-time updating.

A note on the range of levels in these tests is warranted. The top number of about 50 usage-reporting levels was chosen because that matches the largest envelope capacity (in channels) of any span in the test case designs. This number of usage levels corresponds to “real time” reporting from that largest span. In most other cases when the number of levels is high, but a span has fewer actual channels than usage levels, the correct interpretation is that the span is reporting updates in real time on every change. In other words, fractional-channel reporting thresholds are not being suggested in practice, but in the simulations their effect is simply the same functionally as reporting per-channel changes, i.e, the “real time” updating strategy.



(a) Control overhead vs. number of usage-reporting levels

(b) Blocking performance vs. number of usage-reporting levels

Fig. 6. NSFNET uniform traffic load between each node pair: 2.4 Erlangs
(N.B.: number of levels greater than number of channels on a span just implies real time update reporting for that span)

Similar results were obtained for SmallNet and COST239 as well. The only difference from the results of NSFNET is that the blocking performance of PWCE is always smaller than that of SBPP no matter how many capacity usage levels are assigned. We have ascribed this to the dense network connectivity as in the previous analyses.

6. CONCLUDING DISCUSSION

This paper has provided the first quantitative study of the recently proposed PWCE scheme for simplified dynamic provisioning of protected lightpath services, in comparison to the existing SBPP scheme. The implementation of PWCE developed here is also novel by virtue of employing span-protecting *p*-cycles as the vehicle for survivability. Experimental results support expectations, previously based only on qualitative considerations, that PWCE should be vastly simpler than SBPP in terms of signaling volumes and the amount of network state to be dynamically maintained in every node, and that PWCE should have at least comparable if not better blocking performance than SBPP. The work also compared a range of PWCE design models to maximally utilize and exploit spare capacity to maximize PWCE total envelope volume and to structure the PWCE to fit a plausible model for the relative load intensities expected. A total of eight PWCE design scenarios were considered.

Some interesting insights were also developed and tested which largely explain the relative blocking performance of PWCE and SBPP. In higher degree test cases the intrinsic efficiency of span and path-based protection techniques is not that different. In these cases, the global scope that is inherent in the PWCE design model (which is in effect designed once, off-line, and with a globally optimal model) gains a certain edge over SBPP which handles every arrival as a greedily-solved incremental sub-problem. On the other hand, in the sparser network first tested here, SBPP being a path-oriented scheme, benefits more from its intrinsically lower need to use up channels for protection purposes, and consequently yields lower blocking. It was also seen, however, that under PWCE a trade-off is possible between the frequency of signaling for network state updating, and improved blocking obtained by virtue of a Least Load (LL) routing algorithm rather than simple least-hop routing. Under the latter scheme, several usage reporting levels are defined for LSA updates from each span. With that more finely resolved span usage information, the LL algorithm is able to provide a more load-leveled routing result that tends to hedge against exhausting any span. With a moderate number of usage levels, PWCE blocking dropped below that of SBPP even in the sparsest of test cases.

From a practical standpoint, however, all that is important about blocking is that the PWCE schemes not have any significant disadvantage relative to SBPP blocking performance. Indeed it is usually slightly better. But this was the main aspect that required experimental validation because it was not so clear-cut as the other predicted advantages. Given that blocking considerations seem, however, not to be an impediment for PWCE, it means that all the other practical advantages of PWCE over SBPP should be taken seriously into consideration for future networks. The main advantages are: (i) within the network itself; vastly reduced state maintenance and signaling overheads to support dynamic protected service provisioning, (ii) from a lightpath service-users perspective; a greatly simplified user interface paradigm: source route the service path as desired and simply designate it protected or not, and (iii) from an optical transmission engineering standpoint; protection path structures (derived from *p*-cycles) that are fully pre-connected and known to meet transmission integrity objectives before failure.

REFERENCES

1. S. Kini, M. Kodialam, T. V. Laksham, C. Villamizar, "Shared Backup Label Switched Path Restoration," Internet Draft, *draft-kini-restoration-shared-backup-00.txt*, Aug. 2000.
2. M. Ammar (and 26 others), Report of NSF Workshop on Fundamental Research in Networking, Report to the National Science Foundation Directorate for Computer and Information Science and Engineering (CISE), April 2003, http://www.cs.virginia.edu/~jorg/workshop1
3. R. Boutaba, G. Goldszmidt, H. G. Hegering, J. Schönwälder, M. Sloman, "Autonomic Communication Systems – Special Issue," *IEEE Journal on Selected Areas in Communications*, to appear.
4. W. D. Grover, *Mesh-based Survivable Networks, Options and Strategies for Optical, MPLS, SONET, and ATM Networking*, Prentice Hall PTR, Upper Saddle River, New Jersey, 2003. (Ch.5 for span-restorable mesh-based network design theory, Ch. 10 for references on *p*-cycle related topics).
5. W. D. Grover, "The Protected Working Capacity Envelope Concept: An Alternate Paradigm for Automated Service Provisioning," *IEEE Communications Magazine*, vol. 42, no. 1, Jan. 2004, pp. 62-69.
6. G Shen, W. D. Grover, "Adaptive Self-Protecting Transport Networks based on *p*-Cycles under the Working Capacity Envelope Concept," submitted to *IEEE Journal on Selected Areas in Communications, Special Issue on Autonomic Communication Systems*, October 2004.
7. W. D. Grover and D. Stamatelakis, "Cycle-oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration," in *ICC'98*, 1998, pp. 537–543.
8. The *p*-Cycles Home Page, http://tomato.edm.trlabs.ca/p-cycles/, Network Systems Group, TRLabs, Canada. (If the URL changes in future, check www.ee.ualberta.ca/~grover/ for an updated link.)
9. W. D. Grover, D. Y. Li, "The Forcer Concept and Express Route Planning in Mesh-survivable Networks," *Journal of Network and Systems Management*, vol. 7, no. 2, Jun. 1999, pp. 199-223.
10. G. Shen, W. D. Grover, "Exploiting Forcer Structure to Serve Uncertain Demands and Minimize Redundancy of *p*-Cycle Networks," *OptiComm'03*, October 2003, pp. 59-70.
11. G. Shen, W.D. Grover, "Design of Protected Working Capacity Envelopes Based on *p*-Cycles: An Alternative Framework for Survivable Automated Lightpath Provisioning," to appear in *Performance evaluation and planning methods for the next generation Internet*, Kluwer Academic Publishers, 2004.
12. G. Shen, "Protected Working Capacity Envelopes Based on *p*-Cycles," PhD Thesis, Department of Electrical and Computer Engineering, University of Alberta, work in progress.
13. G. Shen, S. K. Bose, T. H. Cheng, *et al.*, "Heuristic Algorithm for Efficient Light-Path Routing and Wavelength Assignment in WDM Networks under Dynamically Varying Loads," *Computer Communications*, vol. 24, no. 3-4, Feb. 2001, pp. 364-373.