Globally Optimal Distributed Synchronous Batch Reconfiguration for Efficient Hazard-free Dynamic Provisioning: How an Entire Network can “Think Globally and Act Locally”

Wayne D. Grover
TRLabs and Dept. of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada

Abstract—This paper proposes an alternative concept for handling dynamic provisioning changes and ongoing network reconfiguration and reoptimization in general. A prime motivation is concern about the dependency of existing concepts for dynamic provisioning on the real-time coherence of databases of network state at diverse geographic locations. Not only is the continual updating of such global state everywhere in the network an intensive real-time load, but inevitable incoherencies pose hazards in network operation. The alternative proposal is a framework that makes use of precise time synchronization and the computational power of network nodes to solve identical local instances of incremental reoptimization problems in situ. The scheme removes the database coherency hazard, reduces signaling volumes, and increases resource efficiencies in service provisioning. It also provides a framework in general for continually ongoing distributed reoptimization of network configuration. The proposal also points directions towards some new research questions such as implementation of optimization models for identical solution on disparate platforms and definition of a variety of incremental batch reoptimization problems. It is thought that this may be one of the first application concepts for transport network management that would involve on-line embedded use of operations research methods in communication networks.

Keywords: dynamic provisioning, incremental optimization, database hazard avoidance, on-line O.R.

I. INTRODUCTION

Present concepts for dynamic operation of transport networks are based either on a presumption of complete centralized control, or on distributed path provisioning operations undertaken independently and asynchronously by end-nodes, relying on local copies of global network state which are synchronized network-wide by “TE” type link-state updates broadcast from each node as it effects changes. The disadvantages of completely centralized control lie mainly in single-point vulnerability, signaling volumes, and scalability and are well recognized. The fully distributed peer-to-peer alternative avoids some of these drawbacks, and generally seems to be the only approach assumed for operating a dynamic survivable optical transport network. In the view of some, however, a significant but almost ignored issue which the prospective automated WDM (or MPLS) networks have to face is hazards from inconsistency in network state inconsistency, especially as network diameter increases and/or the time-scale of connections request arrivals and departures decreases. In the current thinking, connection admission control and network resource allocation functions are implemented independently at each node in a network for connections originating/terminating at that node. While this removes vulnerabilities of having a single control center and telemetry to/from that center from all network nodes, its own peer-to-peer signaling intensity still grows at least as $O(n^2)$ where $n$ is the number of nodes in the network and $\lambda$ is the arrival rate of connection requests at each node and all of this state update information is real-time critical information. In addition, the globally distributed database of network state, including tracking the routes of all paths in service and spare channel sharing relationships on backup paths could be growing as $O(\lambda nth^4)$ where $h$ is the average holding time of (protected) connections. As computational complexity arguments go, these are not extremely high growth rates for a standalone computational problems and/or database sizes, but it is hard to see why this is so often considered “scalable” in the context of a continental-scale transport network where all such signaling and database coherence is actually time-critical and mission critical because correct ongoing operation of the network relies on maintaining a globally coherent database of network state in all nodes.

In simple language, the hazard exists under asynchronous distributed provisioning because some nodes are making changes to the common state information while other nodes are relying on that information, acting on it, and making more changes based on it. Intuitively we can easily see that sooner or later this will lead to problems of almost unpredictable severity. But more theoretically that intuition is confirmed in the “Fischer, Lynch, Paterson” (FLP) theorem [5] which states:

“The consensus problem involves an asynchronous system of processes, some of which may be unreliable. The problem is for the reliable processes to agree on a binary value. It is shown that every protocol for this problem has the possibility of non-
termination, even with only one faulty process. By way of contrast, solutions are known for the synchronous case, the “Byzantine Generals” problem.” (JACM 32:2, 1985)

Although the FLP theorem uses formal language, it tells us that if the processes involved cannot be relied upon to hold a constant value while the consensus is being attempted, a stable outcome may never be reached. But this is what engineers already know in digital logic design. We “clock” our logic circuits so that at significant time instants all states are frozen, allowing for propagation time through combinatorial circuits, and time for differential delays and rise/fall transition times, and so on, so that at the next clock instant, an assured correct next-state is adopted throughout the entire circuit. The hazard exists only if there is no coordination of the times at which changes and actions will be allowed and not allowed by nodes. Thus, if time synchronization is effected, we could make an entire network operate with the stability of a large clocked digital logic circuit. Thus, the role of the FLP theorem here is to explain why asynchronous operation is unassured. But conversely lets us see why synchronous operation can be robust in this regard—it is because no one will be trying to make changes while others are acting on the same information.

In the following scheme, the equivalent of “consensus” can be reached because data is exchanged only during a time phase when all nodes are in agreement to temporarily not make any more changes to the data.2

In the context of an optical transport network, a relatively benign outcome of temporary state inconsistency occurs if a resource is incorrectly considered not available. Then the connection admission control algorithm running at a node may reject a connection that could, in fact, have been admitted at the moment. This affects only the single connection request, however. On the other hand, if a particular resource is marked as available in the network state database of a node while it is already not available in reality, then connections may be admitted without enough resources in the network to serve them. This will usually lead to a failure of one or more conflicting path forming signaling attempts following the locally determined route choices. Normally, this too will not be a severe problem. Crank-back protocols will release the resources of the failed, but partially formed paths, and again update network state globally. End nodes may then re-attempt.

If a reader contemplates seriously that such networks are to one-day operate dynamically, independent for thousands of connections a minute (network-wide), hour after hour, 24/7 for months and years, then one must be concerned about the possible outcomes of randomly arising interactions of effects from state inconsistency. It is possible to conceive worst-case event sequences that lead to the “meltdown” of the entire network because of repeated interacting resource allocation failures and runaway crankback and state updating dynamics among, in addition to loss of network state needed to correctly active protection arrangements. Any one such scenario may be vanishingly improbable but one is “running the experiment” often over a very long time. The interactions that lead to the collapse of the AT&T switching network some years were extremely improbable. Many Internet problems are also typically understood to arise from combinations of signaling and state-update interactions. Each exact sequence of interactions that leads to a brown-out or collapse is individually very improbable, but at the large scale, happen all to often. It is not possible to give an a priori proof that a serious crash of a network will arise within so much time, given so and so size and frequency of provisioning action. Rather, the point is made for us by real-world experience with crashes in systems involving numerous asynchronously acting processes and events for which correct operation relies on the real-time coherency of a common state database. Many measures can be thought of within the existing peer-to-peer framework to reduce the likelihood of such adverse complex interactions, but no such accumulation of measures guarantees that Murphy’s Law won’t eventually prevail. (How many measures have been developed to avoid deadlock and so on in computer OS’s?)

Ultimately, however, to motivate what follows, we do not think a reader needs to be convinced that such crashes are certain or will be noticeably frequent, only that they agree that the risk logically exists within the existing framework. If so, this sets the stage for, our present thesis which is to at least propose and explain an alternate framework which is free of the hazard altogether, and provide other advantages as well.

Prior research that explicitly addresses the risk that is posed has been targeted at essentially two types of workaround so far. The first is to propose connection admission mechanisms that tolerate the inaccuracy of the network state information and alleviate its impact at the price of increased connection blocking. The second one is to drop the idea of distributed operation by introducing a central entity in charge of connection admission decisions. The latter is obviously less robust as the critical point of the system is the central entity, whose substitution may entail additional problems in case of a failure. A review of related work is presented in [1].

II. AN ALTERNATE APPROACH

In this section we explain the alternate concept for what is still distributed autonomous network operation, but is hazard-free, and introduces the ability to operate with globally optimal solutions to network reconfiguration problems. The key concepts involved are:

- Small-batch change provisioning, not single path provisioning.
- Globally synchronous change actions, not asynchronous actions.
- Relegating all signaling for state update to non real-time communication requirements, rather than being real-time-critical.

---

2 The author wishes to thank one of the anonymous reviewers who pointed out the relevance of the FLP theorem to the present thesis. Upon study of the reference, however, it was found to bear on the issue in a slightly different way than the reviewer suggested. I find that the FLP theorem confirms the hazard of asynchronous operation, and only clarifies why synchronous operation does not face the same “consensus” question—because nodes will no longer be making changes while consensus-reaching is being undertaken. As long as sufficient propagation and checking time is provided, global consensus will be reachable under synchronous operation. FLP tells us only that it is not guaranteed under asynchronous operation.
Creating the opportunity for an absolutely robust confirmation of global state database coherence before any reliance upon it for network actions,

Solving and acting on globally optimal reconfiguration solutions, not node-local solutions to greedy problem models with heuristics,

Nodes acting locally to put into effect their parts only of globally optimal reconfiguration plans, instead of source-routed signaling sequences to activate new paths, etc.

The proposed framework applies provisioning requests by batches via the synchronous execution of an accumulation step (during which incoming provisioning requests are recorded), a synchronization step (during which all nodes are made aware of those provisioning requests pending at other nodes), an optimization step (whereby all nodes compute an identical new network state) and an activation step (during which the network nodes materialize the new network state). Let us now describe the key aspects.

**Batching:** First, in this approach we do not assume or accept that new connection requests, especially for transport-level paths (OCn’s or lightpaths) have to be provided instantaneously. Even though a wide range of applications may be envisioned to require on-demand connection provisioning, it seems reasonable to assume that the delay users can tolerate between connection request and setup could still be in the range of minutes at least. After all, it can take 10 s or more to establish an “on-demand” long distance phone call. So we wonder where the expectation arose that a provisioning delay of a few minutes, or more, would not be acceptable in the establishment of a lightpath that may convey 10 Gb/s and cost thousands of dollars to use?—especially if there are other benefits for both user and network operator to permitting a batch-change or pre-scheduled mode of service provisioning. Even in the most automated context of a router seeing increasing load, and “dialing up” an additional lightpath from the transport layer, it is hard to see the downside of requiring the router to make such a request on the basis of an observed trend, slightly before the added capacity is fully needed, allowing for a short operational provisioning delay. In this work, therefore, we take it as an axiom that some reasonably short (but noticeable) implementation delay upon provisioning requests will be acceptable to most, if not all, users of carrier-path signals through a transport network. 3

**Time synchronization:** This scheme also exploits the existence of precise time synchronization amongst nodes involved in the provisioning of new service paths. In today’s transport networks, all nodes have access to precise time and frequency, traceable to national atomic reference standards. This is needed for SONET network synchronization and prior to that, for slip-free digital PCM switching. Very high absolute clock time synch is obtainable either from GPS receivers or through existing terrestrial based precise-time synch procedures. Thus, all provisioning-enabled nodes can participate in network-time synchronization. With precise time as a common asset, all nodes can observe a repeating definition of time steps. For more on the technology and performance levels of precise time synchronization achieved in today’s networks see [4]. Depending on the time scale of averaging, time precision countable in nanoseconds of difference between network clocks and the national atomic reference standard are typical.

Using the precise time asset in every node, all nodes synchronously conduct the following phases. Conceptually, if a network re-optimization and reprovisioning interval of 10 minutes was chosen, then the first phase could, for example, be defined as starting on every 10th minute of the hour.

**Phase 1: (Accumulate change requests)** During the first phase no provisioning operations or changes are made at all. The existing state of all connections is retained but new connection requests are accumulated as they arrive at each provisioning access node. Departures are accepted at any time. This phase lasts for a predefined period, δ. A reasonable range for the values of δ is from a few seconds to perhaps minutes. More generally, it would be set in relation to the typical inter-arrival time of new connection requests. Phase 1 ends synchronously and precisely at a predefined time at all nodes of the network. By this we mean that, to within the sub-microsecond accuracy provided by network precise time, each node closes the request recording phase. As soon as this occurs, phase 2 is entered with respect to the provisioning requests accumulated in that interval and a new accumulation period begins immediately.

**Phase 2: (Share change requests)** In the next phase all nodes create a data packet which summarizes the list of the accumulated new connection requests (including the class of protection requested) and departures which they recorded in the preceding accumulation phase. This packet can be invested with strong error-checking and/or correction encoding, using known methods. The error-protected change summary packet from each node is disseminated to all other nodes and each node receives such “change list” packets from all other nodes. This can be done by standard packet mechanisms, somewhat like a link-state advertisement in ordinary Internet operation. Importantly, however, this is a completely non-time-critical process, with the luxury of considering any method needed to ensure correct mutual data exchange amongst all nodes before ever proceeding to act on the data. All sorts of known measures, such as checksums, can be applied to ensure that at the end of the dissemination phase, every node has an identical copy of the global list of all new connections and departures that arose in the preceding first phase. The philosophy is that although this is a critical information exchange, it is not under any real-time pressure, so the robustness of this process can be made almost arbitrarily reliable. Moreover, unless there is a positive global confirmation of correct update data synchronization, there is no need to act. For instance, we could envisage: (i) checksums in each summary packet, (ii) a checksum run by each node on the integrated set of all change summary packets, (iii) all nodes disseminating the integrated

---

3 Although virtually all academic literature assumes such “instantaneous” provisioning, this may have more to do with convenience in traffic simulation exercises than real business requirements, especially given the resource implications of strictly on-demand (Erlang B-like) trunk provisioning for low blocking, as opposed to scheduled or batch provisioning. We note in at least one other line of work the recognition of how much more practical and cost-effective a scheduled or batch paradigm for provisioning could be [7].
Phase 3: (On-line O.R.) In the next phase all nodes locally and individually solve an instance of the optimal global incremental batch provisioning problem. The problem solved at this stage is an optimal model for simultaneously taking advantage of any resource releases in the last phase, routing the new working path requests, and re-optimizing the overall protection plan for the new set of paths in service, taking into account a variety of possible protection service classes, and optionally, either keeping existing protection arrangements in place and just adding the new ones, or globally reoptimizing the configuration of protection resources for the entire set of paths in service. The key property of this process is that although computed locally at each node, on possibly different platforms, every node arrives at an identical solution in terms of working channel, spare channel, and path assignment details. The computation is done in a way that every node locally obtains a solution that is identical in detail to that obtained at every other node starting with the same confirmed initial state and incremental change list. If desired, at the end of a pre-defined time allowed for solution of the global optimal reconfiguration problem, a checksum run on the solution can be disseminated for mutual confirmation of an identical solution at all nodes. In the event a node failed to meet this synchronization deadline for having completed solving the reconfiguration problem, or it publishes a checksum that mismatched others, the node is then diagnosed as having a control failure. It would then retain all connections presently in place and simply not participate further in the ongoing provisioning change processes. The time available for the local computation at each node is a large fraction of the basic cycle time chosen for the scheme, because it is overlapped with the next accumulation phase. For instance if the overall cycle is defined to start every 10 minutes, and 10 seconds is allowed for the change update dissemination, the 9 min, 50 sec is available for computation. (Although in the next step we will take back some of this for cross-connection activation time.)

Phase 4: (Local Activation of Globally Optimal Reconfiguration) On the onset of the next pre-defined time instant within the overall cycle, every node implements its part of the global reconfiguration solution. In other words each node locally makes and releases all the cross point connections within its own local matrix which correspond to implementing it part only of the globally optimal incremental reconfiguration solution. This includes creating cross-connections that transit connections for paths between other end-nodes and creating connections at source and sink nodes which connect the new path to the local access port of the user of the respective path. The end-user of the service is then notified that the path is now in service and the end-node cross-connects and/or the end-user equipment, by their own time and measure can then validate the path integrity. Note importantly (in the basic proposal) that no existing path in service (that was not released in the prior accumulation period) is ever disconnected or touched in this change-activation stage. After phase 4, a few seconds of guard-time can be invested, leading up to the instant at which the next accumulation interval will be ending, and the next change-request summary dissemination phase begins.

Several sub-problems can be identified as research topics under this concept. One is the definition of the incremental batch provisioning optimization problem, including multiple potential service priorities. There are also technical details to be developed concerning the time-synch method, the encapsulated accumulated change distribution method, and the local identical computation method. The format for notification to end customers could, however, be using existing user-network interface protocols and the details of customer end-to-end confirmation of the requested path also the same as used under GMPLS type service provisioning proposals. In later discussion we will consider some of these aspects further. The main aspect considered in the rest of this paper is the illustration of an example of the incremental batch re-optimization policy that could be used within the proposed framework, with a first demonstration of the added efficiencies of batch incremental provisioning over asynchronous individual arrival provisioning.

III. EXAMPLE OF AN INCREMENTAL OPTIMAL BATCH Provisioning Model

For study purposes assume connection requests arrive at random at each node, under standard memoryless Poisson arrival/departure assumptions and requires a working path routing and shared backup path protection for survivability. An Integer Linear Programming (ILP) model for optimal “green fields” design to serve a set of demands under SBPP is available from [3]. The optimization model in [3] is adapted here to define the incremental batch reconfiguration problem that is given in Appendix A. It embodies the following main ideas to adapt it to the context of incremental re-optimized batch provisioning with pre-existing capacities and connections and protection arrangements already in service. When executed, it takes into account (i) the set of all existing connections in progress that continue through to the next period (and are not disturbed at all), (ii) the set of all new connection requests to be served, and (iii) the new set of unused capacities on all links which includes any channels released by departures in the past operating period. It then allows for any desired relative emphasis on the simultaneous objectives of (i) maximizing the number of new demands served, (ii) minimizing the working resources allocated to new demands, and/or (iii) minimizing the total resources used for protection. Depending on relative weightings of parameters in

---

4 The postulated node-failure scenario is not particular to this scheme. It corresponds to the same node having a “brain failure” under present concepts of asynchronous GMPLS type provisioning operations. At worst the node would simply stop being available to participate in further ongoing provisioning requests.
the model, a range of operating policies from “serve new demands at all costs” \( (\alpha = 1, \beta = 0, \gamma = 0) \) to “serve new demands but conserve capacity used for their routing” \( (\alpha = 1, \beta > 0, \gamma = 0) \) to “serve new demands but only if it is not expensive to protect all demands in the new configuration” \( (\alpha = 1, \beta = 0, \gamma > 0) \), and so on.

An addition feature of the incremental batch reconfiguration model is a choice as to whether a general re-optimization of all backup resources for protection is permitted or not. This is a reasonable and possibly powerful option in any survivable network because this only re-arranges the assignments of spare capacity to implement the full level of protection desired. This does not imply that working paths are touched in any way, just the pre-plans for their protection may be either completely globally reoptimized, or left as is and only new paths backup arrangements optimized. The option to globally reoptimize backup paths at the time of an incremental batch provisioning update provides possibly large opportunities to release new operating capacity at each interval. It is not an option that can be as easily considered in independent asynchronous arrival provisioning because each end node pair only controls the backup arrangements for their own paths and there is no single defined time at which an entire reoptimized backup plan could be globally adopted. In this scheme, however, global updates to the backup plan just become part of the new configuration data solved for by every node and “switched into” at the next global time tick.

IV. SIMULATION OF INCREMENTAL OPTIMAL BATCH
PROVISIONING

To study and demonstrate the efficiency benefits of incremental optimal batch provisioning, a simulator was implemented by Zsolt Pandi during a COST 270 “short term scientific mission” to TRLabs with the aim of testing the behavior and performance of networks working according to this proposal. The complete simulator includes random connection arrival and departure generating process on each node pair, and the “change accumulation” process at each node. It then assumes the dissemination phase and uses an AMPL/CPLEX solver to determine the optimal solution of each successive incremental batch provisioning problem on the network as a whole. A series of experiments was carried out to demonstrate the benefits of the proposed mode of network operation. A few of these results are now presented.

For the experiments a European WDM network topology and a sparser version of itself (obtained by deleting some links) were used, as shown in Figure 1. The original topology has 19 nodes and 39 links (average node degree ~ 4.1). The sparser topology has 24 links (average node degree ~ 2.5). Each link is assumed to have 4 wavelength channels, and full wavelength conversion capabilities are assumed at network nodes.

The offered traffic is generated either uniformly or in a spatial pattern of intensity on each node pair with a Poisson arrival process. Connection holding times are exponentially distributed with a mean of one time unit so the arrival rate can be varied directly and the offered network load per node pair in Erlangs is numerically the same as the arrival rate.

V. RESULTS AND DISCUSSION

In a first set of trials, load was uniform on all node pairs, to simulate ongoing network operation under a constant moderate load. The benefits of incremental batch provisioning were particularly evident in the sparse network. Figure 2 shows the results in the sparse topology at 22 Erl (total network load). (Intuitively, this means that there are ~22 paths in service in the network on average and they and their shared backup protection paths are all supported over the four channels on each edge.) Figure 2 and the others of this type show the total count of connection blocking events through the simulation time. The solid line represents blocking under one-by-one handling of arrivals. The other lines represent to blocking total history of batch incremental provisioning with the accumulation interval varying from 0.2 to 0.4 of the average connection holding time and the option to re-optimize protection sharing relationships at each stage. The benefit seems especially visible in the sparse topology where routing decisions are in a sense more critical to get right if the network performance is to be its best.
Another scenario when incremental batch provisioning is expected to perform better is when the network has to handle changes in the statistical parameters of the load pattern. Two types of such changes may be imagined for test cases:

1) evenly distributed connection demands, temporary increase in arrival rate
2) temporary change in spatial distribution of connection demands, constant arrival rate

Under a temporary but spatially uniform “overload” situation, the benefits of incremental batch provisioning are illustrated in a test case where we simulated all nodes undergoing the same Erlang load intensity involving a temporary general overload. In this simulation the overall load was a baseline of 15 Erlangs for eight time steps, in the middle of which for two time periods the load increased to 120 Erlangs. The purpose of the experiment is to test the reaction to a temporary overload, under batch provisioning compared to individual provisioning. Figure 2 shows that under the baseline of 15 Erl, neither approach is blocking. Then, when the intense overload begins, blocking counts start rising steeply, but by the time the whole transient has passed, batch handling with re-optimization at each step resulted in significantly greater load carrying ability in the network through the transient.

Next, to simulate a spatial as well as temporal dynamic evolution in demand we forced modulated the pair-wise arrival rates individually as follows. The total connection arrival rate on the network as a whole remains constant but in end node pairs are grouped into certain spatial orientations and the load on these groups is time-varying. For example, in Figure 4 “northern” nodes have an arrow that points downwards, while southern nodes have an arrow pointing upwards, and a node-pair in the North-South group has one end node from each set. To illustrate Figure 3 shows the N-S and E-W spatial groupings as defined for tests in the normal topology.

Figure 4. Illustrating switched directional constraints on load used to simulate spatial transient evolution of demand pattern in the experiments.

Figure 4 shows the results of a spatial load evolution experiment carried out in the normal topology with a total load of 30 Erl which is “switched” to flowing in three distinct patterns through the experiment: [0-3]=even, [3-3.5]=North-South, [3.5-4]=East-West, [4-]=even. Figure 4 shows the superior ability of batch incremental provisioning to adapt to the time-and-space evolution of the demand pattern. The differences are even greater in the sparser topology.
VI. CONCLUDING DISCUSSION

We have proposed a new framework for network operations in the face of randomly arriving requirements for protected (or non-protected) connections through a transport network. Like the current “peer-to-peer” GMPLS-based concept for dynamic service provisioning, this scheme is also without any dependence on centralized network control, but it has the following further advantages: (i) no critical data dissemination or database synchronization occurs under real-time pressure, (ii) no data dissemination or state update occurs except at pre-defined times, in a summarized way, with robust error detection, (iii) groups of path change requests (new arrivals and departures) are taken into effect as a group and their treatment optimized, (iv) if backup reconfiguration is permitted, at each step even greater efficiency gains are possible by re-optimizing and reconfiguring the protection capacity planning at each step. Results show that efficiencies and performance gains can be made with the batch incremental optimization approach, especially in sparse topologies facing an uncertain time-space demand pattern.

The price to access the benefits of eliminating the data-base hazard and realizing the added efficiencies and adaptability shown above are:

(i) accepting a dynamic service offering model where there is a small delay in realizing provisioning requests at the transport path level, or such requests are pre-scheduled.

(ii) determination of suitable techniques for solving instances of the global optimization problem so that on every platform the outcome is not just equivalent but identical at the level of detail needed to correctly assign port numbers and so on to correctly realize the end-to-end paths. This is an aspect of ongoing research which clearly seems solvable. Some ideas are discussed below.

In closing, let us discuss a few questions this concept raises. One is to ask “How critical is the time synch?” There are two places in Figure 1 where we have indicated that the time coordination between nodes seems important. One is in closing off the accumulation intervals. Ideally, this happens at the same absolute time at all nodes, so that the next state the network enters reflects exactly the set of new provisioning requests that arrived at nodes over the same preceding intervals of the networks life. Upon reflection, the simultaneity of this event at all nodes is not actually that critical, however, because request (or release) events pertaining to a node-pair actually arrive at only one of the two nodes in the pair. Therefore even if the interval closes early or late at one node with respect to another, the worst-case effect is only that a request may have to wait until the next interval to be provisioned. More generally, a spread of the exact times of internal events at nodes of a even few microseconds is not a serious impediment, because this time is still vanishingly small compared to the holding time of connections or the interarrival time of new connection requests.

And, overriding everything is the consideration that nodes will not proceed to make any actual network changes until the positive indication of coherent global state is confirmed everywhere. Therefore, delay of event handling into the next cycle is strictly possible, but deleterious uncoordinated changes on the network by asynchronously acting nodes cannot happen in this scheme.

The other point in the cycle where time precision is desirable, but again not actually crucial (in the sense that things would fail otherwise), is the instant at which each node puts its own local cross-connections into service. If these actions are simultaneous everywhere, then the new paths in service appear to form more instantly, in parallel. But a spread in exact this of cross-connect completion of a few microseconds is also of no practical concern. One aspect in which the precision could be argued to be of technical concern is when the protection arrangements have been re-optimized in the preceding computation phase and are now to be put into effect in conjunction with activating the new service paths as well. Here, if one node is acting say 1 millisecond behind the others, then there could be a 1 millisecond gap in protection coverage of some existing or new paths. In practice, however, this is a trivial amount of unprotected time and familiarity with the level of precise time accuracy available in today’s network actually suggests that the worst-case coverage gap due to this consideration is more like microseconds of not milliseconds.

Another question for consideration is based on recognizing that different nodes might not all use the same computing infrastructure, so it might not always be assured that even with identical input data, and an identical problem statement, that the reconfiguration solution at each node would be identical in detail to that at other nodes. There may be solutions which are equivalent in “cost” but not identical in that they all pertain to a different detailed assignment of ports, wavelengths, time-slots, etc. This could, however, be addressed in one of several ways in a system implementation. First all nodes may indeed be required to have the same CPU types. If not the same CPU types, they may be required to run identical software (for instance FORTRAN implementations have been standardized by IEEE in a way that requires identical numerical behaviors on any compliant platform.) Another approach is even available that the level of the optimization model definition itself: by adding uniqueness-forcing details to the problem definition (for example adding a second objective function term to be minimized which is the product of a small number and the sum of all channel and port numbers used in the solution). Another method is the addition of unique dithering
noise on cost coefficients in the problem model or use of systematic port numbering schemes at nodes. Another simple approach if needed is that if multiple “cost equivalent” but not identical solutions exist then a checksum can be run the sum of all identity number of channels and ports employed and the convention adopted that all nodes select that with the lowest checksum.) During review of this paper, one reviewer even proposed within the spirit of this framework “why not even relax the uniqueness constraint and use the network computing power so as to explore the search space more efficiently. In this context each node would compute a solution and those solutions would be disseminated to the other node, each node keeping the best solution.” Also expressed was the view that “a single proven implementation in a high level language” run on all nodes would even be a suitable solution in practice. So it seems clear that there are many ideas and approaches to the identical computation problem and even some straightforward solutions may be acceptable. The overall scheme does not thus seem to encounter any “show-stoppers” on this account.

A final closing comment: This is a proposal made in the spirit of many recent calls by scientific agencies for research on entirely new ideas about how to operate future communication networks. It is hoped it is seen as contributing ideas towards that end even though many may disagree at present with notions such as batch provisioning with a slight user delay for lightpaths or the computational power to solve ILP models in each node as opposed to centrally, and so on. In addition, space has limited simulation results of the adaptability and resource efficiency to only two network cases, but this suffices to illustrate that benefit exists in principle. Further work can study a wider range of network, traffic, and batching scenarios. Many detail changes would also obviously be possible while still realizing this same overall new framework. Philosophically it can also be noted that in a sense the proposal amounts to operating an entire communication network as a single large (clocked) digital logic circuit, from a control configuration standpoint.

ACKNOWLEDGMENT

The simulation results presented in this paper were obtained for this work through collaboration with Zsolt Pandi who was then with Budapest University of Technology, Budapest, Hungary on a research visit to TRLabs following conception and internal documentation of the basic “synchronous distributed optimal batch provisioning” system concept at TRLabs.

REFERENCES


APPENDIX A: INCREMENTAL BATCH REOPTIMIZATION MODEL USED IN THE EXPERIMENTS

# Formulation for single priority incremental batch change optimization with SBPP protection adapted from the static optimization model for SBPP in [3]. Zsolt Pandi and Wayne Grover, TRLabs, Edmonton, March 2006.
set LINKS; set PATHS; set LINKS_OF [PATHS] within LINKS; # set of demands already in the network set D_OLD; # set of new demands; set D_NEW; # sets of links of working lightpaths of demands already in the network set W [D_OLD] within LINKS; # set of backup path candidates for demands already in the network set B [D_NEW] within PATHS; # set of (working,backup) candidate pairs for new demands set WB [D_NEW] within [PATHS, PATHS]; # free capacity on links param S {LINKS}; # mapping function between opposite links; param OPPOSITE {LINKS}; # weight of number of total new demands served in the cost function; param ALPHA default 1; # weight of working resources allocated to new demands in the cost function; param BETA default 0; # weight of total allocated backup resources in the cost function; param GAMMA default 0;

# Y[d,w,b] binary decision variable: # 1, if new demand d uses (w,b) working-backup path pair, 0 otherwise
# X[d,b] binary decision variable: # 1, if old demand d uses backup path b, 0 otherwise
var Y {d in D_NEW, WB[d]} binary;
var X {d in D_OLD, B[d]} binary;

# OBJECTIVE FUNCTION
maximize Total_served_minus_resources:

ALPHA * (sum {d in D_NEW, (w,b) in WB[d]} Y[d,w,b] - BETA * (sum {d in D_NEW, (w,b) in WB[d]} Y[d,w,b] * card(LINKS_OF[w]) - GAMMA * (sum {i in LINKS, j in LINKS: i <> j and i <> OPPOSITE[j]} (sum {d in D_OLD, b in B[d]: (i in W[d] or OPPOSITE[i] in W[d]) and j in LINKS_OF[b]} X[d,b] + sum {d in D_NEW, (w,b) in WB[d]: (i in LINKS_OF[w] or OPPOSITE[i] in LINKS_OF[w] and j in LINKS_OF[b]} Y[d,w,b]) / (card(LINKS) - 2);

# CONSTRAINTS
# constraint 1: each old demand must have exactly one backup path
subject to Served {d in D_OLD}: sum {b in B[d]} X[d,b] = 1;
# constraint 2: new demands may use at most one eligible working-backup path pair
subject to At_most_one {d in D_NEW}: sum {w,b} in WB[d] Y[d,w,b] <= 1;
# constraint 3: the capacity used on link j if link i fails must not exceed Sj]]
subject to Cap_share {i in LINKS, j in LINKS: i <> j and i <> OPPOSITE[j]}:
sum {d in D_OLD, b in B[d]: (i in W[d] or OPPOSITE[i] in W[d]) and j in LINKS_OF[b]} X[d,b] + sum {d in D_NEW, (w,b) in WB[d]: (i in LINKS_OF[w] or OPPOSITE[i] in LINKS_OF[w] and j in LINKS_OF[b]} Y[d,w,b] <= S j];

#