**A Deep Inspection of**

**High-Performance Bandwidth Scheduling Algorithm Complexity and Design**

***Research by:***

***Karan Kohli***

***Michael Mar***

***Ibraheem Saleh***

*Term Paper*

*November 30, 2017*

*37-pages*

**A Deep Inspection of High-Performance Bandwidth Scheduling Algorithm Complexity and Design**

|  |  |  |
| --- | --- | --- |
| Karan Kohli  kkohli@cpp.edu | Michael Mar  mhmar@cpp.edu | Ibraheem Saleh  iysaleh@cpp.edu |

**Abstract**

The increasing number of large scale data in applications onset by technological advancements and increased human-computer integration in various public and private sectors has prompted research into how to handle the rapid transfer of terabytes and eventually petabytes and exabytes of data. Investigation of a number of high performance research network testbeds and production networks have demonstrated the capability of provisioning dedicated channels for high speed data transfer in support of large scale scientific applications. Each dedicated channel in these networks typically consists of one or more physical links that are shared by multiple applications through advanced reservations. In this paper, we investigate six bandwidth scheduling problems and their associated optimal, and, in some cases, heuristic algorithms, and their related time complexities, all aimed at the goal of minimizing the total data transfer time under different path switching and bandwidth constraints. The bandwidth scheduling problems we investigate include fixed path with fixed bandwidth (FPFB), fixed path with variable bandwidth (FPVB), variable path with fixed bandwidth with a negligible path switching penalty (VPFB-0), variable path fixed bandwidth considering path switching (VPFB-1), variable path with variable bandwidth with a negligible path switching penalty (VPVB-0) and variable path with variable bandwidth while considering the cost of path switching (VPVB-1).

1. **Introduction**

With the advent of the era of big data, a significant number of science, engineering and business entities have begun to generate colossal amounts of digitally stored information which must be shared between sites in faraway remote-locations. Traditionally, these entities would simply establish internet-based connections to their remote sites and transfer their data as needed, however, since the data that is being transferred is now in the order of hundreds of terabytes, petabytes or soon even exabytes of data, these widely available internet resources simply are not performant enough for the data transfer task at hand [6]. In recent years, dedicated high-bandwidth links have begun to be laid my numerous organizations and wide-area-networks are being created that run apart from the Internet. For example, the Department of Energy maintains an “UltraScience Net” that runs Dual 10Gbps lines between Seattle, Chicago, Sunnyvale, Atlanta as shown in figure 1 below. [2]

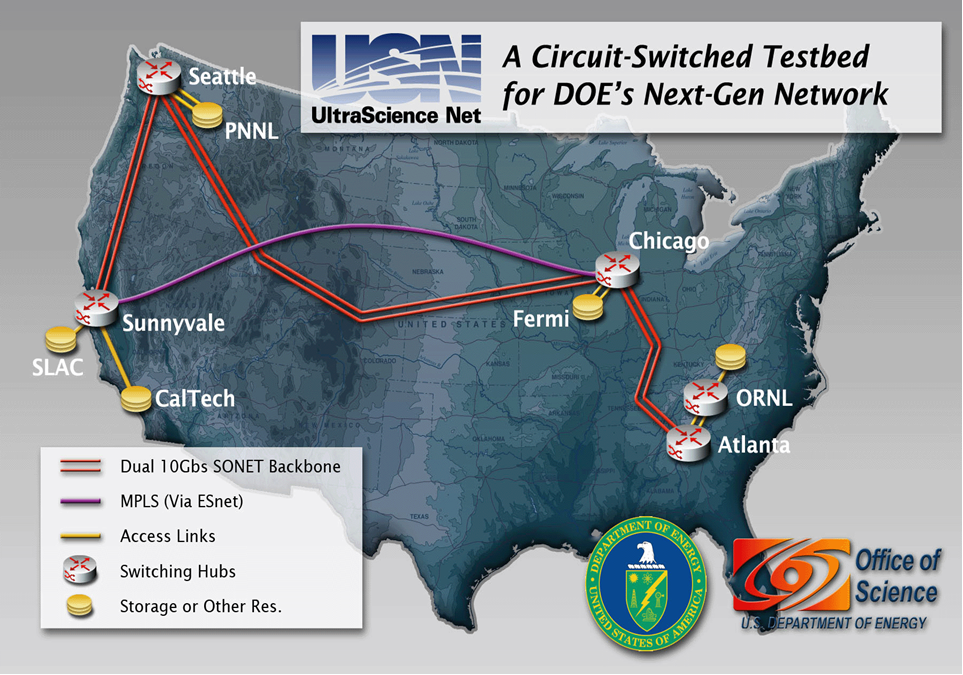


Figure 1: Diagram of the Department of Energy’s UltraScience Net that connects various important sites with dedicated high bandwidth links. [2]

Among other things, the department of energy uses this dedicated high bandwidth network to run environment simulations that utilizes datasets that can be over 20TB in size. These simulations often need to rapidly transfer data between nodes and can be controlled remotely from different parts of the country. Similar dedicated networks such as the User-Controlled Light Controlled Path (UCLP), Circuit-Controlled High-speed End-to-End Transport ArcHitecture (CHEETAH) [3], On-demand Secure Circuits and Advance Reservation System (OSCARS), Japanese Gigabit Network II, EnLightened (ENL) [4] and more have begun to be built throughout the world to meet the constantly growing data transfer demand. [6]

These high bandwidth connections achieve superior performance from traditional WAN type networks by utilizing scheduling and advanced bandwidth reservation techniques to control when and how fast data is transferred. This is established generally through a management framework layer, a “control plane”, that exist above the edge devices, core switches, and other network infrastructure components [6]. The management layer is responsible for reserving link bandwidth, setting up end-to-end network paths, and releasing resources when tasks are completed. Inside the control plane, there is a bandwidth scheduler that computes the network paths and allocates link bandwidth to meet specific user requests based on network topology and bandwidth availability. The scheduler only needs to know ahead of time, through advanced reservation, the size of the data transfer to be completed, a destination and, optionally, a deadline for when the transfer needs to be received at the destination node.

The design of efficient bandwidth scheduling algorithms is critical to maximizing the utilization of dedicated network resources and meeting diverse end-to-end transport performance requirements that can go beyond the capabilities of the classical Dijkstra and Bellman-Ford algorithms. In this paper, we investigate the following six bandwidth problems:

1. **Fixed Path with Fixed Bandwidth** (**FPFB**), which computes a fixed path with a constant bandwidth;
2. **Fixed Path with Variable Bandwidth** (**FPVB**), which computes a fixed path with varying bandwidth across different time slots.
3. **Variable Path with Fixed Bandwidth-0** (**VPFB-0**),which computes for a variable path in each time slot with the same bandwidth and ignores time-cost of path switching;
4. **Variable Path with Fixed Bandwidth-1** (**VPFB-1**),which computes for a variable path in each time slot with the same bandwidth and takes into account the time-cost of path switching;
5. **Variable Path with Variable Bandwidth-0** (**VPVB-0**), which computes the widest (highest bandwidth) path in each time slot and ignores time-cost of path switching;
6. **Variable Path with Variable Bandwidth-1** (**VPVB-1**), which computes the widest (highest bandwidth) path in each time slot and takes into account the time-cost of path switching;

The rest of the paper is structured as follows: Section II introduces the terminology and nomenclature that will be used throughout the paper and in the algorithms to describe the concepts, variables and methodologies which are pertinent to developing an understanding of the advanced reservation bandwidth scheduling problems which this paper seeks to address. Section III goes into significant detail for each of the six-bandwidth scheduling problems. For each problem space, a simple example problem is utilized and its solution presented to demonstrate how an algorithm that solves the problem might function. Then, the optimal algorithms and an explanation of how they work and their time complexities are discussed. If the optimal algorithm has an impractical time-complexity, IE it is not polynomial time solvable, then an efficient heuristic algorithm and its related complexity is also introduced to demonstrate that the problem-solution can be tackled in difficult real-world scenarios. Finally, section IV concludes the paper and section V mentions the references used for the purposes of this research.

1. **Terminology and Nomenclature**

It can be extremely difficult to comprehend what the algorithms are doing and what abbreviations are referring to simply due to the sheer volume of the ones that are utilized throughout this paper. The purpose of this section is to create a kind of index reference with definitions for the common terms and variables that can be backtracked to by the reader when they are trying to understand the algorithms detailed throughout this paper.

|  |  |  |
| --- | --- | --- |
| **Concept/Term/Abbreviation** | **Definition** | **Additional Notes** |
| G = (V,E) | Graph G consists of a set of Vertices V and Edges E | Each graph has a set of nodes (n) belonging to V and links/edges (m) belonging to E |
| l ∈ E | Each link l belongs to the set of Edges E |  |
| m | m Links/Edges in a graph | Used in O-complexities |
| n | n nodes in a graph | Used in O-complexities |
| TB | Time-Bandwidth | A 3-tuple (tl[i], tl[i+1],b1[i]) to represent the residual bandwidth that can be transferred over link l during time slot i (The subscript is an L not a 1) |
| T | Time Slot Links | Used to represent total number of time slot links |
| ATB | Aggregated Total Bandwidth list | Stores the residual bandwidths of all links in each intersected time-slot. |
| 𝛿 | Data Size | Size of data to be transferred from source to destination nodes |
| β | Bandwidth of widest path for Vs to Vd during timeslot i | Used in VPVB and more |
| β(0/1 TB) | Number of time slots in which total bandwidth is 1 | Used in 0/1 Total Bandwidth problem! |
| MPA | Maximum Permutation Algorithm | A recursive divide and conquer algorithm to help solve a subset of FPVB |
| 𝜏 | Path Switching | Used to indicate the number of path switching during an algorithm. |
| p | Start time | Start time for data transfer |
| q | End time | End time for data transfer |

Table 1: Terminology and Variable Definitions used throughout this research.

1. **Algorithms and Analysis**

Any algorithm that is suited to solving the bandwidth scheduling problem that is at the heart of this paper must define a set of paths from the given source node to the given destination node and must also return the bandwidth utilized along each link at each timeslot for the returned set of paths. Depending on the restrictions assumed by the bandwidth scheduling algorithm, the complexity and dynamic nature of what gets returned may vary. With the current state of high-speed dedicated network switches in place, there are six-different path and bandwidth problem-assumptions that all have varying degrees of freedom: Fixed-Path-Fixed-Bandwidth, Fixed-Path-Variable-Bandwidth, Variable-Path-Fixed-Bandwidth-0, Variable-Path-Fixed-Bandwidth-1, Variable-Path-Variable-Bandwidth-0, Variable-Path-Variable-Bandwidth-1.

To demonstrate the optimal solutions unique to each scheduling problem, we will utilize the following network graph structure for an example problem with the following conditions: data size 𝛿=8, bandwidth transfer start time t=0, the path switching cost 𝜏=0.1 and we want to transfer that data from the source node Vs to the destination node Vd in the most efficient way possible.

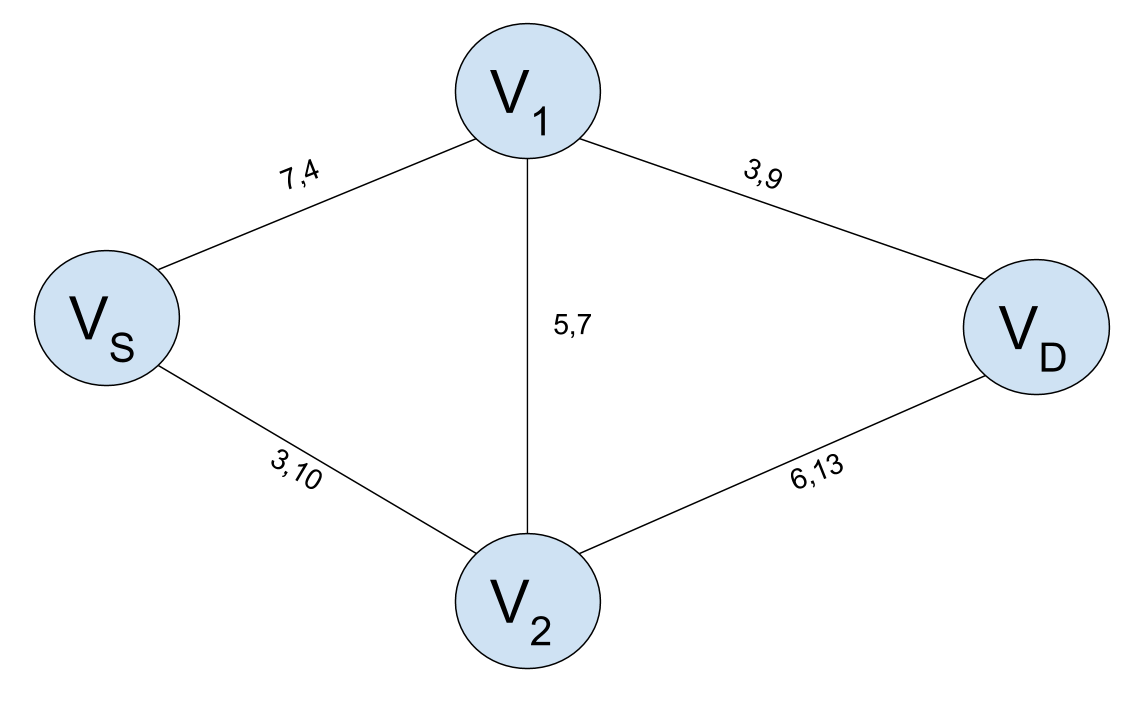


Figure 2: Example Bandwidth Scheduling problem graph with four nodes. Vs is the source node and Vd is the destination node. Each link connecting the nodes has a set of available bandwidth values which correspond to the time slots (t1,t2).

* 1. **Fixed Path Fixed Bandwidth (FPFB)**

The Fixed Path Fixed Bandwidth (FPFB) represents the most precise transport conditions by fixing path and bandwidth. It limits the bandwidth to a fixed value so that the data transfer can be started at the first possible slot. This scheme is most suited for the transport methods and can be stabilized at a fixed target rate.

FPFB computes a fixed path from Vs  to Vd with a constant bandwidth. Its optimal path, given the example problem from figure 2, is Vs-V2-Vd with the minimum data transfer endtime 1+8/10= 1.8. The transfer must start at time point 1 instead of 0 because the available bandwidth 10 in the second slot is much larger than the first time slot.

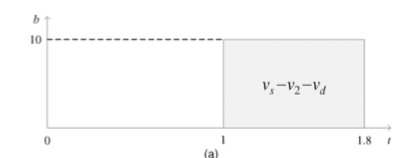


Figure 3: Diagram of optimal solution to example problem with FPFB Constraints [6]

* + 1. **Algorithm Example**

The reasonable solution to a bandwidth scheduling problem consists of a number of scheduling components. Each component is a 4-tuple that consists of a path from source to destination, the bandwidth along its path and the data transfer start and end point. But a reasonable solution to FPFB has one scheduling component.

* + 1. **FPFB Optimal Algorithm**

FPFB takes the input graph G= (V,E) with an ATB list for all links l ∈ E, its source Vs and the destination Vd  and the data size is 𝛿, it computes the fixed path with a constant bandwidth from source to the destination.

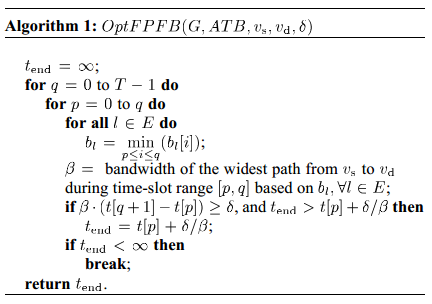


Figure 4: Optimal FPFB Algorithm as proposed by Lin et Al [6]

An optimal complete Start Algorithm for FPFB, referred to as Opt F P F B, whose pseudocode is provided in the above algorithm. As the output of this algorithm is minimal data transfer end time tend but it may or may not be always optimal to start data transfer immediately, because it varies the transfer time slot q and checks whether there is any feasible p such that the size of 𝛿 can be transferred during the time slot range [p.q]. If there is no existence of the feasible path the algorithm repeatedly increase q by 1, or otherwise it computes the optimal start time p and data transfer end time tend by considering all the possible values of p and terminates.

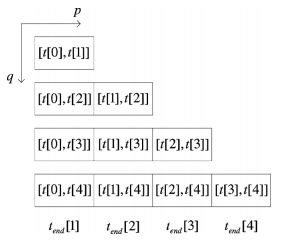
****

Figure 5: Example of solving an FPFB problem logically through steps. [6]

* + 1. **Complexity Analysis**

The time complexity of this algorithm is O(T2. m . logn + T3 . m) which indicates that it can be solved in polynomial time and is feasible under large scale FPFB bandwidth scheduling applications.

* 1. **Fixed Path Variable Bandwidth (FPVB)**

The Fixed Path Variable Bandwidth (FPVB) problem imposes the restriction that any feasible solution to the bandwidth scheduling task is not allowed to switch paths after the bandwidth transfer begins, however, the transfer-rate can be dynamic and change per any given time-slot. Since we assume that we are only trying to find the best solution for the current problem and not solving the task of multi-job scheduling, for any given path at any given time t, a bandwidth scheduling algorithm job will schedule the maximum available bandwidth along a path--IE the maximum bandwidth of the slowest link on the path. This fact is central to the best solution that can be discovered for the FPVB problem

* + 1. **Example Optimal FPVB Solution**

To best understand how an optimal solution to the FPVB problem might be determined, we look towards the bandwidth graph shown in figure 2 above and assume the problem details as follows: data size 𝛿=8, bandwidth transfer start time t=0 and that we want to transfer that data from the source node Vs to the destination node Vd in the most efficient way possible. To solve this, we do an exhaustive iteration through all the possible solutions assuming FPVB problem constraints.

Since there are only four-paths in total between the source and the destination node and FPVB forces a no path-switching constraint, if we assume that we utilize the maximum available throughput on each path during each available time slot, there are only four-possible solutions to this simple problem as demonstrated in figure 6 below.

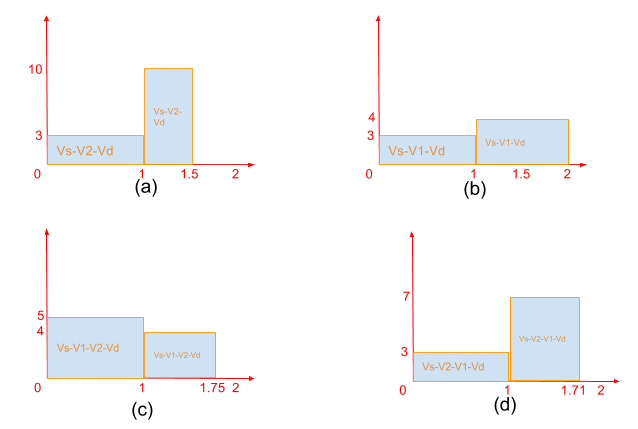


Figure 6: All possible solutions using FPVB constraints assuming the maximum available throughput is allocated at any given time slot on the path.

The optimal solution to the aforementioned problem given fixed path and variable bandwidth constraints can be given seen by chart d in figure 6 above.

* + 1. **FPVB Optimal Algorithm and Complexity Analysis**

In order to formulate an optimal scheduling algorithm for the FPVB problem proposed by Lin et al [6], it is first necessary to understand the base Maximum Permutation Algorithm (MPA) which serves the purpose of determining if there is a path from a source node Vs to a destination node Vd such that the data size can be transferred within the time-slot range [p,q]. As seen in the MPA algorithm depicted in figure 6 below, MPA starts from time-slot p and recursively calls itself with a different sub-graph containing just the bandwidth values for the current time-slot.

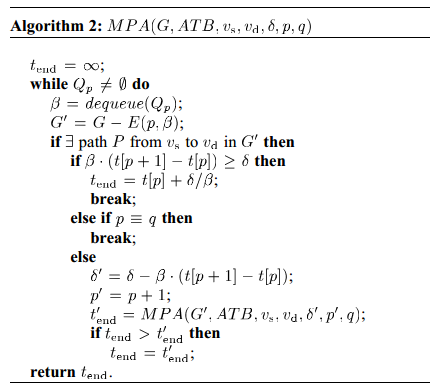


Figure 7: Maximum Permutation Algorithm as formulated by Lin et Al in [6]

Each iteration of MPA subtracts the maximum possible data that can be transferred over the current time slot and then passes on the remaining data size to be transferred to the next recursive call of MPA. If it reaches the final timeslot and the remaining data size is non-zero, MPA returns infinity. If it ever reaches a data size of zero, it returns the exact point in the current timeslot that this end-condition was reached. The Maximum Permutation Algorithm is a divide-and-conquer technique that is inefficient but effective in finding the time point, if it exists, at which a solution for the bandwidth graph for a given data size can be found. Since MPA ultimately recursively calls itself q-p times and must iterate through all of the edges in each recursive call, the computational complexity of MPA is O(mq-p+1).

Now that we understand how the Maximum Permutation Algorithm can determine if there is a path from the source to the destination nodes for any given time-slot range, it is only logical that we move forward to understand how an algorithm might be formulated to use MPA and find the optimal solution to any given FPVB graph problem. In figure 8 below, Lin et al [6] propose an algorithm that returns the earliest time that a datafile may be transferred over a network given the graph, and aggregated total bandwidth list, the source and destination nodes and the datafile size.

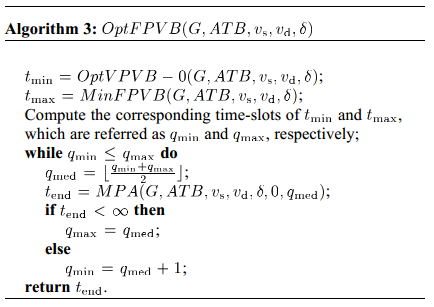


Figure 8: Optimal FPVB algorithm as formulated by Lin et al [6]

Optimal FPVB works by first calculating the minimum and maximum transfer times that could possibly be required to transfer the input datafile size over the network from the source to the destination nodes using heuristic algorithms that are described later. Once the algorithm has that information, it can then do a binary-search over the graph by calling MPA over just the tmin to tmax time interval. As previously demonstrated, MPA will return the minimum data transfer end time for any given datafile and time-slot range. Since OptFPVB is utilizing a binary-search approach over the graph, its worst-case time complexity is O(log T) where T is the number of time slots for any given graph. The total complexity for OptFPVB is O(mT.logT). Since the complexity of OptFPVB is exponentially dependent on the number of timeslots which could be necessary for the solution to a given problem, it is clear that it is impractical to use it in large-scale real-world scenarios.

* + 1. **Optimal FPVB is NP-Complete**

As demonstrated in the previous section, the algorithm that Lin et Al formulated for the Optimal FPVB solution has an exponential time-complexity that is infeasible to apply in significantly challenging bandwidth scheduling problems with large numbers of time slots. The purpose of this section is to delve further into the FPVB problem and demonstrate that it is, in fact, NP-Complete. To prove the NP-Completeness of any problem, you must first show that the decision version of the problem is in NP and then secondly prove that it is reducible to a known NP-Complete problem.

* + - 1. **Decision Version of FPVB is in NP**

The decision version of the Fixed-Path-Variable-Bandwidth problem can be stated as follows:

***Input****: Given a graph G = (E,V) with an ATB list for all links l ∈ E, source Vs, destination Vd, and data size 𝛿...*

***Question****: Does there exist a fixed path from Vs to Vd with varying bandwidths across multiple time-slots such that the data can be completely transferred along the path during the time interval [0,tend]?*

We can easily show that the decision version of FPVB exists in NP by producing a non-deterministic polynomial time algorithm that solves the answer as follows:

*For Every ${Path} from Vs to Vd in G:  
 guess\_max\_data\_size\_transferrable(${Path},0,tend) >= 𝛿*

* + - 1. **Decision FPVB reduces to a known NP-Complete Problem**

The decision version of the Fixed-Path-Variable-Bandwidth problem closely resembles the 0/1 Total Bandwidth (TB) problem that Guerin et al demonstrated to be NP-complete in [7]. Similar to the FPVB problem, the 0/1 Total Bandwidth problem, takes as input a graph G, an aggregated total bandwidth list ATB, a source node Vs and a destination node Vd. 0/1 TB also takes as input a tend and β which represents the number of time slots which all links have to have a value of 1-available. The 0/1 TB problem, however, differs in that each edge of the graph, the set of available bandwidths at any given time slot, can only be a 0, not available, or a 1, available. To demonstrate the NP-Completeness of 0/1 TB, [7] transformed an instance of the Satisfiability problem into an instance of the 0/1 TB problem as shown in figure 9 below.

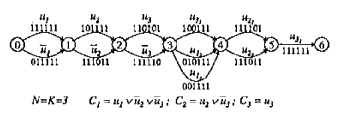


Figure 9: Satisfiability problem transformation to 0/1 Total Bandwidth Problem

The transformation works by mapping each of the variables in any given clause to the time-slot for an edge of the bandwidth network graph. A boolean true maps to a bandwidth-available 1 while a boolean false maps to a bandwidth-unavailable 0 in the TB problem. A path only exists between the source and destination at any given time-slot if the clauses in the transformed input instance are satisfiable for that time-slot.

Since the 0/1 Total Bandwidth problem is known to be NP-Complete, all that is left is to reduce it to the FPVB problem. To achieve this, construct an instance of 0/1 TB as (G, ATB, Vs, Vd, tend, β). From this instance of 0/1 TB, transform it into an instance of the FPVB problem as (G’, ATB’, Vs’, Vd’, tend’, 𝛿). In this case, G=G’, ATB=ATB’, Vs=Vs’, Vd=Vd’,tend=tend’ and β=𝛿*.* Beta easily transforms to the FPVB delta since there must exist a path from the source to the destination node where the bandwidth is 1 is essentially equivalent to the available bandwidth being equal to or greater than the datafile size for a given timeslot. The only challenging part of this polynomial transformation is that of the ATB lists which Lin et al diagram in figure 10 below.

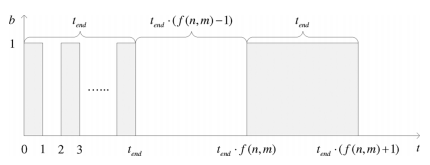


Figure 10: An ATB transformation for an FPVB instance.

The ATB transformation works by transforming a single time slot in the FPVB problem with a given available bandwidth to multiple time-slots in the 0/1 TB problem with 0 or 1 bandwidth. Since all of these transformations can be done in polynomial time and 0/1 TB has been proven to be NP-Complete, by reduction, FPVB must also be NP-Complete since FPVB can be solved in polynomial time if and only if 0/1 TB can be solved in polynomial time which would mean that P=NP.

* + 1. **Heuristic FPVB Algorithm**

Since the optimal solution to the Fixed-Path-Variable-Bandwidth problem is NP-complete which means that it does not have a an optimal solution solvable in polynomial time unless P=NP, the next logical step is to find a heuristic algorithm that produces a reasonable good-solution and has an acceptable time complexity so that it can be utilized in large bandwidth scheduling scenarios with numerous time-slots.

The efficient heuristic algorithm proposed by Lin et al [6] can be seen in Algorithm 5 minFPVB depicted in figure 11 below.

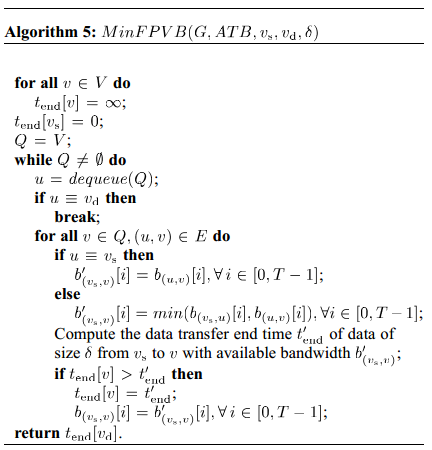


Figure 11: Heuristic MinFPVB algorithm as formulated by Lin et al in [6]

The heuristic MinFPVB function works in a similar manner to dijkstras algorithm in that it first computes the data transfer end time over every single link and assigns it as a weight to each link. It then modifies dijkstras to compute the narrowest path from the source to the destination on which the maximum weight is minimized. To make this heuristic more efficient, it considers the fact that bandwidth transfer rates are only as fast as the slowest bottleneck link on any given path. For each time step on each path, the bottleneck bandwidth from Vs to the current node b(vs,v)[i] is updated instead of computed separately on a per link basis. The total computational complexity for MinFPVB is O(m.(T+logn)) which means that it is a kind of pseudo-polynomial time algorithm that scales well in high-bandwidth multiple time-slot scheduling problems.

* + 1. **FPVB Performance Simulations - Heuristic vs Greedy**

In order to prove that the heuristic MinFPVB algorithm formulated by Lin et al [6] performs within reasonable bounds and obtains acceptable answers, they ran a series of simulations under 200 simulated networks with various reservation loads. As shown in figure 12, the heuristic algorithm achieves optimality at worst .955 percent of the time.

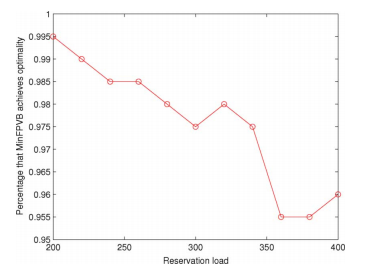


Figure 12: MinFPVB Heuristic algorithm percentage that it produced the optimal algorithm in a series of 200 simulated networks of 8 nodes and 12 links with various reservation loads. [6]

Then after, to demonstrate the performance gains of the heuristic algorithm, they executed both their greedy FPVB algorithm [6] versus the heuristic MinFPVB algorithm and compared both average transfer end time vs reservation load and average transfer end time vs the data size.

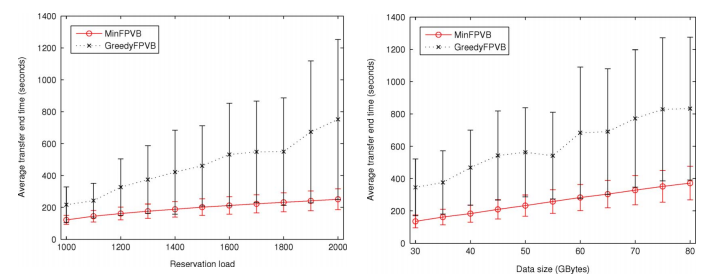


Figure 13: Average transfer end time versus Reservation Load and Data Size in 50 simulated networks of 50 nodes and 200 links. [6]

As shown in figure 13 above, the heuristic algorithm increasingly out-performed the Greedy algorithm as both advanced reservation load and the input data size increased. It is important to note that the optimal solution would take even longer and was infeasible to run with these simulated networks since it would simply take way too much time to execute.

* 1. **Variable Path Fixed Bandwidth-0**

Variable Path Fixed Bandwidth-0 is an algorithm that adjusts the path of transfer at different time-slots but maintains the bandwidth with no or negligible time delay as indicated with the “0”. Refer to the graph with the two time-slots, the widest path for the first time slot is vs-v1-v2-vd has the bandwidth of 5 while the widest path for the second time slot is vs-v2-vd with a bandwidth of 10 [8]. This results with a transfer rate of end time rate of 8/5=1.6.

* + 1. **Algorithm Examples**

The algorithm associated with VPFB-0 is the Optimal VPFB-0

VPFB-0 computes a set of paths from source to destination at different time-slots with a fixed bandwidth with a negligible time delay.

* + 1. **Algorithms - Optimal and Heuristics**

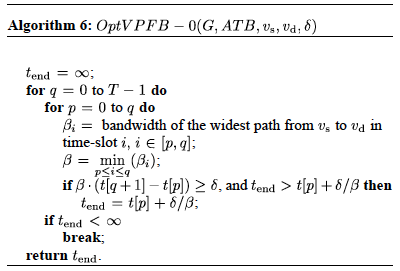


Figure 14: Optimal VPFB algorithm as formulated by Lin et al in [6]

As shown from Algorithm 6, the bandwidth is computed from the widest path from vs to vd in each time-slot with the time range [p,q]. It then computes the smallest bandwidth, which is considered the bottleneck bandwidth across the time-slots and is the fixed bandwidth for data transfer. The end time (q) is increased by 1 until the amount of data transferred up to q is greater than the data size 𝛿 and computes the minimal data transfer end time by considering all possible p values.

* + 1. **Complexity Analysis**

To calculate the widest path in each time-slot takes O(T\*m\*lg n). Since this is performed in advance, the total time complexity is O(T\*m\*lg n+T3).

* 1. **Variable Path Fixed Bandwidth-1**

In VPVB-1 the 1 indicates the switching is non negligible. Since the bandwidth of the widest path in the widest path is smaller than that in the second time-slot (see below graph), the path switching is performed at the end of the first time-slot [0.9, 1] and the data transfer end time is 1+ 3.5/10=1.35.

* + 1. **Optimal VPFB-1 Algorithm**

It considers the non-eligible positive path switching delay 𝜏 > 0.

B[p,q,k] be the maximum available bandwidth during the time slot range [p,q] with k path switching. The maximum amount of transferred data will be : B[p,q,k].(t[q+1]-t[p]-𝜏.k0 (1)

If the above max amount of time is greater than or equal to the size 𝛿, the data transfer end time is obtained :

tend=t[p]+𝛿/B[p,q,k]+𝜏 .k (2)

Following the concept of the dynamic programming, the recursive form of the optimal solutions to the subproblem, divide the problem of computing B[p,q,k] by the time slot after which the path switching is scheduled : B[p,q,k] = max { min(B[p,i,0], B[i+1, q, k-1])} (3)

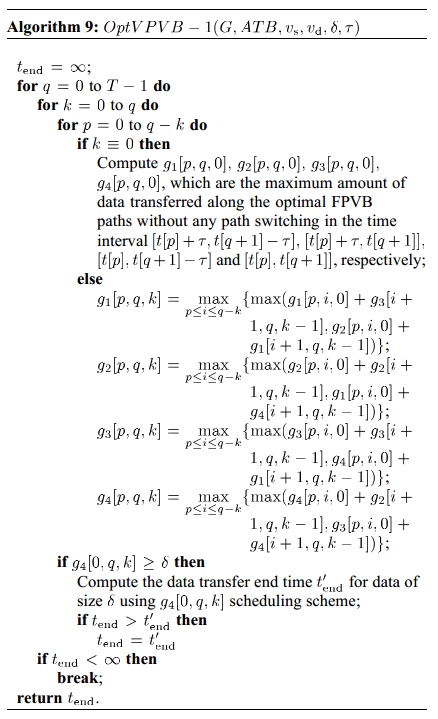


Figure 15: Optimal VPVB-1 algorithm as formulated by Lin et al in [6]

The pseudocode of optimal VPFB-1 is based on dynamic programming as it’s shown in the above algorithm. The data transfer end time slot q starts from 0, and for a given q, the algorithm computes 𝛃[p,q,0] as base condition when K= 0 and recursively puts 𝛃 [p,q,0] based on 3 when k>0. 𝛃[p,q,0] is computed using a tabular bottom up approach, as shown in the figure below. Where its values are stored in a three dimensional table and the shadowed entries in the table represents the optimal solutions to base condition when k=0. The curved lines indicates which entries are used for computing. The dynamic programming procedure repeatedly increase q by 1 and considers all the possible p and k values to compute the amount of data transfer defined in (1) until it is greater than or equal to the data size *𝛿,* in which the minimal tend is updated.

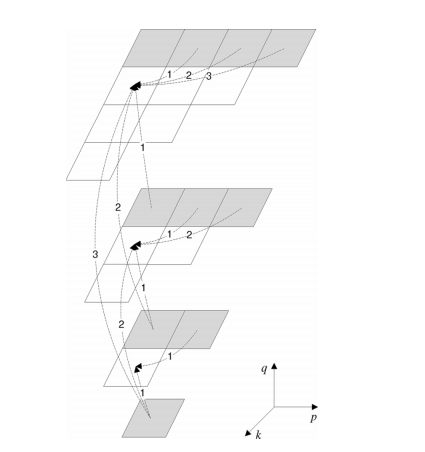
**

Figure 16: Dynamic programming procedure that computes 𝛃 [p,q,k] using a tabular, bottom-up approach [6]:

* + 1. **Complexity Analysis**

The time for computing the optimal solution to all base entries (K=0) O(T2.m.logn). When K>0 time is O(T4) total time O(T2.(m.lgn+T2)) in the worst case.

* 1. **Variable Path Variable Bandwidth-0 (IS)**

The Variable Path Variable Bandwidth-0 (VPVB-0) problem does not force any restrictions on path-switching or bandwidth changes at different time slots. It is important to not that any path switch or bandwidth variations happen at the time-slot borders and not in the middle of a time-slot. The 0 appended to VPVB refers to the fact that in VPVB-0 path switching is considered a negligible operation. Negligible path switching does not mean that there is no cost associated with the action, it means that the time-cost of path-switching is so insignificant compared with the time-cost of the data transfer that it is essentially ignorable in any final paths associated with the bandwidth scheduling task.

* + 1. **Optimal VPVB-0 Example**

To best understand how an optimal solution to the VPVB-0 problem might be determined, we look towards the bandwidth graph shown in figure 2 above and assume the problem details as follows: data size 𝛿=8, bandwidth transfer start time t=0 and that we want to transfer that data from the source node Vs to the destination node Vd in the most efficient way possible. To solve this, we do an exhaustive iteration through all the possible solutions assuming VPVB-0 problem constraints.

Since there are four-paths in total between the source and the destination node and that we assume that we utilize the maximum available throughput on each path during each available time slot, there are only four-possible solutions paths for the first time slot T[0] and then four-more possible paths to the second time slot T[0]. This means that there are 4x4=16 potential solutions to this problem given VPVB-0 constraints. It is easy to see that the optimal solution to this problem involves just taking the paths which consist of the maximum or widest bandwidth available for any given time slot. Thusly we only, need to consider 8 paths, 4 for each time slot, to know the most efficient way to transfer a file from the source to the destination. The optimal path for the example problem with VPVB-0 can be seen in Figure 17 below.

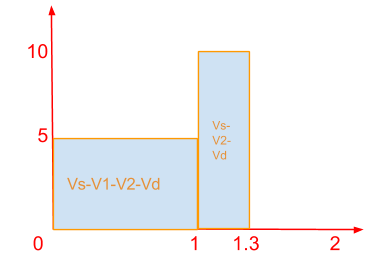


Figure 17: Optimal VPVB-0 solution given the problem (𝛿=8,𝜏=0.1,t=0) and the Network Graph depicted in figure ? above.

* + 1. **VPVB-0 Optimal Algorithm**

As demonstrated in the example problem in section i above, the optimal algorithm to the VPVB-0 problem is both extremely simple and polynomial time solvable.

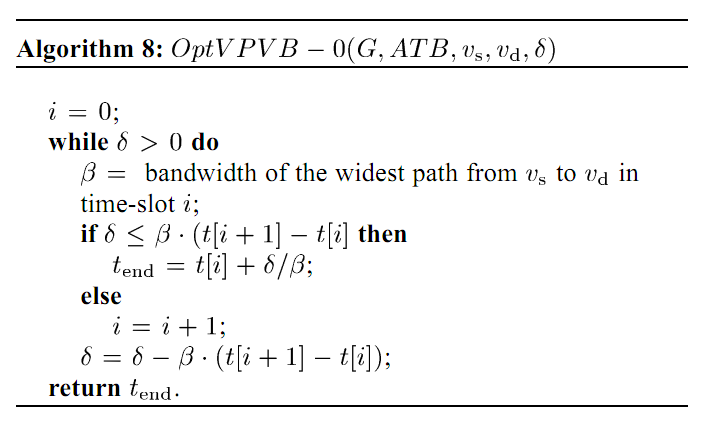


Figure 18: Optimal VPVB-0 algorithm as formulated by Lin at Al in [6]

OptVPVB-0 works by decomposing a network bandwidth graph of multiple time slots into multiple network bandwidth graphs, one for each time slot, taking the widest path of each subgraph and utilizing it to transfer the data at each time-point until the data is less than the available bandwidth on a given path. At that point, it returns the time at which the data would complete its transfer for the current time slot. A visualization of this can be seen in figure 19 below.

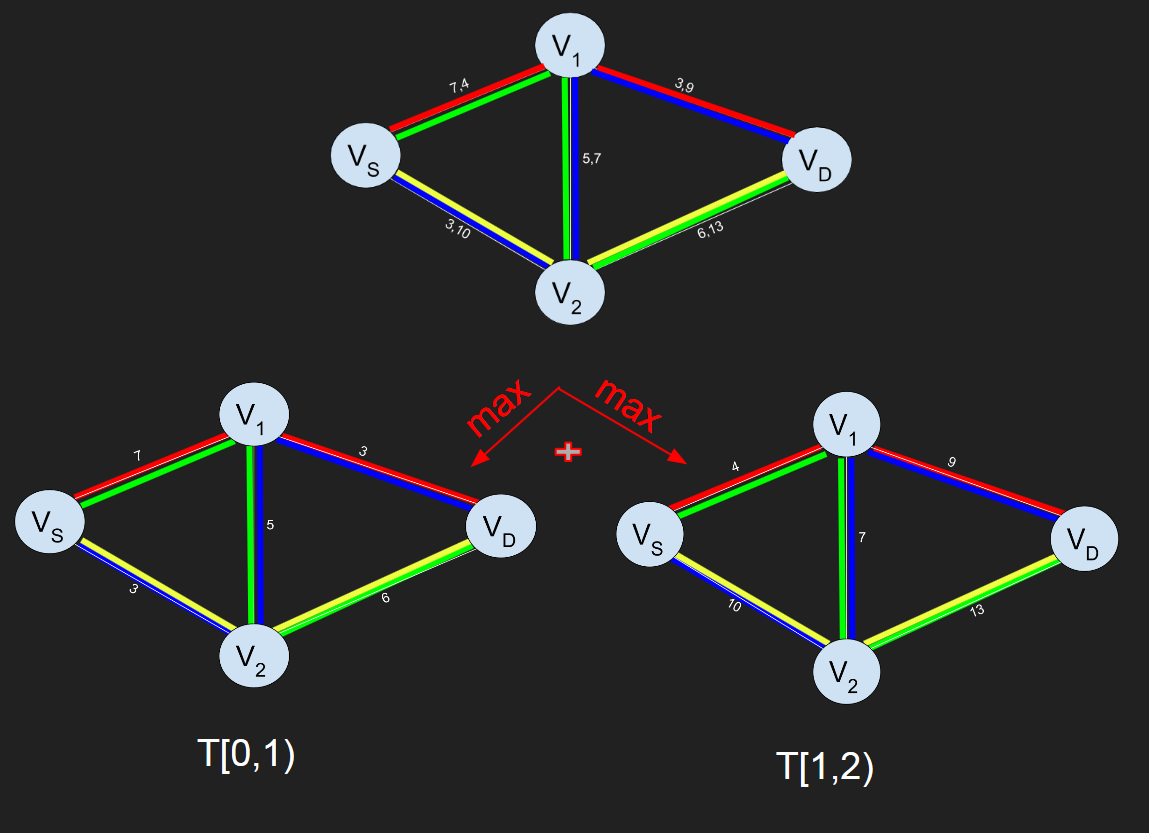


Figure 19: Visualization of OptVPVB-0 algorithm at work on example problem.

* + 1. **OptVPVB-0 Complexity Analysis**

The time complexity of performing the widest-path calculation at each time-slot is O(T.m.log n) since the algorithm must go through each time slot link once for every link and node. The algorithm for the Optimal Variable Path Variable Bandwidth-0 problem ultimately only needs to iterate through each of the possible paths from Vs to Vd once for every time slot so the overall time complexity is O(T.m.log n+T3). Clearly, optimal VPVB-0 is polynomial time solvable.

* 1. **Variable Path Variable Bandwidth-1 (MM)**

Variable Path Variable Bandwidth-1 occurs when the the time delay to

switch path (𝜏) is so large cannot be ignored and causes a negative impact on the performance.

* + 1. **Algorithm Examples**

The algorithms associated with VPVB-1 is Optimal VPVB-1(OptVPVB-1) and Minimum VPVB-1 (MinVPVB-1):

* OptVPVB-1 has a time delay that is so large that it causes a negative impact on the performance of the data transferred. When it occurs, VPVB-1 reduces to the FPVB algorithm.
* MinVPVB-1 is similar to OptVPVB-1 but uses the MinFPVB instead of the Maximum Permutation Algorithm (MPA) for the optimal algorithm when the base conditions of path switching (k) at zero. MinVPVB-1 computes gi[p,q,0] by using the input of MinFPVB to (G, ATB, vs, vd, p, q) and the output of MinFPVB to the maximal amount of data transfer during the time-slot range [p,q]
  + 1. **Algorithms - Optimal and Heuristics**

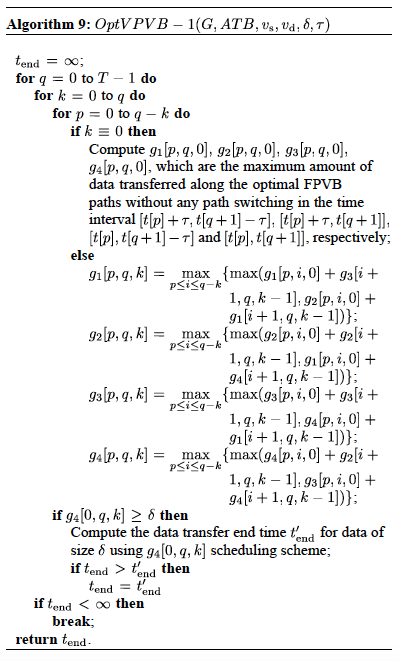


Figure 20: Optimal VPVB-1 algorithm as formulated by Lin et al in [6]

To prove that VPVB-1 is NP-complete and inapproximable, there has to be an optimal algorithm with exponential complexity for small-scale networks and a heuristic algorithm for large-scale ones [9].

A path switching can be performed either at the end of a time-slot or at the beginning of the next time-slot. There are four different data transfer scenarios in the figure below.

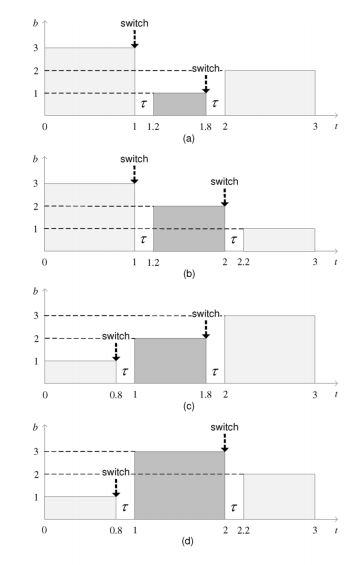


Figure 21: Data transfer in different path switching schemes for VPVB-1. Examples of computing (a) g1[p,q,k], (b) g2[p,q,k], (c) g3[p,q,k], and (d) g4[p,q,k], where p=1, q=1, and k=0.

The four different data transfer rates are embedded in Algorithm 9 in the else statement. The shaded area represent the data being transferred in the four different time intervals due to the different path switching schemes. The scenario in first graph (a) in figure 21 shows that the path bandwidth in time-slot 1 is less than that in the time-slot 0 and and 2, so both paths are switched in time-slot 1. The graphs b to d also show similar scenarios. Graph d shows the data size of 𝛿 being equal or greater than the data being transferred. The data transfer can be computed from the time-slot q but the k path switching during this time period is maximized.

* + 1. **Complexity Analysis**

By looking at the Algorithm 9, the data transfer time starts at 0 and increases q (the end time slot) by 1 and considers all possible k values to compute the amount of data transfer. Since it includes the MPA algorithm and that has the time complexity of O(mT) and the other entries of the table (when k>0 is O(T4), so the time complexity is O(mT+T4) in the worst case for optimal scheduling.

For the heuristic scheduling algorithm, MinVPVB-1, which uses the MinFPVB heuristic instead of the MPA for the OptVPVB-1, the time complexity is derived from that algorithm which gives a polynomial time complexity of O(T2\*m\*(T+lg n) + T4) in the worst case.

1. **Conclusion**

In this paper, we have investigated a series of six advanced reservation bandwidth scheduling problems all designed to enable the optimally scheduled high-performance transfer of files between nodes in a dedicated link network. In section II, we demonstrated feasible optimal and heuristic algorithms and their associated complexity for the fixed-path-fixed-bandwidth, fixed-path-variable-bandwidth, variable-path-fixed-bandwidth-0/1 and variable-path-variable-bandwidth-0/1 scheduling problem constraints. The summary of our findings is presented in table 2 below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem** | **Complexity** | **Algorithm** | **Big O Time** |
| **FPFB** | **P** | **OptFPFB** | **O(T2 .m.logn+T3.m)** |
| **FPVB** | **NP-Complete** | **OptFPVB**  **MinFPVB** | **O(mTlogT)**  **O(m.(T+log(n))** |
| **VPFB-0** | **P** | **OptVPFB-0** | **O(T.m.lg n + T3)** |
| **VPFB-1** | **P** | **OptVPFB-1** | **O(T2.m.logn)** |
| **VPVB-0** | **P** | **OptVPVB-0** | **T.m.log(n)** |
| **VPVB-1** | **NP-Complete** | **OptVPVB-1,MinVPVB-1** | **O(mT+T4), O(T2.m.(T+lg n)+T4** |

Table 2: Summary of algorithms and complexities to solve the various bandwidth scheduling problems addressed throughout this paper.

In conclusion, the transfer of significant amounts of data over dedicated high-speed networks using advanced bandwidth reservation techniques given varying scheduling constraints is feasible through the optimal and heuristic algorithms explored in this paper and present a realistic solution to the real world application of these networks to be used in various science, business and public sectors.

1. **References**
2. N. Rao, Q. Wu, S. Carter, W. Wing, D. G. A. Banerjee, and B. Mukherjee, “Control plane for advance bandwidth scheduling in ultra high-speed networks,” in Proc. INFOCOM Workshop Terabits Netw., 2006, pp. 1–5. UCLP, “UCLP: User Controlled LightPath Provisioning,” 2011 [On-line]. Available: http://www.uclp.ca
3. N. Rao, W. Wing, S. Carter, and Q. Wu, “Ultrascience net: Network testbed for large-scale science applications,” IEEE Commun. Mag. vol. 43, no. 11, pp. s12–s17, Nov. 2005 [Online]. Available: http://www.csm.ornl.gov/ultranet
4. CHEETAH, “CHEETAH: Circuit-switched High-speed End-to-End Transport ArcHitecture,” 2008 [Online]. Available: http://www.ece.virginia.edu/cheetah
5. Enlightened computing: An architecture for co-allocating network, compute, and other grid resources for high-end applications,” in Proc. IEEE HONET, Dubai, UAE, Nov. 2007, pp. 1–8.
6. Liudong Zuo, Khaleel, Zhu, & Wu. (2013). On Fixed-Path Variable-Bandwidth Scheduling in High-Performance Networks. Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, 23-30.
7. Lin, Y., & Wu, Q. (2013). “Complexity Analysis and Algorithm Design for Advance Bandwidth Scheduling in Dedicated Networks”. IEEE/ACM Transactions on Networking, 21(1), 14-27. doi:10.1109/tnet.2012.2189127
8. R. Guerin and A. Orda, “Networks with advance reservations: The routing perspective,” in Proc. 19th IEEE INFOCOM, 2000, vol. 1, pp.118–127.
9. C. Castillo, G. N. Rouskas and K. Harfoush, "On the Design of Online Scheduling Algorithms for Advance Reservations and QoS in Grids," *2007 IEEE International Parallel and Distributed Processing Symposium*, Rome, 2007, pp. 1-10. doi: 10.1109/IPDPS.2007.370226
10. Y. Lin and Q. Wu, "Path Computation with Variable Bandwidth for Bulk Data Transfer in High-Performance Networks," *IEEE INFOCOM Workshops 2009*, Rio de Janeiro, 2009, pp. 1-6. doi: 10.1109/INFCOMW.2009.5072176