Cross-Virtual Concatenation for Ethernet-over-SONET/SDH Networks

Satyajeet S. Ahuja and Marwan Krunz
{ahuja,krunz}@ece.arizona.edu*

Dept. of Electrical and Computer Engineering, The University of Arizona.

Abstract. Ethernet-over-SONET/SDH (EoS) with virtual concatenation is a popular approach for interconnecting geographically distant Ethernet segments using the SDH transport infrastructure. In this paper, we introduce a new concatenation technique, referred to as *cross-virtual concatenation* (CVC), which involves the concatenation of *virtual channels* (VCs) of hetrogenous capacities and can be implemented by a simple upgrade at SDH end nodes, thus utilizing the existing legacy SDH infrastructure. By employing CVC for EoS systems, we show that the SDH bandwidth can be harvested more efficiently than in conventional virtual concatenation. We later consider the routing problems associated with CVC connections, namely the connection establishment problem and the connection upgrade problem. We propose ILP and heuristic solutions to solve such problems. Simulations are conducted to evaluate the performance of the proposed heuristic and to demonstrate the advantages of employing CVC.

1 Introduction

Current optical transport infrastructure is dominated by the SDH technology [1]. SDH uses a bandwidth hierarchy indicated by STM-n, where $n = 1, 4, 16, 64, \ldots$. The basic unit in this hierarchy is the STM-1 channel (155.52 Mbps), which can support various smaller payloads, including VC-11 (1.5 Mbps), VC-12 (2 Mbps), and VC-3 (45 Mbps) channels. SDH was originally developed to support voice traffic. "Data" services are supported over SDH using Ethernet-over-SONET/SDH (EoS) with virtual concatenation [2].

In EoS with virtual concatenation, the aggregate bandwidth used for interconnecting two Ethernet segments is obtained by concatenating several SDH payloads (VC-n channels) of the same type, which can be independently routed to the destination. These channels, which we simply

* This work was supported by NSF under grants ANI-0095626, ANI-0313234, and ANI-0325979, and by the Center for Low Power Electronics (CLPE) at the University of Arizona. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not reflect the views of NSF.

refer to as *virtual channels* (VCs), form a *virtually concatenated group* (VCG). Data is byte-interleaved over the various VCs of the VCG.

Although the use of virtually concatenated EoS circuits has enabled efficient utilization of the SDH bandwidth, it may result in a large number of circuits between two end points. For example, consider two Ethernet LANs (with an average traffic of 100 Mbps between them) connected using a VCG of fifty VC-12 channels. Although such a connection greatly increases the bandwidth efficiency, the large number of circuits incur high maintenance overhead (the network management system has to maintain a large database to maintain these circuits). An alternative is to use three VC-3 channels (3×45 Mbps = 135 Mbps) at the expense of excessive bandwidth wastage. To achieve efficient bandwidth utilization using a relatively smaller number of connections, an EoS system needs to be able to combine VCs of different payloads in the same VCG. For example, the 100 Mbps traffic can be supported using two VC-3 channels (90 Mbps) plus five VC-12 channels (10 Mbps). We refer to such concatenation of SDH payload channels of different payload capacities as *cross-virtual concatenation* (CVC). In addition to reducing the maintenance overhead, CVC can also reduce the actual bandwidth usage. Due to hierarchical implementation of the SDH frame [1], only twenty one VC-12 (21×2Mbps = 42Mbps) channels or one VC-3 channel (45Mbps) can be transported over a TUG-3. This is because VCs typically incur some bandwidth overhead. For example, if we employ fifty VC-12 channels to support the EoS connection then it actually consuming 107 Mbps ($45 + 45 + (8/21)\times45$) worth of SDH bandwidth. For the CVC case (two VC-3 channels plus five VC-12 channels), the actual SDH bandwidth consumed is 101.9 Mbps ($2\times45 + (5/21)\times45$).

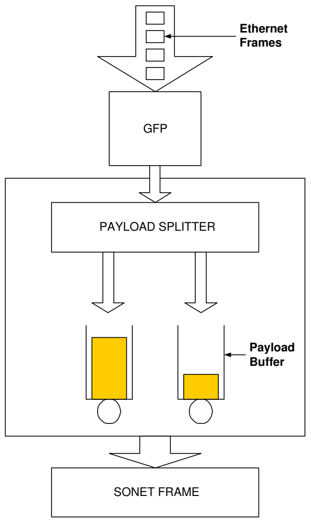
In this paper, we outline the general structure of the proposed CVC architecture and describe its advantages. We show a typical implementation of CVC using existing control overheads defined in SDH. The implementation requires a simple end-point upgrade. We study the path selection problems associated with CVC connection establishment. First, we consider the case of a new-connection establishment with a given bandwidth requirement. Then, we consider the case of bandwidth upgrade. We propose heuristic solutions for these problems. We study the performance of these heuristics and compare them with conventional VC case. Simulation results show that by employing CVC to establish EoS connections, we can harvest the SDH bandwidth efficiently.

2 Implementation of Cross-Virtual Concatenation

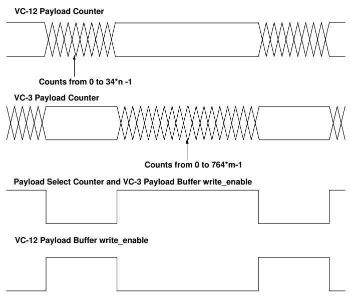
In this section, we describe how CVC can be implemented with a simple end-point upgrade. For illustration purposes, we describe a particular instance of CVC applied to the concatenation of VC-3 and VC-12 channels.

2.1 Transmit Side

In a typical transmit-side implementation of EoS, Ethernet frames are encapsulated using GFP (Generic Framing Procedure) [3].The resultant stream of bytes is then interleaved into various constituent members of the VCG. In the CVC implementation (see Figure 1(a)), we use a special payload splitter and a buffer assembly, just after packet encapsulation.



(a) Transmit side implementation of CVC.


VC-12 Payload Counter
Counts from 0 to 34*n -1
VC-3 Payload Counter
Counts from 0 to 764*m-1
Payload Select Counter and VC-3 Payload Buffer write_enable
VC-12 Payload Buffer write_enable

(b) Timing diagram for the transmit side implementation of CVC.

Fig. 1. Transmit side implementation and timing diagram

The splitter is essentially a set of payload counters associated with each payload type. Each payload counter maintains the number of bytes of a particular payload in the SDH frame. For example, a VC-12 payload counter counts from 0 to $34n - 1$ for n VC-12 channels in the VCG.

payload-select counter counts from 0 to $J-1$, where J is the number of different types of payloads participating in CVC; in this case, $J=2$. The various states of the payload-select counter are used to generate enable signals for various payload counters. At any instant, only one payload counter is enabled. There are J buffers (one for each payload type) to store the incoming payload bytes. When a byte is received after GFP encapsulation, it is stored into the payload buffer for which the payload buffer *write_enable* signal is high (see Figure 1(b)).

2.2 Receive Side

In a typical receive side implementation of EoS, the received SDH payload is stored in the differential delay compensator, which is essentially a payload buffer. Once the frames of all members of the VCG with the same multi-frame number (time-stamp) have arrived, the payload bytes are sequentially removed from the buffer based on the associated sequence number and are input to the GFP de-encapsulator. In a CVC implementation, we use different buffers for different payload types to compensate for the differential delay.

3 Connection Establishment Problem

Consider two inter-office Ethernet LANs that are to be connected using EoS with a given bandwidth requirement L. We define the *packing factor* α for the two payload types as the amount of bandwidth wastage incurred when using a smaller sized payload. For example, twenty one VC-12s consume the same SDH bandwidth as one VC-3, although the effective capacity of one VC-3 is 22.5 times that of a VC-12 channel. Hence, for these two payloads $\alpha = (22.5-21)/21$. In general, consider two payload types X and Y with bandwidth of B_x and B_y ($B_y > B_x$), respectively, and a packing factor α. Each link (i,j) is associated with two nonnegative integer capacity parameters C_{ij}^{x} and C_{ij}^{y} for the two payload types X and Y respectively. For a given source s and destination t, let $\mathbb{P}_x$ and $\mathbb{P}_y$ be k precomputed paths between s and t with capacity $f_x(p)$ ($\min_{(i,j)\in p} C_{ij}^{x} > 0$) and $f_y(p)(>0)$ w.r.t. payload types X and Y, respectively. We now formally define the connection establishment problem.

Problem 1. (Connection Establishment): For a given graph $G(V,E)$, find a set of paths from s to t such that the total capacity of these paths is greater than L and the maximum differential delay between any two of them is less than a given constraint J. If there are multiple solutions, then find the one with the minimum bandwidth wastage.

In [4] the authors considered the so called Differential delay routing (DDR) problem and showed that it is NP-complete. The problem at hand is more complicated than the standard DDR problem, and in the best case is equivalent to DDR problem (when the two payload types are of the same capacity). Hence, the connection establishment problem is also NP-complete. This problem can be addressed by splitting it into two parts. First, we check if the requested bandwidth requirement is feasible or not (using standard maximum-flow algorithm [5]). Then, we find the set of paths that satisfy the differential delay requirement. This step requires enumeration of paths, and is therefore NP-hard. Instead of computing these paths at the time of connection establishment, we can precompute a set of K paths, for each of the payloads and then focus on choosing a set of paths that satisfy the differential delay and bandwidth requirement. We first propose an ILP formulation to solve such a problem and later propose an efficient heuristic based on the sliding-window algorithm.

3.1 ILP formulation

Consider two sets of k paths $\mathbb{P}_x = \{P_1^x, \ldots, P_k^x\}$ and $\mathbb{P}_y = \{P_1^y, \ldots, P_k^y\}$ for the two payload types X and Y $(B_x < B_y)$, respectively. Let x_{jk} and y_{jk} be the integer flow on on links (j, k) w.r.t. payload types X and Y, respectively. Let X_i and Y_i be the flow along path P_i^x and P_i^y. Constraint 1, 2, and 3 are the flow conservation constraints on nodes (see Figure 2). Constraint 4 is the capacity constraints on edges. Constraint 5 relate flows on edges to the flows on paths, and constraint 6 force integrality constraint on flows of payload types X and Y, respectively.

3.2 Sliding Window Algorithm

The set of paths returned by the ILP will minimize the bandwidth wastage but worst-case complexity associated with this algorithm can be exponential. Hence, we now propose a computationally practical heuristic, which is a variant of the sliding-window algorithm [4]. The algorithm sequentially tries a set of precomputed paths to find a feasible solution. In case of multiple solutions, it attempts to return a solution with minimum bandwidth wastage. The heuristic uses K-shortest path algorithm [6] to find the set of precomputed paths. As shown in the pseudocode in Figure 3, the inputs to the algorithm are the graph $G(V, E)$, source s, destination t, bandwidth requirement L, and differential delay J. The algorithm precomputes two sets of k paths, $\mathbb{P}_x$ and $\mathbb{P}_y$, for payload types X and Y. Create $\mathbb{P} = \mathbb{P}_x \cup \mathbb{P}_y$, ordered according to their delay values. In the j^{th}

$$\text{Minimize} \sum_{\forall(j,k)\in E} x_{jk} + \sum_{\forall(j,k)\in E} y_{jk}$$

Subject to:

$$\sum_{k:(j,k)\in E} x_{jk} - \sum_{k:(k,j)\in E} x_{kj} = 0, \quad j \in V - \{s,t\} \tag{1}$$

$$\sum_{k:(j,k)\in E} y_{jk} - \sum_{k:(k,j)\in E} y_{kj} = 0, \quad j \in V - \{s,t\} \tag{2}$$

$$\left\{ \sum_{k:(j,k)\in E} (B_x x_{jk} + B_y y_{jk}) - \sum_{k:(k,j)\in E} (B_x x_{kj} + B_y y_{kj}) \right\} \begin{array}{l} \geq U,\ j = s \\ \leq -U,\ j = t \end{array} \tag{3}$$

$$0 \leq x_{jk} \leq C^x_{jk},\ 0 \leq y_{jk} \leq C^y_{jk}, \quad (j,k) \in E \tag{4}$$

$$x_{jk} = \sum_{i:(j,k)\in P_i} X_i, \quad y_{jk} = \sum_{i:(j,k)\in P_i} Y_i, \quad (j,k) \in E \tag{5}$$

$$X_i \in \{0,1,2,...,\lceil U/B_x \rceil\},\ Y_i \in \{0,1,2,...,\lfloor U/B_y \rfloor\}, i = 1,2,...,N \tag{6}$$

Fig. 2. ILP formulation for the connection establishment problem.

iteration, the algorithm considers all the paths $P_j, ..., P_r$, where P_r is the path with highest delay such that $d_r - d_j \leq J$, where d_i is delay of path P_i. The algorithm routes maximum flow along them, starting with the paths of positive payload capacities w.r.t. Y and then w.r.t. X. If the total flow is greater than L, the algorithm stores the solution with minimum bandwidth wastage. The wastage associated with any solution is the wastage incurred due to the use of X type payload.

4 Connection Upgrade Problem

We now consider the connection upgrade (CU) problem. Given an established EoS connection, we want to *upgrade* it with additional L bits/sec without affecting the traffic. LCAS [7] can be modified to incorporate CVC and additional bandwidth can be harvested multiple types of channels. In [8] the CU problem for the conventional virtually concatenated EoS system was studied by modelling it as a TSCP (two-sided constraint path) problem and was shown to be NP-complete in [9]. The problem was heuristically solved using MLW-KSP algorithm. CU problem is a generalization of TSCP and hence is also NP-Complete.

```
Sliding_Window($G(V,E), s, t, w(.), C_x, C_y, B_x, B_y, L, J$)
1.  Set $Flow_x = 0, Flow_y = 0$, $W = \infty$, $R = \emptyset$
2.  Precompute $k$ paths $\mathbb{P}_x = \{P_{x1}, P_{x2}, ..., P_{xk}\}$ and $\mathbb{P}_y = \{P_{y1}, P_{y2}, ..., P_{yk}\}$
    for payload type X and Y respectively with increasing order of delays
3.  $P = \mathbb{P}_x \cup \mathbb{P}_y$, sort $P$ based on increasing values of path delays
4.  For $i$ = 1, 2,...,2$k$,
5.    Consider paths $P_i, ..., P_r$, s.t. $P_r$ is highest delay path satisfying $d_r - d_i \leq J$
6.    For $j = i, i+1, ..., r$,
7.        Route maximum flow w.r.t. $Y$ along path $P_j$, $Flow_y = Flow_y + f_{yj}$
8.        Route maximum flow w.r.t. $X$ along path $P_j$, $Flow_x = Flow_x + f_{xj}$
9.    If $Flow_x * B_x + Flow_y * B_y > L$, /*A feasible solution found */
10.       $W_i$ = Wastage($Flow_x, Flow_y, B_x, B_y, L, \alpha$)
11.       If $W \geq W_i$,
12.           Store paths $R = \{P_i, P_{i+1}, ..., P_r\}$, Update $W = W_i$
13.   Remove all flow routed in the network, Reset $Flow_x = 0, Flow_y = 0$
14. Return $R$
Function: Wastage($x, y, B_x, B_y, L, \alpha$) /*For $B_y > B_x$*/
1. If $L \geq B_y y$,
2.    $L = L - B_y y$, Return $\lfloor \frac{L}{B_x} \rfloor \alpha$
3. else    /*$L < B_y y$*/
4.    $L = L - B_y \lfloor \frac{L}{B_y} \rfloor$
5.    If $L > B_x x$, Return $B_y - L$
6.    else Return $\lfloor \frac{L}{B_x} \rfloor \alpha$
```

Fig. 3. Pseudocode for the sliding-window algorithm.

Problem 2. Connection Upgrade: Given a graph $G(V, E)$, source s and destination t with an EoS connection of m members in the VCG, with path delays $D_1, \ldots, D_m$. Let J be the maximum allowable differential delay. Given a required upgrade bandwidth L, find set of paths S between s and t that satisfy the following: (1)Total bandwidth that can flow along these paths is greater than L, (2) Delay associated with $s_i \in S$ satisfies differential delay constraint with all paths in S and with all existing members of VCG. Specifically, $|D_{s_j} - D_{s_k}| \leq J$, $\forall s_j, s_k \in S$ and $\max_{1 \leq i \leq m} |D_i - D_{s_k}| \leq J$, $\forall s_k \in S$ and (3) wastage associated with S is minimum among all possible solutions.

We now discuss a heuristic approach for finding the set S. The inputs to the upgrade algorithm are the graph $G(V, E)$, $(s - t)$ pair, capacity constraint L, two positive constraints C_1 and C_2 representing possible maximum and minimum delay of paths in solution set S, and the packing factor α. The algorithm precomputes a set $\mathbb{P}$ using the MLW-KSP algorithm [8]. We then use a modified version of the sliding-window approach discussed in Section 3 to find the feasible and the most optimal solution in terms of bandwidth wastage. In the jth iteration, the algorithm considers the P_j^{th} path in $\mathbb{P}$, routes the maximum flow along P_j, and finds the new values of C_1 and C_2 after incorporating P_j in the VCG. The algorithm then finds the path with the maximum capacity among the set of paths in $\mathbb{P}$, which is also feasible with respect to the new values of C_1 and C_2. The algorithm continues to find the feasible path and route the flow until there are no feasible paths with positive flow. The pseudocode of the upgrade algorithm is presented in Figure 4.

```
**Upgrade_Algo**$(G(V,E), s, t, w(), C_x, C_y, C_1, C_2, \alpha)$
1. $Flow_x = 0, Flow_y = 0$, $W = \infty$, $R = \emptyset$ and $S = \emptyset$
2. Precompute $\mathbb{P} = \mathbb{P}_x \cup \mathbb{P}_x$ using MLW-KSP
3. For $i = 1, 2, ..., 2k$,
4.     $A_1 = C_1, A_2 = C_2$
5.     Augment $f_y(P_i)$ and $f_x(P_i)$ flow along path $P_i$
6.     $Flow_y = f_y(P_i), Flow_x = f_x(P_i)$, Update $A_1$ and $A_2, \mathbb{G} = \mathbb{P} - \{P_i\}, R = \{P_i\}$
7.     Calculate capacity of each path in $\mathbb{G}$, remove infeasible and zero capacity paths.
8.     While (G not empty),
9.         Find path $P_r$ with maximum capacity
10.        Augment maximum flow along $P_r$, $Flow_y = Flow_y + f_y(P_r)$, $Flow_x = Flow_x + f_x(P_r)$
11.        $R = R \cup \{P_i\}$, Update $A_1$ and $A_2$
12.        Re-Calculate capacity of each path in $\mathbb{G}$, remove infeasible and zero capacity paths.
13.    If $Flow_x * B_x + Flow_y * B_y > L$, /*A feasible solution found */
14.        $W_i$ = Wastage$(Flow_x, Flow_y, B_x, B_y, L, \alpha)$
15.        If $W \geq W_i$,
16.            $S = R$, $W = W_i$
17.    Remove all flow routed in the network
18.    $Flow_x = 0, Flow_y = 0$
19.Return $S$
```

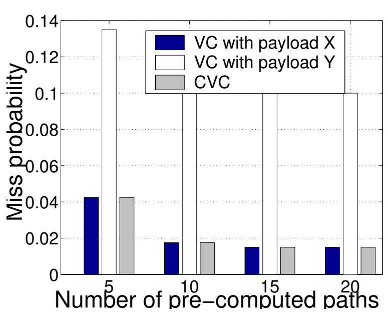
Fig. 4. Pseudocode for the connection upgrade algorithm.

5 Simulation Results

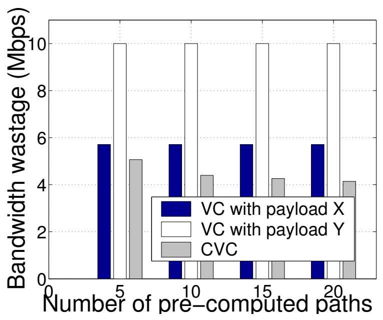
In this section, we use simulations to study the performance of the proposed algorithms. We report the gain achieved using CVC over the standard VC approach. Our simulations are based on random topologies that obey recently observed power laws [10] and are generated using the BRITE topology generator [11]. For a link (i, j), $f_x(i, j)$ and $f_y(i, j)$ are sampled from uniform distributions in the range $[0, 50]$ and $[1, 5]$, respectively. We randomize the selection of the *s*-*t* pair and fix the values of L and J for a simulation run. To model a meaningful EoS scenario, we choose the values of B_x and B_y such that they represent VC-12 and VC-3 SDH payload types. Specifically, we let $B_x = 2$ Mbps and $B_y = 45$ Mbps.

5.1 Results for the Connection Establishment Problem

We study the performance of the sliding-window algorithm proposed in Section 3 and compare it with standard VC. Our performance metrics are the bandwidth wastage and the probability of a miss. If the algorithm finds a set of paths that satisfy the required bandwidth, then we call it a *hit*; otherwise, it is a *miss*. For the standard VC case, we separately consider two types of payload and execute the sliding-window algorithm for each type. We assume all nodes are capable of using payload type Y by converting them into payload type X with a packing factor α. Figure 5(a) depicts the miss probability of various methods versus the number of precomputed paths used by the algorithm. The standard VC with payload type X and CVC perform significantly better than the standard VC with payload type Y because each node has a cross-connect of granularity equivalent to payload type X. Hence every solution to a connection establishment problem that uses only payload type Y is a solution to the connection upgrade problem with CVC and in most cases is a solution to the standard virtual concatenation with payload type X. In Figure 5(b),



(a) Miss probability vs. number of precomputed paths.



(b) Bandwidth wastage vs. number of precomputed paths.

Fig. 5. Comparison of CVC with sliding-window algorithm and standard VC with the two types of payloads (L = 80 Mbps, J = 70, and a network of 100 nodes).

we study the performance in terms of the average bandwidth wastage for various values of precomputed paths in the sliding-window algorithm. In the three considered methods, we count all the successful attempts to find the set of paths and average the bandwidth wastage associated with the solution. The performance of CVC with sliding-window algorithm improves when we consider more precomputed paths because of the larger solution space. Compared to standard VC techniques, the performance of CVC in terms of bandwidth wastage is significantly better and it further improves with the number of precomputed paths. The performance of the standard VC technique does not improve with the number of precomputed paths because for a given bandwidth and a standard VC technique, the amount of bandwidth wastage is fixed. To demonstrate the effectiveness of CVC, we plot the bandwidth wastage incurred by employing three types of concatenation techniques as a function of the traffic demand (L). As shown in Figure 6, the bandwidth wastage of the standard virtual concatenation with payload type Y follows a saw-tooth pattern, whereas it increases linearly in the standard VC with payload type X. CVC outperforms other techniques because it efficiently uses both payloads to achieve a lower bandwidth wastage. Recall that CVC has a significant advantage over standard VC with payload type X in terms of connection management (i.e., it requires fewer chennels).

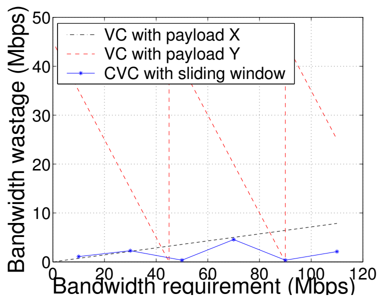


Fig. 6. Comparison between CVC with the sliding-window algorithm and the standard VC with the two types of payloads ($k = 15$, $J = 70$, and a network of 100 nodes.)

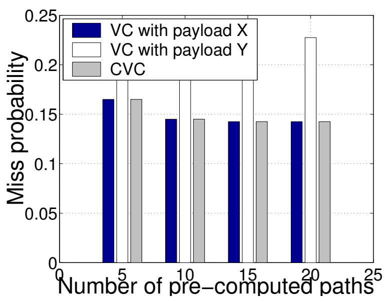
5.2 Results for the Connection Upgrade Problem

For the connection upgrade problem, we study the performance of Upgrade algorithm proposed in Section 4. For a given simulation run, the values of C_1 and C_2 fall into the following cases: First, The shortest path between s and t w.r.t $w(.,.)$ is greater than C_2. In this case, there is no feasible solution. Second, The shortest path between s and t w.r.t $w(.,.)$ is less than or equal to C_1. The second case is nontrivial, and is the one considered in our simulations. Specifically, we let $C_1 = W(p*) + A + U(0, 50)$

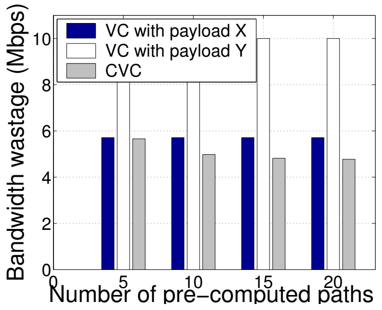
($U(x, y)$ is a uniform random variable with range (x, y)) and $C_2 = C_1 + J$, where $p*$ is the shortest path between s and t w.r.t. $w(.,.)$ and A is a positive constant. Figure 7(a) compares the performance of upgrade algorithm with the standard virtual concatenation in terms of miss probability for different values of precomputed paths used in the upgrade algorithm ($L = 80$ Mbps and $J = 70$). The performance of CVC is better than the performance of standard virtual concatenation with payload type Y and is same as the performance of payload type X with standard virtual concatenation. This is because the cross-connect chosen at each node assumes a granularity of B_x, and hence a solution to the upgrade problem using standard virtual concatenation of payload type Y is also a solution of CVC and in most cases is the solution of standard virtual concatenation with payload type X. The standard virtual concatenation with payload type X and CVC performs significantly better than the payload type Y. The performance of CVC and standard concatenation with payload type X further improves by increasing the number of precomputed paths. In Figure 7(b), we study the performance of upgrade algorithm with the standard virtual concatenation in terms of average bandwidth wastage by varying the values of precomputed paths in the upgrade algorithm. The performance of upgrade algorithm improves when we increase the number of precomputed paths. Compared to the standard concatenation techniques the performance of CVC in terms of bandwidth wastage is significantly better and improves slightly when we use more number of precomputed paths in the upgrade algorithm. This is because with a larger solution space, the upgrade algorithm is able to efficiently allocate the resources within the two payload types such that the bandwidth wastage is minimized.

6 Conclusions

In this paper, we introduced the general structure of CVC (cross-virtual concatenation) for EoS circuits. We proposed a simple implementation of CVC that involves a simple upgrade at the end nodes and that reuses the existing SDH overhead. The algorithmic problems associated with connection establishment and connection upgrade were studied for CVC. We have proposed efficient heuristics to solve these problems using a sliding window approach. Extensive simulations were conducted to show the effectiveness of employing CVC for EoS systems.



(a) Miss probability vs. number of precomputed paths.



(b) Bandwidth wastage vs. number of precomputed paths.

Fig. 7. Comparison of CVC with Upgrade algorithm and the standard VC with the two types of payloads, $L = 80$ Mbps, $J = 70$, and a network of 100 nodes.

References

1. ITU-T Standard G.707: Network node interface for the synchronous digital hierarchy (2000)
2. Ramamurti, V., Siwko, J., Young, G., Pepe, M.: Initial implementations of point-to-point Ethernet over SONET/SDH transport. IEEE Communications Magazine **42** (2004) 64–70
3. ITU-T Standard: G.7041 Generic Framing Procedure. (2003)
4. Srivastava, A., Acharya, S., Alicherry, M., Gupta, B., Risbood, P.: Differential delay aware routing for Ethernet over SONET/SDH. Proceedings of the IEEE INFOCOM Conference (2005, Miami)
5. Ahuja, R., Magnanti, T., Orlin, J.: Network flows: Theory, Algorithm, and Applications. Prentice Hall Inc. (1993)
6. Chong, E., Maddila, S., Morley, S.: On finding single-source single-destination shortest paths. Proceedings of the Seventh International Conference on Computing and Information (ICCI '95) (1995) 40–47
7. ITU-T Standard G.7042: Link capacity adjustment scheme for virtually concatenated signals. (2001)
8. Ahuja, S., Korkmaz, T., Krunz, M.: Minimizing the differential delay for virtually concatenated Ethernet over SONET systems. Proceedings of the IEEE 13th International Conference on Computer Communications and Networks, ICCCN **5** (2004) 205–210
9. Ahuja, S., Krunz, M., Korkmaz, T.: Optimal path selection for Ethernet over SONET under inaccurate link-state information. Proceedings of the Second International Conference on Broadband Networks (2005)
10. Faloutsos, M., Faloutsos, P., Faloutsos, C.: Power-laws of the Internet topology. Proceedings of the ACM SIGCOMM Conference (1999) 251–262
11. BRITE: Boston university representative Internet topology generator. (http://www.cs.bu.edu/brite/)