Network Topology Aware Scheduling of Collective Data Exchanges

Emin Gabrielyan, Roger D. Hersch
École Polytechnique Fédérale de Lausanne, Switzerland
{Emin.Gabrielyan,RD.Hersch}@epfl.ch

Abstract

We propose a method for the optimal scheduling of collective data exchanges relying on the knowledge of the underlying network topology. We introduce the concept of liquid schedules. Liquid schedules ensure the maximal utilization of a network's bottleneck links and offer an aggregate throughput as high as the flow capacity of a liquid in a network of pipes. The collective communication throughput offered by liquid schedules in highly loaded networks may be several times higher than the throughput of traditional topology-unaware brute force scheduling techniques such as round-robin or random schedules. To create a liquid schedule we need to find the smallest partition of a the whole set of transfers consisting of subsets of mutually non-congesting transfers. For interconnection network topologies (switches, communication links) having equilibrated throughput capabilities, the number of combinations of non-overlapping subsets of mutually non-congesting transfers grows exponentially with the number of transfers. We propose several methods to reduce the search space without affecting the solution space. On a real 32 node computer cluster, the measured liquid throughputs of data exchanges scheduled according to our method are very close to the theoretical liquid throughputs.

Keywords: Optimal network utilization, collective data exchange, liquid schedules, network topology, topology-aware scheduling.

1. Introduction

The interconnection topology is one of the key factors of a computing cluster. It determines the communication performance of the communications, which is are often a limiting factor of parallel applications [1], [2], [3], [4].

Interconnection topology is one of the key elements determining the global communication throughput. In high-speed networks such as those used in cluster computing [Boden95], [Duato01] and in optical communication networks [Stern99], the network is often formed by a set of full crossbar switches (called optical switches in optical networks). TDMA and CMDA wireless networks [Battiti99] also can be viewed as interconnection topologies whose links represent the orthogonal frequency spectres. Full crossbar switches allow to dynamically route packets from their input ports to their output ports. The crossbar switches we consider are cut-through switches with full $k \times k$ connectivity allowing for simultaneous transfers from any input to any output. between any pair of the k input and output links.

In the present contribution, we deal with the problem of collective communications through networks made of cut-through switches. We assume that end nodes do not perform store and forward operations. We also assume that the collective communication pattern is known in advance.

The aggregate throughput of a collective data exchange depends on the underlying network topology and on the number of receiving and emitting nodes (end nodes). The total amount of data together with the longest transfer time across the most loaded links (bottlenecks) gives an estimation of the aggregate throughput. We define this estimation as the liquid throughput of the network. It corresponds to the flow capacity of a non-compressible fluid in a network of pipes [6]. In a real wormhole and cut-through networks during the transmission of a message the links lying on the path from the source node to the destination nodes are kept occupied. Therefore due to transfers of data packets messages, congestions may occur and the aggregate throughput of a collective data exchange may be lower than the liquid throughput. The rate of congestions of a given data exchange may considerably vary according to the chosen sequence of transfers from the order according to which the transfers are actually carried out.

This paper presents an algorithm able to compute a liquid schedule whenever a liquid schedule exists. We test the proposed scheduling algorithm on a K-ring switching network and verify by measurements that it achieves the expected liquid throughput. We limit ourselves to fixed packet sizes and neglect network latencies. Switches are assumed to be full cross-bar, also with negligible latencies.

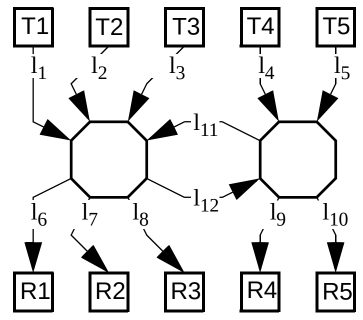
The problem we present is similar to the problem of wavelength routing in all-optical networks. In that context, it was shown that the computation of an optimal routing scheme can be formulated as a graph-coloring problem and that for general types of networks, the problem is NP-hard [Beauquier97]. However, greedy algorithms exist which provide sub-optimal solutions in polynomial time. There has been work on theoretical considerations about the required number of wavelengths and the complexity of finding a solution according to the type of network [Bermond96], [Caragiannis02]. Specific classes of topologies were also analysed in the context of satellite-switch time division multiplexing networks [8].

Unlike flow control based congestion avoidance mechanisms [9] [10], we establish schedules for the data transfers without trying to regulate the sending processors' data rate. We specifically address the problem of reaching the flow capacity of a fluid in a network by trying to optimally schedule accordingly the set of transfers of a collective data exchange.

There are numerous applications requiring highly efficient network resources: parallel acquisition and/ or distribution of multiple video streams each one forwarded being further distributed to a set of target nodes [11], [12]; switching of simultaneous voice communication sessions [13], [14]; high energy physics data acquisition and all-to-all transmission from a large number of detectors to a cluster of processing nodes for filtering and event assembling [16], etc.

The solution we propose may also be helpful to schedule collective communications in switched optical networks with wavelength conversion [Bermond96], [Caragiannis02]. It may allow within single hop all-optical networks, to compute for a given network and routing scheme, the minimal number of wavelengths to be assigned to bottleneck links.

2. The scheduling problem



$$\left\{\begin{array}{c}
\{l_1, l_6\}, \{l_1, l_7\}, \{l_1, l_8\}, \{l_1, \mathbf{l_{12}}, l_9\}, \{l_1, \mathbf{l_{12}}, l_{10}\}, \\
\{l_2, l_6\}, \{l_2, l_7\}, \{l_2, l_8\}, \{l_2, \mathbf{l_{12}}, l_9\}, \{l_2, \mathbf{l_{12}}, l_{10}\}, \\
\{l_3, l_6\}, \{l_3, l_7\}, \{l_3, l_8\}, \{l_3, \mathbf{l_{12}}, l_9\}, \{l_3, \mathbf{l_{12}}, l_{10}\}, \\
\{l_4, \mathbf{l_{11}}, l_6\}, \{l_4, \mathbf{l_{11}}, l_7\}, \{l_4, \mathbf{l_{11}}, l_8\}, \{l_4, l_9\}, \{l_4, l_{10}\}, \\
\{l_5, \mathbf{l_{11}}, l_6\}, \{l_5, \mathbf{l_{11}}, l_7\}, \{l_5, \mathbf{l_{11}}, l_8\}, \{l_5, l_9\}, \{l_5, l_{10}\}
\end{array}\right\}$$

Fig. 1. Example of a data exchange composed of 25 transfers

For example, consider the all-to-all collective data exchange shown in Fig. 1. There are 5 transmitting processors (T1,... T5), each of them sending a packet to each of the receiving processors (R1... R5). The network consists of 12 links. Links l_{11} and l_{12} are the most loaded links, since each of them will be used by 6 transfers. The most loaded links are the bottlenecks of the collective data exchange. They have the longest active time. In the best case, the duration of a collective data exchange is as long as the active time of the bottleneck links.

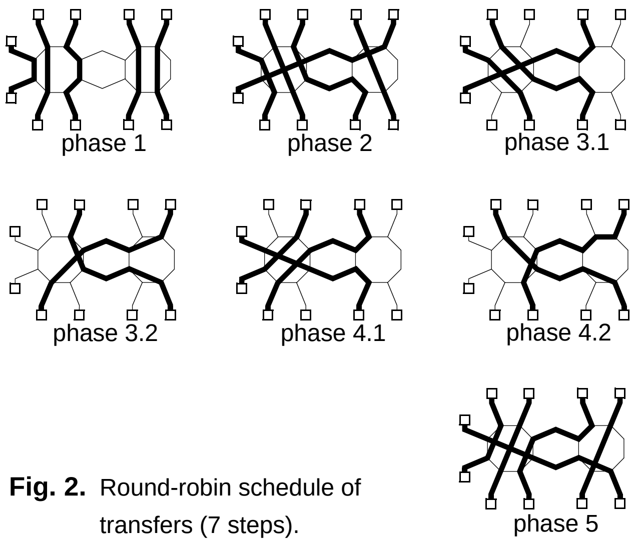


Fig. 2. Round-robin schedule of transfers (7 steps).

A *round-robin* schedule is carried out in 5 phases: *(1)* {T1→R1, T2→R2 ... T5→R5}, *(2)* {T1→R2, T2→R3 ... T5→R1}, etc. The round-robin schedule's throughput is however lower than the liquid throughput, since bottleneck links l_{11} and l_{12} are idle in phase 1 (Fig. 2). Phases 3 and 4 are carried out in two steps time-frames, since they contain congesting transfers.

Fig. 3 shows that a schedule achieving the liquid throughput for the considered collective data exchange exists.

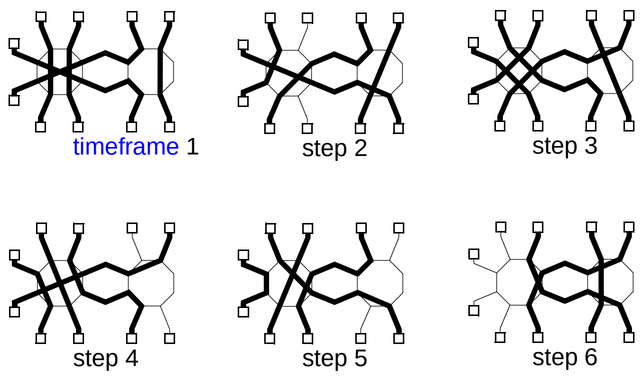


Fig. 3. An optimal schedule (6 steps).

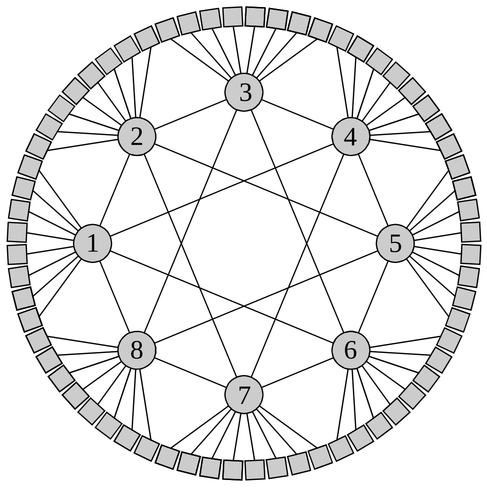
Section 2 shows how to describe introduces a testbed consisting of 363 sub-topologies and represents the liquid throughput as a function of the number of contributing processing nodes and their underlying network topologies. The construction of liquid schedules is presented in section 3. In section 4, we present measurements for the considered sub-topologies and in section 5 we draw the conclusions.

2. Throughput as a function of sub-topology

Let us compute introduce a few test topologies for evaluation of our results. for which liquid schedules will be computed.

In order to plot the throughput of collective data exchanges as a function of the network topology, we specify along an independent axis the number of contributing processing nodes as well as significant variations of their underlying network topologies. For the sake of simplicity, each node incorporates a transmitting and a receiving processor. The applications perform all-to-all data exchanges over the allocated nodes (each transmitting processor sends one packet to each receiving processor).

Let us create variations of processing node allocations on the Swiss-T1 cluster (called henceforth T1, see Fig. 4). The network of the T1 forms a K-ring [17] and has a static routing scheme. The throughputs of all links are identical, equal to 86*MB/s*. The cluster consists of 64 processors paired into 32 nodes [18].



Routing Table

	1	2	3	4	5	6	7	8
1	×	✓	**2**	✓	**4**	✓	**8**	✓
2	✓	×	✓	**7**	✓	**3**	✓	**5**
3	**2**	✓	×	✓	**4**	✓	**8**	✓
4	✓	**7**	✓	×	✓	**7**	✓	**3**
5	**4**	✓	**4**	✓	×	✓	**6**	✓
6	✓	**3**	✓	**7**	✓	×	✓	**1**
7	**8**	✓	**8**	✓	**6**	✓	×	✓
8	✓	**5**	✓	**3**	✓	**1**	✓	×

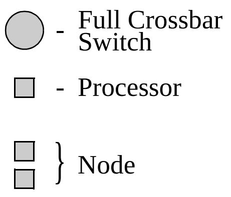


Fig. 4. Architecture of the T1 cluster computer.

Since there may be between 0 and 4 allocated nodes in front of each of 8 switches, we have $5^8 = 390625$ possible processing node allocations. With the given network topology and routing tables, we can compute for each combination of node allocation the liquid throughput of the all-to-all traffic.

Because of various symmetries within the network, many of these node allocations yield an identical liquid throughput. We extracted a set of 363 different node allocations representing each possible liquid throughput only once. Each node allocation being considered as an underlying network's sub-topology is characterized by its liquid throughput and the number of allocated nodes (see Fig. 5). Depending on the sub-topology, the liquid throughput for a given number of nodes may considerably vary.

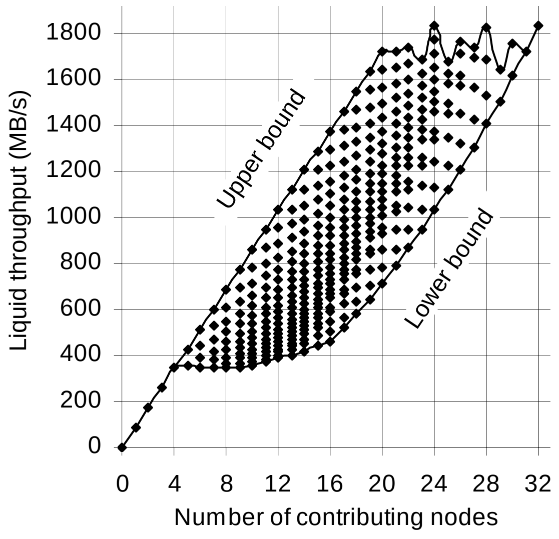


Fig 5. Liquid throughput in function of the number of nodes with variations according to sub-topologies.

These 363 sub-topologies are placed on one axis, sorted first by the number of nodes and then according to their liquid throughput. Fig. 6 shows the predicted liquid throughput values together with the measured throughput of a round-robin schedule.

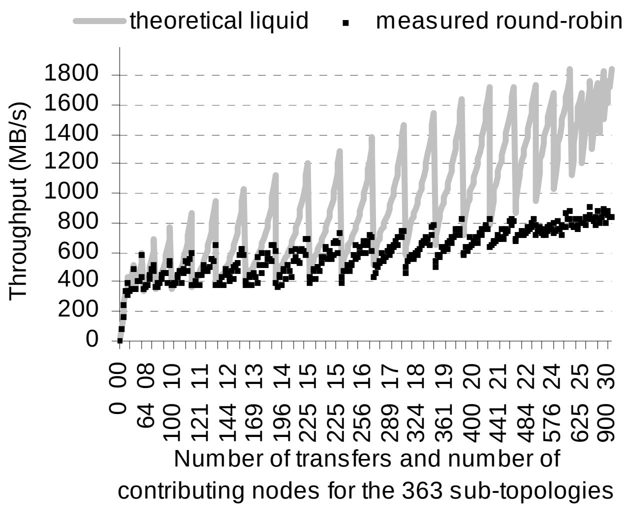


Fig. 6. Theoretical liquid throughput and measured round-robin schedule throughput for 363 network sub-topologies.

For many sub-topologies, the theoretical liquid throughput is twice as large as the round-robin throughput.

3. Liquid schedules

This section presents a method for building liquid schedules on any topology. As in many computer cluster networks, we assume a static routing scheme. The presented method is valid for any combination of transmitting and receiving processors performing any type of collective exchange (not limited to all-to-all exchanges). We neglect network latencies and assume a constant packet size for all data exchanges. A sending processor may transfer a packet to a given receiving processor not more than once.

Let us introduce a formal model of a collective data exchange.

DEFINITIONS. A *transfer* is a set of links (i.e. the links forming the path from a sending processor to a receiving processor). A *traffic* is a set of transfers (i.e. the transfers forming the collective exchange, see Fig. 1). A link l is *utilized* by a transfer x if $l \in x$. A link l is utilized by a traffic X if l is utilized by a transfer of X. Two transfers of a traffic X congest if they use a common link. A sub-traffic of X (a subset of X) is *simultaneous* if it forms a collection of non-congesting transfers.

A simultaneous subset of a traffic is processed in the time frame of a single transfer. The *load* of a link l in the traffic X is the number of transfers in X using l. The maximally loaded links are called *bottlenecks*. The *duration* $\Lambda(X)$ of a traffic X is the load of its bottlenecks. The size of the traffic $\#(X)$ is the number of its transfers. The *liquid throughput* of a traffic X is the ratio $\#(X)/\Lambda(X)$ multiplied by the single link throughput.

For example, the traffic X shown in Fig. 1 has a number of transfers $\#(X) = 25$ and the duration of the traffic is $\Lambda(X) = 6$. Therefore the aggregate liquid throughput is the ratio $25/6$ of a single link throughput, i.e. $(25/6) \times 100MB/s = 416.67MB/s$, assuming a single link throughput of $100MB/s$.

In the parlance of the graph theory the traffic is a graph, whose vertices are the transfers of the traffic and two vertices are joined if the two transfers are congesting. The duration of a traffic is a lower bound of a chromatic number of the graph. Transfers utilizing a common link form a clique in the graph.

3.1. Partitioning

A *partition* of X is a disjoint collection of non-empty subsets of X whose union is X [19]. A *schedule* α of a traffic X is a collection of simultaneous sub-traffics of X partitioning the traffic X. A *step timeframe* of a schedule α is an element of the schedule α. The *length* $\#(\alpha)$ of a schedule gives the number of *step timeframe* in α. A schedule of a traffic is *optimal* if the traffic does not have any shorter schedule (and the length of an optimal schedule is therefore the chromatic number of the corresponding graph). If the length of a schedule is equal to the duration of the traffic, then the schedule is *liquid*. A liquid schedule is optimal, but the inverse is not always true, meaning that a traffic may not have a liquid schedule. Fig. 7 shows a liquid schedule for the collective traffic shown in Fig 1.

The duration of a traffic X is the load of its bottlenecks. If a schedule is liquid, then each of its steps must use all bottlenecks. Inversely, if all steps of a schedule use all bottlenecks, the schedule is liquid.

$$\left\{\begin{array}{ccc}
\underset{\text{step 1}}{\begin{Bmatrix}\{l_1, \mathbf{l_{12}}, l_9\}, \\ \{l_2, l_7\}, \\ \{l_3, l_8\}, \\ \{l_4, \mathbf{l_{11}}, l_6\}, \\ \{l_5, l_{10}\}\end{Bmatrix}} &
\underset{\text{step 2}}{\begin{Bmatrix}\{l_1, \mathbf{l_{12}}, l_{10}\}, \\ \{l_2, l_6\}, \\ \{l_4, \mathbf{l_{11}}, l_7\}, \\ \{l_5, l_9\}\end{Bmatrix}} &
\underset{\text{step 3}}{\begin{Bmatrix}\{l_1, \mathbf{l_8}\}, \\ \{l_2, \mathbf{l_{12}}, l_9\}, \\ \{\mathbf{l_3}, l_6\}, \\ \{l_4, l_{10}\}, \\ \{l_5, \mathbf{l_{11}}, l_7\}\end{Bmatrix}} \\
\underset{\text{step 4}}{\begin{Bmatrix}\{l_1, l_7\}, \\ \{l_2, \mathbf{l_8}\}, \\ \{\mathbf{l_3}, \mathbf{l_{12}}, l_9\}, \\ \{l_5, \mathbf{l_{11}}, l_6\}\end{Bmatrix}} &
\underset{\text{step 5}}{\begin{Bmatrix}\{l_1, l_6\}, \\ \{l_2, \mathbf{l_{12}}, \mathbf{l_{10}}\}, \\ \{\mathbf{l_3}, l_7\}, \\ \{\mathbf{l_4}, \mathbf{l_{11}}, \mathbf{l_8}\}\end{Bmatrix}} &
\underset{\text{step 6}}{\begin{Bmatrix}\{\mathbf{l_3}, \mathbf{l_{12}}, \mathbf{l_{10}}\}, \\ \{\mathbf{l_4}, \mathbf{l_9}\}, \\ \{\mathbf{l_5}, \mathbf{l_{11}}, \mathbf{l_8}\}\end{Bmatrix}}
\end{array}\right\}$$

Fig. 7. A liquid schedule for the collective traffic shown in Fig. 1. Bold links in a step indicate bottlenecks in the reduced traffic, i.e. the original traffic minus the transfers of the preceding steps.

The necessary and sufficient condition for the liquidity of a schedule is that all bottlenecks be used by each step of the schedule. Since a simultaneous sub-traffic of X is defined as a *team* of X, if it uses all bottlenecks of X, an equivalent condition for the liquidity of a schedule α on X is that each step of α be a team of X.

Our strategy for finding a liquid schedule will therefore rely on searching for simultaneous sub-traffics using all bottlenecks, i.e. teams of a traffic. Hence, we need to partition a traffic by collections of teams (whenever possible).

Let us show that by removing an element (*step timeframe*) from a liquid schedule, we form a new liquid schedule on the remaining traffic. Note that the remaining traffic may have additional bottlenecks. For example, in Fig. 7, from step 3 on, links $\boldsymbol{l_3}$ and $\boldsymbol{l_8}$ appear as additional bottlenecks. Emerging additional bottlenecks allow us to reduce the search space when creating a liquid schedule.

THEOREM 1. Let α be a liquid schedule on X and A be a *step timeframe* of α. Then $\alpha - \{A\}$ is a liquid schedule on $X - A$.

PROOF. Clearly A is a team of X. Remove the team A from X so as to form a new traffic $X - A$. The duration of the new traffic $X - A$ is the load of the bottlenecks in $X - A$. Bottlenecks of $X - A$ include the bottlenecks of X. The load of a bottleneck of X is decreased by one in the new traffic $X - A$ and therefore the duration of $X - A$ is the duration of X decreased by one, i.e. $\Lambda(X - A) = \Lambda(X) - 1$. The schedule α without the element A is a schedule for $X - A$ with the previous length decreased by one. Therefore the new schedule $\alpha - \{A\}$ has as many steps as the duration of the new traffic $X - A$. Hence $\alpha - \{A\}$ is a liquid schedule on $X - A$.

In other words, if the traffic has a liquid schedule, then a schedule reduced by one team is a liquid schedule on the reduced traffic. The repeated application of Theorem 1 implies that any non-empty subset of a liquid schedule is a liquid schedule on the correspondingly reduced traffic.

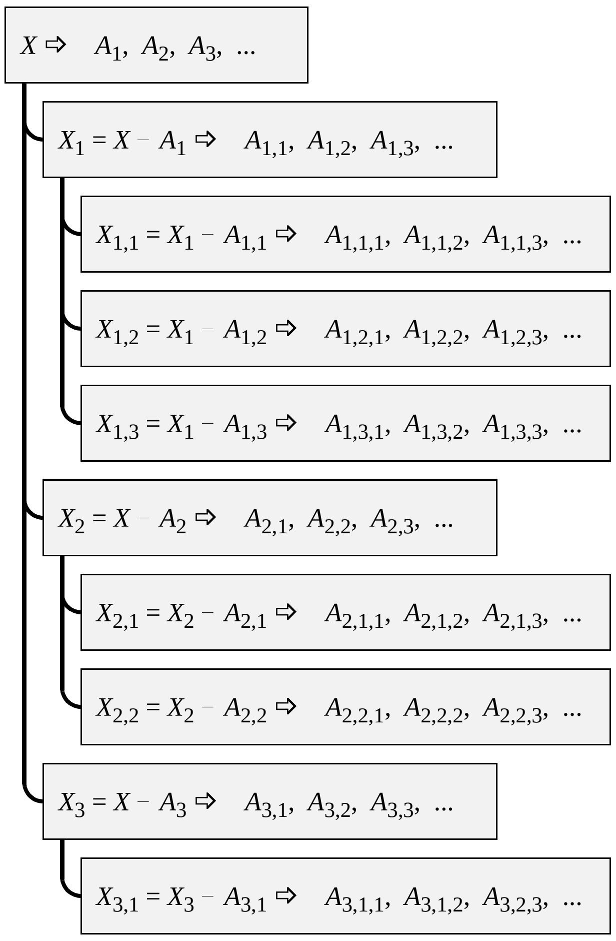
3.2. Construction

THEOREM 2. If, by traversing each team A of a traffic X none of the sub-traffics $X - A$ has a liquid schedule, then the traffic X does not have a liquid schedule either.

PROOF. Let us suppose that X has a liquid schedule α. Then a step A of α shall be a team of X. Further, according to Theorem 1 the schedule $\alpha - \{A\}$ shall be a liquid schedule for $X - A$. Therefore for at least one team A of X the sub-traffic $X - A$ has a liquid schedule. This proves the theorem by contraposition.

Theorem 2 implies that if X has a liquid schedule at least one team A of X will be found, such that the sub-traffic $X - A$ has a liquid schedule β. Obviously $\beta \cup \{A\}$ will be a liquid schedule for X.

Let us give an overall view to the liquid schedule search algorithm. The algorithm recursively searches for a solution by traversing a tree in depth-wise order (Fig. 8). The root of the tree is the original traffic *X*. Associated to the traffic *X* is the collection of all possible steps of a liquid schedule $\{A_1, A_2, \ldots, A_n\}$. Successor nodes are formed by subtraffics $X - A_1$, $X - A_2, \ldots$ $X - A_n$. Each of these successor nodes has its own collection of all possible steps. As before, each member of this collection will produce successor nodes at the next level of the tree.



$X ⇨ A_1, A_2, A_3, \ldots$
$X_1 = X - A_1 ⇨ A_{1,1}, A_{1,2}, A_{1,3}, \ldots$
$X_{1,1} = X_1 - A_{1,1} ⇨ A_{1,1,1}, A_{1,1,2}, A_{1,1,3}, \ldots$
$X_{1,2} = X_1 - A_{1,2} ⇨ A_{1,2,1}, A_{1,2,2}, A_{1,2,3}, \ldots$
$X_{1,3} = X_1 - A_{1,3} ⇨ A_{1,3,1}, A_{1,3,2}, A_{1,3,3}, \ldots$
$X_2 = X - A_2 ⇨ A_{2,1}, A_{2,2}, A_{2,3}, \ldots$
$X_{2,1} = X_2 - A_{2,1} ⇨ A_{2,1,1}, A_{2,1,2}, A_{2,1,3}, \ldots$
$X_{2,2} = X_2 - A_{2,2} ⇨ A_{2,2,1}, A_{2,2,2}, A_{2,2,3}, \ldots$
$X_3 = X - A_3 ⇨ A_{3,1}, A_{3,2}, A_{3,3}, \ldots$
$X_{3,1} = X_3 - A_{3,1} ⇨ A_{3,1,1}, A_{3,1,2}, A_{3,2,3}, \ldots$

Fig. 8. Liquid schedule search tree. The symbol "⇨" points to all possible steps for the current reduced traffic.

Let us discuss how to build the collection of all possible steps for the current node. For being liquid, it is sufficient that all the steps of a schedule be teams of the original traffic *X*. A possible step at each sub-traffic is any team of *X* formed by not yet carried out transfers, i.e. each team *A* of the original traffic *X* included in the current sub-traffic $X_{reduced}$, i.e. $\{A \in \Im(X) | A \subset X_{reduced}\}$, the operator $\Im$ associating with a traffic the set of all its teams.

We would like to reduce the search space. Instead of forming the set of possible steps by using teams of the original traffic $\{A \in \Im(X) | A \subset X_{reduced}\}$, we propose to form the set of all possible steps at the current node using all teams of the current sub-traffic, i.e. $\Im(X_{reduced})$. It can be shown that the number of teams of the current subtraffic $\Im(X_{reduced})$ is smaller or equal to the number of teams of the original traffic whose transfers belong to the current subtraffic, i.e.
$\#(\Im(X_{reduced})) \leq \#(\{A \in \Im(X) | A \subset X_{reduced}\})$.
Therefore less possible teams need to be considered when building the schedule and the solution space is not affected, since theorem 2 is valid at any level of the search tree.

By traversing the tree in depth-wise order, we cover the full solution space. A solution is found when the current node (sub-traffic) forms a single team. The path from the root to that leaf node forms the set of teams yielding the liquid schedule. A node presents a dead end if it is not possible to create a team out of that sub-traffic. In that case we have to backtrack to evaluate other choices. Evaluation of all choices ultimately leads to a solution if it exists.

If a solution exists for *X*, then the algorithm will find it. If the algorithm does not find a solution for *X*, and since we explored the full solution space, we conclude that *X* does not have a liquid schedule.

Let us describe a further simple and efficient search space reducing technique.

DEFINITIONS. A simultaneous subset *A* of a traffic *X* is *full* with respect to *X* if each transfer of $X - A$ is in congestion with a transfer of *A*. A team of *X* is called *full* team if it is a full simultaneous subset of *X*.

Let us modify a liquid schedule so as to convert one of its teams into a full team. Let a traffic *X* have a liquid schedule α. Let *A* be a step of α. If *A* is not a full team of *X*, then, by moving the necessary transfers from other steps of α, we can convert step *A* to a full team. Evidently, the properties of liquidity (partitioning, simultaneousness and length) of α will not be affected. Therefore if *X* has a solution then it has also a solution when one of its steps is full, hence the choice of the teams in the construction may be narrowed from the set of all teams to the set of full teams only. Fig. 7 shows a liquid schedule constructed with full teams.

In order to be able to explore the full solution space for obtaining a liquid schedule, we need to successively build all full teams. We designed a procedure capable of generating (without repetitions) all full teams of an arbitrary traffic. It first builds *skeletons*, an intermediate collection of teams from a sub-traffic including only those transfers which comprise bottlenecks. Then it expands each skeleton by applying variations of all non-congesting transfers in order to build up all full teams.

As briefly described in the annex, the problem of creating a liquid schedule seems can also be expressed as a graph coloring problem.

The annex introduces a further relationship between the problem of creating a liquid schedule and the graph coloring problem.

4. Results

Let us compare the predicted theoretical values of the liquid throughputs with the measurements of the actual data exchanges carried out according to the liquid schedules we have found.

Fig. 9 shows the measured aggregate throughput of an all-to-all collective data exchange executed on a T1 computer cluster, optimized by applying our liquid schedule based traffic partitioning technique. Each black dot represents the median of 7 measurements. The horizontal axis represents the 363 sub-topologies as well as the number of contributing nodes. Processor to processor transfers have a size of 5*MB*, transferred as a single message of 5*MB*. The measured all-to-all aggregate throughputs (black dots) are close to the theoretically computed liquid throughput (gray line). For many sub-topologies, the proposed scheduling technique allows to increase the aggregate throughput by a factor of two compared with a simple round-robin schedule (Fig. 6).

Thanks to the presented search space reduction algorithms, the computation time of a liquid schedule takes for more than 97% of the considered sub-topologies of the T1 cluster less than 1/10 of a second on a single Compaq 500MHz Alpha processor.

5. Conclusion

We propose a method for scheduling collective data exchanges in order to obtain an aggregate throughput equal to the network's liquid throughput. This is achieved by building at each step of the schedule mutually non-congesting sets of transfers using all bottleneck links. Exploration of the full solution space yields a liquid schedule if it exists. The proposed search space reduction techniques make the approach practical for networks having in the order of ten interconnected crossbar switches.

On the Swiss T1 cluster computer, the proposed scheduling technique allows for many sub-topologies to increase the collective data exchange throughput by a factor of two.

An alternative to the search for the liquid schedule is to color the conflict graph associated to the set of collective transfers by applying a greedy algorithm. Slightly sub-optimal scheduling solutions are obtained at a much higher speed, which is of high interest for concrete scheduling applications.

In the future, we intend to explore how to extend the presented scheduling techniques in order to dynamically

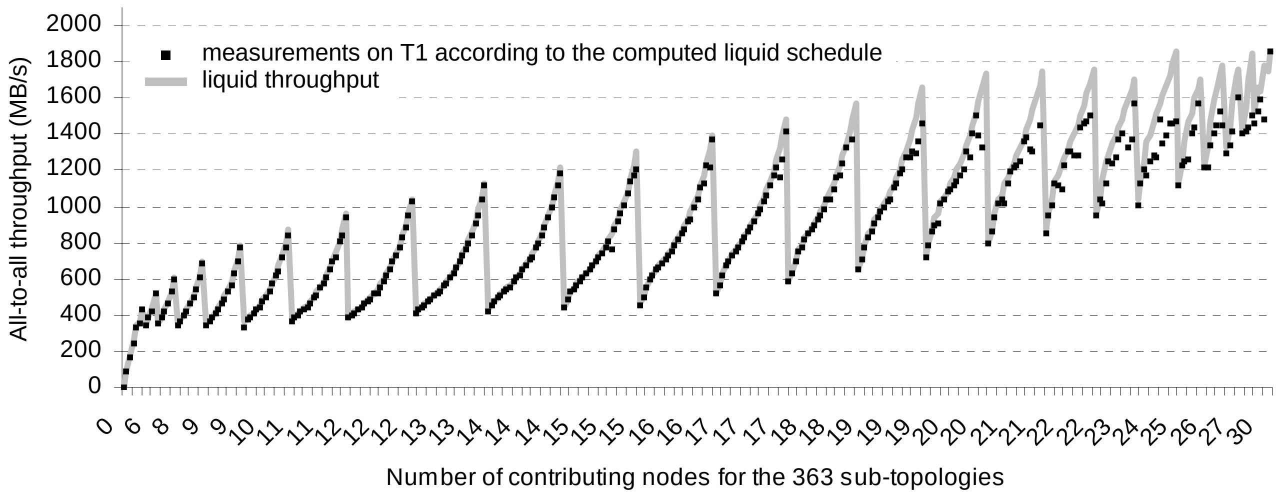


Fig. 9. Predicted liquid throughput and measured throughput according to the computed liquid schedule.

reschedule collective data exchanges when the set of planned exchanges evolves over time.

Annex

The search for a liquid schedule requires to partition the traffic into a set of non-overlapping sets consisting of mutually non-congesting transfers. The problem can also be formulated as the problem of coloring the conflict graph [Beauquier97]. Vertices of the conflict graph are formed by transfers. Edges between vertices represent congestions between transfers.

Fig. 10 shows the graph whose vertices are to be colored for the collective data exchange of Fig. 1. Vertex $x_{n,\,m}$ corresponds to a transfer from an emitting processor *n* to a receiving processor *m*. For example vertex $x_{4,\,1}$ represents the transfer T4→R1={l_4, $\mathbf{l_{11}}$, l_6}. The bold edges of the graph show congestions of transfers due to specific bottleneck links.

Whenever a liquid schedule exists, an optimal solution of the graph coloring problem is a liquid schedule. The chromatic number of the graph's optimal coloring is the length of the liquid schedule. Vertices having the same color represent a timeframe of the liquid schedule.

The graph to be colored is characterised by the relatively low density of its edges. We can label each edge of the graph by the link(s) causing the congestion. An all-to-all data exchange on the Swiss T1 cluster with 32 transmitting and 32 receiving processors forms a graph with $32 \times 32 = 1024$ vertices and 48704 edges. The approach we propose allows to compute in advance the chromatic number of the graph's optimal coloring (length of the liquid schedule) if a liquid schedule exists. Furthermore, we further reduce the problem by first trying to "color" vertices having edges representing all bottleneck links (creation of teams). Then we work on the reduced graph, formed by the original graph minus the colored vertices (forming teams on sub-traffics).

We compared our method of finding a liquid schedule with the results obtained by applying a greedy high-speed graph coloring *Dsatur* algorithm [Brelaz79] which carries out the following operations:

1. Arrange the vertices by decreasing order of degrees.
2. Color a vertex of maximal degree with color 1.

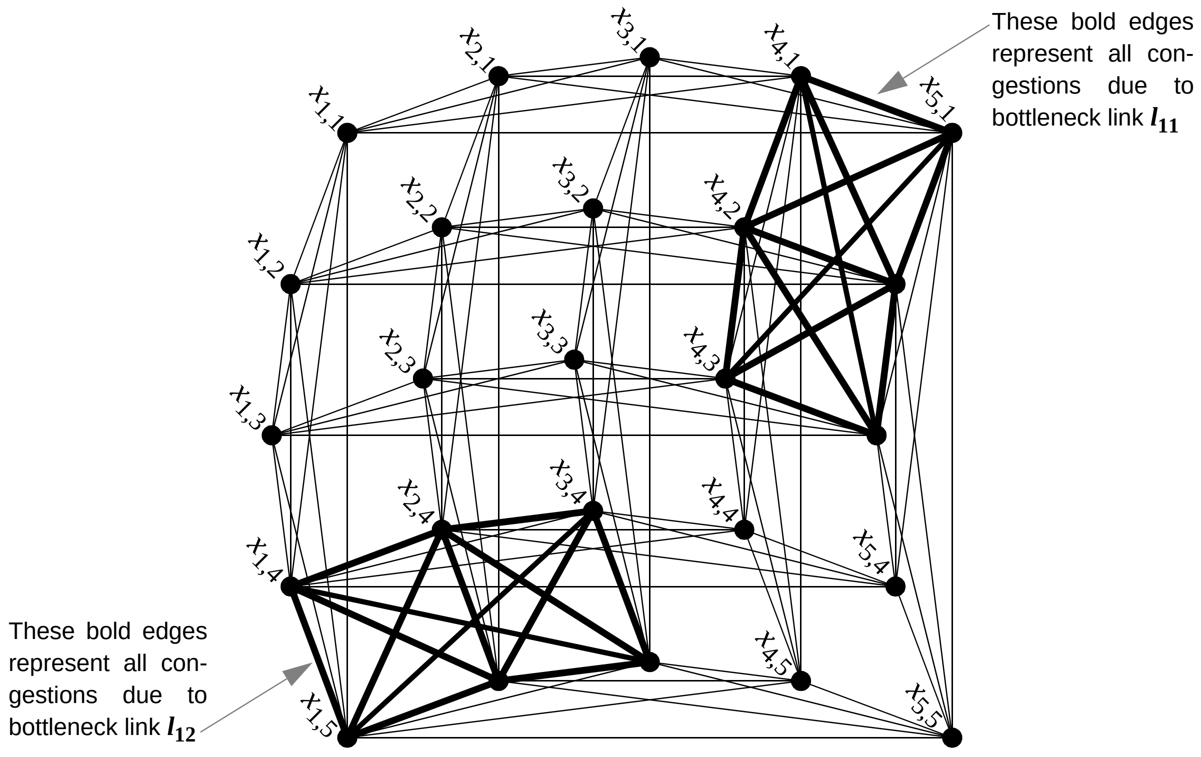


Fig. 10. Graph corresponding to the data exchange shown in Fig. 1. The 25 vertices of the graph represent the transfers. The edges represent congestion relations between transfers, i.e. each edge represents one or more communication links shared by two transfers.

3. Choose a vertex with a maximal saturation degree (defined as the number of different colours to which it is adjacent). If there is an equality, choose any vertex of maximal degree in the uncoloured subgraph.
4. Color the chosen vertex with the least possible (lowest numbered) color.
5. If all the vertices are colored, stop. Otherwise, return to 3.

Fig. 11 shows the loss of performance on the T1 sub-topologies due to additional unnecessary colours induced by the greedy graph coloring algorithm, compared with the liquid schedule algorithm.

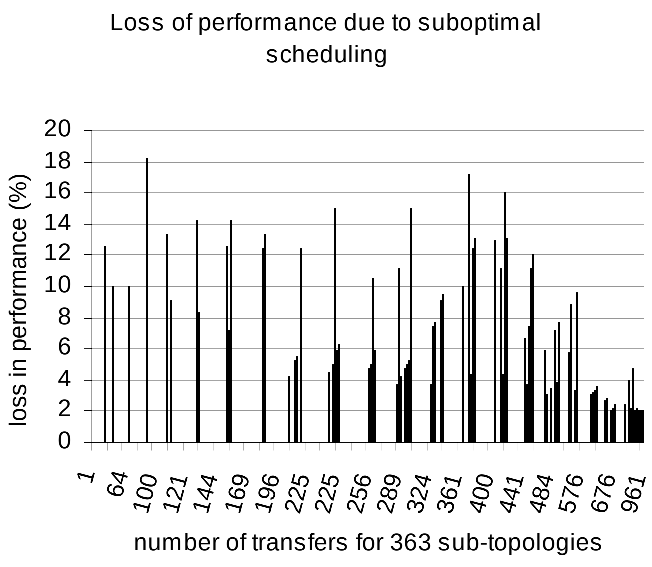


Fig. 11. Comparison of the throughputs of liquid schedules with the schedules obtained by the Dsatur heuristic algorithm.

For 74% of the topologies there is no loss of performance. For 18% of the topologies, the performance loss is below 10% and for 8% of the topologies, the loss of performance is between 10% and 20%.

The computation time of the greedy algorithm is polynomial and compares therefore quite favourably with the algorithm searching for the liquid schedule.

References

[1] H. Sayoud, K. Takahashi, B. Vaillant, "Designing communication network topologies using steady-state genetic algorithms", IEEE Communications Letters, Vol. 5, No. 3, March 2001, 113-115.

[2] Pangfeng Liu, Jan-Jan Wu, Yi-Fang Lin, Shih-Hsien Yeh, "A simple incremental network topology for wormhole switch-based networks", Proc. 15th International Parallel and Distributed Processing Symposium, 2001, 6-12.

[3] P.K.K. Loh, Wen Jing Hsu, Cai Wentong, N. Sriskanthan, "How network topology affects dynamic loading balancing", IEEE Parallel & Distributed Technology: Systems & Applications, Vol. 4, No. 3, 25-35.

[4] V. Puente, C. Izu, J. A. Gregorio, R. Beivide, J. M. Prellezo, F. Vallejo, "Improving parallel system performance by changing the arrangement of the network links", Proc. of the International Conference on Supercomputing, May 2000, 44-53.

[5] M. Naghshineh, R. Guerin, "Fixed versus variable packet sizes in fast packet-switched networks", Proc. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM '93., Networking: Foundation for the Future, IEEE Press, Vol. 1, 1993, 217-226.

[6] Benjamin Melamed, Khosrow Sohraby, Yorai Wardi, "Measurement-Based Hybrid Fluid-Flow Models for Fast Multi-Scale Simulation", DARPA/NMS BAA 00-18 AGREEMENT No. F30602-00-2-0556, *http://www.darpa.mil/ito/research/nms/meetings/nms2001apr/Rutgers-SD.pdf*

[7] J.-C. Bermond, L. Gargano, S. Perennes, A. A. Rescigno, and U. Vaccaro, "Efficient collective communication in optical networks", Proc. of ICALP'96. Lecture Notes in Computer Science, 574-585, 1996.

[8] R. Jain, G. Sasaki, "Scheduling packet transfers in a class of TDM hierarchical switching systems", IEEE International Conference on Communications ICC '91, Vol. 3, 1991, 1559-1563.

[9] Dah-Ming Chiu, Raj Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", Computer Networks and ISDN Systems, 1989, Vol. 17, 1-14.

[10] H. Ozbay, S. Kalyanaraman, A. Iftar, "On rate-based congestion control in high-speed networks: Design of an H-infinity based flow controller for single bottleneck", Proc. of the American Control Conference, June 1998, 2376-2380.

[11] S.-H.G. Chan, "Operation and cost optimization of a distributed server architecture for on-demand video services", IEEE Communications Letters, Vol. 5, No. 9, Sept. 2001, 384-386.

[12] Dinkar Sitaram, Asit Dan, *Multimedia Servers*, Morgan Kaufmann Publishers, San Francisco California, ISBN 1-55860-430-8, 2000, 69-73.

[13] H.323 Standards, *http://www.openh323.org/standards.html*

[14] D.A. Fritz, D.W. Moy, R.A. Nichols, "Modeling and simulation of Advanced EHF efficiency enhancements", Proc. of Military Communications Conference, IEEE MILCOM 1999, Vol. 1, 354-358.

[15] ATLAS Collaboration, CERN, Technical Progress Report, *http://press.web.cern.ch/Atlas/GROUPS/DAQTRIG/TPR/PDF_FILES/TPR.bk.pdf*

[16] Large Hadron Collider, Computer Grid project, CERN, 20.09.2001, *http://press.web.cern.ch/Press/Releases01/PR10.01EGoaheadGrid.html*

[17] P. Kuonen, “The K-Ring: a versatile model for the design of MIMD computer topology”, Proc. of the High-Performance Computing Conference (HPC'99), San Diego, USA, April 1999, 381-385.

[18] Pierre Kuonen, Ralf Gruber, “Parallel computer architectures for commodity computing and the Swiss-T1 machine”, EPFL Supercomputing Review, Nov 99, pp. 3-11, *http://sawww.epfl.ch/SIC/SA/publications/SCR99/scr11-page3.html*

[19] Paul R. Halmos, *Naive Set Theory*, Springer-Verlag New York Inc, ISBN 0-387-90092-6, 1974, 26-29.

[20] G. Campers and O. Henkes and J. P. Leclerq “Graph Coloring Heuristics: A Survey, Some New Propositions and Computational Experiences on Random and ‘{L}eighton's’ Graphs” Proc. Operational Research, 917-932, 1988.

[21] A. Hertz and D. de Werra “Using Tabu Search Techniques for Graph Coloring” in Computing(39) 345-351, 1987.

[Beauquier97] B. Beauquier, J.C. Bermond, L. Gargano, P. Hell, S. Pérennes, U. Vaccaro, “Graph Problems Arising from Wavelength-Routing in All-Optical Networks”, 2rid IEEE Workshop on Optics and Computer Science (WOCS, part of IPPS '97). IEEE Press, April 1997.

[Bermond96] J.-C. Bermond, L. Gargano, S. Perennes, A. A. Rescigno, and U. Vaccaro, “Efficient collective communication in optical networks”, *Proc. of ICALP'96. Lecture Notes in Computer Science*, 1996, 574-585

[Caragiannis02] I. Caragiannis and Ch. Kaklamanis and P. Persiano, “Wavelength Routing in All-Optical Tree Networks: {A} Survey”, *Bulletin of the European Association for Theoretical Computer Science*, 2002, Vol. 76, 104-112

[Boden95] N.J. Boden, et al., “Myrinet - A gigabit per second local area network,” *IEEE Micro*, pp. 29-36, February 1995.

[Duato01] J. Duato, A. Robles, F. Silla, R. Beivide, “A Comparison of Router Architectures for Virtual Cut-Through and Wormhole Switching in a NOW Environment”, *ACM Journal of Parallel and Distributed Computing*, February 2001, Vol. 61, Issue 2, 224-253

[Stern99] Thomas E. Stern, Krishna Bala, Hardcover, “Multi-wavelength Optical Networks: A Layered Approach”, Addison-Wesley, ISBN: 020130967X, May 1999

[Battiti99] Roberto Battiti, Alan A. Bertossi, Maurizzio A. Bonuccelli, “Assigning Codes in Wireless Networks: Bounds and Scaling Properties.”, ACM/Baltzer Wireless Networks, Vol. 5, 1999, 195-209

[Brelaz79] Daniel Brelaz, “New Methods to Color the Vertices of a Graph”, *CACM(22)*, 1979, 251-256.