

Broadcasting on Large Scale Heterogeneous Platforms with connectivity artifacts under the Bounded Multi-Port Model

Olivier Beaumont, Nicolas Bonichon, Lionel Eyraud-Dubois, Przemysław Uznański
INRIA Bordeaux – Sud-Ouest
University of Bordeaux,
Bordeaux, France
Email: obeaumont@labri.fr, bonichon@labri.fr, eyraud@labri.fr, uznanski@labri.fr

Abstract—We consider the classical problem of broadcasting a large message at an optimal rate in a large scale distributed network. The main novelty of our approach is that we consider that the set of participating nodes can be split into two parts: "green" nodes that stay in the open-Internet and "red" nodes that lie behind firewalls or NATs. Two red nodes cannot communicate directly, but rather need to use a green node as a gateway for transmitting a message. In this context, we are interested in both maximizing the throughput (*i.e.* the rate at which nodes receive the message) and minimizing the degree at the participating nodes, *i.e.* the number of TCP connections they must handle simultaneously. We both consider cyclic and acyclic solutions for the flow graph. In the cyclic case, our main contributions are a closed form formula for the optimal cyclic throughput and the proof that the optimal solution may require arbitrarily large degrees. In the acyclic case, we prove that it is possible to achieve the optimal throughput with low degree. Then, we prove a worst case ratio between the optimal acyclic and cyclic throughput and show through simulations that this ratio is on average very close to 1, which makes acyclic solutions efficient both in terms of the throughput and the number of connections.

Keywords-Broadcast; Scheduling; Resource Augmentation; firewall; Approximation Algorithms; Communication modeling

I. INTRODUCTION

Disseminating large data files in parallel and distributed platforms has been the subject of a vast literature. In this paper, we are interested in broadcasting a large file to a set of nodes in the context of large scale, Internet level platforms. In the case where the Internet is the underlying network, we cannot assume that the exact topology of the network is known and can be used to design an efficient algorithm. Therefore, designing a broadcasting algorithm on a large scale distributed platform over the Internet consists in both finding an overlay network (*i.e.* deciding who will communicate with who) and finding a schedule for communication (when and how much will be sent on the edges of the overlay network). In the context of content distribution systems, finding a good overlay is at the core of live streaming distribution systems such as CoolStreaming [1], PPLive [2] or SplitStream [3]. In the context of efficient data dissemination, many algorithms devoted to

specific parallel architectures have been designed [4], [5], [6]) and more recently randomized dissemination algorithms achieving optimal throughput have been proposed [7]. This paper is a follow-up of [8], where the problem of building an overlay and designing a scheduling algorithm to achieve high throughput have been considered. The main contribution of this work is to consider a more realistic communication model. Indeed, in [8], it is assumed that each node has the possibility to communicate with any other node, whereas we consider here the case of nodes located behind firewalls or NATs that therefore require the use of other nodes susceptible of acting as communication relays to use their outgoing bandwidth, as proposed in TURN [9]. As we will see it, considering nodes behind NATs and firewalls deeply changes the results we can achieve and the algorithms.

Even if the topology of a large scale platform cannot be determined dynamically, several embedding tools have been proposed, whose goal is to map a set of nodes into a metric space [10], [11] (*i.e.* to give them coordinates) so that their distance in the metric space estimates well the metric of interest (usually the latency between two nodes or the bandwidth a communication between them can achieve). In the case of latencies, the most well known embedding tools are Vivaldi [12], which embeds nodes into a 2D+1 metric space and relies on direct measurements to adapt dynamically node coordinates, and Sequoia [13], which embeds the nodes as the leaves of a weighted tree and relies on the distance in the tree to estimate the latency. Both coordinate systems have been extended to estimate bandwidths in [13], but it has been recently proved experimentally [14] on the PlanetLab dataset that the basic LastMile or bounded multiport model, where each node is associated to a incoming and an outgoing bandwidth limit, and where the achievable bandwidth between C_i and C_j is the minimum of the outgoing bandwidth of C_i and the incoming bandwidth of C_j , is more accurate than those more sophisticated models with respect to bandwidth prediction.

The bounded multiport model has already been advocated by Hong et al. [15] for independent tasks distribution on heterogeneous platforms. In this model, node C_i can communicate with any number of nodes C_j simultaneously,

each using a bandwidth $c(C_i, C_j)$ provided that its outgoing bandwidth is not exceeded, *i.e.*, $\sum_j c(C_i, C_j) \leq b_i^{\text{out}}$. Similarly, node C_i can receive messages from any number of nodes C_j simultaneously, each using a bandwidth $c(C_j, C_i)$, provided that its incoming bandwidth is not exceeded, *i.e.*, $\sum_j c(C_j, C_i) \leq b_i^{\text{in}}$. This corresponds well to modern network infrastructure, where each communication is associated to a TCP connection.

This model strongly differs from the traditional one-port model used in scheduling literature, where connections are made in exclusive mode: each node can communicate with a single node at any time step. Nevertheless, in the context of large scale platforms, the networking heterogeneity ratio may be high, and it is unacceptable to assume that a 100MB/s server may be kept busy for 10 seconds while communicating a 1MB data file to a 100kB/s DSL node. Therefore, in our context, we will assume that all communications are directly handled at TCP level. Nevertheless, in order to keep some flavor of the one-port model, we will bound the number of connections that can be handled simultaneously at a given node. This constraint is particularly important in the context where QoS mechanisms are used to fix or bound the bandwidth associated to each communication (each TCP connection in practice). It is worth noting that at the operating system level, several QoS mechanisms enable a prescribed sharing of bandwidth [16], [17]. In particular, it is possible to handle simultaneously several connections and to fix the bandwidth allocated to each connection. In our context, it has been proved in [18] that these mechanisms are necessary since the bandwidth allocated to the connection between C_j and C_i may be lower than both b_j^{out} and b_i^{in} . In order to model the limit on the number of simultaneous communications, we introduce another parameter d_j in the bounded multi-port model ($d = 1$ corresponds to the one port model, whereas $d = +\infty$ corresponds to the Lastmile model), that represents the maximal number of connections that can simultaneously be opened and handled with QoS mechanisms at node C_j . Therefore, the model we propose encompasses the benefits of both the bounded multi-port model and the one-port model. It enables several communications to take place simultaneously, which is compulsory in the context of large scale distributed platforms, and practical implementation is achieved by using TCP QoS mechanisms and by bounding the number of connections.

Nevertheless, the bounded degree multi-port models fails to correctly model the behavior of the nodes located behind a NAT or a firewall. This issue is crucial in the context of Peer-to-Peer applications running over the Internet. For instance, in distributed applications such as Skype [19], [20] or Bittorrent [21], NATs play a crucial role, since in certain situations where "hole punching" techniques [22] fail, it can be impossible for a pair of nodes to communicate directly. In this case, the technique consists of using a third party node

that will act as a relay for the packets. At a higher level, we can classify the nodes between green and red nodes, where green-green, green-red (or red-green) connections are possible, but not red-red. As we will see, adding this constraint on node connectivity capabilities strongly modifies the algorithms and the theoretical results.

The rest of the paper is organized as follows. In Section II, we present the communication model we use with the two sets of nodes, formalize the scheduling problem we consider, and prove that the problem of finding the optimal acyclic solution is NP-complete. Then, we provide in Section III a cyclic solution that achieves maximal throughput without the limit on the number of connections a node can handle simultaneously, and we prove that in some cases, a high degree may be required to achieve optimal throughput. In Section IV, we propose a greedy algorithm that achieves optimal throughput (among acyclic solutions) at the price of a small increase in the degree of the nodes (based on resource augmentation techniques). In Section V we propose a comparison (for worst-case instances and random instances) between the optimal throughput that can be achieved using cyclic and acyclic solutions. These results indicate that the cost of considering only acyclic solutions is bounded in the worst case, and is negligible in the average case. Finally, we provide in Section VI some future works and concluding remarks.

II. PROBLEM MODELING

Let us consider a set of nodes $(C_0, C_1 \dots)$ and let us partition this set into two sets: on the one hand the nodes that belong to the *open-internet*, *i.e.* nodes that can communicate with each other freely, and on the other hand nodes that can communicate only with nodes of the open-internet, *i.e.* nodes that are behind a firewall or behind a NAT router. In the following, we assume that the set of nodes in the open-internet is of size $n+1$ and the set of nodes behind a firewall or a NAT router is of size m . Let us denote by \mathcal{N} the index set of the open-internet nodes, and by \mathcal{M} the index set of nodes behind a firewall/NAT router. In the rest of the paper, a node in the open-internet is said to be *green* and a node behind a firewall/NAT router is said to be *red*.

W.l.o.g., let us assume that $\mathcal{N} = \llbracket 1, n \rrbracket$ and $\mathcal{M} = \llbracket n+1, n+m \rrbracket$. For any $i \in \llbracket 0, n+m \rrbracket$, the i -th node is denoted by C_i . The node C_0 is called the *source*.

Let us denote by b_j the outgoing bandwidth of node C_j and by d_j the maximal number of outgoing connections that it can handle simultaneously (its degree). Moreover, let $N = \sum_{i=1}^n b_i$ and $M = \sum_{i=n+1}^m b_i$ denote respectively the overall outgoing bandwidth of green and red nodes. Moreover, let us assume that both node sets are ordered in non-increasing order of bandwidths, except the source that is always the first node: $\forall (i, j) \in \mathcal{N}^2 : i < j \Rightarrow b_i \geq b_j$ and $\forall i < j \in \mathcal{M}^2 : i < j \Rightarrow b_i \geq b_j$.

Our aim is to broadcast a given message at rate (or throughput) T to all the nodes of the platform. Clearly, all nodes need to receive the message exactly once, so that all incoming bandwidths should be at least T . Therefore, in what follows, we assume that the incoming bandwidths of all nodes are larger than T and we concentrate only on outgoing communication capacities.

A broadcast scheme is given by a matrix $\{c_{i,j}(i,j) \in \llbracket 0, n+m \rrbracket^2\}$ such that:

- $\forall i \in \llbracket 0, n+m \rrbracket, \sum_j c_{i,j} \leq b_i$ (bandwidth constraint)
- $\forall i \in \llbracket 0, n+m \rrbracket, |j, c_{i,j} > 0| \leq d_i$ (degree constraint)
- $\forall (i,j) \in \mathcal{M}^2, c_{i,j} = 0$ (firewall constraint)

The *throughput* of a broadcast scheme is $T = \min_{i \in \llbracket 1, n+m \rrbracket} \{maxflow(C_0 \rightarrow C_i)\}$. For a given platform, we denote by T^* the optimal throughput. The problem of computing T^* is NP-complete, even for the special case where all nodes are green ($m = 0$) [8]. We are thus interested in this paper in *approximate* solutions, both in terms of throughput and resource augmentation on the degrees.

It is interesting to note that in a solution of throughput T , the bandwidth that node i can actually use is bounded by $b'_i = \min(b_i, Td_i)$. The throughput $T^*(I')$ achievable on the instance I' with bandwidth b'_i , and no degree constraint is an upper bound on the optimal throughput T^* . A solution that achieves (a fraction of) $T^*(I')$ by using an outdegree $o_i \leq \left\lceil \frac{b'_i}{T_{opt}(I')} \right\rceil + d$ is thus a d -resource augmentation (approximation) algorithm. The results presented in this paper are in the context of this transformation. As a consequence, we do not consider strict degree constraints, but rather analyze the outdegrees used by our solutions in terms of $\left\lceil \frac{b'_i}{T} \right\rceil$.

As a special case, we analyze more precisely *acyclic* solutions. A broadcast scheme is said to be *acyclic* if there exists an order σ on the nodes such that

$$\forall i, j \in \llbracket 0, n+m \rrbracket, i > j \Rightarrow c_{\sigma(i), \sigma(j)} = 0.$$

This condition states that $\sigma(i)$, the node at position i in the ordering σ cannot feed $\sigma(j)$, the node at position j , if $i > j$. For a given instance and a given order σ , we denote by $T_{ac}^*(\sigma)$ the optimal acyclic scheme compatible with the order σ . For a given instance, we denote by T_{ac}^* the optimal acyclic throughput:

$$T_{ac}^* = \min_{\sigma} T_{ac}^*(\sigma)$$

III. CYCLIC ANALYSIS

In this section, we give a closed formula for the optimal throughput for a broadcast scheme. We then show that in an optimal broadcast scheme, a node of bandwidth T^* may have to have an arbitrary large degree, what strongly differs from the case where all the nodes are green [8]).

A. Closed formula for the optimal throughput

Theorem 3.1:

$$T^* = \min\left(b_0, \frac{b_0 + N}{n}, \frac{b_0 + N + M}{n+m}\right).$$

Proof: Let us first prove that $T^* \leq \min\left(b_0, \frac{b_0 + N}{n}, \frac{b_0 + N + M}{n+m}\right)$. Clearly $T^* \leq b_0$, since the whole message has been sent at least once by the source. Then, the m red nodes have to receive the message at rate T^* and therefore consume mT^* bandwidth. Since this bandwidth must come from the source and the green nodes, then $mT^* \leq b_0 + N$. Finally, all $n+m$ nodes must receive the whole message at rate T^* and the bandwidth must come from the source, the green and the red nodes, so that $(m+n)T^* \leq b_0 + N + M$.

Let us now prove that $T^* \geq \min\left(b_0, \frac{b_0 + N}{n}, \frac{b_0 + N + M}{n+m}\right)$, by building a solution of throughput $T = \min\left(b_0, \frac{b_0 + N}{n}, \frac{b_0 + N + M}{n+m}\right)$ as a weighted sum of four types of trees (see Figure 1),

- $T_1(i, j)$ for $i \in \mathcal{N}$ and $j \in \mathcal{M}$, where the source feeds red node j , red node j feeds all green nodes, and green node i feeds all other red nodes;
- $T_2(i)$, for $i \in \mathcal{N}$, where the source feeds green node i , and green node i feeds all other nodes (green and red);
- $T_3(j)$, for $j \in \mathcal{M}$, where the source feeds all red nodes, and red node j feeds all green nodes; and
- T_4 where the source feeds all green and red nodes.

Provided that none of the outgoing bandwidths of the source, the green nodes and the red nodes are exhausted, then it is always possible to add to the solution a tree $T_1(i, j)$ of weight $\alpha_1(i, j)$ such that the bandwidth of the source or C_i or C_j becomes 0. We first build the solution by adding greedily those trees and set $\alpha_1 = \sum \alpha_1(i, j)$. This process stops when no type 1 tree can be added, *i.e.* $\alpha_1 = b_0$, $n\alpha_1 = M$ or $(m-1)\alpha_1 = N$.

Let us consider the associated three cases:

- $\alpha_1 = b_0$.
The overall weight of this sum of trees is $T = b_0 \geq T^*$.
- $n\alpha_1 = M$.
Provided that none of the outgoing bandwidths of the source and the green nodes are exhausted, it is always possible to add to the solution a tree $T_2(i)$ of weight $\alpha_2(i)$ such that the bandwidth of the source or C_i becomes 0. Let us add greedily those trees, until no type 2 tree can be added and let us set $\alpha_2 = \sum \alpha_2(i)$. Then,
 - If the capacity of the source has been exhausted, then $T = \alpha_1 + \alpha_2 = b_0 \geq T^*$.
 - If the capacity of the green nodes has been exhausted, then we can add to the solution the tree T_4 with weight α_4 such that $\alpha_1 + \alpha_2 + (n+m)\alpha_4 = b_0$. Then, the overall throughput of type 1,2 and 4 trees

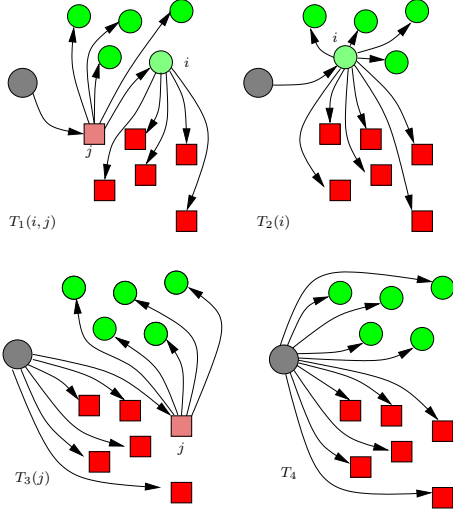


Figure 1. The four types of trees used to reach T^* .

is given by

$$\begin{aligned} (n+m)T &= n\alpha_1 + (m-1)\alpha_1 + (n+m-1)\alpha_2 \\ &\quad + \alpha_1 + \alpha_2 + (n+m)\alpha_4 \\ &= M + N + b_0. \end{aligned}$$

- $(m-1)\alpha_1 = N$.

Provided that none of the outgoing bandwidths of the source and the red nodes is exhausted, it is always possible to add to the solution a tree $T_3(j)$ of weight $\alpha_3(j)$ such that the bandwidth of the source or C_j becomes 0. Let us greedily add those trees until no type 3 tree can be added, and set $\alpha_3 = \sum \alpha_3(j)$. Then,

- If the capacity of the source has been exhausted, then $mT = m(\alpha_1 + \alpha_3) = (m-1)\alpha_1 + \alpha_1 + m\alpha_3 = N + b_0 \geq T^*$.
- If the capacity of the red nodes has been exhausted, then we can add to the solution the tree T_4 with weight α_4 such that $\alpha_1 + m\alpha_3 + (n+m)\alpha_4 = b_0$. Then, the overall throughput of type 1,2 and 4 trees is given by

$$\begin{aligned} (n+m)T &= n\alpha_1 + n\alpha_3 + (m-1)\alpha_1 + \alpha_1 \\ &\quad + m\alpha_3 + (n+m)\alpha_4 \\ &= M + N + b_0. \end{aligned}$$

Therefore, in all cases,

$$T \geq T^* = \min\left(b_0, \frac{b_0 + N}{n}, \frac{b_0 + N + M}{n+m}\right),$$

which achieves the proof of the theorem. \blacksquare

B. Unbounded degree is necessary to achieve optimal throughput

In [8], which corresponds to our model without the red nodes, it has been proved that there always exists a cyclic

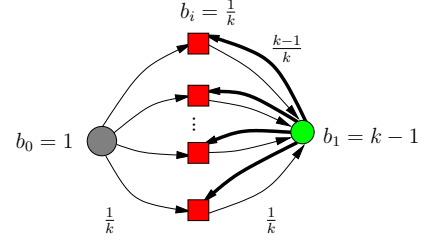


Figure 2. High degree may be necessary to reach T^* .

solution of optimal throughput T^* such that the number of connections that needs to be opened at a node of bandwidth b is at most $\lfloor \frac{b}{T^*} \rfloor + 2$, *i.e.* is almost optimal (up to an additive ratio of 2) if the bandwidth b is necessary in the optimal solution.

Unfortunately, the situation strongly differs when we consider both green and red nodes, as shown in the following example. Let us assume that there is a source node with capacity $b_0 = 1$, one green node with capacity $b_1 = k-1$ and k red nodes with capacity $b_2 = \frac{1}{k}$. Then, using Theorem 3.1, $T^* = \min(b_0, \frac{b_0+b_1}{k}, \frac{b_0+b_1+k b_2}{k+1}) = 1$ and an optimal solution is depicted in Figure 2. In this solution, the source has degree k , whereas $\lfloor \frac{b_0}{T^*} \rfloor = 0$, which strongly differs from the green-only case. In fact, this high degree at the source node is unavoidable to achieve a throughput of T^* . Indeed, since $\frac{b_0+b_1+k b_2}{k+1} = 1$, all bandwidth must be used and since $\frac{b_0+b_1}{k} = 1$, all the bandwidth of the green nodes must be used to feed the red nodes. Therefore, the green node must be fed by all k red nodes at rate $\frac{1}{k}$, and the source node must feed each red node with rate $\frac{1}{k}$. Thus, in some cases, in order to reach the optimal acyclic throughput, the degree of a given node with bandwidth b may be arbitrarily larger than $\lfloor \frac{b_0}{T^*} \rfloor$, what strongly contrasts with the green-only case.

In what follows, we will prove that this property does not hold true in the acyclic case and that it is always possible to achieve the optimal acyclic throughput with degree at most $\lfloor \frac{b_0}{T^*} \rfloor + 2$. Then, we will prove a worst case ratio between the acyclic and the cyclic throughput and show through simulations that the situation is even better in practice, and that optimal acyclic and cyclic throughputs are very close.

IV. ACYCLIC ANALYSIS

In order to design broadcast schemes with reasonably low degree, we study in this section only *acyclic* broadcast schemes. We start by proving dominance relations in order to characterize optimal acyclic schemes, and then propose an algorithm for testing if a given value T is achievable using an acyclic scheme. This algorithm can be combined to a binary search to find the optimal acyclic throughput. We also prove that the solutions computed by this algorithm have low degree. Furthermore, we analyze the worst-case ratio between optimal acyclic and cyclic solutions, to prove that

our algorithm is an approximation algorithm for the original problem.

A. Dominance relations

An ordering σ is said to be *increasing* if its restriction to \mathcal{N} is the identity on \mathcal{N} , and its restriction to \mathcal{M} is the identity on \mathcal{M} . This means that nodes of the same color are ordered by non-increasing order on their bandwidth.

Lemma 4.1:

$$T_{ac}^* = \min_{\sigma: \text{increasing}} \{T_{ac}^*(\sigma)\}.$$

Due to space limitation, the proof of the Lemma has been moved to the Appendix.

An increasing order can be naturally encoded by a binary word π with n letters \bigcirc (corresponding to green nodes) and m letters \square (corresponding to red nodes): it is sufficient to specify if $\sigma(i)$ belongs to \mathcal{N} or to \mathcal{M} . We denote by $|\pi|$ the length of the word π , and by *green*(π) (resp. *red*(π)) the number of letters \bigcirc (resp. \square) in π . $\pi' \sqsubseteq \pi$ (resp. $\pi' \sqsubset \pi$) means that π' is a prefix (resp. a strict prefix) of π . At last, $\pi[i]$ denotes the prefix of length i of π .

From now on, when no confusion is possible, π will be identified with its corresponding increasing order. For instance, $T_{ac}^*(\pi)$ corresponds to the optimal acyclic throughput associated with the order encoded by π . A word π is said to be *valid* (with respect to an instance I and a throughput T) if $T_{ac}^*(\pi) \geq T$.

A solution c is said to be *conservative* with respect to order σ , if there is no quadruple of distinct indices i, j, k, l , such that $i < k < l$ and $j < k < l$, $\sigma(i) \in \mathcal{M}$, $\sigma(j), \sigma(k), \sigma(l) \in \mathcal{N}$, and $c_{\sigma(j), \sigma(k)} > 0$ and $c_{\sigma(i), \sigma(l)} > 0$ simultaneously.

The idea behind this definition is to consider solutions which feed the green nodes from red nodes whenever it is possible. Indeed, the firewall constraint prevents transfer from red nodes to red nodes: transfer from green nodes is thus a valuable resource, and it is a "waste" to use it to feed green nodes when it is not necessary. This means that when creating a *conservative* solution incrementally (by satisfying the nodes in a given order σ), there is no choice for the type of nodes which should feed the next node to add: a red node must be fed by green nodes (because of the firewall constraint), and a green node should be fed by a red node as long as some of them have remaining outgoing capacity.

Lemma 4.2: For every order σ there exists a *conservative* solution c that achieves $T_{ac}^*(\sigma)$.

Due to space limitation, the proof of the Lemma has been moved to the Appendix.

Given a throughput T , and a coding word π with $0 \leq i \leq n$ letters \bigcirc and $0 \leq j \leq m$ letters \square , let \mathcal{C}_π be the set of partial conservative solutions on the partial increasing order encoded by π (that feeds nodes C_1, \dots, C_i and C_{n+1}, \dots, C_{n+j}).

All partial conservative solutions of \mathcal{C}_π have the same amount of available throughput of each type. Let us denote by $G(\pi)$ (respectively $R(\pi)$) the green (respectively red) bandwidth available at the end of the partial solutions of \mathcal{C}_π . G and R satisfy the following recursive equations:

$$\begin{aligned} G(\epsilon) &= b_0, \\ R(\epsilon) &= 0, \\ G(\pi\square) &= G(\pi) - T, \\ R(\pi\square) &= R(\pi) + b_{n+j+1}, \\ G(\pi\bigcirc) &= G(\pi) + b_{i+1} - \max(0, T - R(\pi)), \\ R(\pi\bigcirc) &= \max(0, R(\pi) - T). \end{aligned}$$

The values G and R encompass all the capacity constraints of solutions in \mathcal{C}_π . Indeed, it is easy to see that a coding word π is valid for a throughput T if and only if

$$\begin{aligned} &\text{for all prefixes } \pi'\square \text{ of } \pi, G(\pi') \geq T \\ &\text{for all prefixes } \pi'\bigcirc \text{ of } \pi, G(\pi') + R(\pi') \geq T \end{aligned}$$

Another parameter that is common to each partial conservative solution of \mathcal{C}_π is $W(\pi)$, the amount of transfer going from \mathcal{N} to \mathcal{M} . This parameter satisfies the following recursive equations

$$\begin{aligned} W(\epsilon) &= 0, \\ W(\pi\square) &= W(\pi), \\ W(\pi\bigcirc) &= W(\pi) + \max(0, T - R(\pi)). \end{aligned}$$

From above, we obtain

$$\begin{aligned} R(\pi) &= b_{n+1} + \dots + b_{n+j} - i.T + W(\pi) \quad (1) \\ G(\pi) &= b_0 + b_1 + \dots + b_i - j.T - W(\pi), \quad (2) \end{aligned}$$

and thus $G(\pi) + R(\pi) = \sum_{k=0}^{\text{green}(\pi)} b_k + \sum_{k=n+1}^{n+1+\text{red}(\pi)} b_k - |\pi|T$.

B. Greedy algorithm

In this section, we present Algorithm 1 for testing if a given throughput T is feasible for an instance. It works by iteratively building a partial conservative solution π , deciding at each step how to extend the partial solution (by \bigcirc or by \square). This decision is made greedily, by choosing \square whenever it is possible. The algorithm is forced to take \bigcirc (see line 15):

- when it is not possible to choose \square at the current step ($G(\pi) < T$);
- or when choosing \square would make it impossible to continue afterwards ($G(\pi\square) + R(\pi\square) < T$).

Of course, if all red nodes have been used (line 7), the algorithm chooses \bigcirc . Another special case is when only one red node is left. In that case (see lines 9-14), the algorithm chooses at each step the node with the largest b_i (unless it is red and $G(\pi) < T$).

Algorithm 1 GreedyTest (T)

```
1:  $\pi \leftarrow \epsilon$ 
2: while  $|\pi| < n + m$  do
3:   if  $G(\pi) + R(\pi) < T$  then return FAIL
4:    $i \leftarrow \text{green}(\pi); j \leftarrow \text{red}(\pi)$ 
5:   if  $i = n$  then
6:      $\pi \leftarrow \pi \square$ 
7:   else if  $j = m$  then
8:      $\pi \leftarrow \pi \circ$ 
9:   else if  $j = m - 1$  then
10:    if  $G(\pi) < T$  or  $b_{n+j+1} < b_{i+1}$  then
11:       $\pi \leftarrow \pi \circ$ 
12:    else
13:       $\pi \leftarrow \pi \square$ 
14:    end if
15:  else if  $G(\pi) < T$  or  $G(\pi \square) + R(\pi \square) < T$  then
16:     $\pi \leftarrow \pi \circ$ 
17:  else
18:     $\pi \leftarrow \pi \square$ 
19:  end if
20:  if  $G(\pi) < 0$  then return FAIL
21: end while
22: return  $\pi$ 
```

In the rest of this section, we prove that this algorithm finds an optimal acyclic solution. The first lemma expresses the fact that this algorithm uses green nodes as late as possible, and is as conservative as possible.

Lemma 4.3: Let π_k be the value of π in Algorithm 1 when the k -th green node has just been added to π . ($\text{green}(\pi_k) = k$, and π_k ends with a \circ).

If $\text{red}(\pi_k) < m - 1$, then for every π'_k ending with a \circ such that $\text{green}(\pi'_k) = k$, we have:

$$W(\pi'_k) \geq W(\pi_k) \quad \text{and} \quad \text{red}(\pi'_k) \leq \text{red}(\pi_k).$$

Due to space limitation, the proof of the Lemma has been moved to the Appendix.

We are now ready to prove the main theorem.

Theorem 4.4: Given an instance I and a throughput T , Algorithm 1 returns a valid word (a word π such that $T_{ac}^*(\pi) \geq T$) if and only if T is feasible for this instance ($T_{ac}^* \geq T$).

Proof: The first implication is trivial, since the tests performed at each step of Algorithm 1 ensure that the word returned is always valid.

For the reverse implication, we prove that if Algorithm 1 fails to find a solution, then there does not exist a valid ordering of the nodes with respect to throughput T . According to Lemmas 4.1 and 4.2, we only consider encoding words.

Let ω be the partial solution builded by Algorithm 1 (before it failed), and let $i = \text{green}(\omega)$ and $j = \text{red}(\omega)$.

There are four different cases to consider:

- $j < m - 1$ and ω ends with \circ .

Since Algorithm 1 failed after ω , $G(\omega) + R(\omega) < T$. On the other hand, $G(\omega) \geq b_i$, what implies $b_i < T$ and $\forall k \geq i, b_k < T$.

Let π be an encoding word, and let us consider the largest sub-word π' such that $\text{red}(\pi') = \text{red}(\omega)$. If $\text{green}(\pi') < \text{green}(\omega)$, then there exists a word $\rho \sqsubseteq \pi$ such that $\text{green}(\rho) = \text{green}(\omega)$ and $\text{red}(\rho) > \text{red}(\omega)$. Since this violates the conclusions of Lemma 4.3, π is not valid.

If $\text{green}(\pi') \geq \text{green}(\omega)$, then

$$\begin{aligned} G(\pi') + R(\pi') &= G(\omega) + R(\omega) + \sum_{k=i+1}^{\text{green}(\pi')} (b_k - T) \\ &\leq G(\omega) + R(\omega) < T. \end{aligned}$$

In conclusion, $G(\pi') < T$ and thus π is not valid.

- $j \leq m - 1$ and ω ends with \square .
Because of the test at line 15, this implies that the last \square was added by the instruction on line 6, and thus $\text{green}(\omega) = n$.
Let π be an encoding word. We can decompose ω and π as $\omega' \circ \square^a$ and $\pi' \circ \square^b$, and we can apply Lemma 4.2 to words $\omega' \circ$ and $\pi' \circ$

$$W(\omega) = W(\omega') \leq W(\pi') = W(\pi)$$

Since $\text{green}(\omega) = n$ and since Algorithm 1 failed, then either $G(\omega) + R(\omega) < T$ or $G(\omega \square) < 0$. In both cases, $G(\omega) < T$, and since $G(\omega) = N - jT - W(\omega)$, we get $N < mT + W(\pi)$, and thus $G(\pi) < 0$. Hence π is not valid.

- $j = m$.
(due to space limitation, the proof has been moved to the Appendix)
- $j = m - 1$ and ω ends with \circ .
(due to space limitation, the proof has been moved to the Appendix)

■

The output of Algorithm 1 is an encoding word, an ordering in which to satisfy the nodes, together with the amounts of red or green bandwidth used for this purpose, but not the actual values of $c_{i,j}$. There are several possibilities for the $c_{i,j}$. However, in order to prove bounds on the degree of the nodes, we will feed each node by the earliest possible nodes with unused upload bandwidth.

Theorem 4.5: From the word π given by Algorithm 1, it is possible to build a broadcast scheme such that

- for every red node $j \in \mathcal{M}$, outdegree d_j is bounded:
 $d_j \leq \left\lceil \frac{b_j}{T} \right\rceil + 1$;
- for at most one green node i , $d_i \leq \left\lceil \frac{b_i}{T} \right\rceil + 3$;
- for the other green nodes, $d_i \leq \left\lceil \frac{b_i}{T} \right\rceil + 2$.

Proof: Because red nodes can only upload to green nodes, and green nodes always receive from the earliest red node available, we have that every red node uploads to a

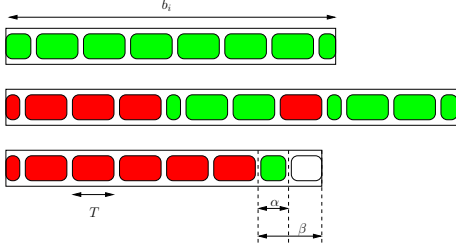


Figure 3. Repartition of the upload of nodes. A red node feeds at most 2 nodes partially (first example). A green node that is the first to feed the last red node (second example). General case for a green node (last example).

consecutive interval of green nodes. So at most 2 nodes will be partially fed by a specific red node: the first and the last one of the interval (see first example of Figure 3).

Now consider a green node i . Because Algorithm 1 rather chooses red nodes when it is possible, as long as there is enough green bandwidth available, node i will feed a consecutive interval of red nodes. When the amount of green upload available gets low, there are two cases to consider:

- i is the earliest green node that feeds the last red node. The sequence of nodes fed by i is a sequence of red nodes, then a sequence of green nodes, then the last red node and another sequence of green nodes (see second example of Figure 3). Since conservatism implies that $R(\pi \circ) = 0$ after feeding a green node from node i , a partially fed green node can only take place as the first node after red nodes. Hence, the only nodes partially fed by i are the first one, the last one and the opening nodes of the green sequences. In total, at most 4 nodes are partially fed by node i .
- Otherwise (see last example of Figure 3). Algorithm 1 feeds red nodes with the upload of i as long as there is enough bandwidth. At one point, $G + R + r_{\text{next}} < 2T$, where r_{next} is the bandwidth of the next red node to be fed. Let β be the remaining bandwidth of node i at that point. By the definition of G , $\beta \leq G$. At this moment, the algorithm decides to switch to green nodes. Green nodes are fed using red bandwidth at first. If any green node is fed using $\alpha = T - R$ upload from i , the remaining upload of i is equal to $\beta - \alpha \leq G + R - T \leq T - r_{\text{next}} \leq T$. Thus, the next node fed by i uses all the remaining bandwidth of node i . Hence, node i feeds partially at most 3 nodes: the first node, one green node and the last node. ■

C. Throughput associated to a given node sequence

For a given instance I and a given sequence π , it is possible to give a formula for the throughput achievable by this sequence. Let $g_i = \text{green}(\pi[i])$ and $bg_i = \sum_{k=1}^{g_i} b_k$ be the output bandwidth of the first g_i green nodes (define r_i and br_i similarly).

Theorem 4.6:

$$T_{ac}^*(\pi) = \min_{i \geq j} \frac{b_0 + bg_i + br_j}{g_{i+1} + r_{j+1}}.$$

Proof: Throughput T is achievable with the sequence π if and only if there exists a non-decreasing sequence $(t_i)_{0 \leq i \leq n+m}$ (which represents the transfer from \mathcal{N} to \mathcal{N} in the prefix of length i) such that

$$\begin{aligned} \forall i, \quad b_0 + bg_i - t_i &\geq Tr_{i+1} && \text{feed the red nodes, and} \\ \forall i, \quad br_i + t_i &\geq Tg_{i+1} && \text{feed the green nodes.} \end{aligned}$$

If T is achievable, then by the assumption on the t_i s, we have $\forall i, \forall j \leq i, t_i \geq Tg_{j+1} - br_j$, which can be combined with the first inequality to yield to $\forall j \leq i, Tr_{i+1} \leq b_0 + bg_i - Tg_{j+1} + br_j$. Hence $T_{ac}^*(\pi) \leq \min_{i \geq j} \frac{b_0 + bg_i + br_j}{g_{i+1} + r_{j+1}}$.

Furthermore, it is easy to see that the throughput $\min_{i \geq j} \frac{b_0 + bg_i + br_j}{g_{i+1} + r_{j+1}}$ is achievable, since sequence $t_i = \max_{j \leq i} T\pi g_{j+1} - br_j$ is non-decreasing and satisfies both inequalities. Hence the second inequality on $T_{ac}^*(\pi)$ also holds. ■

V. ACYCLIC VS CYCLIC

In this section, we compare the optimal acyclic throughput with the optimal (cyclic) throughput. On the one hand we show that the ratio $\frac{T_{ac}^*}{T^*}$ can be as small as $\frac{5}{7}$ for some (small-size) instances and as small as $\frac{1+\sqrt{41}}{8}$ for some arbitrary large instances. On the other hand, we show that this ratio is larger than $\frac{1}{2}$ for any instance. Then, by studying the instances that are hard for the acyclic scheme, we explain why we conjecture that $\frac{T_{ac}^*}{T^*} \geq \frac{5}{7}$ holds true for any instance. Finally we present experimental results on the ratio $\frac{T_{ac}^*}{T^*}$ on random instances, that prove that acyclic solutions provide very good results in practice.

A. First Bounds

Theorem 5.1: There exists an instance such that

$$\frac{T_{ac}^*}{T^*} = \frac{5}{7}.$$

Proof: Let us consider the following instance consisting of one source of throughput 1, one green node of throughput $1+2\epsilon$ and two red nodes with throughput of $1/2-\epsilon$ each. For this instance, $T^* = 1$ (see Theorem 3.1). There also exist 3 increasing orderings $\sigma_1 = 123, \sigma_2 = 213$ and $\sigma_3 = 231$. Ordering σ_1 achieves a throughput of $T_{ac}^*(\sigma_1) = (2/3) \cdot (1 + \epsilon)$ and ordering σ_2 achieves a throughput of $T_{ac}^*(\sigma_2) = 3/4 - \epsilon/2$ (see Figure 4). The throughput of the last ordering is always smaller than the maximum of the two previous ones. When $\epsilon = 1/14$, orderings σ_1 and σ_2 achieve the same throughput: $T_{ac}^* = 5/7$. ■

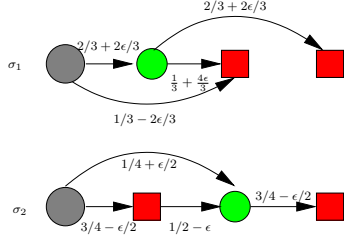


Figure 4. Optimal acyclic schemes of σ_1 and σ_2 .

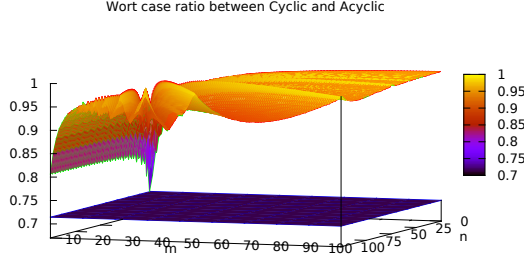


Figure 5. Worst case ratio between cyclic and acyclic optimal solutions on tight homogeneous instances. The bottom plane is $\frac{5}{7} \approx 0.714$.

Theorem 5.2: For every $\epsilon > 0$ and every $K \in \mathbb{N}$, there exist instances with at least K green nodes and K red nodes such that

$$\frac{T_{ac}^*}{T^*} \leq \frac{1 + \sqrt{41}}{8} + \epsilon \approx 0.925 + \epsilon.$$

Due to space limitations, the proofs of the theorem is omitted here and can be found in the Appendix.

An instance is said to be *homogeneous* if all green nodes except the source have the same throughput g and all red nodes have the same throughput r . An instance is said to be *tight* if $b_0 = \frac{b_0 + N + M}{n + m} = T^*$ (i.e. if no bandwidth can be wasted in the optimal cyclic solution). Observe that the instances given in the proofs of Theorems 5.1 and 5.2 are tight and homogeneous. Due to space limitations, the proofs of the following lemma, that state that we can restrict the search of worst case to homogeneous and tight instances, and the proof of the following theorem that provides a lower bound for the worst case ratio, can be found in the Appendix.

Lemma 5.3: Let $\alpha > 0$. If for every tight homogeneous instance, $\frac{T_{ac}^*}{T^*} \geq \alpha$, then for every instance $\frac{T_{ac}^*}{T^*} \geq \alpha$.

Theorem 5.4: For any instance, $\frac{T_{ac}^*}{T^*} \geq \frac{1}{2}$.

B. Worst-case exploration

We conjecture that the bound $\frac{5}{7}$ for the ratio $\frac{T_{ac}^*}{T^*}$ given in Theorem 5.1 is tight (i.e., the bound of Theorem 5.4 can be improved to $\frac{5}{7}$). Figure 5 shows the worst-case ratio for all tight and homogeneous instances, for n and m between

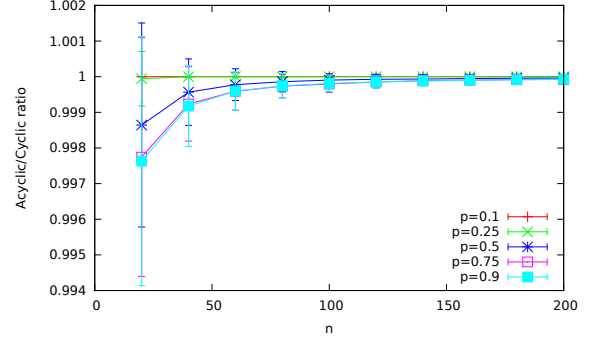


Figure 6. Average ratio between cyclic and acyclic optimal solutions on randomly generated instances

0 and 100. Thanks to Lemma 5.3, this figure gives strong intuitions that $\frac{T_{ac}^*}{T^*} \leq \frac{5}{7}$ for all instances. It is also possible to observe the result of Theorem 5.1: when $m \simeq \frac{\sqrt{41}-3}{8}n$ (for example $n = 100$ and $m = 42$), the ratio remains below 1, even for large values of n and m .

C. Average case exploration

We also analyze the average ratio between acyclic and cyclic throughput on randomly generated instances. The bandwidth values of the nodes is generated with a uniform distribution between 1 and 100, and each node is independently chosen to be a "green" node with probability p (and red with probability $(1-p)$). The results are shown on Figure 6, for different numbers of nodes and different values of p . For each set of parameters, 1000 random instances were generated, and the figures show average values and confidence intervals at 5%.

These simulations show that random instances with a large proportion of green nodes are slightly more difficult. Nevertheless, even on instances with a large proportion of green nodes, the throughput of the acyclic solution is very close to the optimal (cyclic) throughput.

VI. CONCLUSIONS

We have considered the problem of broadcasting a large message in a large scale distributed network, in the case where some participating nodes lie behind firewalls and cannot communicate directly, but rather need to use as gateways nodes that lie in the open-Internet. The impact of the presence of nodes behind firewalls and NATs, although widely observed in P2P networks, has not been widely studied from a theoretical point of view. In this paper, we prove that when we consider the problem of maximizing the rate of the broadcast operation while minimizing the number of TCP connections opened at each node, the results strongly differ from the case of the fully open-Internet case. On the other hand, we prove that restricting the search to acyclic solutions is efficient for the design of both low degree and high throughput solutions.

REFERENCES

- [1] X. Zhang, J. Liu, B. Li, and Y. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, 2005.
- [2] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Mapping the PPLive network: Studying the impacts of media streaming on P2P overlays," *Department of Computer Science, University of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS*, pp. 2006–275.
- [3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 298–313, 2003.
- [4] S. Johnsson and C. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Transactions on Computers*, vol. 38, no. 9, pp. 1249–1268, 1989.
- [5] J. Watts and R. Geijn, "A pipelined broadcast for multidimensional meshes," *Parallel Processing Letters*, vol. 5, no. 2, pp. 281–292, 1995.
- [6] Y. Tseng, S. Wang, and C. Ho, "Efficient broadcasting in wormhole-routed multicomputers: a network-partitioning approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 1, pp. 44–61, 1999.
- [7] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, 2007, pp. 1073–1081.
- [8] O. Beaumont, L. Eyraud-Dubois, and S. Agrawal, "Broadcasting on large scale heterogeneous platforms under the bounded multi-port model," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–11.
- [9] J. Rosenberg, R. Mahy, and P. Matthews, "Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun)," 2010.
- [10] T. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *IEEE INFOCOM 2002*, New York, NY, USA, 2002, pp. 170–179.
- [11] J. Ledlie, P. Gardner, and M. Seltzer, "Network coordinates in the wild," in *4th USENIX Symposium on Networked Systems Design & Implementation*, 2007, pp. 299–311.
- [12] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2004, pp. 15–26. [Online]. Available: <http://doi.acm.org/10.1145/1015467.1015471>
- [13] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella, "On the treeness of internet latency and bandwidth," in *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2009, pp. 61–72.
- [14] O. Beaumont, L. Eyraud-Dubois, and Y. Won, "Using the last-mile model as a distributed scheme for available bandwidth prediction," in *Proceedings of the EuroPar 2011 conference*, 2011.
- [15] B. Hong and V. Prasanna, "Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput," *International Parallel and Distributed Processing Symposium*, 2004.
- [16] M. A. Brown, "Traffic Control HOWTO. Chapter 6. Classless Queuing Disciplines," <http://tldp.org/HOWTO/Traffic-Control-HOWTO/classless-qdiscs.html>, 2006.
- [17] B. Hubert *et al.*, "Linux Advanced Routing & Traffic Control. Chapter 9. Queuing Disciplines for Bandwidth Management," <http://lartc.org/lartc.pdf>, 2002.
- [18] O. Beaumont and H. Rejeb, "On the importance of bandwidth control mechanisms for scheduling on large scale heterogeneous platforms," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. IEEE, pp. 1–12.
- [19] S. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," *Arxiv preprint cs/0412017*, 2004.
- [20] S. Guha, N. Daswani, and R. Jain, "An experimental study of the skype peer-to-peer voip system," in *Proceedings of IPTPS*, vol. 6, 2006.
- [21] R. Jimenez, F. Osmani, and B. Knutsson, "Connectivity properties of mainline bittorrent dht nodes," in *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE, 2009, pp. 262–270.
- [22] P. Srisuresh, B. Ford, and D. Kegel, "State of peer-to-peer (p2p) communication across network address translators (nats)," 2008.

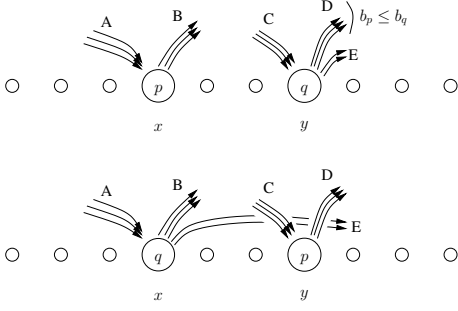


Figure 7. Exchange argument for dominance of increasing solutions

APPENDIX

A. Missing proofs of Section IV

Proof: of Lemma 4.1. Let c be an acyclic solution with order σ which is not increasing. Then there exists two indices $x < y$ such that $p = \sigma(x) > q = \sigma(y)$ (and thus $b_p \leq b_q$). We will exhibit another acyclic solution c' with order $\sigma' = \sigma \circ (x, y)$ (where (x, y) denotes the transposition which exchanges x and y , which means that the nodes in position x and y are swapped) and whose throughput is not smaller than c .

The transformation is depicted on Figure 7. For most indices i, j , it is sufficient to set $c'_{\sigma'(i), \sigma'(j)} = c_{\sigma(i), \sigma(j)}$. However this would break the bandwidth constraint of node $p = \sigma(x)$, and the solution is to give the connections in excess (denoted as E in Figure 7) to node $q = \sigma'(x)$. Since $x < y$, this does not break acyclicity.

Recursively, we can thus transform any acyclic solution into an increasing acyclic solution with at least the same throughput. ■

Proof: of Lemma 4.2. Let c be a solution that achieves $T_{ac}^*(\sigma)$. If there exists a quadruple of indices i, j, k, l that violates *conservativeness*, we can build a solution c' which is conservative with respect to these indices (see Figure 8). Let $M = \min\{c_{\sigma(i), \sigma(l)}, c_{\sigma(j), \sigma(k)}\}$, and set:

$$\begin{aligned} c'_{\sigma(i), \sigma(l)} &= c_{\sigma(i), \sigma(l)} - M & c'_{\sigma(j), \sigma(k)} &= c_{\sigma(j), \sigma(k)} - M \\ c'_{\sigma(i), \sigma(k)} &= c_{\sigma(i), \sigma(k)} + M & c'_{\sigma(j), \sigma(l)} &= c_{\sigma(j), \sigma(l)} + M \end{aligned}$$

and c and c' coincide on all other indices. It is easy to see that c' is a valid solution of the same throughput, and that the number of quadruples of indices violating *conservativeness* is lower in c' . Recursively, we create a *conservative* acyclic solution with respect to order σ , with throughput $T_{ac}^*(\sigma)$. ■

The following technical lemma will be useful:

Lemma 6.1: Let π_1, π_2 be two conservative partial solutions such that $green(\pi_1) = green(\pi_2)$ and $red(\pi_1) = red(\pi_2)$. If $W(\pi_1) \leq W(\pi_2)$, then $\forall \omega \in \{\circ, \square\}^*$, $W(\pi_1\omega) \leq W(\pi_2\omega)$.

Proof: To prove the lemma we only have to consider the cases where $\omega \in \{\circ, \square\}$. The case $\omega = \square$ is trivial since $W(\pi\square) = W(\pi)$.

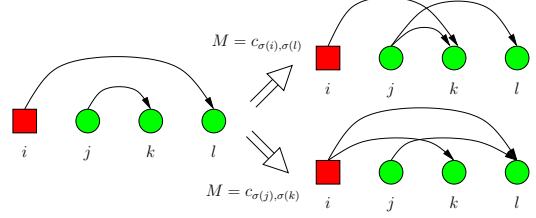


Figure 8. Exchange argument for dominance of conservative solutions

Let us consider now the case $\omega = \circ$:

$$\begin{aligned} W(\pi_1\circ) &= \max(W(\pi_1), W(\pi_1) + T - R(\pi_1)) \\ &= \max(W(\pi_1), T + i \cdot T - b_{n+1} - \dots - b_{n+j}) \\ &\leq \max(W(\pi_2), T + i \cdot T - b_{n+1} - \dots - b_{n+j}) \\ &\leq W(\pi_2\circ) \end{aligned}$$

■

Proof: of Lemma 4.3. We prove this lemma by induction on k . Clearly the lemma holds for $k = 0$, since $\pi_0 = \epsilon = \pi'_0$.

Assume now that the conclusions of the lemma hold for $k - 1$, and decompose the words maximally as follows:

$$\begin{aligned} \pi_k &= \pi_{k-1} \square^a \circ & \text{and note } \delta &= \pi_{k-1} \square^a, \\ \pi'_k &= \pi'_{k-1} \square^{a'} \circ. \end{aligned}$$

Let $l = red(\pi_k)$ and $l' = red(\pi'_k)$. From (1) and (2), we get:

$$\begin{aligned} G(\delta) &= b_1 + \dots + b_{k-1} - l \cdot T - W(\pi_{k-1}), \\ R(\delta) &= b_{n+1} + \dots + b_{n+l} - (k-1) \cdot T + W(\pi_{k-1}). \end{aligned}$$

Since Algorithm 1 chooses \circ (after choosing δ), and $red(\delta) < m - 1$, we have: $G(\delta) < T$ or $G(\delta) + R(\delta) + b_{n+l+1} < 2T$.

Let us first prove by contradiction that $red(\pi'_k) \leq red(\pi_k)$. Assume that $red(\pi'_k) > red(\pi_k)$. In this case, there exists $\delta' \sqsubseteq \pi'$ such that $|\delta'| = |\delta|$. By inductive assumption, $red(\pi_{k-1}) \geq red(\pi'_{k-1})$, which implies that $|\pi'_{k-1}| \leq |\pi_{k-1}| \leq |\delta|$. Hence, $green(\delta') = green(\pi') - 1 = k - 1$. We can thus compute:

$$\begin{aligned} G(\delta') &= b_1 + \dots + b_{k-1} - l \cdot T - W(\pi'_{k-1}) \\ &\leq b_1 + \dots + b_{k-1} - l \cdot T - W(\pi_{k-1}) = G(\delta), \\ G(\delta') + R(\delta') &= \sum_{i=1}^{k-1} b_i - l \cdot T + \sum_{i=n+1}^{n+l} b_i - (k-1) \cdot T \\ &= G(\delta) + R(\delta). \end{aligned}$$

So either $G(\delta') < T$ or $G(\delta') + R(\delta') + b_{n+l+1} < 2T$. Both lead to a contradiction when we try to continue δ' with \square . This proves that $red(\pi'_k) \leq red(\pi_k)$.

Let us now prove that $W(\pi'_k) \geq W(\pi_k)$. As π_k and π'_k end with \bigcirc :

$$\begin{aligned} W(\pi_k) &= W(\pi_{k-1}) + \max(0, T - R(\delta)) \\ &= \max(W(\pi_{k-1}), T \cdot k - (b_{n+1} + \dots + b_{n+l})), \\ W(\pi'_k) &= \max(W(\pi'_{k-1}), T \cdot k - (b_{n+1} + \dots + b_{n+l'})). \end{aligned}$$

Since $l' \leq l$ and $W(\pi'_{k-1}) \geq W(\pi_{k-1})$ (the inductive assumption), we have $W(\pi'_k) \geq W(\pi_k)$. ■

B. Missing cases of the proof of Theorem 4.4

- $j = m$. (skipped) Since Algorithm 1 chooses the last red node at some point (line 14), we have $b_{i+1} \leq b_{n+m}$. The failure of the algorithm implies $G(\omega) + R(\omega) < T$. Let $\omega = \omega'\alpha$. We know that $G(\omega') + R(\omega') \geq T$, and also

$$\begin{aligned} G(\omega) + R(\omega) &= G(\omega') + R(\omega') - T + b_i & \text{if } \alpha = \bigcirc, \\ G(\omega) + R(\omega) &= G(\omega') + R(\omega') - T + b_{n+m} & \text{if } \alpha = \square. \end{aligned}$$

So either $b_{n+m} < T$ or $b_i < T$. In both cases, we have

$$b_n \leq b_{n-1} \leq \dots \leq b_{i+1} < T.$$

Let $\pi = \pi'\beta$ be an encoding word. If $\beta = \bigcirc$, then

$$\begin{aligned} G(\pi') + R(\pi') &= b_0 + N - b_n - (n-1)T + M - mT \\ &= G(\omega) + R(\omega) + \sum_{k=i+1}^{n-1} (b_k - T) \\ &\leq G(\omega) + R(\omega) < T \end{aligned}$$

Hence π is not valid. Otherwise, $\beta = \square$, and $G(\pi') + R(\pi') = b_0 + N - nT + M - b_{n+m} - (m-1)T$. Since $b_{n+m} \geq b_n$, we get the same conclusion.

- $j = m-1$ and ω ends with \bigcirc . As in the first case, we have $\forall k \geq i, b_k < T$. Let us decompose ω as $\omega = \omega'\bigcirc^a$ ($a \geq 0$). We begin by showing that the words $\pi(x) = \omega'\square\bigcirc^x$ $\square\bigcirc^{a-x}$ are invalid for throughput T . The following lemma shows that it is possible to consider only words where the last \square is followed only by \bigcirc with smaller bandwidth.

Lemma 6.2: If the word $\pi = \pi_1\square\bigcirc\bigcirc^a$ is a valid word in which the last \square has bandwidth r , the following \bigcirc has bandwidth g , and $g \geq r$, then the word $\pi' = \pi_1\bigcirc\square\bigcirc^a$ is also valid.

Proof: Let $G = G(\pi_1)$, $R = R(\pi_1)$ and $\pi_2 = \bigcirc^a$. Since π is valid, we have $G \geq T$ and $G - T + R + r \geq T$. We can thus bound $G(\pi_1\bigcirc)$:

$$\begin{aligned} G(\pi_1\bigcirc) &= G + g - \max(T - R, 0) \\ &= \min(G + g + R - T, G + g) \geq T. \end{aligned}$$

This ensures that $\pi_1\bigcirc\square$ is a valid sequence.

Since π_2 is composed only of \bigcirc , and $G(\pi_1\square\bigcirc) + R(\pi_1\square\bigcirc) = G(\pi_1\bigcirc\square) + R(\pi_1\bigcirc\square)$, $\pi_1\bigcirc\square\pi_2$ is a valid sequence. ■

If $\pi(x)$ is valid, we can iteratively use Lemma 6.2 to prove the existence of a valid $\pi(y)$ in which the last \square is followed by a \bigcirc with smaller upload. If $y < a$, since Algorithm 1 at that point chooses \bigcirc instead of \square , we know that $G(\omega'\square\bigcirc^y) < T$ and $\pi(y)$ is invalid. If $y = a$, then $\pi(y) = \omega\square$, which is invalid because Algorithm 1 failed.

Consider now any encoding word π . Let $\pi = \pi_1\pi_2\square\bigcirc^k$ be the decomposition with minimal π_1 having $green(\pi_1) = green(\omega')$ (applying Lemma 4.3 we have $red(\pi_1) \leq red(\omega') = m-2$, so decomposing is always possible).

For any word δ we have:

$$\begin{aligned} W(\delta\bigcirc\square) &= W(\delta\bigcirc) = W(\delta) + \max(0, T - R(\delta)) \geq \\ &\geq W(\delta\square) + \max(0, T - R(\delta\square)) = W(\delta\square\bigcirc). \end{aligned}$$

We can apply this to the word π :

$$W(\pi_1\pi_2) \geq W(\pi_1\square^{red(\pi_2)}\bigcirc^{green(\pi_2)}).$$

Since Lemma 4.3 applies to π_1 and ω' :

$$W(\pi_1) \geq W(\omega').$$

So by Lemma 6.1, (since $green(\pi_1) = green(\omega')$, $red(\pi_1\pi_2) = m-1 = red(\omega'\square)$):

$$W(\pi_1\square^{red(\pi_2)}\bigcirc^{green(\pi_2)}) \geq W(\omega'\square\bigcirc^{green(\pi_2)}).$$

Composing it, we have:

$$W(\pi_1\pi_2) \geq W(\omega'\square\bigcirc^{green(\pi_2)}) \text{ and}$$

$$\forall x, W(\pi_1\pi_2\square\bigcirc^x) \geq W(\omega'\square\bigcirc^{green(\pi_2)}\square\bigcirc^x).$$

So if $\pi = \pi_1\pi_2\square\bigcirc^k$ is valid, then $\omega'\square\bigcirc^{green(\pi_2)}\square\bigcirc^k$ is also valid. But we proved previously that no such solution can exist. So we have reached a contradiction.

C. Missing proofs of Section V

Proof: of Theorem 5.2. For a given $\alpha = \frac{p}{q} < 1$, with p and q integers, and for any k , let us consider the instance $I(\alpha, k)$ such that:

- $b_0 = 1$;
- $n = kq$ green nodes have bandwidth α ; and
- $m = kp$ red nodes have bandwidth $\frac{1}{\alpha}$.

The first observation is that for all α and k , Theorem 3.1 yields that the optimal throughput T^* is equal to 1.

For the second observation, let S be an acyclic solution to $I(\alpha, k)$ and x be the number of green nodes before the second red node in S . In other words, S starts with a prefix $\pi = \bigcirc^u\square\bigcirc^v\square$ with $u + v = x$. The throughput T achievable by S is bounded by two constraints:

- the source and the x first green nodes should be able to feed the 2 first red nodes: $\alpha x + 1 \geq 2T$; and

- the bandwidth of the source and of the $x+1$ first nodes should be enough to feed the $x+2$ nodes: $\alpha x + \frac{1}{\alpha} + 1 \geq (x+2)T$.

Hence $T \leq \frac{\alpha x + 1}{2} = f_\alpha(x)$ and $T \leq \frac{\alpha x + \frac{1}{\alpha} + 1}{x+2} = g_\alpha(x)$. Since an optimal acyclic scheme must respect these constraints as well for some x , we have $T_{ac}^* \leq \max_{x \in \mathbb{N}} \min(f_\alpha(x), g_\alpha(x))$.

Observe now that the function f_α is increasing, and g_α is decreasing (since $\alpha < 1$), and that they coincide (with value 1) for $x = \frac{1}{\alpha}$. The minimum is thus realised by f_α for $x < 1/\alpha$, and by g_α for $x > 1/\alpha$, and this minimum is maximised for $x = 1/\alpha$. However, $\frac{1}{\alpha}$ is not necessarily an integer, so the maximal value is attained for $x = \lfloor \frac{1}{\alpha} \rfloor$ or $x = \lceil \frac{1}{\alpha} \rceil$:

$$T_{ac}^* \leq \max\left(f_\alpha\left(\left\lfloor \frac{1}{\alpha} \right\rfloor\right), g_\alpha\left(\left\lceil \frac{1}{\alpha} \right\rceil\right)\right).$$

If $\alpha = \frac{\sqrt{41}-3}{8}$, simple computations show that $\lfloor \frac{1}{\alpha} \rfloor = 2$, $\lceil \frac{1}{\alpha} \rceil = 3$, and $f_\alpha(2) = g_\alpha(3) = \frac{\sqrt{41}+1}{8}$. Since this value of α can be approximated arbitrarily close with a rational number, and since the expressions $f_\alpha(2)$ and $g_\alpha(3)$ are continuous in α , we have the desired result. ■

Proof: of Lemma 5.3. To prove this lemma we will show that given an instance, we can associate with it a tight homogeneous instance with the same optimal throughput T^* and with no greater optimal acyclic throughput T_{ac}^* .

Firstly, if the instance is such that $\frac{b_0 + N + M}{n + m} > T^*$, by reducing the throughput of the red nodes it is possible to make this inequality an equality. This transformation doesn't change the optimal throughput T^* and any acyclic solution for the transformed instance is also an acyclic solution for the original one.

Consider now a non-homogeneous instance I such that $\frac{b_0 + N + M}{n + m} = T^*$. Let I' be the homogeneous instance obtained from I as follows: $b'_0 = T^*$, $b'_i = (N + b_0 - T^*)/n$ for $i \in \llbracket 1, n \rrbracket$ and $b'_i = M/m$ for $i \in \llbracket n + 1, n + m \rrbracket$ where b'_i is the throughput of the node C_i in I' . Clearly I and I' have the same optimal throughput T^* and I' is tight and homogeneous. Observe that since nodes of same color are ordered in the non-increasing order of their throughput, $\forall k \in \llbracket 0, n \rrbracket, \sum_{i=0}^k b_i \geq \sum_{i=0}^k b'_i$ and $\forall k \in \llbracket n + 1, n + m \rrbracket, \sum_{i=1}^k b_i \geq \sum_{i=1}^k b'_i$. Hence, any acyclic scheme of I' can be turned into a scheme of I communications ensured by the k -th green (resp. red) node in I' are ensured by the k first green (resp. red) nodes in I . The scheme thus produced is also acyclic and of same throughput. ■

Proof: of Theorem 5.4. From Lemma 5.3, we only have to consider tight homogeneous instances. Without loss of generality, we can also assume that $T^* = 1$. Set ϵ such that $\frac{b_0 + N}{m} = (1 + \epsilon)T^*$. It is easy to compute the bandwidth of green and red nodes: $g = (m + m \cdot \epsilon - 1)/n$ and $r = n/m - \epsilon$, and $0 \leq \epsilon \leq n/m$.

For each instance, we build an order σ which is valid for throughput $\frac{1}{2}$, hence $T_{ac}^*(\sigma) \geq 1/2$.

Let $k = \lceil n/m \rceil$ and $k' = \lceil m/n \rceil$.

There are several cases to consider:

- $n \geq m$ and $r \geq g$.

We consider blocs composed of 1 red node followed by k green nodes. After these blocs, we can have a bloc with less than k green nodes, and then some red nodes. One can check that once the red node of a bloc receives the message, it is possible to build an acyclic solution in this bloc: $r + g(k-1) \geq \frac{1}{2}k$ (cf. Algorithm 1 of [8]). Moreover, one can also check that the remaining green throughput of the bloc is able to feed the red node of the next bloc: $gk - \max(\frac{1}{2}k - r, 0) \geq \frac{1}{2}$ (observe that $\max(\frac{1}{2}k - r, 0)$ is the green-green transfer inside the bloc). Finally, computations prove that $gn - \lfloor \frac{n}{k} \rfloor \max(\frac{k}{2} - r, 0) \geq \frac{1}{2}m$: the green throughput available at the end of all the blocs is enough to feed all the red nodes.

- $n \geq m$ and $r \leq g$.

We consider blocs composed of k green nodes followed by one red node. After these blocs, we can have a bloc with less than k green nodes, and then some red nodes. Since $r \leq g$, we have $g \geq 1/2$. Hence it is possible to build an acyclic solution in each bloc providing that the first node of the bloc is fed. This is possible because each complete bloc has an excess of throughput: $gk + r - (k-1)/2 \geq 0$.

Now, let us notice that a complete bloc consumes all the red throughput provided by the previous bloc. From this observation it is possible to estimate the green throughput available for red nodes:

$$1 + ng - \left(\lfloor \frac{n}{k} \rfloor - 2\right)r - n/2.$$

One can check that this throughput is enough to feed the red nodes (it is greater than $m/2$).

- $n < m$.

We consider blocs composed of 1 green node and k' red nodes. As before, the last bloc may contain less red nodes, and after the blocs there may be some green nodes. In each bloc, the green node is able to feed the red nodes of its bloc. Moreover each bloc (complete or not) is able to feed the green node of the next bloc.

For each tight homogeneous instance, we have provided a scheme that produces a throughput of $1/2$ where $T^* = 1$. ■