Competitive Multicast Routing

Baruch Awerbuch [*] **Yossi Azar** [†]

Abstract

In this paper, we introduce and solve the multicast routing problem for virtual circuit environment *without making any assumptions* about the communication patterns, or about the network topology. By multicast we refer to the case were one source transmits to several destination the same information. Also, we allow arbitrary interleaving of subscription patterns for different multicast groups, i.e. the destinations for each group arrive at an arbitrary order and may interleaved with destinations of other groups. Our goal is to make route selection so as to minimize congestion of the bottleneck link. This is the first analytical treatment for this problem in its full generality.

The main contribution of this paper is an online competitive routing strategy that has an $O(\log n \log d)$ competitive factor where n is the size of the network and $d(\leq n)$ is the maximum size of a multicast group.

[*]Lab. for Computer Science, MIT. Supported by Air Force Contract TNDGAFOSR-86-0078, ARO contract DAAL03-86-K-0171, NSF contract 9114440-CCR, DARPA contract N00014-J-92-1799, and a special grant from IBM.

[†]Department of Computer Science Tel-Aviv University. E-mail: azar@math.tau.ac.il

1 Introduction

1.1 Informal statement of the problem and the result

Bandwidth utilization is the bottleneck resource for communication-intensive applications such as multicast. The problem could be particularly acute in low-bandwidth wireless networks. In this paper, we show how to cleverly select the multicast routes, so as to satisfy all the multicast subscription requests, while minimizing the "bottleneck congestion", i.e. maximal traffic over a network link. Performance of wireless network in terms of quality of service, delay, and reliability would definitely improve if we could reduce the packet load.

For the sake of simplifying the arguments, the discussion in this paper applies to source-routed circuit switching environments with permanent virtual circuits; using the methods in [AKP$^+$93] it can be extended to either temporary virtual circuits or packet-switched environments. However, the performance guarantee of the latter cases deteriorates by a factor of $O(\log T)$.

We assume the most general case of arbitrarily interleaved subscription patterns for different multicast groups. That is, the subscription requests are of the form:

1. Bill Clinton (White House) subscribes to CNN Headline News, from 6 to 6.30,

2. Bob Dole (DC) subscribes to CNN Headline News, from 6 to 6.30,

3. Madonna (LA) subscribes to MTV, from 6 to 7,

4. Saddam Hussein (Baghdad) subscribes to CNN Headline News, from 6 to 6.30, etc.

Clearly, at the time Bob Dole makes his request for CNN, we can save bandwidth and avoid routing CNN traffic all the way from Atlanta, by forwarding to him CNN traffic from the White House, since Clinton's subscription has already been accommodated.

Also, note that in between two subscriptions to CNN (Clinton, Hussein) other users subscribe to different groups and the communication path established for Madonna may block the future subscription for CNN by Hussein. This is in fact one of the complications of the "interleaved" subscription pattern.

In reality, subscription patterns will not be predictable. Therefore, this paper introduces and solves the multicast routing problem for virtual circuit environment *without making any assumptions* about the communication patterns, or about the network topology. Also, we allow arbitrary interleaving of subscription patterns for different multicast groups. Our goal is to make route selection so as to minimize congestion of the bottleneck link. This is the first analytical treatment for this problem in its full generality.

The main contribution of this paper is an online routing strategy which performs within a $O(\log n \log d)$ factor away from optimal prescient strategy, on each input instance, where n denote the size of the network and $d \leq n$ is the largest size of a broadcast group in the network.

1.2 The complications of online decision making

If we knew ahead of time the set of all users in all multicast connections, then, in principle, we could have solved this *offline* problem. Unfortunately, this combinatorial optimization problem is very complex (NP-hard). In other words, exact solution, in all likelihood, will require an equivalent of an exhaustive search. The best known approximation schemes guarantee worst-case performance ratio that is at least logarithmic ($O(\log n)$) in the number n of network nodes.

In addition to the inherent algorithmic complexity, the situation is further complicated by the aforementioned fact that the knowledge of future subscriptions to multicast connections is not available at the time when routing decisions are performed.

Much of the existing analytical work in the area of networking, both in virtual circuit and packet-switched models, deals with these kind of problems by imposing statistical assumptions on the traffic patterns, e.g. see [Kel86]. Some of these assumptions, e.g. traffic independence assumption [Kel86], is particularly inaccurate for the virtual circuit setting. Other assumptions, typically used in the telecommunications context, e.g. Poisson arrival rate, exponential etc. may or may not be applicable to new high-speed environments. In any case, even if *perfect* statistical knowledge about communication pattern were available, the corresponding combinatorial optimization problem is still computationally hard.

Another complication here is that once a specific user joins the group and starts receiving traffic over a certain path, it is difficult, if not impossible, to reroute a different communication path, without disruption of the real-time service and thus violating the performance guarantees. Even in somewhat easier case of uni-cast (point to point) communication rerouting is considered difficult [GKR94] and it is not used in the current of Gigabit rate networks (*e.g.* PARIS/plaNET [ACG+90, CG88, CGG91, CGG+93] or in ATM standards [XVI88, Bou92]). We also will not consider preemption or rejection of connections [GG92, GGK+93].

Because of the unfeasibility of the rerouting or preemption options, the routing decisions are in effect "irreversible". Since connections arrive in an online manner, knowledge about future connections is not available at the time decisions are made about routing of previous connections. [GG92, GGK+93].

For example, at the time the connection was routed over a specific link, that link may be lightly loaded, and yet, in the future, this link may turn out to be the "bottleneck" link, that must be used for most of the subsequent connections. Thus, our earlier decision to route along that link has proved to be a grave mistake.

This is exactly the problem with the "natural" online heuristics, both from the theoretical and the experimental perspective [GKR94]. These "natural" heuristics include, among others,

- *min-hop:* the new subscription is established on the shortest path, in terms of the number of links ("hops"), from the requesting node to any node previously subscribed to the requested multicast group.

- *min-max:* the new subscription is established on any path, from the requesting node to any node, previously subscribed to the requested multicast group, so as to minimize the traffic over the most congested ("bottleneck") link.

1.3 Competitive algorithmic design

Recently, network implementors and architects have embraced a different design philosophy [GKR94], that can be characterized as "competitive algorithmic design". Namely, in contrast to existing work, one seeks algorithms whose performance does *not* depend on statistical or other assumptions about the communication patterns, network topology, etc. Rather, the algorithms we are seeking ought to be "uniformly-efficient" on *all* inputs, not just on some "benchmarks" or "typical cases".

More precisely, the standard way to measure performance of online algorithms is to consider their "competitive ratio", which is the worst-case performance ratio between online and optimal offline algorithms on a specific input instance (maximized over all input instances). In our case, the performance measure is the maximum edge congestion, which was also studied, in the context of *uni-cast* communication, in [ANR92, ABK92, AAF$^+$93, AAPW94]. The latter work provided also $O(\log n)$ competitive algorithms for the uni-cast communication, which forms the basis for the work in this paper. In the multicast setting, this work can be directly applied to yield competitiveness results in the special case in which all the users of a specific multicast group are known at the time that the group is formed. In contrast, in this paper, we consider the general case of unconstrained subscription pattern.

An algorithm with "small" competitive ratio is efficient in a very robust mathematical sense. Also, such "uncertainty-tolerant" algorithms tend to use natural "greedy-like" heuristics, which are attractive to implementors because of their computational simplicity.

Even though the "algorithmic theory" only guarantees logarithmic performance ratio, which may seem excessive in practice, one should remember that logarithmic performance gap is w.r.t. offline, i.e. optimum prescient strategy, with complete knowledge of the future, and infinite computational resources, which, in fact, is a non-option. The correct interpretation of logarithmic competitive ratio is that the algorithm cannot be "embarrassed" on any particular instance. In order to evaluate performance of the algorithm in practice, one needs to compare it against other online heuristic by experimentation on realistic topology and traffic patterns. In fact, the experimental evidence obtained by considering topologies and communication patterns from real commercial networks [GKR94] indicates that the natural heuristics (min-max and a number of variations of min-hop) are inferior to logarithmically-competitive algorithms in terms of performance.

1.4 Our approach to multicast

Our goal in this paper is to extend the "competitive methodology" to the setting of multicast. We are suggesting a new competitive multi-cast routing strategy for circuit switched network environments, with $O(\log n \log d)$ competitive ratio, where n denote the size of the network and $d \leq n$ is the largest size of a broadcast group in the network. That is, the bottleneck congestion of our online strategy is within $O(\log n \log d)$ factor away from optimal prescient strategy, on each input instance. Note that the case $d = 2$ (i.e. two participants in each multicast group) reduces to the uni-cast connections (one the participants is source and another is destination), in which case $O(\log n)$ competitiveness is achieved, as in [AAF$^+$93, AAPW94]. Based on experimental results with analogous uni-cast algorithms in [GKR94], we expect our work to be of significant practical value.

Our algorithm works as follows. A cost is associated with each network edge, that grows with utilization of each link. Subscription request will be connected to the closest node that has issued previous request, along the shortest path based on the above edge costs. Thus, in an unloaded network, we simply use min-hop strategy. As the link congestion grows, the cost of that link "sky-rockets", so that in the highly loaded network, we use min-max strategy.

In between, a certain "pricing function" Ψ is used to trade off the number of hops with the bottleneck link congestion. For example, a constant function reduces the min-hop strategy and a very fast growing function reduces to the min-max strategy. We examine the set of all possible pricing functions Ψ, and establish the sufficient conditions on such a function that make the resulting algorithm competitive. As a corollary, we deduce that the exponential function $\Psi(x) = e^{\alpha x}$ (α is a constant) and delay function $\Psi(x) = 1/(c - x)$ (c is a constant related to edge capacity) previously used in PARIS/plaNET network [CGG91, CGG$^+$93], both work for our purposes.

It is worth mentioning that our scheme can be easily extended for the case in which group communication is restricted to a sub-network (e.g. some links will admit one type of traffic, some will admit all, etc.). In this case, the shortest paths would be restricted to appropriate sets of nodes and edges.

1.5 Other settings considered in literature

In [VG93], the existence of multiple multi-cast groups is modeled by introducing the notion of link cost, and considering single group decision making when all the users subscribe simultaneously. In [BFC93], different (TCP/IP packet-switched) network environments is assumed, without explicit real time guarantees. More results about statistical models can be found in [BFC93, CESZ91, Raj92].

An alternative measure of network performance is the amortized throughput defined as the average over time of the number of bits transmitted by the accepted connections. In this setting, the network's bandwidth is assumed to be insufficient to satisfy all the requests so some of the requests may need to be rejected upon their arrival. An online algorithm in this setting is a combination of a decision mechanism that determines which requests to satisfy together with a strategy that specifies how to route these requests. The goal is to maximize the amortized throughput.

Competitive algorithms to maximize the throughput were provided by Garay and Gopal [GG92] (for the case of a single link); by Garay, Gopal, Kutten, Mansour and Yung [GGK$^+$93] (for a line network); and by Awerbuch, Azar and Plotkin [AAP93] (for general network topologies). The latter work also provided $O(\log n)$ competitive algorithm for the multicast setting, but only when all the users for a specific multicast are known at the time that the group formed and when all the group has to be accepted or all the group has to be rejected. Some results about selective multicast in the throughput model can be found in [AAG94].

2 The model

To simplify, we are considering here the case of *permanent* subscription, or, equivalently, case in which all subscriptions are for identical slots of time, say, 1/2 hour slot for ABC news from 7 till 7.30 p.m., and the customers can tune in at any time, even in the middle of the broadcast. We comment that our algorithms can be easily extended to the case in which different broadcast groups subscribe for different time slots, e.g. there is ABC news slot from 7 till 7.30 and MTV slot from 7 till 8. In the rest of this paper, we only deal with the case of permanent subscription, and delay the generalizations to the final version of this paper.

Formally, consider an (arbitrary) undirected graph $G(V, E, cap)$ with (arbitrary) capacity $cap(e)$ assigned to each edge $e \in E$. We assume a collection of K different broadcast *groups*, with rates $Rate_i$, $1 \le i \le K$ assigned to each group, and with a different source s_i for each group. In fact, the algorithm can be easily adapted to the case in which groups can be initiated in an online manner.

The "subscription requests" consist of pairs (i, v) with the meaning that node $v \in V$ needs to subscribe to broadcast group i, $1 \le i \le K$. We consider the most general case where requests of different groups can be interleaved in an arbitrary way.

Algorithm's *output* is the collection of trees $\mathcal{P} = \bigcup_i \mathcal{P}_i$, where $\mathcal{P}_i$ is a tree consisting of the collection of paths currently existing to route traffic of group i, that must span s_i as well as all nodes $\mathcal{S}_i$ that requested to join the group until the current time.

The output is updated incrementally, i.e. the algorithm responds to subscription request (i, v) by adding to tree $\mathcal{P}_i$ a path P connecting node v to any one of nodes in $\mathcal{P}_i$ and reserving bandwidth of $Rate_i$ along all edges on that path. Note that we somewhat abuse notation and treat $\mathcal{P}_i$ as a tree and as a collection of paths that created the tree.

The algorithm's decision are centralized, i.e. based on global knowledge, in the sense that any subscription request is granted with full knowledge of the previous requests, and the current network utilization, i.e. amount of committed bandwidth on each network link. This can be implemented by either making all decisions at some "center" node, or by broadcasting all routing decisions thru-out the network.

The algorithm's goal is to minimize the "bottleneck link" utilization, i.e.

$$load(\mathcal{P}) = \max_{e \in E} \frac{1}{cap(e)} \sum_{i : e \in \mathcal{P}_i} Rate_i$$

In the competitive framework, the goal is to achieve the best (smallest) "competitive ratio", namely, the maximum (supremum), over all possible sequences, of the ratio of the bottleneck load for offline and online algorithms:

$$\frac{load(\mathcal{P})}{load(\mathcal{R})}$$

where the set of trees $\mathcal{R} = \bigcup_i \mathcal{R}_i$ is chosen by offline (optimum prescient) algorithm for the *same* sequence of requests.

3 Routing Algorithm

3.1 Notations

We denote the "normalized" contribution of i's group to "utilization" of link e as

$$util_i(e) = \frac{Rate_i}{cap(e)}$$

For convenience, we assume that, a certain parameter, $Util(\mathcal{Q})$ is known to the online algorithm. This parameter is an upper bound on the load of a best restricted off-line algorithm, which will be defined later. We can easily dispense of this assumption by standard doubling technique: starting with some lower bound and then doubling this value when necessary. In the future, we will normalize all traffic by $Util(\mathcal{Q})$. In particular, we will denote

$$\rho_i(e) = \frac{util_i(e)}{Util(\mathcal{Q})}$$

Also denote

$$\ell(e) = \sum_{i:e\in\mathcal{P}_i} \rho_i(e) \tag{1}$$

and for the load induced by an algorithm $\mathcal{A}$

$$\ell(\mathcal{A}) = \max_{e\in E} \rho_i(e) \tag{2}$$

3.2 The algorithm

To serve a request (i, v) the algorithm finds a (weighted) shortest path from v to the tree $\mathcal{P}_i$ in the weighted graph $(G, V, weight)$ with weights

$$weight(e) = \Delta\Psi_e = \Psi(\ell(e) + \rho_i(e)) - \Psi(\ell(e)).$$

Here, $\Psi(\ell)$ is any positive infinitely differentiable function, with all of its derivatives defined for $\ell \geq 0$, and positive in that range.

Also define

$$\gamma \overset{def}{=} \max_{0\leq\ell} \left(\frac{\Psi^{(k)}(\ell)}{\Psi(\ell)}\right)^{1/k} \tag{3}$$

We also require that $\gamma < \ln 2$. In particular, for a given γ, we choose the "fastest growing" function with the derivative property, namely

$$\Psi(\ell) = e^{\gamma\ell}.$$

The algorithm is formally presented in Figure 1. The main theorem, which we prove, is the following.

Theorem 3.1 The on-line algorithm is $O(\log n \log d)$ competitive.

```
SUBSCRIBE($v, i$)                                                          /* subscription request */
  $\forall e \in E:\ \Delta\Psi_e = \Psi(\ell(e) + \rho_i(e)) - \Psi(\ell(e))$          /* compute weights */
  $P \leftarrow$ shortest path from $v$ to closest node in $\mathcal{P}_i$ in weighted graph $(V, E, \Delta\Psi)$
  route group $i$ to $v$ along $P$
  $\mathcal{P}_i \leftarrow \mathcal{P}_i \bigcup P$
  $\forall e \in P, \ell(e) \leftarrow \ell(e) + \rho_i(e)$                                   /* load increase update*/
```

Figure 1: The SHORTEST-WEIGHTED-PATH Algorithm.

4 Competitive analysis

4.1 Structure of the proof

The on-line algorithm has to maintain for each i a connected subgraph (a tree) $\mathcal{P}_i$ that spans the current vertices that belong to group i with a reserved bandwidth $Rate_i$ on each of its edges. Since the decisions are made in an on-line fashion, $\mathcal{P}_i$ evolve in time such that edges are only added to the tree and none of the edges are deleted. For a request (i, v) we denote by $P_{i,v}$ the path that the on-line augmented to the tree in order to connect vertex v to the previous tree of group i. Let $\mathcal{S}_i$ be the set of vertices that join group i up to the current time. Clearly, in this way the collection of paths $\mathcal{P}_i$ defines the broadcast tree $\mathcal{P}_i$: $\mathcal{P}_i = \bigcup\limits_{v \in \mathcal{S}_i} P_{i,v}$.

The off-line knows the whole sequences of requests in advance and thus can construct a tree $\mathcal{R}_i = \bigcup\limits_{v \in \mathcal{S}_i} R_{i,v}$ that spans all the vertices of group i in the sequence such that each edge of that tree has bandwidth $Rate_i$ reserved for this group.

Observe that, in principle, both the online and the optimal offline solution may use "relay" points to serve subscription request (i, v), e.g. path $R_{i,v}$ from requesting vertex v (generated by optimal offline) may end in the middle of existing communication path. (See Figure 2.)

For the purpose of the proof, we define the notion of *restricted* offline. The restricted offline must connect v via path $Q_{i,v}$ of bandwidth $Rate_i$ to one of the (previously subscribed) vertices ($\mathcal{S}_i$), rather than to an intermediate "relay point" on one of the existing paths. If this path intersects other paths of optimal offline, restricted offline is not allowed to merge these paths together. Also, restricted offline is only allowed to use such paths $Q_{i,v}$ that belong to the spanning tree R_i of the optimal offline. The essence of the restriction is that hooking into the middle of existing communication paths is disallowed.

Note that such an off-line may need to use the same edge of $\mathcal{R}_i$ several times. In this case $\mathcal{Q}_i = \bigcup\limits_{v \in \mathcal{S}_i} Q_{i,v}$ where $\mathcal{Q}_i$ is a "generalized tree", i.e. tree such that each edge has some multiplicity. Thus the load on an edge increases by the rate of the tree multiple times. In Figure 3 we show output of restricted offline, which corresponds one of the multicast groups of optimal offline in Figure 2.

In fact, our proof of competitiveness will assume that online is operating under analo-

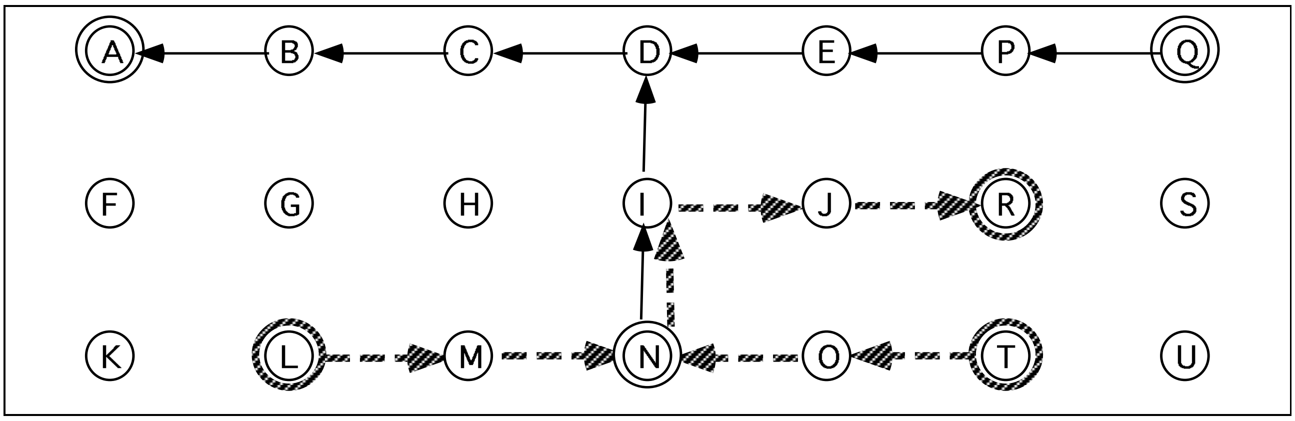

Figure 2: The collection of trees tree $\mathcal{R} = \bigcup_{i=1,2} \mathcal{R}_i$ constructed by optimal offline for two different multicast groups. The two different spanning trees for each group are represented by solid and dashed arrows, accordingly. The underlying topology is a grid. The congestion is caused by overlap between trees of different groups, e.g. on link from I to N. Double-circles represent subscription requests, e.g., N and Q are locations of requests for first group, with source at A, and L, T are locations of requests for second group, with source at R. Single circles represent relay points. Note that optimal offline connects request from N to intermediate relay point (D) on existing communication path from A to Q.

gous "no-relay-points" restriction, and must maintain separately intersecting communication paths, thus paying higher load penalty. (This only strengthens our result.) In Figure 4 we show output of restricted offline, which corresponds to the same subscription request sequence as in Figure 2.

The proof of performance of the algorithm is divided into two parts.

- We show that, for any subscription pattern,

$$\frac{\ell(\mathcal{P})}{\ell(\mathcal{Q})} = O(\log n) \tag{4}$$

  i.e. the on-line algorithm is $O(\log n)$ competitive to the restricted off-line, where n is the number of network nodes.

- We show that, for any sub-scription pattern,

$$\frac{\ell(\mathcal{Q})}{\ell(\mathcal{R})} = O(\log d) \tag{5}$$

  i.e. the restricted off-line generates a load which is *at most* $\log d$ times the optimal off-line, where d is the maximum number of users in any given broadcast group.

Combining these two parts we get the desired result:

$$\frac{\ell(\mathcal{P})}{\ell(\mathcal{R})} = O(\log n \log d) \tag{6}$$

i.e. our online algorithm in Figure 1 is $O(\log n \log d)$ competitive.

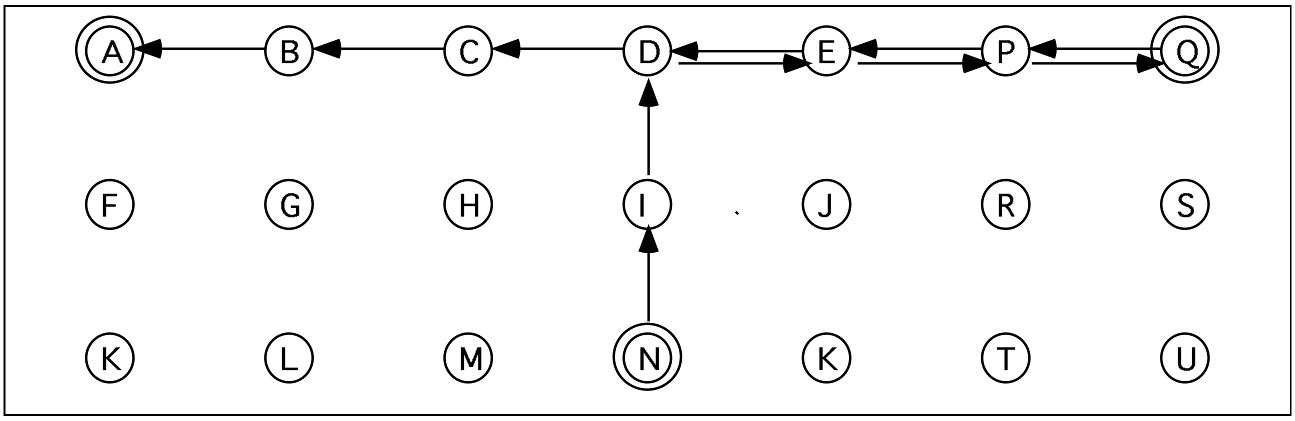

Figure 3: The tree $\mathcal{Q}_1$ constructed by restricted offline. Uses paths belonging to tree $\mathcal{R}_1$ of optimal offline. Now, request from N cannot be accommodated by connecting to D, since D was not a previously requested vertex, and thus connection is made thru Q. Paths from N to Q and from A to Q intersect, yet they cannot be merged together, thus leading to twice higher load.

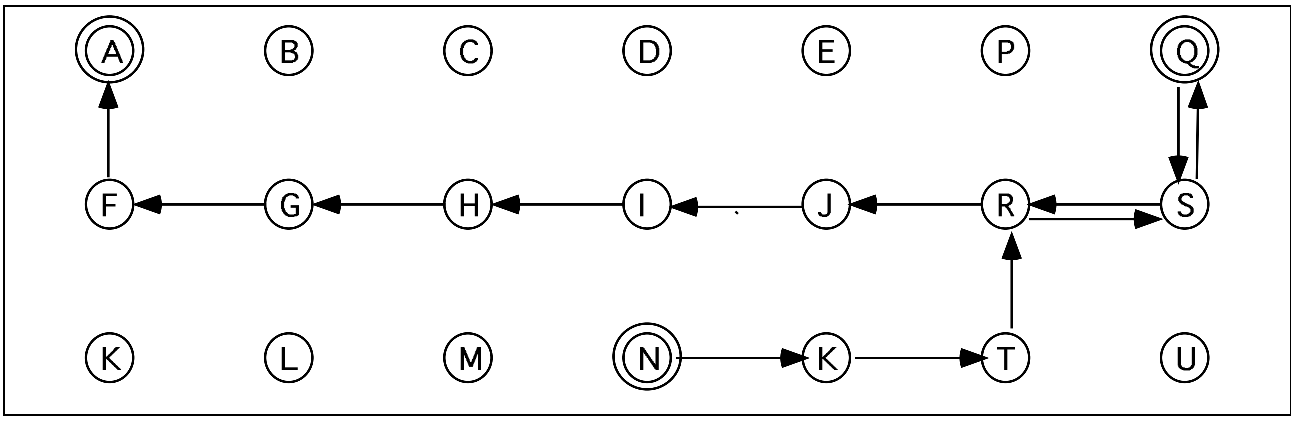

Figure 4: The tree $\mathcal{P}_1$ constructed by restricted offline that is completely different from optimal offline tree $\mathcal{R}_1$. Note that the restricted offline cannot use any relay points, either. Thus, edge multiplicity is possible, e.g., edge from S to Q is counted twice.

4.2 Online is competitive with Restricted offline

Theorem 4.1 The on-line algorithm is $O(\log n)$ competitive to the restricted off-line.

The proof uses the methods developed in [AAF$^+$93, AAPW94] for point to point connection. We present here a proof which is an extension of the proof in [AAPW94].

We start first with the following useful fact.

Fact 4.2 For all $0 \leq \delta \leq 1$

$$\Psi(x+\delta) - \Psi(x) \leq \Psi(x)\delta(e^{\gamma} - 1)$$

Proof: Taking advantage of Equation (3), and expanding $\Psi(x+\delta)$ into Taylor series around x, we get

$$
\begin{aligned}
\Psi(x+\delta)-\Psi(x) &= \sum_{k=1}^{\infty}\Psi^{(k)}(x)\frac{\delta^k}{k!} && (7)\\
&\le \sum_{k=1}^{\infty}\Psi(x)\frac{\delta^k\gamma^k}{k!} && (8)\\
&\le \Psi(x)\delta\sum_{k=1}^{\infty}\frac{\gamma^k}{k!} && (9)\\
&\le \Psi(x)\delta(e^{\gamma}-1) && (10)
\end{aligned}
$$

∎

We now show

Lemma 4.3 At any time
$$\sum_{e\in E}\Psi(\ell(e)) \le |E|/(2-e^{\gamma}).$$

Proof: Let $h_P(e)$ be the load on edge e at the arrival time of vertex v. The on-line algorithm connects v to its group i through a path P. The restricted off-line connects v to a vertex in the current $\mathcal{S}_i$ by a path Q.

In order to connect v to the current $\mathcal{P}_i$, the on-line algorithm uses P the shortest path with respect to the weight $\Delta\Psi$ at the time of arrival of v. The restricted off-line connects v to some vertex in the current $\mathcal{S}_i$. All these vertices are also on the tree $\mathcal{P}_i$ of the on-line. Therefore

$$
\begin{aligned}
\sum_{e\in P}(\Psi(h_P(e)+\rho_i(e))-\Psi(h_P(e))) &\le \sum_{e\in Q}(\Psi(h_P(e)+\rho_i(e))-\Psi(h_P(e)))\\
&\le \sum_{e\in Q}(e^{\gamma}-1)\Psi(h_P(e))\rho_i(e)\\
&\le \sum_{e\in Q}(e^{\gamma}-1)\Psi(\ell(e))\rho_i(e)
\end{aligned}
$$

The second inequality above follows from Fact 4.2 with $\delta=\rho_i(e)$, and $h_P(e)$. Note that each $e\in Q$ we have $0\le\rho_i(e)\le 1$ since the restricted offline algorithm connects the i'th request through Q.
Summing over all subscription requests of all groups yields

$$\sum_i\sum_{P\in\mathcal{P}_i}\sum_{e\in P}(\Psi(h_P(e)+\rho_i(e))-\Psi(h_P(e))) \le (e^{\gamma}-1)\sum_i\sum_{Q\in\mathcal{Q}_i}\sum_{e\in Q}\Psi(\ell(e))\rho_i(e) \qquad (11)$$

BY exchanging the order of summation we get

$$\sum_{e\in E}\sum_{i,e\in\mathcal{P}_i}(\Psi(h_P(e)+\rho_i(e))-\Psi(h_P(e))) \le (e^{\gamma}-1)\sum_{e\in E}\Psi(\ell(e))\sum_{i,e\in\mathcal{Q}_i}\rho_i(e)$$

Clearly, the sum of left hand-side is a telescopic sum for each edge e. Observe that the fact that the normalized load of the restricted offline algorithm never exceeds 1, implies that

$$\sum_{i,e\in\mathcal{Q}_i} \rho_i(e) \leq 1.$$

Thus we conclude that

$$\sum_{e\in E}(\Psi(\ell(e)) - \Psi(0)) \leq (e^{\gamma} - 1)\sum_{e\in E}\Psi(\ell(e))$$

Using the fact that $e^{\gamma} < 2$, we get

$$\sum_{e\in E}\Psi(\ell(e)) \leq |E|\Psi(0)/(2 - e^{\gamma}).$$

▮

The above lemma immediately implies that

$$\ell(\mathcal{P}) = O(\Psi^{-1}(|E|\Psi(0)/(2 - e^{\gamma})))$$

For our choice of $\Psi(\ell) = e^{\gamma\ell}$, we get

$$\ell(\mathcal{P}) = O(\log|E|).$$

Now, recall that ℓ is the normalized load of online, and thus

$$load(\mathcal{P}) = O(load(\mathcal{Q})\log|E|) = O(load(\mathcal{Q})\log n)$$

which implies the Theorem.

▮

4.3 Restricted offline is competitive with optimal offline

Next we relate the load of the restricted offline, that is only allowed to connect to previously connected vertices, to the optimal offline, that may connect to "relay points" of existing communication paths, and not just to the terminal points.

We will denote by $\mathcal{R}$ the tree computed by optimal offline and by $\mathcal{Q}$ the tree computed by restricted offline.

Theorem 4.4 At any time

$$\ell(\mathcal{Q}) \leq 2\ell(\mathcal{R})\log d$$

Proof: We construct a restricted off-line solution such that for serving group i uses only edges of $\mathcal{R}_i$ and each such edge is used at most $2\log d$ times. Recall that the total load on an edge is just the sum of the bandwidth of all the trees which use this edge and thus the theorem will follow from the above construction.

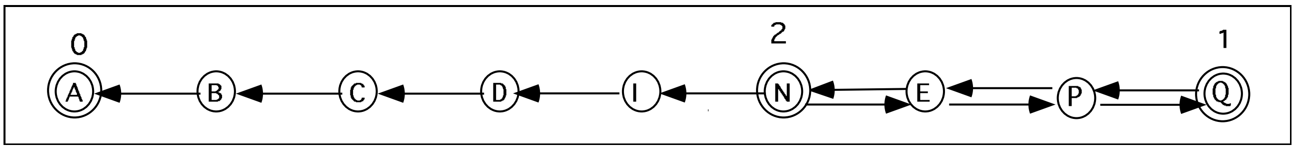

Figure 5: Generating the line graph for the restricted offline. Vertices are numbered (0,1,2) according to the order of their appearance, and then laid out from left to right, in the increasing order of DFS numbers in the induced line graph. The arrows represents the connections made by restricted offline. Notice that the path from N to Q is shorter than the path from N to A is this line graph, and thus the former path is chosen. The segments representing the connections from N to Q, and from A to Q, overlap. We prove that total overlap is at most logarithmic in the size of a group.

Given a tree $T = \mathcal{R}_i$ rooted at v_0 with vertices $v_1, v_2, \ldots v_l$, $l \leq d$ which joined the group in this order, we first traverse the tree T in a DFS fashion. This defines an ordered list $v_{i_0}(= v_0), v_{i_1}, \ldots, v_{i_l}$ of the vertices. We add to the end of this list $v_{i_{l+1}} = v_0$. We build a line which consists of the vertices in the order of their DFS numbering. That is the DFS numbers grow as we scan this line from left to right. Figure 5 illustrates this graphically for the restricted offline in Figure 3 corresponding to optimal offline in Figure 2.

The restricted offline needs to connect each vertex v_j to some vertex $v_{j''}$ where $j'' < j$. To do so the restricted off-line finds the nearest neighbor among $v_{j''}$ in the line to its right and to its left. Note that v_0 appears both as the first and the last element in the list. Thus each vertex has always nearest neighbor to its right and to its left. Denote by $v_{j'}$ the nearest element to v_j among these two (ties are broken arbitrary).

We emphasize that the list is unweighted i.e., the distances between any two consecutive elements is defined to be 1. Then, if $v_{j'}$ is to the right of v_j it connects v_j to $v_{j'}$ by the partial path of the DFS from v_j to $v_{j'}$ and otherwise by the partial path of the DFS from $v_{j'}$to v_j. We call this path the *augmenting path* of vertex v_j.

Clearly, the paths constructed is not necessarily simple but it must contain the simple unique path between v_j and $v_{j'}$.

The proof of the Theorem immediately follows from the following

Claim 4.5 The above construction uses each edge at most $\log d$ times in each direction.

Proof: (of the claim): The DFS defines a path in the original graph between any two consecutive nodes in the line. The union of these paths covers each edge of the tree precisely once in each direction (by the definition of DFS).

Consider an edge e of the original graph in one direction. In the DFS traversal the edge is between some consecutive vertices of the line v_{i_j} to $v_{i_{j+1}}$. Each augmenting path to the tree corresponds to a segment in the line such that any two segments are either disjoint (excluding endpoints) or one is contained in the other. Clearly, in the latter case the augmenting path of the vertex with the higher index (i.e the one which arrived later) is contained in the augmenting path of the vertex with the smaller index. Moreover, given two consecutive vertices in the line, every segment that contains both of them has (unweighted) length (in the line) of at most half the previous segment. Otherwise, the augmenting path of the later vertex would have been connected to the other endpoint of the earlier segment. Thus, there are at most $\log d$ segments which contain both v_i and v_{i+1}. Since these are precisely all the segments that contains e the proof of the claimed is completed. As we already mentioned that completes the proof of the Theorem. ▮

References

[AAF+93] Jim Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 623–631, May 1993.

[AAG94] B. Awerbuch, Y. Azar, and R. Gawlick. Dense trees and competitive selective multicast. Unpublished manuscript, 1994.

[AAP93] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive on-line routing. In *focs34*, November 1993.

[AAPW94] Baruch Awerbuch, Yossi Azar, Serge Plotkin, and Orli Waarts. Competitive routing of virtual circuits with unknown duration. In *Proc. 5'th ACM-SIAM Symp. on Discrete Algorithms*, pages 321–327, 1994.

[ABK92] Yossi Azar, Andrei Broder, and Anna Karlin. On-line load balancing. In *Proc. 33rd IEEE Symp. on Found. of Comp. Science*, pages 218–225, October 1992.

[ACG+90] Baruch Awerbuch, Israel Cidon, Inder Gopal, Marc Kaplan, and Shay Kutten. Distributed control for PARIS. In *Proc. 9th ACM Symp. on Principles of Distrib. Computing*, pages 145–160, 1990.

[AKP+93] Yossi Azar, B. Kalyanasundaram, Serge Plotkin, K. Pruhs, and Orli Waarts. On-line load balancing of temporary tasks. In *Proc. Workshop on Algorithms and Data Structures*, pages 119–130, August 1993.

[ANR92] Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignment. In *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms*, pages 203–210, 1992.

[BFC93] Tony Ballardi, Paul Francis, and Jon Crowcroft. Core based trees (cbt). In *Proc. of the Annual ACM SIGCOMM Symposium San Fransisco, CA*, pages 85–95, September 1993.

[Bou92] J.Y. Le Boudec. The asynchronous transfer mode: a tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.

[CESZ91] Ron Cocchi, Deborah Estrin, Scott Shenker, and Lixia Zhang. A study of priority pricing in multiple service class networks. In *Proc. of the Annual ACM SIGCOMM Symposium on Communication Architectures and Protocols, Zurich, Switzerland*, 1991.

[CG88] I. Cidon and I. S. Gopal. PARIS: An approach to integrated high-speed private networks. *International Journal of Digital & Analog Cabled Systems*, 1(2):77–86, April-June 1988.

[CGG91] I. Cidon, I. Gopal, and R. Guérin. Bandwidth management and congestion control in plaNET. *IEEE Commun. Mag.*, 29(10):54–63, October 1991.

[CGG$^+$93] I. Cidon, I. Gopal, P. M. Gopal, R. Guérin, J. Janniello, and M. Kaplan. The plaNET/ORBIT high speed network. *Journal of High Speed Networks*, 2(3):1–38, September 1993.

[GG92] J.A. Garay and I.S. Gopal. Call preemption in communication networks. In *Proceedings of INFOCOM '92*, volume 44, pages 1043–1050, Florence, Italy, 1992.

[GGK$^+$93] Juan Garay, Inder Gopal, Shay Kutten, Yishay Mansour, and Moti Yung. Efficient on-line call control algorithms. In *Proceedings of 2'nd Annual Israel Conference on Theory of Computing and Systems*, 1993.

[GKR94] Rainer Gawlick, Charles Kamanek, and K.G. Ramakrishnan. On-line routing for virtual private networks. unpublished manuscript, February 1994.

[Kel86] F.P. Kelly. Blocking probabilities in large circuit-switched networks. *Advances in Applied Probablity*, 18:473–505, 1986.

[Raj92] Bala Rajagopalan. Reliability and scaling issues in multicast communication. In *Proc. of the Annual ACM SIGCOMM Symposium on Communication Architectures and Protocols, Baltimore, MD*, pages 188–197, August 1992.

[VG93] Dinesh C. Verma and P.M. Gopal. Routing reserved bandwidth multi-point connections. In *Proc. of the Annual ACM SIGCOMM Symposium San Fransisco, CA*, pages 96–105, September 1993.

[XVI88] CCITT SG XVIII. Special issue on asynchronous transfer mode. *International Journal of Digital & Analog Cabled Systems*, 1(4), 1988.