which implies the lemma. Let T' be an optimal broadcast schedule for for n processors that uses only q_k. Certainly $height(T) \leq height(T')$. However, $height(T') = O(poly(\lg n))$ [3, 9]. ■

Proof sketch of Lemma 3.7: The conditions of the Lemma attempt to characterize a schedule T for inputs S, B, n that ends within time H. We first explain the meaning of the conditions in the Lemma. In general, these conditions assert that each level (time point) in the schedule is consistent, based on the consistency of previous levels. This fact, together with a consistent starting/ending level, and a total time/budget as required, is enough to assert the existence of the schedule.

Now, n_t represents the number of new nodes at time t, namely these nodes that correspond to processors that have received the message at time t. These nodes are available to immediately start transmitting the message to other nodes. o_t represents the number of available old nodes at time t; namely, these nodes that have received the message at time $t - i \times \ell$ for some i (we assume here that such nodes transmit whenever possible, that is: every ℓ units of time). There are other nodes that might be in the middle of the setup process for their next message, but these are not involved in any action at time t, and will not be accounted for. e_t^i represents the number of services of type i that are used at time t. Given this, Conditions 1-4 assert the consistency of the schedule tree: Condition 1 is the initial situation: there is a single node which is available at time $t = 0$. Condition 2 relates the edges to the nodes; it asserts that with every service e_t^i there is a node (at the end of this edge) at time $t + q_i$. Condition 3 states that the available old nodes at time t are exactly the old and new nodes of time $t - \ell$ (for which the setup time is just over). Condition 4 relates the available nodes at time t to services used at time t, namely, available nodes to send messages at time t are exactly the old and new nodes at that time.

Conditions 5 and 6 assert the resource bounds of the schedule: Condition 5 asserts that the schedule is a broadcast tree for n nodes; Condition 6 asserts that the total cost of the schedule is bounded by B.

The formal proof is omitted here for lack of space.

4. Conclusion

Many "shopping problems" can be formalized in the context of the model presented in this paper. In the context of communication services, we gave an algorithm which computes an optimal broadcast schedule.

There are still many related open questions. In particular the complexity of the broadcast decision problem is not completely settled. A different direction is to enrich the model by introducing probabilities of success, dealing with randomized strategies, varying service loads, and other real-world parameters. This direction will result in a "multi-(parameter,cost)" optimization problems.

References

[1] A. Bar-Noy, J. Bruck, C. Ho, S. Kipnis, and B. Schieber. Computing global combine operations in the multi-port postal model. *IEEE Transactions on Parallel and Distributed Systems*, 6(8):896–900, August 1995.

[2] A. Bar-Noy and S. Kipnis. Multiple message broadcasting in the postal model. In *Proc. 7th International Parallel Processing Symposium*, pages 463–470. IEEE, April 1993.

[3] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. *Mathematical Systems Theory*, 27(5):431–452, 1994.

[4] A. Bar-Noy, S. Kipnis, and B. Schieber. Optimal computation of census functions in the postal model. *Discrete Applied Mathematics*, 58:213–222, 1995.

[5] J. Bruck, R. Cypher, and C. Ho. Multiple message broadcasting with generalized Fibonacci trees. In *Proc. 4th Symposium on Parallel and Distributed Processing*, pages 424–431. IEEE, December 1992.

[6] J. Bruck and C. Ho. Efficient global combine operations in multi-port message-passing systems. *Parallel Processing Letters*, 3(4):335–346, December 1993.

[7] J. Bruck, C. Ho, S. Kipnis, and D. Weathersby. Efficient algorithms for all-to-all communications in multi-port message-passing systems. In *Proc. 6th Annual Symposium on Parallel Algorithms and Architectures*, pages 298–309. ACM, June 1994. To appear in *IEEE Transactions on Parallel and Distributed Systems*.

[8] S.-N. Choi and M. Golin. Lopsided trees: Algorithms, Analyses and Applications. In *Proc. of the 23rd Intl. Colloquium on Automata Languages and Programming*, 1996.

[9] M. Golin and A. Schuster. Optimal Point-to-Point Broadcast Algorithms via Lopsided Trees. In *Proc. 5th Israeli Symp. on Theory of Computing and Systems*, pages 63–73, 1997. To appear in Disc. Appl. Math.

[10] O. Etzioni and S. Hanks and T. Jiang and R.M. Karp and O. Madani and O. Waarts. Efficient Information Gathering on the Internet. In *Proc. 37th Symp. on Foundations of Computer Science*, pages 234–243, 1996.

Claim 3.4 *For input $S = \{(c_i, q_i)|i = 1, ..., k\}$,$l$, $n \geq 1$ and t, $1 \leq t \leq q_k + (n-1)l$, Algorithm $A2$ computes $B(t, n)$ in $O(k^2 n^3 l \lg q_k)$ steps.* ■

We note here that the dependency in l is not polynomial in the length of its description - this is indeed unsettled here, although for practical purposes l should be taken as fairly small, usually $O(1)$ (see also Sec. 3.3).

3.3. Complexity issues

We have shown that for input $S = \{(c_i, q_i)|i = 1, ..., k\}$ representing the services, l representing the setup time, and integers B, n, representing a total budget and number of targets respectively, algorithm $A2$ takes $O(k^2 n^3 l \lg q_k)$ steps. This time bound is polynomial in the input length only if n is given in unary and $l = O(max(n, \lg q_k))$. We can't expect a polynomial algorithm whose dependency in n is $poly(\lg n)$ as the output size (the broadcast tree) is of size $\theta(n)$. However, the following is an associated decision problem whose complexity is unsettled.

The broadcast decision problem: Given a set S of k services, budget B, a set-up delay l, the number of processors n and the deadline H, is there a broadcast schedule T on n nodes with $height(T) \leq H, B(T) \leq B$ (i.e., can we broadcast to n processors within time H and budget B?) Corresponding optimization problems would be to compute the functions $B(), H(), N()$.

We assume that $l \leq q_1$ as we take the setup time to be included in the delay and may always assume that $q_i \leq H$ and $c_i < B$, for all i (otherwise the ith service cannot be used and so can be ignored). Hence the length of the input is $poly(k, \lg H, \lg B, \lg n)$ and algorithm $A2$ does not provide a poly-time algorithm. Few remarks are due here.

- For $k = 1$ (no optimization is needed) there is a poly-time algorithm for the broadcast decision problem [8, 9].
- For $k > 1$ and general input service set S we do not know any upper bounds for the complexity of the broadcast decision problem except exponential-time even for $k = 2$.

For the restricted problem in which S is such that $q_k = O(poly(\lg n))$ the broadcast decision problem is in NP as stated by the following theorem:

Theorem 3.5 *Let $S = \{(c_i, q_i)||i = 1, \ldots, k\}$ be a set of services, l the setup time, and B, n, H, representing a total budget, number of targets, and deadline respectively. Assume that $l \leq q_k = O(poly(\lg n))$. Then the broadcast decision problem is in NP.*

The intuition behind the proof is the following: We first show that if the services time is relatively small ($q_k = O(poly(\lg n))$) then there is a shallow broadcast tree (this is shown in Lemma 3.6). Next we show that any shallow broadcast tree can be witnessed (and verified) by examining a relatively small amount of information, namely proportional to the tree depth rather than to the tree size (this is shown in Lemma 3.7). The reason for this is that to witness a schedule, the tree structure is of no importance - the only important data is how many services of each type are being used at each point in time. However, this data is proportional to the tree depth rather than to its size.

We now present the formal proof of the theorem.

Proof: The proof will follow from the Lemmata below.

Lemma 3.6 *Let $S = \{(c_i, q_i)||i = 1, \ldots, k\}, q_k = O(poly(\lg n))$. Then, $H(n, B, S) = O(poly(\lg n))$.*

Lemma 3.7 *Let $S = \{(c_i, q_i)||i = 1, \ldots, k\}$, and l, B, n, H as before. Then a broadcast tree T: $\|T\| = n, B(T) \leq B, height(T) \leq H$, exists iff there are k sequences of non-negative integers $\{e_t^i\}_{t=0}^{t=H}$, $i = 1, ..., k$ (representing the number of edges (services) of type i at time t), two sequences $\{n_t\}_{t=0}^{t=H}$ and $\{o_t\}_{t=0}^{t=H}$ (that represent the number of new processors and available processors at time t) that satisfy the following conditions:*

1. $n_0 = 0, \ o_0 = 1$ - *initial conditions.*
2. $n_t = \Sigma_{i=1}^{i=k} e_{t-q_i}^i$ - *new nodes at time t.*
3. $o_t = o_{t-l} + n_{t-l}$ - *nodes become available again after l steps of setup time.*
4. $\Sigma_{i=1}^{i=k} e_t^i \leq n_t + o_t$ - *number of transmissions at time t.*
5. $1 + \Sigma n_i = n$ - *total number of processors.*
6. $\Sigma_{t=0}^{t=H-1} \Sigma_{i=1}^{i=k} c_i e_t^i \leq B$ - *total budget.*

The theorem follows, as from the assumptions of Lemma 3.6 there is a valid broadcast schedule T of time $H' = O(poly(\lg n))$. A Non-deterministic procedure to verify the existence of such schedule T is to guess H', k sequences $\{e_t^i\}_{t=0}^{t=H'}$, $i = 1, ..., k$ (representing the number of edges (services) of type i at time t), two sequences $\{n_t\}_{t=0}^{t=H'}$ and $\{o_t\}_{t=0}^{t=H'}$ (that represent the number of new processors and available processors at time t) and to verify that the conditions of Lemma 3.7 are met (which can be done in $O(poly(\lg n, k, \lg B))$ time). ■

We now proceed to prove the Lemmata.

Proof (of Lemma 3.6): Given $S = \{(c_i, q_i)||i = 1, \ldots, k\}$, n, and a budget B. Let T be an optimal broadcast schedule; we prove that $height(T) = poly(\lg n)$

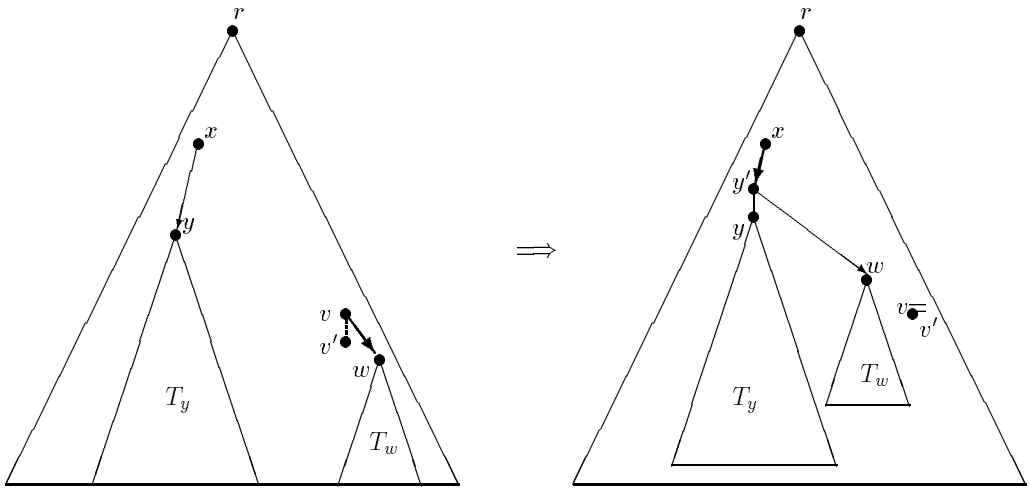

Figure 3. Switching fast and slow edges for Claim 2.2.

3.2. General Broadcast

In this section we present algorithms to compute optimal schedules for the broadcast on a budget and broadcast with a deadline. Let $|S| = k$, i.e. there are k services available. Since S remains fixed throughout this section, for the sake of simplicity we omit it from all the notations.

Clearly, the cost of the cheapest possible broadcast is $c_k n$ (by using only the kth –cheapest– service), and this can be done in time $H(n, q_k)$; the fastest broadcast can be done in $H(n, q_1)$ using only the fastest service.

We now present our first algorithm, $A1$, that computes $B(t, n)$ and the corresponding schedule in $O(kn^2(nl+q_k))$ steps. This time is polynomial in n, k (this is expected since the output size is $\Omega(n)$). However, it is proportional to q_k (and $l \leq q_k$) which is exponentially larger than its description. We then show how to improve algorithm $A1$ to become polynomial in the description of q_k too.

The algorithms use dynamic programming based on the following claim.

Claim 3.2 *For every $m \geq 0$ and every t,*

$$B(t,m) = \infty \text{ for } t < H(m, q_1)$$

$$B(t,0) = 0 \text{ for } t \geq 0 = H(0, q_1)$$

$$B(t,m) = \min_{1 \leq i \leq k} \min_{1 \leq r \leq m-1} c_i + B(t - q_i, r-1) + B(t-l, m-r) \quad (1)$$

Proof: The first two equalities are by convention. For the third, any broadcast to m processors must start with sending a message using one of the k services, after which the receiving processor will broadcast to some other $r-1$ processors. In the meantime, the origin is free to take care of the remaining $m - r$ processors, starting at the next time step. ■

Algorithm $A1$.

for $m = 1, \ldots, n$
  for $t = H(m, q_1), \ldots, H(m, q_k)$
    compute $B(t, m)$ using Equation (1)

The corresponding schedule T can be obtained by straight-forward adjustments of the above algorithm.

Claim 3.3 *Algorithm $A1$ computes $B(t, n)$ in $O(kn^2(nl + q_k) \lg q_k)$ steps.*

Proof: Each application of Equation (1) takes $O(kn \lg q_k)$ steps ($\lg q_k$ is for the numerical computations involving q_i). In order to compute $B(t, m)$ over all the range, no more than $n \cdot (q_k + nl)$ such application are needed. ■

To improve the algorithm complexity to be polynomial in $\lg q_k$ (rather then in q_k) we note that $B(t, m)$ need not be computed for any t in the range $q_{k-1} + (m-1)l < t < q_k$. The reason for this is that when $t < q_k$ the kth service can not be used at all. Thus, the cheapest possible cost will be $c_{k-1}m$, which is obtained for $t = q_{k-1} + (m-1)l$. Namely, $B(t, m) = c_{k-1}m$ for each t in the above range.

A similar argument applies to t in the range $q_i + (m-1)l < t < q_{i+1}$, for each $i = 1, \ldots, k-1$:

$$\text{for each } i = 1, \ldots, k-1 \quad q_i + (m-1)l < t < q_{i+1} \longrightarrow B(t,m) = c_i m \quad (2)$$

We get the following improved algorithm

Algorithm $A2$

for $i = 1, \ldots, k$
  for $t = q_i i, \ldots, q_i i + (n-1)l$
    for $m = 1, \ldots, n$ compute $B(t, m)$
      using Equations (1) and (2).

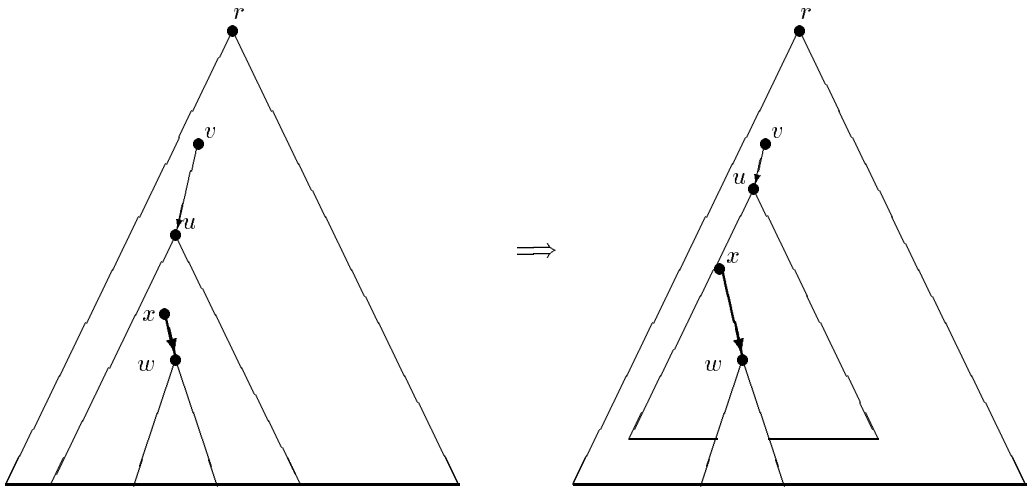

Figure 2. Fast transmission edges should precede the slow ones in any trajectory (see Claim 2.1). Nodes v, u, x, w all are on the same trajectory, in both T and T'.

$t'_y = t_x + q_f + l = t_y - q_s + q_f + l$. Thus, for any z in T_y (but not in T_w, for which the above takes precedence), $t'_z = t_z - q_s + q_f + l$.

Finally, the result of moving v' up l steps is that all its children have their time improved by l (if v' is in T_y this saving is added to the above). ■

So, w.l.o.g. we can assume that all schedules are monotone and service-layered.

3. Algorithms

3.1. Broadcasting with two services

Let $S = \{(c_f, q_f), (c_s, q_s)\}$, $q_s \geq q_f + l$. When the number of services is only two it is easy to compute how much each service should be used for an optimal broadcast on a budget schedule: Let f be the number of expensive transmissions and let s be the number of cheap transmission. Then

$$f \cdot C_f + s \cdot C_s \leq B \quad \text{and} \quad f + s = n.$$

Substituting $s = n - f$ into the inequality we get

$$f \cdot C_f + (n - f) \cdot C_s = f \cdot (C_f - C_s) + n \cdot C_s \leq B$$

so we want the largest integral f such that

$$f \leq \frac{B - n \cdot C_s}{C_f - C_s}$$

and for simplicity, we assume in the sequel that equality holds in the above equation.

Next, we consider a linear algorithm to obtain a nearly optimal schedule. Denote by $H(m, q) = H(m, \{(c, q)\}, mc)$ the optimal broadcast time to m nodes using a single service with the delay q. The optimal single service schedules and their heights are computed in linear time as in [8, 9].

Consider an optimal single service schedule T^f for broadcasting a message to f processors using the fast service only. Its height is $height(T^f) = H(f, q_f)$ and it can be constructed (in linear time) by the algorithm of [8, 9]. Now using the same algorithm construct an optimal schedule T^s of height $height(T^s) = H(\lceil \frac{s}{f+1} \rceil, q_s)$, for broadcasting to $\lceil \frac{s}{f+1} \rceil$ processors using only the slow service.

Construct the final schedule T by appending a copy of T^s to each of $f + 1$ leaf of T^f (so that each leaf of T^f becomes a root of a copy of T^s; some resulting nodes will be unused).

Then $height(T) \leq H(f, q_f) + H(\lceil \frac{s}{f+1} \rceil, q_s)$. We show that an optimal schedule can not be much better:

Claim 3.1 *Let n, S, B, f, s be defined as above. Then $H(n, S, B) > H(f, q_f) + H(\lceil \frac{s}{f+1} \rceil, q_s) - q_s - q_f$.*

Proof: By Claim 2.2 there exists an optimal schedule T such that if $v\vec{w}$ is the latest fast and $x\vec{y}$ the earliest slow message, then $t_v - t_x < q_s$. Also, clearly $t_w \geq H(f, q_f)$, so $t_v \geq H(f, q_f) - q_f$. Combining these we get $t_x > t_v - q_s \geq H(f, q_f) - q_f - q_s$.

W.l.o.g., there are $\lceil \frac{s}{f+1} \rceil$ processors in T_x. So, $height(T_x) \geq H(\lceil \frac{s}{f+1} \rceil, q_s)$, and therefore $height(T) = H(n, S, B) > H(f, q_f) - q_f - q_s + H(\lceil \frac{s}{f+1} \rceil, q_s)$. ■

Thus, schedules within $q_s + q_s$ from optimal can be computed with only two applications of algorithms of [8, 9].

Broadcast on a budget: given n, S and budget B, compute the fastest schedule T to broadcast to n processors and satisfying the budget: $height(T) = H(n, B, S), B(T) \leq B, \|T\| = n$;

Broadcast with a deadline: given n, S and deadline t, compute the cheapest schedule T satisfying the deadline : $B(T) = B(t, n, S), height(T) \leq t, \|T\| = n$;

Broadcast with a deadline and a budget: given services S, budget B and deadline t, compute a schedule T to broadcast to the maximum number of processors $N(t, B, S)$ while satisfying both the deadline and the budget : $\|T\| = N(t, B, S), height(T) \leq t, B(T) \leq B$.

Given an algorithm to solve one of these problems, each of the other two can be solved by binary search.

Let $p_v = p$ and the path in T from the root r to v contain no $w \neq v$, such that $p_w = p$. Then this path is called *trajectory* τ_p, and $|\tau_p|$ is the *receiving time* of p. Whenever two schedules T, T' are considered, t, t' (resp. τ, τ') denote times (resp. trajectories) in the corresponding schedules. Say, schedule T' is a *refinement* of T (write $T' \leq T$), if for every processor p of T its receiving time $|\tau'_p|$ in T' is no worse than in T: $|\tau'_p| \leq |\tau_p|$.

We denote by T_v a sub-tree (sub-schedule) of T rooted in v.

W.l.o.g we assume that l and the q_is are integral, otherwise some normalization factor may be applied.

We note here that for the sake of generality we consider S and l to be a part of the input and with unlimited size (in terms of n). However, the interesting practical applications will commonly have $l = O(1)$ as it is a fixed parameter of the corresponding system, and typically S will be a fixed set of services too, in which case the q_i's are all $O(1)$.

2.2. Properties of Schedules

Intuitively, it seems that the faster messages should be used first to maximize the number of sending nodes as fast as possible. This intuition is only partly correct. For example, if $S = \{(c_1 = 2, q_1 = 1), (c_2 = 1, q_2 = 2)\}$ and $l = 1$, broadcasting to two processors with a budget $B = 3$ is possible in time 2, but only if the root sends a slow message first and the fast message second.

However, if we consider the messages along any single trajectory, then the intuition is correct:

Say a trajectory is *monotone* if the transmission cost does not increase when traversing the trajectory from the root down. A schedule is monotone if all its trajectories are monotone.

Claim 2.1 *Any schedule has a monotone refinement.*

Proof: Let schedule T be non-monotone. Then T has a trajectory with a slower transmission edge $e_s = v\vec{u}$ preceding the faster $e_f = x\vec{w}$, $t_v < t_x, q_f < q_s$. Let T' be obtained from T by switching the services of e_s and e_f (see Fig. 2). Then $t_v = t'_v$, $|t_w| = |t'_w|$. Similarly, the receiving times of the processors in T_w, T'_w and outside the subtree T_u, T'_u are the same. For any x in $T_u - T_w$, $t'_x = t_x - (q_s - q_f)$. Repeating the above, obtain the desired monotone schedule. ■

In fact, the receiving times can be improved for some nodes (without delays for the others) by making sure that even on different trajectories the faster edges are not preceded too much by the slower ones:

Say, a schedule T is *service-layered* if whenever in T transmission $e_f = v\vec{w}$ uses service (c_f, q_f), and transmission $e_s = x\vec{y}$ uses (c_s, q_s) such that $q_s - q_f \geq l$, then $t_v < t_x + q_s$.

Claim 2.2 *Every schedule has a service layered refinement.*

Proof: Let T be a schedule with a faster edge $e_f = v\vec{w}$, a setup edge $v\vec{v'}$ and a slower edge $e_s = x\vec{y}$, such that $q_s - q_f \geq l$ and $t_y = t_x + q_s < t_v$. We prove the claim by showing that there is a schedule T' such that

- for each processor p in T_w its receiving time in T' is $|\tau'_p| = |\tau_p| - (t_v - t_x - q_s)$;
- for each p in T_y but not in T_v its receiving time in T' is $|\tau'_p| = |\tau_p| - (q_s - q_f - l)$;
- for each p in $T_{v'}$ but not in T_y its receiving time is $|\tau'_p| = |\tau_p| - l$;
- for each p in both $T_{v'}$ and T_y its receiving time is $|\tau'_p| = |\tau_p| - (q_s - q_f - l) - l$;
- for all other processors the receiving times in T, T' are the same.

Thus, if $q_s \geq q_f + l$ and $t_v - t_x \geq q_s$ then T' is a refinement of T. By repeating the above till there are no such e_s, e_f obtain the service-layered refinement of T.

Consider the effect of changing T to T' as shown in Fig. 3: slow transmission edge $x\vec{y}$ is replaced with a fast one $x\vec{y'}$ followed by a setup $y\vec{'}y$, w now receives a slow message from y' (that is why a setup edge $y\vec{'}y$ is added); since v no longer sends a message (to w) its setup edge $v\vec{v'}$ is no longer needed, so v' can be moved up into v; the rest is unchanged.

Clearly, the nodes/processors outside T_y and T_v are not affected by the change from T to T'.

$t'_w = t_x + q_f + q_s$; $t_x = t_w - q_f - (t_v - t_x)$. So, $t'_w = t_w - t_v + t_x + q_s$. Thus, for every z in T_w, $t'_z = t_z - (t_v - t_x - q_s)$.

communication model [1, 6, 7]. Using multiple communication services is reminiscent of the MULTI_MEDIA communication model [9]. None of the previous communication models, however, is dealing with the quality-of-service dimension.

The analysis of Lopsided Trees was given in [8]. The relation of these trees and broadcast algorithms is discussed in [9]. Algorithms that were given in these works are used here as building blocks.

A previous work that does take quality-of-service into account consider information gathering and searching on the Internet [10]. Similar to this work, [10] is also dealing with speed related quality-of-service in the context of data gathering. [10] consider sources that have prices and probabilities of providing the answer for the search, and presents two models: the *cost model*, where the cost of a search is the sum of the costs of the queried sources, and the *reward model* in which there is a certain delay for each source and the price is the reward given to the first source which supplies the answer. However, the MULTI_SERVICE model is drastically different from these two models in the use of parallelism: In the models of [10] there is unbounded parallelism - namely the client can activate as many services it wants. The task is then finished when the first of services successfully ends. Solution is thus a linear schedule of the services. In the MULTI_SERVICE model, at each point, a node (client or a commissioned service) may activate only one service. The parallelism is obtained by the tree-like structure of the schedule - namely, the fact that commissioned services that already have been activated get to activate other services for the same original task.

2. Model

2.1. Notations and Definitions

Let there be n processors which must receive the broadcast message. Since each processor needs to receive it exactly once, exactly n messages must be sent during the broadcast. Let the set of available services be $S = \{(c_i, q_i) \| i = 1, \ldots, k\}$, where the i-th service costs c_i for a point-to-point message transmission with the quality of the service q_i denoting (an upper bound on) the transmission time.

W.l.o.g., we assume that $c_i > c_{i+1}$ and $q_{i+1} > q_i$, for $1 \leq i \leq k-1$, i.e. the services are listed in the order of decreasing costs and the cheaper services are slower (a service which is more expensive and slower than another available one would not be used and can be ignored). Also, we assume that for any service the delay includes the setup for the next message, so $q_i > l$ for all i. Since the leaves do not need to setup for the next message, the broadcasts' times we compute below can be reduced by l steps. We ignore this saving for the sake of simplicity.

A *broadcast schedule* indicates where, in what order, and using which service should each processor forward the message M when it receives it. We represent it as a tree (e.g., as in Figure 1): the nodes represent processors at different times, and each edge represents either message transmission or setup (cf. [9]).

More formally: A broadcast *schedule* T is a directed tree. Each node v in the tree corresponds to some processor p_v at a time t_v when p_v is receiving and/or sending a message. The root r corresponds to processor p_0 (the original sender of the message) at time $t_r = 0$. Each non-leaf node v has exactly two out-edges:

- A *transmission* edge $e = v\vec{w}$, corresponds to a message sent, using some service i, by processor p_v at time t_v to $p_w (\neq p_v)$, and received at $t_w = t_v + q_i$.

- A *setup* edge $e = v\vec{u}$ corresponds to some processor $p_v = p_u$ during time from t_v to $t_u = t_v + l$ when p_v is preparing to send the next message.

In the figures we represent the setup edges with the dotted vertical lines, while the solid lines depict the transmission edges.

In our model we restrict each node to have at most one in-edge, at most one setup out-edge and at most one transmission out-edge.[1]

The *cost* $B(T)$ of the broadcast is the sum over all transmission edges of T of the costs associated with the corresponding services. $S(T)$ is the set of services used by T. The number of leaves in a schedule T is the number of processors participating in the broadcast. $\|T\|$ denotes the number of receiving processors, i.e., the number of leaves in T minus one.

Define $H(n, B, S) = \min\{height(T) : \|T\| = n, B(T) \leq B, S(T) \subseteq S\}$ be the minimum time required to broadcast a message to n processors, given the services S and budget B. Similarly, let $B(t, n, S) = \min\{B(T) : height(T) \leq t, \|T\| = n, S(T) \subseteq S\}$ be the minimal cost of a broadcasting to n processors within time bound t and using services S; and $N(t, B, S) = \max\{\|T\| : height(T) \leq t, B(T) \leq B, S(T) \subseteq S\}$ denote the maximal number of processors to which it is possible to broadcast a message within time t and budget B, using services S.

We consider the following related problems (assume $S(T) \subseteq S$ everywhere below):

[1] Relaxing the last requirement one can obtain MULTI_PORT (rather than SINGLE_PORT) model. Relaxing the bound on in-edges is meaningless, and allowing multiple setup edges makes sense only in some MULTI_PORT models with different and parallel setup times.

taken into account when the related service is provoked. We thus assume that it takes some overhead of l steps for the client to call a service. In particular, l may depend on the size of some information which the client asks the service to manipulate; In this paper we consider the distribution of a single message, hence the setup time l will remain fixed for all the transmissions throughout the paper.

After the service activation time, the client is free to start another round, while the submitted message may still be on its way to its target using the service's resources. We note that this mode of service-activation is different from the one in [10] as it limits the amount of parallelism in calling the services.

1.2. Broadcasting on a budget

The question we consider in this paper is how to disseminate information to many cooperating clients as fast as possible using a variety of point-to-point communication services and a limited budget. We call this problem *broadcasting on a budget using multiple communication media*, or simply, *broadcasting*. A related problem is to broadcast when some bound, or, a deadline, is put on the time to complete it, and the objective is to minimize the cost. As it turns out these two problems can be solved by using the same tools.

Broadcasting in the MULTI_SERVICE model involves some characteristic features as follows. In the broadcasting problem clients correspond to processors. The underlying services that we consider are message passing systems, in which any processor can submit to the network a *point-to-point* message destined at any other processor. The service is responsible for delivering the messages from their sources to their destinations. Each service corresponds to some communication medium (or, bandwidth) which involves a latency parameter measuring the time it takes the communication service provider to deliver a message to its destination.

The broadcast operation is as follows: There is a given budget B, and a message M stored initially at a certain processor, say, processor 0, and which is to be communicated to n other processors. There is a set $S = \{(c_i, q_i) \| i = 1, \ldots, k\}$ of k point-to-point communication services, where c_i is the cost of using the ith service, and q_i denotes the time within which the ith service is guaranteed to deliver message M to the indicated (single) destination. Calling a communication service involves a setup time of l steps. While handing another processor the message M, the sending processor may ask the receiving one to be in charge of broadcasting M to a subset of the remaining destinations. Hence the broadcast schedule has a tree-like structure, as shown in Figure 1.

The rest of the paper is as follows. In Subsec. 1.3 we list some related work. In the Sec. 2 we give formal definitions (Subsec. 2.1) and general properties of broadcast schemes in our model (Subsec. 2.2). Section 3 gives algorithms for computing efficient broadcasting schedules. Subsection 3.1 considers the case of just two services, while Subsection 3.2 solves the general problem for arbitrary number of services. The complexity issues of these algorithms are addressed in Subsec. 3.3.

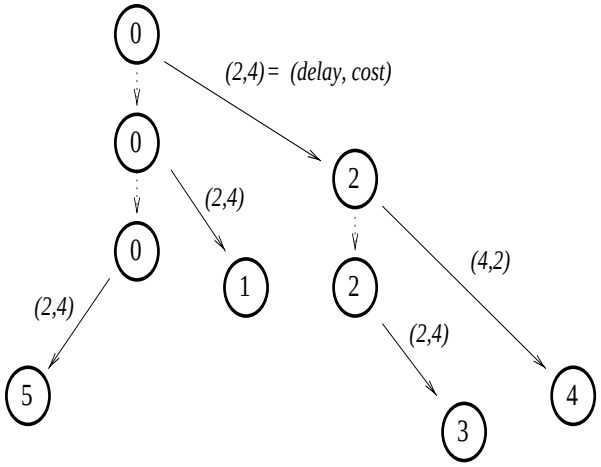

Figure 1. A tree representing a broadcast schedule for $n = 5$ processors. The algorithm for scheduling uses two services of costs $2, 4$ and respective delays $4, 2$. The arrows between tree nodes of different id represent sending a message, and the vertical arrows represent a setup time of l steps between successive message transmissions by the same processor. Initially, processor 0 transmits the message to processor 2 in step 1 using a service of delay 2 and cost 4, then after a setup time of l steps it uses the same service to send the message to processor 1. Meanwhile processor 2 uses the service of delay 4 and cost 2 to transmit the message to processor 4, etc. The total cost of this algorithm is 18 and its total time is $\max\{2l + 2, l + 4, 6\}$.

1.3. Related Work

Our communication-oriented service model was inspired by, and is generalizing, the successful POSTAL communication model that was introduced by Bar-Noy and Kipnis in [3]. Indeed the two models coincide when the budget is unlimited, or when there is only a single service available. Querying each service at a time corresponds to the SINGLE_PORT communication model [4, 3, 2, 5]. Using multiple services has some similarity with the MULTI_PORT

Broadcasting on a Budget in the Multi-Service Communication Model

Gene Itkis
NDS Technologies Israel
Jerusalem, Israel
itkis@ndsisrael.com

Ilan Newman
Math and Computer Science
Haifa University, Haifa, Israel
ilan@mathcs.haifa.ac.il

Assaf Schuster
Dept. of Computer Science
Technion, Haifa, Israel
assaf@cs.technion.ac.il

Abstract

In this paper we introduce the MULTI_SERVICE *model of network communication. This model attempts to capture recent communication technology trends, such as aspects of quality-of-service and their relation to the emerging technology of automatic pricing, e.g. for Internet services. The* MULTI_SERVICE *model differs from related models by taking communication and service activation time into account, thus restricting parallelism to better fit reality. Thus, our model extends and refines previous successful models for network communication.*

We consider the application of this model to communication problems, where the services are certain communication media, or, connection providers, with respective pricing policies. We give some insights and an algorithm for optimal dissemination of information in this model when given a fixed, limited budget.

1. Introduction

1.1. Background

Many information services are available on the Internet with the touch of a fingertip. Currently many of these are offered free of charge. In part, this is probably due to the lack of reliable pricing infrastructure. Once such pricing technology is matured it will be integrated with the existing information browsing mechanisms, thus extending the current information-fetching device into an ultimate generic servicing-and-billing platform. The variety of choices and the related complexity involved in taking intelligent decisions call for the development of computational models and efficient algorithms for these purposes.

In the future, the services that will be provided over the Internet are likely to include the following two parameters:

Price is what the service provider charges per service use,

Quality-of-service is a certain quality-related guarantee (speed, reliability, etc.) that the vendor is committed to provide for the service at the corresponding price.

For example, consider courier vs. regular mail where the corresponding quality-of-service is the speed of delivery.

There may be some additional associated features that are characteristic for any specific type of service, and must be described in the context of the shopper's objective function. Optimization becomes more complicated when the same task is to be performed repeatedly several times, and there is a given total budget which is not likely to cover the best available service for all of these uses ("to which of the addresses should I use the courier?"). Choice may get even more complicated when the optimization of the global objective function depend in some non-trivial way on the order of the chosen services, and their corresponding performance (as in the case of broadcasting on a budget, considered below).

Commonly, a shopper will be faced with a set of k services $S_1, \cdots, S_k$. To the i'th service S_i corresponds a pair (c_i, q_i), where c_i corresponds to the cost of using the service S_i, and q_i is this service performance, or: its guaranteed quality-of-service. This model may be used to deal with the general shopping question, involving the coordination of several different tasks towards a common goal, and finding service providers for each of these tasks.

Given a certain task and an allocated budget, the user will have to choose a subset of the service providers and a schedule for their activation which will guarantee that the task will be carried satisfactorily while meeting the budget constraint. For example, the work of Etzioni et. al. [10] may be viewed in this framework, where the quality-of-service corresponds to the probability of success and the needed time. Alternatively, given a set of performance constraints (e.g., deadlines) the user needs to select and schedule the services to minimize the cost. In this work we consider *communication services*, of which the service quality is commonly measured by *speed*. Since the electronic communication delays of activating these services seem to be very large (as experienced by anyone browsing the Internet), they must be