THE GENERALIZED POSTAL MODEL - BROADCASTING IN A SYSTEM WITH NON-HOMOGENEOS DELAYS[*]

Amotz Bar-Noy and Udi Nir

Faculty of Engineering - Tel-Aviv University
amotz@eng.tau.ac.il udi_nir@ncc.co.il

ABSTRACT

In many communication systems we may assume that every node is directly connected to every other node. It does not matter whether the system is a massively parallel computer where each processor is a node and the connectivity is achieved using a software communication library or if the system is the Internet where each node is a computer and the connectivity is achieved using IP.
The Postal model started with this assumption and in addition assumed that the process of delivering a message between two nodes takes a constant amount of time called the communication delay (greater than one unit of time), and that when a node sends a message or receives a message it wastes a single time unit.
In this paper, we enhanced the Postal model by removing the assumption that the communication delay is constant. Without this constraint, the broadcasting problem is NP-Complete and therefore we study two directions : (i) We present heuristic algorithms and analyze them. (ii) We evaluate simple topologies of networks (by assuming large delays when there is not a direct connection). We have also tested the heuristic algorithms with simulations and compared them to the optimal solutions.

1. INTRODUCTION

In many communication networks the participating nodes communicate with each other by sending and receiving messages. The communication networks may be global ones such as the Internet, smaller ones such as LANs and even a set of processors in a parallel computer. Usually, each node makes use of a service supplied by an underlying communication layer that hides the specific communication mechanisms and supplies the ability to send and receive messages to and from any other node in the system. In the Internet for example, this layer is IP.
In parallel computers such as Intel Delta Machine [2] it is assumed that the process of passing a message between any two nodes using the point to point connectivity supplied by the communication layer takes the same time. In some communication networks this assumption is close to reality as well.
Many applications require primitives that involve communication among the nodes in the system. For example, the broadcast primitive that allows the dissemination of a message from one node to a set of remote nodes (e.g. video conferencing over the Internet or parallel algorithms in a multi-processor computer). Another example is the primitive of adding numbers that are stored in different nodes. These applications require the primitive to be as fast and as efficient as possible. In most systems however, the implementations of these primitives use the Telephone model that assumes that the process of transferring a message between two node requires the participation of the nodes during the whole process. This is true in systems that use circuit switching but not in systems that use packet switching.
In [1], Bar-Noy and Kipnis present the Postal Model in which all processors are connected, the transmission time is λ for every pair (and is greater than one unit of time) but after one unit of time the sending processor is free and can do other things. In particular it may send the message to a different processor. Bar-Noy and Kipnis show an optimal broadcast algorithm for the Postal Model that yields a broadcast time of $\Theta(\lambda \log n/\log(\lambda+1))$. The doubling process implied by the Telephone model yields a broadcast time of $\Theta(\lambda \log n)$.
All the above models and research were limited to systems with homogeneous delays and addressed mainly the parallel computers arena. In today's world however, communication systems cannot be characterized by a homogeneous delay.
In this paper we investigate the Generalized Postal Model - a system with full connectivity but with different delays between each pair of nodes. We study the problem of broadcasting a single message in the Generalized Postal Model and investigate it in different scenarios. Because the problem is NP-Complete [4], we search for heuristic algorithms (and find bounds and approximation ratios for them) on one hand, and try to simplify the problem by forcing certain delays to be ∞ on the other hand.

Proofs for the various lemmas and equations and detailed results can be found in the complete master thesis [3].

2. THE GENERALIZED POSTAL MODEL

Let us first define the *Postal Model* according to [1]:
Definition 2-1. A message passing system with n nodes - *MPS(n)* - consists of a set of nodes $\{ p_0, p_1, \ldots, p_{n-1} \}$ with the following two properties :

- Full connectivity: each node p in the system can send a point-to-point message to any other node q in the system.
- Simultaneous I/O: each node p can simultaneously send one message to node q and receive another message from node r (r does not have to be q).

In the postal model we use the term *message* to refer to an atomic piece of data communicated between nodes. A message cannot be broken into smaller pieces at the sending node, at the communication network, or at the receiving node. The size of an atomic message is defined as one unit of size, and the time it takes to send or receive an atomic message is defined as one unit of time.

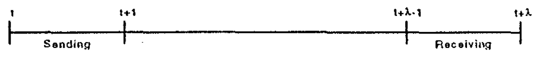
Definition 2-2. A message passing system with n nodes and a communication latency $\lambda \geq 1$- *MPS(n,λ)* - is a message passing system *MPS(n)* with the additional property :

[*] This paper is an abstract of a master thesis submitted by Udi Nir under the supervision of Amotz Bar-Noy

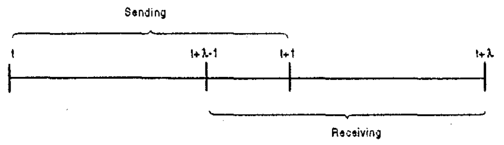
- Communication latency: if at time t node p sends a message M to node q, then node p is busy sending message M during the time interval [t,t+1], and node q is busy receiving message M during the time interval [t+λ-1,t+λ].

Note that when $1 \leq \lambda < 2$ the receiving process overlaps the sending process and when λ=1, the Postal model reduces to the Telephone model in which both nodes are engaged all the time in the sending and receiving process.

The case $\lambda \geq 2$:



The case $1 \leq \lambda < 2$:



We now define the *Generalized Postal Model* (GPM) :

Definition 2-3. A message passing system with *n* nodes and a communication latencies $\lambda_{i,j} \geq 1$ - $MPS(n,\lambda_{i,j})$ - is a message passing system $MPS(n,\lambda)$ with the additional property:

- Pair dependent communication latency: if at time t node p sends a message M to node q, then node p is busy sending message M during the time interval [t,t+1], and the node q is busy receiving message M during the time interval $[t+\lambda p,q-1, t+\lambda p,q]$.

Remark : When $\lambda_{i,j} = \infty$, we view it as node i cannot send messages directly to node j. In such cases we discuss "topologies" which are cases where all present edges have $\lambda_{i,j} < \infty$ and the edges that have $\lambda_{i,j} = \infty$ are missing. When $\lambda_{i,j} = C$ for all pairs, we return to the Postal model described in [1].

2. 1. The Broadcasting Problem

The problem of broadcasting one message in $MPS(n,\lambda_{i,j})$ is defined as follows: At time t=0, node p_0 has a message M to broadcast to nodes $p_1,\ldots,p_{n-1}$. The goal is to minimize the time at which the last node receives message M. The running time of algorithm A in $MPS(n,\lambda_{i,j})$ is denoted as $T_A(n,\lambda_{i,j})$.
Let us denote this broadcasting problem as $GBP(n, \lambda_{i,j})$.
Slater, Cockayne and Hedetniemi [4] have proven that $GBP(n, \lambda_{i,j})$ where $\lambda_{i,j} = \infty$ or 1 is already NP-Complete. In the thesis we present another proof that shows that $GBP(n, \lambda_{i,j})$ is NP-Complete by a reduction to the Hamiltonian circle problem.

2. 2. The Broadcast Tree

A solution to a broadcasting problem can be represented by a broadcast ordered tree. In this tree, the root corresponds to P_0, the rest of the nodes in the tree correspond to the rest of the nodes in the system. The i-th branch that emanates from node p to node q, represents the fact that i-1 units of time after receiving the message, node p sent the message to node q . In the figures depicting the broadcast tree, the order of transmission is from left to right.
For each branch there is a delay parameter that describes the communication delay between the two nodes that are at the ends of this branch.
To calculate the time it takes for the message to reach a specific node we sum all delays in the path that leads from the root to that node and to that we add the number of branches that emanate from any of the nodes on the path which are to the left of our path. In order to find the broadcast time for the whole system according to the broadcast tree we take the maximum among the times of all the leaves in the tree.

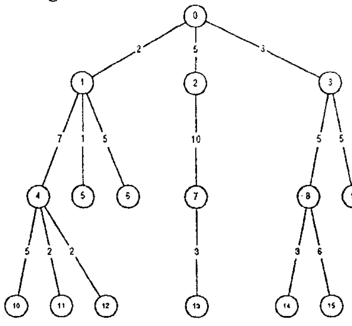


Figure 2-1 A Broadcast Tree Example

In the example of Figure 2-1, the root is node 0 and this node passes the message to node 1, then to node 2 and then to node 3. The path to node 15 is as follows : node 0 (the root) to node 3, then to node 8 and from there to node 15. The time it takes for the message to reach node 15 according to this broadcast tree is as follows : at time t=0, the root sends the message to node 1, at t=1 to node 2 and only at t=2 to node 3. Node 3 receives the message after 3 time units which is at t=5. It immediately sends it to node 8 which receives it at t=10 (5 time units later). Node 8 first sends the message to node 14 and only after 1 more time unit (t=11) it send it to node 15. The message is delayed for 6 time units thus reaching its destination, node 15, at time t=17.
Note that the broadcast time of this tree is 19, the time node 13 receives the message.

2. 3. The Generalized Fibonacci Function

The *Generalized Fibonacci Function* is defined as follows :

$$F_\lambda(t) = \begin{cases} 1 & if\ \ 0 \leq t < \lambda \\ F_\lambda(t-1) + F_\lambda(t-\lambda) & if\ \ t \geq \lambda \end{cases}$$

Define $f_\lambda(n)$ to be the index function of $F_\lambda(t)$. I.e. $f_\lambda(n) = t$ implies that $F_\lambda(t) \geq n$ and $F_\lambda(t-1) < n$

Note that for λ=1 we get $F_1(t) = 2^t$ and $f_1(n) = \lceil \log(n) \rceil$ and that for λ=2 we get that $F_\lambda(t)$ is the standard Fibonacci progression where $f_2(n)$ is the index of the first Fibonacci number that is no smaller than n.
We now have the following Lemma :

Lemma 2-1: $\sum_{i=0}^{k} F_\lambda(i) = F_\lambda(k+\lambda) - 1$

In [1], Bar-Noy and Kipnis prove that $f_\lambda(n)$ is the minimal time required to broadcast a message in the Postal model with n nodes and delay λ. They also prove that $F_\lambda(t)$ is the maximal number of nodes that a message can be broadcast to in the Postal model in a time period t.

Using the above we have the next Lemma :

Lemma 2-2: $f_\lambda(x \cdot y) + (\lambda - 1) \geq f_\lambda(x) + f_\lambda(y) \geq f_\lambda(x \cdot y)$

3. HEURISTIC ALGORITHMS FOR GPM

3. 1. Dijkstra's Algorithm

Dijkstra's algorithm solves the single-source shortest-path problem on a given graph G(V,E) and a weight function w(u,v) ≥ 0. We can apply this algorithm to solve *GBP(n, $\lambda_{i,j}$)* where w(u,v) = $\lambda_{u,v}$.
Let us denote the broadcast tree as S, the parent of a node v in S as p[v] and the time a node v in S received the message and therefore is able to send it to one of its' neighbors as r[v] . In Dijkstra's algorithm, this time remains constant once it has been set, ignoring the fact that sending the message requires one unit of time.
The following is a high level code of the algorithm :

```
A) $r[i] \leftarrow \infty;\quad p[i] \leftarrow NIL;\quad 0 \le i < n$
B) $S \leftarrow \{v_0\};\quad r[0] \leftarrow 0;\quad Q \leftarrow V - \{v_0\}$
C) Let $u,v$ be s.t. $r[u] + w(u,v) = \min_{i \in S, j \in Q}(r[i] + w(i,j))$  ($u \in S$, $v \in Q$)
D) $S \leftarrow S \cup \{v\}; p[v] \leftarrow u; Q \leftarrow Q - \{v\}$
E) If $Q \ne \varnothing$ go to step ( C )
```

Complexity (when no complex data structures are used)
The loop occurs n-1 times
Step (C) takes at most O(n/2 * n/2) = O(n^2)
Step (E) takes O(1) time.
Thus the complexity is O(n^3).

The complexity also satisfies $\sum_{i=1}^{n-1}\Omega(i \bullet (n-i)) = \Omega(n^3)$

Therefore the complexity of this algorithm is $\Theta(n^3)$

It is known however, that Dijkstra's algorithm can be solved in O(m + log(n)) time using Fibonacci heaps (where m is the number of edges in the graph).

Lower Bound for Dijkstra's Algorithm

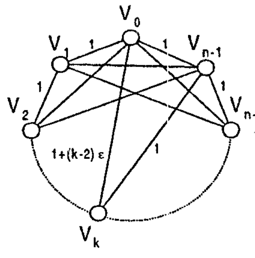


Figure 3-1 Lower Bound for Dijkstra's Algorithm

Consider a graph with the following properties (Figure 3-1) :
- The graph is fully connected,
- The delay of all edges ei,j for i,j ≠ 0 is 1 (one),
- The delay of e0,k for 1 ≤ k ≤ n-1 is 1+ (k-2)ε where 0<ε<<1

Dijkstra's algorithm adds to the broadcast tree the edges $e_{0,1}$, $e_{0,2}$, .., $e_{0,n-1}$ resulting in a star formation and a broadcast time of n-2+1+(n-3)ε which is O(n) .
The optimal algorithm (described in [1]) broadcasts the message in O(log(n))
The ratio between Dijkstra's algorithm and the optimal one for this example is $\Omega\left(\frac{n}{\log(n)}\right)$.

3. 2. The Greedy Algorithm

The Greedy algorithm solves *GBP(n, $\lambda_{i,j}$)* on a given weighted graph G(V,E) with n nodes and w(u,v) = $\lambda_{u,v}$.
This algorithm takes into consideration the fact that a node "wastes" one time unit while sending the message.
Let us denote the broadcast tree as S, the parent of a node v in S as p[v] and the time a node v in S can send the message to one of its' neighbors as r[v] (this time r[v] does not remain constant but is altered according to the actual time the node is able to send the message).
The following is a high level code of the algorithm :

```
A) $r[i] \leftarrow \infty;\quad p[i] \leftarrow NIL;\quad 0 \le i < n$
B) $S \leftarrow \{v_0\};\quad r[0] \leftarrow 0;\quad Q \leftarrow V - \{v_0\}$
C) Let $u,v$ be s.t. $r[u] + w(u,v) = \min_{i \in S, j \in Q}(r[i] + w(i,j))$  ($u \in S$, $v \in Q$)
D) $S \leftarrow S \cup \{v\}; p[v] \leftarrow u; Q \leftarrow Q - \{v\}$
E) $r[u] = r[u] + 1$
F) If $Q \ne \varnothing$ go to step ( C )
```

Note the difference : the addition of step (E).

Complexity (when no complex data structures are used)
The complexity of this algorithm is the same as the previous one : $\Theta(n^3)$

Lower Bound for the Greedy Algorithm
Consider the following graph :
- There are n nodes in the graph
- The root is connected to k - 1 nodes with λ = 1
- Let 0<ε<<1
- The root is connected to one more node with λ = 1 + ε
- The node that is connected to the root with λ = 1 + ε is connected to k new nodes in the same way (k - 1 nodes with λ = 1 and one node with λ = 1 + ε)
- The new node that is connected with λ = 1 + ε is connected to k new nodes in the same manner and so on.

The broadcast time of the Greedy algorithm is $(k+\varepsilon) \bullet (n-1)/k$ which is n for $\varepsilon = k/(n-1)$
The broadcast time of the optimal algorithm is $(1+\varepsilon) \bullet (n-1)/k + k = \frac{n-1+k+k^2}{k}$
The ratio between the Greedy algorithm and the optimal one for this example when $k \ge 1$ is : $nk/(n+k^2)$
When $k = \sqrt{n}$ the ratio is $\Omega(\sqrt{n})$.

The Greedy Algorithm in the Postal Model
In the case of $\lambda_{i,j}$ = λ (a fully connected graph with edges of equal length) the greedy algorithm makes each node that has the message, send it to another node as soon as possible. The algorithm does not allow for idle time.
The optimal algorithm for the Postal model [1], performs in the same manner and therefore the Greedy algorithm is optimal for this case.

3. 3. Simulated Annealing

The process of simulated annealing is based on the simulation of
the annealing of solids [5].

The following pseudo-code outlines the algorithm :

```
repeat
    repeat
        PERTURB(configuration.i→configuration.j, $\Delta C_{ij}$)
        if $\Delta C_{ij} \le 0$ then accept else
           if $\exp(-\Delta C_{ij}/C_M) >$ random[0,1) then accept;
        if accept then UPDATE(configuration.j);
    until equilibrium is approached sufficiently closely;
$C_{M+1} := f(C_M)$;
M := M+1;
until stop criterion = true (system is 'frozen');
```

In this work we use the following implementation for the algorithm :

- The initial solution to the problem is a broadcast tree chosen randomly. The process used yields a tree with a higher degree for nodes closer to the root of the tree.
- The perturbation is cutting of an edge in the tree thus splitting a sub-tree off the main broadcast tree. Then we rejoin the sub-tree to the main one in a different place
- The cost function is of course the broadcast time using the chosen tree.
- Equilibrium is reached if no change in cost happens in a certain number of iterations
- The initial temperature, the cooling factor and the stop criterion (which is the minimal temperature) are chosen arbitrarily as 100, 0.99 times the Temperature and 10

We have also added the following obvious improvements to the "classic" algorithm :

- We always keep the best solution and return to it if at the end of the simulated annealing process no better solution was found.
- We rearrange the edges in the solution broadcast tree using the optimal algorithm described in section 4. 1.

4. TREES

4. 1. Optimal Broadcast

Stars

Star is a very basic topology. The only freedom an algorithm has is the order of the transmissions.

Lemma 4-1 : In order to minimize the broadcast time from P_0 to $P_1..P_{n-1}$ in a star shaped graph whose root is P_0, the order of transmission is from the longest edge to the shortest one.

Trees

In a tree topology the freedom a broadcast algorithm has, is to choose the order of the children of each node.

Lemma 4-2 - In order to minimize the broadcast time from P_0 to $P_1..P_{n-1}$ in a tree shaped graph whose root is P_0, we consider , recursively, each node to be a root of a star (organized according to Lemma 4-1) whose edges' length are the sum of the length of the edge and the broadcast time of their sub-tree.

4. 2. Approximation Ratios

Stars

Lemma 4-3 - The approximation ratio of any broadcast algorithm for a star is 2.

K-ary Tree

Lemma 4-4 - The approximation ratio of any broadcast algorithm for a k-ary tree is *k*.

5. THE GREEDY ALGORITHM ON A TWO CLUSTER GRAPH

Let us examine the behavior of the Greedy algorithm in the following graph :

- There are two clusters in the graph, C_1 and C_2
- There are n_1 and n_2 nodes in each cluster respectively ($n_1+n_2 = n$)
- All the nodes are connected among themselves
- The delay between two nodes in the same cluster is λ and the delay between two nodes in different clusters is λ_{intra}
- Node P_0 (which has the message to be broadcast) is located in C_1

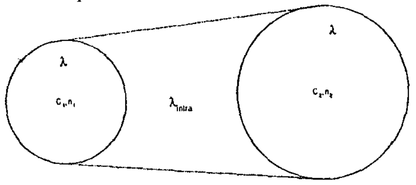


Figure 5-1 Two Clusters Topology

The Greedy algorithm passes the message to the nodes in C_1, then passes it from C_1 to C_2 (possibly many times) and then passes it internally in C_2.
If we force the process to wait until all nodes of C_1 have received the message and only then we let it continue to send the messages to C_2 then we may loose up to λ-1 time units due to nodes in C_1 that are waiting for the broadcast in C_1 to end.
Let $T_G(n_1, n_2, \lambda, \lambda_{intra})$ be the broadcast time for the Greedy algorithm.
Hence the time for the Greedy algorithm to broadcast the message to all nodes is :

Equation 5-1:

$$f_\lambda(n_1) + \lambda_{intra} + k - (\lambda - 1) \le T_G \le f_\lambda(n_1) + \lambda_{intra} + k$$

where k is the time it takes to pass the message to all nodes of C_2 since the message first arrives to the cluster.
Each node in C_2 that receives the message from C_1 can broadcast in the i time units until the process ends to $F_\lambda(i)$ nodes. Each time unit since the first n_1 messages arrived at C_2 , another n_1 messages arrive from C_1 and therefore

Equation 5-2: $n_2 = n_1 \sum_{i=0}^{k} F_\lambda(i)$

According to Equation 5-1 we can now write

Equation 5-3:

$$f_\lambda(n_1) + f_\lambda\left(\frac{n}{n_1}\right) + \lambda_{intra} - 2\lambda - 1 \le T_G \le f_\lambda(n_1) + f_\lambda\left(\frac{n}{n_1}\right) - \lambda + 1 + \lambda_{intra}$$

According to Lemma 2-2 we have :

Corollary 5-1

$$f_\lambda(n) + (\lambda - 1) \ge f_\lambda(n_1) + f_\lambda\left(\frac{n}{n_1}\right) \ge f_\lambda(n)$$

Using the corollary and Equation 5-3 we get

$$f_\lambda(n) + \lambda_{intra} - 2\lambda + 1 \le T_G \le f_\lambda(n) + \lambda_{intra}$$

The optimal algorithm uses T_{opt} time units to complete the broadcast. We know that :

- Since the algorithm has to broadcast to n-1 nodes and $\lambda \leq \lambda intra$, $T_{opt} \geq f_{\lambda}(n)$
- Since the algorithm has to pass a message between the two clusters at least one time $T_{opt} \geq \lambda_{\mathrm{int}\,ra}$

Therefore, $T_{opt} \geq \dfrac{f_{\lambda}(n) + \lambda_{\mathrm{int}\,ra}}{2}$

and hence the approximation ratio of the Greedy algorithm 2.

We now show examples in which the approximation ratio of the Greedy algorithm is approaching 2 :

When n_2=1 and $\lambda_{intra}=1+f_{\lambda}(n_1)$, the optimal algorithm could send the message to C_2 (the only node there) and then to the nodes in C_1. This process will require λ_{intra} time units. The Greedy algorithm will send the message in C_1 and only then to C_2. This will require $f_{\lambda}(n_1)+\lambda_{intra}-(\lambda-1)$ time units.

The approximation ratio in this situation is therefore

$$\frac{2\cdot\lambda_{\mathrm{int}\,ra}-1}{\lambda_{\mathrm{int}\,ra}} = 2 - \frac{1}{\lambda_{\mathrm{int}\,ra}} \xrightarrow[\lambda_{\mathrm{int}\,ra}\to\infty]{} 2$$

6. SIMULATIONS - TESTS, RESULTS AND ANALYSIS

6. 1. The Tests

The tests were divided into two categories according to the graph topology :

- Postal graphs .
- Two clusters graph.

The Postal Graph

We tested graphs of 8, 16, 32, 64, 128, 256 and 512 nodes. In each graph we use λ of 2, 4, 6, 8 and 10 in the following manner:

The first test was running the Greedy algorithm and getting the optimal result. We denote this result as the Postal broadcast tree.

Next, we use the λs as the mean for a randomly distributed delay for each edge in the graph. The distributions used were exponential, uniform and normal. For the uniform and normal distribution we also uses a varying variance of 10%, 30% and 50% of the mean (e.g. if the mean was 8 and the variance was 50% then for the uniform distribution we used weight = U[4,12]).

After the delays were calculated, we tested the delay of the broadcast tree that was formed by the Postal broadcast tree with the random weights. In this test we actually look at the case were we believe that the delays in the graph are a random variable with known average and variance but apply the Postal model as if they are constant (with the average delay).

Following this test, we applied the Greedy algorithm and the Simulated Annealing process using the actual random weights, thus assuming that each and every delay is known before applying the algorithm.

The Two Clusters Graph

We tested graphs of 8, 16, 32, 64, 128, 256, 512 and 1024 nodes. In each graph we divided the number of nodes between the two clusters in the following way: 1/7, 2/6, 3/5, 4/4, 5/3, 6/2 and 7/1. We also used λs (within the clusters) of 1, 2 and 3 and λ_{intra} (between the two clusters) of 10, 100 and 1000.

For each choice we have tested the Greedy algorithm and the Simulated Annealing process.

6. 2. Results and Analysis

It is clearly seen that in the Postal (and near Postal) model the Greedy algorithm outperforms the other algorithms.

In general, we can assume that an algorithm that has more information can perform better than one that has not. This explains why the Greedy algorithm outperforms the Postal one. Still we have to account for the fact that the Postal is sometimes better than the Simulated Annealing. Let us remember that the process starts at a random state and gradually moves to better and better states until it cannot improve the "energy level" any more. The results we achieved, show that when the number of nodes is small, the Simulated Annealing process is better than the Postal algorithm. In these cases the number of possible solutions that the Simulated Annealing may pass through is small and therefore it may find (within the cooling process that we set for it) optimal or near optimal solutions. As the number of nodes increases, the amount of possible solutions rises exponentially thus in the same cooling scheme, it is much less likely to encounter good (not to say optimal) solutions. We did see (as mentioned before) that as the cooling is done more slowly the performance of the Simulated Annealing improves.

If we look at the difference between the performance of the Postal algorithm and that of the Greedy algorithm we see that as the variance increases so does the difference. The absolute numbers, however, are very small and tell us that the penalty for not knowing the real delays in a communication system but instead knowing an average one is minor.

In the Two Clusters graph, we see again that when the complexity is low, the Simulated Annealing process is better than the Greedy algorithm (and is even optimal in cases). This is especially true in the cases when the number of nodes in the second cluster (the one without the root) is small thus yielding less possible "bad" solutions that the Simulated Annealing process can go through.

7. BIBLIOGRAPHY

[1] Bar-Noy A. and Kipnis S., Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems, Mathematical Systems Theory 27, 431-452 (1994).

[2] Bruck J., De Coster L., Dewulf N., Ho C. H. and Lauwereins R., On the Design and Implementation of Broadcast and Global Combine Operations Using the Postal Model, IEEE Transactions on Parallel and Distributed Systems, Vol. 1, Jan-June 1996, pp. 256-265.

[3] Nir Udi, The Generalized Postal Model, Master Thesis, Faculty of Engineering, Tel-Aviv University, June 1997

[4] Slater P. J., Cockayne E. J. and Hedetniemi S. T., Information Dissemination in Trees, Siam Journal on Computing 10(4), pp. 692-201, 1981

[5] van Laarhuven P. J. M. and Aarts E. H. L., Simulated Annealing : Theory and Applications.