Signature Searching in a Networked Collection of Files

Zhongwen Ying and Thomas G. Robertazzi, *Fellow, IEEE*

Abstract—A signature is a data pattern of interest in a large data file or set of large data files. Such signatures that need to be found arise in applications such as DNA sequence analysis, network intrusion detection, biometrics, large scientific experiments, speech recognition and sensor networks. Related to this is string matching. More specifically we envision a problem where long linear data files (i.e., flat files) contain multiple signatures that are to be found using a multiplicity of processors (parallel processor). This paper evaluates the performance of finding signatures in files residing in the nodes of parallel processors configured as trees, two dimensional meshes and hypercubes. We assume various combinations of sequential and parallel searching. A unique feature of this work is that it is assumed that data is pre-loaded onto processors, as may occur in practice, thus load distribution time need not be accounted for. Elegant expressions are found for average signature searching time and speedup, and graphical results are provided.

Index Terms—Database search, signatures, tree networks, mesh networks, hypercube networks

1 Introduction

A signature is a relatively small data pattern of interest embedded in a very large (in this paper sequential) data file. It is assumed signatures are temporally distinct and do not overlap each other. That is, there can be multiple signatures in a file. Because the files we study are much longer than the signatures, it is assumed that signatures have infinitesimally small length. Such signature searching occurs in network security, signal processing, medicine, image processing, and sensor technology and many other fields.

Most previous work on signature searching (known as template matching and string matching) develop algorithm for the detailed matching process. This paper, like [1], [2] addresses an upper level view of signature searching involving system performance evaluation. However we now briefly summarize some string matching work.

String searching, which is similar to our concept of signature searching, is a special case of pattern searching. String searching generically involves finding a pattern of length m in a text of length n over some alphabet. The worst case complexity of exact string matching is O(n) but the proportionality constant of the linear term can be very different depending on the string matching algorithm, ranging from m for the naïve algorithm to 2 for the Knuth-Morris-Pratt algorithm [3].

Approximate string matching involves string matching that allows errors. That is, the pattern and/or text suffer some corruption. Applications include noisy channels, speech recognition, hand writing recognition, finding DNA sequences in the presence of mutations and text searching. Approximate string matching algorithms utilize some distance metric to quantify the amount of difference between two strings. For instance, the edit distance is the number of differences between two strings. The computational complexity of approximate string matching can range from linear to NP complete depending on the error mechanism [4].

For on-line algorithms, it is assumed that the text is not known in advance. For off-line algorithms the text can be pre-processed, thus indexing can be used [5]. On-line algorithms generally consist of a phase of entering the string into a data structure and a phase of looking for a match using the data structure [6].

In this paper we assume the data is stored in flat files (i.e., very long linear sequences of data) stored at nodes of certain interconnection networks. We assume linear (in the file size) computational complexity which applies to exact string matching and some approximate string matching. Naturally more sophisticated database methodologies are possible and flat file are often converted into other structures [7], [8] but for initial raw data processing flat files are natural [9].

We envision a scenario where files containing signatures are placed on a multiplicity of (parallel) processors tied together by an interconnection network [1], [2]. Unlike the work in much of the divisible load theory literature [2], [10], [11], [12], we do not take the time to distribute the load to the processors and links into account. Rather we assume the files are pre-loaded onto the processors prior to time $t = 0$. This is relevant in certain applications. Load is often spontaneously distributed to processors without being scheduled, monitored or timed. Our goal is to determine expected search time and speedup under a variety of search protocols that largely differ in a number of aspects. These include:

1. Whether the number of signatures in a file at a node is known or unknown *a priori*. The latter is more likely in general but the former could also occur. For instance when a search is done for ''management''

• *The authors are with the Department of Electrical and Computer Engineering, Stony Brook University.*

Manuscript received 21 Aug. 2012; revised 27 Aug. 2013; accepted 24 Sept. 2013. Date of publication 7 Oct. 2013; date of current version 21 Mar. 2014. Recommended for acceptance by K. Li.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.258

in a company database, the number of signatures matches may be generally known *a priori*.

2. Whether a data file stored at a node contains one or more than one signature. For instance, a database of company employee information may contain one match for a given individual, but multiple matches for individual who are ''management''.
3. If no signatures are stored in a node, whether it implies that there are no signatures stored in the children nodes of the node. For instance a node may store aggregated/summarized national data and children nodes store more detailed state/provincial data. It may be possible in searching the national nodes to find children nodes that should be searched and those that need not be searched.
4. Differing degrees of sequentiality and concurrency in the search strategy. Both have been investigated in the divisible load scheduling literature over the years [10], [11], [12].

These are important assumptions that may be made in specific applications and we try to study this problem in a complete manner. In this context this work examines tree, mesh and hypercube interconnection networks [13], [14], [15], [16]. We look at trees, meshes and hypercubes mainly because these are fundamental interconnection networks. Trees are often used as spanning trees to distribute load in other types of interconnection networks. Meshes are often used as interconnection networks in parallel processors. Meshes are particularly well suited to networks on chips [17], [18], [19]. Hypercubes are widely used in parallel processors. One could certainly examine other interconnection networks but space limitations prevent this in this paper.

In earlier work the expected search time to find either one [1] or multiple signatures [2] with a uniform distribution of signatures in a flat file was found analytically. This earlier work involved trees and daisy chains and incorporated, unlike this work, load distribution time to processors over a network.

The rest of the paper is organized as follows. Section 2 discusses searching time and speedup in tree networks for different cases. The searching time and speedup in mesh networks (store and forward) for different cases are discussed in Section 3. A comparison of the speedups for different type of networks is presented in Section 4. Finally, this paper concludes with some possible extensions in Section 5.

2 Tree Networks

A general tree is shown in Fig. 1. Assume that in a tree a node is a structure which contains exactly one file. Each node has one, two or more child nodes, which are below it in the tree space. A node that has a child is called the child's parent node.

A node has at most one parent. Nodes that do not have any children are called leaf nodes. They are also referred to as terminal nodes. A node's height is the length of the longest downward path to a leaf from that node. The root's height is the height of the tree. The depth of a node is the length of the path to its root. The nodes in the same depth of the tree are said to be at the same level.

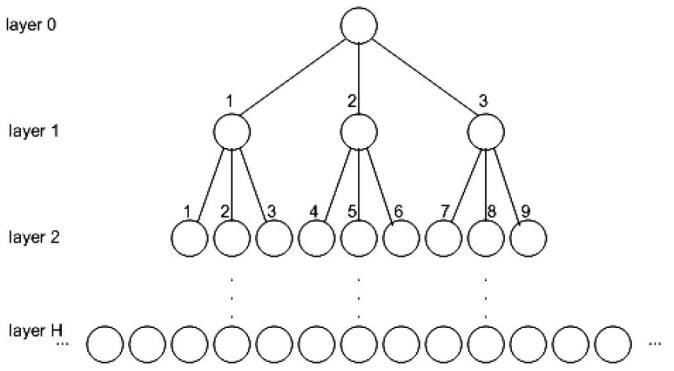


Fig. 1. Multi-level tree network.

The height of the multiple-level tree in this paper is H. Throughout this paper there is a single file, possibly containing signatures, in each node. In all the cases for tree networks which we examine the number of children nodes n_i per node is a random variable from 1 to N, possibly different for each level. But at the same level the number of children in each subtree is the same. In this section if a parent node has a signature(s), its children nodes may also have signatures. If the parent node does not have any signature of interest, its children nodes also have no signatures and do not need to be searched. Some dependency between signature occurrences is thus being assumed. This may model locality of reference—if a node has a signature(s), there may be related information (signatures) on its children.

The lengths of all the files in the tree are same, L (bits). The inverse searching speed for one processor is w (s/bit). Suppose that the expected time we need to search a file with signature is $\overline{X}$, because the expected position of signature is in the middle of such file (i.e. uniform distribution in linear file assumption). The time to search the whole file is $2\overline{X}$. while $2\overline{X} = L \cdot w$. The purpose of this paper is to calculate the time we need to find all of the files with signatures. We will discuss several cases in Sections 2.1 and 2.2 using the assumptions listed in the introduction.

2.1 Number of Signatures is Unknown

2.1.1 Each File Has at Most One Signature

Let M_i be the number of signatures in the i_{th} layer and n_i be the number of children nodes in each subtree in the i_{th} layer. If there is at most one signature in one file, $M_i \leq M_{i-1}n_i \leq M_{i-2}n_{i-1}n_i \leq \cdots \leq n_1 n_2 \ldots n_i$. The expected time to search one file is $\overline{X}$, the searching process for a file will stop as soon as we find one signature (as also true in Sections 3.1.1 and 3.2.1).

Only one node in the tree commands all the nodes: First, assume that only the top most root node in the multi-level tree commands all the nodes to search, for every layer, every file in the layer and each node replies to the top node whether there is a signature in the node. That is, one layer is searched sequentially at a time. The expected time to search the entire tree is

$$T_{u1} = n_1\overline{X} + M_1 n_2\overline{X} + \cdots + M_{H-1}n_H\overline{X} = \sum_{i=1}^{H} M_{i-1}n_i \cdot \overline{X}. \quad (1)$$

Here, $M_0 = 1$.

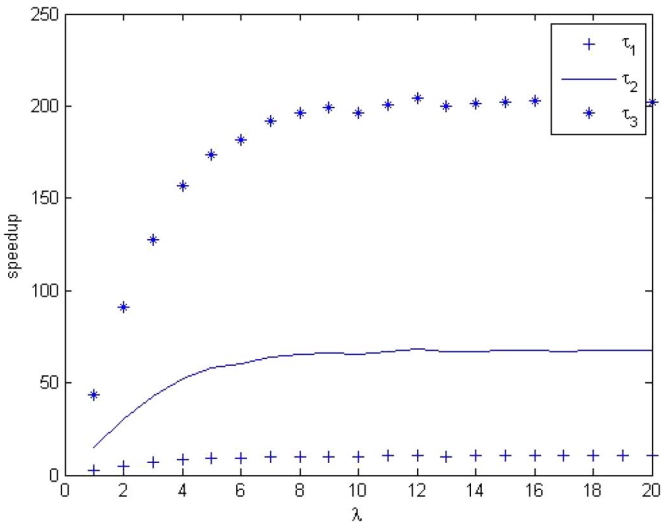


Fig. 2. Speedup when the number of signatures is unknown and each file can have multiple signatures.

Now we consider a homogeneous case that for all the subtrees in a certain layer the probability of a signature is the same: p_i for the i_{th} layer. Equation (1) will be $T_{u1} = n_1\overline{X} + p_1 n_1 n_2 \overline{X} + \cdots + (\prod_{j=1}^{H-1} p_j n_j) n_H \overline{X} = \sum_{i=1}^{H} (\prod_{j=1}^{i-1} p_j n_j) n_i \overline{X}$. A homogeneous case occurs if $p_1 = p_2 = \cdots = p_{H-1} = p$, Equation (1) will be $T_{u1} = \sum_{i=1}^{H} (\prod_{j=1}^{i} n_j) p^{i-1} \overline{X}$. The fully homogeneous case occurs if the number of files in all of the subtrees is the same: n, and the probability of signature is also the same: p, then the expected search time is $T_{u1} = \sum_{i=1}^{H} n^i p^{i-1} \overline{X} = \frac{p^H n^{H+1} - n}{pn-1} \overline{X}$.

Every root node of each local tree commands the nodes below it: In this case each level is searched sequentially, within each subtree the search is sequential but all subtrees at the same level are searched in parallel. The time to search the i_{th} layer is $n_i\overline{X}$. The total expected time to search the multi-level tree is

$$T_{u2} = n_1\overline{X} + n_2\overline{X} + \cdots + n_H\overline{X} = \sum_{i=1}^{H} n_i\overline{X}. \tag{2}$$

The speedup compared with case 2.1.1.1 is

$$\tau_{u1} = \frac{\sum_{i=1}^{H} M_{i-1} n_i}{\sum_{i=1}^{H} n_i}. \tag{3}$$

For the fully homogeneous case, $\tau_{u1} = \frac{p^H n^{H+1} - n}{pn-1} \frac{1}{nH} = \frac{(pn)^H - 1}{(pn-1)H}$.

All the files in every Level are Learched in Parallel, Levels are Searched Sequentially: In this case each level is searched sequentially and within a level the nodes are searched in parallel. For every level the search time is only $\overline{X}$. The total search time is $T_{u3} = H\overline{X}$. The speedup compared with case 2.1.1.1 is

$$\tau_{u2} = \frac{\sum_{i=1}^{H} M_{i-1} n_i}{H}. \tag{4}$$

For the fully homogeneous case, τ_{u2} is $\frac{((pn)^H - 1)n}{(pn-1)H}$.

All the files in the tree are searched in parallel: This case is the fastest search method. All the files in the entire tree are searched in parallel. The total search time is $T_{u4} = \overline{X}$ and the speedup is

$$\tau_{u3} = \sum_{i=1}^{H} M_{i-1} n_i. \tag{5}$$

For the fully homogeneous case, τ_{u3} is $\frac{((pn)^H - 1)n}{(pn-1)}$.

It is apparent that as the height increases, all the three speedups also increase, and $\tau_{u1} < \tau_{u2} < \tau_{u3}$, τ_{u3} is much larger than τ_{u1} and τ_{u2} when the height is high. From equations (3), (4), (5) we know $\tau_{u3} = H \cdot \tau_{u2}$, $\tau_{u2} = \frac{\sum_{i=1}^{H} n_i}{H} \tau_{u1}$, the expected mean number of $\frac{\sum_{i=1}^{H} n_i}{H}$ is $\frac{\frac{N}{2}H}{H} = \frac{N}{2}$. This confirms that the highest performing strategies, if it can be implemented, is to search all files in parallel.

2.1.2 Multiple Signatures Can Exist in One File

One should search the whole file to find all of the signatures in each file. If there are multiple signatures in one file, the expected search time is $2\overline{X}$ (Sections 3.1.2 and 3.2.2), M_i can be larger than $M_{i-1} \cdot n_i$, and we assume that those M_i signatures exist in N_i files (each of N_i files with at least one signature).

Only one node in the tree commands all the nodes: The expected search time in layer i is $N_{i-1} n_i \cdot 2\overline{X}$, for N_i files with signatures, $N_i \leq M_i$. The total expected search time is:

$$\acute{T}_{u1} = \sum_{i=1}^{H} N_{i-1} n_i \cdot 2\overline{X}. \tag{6}$$

Every root node of each local tree commands the nodes below it: Again the levels are searched sequentially, within each subtree the search is sequential but all subtrees at the same level are searched in parallel. The expected search time in layer i is $n_i \cdot 2\overline{X}$. The total expected search time is:

$$\acute{T}_{u2} = \sum_{i=1}^{H} n_i \cdot 2\overline{X}. \tag{7}$$

The speedup is:

$$\acute{\tau}_{u1} = \frac{\sum_{i=1}^{H} N_{i-1} n_i \cdot 2\overline{X}}{\sum_{i=1}^{H} n_i \cdot 2\overline{X}} = \frac{\sum_{i=1}^{H} N_{i-1} n_i}{\sum_{i=1}^{H} n_i}. \tag{8}$$

All the files in every level are searched in parallel, levels are searched sequentially: The same case as before, $\acute{T}_{u3} = H \cdot 2\overline{X}$. The speedup is $\acute{\tau}_{u2} = \frac{\sum_{i=1}^{H} N_{i-1} n_i}{H}$.

All the files in the tree are searched in parallel: The same case as before, $\acute{T}_{u4} = 2\overline{X}$. The speedup is $\acute{\tau}_{u3} = \sum_{i=1}^{H} N_{i-1} n_i$.

The speedup is plotted in Fig. 2. As a baseline result, it is assumed that the distribution of signatures in every file is a Poisson distribution: $P(m = k) = \frac{\lambda^k}{k!} e^{-\lambda}$ (m is the number of signatures in every file). We can see from Fig. 3 that when λ goes up from 1 to 20, the speedup $\acute{\tau}_{u1}$ increases quickly first and then saturates. That is because initially the number of subtrees needed to be searched increased and later one winds up searching all, but not more than all, files in a level, then the speedup saturates.

2.2 Number of Signatures is Known

When the number of signatures is known, the searching process will stop as soon as the last signature is found,

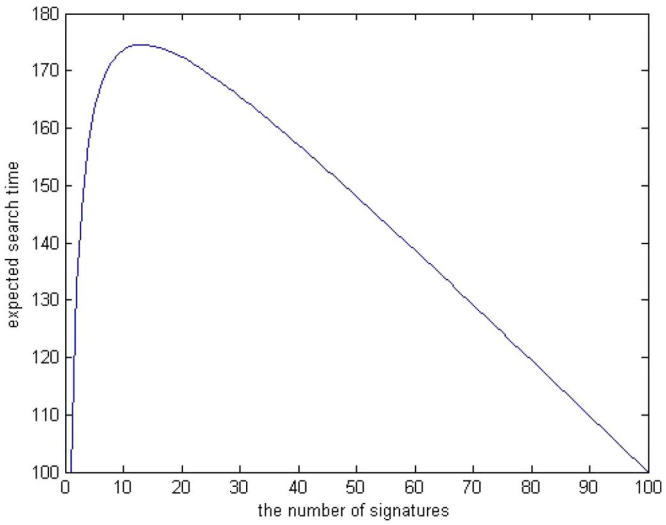


Fig. 3. Single level tree when the number of signatures is known.

thus the time should be shorter than that of the case where the number of signatures is unknown. Two cases are discussed.

2.2.1 Each File Has at Most One Signature

In this case assume that there are m signatures in n files in a single level of the tree. Each file can have at most one signature. Each node holds one file. We focus on the last signature because the searching process will not stop until the last signature is found, thus the search time depends on the position of the last signature. Because there are m signatures and every file can have at most one signature, the last signature should be in the $m_{th}, (m+1)_{th}, \ldots, n_{th}$ file. If the last signature is in the $(m-1)_{th}$ or even a file to the left of m_{th} file, there will be multiple signatures in some files, which is a contradiction. The probability that the last signature is in the m_{th} file is (there are $m-1$ signatures in $m-1$ files)

$$P_m = \frac{\binom{m-1}{m-1}}{\binom{n}{m}}. \tag{9}$$

And the expected search time is

$$t_m = P_m(m\overline{X}). \tag{10}$$

The probability that the last signature is in the i_{th} file is (there are $m-1$ signatures in $i-1$ files) for $i \geq m-1$ is

$$P_i = \frac{\binom{i-1}{m-1}}{\binom{n}{m}}. \tag{11}$$

And the expected search time is

$$t_i = P_i \cdot \left(m\overline{X} + (i-m)2\overline{X}\right) = \frac{\binom{i-1}{m-1}}{\binom{n}{m}}(2i-m)\overline{X}. \tag{12}$$

Thus, the expected search time T_s for this single level is

$$T_{n,m} = \sum_{i=m}^{n} t_i = \sum_{i=m}^{n} \frac{\binom{i-1}{m-1}}{\binom{n}{m}}(2i-m)\overline{X}. \tag{13}$$

We plot the expected search time as the number of signatures increases. In Fig. 3 the number of files $n = 100$, and we can conclude that when $m = 13$, the expected search time is the largest.

We derive the expression for the maximum value of the function of Fig. 3 in the supplementary file which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/258.

$$\frac{\mathrm{d}}{\mathrm{d}m}T_{m,n} = \frac{2n - m^2 - 2m + 1}{(m+1)^2} \cdot \overline{X} = 0 \tag{14}$$

Then we obtain $m = \sqrt{2n+2} - 1$. If $n = 100$, then equation (14) will be $\frac{\mathrm{d}}{\mathrm{d}m}T_{m,n} = \frac{200 - m^2 - 2m + 1}{(m+1)^2}$, let $\frac{\mathrm{d}}{\mathrm{d}m}T_{m,n} = 0$, then we get the maximum expected search time occur when $m = 13.21$ which is close to $m = 13$. For this graph if a file has a signature the expected search time is $\overline{X}$, and if there is no signature, it is $2\overline{X}$. Initially the mean time to find all signatures increases as more signatures are located closer to the last file. Eventually though as the number of signatures increases most files have a signature forcing the average search time per file closer to $\overline{X}$ rather than $2\overline{X}$ and the curve decreases.

Up to this point in this section the expected time for one single level has been solved. Next the expected time to search an H level tree is considered. There are four cases just as in the case where the number of signatures is unknown.

Only one node in the tree commands all the nodes: Assume that there are $m_{i,j}$ signatures in the j_{th} subtree in layer i, and $M_i = \sum_{j=1}^{M_{i-1}} m_{i,j}$, M_i stands for the total number of signatures in the i_{th} layer. Then for the j_{th} subtree in layer i, the expected search time $T_{n_i,m_{i,j}} = \sum_{i=m_{i,j}}^{n_i} \frac{\binom{i-1}{m_{i,j}-1}}{\binom{n_i}{m_{i,j}}}(2i - m_{i,j})\overline{X}$. The total expected search time is thus:

$$T_1 = \sum_{i=1}^{H} \sum_{j=1}^{M_{i-1}} T_{n_i,m_{i,j}}. \tag{15}$$

Assume $m_{1,1} = m_1 = M_1$ and $M_0 = 1$.

Every root node at local tree to command the nodes below it: Again, the levels are searched sequentially, within each subtree the search is sequential. However all subtrees at the same level are searched in parallel. For a given layer, the search time in this layer is the longest expected search time in the layer's subtrees. Assume that in the i_{th} layer the largest expected search time is $max[T_{n_i,m_{ij}}]$ (j is from 1 to M_{i-1} while M_i has been defined before), The total time to search the entire tree is:

$$T_2 = \sum_{i=1}^{H} \max\left[T_{n_i,m_{ij}}\right]. \tag{16}$$

The speedup is:

$$\tau_1 = \frac{\sum_{i=1}^{H}\sum_{j=1}^{M_{i-1}} T_{n_i,m_{ij}}}{\sum_{i=1}^{H} \max\left[T_{n_i,m_{ij}}\right]}. \tag{17}$$

For the fully homogeneous case where $n_1 = n_2 = \ldots = n_H = n$ and $m_1 = m_{2,1} = \ldots = m_{H,1} = \ldots = m$, the number of signatures in every subtree is the same in every layer in the first case. Then

$$\tau_1 = \frac{\sum_{i=1}^{H} m^{i-1} T_{n,m}}{H \cdot T_{n,m}} = \frac{m^H - 1}{H(m-1)}. \tag{18}$$

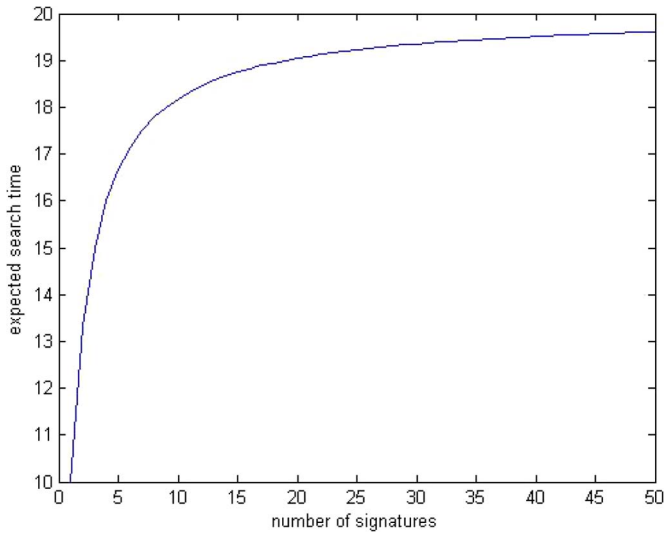


Fig. 4. The expected search time when the number of signatures is known and multiple signatures can exist in one file for a single level tree.

All the files in every level are searched in parallel, levels are searched sequentially: Discussed as before, $T_3 = H\overline{X}$ and the speed up compared with case 2.2.2.1 is

$$\tau_2 = \frac{\sum_{i=1}^{H}\sum_{j=1}^{M_{i-1}} T_{n_i,m_{i,j}}}{H\overline{X}}. \tag{19}$$

For the fully homogeneous case that $n_1 = n_2 = \ldots = n_H = n$ and $m_1 = m_{2,1} = \ldots = m_{H,1} = \ldots = m$, $\tau_2 = \frac{m^H-1}{H(m-1)} T_{n,m} = \frac{m(m^H-1)(2n-m+1)}{H(m^2-1)}$.

All the files in the tree are searched in parallel: Discussed as before $T_4 = \overline{X}$, and the speedup $\tau_3 = \frac{\sum_{i=1}^{H}\sum_{j=1}^{M_{i-1}} T_{n_i,m_{i,j}}}{\overline{X}}$. For the fully homogeneous case that $n_1 = n_2 = \cdots = n_H = n$ and $m_1 = m_{2,1} = \cdots = m_{H,1} = \cdots = m$, then $\tau_3 = \frac{m^H-1}{H(m-1)} T_{n,m} = \frac{m(m^H-1)(2n-m+1)}{m^2-1}$.

This is apparent that $\tau_3 > \tau_2 > \tau_1$, because for the fully homogeneous case, $\tau_3 = H\tau_2$, $\tau_2 = \frac{m(2n-m+1)}{m+1}\tau_1$, absolutely $\frac{m(2n-m+1)}{m+1} > 1$.

2.2.2 Multiple Signatures Can Exist in One File

First we consider a single level tree which has n files and m signatures. We combine all the files logically in one level into a single file. We also assume that the capacity of one file is L bits, so that the length of the combined file is $n \cdot L$. The position of the signatures is $X_1, X_2, \ldots, X_m$, $(nL \geq X_1, X_2, \ldots, X_m \geq 0)$, $X \sim U(0, nL)$.

$$P(X_{max} = x) = \binom{m}{1} \cdot \frac{1}{nL} \cdot \left(\frac{x}{nL}\right)^{m-1}. \tag{20}$$

The expected length of the last signature position:

$$\overline{X_{max}} = \int_0^{nL} \frac{m}{nL} \cdot \left(\frac{x}{nL}\right)^{m-1} \cdot x dx = \frac{m}{m+1} \cdot nL. \tag{21}$$

The search inverse speed is w s/bit. Then the expected time we need to search one single level is $\acute{T}_{n,m} = \overline{X_{max}} \cdot w = \frac{m}{m+1} \cdot n \cdot L \cdot w$, here $L \cdot w = 2\overline{X}$, thus

$$\acute{T}_{n,m} = \frac{2m}{m+1} n\overline{X}. \tag{22}$$

This is plotted in Fig. 4. Here the number of files is 10 and the maximum number of signatures is 50. From the figure we can see that the expected search time will increase and saturate as the number of signatures increase. That can be seen from Equation (22), as m increases, $\acute{T}_{n,m}$ will be close to $2n\overline{X}$.

Until now in this section we have solved the expected time in one single level, now we consider the expected time we need to search the H level tree. Assume that the m_i signatures are in $\acute{n}_i$ files. Apparently $\acute{n}_i \leq m_i$. The four cases are discussed as below.

Only one node in the tree commands all the nodes: For the i_{th} layer, there are N_{i-1} files that should be searched, there are $m_{i,1}, m_{i,2}, \ldots, m_{i,N_{i-1}}$ signatures in $\acute{n}_{i,1}, \acute{n}_{i,2}, \ldots, \acute{n}_{i,N_{i-1}}$ files, $N_i = \sum_{j=1}^{N_{i-1}} \acute{n}_{i,j}$. The total time is

$$\acute{T}_1 = \sum_{i=1}^{H}\sum_{j=1}^{N_{i-1}} \acute{T}_{n_i,m_{i,j}}. \tag{23}$$

Note that $N_0 = 1$ and $\acute{n}_{i,j} \leq m_{i,j}$.

Every root node at local tree to command the nodes below it: In this case the search time in the i_{th} layer is the time to search one of the subtrees which has the largest number of signatures (like the case discussed before). The total search time is:

$$\acute{T}_2 = \sum_{i=1}^{H} \max\left[\acute{T}_{n_i,m_{i,j}}\right]. \tag{24}$$

Note that j is from 1 to N_{i-1}. The speed up is:

$$\acute{\tau}_1 = \frac{\sum_{i=1}^{H}\sum_{j=1}^{N_{i-1}} \acute{T}_{n_i,m_{i,j}}}{\sum_{i=1}^{H} \max\left[\acute{T}_{n_i,m_{i,j}}\right]}. \tag{25}$$

All the files in every level are searched in parallel, levels are searched sequentially: Discussed as before, $\acute{T}_3 = 2H\overline{X}$. The speedup compared with case 2.2.2.1 is

$$\acute{\tau}_2 = \frac{\sum_{i=1}^{H}\sum_{j=1}^{N_{i-1}} \acute{T}_{n_i,m_{i,j}}}{2H\overline{X}}. \tag{26}$$

All the files in the tree are searched in parallel: The total search time in this case is $2\overline{X}$. The speedup here is:

$$\acute{\tau}_3 = \frac{\sum_{i=1}^{H}\sum_{j=1}^{N_{i-1}} \acute{T}_{n_i,m_{i,j}}}{2\overline{X}}. \tag{27}$$

The speedup figure for these cases has been plotted in Fig. 5. From the figure we can see that $\acute{\tau}_3 > \acute{\tau}_2 > \acute{\tau}_1$. The reason is similar to the case in 2.2.1.

3 Mesh Networks: Store and Forward

A regular two dimensional mesh network of processors [11] is shown in Fig. 6. It is a commonly used interconnection network. In this network structure each processor is located in the corners of four rectangles and has four neighbors. The central processor is called the originator, it can communicate information or transport data to its four

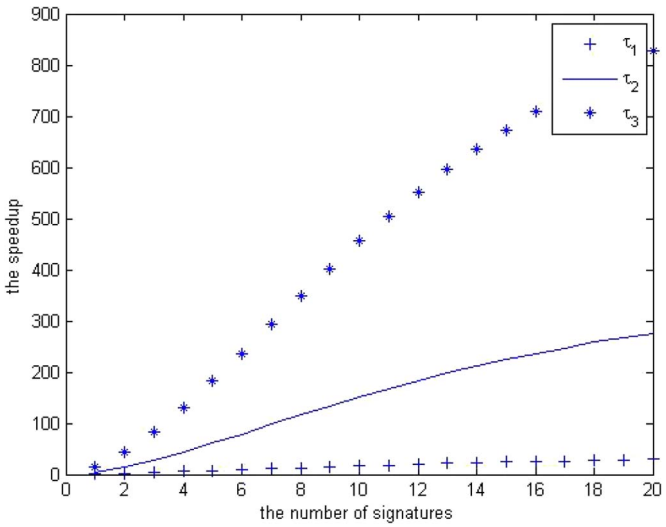


Fig. 5. Speedup when there can be multi-signatures in one file and the number of signatures in a level is known.

neighboring nodes. As in trees, the central node is assumed to be layer 0, and its four neighboring nodes in layer 1, the nodes which are neighbors to the nodes in layer 1 but not in layer 0 are in layer 2, ..., the nodes which are neighbors to the nodes in layer i but not in layer $i-1$ are in layer $i+1$, as shown in the mesh structure. A node can only send messages or data to its neighboring nodes. If the node in the north of layer 2 wants to send a message to the node in layer 0, first it should send message to the node in the north top of layer 1, then the message will be transferred to layer 2.

Assume that there are N layers (as shown in Fig. 6) and there is at most one signature in every node in a mesh network. There are $4i$ nodes in the i_{th} layer. Different from the tree network, the case that every node can have a signature will be discussed in Section 3.1. That is, if a node has no signature, its children nodes can still have signatures. Assume the average time to search every node in the mesh network is the same: $\overline{X}$, and again there is at most one signature in one file.

3.1 Every File Can Have Its Own Signature Regardless of the Upper Layer Node

3.1.1 The Number of Signatures Is Unknown

In this case every node should be searched. There are three cases.

Only one node in the network commands all the nodes: The central node which is in layer 0 commands each node to search for signatures. As soon as the central node sends a message to command one node to start to search its file, the central node will wait until it receives a message indicating whether there is any signature in the file. All of the nodes in the mesh network are searched one by one. It is apparent that the searching time is the longest, the searching time in the i_{th} layer is $4i \cdot \overline{X}$. The total expected search time is:

$$T_1 = 4\overline{X} + 8\overline{X} + \ldots + 4N\overline{X} = 2N(N+1)\overline{X}. \quad (28)$$

Every root node of each local network commands the node below it: The searching time in layer one is $\overline{X}$. For layer two, there are 8 nodes to be searched. We assign two nodes in layer two to be searched by each node in layer one, the search time should be $2\overline{X}$. For layer three, there are two cases, the first case is for the four nodes in the upmost, or downmost, or leftmost, or rightmost direction. The expected search time for these four nodes is $3\overline{X}$, because there are three neighboring nodes in their lower layer; and for the second case (the remaining nodes), there are only two neighboring nodes in their lower layer. Thus the expected search time is $2\overline{X}$. However, we can eliminate $\overline{X}$ searching time for both cases, because every node in layer three except the four top nodes in four directions have two upper nodes (different from the tree network). If one node for the node in layer three is commanded by one node in layer two, the other node in layer two need not command it to search for a signature. For layer three, the searching time is $2\overline{X}$, the same for layer four, five, ... The total expected search time is $T_2 = 4\overline{X} + 2\overline{X} + 2\overline{X} + \cdots + 2\overline{X} = 4\overline{X} + 2(N-1)\overline{X} = (2N+2)\overline{X}$.

All the files in every layer are searched in parallel, layers are searched sequentially: Like the third case in the tree network, the search time in every layer is $\overline{X}$. The total time is $T_3 = N\overline{X}$.

All the files in the mesh network are searched in parallel: The expected search time in this case is only $T_4 = \overline{X}$.

Then we calculate the speedup in these four cases: $\tau_1 = \frac{T_1}{T_2} = N$, $\tau_2 = \frac{T_1}{T_3} = 2(N+1)$, $\tau_3 = \frac{T_1}{T_4} = 2N(N+1)$, and the speedup figure has been plotted in Fig. 7. From the figure we know that $\tau_3 > \tau_2 > \tau_1$, that is because $\frac{\tau_2}{\tau_1} = \frac{2(N+1)}{N} > 2$ and $\frac{\tau_3}{\tau_2} = N$.

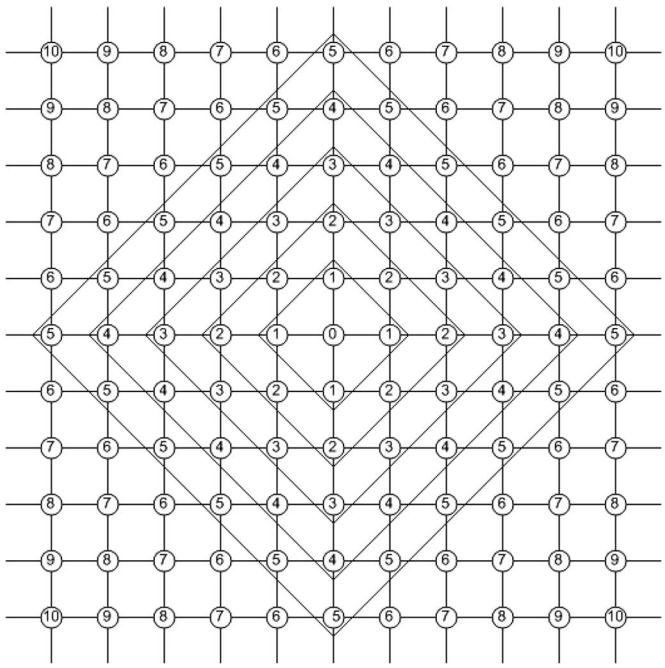


Fig. 6. Mesh network.

3.1.2 The Number of Signatures Is Known

Only one node in the network commands all the nodes: This case is mostly similar to the case in the tree network: assume there are m_i signatures in the i_{th} layer, we focus on the last signature. There are $4i$ files in the i_{th} layer, the probability that the last signature is in the j_{th} file in the i_{th} layer is:

$$P_j = \frac{\binom{j-1}{m_i-1}}{\binom{4i}{m_i}}. \quad (29)$$

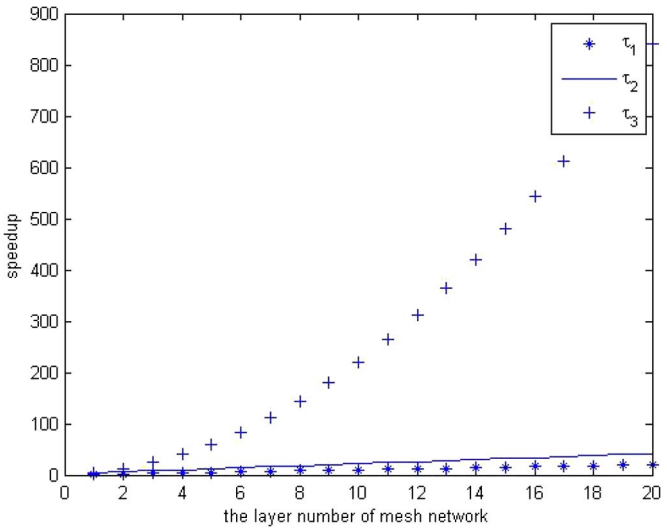


Fig. 7. Speedup when the number of signatures is unknown in a mesh network and each node can have at most one signature.

The expected searching time in the i_{th} layer is:

$$\acute{T}_i = \sum_{j=m_i}^{4i} \frac{\binom{j-1}{m_i-1}}{\binom{4i}{m_i}}(2j - m_i)\overline{X}. \tag{30}$$

The total time to search the mesh network is $\acute{T}_1 = \sum_{i=1}^{N} \acute{T}_i$. The time should be less than T_1.

Every root node of each local network commands the nodes below it: For layer one, the expected search time is the same as $T_{2.1} = \sum_{j=m_1}^{4} \frac{\binom{j-1}{m_1-1}}{\binom{4}{m_1}}(2j - m_1)\overline{X}$. For layer two, three and the next layers, there are two types of nodes. The first type of nodes is those nodes which have three children nodes in their lower layer, from Fig. 2 the first type of nodes are in the top most position of east, west, south, north four directions. The other type of nodes are the other nodes, which only have two children nodes in their lower layers. However, the searching time for the two types nodes are not $2\overline{X}$ and $3\overline{X}$, because every node except the four top nodes have two parent nodes. Assume that in the first $\overline{X}$ searching time all the nodes except the four top nodes are searched, and for the parent node only one node commands one node. If all the signatures are found, the searching time in this layer is $\overline{X}$. If there are some signatures in the top four nodes, we need to add $\overline{X}$ more time to find the signature, then the searching time should be $2\overline{X}$. The probability that there are no signatures in the top four nodes in the i_{th} layer is:

$$P_{2.i} = \frac{\binom{4i-4}{m_i}}{\binom{4i}{m_i}}. \tag{31}$$

The total time of the mesh network in this case is:

$$\acute{T}_2 = \acute{T}_1 + \sum_{i=2}^{N}\left(P_{2,i}\overline{X} + (1 - P_{2,i})2\overline{X}\right). \tag{32}$$

All the files in every layer are searched in parallel, layers are searched sequentially: The same as the last case we have discussed, the total time is $\acute{T}_3 = N\overline{X}$.

All the files in the mesh network are searched in parallel: The total time is $\acute{T}_4 = \overline{X}$.

Now we calculate the speedup for these four cases. $\acute{\tau}_1 = \frac{\acute{T}_1}{\acute{T}_2} = \frac{\sum_{i=1}^{N}\acute{T}_i}{\acute{T}_1 + \sum_{i=2}^{N} P_{2,i}\overline{X} + (1-P_{2,i})2\overline{X}}$, while $\acute{T}_i$ and $P_{2,i}$ have been defined before (see Equation(30) and (31)). $\acute{\tau}_2 = \frac{\acute{T}_1}{\acute{T}_3} = \frac{\sum_{i=1}^{N}\acute{T}_i}{N\overline{X}}$, $\acute{\tau}_3 = \frac{\acute{T}_1}{\acute{T}_4} = \frac{\sum_{i=1}^{N}\acute{T}_i}{\overline{X}}$. The speedup figure has been plotted in Fig. 8. From the figure we can know that $\acute{\tau}_3 > \acute{\tau}_2 > \acute{\tau}_1$. That is because $\acute{\tau}_3 = N \cdot \acute{\tau}_2$, $\frac{\acute{\tau}_2}{\acute{\tau}_1} = \frac{\acute{T}_1 + \sum_{i=2}^{N} P_{2,i}\overline{X} + (1-P_{2,i})2\overline{X}}{N\overline{X}} > 1$.

3.2 Only the Nodes Whose Parent Node Has a Signature Can Have a Signature

In this case those nodes whose parent node do not have a signature, will also have no signatures. Only those nodes whose parent node has a signature need to be searched. This case is so complex that one can not find a general equation to solve it. We simulated 100,000 mesh networks with different probabilities of finding signatures in the files, and calculated the search times. In this section we only discuss the case that the number of signatures is unknown.

3.2.1 *Only One Node in the Network Commands All the Nodes*

The mesh network structure is divided into four parts: up-left, up-right, down-left and down-right. The nodes in these four parts will search their children nodes in four different directions, which is also up-left, up-right, down-left and down-right. It is probable that those nodes, whose parent node has a signature, will have signatures. The nodes that are not in the top position of every layer have two parent nodes, the case that either of them or both of them have a signature will lead the child node to possibly have a signature. We can get the mesh structure by creating the signature distribution layer by layer. The next step is to

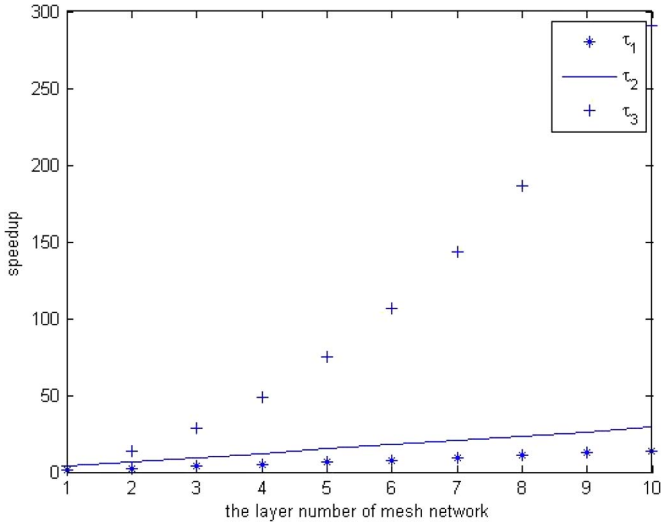


Fig. 8. Speedup when the number of signatures is known in a mesh network and there is at most one signature in each node.

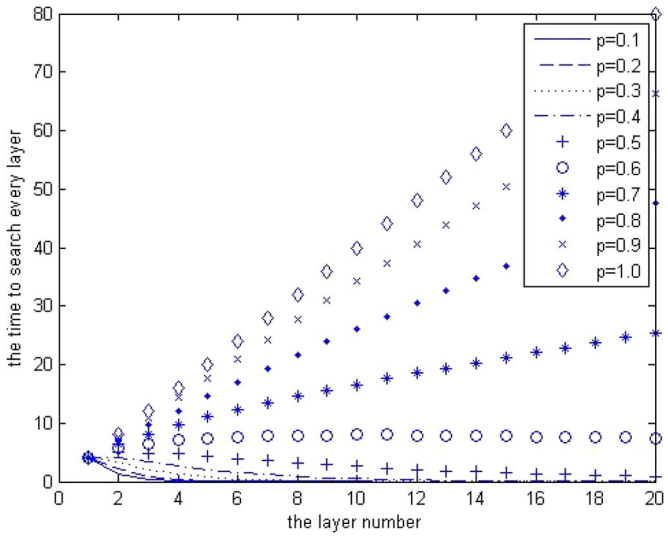


Fig. 9. Searching time for the mesh network in case 3.2.1 as signature probability is varied.

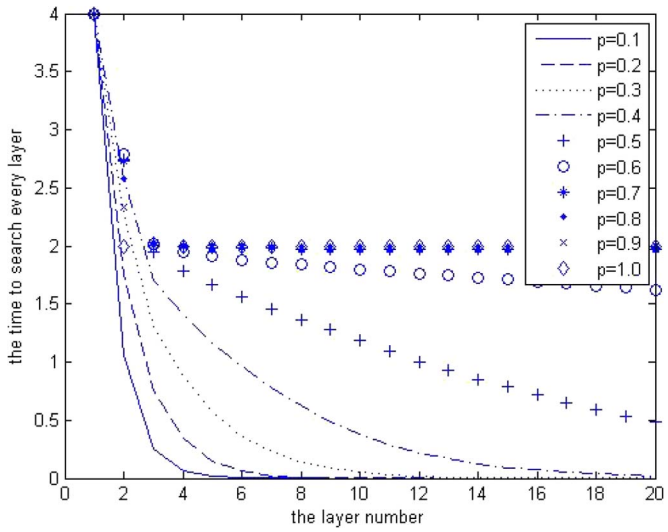


Fig. 10. Searching time for the mesh network in case 3.2.2 as signature probability is varied.

calculate the search time in every layer, which depends on the signature distribution in the upper layer. The four top nodes have three children nodes, and the other nodes have two children nodes. Thus the total search time for a certain layer is the number of top nodes, which have a signature to be searched, times $3\overline{X}$, plus the number of the other nodes, which have a signature, times $2\overline{X}$. For this algorithm Fig. 9 shows the search time for every layer as the probability of signatures increases. Smaller values of p result in the signature searching processing stopping earlier as irregular signature free boundaries form around the mesh network that prevent further signature searching.

Fig. 9 shows that when the probability of finding a signature in every node is higher, the expected time to search every layer will be longer. When the probability of a signature is exactly 1, that means every node will have a signature, the time to search the i_{th} layer is $4i$. When the probability of a signature is equal or less than 0.5, the number of nodes which have signatures will decrease as the layers increase until no node will have signatures in a certain layer. When the probability of signature is equal to 0.6, there are almost always nodes which have signatures and the number of nodes which have signatures in every level remains almost constant. When the probability of signature is equal or more than 0.7, the number of nodes which have signatures increases.

3.2.2 Every Root Node of Each Local Network Commands the Nodes Below it

For the four top nodes in the east, west, north, and south directions, if both of the two neighboring nodes in their layer have no signatures (i.e they will not command their lower layer node to search their files), the searching time for the top nodes is $3\overline{X}$. If one of the neighboring nodes or both of them have signatures, we can assign the node whose parent nodes are the top node and its neighboring node to be commanded by the node neighboring the top node. Thus the searching time will be cut to $2\overline{X}$. For the other node, the maximum searching time is $2\overline{X}$, when these nodes have signatures. All the local mesh networks (the node and its children nodes) search their own nodes in parallel. The searching time for a certain layer depends on the signature distribution of the top node of the upper layer, as mentioned before, it will be $3\overline{X}$, $2\overline{X}$, or 0 (there are no signatures in the whole parent nodes). We simulate this case in Fig. 10.

From Fig. 10, when the probability of a signature is equal or less than 0.6, the searching time is shorter and shorter until we need not search the signature in the lower layer as the layer is larger. That's because smaller values of p result in the signature searching processing stopping earlier as irregular signature free boundaries form around the mesh network that prevent further signature searching. When the probability of signature is equal or higher than 0.7, for those higher layer the searching time is exactly $2\overline{X}$. That is because the probability that the top node has a signature and meanwhile the two neighboring nodes have no signatures is almost impossible. Thus the searching time for those high layers is exactly $2\overline{X}$.

The speedup for case 3.2.1 and case 3.2.2 (speedup 1) has been depicted in Fig. 11. From it we can conclude that when the probability of a signature is below 0.6, the speedups are saturated at less than 4. We can not reduce much searching time by using the method in case 3.2.2 when the probability of a signature is below 0.6 as the number of layers increases. That is because when the probability of a signature is below 0.6, the searching time for every layer in case 3.2.1 will decrease as the number of layers increases (see Fig. 9). While the probability of a signature is more than 0.6, the speedup rises much faster. That is because in Fig. 9 we know the the searching time will go up quickly, while the searching time will be $2\overline{X}$ in Fig. 10.

3.2.3 All the Files in Every Layer Are Searched in Parallel, Layers Are Searched Sequentially

The searching time for every layer is $\overline{X}$. The total search time is thus $N\overline{X}$. The speedup for 3.2.1 and case 3.2.3 (speedup 2) has been depicted in Fig. 12. It can be seen from Fig. 12 the speedup is larger than the speedup in Fig. 11

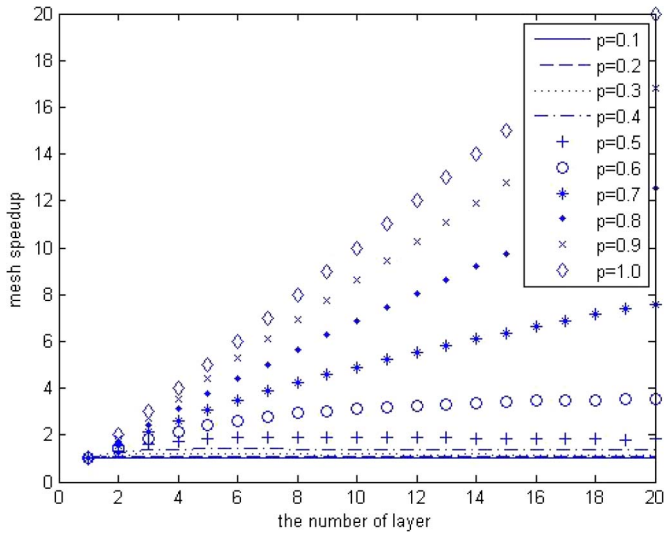


Fig. 11. Speedup 1 when the number of signatures is unknown and only the nodes whose parent node has a signature can have a signature in the mesh network.

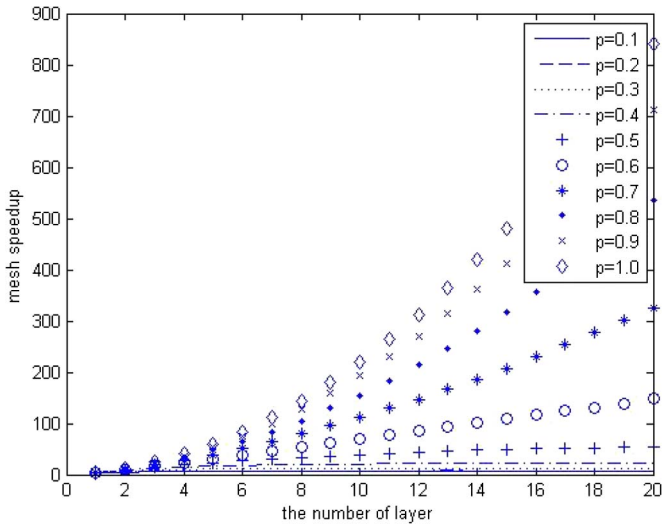


Fig. 13. Speedup 3 when the number of signature is unknown and only the nodes whose parent node has a signature can have a signature in mesh network.

when the probability of signature is larger than 0.6, for the searching time for every layer is the same $\overline{X}$. When the probability of signature is lower than 0.6, the same as Fig. 11, the speedup does not increase when the number of layers increases.

3.2.4 *All the Files in the Mesh Network Are Searched in Parallel*

The expected search time in this case is only $\overline{X}$. The speedup (speedup 3) for case 3.2.4 and case 3.2.1 has been plotted in Fig. 13. It can be seen that the speedup increases much faster as the number of layers increase, than the speedup in Fig. 12 when the probability of a signature is larger than 0.6, and when the probability of a signature is lower than 0.6, the speedup does not increase as the number of layers increase.

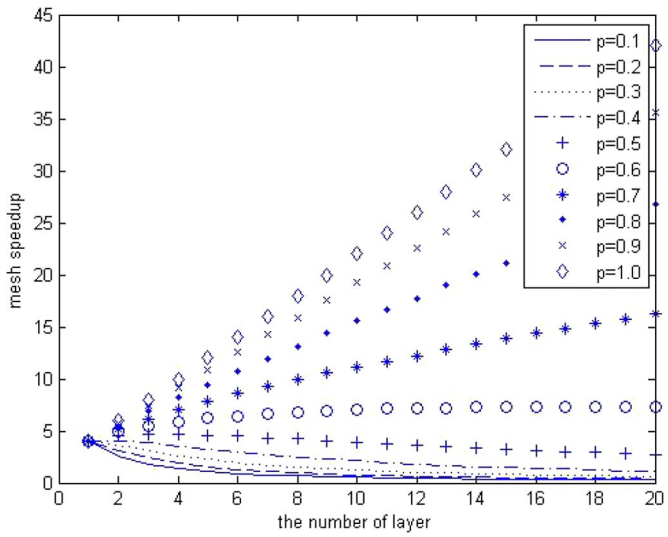


Fig. 12. Speedup 2 when the number of signatures is unknown and only the nodes whose parent node has a signature can have a signature in the mesh network.

4 Searching Time Comparison for Different Types of Network

The four types of networks considered are tree networks, mesh networks: store and forward, mesh networks: circuit switched and wormhole routing (discussed in the supplementary file available online) and hypercubes (discussed in supplementary file available online). Each of these has advantages and disadvantages. Now we want to compare their searching time if the total number of nodes is fixed: N. For the tree network, assume that the number of nodes in every subtree, n, is the same, which is a homogeneous case. The height of the tree should be: $1+n+n^2+\ldots+n^{H-1}=N$, then $H=\log_n((n-1)N+1)$. The total searching time for case 3.1.1.2 is $T_1=nH\overline{X}=n\log_n((n-1)N+1)\overline{X}$. The searching time is plotted in Fig. 14 (here $N=100$) for these four types of network. We can see that when the number of nodes in every subtree increases, the searching time also increases. For the store-and-forward mesh

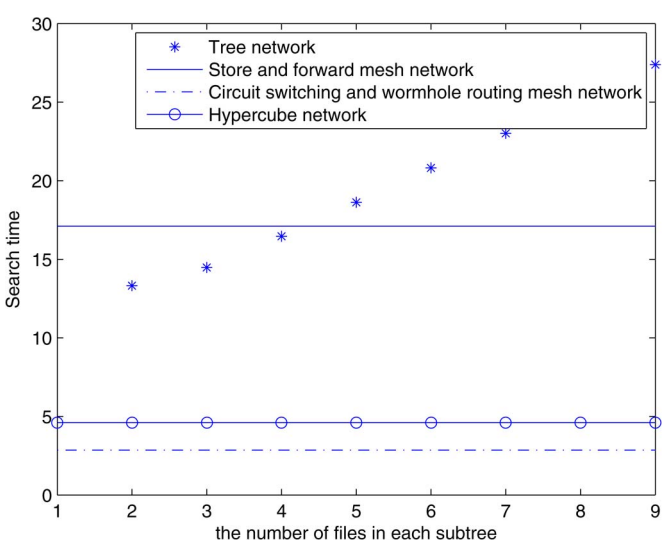


Fig. 14. Four types of networks comparison.

network, the number of layers should be: $1+4+\cdots+4(H-1)=N$, then $H=\frac{1+\sqrt{2N-1}}{2}$. The searching time should be $T_2=(2H+2)\overline{X}=(3+\sqrt{2N-1})\overline{X}$. For the circuit switching and wormhole routing mesh network, the searching time is $T_3=\log_{p+1}N\overline{X}$. For the hypercube network, the searching time is $T_4=log_2N\overline{X}$. In Fig. 14, we can see that the searching time for circuit switching and the wormhole routing mesh network is short, but this type network is hard to realize. The searching time for hypercubes is also very short, and this type can be easily realized in low dimensionality. For the tree network and store-and-forward network, which are easy to realize, if the number of nodes in every subtree is more than 4, the searching time for the store and forward mesh network is shorter, for the tree network when the number of nodes in every subtree is less than or equal to 4, the searching time for the tree network is shorter.

5 Conclusion

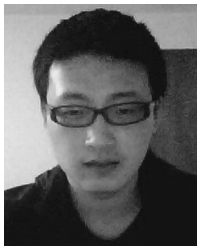
It has been demonstrated that the expected search time for signatures in a wide variety of search scenarios for tree, mesh and hypercube networks, where load distribution time is not considered, can be calculated either analytically or through simulation. This should also be possible for other types of interconnection networks. Future research should consider other types of file structures or statistical assumptions. This work is of interest in a wide variety of applied areas involving signature searching.

Acknowledgment

The authors acknowledge the support of DOE grant DE-SC0003361.

References

[1] K. Ko and T. Robertazzi, ''Signature Search Time Evaluation in Flat File Databases,'' *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 2, pp. 493-502, Apr. 2008.
[2] Y. Kyong and T.G. Robertazzi, ''Greedy Signature Processing with Arbitrary Location Distributions: A Divisible Load,'' *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 4, pp. 3027-3041, Oct. 2012.
[3] R. Baeza-Yates, ''Algorithms for String Searching,'' *AcM sIGIR Forum*, vol. 23, no. 3/4, pp. 34-58, Apr. 1989.
[4] G. Navarro, ''A Guided Tour to Approximate String Matching,'' *ACM Comput. Surveys*, vol. 33, no. 1, pp. 31-88, Mar. 2001.
[5] P. Michailidis and K. Margaritis, ''String Matching Algorithms: Survey and Experimental Results,'' *Int'l J. Comput. Math.*, vol. 76, no. 4, pp. 411-434, Jan. 2001.
[6] Z. Galil and R. Giancarlo, ''Data Structures and Algorithms for Approximate String Matching,'' *J. Complex.*, vol. 4, no. 1, pp. 33-72, Mar. 1988.
[7] H. Kitakami, T. Shin-I, K. Ikeo, Y. Ugawa, N. Saitou, T. Gojobori, and Y. Tateno, ''Yamato and Asuka: DNA Database Management System,'' in *Proc. 28th Hawaii Int'l Conf. Syst. Sci.*, 1995, vol. 5, pp. 72-80.
[8] M. Hoffman and D. Carver, ''Reverse Engineering Data Requirements,'' in *Proc. IEEE Aerosp. Appl. Conf.*, 1996, vol. 2, pp. 269-277.
[9] M. Lubeck, D. Geppert, and K. Nienartowicz, ''An Overview of a Large-Scale Data Migration,'' in *Proc. 20th IEEE 11th NASA Goddard Conf. MSST*, 2003, pp. 49-55.
[10] T. Robertazzi, *Networks and Grids: Technology and Theory*. New York, NY, USA: Springer-Verlag, 2010.
[11] M. Drozdowski, *Scheduling for Parallel Processing*. New York, NY, USA: Springer-Verlag, 2009.
[12] H. Casanova, A. Legrand, and Y. Robert, *Parallel Algorithms*. Boca Raton, FL, USA: CRC Press, 2009.
[13] W. Głazek, ''A Multistage Load Distribution Strategy for Three-Dimensional Meshes,'' *Cluster Comput.*, vol. 6, no. 1, pp. 31-39, Jan. 2003.
[14] M. Drozdowski and W. Glazek, ''Scheduling Divisible Loads in a Three-Dimensional Mesh of Processors,'' *Parallel Comput.*, vol. 25, no. 4, pp. 381-404, Apr. 1999.
[15] K. Li, ''Speed-Up of Parallel Processing of Divisible Loads on k-Dimensional Meshes and Tori,'' *Comput. J.*, vol. 46, no. 6, pp. 625-631, 2003.
[16] J. Błażewicz and M. Drozdowski, ''Performance Limits of Two-Dimensional Network of Load-Sharing Processors,'' *Found. Comput. Decision Sci.*, vol. 21, no. 1, pp. 3-15, 1996.
[17] T. Bjerregaard and S. Mahadevan, ''A Survey of Research and Practices of Network-On-Chip,'' *ACM Comput. Surveys*, vol. 38, no. 1, p. 1, 2006.
[18] E. Salminen, T. Kangas, V. Lahtinen, J. Riihimäki, K. Kuusilinna, and T. Hämäläinen, ''Benchmarking Mesh and Hierarchical Bus Networks in System-On-Chip Context,'' *J. Syst. Architect.*, vol. 53, no. 8, pp. 477-488, Aug. 2007.
[19] S. Stuijk, T. Basten, M. Geilen, A. Ghamarian, and B. Theelen, ''Resource-Efficient Routing and Scheduling of Time-Constrained Streaming Communication on Networks-on-Chip,'' *J. Syst. Architect.*, vol. 54, no. 3/4, pp. 411-426, Mar. 2008.



Zhongwen Ying received the BS degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2009. He is currently pursuing the PhD degree in Stony Brook University since 2009. His research interests are in the area of distributed and parallel computing and algorithm design and he is currently working on signature searching algorithm and divisible load theory.

Thomas G. Robertazzi received the BEE from the Cooper Union, New York, NY, in 1977 and the PhD degree from Princeton University, Princeton, NJ, in 1981. He is presently a Professor in the Department of Electrical and Computer Engineering at Stony Brook University, Stony Brook NY. In supervising a very active research group, he has published extensively in the areas of parallel processing and grid scheduling, telecommunications network planning, ATM switching, queueing and Petri networks. Prof. Robertazzi has also authored, co-authored or edited six books in the areas of networking, performance evaluation, scheduling and network planning. For eleven years Prof. Robertazzi was the faculty director of the Stony Brook Living Learning Center in Science and Engineering. For the past five years he has co-chaired the Stony Brook University Senate Research Committee. He is a Fellow of the IEEE.