A Note on Weighted Distributed Match-Making

Evangelos Kranakis

Centre for Mathematics and Computer Science
Kruislaan 413, 1098SJ Amsterdam, The Netherlands

Paul M. B. Vitányi

Centre for Mathematics and Computer Science
Kruislaan 413, 1098SJ Amsterdam, The Netherlands

ABSTRACT

This is a preliminary draft version of [Mathematical Systems Theory, 25(1992), 123-140].

In many distributed computing environments, processes are concurrently executed by nodes in a store-and-forward network. Distributed control issues as diverse as name-server, mutual exclusion and replicated data management, involve making matches between processes. The generic paradigm is a formal problem called ‘‘distributed match-making’’. We define multi-dimensional and weighted versions, and the relations between the two, and develop a very general method to prove lower bounds on the complexity as a trade-off between number of messages and ‘‘distributedness’’. The resulting lower bounds are tight in all cases we have examined. We present a success-stop version of distributed match-making that is analysed in terms of a weight distribution that in all cases results in approximately halving the (expected) number of messages required in the corresponding strategy that does not use these weights.

1. Introduction

A distributed system consists of computers (*nodes*) connected by a communication network. Nodes can communicate with each other through the network. There is no other communication between nodes. Distributed computation entails the concurrent execution of more than one process, each process being identified with the execution of a program on a computing node. Communication networks come in two types: broadcast

---

The second author did part of this work at the Laboratory for Computer Science, M.I.T., Cambridge, Mass.. He was supported in part by the Office of Naval Research under Contract N00014-85-K-0168, by the Office of Army Research under Contract DAAG29-84-K-0058, by the National Science Foundation under Grant DCR-83-02391, and by the Defence Advanced Research Projects Agency (DARPA) under Contract N00014-83-K-0125. A preliminary version of this paper appeared in: Proc. VLSI Algorithms and Architectures, 3rd Aegean Workshop on Computing (AWOC 89), *Springer Lecture Notes in Computer Science*, vol. 319, 1988, pp. 361-368.

networks and store-and-forward networks. In a broadcast network a message by the sender is broadcasted and received by all nodes, including the addressee. In such networks the communication medium is usually suited for this (like Ethernet). But here we are primarily interested in store-and-forward networks, where a message is routed from node to node to its destination. Such networks occur in the form of wide area networks like Arpanet, but also as the communication network of a single multicomputer. The necessary coordination of the separate processes in various ways constitutes distributed control. The situation gets more complicated by assuming that processes can migrate from host to host, for instance to balance the load in the system. We focus on a common aspect of seemingly unrelated issues in this area, such as name server, mutual exclusion and replicated data management.

Processes residing in different nodes may need to find each other, without knowing the host addresses of each other in advance. For example, in a name-server a client process wants to know the host address of a server process providing a particular service; in distributed mutual exclusion a process that wants to enter the critical section needs to know whether some other process wants to do so as well [7]. The common aspect of these problems of these problems is formalized in [4] as the paradigm 'distributed match-making.' That is, associate with each node a in the network two sets of network nodes, $P(a)$ and $Q(a)$, such that the intersection $P(a) \cap Q(b)$ is nonempty, for each ordered node pair (a, b). Such a pair P, Q is called a strategy. We want to minimize the average of $|P(a)| + |Q(b)|$, the average taken over all pairs (a, b) of nodes. ($|X|$ denotes the number of elements in a finite set X.) This average is related to the amount of *communication* (number of messages) involved in implementations of the distributed control issues mentioned. In terms of the name-server, a is a server that posts its whereabouts in all nodes of $P(a)$, and b is a client that looks for a particular service (as provided by a) in all nodes of the query set $Q(b)$. Nodes in $P(a) \cap Q(b)$ can establish contact between a and b by sending a message to a with the address of b. In distributed mutual exclusion the interpretation is about the same, except that there is no difference between client and server, $P(a) = Q(a)$, see [3], [4]. For application to replicated data management see [4]. We make the simplifying assumption that hosts do not crash, and the involved processes do not migrate from host to host, during execution of a match-making instance. The question of how to determine P and Q, and how they are to be set, is not addressed here.

Example. A match-making strategy $P, Q : N \to 2^N$ can be represented by an $|N| \times |N|$ matrix $\{r_{a,b}\}$, called the *rendez-vous matrix* in [4], where each entry $r_{a,b}$ is the set of nodes $P(a) \cap Q(b)$ at which node a posts and node b queries. Since we are interested in lower bounds, we can assume the entries to be singleton sets. For example, we have a network with a set of nodes $N = \{1, 2, 3\}$, and with rendez-vous matrix:

$$\begin{matrix} 2 & 3 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 2 \end{matrix}$$

A server at node 1 and a client at node 2 have node 3 as rendez-vous node. To make a match between node 1 and node 3 takes five messages (node 1 posts its address at nodes

1,2 and 3; node 3 queries nodes 1 and 2), while to make a match between a server at node 2 and a client at node 1 is cheaper at four messages. One and the same strategy can have a pair which costs a lot of messages to match and a pair which is very cheap. It seems therefore meaningful to measure the communication complexity of a strategy as the average number of messages to match a pair of nodes. In this example the average is $4\frac{1}{3}$ messages.

1.1. The Problem in the Two-Dimensional Case

We recall the basic definitions and relevant results of [4], to make this article self-contained. If U is a set, then $|U|$ denotes the number of elements, and 2^U denotes the set of all subsets of U. We are given a set of elements $U = \{1,2, \ldots, n\}$ and total functions

$$P,\ Q : U \rightarrow 2^U \quad ,$$

such that $|P(a) \cap Q(b)| = 1$ for all a,b, $1 \le a,b \le n$.

Question 1. Find a lower bound on the average of $|P(a)| + |Q(b)|$, the average taken over all ordered pairs $(a,b) \in U^2$. Investigate this lower bound when $|P(a)|$ and $|Q(b)|$ can be chosen freely, and when either one of $|P(a)|$ or $|Q(b)|$ has a prescribed value.

Question 2. If $S(i) = \{j : i \in P(j)\}$, then find trade-offs between the lower bound of Question 1, and the average number of elements (or worst case number of elements) in $S(i)$, the average taken over all i in U.

If the elements of $P(a)$ and $Q(b)$ are randomly chosen then the probability for any one element of U to be an element of $P(a)$ (or $Q(b)$) is $|P(a)|/n$ (or $|Q(b)|/n$). If $P(a)$ and $Q(b)$ are chosen independently then the probability for any one element of U to be an element in both $P(a)$ and $Q(b)$ is $|P(a)||Q(b)|/n^2$. Since there are n elements in U, the expected size of $P(a) \cap Q(b)$ is given by

$$E(|P(a) \cap Q(b)|) = \frac{|P(a)||Q(b)|}{n} \quad .$$

Therefore, to expect precisely one element in $P(a) \cap Q(b)$, we must have $|P(a)| + |Q(b)| \ge 2\sqrt{n}$. The above analysis holds for each ordered pair (a,b) of elements of U, since all nodes are interchangeable. Consequently, the minimal average value of $|P(a)| + |Q(b)|$ over all ordered pairs (a,b) in U^2 is $2\sqrt{n}$.

By deliberate choice of the sets $P(a)$ and $Q(b)$, as opposed to random choice, the result may improve in two ways.

(1) The intersection of $P(a)$ and $Q(b)$ with certainty contains one element, as opposed to an expectation of one element, and

(2) $|P(a)| + |Q(b)| < 2\sqrt{n}$ suffices, for selected pairs, or even on the average.

Option (2) is suggested by the fact that the elements of U need not be treated as symmetric. For instance, with one distinguished element in U we can get by with $|P(a)| + |Q(b)| = 2$ on the average.

1.1.1. Complexity

Denote the singleton set $P(a) \cap Q(b)$ by $r_{a,b}$, and call $r_{a,b}$ the *rendez-vous* element. (For convenience, identify a singleton set with the element it contains.)

Definition. The $n \times n$ matrix, R, with entries $r_{a,b}$ $(1 \le a,b \le n)$ is the *rendez-vous* matrix. Note that:

$$\bigcup_{b=1}^{n} r_{a,b} \subseteq P(a) \quad \& \quad \bigcup_{a=1}^{n} r_{a,b} \subseteq Q(b) \quad . \tag{M1}$$

By choice of $P(a)$'s and $Q(b)$'s we can always replace the inclusions in (M1) by equalities. We also say that R represents a *match-making strategy* between each ordered pair (a,b) of nodes in U. The interpretation is that a sends messages to all elements in $P(a)$, and b sends messages to all elements in $Q(b)$, to effect a match of the ordered pair (a,b) at $r_{a,b}$. In many applications we can assume that a node needs to send no messages to itself, which corresponds to empty elements $r_{a,a}$ on the main diagonal. This gives minor changes in the results below. For simplicity we do not make this assumption. Examples of *rendez-vous* matrices for different strategies, ranging from centralized via hierarchical to distributed, are given in the Appendix of [4]. We give two examples in the Appendix. The reader may find it useful to look at the examples before continuing.

1.1.2. Lower Bound

The *number of messages* $m(a,b)$ involved in the *match-making instance* associated with (a,b) is:

$$m(a,b) = |P(a)| + |Q(b)| \quad . \tag{M2}$$

We can determine the quality and complexity of a match-making strategy by the *minimum* of $m(a,b)$, or the *maximum* of $m(a,b)$. But the most significant measure appears to be the *average* of $m(a,b)$, for a,b ranging from 1 to n.

Definition. The *average* number of messages m of a match-making strategy (as determined by the *rendez-vous* matrix R) is:

$$m = \frac{1}{n^2} \sum_{a=1}^{n} \sum_{b=1}^{n} m(a,b) \quad . \tag{M3}$$

We call m the *communication* complexity of R. We denote by $m(n)$ the *optimal* communication complexity, i.e., $m(n)$ equals the minimum value of m associated with R, where R ranges over all $n \times n$ *rendez-vous* matrices. Distributed methods are preferable since they can tolerate failures and distribute the message load better than centralized ones. A question is how to express the trade-off between communication efficiency and distributedness of these algorithms? It appears that communication efficiency is intimately tied up with the frequencies with which the respective nodes occur in the *rendez-vous* matrix.

Define the *frequency* k_i of i in R as the number of times element i occurs as an entry in R, i.e., how often i is used as *rendez-vous* for an ordered pair (a,b) of elements

($1 \le a, b \le n$). Clearly,

$$\sum_{i=1}^{n} k_i = n^2 \quad . \tag{M4}$$

We call the n-tuple $(k_1, \ldots, k_n)$ the *distribution vector* of R, and we consider it as a measure for the *distributedness* of strategy R. Looking at two extremes we see that this makes sense. If there is an i such that $k_i = n^2$ and $k_j = 0$ for $j \neq i$, then the strategy is *centralized* If $k_i = n$ for all i, then we call the strategy *distributed*. Intuitively, the statistical variation of the k_i's measures the distributedness of a strategy in a single figure. We derive a lower bound (Proposition 2) on $m(n)$ expressed in terms of the k_i's. We show that this lower bound optimal for distribution vectors $(n, \ldots, n)$ and $(0,...,0,n^2,0,...,0)$, by exhibiting strategies R which achieve it. We conjecture that the lower bound is optimal for all distribution vectors. To prove Proposition 2, it is useful to proceed by way of Proposition 1. Not only is Proposition 1 combinatorially more interesting than Proposition 2, which is an easy corollary, but it also quantifies the optimal trade-off between the sizes of the P-sets and the Q-sets. We will generalize this in the present paper to obtain results on many-dimensional and weighted versions of distributed match-making.

Proposition 1. *Consider the rendez-vous matrix R as defined above. Then,*

$$\sum_{a=1}^{n} \sum_{b=1}^{n} |P(a)|\,|Q(b)| \geq \left(\sum_{i=1}^{n} \sqrt{k_i} \right)^2 \tag{M5}$$

Proposition 2.

$$m(n) \geq \frac{2}{n} \sum_{i=1}^{n} \sqrt{k_i} \quad .$$

For the proofs see [4] or below: these results are the special case of Theorem 1 with $s=2$ and $n_1 = n_2 = n$. It is not difficult to see that Propositions 1 and 2 hold *mutatis mutandis* for nonsquare matrices R. For totally *distributed* strategies they specialize to:

Corollary. *Let R be a rendez-vous matrix such that $k_1 = k_2 = \cdots = k_n = n$. Then:*

$$\sum_{a=1}^{n} \sum_{b=1}^{n} |P(a)|\,|Q(b)| \geq n^3 \text{ , and } m \geq 2\sqrt{n} \quad .$$

The second inequality is the same lower bound we saw in the probabilistic analysis. Note that in the latter case the elements were also symmetric in the sense of interchangeable. These lower bounds are matched *precisely* by arranging the *rendez-vous* matrix R as a checker board consisting of $\sqrt{n} \times \sqrt{n}$ squares, of n entries each. Each square contains n copies of a single element of U, a different one for each square.

Singling out one element gives *centralized* match-making as follows:

Corollary. *Let R be a rendez-vous matrix such that $k_2 = k_3 = \cdots = k_n = 0$ and $k_1 = n^2$, that is, 1 is the central element. Then:*

$$\sum_{a=1}^{n} \sum_{b=1}^{n} |P(a)|\,|Q(b)| \geq n^2 \text{ , and } m \geq 2 \quad .$$

These lower bounds can be matched by upper bounds.

2. The Question of Weighted Match-Making

The constraints (M1)-(M5) and Proposition 1 give a trade-off between the $P(a)$'s and $Q(b)$'s, which is much stronger than the one implied by Proposition 2. We can illustrate this by a simple example. If $P(a)=p$ and $Q(b)=q$ for $1 \le b \le n$, then by Proposition 1 we have $pq \ge n$. If we set $p=n^{1/4}$, then it follows that $q \ge n^{3/4}$, which gives $p+q \ge n^{3/4}+n^{1/4}$. Proposition 2 gives, for $p=n^{1/4}$ only $q \ge 2n^{1/2}-n^{1/4}$, while $p+q > 2n^{1/2}$ does not change. As suggested by this example, we can use the trade-off in Proposition 1 to adjust distributed match-making strategies so as to minimize the *weighted* overall number of messages.

In a nameserver the average call for a service a by a client b occurs more often than the average posting of a service available at a. Or, in a match-making instance (a,b) the server a may be allowed to post $\alpha(a,b)$-many times to the nodes in $P(a)$ and the client b is allowed to query $\beta(a,b)$-many times the nodes in $Q(b)$ (increasing availability/reliability of the network). Therefore, in many applications such as nameserver, mutual exclusion, replicated version management, we may actually be interested in minimizing m with (M2) replaced by (M2′):

$$m(a,b) = \alpha(a,\ b)\,|P(a)| + \beta(a,b)\,|Q(b)| \ . \tag{M2′}$$

This question was the incentive for the present article and is resolved in Theorem 2. It turns out that it is advantageous to solve an s-dimensional generalization of distributed match-making first.

3. Generalization of the (Unweighted) Problem to the s-Dimensional Case

Let $\mathbf{P} = (P_1,\ldots,P_s)$ be a communication strategy in a given network, on a set of nodes N, be defined as follows. For each $j = 1,\ldots,s$, let $P_j : N \to 2^N$ be a total function, such that for each s-tuple $(a,b,\ldots,c)$ of nodes $P_1(a) \cap P_2(b) \cap \cdots \cap P_s(c) \ne \emptyset$. For any s-tuple $(a,b,\ldots,c)$ of nodes let

$$m(a,\ b,\ldots,c) = |P_1(a)| + |P_2(b)| + \cdots + |P_s(c)|$$

be the number of messages required for the match-making instance $(a,b,\ldots,c)$ following strategy $\mathbf{P}$. The average number M of point-to-point messages necessary for match-making is:

$$M = \frac{1}{|N|^s} \sum m(a,b,\ \cdots,c), \tag{1}$$

with the sum taken over $(a,b,\ldots,c) \in N^s$.

Let us interpret the case $s=2$ in terms of the name-server, in order to give the intuitive background for considering weighted versions. Since a server a posts its whereabouts at all the nodes in $P(a)$, by sending messages to all these nodes, and a client b queries each node in $Q(b)$, we have $\mathbf{P} = (P,Q)$. The number $m(a,b)$ of point-to-point messages in the match-making instance (a,b) must be at least* $|P(a)| + |Q(b)|$. In contrast to the *post-query* case ($s = 2$), which is best visualized in two dimensions, the more general case ($s > 2$) is best visualized in s dimensions. (Each axis is marked with a node from N and the vertex $(a, b,...,c)$ is the location of $P_1(a) \cap ... \cap P_s(c)$.) It turns out that to analyse the weighted version of the problem, we have to solve the *multi-dimensional* versions first.

In the last section, we analyse one weight distribution in particular, that can be naturally interpreted as expressing a modified form of distributed match-making in terms of the standard definition. This is related to the least expected number of messages to make the match in a ‘‘success-stop’’ strategy as follows. Namely, b accesses first the node in $Q(b)$ were most hosts a make a rendez-vous, second the next most frequently used node in $Q(b)$, etc., until it exhausts all the nodes in $Q(b)$. Hence, to find a the expected number of messages is usually less than $|Q(b)|$, and b can stop once it has reached the rendez-vous node. Similarly, a accessing nodes in $P(a)$ often can use less than $|P(a)|$ messages to reach the rendez-vous node for node b.

3.1. Outline of the Paper

In the next section we derive the lower bound trade-off (Theorem 1) on the multidimensional case. We then show that the lower bound is tight for the multidimensional mesh, binary n-cube, and finite projective space, by exhibiting distributed algorithms that match the lower bound. In the penultimate section, we derive the promised lower bound on the weighted version of distributed match-making (Theorem 2). In the final section we show that we can improve the previous lower bounds by minimizing the expected number of messages in a slightly modified setting. We show that the ‘‘success-stop’’ version of a match-making algorithm has expected average number of messages that improves the average number of messages required by the corresponding plain version by a factor of about $1/2$ (Theorem 3).

4. The s-Dimensional Lower Bounds

To be able to prove the most general results possible it will be necessary to formulate the required concepts with a higher level of abstraction than in the introduction. The motivation however is derived from the previous section, and the results are necessary to resolve weighted match-making in the next section.

Let $N, N_1,...,N_s$ be nonempty sets of positive integers (node i.d.’s), and $n = |N|$, $n_1 = |N_1|,..., n_s = |N_s|$. It is important to note that, in this setting, the N_i’s are *arbitrary* finite sets, in particular, they can have more elements than N. A *strategy* is defined as

$$\mathbf{P} = \{P_1(a), P_2(b), \cdots, P_s(c) : (a, b,...,c) \in N_1 \times N_2 \times \cdots \times N_s\},$$

---

* It does not cost anything to have a machine send a message to itself. Thus, if $P(a)$ contains a and $Q(b)$ contains b, then we could have set $m(a, b) = |P(a)| + |Q(b)| - 2$. This modification results in about the same theory.

where the $P_i : N_i \rightarrow 2^N$ are total mappings. For $1 \leq i \leq s$ and $a \in N_i$, let $p_i(a) = |P_i(a)|$. Let k_d, $d \in N$, be the number of s-tuples $(a,b,...,c)$ such that $d \in P_1(a) \cap P_2(b) ... \cap P_s(c)$. (If each of these intersections is nonempty then $\sum_{d \in N} k_d \geq n_1 \cdots n_s$, with equality for all intersections are singleton sets.) For the given strategy **P** define the *product* Π and the *sum* M by:

$$\Pi = \frac{1}{n_1 n_2 \cdots n_s} \sum p_1(a) p_2(b) \cdots p_s(c) ,$$

$$M = \frac{1}{n_1 n_2 \cdots n_s} \sum [\, p_1(a) + p_2(b) \cdots + p_s(c) \,] ,$$

with the sums taken over all $(a,b,...,c) \in N_1 \times N_2 \times ... \times N_s$. For $i = 1,2,...,s$ define

$$M_i = \frac{1}{n_i} \sum p_i(a)$$

(with summation over $a \in N_i$) so that

$$\Pi = M_1 M_2 \cdots M_s \text{ and } M = M_1 + M_2 + \cdots + M_s . \tag{2}$$

The main result of the section is the following lower bound on the number of messages for match-making. We first state the theorem and its proof, and then discuss what it means.

Theorem 1.

For any strategy **P** *the following inequalities hold:*

$$\Pi \geq \frac{1}{n_1 n_2 \cdots n_s} \left(\sum_{i \in N} k_i^{1/s} \right)^s$$

$$M \geq \frac{s}{(n_1 n_2 \cdots n_s)^{1/s}} \left(\sum_{i \in N} k_i^{1/s} \right) .$$

Proof of Theorem 1.

The following inequality, also known as *inequality of the arithmetic and geometric means*, holds for s-many nonnegative real numbers $\alpha, \ldots, \sigma$,

$$\alpha + \cdots + \sigma \geq s(\alpha \cdots \sigma)^{1/s}. \tag{3}$$

Equality holds when all the summands are equal [2]. Thus, the inequality in the theorem concerning the sum M follows immediately from the inequality concerning the product Π, identities (2), and inequality (3). It is only left to prove the inequality concerning Π. For each $i \in \{1,2,...,s\}$ and each $j \in N$, define $H_{i,j} \subseteq N_i$ as the the set of nodes d such that some s-tuple $(a,b,...,d,...,c)$, where d is the ith coordinate, satisfies

$$j \in P_1(a) \cap P_2(b) \cdots \cap P_i(d) \cap \cdots \cap P_s(c).$$

Denote the cardinality $|H_{i,j}|$ by $h_{i,j}$. Then, for all $j = 1,...,n$,

$$h_{1,j} h_{2,j} \cdots h_{s,j} = |H_{1,j} \times H_{2,j} \times \cdots \times H_{s,j}|$$

$$\geq |\{(a,b,\cdots,c) : j \in P_1(a) \cap P_2(b) \cdots \cap P_s(c)\}|$$
$$= k_j \ . \qquad (4)$$

Also, for all $i \in \{1,...,s\}$,

$$\sum_{j \in N} h_{i,j} \leq \sum_{j \in N} |\{a : j \in P_i(a)\}|$$
$$= \sum_{j \in N} \sum_{a \in N_i} |\{(j,a) : j \in P_i(a)\}|$$
$$= \sum_{a \in N_i} |\{j : j \in P_i(a)\}|$$
$$= \sum_{a \in N_i} p_i(a) = n_i \ M_i \qquad (5)$$

To obtain the lower bound on Π, proceed as follows.

$$\Pi = M_1 M_2 \cdots M_s \qquad \text{(by (2))}$$
$$\geq \frac{1}{n_1 n_2 \cdots n_s} \left(\sum_{j \in N} h_{1,j} \right) \cdots \left(\sum_{j \in N} h_{s,j} \right) \qquad \text{(by (5))}$$
$$= \frac{1}{n_1 \cdots n_s} \sum_{j_1,...,j_s \in N} S(j_1, j_2, \ldots, j_s), \qquad (6)$$

where $S(j_1,...,j_s) = h_{1,j_1} \cdots h_{s,j_s}$. By interchanging the order of summations and then renumbering the dummy indices, we can rewrite this last quantity as the sum

$$\frac{1}{n_1 n_2 \cdots n_s} \sum_{j_1=1}^{n} \cdots \sum_{j_s=1}^{n} S(j_{\pi(1)}, \ldots, j_{\pi(s)})$$

for any permutation π of the numbers from 1 through s. For our purposes it is convenient to rewrite with π equal, in turn, to each of the cyclic permutations of 1 to s, and then to consider the average of the s resulting equal quantities. This average is given by a sum of the same form with the summand

$$\frac{S(j_1, \ldots, j_s) + S(j_2, \ldots, j_s, j_1) + ... + S(j_s, j_1, ..., j_{s-1})}{s}$$

Using inequalities (3) and (4), it is easy to see that this summand must be at least $(k_{j_1} \cdots k_{j_s})^{1/s}$. From this the claimed lower bound on Π follows, and hence the proof of the theorem is complete. □

Corollary. Both Propositions 1 and 2 of [4] are immediate consequences of Theorem 1. □

Remark 1. For simplicity in the following discussion let $N = \{1, 2, ..., n\}$. As in the 2-dimensional case, the n-dimensional vector $(k_1, k_2, ..., k_n)$ can be viewed as a measure of the ‘distributedness’ of the strategy **P**. Namely, it tells us how evenly the load of making matches is distributed among the different nodes in N. One extreme is if $k_1 = n^s$ and

$k_2 = ... = k_n = 0$. Then the algorithm is centralized since for each match-making instance $(a, b, ..., c)$, node 1 is the only node to which nodes a, b, and c send messages. As another extreme we can view the case $k_1 = ... = k_n = n^{s-1}$. Then the algorithm may be called 'truly distributed', since each node i $(1 \le i \le n)$ functions as the rendez-vous node (node where the match is made) for an equal number n^{s-1} of match-making instances $(a, b, ..., c)$. In other words, in this case the load of match-making is distributed evenly over all nodes in the network. Therefore, the lower bounds in Theorem 1 can be interpreted as trade-offs between the average message cost and the measure of distributedness of a match-making strategy.

Remark 2. If $n_1 = \cdots = n_s = n$, and for simplicity $N = \{1, 2, ..., n\}$, then

$$\Pi \ge \left(\frac{1}{n} \sum_{i=1}^{n} k_i^{1/s} \right)^s \quad \text{and} \quad M \ge \frac{s}{n} \left(\sum_{i=1}^{n} k_i^{1/s} \right).$$

Additionally, consider the symmetric case where all k_i's are equal, viz., $k_i = n^{s-1}$, $i = 1, ..., n$. Then Theorem 1 specializes to the 'truly distributed' case $\Pi \ge n^{s-1}$ and $M \ge sn^{(s-1)/s}$, for which we establish matching upper bounds in the sequel.

Remark 3. The lower bound on Π gives more information than the lower bound on M (which it implies). As a (simple) example consider the truly distributed case of Remark 2 with $s = 2$. We have $M = M_1 + M_2$. Suppose we know $M_1 = n^{1/4}$ but we do not know M_2. The lower bound $M \ge 2n^{1/2}$ allows us to conclude that $M_2 \ge 2n^{1/2} - n^{1/4}$. However, the lower bound $\Pi \ge n$ shows $M_2 \ge n^{3/4}$, which implies in turn that $M \ge n^{3/4} + n^{1/4} \gg 2n^{1/2}$.

Remark 4. M equals the right-hand side of the inequality in which it occurs, exactly when $M_1 = \cdots = M_s$, which means that the strategy $\mathbf{P}$ is optimal exactly when the average number of messages is equally balanced in all directions.

5. Optimality

We show that Theorem 1 is optimal in some special cases (which are of sufficient generality), by exhibiting matching strategies. (We only consider the number of messages needed for the match-making part of an algorithm.) For more examples the reader is referred to [4].

(Manhattan Network) Consider the s-dimensional grid of nodes that is $n^{1/s}$ nodes long in each dimension: the s-dimensional Manhattan network. Each query transmits to all nodes in the hyperplane through the querying node that is perpendicular to that query's axis. This uses $sn^{(s-1)/s}$ messages, which is optimal by Theorem 1.

(Cube Network) Let the number of nodes be $n = 2^d$ and suppose that s is a divisor of d. Addresses of nodes consist of d bits, like $u_1 u_2 \cdots u_d$. Nodes are connected by an edge exactly when they differ by a single bit. Let $\mathbf{P} = (P_1, ..., P_s)$ be a strategy, and, for each $r \in \{1, ..., s\}$, let $P_i(u_1 ... u_d)$ be the set

$$\{x_1 \cdots x_{(r-1)d/s} u_{(r-1)d/s+1} \cdots u_{id/s} x_{id/s+1} \cdots x_d : x_i \in \{0,1\}\}.$$

Clearly, each of the above sets has size $2^{(s-1)d/s}$ and $k_i = 2^{(s-1)d} = n^{s-1}$. Thus, one easily obtains that $M \le sn^{(s-1)/s}$, that is, the average number of point-to-point message transmissions is at most $sn^{(s-1)/s}$. By Theorem 1 this strategy is also optimal.

(Finite Projective Space). Consider generalized mutual exclusion in a distributed setting, where $s-1$ processors are allowed to be in the critical section simultaneously, but not s or more processors. For background and nondistributed solutions we refer to [1]. In [3], Maekawa considers the distributed version of mutual exclusion for $s = 2$, the commonly studied variant. In our terminology, for mutual exclusion with $s = 2$ we can set $P_1(i) = P_2(i)$, which is some sort of symmetry condition. Each instance of mutual exclusion contains a match-making instance [4]. For the truly distributed case, with $k_1 = \cdots = k_n = n$ and $s = 2$ we find that on the average each match-making instance takes at least $2\sqrt{n}$ messages [4]. Maekawa obtains a similar lower bound, and exhibits an algorithm that achieves $5\sqrt{n}$ [3]. Theorem 1 gives a lower bound of $sn^{(s-1)/s}$ for the generalized version. The optimal solution there is achieved by the finite projective space $PG(s,k)$ (k a prime power), where the network has $k^s + k^{s-1} + \cdots + 1 = n$ nodes, each node is incident to all $k^{s-1} + k^{s-2} + \cdots + 1$ hyperplanes, and each hyperplane contains $k^{s-1} + k^{s-2} + \cdots + 1$ nodes. Each family of s hyperplanes intersects in precisely one node. Therefore, each query set $S(i) = P_1(i) = \cdots = P_s(i)$ of a node i consists of the set of $k^{s-1} + k^{s-2} + \cdots + 1$ nodes incident on some hyperplane containing node i. It does not matter which hyperplane we pick, because any s hyperplanes intersect in a single node. The average cost M of point-to-point messages associated with a particular mutual exclusion instance is therefore $O(s(k^{s-1} + k^{s-2} + \cdots + 1)) \approx O(sn^{(s-1)/s})$ (generalizing Maekawa's method to $s \ge 2$ [3]). By Theorem 1 this strategy is also optimal.

6. Weighted Distributed Match-Making

We now examine weighted distributed match-making. Given a strategy $\mathbf{P} = (P_1, \ldots, P_s)$, with all parameters as before, we define a weighted version of the number of messages required for match-making. For each match-making instance $S = (a, b, \ldots, c)$, let $(l_1(S), l_2(S), \ldots, l_s(S))$ be an s-dimensional vector of positive integer *weights*. The weighted cost of match-making instance S is:

$$m(S) = l_1(S)p_1(a) + l_2(S)p_2(b) + \cdots + l_s(S)p_s(c),$$

The *weighted cost* M of strategy $\mathbf{P}$ is:

$$M = \frac{1}{n_1 \cdots n_s} \sum l_1(S)p_1(a) + \cdots + l_s(S)p_s(c),$$

the sum taken over all $S \in N_1 \times \ldots \times N_s$. Then,

$$n_1 \cdots n_s M = \sum_{j_1 \in N_1} \left(\sum_{S \in \mathbf{S}_1} l_1(S) \right) p_1(j_1) + \cdots + \sum_{j_s \in N_s} \left(\sum_{S \in \mathbf{S}_s} l_s(S) \right) p_s(j_s)$$

$$= \sum_{j_1 \in N_1} N_{1,j_1} p_1(j_1) + \cdots + \sum_{j_s \in N_s} N_{s,j_s} p_s(j_s) \,, \tag{7}$$

where, for $i = 1, 2, \ldots, s$:

$$N_{i,j_i} = \textstyle\sum_{S \in \mathbf{S}_i} l_i(S), \ \ \mathbf{S}_i = N_1 \times ... \times N_{i-1} \times \{j_i\} \times N_{i+1} \times ... \times N_s.$$

Consider the following related strategy $\mathbf{Q}$ for the sets N, and, for $i = 1,2,...,s$, $N_i' = N_i \times \{1,2,...,N_{i,j}\}$:

$$\mathbf{Q} = \{Q_1(a), Q_2(b), \cdots, Q_s(c) \colon (a,b,\ldots,c) \in N_1' \times N_2' \times \cdots \times N_s'\}.$$

For $1 \le i \le s$ and $j \in N_i'$, let $q_i(j) = |Q_i(j)|$. Note that we have chosen the definitions in such a way that

$$\sum_{j \in N'_i} q_i(j) = \sum_{j \in N_i} N_{i,j} p_i(j) \,, \tag{8}$$

for all i from 1 through s. In the following we shall use primed identifiers for the variables associated with $\mathbf{Q}$ to distinguish them from the corresponding unprimed variables associated with $\mathbf{P}$. We relate M with Π' as follows.

$$\begin{aligned}
M &= \frac{1}{n_1 \cdots n_s} \left(\sum_{j_1 \in N'_1} q_1(j_1) + \cdots + \sum_{j_s \in N'_s} q_s(j_s) \right) \quad \text{(by (7), (8))} \\
&\ge \frac{s}{n_1 \cdots n_s} \left(\left(\sum_{j_1 \in N'_1} q_1(j_1) \right) \cdots \left(\sum_{j_s \in N'_s} q_s(j_s) \right) \right)^{1/s} \quad \text{(by (3))} \\
&= \frac{s}{n_1 \cdots n_s} (N'_1 \cdots N'_s)^{1/s} \Pi'^{1/s} \quad \text{(by definition)} \\
&\ge \frac{s}{n_1 \cdots n_s} \sum_{i \in N} k_i'^{1/s}. \quad \text{(by Theorem 1)}
\end{aligned}$$

It remains to compare the quantities k_i and k_i'. This is easy. Namely, for each s-tuple $(j_1, \ldots, j_s)$ such that $i \in P_1(j_1) \cap \cdots \cap P_s(j_s)$ we have choosen the N_i' such that there are at least $N_{1,j_1} \cdots N_{s,j_s}$-many s-tuples $(j_1', \ldots, j_s')$ such that $i \in Q_1(j_1') \cap \cdots \cap Q_s(j_s')$. In other words, to each match-making instance $(a,b,...,c)$ with rendez-vous node i in $\mathbf{P}$, there correspond at least $N_{1,a} \cdots N_{s,c}$ distinct match-making instances $(a', b', \ldots, c')$ with rendez-vous node i in $\mathbf{Q}$. Hence, we conclude that

$$M \ge \frac{s}{n_1 \cdots n_s} \sum_{i \in N} \left(\sum \{ N_{1,a} N_{2,b} \cdots N_{s,c} : (a,b,...,c) \text{ matches at } i \text{ in } \mathbf{P} \} \right)^{1/s}. \tag{9}$$

With some computation we can specialize the general result (9) to ‘instance-independent’ weights, to obtain the more pleasant looking theorem below.

Theorem 2. *For any strategy* $\mathbf{P}$, *if there are positive integers* $\lambda_1,...,\lambda_s$ *such that for each* $(a,b,...,c)$ *in* $N_1 \times N_2 \times ... \times N_s$ *we have that* $m(a,b,...,c) = \lambda_1 p_1(a) + \lambda_2 p_2(b) + ... + \lambda_s p_s(c)$, *then*

$$M \ge \frac{s(\lambda_1 \cdots \lambda_s)^{1/s}}{n} \sum_{i \in N} k_i^{1/s}. \tag{10}$$

The quantity M *equals the right-hand side of the inequality above, exactly when*

$\lambda_1 M_1 = \cdots = \lambda_s M_s$. □

Corollary. Theorem 2 also holds for rational λ's.

Hint. Let $\lambda_i = p_i / q_i$, $i \in \{a,...,s\}$, and denote the corresponding M-quantity by M_λ. Let $\mu_i = C\lambda_i$ with $C = q_1 \cdots q_s$, $i \in \{a,...,s\}$, and denote the corresponding M-quantity by M_μ. Since all μ-weights are C times the λ-weights, $M_\mu = C\, M_\lambda$. Substituting the integer μ's in Equation (10) gives

$$C\, M_\lambda = M_\mu \geq \frac{s(C^s\, \lambda_1 \cdots \lambda_s)^{1/s}}{n} \sum_{i \in N} k_i^{1/s}.$$

Dividing both sides by C yields the corollary. □

Remark 5. Hence we have demonstrated a very general lower bound on weighted distributed match-making. Let us illustrate its use in a very simple case. Consider the 'truly distributed' case for $s = 2$, $n_1 = n_2 = n$, and $k_i = n$ for all $i \in N$. We interpret this for the name-server with strategy $\mathbf{P} = (P,\, Q)$. Assume that each posted service gets 100 queries in its lifetime. Then each match-making instance $(a,\, b)$ carries a cost of $m(a,\, b) = |P(a)|/100 + |Q(b)|$. That is, $\lambda_1 = 1/100$ and $\lambda_2 = 1$. By (10), therefore, the weighted match-making cost M is bounded below by

$$M \geq \frac{2 \cdot \sqrt{1/100 \cdot 1}}{n} \sum_{i \in N} n^{1/2} = \frac{2}{10}\, n^{1/2}.$$

7. Success-Stop Strategies

The above investigations establish the number of messages necessary for distributed match-making, while the examples given here and in [7] suggest that this is sufficient as well for the current problem formulation. While it thus appears that the lower bound provided cannot be improved any further, Paris Kanellakis has drawn our attention to the fact that the lower bound is with respect to the assumption that the querying node sends messages to all nodes to be queried at once. We may hope to improve the performance when we develop strategies that query the nodes one node at a time and stop when the queried node is the one looked for. For any given strategy $\mathbf{P}$ we will construct a new "success-stop" strategy. The "expected" number of messages for this new strategy will be shown to be $\leq (1/2) \cdot M$, with M as in (1).

7.1. The Two-Dimensional Case

By way of illustration we first formulate such a strategy for the unweighted, two-dimensional case of the distributed match-making problem.

Let $\mathbf{P} = (P,\, Q)$ with $P,\, Q : N \rightarrow 2^N$. The rendez-vous matrix is the matrix whose (a,b)-entry is the (non-empty) set $P(a) \cap Q(b)$. The modification we look at may be called *success-stop*. The idea is the following. Let us take the name-server interpretation again. Each server a posts at a set $P(b)$. Each client b queries the elements of a set of nodes $Q(b) = Q_1(b) \cup ... \cup Q_n(b)$ as follows. First it queries the elements of $Q_1(b)$, then $Q_2(b)$, up to $Q_n(b)$. All sets $Q_k(b)$ are disjoint, and may be empty from $k = m$, for some *m*, onwards. In our earlier set-up, the cost to make a match between a server located at *a* and a client located at *b* is $|P(a)| + |Q_1(b)| + \cdots + |Q_n(b)|$. In our new set-up, the client stops querying after it has found the server. That is, if *m* is the least index such that $P(a) \cap Q_m(b) \neq \emptyset$, then the client does not need to query $Q_{m+1}(b),...,Q_n(b)$. The new cost to make the match is $P(a)+Q_1(b)+...+Q_m(b)$. This cost is minimized for a particular rendez-vous matrix, if in column *b* we choose as $Q_1(b)$ the singleton set with the node occurring most often in that column, as $Q_2(b)$ the singleton set with the node occurring next most often, and so on. (With such a method we may also address questions of fault-tolerance by setting $Q_1(b) \cup \cdots \cup Q_n(b) = \{1,...,n\}$, or anyway a large subset of *N* for all *b*.)

Example. Consider the rendez-vous matrix of a three node network:

$$\begin{matrix} 2 & 3 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 2 \end{matrix}$$

We add some information to the rendez-vous matrix to represent the success-stop strategy:

$$\begin{matrix} (1,2) & (2,3) & (1,1) \\ (1,2) & (1,1) & (1,1) \\ (2,1) & (1,1) & (2,2) \end{matrix}$$

Now each element consists of a pair (try number, rendez-vous node) the first coordinate specifies the sequence number of the try, and the second coordinate specifies the rendez-vous node which is used in that try. Since rows are associated with servers and columns with clients, then the cost for matching server one with client two at rendez-vous node three is now $2 + 3 = 5$. That is, client two queries both nodes one and three, and server one posts at nodes one, two, and three. However, the cost of making a match between client two and server three is only $2+1=3$. Namely, client two is successful in the first query for server three since the node at the third row is queried already in the first try, and server three posts at both nodes one and two. Similarly, to match server three with client three takes $2 + 2 = 4$ messages.

A similar idea applies for the case of posting. For a match between a server at node *a* and a client at node *b*, we only count the repeated number of postings the server has to do to reach the proper rendez-vous node. For each process *a* the set $P(a)$ is split into singletons $P_1(a),...,P_n(a)$, where all $P_k(b)$ are disjoint and may be empty from some *k* onwards. As before, $P_1(a)$ is the singleton with the most frequently "occurring" node in the *a*th row of the rendez-vous matrix, $P_2(a)$ is the singleton with the second most frequently "occurring" node in the *a*th row, etc. The above modification of posting is called *repeated posting*.

Remark 6. It clearly works to post everywhere first, and then query one site at a time, stopping as soon as the posted site is found. Conversely, it would also work to query everywhere and then post to one site at a time, stopping as soon as a queried site is found. But if posting and querying are happening one site at a time, in what order are they interleaved? The number of messages would, of course, depend on the interleaving. There is another issue that would also arise in the simultaneous case. Suppose that repeated posting and repeated querying are happening concurrently, and that the rendez-vous node (which was posted some time ago, say) has just been queried. This successful rendez-vous stops the querying process; but does it also stop the posting process? In this preliminary investigation we formulate the abstract problem below, and simply ignore these interpretational difficulties (left as open problems). The results could be interpreted as setting outer limits to the possible gain of using success-stop strategies.

We modify the rendez-vous matrix once more to represent this strategy: each entry is a triple (try number server, try number client, rendez-vous node). For instance, continuing our example,

$$\begin{array}{ccc} (2,1,2) & (1,2,3) & (3,1,1) \\ (2,1,2) & (1,1,1) & (1,1,1) \\ (1,2,1) & (1,1,1) & (2,2,2) \end{array}$$

Now to match server one with client two costs $1+2=3$ messages, and to match server three with client two costs only $1+1=2$ messages.

7.2. Analysis

In this section we analyse a particular form of the problem. Here, it is just important to us to point out that the analysis of this matter is not restricted to trivialities. We are given a communication strategy $\mathbf{P}=(P_1,...,P_s)$ on a network with set of nodes $N=\{1,2,...,n\}$. Let k_l be the number of s-tuples $(a,b,...,c)$ such that $l \in P_1(a) \cap P_2(b) \cdots \cap P_s(c)$. (In general, we use the same notation as in the previous sections.) For each pair i,j, consider the nodes l occurring in the query set $P_i(j)$ and define

$$k_l(i,j) = |\{(a,b,...,j,...,c) : l \in P_1(a) \cap ... \cap P_i(j) \cap ... \cap P_s(c)\}|.$$

Clearly, if l is not in $P_i(j)$ then $k_l(i,j)=0$. It is also immediate that for any direction i, and for any node l,

$$k_l = \sum_{i=1}^{s} \sum_{j \in N_i} k_l(i,j) \le n. \tag{11}$$

Next arrange the nodes l in $P_i(j)$ in order of decreasing $k_l(i,\ j)$'s such that:

$$k_1(i,\ j) \ge k_2(i,\ j) \ge \cdots \ge k_m(i,\ j),$$

where $m = |P_i(j)|$ and we have renamed the nodes by the integers 1 through m. Finally consider the following new strategy:

Success-Stop Strategy

Instead of sending messages to each element in $P_i(j)$, process j executes the following algorithm. We assume that j knows the order of the $k_l(i,\ j)$'s. Moreover, we assume that a rendez-vous node sends a message back to a querying node informing it whether or not the intended match was successful.

for $l = 1, 2, ..., m$ *do:*

send message to node l;

if match successful *then* exit

The expected number of messages required for match-making of node j using P_i, *ignoring* the response messages of the rendez-vous nodes, is defined as

$$E(i,j) = \frac{1 \cdot k_1(i,j)}{n} + \frac{2 \cdot k_2(i,j)}{n} + \cdots + \frac{m \cdot k_m(i,j)}{n}.$$

After regrouping terms and simplifying, we obtain that the righthand side of the above equality is:

$$\leq \frac{m+1}{2n} \cdot (k_1(i,j) + k_2(i,j) + \cdots + k_m(i,j))$$

$$\leq \frac{m+1}{2n} \cdot n = \frac{m+1}{2} \quad \text{(by (11))}.$$

Averaging out over $j \in N_i$ we conclude that the expected average number of messages send by the $j \in N_i$ according to P_i in the success-stop strategy satisfies

$$\overline{M}_i \leq \frac{1}{n_i} \cdot \sum_{j \in N_i} \frac{|P_i(j)| + 1}{2}.$$

The expected average number of messages $\overline{M}$ for the success-stop strategy is given by

$$\overline{M} = \sum_i \overline{M}_i.$$

Using the definitions of M, M_i, and identity (2), we obtain the following theorem.

Theorem 3. *For any s-dimensional match-making strategy* **P** *the expected average number of messages* $\overline{M}$ *of the associated success-stop strategy satisfies*

$$\overline{M} \leq \frac{M}{2} + \frac{s}{2}. \quad \square$$

Remark 7. Note that without the assumption that the nodes in the P-sets have a known order according to their frequency of occurrence as rendez-vous node, the upper bound in Theorem 3 may increase. If we also include the messages send back by the rendez-vous nodes to inform a sender of success/failure of a desired match, then $\overline{M}$ doubles, so the upper bound on the expected value becomes slightly worse again than M.

Acknowledgements.

The idea of success-stop strategies was suggested first to us by Paris Kanellakis. We thank the referees for detailed comments, which resulted in considerable improvement of the paper.

References

[1] Fischer, M.J., Lynch, N.A., Burns, J.E., and Borodin, A., *Distributed FIFO Allocation of Identical Resources Using Small Shared Space*, ACM Trans. Program. Lang. Syst., **11**(1989), pp. 90-114.

[2] Hardy, G. H., Littlewood, J. E. and Polyá, G., *Inequalities,* Cambridge University Press,

1934.

[3] Maekawa, M., *A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems*, ACM Transactions on Computer Systems **3** (1985), pp. 145-159.

[4] Mullender, S. J. and Vitányi, P. M. B., *Distributed Match-Making*, Algorithmica, **3** (1988), pp. 367-391.

[5] Powell, M.L. and Miller, B. P., *Process Migration in DEMOS/MP*, Proceedings of the 9th ACM Symposium on Operating Systems Principles, 1983, pp. 110-119.

[6] Tanenbaum, A.S. and Mullender, S.J., *The Design of a Capability Based Distributed Operating System*, The Computer Journal, 29(1986).

[7] Tanenbaum, A.S., *Computer Networks*, Prentice-Hall, 1981.

Appendix: Examples

1. *Broadcasting*: $m = n + 1$.

	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

2. *9 nodes in a* 3×3 *Manhattan network* : $m = 2\sqrt{n}$.

	1	2	3	4	5	6	7	8	9
1	1	2	3	1	2	3	1	2	3
2	1	2	3	1	2	3	1	2	3
3	1	2	3	1	2	3	1	2	3
4	4	5	6	4	5	6	4	5	6
5	4	5	6	4	5	6	4	5	6
6	4	5	6	4	5	6	4	5	6
7	7	8	9	7	8	9	7	8	9
8	7	8	9	7	8	9	7	8	9
9	7	8	9	7	8	9	7	8	9