PACKET DISTRIBUTION ON A RING

David Peleg § and *Allen Van Gelder* ‡
Dept. of Computer Science, Stanford University, Stanford, CA 94305

Abstract: The balanced packet distribution problem on a ring of n processors requires that randomly arriving packets be stored at nodes as evenly as possible by passing packets (and other messages) around the ring. We give a protocol to achieve balanced distribution with average message complexity of $\sqrt{n}$ per packet, and show that the protocol is optimal, up to lower order terms, for unidirectional rings.

Key words. distributed computation, ring networks, linear probing.

1. Introduction

This note deals with a problem of packet distribution on a ring of processors, presented by Bodlaender and van Leeuwen [1]. Consider a network of n processors connected in a ring, so that each processor can directly communicate with its two immediate neighbors. Assume the network operates in rounds. In each round, an arbitrary (i.e., random) processor receives a packet of information from some external source. This packet must be stored at some processor on the ring. The receiving processor either stores the packet locally, or moves it along the ring (according to some protocol) until it is stored at some processor. There is enough time between consecutive rounds for the protocol to complete this operation. (The assumption of operation in "sufficiently long" rounds is made here following [1]. In Section 5 we discuss how this assumption may be relaxed and replaced with a random arrival rate.)

The problem asks for protocols that attempt to minimize the maximum number of packets stored at a node and the number of messages needed to achieve a reasonable distribution on the average. A special case of particular interest is that of achieving a *completely balanced distribution*. In this case we require the protocol to keep the difference between the minimum and the maximum number of packets held at a processor at most 1.

The motivation given in [1] for this problem concerns modeling a distributed database on which queries are processed in a pipelined fashion. The bottleneck of the querying process is at the nodes having the largest number of records, hence the need to balance the distribution. More generally, the problem may be interpreted as dealing with the average communication complexity of a restricted form of load balancing, in which one attempts to balance only the *number* of tasks awaiting execution in each processor, making some simplifying assumptions on the input

§ Supported in part by a Weizmann fellowship and by contract ONR N00014-85-C-0731
‡ Current address: CIS, Applied Sciences, UCSC, Santa Cruz, CA 95064

parameters of the system. The results can thus serve as a first step towards obtaining an estimate of the communication complexity of load balancing protocols when additional complicating factors are taken into consideration.

Bodlaender and van Leeuwen [1] present several token-based protocols for the packet distribution problem, some of which also satisfy the stronger "complete balancing" requirement. Their basic protocol makes use of a token which circulates the ring. Each incoming packet is forwarded along the ring until it reaches the node currently holding the token. The packet is then stored in that node, and the token is passed on to the next node. Variations of this protocol are considered, such as using two or more tokens, moving the tokens and the packets in opposite directions, etc. However, these protocols require $\Theta(n)$ messages per round on the average in order to maintain completely balanced distribution.

In this note we present a protocol which maintains completely balanced distribution while requiring only $\Theta(\sqrt{n})$ messages per round on the average. The main idea behind our protocol is the application of the *linear probing* technique [2]. The protocol works on either unidirectional or bidirectional rings, and it can be extended in a natural way to handle packet deletions, in the same ways as studied in [1; Section 7] w.r.t. the token-based protocols. Furthermore, we show that our protocol is optimal (up to lower order terms) for unidirectional rings.

2. The distribution protocol

We refer to the round in which the $(p+1)$st packet arrives (i.e., where p packets are already stored in the network) as round p ($p = 0, 1, 2, \ldots$). Consider the distribution of packets at the beginning of round p. Our goal is to keep the number of packets stored in different nodes either $\lfloor \frac{p}{n} \rfloor$ or $\lceil \frac{p}{n} \rceil$. In order to achieve this goal we use the idea of *linear probing*, which was used before as a hashing method (cf. [2]). In every given moment every node knows whether it has $\lfloor \frac{p}{n} \rfloor$ or $\lceil \frac{p}{n} \rceil$ packets. This can be represented by a bit b holding 0 or 1, respectively. If the node receiving a packet (from an external source or from its neighbor) has $b = 0$ (i.e., it has $\lfloor \frac{p}{n} \rfloor$ packets), then it stores the packet locally and sets its bit to 1. Otherwise, it passes the packet to its neighbor (say, clockwise). The packet continues traveling around the ring until it finds a node with $b = 0$, which will store it locally. In case the packet completes a cycle around the ring and returns to the node which received it from the external source, this means that all nodes have $b = 1$, so in fact, all of them hold precisely $\frac{p}{n}$ packets, and the ring is said to have completed a *phase*. Now the receiving node initiates a message $\langle reset \rangle$, which circles the ring and causes all nodes to reset their bit to 0. When the $\langle reset \rangle$ reaches the receiving node, it stores the packet locally, and sets its bit to 1.

Protocol H

Each node v has a boolean variable $b(v)$. Before a phase has been initialized, $b(v) = 1$ for all nodes v and the nodes have no packets.

1. If v receives a packet X for the first time (either from another node or, at the beginning of a round, from an external source), then v executes:

   if $b(v) = 0$ then set $b(v) \leftarrow 1$ and store X,

   else send X to the next node.

2. If v receives the packet X for the *second* time then v executes:

   store X.

   send the message $\langle reset \rangle$ to the next node.

3. If a node v receives a message $\langle reset \rangle$ not initiated by itself then it executes:

   reset $b(v) = 0$

   forward the message $\langle reset \rangle$ to the next node.

3. An upper bound on the average number of messages

The analysis of the average number of messages used in a round by Protocol H exhibits a difference in behavior, compared to the token-based protocols of [1]. There, the average in every round is the same (and is about $n/2$). In contrast, in this protocol, the cost of early rounds in a phase is very low, and it increases towards $n/2$ near the end of the phase. In addition, there is a cost of $2n$ messages to detect that a phase has ended and initiate the next phase. It is clear that the behavior has period n, so the cost of a round amortized over a phase has to be studied. As our analysis will show, the average number of messages per round is bounded above by $\sqrt{n} + \frac{3}{2}$.

Let us denote the average number of messages required in round p by $\bar{M}_p$. Note that it is sufficient to consider rounds 0 through $n-1$, i.e., $\bar{M}_{p+n} = \bar{M}_p$ for all p. It is clear that for a round number $k > 0$ only step (1) is executed (by a varying number of nodes). Further, for round 0 exactly one node executes step (2), namely, the node which receives the packet from the external source. All other nodes execute step (3) in that round. Hence in that round the packet travels one complete cycle around the ring and then is stored in the node that received it originally, and the message $\langle reset \rangle$ travels one complete cycle around the ring too, and overall, $\bar{M}_0 = 2n$.

It thus remains to analyze $\bar{M}_k$ for $1 \leq k \leq n-1$. The space of possible inputs for the protocol which are relevant for analyzing the behavior of the protocol in round k can be modeled by the set of sequences $(a_0, \ldots, a_k)$, where $0 \leq a_i \leq n-1$ describes the node that received the packet in round i, for $i = 0, \ldots, k$. For the average case analysis we assume (as in [1]) that in each round each processor has an equal probability of receiving the incoming packet. Therefore all of these n^{k+1} sequences have equal probability. For every $1 \leq k \leq n-1$, the number of messages sent in

round k is one less than the number of nodes examined until finding a suitable host (i.e., a node v with $b(v) = 0$). For an input sequence $(a_0, \ldots, a_k)$, this number is identical to the number of probes required in the case of an unsuccessful search for a_k in the hashing scheme of linear probing [2] when the table contains k values, assuming that the initial hash entries used to insert the first k values were $(a_0, \ldots, a_{k-1})$. With the above assumption on the distribution of input sequences, the analysis of [2] applies directly to our model. Therefore we have that for $1 \leq k \leq n-1$, $\bar{M}_k = \frac{1}{2}(1 + Q_1(n,k)) - 1$, where

$$Q_1(n,k) = 1 + \sum_{i \geq 1}^{k} (i+1) \left(\frac{k}{n}\right) \left(\frac{k-1}{n}\right) \cdots \left(\frac{k-i+1}{n}\right)$$

or, denoting $k^{\underline{i}} = k!/(k-i)! = k(k-1)\cdots(k-i+1)$, and $k^{\underline{0}} = 1$,

$$Q_1(n,k) = \sum_{i \geq 0}^{k} (i+1) \frac{k^{\underline{i}}}{n^i}$$

The average number of messages per round, amortized over a phase, is

$$\bar{M} = \frac{1}{n}\left(2n + \sum_{k=1}^{n-1} \bar{M}_k\right) = \frac{3}{2} + \frac{1}{2n} \sum_{k=0}^{n-1} Q_1(n,k)$$

since $Q_1(n,0) = 1$.

Thus we have $\bar{M} = \frac{1}{2n} P + \frac{3}{2}$, where

$$P = \sum_{k=0}^{n-1} \sum_{i=0}^{k} (i+1) \frac{k^{\underline{i}}}{n^i}$$

To estimate P, we interchange the summations and sum by parts, giving

$$P = \sum_{i=0}^{n-1} \frac{1}{n^i} \sum_{k=i}^{n-1} (i+1) k^{\underline{i}} = \sum_{i=0}^{n-1} \frac{n^{\underline{i+1}}}{n^i} = n \sum_{i=0}^{n-1} \frac{(n-1)^{\underline{i}}}{n^i}$$

The sum is easily estimated by breaking it into two at $\sqrt{n}$.

$$P = n \sum_{0 \leq i < \sqrt{n}} \prod_{j=1}^{i} \left(\frac{n-j}{n}\right) + n \sum_{\sqrt{n} \leq i < n} \prod_{j=1}^{i} \left(\frac{n-j}{n}\right)$$

The left sum is clearly bounded above by $\sqrt{n}$ and the right sum is dominated by the geometric series $\sum_{i \geq 0} (1 - 1/\sqrt{n})^i$. Thus $P < 2n\sqrt{n}$.

Note: For the lower bound of the next section, we observe that the summands are monotonically decreasing, and the last summand of the left sum is greater than $e^{-0.5 - n^{-0.5}}$ for large n, which is bounded away from zero. Thus $P = \Omega(n^{\frac{3}{2}})$ also.

Returning to the upper bound, $\bar{M} < \sqrt{n} + \frac{3}{2}$, which proves

Theorem 1: *There is a protocol for the completely balanced distribution problem on a ring of n processors which operates with average (amortized) message complexity of $O(\sqrt{n})$ messages per round.* ∎

Note: The *worst case* is when in n consecutive rounds the packet is received in the same node v, and the total number of messages in this case will be $2n + 1 + \ldots + (n-1)$, or $\frac{n+3}{2}$ messages per round.

4. A lower bound

In this section we consider only unidirectional rings, namely, rings in which messages may move only in one direction (say, clockwise). Note that Protocol H is executable in such a model. We prove that no other protocol for balanced distribution on unidirectional rings can do significantly better than Protocol H.

Let us first give a lower bound for the message complexity of Protocol H presented above. For this protocol, as noted above, $P = \Omega(n^{\frac{3}{2}})$, hence also $\bar{M} = \Omega(\sqrt{n})$, which proves

Theorem 2: *Protocol H has average (amortized) message complexity of $\Omega(\sqrt{n})$ messages per round.* ∎

Next, we need the following definitions and lemma. For two nodes v and w let $d(v,w)$ denote the clockwise distance from v to w. At any given moment, the *configuration* of the ring is characterized by the setting of the $b(v)$ bits in the various nodes v. (Note that these bits need not be actually maintained in the system, but are rather thought of as an external description of the state of the system, representing the "available" and "occupied" nodes.) An *i-configuration* is a configuration with a total of i nodes at which $b = 1$. The possible configurations at the beginning of round $p = kn + i$, for every $k \geq 0$, are precisely the i-configurations.

Lemma 3: *Let two i-configurations C_1 and C_2 differ only in that C_1 has $b(v) = 0$ and $b(w) = 1$, while C_2 has the $b(v) = 1$ and $b(w) = 0$. For some specific sequence of packet arrivals in rounds $i+1$ through $n-1$, let R_1 and R_2 denote the runs of Protocol H starting from C_1 and C_2, respectively, and let M_1 and M_2 be the total number of messages used in R_1 and R_2, respectively. Then $M_1 - M_2$ is either $-d(v,w)$ or $d(w,v)$.*

Proof: The proof is by backward induction on i. The base case, $i = n-1$ is immediate. Assume the lemma holds for $i+1$, where $0 \leq i < n-1$. Let the packet for round i arrive at node u. There are several cases.

1. The packet is stored at the same node in both runs: Then the lemma follows for i by the inductive hypothesis.
2. The packet is stored in v in R_1: Then in R_2 the packet is stored in some node v' that is clockwise

of v, but no further than w. If $v' = w$, the lemma is immediate for i. Otherwise $d(v', w) > 0$. R_1 used $d(v, v')$ fewer messages than R_2 in round i, and the new configurations differ only at v' and w. Therefore by the inductive hypothesis $M_1 - M_2$ is either $-d(v', w) - d(v, v')$ or $d(w, v') - d(v, v')$. But these two quantities are just $-d(v, w)$ and $d(w, v)$.

3. The packet is stored in w in R_2: Analogous to Case 2.

And the lemma is proved. ∎

We prove the lower bound for a slightly stronger model in which processors know the number of the round, in every given moment. In this stronger model, Protocol H can be modified into H' by simplifying round $p = kn$ for every $k \geq 0$; the $2n$ messages (used to detect the end of one phase and initiate the next) are not required here, so

$$\bar{M}_H = \bar{M}_{H'} + 2n. \qquad (*)$$

Assume that U is a balanced distribution protocol using an average of $\bar{M}_U \leq \bar{M}$ messages per round. Consider a sequence of n protocols U_i, $0 \leq i \leq n$, where each U_i is defined as follows: in round $0, \ldots, i-1$ behave as U, and from then on (rounds $i, \ldots, n-1$) behave as H'. (These protocols are well-defined assuming a given U and the above assumption on the model.)

Lemma 4: *For every* $0 \leq i \leq n-1$, $\bar{M}_{U_i} \leq \bar{M}_{U_{i+1}}$.

Proof: Consider some specific sequence of packet arrivals, and the corresponding runs of U_i and U_{i+1}. In rounds 0 to $i-1$ both protocols act the same. In round i, U_i stores the $(i+1)$st packet at the first available node, v. If U_{i+1} stores the arriving packet at the same node, then since the behavior of the two protocols through the rest of the phase is identical too, both use the same number of messages in their runs. Now suppose U_{i+1} stored the packet at some other node w. This means that U_{i+1} has already spent $d(v, w)$ more messages than U_i. By Lemma 3, since both protocols behave like Protocol H' through the rest of the run, the number of messages used by U_{i+1} in the remainder of the run less the number used by U_i is either $-d(v, w)$ or $d(w, v)$. Hence over all the run either both protocols used the same number of messages or U_{i+1} used n more messages than U_i. Hence the lemma follows. ∎

We now complete the proof of the lower bound. By Lemma 4 and $(*)$,

$$\bar{M}_H - 2n = \bar{M}_{H'} = \bar{M}_{U_0} \leq \bar{M}_{U_n} = \bar{M}_U.$$

Hence H spends at most $2n$ more messages per phase than U. This amortizes to 2 per round, so $\bar{M} \leq \bar{M}_U + 2$, which proves

Theorem 5: *For unidirectional rings, Protocol H has average (amortized) message complexity that is optimal up to lower order terms.* ∎

5. Eliminating the assumption of rounds

So far we adopted the assumption made in [1] regarding packet arrival rate, namely, exactly one packet arrives in each round, and the rounds are long enough to enable the processors to complete processing one packet before the next one arrives. This assumption may be unrealistic in many cases, and one would usually like to assume a random arrival rate, and in particular, allow two packets to arrive different nodes at the same time.

It is possible to relax this assumption and replace it with weaker ones. We make the following assumptions. First, the propagation time of a message over an edge is one time unit (so the entire "round trip" around the ring takes n time units, and a "phase" of the algorithm takes about $O(n^2)$ time units). Second, the channels obey the FIFO rule, i.e., messages arrive at the other side of a channel in the order in which they are sent. Third, packet arrival rate is random but "small enough." That is, packets arrive each processor v according to an exponential interarrival-time probability density with mean arrival rate $\lambda_v \leq n^{-2}$. Therefore the mean arrival rate for the entire ring is $\lambda \leq n^{-1}$, so there is enough time (on average) for handling each single packet.

This relaxation requires us to change the algorithm somewhat in order to ensure its correctness. To see where a problem might arise, consider the following scenario. Suppose the system is in *full* state, i.e., all flag bits $b(v)$ are 1. Suppose further that two different processors u and v get a new packet and issue reset messages $\langle reset_u \rangle$ and $\langle reset_v \rangle$, respectively. A third node w may get the first message $\langle reset_u \rangle$, reset $b(w)$ to 0, and then get a new packet and store it locally, setting $b(w)$ to 1 again. Upon receiving the second reset message $\langle reset_v \rangle$, c will erroneously reset its bit once again.

Thus it is clear that we need to make sure that only one node may issue a $\langle reset \rangle$ message in the special reset round. This can be achieved by simply designating a single node (say, 1) which will be responsible for resetting the system. The reset process will now compose of three principal stages, instead of the previous two. In the, "sensing" stage, a processor senses a full state by receiving back a packet it sent around the ring. In the second stage, each node sensing a full state sends a $\langle reset\_request \rangle$ message to node 1. In the third stage, node 1 issues a $\langle reset \rangle$ message which travels around the ring. After resetting, the processors currently holding packets handle them as before; in particular, a processor receiving two or more packets during the reset round stores one locally and forwards the rest. (The FIFO assumption on channels is used here to prevent faulty scenarios such as the one illustrated earlier.)

Let us now consider the complexity analysis. Under the new assumption it makes no sense to talk about the message complexity of a round, and we need to discuss the complexity of a *phase*. Just as before, in the *worst case* the cost of a phase may be $O(n^2)$. In fact, in this setting the worst case cost of the reset round alone may be as high as $O(n^2)$. This might happen if n packets arrive simultaneously when the system is in full state. In this case, $O(n)$ messages are wasted by

each processor just in order to realize that a reset is necessary, which sums up to an overall of $O(n^2)$ messages in the phase. However, the *average* complexity of the reset round is not seriously affected, since in most phases only few packets will arrive when the system is in full state.

In order to analyze the average message complexity $\bar{M}$ of a phase, we need again to distinguish between the "main" part of the phase and the reset round, and treat their average complexities $\bar{M}_m$ and $\bar{M}_r$ separately. Consider some particular phase, and let $\bar{X} = (X_1, \ldots, X_n)$ be the n-tuple of packets that arrived during the main part of the phase. Let $\bar{v}^A = \{v_1^A, \ldots, v_n^A\}$ be the tuple of *arrival processors* (i.e., such that packet X_i was received by processor v_i^A). Let $\bar{t}^A = \{t_1^A, \ldots, t_n^A\}$ be the *arrival times* of the packets. Similarly, denote the tuple of *storing processors* (in which the packets were stored) and the *storing times* of the packets under Protocol H by $\bar{v}^S = \{v_1^S, \ldots, v_n^S\}$ and $\bar{t}^S = \{t_1^S, \ldots, t_n^S\}$, respectively. For the sake of the analysis we assume that all these times are distinct, although they may be arbitrarily close (obviously this does not cause any loss of generality). It is clear that the tuples $\bar{v}^S$ and $\bar{t}^S$ are determined by the tuples $\bar{v}^A$ and $\bar{t}^A$. Consequently the number of messages used during the main part of the phase is a function of $\bar{v}^A$ and $\bar{t}^A$, $M_m = M_m(\bar{v}^A, \bar{t}^A)$. We now claim that in fact, M_m is completely determined by the multiset of arrival processors alone, and is independent of the order and timing of the packet arrivals.

We prove this claim in two stages. Let T denote the set of all possible n-tuples of arrival times, and let TR denote the set of all n-tuples of arrival times obeying the "rounds" assumption, i.e., such that $|t_i - t_j| > n$ for every $1 \leq i \leq j \leq n$. Fix the tuple $\bar{v}^A$ of arrival processors.

Lemma 6: *For every two tuples* $\bar{t}^1, \bar{t}^2 \in TR$, $M_m(\bar{v}^A, \bar{t}^1) = M_m(\bar{v}^A, \bar{t}^2)$.

Proof: Clearly, the exact interarrival time between consecutive packets does not make any difference here, and we only need to worry about the order in which the packets arrive. Order the tuples $\bar{X}$, $\bar{v}^A$ and $\bar{t}^1$ by increasing order of arrival time (i.e., such that $t_i^1 < t_{i+1}^1$ for every i). It suffices to look at adjacent "exchange" permutations, i.e., tuples $\bar{t}^2 \in TR$ which are identical to $\bar{t}^1$ except that t_i^1 and t_{i+1}^1 are interchanged in $\bar{t}^2$ for some i. For this case the claim can be proved by an argument similar to that of Lemma 3. ∎

Finally the claim is extended to any tuple of arrival times.

Lemma 7: *For every tuple* $\bar{t}^A \in T$ *there exists a tuple* $\bar{t}^2 \in TR$ *such that* $M_m(\bar{v}^A, \bar{t}^A) = M_m(\bar{v}^A, \bar{t}^2)$.

Proof: The tuples $\bar{v}^A$ and $\bar{t}^A$ determine the tuples $\bar{v}^S$ and $\bar{t}^S$ of storing processors and times. Order the tuples $\bar{X}$, $\bar{v}^A$, $\bar{t}^A$, $\bar{v}^S$ and $\bar{t}^S$ by increasing order of storing time (i.e., such that $t_i^S < t_{i+1}^S$ for every i). Now choose $t_i^2 = 2ni$ for every i. It is easy to verify (e.g., inductively) that the tuple of storing processors does not change, hence M_m is not affected either. ∎

Consequently, since all tuples $\bar{v}^A$ of arrival processors are given equal probability, the analysis

of Section 3 is valid in the new relaxed setting, and $\bar{M}_m = O(n^{1.5})$.

We now turn to the reset round. Consider a particular reset round and let v be the first node to sense a full state. The maximum number of steps since v received its packet until the reset round is completed is $3n$. Let K denote the number of nodes which received a packet (or several packets) from their external source during these $3n$ steps. The average number $\bar{M}_r$ of messages sent during the reset round given K is bounded above by $\bar{M}_r \leq (K+1)\lceil \frac{n}{2} \rceil + n$. Using the Poisson law for $M/M/1$ queues we get that for $1 \leq k \leq n-1$, $Prob(K=k) = P_k(3n) = (\lambda \cdot 3n)^k e^{-\lambda \cdot 3n}/k!$, thus the expected value of $\bar{M}_r$ is bounded above by

$$\bar{M}_r \leq \sum_{k=0}^{n-1} Prob(K=k)\left((K+1)\left\lceil \frac{n}{2} \right\rceil + n\right) \leq \frac{3n}{2} + \frac{n}{2}\sum_{k=0}^{n-1} \frac{(\lambda \cdot 3n)^k e^{-\lambda \cdot 3n}}{k!} \cdot k,$$

so taking $\lambda = n^{-1}$ we get $\bar{M}_r = O(n)$, and the overall average message complexity remains $\bar{M} = O(n^{1.5})$ messages per phase.

Acknowledgement

We thank the anonymous referees for their comments and suggestions.

References

[1] H.L. Bodlaender and J. van Leeuwen, Distribution of Records on a Ring of Processors, Tech. Report RUU-CS-86-6, University of Utrecht, Utrecht, the Netherlands, March 1986.

[2] D.E. Knuth, *The Art of Computer Programming, Vol. 3*, Sect. 6.4, Addison-Wesley, Reading, Mass., 1973.