Efficient Emulation of Single-Hop Radio Network with Collision Detection on Multi-Hop Radio Network with no Collision Detection

Reuven Bar-Yehuda *Oded Goldreich* *Alon Itai*
Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, ISRAEL

ABSTRACT

This paper presents an efficient randomized emulation of *single-hop* radio network *with* collision detection on *multi-hop* radio network *without* collision detection. Each step of the single-hop network is emulated by $O((D+\log\frac{n}{\varepsilon})\log\Delta)$ rounds of the multi-hop network and succeeds with probability $\geq 1-\varepsilon$. (n is the number of processors, D the diameter and Δ the maximum degree). It is shown how to emulate any polynomial algorithm such that the probability of failure remains $\leq \varepsilon$.
A consequence of the emulation is an efficient randomized algorithm for choosing a leader in a multi-hop network.

The second author was partially supported by grant No. 86-00301 from the United States - Israel Bi-national Science Foundation (BSF), Jerusalem, Israel.

1. INTRODUCTION

The purpose of this paper is to present a close relation between several different models of radio communication. In particular, the most restricted model is emulated by the most general one.

The easiest model to work with (i.e. for developing protocols) is that of the Ethernet [6, 7]. In this model all processors share a single "broadcast channel", through which they communicate in synchronous rounds*. In each round, each processor may place a message on the channel. In case a single processor placed a message on the channel, all processors receive it at the end of the round. If no processor placed a message, all processor receive nothing. In case two or more processors placed messages on the channel, all processors, including the transmittors, receive noise at the end of the round. This last feature is called *Collision Detection*, and is hereafter abbreviated by CD.

The Ethernet is a good model for local networks, but is somewhat unrealistic for describing radio communication between distant stations. A better model for distant radio communication postulates that the underlying graph of the network is an arbitrary connected graph, rather than a complete graph (as in the Ethernet model). Such a network is called *multi-hop* (contrasted with the Ethernet model which is *single-hop*).

As before, communication in this network proceeds in (synchronous) rounds in which each processors can act either as a transmitter or as a receiver. A processor acting as a receiver receives a message only when some of its neighbors transmit. When one neighbor transmits then the message is indeed received. When more than one neighbor transmits the receiver may either receive one of the messages transmitted, receive noise or not receive anything. In particular, a receiver cannot necessarily distinguish the case of no transmission from the case of multi-transmission; namely, there is no Collision Detection (CD). The absence of CD characterizes noisy networks since the noise does not allow a processor to distinguish no transmission from multi-transmission. Note that the presence of CD does not invalidate our results, it has not been postulated because the proposed protocols do not make use of it. For the same reason

---

* The term Ethernet is sometimes used in a broader sense to describe a network consisting of several segments. Here the term Ethernet refers to the basic single segment network.

we do not require the processors to have (unique) identities.

Due to the practical importance of multi-hop radio networks, see [7 section IVa, 8, 9 section 6.1.2], the development of protocols for the second model is of interest. However, developing such protocols is a complicated task. The difficulties emerge both from the unknown topology of the network and from the absence of CD mechanism. A useful methodology is to first design protocols for the Ethernet and then emulate them to get protocols for the multi-hop radio network without CD. This methodology becomes more attractive if this compilation can be done automatically without losing too much efficiency.

In this paper, we present an efficient emulation of the Ethernet on arbitrary (multi-hop) radio networks without CD. In section 2 we show how to emulate a single Ethernet step, then in section 3 we show how to implement any Ethernet protocol.

Throughout the paper n denotes the number of processors, Δ and D the maximum degree and diameter of the multi-hop network, respectively. All logarithms are to base 2.

1.1. Previous Work Chlamtac and Kutten [4] showed that, given a network and a designated source, finding an optimal broadcast schedule (i.e., broadcast schedule which uses the minimum number of rounds) is NP-Hard. Chlamtac and Weinstein [5] presented a polynomial-time (centralized) algorithm for constructing a broadcast schedule which uses $O(D\log^2 n)$ time-slots. This centralized algorithm can be implemented in a distributed system assuming the availability of special control channels, but the number of control messages sent may be quadratic in the number of nodes of the network [10].

Bar-Yehuda et al. [2] described a randomized single-source broadcast protocol. To ensure that with probability $1-\varepsilon$ all nodes receive the message the protocol requires an average of $O((D + \log\frac{n}{\varepsilon})\log\Delta)$ time slots. For $D=O(1)$, they have also shown a $\Omega(n)$ lower bound for deterministic protocols. Thus, for this problem there exist randomized protocols that are much more efficient than any deterministic one.

Alon et al. [1] presented networks with diameter $D=2$ in which every broadcast schedule has length $\Omega(\log^2 n)$. The randomized protocol of [2] is thus optimal for these networks. Alon et al. also showed how to emulate a point-to-point *message-passing* model on a radio network. The main difference between the message-passing model and a (multi-hop) radio network (with CD is) that in the first model a processor must receive all messages sent to it by its neighbours in the current round, while in radio network receipt of messages in only required in case of no conflict. Thus the emulators described in [1] address a completely different model and are not applicable for our setting.

In [3], Bar-Yehuda et al. discuss several other radio communication tasks; namely, they study multiple broadcast and point-to-point communication. Efficient probabilistic protocols of the Las Vegas type (i.e. no error in case an acknowledgement is received) are presented. In particular, k point-to-point requests are handled in $O((k+D)\log\Delta)$ rounds (on the average), and k broadcast requests are handled in $O((k+D)\log\Delta\log n)$ rounds.

1.2. Our Results.

Our main result is a probabilistic emulation of a single round of a single-hop radio network with CD on an arbitrary multi-hop radio network without CD mechanism. The emulation fails if there exists a single vertex which did not receive the broadcast. The emulation of a single round requires $O(B_\varepsilon)$ time slots, where $B_\varepsilon = O((D+\log\frac{n}{\varepsilon})\log\Delta)$ is the time required to implement broadcast on the underlying multi-hop network, such that the probability of failure is bounded by ε

A simpler and more efficient method for implementing a CD mechanism on an arbitrary multi-hop network is also presented. The emulation of one round with CD mechanism, on a network with the same topology but without CD, requires $O(\log(1/\varepsilon)\log\Delta)$ (where ε is the probability that there exists a vertex that did not detect the collision).

1.3. Subprotocols Used. Our emulations uses two protocols *Decay* and *Broadcast*, first discussed in [2].

Decay is a protocol that enables a processor to receive, with probability greater than ½ , a message sent by one of its neighbours regardless of the number of neighbours wishing to send it a message.

procedure $Decay(m)$;
repeat at most $2\log\Delta$ times
transmit m to all neighbors;
flip $coin \in_R \{0,1\}$
until $coin = 0$;
wait until round $2\log\Delta$.

Decay is a probabilistic protocol, with the following properties:

(1) It consists of $2\log\Delta$ rounds.

(2) If several neighbors of a node v use *Decay* to send messages then with probability greater than ½ the node v receives one of the messages.

(3) *Decay* is oblivious of the contents of the messages sent.

The second protocol is *Broadcast* [2]. It makes use of *Decay* and has the following properties:

(1) It terminates within $B_\varepsilon = O((D + \log\frac{n}{\varepsilon})\log\Delta)$ rounds.

(2) If several nodes initiate *Broadcast* at round 0 then at round B_ε with probability $> 1-\varepsilon$ each node has received a message of one of the initiators. (In particular, if there is only one initiator then all the nodes received the same message.)

(3) *Broadcast* is oblivious of the contents of the message.

2. THE EMULATION OF A SINGLE ROUND.

Consider the following two models of radio networks.

Model 1: Complete graph with conflict detection.

Model 2: Arbitrary connected graph, in case of a conflict at vertex v any of the the following may occur:

(1) The conflict is detected, or
(2) v receives one of the messages, or
(3) v does not receive any message and is not aware that any message has been sent.

Aim: Show how one round in Model 1 can be emulated by several rounds in Model 2.

2.1 The emulation procedure.

An *initiator* is a processor which wishes to transmit in the current round (of Model 1). Let ε be the desired bound on the failure probability and $k = \lceil 2.5 \log(3/\varepsilon) \rceil$.

For each processor v we use the following variables

msg_v	A message v wishes to transmit in the current round (of Model 1).
$conflict_v$, m_v	Output variables holding the result of the current round: $conflict_v$ is a Boolean variable while m_v is assigned messages.
tag_v , t_v	k-bit variables used in the program.

The emulation consists of three phases: *propagation, detection* and *notification*.

Propagation: Each initiator v selects tag_v at random in $\{0,1\}^k$ and initiates *Broadcast* of the pair (msg_v, tag_v) with error probability $\varepsilon/3$. Every processor sets (m_v, t_v) to be the first message received during this phase (for initiators we take the natural convention of setting (m_v, t_v) to (msg_v, tag_v)).

Detection: The purpose of this phase is to detect if there was more than one initiator. To this end, each initiator and each processor v that received a message during the propagation phase, proceeds as follows:

$conflict_v \leftarrow 0$
for $i \leftarrow 1$ **to** k **do**
  if the i-th bit of t_v is 1
    then $Decay(t_v)$
    else if v receives a message during the next $2\log\Delta$ rounds
    then it sets $conflict_v \leftarrow 1$.

(The processors that did not receive a message remain silent throughout the entire detection phase.)

Notification: Each processor, v, having $conflict_v = 1$ initiates a broadcast of a standard message (e.g., ‘‘conflict’’). The processors use *Broadcast* with failure probability $\varepsilon/3$. A processor, u, receiving this (‘‘conflict’’) message sets $conflict_u \leftarrow 1$. (In case no such message was received, $conflict_u$ remains unchanged.)

2.2 Analysis

Lemma 1: Suppose there is at least one initiator and that the propagation phase succeeded (every vertex received at least one message). Then

(i) If there is a single initiator, denoted u, then at the end of the detection phase for all vertices v, $conflict_v = 0$ and $m_v = msg_u$.

(ii) If there is more than one initiator then with probability $\geq 1-\varepsilon/3$ at the end of the detection phase there exists a vertex v for which $conflict_v = 1$.

Proof:

(i) If there is a single initiator (u) all vertices v have the same value of t_v (which equals tag_u). Thus in every iteration of the detection phase, either all the vertices participate in *Decay* or all of them listen. Therefore, whenever a vertex listens no vertex transmits and the value of $conflict_v$ remains 0. Clearly, in this case $m_v = msg_u$.

(ii) Assume that there was more than one initiator. Since by assumption the propagation phase succeeded, every vertex received a message from some initiator. Since the network is connected, there exist two adjacent processors u and v which have received messages initiated by two different processors r and s (u may be equal to r – this is the case iff u is an initiator itself). It follows that $t_u = tag_r$ and $t_v = tag_s$. Property (3) of *Broadcast* states that it is oblivious of the contents of the messages. Therefore, the distribution of tag_u and tag_v is independent of the fact that these messages have reached u and v respectively. Thus, for every i, with probability ½, the strings t_u and t_v differ in the i-th bit. In this case, in the i-th iteration of the detection phase, one of them, say u, transmits and the other, v, listens, and there is probability $\geq$ ½ that the listening vertex, v, received a message, whereupon it sets $conflict_v$ to 1. Thus with probability $\geq$ ¼ during some iteration, some $conflict_w$ variable is set to 1. The probability that this did not occur during any iteration is $\leq (1-\frac{1}{4})^k \leq \frac{3}{4}^{2.5\log(3/\varepsilon)} < \varepsilon/3$. □

Lemma 2: The entire protocol requires $(2+o(1))B_\varepsilon$ rounds.

Proof: The propagation and notification phases consist of executing *Broadcast*, which by [2] can be implemented in $O(B_\varepsilon)$ rounds. The detection phase consists of $k = \lceil 2.5\log(3/\varepsilon) \rceil$

iterations of *Decay* each of which requires $2\log\Delta$ rounds. Thus, the time is dominated by the number of rounds required for *Broadcast*. □

Theorem 1: Let the vertices of Model 2 follow the above protocol. If there is a single initiator then with probability $\geq 1-\varepsilon/3$ all the vertices receive its message, and if there are more initiators then with probability $\geq 1-\varepsilon$ all vertices detect a conflict. The above protocol requires $O(B_\varepsilon) = O((D+\log(n/\varepsilon))\log\Delta)$ time.

Proof: In case there is one or more initiators, *Broadcast* is executed in the propagation phase and it may fail with probability $\leq \varepsilon/3$. In case there are several initiators, conflict detection (in the second phase) fails with probability $\leq \varepsilon/3$. The failure probability in the (possible) broadcast of the notification phase is again bounded above by $\varepsilon/3$. Hence the theorem. □

2.3 Implementing a CD mechanism in arbitrary multi-hop networks.

In this subsection, we take a small detour, presenting a method for implementing a CD mechanism in arbitrary multi-hop radio networks. Namely, we show how to probabilistically emulate the following model $1'$ on Model 2:

Model $1'$: A multi-hop network where conflicts are detected with probability $1-\varepsilon$ at the potential receiver.

Model 2: A network with the same underlying graph but no guarantee whatsoever concerning conflict detection.

Let $k = \lceil 2.5\log(3/\varepsilon) \rceil$. The emulation of a single round of Model $1'$ proceeds as follows. Processor v wishing to transmit a message msg_v in the current round (of Model $1'$) selects uniformly $tag_v \in \{0,1\}^k$ and repeats $Decay(msg_v, tag_v)$ for k times. Processor u acting as a receiver in the current round (of Model $1'$) listens during the $2k\log\Delta$ rounds (i.e. the duration of k executions of *Decay*) and sets $conflict_u \leftarrow 1$ if it heard two messages with different tags (otherwise $conflict_u$ remains 0). Processor u sets m_u to be the message field in the first message it has received during the above rounds.

The reader may easily verify that the above procedure guarantees collision detection (at a single processor in one round) with probability $> 1-\varepsilon$.

In Model $1'$, collision is detected at the potential receiver. Model $1'$ can easily emulate the more standard model in which collisions are detected by the transmittors: Each round of the standard model is emulated by two rounds of Model $1'$. If there are collisions they are detected by the potential receivers in the first round. In the next round, the processors that detected a collision transmit that a collision occurred (and the remaining processors remain silent). Hence in the second round, the original transmittors either receive a "collision occurred" message or detect a collision. In either case, they deduce that a collision occurred in the first round.

3. EMULATING AN ENTIRE ALGORITHM

In the previous section we presented a probabilistic protocol to emulate a single round of a single-hop network with CD (i.e., Model 1) by a multi-hop radio network with no CD (i.e., Model 2). The simulation is probabilistic and the probability of failure is bounded by a parameter ε. For a given ε the number of rounds is $S_\varepsilon < 3B_\varepsilon = O((D + \log\frac{n}{\varepsilon})\log\Delta)$.

In general, the emulation of an algorithm (designed for Model 1) on Model 2, requires several such rounds. If the probability of error for a single round is ε', then the probability that the entire t rounds of the algorithm are error free is $(1-\varepsilon')^t > 1-t\varepsilon'$. To make sure that the algorithm succeeds with probability ε, we must choose $\varepsilon' = \varepsilon/t$. Thus a single round requires $O((D + \log\frac{n}{\varepsilon'})\log\Delta) = O((D + \log\frac{n}{\varepsilon/t})\log\Delta) = O((\log t + D + \log\frac{n}{\varepsilon})\log\Delta) = O(S_\varepsilon + \log t \log\Delta)$, and the entire algorithm requires $O(t(\log t\log\Delta + S_\varepsilon))$.

In order to proceed in such a simulation, one needs to precompute the value of ε', using an upper bound on the running time of the algorithm. In some cases, such a bound may not be known a priori. An adaptive approach, which does not require a priori knowledge of the running time, is to gradually decrease the ε' used in the single-round emulation.

Let $\varepsilon_i = \frac{6\varepsilon}{\pi^2 i^2}$. In the emulation of the i-th round of the algorithm, we use ε_i instead of ε'. Thus the i-th round requires $S_{\varepsilon_i} = O((D + \log\frac{n}{\varepsilon_i})\log\Delta) = O((D + \log\frac{n}{\varepsilon/i^2})\log\Delta) =$

$O((2\log i + D + \log\frac{n}{\varepsilon})\log\Delta)$. Consequently, the emulation of a t-round algorithm requires

$O(t(\log t + D + \log\frac{n}{\varepsilon})\log\Delta) = O(t(\log t \log\Delta + S_\varepsilon))$.

The probability that the i-th round failed is ε_i. Thus the probability that no round failed is

$$\prod_{i=1}^{t}(1-\varepsilon_i) > 1 - \sum_{i=1}^{t}\varepsilon_i = 1 - \sum_{i=1}^{t}\frac{6\varepsilon}{\pi^2 i^2} > 1 - \sum_{i=1}^{\infty}\frac{6\varepsilon}{\pi^2 i^2} = 1 - \varepsilon.$$

Theorem 2: For every ε, any Ethernet algorithm can be probabilistically emulated on a multi-hop network with no CD, such that

(1) The emulation fails with probability $\leq\varepsilon$;

(2) The emulation takes $O(t(\log t \log\Delta + B_\varepsilon))$, where t is the round complexity of the Ethernet algorithm and $B_\varepsilon = O((D + \log\frac{n}{\varepsilon})\log\Delta)$ is the round complexity of broadcast (in Model 2).

Note that an Ethernet algorithm with round complexity polynomial in the network size or in the inverse of the error probability (i.e. $t = poly(n/\varepsilon)$) can be emulated at an average cost of $O(B_\varepsilon)$ rounds per each round of the algorithm.

4. APPLICATIONS AND CONCLUSIONS

We now show how the emulation of section 3 can be used to choose a leader in a multi-hop radio network. Willard [11] proposed an algorithm for leader election in the Ethernet model. The algorithm uses conflict detection and requires $O(\log\log n)$ rounds. Applying our emulation yields a probabilistic algorithm of $O(B_\varepsilon \log\log n)$ rounds for leader election in a multi-hop radio networks with no CD.

Our emulation results are general and apply to all Ethernet algorithms. In many cases (e.g. for polynomial-time algorithms) the overhead of the emulation is merely a multiplicative factor of the time required to complete broadcast on the underlying (multi-hop) network. As a general result this seems the best possible, but it may be improved in special cases. In particular, it is interesting to investigate whether the $O((D + \log\frac{n}{\varepsilon})\log\Delta \log\log n)$ round algorithm for leader election in arbitrary radio networks can be improved.

Acknowledgements It is a pleasure to thank the anonymous referees for their helpful comments.

REFERENCES

(1) Alon N., Bar-Noy A., Linial N. and Peleg D., "On the complexity of radio communication", *21st STOC*, pp. 274-285, 1989.

(2) Bar-Yehuda R., Goldreich O. and Itai A., "On the time complexity of broadcast in radio networks: an exponential gap between determinism and randomization", 6th Symposium on Principles of Distributed Systems, (Aug. 1987). To appear in J. Comp. and System Science.

(3) Bar-Yehuda R., Israeli A. and Itai A., Multiple Communication in Multi-Hop Radio Networks, 8th Symposium on Principles of Distributed Systems, (Aug. 1989). To appear in SIAM J. Comp.

(4) Chlamtac, I. and Kutten, S., "On Broadcasting in Radio Networks- Problem Analysis and Protocol Design", December (1985), Vol COM-33, No. 12.

(5) Chlamtac, I. and Weinstein O., "The wave expansion approach to broadcasting in multihop radio networks", *INFOCOM* (April 1987).

(6) Digital-Intel-Xerox, "The Ethernet data link layer and physical layer specification 1.0" (Sept. 1980).

(7) Gallager, R., "A perspective on multiaccess channels", *IEEE Trans. on Inf. Theory*, Vol. IT-31 (1985), 124-142.

(8) L.G. Roberts, "Aloha packet system with and without slots and capture", ASS Notes 8, Advanced Research Projects Agency, Network Information Center, Stanford Research Institute, Stanford, June 1972.

(9) A.C. Tanenbaum, *Computer Networks*, Prentice-Hall, Inc., 1981.

(10) Weinstein O., "The wave expansion approach to broadcasting in multihop radio networks", M.Sc. Thesis, Computer Science Dept., Technion, Haifa, Israel, (1987).

(11) Willard D.E., "Log-logarithmic selection resolution protocols in a multiple access channel", SIAM J. on Comput. 15(2), 468-477, (1986).