Contention Resolution with Constant Expected Delay[*]

Leslie Ann Goldberg[†] Philip D. MacKenzie[‡] Mike Paterson [§]
Aravind Srinivasan[¶]

March 25, 1998

Abstract

We study contention resolution in a multiple-access channel such as the Ethernet channel. In the model that we consider, n users generate messages for the channel according to a probability distribution. Raghavan and Upfal have given a protocol in which the expected *delay* (time to get serviced) of every message is $O(\log n)$ when messages are generated according to a Bernoulli distribution with generation rate up to about $1/10$. We present a protocol in which the expected average message delay is $O(1)$ when messages are generated according to a Bernoulli distribution with a generation rate smaller than $1/e$. To achieve this result we first consider an analogous model in which users are synchronized (i.e., they agree about the time), there are potentially an infinite number of users, and messages are generated according to a Poisson distribution with generation rate up to $1/e$. (Each message constitutes a new user.) We give a protocol in which the expected delay of any message is $O(1)$. Next we show how to simulate the protocol using n synchronized users. Finally, we show how to build the synchronization into the protocol.

[*]Preliminary versions of this work appeared in a paper written by the third and fourth authors (*Proc. IEEE Symposium on Foundations of Computer Science*, pages 104–113, 1995), and in a paper written by the first and second authors (*Proc. IEEE Symposium on Foundations of Computer Science*, pages 213–222, 1997).

[†]`leslie@dcs.warwick.ac.uk`. Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom. This work was supported by EPSRC Research Grant GR/L60982 — Design and Analysis of Contention-Resolution Protocols, by ESPRIT Project 21726 — RAND-II and by ESPRIT LTR Project 20244 — ALCOM-IT.

[‡]`philmac@cs.idbsu.edu`. Department of Mathematics and Computer Science, Boise State University, Boise, ID, 83725. Part of this work was performed at Sandia National Labs and supported by the U.S. Dept. of Energy under contract DE-AC04-76DP00789.

[§]`msp@dcs.warwick.ac.uk`. Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom. This work was supported by ESPRIT LTR Project 20244 — ALCOM-IT.

[¶]`aravind@iscs.nus.edu.sg`. Department of Information Systems and Computer Science, National University of Singapore, Singapore 119260. Part of this work was supported by ESPRIT LTR Project 20244 — ALCOM-IT, and was done while this author was visiting the University of Warwick. Part of this work was done while visiting the Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. Part of this work was supported by National University of Singapore Academic Research Fund Grants RP960620 and RP970607, and was done at the National University of Singapore.

1 Introduction

A *multiple-access channel* is a broadcast channel that allows multiple users to communicate with each other by sending messages onto the channel. If two or more users simultaneously send messages, then the messages interfere with each other (collide), and the messages are not transmitted successfully. The channel is not centrally controlled. Instead, the users use a *contention-resolution protocol* to resolve collisions. Although the most familiar multiple-access channels are local-area networks (such as the Ethernet network) which are implemented using cable, multiple-access channels are now being implemented using a variety of technologies including optical communications. Thus, good contention-resolution protocols can be used for communication between computers on local-area networks, for communication in optical networks, and (therefore) for simulating shared-memory parallel computers (such as PRAMs) on optical networks.

Raghavan and Upfal considered the model in which n users generate messages according to a Bernoulli distribution with total generation rate up to about 1/10 [21]. (More details about the arrival distribution are given in Section 1.1.) They gave a protocol in which the expected *delay* (time to get serviced) of every message is $O(\log n)$. Using the same model, we present a protocol in which the expected average message delay is $O(1)$ provided that the total generation rate is sufficiently small (less than $1/e$). We derive our result by considering an analogous model in which users are synchronized (i.e., they agree about the time), the number of users is potentially infinite, and messages arrive according to a Poisson distribution with parameter up to about $1/e$. Each message constitutes a new user. We give a protocol in which the expected delay of any message is $O(1)$. The synchronizing of our users allows our protocol to use different time steps for different purposes. Thus, for example, those time steps that are equal to 1 modulo 2 can be used for messages making their first attempt, time steps equalling 2 modulo 4 can be used for messages making their second attempt, and so on. The partitioning of time steps is what makes it possible to have bounded expected delay.

Once we have proved that the expected delay of each message is $O(1)$, we show how to simulate the protocol using n synchronized users. Here each user is responsible for a potentially infinite number of messages (rather than for a single message) and the difficult part is dealing with all of the messages in constant time.

The analysis of our n-user protocol requires the n users to have synchronized clocks. We next show how to simulate the synchronized clocks (for reasonably long periods of time) by building synchronization into the protocol. Thus, our final protocol consists of "normal" phases in which the users are synchronized and operating as described above and "synchronization phases" in which the users are synchronizing. The synchronization phases are robust in the sense that they can handle pathological situations (such as users starting in the middle of a synchronization phase). Thus, we are able to achieve constant expected message delay even for models in which users are allowed to start and stop (see Section 1.1 for details).

1.1 The Multiple-Access Channel Model

Following previous work on multiple-access channels, we work in a *time-slotted* model in which time is partitioned into intervals of equal length, called *steps*. During each step, the users generate messages according to a probability distribution. During each step, each user may attempt to send at most one message to the channel. If more than one attempt is made during a given time step, the messages collide and must be retransmitted. If just a single user attempts to send to the channel, it receives an acknowledgment that the transmission was successful. Users must queue all unsuccessful messages for retransmission and they use a contention-resolution protocol to decide when to retransmit.

In the *Synchronized Infinitely-Many Users Model*, there is a single parameter λ. The number of messages generated at each step is determined according to a Poisson distribution with parameter λ. Each message is deemed to be a new user. After a user has sent its message successfully, it leaves the system.

There are two variants of the *Finitely-Many Users Model.* In both variants, there are n users. The first variant (which we consider in Section 3) is the *Synchronized Finitely-Many Users Model.* In this model, the n users are synchronized and they all run for the entire time that the protocol is running. When we consider this model, we will need to consider a variety of message-arrival distributions. In particular, we will say that an arrival distribution is $\{\lambda_i\}_{1\le i\le n}$-*dominated* (for $\lambda_1, \ldots, \lambda_n > 0$) if $\sum_i \lambda_i$ is sufficiently small (i.e., at most λ for $\lambda < 1/e$) **and** for every user i, every time step t and every event E concerning the arrival of messages at steps other than t or to users other than i, the probability, conditioned on event E, that user i generates a message at step t is at most λ_i.

The second variant of the Finitely-Many Users Model is called the *Unsynchronized Finitely-Many Users Model.* In this model, the n users are not synchronized and are allowed to start and stop over time, provided that each user runs for at least a certain polynomial number of steps every time it starts (see Section 4 for details). We will generalize the definition of a $\{\lambda_i\}_{1\le i\le n}$-dominated distribution so that it applies to this model by stipulating that no messages are generated at users which are stopped. We will be most interested in the $\{\lambda_i\}_{1\le i\le n}$-*Bernoulli* arrivals distribution, in which each user i is associated with a positive probability λ_i and it generates a message independently with probability λ_i during each time step that the user is running. Once again, we require $\sum_i \lambda_i < 1/e$. The result of Raghavan and Upfal applies to any $\{\lambda_i\}_{1\le i\le n}$-Bernoulli arrivals distribution in which $\sum_i \lambda_i \le \lambda'$ where $\lambda' \approx 1/10$.

In the Synchronized Infinitely-Many Users Model we will show that the expected delay of *any* message is $\mathrm{O}(1)$. In the Unsynchronized Finitely-Many Users Model we will show only that the expected *average* delay of messages is $\mathrm{O}(1)$. To be precise, let W_i be the delay of the ith message, and let

$$W_{\mathrm{avg}} = \lim_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} W_i.$$

(Intuitively, W_{avg} is the average waiting time of messages in the system.) We will show that if the message generation rate is sufficiently small (less than $1/e$), then $\mathrm{E}[W_{\mathrm{avg}}] = \mathrm{O}(1)$.

The multiple-access channel model that we have described is *acknowledgment-based* because the only information that a user receives about the state of the channel is the history of its own transmission attempts. (In the Unsynchronized Finitely-Many Users Model, we also assume that the users all know some upper bound on the number of simultaneous live users.) Other models have been considered. One popular model is the *ternary feedback* model in which, at the end of each time step, *each* user receives information indicating whether zero, one, or more than one messages were sent to the channel at that time step. Stable protocols are known [12, 23] for the case in which λ is sufficiently small (at most $.4878\cdots$). However, Tsybakov and Likhanov [22] have shown that, in the infinitely-many users model, no protocol achieves a throughput better than 0.568. (That is, in the limit, only a 0.568 fraction of the time-steps are used for successful sends.) By contrast, Pippenger [20] has shown that if the exact number of messages that tried at each time step is known to all users, there is a stable protocol for every $\lambda < 1$. We believe that the weaker acknowledgment-based model is more realistic for purposes such as PRAM emulation and optical routing and we follow [13, 17, 21] in focusing on this model henceforth.

In this paper we focus on the *dynamic* contention-resolution problem in which messages arrive according to a probability distribution. Other work [17] has focussed on the *static* scenario in which a given set of users start with messages to send. Similar static contention-resolution problems arise in optical routing [4, 8, 9] and in simulating shared memory computers on distributed networks [7, 10, 17].

1.2 Previous work

There has been a tremendous amount of work on protocols for multiple-access channels. Here we will only discuss theoretical results concerning dynamic protocols in the acknowledgment-based model that we use. We refer the reader to the papers cited here and in Section 1.1 for work on protocols using different assumptions or models.

The multiple-access channel first arose in the context of the ALOHA system, which is a multi-user communication system based on radio-wave communication [1]. As we noted earlier, it also arises in the context of local-area networks. For example, the Ethernet protocol [19] is a protocol for multiple-access channels. Much research on multiple-access channels was spurred by ALOHA, especially in the information theory community; see, for example, the special issue of *IEEE Trans. Info. Theory* on this topic [15].

We now give an informal description of a common idea that runs through most known protocols for our problem; this is merely a rough sketch, and there are many variants. In the Infinitely-Many Users Model, consider a newly-born message P. P could try using the channel a few times with some fairly high probability. If it is successful, it leaves the system; if not, then P could guess that its trial probability was "too high", and try using the channel with lower and lower probability until it successfully leaves the system.

One way to formalize this is via *backoff protocols*, which are parameterized by a non-decreasing function $f : \mathbf{Z}^+ \to \mathbf{Z}^+$, where $\mathbf{Z}^+$ denotes the set of non-negative integers. In the Infinitely-Many Users Model, a message P that has made $i \geq 0$ unsuccessful attempts at the channel, will pick a number r uniformly at random from $\{1, 2, \ldots, f(i)\}$, and will next attempt using the channel r time steps from then. If successful, P will leave the system, otherwise it will increment i and repeat the process. In the Finitely-Many Users Model, each user queues its messages and conducts such a protocol with the message at the head of its queue; once this message is successful, the failure count i is reset to 0. If $f(i) = (i+1)^{\Theta(1)}$ or 2^i, then such a protocol is naturally termed a polynomial backoff protocol or a binary exponential backoff protocol, respectively. (The function f, if it exists, must be chosen judiciously: if it grows too slowly, the messages will tend to try using the channel too often, thus leading to frequent collisions and hence long message lifetimes. But if f grows too quickly, the messages will tend to use the channel too infrequently, and again the throughput rate will suffer as messages are retained in the system.)

For our model of interest, the dynamic setting with acknowledgment-based protocols, the earliest theoretical results were negative results for the Unsynchronized Infinitely-Many Users Model. Kelly [16] showed that, for any $\lambda > 0$, any backoff protocol with a backoff function $f(i)$ that is smaller than any exponential function is unstable in the sense that the expected number of successful transmissions to the channel is finite. Aldous [2] showed, for every $\lambda > 0$, that the binary exponential backoff protocol is unstable in the sense that the expected number of successful transmissions in time steps $[1, t]$ is $o(t)$ and that the expected time until the system returns to the empty state is infinite.

In striking contrast to Kelly's result, the important work of [13] showed, among other things, that in the Unsynchronized Finitely-Many Users Model for all $\{\lambda_i\}_{1 \le i \le n}$-Bernoulli distribution with $\sum_i \lambda_i < 1$ all superlinear polynomial backoff protocols are stable in the sense that the expected time to return to the empty state and the expected average message delay are finite. However, they also proved that the expected average message delay in such a system is $\Omega(n)$. Raghavan and Upfal showed that, for any $\{\lambda_i\}_{1 \le i \le n}$-Bernoulli distribution with $\sum_i \lambda_i$ up to about 1/10, there is a protocol in which the expected delay of any message is $\mathrm{O}(\log(n))$ [21]. It is also shown in [21] that, for each member $\mathcal{P}$ of a large set of protocols that includes all known backoff protocols, there exists a threshold $\lambda_{\mathcal{P}} < 1$ such that if $\lambda > \lambda_{\mathcal{P}}$ then $\mathrm{E}[W_{ave}] = \Omega(n)$ must hold for $\mathcal{P}$.

1.3 Our results

We first consider the Synchronized Infinitely-Many Users Model and give a protocol in which the expected delay of any message is $\mathrm{O}(1)$ for message generation rates up to $1/e$. (Note that this arrival rate threshold of $1/e$ is higher than the threshold of approximately 1/10 allowed in [21]. We argue in Section 5 that handling arrival rates greater than $1/e$ is a challenging problem.) As far as we know our protocol is the first acknowledgment-based protocol which is provably stable in the sense of [13]. An interesting point here is that our results are complementary to those of [13]: while the work of [13] shows that (negative) results for the Infinitely-Many Users Model may have no bearing on the Finitely-Many Users Model, our results suggest that better intuition and positive results for the Finitely-Many Users Model may be obtained via the Infinitely-Many Users Model.

Our infinite-users protocol is simple. We construct an explicit, easily computable collection $\{S_{i,t} : i, t = 0, 1, 2, \ldots\}$ of finite sets of nonnegative integers $S_{i,t}$ where, for all i and t, every element of $S_{i,t}$ is smaller than every element of $S_{i+1,t}$. A message born at time t which has made i (unsuccessful) attempts to send to the channel so far, picks a time r uniformly at random from $S_{i,t}$, and tries using the channel at time r. If it succeeds, it leaves the system. Otherwise, it increments i and repeats this process. We give bounds on the probability that the delay of the message is high and we use these bounds to show that the expected number of messages (and hence the expected total storage size) in the system at any given time is $\mathrm{O}(1)$, improving on the $\mathrm{O}(\log n)$ bound of [21].

Once we have proved that the expected delay of each message is $\mathrm{O}(1)$, we show how to simulate the Infinitely-Many Users Protocol using n synchronized users, achieving low expected delay for a variety of message-arrival distributions.

Finally, we consider the Unsynchronized Finitely-Many Users Model. Our earlier analysis required synchronized clocks and we show how to simulate this for reasonably long periods of time by building synchronization into our final protocol. The synchronization is complicated by the fact that the model allows users to start and stop over time.

The structure of our final protocol is simple. Most of the time, the users are simulating our Infinitely-Many Users Protocol from Section 2. The users occasionally enter a synchronizing phase to make sure that the clocks are synchronized (or to resynchronize after a user enters the system). Note that the synchronizing phase has some probability of (undetectably) failing, and thus it must be repeated periodically to guarantee constant expected message delay.

The idea of the "synchronization phase" was inspired by the "reset state" idea of [21]. The key idea that allowed [21] to achieve low expected delay is to have users detect "bad events" and to enter a "reset state" when bad events occur. In some sense, the structure of our protocol (normal phases, occasionally interrupted by synchronization phases) is similar to the structure of [21]. However, there are major differences between them. One difference is that, because lack of synchronization cannot be reliably detected, synchronizing phases must be entered periodically even when no particular bad event is observed. Another difference is that users in a reset state are only allowed to send messages with very low probability, and this helps other users to access the channel. However, our synchronization phase is designed to accomplish the more-difficult task of synchronizing the users (this is needed to obtain constant expected delay rather than logarithmic expected delay), and accomplishing this task requires many transmissions to the channel, which prevent access to the channel by the other users. Thus, synchronization phases are costly in our protocol. A third difference is that in [21] a normal phase always tends towards low expected delay. When bad situations arise, there is a good probability of them being caught, thus causing a reset state to occur. In our protocol, a normal phase tends towards even lower (constant) expected delay if the users are synchronized. However, if they are not synchronized, the normal phase does not necessarily tend towards low expected delay, and there is no sure way to detect that the users are unsynchronized. Thus, the bad situation can only be remedied during the next time the users start a synchronizing phase, which may be after quite a long time! Fortunately, the effects of this type of behavior can be bounded, so we do achieve constant expected message delay.

The synchronizing phase of our protocol is somewhat complicated, because it must synchronize the users even though communication between users can only be performed through acknowledgments (or lack thereof) from the multiple-access channel. The analysis of our protocol is also complicated due to the very dynamic nature of the protocol, with possibilities of users missing synchronizing phases, trying to start a synchronizing phase while one is already in progress, and so on. Our synchronizing phases are robust, in the sense that they can handle these types of events, and eventually the system will return to a normal synchronized state.

To give an idea of the problems that arise when designing a robust synchronization phase, consider the following scenario. Suppose that the set L of all live users enters a synchronization phase and, halfway through it, another set L' of users starts up. Since the users in L' have missed a large part of the synchronization phase, they will not be able to synchronize with the other users. (This seems to be inherent in any conceivable synchronization protocol.) There are two possible approaches for solving this problem. One is to try to design the protocol so that the users in L detect the newly started users during the synchronization phase. Then they must somehow resynchronize with the newly joined users. However, any synchronization protocol must perform various tasks (such as electing a leader) and it is difficult to detect the presence of the users in L' during some of these tasks. A second approach is to allow the users in L to ignore the users in L' and to finish their synchronization phase (either synchronized amongst themselves, or not). Then the set L' of users in the synchronization phase will very likely disrupt the normal operations of the users in L, causing them to synchronize again. But now the users in L' will be about halfway through their synchronization, whereas the users in L are just starting synchronization! Our solution to this problem is a combination of the two approaches, and is described in Section 4.

1.4 Outline

In Section 2 we consider the Synchronized Infinitely-Many Users Model. Subsection 2.1 gives notation and preliminaries. Subsection 2.2 gives our protocol. Subsections 2.3 and 2.4 bound the expected delay of messages. In Section 3 we consider the Synchronized Finitely-Many Users Model and show how to simulate our protocol on this model, achieving bounded expected delay for a large class of input distributions. In Section 4 we consider the Unsynchronized Finitely-Many Users Model. Subsection 4.1 gives notation and preliminaries. Subsection 4.2 gives our protocol. In Section 4.3 we prove the key features of our protocol, namely, a message generated at a step in which no users start or stop soon before or after will have constant expected delay, and a message generated at a step in which a user starts soon before or after will have an expected delay of $O(n^{37})$ steps. In Section 4.4 we show that our protocol achieves constant expected message delay for a fairly general multiple access channel model, with users starting and stopping.

2 The Infinitely-Many Users Protocol

2.1 Notation and Preliminaries

For any $\ell \in \mathbf{Z}^+$, we denote the set $\{1, 2, \ldots, \ell\}$ by $[\ell]$; logarithms are to the base two, unless specified otherwise. In any time interval of a protocol, we shall say that a message P *succeeded* in that interval if it reached the channel successfully during that interval.

Theorem 2.1 presents the Chernoff-Hoeffding bounds [6, 14]; see, e.g., Appendix A of [3] for details.

Theorem 2.1 *Let R be a random variable with* $\mathrm{E}[R] = \mu \geq 0$ *such that either: (a) R is a sum of a finite number of* independent *random variables $X_1, X_2, \ldots$ with each X_i taking values in* $[0,1]$, *or (b) R is Poisson. Then for any $\nu \geq 1$,* $\Pr[R \geq \mu\nu] \leq H(\mu,\nu)$, *where* $H(\mu,\nu) \doteq (e^{\nu-1}/\nu^{\nu})^{\mu}$.

Fact 2.2 is easily verified.

Fact 2.2 *If $\nu > 1$ then $H(\mu,\nu) \leq e^{-\nu\mu/M_\nu}$, where M_ν is positive and monotone decreasing for $\nu > 1$.*

We next recall the "independent bounded differences tail inequality" of McDiarmid [18]. (The inequality is a development of the "Azuma martingale inequality"; a similar formulation was also derived by Bollobás [5].)

Lemma 2.3 ([18, Lemma 1.2]) *Let $x_1, \ldots, x_n$ be independent random variables, with x_k taking values in a set A_k for each k. Suppose that the (measurable) function $f : \prod A_k \to \mathbf{R}$ (the set of reals) satisfies*
$|f(\overline{x}) - f(\overline{x}')| \leq c_k$ whenever the vectors $\overline{x}$ and $\overline{x}'$ differ only in the kth coordinate. Let Y be the random variable $f(x_1, \ldots, x_n)$. Then for any $t > 0$,

$$\Pr\left[|Y - \mathrm{E}[Y]| \geq t\right] \leq 2\exp\left(-2t^2/\textstyle\sum_{k=1}^{n} c_k^2\right).$$

Suppose (at most) s messages are present in a *static* system, and that we have s time units within which we would like to send out a "large" number of them to the channel, with high probability. We give an informal sketch of our ideas. A natural scheme is for each message *independently* to attempt using the channel at a randomly chosen time from $[s]$. Since a message is successful if and only if no other message chose the same time step as it did, the "collision" of messages is a dominant concern; the number of such colliding messages is studied in the following lemma.

Lemma 2.4 *Suppose at most s balls are thrown uniformly and independently at random into a set of s bins. Let us say that a ball* collides *if it is not the only ball in its bin. Then, (i) for any given ball B,* $\Pr[B \text{ collides}] \leq 1 - (1 - 1/s)^{s-1} < 1 - 1/e$, *and (ii) if C denotes the total number of balls that collide then, for any $\delta > 0$,*

$$\Pr[C \geq s(1 - 1/(e(1+\delta)))] \leq F(s,\delta), \text{ where } F(s,\delta) \doteq e^{-s\delta^2/(2e^2(1+\delta)^2)}.$$

Proof: Part (i) is direct. For part (ii), number the balls arbitrarily as $1, 2, \ldots$. Let X_i denote the random choice for ball i, and $C = f(X_1, X_2, \ldots)$ be the number of colliding balls. It is easily seen that, for any placement of the balls and for any movement of any desired ball (say the i^{th}) from one bin to another, we have $c_i \leq 2$, in the notation of Lemma 2.3. Invoking Lemma 2.3 concludes the proof. □

Lemma 2.4 suggests an obvious improvement to our first scheme if we have many more slots than messages. Suppose we have s messages in a static system and ℓ available time slots $t_1 < t_2 < \cdots < t_\ell$, with $s \le \ell/(e(1+\delta))$ for some $\delta > 0$. Let

$$\ell_i(\delta) \doteq \frac{\ell}{e(1+\delta)} \left(1 - \frac{1}{e(1+\delta)}\right)^{i-1} \text{ for } i \ge 1; \tag{1}$$

thus, $s \le \ell_1(\delta)$. The idea is to have each message try using the channel at some randomly chosen time from $\{t_i : 1 \le i \le \ell_1(\delta)\}$. The number of remaining messages is at most $s(1 - \frac{1}{e(1+\delta)}) \le \ell_2(\delta)$ with high probability, by Lemma 2.4(ii). Each remaining message attempts to use the channel at a randomly chosen time from $\{t_i : \ell_1(\delta) < i \le \ell_1(\delta) + \ell_2(\delta)\}$; the number of messages remaining is at most $\ell_3(\delta)$ with high probability (for s large). The basic "random trial" user of Lemma 2.4 is thus repeated a sufficiently large number of times. The total number of time slots used is at most $\sum_{j=1}^{\infty} \ell_j(\delta) = \ell$, which was guaranteed to be available. In fact, we will also need a version of such a scenario where some number z of such protocols are run *independently*, as considered by Definition 2.5. Although we need a few parameters for this definition, the intuition remains simple.

Definition 2.5 *Suppose ℓ, m and z are positive integers, $\delta > 0$, and we are given sets of messages $P_1, P_2, \ldots, P_z$ and sets of time slots $T_1, T_2, \ldots, T_z$ such that: (i) $P_i \cap P_j = \phi$ and $T_i \cap T_j = \phi$ if $i \ne j$, and (ii) $|T_i| = \ell$ for all i. For each $i \in [z]$, let $T_i = \{t_{i,1} < t_{i,2} < \cdots < t_{i,\ell}\}$. Define $\ell_0 = 0$, and $\ell_i = \ell_i(\delta)$ as in (1) for $i \ge 1$.*

Then, RT$(\{P_i : i \in [z]\}, \{T_i : i \in [z]\}, m, z, \delta)$ denotes the performance of z independent *protocols $E_1, E_2, \ldots, E_z$ ("RT" stands for "repeated trials"). Each E_i has m iterations, and its j^{th} iteration is as follows: each message in P_i that collided in all of the first $(j-1)$ iterations picks a random time from $\{t_{i,p} : \ell_0 + \ell_1 + \cdots + \ell_{j-1} < p \le \ell_0 + \ell_1 + \cdots + \ell_j\}$, and attempts using the channel then.*

Remark. Note that the fact that distinct protocols E_i are independent follows directly from the fact that the sets T_i are pairwise disjoint.

Since no inter-message communication is needed in RT, the following definition is convenient.

Definition 2.6 *If $P \in P_i$ in a protocol RT$(\{P_i : i \in [z]\}, \{T_i : i \in [z]\}, m, z, \delta)$, the protocol for message P is denoted P-RT(T_i, m, δ).*

The following useful lemma shows that, for any fixed $\delta > 0$, two desirable facts hold for RT *provided $|P_i| \le \ell_1(\delta)$ for each i (where $\ell = |T_i|$)*, if ℓ and the number of iterations m are chosen large enough: (a) the probability of any given message not succeeding at all can be made smaller than any given small positive constant, and (b) the probability of there remaining any given constant factor of the original number of messages can be made exponentially small in ℓ.

Lemma 2.7 *For any given positive ϵ, δ and η ($\eta \le 1/2$), there exist finite positive $m(\epsilon, \delta, \eta)$, $\ell(\epsilon, \delta, \eta)$ and $p(\epsilon, \delta, \eta)$ such that, for any $m \ge m(\epsilon, \delta, \eta)$, any $\ell \ge \ell(\epsilon, \delta, \eta)$, any $z \ge 1$, and $\ell_i = \ell_i(\delta)$ defined as in (1), the following hold if we perform RT$(\{P_i : i \in [z]\}, \{T_i : i \in [z]\}, m, z, \delta)$, provided $|P_i| \le \ell_1$ for each i.*

(i) For any message P, $\Pr[P$ *did not succeed*$] \leq \epsilon$.

(ii) $\Pr[$*in total at least* $\ell z \eta$ *messages were unsuccessful*$] \leq z e^{-\ell \cdot p(\epsilon,\delta,\eta)}$.

Proof: Let $P \in P_i$. Let $n_j(i)$ denote the number of unsuccessful elements of P_i before the performance of the j^{th} iteration of protocol E_i, in the notation of Definition 2.5. By assumption, we have $n_1(i) \leq \ell_1$; iterative application of Lemma 2.4 shows that

$$\Pr[n_{m+1}(i) \geq \ell_{m+1}] \leq \sum_{j \in [m]} F(\ell_j, \delta).$$

It is also easily seen, using Lemma 2.4, that the probability of P failing throughout is at most

$$(1 - 1/e)^m + \sum_{j=1}^{m-1} F(\ell_j, \delta).$$

These two failure probability bounds imply that if we pick

$$m(\epsilon, \delta, \eta) > \log(\epsilon/2)/\log(1 - 1/e)$$

and then choose $\ell(\epsilon, \delta, \eta)$ large enough, we can ensure part (i). Also, if we pick $m(\epsilon, \delta, \eta) \geq \log(\eta e(1+\delta))/\log(1 - 1/(e(1+\delta)))$ and then choose $\ell(\epsilon, \delta, \eta)$ large enough and $p(\epsilon, \delta, \eta)$ appropriately, we also obtain (ii). $\square$

A variant. The following small change in *RT* will arise in Lemmas 3.1 and 3.2. Following the notation of Definition 2.5, for each $i \in z$, there may be one known time $t_{i,g(i)} \in T_i$ which is "marked out": messages in P_i cannot attempt using the channel at time $t_{i,g(i)}$. To accommodate this, we modify *RT* slightly: define $j = j(i)$ to be the unique value such that $\ell_0 + \ell_1 + \cdots + \ell_{j-1} < g(i) \leq \ell_0 + \ell_1 + \cdots + \ell_j$. Then any message in P_i that collided in all of the first $(j-1)$ iterations, will, in the jth iteration, attempt using the channel at a time chosen randomly from $\{t_{i,p} : (p \neq g(i))$ **and** $\ell_0 + \cdots + \ell_{j-1} < p \leq \ell_0 + \cdots + \ell_j\}$. All other iterations are the same as before for messages in P_i, for each i.

We now sketch why Lemma 2.7 remains true for this variant, if we take $m(\epsilon, \delta, \eta)$ and $\ell(\epsilon, \delta, \eta)$ slightly larger and reduce $p(\epsilon, \delta, \eta)$ to a slightly smaller (but still positive) value. We start by stating the analogue of Lemma 2.4, which applies to the variant. (The proof that the analogue is correct is the same as the proof of Lemma 2.4.) Note that, for $s \geq 2$, $1 - (1 - 1/s)^s \leq 1 - 1/e + K_0/s$, for some absolute constant $K_0 > 0$.

Lemma 2.4$'$ *There are positive constants K_0, K_1, K_2 such that the following holds. For $s \geq 2$, suppose at most $s + 1$ balls are thrown uniformly and independently at random into s bins. Then (i) for any given ball B,* $\Pr[B$ *collides*$] = 1 - (1 - 1/s)^s \leq 1 - 1/e + K_0/s$, *and (ii) if C denotes the total number of balls that collide then, for any $\delta > 0$,*

$$\Pr[C \geq s(1 - 1/(e(1+\delta)))] \leq G(s, \delta), \text{ where } G(s, \delta) \doteq K_1 e^{-K_2 s \delta^2/(1+\delta)^2}.$$

Now note that the proof of Lemma 2.7 applies to the variant by using Lemma 2.4$'$ in place of Lemma 2.4.

2.2 The protocol

We present the ideas parameterized by several constants. Later we will choose values for the parameters to maximize the throughput. There will be a trade-off between the maximum throughput and the expected waiting time for a message; a different choice of parameters could take this into consideration. The constants we have chosen guarantee that our protocol is stable in the sense of [13] for $\lambda < 1/e$.

From now on, we assume that $\lambda < 1/e$ is given. Let $\Delta \geq 3$ be any (say, the smallest) positive integer such that

$$\lambda \leq (1 - 2/\Delta)/e. \tag{2}$$

We define δ_0 by

$$1 + \delta_0 = \frac{1}{e\lambda + 1/\Delta}. \tag{3}$$

Note that $\delta_0 > 0$ by our assumptions on λ and Δ.

Three important constants, b, r and k, shape the protocol; each of these is a positive integer that is at least 2. At any time during its lifetime in the protocol, a message is regarded as residing at some node of an infinite tree T, which is structured as follows. There are countably infinitely many leaves ordered left-to-right, with a *leftmost leaf*. Each non-leaf node of T has exactly k children, where

$$k > r\ . \tag{4}$$

As usual, we visualize all leaves as being at the same (lowest) *level*, their parents being at the next higher level, and so on. (The leaves are at level 0.) Note that the notions of left-to-right ordering and leftmost node are well-defined for every level of the tree. T is not actually constructed; it is just for exposition. We associate a finite nonempty set of non-negative integers $\mathit{Trial}(v)$ with each node v. Define $L(v) \doteq \min\{\mathit{Trial}(v)\}$, $R(v) \doteq \max\{\mathit{Trial}(v)\}$, and the *capacity* $\mathit{cap}(v)$ of v, to be $|\mathit{Trial}(v)|$. A required set of properties of the *Trial* sets is the following:

P1. If u and v is any pair of distinct nodes of T, then $\mathit{Trial}(u) \cap \mathit{Trial}(v) = \phi$;

P2. If u is either a proper descendant of v, or if u and v are at the same level with u to the left of v, then $R(u) < L(v)$.

P3. The capacity of all nodes at the same level is the same. Let u_i be a generic node at level i. Then, $\mathit{cap}(u_0) = b$ and $\mathit{cap}(u_i) = r \cdot \mathit{cap}(u_{i-1}) = br^i$, for $i \geq 1$.

Suppose we have such a construction of the *Trial* sets. (Note (P1): in particular, the *Trial* set of a node is *not* the union of the sets of its children.) Each message P injected into the system at some time step t_0 will initially enter the leaf node $u_0(P)$ where $u_0(P)$ is the leftmost leaf such that $L(u_0(P)) > t_0$. Then P will move up the tree if necessary, in the following way. In general, suppose P enters a node $u_i(P)$ at level i, at time t_i; we will be guaranteed the invariant "**Q:** $u_i(P)$ is an ancestor of $u_0(P)$, and $t_i < L(u_i(P))$." P will then run protocol $\mathrm{RT}(P_{u_i(P)}, \mathit{Trial}(u_i(P)), m, 1, \delta_0)$, where $P_{u_i(P)}$ is the set of messages entering $u_i(P)$ and m is a suitably large integer to be chosen later. If it is successful, P

will (of course) leave the system, otherwise it will enter the parent $u_{i+1}(P)$ of $u_i(P)$, at the last time slot (element of $\mathit{Trial}(u_i(P))$) at which it tried using the channel and failed, while running $\mathrm{RT}(P_{u_i(P)}, \mathit{Trial}(u_i(P)), m, 1, \delta_0)$. ($P$ knows what this time slot is: it is the m^{th} step at which it attempted using the channel, during this performance of RT.) Invariant $\mathbf{Q}$ is established by a straightforward induction on i, using Property P2. Note that the set of messages P_v entering any given node v perform protocol $\mathrm{RT}(P_v, \mathit{Trial}(v), m, 1, \delta_0)$, and, if v is any non-leaf node with children $u_1, u_2, \ldots, u_k$, then the trials at its k children correspond to $\mathrm{RT}(\{P_{u_1}, \ldots, P_{u_k}\}, \{\mathit{Trial}(u_1), \ldots, \mathit{Trial}(u_k)\}, m, k, \delta_0)$, by Property P1. Thus, each node receives all the unsuccessful messages from each of its k children; an unsuccessful message is imagined to enter the parent of a node u, immediately after it found itself unsuccessful at u.

The intuition behind the advantages offered by the tree is roughly as follows. Note that in a multiple-access channel problem, a solution is easy if the arrival rate is always close to the expectation (e.g., if we always get at most one message per step, then the problem is trivial). The problem is that, with probability 1, infinitely often there will be "bulk arrivals" (bursts of a large number of input messages within a short amount of time); this is a key problem that any protocol must confront. The tree helps in this by ensuring that such bursty arrivals are spread over a few leaves of the tree and are also handled *independently*, since the corresponding *Trial* sets are pairwise disjoint. One may expect that, even if several messages enter one child of a node v, most of the other children of v will be "well-behaved" in not getting too many input messages. These "good" children of v are likely to successfully transmit most of their input messages, thus ensuring that, with high probability, not too many messages enter v. Thus, bursty arrivals are likely to be smoothed out, once the corresponding messages enter a node at a suitable level in the tree. In short, our assumption on time-agreement plays a symmetry-breaking role.

Informally, if the proportion of the total time dedicated to nodes at level 0 is $1/s$, where $s > 1$, then the proportion for level i will be approximately $(r/k)^i/s$. Since the sum of these proportions for all i can be at most 1, we require $s \geq k/(k-r)$; we will take

$$s = k/(k-r)\ . \tag{5}$$

More precisely, the *Trial* sets are constructed as follows; it will be immediate that they satisfy Properties P1, P2, and P3. First define

$$s = \Delta/(\Delta-1),\ \ k = 4\Delta^2,\ \text{ and } r = 4\Delta. \tag{6}$$

We remark that though we have fixed these constants, we will use the symbols k, s and r (rather than their numerical values) wherever possible. Also, rather than present the values of our other constants right away, we choose them as we go along, to clarify the reasons for their choice.

For $i \geq 0$, let

$$F_i = \{j > 0 : \exists h \in [\Delta - 1] \text{ such that } j \equiv h\Delta^i \pmod{\Delta^{i+1}}\}. \tag{7}$$

Note that F_i is just the set of all j which, when written in base Δ, have zeroes in their i least significant digits, and have a non-zero in their $(i+1)^{\mathrm{st}}$ least significant digit. Hence, the sets F_i form a partition of $\mathbf{Z}^+$. Let v_i be a generic node at level i; if it is not the leftmost node in its level, let u_i denote the node at level i that is immediately to the left of v_i. We will ensure that all elements of $\mathit{Trial}(v_i)$ lie in F_i. (For any large enough interval I in $\mathbf{Z}^+$, the fraction of I lying in F_i is roughly $(\Delta - 1)/\Delta^{i+1} = (r/k)^i/s$; this was what we meant informally above, regarding the proportion of time assigned to level i of the tree being $(r/k)^i/s$.)

We now define $\mathit{Trial}(v_i)$ by induction on i and from left-to-right within the same level, as follows. If $i = 0$, then if v_0 is the leftmost leaf, we set $\mathit{Trial}(v_0)$ to be the smallest $\mathit{cap}(v_0)$ elements of F_0; else we set $\mathit{Trial}(v_0)$ to be the $\mathit{cap}(v_0)$ smallest elements of F_0 larger than $R(u_0)$. If $i \geq 1$, let w be the rightmost child of v_i. If v_i is the leftmost node at level i, we let $\mathit{Trial}(v_i)$ be the $\mathit{cap}(v_i)$ smallest elements of F_i that are larger than $R(w)$; else define $\mathit{Trial}(v_i)$ to be the $\mathit{cap}(v_i)$ smallest elements of F_i that are larger than $\max\{R(u_i), R(w)\}$. In fact, it is straightforward to show by the same inductive process that, if u_i is defined, then $R(w) > R(u_i)$; hence for every node v_i with $i \geq 1$,

$$L(v_i) \leq R(w) + s(k/r)^i; \quad R(v_i) \leq R(w) + s(k/r)^i \cdot br^i = R(w) + sbk^i. \tag{8}$$

2.3 Waiting times of messages

Our main random variable of interest is the time that a generic message P will spend in the system, from its arrival. Let

$$a = e(1 + \delta_0) \tag{9}$$

and d be a constant greater than 1.

Definition 2.8 *For any node $v \in T$, the random variable $\mathit{load}(v)$, the* load of v, *is defined to be* the number of messages that enter v; *for any positive integer t, node v at level i is defined to be t-*bad *if and only if $\mathit{load}(v) > br^i d^{t-1}/a$. Node v is said to be t-*loaded *if it is t-bad but not $(t+1)$-bad. It is called* bad *if it is 1-bad, and* good *otherwise.*

It is not hard to verify that, for any given $t \geq 1$, the probability of being t-bad is the same for any nodes at the same level in T. This brings us to the next definitions.

Definition 2.9 *For any (generic) node u_i at level i in T and any positive integer t, $p_i(t)$ denotes the probability that u_i is t-bad.*

Definition 2.10 *(i) The* failure probability *q is the maximum probability that a message entering a* good *node will* not *succeed during the functioning of that node. (ii) For any message P, let $u_0(P), u_1(P), u_2(P), \ldots$ be the nodes of T that u_i is allowed to pass through, where the level of $u_i(P)$ is i. Let $E_i(P)$ be the event that P enters $u_i(P)$.*

If a node u at level i is good then, in the notation of Lemma 2.7, its load is at most $\ell_1(\delta_0)$, where $\ell = \mathit{cap}(u)$; hence, Lemma 2.7(i) shows that, for any fixed $q_0 > 0$, $q < q_0$ can be achieved by making b and the number of iterations m large enough.

Note that *the distribution of $E_i(P)$ is independent of its argument.* Hence, for any $i \geq 0$, we may define $f_i \doteq \Pr[E_i(P)]$ for a generic message P. Suppose P was unsuccessful at nodes $u_0(P), u_1(P), \ldots, u_i(P)$. Let $A(i)$ denote the maximum total amount of time P could have spent in these $(i+1)$ nodes. Then, it is not hard to see that $A(0) \leq s\ cap(u_0) + s\ cap(u_0) = 2sb$ and that, for $i \geq 1$, $A(i) \leq kA(i-1) + (k/r)^i sbr^i$, using (8). Hence,

$$A(i) \leq (i+2)sbk^i \text{ for all } i. \tag{10}$$

The simple, but crucial, Lemma 2.11 is about the distribution of an important random variable $W(P)$, the time that P spends in the system.

Lemma 2.11 *(i) For any message P,* $\Pr[W(P) > A(i)] \leq f_{i+1}$ *for all $i \geq 0$, and* $\mathrm{E}[W(P)] \leq \sum_{j=0}^{\infty} A(j) f_j$. *(ii) For all $i \geq 1$, $f_i \leq q f_{i-1} + p_{i-1}(1)$.*

Proof: Part (i) is immediate, using the fact that, for a non-negative integer-valued random variable Z, $\mathrm{E}[Z] = \sum_{i=1}^{\infty} \Pr[Z \geq i]$. For part (ii), note that

$$f_i = f_{i-1} \Pr[E_i \mid E_{i-1}]. \tag{11}$$

Letting $c_i = \Pr[u_{i-1}(P) \text{ was good} \mid E_{i-1}]$,

$$\begin{aligned}
\Pr[E_i \mid E_{i-1}] &= c_i \Pr[E_i \mid u_{i-1}(P) \text{ was good} \wedge E_{i-1}] + \\
&\qquad + (1-c_i) \Pr[E_i \mid u_{i-1}(P) \text{ was bad} \wedge E_{i-1}] \\
&\leq \Pr[E_i \mid u_{i-1}(P) \text{ was good} \wedge E_{i-1}] + \\
&\qquad + \Pr[u_{i-1}(P) \text{ was bad} \mid E_{i-1}] \\
&\leq q + \Pr[u_{i-1}(P) \text{ was bad} \mid E_{i-1}] \\
&\leq q + \Pr[u_{i-1}(P) \text{ was bad}] / \Pr[E_{i-1}].
\end{aligned}$$

Thus, by (11), $f_i \leq f_{i-1} q + \Pr[u_{i-1}(P) \text{ was bad}] = q f_{i-1} + p_{i-1}(1)$. □

2.4 The improbability of high nodes being heavily loaded

As is apparent from Lemma 2.11, our main interest is in getting a good upper bound on $p_i(1)$. However, to do this we will also need some information about $p_i(t)$ for $t \geq 2$, and hence Definition 2.9. The basic intuition is that if a node is good then, with high probability, it will successfully schedule "most" of its messages; this is formalized by Lemma 2.7(ii). In fact, Lemma 2.7(ii) shows that, for any node u in the tree, the *good* children of u will, with high probability, pass on a total of "not many" messages to u, since the functioning of each of these children is independent of the other children.

To estimate $p_i(t)$, we first handle the easy case of $i = 0$. Recall that if X_1 and X_2 are independent Poisson random variables with means λ_1 and λ_2 respectively, then $X_1 + X_2$ is Poisson with mean $\lambda_1 + \lambda_2$. Thus, u_0 being t-bad is a simple large-deviation event for a Poisson random variable with mean $sb\lambda$. If, for every $t \geq 1$, we define $\nu_t \doteq d^{t-1}/(sa\lambda)$ and ensure that $\nu_t > 1$ by guaranteeing

$$sa\lambda < 1, \tag{12}$$

then Theorem 2.1 shows that

$$p_0(t) = \Pr[u_0 \text{ is } t\text{-bad}] \leq H(sb\lambda, \nu_t)\ . \tag{13}$$

Our choices for s and a validate (12): see (3), (9), (6) and (2).

We now consider how a generic node u_i at level $i \geq 1$ could have become t-bad, for any given t. The resulting recurrence yields a proof of an upper bound for $p_i(t)$ by induction on i. The two cases $t \geq 2$ and $t = 1$ are covered by Lemmas 2.12 and 2.13 respectively. We require

$$d^2 + k - 1 \leq dr \; ; \tag{14}$$

this is satisfied by defining

$$d = 2\Delta.$$

Lemma 2.12 *For $i \geq 1$ and $t \geq 2$, if a node u_i at level i in T is t-bad, then at least one of the following two conditions holds for u_i's set of children: (i) at least one child is $(t+1)$-bad, or (ii) at least two children are $(t-1)$-bad. Thus,*

$$p_i(t) \leq k p_{i-1}(t+1) + \binom{k}{2} \left(p_{i-1}(t-1)\right)^2 .$$

Proof: Suppose that u_i is t-bad but that neither (i) nor (ii) holds. Then u_i has at most one child v that is either t-loaded or $(t-1)$-loaded, and none of the other children of u_i is $(t-1)$-bad. Node v can contribute a load of at most $br^{i-1}d^t/a$ messages to u_i; the other children contribute a total load of at most $(k-1)br^{i-1}d^{t-2}/a$. Thus the children of u_i contribute a total load of at most $br^{i-1}d^{t-2}(d^2+k-1)/a$, which contradicts the fact that u_i is t-bad, since (14) holds. □

In the case $t = 1$, a key role is played by the intuition that the good children of u_i can be expected to transmit much of their load successfully. We now fix q and m, and place a *lower bound* on our choice of b. Note that (14) implies $r > d$. Define $\eta_1, \eta_2 > 0$ by

$$\eta_1 = \min\{\frac{r-d}{a(k-1)}, \frac{1}{2}\} \text{ and } \eta_2 = \min\{\frac{r}{ak}, \frac{1}{2}\} \; .$$

For q, we treat it as a parameter that satisfies

$$0 < q < 1/k. \tag{15}$$

(Lemmas 3.1 and 3.2 will require that q is sufficiently small.) In the notation of Lemma 2.7, we define

$$m = \max\{m(q, \delta_0, \eta_1), m(q, \delta_0, \eta_2)\} \tag{16}$$

and require

$$b \geq \max\{\ell(q, \delta_0, \eta_1), \ell(q, \delta_0, \eta_2)\}. \tag{17}$$

Lemma 2.13 *For any $i \geq 1$,*

$$p_i(1) \leq k p_{i-1}(2) + \binom{k}{2} \left(p_{i-1}(1)\right)^2 + k(k-1)p_{i-1}(1)e^{-br^{i-1}p(q,\delta_0,\eta_1)} + ke^{-br^{i-1}p(q,\delta_0,\eta_2)}.$$

Proof: Suppose that u_i is 1-bad. There are two possibilities: that at least one child of u_i is 2-bad or that at least two children are 1-bad. If neither of these conditions holds, then either (A) u_i has exactly one child which is 1-loaded with no other child being bad, or (B) all children are good.

In case (A), the $k-1$ good children must contribute a total of at least

$$\frac{cap(u_i)}{a} - \frac{cap(u_{i-1})d}{a} = \frac{br^{i-1}(r-d)}{a} \geq br^{i-1}(k-1)\eta_1$$

messages to u_i. In the notation of Lemma 2.7, $z = k-1$, $\ell = br^{i-1}$ and $\eta = \eta_1$. Since there are k choices for the 1-loaded child, Lemma 2.7(ii) shows that the probability of occurrence of case (A) is at most

$$k(k-1)p_{i-1}(1)e^{-br^{i-1}p(q,\delta_0,\eta_1)}.$$

In case (B), the k good children contribute at least $cap(u_i)/a = br^i/a$. By a similar argument, the probability of occurrence of case (B) is at most

$$ke^{-br^{i-1}p(q,\delta_0,\eta_2)}.$$

The inequality in the lemma follows. □

Next is a key theorem that proves an upper bound for $p_i(t)$, by induction on i. We assume that our constants satisfy the conditions (4, 5, 9, 12, 14, 15, 16, 17).

Theorem 2.14 *For any fixed $\lambda < 1/e$ and any $q \in (0, 1/k)$, there is a sufficiently large value of b such that the following holds. There are positive constants α, β and γ, with $\alpha, \beta > 1$, such that*

$$\forall i \geq 0 \ \forall t \geq 1, \ p_i(t) \leq e^{-\gamma \alpha^i \beta^{t-1}}.$$

Before proving Theorem 2.14, let us see why this shows the required property that E$[W(P)]$, the expected waiting time of a generic message P, is finite. Theorem 2.14 shows that, for large i, $p_{i-1}(1)$ is negligible compared to q^i and hence, by Lemma 2.11(ii), $f_i = \mathrm{O}(q^i)$. Hence, Lemma 2.11(i) combined with the bound (10) shows that, for any choice $q < 1/k$, E$[W(P)]$ is finite (and good upper tail bounds can be proven for the distribution of $W(P)$). Thus (15) guarantees the finiteness of E$[W(P)]$.

Proof: (of Theorem 2.14) This is by induction on i. If $i = 0$, we use inequality (13) and require that

$$H(sb\lambda, \nu_t) \leq e^{-\gamma\beta^{t-1}}. \tag{18}$$

From (12), we see that $\nu_t > 1$; thus by Fact 2.2, there is some $M = M_{\nu_t}$ such that $H(sb\lambda, \nu_t) \leq e^{-\nu_t sb\lambda/M}$. Therefore to satisfy inequality (18), it suffices to ensure that $d^{t-1}b/(aM) \geq \gamma\beta^{t-1}$. We will do this by choosing our constants so as to satisfy

$$d \geq \beta \text{ and } b \geq \gamma a M\ . \tag{19}$$

We will choose α and β to be fairly close to (but larger than) 1, and so the first inequality will be satisfied. Although γ will have to be quite large, we are free to choose b sufficiently large to satisfy the second inequality.

We proceed to the induction for $i \geq 1$. We first handle the case $t \geq 2$, and then the case $t = 1$.
Case I: $t \geq 2$. By Lemma 2.12, it suffices to show that

$$ke^{-\gamma\alpha^{i-1}\beta^t} + \binom{k}{2} e^{-2\gamma\alpha^{i-1}\beta^{t-2}} \leq e^{-\gamma\alpha^i\beta^{t-1}}.$$

It is straightforward to verify that this holds for some sufficiently large γ, provided

$$\beta > \alpha \text{ and } 2 > \alpha\beta \ . \tag{20}$$

We can pick $\alpha = 1+\epsilon$ and $\beta = 1+2\epsilon$ for some small positive ϵ, $\epsilon < 1$, to satisfy (20).
Case II: $t = 1$. The first term in the inequality for $p_i(1)$ given by Lemma 2.13 is the same as for Case I with $t = 1$; thus, as above, an appropriate choice of constants will make it much smaller than $e^{-\gamma\alpha^i}$. Similarly, the second term in the inequality for $p_i(1)$ can be handled by assuming that $\alpha < 2$ and that γ is large enough. The final two terms given by Lemma 2.13 sum to

$$k(k-1)p_{i-1}(1)e^{-br^{i-1}p(q,\delta_0,\eta_1)} + ke^{-br^{i-1}p(q,\delta_0,\eta_2)}. \tag{21}$$

We wish to make each summand in (21) at most, say, $e^{-\gamma\alpha^i}/4$. We just need to ensure that

$$br^{i-1}p(q,\delta_0,\eta_1) \geq \gamma\alpha^i + \ln(4k^2) \text{ and } br^{i-1}p(q,\delta_0,\eta_2) \geq \gamma\alpha^i + \ln(4k) \ . \tag{22}$$

Since $r > \alpha$, both of these are true for sufficiently large i. To satisfy these inequalities for small i, we choose b sufficiently large to satisfy (17,19,22), completing the proof of Theorem 2.14. □

It is now easily verified that conditions (4,5,12,14,19,20) are all satisfied. Thus, we have presented stable protocols for $\lambda < 1/e$.

Theorem 2.15 *Fix any $\lambda < 1/e$. In the Synchronized Infinitely-Many Users Model, our protocol guarantees an expected waiting time of* $\mathrm{O}(1)$ *for every message.*

We also get a tail bound as a corollary of Theorem 2.14:

Corollary 2.16 *Let ℓ' be a sufficiently large constant. Fix any $\lambda < 1/e$ and $c_1 > 1$. We can then design our protocol such that, for any message P, in addition to having* $\mathrm{E}[W(P)] = \mathrm{O}(1)$*, we also have for all $\ell \geq \ell'$ that* $\Pr[W(P) \geq \ell] \leq \ell^{-c_1}$.

Proof: Using (10) we see that if $W(P) \geq \ell$ then P enters j levels where $\sum_{i=1}^{j}(i+2)k^i > \ell/(2sb)$, so $j(j+2)k^j \geq \ell/(2sb)$. This implies that

$$j \geq (\log_k(\frac{\ell}{2sb}) - 2\log_k\log_k(\frac{\ell}{2sb})).$$

As we mentioned in the paragraph preceding the proof of Theorem 2.14, $f_j = \mathrm{O}(q^j)$. Thus,

$$Pr[W(P) \geq \ell] = \mathrm{O}(q^{\log_k(\ell/(2sb)) - 2\log_k\log_k(\ell/(2sb))}).$$

The result follows by designing the protocol with $q \leq k^{-c_2c_1}$ for a sufficiently large positive constant c_2. □

Remark. In practice, the goal is often simply to ensure that the probability that any given packet is delivered to the channel is at least $1-\epsilon$ for some constant ϵ. By the corollary, we can achieve this goal by truncating each packet after $(1/\epsilon)^{1/c_1}$ steps, or equivalently by truncating the infinite tree after $\mathrm{O}(\log_k(1/\epsilon))$ levels.

3 The Synchronized Finitely-Many Users Protocol

We transfer to the Synchronized Finitely-Many Users Model (see Section 1.1). Here, we shall let $\lambda = \sum_i \lambda_i$ be any constant smaller than $1/e$, and show how to simulate the Infinitely-Many Users Protocol on n synchronized users. Suppose for the moment that each message can do its own processing independently (this assumption will be removed shortly). With this assumption, the difference between the synchronized infinitely-many users model which we have been considering and the synchronized finitely-many users model is that, instead of being a Poisson distribution with parameter λ, the input arrival distribution can be any $\{\lambda_i\}_{1\leq i\leq n}$-dominated distribution (see Section 1.1). Although the arrivals may not be independent, the strong condition in the definition of "$\{\lambda_i\}_{1\leq i\leq n}$-dominated" allows us to apply Theorem 2.1 (a) to the message arrivals (using stochastic domination). Therefore, (13) still holds in the synchronized finitely-many users model.

We need to avoid the assumption that each message is processed separately. The difficulty is that each user must be responsible for a potentially unbounded number of messages and must manage them in constant time at each step. We first sketch how to manage the messages and then give further details. Each user s maintains, for each $i \geq 0$, a linked list $L(s,i)$ of the messages belonging to it that are at level i of the tree. If it is the turn of messages at level i of the tree to try in the current time step t, then each user s will compute the probability $p_{s,t}$ of *exactly one* message in $L(s,i)$ attempting to use the channel in our Synchronized Infinitely-Many Users Protocol. Then, each s will independently send the message at the head of $L(s,i)$ to the channel with probability $p_{s,t}$. (The reader may have noticed that in order to simulate faithfully our infinitely-many users protocol, s should also calculate the probability $r_{s,t}$ that more than one message in $L(s,i)$ attempts to use the channel. It should send a dummy message to the channel with probability $r_{s,t}$. This solution works, but we will show at the end of this section that dummy messages are not necessary.)

We now present the details of this message-management scheme. Let all the parameters such as k, Δ, etc., be as defined in Section 2. For each $t \in \mathbf{Z}^+$, define *active*(t) to be the index of the least significant digit of t that is nonzero, if t is written in base Δ. Recall from (7) that if the current time is t then the messages in $L(s_j, \text{active}(t))$, taken over all users s_j, are precisely those that may attempt using the channel at the current step. Thus, if $\text{active}(t) = i$, each user s first needs access to the head-pointer of $L(s,i)$ in $\mathrm{O}(1)$ time. For this, it suffices if s counts time in base Δ and has an infinite array whose ith element is the head-pointer of $L(s,i)$. However, such static infinite storage is not required: s can count time in base Δ using a linked list, where the ith element of the list additionally contains the head-pointer of $L(s,i)$. This list can be augmented with pointers to jump over substrings (of the base-Δ representation of t) that are composed of only $\Delta - 1$, so that s can maintain t and $\text{active}(t)$ in $\mathrm{O}(1)$ time. We leave the tedious but straightforward details of this to the reader. (Alternatively, as mentioned in the remark following Corollary 2.16, we may simply truncate the tree to a certain finite height, if we only desire that each message reaches the channel with sufficiently high probability. Then, of course, s may simply have a *finite* array that contains head-pointers to the $L(s,i)$.) Thus, we assume that s can access the head-pointer to $L(s, \text{active}(t))$ in $\mathrm{O}(1)$ time.

Each list $L(s,i)$ will also have the value of $|L(s,i)|$ at its head. In addition, $L(s,i)$ will have a pointer to its last element, so that concatenating two such lists can be done in $\mathrm{O}(1)$ time. Each $L(s,i)$ will also have the important property that the rank of any message P in the list order is uniformly distributed. For each s, we maintain these properties by induction on i. To establish these properties for the base case $i = 0$, we shall require the following assumption on the message arrivals: in each step t, the messages arriving at user s arrive in *random* order (among each other) and, when arriving, they increment $|L(s,0)|$ and get appended to the head of $L(s,0)$. Once these properties are true for level i, they are easily maintained for $i+1$: since $L(s,i+1)$ is the disjoint union of at most $r = \mathrm{O}(1)$ such lists from level i (one from each child of a node). We just need to generate a random permutation of $[r]$ and concatenate these lists in the permuted order.

We need to show the probability computations to be done by s. Recall that the set of messages P_v entering any given node v perform protocol $\mathrm{RT}(P_v, \mathit{Trial}(v), m, 1, \delta_0)$. Suppose s is managing its messages at node v in level i of the tree at time step t. Let $\mathit{Trial}(v) = \{t_1 < t_2 < \cdots < t_\ell\}$. Recall from Definition 2.5 that the messages in P_v proceed in m iterations. Suppose s is conducting the jth iteration at time t; thus,

$$t \in S \doteq \{t_p : \ \ell_0 + \ell_1 + \cdots + \ell_{j-1} < p \le \ell_0 + \ell_1 + \cdots + \ell_j\}.$$

User s needs to compute the probability $p_{s,t}$ of *exactly one* message in $L(s,i)$ attempting to use the channel. We show how to do this, for each t_p such that $(\sum_{h=0}^{j-1} \ell_h) < p \le (\sum_{h=0}^{j} \ell_h)$. Recall that s knows the value of $N \doteq |L(s,i)| = |P_v|$: this is present at the head of $L(s,i)$. At time step t_q where $q = 1 + \lfloor(\sum_{h=0}^{j-1} \ell_h)\rfloor$, s generates a random integer $r_1 \in \{0\} \cup [N]$, where

$$\Pr[r_1 = j] = \binom{N}{j} \left(\frac{1}{|S|}\right)^j \left(1 - \frac{1}{|S|}\right)^{N-j}.$$

Note that r_1 has the same distribution as the number of messages in $L(s,i)$ that would have attempted using the channel at step t_q in our Synchronized Infinitely-Many Users Protocol. At time step t_q, if $r_1 = 1$, s will send the message at the head of $L(s,i)$ to the channel. Similarly, if $t = t_{q+1}$, s will generate a random integer $r_2 \in \{0\} \cup [N - r_1]$ such that

$$\Pr[r_2 = j] = \binom{N - r_1}{j} \left(\frac{1}{|S| - 1}\right)^j \left(1 - \frac{1}{|S| - 1}\right)^{N - r_1 - j}.$$

Once again, r_2 has the same distribution as the number of messages in $L(s,i)$ that would have attempted using the channel at step t_{q+1}; as before, s will send the message at the head of $L(s,i)$ to the channel at time step t_{q+1} if and only if $r_2 = 1$. It is immediate that, at each step, s correctly computes the probability of a "unique send".

At this point, it is clear that the infinitely-many users protocol can be simulated by finitely-many users provided that the users send "dummy messages" as explained previously. We now argue that sending dummy messages is unnecessary because the protocol is "deletion resilient" in the sense that if an adversary deletes a message (for example, one that would have collided with a dummy), the expected lifetime of other messages can only shorten. Formally, we must show that the simulated system without dummy messages evolves with no worse probabilities than in the infinite case. We observe from our proof (for the Synchronized Infinitely-Many Users Model) that it suffices to show the following analogue of Lemma 2.4. We need to show that if the number of available time slots (elements of the set S) is at least as high as $\sum_j |L(s_j, i)|$ (the sum taken over all users s_j), then: (a) for any s and any message $P \in L(s, i)$, the probability that P succeeds in the $|S|$ time slots above is greater than $1/e$, and (b) the total number of colliding messages C satisfies the tail bound in part (ii) of Lemma 2.4.

It is not hard to see that the probability of a collision in any one of the time steps above is at most $1/e$. Thus (b) follows by the same proof as for part (ii) of Lemma 2.4. So, let us show (a) now. Let $|L(s, i)| = N$, and let $M \in [N, |S|]$ denote $\sum_j |L(s_j, i)|$. In any given step among the $|S|$ steps, the probability that s *successfully* transmitted a message, is at least

$$\frac{N}{|S|}\left(1-\frac{1}{|S|}\right)^{M-1} \geq \frac{N}{|S|}\left(1-\frac{1}{|S|}\right)^{|S|-1} > \frac{N}{e|S|}.$$

Thus, by linearity of expectation, the expected number of successful transmissions by s is more than N/e. Once again by linearity, this equals the sum of the success probabilities of the messages in $L(s, i)$, each of which is the same by symmetry. Thus, for any given message $P \in L(s, i)$, P succeeds with probability more than $1/e$.

This completes the proof for the Synchronized Finitely-Many Users Model.

3.1 A Variant

We will take n to be sufficiently large (if n is smaller than a certain constant, we can use the protocol of [13], which can handle any arrival rate $\lambda < 1$). We will assume without loss of generality that n is even; if n is odd, just add a dummy user which gets no messages and does nothing.

Let $\mathcal{P}$ be a protocol running on n completely synchronized users which simulates the Synchronized Infinitely-Many Users Protocol from Section 2 for $n^2 - 1$ steps then skips a step and continues; this "skip" happens at every step of the form $jn^2 - 1$, where $j \in \mathbf{Z}^+$. Inputs might, however, arrive during the skipped step. To simplify $\mathcal{P}$, note from (2) that we can take Δ to be even. Now (7) shows that, for all $i \geq 1$, all elements of F_i will be even; thus, since all skipped steps (which are of the form $jn^2 - 1$) are odd since n is even, we see that no skipped step occurs in the *Trial* set of nodes at level $i \geq 1$. Thus, the skipped steps occur only during the time slots assigned to the nodes at the leaf level. Since the *Trial* sets of the leaves have cardinality b and as we may take $n > \sqrt{b}$, we have that such "marked out" (skipped) steps occur at most once in the *Trial* set of any leaf. Thus, as long as b is sufficiently large (and n is chosen larger), the "variant" discussed after Lemma 2.7 shows that $\mathcal{P}$ is essentially the same as the Synchronized Infinitely-Many Users Protocol as far as our analysis is concerned.

We prove the following two useful lemmas about $\mathcal{P}$. In both lemmas, $\mathcal{P}$ is run for at most n^{40} steps.

Lemma 3.1 *Suppose $\lambda < 1/e$ and that $\mathcal{P}$ is run with a $\{\lambda_i\}_{1 \le i \le n}$-dominated arrival distribution for $\tau \le n^{40}$ steps. Then the expected delay of any message that arrives is* $\mathrm{O}(1)$*. Furthermore, the probability that any message has delay more than $n^7/2$ is at most n^{-60}.*

Proof: As discussed above, we can handle $\mathcal{P}$ just as if it were the Synchronized Infinitely-Many Users Protocol. Thus, by choosing $c_1 \ge 18$ in the notation of Corollary 2.16 and as long as n is sufficiently large, Corollary 2.16 shows that the lemma holds. $\Box$

Lemma 3.2 *Suppose $\lambda < 1/e$ and that $\mathcal{P}$ is run with a $\{\lambda_i\}_{1 \le i \le n}$-dominated arrival distribution for $\tau \le n^{40}$ steps. Suppose further that a message arrives at user p at step $t' \le \tau$. Then the expected delay of any message that arrives is* $\mathrm{O}(1)$*. Furthermore, the probability that any message has delay more than $n^7/2$ is at most n^{-60}.*

Proof: The only place where the proof in Section 2 uses the arrival distribution is in bound (13). We argued at the beginning of this section that (13) still holds for any $\{\lambda_i\}_{1 \le i \le n}$-dominated arrival distribution. We now show that a similar bound holds even if the arrival distribution is conditioned on a message arriving at user p at step $t' \le \tau$. Recall that a leaf u_0 is t-bad if and only if its load (the number of arrivals in the relevant period of sb steps) exceeds bd^{t-1}/a. The number of arrivals in sb steps is at most 1 plus the sum of nsb random variables $X_{i,j}$ where, for $1 \le i \le n$ and $1 \le j \le sb$, $X_{i,j}$ is a random variable that has value 1 with probability at most λ_i (even conditioned on other arrivals) and value 0 otherwise. Using stochastic domination, we can apply Theorem 2.1. We let $\nu'_t = (bd^{t-1} - a)/(asb\lambda)$. Since $sa\lambda < 1$ (12), b can be chosen sufficiently large to make $\nu'_t > 1$. By Theorem 2.1, the probability that the sum of the random variables exceeds $((db^{t-1})/a) - 1 = (sb\lambda)\nu'_t$ is at most $H(sb\lambda, \nu'_t)$. Thus, in place of (13), we now have "$\Pr[u_0 \text{ is } t\text{-bad}] \le H(sb\lambda, \nu'_t)$". A small further change to be made to our proof for the Synchronized Infinitely-Many Users Protocol is, in the sentence following (18), to define $M = M_{\nu'_t}$. The whole proof goes through now. $\Box$

4 The Unsynchronized Finitely-Many Users Protocol

4.1 Notation and Preliminaries

In our basic model, we have n users which can start and stop at arbitrary steps, with the constraint that each time a user starts, it runs for at least a certain polynomial number of steps. (For the constant expected message delay results in Section 4.4, we require this polynomial to be $8n^{71}$; however, n^{33} is sufficient for all proofs in Section 4.3. No attempt has been made to optimize these polynomials.) Also recall that n is taken to be sufficiently large and that $\lambda = \sum_i \lambda_i < 1/e$.

4.2 The Protocol

The users typically simulate protocol $\mathcal{P}$ from Section 3. However, the starting and stopping of users causes the system to become unsynchronized, so the protocol synchronizes itself from time to time.

Here is an informal description of our protocol. In the normal state a user maintains a buffer B of size n^7 and an unbounded queue Q, each containing messages to be sent. When a message is generated it is put into B. For each message $m \in B$ the user maintains a variable $\text{trial}(m)$ which contains the next step on which the user will attempt to send m. The step $\text{trial}(m)$ will be chosen using protocol $\mathcal{P}$. When $\mathcal{P}$ is "skipping a step" our protocol will take the opportunity to try to send some messages from Q: at such steps, with probability $1/(3n)$, the user attempts to send the first message in Q. Each user also maintains a list L which keeps track of the results (either "failure" or "success") of the (up to n^2) most recent message sending attempts from Q.

A user goes into a synchronizing state if any message has remained in the buffer for n^7 steps or if L is full (contains n^2 results) and only contains failures. It also goes into a synchronizing state from time to time even when these events do not occur. (It synchronizes if it has been simulating $\mathcal{P}$ for at least n^{40} steps, and it synchronizes with probability n^{-30} on any given step.) If the user does go into a synchronizing state, it transfers all messages from B to the end of Q.

In the synchronizing state, a user could be in one of many possible stages, and its actions depend on the stage that it is in. It will always put any generated messages into the queue. Also, it sends only dummy messages in the synchronizing state. (The dummy messages are used for synchronizing. Real messages that arrive during the synchronization phase must wait until the next normal phase to be sent.[1]) The sequence of synchronization stages which a user goes through is as follows.

Definition: Let $W = 12n^4$.

JAMMING The user starting the synchronization jams the channel by sending messages at every step. In this way, it signals other users to start synchronizing also.

FINDING_LEADER Each user sends to the channel with probability $1/n$ on each step. The first user to succeed is the leader.

ESTABLISHING_LEADER In this stage, a user has decided it is the leader, and it jams the channel so no other user will decide to be the leader.

SETTING_CLOCK In this stage, a user has established itself as the leader, and it jams the channel once every $4W$ steps, giving other users a chance to synchronize with it.

COPYING_CLOCK In this stage, a user has decided it is not the leader, and it attempts to copy the leader's clock by polling the channel repeatedly to find the synchronization signal (namely, the jamming of the channel every $4W$ steps by the leader). Specifically, it sends to the channel with probability $1/(3n)$ on each step and, if it succeeds, it knows that the current step (mod $4W$) does not correspond to the leader's clock. After many attempts, it should be left with only one step (mod $4W$) that could correspond to the leader's clock. At the end of this stage, it synchronizes its clock to the leader's clock.

[1] Of course, there is no harm in using real messages for synchronizing, but this does not improve the *provable* results, so we prefer to use dummy messages for synchronizing in order to keep the exposition clear.

WAITING This stage is used by a user after COPYING_CLOCK in order to synchronize with the leader's clock. The user idles during this stage.

POLLING A user in this stage is simply "biding its time" until it switches to a normal stage. While doing so, it attempts to send to the channel occasionally (with probability $1/(3n)$ on each step) in order to detect new users which might be joining the system and re-starting a synchronization phase. If new users are detected, the user re-starts the synchronization phase. Otherwise, it begins the normal phase of the protocol.

The length of each of these stages is very important in terms of achieving both a high probability of synchronization and a high level of robustness. The high probability of synchronization is achieved by making the "preliminary" stages (i.e., JAMMING, FINDING_LEADER, and ESTABLISHING_LEADER) of length $\Theta(W)$ (this is long enough to guarantee all users in a normal state will detect a synchronization), and the "synchronizing" stages (i.e., SETTING_CLOCK, COPYING_CLOCK, and WAITING) of length $\Theta(Wn^2)$ (this gives users enough time to determine the leader's clock modulo $4W$ with high probability). The high level of robustness is achieved by the following properties:

1. the lengths of the "preliminary" and "synchronizing" stages are as above,
2. only the preliminary stages can cause the channel to be jammed,
3. the "synchronizing" stages cannot detect a new synchronization occurring,
4. the POLLING stage is of length $\Theta(Wn^3)$ (longer than all of the other stages combined), and
5. the POLLING stage is able to detect new synchronizations.

The differing lengths of time for the "preliminary", "synchronizing" and POLLING stages, and the fact that only the POLLING stage could cause another synchronization to occur, guarantee that bad events as described at the end of Section 1.3 cannot occur, even when up to n users are starting at different times (and stopping periodically).

Whenever a user joins the multiple-access channel, it starts the protocol with state = SYNCHRONIZING, sync_stage = JAMMING, clock = 0, and L empty. We now give the details of the protocol.

Protocol

```
    At each step do
        If (state = NORMAL) call Procedure Normal
        Else call Procedure Synchronizing

Procedure Normal
    If a message $m$ is generated
        Put $m$ in $B$
        Choose trial($m$) by continuing the simulation of $\mathcal{P}$
    If ((clock mod $n^2$) = $n^2 - 1$) call Procedure Queue_Step
    Else call Procedure Normal_Step
```

```
Procedure Begin_Sync
    Move all of the messages in $B$ to $Q$
    Empty $L$
    state ← SYNCHRONIZING, sync_stage ← JAMMING, clock ← 0
```

```
Procedure Normal_Step
    If (clock $\geq n^{40}$ or any message in $B$ has waited more than $n^7$ steps)
        Call Procedure Begin_Sync
    Else With Probability $n^{-30}$, call Procedure Begin_Sync
        Otherwise
            If more than one message $m$ in $B$ has trial($m$) = clock
                For each $m \in B$ with trial($m$) = clock
                    Choose a new trial($m$) by continuing the simulation of $\mathcal{P}$
            If exactly one message $m$ in $B$ has trial($m$) = clock
                Send $m$
                If $m$ succeeds, remove it from $B$
                Else choose a new trial($m$) by continuing the simulation of $\mathcal{P}$
            clock ← clock + 1
```

```
Procedure Queue_Step
    With probability $1/(3n)$
        If ($Q$ is empty) send a dummy message
        Else
            Send the first message in $Q$
            If the outcome is "success", remove the message from $Q$
        Add the outcome of the send to $L$
    Otherwise add "failure" to $L$
    If ($|L| = n^2$ and all of the entries of $L$ are "failure")
        Call Procedure Begin_Sync
    Else clock ← clock + 1
```

```
Procedure Synchronizing
    If a message arrives, put it in $Q$
    If (sync_stage = JAMMING) call Procedure Jam
    Else If (sync_stage = FINDING_LEADER) call Procedure Find_Leader
    Else If (sync_stage = ESTABLISHING_LEADER) call Procedure Establish_Leader
    Else If (sync_stage = SETTING_CLOCK) call Procedure Set_Clock
    Else If (sync_stage = COPYING_CLOCK) call Procedure Copy_Clock
    Else If (sync_stage = WAITING) call Procedure Wait
    Else If (sync_stage = POLLING) call Procedure Poll
```

```
Procedure Jam
    Send a dummy message
    If (clock $< W/2 - 1$), clock $\leftarrow$ clock $+ 1$
    Else sync_stage $\leftarrow$ FINDING_LEADER, clock $\leftarrow 0$

Procedure Find_Leader
    With probability $1/n$
        Send a dummy message
        If it succeeds
            sync_stage $\leftarrow$ ESTABLISHING_LEADER, clock $\leftarrow 0$
    If (clock $< W - 1$) clock $\leftarrow$ clock $+ 1$
    Else
        for $i = 0$ to $4W - 1$
            possibletime[$i$] $\leftarrow$ Yes
        sync_stage $\leftarrow$ COPYING_CLOCK, clock $\leftarrow 0$

Procedure Establish_Leader
    Send a dummy message
    If (clock $< 2W - 1$) clock $\leftarrow$ clock $+ 1$
    Else sync_stage $\leftarrow$ SETTING_CLOCK, clock $\leftarrow 0$

Procedure Set_Clock
    If (clock $= 0 \bmod 4W$)
        Send a dummy message
    If (clock $< 20Wn^2 - 1$) clock $\leftarrow$ clock $+ 1$
    Else sync_stage $\leftarrow$ POLLING, clock $\leftarrow 0$

Procedure Copy_Clock
    With probability $1/(3n)$
        Send a dummy message
        If it succeeds
            possibletime[clock mod $4W$] $\leftarrow$ No
    If (clock $< 20Wn^2 - 1$) clock $\leftarrow$ clock $+ 1$
    Else
        If possibletime[$j$] $=$ Yes for exactly one $j$,
            clock $\leftarrow -j$
            If ($j = 0$) sync_stage $\leftarrow$ POLLING
            Else sync_stage $\leftarrow$ WAITING
        Else sync_stage $\leftarrow$ POLLING, clock $\leftarrow 0$

Procedure Wait
    clock $\leftarrow$ clock $+ 1$
    If (clock $= 0$), sync_stage $\leftarrow$ POLLING
```

```
Procedure Poll
    With Probability $1/(3n)$
        Send a dummy message
        Add the outcome of this send to the end of $L$
    Otherwise Add “failure” to $L$
    If ($|L| = n^2$ and all of the entries of $L$ are “fail”)
        Empty $L$
        sync_stage $\leftarrow$ JAMMING, clock $\leftarrow 0$
    Else
        If (clock $< Wn^3 - 1$), clock $\leftarrow$ clock $+ 1$
        Else
            Empty $L$
            state $\leftarrow$ NORMAL, clock $\leftarrow 0$
```

4.3 The Main Proof

Step 0 will be the step in which the first user starts the protocol. Users will start and stop (perhaps repeatedly) at certain predetermined times throughout the protocol. We say that the sequence of times at which users start and stop is *allowed* if every user runs for at least n^{33} steps each time it starts. Just before any step, t, we will refer to the users that are running the protocol as *live* users. We will say that the state of the system is *normal* if all of these users are in state NORMAL. We will say that it is *good* if

1. it is normal, and
2. for some $C < n^{40} - n^7$, every user has clock $= C$, and
3. every user with $|L| \geq n^2/2$ has a success in the last $n^2/2$ elements of L, and
4. no message in any user's buffer has been in that buffer for more than $n^7/2$ steps.

We say that the state is a *starting* state if the state is good and every clock $= 0$. We say that it is *synchronizing* if

- every user has state = NORMAL, or has state = SYNCHRONIZING with either sync_stage = JAMMING or sync_stage = POLLING, and
- some user has state = SYNCHRONIZING with sync_stage = JAMMING and clock $= 0$.

We say that the system *synchronizes* at step t if it is in a normal state just before step t and in a synchronizing state just after step t. We say that the synchronization is *arbitrary* if every user with state = SYNCHRONIZING, sync_stage = JAMMING and clock $= 0$ just after step t had its clock $< n^{40}$, had no message waiting more than n^7 steps in its buffer, and either had $|L| < n^2$ or had a success in L, just before step t.
Definition: The *interval* starting at any step t is defined to be the period $[t, \ldots, t + n^{33} - 1]$.

Definition: An interval is said to be *productive* for a given user if at least $n^{29}/2$ messages are sent from the user's queue during the interval, or the queue is empty at some time during the interval.
Definition: An interval is said to be *light* for a given user if at most n^{17} messages are placed in the user's queue during the interval.
Definition: Step t is said to be an *out-of-sync* step if either the state is normal just before step t, but two users have different clocks, or the state was not normal just before any step in $[t - 13n^7 + 1, \ldots, t]$. (Intuitively, an out-of-synch step is the result of an "unsuccessful" synchronizing phase.)

Procedure Normal_Step simulates protocol $\mathcal{P}$ from Section 3. Thus, from any starting state until a synchronization, our system simulates $\mathcal{P}$. This implies that our system stops simulating $\mathcal{P}$ when a user starts up, since that user will immediately start a synchronization. Then $\mathcal{P}$ is simulated again once a starting state is reached. We will use the following lemma.

Lemma 4.1 *Given a random variable X taking on non-negative values, and any two events A and B,* $\mathrm{E}[X|A \wedge B] \leq \mathrm{E}[X|B]/\Pr[A|B]$.

Proof: $\mathrm{E}[X \mid B] = \mathrm{E}[X \mid A \wedge B]\Pr[A \mid B] + \mathrm{E}[X \mid \overline{A} \wedge B]\Pr[\overline{A} \mid B]$. □

Lemmas 4.2 to 4.6 outline the analysis of the normal operation of the synchronization phase of our protocol.

Lemma 4.2 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops between steps t and $t + W$. If the system is in a synchronizing state just before step t, then every live user sets* sync_stage *to* FINDING_LEADER *just before some step in $[t, \ldots, t + W]$.*

Proof: A user can have state = SYNCHRONIZING and sync_stage = JAMMING for only $W/2$ steps. Also, every user with state = SYNCHRONIZING, sync_stage = POLLING, and clock $< Wn^3 - n^2$ will set sync_stage to JAMMING after at most n^2 steps; every user with state = SYNCHRONIZING sync_stage = POLLING, and clock $\geq Wn^3 - n^2$ will either set sync_stage to JAMMING within n^2 steps, or switch to state = NORMAL within n^2 steps, and set sync_stage to JAMMING after at most an additional n^4 steps (since when state = NORMAL, a queue step is taken only once every n^2 steps); and every user with state = NORMAL will set sync_stage to JAMMING after at most n^4 steps. The lemma follows by noting that $n^2 + n^4 < W/2$, and that a user remains in sync_stage = JAMMING for $W/2$ steps. □

Lemma 4.3 *Suppose that the protocol is run with a sequence of user start/stop times in which no users start or stop between steps t and $t + 4W$. If every user sets* sync_stage = FINDING_LEADER *before some step in $[t, \ldots, t + W]$ then, with probability at least $1 - e^{-n^3}$, exactly one user sets* sync_stage = SETTING_CLOCK *just before some step in $[t + 2W + 1, \ldots, t + 4W]$ and every other user sets* sync_stage = COPYING_CLOCK *just before some step in $[t + W, \ldots, t + 2W]$.*

Proof: At most one leader is elected since, after being elected it does not allow any users to access the channel for $2W$ steps. Also no user will have sync_stage =

FINDING_LEADER just before step $t+2W$, since sync_stage = FINDING_LEADER for at most W steps.

Suppose P is the last user to set sync_stage = FINDING_LEADER. Then as long as no leader has been elected, the probability that P is elected at a given step is at least $(1/n)(1-(1/n))^{n-1} \geq 1/(en)$. Thus the probability that no leader is elected is at most $(1-1/(en))^{W}$, which is at most e^{-n^3}. Then the leader will spend $2W$ steps with sync_stage = ESTABLISHING_LEADER before setting sync_stage to SETTING_CLOCK, while each of the other users will directly set sync_stage to COPYING_CLOCK. $\square$

Lemma 4.4 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops between steps $\tau - 3W$ and $\tau + 20Wn^2$. If exactly one user sets* sync_stage = SETTING_CLOCK *just before step τ in $[t+2W, \ldots, t+4W]$ and every other user sets* sync_stage = COPYING_CLOCK *just before some step in $[\tau - 3W, \ldots, \tau]$, then, with probability at least $1 - 4Wne^{-n}$, all users set* sync_stage = POLLING *with* clock = 0 *just before step $\tau + 20Wn^2$.*

Proof: The statement in the lemma is clearly true for the user that sets sync_stage = SETTING_CLOCK. Suppose that P is some other user. For each i in the range $0 \leq i < 4W$, if P's clock $= i \bmod 4W$ when the leader's clock $= 0 \bmod 4W$, possibletime[i] will be Yes. If not, P has at least $\lfloor (20Wn^2 - 3W)/(4W) \rfloor$ chances to set possibletime[i] to No, i.e., it has that many chances to poll when its clock $= i \bmod 4W$ and the leader has already set sync_stage = SETTING_CLOCK. Now, $5n^2 - 1 = \lfloor (20Wn^2 - 3W)/(4W) \rfloor$. The probability that P is successful on a given step is at least $\frac{2}{3}(\frac{1}{3n})$, and so the probability that it is unsuccessful in $5n^2-1$ steps is at most $(1 - \frac{2}{9n})^{5n^2-1} \leq e^{-n}$. The lemma follows by summing failure probabilities over all users and moduli of $4W$. $\square$

Lemma 4.5 *Suppose that the protocol is run with a sequence of user start/stop times in which no users start or stop between steps τ and $\tau + Wn^3$. If all users set* sync_stage = POLLING *with* clock = 0 *just before step τ then, with probability at least $1 - Wn^4e^{-n/10}$, all users set* state = NORMAL *and* clock = 0 *just before step $\tau + Wn^3$.*

Proof: Say a sequence of $n^2/2$ steps is *bad* for user P if P does not have a successful transmission on any step in the sequence. Then the probability that a given user P is the first to set sync_stage = JAMMING is at most the probability that it has a bad sequence of $n^2/2$ steps, assuming all other users still have sync_stage = POLLING. This is at most the probability that it either does not send, or is blocked on each step of the sequence, which is at most

$$\left[1 - \frac{1}{3n} + \frac{1}{3n}\left(\frac{1}{3}\right)\right]^{n^2/2} = \left(1 - \frac{2}{9n}\right)^{n^2/2} \leq e^{-n/10}.$$

The lemma follows from summing over all steps (actually this overcounts the number of sequences of $n^2/2$ steps) and all users. $\square$

Lemma 4.6 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops between steps t and $t + 13n^7$. If the system is in a synchronizing state just before step t then, with probability at least $1 - 2Wn^4e^{-n/10}$, there is a t' in $[t + 12n^7, \ldots, t + 13n^7]$ such that it is in the starting state just before step t'.*

Proof: The lemma follows from Lemmas 4.2, 4.3, 4.4 and 4.5. □

Lemmas 4.7 to 4.12 outline the analysis of the robustness of the synchronization phase. Lemma 4.7 shows that no matter what state the system is in (i.e., possibly normal, possibly in the middle of a synchronization), if some user starts a synchronization (possibly because it just started) then, within $W/2$ steps, every user will be in an early part of the synchronization phase. Then Lemma 4.8 shows that with high probability, within a reasonable amount of time, all users will be beyond the stages where they would jam the channel, and furthermore there is a low probability of any going back to those stages (i.e., a low probability of any synchronization starting). Finally, Lemma 4.9 shows that soon all users will be in the polling stage. At this point, as shown in Lemma 4.10, they will either all proceed into the normal state, or if a synchronization is started, they will all detect it and with high probability proceed into a good state as in Lemma 4.6.

Note that these lemmas require the assumption that no users start or stop. This is because they are used for showing that the system returns to a normal state from any situation, even from a bad situation such as a user just having started in the middle of a synchronization phase. If another user starts before the system returns to normal, then we would again use these lemmas to show that the system will return to normal within a reasonable amount of time after that user started.

Lemma 4.7 *If the protocol is run and some user sets* sync_stage = JAMMING *just before step t, and that user does not stop for $W/2$ steps, then there is a t' in $[t, \ldots, t + (W/2)]$ such that just before step t' no user has* state = NORMAL, *and every user that has* sync_stage = POLLING *has* clock $\leq W/2$.

Proof: Every user P that has state = NORMAL or sync_stage = POLLING just before step t will detect the channel being jammed and set state = SYNCHRONIZING and sync_stage = JAMMING just before some step in $[t + 1, \ldots, t + (W/2)]$. The lemma follows. □

Lemma 4.8 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops between steps t and $t + 5nW$. If, just before step t, no user has* state = NORMAL *and every user with* sync_stage = POLLING *has* clock $\leq W/2$, *then, with probability at least $1 - 5Wn^2e^{-n/10}$, there is a t' in $[t, \ldots, t + 5nW]$ such that, just before step t', each user has* state = SYNCHRONIZING *with* sync_stage *set to* SETTING_CLOCK, COPYING_CLOCK, WAITING, *or* POLLING. *Furthermore, if a user has* sync_stage = POLLING, *it has* clock $\leq 5nW + W/2$ *and either it has* clock $\leq n^2/2$ *or it has had a success in the last $n^2/2$ steps.*

Proof: Say a user is *calm* at a given step if it has state = SYNCHRONIZING, and sync_stage set to SETTING_CLOCK, COPYING_CLOCK, WAITING, or POLLING, and if sync_stage = POLLING then its clock is at most $W/2 + 5nW$. Note that each user is uncalm for at most $4W$ steps in $t, \ldots, t + 5nW$, so there is a sequence of W steps in $t, \ldots, t + 5nW$ in which every user is calm. Let t' be the random variable denoting the $(n^2/2 + 1)$st step in this sequence.

Say a sequence of $n^2/2$ steps is *bad* for a user P if P has sync_stage = POLLING just before every step in the sequence, and all of its transmissions during the sequence are blocked by other calm users. The probability that a user with sync_stage = POLLING adds a failure to L on a given step, either due to not transmitting or due to being blocked by a calm user, is at most $1 - 1/(3n) + (1/(3n))(1/3) = 1 - 2/(9n)$. Thus, the probability that a given sequence of $n^2/2$ steps is bad for a given user is at most $(1 - 2/(9n))^{n^2/2} \leq e^{-n/10}$. Thus, with probability at least $1 - 5Wn^2e^{-n/10}$, no sequence of $n^2/2$ steps in $t, \ldots, t + 5nW$ is bad for any user. In particular, the sequence of $n^2/2$ steps preceding t' is not bad for any user, so any user that has sync_stage = POLLING just before step t' with clock $> n^2/2$ has a success in the sequence of $n^2/2$ steps preceding t'. □

Lemma 4.9 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops between steps t and $t + 5nW + (W/2) + 20Wn^2$. If some user sets* sync_stage = JAMMING *just before step t then, with probability at least $1 - 21Wn^3e^{-n/10}$, there is a t' in $[t, \ldots, t + 5nW + (W/2) + 20Wn^2]$ such that, just before step t', each user has* sync_stage = POLLING.

Proof: We know by Lemmas 4.7 and 4.8 that, with probability at least $1 - 5Wn^2e^{-n/10}$, there is a τ in $[t, \ldots, t + 5nW + (W/2)]$ such that, just before step τ, each user has state = SYNCHRONIZING and sync_stage set to SETTING_CLOCK, COPYING_CLOCK, WAITING, or POLLING. Furthermore, if a user has sync_stage = POLLING, it has clock $\leq 5nW + W/2$, and either it has clock $\leq n^2/2$ or it has had a successful poll in the last $n^2/2$ polls.

Unless a user sets sync_stage = JAMMING in the next $20Wn^2$ steps, there will be a step t' such that each user has sync_stage = POLLING. But to set sync_stage = JAMMING, a user with sync_stage = POLLING must be unsuccessful in all transmission attempts during some $n^2/2$ consecutive steps. For a single user and a single set of $n^2/2$ consecutive steps, the probability of this is at most $e^{-n/10}$ (as in the proof of Lemma 4.5). For all users and all possible sets of $n^2/2$ consecutive steps in $\tau, \ldots, \tau + 20Wn^2$, this probability is bounded by $20Wn^3e^{-n/10}$. The lemma follows. □

Lemma 4.10 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops between steps t and $t + Wn^3 + 13n^7$. If the system is in a state in which every user has* state = *NORMAL or* sync_stage = POLLING *just before step t then, with probability at least $1 - 2Wn^4e^{-n/10}$, there is a t' in $[t, \ldots, t + Wn^3 + 13n^7]$ such that the system is in a normal state just before step t'.*

Proof: If no user sets sync_stage = JAMMING during steps $[t, \ldots, t + Wn^3 - 1]$ then the system reaches a normal state before step $t + Wn^3$. Otherwise, suppose that some user sets sync_stage = JAMMING just before step $t'' \leq t + Wn^3 - 1$. By

Lemma 4.6, with probability at least $1-2Wn^4e^{-n/10}$, the system will enter a starting state by step $t''+13n^7$. $\square$

Observation 4.11 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts between steps t and $t+21Wn^2-1$. Suppose that no user sets* sync_stage = JAMMING *during steps $t,\ldots,t+21Wn^2-1$. Then every user has* state = NORMAL *or* sync_stage = POLLING *just before step $t+21Wn^2$.*

Lemma 4.12 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops between steps t and $t+n^8$. Given any system state just before step t, with probability at least $1-3Wn^4e^{-n/10}$, there is a t' in $[t,\ldots,t+n^8]$ such that the system is in a normal state just before step t'.*

Proof: The lemma follows from Lemma 4.10, Observation 4.11 and Lemma 4.9. $\square$

Lemmas 4.13–4.16 and Theorem 4.17 show that if the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated message arrivals distribution then the system is usually in a good state (i.e., synchronized and running the $\mathcal{P}$ protocol), and thus the expected time that messages wait in the buffer is constant.

Lemma 4.13 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops during steps $t,\ldots,t+n^{31}/4-1$. Given any system state just before step t, with probability at least $1-6Wn^4e^{-n/10}$, there is a t' in $[t,\ldots,t+n^{31}/4]$ such that the system is in a starting state just before step t'.*

Proof: By Lemma 4.12, no matter what state the system is in at step t, with probability at least $1-3Wn^4e^{-n/10}$ it will be in a normal state within n^8 steps. Then the probability that it does not enter a synchronizing state within $n^{31}/8$ steps is at most $(1-n^{-30})^{(n^{31}/8)-(n^{29}/8)}\le e^{-n/10}$. Then by Lemma 4.6, once it enters a synchronizing state, with probability at least $1-2Wn^4e^{-n/10}$ it will be in a starting state within $13n^7$ steps. The lemma follows directly from summing failure probabilities. $\square$

Lemma 4.14 *Suppose that the protocol is run with a sequence of user start/stop times in which no user starts or stops between steps t and $t+n^{31}-2n^8$. Given any system state just before step t, with probability at least $1-4Wn^4e^{-n/10}$ there is a t' in $[t,\ldots,t+n^{31}-2n^8]$ such that the system is in a synchronizing state just before step t'.*

Proof: From Lemma 4.12, with probability at least $1-3Wn^4e^{-n/10}$, the system will be in a normal state at some time steps in $[t,\ldots t+n^8]$. Once the system is in a normal state, on every step except one out of every n^2 steps, with probability at least n^{-30} a user will switch to a synchronizing state. The probability of this not happening in the next $n^{31}-3n^8$ steps is at most $(1-n^{30})^{(n^{31}-3n^8-n^{29})}\le e^{-n/2}$. The lemma follows from summing the failure probabilities. $\square$

Arrival distribution. For the remainder of this subsection, we will assume (without further mention) that the arrival distribution is $\{\lambda_i\}_{1\le i\le n}$-dominated.

Lemma 4.15 *Let τ be a non-negative integer less than $n^{40} - n^7$. Suppose that no user starts or stops between steps t and $t+\tau$. If the system is in a starting state just before step t then, with probability at least $1-(13.5)n^{-22}$, the system is in a good state just before step $t+\tau$.*

Proof: Consider the following experiment, in which the protocol is started in a starting state just before step t and run according to the experiment.

```
$i \leftarrow t$
resyncing $\leftarrow$ false
Do forever
    Simulate a step of the protocol
    If (resyncing $=$ false)
        If some message has waited more than $n^7/2$ steps
            FAIL2
        If some user with $|L| \geq n^2/2$ has no success in the last $n^2/2$ elements of $L$
            FAIL1
        If the new state of the system is synchronizing
            If ($i \geq t+\tau-13n^7$), FAIL3
            Else
                resyncing $\leftarrow$ true
                $j \leftarrow 0$
    Else
        If (the new state of the system is a starting state)
            resyncing $\leftarrow$ false
        $j \leftarrow j+1$
        If (($j \geq 13n^7$) and (resyncing $=$ true)), FAIL4
    $i = i+1$
    If ($i \geq t+\tau$), SUCCEED
```

If none of {FAIL1, ..., FAIL4} occurs then the system is in a good state just before step $t+\tau$. As in the proof of Lemma 4.4, the probability that a given element of L is "success" is at least $2/(9n)$, so the probability that FAIL1 occurs is at most $\tau n e^{-n/9}$. By Lemma 3.1, and the fact that at most n^{40}/W starting states occur in the experiment (so $\mathcal{P}$ is started at most n^{40}/W times), the probability that FAIL2 occurs is at most $(n^{40}/W)n^{-60} < n^{-24}$. In the experiment, the clocks of the users never reach n^{40}. If the state is normal, all users have the same value of c, every user with $|L| \geq n^2/2$ has a success in the last $n^2/2$ elements of L, and every user has no message that has waited more than $n^7/2$ steps, then the probability that a given user sets state = SYNCHRONIZING on a given step is at most n^{-30}. Thus, the probability that FAIL3 occurs is at most $13n^{-22}$. By Lemma 4.6, the probability of failing to successfully restart after a given synchronization state is at most $2Wn^4e^{-n/10}$. Hence, the probability of FAIL4 occurring is at most $2\tau W n^4 e^{-n/10}$. □

Definition: Let $T = n^{31}$.

Lemma 4.16 *Suppose that no user starts or stops between steps t and $t+T$. Given any system state just before step t, with probability at least $1-14n^{-22}$, the system is in a good state just before step $t+T$.*

Proof: The lemma follows from Lemma 4.14, Lemma 4.6, and Lemma 4.15. $\square$

Theorem 4.17 *Suppose that no user starts or stops during steps* $[t-T,\ldots,t+n^7]$. *Given any system state just before step* $t-T$, *suppose that a message is generated at step* t. *The expected time that the message spends in the buffer is* $\mathrm{O}(1)$.

Proof: Let X be the time that the message spends in the buffer and let G be the event that the state just before step t is good and has clock less than T. Since X is always at most n^7, $\mathrm{E}[X] \le n^7 \Pr[\overline{G}] + \mathrm{E}[X|G]$. Now, $\Pr[\overline{G}]$ is at most the probability that the state just before step t is not good plus the probability that the state just before step t has clock at least T. By Lemma 4.13, the latter probability is at most $6Wn^4e^{-n/10}$, and, by Lemma 4.16, the former probability is at most $14n^{-22}$. Thus, $\mathrm{E}[X] \le \mathrm{O}(1) + \mathrm{E}[X|G]$. Then $\mathrm{E}[X|G] = \sum_{t'} \mathrm{E}[X|G_{t'}] \Pr[G_{t'}|G]$, where $G_{t'}$ is the event that the good state just before step t has clock $t' < T$. Let $A_{t'}$ be the event that a message p' is born in step t' of the $\mathcal{P}$ protocol. Let B be the event that, prior to that step t' (in the $\mathcal{P}$ protocol), no message has waited more than n^7 steps, and at step t' no message in the buffer has waited more than $n^7/2$ steps. Let Y be the random variable denoting the number of steps required to transmit p' (in $\mathcal{P}$). Then $\mathrm{E}[X|G_{t'}] \le \mathrm{E}[Y|A_{t'} \wedge B]$. (It would be equal except that in our protocol, it is possible to be transferred to the queue before it is successfully sent from the buffer.) So by Lemma 4.1, $\mathrm{E}[X|G_{t'}] \le \mathrm{E}[Y|A_{t'} \wedge B] \le \mathrm{E}[Y|A_{t'}]/\Pr[B|A_{t'}]$. Then by Lemma 3.2, $\mathrm{E}[X|G_{t'}] \le 2\mathrm{E}[Y|A_{t'}] \le \mathrm{O}(1)$, $\forall t' < T$. Thus $\mathrm{E}[X|G] = \mathrm{O}(1)$. $\square$

The remaining results in Subsection 4.3 show that the probability of a message entering a queue is low, the probability of a queue being very full is low, and the rate at which the messages are sent from the queue is high enough that the expected time any given message spends in the queue is low. (Note that most messages will spend no time in the queue.)

Lemma 4.18 *Suppose that the protocol is run with an allowed sequence of user start/stop times. The probability that there is a* t' *in* $[t,\ldots,t+n^{32}]$ *such that the system is in a starting state just before step* t' *is at least* $1 - 6Wn^4e^{-n/10}$, *given any system state just before step* t.

Proof: Divide the interval of n^{32} steps into subintervals of $n^{31}/4$ steps each. Since at most n users can start or stop during the interval, and those that start continue for the remainder of the interval, there must be a subinterval in which no users start or stop. The result follows from Lemma 4.13. $\square$

Lemma 4.19 *Suppose that the protocol is run with a given allowed sequence of user start/stop times in which no user starts or stops between steps* $t-T$ *and* $t+n^7/2$. *Given any system state just before step* $t-T$, *suppose that a message* R *arrives at user* P *at step* t. *The probability that* R *enters the queue is at most* $16n^{-22}$.

Proof: Let X be the event that R enters the queue. Let G be the event that just before step t the state is good and has clock less than T. Then by Lemma 4.16 and Lemma 4.13, $\Pr[X] \le 1\Pr[\overline{G}] + \Pr[X|G] \le 14n^{-22} + 6Wn^4e^{-n/10} + \Pr[X|G]$. Note that $\Pr[X|G] = \sum_{t'} \Pr[X|G_{t'}] \Pr[G_{t'}|G]$, where $G_{t'}$ is the event that the good state just before step t has clock t'. Consider the following experiment (the corresponding intuition and analysis are presented after its description; so the reader is asked to first skip to the end of the description and then study the description as needed):

```
$i \leftarrow 0$
Do forever
    If $i = t'$
        Add a message $R$ to user $P$
        Simulate a step of the protocol (except for the arbitrary synchronizations)
        If some message has been in a buffer more than $n^7/2$ steps
            FAIL1
        If some user with $|L| \geq n^2/2$ has no success in the last $n^2/2$ elements of $L$
            FAIL1
    Else
        Simulate a step of the protocol (except for the arbitrary synchronizations)
        If $(i < t')$ and some message has waited more than $n^7$ steps
            FAIL1
        If $(i > t')$ and some message has waited more than $n^7$ steps
            FAIL3
        If some user with $|L| \geq n^2$ has no success in the last $n^2$ elements of $L$
            FAIL1
    $i = i + 1$
    If $(i \geq t' + n^7/2)$
        If message $Q$ has been sent, SUCCEED
        Else FAIL2
```

This experiment models the system beginning at a start state, and going for $t' + n^7/2 \leq T + n^7/2$ steps, but assumes that there are no arbitrary synchronizations, and that there is a message R generated at P at clock t'. The experiment fails at step $i = t'$ if the system enters a state which is not good at that point. It fails at a step $i < t'$ or $t' < i < t' + n^7/2$ if the system does a non-arbitrary synchronization at that point. It fails at step $i = t' + n^7/2$ if the message R has not been sent successfully. Let A be the event that FAIL1 occurs, B be the event that FAIL2 occurs, C be the event that FAIL3 occurs, and S be the event that the experiment does not fail during steps $1, \ldots, t'$. The probability that R is still in the buffer after step $t + n^7/2 + 1$, or the real system synchronizes before step $t + n^7/2 + 1$, conditioned on the fact that the state just before step t is good and has clock t' and on the fact that message R is generated at P at step t', is at most the sum of (1) $\Pr[C \mid S]$, (2) $\Pr[A \mid S]$, (3) $\Pr[B \mid S]$, and (4) the probability that there is an arbitrary synchronization during steps $t, \ldots, t + n^7/2 - 1$. Probability (4) is at most $n(n^7/2)(n^{-30}) = n^{-22}/2$. Now note that $\Pr[A \mid S] \leq \Pr[A]/\Pr[S]$. By the proof of Lemma 4.15 (using Lemma 3.2),

$$\Pr[S] \geq 1 - [n^{40}(ne^{-n/9}) + n^{-60}] \geq \frac{1}{2}$$

and

$$\Pr[A] \leq n^{40}(ne^{-n/9}) + n^{-60}.$$

Thus $\Pr[A \mid S] \leq 3n^{-60}$.

Note also that $\Pr[B \mid S] \leq \Pr[B]/\Pr[S]$. By Lemma 3.2, $\Pr[B] \leq n^{-60}$. (This can only be decreased by a queue step causing a synchronization.) Then $\Pr[B \mid S] \leq 2n^{-60}$.

Finally, $\Pr[C \mid S] = 0$, since all messages at step t' have waited for at most $n^7/2$ steps, and the experiment stops at step $t' + n^7/2$.

Thus, $\Pr[X|G] \leq n^{-22}$, which completes the proof. $\square$

Lemma 4.20 *Let j be an integer in $[0,\ldots,14]$. Suppose that no user starts or stops during steps $t,\ldots,t+n^{14+j}-1$. If the system is in a starting state just before step t then the probability that the system enters a synchronizing state during steps $t,\ldots,t+n^{14+j}-1$ is at most $2n^{-15+j}$.*

Proof: The probability that an arbitrary synchronization occurs during steps $t,\ldots,t+n^{14+j}-1$ is at most $n \cdot n^{-30} \cdot n^{14+j} = n^{-15+j}$. Following the proof of Lemma 4.15, we see that the probability that a non-arbitrary synchronization occurs during these steps is at most $n^{-60}+n^{15+j}e^{-n/9}$. (The probability that a message waits in a buffer more than n^7 steps is at most n^{-60} by Lemma 3.1 and the probability that some user gets n^2 failures on L is at most $n^{14+j} \cdot n \cdot e^{-n/9}$.) $\square$

Lemma 4.21 *Suppose that no user starts or stops during the interval $[t,\ldots,t+n^{33}-1]$. If the system is in a starting state just before step t then the probability that either some step in the interval is an out-of-sync step or that the system is in a starting state just before more than n^7 steps in the interval is at most $3Wn^{11}e^{-n/10}$.*

Proof: If the system is in a starting state x times, where $x > n^7$, then at least $x - n^7/2$ of these must be followed by fewer than $2n^{26}$ steps before the next synchronization phase. By Lemma 4.20, the probability of fewer than $2n^{26}$ steps occurring between a starting state and the next synchronization phases is at most $2n^{-2}$. Thus, the probability of this happening after at least $x - n^7/2$ of the x starting states is at most $2^x(2n^{-2})^{x-n^7/2}$ which is at most $2^{-n^7/2}$.

If the system is in a starting state just before at most n^7 steps in the interval, then the only time that the system could have an out-of-sync step during the interval is during at most n^7-1 subintervals which start with a synchronizing state and end in a starting state. By the proof of Lemma 4.6, the probability that a given subinterval contains an out-of-sync step is at most $2Wn^4e^{-n/10}$. Thus, the probability that an out-of-sync step occurs in the interval is at most $n^7(2Wn^4e^{-n/10})$. $\square$

Lemma 4.22 *Suppose that the protocol is run with a given allowed sequence of user start/stop times after step t, and a given system state just before step t. Divide the interval starting at step t into blocks of n^4 steps. The probability that the interval has more than $27n^{11}$ blocks containing non-normal steps is at most $7Wn^{12}e^{-n/10}$.*

Proof: Let S contain the first step of the interval and each step during the interval in which a user starts or stops. Then $|S| \leq 2n+1$. Let S' contain S plus for each step $s \in S$, all steps after s until the system returns to a normal state. By Lemma 4.12, with probability at least $1-(2n+1)(3Wn^4e^{-n/10})$, S' can be covered by $2n+1$ sequences of at most n^8 steps each. Then the set S' partitions the other steps in the interval into at most $2n+1$ subintervals, such that the state is normal just before each subinterval, and no users start or stop during any subinterval. We perform the following analysis for each of these subintervals.

By Lemma 4.6, once the system enters a synchronizing state, with probability at least $1-2Wn^4e^{-n/10}$ it will be in a starting state within $13n^7$ steps. Once the system is in a starting state, by Lemma 4.21 with probability at least $1 - 3Wn^{11}e^{-n/10}$, it will enter a synchronizing state at most $n^7 + 1$ times, and each synchronizing phase will last at most $13n^7$ steps.

In total, the probability of not performing as stated above is at most

$$(2n+1)(3Wn^4e^{-n/10} + 2Wn^4e^{-n/10} + 3Wn^{11}e^{-n/10}) \leq 7Wn^{12}e^{-n/10}.$$

Finally, the set S' can intersect at most $(2n+1)((n^8/n^4)+1)$ blocks of size n^4. Then, in each of the $2n+1$ subintervals of steps between those of S', there are at most n^7+2 synchronizing phases, each of which can intersect at most $((13n^7/n^4)+1)$ blocks of size n^4. Altogether, at most $27n^{11}$ blocks of size n^4 will contain non-normal steps. $\square$

Corollary 4.23 *Let x be an integer in the range $0 \leq x \leq n^{29} - 54n^{11}$. Suppose that the protocol is run with a given allowed sequence of user start/stop times after step t, and a given system state just before step t. Focus on a particular non-empty queue at step t. The probability that the queue remains non-empty for the next $xn^4 + 54n^{15}$ steps but fewer than x messages are delivered from it during this period is at most $7Wn^{12}e^{-n/10}$.*

Proof: Divide the next $xn^4 + 54n^{15} \leq n^{33}$ steps into blocks of size n^4. By Lemma 4.22, with probability at least $1-7Wn^{12}e^{-n/10}$, at most $54n^{11}$ of these blocks will either contain a non-normal step, or precede a block which contains a non-normal step. The corollary follows by noting that if block i contains all normal steps and no synchronization is started in block $i+1$, then a message must have been sent from the queue during block i. $\square$

Lemma 4.24 *Suppose that the protocol is run with a given allowed sequence of user start/stop times after step t, and a given system state just before step t. Then the probability that the interval starting at t is light for a given user is at least $1 - 8Wn^{12}e^{-n/10}$.*

Proof: As in the proof of Lemma 4.22, with probability at least $1-7Wn^{12}e^{-n/10}$, the non-normal steps could be covered by at most $(2n+1)+(2n+1)(n^7+2)$ subintervals of at most n^8 steps each, and each of the subintervals would contribute at most n^8+n^7 messages to the queue (including the at most n^7 that could be transferred from the user's buffer). If this were the case, at most $3n^{16}$ messages would be placed in the queue during the interval. $\square$

Lemma 4.25 *Suppose that the protocol is run with a given allowed sequence of user start/stop times after step t, and a given system state just before step t. The probability that the interval starting at t is productive for a given user is at least $1 - 7Wn^{12}e^{-n/10}$.*

Proof: Follows from Corollary 4.23. $\square$

Lemma 4.26 *Suppose that the protocol is run with a given allowed sequence of user start/stop times before step t. The probability that more than $n^{17} + j(n^{33} + n^7)$ messages are in a queue just before step t is at most $e^{-jn/30}$ for $j \geq 1$ and at most $e^{-n/30}$ for $j = 0$.*

Proof: For every non-negative integer j, we will refer to the interval $[t - (j+1)n^{33} + 1, \ldots, t - jn^{33}]$ as "interval j". Choose k such that the queue was empty just before some step in interval k, but was not empty just before any steps in intervals 0 to $(k-1)$. We say that interval j is "bad" if it is not both productive and light for the user. The size of the queue increases by at most $n^{33} + n^7$ during any interval. If interval k is not bad, then the queue size increases by at most n^{17} during interval k. If interval j is not bad for $j < k$, then the queue size decreases by at least $n^{29}/2 - n^{17}$ during interval k. Thus, if b of intervals 0 to k are bad, then the size of the queue just before step t is at most

$$(k+1)(n^{33} + n^7) - (k+1-b)(n^{33} + n^7 + n^{29}/2 - n^{17}) + n^{17}.$$

This quantity is at most $n^{17} + i(n^{33} + n^7)$ unless $b > i/2 + k/(8n^4)$. Thus, the probability that the queue has more than $n^{17} + i(n^{33} + n^7)$ messages just before step t is at most the probability that, for some non-negative integer k, more than $(i/2) + (k/(8n^4))$ of intervals 0 to k are bad. By Lemmas 4.24 and 4.25, the probability that a given interval is bad is at most $16Wn^{12}e^{-n/10}$. Let $X = 16Wn^{12}e^{-n/10}$. Then, for $i \geq 1$, the failure probability is at most

$$\begin{aligned}
&\sum_{k\geq 0} \binom{k}{\lfloor (i/2) + (k/(8n^4)) \rfloor + 1} X^{\lfloor (i/2)+(k/(8n^4))\rfloor + 1} \\
&\qquad \leq \sum_{k\geq 0} (16en^4X)^{\lfloor (i/2)+(k/(8n^4))\rfloor + 1} \\
&\qquad \leq \sum_{k\geq 0} (16en^4X)^{(i/2)+(k/(8n^4))} \\
&\qquad \leq (16en^4X)^{i/2} \sum_{k\geq 0} (16en^4X)^{k/(8n^4)} \\
&\qquad \leq (16en^4X)^{i/2} 8n^4 \sum_{k\geq 0} (16en^4X)^{k} \\
&\qquad \leq 2(8n^4)(16en^4X)^{i/2} \leq e^{-in/30}.
\end{aligned}$$

For $i = 0$, this probability is at most

$$\begin{aligned}
\sum_{k\geq 0} \binom{k}{\lfloor k/(8n^4) \rfloor + 1} X^{\lfloor k/(8n^4)\rfloor + 1} &\leq \sum_{k\geq 0} (16en^4X)^{\lfloor k/(8n^4)\rfloor + 1} \\
&\leq (16en^4X) \sum_{k\geq 0} (16en^4X)^{\lfloor k/(8n^4)\rfloor} \\
&\leq 2(8n^4)(16en^4X) \leq e^{-n/30}.
\end{aligned}$$

□

Lemma 4.27 *Suppose that the protocol is run with a given allowed sequence of user start/stop times after step $t + n^{32}$. Suppose that no users start or stop during steps $[t - T, \ldots, t + n^{32}]$ and that the system state just before step $t - T$ is given. The probability that an out-of-sync step occurs before a starting step after t is at most $4Wn^{11}e^{-n/10}$.*

Proof: By Lemma 4.13, the probability of not having a start state just before any step in the subinterval $[t - T, \ldots, t - T/2]$ is at most $6Wn^4e^{-n/10}$. Then by (the proof of) Lemma 4.21, the probability of having an out-of-synch step before step $t + n^{32}$ is at most $3Wn^{11}e^{-n/10}$. Finally, by Lemma 4.13, the probability of not having a start state in the subinterval $[t, \ldots, t + T/2]$ is at most $6Wn^4e^{-n/10}$. The lemma follows by summing the failure probabilities. □

Lemma 4.28 *Suppose that the protocol is run with a given allowed sequence of user start/stop times after step t, and a given system state just before step t in which queue Q contains at least x messages. Then the expected time until at least x messages have been sent from Q is $\mathrm{O}(xn^4 + n^{15})$.*

Proof: Our first case is when $x \leq n^{29}/2$. Let A be the event that at least x messages are sent in steps $t, \ldots, t + xn^4 + 54n^{15} - 1$. We refer to the interval $[t + xn^4 + 54n^{15} + (k-1)n^{33}, \ldots, t + xn^4 + 54n^{15} + kn^{33} - 1]$ as "interval k". Let C_k be the event that interval k is productive. Let E_x be the expected time to send the x messages. Using Corollary 4.23 and Lemma 4.25,

$$\begin{aligned}
E_x &\leq (xn^4 + 54n^{15}) + n^{33}\Pr[\overline{A}] + \sum_{k>1} n^{33}\Pr[\bigwedge_{1\leq i\leq k-1} \overline{C_i}] \\
&\leq xn^4 + 54n^{15} + \sum_{k\geq 1} n^{33}(7Wn^{12}e^{-n/10})^k \\
&= \mathrm{O}(xn^4 + n^{15}).
\end{aligned}$$

Our second and last case is when $x > n^{29}/2$. Let $r = \lceil 2x/n^{29} \rceil$. Note that after r productive intervals, at least x messages will be sent. Let D_k be the event that intervals 1 to k do not contain at least r productive intervals, but that intervals 1 to $(k+1)$ do contain r productive intervals.

$$\begin{aligned}
E_x &\leq \sum_{k\geq r}(k+1)n^{33}\Pr[D_k] \\
&\leq n^{33}(2r + \sum_{k\geq 2r}(k+1)\Pr[D_k]) \\
&\leq n^{33}(2r + \sum_{k\geq 2r}(k+1)\binom{k}{k-r}(7Wn^{12}e^{-n/10})^{k-r}) \\
&\leq n^{33}(2r + \sum_{k\geq 2r}(k+1)2^k(7Wn^{12}e^{-n/10})^{k-r}) \\
&= \mathrm{O}(n^{33}r) = \mathrm{O}(xn^4).
\end{aligned}$$

□

Theorem 4.29 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution, a given allowed sequence of user start/stop times in which no users start or stop during steps $[t-n^{33},\ldots,t+n^{33}]$. Suppose that a message is generated at step t. The expected time that the message spends in the queue is* $\mathrm{O}(1)$.

Proof: Let I_ℓ be the interval $[t-\ell n^{33}+1,\ldots,t-(\ell-1)n^{33}]$. Let A_0 be the event that the size of the queue is at most $n^{17}-1$ just before step $t-n^{33}+1$, and, for $i\ge 1$, let A_i be the event that the size of the queue just before step $t-n^{33}+1$ is in the range $[n^{17}+(i-1)(n^{33}+n^7), n^{17}+i(n^{33}+n^7)-1]$. Let B the event that interval I_1 is light. Let C be the event that the message enters the queue. Let t' be the random variable denoting the smallest integer such that $t'\ge t$ and the state of the system just before step t' is a starting state. Let t'' be the random variable denoting the smallest integer such that $t''\ge t$ and step t'' is out-of-sync. Let F be the event that $t'<t''$. Let X be the random variable denoting the amount of time that the message spends in the queue. All probabilities in this proof will be conditioned on the state of the fact that no users start or stop during steps $[t-n^{33},\ldots,t+n^{33}]$.

We start by bounding $\sum_{i\ge1}\mathrm{E}[X\mid A_i\wedge C]\Pr[A_i\wedge C]$. By Lemma 4.26, $\Pr[A_i]\le e^{-(\max\{i-1,1\})n/30}$ so $\Pr[A_i\wedge C]\le e^{-(\max\{i-1,1\})n/30}$. By Lemma 4.28,

$$\mathrm{E}[X\mid A_i\wedge C]\le \mathrm{E}[t'-t\mid A_i\wedge C]+\mathrm{O}(n^4(n^{17}+(i+1)(n^{33}+n^7))).$$

(Since A_i holds, there are at most $n^{17}+i(n^{33}+n^7)$ messages in the queue before interval I_1 and at most $n^{33}+n^7$ get added during interval I_1.) By Lemma 4.18, $\mathrm{E}[t'-t\mid A_i\wedge C]$ is at most $\sum_{j\ge1}n^{32}(6Wn^4e^{-n/10})^{j-1}=\mathrm{O}(n^{32})$. Thus, $\mathrm{E}[X\mid A_i\wedge C]=(i+1)\mathrm{O}(n^{37})$. Thus,

$$\sum_{i\ge1}\mathrm{E}[X\mid A_i\wedge C]\Pr[A_i\wedge C]\le\sum_{i\ge1}e^{-(\max\{i-1,1\})n/30}(i+1)\mathrm{O}(n^{37})=\mathrm{O}(1).$$

We now bound $\mathrm{E}[X\mid A_0\wedge\overline{B}\wedge C]\Pr[A_0\wedge\overline{B}\wedge C]$. By Lemma 4.24, $\Pr[\overline{B}]\le 8Wn^{12}e^{-n/10}$, so $\Pr[A_0\wedge\overline{B}\wedge C]\le 8Wn^{12}e^{-n/10}$. As above, $\mathrm{E}[X\mid A_0\wedge\overline{B}\wedge C]=\mathrm{O}(n^{37})$, so

$$\mathrm{E}[X\mid A_0\wedge\overline{B}\wedge C]\Pr[A_0\wedge\overline{B}\wedge C]\le(8Wn^{12}e^{-n/10})\mathrm{O}(n^{37})=\mathrm{O}(1).$$

Next, we bound $\mathrm{E}[X\mid A_0\wedge\overline{F}\wedge C]\Pr[A_0\wedge\overline{F}\wedge C]$. By Lemma 4.27, the probability of $\overline{F}$ is at most $4Wn^{11}e^{-n/10}$, so $\Pr[A_0\wedge\overline{F}\wedge C]\le 4Wn^{11}e^{-n/10}$. As above, $\mathrm{E}[X\mid A_0\wedge\overline{F}\wedge C]$ is at most $\mathrm{E}[t'-t\mid A_0\wedge\overline{F}\wedge C]+\mathrm{O}(n^{37})$. Since C occurs, the system is in a synchronization state just before some state in $[t,\ldots,t+n^7]$. Since $\overline{F}$ occurs, there is an out-of-sync step in $[t,\ldots,t+14n^7]$. By Lemma 4.18, the expected time from this out-of-sync step until a starting state occurs is at most $\sum_{j\ge1}n^{32}(6Wn^4e^{-n/10})^{j-1}=\mathrm{O}(n^{32})$. Thus, $\mathrm{E}[t'-t\mid A_0\wedge\overline{F}\wedge C]=\mathrm{O}(n^{32})$ and $\mathrm{E}[X\mid A_0\wedge\overline{F}\wedge C]=\mathrm{O}(n^{37})$. Thus,

$$\mathrm{E}[X\mid A_0\wedge\overline{F}\wedge C]\Pr[A_0\wedge\overline{F}\wedge C]\le(4Wn^{11}e^{-n/10})\mathrm{O}(n^{37})=\mathrm{O}(1).$$

Finally, we bound $\mathrm{E}[X\mid A_0\wedge B\wedge F\wedge C]\Pr[A_0\wedge B\wedge F\wedge C]$. By Lemma 4.19, the probability of C is at most $16n^{-22}$, so $\Pr[A_0\wedge B\wedge F\wedge C]\le 16n^{-22}$. We now wish to bound $\mathrm{E}[X\mid A_0\wedge B\wedge F\wedge C]$. Since A_0 and B hold, the size of the queue just before step t is at most $2n^{17}$. Suppose that $t' > t + 2n^{21} + 13n^7$. Then, since F holds, no step in $t, \ldots, t + 2n^{21} + 13n^7$ is out-of-sync. Suppose first that no step in $t, \ldots, t + 2n^{21} + 13n^7$ is out-of-sync and that the state is normal before each step in $t, \ldots, t + 2n^{21}$. Then all of the clocks will be the same, so at least $2n^{17}$ messages will be sent from the queue during this period. Suppose second that no step in $t, \ldots, t + 2n^{21} + 13n^7$ is out-of-sync, but that the state is not normal just before some step in $[t, \ldots, t + 2n^{21}]$. Then since no state in $t, \ldots, t + 2n^{21} + 13n^7$ is out-of-sync, $t' \leq t + 2n^{21} + 13n^7$. Finally, suppose that $t' \leq t + 2n^{21} + 13n^7$. By Lemma 4.28, $\mathrm{E}[X \mid A_0 \wedge B \wedge C \wedge F]$ is at most $t' - t + \mathrm{O}(n^4 \cdot 2n^{17}) = \mathrm{O}(n^{21})$. Thus,

$$\mathrm{E}[X \mid A_0 \wedge B \wedge F \wedge C] \Pr[A_0 \wedge B \wedge F \wedge C] \leq 16n^{-22}\mathrm{O}(n^{21}) = \mathrm{O}(1).$$

□

Observation 4.30 *When the protocol is run, every message spends at most n^7 steps in the buffer.*

Theorem 4.31 *Suppose that the protocol is run with a $\{\lambda_i\}_{1 \leq i \leq n}$-dominated arrival distribution and a given allowed sequence of user start/stop times. Suppose that a message is generated at step t. Then the expected time that the message spends in the queue is $\mathrm{O}(n^{37})$.*

Proof: Let X be the random variable denoting the size of the queue just before step t. By Lemma 4.26, for $i \geq 1$, the probability that $X > n^{17} + i(n^{33} + n^7)$ is at most $e^{-in/30}$. Given a particular value of X, Lemma 4.28 shows that the expected time to send the message is $\mathrm{O}(Xn^4 + n^{15})$. Thus, the overall expected time to send the message is

$$\mathrm{O}(n^4(n^{17} + n^{33} + n^7) + n^{15}) + \sum_{i \geq 2} \mathrm{O}(n^4(n^{17} + i(n^{33} + n^7)) + n^{15}) e^{-(i-1)n/30} = \mathrm{O}(n^{37}).$$

□

4.4 Final Results

For $v \in [n]$, let T_v be the set of steps in which user v is running.

Theorem 4.32 *Suppose that the protocol is run with a $\{\lambda_i\}_{1 \leq i \leq n}$-Bernoulli arrival distribution and a given sequence of user start/stop times in which each user runs for at least $8n^{71}$ steps every time it starts. Then $\mathrm{E}[W_{\mathrm{avg}}] = \mathrm{O}(1)$.*

Proof: First note that the sequence of user start/stop times is allowed. Let R be the set of steps within n^{33} steps of the time that a user starts or stops. Lemma 4.33 proves that if the $\{\lambda_i\}_{1 \leq i \leq n}$-Bernoulli arrival distribution is conditioned on having at most m messages arrive by time t, the resulting arrival distribution is $\{\lambda_i\}_{1 \leq i \leq n}$-dominated. Therefore, the system described in the statement of the theorem satisfies the conditions of Lemma 4.34 with (from Theorem 4.17 and Theorem 4.29) $C' = \mathrm{O}(1)$

and (from Theorem 4.31 and Observation 4.30) $C = O(n^{37})$. From the condition given in the statement of this theorem, we can see that

$$S = \max_{v\in V} \limsup_{t\to\infty} \frac{|R \cap T_v \cap [t]|}{|T_v \cap [t]|} \le n^{-37}.$$

(The worst case for S is when a user runs for $8n^{71} + 6(n-1)n^{33} + 2n^{33}$ steps, and the other $n-1$ users have [ending, starting, ending, starting] times

$$[2in^{33}, 2(n-1)n^{33} + 2in^{33}, 2(n-1)n^{33} + 2in^{33} + 8n^{71}, 4(n-1)n^{33} + 2in^{33} + 8n^{71}],$$

for $1 \le i \le n-1$. Then $|R| = 8(n-1)n^{33} + 2n^{33}$, including the n^{33} steps just after the user starts and the n^{33} steps just before the user stops.) The theorem then follows from Lemma 4.34. (Note that C and C' are actually functions of λ, but λ is a constant.) □

Lemma 4.33 *Consider the distribution obtained from the $\{\lambda_i\}_{1\le i\le n}$-Bernoulli arrivals distribution by adding the condition that at most m messages arrive by step t. The resulting arrival distribution is $\{\lambda_i\}_{1\le i\le n}$-dominated.*

Proof: Let $A_{v,t'}$ denote the probability that a message arrives at user v at time t' (under the $\{\lambda_i\}_{1\le i\le n}$-Bernoulli arrivals distribution). Let E be any event concerning the arrival of messages at steps other than t' or at users other than v. Let C be the event that at most m messages arrive during steps $1,\ldots,t$. We wish to show that $\Pr[A_{v,t'} \mid C \wedge E] \le \lambda_v$. If $t' > t$ then $\Pr[A_{v,t'} \mid C \wedge E] = \lambda_v$ by the independence of the $\{\lambda_i\}_{1\le i\le n}$-Bernoulli arrivals distribution, so suppose that $t' \le t$. Let E' denote the part of event E concerning arrivals at steps $1,\ldots,t$. By the independence of the $\{\lambda_i\}_{1\le i\le n}$-Bernoulli arrivals distribution, $\Pr[A_{v,t'} \mid C \wedge E] = \Pr[A_{v,t'} \mid C \wedge E']$. Let W be the set containing every possible sequence of message arrivals during steps $1,\ldots,t$ with the arrival at user v and step t' omitted. Let W' be the set of elements of W which satisfy E' and have fewer than m arrivals and let W'' be the set of elements of W which satisfy E' and have exactly m arrivals.

$$\begin{aligned}
\Pr[A_{v,t'} \mid C \wedge E'] &= \sum_{w\in W} \Pr[A_{v,t'} \mid w \wedge C \wedge E'] \Pr[w \mid C \wedge E'] \\
&= \sum_{w\in W'} \Pr[A_{v,t'} \mid w \wedge C] \Pr[w \mid C \wedge E'] \\
&\quad + \sum_{w\in W''} \Pr[A_{v,t'} \mid w \wedge C] \Pr[w \mid C \wedge E'] \\
&= \sum_{w\in W'} \Pr[A_{v,t'} \mid w] \Pr[w \mid C \wedge E'] \\
&= \lambda_v \sum_{w\in W'} \Pr[w \mid C \wedge E'] \ \le\ \lambda_v.
\end{aligned}$$

□

Lemma 4.34 *Suppose that, for every m and t, a protocol running on n users has the property: for all users v, if a message P is generated at user v at step $t \in R$ and* *is one of the first m messages generated, then the expected time before message P is sent is at most C, and if a message P is generated at user v at step $t \in \overline{R}$ and is one of the first m messages generated, then the expected time before message P is sent is at most C'. Then* $\mathrm{E}[W_{\text{avg}}] \leq 2(SC + C')$, *where* $S = \max_{v \in V} \limsup_{t \to \infty} \frac{|R \cap T_v \cap [t]|}{|T_v \cap [t]|}$.

Proof: Recall that $\lambda = \sum_{v \in V} \lambda_v$, that $\lambda_v > 0$ for all $v \in V$ and that $W_{\text{avg}} = \lim_{m \to \infty} \frac{1}{m} \sum_{i=1}^{m} W_i$, where W_i is the delay of the ith message generated in the system.

$$\mathrm{E}[W_{\text{avg}}] = \mathrm{E}\left[\lim_{m \to \infty} \frac{1}{m} \sum_{i=1}^{m} W_i\right] \leq \mathrm{E}\left[\limsup_{m \to \infty} \frac{1}{m} \sum_{i=1}^{m} W_i\right] = \limsup_{m \to \infty} \frac{1}{m} \sum_{i=1}^{m} \mathrm{E}[W_i].$$

Now let $A_{i,v,t}$ be the event that the ith message is generated at user v at step t. Then

$$\begin{aligned} \sum_{i=1}^{m} \mathrm{E}[W_i] &= \sum_{i=1}^{m} \sum_{t \geq 0} \sum_{v \in V} \mathrm{E}[W_i \mid A_{i,v,t}] \Pr[A_{i,v,t}] \\ &= \sum_{v \in V} \sum_{t \in T_v} \sum_{i=1}^{m} \mathrm{E}[W_i \mid A_{i,v,t}] \Pr[A_{i,v,t}]. \end{aligned}$$

Let $B_{m,v,t}$ be the event that one of the first m messages is generated at user v at step t. Now, the properties of the protocol given in the lemma are equivalent to the following: for any $v \in V$, m and $t \in T_v$,

$$\begin{aligned} \sum_{i=1}^{m} \mathrm{E}[W_i \mid A_{i,v,t}] \Pr[A_{i,v,t} \mid B_{m,v,t}] &\leq C, \text{ if } t \in R, \text{ and} \\ \sum_{i=1}^{m} \mathrm{E}[W_i \mid A_{i,v,t}] \Pr[A_{i,v,t} \mid B_{m,v,t}] &\leq C', \text{ if } t \in \overline{R}. \end{aligned}$$

Since, for $i \leq m$, $\Pr[A_{i,v,t}] = \Pr[A_{i,v,t} \wedge B_{m,v,t}] = \Pr[A_{i,v,t} \mid B_{m,v,t}] \Pr[B_{m,v,t}]$,

$$\begin{aligned} \sum_{i=1}^{m} \mathrm{E}[W_i] &= \sum_{v \in V} \sum_{t \in T_v} \sum_{i=1}^{m} \mathrm{E}[W_i \mid A_{i,v,t}] \Pr[A_{i,v,t}] \\ &= \sum_{v \in V} \sum_{t \in T_v} \sum_{i=1}^{m} \mathrm{E}[W_i \mid A_{i,v,t}] \Pr[A_{i,v,t} \mid B_{m,v,t}] \Pr[B_{m,v,t}] \\ &= \sum_{v \in V} \sum_{t \in T_v} \Pr[B_{m,v,t}] \sum_{i=1}^{m} \mathrm{E}[W_i \mid A_{i,v,t}] \Pr[A_{i,v,t} \mid B_{m,v,t}] \\ &\leq \sum_{v \in V} \left(\sum_{t \in R \cap T_v} \Pr[B_{m,v,t}] C + \sum_{t \in \overline{R} \cap T_v} \Pr[B_{m,v,t}] C' \right). \end{aligned}$$

Let $\mu_t = \sum_{v' \in V} \lambda_{v'} |T_{v'} \cap [t]|$, i.e. the expected number of messages generated in the system through time t. Note that $\Pr[B_{m,v,t}] \leq \lambda_v$, and, for $m < \mu_t$, $\Pr[B_{m,v,t}] \leq \lambda_v \exp\{-(\mu_t - m)^2/(2\mu_t)\}$, by a Chernoff bound. Then for any $T^* \subseteq T_v$,

$$\sum_{t \in T^*} \Pr[B_{m,v,t}] \leq \sum_{t \in T^*, \mu_t < 2m} \lambda_v + \sum_{t \in T^*, \mu_t \geq 2m} \lambda_v \exp\{-(\mu_t - m)^2/(2\mu_t)\}$$

$$
\begin{aligned}
&\leq \lambda_v|T^* \cap \{t : \mu_t < 2m\}| + \lambda_v \sum_{t\in T^*, \mu_t \geq 2m} \exp\{-(\mu_t - m)/4\} \\
&\leq \lambda_v|T^* \cap \{t : \mu_t < 2m\}| + \lambda_v \sum_{i\geq 0} \exp\{-(m + i\lambda_v)/4\} \\
&\leq \lambda_v|T^* \cap \{t : \mu_t < 2m\}| + \lambda_v e^{-m/4} \sum_{i\geq 0} (e^{-\lambda_v/4})^i \\
&\leq \lambda_v|T^* \cap \{t : \mu_t < 2m\}| + \mathrm{O}(1).
\end{aligned}
$$

Consequently,

$$
\begin{aligned}
\mathrm{E}[W_{\text{avg}}] &\leq \limsup_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} \mathrm{E}[W_i] \\
&\leq \limsup_{m\to\infty} \frac{1}{m} \sum_{v\in V} [C(\lambda_v|R \cap T_v \cap \{t : \mu_t < 2m\}| + \mathrm{O}(1)) \\
&\quad + C'(\lambda_v|\overline{R} \cap T_v \cap \{t : \mu_t < 2m\}| + \mathrm{O}(1))] \\
&\leq C(\limsup_{m\to\infty} \frac{1}{m} \sum_{v\in V} \lambda_v|R \cap T_v \cap \{t : \mu_t < 2m\}|) \\
&\quad + C'(\limsup_{m\to\infty} \frac{1}{m} \sum_{v\in V} \lambda_v|\overline{R} \cap T_v \cap \{t : \mu_t < 2m\}|).
\end{aligned}
$$

We bound the factor multiplied by C as follows.

$$
\begin{aligned}
&\limsup_{m\to\infty} \frac{1}{m} \sum_{v\in V} (\lambda_v|R \cap T_v \cap \{t : \mu_t < 2m\}|) \\
&= \limsup_{m\to\infty} \sum_{v\in V} \frac{\lambda_v|T_v \cap \{t : \mu_t < 2m\}|}{m} \left(\frac{|R \cap T_v \cap \{t : \mu_t < 2m\}|}{|T_v \cap \{t : \mu_t < 2m\}|} \right) \\
&\leq \limsup_{m\to\infty} \left(\max_{v\in V} \frac{|R \cap T_v \cap \{t : \mu_t < 2m\}|}{|T_v \cap \{t : \mu_t < 2m\}|} \right) \sum_{v\in V} \frac{\lambda_v|T_v \cap \{t : \mu_t < 2m\}|}{m} \\
&\leq \left(\limsup_{m\to\infty} \max_{v\in V} \frac{|R \cap T_v \cap \{t : \mu_t < 2m\}|}{|T_v \cap \{t : \mu_t < 2m\}|} \right) \left(\limsup_{m\to\infty} \sum_{v\in V} \frac{\lambda_v|T_v \cap \{t : \mu_t < 2m\}|}{m} \right) \\
&\leq \left(\max_{v\in V} \limsup_{m\to\infty} \frac{|R \cap T_v \cap \{t : \mu_t < 2m\}|}{|T_v \cap \{t : \mu_t < 2m\}|} \right) \left(\limsup_{m\to\infty} \frac{2m}{m} \right) \\
&\leq \max_{v\in V} \limsup_{t\to\infty} \frac{|R \cap T_v \cap [t]|}{|T_v \cap [t]|} \cdot 2 \;=\; 2S.
\end{aligned}
$$

We bound the factor multiplied by C' as follows.

$$
\begin{aligned}
\limsup_{m\to\infty} \frac{1}{m} \sum_{v\in V} (\lambda_v|\overline{R} \cap T_v \cap \{t : \mu_t < 2m\}|) &\leq \limsup_{m\to\infty} \sum_{v\in V} \frac{\lambda_v|T_v \cap \{t : \mu_t < 2m\}|}{m} \\
&\leq \limsup_{m\to\infty} \frac{2m}{m} = 2.
\end{aligned}
$$

□

5 Conclusions and Open Problems

We have given a protocol which achieves constant expected delay for each message in the Synchronized Infinitely-Many Users Model with $\lambda < 1/e$. We have also given a protocol which achieves constant expected average delay in the Unsynchronized Finitely-Many Users Model for any $\{\lambda_i\}_{1 \le i \le n}$-Bernoulli message-arrivals distribution in which $\sum_i \lambda_i < 1/e$. Several open questions remain:

- Can we get good delay versus arrival rate tradeoffs in our models? Are there fine-tunings of the protocols or constants which ensure short delays for "small" values of λ?

- In the infinitely-many senders models considered, is there a protocol which is stable in the sense of [13] for all $\lambda < 1$? If not, then what is the supremum of the allowable values for λ, and how can we design a stable protocol for all allowed values of λ? We have shown protocols that guarantee stability for all $\lambda < 1/e$. Here is a heuristic argument as to why this may indeed be a limit. Assume that we have a *static* system with some h users (messages), where even the value of h is known to all users. If all users follow the same protocol, the optimal probability of "success" (exactly one message attempting the channel) in one time step, is achieved if each message attempts using the channel with probability $1/h$: in this case, the success probability is $h \cdot (1/h) \cdot (1-1/h)^{h-1} \sim 1/e$. Thus, even if the users are given the additional information on the exact number of messages, it may be that $1/e$ is the best probability of success possible. This seems to suggest that if the arrival rate λ is more than $1/e$, then the system cannot be stable (since the average arrival rate will be more than the average rate of departure). Is this intuition correct? What is a "minimal" assumption that will ensure a stable protocol for all $\lambda < 1$? (As described in the introduction, some sufficient conditions are described in [20, 13] for certain models including finitely-many users models.)

- For which arrivals distributions are our protocols stable? We have shown that our Unsynchronized Finitely-Many Users Model protocol is stable for any $\{\lambda_i\}_{1 \le i \le n}$-Bernoulli message-arrivals distribution in which $\sum_i \lambda_i < 1/e$, that our Synchronized Finitely-Many Users Model protocol is stable for any $\{\lambda_i\}_{1 \le i \le n}$-dominated arrivals distribution with $\sum_i \lambda_i < 1/e$, and that our Synchronized Infinitely-Many Users Model protocol is stable for Poisson arrivals with $\lambda < 1/e$. We believe that our Synchronized Infinitely-Many Users Model protocol is also stable for other input distributions.

  For example, suppose that the distribution of incoming messages to the system has substantially weaker random properties than the independent Poisson distribution. Our protocol can still achieve $\mathrm{E}[W_{ave}] = \mathrm{O}(1)$. From the paragraph immediately following the statement of Theorem 2.14, we see that $p_i(1) = \mathrm{O}(q^i)$ will suffice to maintain the property that $\mathrm{E}[W_{ave}] = \mathrm{O}(1)$; the strong (doubly exponential) decay of $p_i(1)$ as i increases is unnecessary. In turn, by analyzing the recurrences presented by Lemmas 2.12 and 2.13, we can show that rather than the strong bound of (18), it suffices if

$$\Pr[u_0 \text{ is } t\text{-bad}] \leq k^{-3}(2k^2)^{-t}. \tag{23}$$

We can then proceed to show that $p_i(1) = \mathrm{O}(q^i)$ by showing, via induction on i as above, that $p_i(t) \leq k^{-(i+3)}(2k^2)^{-t}$; the proof can then be concluded as before. The bound in (23) just decays singly exponentially in t, as opposed to the doubly-exponential decay we had for Poisson arrivals. Thus, our approach will work with message-arrival distributions that have substantially weaker tail properties than independent Poisson.

Acknowledgments. We thank Michael Kalantar for explaining the practical side of this problem, Prabhakar Raghavan and Eli Upfal for sending us an early version of their paper [21], and the participants of a seminar at Carnegie-Mellon University, whose questions and comments helped us clarify some points.

References

[1] N. Abramson. The ALOHA system. In N. Abramson and F. Kuo, editors, *Computer-Communication Networks.* Prentice Hall, Englewood Cliffs, New Jersey, 1973.

[2] D. Aldous. Ultimate instability of exponential back-off protocol for acknowledgment based transmission control of random access communication channels. *IEEE Trans. on Information Theory*, IT-33(2):219–223, 1987.

[3] N. Alon, J. H. Spencer, and P. Erdős. *The Probabilistic Method.* Wiley–Interscience Series, John Wiley & Sons, Inc., New York, 1992.

[4] R. J. Anderson and G. L. Miller. Optical communication for pointer based algorithms. Technical Report CRI-88-14, Computer Science Department, University of Southern California, 1988.

[5] B. Bollobás. Martingales, Isoperimetric Inequalities and Random Graphs, in *Combinatorics* (eds A. Hajnal, L. Lovász, and V. T. Sós), *Colloq. Math. Soc. János Bolyai* **52**, pages 113–139, North Holland, 1988.

[6] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[7] M. Dietzfelbinger and F. Meyer auf der Heide. Simple, efficient shared memory simulations. In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, pages 110–119, 1993.

[8] M. Geréb-Graus and T. Tsantilas. Efficient optical communication in parallel computers. In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, pages 41–48, 1992.

[9] L. A. Goldberg, M. Jerrum, F. T. Leighton, and S. B. Rao. A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer. In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, pages 300–309, 1993.

[10] L. A. Goldberg, Y. Matias and S. B. Rao. An Optical Simulation of Shared Memory. In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, pages 257–267, 1994.

[11] J. Goodman, A. G. Greenberg, N. Madras, and P. March. Stability of binary exponential backoff. *J. Assoc. Comput. Mach.*, 35(3):579–602, 1988. A preliminary version appeared in *Proc. ACM Symposium on Theory of Computing*, 1985.

[12] A. G. Greenberg, P. Flajolet and R. E. Ladner. Estimating the multiplicities of conflicts to speed their resolution in multiple access channels. *J. Assoc. Comput. Mach.*, 34(2):289–325, 1987.

[13] J. Håstad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access channels. *SIAM J. on Computing*, 25(4):740–774, 1996. A preliminary version appeared in *Proc. ACM Symposium on Theory of Computing*, 1987.

[14] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

[15] Special issue of *IEEE Trans. on Information Theory*, IT-31, 1985.

[16] F. P. Kelly. Stochastic models of computer communication systems. *J. R. Statist. Soc. B*, 47(3):379–395, 1985.

[17] P. D. MacKenzie, C. G. Plaxton, and R. Rajaraman. On contention resolution protocols and associated probabilistic phenomena. In *Proc. ACM Symposium on Theory of Computing*, pages 153–162, 1994.

[18] C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, London Math. Soc. Lecture Notes Series 141, pages 148–188, Cambridge University Press, 1989.

[19] R. Metcalfe and D. Boggs. Distributed packet switching for local computer networks. *Comm. ACM*, 19:395–404, 1976.

[20] N. Pippenger. Bounds on the performance of protocols for a multiple access broadcast channel. *IEEE Trans. on Information Theory*, IT-27:145–151, 1981.

[21] P. Raghavan and E. Upfal. Stochastic contention resolution with short delays. In *Proc. ACM Symposium on Theory of Computing*, pages 229–237, 1995.

[22] B.S. Tsybakov and N. B. Likhanov, Upper Bound on the Capacity of a Random Multiple-Access System, *Problemy Peredachi Informatsii,* **23(3)** (1987) 64–78.

[23] N. D. Vvedenskaya and M. S. Pinsker. Non-optimality of the part-and-try algorithm. In *Abstracts of the International Workshop on Convolutional Codes, Multiuser Communication, Sochi, USSR*, pages 141–148, 1983.