Research Report 328

Contention Resolution with Guaranteed Constant Expected Delay

Leslie Ann Goldberg and Philip D. MacKenzie

RR328

We study contention resolution in multiple-access channels such as the Ethernet. Under a stochastic model of continuous packet generation from a set of n processors, we construct a protocol which guarantees constant expected delay for generation rates up to a fixed constant which is less than 1. Previous protocols which are stable for constant arrival rates do not guarantee constant expected delay. The two protocols that achieved results closest to this are one by Raghavan and Upfal, which only guarantees logarithmic (in n) expected delay, and one by Paterson and Srinivasan, which only guarantees constant expected delay with high probability. (In the latter protocol, there is a non-zero probability that the initial clock synchronization might fail and cause the expected delay to grow unboundedly.) Although those protocols do not guarantee constant expected delay, we have used ideas from them in the construction of our protocol, which does guarantee constant expected delay. We achieve our results using a technique called *Robust Synchronization* which is applied periodically in our protocol. The introduction of this technique and the analysis of this technique are the major contributions of the paper.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

July 1997

Contention Resolution with Guaranteed Constant Expected Delay*

Leslie Ann Goldberg[†] Philip D. MacKenzie[‡]

July 21, 1997

Abstract

We study contention resolution in multiple-access channels such as the Ethernet. Under a stochastic model of continuous packet generation from a set of n processors, we construct a protocol which guarantees constant expected delay for generation rates up to a fixed constant $\lambda_0 < 1$. Previous protocols which are stable for constant arrival rates do not guarantee constant expected delay. The two protocols that achieved results closest to this are one by Raghavan and Upfal, which only guarantees logarithmic (in n) expected delay, and one by Paterson and Srinivasan, which only guarantees constant expected delay with high probability. (In the latter protocol, there is a non-zero probability that the initial clock synchronization might fail and cause the expected delay to grow unboundedly.) Although those protocols do not guarantee constant expected delay, we have used ideas from them in the construction of our protocol, which does guarantee constant expected delay. We achieve our results using a technique called *Robust Synchronization* which is applied periodically in our protocol. The introduction of this technique and the analysis of this technique are the major contributions of the paper.

1 Introduction

There has been an enormous amount of work on contention resolution for Ethernet-like multiple-access channels (see, for example, [3, 4], and the references therein). Most of this work has focused on backoff protocols, including both constant backoff protocols (as in the Aloha network [1]) and increasing backoff protocols (as in the Ethernet network [6]). Backoff protocols work very well in practice and are therefore worthy of study. However, it has been shown that the expected packet delay of backoff protocols with constant generation rate is $\Omega(n)$, where n is the number of processors. (Håstad, Leighton, and Rogoff [4] show this for exponential, polynomial, and constant backoff protocols, but the same lower bound technique would apply to other backoff functions.)

*Research Report RR328, Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom. An extended abstract of this report will appear in the Proceedings of the 38th Symposium on Foundations of Computer Science, 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. Dept. 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.

Raghavan and Upfal [8] were the first develop a contention-resolution protocol with $o(n)$ expected packet delay for a constant generation rate. Specifically, their protocol allows a generation rate up to about 1/10, while achieving expected delay $O(\log n)$. A key idea that allows the low expected delay is that of a "Reset State" which is entered when bad situations are detected. Later, Paterson and Srinivasan [7] achieved constant expected delay for an infinite number of processors, assuming that the processors have synchronized clocks, for generation rates up to about $1/e$. Using their infinite-processor protocol, Paterson and Srinivasan designed a protocol that can be used by n processors whose clocks differ by at most B (for some value B) and, with high probability, achieves constant expected packet delay for generation rates up to about $1/e$. This protocol starts with a "Pre-processing Phase" which takes $\Omega(\max\{B, n\})$ time and succeeds with high probability. If the pre-processing phase fails, then the expected packet delay of the protocol can grow without bound. Thus, the overall expected packet delay of the protocol is unbounded.

In this paper we affirmatively answer the question of the existence of a contention-resolution protocol with constant expected packet delay in the standard multiple-access channel model, i.e., we present the first protocol that achieves constant expected packet delay for generation rates up to a fixed constant $\lambda_0 < 1$, without the added requirement that processors be started with identical clocks. In fact, our protocol allows processors to start and stop repeatedly (with some constraints) with no a priori clock values, and it still achieves constant expected packet delay.

Our protocol uses the protocol of Paterson and Srinivasan as a subroutine. The analysis of this subroutine requires processors to have synchronized clocks, and we are able to simulate this (for reasonably long periods of time) using a new technique for *Robust Synchronization*. In particular, our protocol incorporates repeated, robust "Synchronization Phases". Each "Synchronization Phase" is similar to the "Preprocessing Phase" of Paterson and Srinivasan, except that our synchronization phases are performed repeatedly and are robust in the sense that they do not require processors to have similar clocks, and they can handle any pathological situation (such as processors starting in the middle of a synchronization phase) and eventually return to a normal, synchronized state.

In some sense, the structure of our protocol (normal phases, occasionally interrupted by synchronization phases) is similar to the structure of Raghavan and Upfal's protocol (which has normal phases, occasionally interrupted by reset phases). 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 processors in a reset state are only allowed to send packets with very low probability, and this helps other processors to access the channel. However, our synchronization phase is designed to accomplish the much-more-difficult task of synchronizing the processors (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 processors. Thus, synchronization phases are costly in our protocol. A third difference is that in Raghavan and Upfal's protocol, a normal phase always tends towards low expected delay, and when bad situations arise, there is a good probability of them being caught, and thus causing a reset state to occur. In our protocol, a normal phase tends towards even lower (constant) expected delay if the processors 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 processors are unsynchronized. Thus, the bad situation can only be remedied during the next time the processors start a synchronizing phase, which may be after quite a long time! Fortunately, we are able to bound the effects of this type of behavior and still achieve constant expected packet delay.

1.1 Model and Definitions

Following all of the papers that we mention in Section 1.2, we work in a *time-slotted* model in which time is partitioned into intervals of equal length, called *steps*. In our basic model, we have n processors which can start and stop at arbitrary steps, with the constraint that each time a processor starts, it runs for at least a certain polynomial number of steps.[1]

Packets are generated at the processors according to a probability distribution. Under the $\{\lambda_i\}_{1\leq i\leq n}$-*independent* arrivals distribution, each processor i is associated with a positive probability λ_i and it generates a message independently with probability λ_i during each time step that it is running. For technical reasons, we also consider other arrival distributions. We say that an arrival distribution is $\{\lambda_i\}_{1\leq i\leq n}$-*dominated* (for $\lambda_1,\ldots,\lambda_n > 0$) if the total arrival rate, $\lambda = \sum_i \lambda_i$, is at most ϵ for a sufficiently small positive constant ϵ, and the following condition is satisfied: For every processor i, every step t in which processor i is running, and every event E concerning the arrival of messages at steps other than t or at processors other than i, the probability that processor i generates a message at step t, conditioned on event E, is at most λ_i.

The packets which arrive at the processors must be transmitted to the multiple access channel which handles contention as follows: when multiple processors attempt to transmit to the channel at the same time, none succeed. If a single processor attempts to transmit to the channel, it receives an acknowledgement that the transmission was successful. Processors must queue all unsuccessful packets for retransmission, and they use a contention-resolution protocol to decide when to retransmit. During the time that a processor is trying to send one packet, it may generate more packets that it needs to transmit. These packets must also be queued. An important feature of a good contention-resolution protocol is that, even when packets are generated fairly frequently, the sizes of the queues do not grow unboundedly.

We say a protocol is *stable* if the expected size of the queues remains bounded. More formally, let W_i be the waiting time of the ith packet, and let

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

(Intuitively, W_{avg} is the average waiting time of packets in the system.) We say that a protocol is stable if $\mathrm{E}[W_{\text{avg}}]$ is finite. For stable protocols, we are concerned with the maximum generation rate λ that can be stably sustained, and with the value of $\mathrm{E}[W_{\text{avg}}]$.

1.2 Previous Work

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

[1]For the constant expected packet delay results in Section 4, we require this polynomial to be $8n^{71}$; however, n^{33} is sufficient for all proofs in Section 3. Note that no attempt has been made to optimize these polynomials.

Aldous [2] showed that for any $\lambda > 0$, binary exponential backoff is unstable when the number of processors is infinite. Kelly [5] showed a similar result for polynomial backoff protocols. Håstad, Leighton, and Rogoff [4] showed that polynomial backoff is stable (with an expected delay of $\omega(n)$) for any $\lambda < 1$ when the number of processors is finite, and that binary exponential backoff is unstable for $\lambda > .567$, when the number of processors is finite. Raghavan and Upfal [8] give a protocol which, for $\lambda < 1/10$, is stable for a finite number of processors, n, and achieves $O(\log n)$ expected packet delay. Their protocol has the added benefit of being a fairly simple, clean protocol. Paterson and Srinivasan [7] give a very clever protocol which, for $\lambda < 1/e$, is stable for an infinite number of processors with synchronized clocks, or stable *with high probability* for a finite number of processors with almost synchronized clocks, and achieves constant expected packet delay, again *with high probability*. (With some small probability, the delay of their protocol is unbounded.)

1.3 Our Results

We present a protocol which, for a small constant λ, is stable for any finite number of processors and *guarantees* constant expected packet delay. In our model, we do not make any assumptions about a previously synchronized or almost synchronized clock. (Note that the lambda used in our protocol is very small, a few orders of magnitude smaller than the lambda used in previous results. We do not claim that our protocol is a practical protocol. The significance of our protocol is two-fold: (1) it is the first protocol with guaranteed constant expected delay, and (2) the ideas behind it, mainly the *Robust Synchronization* technique, could be useful in other practical protocols.)

The structure of our protocol is simple. Most of the time, the processors are running the "infinite processors" protocol of Paterson and Srinivasan. The analysis of that protocol assumes that processors have a synchronized clock. Therefore, in our protocol, the processors occasionally enter a synchronizing phase to make sure that the clocks are synchronized (or to resynchronize after a processor 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 packet delay.

The synchronizing phase of our protocol is somewhat complicated, because it must synchronize the processors even though interprocessor communication can only be performed through acknowledgements (or lack of acknowledgements) from the multiple-access channel. (This is the model used in all previous papers.) The analysis of our protocol is also complicated due to the very dynamic nature of the protocol, with possibilities of processors 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 processors enter a synchronization phase, and halfway through it, another set L' of processors start up. Since the processors in L' have missed a large part of the synchronization phase, they will not be able to synchronize with the other processors. (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 processors in L detect the newly started processors during the synchronization phase. Then they must somehow resynchronize with the newly joined processors. However, any synchronization protocol must perform various tasks (such as electing a leader) and it is difficult to detect the presence of the processors in L' during some of these tasks. A second approach is to allow the processors in L to ignore the processors in L' and to finish their synchronization phase (either synchronized amongst themselves, or not). Then the set L' of processors in the synchronization phase will very likely disrupt the normal operations of the processors in L, causing them to synchronize again. But now the processors in L' will be about halfway through their synchronization, whereas the processors in L are just starting synchronization! Our solution to this problem is a combination of the two approaches, and is described in the next section with the description of our protocol.

1.4 Outline

In Section 2 we describe our new protocol. In Section 3 we prove the key features of our protocol, namely, a packet generated at a step in which no processors start or stop soon before or after will have constant expected delay, and a packet generated at a step in which a processor starts soon before or after will have an expected delay of $O(n^{37})$ steps. In Section 4 we show that our protocol achieves constant expected packet delay for a fairly general multiple access channel model, with processors starting and stopping.

2 The Protocol

Let

$$S = \{0, \ldots, n^{40} - 1\} \setminus \{n^2 - 1, 2n^2 - 1, 3n^2 - 1, \ldots, n^{40} - 1\},$$

be a set of steps, namely, the first n^{40} steps, except for the last one of every n^2 steps. Then let T be the tree defined as in Paterson and Srinivasan's protocol, except that (1) it is truncated to the first $n^{40} - n^{38}$ leaves, and (2) if a node in their tree T has step j in its trial set, then the corresponding node in our tree T has the jth step of S in its trial set. Let $W = 12n^4$.

Now we give an informal description of our protocol. In the normal state a processor maintains a buffer B of size n^7 and an infinite queue Q. B and Q contain packets to be sent, and when a packet is generated it is put into B. For each packet $m \in B$ the processor maintains a variable trial(m) which contains the next step on which the processor will attempt to send m. The step trial(m) will be chosen using Paterson and Srinivasan's protocol (but modified as above). The steps not in S (i.e., the last step of every n^2 steps) are used to send packets from Q. At each of these steps, with probability $1/(3n)$, the processor attempts to send the first packet in Q. Each processor also maintains a list L which keeps track of the results (either "failure" or "success") of the most recent packet sending attempts from Q, up to n^2 of them.

A processor goes into a synchronizing state if a packet 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 Paterson and Srinivasan's protocol for at least n^{40} steps, and it synchronizes with probability n^{-30} on any given step.) If the processor does go into a synchronizing state, it transfers all packets from B to the end of Q.

In the synchronizing state, a processor 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 packets into the queue. Also, it only sends dummy packets in the synchronizing state. (The dummy packets are used for synchronizing. Real packets that arrive during the synchronization phase must wait until the next normal phase to be sent.) The various synchronization stages are as follows (a processor goes through these stages in order).

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

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

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

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

COPYING_CLOCK In this stage, a processor has decided it is not the leader, and it attempts to copy the leader clock by polling the channel repeatedly to attempt 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 only be left with 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.

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

POLLING A processor 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 processors which might be joining the system and re-starting a synchronization phase. If new processors are detected, the processor re-starts the synchronization phase. Otherwise, it begins the normal phase of the protocol.

The length of each of these stages is very important, both in terms of achieving 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 processors 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 processors 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. Having the lengths of the "preliminary" and "synchronizing" stages as above,
2. Noticing that only the preliminary stages can cause the channel to be jammed,
3. Noticing that the "synchronizing" stages cannot detect a new synchronization occurring,
4. Having the POLLING stage be of length $\Theta(Wn^3)$ (longer than all of the other stages combined), and

5. having the POLLING stage be 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, guarantees that bad events as described at the end of Section 1.3 cannot occur, even when up to n processors are starting at different times.

Whenever a processor joins the Ethernet, 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 packet, $m$, is generated
        Put $m$ in $B$
        Choose trial($m$) from the trial set of the appropriate leaf of $T$
    If ((clock mod $n^2$) = $n^2 - 1$) call Procedure Queue_Step
    Else call Procedure Normal_Step
```

```
Procedure Normal_Step
    If (clock $\geq n^{40}$ or any packet in $B$ has waited more than $n^7$ steps)
        Move all of the packets in $B$ from $B$ to $Q$
        Empty $L$
        state $\leftarrow$ SYNCHRONIZING, sync_stage $\leftarrow$ JAMMING, clock $\leftarrow$ 0
    Else
        With Probability $n^{-30}$
            Move all of the packets in $B$ from $B$ to $Q$
            Empty $L$
            state $\leftarrow$ SYNCHRONIZING, sync_stage $\leftarrow$ JAMMING, clock $\leftarrow$ 0
        Otherwise
            If more than one packet $m$ in $B$ has trial($m$) = clock
                Send a dummy packet
                For each $m \in B$ with trial($m$) = clock
                    Choose a new trial($m$) from the trial set of the appropriate node
                    at the next level of $T$
            If exactly one packet in $B$, $m$, has trial($m$) = clock
                Send $m$
                If $m$ succeeds, remove it from $B$
                Else choose a new trial($m$) from the trial set of the appropriate node
                at the next level of $T$
            clock $\leftarrow$ clock + 1
```

```
Procedure Procedure Queue_Step
    With probability $1/3n$
        If ($Q$ is empty) Send a dummy packet
        Else
            Send the first packet in $Q$
            If the outcome is "success", Remove the packet 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")
        Move all of the packets in $B$ from $B$ to $Q$
        Empty $L$
        state $\leftarrow$ SYNCHRONIZING, sync_stage $\leftarrow$ JAMMING, clock $\leftarrow 0$
    Else clock $\leftarrow$ clock $+ 1$
```

```
Procedure Synchronizing
    If a packet 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 packet
    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 packet
        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 packet
    If (clock < $2W - 1$) clock $\leftarrow$ clock + 1
    Else sync_stage $\leftarrow$ SETTING_CLOCK, clock $\leftarrow$ 0


Procedure Set_Clock
    If (clock = 0 mod $4W$)
        Send a dummy packet
    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 packet
        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 $\leftarrow -j$
            Empty $L$
            If ($j = 0$) sync_stage $\leftarrow$ POLLING
            Else sync_stage $\leftarrow$ WAITING
        Else
            Empty $L$
            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 packet
        Add the outcome of this send to the end of $L$
    Otherwise Add "failure" to $L_2$
    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 $\leftarrow 0$
```

3 The Main Proof

Suppose that n is sufficiently large and that n processors run the protocol. Step 0 will be the step in which the first processor starts the protocol. Processors will start and stop (perhaps repeatedly) at certain predetermined times throughout the protocol. We say that the sequence of times at which processors start and stop is *allowed* if every processor runs for at least n^{33} steps each time it starts. Just before any step, t, we will refer to the processors that are running the protocol as *live* processors. We will say that the state of the system is *normal* if all of these processors 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 processor has clock $= C$, and
3. every processor with $|L| \geq n^2/2$ has a success in the last $n^2/2$ elements of L, and
4. no packet in any processor'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 processor has state = NORMAL, or has state = SYNCHRONIZING with either sync_stage = JAMMING or sync_stage = POLLING, and
- some processor 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 processor with state = SYNCHRONIZING, sync_stage = JAMMING and clock $= 0$ just after step t had its clock $< n^{40}$, had no packet 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 processor if at least $n^{29}/2$ packets are sent from the processor'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 processor if at most n^{17} packets are placed in the processor'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 processors 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" synchronize phase.)
Definition: $T = n^{31}$.

Procedure Normal_Step simulates a slightly modified version Paterson and Srinivasan's protocol from [7]. (We refer the reader to [7] for a full description and analysis of Paterson and Srinivasan's protocol.) We call our slightly modified version PS'. PS' is based on the variant of Paterson and Srinivasan's protocol with $k = 16$, $s = 2$ and $r = 8$ which was described in [7]. In PS', a is defined to be k^{20} (Paterson and Srinivasan use a smaller a.) Paterson and Srinivasan require $\lambda < 1/sa$. However, in PS', we require $4\lambda < 1/sa$. In our variant, b is chosen such that $b > 2/(4\lambda)$. Given our choice of constants, PS' can only handle smaller values of λ_j than those specified in [7]. The main difference between PS' and the protocol discussed in [7] is that after every $n^2 - 1$ steps, PS' waits one step, generating inputs, but not running the Paterson-Srinivasan protocol. (Any inputs that arrive during this step are deemed to have arrived after at the beginning of the following step, when PS' continues simulating the Paterson-Srinivasan protocol.) Note that from any starting state until a synchronization, our system simulates PS'. This implies that our system stops simulating PS' when a processor starts up, since that processor will immediately start a synchronization. Then PS' is simulated again once a starting state is reached. Lemmas 1 and 2 describe the behavior of PS' and are based on a theorem of Paterson and Srinivasan.

Lemma 1 *Suppose that PS' is run with a $\{\lambda_i\}_{1\leq i\leq n}$-dominated arrival distribution for $\tau \leq n^{40}$ steps. Then the expected delay of any packet that arrives is* O(1)*. Furthermore, the probability that any packet has delay more than $n^7/2$ is at most n^{-60}.*

Proof: The lemma follows from the proof of Theorem 2 of [7]. Note that an extra factor of 2 in the definition of λ allows the theorem to apply to PS' (with the extra step every n^2 steps). □

Lemma 2 *Suppose that PS' is run with a $\{\lambda_i\}_{1\leq i\leq n}$-dominated arrival distribution for $\tau \leq n^{40}$ steps and that a packet arrives at processor p at step $t' \leq \tau$. Then the expected delay of any packet that arrives is* O(1)*. Furthermore, the probability that any packet has delay more than $n^7/2$ is at most n^{-60}.*

Proof: Another extra factor of 2 in the definition of λ and the constraint on b ensure that the expected number of packets in a node of the tree at level 0 (even with the extra packet) can be handled by the protocol. Thus, this Lemma follows from Lemma 1. □

Lemma 3 *Given a random variable X taking on non-negative integer values, and two events A and B over the same space,* $\mathrm{E}[X \mid A \wedge B] \leq \mathrm{E}[X \mid B] / \Pr(A \mid B)$.

Proof: Note for any event C that $\Pr(C|B) \geq \Pr(C \wedge A|B) = \Pr(C|A \wedge B)\Pr(A|B)$. From this we see that

$$\begin{aligned} \mathrm{E}[X|A \wedge B] &= \sum_{r \geq 0} r \Pr[X = r|A \wedge B] \\ &\leq \sum_{r \geq 0} r \Pr[X = r|B] / \Pr[A|B] \\ &\leq \mathrm{E}[X|B] / \Pr[A|B]. \end{aligned}$$

□

Lemmas 4 to 8 outline the analysis of the normal operation of the synchronization phase of our protocol.

Lemma 4 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor starts or stops between steps t and $t+W$. If the system is in a synchronizing state just before step t, then every live processor sets* sync_stage *to* FINDING_LEADER *just before some step in the range $\{t, \ldots, t+W\}$.*

Proof: First note a processor can have state = SYNCHRONIZING and sync_stage = JAMMING for only $W/2$ steps. Then note that every processor 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 processor 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 processor 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 processor remains in sync_stage = JAMMING for $W/2$ steps. □

Lemma 5 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processors start or stop between steps t and $t + 4W$. If every processor sets* sync_stage = FINDING_LEADER *before some step in the range $\{t, \ldots, t+W\}$, then with probability at least $1 - e^{-n^3}$, exactly one processor sets* sync_stage = SETTING_CLOCK *just before some step in the range $t + 2W + 1, \ldots, t + 4W$ and every other processor sets* sync_stage = COPYING_CLOCK *just before some step in the range $t + W, \ldots, t + 2W$.*

Proof: First note that at most one leader is elected since after elected, it does not allow any processors to access the channel for $2W$ steps. Also note that no processor 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 processor 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 the other processors will directly set sync_stage to COPYING_CLOCK. □

Lemma 6 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor starts or stops between steps $\tau - 3W$ and $\tau + 20Wn^2$. If exactly one processor sets* sync_stage = SETTING_CLOCK *just before step* $\tau \in \{t + 2W, \ldots, t + 4W\}$ *and every other processor sets* sync_stage = COPYING_CLOCK *just before some step in the range* $\tau - 3W, \ldots, \tau$, *then with probability at least* $1 - 4Wne^{-n}$. *all processors set* sync_stage = POLLING *with* clock = 0 *just before step* $\tau + 20Wn^2$.

Proof: The statement in the lemma is clearly true for the processor that sets sync_stage = SETTING_CLOCK. Suppose that P is some other processor. 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, $5n^2 - 1 \leq \lfloor (20Wn^2 - 3W)/(4W) \rfloor$. 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 processors and moduli of $4W$. □

Lemma 7 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processors start or stop between steps τ and $\tau + Wn^3$. If all processors set* sync_stage = POLLING *with* clock = 0 *just before step τ then with probability at least* $1 - Wn^4e^{-n/10}$ *all processors set* state = NORMAL *and* clock = 0 *just before step* $\tau + Wn^3$.

Proof: Say a sequence of $n^2/2$ steps is *bad* for processor P if P does not have a successful transmission on any step in the sequence. Then the probability that a given processor 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 processors still have sync_stage = POLLING. This is at most the probability that it either doesn't send, or is blocked on each step of the sequence, which is

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

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

Lemma 8 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor 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 the range $\{t + 12n^7, \ldots, t + 13n^7\}$ such that it is in the starting state just before step t'.*

Proof: The lemma follows from Lemmas 4, 5, 6 and 7. □

Lemmas 9 to 14 outline the analysis of the robustness of the synchronization phase. Lemma 9 shows that no matter what state the system is in (i.e., possibly normal, possibly in the middle of a synchronization), if some processor starts a synchronization (possibly because it just started), then within $W/2$ steps, every processor will be in an early part of the synchronization phase. Then Lemma 10 shows that with high probability, within a reasonable amount of time, all processors 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 11 shows that soon all processors will be in the polling stage. At this point, as shown in Lemma 12, 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 8.

Note that these lemmas require the assumption that no processors 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 processor just having started in the middle of a synchronization phase. If another processor 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 processor started.

Lemma 9 *If the protocol is run and some processor sets* sync_stage = JAMMING *just before step t, and that processor does not stop for $W/2$ steps, then there is a t' in the range $t, \ldots, t+(W/2)$ such that just before step t', no processor has* state = NORMAL, *and every processor that has* sync_stage = POLLING *has* clock $\leq W/2$.

Proof: Every processor 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 the range $t+1, \ldots, t+(W/2)$. The Lemma follows. □

Lemma 10 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor starts or stops between steps t and $t+5nW$. If, just before step t, no processor has* state = NORMAL *and every processor with* sync_stage = POLLING *has* clock $\leq W/2$, *then, with probability at least $1-5Wn^2e^{-n/10}$, there is a t' in the range $t, \ldots, t+5nW$ such that just before step t', each processor has* state = SYNCHRONIZING *with* sync_stage *set to* SETTING_CLOCK, COPYING_CLOCK, WAITING, *or* POLLING. *Furthermore, if a processor 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 processor 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 processor 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 processor 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 processor 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 processors. The probability that a processor 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 processor, 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 processor 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 processor. In particular, the sequence of $n^2/2$ steps preceding t' is not bad for any processor, so any processor 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 11 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor starts or stops between steps t and $t + 5nW + (W/2) + 20Wn^2$. If some processor sets* sync_stage = JAMMING *just before step t then with probability at least $1 - 21Wn^3e^{-n/10}$, there is a t' in the range $t, \ldots, t + 5nW + (W/2) + 20Wn^2$ such that just before step t', each processor has* sync_stage = POLLING.

Proof: We know by Lemmas 9 and 10 that, with probability at least $1 - 5Wn^2e^{-n/10}$, there is a τ in the range $t, \ldots, t + 5nW + (W/2)$, such that just before step τ, each processor has state = SYNCHRONIZING and sync_stage set to SETTING_CLOCK, COPYING_CLOCK, WAITING, or POLLING. Furthermore, if a processor has sync_stage = POLLING, it has a 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.

Note now that unless a processor sets sync_stage = JAMMING in the next $20Wn^2$ steps, there will be a step t' such that each processor has sync_stage = POLLING. But to set sync_stage = JAMMING, a processor with sync_stage = POLLING must be unsuccessful in all transmission attempts during some $n^2/2$ consecutive steps. For a single processor 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 7). For all processors 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 12 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor starts or stops between steps t and $t + Wn^3 + 13n^7$. If the system is in a state in which every processor 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 the range $t, \ldots, t + Wn^3 + 13n^7$ such that the system is in a normal state just before step t'.*

Proof: If no processor 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 processor sets sync_stage = JAMMING just before step $t'' \leq t + Wn^3 - 1$. By Lemma 8, with probability at least $1 - 2Wn^4e^{-n/10}$ the system will enter a starting state by step $t'' + 13n^7$.
□

Observation 13 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor starts between steps t and $t + 21Wn^2 - 1$. Suppose that no processor sets* sync_stage = JAMMING *during steps $t, \ldots, t + 21Wn^2 - 1$. Then every processor has* state = NORMAL *or* sync_stage = POLLING *just before step $t + 21Wn^2$.*

Lemma 14 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor 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 the range $\{t, \ldots, t + n^8\}$ such that the system is in a normal state just before step t'.*

Proof: The lemma follows from Lemma 12, Observation 13 and Lemma 11. □

Lemma 15 through Theorem 19 show that if the protocol is run with a $\{\lambda_i\}_{1 \leq i \leq n}$-dominated packet arrivals distribution then the system is usually in a good state, (i.e., synchronized and running the PS' protocol), and thus the expected time that packets wait in the buffer is constant.

Lemma 15 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor 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 the range $\{t, \ldots, t + n^{31}/4\}$ such that the system is in a starting state just before step t'.*

Proof: By Lemma 14, 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 doesn't enter a synchronizing state within $n^{31}/8$ steps is at most $(1 - n^{-30})^{(n^{31}/8)-(n^{29}/8)} \leq e^{-n/10}$. Then by Lemma 8, once it enters a synchronizing state, with probability $1 - 2Wn^4e^{-n/10}$ it will be in a starting state within $13n^7$ steps. The lemma follows directly from summing failure probabilities. □

Lemma 16 *Suppose that the protocol is run with a sequence of processor start/stop times in which no processor 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 the range $\{t, \ldots, t + n^{31} - 2n^8\}$ such that the system is in a synchronizing state just before step t'.*

Proof: From Lemma 14, 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 processor will switch to a synchronizing state. The probability of this not happening in the next $n^{31} - 3n^8$ steps is at most $(1 - n^{30})^{(n^{31}-3n^8-n^{29})} \leq e^{-n/2}$. The lemma follows from summing the failure probabilities. □

Lemma 17 *Let τ be a non-negative integer less than $n^{40} - n^7$. Suppose that the protocol is run with a $\{\lambda_i\}_{1\leq i\leq n}$-dominated arrival distribution and a sequence of processor start/stop times in which no processor 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 Figure 1, in which the protocol is started in a starting state just before step t and run according to the experiment. 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 6, the probability that a given element of L is "success" is at least $2/(9n)$, so the probability that FAIL1 occurs is at most $\tau ne^{-n/9}$. By Lemma 1, and the fact that at most n^{40}/W starting states occur in the experiment (so PS' 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 processors never reach n^{40}. If the state is normal, all processors have the same value of c, every processor with $|L| \geq n^2/2$ has a success in the last $n^2/2$ elements of L, and every processor has no packet that has waited more than $n^7/2$ steps, then the probability that a given processor 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 8, 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 $2\tau Wn^4e^{-n/10}$. □

Lemma 18 ***Suppose that the protocol is run with a $\{\lambda_i\}_{1\leq i\leq n}$-dominated arrival distribution and a sequence of processor start/stop times in which no processor starts or stops between***

```
$i \leftarrow t$
resyncing $\leftarrow$ false
Do forever
    Simulate a step of the protocol
    If (resyncing $=$ false)
      If some packet has waited more than $n^7/2$ steps
        FAIL2
      If some processor 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
```

Figure 1: Experiment for proof of Lemma 17

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 16, Lemma 8, and Lemma 17. $\square$

Theorem 19 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution and a sequence of processor start/stop times in which no processors start or stop during steps $[t-T,\ldots,t+n^7]$. Given any system state just before step $t-T$, suppose that a packet is generated at step t. The expected time that the packet spends in the buffer is* O(1).

Proof: Let X be the time that the packet 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. Note that X is always at most n^7. Thus, $\mathrm{E}[X] \le n^7 \Pr[\overline{G}] + \mathrm{E}[X|G]$. 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 15, the latter probability is at most $6Wn^4e^{-n/10}$, and, by Lemma 18, the former probability is at most $14n^{-22}$. Thus, $\mathrm{E}[X] \le O(1) + \mathrm{E}[X|G]$. 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 packet p' is born in step t' of the PS' protocol. Let B be the event that prior to that step t' (in the PS' protocol), no packet has waited more than n^7 steps, and at step t' no packet 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 PS'). 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 3, $\mathrm{E}[X|G_{t'}] \le \mathrm{E}[Y|A_{t'} \wedge B] \le \mathrm{E}[Y|A_{t'}]/\Pr[B|A_{t'}]$. Then by Lemma 2, $\mathrm{E}[X|G_{t'}] \le 2\mathrm{E}[Y|A_{t'}] \le O(1)$ for all $t' < T$. Then $\mathrm{E}[X|G] = O(1)$. $\square$

Lemma 20 through Theorem 33 show that the probability of a packet entering a queue is low, the probability of a queue being very full is low, and the rate at which the packets are sent from the queue is high enough so that the expected time any given packet spends in the queue is low. (Note that most packets will spend no time in the queue.)

Lemma 20 *Suppose that the system state just before step t is given in a particular run of the protocol. With probability at least $1-6Wn^4e^{-n/10}$, there is a t' in the range $\{t,\ldots,t+n^{32}\}$ such that the system is in a starting 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 processors 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 processors start or stop. Then use Lemma 15. $\square$

Lemma 21 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution and a given allowed sequence of processor start/stop times in which no processor starts or stops between steps $t-T$ and $t+n^7/2$. Given any system state just before step $t-T$, suppose that a packet R arrives at processor 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 18 and Lemma 15,

$\Pr[X] \leq 1\Pr[\overline{G}] + \Pr[X|G] \leq 14n^{-22} + 6Wn^4e^{-n/10} + \Pr[X|G]$. 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 experiment in Figure 2. This experiment models the system beginning at a start state, and going for $t' + n^7/2 \leq T + n^7/2$ steps, but assuming that there are no arbitrary synchronizations, and that there is a packet 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 packet 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 packet 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 17 (using Lemma 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 $\Pr[A \mid S] \leq 2n^{-60}$.

Note also that $\Pr[B \mid S] \leq \Pr[B]/\Pr[S]$. By Lemma 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 packets 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. □

Lemma 22 *Let i be an integer in $[0, \ldots, 14]$. Suppose that the protocol is run with a $\{\lambda_i\}_{1\leq i\leq n}$-dominated arrival distribution and a sequence of processor start/stop times in which no processors start or stop during steps $t, \ldots, t + n^{14+i} - 1$. If the system is in a starting state just before step t then the probability that the system enters a synchronizing state during steps $t, \ldots, t + n^{14+i} - 1$ is at most $2n^{-15+i}$.*

Proof: The probability that an arbitrary synchronization occurs during steps $t, \ldots, t + n^{14+i} - 1$ is at most $n \cdot n^{-30} \cdot n^{14+i} = n^{-15+i}$. Following the proof of Lemma 17, we see that the probability that a non-arbitrary synchronization occurs during these steps is at most $n^{-60} + n^{15+i}e^{-n/9}$. (The probability that a packet waits in a buffer more than n^7 steps is at most n^{-60} by Lemma 1 and the probability that some processor gets n^2 failures on L is at most $n^{14+i} \cdot n \cdot e^{-n/9}$.) □

Lemma 23 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\leq i\leq n}$-dominated arrival distribution and a sequence of processor start/stop times in which no processor 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 $3Wn^{11}e^{-n/10}$.*

```
$i \leftarrow 0$
Do forever
    If $i = t'$
        Add a packet $R$ to processor $P$
        Simulate a step of the protocol (except for the arbitrary synchronizations)
        If some packet has been in a buffer more than $n^7/2$ steps
            FAIL1
        If some processor 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 packet has waited more than $n^7$ steps
            FAIL1
        If $(i > t')$ and some packet has waited more than $n^7$ steps
            FAIL3
        If some processor 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 packet $Q$ has been sent, SUCCEED
        Else FAIL2
```

Figure 2: Experiment for proof of Lemma 21

Proof: If the system is in a starting state $x > n^7$ times, then at least $x - n^7/2$ must be followed by fewer than $2n^{26}$ steps before the next synchronization phase. By Lemma 22, the probability of fewer than $2n^{26}$ steps occurring between a starting state and the next synchronization phases is at most $2n^{-3}$. Thus, the probability of this happening after at least $x - n^7/2$ of the x starting states is at most $2^x(2n^{-3})^{x-n^7/2}$ 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 8, 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 $n^7(2Wn^4e^{-n/10})$ □

Lemma 24 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution, a given allowed sequence of processor 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 processor starts or stops. Then $|S| \le 2n+1$. 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 14, 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 processors start or stop during any subinterval. We perform the following analysis for each of these subintervals.

By Lemma 8, 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 23 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

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

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. All together, at most $27n^{11}$ blocks of size n^4 will contain non-normal steps. □

Corollary 25 *Let x be an integer in the range $0 \le x \le n^{29} - 54n^{11}$. Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution, a given allowed sequence of processor 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 packets are delivered from it during this period is at most $7Wn^{12}e^{-n/10}$.*

Proof: Divide the next $xn^4 + 54n^{15} \le n^{33}$ steps into blocks of size n^4. By Lemma 24, 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 packet must have been sent from the queue during block i. $\square$

Lemma 26 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution, a given allowed sequence of processor 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 processor is at least $1 - 8Wn^{12}e^{-n/10}$.*

Proof: As in the proof of Lemma 24, 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 $n^8 + n^7$ packets to the queue (including the at most n^7 that could be transferred from the processor's buffer). If this were the case, at most $3n^{16}$ packets would be placed in the queue during the interval. $\square$

Lemma 27 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution, a given allowed sequence of processor 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 processor is at least $1 - 7Wn^{12}e^{-n/10}$.*

Proof: Follows from Corollary 25. $\square$

Lemma 28 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution and a given allowed sequence of processor start/stop times before step t. The probability that more than $n^{17} + i(n^{33}+n^7)$ packets are in a queue just before step t is at most $e^{-in/30}$ for $i \ge 1$ and at most $e^{-n/30}$ for $i = 0$.*

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–$(k-1)$. We say that interval j is "bad" if it is not both productive and light for the processor. 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 $n^{29}/4 - n^{17}$ during interval k. Thus, if b of intervals 0–k are bad, then the size of the queue just before step t is at most

$$(k+1)(n^{33}+n^7) - (k+1-b)(n^{33}+n^7+n^{29}/4-n^{17}) + n^{17}.$$

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 $n^{17} + i(n^{33}+n^7)$ packets 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–k are bad. By Lemmas 26 and 27, 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 $i \ge 1$, the failure probability is at most

$$\sum_{k\ge 0}\binom{k}{\lfloor (i/2)+(k/(8n^4))\rfloor + 1} X^{\lfloor (i/2)+(k/(8n^4))\rfloor+1} \le \sum_{k\ge 0}(16en^4X)^{\lfloor (i/2)+(k/(8n^4))\rfloor+1}$$

$$
\begin{aligned}
&\le \sum_{k\ge 0}(16en^4X)^{(i/2)+(k/(8n^4))}\\
&\le (16en^4X)^{i/2}\sum_{k\ge 0}(16en^4X)^{k/(8n^4)}\\
&\le (16en^4X)^{i/2}8n^4\sum_{k\ge 0}(16en^4X)^{k}\\
&\le 2(8n^4)(16en^4X)^{i/2}\\
&\le e^{-in/30}.
\end{aligned}
$$

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

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

□

Lemma 29 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution and a given allowed sequence of processor start/stop times after step $t+n^{32}$. Suppose that no processors 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 15, 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 Lemma 23, 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 15, 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 30 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\le i\le n}$-dominated arrival distribution, a given allowed sequence of processor start/stop times after step t, and a given system state just before step t in which queue Q contains at least x packets. Then the expected time until at least x packets have been sent from Q is $O(xn^4+n^{15})$.*

Proof: Our first case is when $x \le n^{29}/2$. Let A be the event that at least x packets 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 packets. Using Corollary 25 and

Lemma 27,

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

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 packets will be sent. Let D_k be the event that intervals 1–k don't contain at least r productive intervals, but that intervals 1–$(k+1)$ do contain r productive intervals.

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

□

Theorem 31 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\leq i\leq n}$-dominated arrival distribution, a given allowed sequence of processor start/stop times in which no processors start or stop during steps $[t - n^{33}, \ldots, t + n^{33}]$. Suppose that a packet is generated at step t. The expected time that the packet spends in the queue is* 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 $i \geq 1$, 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 packet enters the queue. Let t' be the random variable denoting the smallest integer such that $t' \geq t$ 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 $t'' \geq t$ 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 packet spends in the queue. All probabilities in this proof will be conditioned on the state of the fact that no processors start or stop during steps $[t - n^{33}, \ldots, t + n^{33}]$.

We start by computing $\sum_{i\geq 1} E(X \mid A_i \wedge C)\Pr(A_i \wedge C)$. By Lemma 28, $\Pr(A_i) \leq e^{-(\max\{i-1,1\})n/30}$ so $\Pr(A_i \wedge C) \leq e^{-(\max\{i-1,1\})n/30}$. By Lemma 30, $E(X \mid A_i \wedge C)$ is at most

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

(Since A_i holds, there are at most $n^{17} + i(n^{33} + n^7)$ packets in the queue before interval I_1 and at most $n^{33} + n^7$ get added during interval I_1.) By Lemma 20, $E(t' - t \mid A_i \wedge C)$ is at most $\sum_{j\geq 1} n^{32}(6Wn^4e^{-n/10})^{j-1} = O(n^{32})$. Thus, $E(X \mid A_i \wedge C) = (i+1)O(n^{37})$. Thus, $\sum_{i\geq 1} E(X \mid A_i \wedge C)\Pr(A_i \wedge C) \leq \sum_{i\geq 1} e^{-(\max\{i-1,1\})n/30}(i+1)O(n^{37}) = O(1)$.

We now compute $E(X \mid A_0\wedge\overline{B}\wedge C)\Pr(A_0\wedge\overline{B}\wedge C)$. By Lemma 26, the probability of $\overline{B}$ is at most $8Wn^{12}e^{-n/10}$, so $\Pr(A_0\wedge\overline{B}\wedge C) \leq 8Wn^{12}e^{-n/10}$. As above, $E(X \mid A_0\wedge\overline{B}\wedge C) = O(n^{37})$, so $E(X \mid A_0 \wedge \overline{B} \wedge C)\Pr(A_0 \wedge \overline{B} \wedge C) \leq (8Wn^{12}e^{-n/10})O(n^{37}) = O(1)$.

Next, we compute $E(X \mid A_0 \wedge \overline{F} \wedge C)\Pr(A_0 \wedge \overline{F} \wedge C)$. By Lemma 29, the probability of $\overline{F}$ is at most $4Wn^{11}e^{-n/10}$, so $\Pr(A_0 \wedge \overline{F} \wedge C) \leq 4Wn^{11}e^{-n/10}$. As above, $E(X \mid A_0 \wedge \overline{F} \wedge C)$ is at most $E(t'-t \mid A_0 \wedge \overline{F} \wedge C) + O(n^{37})$. 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 20, the expected time from this out-of-sync step until a starting state occurs is at most $\sum_{j\geq 1} n^{32}(6Wn^4e^{-n/10})^{j-1} = O(n^{32})$. Thus, $E(t'-t \mid A_0 \wedge \overline{F} \wedge C) = O(n^{32})$ and $E(X \mid A_0 \wedge \overline{F} \wedge C) = O(n^{37})$. Thus, $E(X \mid A_0 \wedge \overline{F} \wedge C)\Pr(A_0 \wedge \overline{F} \wedge C) \leq (4Wn^{11}e^{-n/10})O(n^{37}) = O(1)$.

Finally, we compute $E(X \mid A_0 \wedge B \wedge F \wedge C)\Pr(A_0 \wedge B \wedge F \wedge C)$. By Lemma 21, the probability of C is at most $16n^{-22}$, so $\Pr(A_0 \wedge B \wedge F \wedge C) \leq 16n^{-22}$. We now wish to bound $E(X \mid A_0 \wedge B \wedge F \wedge C)$. 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 $2n^{17}$ processors 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 the range $t,\ldots,t+2n^{21}$. Then since no state in $t,\ldots,t+2n^{21}+13n^7$ is out-of-sync, $t' \leq t+2n^{21}+13n7$. Finally, suppose that $t' \leq t+2n^{21}+13n7$. By Lemma 30, $E(X \mid A_0\wedge B\wedge C\wedge F)$ is at most $t'-t+O(n^4\cdot 2n^{17}) = O(n^{21})$. Thus, $E(X \mid A_0 \wedge B \wedge F \wedge C)\Pr(A_0 \wedge B \wedge F \wedge C) \leq 16n^{-22}O(n^{21}) = O(1)$. □

Observation 32 *When the protocol is run, every packet spends at most n^7 steps in the buffer.*

Theorem 33 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\leq i\leq n}$-dominated arrival distribution, a given allowed sequence of processor start/stop times. Suppose that a packet is generated at step t. Then the expected time that the packet spends in the queue is $O(n^{37})$.*

Proof: Let X be the random variable denoting the size of the queue just before step t. By Lemma 28, 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 30 shows that the expected time to send the packet is $O(Xn^4 + n^{15})$. Thus, the overall expected time to send the packet is

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

□

4 Results

In this section, we need the following definitions. For any k, let $[k]$ denote the set $\{0,\ldots,k\}$. Let V be the set of processors. For $v \in V$, let T_v be the set of steps in which processor v is running.

Theorem 34 *Suppose that the protocol is run with a $\{\lambda_i\}_{1\leq i\leq n}$-independent arrival distribution and a given sequence of processor start/stop times in which each processor runs for at least $8n^{71}$ steps every time it starts. Then* $\mathrm{E}[W_{avg}] = \mathrm{O}(1)$.

Proof: First note that the sequence of processor start/stop times is allowed. Let T be the set of steps within n^{33} steps of the time that a processor starts or stops. Lemma 35 proves that if the $\{\lambda_i\}_{1\leq i\leq n}$-independent arrival distribution is conditioned on having at most m packets 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 36 with (from Theorem 19 and Theorem 31) $C' = O(1)$ and (from Theorem 33 and Observation 32) $C = O(n^{37})$. From the condition given in the statement of this theorem, we can see that

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

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

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

$(1 \leq i \leq n-1)$. Then $|T| = 8(n-1)n^{33} + 2n^{33}$, including the n^{33} steps just after the processor starts and the n^{33} steps just before the processor stops.) The theorem then follows directly from Lemma 36. (Note that C and C' are actually functions of λ, but λ is a constant.) □

Lemma 35 *Consider the distribution obtained from the $\{\lambda_i\}_{1\leq i\leq n}$-independent arrivals distribution by adding the condition that at most m packets arrive by step t. The resulting arrival distribution is $\{\lambda_i\}_{1\leq i\leq n}$-dominated.*

Proof:

Let $A_{v,t'}$ denote the probability that a message arrives at processor v at time t' (under the $\{\lambda_i\}_{1\leq i\leq n}$-independent arrivals distribution). Let E be any event concerning the arrival of messages at steps other than t' or at processors other than v. Let C be the event that at most m messages arrive during steps $1,\ldots,t$. We wish to show that $\Pr(A_{v,t'} \mid C \wedge E) \leq \lambda_v$. If $t' > t$ then $\Pr(A_{v,t'} \mid C \wedge E) = \lambda_v$ by the independence of the $\{\lambda_i\}_{1\leq i\leq n}$-independent arrivals distribution, so suppose that $t' \leq t$. Let E' denote the part of event E concerning arrivals at steps $1,\ldots,t$. By the independence of the $\{\lambda_i\}_{1\leq i\leq n}$-independent 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 processor 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.

$$\begin{aligned}
\Pr(A_{v,t'} \mid C \wedge E') &= \sum_{w\in W} \Pr(A_{v,t'} \mid w \wedge C \wedge E') \Pr(w \mid C \wedge E') \\
&= \sum_{w\in W'} \Pr(A_{v,t'} \mid w \wedge C) \Pr(w \mid C \wedge E') + \sum_{w\in W''} \Pr(A_{v,t'} \mid w \wedge C) \Pr(w \mid C \wedge E') \\
&= \sum_{w\in W'} \Pr(A_{v,t'} \mid w) \Pr(w \mid C \wedge E') \\
&= \lambda_v \sum_{w\in W'} \Pr(w \mid C \wedge E') \\
&\leq \lambda_v.
\end{aligned}$$

□

Lemma 36 *Suppose that for every m and t a protocol running on n processors has the property: for all processors v, if a packet P is generated at processor v at step $t \in T$ and is one of the first m packets generated, then the expected time before packet P is sent is at most C, and if a packet P is generated at processor v at step $t \in \overline{T}$ and is one of the first m packets generated, then the expected time before packet P is sent is at most C'. Then* $\mathrm{E}[W_{avg}] \le 2(SC + C')$, *where* $S = \max_{v \in V} \limsup_{t \to \infty} \frac{|T \cap T_v \cap [t]|}{|T_v \cap [t]|}$.

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

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

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

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

Let $B_{m,v,t}$ be the event that one of the first m packets is generated at processor 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 $t \in T_v$,

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

Furthermore, since for $i \le m$ $\Pr(A_{i,v,t}) = \Pr(A_{i,v,t} \wedge B_{m,v,t}) = \Pr(A_{i,v,t} \mid B_{m,v,t}) \Pr(B_{m,v,t})$,

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

Let $\nu_t = \sum_{v' \in V} \lambda_{v'} |T_{v'} \cap [t]|$, i.e. the expected number of packets generated in the system through time t. Note that $\Pr(B_{m,v,t}) \le \lambda_v$, and for $m < \nu_t$, $\Pr(B_{m,v,t}) \le \lambda_v \exp\{-(\nu_t -$

$m)^2/(2\nu_t)\}$, by a Chernoff bound. Then for any $T^* \subseteq T_v$,

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

Consequently,

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

We bound the factor multiplied by C as follows.

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

We bound the factor multiplied by C' as follows.

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

□

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 acknowledgement based transmission control of random access communication channels. *IEEE Trans. on Information Theory*, IT-33(2):219–223, 1987.

[3] 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 STOC 85.

[4] J. Håstad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access channels. Pre-print - a preliminary version appeared in STOC 87, 1993.

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

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

[7] M. Paterson and A. Srinivasan. Contention resolution with bounded delay. to appear in Proc. 34th Symp. on Found. of Comp. Sci., 1995.

[8] P. Raghavan and E. Upfal. Stochastic contention resolution with short delays. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 229–237, 1995.