The Need for Headers: An Impossibility Result for Communication over Unreliable Channels

Alan Fekete
Software Systems Research Group
Department of Computer Science
University of Sydney
NSW 2006
AUSTRALIA

Nancy Lynch
MIT Lab for Computer Science
545 Technology Square
Cambridge, MA 02139
U.S.A.

Abstract

It is proved that any protocol that constructs a reliable data link service using an unreliable physical channel service necessarily includes in the packets some header information that enables the protocol to treat different packets differently. The physical channel considered is permitted to lose, but not reorder or duplicate packets. The formal framework used for the proof is the I/O automaton model.

1 Introduction

Probably the most common use for formal models of concurrent programming is to verify that protocols meet their specifications. A less common, but nonetheless important use for such models is to prove impossibility results, showing that no protocol can possibly solve a particular problem. We believe that it is important to the coherence of our research field that the same formal models be used as the foundation for both kinds of activities.

Some of the features of a formal model that are needed to support impossibility results are the same as those needed for verification; for instance, both kinds of models must allow separate description of the problem to be solved and the allowable implementations. However, some other features may be different. A useful model for proving impossibility results should include a notion of "fair" or "admissible" execution, which describes exactly when system components are required to continue taking steps; otherwise, it will be impossible to guarantee that the system satisfies any liveness properties. Also, a model for proving impossibility results needs to designate which activities are under the control of the protocol and which are under the control of the environment; otherwise, a protocol might "solve" a problem in a trivial way, simply by preventing some of the inputs from occurring. A more extensive discussion of these features can be found in [12].

We believe that the I/O automaton model [10, 11] can serve as a reasonable basis for both verifying concurrent algorithms and for proving impossibility results. The model has already been used in verifying a wide range of algorithms (see, for example, [10, 3, 4, 5, 7, 14, 16, 17]). In this paper, we use it to prove an impossibility result, namely, that any protocol that implements a reliable data link service by using an unreliable physical channel service necessarily includes in

The authors were supported in part by the National Science Foundation under grant CCR-86-11442, by the Office of Naval Research under contract N00014-85-K-0168 and by the Defense Advanced Research Projects Agency under contracts N00014-83-K-0125 and N00014-89-J-1988. The first author was supported in part by the Research Foundation for Information Technology of the University of Sydney.

the packets some header information that enables the protocol to treat different packets differently. We believe that the main interest of this work is not so much in the result itself (no one has ever suggested using a protocol without header information) but rather in the way the model is used to show nonexistence of a protocol with certain properties.

There has recently been a lot of research in the distributed computing theory research community on the problem of constructing a reliable message transmission service using an underlying unreliable packet transmission service (see [9, 1, 18, 13], for example). Most of this work has addressed the case where the physical channel is especially unreliable, in that it can lose packets and also deliver packets out of order. In these cases the natural protocol, due to Stenning ([15]) places each message in a packet with a sequence number as header, and repeatedly sends the packet until its receipt has been acknowledged. The difficulty with this protocol is that the sequence numbers increase without bound, and the papers mentioned above explore the possibility of using a fixed size header. By contrast, in this paper we consider using a FIFO (but possibly lossy) physical channel. There are many protocols known for this situation, most being variants on the Alternating Bit protocol [2], in which packets and acknowledgments contain a single bit header. We show that this header is needed, in that there is no protocol that solves the same problem without using some header to distinguish between packets. A key modeling issue is how to define "headers" in an arbitrary protocol, without assuming a particular structure (such as [sequence-number,message]) for the packets. The definitions we use are adapted (and simplified) from those in [9, 6].

The rest of the paper is organized as follows. In Section 2, we show how we model physical channels, and also show the existence of a "universal" physical channel, which exhibits all the behaviors that can arise in any physical channel. In Section 3, we give the specification for correct data link behavior. In Section 4, we define data link protocols and show what it means for them to implement correct data link behavior using two unidirectional physical channels. In Section 5 we prove a preliminary impossibility result, namely, we show the impossibility of implementing a data link service using *identical* packets in each direction. This result contains most of the complexity of our main result, but avoids the issue of how to model the headers. In Section 6, we discuss how to define the headers used by an arbitrary protocol, and invoke our preliminary result to obtain the main theorem. A summary of the results we need about the I/O automaton model is given in Appendix A, and the construction of a universal physical channel is given in Appendix B.

2 The Physical Layer

The physical layer is the lowest layer in the OSI Reference Model hierarchy, and is implemented directly in terms of the physical transmission media. A standard interface to the physical layer permits implementation of the higher layers independently of the transmission media. In a typical setting, a physical layer interacts with higher layers at two endpoints, a "transmitting station" and a "receiving station". The physical layer receives messages called "packets" from the higher layer at the transmitting station, and delivers some of the packets to the higher layer at the receiving station. The physical layer can lose packets. While it is also possible for packets to be corrupted by the transmission medium, we assume that the physical layer masks such corrupted packets using error-detecting codes. Thus, the only faulty behavior we consider is loss of packets.

In this section, we give a specification for physical layer behavior; in particular, we specify a channel that ensures FIFO delivery of packets. It is convenient to parameterize the specification by a channel name n and by an alphabet P of legal *packets*. The specification will be given (as usual in the I/O automaton model) by a pair, $PL^{n,P}$, consisting of an action signature, $sig(PL^{n,P})$, describing the interface the layer provides, and a set of sequences of actions, $scheds(PL^{n,P})$, describing the allowed interactions. $PL^{n,P}$ has the action signature given formally as follows.

Input actions:
$send\_pkt^n(p)$, $p \in P$
Output actions:
$rec\_pkt^n(p)$, $p \in P$

The $send\_pkt^n(p)$ action represents the sending of packet p on the physical channel by the transmitting station, and the $rec\_pkt^n(p)$ represents the receipt of packet p by the receiving station. We will refer to these actions as *physical layer actions* (for n and P). In order to define the set of sequences, $scheds(PL^{n,P})$, we define first a collection of properties, reflecting the operation of a "good" physical channel. The properties are defined with respect to $\beta = \pi_1\pi_2\ldots$ a (finite or infinite) sequence of physical layer actions, and a *correspondence relation*, a binary relation between the $send\_pkt^n$ events and the rec_pkt^n events in β. The correspondence relation is intended to model the association that can be set up between the event modeling the sending of a packet and the event modeling the receipt of the same packet. Complications are caused by the fact that the same data might be sent repeatedly, and so the sending of two such identical packets is modeled by two occurrences of the same action $send\_pkt^n(p)$. The first property gives basic requirements on the correspondence.

(PL1) 1. If an event $\pi_i = rec\_pkt^n(p)$ corresponds to an event $\pi_j = send\_pkt^n(q)$, then $p = q$, and also $j < i$, that is, the event π_j precedes π_i in β.
2. Each rec_pkt^n event corresponds to exactly one $send\_pkt^n$ event.
3. Each $send\_pkt^n$ event corresponds to *at most* one rec_pkt^n event.

We next define the FIFO property. It says that those packets that are delivered have their *receive_pkt* events occurring in the same order as their *send_pkt* events. Note that (PL2) may be true even if a packet is delivered and some packet sent earlier is not delivered: there can be gaps in the sequence of delivered packets representing lost packets.

(PL2) (FIFO) Suppose that the event $\pi_i = send\_pkt^n(p)$ in β corresponds to the event $\pi_j = rec\_pkt^n(p)$, and $\pi_k = send\_pkt^n(p')$ corresponds to $\pi_l = rec\_pkt^n(p')$. Then $i < k$ if and only if $j < l$.

So far, the properties listed have been safety properties, that is, when they hold for a sequence they also hold for any prefix of that sequence. The final property is a liveness property. It says that if repeated send events occur, then eventually some packet is delivered.

(PL3) If infinitely many $send\_pkt^n$ actions occur in β, then infinitely many rec_pkt^n actions occur in β.

Now we define $scheds(PL^{n,P})$ to be the set of those sequences β of physical layer actions for which there exists a correspondence such that (PL1), (PL2), and (PL3) are all satisfied for β and that correspondence. A *FIFO physical channel* for n and P is any I/O automaton whose external signature is $sig(PL^{n,P})$ and whose fair behaviors are all in the set $scheds(PL^{n,P})$.

We close this section by defining "universal FIFO physical channels", that is, automata whose fair behaviors are exactly the set of sequences permitted by the specification above. Namely, we say that an I/O automaton is a *universal physical channel* for n and P if it is a FIFO physical channel for n and P and the set of its fair behaviors is exactly the set $scheds(PL^{n,P})$. We give the construction of a universal FIFO physical channel in Appendix B, but for most of this paper all that we will need is the fact that one exists:

Lemma 2.1 *For any n and P, there exists a universal FIFO physical channel for n and P.*

The following lemma will allow us to argue in Section 5 that certain sequences of actions which we will construct by extending behaviors of a universal FIFO physical channel are themselves fair behaviors of the channel. The intuition behind this result is that after any history, every packet not yet delivered might be lost, and then subsequent activity of the channel would be as if it restarted in an initial state.

Lemma 2.2 *Let C be a universal physical channel for n and P. Let β be a finite behavior of C and γ any fair behavior of C. Then $\beta\gamma$ is a fair behavior of C.*

3 The Data Link Layer

The data link layer is the second lowest layer in the hierarchy, and is implemented using the services of the physical layer. Generally, it is implemented in terms of two physical channels, one in each direction. It provides a reliable one-hop message delivery service, which can in turn be used by the next higher layer.

In this section, we specify a weak form of data link behavior. We again assume that there are two endpoints, a "transmitting station" and a "receiving station". The data link layer receives messages from the higher layer at the transmitting station, and delivers them at the receiving station. The data link layer guarantees that every message that is sent is eventually received. The order of the messages need not be preserved, however. We give a parameterized specification for data link layer behavior[1] denoted DL^M, where M is an alphabet of legal *messages*. The interface is described by $sig(DL^M)$, which is given formally as follows.

Input actions:
    $send\_msg(m)$, $m \in M$
Output actions:
    $rec\_msg(m)$, $m \in M$

The $send\_msg(m)$ action represents the sending of message m on the data link by the transmitting station, and the $rec\_msg(m)$ represents the receipt of message m by the receiving station. We will refer to these actions as *data link layer actions.* In order to define the set of allowed interactions $scheds(DL^M)$, we again define a collection of auxiliary properties. They are defined for a sequence $\beta = \pi_1\pi_2\ldots$ of data link layer actions and a correspondence relationship, a binary relation between the $send\_msg$ events and the rec_msg events in β. The first property is analogous to (PL1) and gives elementary requirements on the correspondence.

(DL1) 1. If an event $\pi_i = rec\_msg(m)$ corresponds to an event $\pi_j = send\_msg(n)$, then $m = n$, and also $j < i$, that is, the event π_j precedes π_i in β.
2. Each $rec\_msg(m)$ event corresponds to exactly one $send\_msg(m)$ event.
3. Each $send\_msg(m)$ event corresponds to *at most* one $rec\_msg(m)$ event.

The remaining property is the data link layer liveness property. It says that all messages that are sent are eventually delivered. This property expresses the reliability of the message delivery guaranteed by the data link layer.

[1] A stronger form of data link behavior, including a FIFO message delivery property, is described in [9]. While the specification in this paper is less interesting for describing properties of a useful data link layer, it is adequate for proving our impossibility result. A similar impossibility result for the stronger data link specification following immediately from ours.

(DL2) If π is a *send_msg*(m) event occurring in β, then there is a *rec_msg*(m) event in β corresponding to π.

Note that (DL1) and (DL2) together imply that there is exactly one *rec_msg*(m) event corresponding to each *send_msg*(m) event. Now we define $scheds(DL^M)$ to be the set of sequences β of data link layer actions for which there exists a correspondence relation such that (DL1) and (DL2) are satisfied for β and the correspondence relation.

We have the following immediate consequence of the definition:

Lemma 3.1 *If β is in $scheds(DL^M)$ then the number of send_msg events in β is equal to the number of rec_msg events in β.*

4 Data Link Implementation

In this section, we define a "data link protocol", which is intended to be used to implement the data link layer using the services provided by the physical layer. A data link protocol consists of two automata, one at the transmitting station and one at the receiving station. These automata communicate with each other using two physical channels, one in each direction. They also communicate with the outside world, through the data link layer actions we defined in Section 3.

Let t and r again be names (for the transmitting and receiving station respectively). Let M, P^{tr} and P^{rt} be alphabets (of messages, forward packets and backwards packets, respectively). Then a *transmitting automaton* for (t, r) and (M, P^{tr}, P^{rt}) is any I/O automaton having the following external action signature.

Input actions:
  send_msg(m), $m \in M$
  rec_pkt$^{rt}(p)$, $p \in P^{rt}$
Output actions:
  send_pkt$^{tr}(p)$, $p \in P^{tr}$

In addition, there can be any number of internal actions. That is, a transmitting automaton receives requests from the environment of the data link layer to send messages to the receiving station r. It sends packets to r over the physical channel to r. It also receives packets over the physical channel from r. Similarly, a *receiving automaton* for (t, r) and (M, P^{tr}, P^{rt}) is any I/O automaton having the following external signature.

Input actions:
  rec_pkt$^{tr}(p)$, $p \in P^{tr}$
Output actions:
  send_pkt$^{rt}(p)$, $p \in P^{rt}$
  rec_msg(m), $m \in M$

Again, there can also be any number of internal actions. That is, a receiving automaton receives packets over the physical channel from t. It sends packets to t over the physical channel to t, and it delivers messages to the environment of the data link layer. A *data link protocol* for (t, r) and (M, P^{tr}, P^{rt}) is a pair (A^t, A^r), where A^t is a transmitting automaton for (t, r) and (M, P^{tr}, P^{rt}), and A^r is a receiving automaton for (t, r) and (M, P^{tr}, P^{rt}). (Often we will omit mention of the station names and the alphabets, if these are clear from context.)

Now we are ready to define correctness of data link protocols. Informally, we say that a data link protocol is "correct" provided that when it is composed with any "correct physical layer" (i.e. a pair of FIFO physical channels from t to r and from r to t, respectively), the resulting system yields correct data link layer behavior. This reflects the fundamental idea of layering, that the implementation of one layer should not depend on the details of the implementation of other layers, so that each layer can be implemented and maintained independently. Formally, suppose (A^t, A^r) is a data link protocol for (t, r) and (M, P^{tr}, P^{rt}). We say that (A^t, A^r) is *correct* provided that the following is true. For all C^{tr} and C^{rt} such that C^{tr} is a FIFO physical channel for tr and P^{tr}, and C^{rt} is a FIFO physical channel for rt and P^{rt}, and for every sequence β that is the projection on the actions of data link layer actions of a fair behavior of the composition of A^t, A^r, C^{tr} and C^{rt}, it is the case that β is in $scheds(DL^M)$.

The definition of correctness for a data link protocol requires us to examine its behavior when combined with any possible FIFO physical channels. However, examining the definition shows that we are able to prove the impossibility of a correct protocol satisfying certain requirements by merely demonstrating that no such protocol works when combined with a specific pair of physical channels. In fact we will do our impossibility proofs by considering a system constructed with any arbitrarily chosen universal FIFO physical channels.[2]

If $A = (A^t, A^r)$ is a data link protocol for (t, r) and (M, P^{tr}, P^{rt}), C^{tr} is a universal FIFO physical channel for tr and P^{tr}, and C^{rt} is a universal FIFO physical channel for rt and P^{rt}, then we denote by $D(A)$ an automaton that is the composition of A^t, A^r, C^{tr} and C^{rt}. Then by virtue of Theorem A.4, we have the following result:

Lemma 4.1 *A data link protocol A is correct if and only if for every fair behavior β of $D(A)$, the projection of β on the data link layer actions is an element of $scheds(DL^M)$.*

Corollary 4.2 *Suppose that A is a correct data link protocol. Then in every fair behavior of $D(A)$, the number of send_msg events is equal to the number of rec_msg events.*

Proof: This is immediate from Lemma 4.1 and Lemma 3.1. ■

5 Impossibility of Having All Packets Identical

We show here the impossibility of constructing a correct data link protocol that uses only a single type of packet (and so needs no header) to transmit a sequence of identical messages. This result seems weaker than the result we want (that it is impossible to construct a correct data link protocol where all packets contain the same header) but in Section 6 we will show that in fact the desired result follows from this. By making this simplification we postpone some of the difficult modeling issues, and allow the reader to see the style of impossibility proof in a simpler setting. The technique, of measuring the number of headers by the size of the packet alphabet when the message alphabet has size one, was used earlier without formal proof of a reduction result in [13].

The proof takes the following form: we assume for the purpose of obtaining a contradiction that A is a correct data link protocol in which, in each direction, all packets are identical. Then Corollary 4.2 implies that, in every fair behavior of $D(A)$, the number of *send_msg* events is equal to the number of *rec_msg* events. Then in Lemmas 5.1 - 5.3, we deduce a series of three preliminary facts about the states of the end stations during executions of $D(A)$, by showing that the failure of one of these facts, coupled with the previously derived facts, would enable us to construct a fair

[2]We do not use the fact in this paper, but we note that since the universal FIFO physical channels have all possible behaviors of any FIFO physical channels, we can prove that a particular protocol is correct by analyzing it in a system with just universal FIFO physical channels.

behavior in which the number of *send_msg* events is unequal to the number of *rec_msg* events. Finally, we use these facts in Lemma 5.4 to construct two fair behaviors of $D(A)$ with identical projections at the receiving automaton, but in one of which two messages are sent while in the other only one message is sent. Since the projections at the receiver are equal, the two executions contain the same number of *rec_msg* events. Thus one of them will have the number of *send_msg* events unequal to the number of *rec_msg* events. This yields a contradiction to the original assumption that the protocol was correct.

5.1 Preliminary Lemmas

In this subsection, we assume that A is a correct data link protocol for (t, r) and (M, P^{tr}, P^{rt}) with $|P^{tr}| = |P^{rt}| = 1$. We also assume that each of A^t and A^r is deterministic, that is, it has only one initial state, at most one locally controlled action is enabled in each state, and at most one new state can be reached by applying an action in a state. As we will see later, this involves no loss of generality. Given a state of an end station (i.e., A^t or A^r) in such a protocol A, there is a *unique* fair execution fragment of that automaton that commences with the given state and includes no input actions. (This execution corresponds to running the automaton from the given state in such a way that it receives no inputs, for as long as it can keep taking steps.) We will say that the given state is *quiescing* if this fragment contains only finitely many *send_pkt* events.

Our first lemma says that from any state in any execution, if the transmitting automaton is run without receiving any inputs (that is, with no *send_msg* or rec_pkt^{rt} events) then it must send only finitely many packets to the receiver.

Lemma 5.1 *If α is a finite execution of $D(A)$, then A^t is quiescing in the final state of α.*

Proof: The idea of the proof is as follows. If A^t sends infinitely many packets with no response then A^r has no hope of determining how many *send_msg* events have happened. In particular, we show that A^r cannot tell the difference between the situation in which one additional *send_msg* event occurs after the given finite execution and the situation in which no such events occur.

More precisely, suppose that A^t is not quiescing in the final state of α. Let $\beta = beh(\alpha)$. Then consider the behavior $\beta send\_msg(m)$ where m is some arbitrary message in the message alphabet of A. This behavior has an extension that is fair and contains no extra *send_msg* events (by Theorem A.1). By Corollary 4.2, there is a finite prefix of this extension, say $\beta send\_msg(m)\beta'$ which contains as many *rec_msg* events as there are *send_msg* events in $\beta send\_msg(m)$, namely, one more than the number of *send_msg* events in α.

Let k be the number of rec_pkt^{tr} events in β'. Since A^t is not quiescing in the final state of α, there is a finite behavior of A^t that is an extension of $\beta|A^t$, say $(\beta|A^t)\gamma$, where γ consists of exactly k $send\_pkt^{tr}$ events (but no *send_msg* or rec_pkt^{rt} events).

Now we consider the sequence $\beta\gamma(\beta'|A^r)$ of actions of $D(A)$. We show that this sequence is a (not necessarily fair) behavior of $D(A)$, by showing that its projection on each of the four components of the system is a behavior of that component.

1. The projection on A^t is $(\beta|A^t)\gamma$, which is a behavior of A^t by construction.

2. The projection on A^r is $(\beta\beta')|A^r$ (since γ involves only actions of A^t). This is a behavior of A^r since it is equal to $(\beta send\_msg(m)\beta')|A^r$.

3. The projection on C^{tr} is $(\beta|C^{tr})(\gamma|C^{tr})((\beta'|A^r)|C^{tr})$. Since $\gamma|C^{tr}$ is a sequence of k $send\_pkt^{tr}(p)$ events and $(\beta'|A^r)|C^{tr}$ is a sequence of k $rec\_pkt^{tr}(p)$ events, where p is the unique element of the packet alphabet P^{tr}, it follows from the universality of C^{tr} that $(\gamma|C^{tr})((\beta'|A^r)|C^{tr})$ is a fair behavior of C^{tr}. Then Lemma 2.2 implies that $(\beta|C^{tr})(\gamma|C^{tr})((\beta'|A^r)|C^{tr})$ is a behavior of C^{tr}.

4. The projection on C^{rt} is $(\beta|C^{rt})((\beta'|A^r)|C^{rt})$ (since γ involves no actions of C^{rt}) and this is a behavior of C^{rt} since $(\beta'|A^r)|C^{rt}$ consists only of $send\_pkt^{rt}$ events, which are inputs to C^{rt} (and I/O automata are input-enabled).

Since its projection on each component is a behavior of that component, the sequence $\beta\gamma(\beta'|A^r)$ is a behavior of $D(A)$, by Theorem A.4.

Now consider the two behaviors $\beta\gamma(\beta'|A^r)$ and $(\beta)send\_msg(m)\beta'$. They both have the same projection on A^r and hence contain the same number of rec_msg events. By the argument at the beginning of this proof, this number is exactly one more than the number of $send\_msg$ events in β. On the other hand, the number of $send\_msg$ events in $\beta\gamma(\beta'|A^r)$ is the same as the number of $send\_msg$ events in $\beta\gamma$ (the two sequences having the same projection on A^t), and this is the same as the number of $send\_msg$ events in β. Thus, the number of $send\_msg$ events in the behavior $\beta\gamma(\beta'|A^r)$ is one fewer than the number of rec_msg events in the same behavior $\beta\gamma(\beta'|A^r)$.

Now when we consider a fair extension of $\beta\gamma(\beta'|A^r)$ that contains no further $send\_msg$ events, as given by Theorem A.1, we find that it contains more rec_msg events than $send\_msg$ events. This contradicts Corollary 4.2. ■

Our second lemma says that from any state in any execution, if the receiving automaton is run without receiving any inputs, then it must send infinitely many packets to the transmitter.

Lemma 5.2 *If α is a finite execution of $D(A)$, then A^r is not quiescing in the final state of α.*

Proof: Suppose the contrary: that A^r is quiescing in the final state. Once again, we reach a contradiction by constructing two fair behaviors of $D(A)$ with the same number of rec_msg events, but different numbers of $send\_msg$ events.

Let α_1 be the fair execution fragment of A^t containing no inputs and starting from the state of A^t at the end of α. Similarly let α_2 be the fair execution fragment of A^r containing no inputs and starting from the state of A^r at the end of α. By definition, the projection of $\alpha\alpha_1$ on A^t is a fair execution of A^t, and likewise the projection of $\alpha\alpha_2$ on A^r is a fair execution of A^r. By Lemma 5.1, α_1 contains only finitely many $send\_pkt^{tr}$ events, and by assumption α_2 contains only finitely many $send\_pkt^{rt}$ events. Let γ be any sequence of actions formed by interleaving the sequences $beh(\alpha_1)$ and $beh(\alpha_2)$.

We claim that $beh(\alpha)\gamma$ is a fair behavior of $D(A)$. We show this by showing that its projection on each of the four components of the system is a fair behavior of that component.

1. The projection on A^t is just $beh((\alpha|A^t)\alpha_1)$, which is a fair behavior of A^t by the definition of α_1.

2. The projection on A^r is $(beh(\alpha|A^r)\alpha_2)$ which is a fair behavior of A^r by the definition of α_2.

3. The projection on C^{tr} is just the projection of beh(α) on that channel followed by a finite number of $send\_pkt^{tr}$ events. This is a fair behavior of C^{tr} by the universality of C^{tr} and Lemma 2.2.

4. Similarly the projection on C^{rt} is a fair behavior of C^{rt}.

Since its projection on each component is a fair behavior of that component, Theorem A.4 implies that $beh(\alpha)\gamma$ is a fair behavior of $D(A)$. By Corollary 4.2 the number of rec_msg events in $beh(\alpha)\gamma$ equals the number of $send\_msg$ events in the same sequence.

Now let m be an arbitrary element of the message alphabet. We construct another fair behavior, $beh(\alpha)send\_msg(m)\gamma'$, which contains the same number of rec_msg events as in $beh(\alpha)\gamma$, but contains one more $send\_msg$ event, which yields a contradiction.

Let α_3 be the fair execution fragment of A^t containing no inputs and starting from the state of A^t at the end of $\alpha send\_msg(m)$. By Lemma 5.1, α_3 contains only finitely many $send\_pkt^{tr}$ events. Now we consider the sequence of actions γ' formed by interleaving (in any fashion) the sequences $beh(\alpha_3)$ and $beh(\alpha_2)$. Just as above, $beh(\alpha)send\_msg(m)\gamma'$ is a fair behavior of $D(A)$, and so by Corollary 4.2, the number of rec_msg events in $beh(\alpha)send\_msg(m)\gamma'$ is equal to the number of $send\_msg$ events. However, since $beh(\alpha)send\_msg(m)\gamma'|A^r$ and $beh(\alpha)\gamma|A^r$ are both equal to $beh(\alpha|A^r)beh(\alpha_2)$, we see that the number of rec_msg events in $beh(\alpha)send\_msg(m)\gamma'$ is the same as the number of rec_msg events in $beh(\alpha)\gamma$. By the equalities proved above, $beh(\alpha)\gamma$ and $beh(\alpha)send\_msg(m)\gamma'$ contain the same number of $send\_msg$ events, which is false. This is a contradiction. ■

The third lemma further characterizes the behavior of a correct data link protocol by showing that the transmitter must both send and receive infinitely many packets.

Lemma 5.3 *If α is a fair execution of $D(A)$ that contains a finite nonzero number of $send\_msg$ actions, then $\alpha|A^t$ contains infinitely many $send\_pkt^{tr}$ actions and infinitely many rec_pkt^{rt} actions.*

Proof: We show that every other possibility leads to a contradiction.

1. $\alpha|A^t$ contains infinitely many $send\_pkt^{tr}$ actions and finitely many rec_pkt^{rt} actions. Then there is a suffix of $\alpha|A^t$ that contains no input actions (neither $send\_msg$ nor rec_pkt^{rt} actions) but contains infinitely many $send\_pkt^{tr}$ actions. The state of A^t at the start of this suffix must be not quiescing, which contradicts Lemma 5.1.

2. $\alpha|A^t$ contains finitely many $send\_pkt^{tr}$ actions and finitely many rec_pkt^{rt} actions. Then $\alpha|A^r$ contains finitely many rec_pkt^{tr} actions (since the channel C^{tr} delivers at most as many packets as were sent) and contains finitely many $send\_pkt^{rt}$ actions (since a fair execution of C^{rt} would contain an infinite number of rec_pkt^{rt} events if it contained an infinite number of $send\_pkt^{rt}$ events). Thus there is a suffix of $\alpha|A^r$ that contains no input events and only a finite number of $send\_pkt^{rt}$ events. The state of A^r at the start of this suffix is quiescing, which contradicts Lemma 5.2.

3. $\alpha|A^t$ contains finitely many $send\_pkt^{tr}$ actions and infinitely many rec_pkt^{rt} actions. First consider the maximal execution of A^r starting from the initial state of A^r and containing no input actions. This execution is a fair execution of A^r. Let β be the behavior of this execution. By Lemma 5.2, β contains infinitely many $send\_pkt^{rt}$ actions. Let γ be the sequence of actions obtained by interleaving β and $beh(\alpha|A^t)$ in such a way that for each i the i-th rec_pkt^{rt} action is immediately preceded by the i-th $send\_pkt^{rt}$ action. We claim that γ is a fair behavior of $D(A)$. Its projections on A^t and A^r are fair behaviors by construction. Its projection on C^{rt} is just $send\_pkt^{rt}(p)rec\_pkt^{rt}(p)send\_pkt^{rt}(p)rec\_pkt^{rt}(p)\ldots$ (where $P^{rt} = \{p\}$) which is a fair behavior since (PL1)-(PL3) are satisfied using the obvious correspondence between each rec_pkt^{rt} event and the immediately preceding $send\_pkt^{rt}$ event. Finally, its projection on C^{tr} is a fair behavior since it consists of the sending of a finite number of packets and the delivery of none (as β contains no inputs to A^r, in particular no rec_pkt^{tr} actions) and this clearly satisfies (PL1)-(PL3).

   We observe that β (and hence γ) cannot contain any rec_msg action. (Otherwise, take the prefix of β up to and including the first rec_msg event, regard it as a behavior of $D(A)$ where all the actions take place at A^r, and extend it to a fair behavior of $D(A)$ which contains no inputs to $D(A)$, that is, no $send\_msg$ actions. This contradicts Corollary 4.2.) However, γ contains a nonzero number of $send\_msg$ events (the same ones as in α). Thus γ is a fair behavior of $D(A)$ which does not contain the same number of rec_msg events as of $send\_msg$ events, contradicting Corollary 4.2.

5.2 The Theorem

We now use the facts proved in the previous subsection to construct two executions that look identical to the receiver, but have different numbers of messages sent at the transmitting end.

Lemma 5.4 *Suppose that A is a correct data link protocol for (t,r) and (M, P^{tr}, P^{rt}) with $|P^{tr}| = |P^{rt}| = 1$, and suppose that each of A^t and A^r is deterministic.*

Let m be an arbitrary element of the message alphabet. Let β_1 and β_2 be two fair behaviors of $D(A)$ such that β_1 begins with send_msg(m) and contains no other send_msg event, and β_2 begins with send_msg(m)send_msg(m) and contains no other send_msg event. Then there exist fair behaviors $\hat{\beta}_1$ and $\hat{\beta}_2$ of $D(A)$ such that $\hat{\beta}_j|A^t = \beta_j|A^t$ for $j = 1, 2$, and such that $\hat{\beta}_1|A^r = \hat{\beta}_2|A^r$.

Proof: Applying Lemma 5.3 to the behavior β_1, we see that we may express $\beta_1|A^t$ as an infinite sequence

$$send\_msg(m)\beta_1^1\gamma_1^1\beta_1^2\gamma_1^2\beta_1^3\ldots$$

where each β_1^i consists only of $send\_pkt^{tr}$ actions, each γ_1^i consists only of rec_pkt^{rt} actions, and where each β_1^i (except possibly β_1^1) and each γ_1^i contains a finite, nonzero number of $send\_pkt^{tr}$ or rec_pkt^{rt} events. [3] Similarly, we may express $\beta_2|A^t$ as an infinite sequence

$$send\_msg(m)send\_msg(m)\beta_2^1\gamma_2^1\beta_2^2\gamma_2^2\beta_2^3\ldots$$

where each β_2^i consists only of $send\_pkt^{tr}$ actions, each γ_2^i consists only of rec_pkt^{rt} actions, and where each β_2^i (except possibly β_2^1) and each γ_2^i contains a finite, nonzero number of $send\_pkt^{tr}$ or rec_pkt^{rt} events. Let a_j^i denote the number of rec_pkt^{rt} events in γ_j^i, for $j = 1, 2$.

We next claim that there exists an infinite fair behavior of A^r of the following form:

$$\eta^1 rec\_pkt^{tr}(p)\eta^2 rec\_pkt^{tr}(p)\eta^3 rec\_pkt^{tr}(p)\ldots$$

(where p is the unique element of P^{tr}), such that each η^i is a finite sequence containing exactly $\max(a_1^i, a_2^i)$ $send\_pkt^{rt}$ events and no input events of A^r. In order to prove this claim, we first show, inductively on i, that $\eta^1 rec\_pkt^{tr}(p)\eta^2 rec\_pkt^{tr}(p)\eta^3 rec\_pkt^{tr}(p)\ldots\eta^i rec\_pkt^{tr}(p)$ is a behavior of A^r; this follows because any state of A^r after $\eta^1 rec\_pkt^{tr}(p)\eta^2 rec\_pkt^{tr}(p)\eta^3 rec\_pkt^{tr}(p)\ldots\eta^i rec\_pkt^{tr}(p)$ is not quiescent according to Lemma 5.2, and $rec\_pkt^{tr}(p)$ is an input action of A^r. We then observe that since A^r is deterministic and the given infinite sequence contains infinitely many locally controlled events of A^r, any execution with this behavior is a fair execution, so the behavior is a fair behavior.

Now we define $\hat{\beta}_1$ to be the infinite sequence

$$send\_msg(m)\beta_1^1\eta^1\gamma_1^1\beta_1^2 rec\_pkt^{tr}(p)\ldots\eta^{i-1}\gamma_1^{i-1}\beta_1^i rec\_pkt^{tr}(p)\ldots$$

and similarly define $\hat{\beta}_2$ to be the infinite sequence

$$send\_msg(m)send\_msg(m)\beta_2^1\eta^1\gamma_2^1\beta_2^2 rec\_pkt^{tr}(p)\ldots\eta^{i-1}\gamma_2^{i-1}\beta_2^i rec\_pkt^{tr}(p)\ldots$$

We claim that these sequences have all the required properties.

First we see that $\hat{\beta}_1|A^t = send\_msg(m)\beta_1^1\gamma_1^1\beta_1^2\ldots\gamma_1^{i-1}\beta_1^i\ldots$, since η^i and $rec\_pkt^{tr}(p)$ consist entirely of actions of A^r. Thus $\hat{\beta}_1|A^t = \beta_1|A^t$. Similarly, $\hat{\beta}_2|A^t = \beta_2|A^t$. Also, $\hat{\beta}_1|A^r = \eta^1 rec\_pkt^{tr}(p)\ldots\eta^{i-1} rec\_pkt^{tr}(p)\ldots$ since $send\_msg(m)$, β_1^i and γ_1^i consist entirely of events of A^t, Also, $\hat{\beta}_2|A^r = \eta^1 rec\_pkt^{tr}(p)\ldots\eta^{i-1} rec\_pkt^{tr}(p)\ldots$. Thus, $\hat{\beta}_1|A^r = \hat{\beta}_2|A^r$.

We show that $\hat{\beta}_1$ is a fair behavior of $D(A)$ by examining its projection on each component.

[3] The exception is due to the fact that we do not know whether the first packet sent by A^t precedes or follows the first packet received by A^t.

1. The projection on A^t is equal to $\beta_1|A^t$ as we observed above, which is a fair behavior of A^t by Theorem A.3 since β_1 is a fair behavior of $D(A)$.

2. The projection on A^r is equal to $\eta^1 rec\_pkt^{tr}(p)\ldots\eta^{i-1}rec\_pkt^{tr}(p)\ldots$ as observed above, which was shown in the earlier claim to be a fair behavior of A^r.

3. The projection on C^{tr} is equal to $\beta_1^1\beta_1^2 rec\_pkt^{tr}(p)\ldots\beta_1^i rec\_pkt^{tr}(p)\ldots$, i.e., a finite (possibly zero length) sequence of $send\_pkt^{tr}(p)$ events followed by an infinite sequence of segments, each consisting of a finite nonzero number of $send\_pkt^{tr}(p)$ events, followed by one $rec\_pkt^{tr}(p)$ event; this is because none of $send\_msg(m)$, η^i or γ_1^i contains any $send\_pkt^{tr}$ or rec_pkt^{tr} events. It is clear that a correspondence can be found for which this satisfies (PL1)-(PL3). Since C^{tr} is a universal FIFO physical channel, this is a fair behavior of C^{tr}.

4. The projection on C^{rt} is equal to $\eta^1\gamma_1^1\ldots\eta^{i-1}\gamma_1^{i-1}\ldots$, i.e., a sequence of $\max(a_1^1, a_2^1)$ $send\_pkt^{rt}(p')$ events followed by a_1^1 $rec\_pkt^{rt}(p')$ events, then $\max(a_1^2, a_2^2)$ $send\_pkt^{rt}(p')$ events followed by a_1^2 $rec\_pkt^{rt}(p')$ events, and so on, where p' is the unique element of P^{rt}. Since each a_1^i is nonzero but finite, it is clear that a correspondence can be found for which this sequence satisfies (PL1)-(PL3), and so this is a fair behavior of C^{rt}.

Now by Theorem A.4 this shows that $\hat{\beta}_1$ is a fair behavior of $D(A)$. By exactly similar arguments, $\hat{\beta}_2$ is a fair behavior of $D(A)$, which completes the proof. ■

We can now put the pieces together to prove our impossibility result for packet alphabets of size 1.

Theorem 5.5 *There is no correct data link protocol for (t, r) and (M, P^{tr}, P^{rt}) with $|P^{tr}| = |P^{rt}| = 1$.*

Proof: Suppose for the purpose of reaching a contradiction that A is a correct data link protocol with $|P^{tr}| = |P^{rt}| = 1$.

First, we deal with the potential nondeterminism of the end-stations. By Theorem A.2, there is a deterministic automaton B^t (respectively, B^r) with fair behaviors that are a subset of the fair behaviors of A^t (respectively, A^r). Let $B = (B^t, B^r)$, which is also a data link protocol with $|P^{tr}| = |P^{rt}| = 1$. Now by Theorems A.3 and A.4, $fairbehs(D(B)) \subseteq fairbehs(D(A))$, and so B is correct (using Lemma 4.1 and the fact that A is correct).

Now, let m be an arbitrary element of the message alphabet. By Theorem A.1, there are fair behaviors β_1 and β_2 of $D(B)$ such that β_1 begins with $send\_msg(m)$ and contains no other $send\_msg$ event, and β_2 begins with $send\_msg(m)send\_msg(m)$ and contains no other $send\_msg$ event. Consider the fair behaviors $\hat{\beta}_1$ and $\hat{\beta}_2$ whose existence is shown in Lemma 5.4. Since the protocol B is correct, each $\hat{\beta}_j$ satisfies Corollary 4.2 (that is, the number of rec_msg events in $\hat{\beta}_j$ equals the number of $send\_msg$ events in $\hat{\beta}_j$). Now the number of $send\_msg$ events in $\hat{\beta}_j$ is just the number of $send\_msg$ events in $\hat{\beta}_j|B^t$ which equals the number of $send\_msg$ events in $\beta_j|B^t$ by the properties of $\hat{\beta}_j$. Since $\beta_j|B^t$ contains j $send\_msg$ events, we deduce that $\hat{\beta}_j$ contains j rec_msg events, contradicting the fact that $\hat{\beta}_1|B^r$ and $\hat{\beta}_2|B^r$ are equal, and so contain the same number of rec_msg events. ■

6 Impossibility of Having No Headers

Most data link protocols in the literature use a finite packet alphabet in each direction, since packets are required to be of limited size. However, it is normally the case that the packets are treated as having two separate parts: a header (which determines what is to be done with the packet) and an

encapsulated message (treated as an uninterpreted bit string). Indeed, one can envisage protocols that allow packets of unbounded size because the included messages may have unbounded size, and yet use only a fixed size of header (and thus a finite number of headers). Here we sketch one way in which one can model the existence of headers in a protocol, without assuming that the packets are necessarily structured explicitly with two parts, one a control field and the other an uninterpreted message. We then show, using a reduction, how our impossibility result for identical packets implies a corresponding impossibility result for the case of infinite packet alphabets without headers.

We model the "headers" used by a protocol as follows. Let $A = (A^t, A^r)$ be a data link protocol for (t,r) and (M, P^{tr}, P^{rt}). Let $\equiv$ be an equivalence relation on the domain $M \cup P^{tr} \cup P^{rt} \cup states(A^t) \cup states(A^r) \cup acts(A^t) \cup acts(A^r)$. Then $\equiv$ is said to be a *header relation* for A provided that the following conditions hold.

1. $\equiv$ only relates elements of the same kind, i.e., elements of M, or P^{tr}, or $states(A^t)$, etc. Also, a start state cannot be related to a non-start state. Moreover, if $a \equiv a'$ for two actions a and a', then a and a' are identical except possibly for a difference in their message or packet parameter. Further, every pair a and a' of locally controlled events of A^t (respectively, of A^r) such that $a \equiv a'$, a and a' are in the same class of $part(A^t)$ (respectively, of $part(A^r)$).
2. For each pair m, m' of messages in M, $send\_msg(m) \equiv send\_msg(m')$ if and only if $m \equiv m'$, and $rec\_msg(m) \equiv rec\_msg(m')$ if and only if $m \equiv m'$.
3. For each pair p,p' of packets in P^{tr}, $send\_pkt^{tr}(p) \equiv send\_pkt^{tr}(p')$ if and only if $p \equiv p'$, and $rec\_pkt^{tr}(p) \equiv rec\_pkt^{tr}(p')$ if and only if $p \equiv p'$.
4. For each pair p,p' of packets in P^{rt}, $send\_pkt^{rt}(p) \equiv send\_pkt^{rt}(p')$ if and only if $p \equiv p'$, and $rec\_pkt^{rt}(p) \equiv rec\_pkt^{rt}(p')$ if and only if $p \equiv p'$.
5. For every two states q and q' of A^t (respectively, of A^r) with $q \equiv q'$, if action a is enabled in q then there is an action a' with $a \equiv a'$, such that a' is enabled in q'.
6. For every two states q and q' of A^t (respectively, of A^r) and every two actions a and a' of A^t (respectively, of A^r) such that $q \equiv q'$ and $a \equiv a'$, if r is a state such that (q,a,r) is a step of A^t (respectively, of A^r) and action a' is enabled in state q', then there exists a state r' such that $r \equiv r'$ and (q',a',r') is a step of A^t (respectively, of A^r).

For a data link protocol A for (t,r) and (M, P^{tr}, P^{rt}) with a header relation $\equiv$, we define the set $headers(A,t,r,\equiv)$ to be the set of equivalence classes of packets in P^{tr}. Similarly $headers(A,r,t,\equiv)$ is the set of equivalence classes of packets in P^{rt}. We think of each equivalence class of packets as being those (in one direction) with the same pattern of bits in the header. Informally, the way a packet is processed must depend only on the header – for example, if receiving a packet takes the protocol to a state where release of a message to the higher layer is possible, then receiving any other packet containing the same header will also take the protocol to a state where release of a message to the higher layer is possible (however, it may be a different message that is released!) We note that for a data link protocol A, the diagonal relation, where each message, action etc. is equivalent only to itself, is a header relation for A. We say that A has *no header under* $\equiv$ if each of $headers(A,t,r,\equiv)$ and $headers(A,r,t,\equiv)$ is a singleton set, that is, all packets in P^{tr} are related by $\equiv$, as are all packets in P^{rt}. We say that A has *no header* if there exists a header relation $\equiv$ for A such that A has no header under $\equiv$.

In order to prove that headers are necessary for a data link protocol, we show how to reduce the question of the existence of a protocol with sets of header equivalence classes of a given size, to the question of the existence of a protocol using packet alphabets of that size. This will allow us to show that there is no correct data link protocol that has no header using our earlier result that there is no correct data link protocol with packet alphabets of size one. The intuition behind

this reduction comes from the case where packets have the simple form (header, message) and the protocol works uniformly over all message alphabets. In this case, if the protocol is applied to a message alphabet of size one, the packet alphabet will be identical to the set of possible headers. Of course, the proof must be more complicated than this, since we do not assume any simple structure for packets.

Theorem 6.1 *Suppose $A = (A^t, A^r)$ is a correct data link protocol for (t,r) and (M, P^{tr}, P^{rt}). If $\equiv$ is a header relation for A such that $|headers(A,t,r,\equiv)| = h_1$ and $|headers(A,r,t,\equiv)| = h_2$, then there are alphabets M', Q^{tr} and Q^{rt} with $|M'| = 1$, $|Q^{tr}| = h_1$ and $|Q^{rt}| = h_2$ and a correct data link protocol $B = (B^t, B^r)$ for (t,r) and (M', Q^{tr}, Q^{rt}).*

Proof: Choose m to be an arbitrary element of M and put $M' = \{m\}$, $Q^{tr} = headers(A,t,r,\equiv)$ and $Q^{rt} = headers(A,r,t,\equiv)$. [4] These alphabets clearly have the correct cardinalities. Now let B^t be the transmitting automaton for (t,r) and (M', Q^{tr}, Q^{rt}) defined as follows. The input actions of B^t are $send\_msg(m)$ and $rec\_pkt^{rt}(p')$ where p' is an element of Q^{rt}, the output actions are $send\_pkt^{tr}(p')$ where p' is an element of Q^{tr}, and the internal actions are the internal actions of A^t. We say that an action π' of B^t *represents* an action π of A exactly when one of the following conditions holds:

- π' is either $send\_msg(m)$ or an internal action of A^t and $\pi = \pi'$
- π' is $send\_pkt^{tr}(p')$ and $\pi = send\_pkt^{tr}(p)$ for some p which is an element of p'
- π' is $rec\_pkt^{rt}(p')$ and $\pi = rec\_pkt^{rt}(p)$ for some p which is an element of p'.

The states and start states of B^t are the same as those of A^t. The transition relation of B^t includes (s', π', s) exactly when there exists some π that is represented by π' for which $(s', \pi, s) \in steps(A^t)$. The partition $part(B^t)$ relates locally controlled actions π_1' and π_2' exactly when $part(A^t)$ relates some (and hence all) pairs π_1 and π_2 such that π_1 is represented by π_1' and π_2 is represented by π_2'.

Similarly, let B^r be the receiving automaton for (t,r) and (M', Q^{tr}, Q^{rt}) defined as follows. The input actions of B^r are $rec\_pkt^{tr}(p')$ where p' is an element of Q^{tr}, the output actions are $rec\_msg(m)$ and $send\_pkt^{rt}(p')$ where p' is an element of Q^{rt}, and the internal actions are the internal actions of A^r. We say that an action π of A^r *is represented by* an action π' of B^r exactly when one of the following conditions holds:

- π' is either $rec\_msg(m)$ or an internal action of A^r and $\pi = \pi'$
- π' is $send\_pkt^{rt}(p')$ and $\pi = send\_pkt^{rt}(p)$ for some p which is an element of p'
- π' is $rec\_pkt^{tr}(p')$ and $\pi = rec\_pkt^{tr}(p)$ for some p which is an element of p'.

The states and start states of B^r are the same as those of A^r. The transition relation of B^r includes (s', π', s) exactly when there exists some π that is represented by π' for which $(s', \pi, s) \in steps(A^r)$. The partition $part(B^r)$ relates locally controlled actions π_1' and π_2' exactly when $part(A^r)$ relates some (and hence all) pairs π_1 and π_2 such that π_1 is represented by π_1' and π_2 is represented by π_2'.

It is easy to check that (B^t, B^r) is a data link protocol for (t,r) and (M', Q^{tr}, Q^{rt}). We claim that it is correct, proving the theorem.

To prove the claim, it suffices to take an arbitrary fair behavior of $D(B)$ and produce a correspondence for its projection on the data link layer actions such that (DL1) and (DL2) are satisfied. For this we use the specific universal FIFO physical channels $C^{tr,P^{tr}}$ and $C^{rt,P^{rt}}$ whose construction is given in Appendix B, rather than arbitrary ones as we have done previously. The universal

[4] Thus, each packet name in the alphabet Q^{tr} is a set of packet names from the alphabet P^{tr}.

channel used clearly does not affect the set of behaviors of $D(A)$, but it will make it easier to relate executions of different systems, since the construction we give has no internal actions and works uniformly for different packet alphabets.

Thus we consider an arbitrary fair execution $s'_0, \pi'_1, s'_1, \pi'_2, s'_2, \ldots$ of $D(B)$. From this we can construct an execution $s_0, \pi_1, s_1, \pi_2, s_2, \ldots$ of $D(A)$ such that π_i is represented by π'_i for each i, the state of A^t (respectively, of A^r) in s_i is the same as the state of B^t (respectively, of B^r) in s'_i, and the state of $C^{tr,P^{tr}}$ (respectively, of $C^{rt,P^{rt}}$) in s_i is related to the state of $C^{tr,Q^{tr}}$ (respectively, of $C^{rt,Q^{rt}}$) in s'_i in the natural way: the values for the variables S, $counter_1$ and $counter_2$ are the same in $C^{tr,P^{tr}}$ (respectively, in $C^{rt,P^{rt}}$) as in $C^{tr,Q^{tr}}$ (respectively, in $C^{rt,Q^{rt}}$), and for each n the value of $packet(n)$ in $C^{tr,P^{tr}}$ (respectively, in $C^{rt,P^{rt}}$) is one element of its value in $C^{tr,Q^{tr}}$ (respectively, in $C^{rt,Q^{rt}}$) except when both values are undefined.[5] This execution is in fact a fair execution of $D(A)$, as is seen by observing that no action $rec\_msg(m')$ for $m' \neq m$ is enabled in any state s_i (using the correctness of A and the fact that no action π_j is $send\_msg(m')$), and that therefore if a locally controlled action of $D(A)$ is enabled in s_i then it is represented by a locally controlled action of $D(B)$ that is enabled in s'_i. Since this execution is fair, its behavior has projection on the data link layer actions that is an element of $scheds(DL^M)$. However the two executions have identical projections on the data link layer actions (the actions differ only for *send_pkt* and *receive_pkt* events, which are not included in the projection). Thus for $s'_0, \pi'_1, s'_1, \pi'_2, s'_2, \ldots$, its projection on the data link layer actions has a correspondence between *send_msg* and *rec_msg* events that satisfies (DL1) and (DL2). Thus by Theorem 4.1, B is correct. ■

Theorem 6.2 *There is no correct data link protocol that has no header.*

Proof: Immediate from Theorems 5.5 and 6.1. ■

7 Conclusions

In this paper, we have proved the impossibility of constructing a reliable data link service using an unreliable physical channel service, where the packets sent over the physical channels carry no header information. This result was proved using the I/O automaton formal model for concurrent systems.

Some remarks are in order about the style of the proof. The proof is carried out entirely at the "semantic level", i.e., in terms of set-theoretic concepts such as states and sequences of actions. Many formal models (including I/O automata) come equipped with a syntactic component designed to facilitate protocol description and verification. It appears that such syntactic structure is of little value (in fact, may get in the way) for carrying out impossibility proofs. We also note that the proof (as all other impossibility proofs so far discovered) is designed to be understood by people, not by machines. Because of their complexity, we see little evidence that interesting impossibility proofs can (or should) be automated. This is in contrast to the situation for proofs of protocol correctness.

The proof is not an simple as one might expect from the simple and natural statement of the result. It would be desirable to polish and simplify the proof further. But in carrying out such a simplification, one must be careful not to gloss over any of the genuinely subtle issues (e.g., the treatment of fairness or the control of actions) that arise in the proof.

It remains to consider how the same proof can be carried out, or the same result can be proved in other ways, using other popular formal models for concurrency. We suggest this as a challenge for users of those other models. It seems to us that key issues that must be addressed in doing this are those of fairness and control of actions.

[5] The inductive construction of the execution $s_0, \pi_1, s_1, \ldots$ is straightforward.

Finally, we wish to reiterate our belief that the theory of concurrency should be developed in a way that uses the same underlying formal model as a basis for both verification work and for impossibility proofs (and more broadly, for other combinatorial and complexity-theoretic results).

Acknowledgements
We thank Yishay Mansour for his help in our initial discussions about this work, and also Steve Ponzio and Isaac Saias for their useful suggestions for improving the presentation.

References

[1] Attiya, H., Fischer, M., Wang, D.-W., and Zuck, L., "Reliable Communication Using Unreliable Channels", manuscript.

[2] Bartlett, K., Scantlebury, R., and Wilkinson, P., "A Note on Reliable Full-Duplex Transmission over Half-Duplex Links" *Communications of the ACM*, 12(5):260-261, May 1969.

[3] Bloom, B., "Constructing Two-Writer Atomic Registers" *Proceedings of 6th ACM Symposium on Principles of Distributed Computing*, pp. 249-259, August 1987.

[4] Chou, C.-T., and Gafni, E., "Understanding and Verifying Distributed Algorithms Using Stratified Decomposition" *Proceedings of 7th ACM Symposium on Principles of Distributed Computing*, pp. 44-65, August 1988.

[5] Fekete, A., Lynch, N., and Shrira, L., "A Modular Proof of Correctness for a Network Synchronizer" *Proceedings of the 2nd International Workshop on Distributed Algorithms*, Amsterdam, Netherlands, July 1987, (J. van Leeuwen, ed), pp. 219-256. Lecture Notes in Computer Science 312, Springer-Verlag.

[6] Fekete, A., Lynch, N. A., Mansour, Y. and Spinelli, J., "The Data Link Layer: The Impossibility of Implementation in Face of Crashes", Technical Memo, TM-355b, Laboratory for Computer Science, Massachusetts Institute of Technology.

[7] Fekete, A, Lynch, N., Merritt, M., and Weihl, W., "Commutativity-Based Locking for Nested Transactions" to appear in JCSS.

[8] Lynch, N., and Goldman, K., "Distributed Algorithms" Research Seminar Series MIT/LCS/RSS-5, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1989.

[9] Lynch, N. A., Mansour, Y. and Fekete, A., "Data Link Layer: Two Impossibility Results," *Proceedings of 7th ACM Symposium on Principles of Distributed Computing*, pp. 149-170, August 1988.

[10] Lynch N. A. and Tuttle M. R., "Hierarchical Correctness Proofs for Distributed Algorithms," *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, pp. 137-151, August 1987.

[11] Lynch, N., and Tuttle, M., "An Introduction to Input/Output Automata" *CWI Quarterly*, 2(3):219-246, September 1989.

[12] Lynch, N., "A Hundred Impossibility Proofs for Distributed Computing", *Proceedings of 8th ACM Symposium on Principles of Distributed Computing*, pp. 1-28, August 1989.

[13] Mansour, Y., and Schieber, B., "The Intractability of Bounded Protocols for non-FIFO Channels" *Proceedings of 8th ACM Symposium on Principles of Distributed Computing*, pp. 59-72, August 1989.

[14] Nipkow, T., "Proof Transformations for Equational Theories" *Proceedings of 5th Annual IEEE Symposium on Logic in Computer Science*, pp. 278-288, June 1990.

[15] Stenning, N., "A Data Transfer Protocol" *Computer Networks*, 1:99-110, 1976.

[16] Troxel, G., "A Hierarchical Proof of an Algorithm for Deadlock Recovery in a System using Remote Procedure Calls" MS thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science Cambridge, MA. January, 1990.

[17] Welch, J., Lamport, L., and Lynch, N., "A Lattice-Structured Proof of a Minimum Spanning tree Algorithm" *Proceedings of 7th ACM Symposium on Principles of Distributed Computing*, pp. 28-43, August 1988.

[18] Wang, D.-W., and Zuck, L., "Tight Bounds for the Sequence Transmission Problem", *Proceedings of 8th ACM Symposium on Principles of Distributed Computing*, pp. 73-84, August 1989.

A The I/O Automaton Model

The *input/output automaton* model was defined in [10] as a tool for modeling concurrent and distributed systems. We refer the reader to [10] and to the expository paper [11] for a complete development of the model, plus motivation and examples. Here, we mention two extensions to the definitions of [10] and [11], and also provide a summary of the theorems about I/O automata that are needed for our results.

We extend the definition of fairness to execution fragments, not just executions. Namely, an execution fragment α of an automaton A is said to be *fair* if the following condition holds for each class C of $part(A)$: if α is finite, then no action of C is enabled in the final state of α, while if α is infinite, then either α contains infinitely many events from C, or else α contains infinitely many occurrences of states in which no action of C is enabled. Thus, a fair execution fragment gives "fair turns" to each class of $part(A)$.

An I/O automaton is said to be *deterministic* if it has only one initial state, at most one locally controlled action is enabled in each state, and at most one new state can be reached by applying an action in a state.

The following theorem says that no matter what has happened in any finite execution, and no matter what inputs continue to arrive from the environment, an automaton can continue to take steps to give a fair execution.

Theorem A.1 *Let A be an I/O automaton and let γ be a sequence of input actions of A. Suppose that β is a finite schedule of A. Then there exists a fair schedule β' of A such that β' is an extension of β and $\beta'|in(A) = (\beta|in(A))\gamma$. Moreover, the same is true for behaviors in place of schedules.*

The following theorem shows that there is no loss of generality in restricting attention to deterministic solutions to specifications.

Theorem A.2 *If A is an I/O automaton, then there is a deterministic I/O automaton B such that $fairbehs(B) \subseteq fairbehs(A)$.*