A DISCIPLINE FOR CONSTRUCTING
MULTIPHASE COMMUNICATION PROTOCOLS*

Ching-Hua Chow†, Mohamed G. Gouda and Simon S. Lam†

Department of Computer Sciences
University of Texas at Austin

TR-233 June 1983
Revised, October 1983
Revised, March 1985


†Work supported in part by National Science Foundation Grants No. ECS78-01803
and No. ECS83-04734, and in part by a research grant from IBM Corporation.
ABSTRACT

Many communication protocols can be observed to go through different phases performing a distinct function in each phase. We present a multiphase model for such protocols. A phase is formally defined to be a network of communicating finite state machines with certain desirable correctness properties; these include proper termination, and freedom from deadlocks and unspecified receptions. A multifunction protocol is constructed by first constructing separate phases to perform its different functions. We show how to connect these phases together to realize the multifunction protocol such that the resulting network of communicating finite state machines is also a phase (i.e., it possesses the desirable properties defined for phases).

The modularity inherent in multiphase protocols facilitates not only their construction but also their understanding and modification. We found an abundance of protocols in the literature that can be constructed as multiphase protocols. Three examples are presented herein: two versions of IBM's BSC protocol for data link control and a token ring network protocol.

Categories and Subject Descriptors: B.4.4 [Input/Output and Data Communications]: Performance Analysis and Design Aids - formal models, verification; C.2.2 [Computer-Communication Networks]: Network Protocols - protocol architecture, protocol verification; D.1.1 [Programming Techniques]: Concurrent Programming; D.2.2 [Software Engineering]: Tools and Techniques - modules and interfaces, structured programming; D.2.4 [Software Engineering]: Program Verification - correctness proofs, validation.


Additional Key Words and Phrases: Communication protocols, protocol design, modularity, data link control, BSC protocol, token ring protocol, multiphase protocols, proper termination, deadlock freedom, communicating processes, finite state machines.
Table of Contents

1. Introduction 1
2. Networks of Communicating Finite State Machines 3
3. Phases 6
4. Modeling Errors and Timeouts 10
5. Constructing Multiphase Networks 11
6. Boundedness of Multiphase Networks 14
7. A Token Ring Protocol Example 16
8. Achieving Modularity 18
9. Concluding Remarks 20
1. Introduction

A layered communications architecture facilitates the construction of networking software in a modular fashion. Nevertheless each protocol layer is a set of complex parallel programs. Several distinct functions can usually be identified among the tasks designated for a protocol layer to perform. For example, a data link control protocol may be thought of as having at least three functions: connection management and one-way data transfers in opposite directions. In both the analysis and the construction of protocols, however, it is preferable to think about the individual functions of a multifunction protocol one at a time. In fact, most protocol analyses published in the literature have been illustrated with single-function protocols. For examples, both the alternating-bit protocol analyzed by Bochmann [4] and Stenning's protocol analyzed by Stenning [31] and Hailpern and Owicki [17] are concerned with a one-way data transfer function only. The protocol analyses of Kurose and Yemini [20] and Razouk [27] are concerned with the connection management function only.

Of interest to us are methods for reducing the analysis/construction of a multifunction protocol to the analysis/construction of smaller single-function protocols. Lam and Shankar [21] presented a method for constructing "image protocols" from a given multifunction protocol. An image protocol is an abstraction of the original protocol but is specified like any real protocol. It is constructed to preserve all safety and liveness properties of the original protocol concerning one of its functions. Thus, their method reduces the analysis of a multifunction protocol to the analyses of several smaller single-function protocols. An application of their method to verify a version of the HDLC protocol is presented in [30].

This paper is concerned with the construction of a multifunction protocol from a composition of single-function protocols. In general, this is a difficult problem. However, many real-life protocols can be observed to go through different phases of behavior. In particular, these protocols go through their phases one at a time with a distinct function performed in each phase. For protocols characterized by this model of multiphase behavior, the following three-step methodology for constructing a
multifunction protocol is proposed:

(1) Divide the protocol's functionality into separate functions.

(2) Construct and verify a phase to perform each such function. (A phase, to be formally defined in Section 3 below, is a network of communicating finite state machines that satisfies certain general properties of correctness, including proper termination, and freedom from deadlocks and unspecified receptions.)

(3) Connect individual phases together to form the required protocol. The resulting protocol should satisfy the same general properties of correctness as the individual phases.

Step (1) of the above methodology is straightforward; a protocol's functions can often be divided quite naturally. For example, a half-duplex data link control protocol such as IBM's BSC protocol has three distinct functions [18,22]: a call setup function, a data transfer function, and a call clear function.

To carry out step (2) of the methodology, there are two basic approaches. In the first approach, each phase is constructed based on the designer's knowledge and experience. It is then verified using available verification techniques, e.g., the reachability analysis techniques of Bochmann [4], Rubin and West [28], Yu and Gouda [33,34], and Gouda and Yu [16], the proof methods of Good [12], Hailpern and Owicki [17], and Misra and Chandy [24,25], the symbolic execution method of Brand and Joyner [2], etc. If an error is found in a phase, the phase is modified and the verification repeated. This procedure goes on until a provably correct phase is obtained. In the second approach, each phase is constructed according to some design rules that automatically result in correct phases. See, for examples, Bochmann and Sunshine [5], Zaffirropulo et al. [35], Merlin and Bochmann [23], and Gouda and Yu [15].

Step (3) of the methodology has received little attention so far, although both Razouk and Estrin [26] and West and Zaffirropulo [32] observed that many errors in a protocol are caused by improper connections between different phases of the protocol. In this paper, we formally characterize the concept of a phase, and present a methodology to carry out step (3). We shall also demonstrate how some realistic
protocols can be constructed (and understood) using the three-step methodology.

The model of communicating finite state machines has been used successfully to model and analyze many existing protocols [4,14,26,32]. For simplicity, our results will be developed using such a model, although these results can be extended to other models as well.

This paper is organized as follows. In Section 2, the model of communicating finite state machines is presented. The concept of phases is formally defined in Section 3. The modeling of errors and timeouts is discussed in Section 4. Our method for constructing multiphase protocols is presented in Section 5; the construction method guarantees that the resulting multiphase protocol terminates properly and is free from deadlocks and unspecified receptions. In Section 6, we discuss a sufficient condition for multiphase protocols to be bounded. A version of IBM's BSC protocol for data link control [18,22] is used as a running example in Sections 3, 4, 5, and 6 for illustration. In Sections 7 and 8, we present two multiphase protocol examples, namely a token ring network protocol [1] and a modified BSC protocol with fair call connection. The advantages of our construction methodology are discussed in Section 8, and concluding remarks are in Section 9. In Appendix I, we present the method of closed covers that can be used to verify that a network of communicating finite state machines satisfies the properties of a phase. Proofs of all our theorems are in Appendix II.

2. Networks of Communicating Finite State Machines

A communicating finite state machine $M$ is a directed labelled graph with two types of edges, namely sending and receiving edges. A sending (or receiving) edge is labelled $-g$ (or $+g$, respectively) for some message $g$ in a finite set $G$ of messages. A node in $M$ whose outgoing edges are all sending (or all receiving) edges is called a sending (or receiving, respectively) node. A node in $M$ whose outgoing edges include both sending and receiving edges is called a mixed node, and a node in $M$ that has no outgoing edges is called a final node. One of the nodes in $M$ is identified as its initial node, and each node in $M$ is reachable by a directed path from the initial node.
Let $M$ and $N$ be two communicating finite state machines with the same set $G$ of messages; the pair $(M,N)$ is called a *network* of $M$ and $N$.

A *state* of network $(M,N)$ is a four-tuple $[v,w,x,y]$, where $v$ and $w$ are nodes in $M$ and $N$ respectively, and $x$ and $y$ are strings over the messages in $G$. Informally, a state $[v,w,x,y]$ means that the executions of $M$ and $N$ have reached nodes $v$ and $w$ respectively, while the input channels of $M$ and $N$ store the strings $x$ and $y$ respectively.

The *initial state* of network $(M,N)$ is $[v_0,w_0,E,E]$ where $v_0$ and $w_0$ are the initial nodes in $M$ and $N$ respectively, and $E$ is the empty string.

Let $s=[v,w,x,y]$ be a state of network $(M,N)$; and let $e$ be an outgoing edge of node $v$ or $w$. A state $s'$ is said to *follow* $s$ over $e$ iff exactly one of the following four conditions is satisfied:

- $e$ is a sending edge, labelled $-g$, from $v$ to $v'$ in $M$, and $s'=[v',w,x,y,g]$, where "." is the concatenation operator.

- $e$ is a sending edge, labelled $-g$, from $w$ to $w'$ in $N$, and $s'=[v,w',x,g,y]$.

- $e$ is a receiving edge, labelled $+g$, from $v$ to $v'$ in $M$, and $s'=[v',w,x',y]$, where $x=g.x'$.

- $e$ is a receiving edge, labelled $+g$, from $w$ to $w'$ in $N$, and $s'=[v,w',x,y']$, where $y=g.y'$.

Let $s$ and $s'$ be two states of network $(M,N)$, $s'$ *follows* $s$ iff there is a directed edge $e$ in $M$ or $N$ such that $s'$ follows $s$ over $e$.

Let $s$ and $s'$ be two states of $(M,N)$, $s'$ is *reachable from* $s$ iff $s=s'$ or there exist states $s_1,...,s_r$ such that $s=s_1$, $s'=s_r$ and $s_{i+1}$ follows $s_i$ for $i=1,...,r-1$.

A state $s$ of network $(M,N)$ is said to be *reachable* iff it is reachable from the initial state of $(M,N)$. Next, we use the concept of reachable states to define what it means for the communication of a network $(M,N)$ to terminate properly and to be free
from deadlocks and unspecified receptions, and to be bounded.

The communication of a network \((M,N)\) is said to terminate properly iff the following two conditions are satisfied:

- For any reachable state \([v,w,x,y]\) of \((M,N)\), if \(v\) is a final node of \(M\), then \(x\) must be the empty string and there must be a directed path of all receiving edges from node \(w\) to a final node \(w'\) in \(N\), such that the string \(y\) is received.

- For any reachable state \([v,w,x,y]\) of \((M,N)\), if \(w\) is a final node of \(N\), then \(y\) must be the empty string and there must be a directed path of all receiving edges from node \(v\) to a final node \(v'\) in \(M\), such that the string \(x\) is received.

A reachable state \([v,w,E,E]\) of \((M,N)\) is called a proper terminating state iff both node \(v\) and \(w\) are final nodes.

A reachable state \([v,w,x,y]\) of a network \((M,N)\) is a deadlock state iff (i) both \(v\) and \(w\) are receiving nodes, and (ii) \(x=E=y\) (the empty string). If no reachable state of network \((M,N)\) is a deadlock state, then the communication of \((M,N)\) is said to be deadlock-free.

A reachable state \([v,w,x,y]\) of a network \((M,N)\) is an unspecified reception state iff one of the following two conditions is satisfied:

- \(x=g_1g_2\ldots g_k\) (\(k\geq 1\)), and \(v\) is a receiving node and none of its outgoing edges is labelled \(+g_1\).

- \(y=g_1g_2\ldots g_k\) (\(k\geq 1\)), and \(w\) is a receiving node and none of its outgoing edges is labelled \(+g_1\).

If no reachable state of \((M,N)\) is an unspecified reception state, then the communication of \((M,N)\) is said to be free from unspecified receptions.

The communication of a network \((M,N)\) is said to be bounded by \(K\), where \(K\) is a nonnegative integer, iff for every reachable state \([v,w,x,y]\) of \((M,N)\), \(|x|\leq K\) and \(|y|\leq K\) where \(|x|\) is the number of messages in string \(x\). The communication is said to be bounded iff it is bounded by some nonnegative integer \(K\); otherwise it is unbounded.
3. Phases

Let $M$ and $N$ be two communicating finite state machines. The network $(M,N)$ is called safe iff its communication terminates properly and is free from deadlocks and unspecified receptions.

Let $(M,N)$ be a safe network, and let $v$ and $w$ be two final nodes in machines $M$ and $N$ respectively. The node pair $(v,w)$ is called an exit node pair of $(M,N)$ iff the state $[v,w,E,E]$ of $(M,N)$ is reachable.

The exit set of a safe network $(M,N)$ is the set of all exit node pairs of $(M,N)$.

A safe network $(M,N)$ is called a phase iff every final node in $M$ or $N$ appears in exactly one exit node pair in the exit set of $(M,N)$.

Is it decidable whether an arbitrary network is a phase? In general, the answer is negative as discussed by Brand and Zafiropulo [3]. However, the problem can be decided in some special cases: For instance, if the communication of the given network $(M,N)$ is bounded, then the problem can be decided by generating and checking all the reachable states of $(M,N)$. Further, we discuss a technique in Appendix I that can be used to verify that a given network is a phase even if the number of its reachable states is infinite. The technique is based upon the concept of closed covers of Gouda [13].

Example 1 (A Call Setup Phase): Consider the two communicating finite state machines $M_1$ and $N_1$ in Figure 1; they model the call setup procedure in the BSC protocol [18,22]: $M_1$ models the primary station, $N_1$ models the secondary station, and the messages have the following meanings:

- **ENQ** denotes an "enquiry" message.
- **ACKO** denotes an "affirmative acknowledgement" message.
- **NAK** denotes a "negative acknowledgement" message.
- **WACK** denotes a "temporarily not ready to receive" message.

Starting from node 1, if $M_1$ wants to set up a call with $N_1$, it sends an ENQ message to $N_1$ and waits at node 3. There are four possibilities:
- $N_1$ accepts the request with an ACK0 message; then each of $M_1$ and $N_1$ reaches its final node 7, and exits the Call Setup Phase.

- $N_1$ rejects the request with a NAK message; then each of $M_1$ and $N_1$ returns to node 1.

- $N_1$ replies with a WACK message, asking $M_1$ to try again later, then each of $M_1$ and $N_1$ returns to node 1.

- $M_1$ receives an ENQ message. This is a message collision situation implying that both machines want to set up a call. In this case, according to the BSC protocol, the primary $M_1$ is given priority to set up its call, and the secondary $N_1$ has to relinquish its request and decide whether or not it is ready to accept the call from $M_1$.

To show that the network $(M_1,N_1)$ is a phase, it is sufficient to prove the following three propositions:

i. The communication of $(M_1,N_1)$ terminates properly.

ii. None of the reachable states of $(M_1,N_1)$ is a deadlock state or an unspecified reception state.

iii. The exit set of $(M_1,N_1)$ is $\{(5,5),(6,6),(7,7)\}$, where each final node in $M_1$ or $N_1$ appears exactly once.

These three propositions can be proved by generating and examining all the reachable states of network $(M_1,N_1)$; there are 32 of them. Alternatively, we can prove that $(M_1,N_1)$ is a phase using the closed cover technique in Appendix I: It is straightforward to show that the set $\{[1,1,E,E],[5,5,E,E],[6,6,E,E],[7,7,E,E]\}$ is a closed cover for the network $(M_1,N_1)$; hence $(M_1,N_1)$ is a safe network by Theorem A1 of Appendix I. It is also straightforward to show that this closed cover satisfies the condition in Theorem A2 of Appendix I; therefore $(M_1,N_1)$ is a phase.

Example 2 (A Data Transfer Phase): Consider the two communicating finite state machines $M_2$ and $N_2$ in Figure 2; they model the data transfer procedure in BSC: $M_2$ models a sender, $N_2$ models a receiver, and the messages have the following
meanings:

D denotes a data message.
ACK0 denotes an "affirmative acknowledgement zero" message; it is used to acknowledge the reception of an odd-numbered message.
ACK1 denotes an "affirmative acknowledgement one" message; it is used to acknowledge the reception of an even-numbered message.
ENQ denotes an "enquiry" message.
WACK denotes a "temporarily not ready to receive" message.
RVI denotes a "reverse interrupt" message.
EOT denotes an "end of transmission" message.
Er is a special message that models a corrupted data message.
Ls is a virtual message that models a message loss.
Tm is a virtual message that models a time-out occurrence.

Starting from node 1, the sender M2 can send a data message to the receiver N2; it then waits at node 2. There are three possibilities:

- **The data message is correctly delivered:** This is modeled by M2 sending a data message D. The receiver acknowledges the reception with an ACK1 message; then the sender and receiver will each reach node 3. At node 2, the receiver may request the end of transmission by sending back WACK or RVI instead of ACK1. We discuss this feature later.

- **The data message is corrupted:** This is modeled by M2 sending a corrupted message Er. The receiver indicates this data corruption with an ACK0 message; then the receiver and sender will each return to node 1.

- **The data message is lost:** This is modeled by M2 sending a virtual message Ls and the receiver sending back a virtual message Tm. (The reception of Tm models the occurrence of a timeout event in the sender. These messages are not transmitted or received in reality.) The sender then sends an ENQ message to the receiver. The receiver responds with ACK0 since the data message has not been received. When the sender gets ACK0, both the sender and receiver are back at node 1.

The above mechanism models the delivery of odd-numbered data messages. Delivery of even-numbered data messages starts at node 3 in M2. The mechanism is similar to the above except that ACK0 is used to acknowledge the correct delivery of the data message, while ACK1 is used to acknowledge a message corruption.
After delivering its data messages, the sender $M_2$ (at node 1 or 3) can send an EOT message (indicating an end of transmission) to the receiver $N_2$; both sender and receiver will then exit the data transfer phase.

There are two ways by which the receiver can request the sender to terminate its transmission; one is via sending WACK messages; the other is via sending RVI messages. The difference between using WACK and RVI messages is as follows. By repeatedly sending WACK messages, the receiver prevents the progress of data transmission, and eventually forces the sender to send EOT. On the other hand, after sending RVI, the receiver is still ready to receive the next data message from the sender, and the data transmission can still proceed effectively.

To show that network $(M_2,N_2)$ is a phase, it is sufficient to prove the following three propositions:

i. The communication of $(M_2,N_2)$ terminates properly.

ii. None of the reachable states of $(M_2,N_2)$ is a deadlock state or an unspecified reception state.

iii. The exit set of $(M_2,N_2)$ is $\{(0,0),(10,10),(11,11),(12,12)\}$, where each final node in $M_2$ or $N_2$ appears exactly once.

These three propositions can be proved by generating and examining all the reachable states of $(M_2,N_2)$; there are 40 of them. Alternatively, we can prove that $(M_2,N_2)$ is a phase using the closed cover technique in Appendix I: It is straightforward to show that the following set is a closed cover for $(M_2,N_2)$:

$$\{[1,1,E,E],[2,2,E,E],[2,14,E,E],[2,13,E,E],[9,9,E,E],[3,3,E,E],[5,3,E,E],
[5,7,E,E],[7,7,E,E],[5,1,E,E],[4,4,E,E],[4,16,E,E],[4,17,E,E],[11,11,E,E],
[2,6,E,E],[2,18,E,E],[2,17,E,E],[2,15,E,E],[10,10,E,E],[2,5,E,E],[6,1,E,E],
[8,8,E,E],[6,8,E,E],[6,3,E,E],[4,5,E,E],[4,13,E,E],[4,14,E,E],[12,12,E,E],
[4,18,E,E],[4,6,E,E]\}.$$  

Moreover, this closed cover satisfies the condition in Theorem A2 of Appendix I. Therefore, $(M_2,N_2)$ is a phase.

Example 3 (A Call Clear Phase): Consider the two communicating finite state
machines $M_3$ and $N_3$ in Figure 3; they model the call clear procedure in BSC. It is
trivial to show that $(M_3, N_3)$ is a phase and that $S_3 = \{(2, 2)\}$ is its exit set.

Examples 1, 2, and 3 show the three basic phases of the BSC protocol for data
link control. In Section 5, we will show how to connect together five instances of these
three phases to form the BSC protocol.

4. Modeling Errors and Timeouts

In the phases presented in Section 3, we have followed a peculiarity of the BSC
protocol and assumed that only data messages, but not control messages, can be
corrupted or lost by the communication channels. In fact, BSC control messages are
very short, consisting of one or two control characters, and do not even have CRC
checksums for error detection [18]. (This is often cited as one of the weaknesses of BSC
[22].) The BSC manual does not specify recovery procedures for handling corruption or
loss of control messages. Since the intent of these examples is to illustrate the
multiphase protocol model, we decided to present them as they are described in the
manual and not to add our own versions of recovery procedures to them.

In general, if messages received can have undetected errors, there is no good
recovery procedure that we are aware of. If errors are always detected, timeouts can be
used to recover from the loss of messages due to errors. In the data transfer phase
presented in Section 3, we use virtual messages to model (simulate) the logical behavior
that a timeout for a data message occurs only if the message (or its acknowledgment) is
lost. This same trick can be extended to specify BSC phases that include timeouts for
recovery from the loss of control messages; in this case, the finite state machines shown
in Figures 1, 2 and 3 would be substantially larger.

Timeouts modeled as described above are said to be nonpremature timeouts. In a
recent technical report, Joseph, Raeuchle and Toueg [19] employed essentially the same
idea as ours to model nonpremature timeouts. Their model also allows the occurrence
of premature timeouts. However, they found that an alternating bit protocol can be
proved correct only if timeout occurrences are always nonpremature. Instead of simulating the occurrences of timeouts, Shankar and Lam [29,30] model timers and clocks explicitly in distributed systems. Given timers and clocks, timeout events and real-time constraints of protocol systems can be specified in a straightforward manner.

5. Constructing Multiphase Networks

In this section we discuss a discipline to connect a number of phases together to construct a multiphase network that is also a phase (thus guaranteeing that its communication terminates properly and is free from deadlocks and unspecified receptions). Phases are connected by joining the exit node pairs of one phase to the initial node pair of another phase, or the same phase.

Let \( p_1 = (M_1, N_1) \) and \( p_2 = (M_2, N_2) \) be two phases, with exit sets \( S_1 \) and \( S_2 \) respectively, and let \( C \) be a subset of \( S_1 \). We define a composite network of \( p_1, C, \) and \( p_2 \), denoted by \( <p_1, C, p_2> \), to be the network \( (M,N) \) where

- \( M \) is the communicating finite state machine constructed (from \( M_1, C, \) and \( M_2 \)) by joining all the final nodes of \( M_1 \) in \( C \) to the initial node of \( M_2 \). The initial node of \( M_1 \) becomes the initial node of \( M \).

- \( N \) is the communicating finite state machine constructed (from \( N_1, C, \) and \( N_2 \)) by joining all the final nodes of \( N_1 \) in \( C \) to the initial node of \( N_2 \). The initial node of \( N_1 \) becomes the initial node of \( N \).

The two phases \( p_1 = (M_1, N_1) \) and \( p_2 = (M_2, N_2) \) are called the constituent phases of the composite network \( <p_1, C, p_2> \). In this case, machines \( M_1 \) and \( M_2 \) are called the constituent machines of \( M \), and machines \( N_1 \) and \( N_2 \) are called the constituent machines of \( N \).

As an example, Figure 4(a) shows two phases \( p_1 = (M_1, N_1) \) and \( p_2 = (M_2, N_2) \). In phase \( p_1 \), the node pair \((1,1)\) is its initial node pair and \(\{(2,2),(3,3)\}\) is its exit set. In phase \( p_2 \) the node pair \((4,4)\) is its initial node pair and \(\{(5,5)\}\) is its exit set. By joining the exit node pair \((2,2)\) of \( p_1 \) to the initial node pair \((4,4)\), we have the composite network \( p_{1,2} = <p_1, \{(2,2)\}, p_2> \) shown in Figure 4(b).
Theorem 1: Let $p_1$ and $p_2$ be two phases, with exit sets $S_1$ and $S_2$ respectively, and let $C$ be a subset of $S_1$. Then, the composite network $<p_1, C, p_2>$ is a phase whose exit set is $(S_1 \cup S_2) - C$.

By theorem 1, network $p_{1,2}$ in Figure 4(b) is also a phase whose exit set is $\{(3,3),(5,5)\}$.

So far we have discussed how to connect one phase to another. Next, we discuss how to connect a phase to itself.

Let $p_1=(M_1,N_1)$ be a phase whose exit set is $S_1$, and let $C$ be a subset of $S_1$. The composite network of $p_1$ and $C$, denoted $<p_1,C>$, is a network $(M,N)$ where

- $M$ is the communicating finite state machine constructed (from $M_1$ and $C$) by joining all the final nodes of $M_1$ in $C$ to the initial node of $M_1$. The initial node of $M_1$ becomes the initial node of $M$.

- $N$ is the communicating finite state machine constructed (from $N_1$ and $C$) by joining all the final nodes of $N_1$ in $C$ to the initial node of $N_1$. The initial node of $N_1$ becomes the initial node of $N$.

Phase $p_1=(M_1,N_1)$ is called the constituent phase of the composite network $<p_1,C>=(M,N)$. In this case, machines $M_1$ and $N_1$ are called the constituent machines of $M$ and $N$ respectively.

For example, consider phase $p_{1,2}$ in Figure 4(b), if we join the exit node pair $(5,5)$ of $p_{1,2}$ to its initial node pair, then we get the composite phase $<p_{1,2},\{(5,5)\}>$ shown in Figure 4(c).

Theorem 2: Let $p$ be a phase whose exit set is $S$, and let $C$ be a subset of $S$. Then, the composite network $<p,C>$ is a phase whose exit set is $S - C$.

By theorem 2, network $<p_{1,2},\{(5,5)\}>$ in Figure 4(c) is also a phase whose exit set is $\{(3,3)\}$.
The process of constructing the multiphase network $p^*$ in Figure 4(c) from the two phases $p_1$ and $p_2$ in Figure 4(a) can be represented by the following sequence of equations:

\[
\begin{align*}
p_1 &= (M_1, N_1) \\
p_2 &= (M_2, N_2) \\
p_{1,2} &= <p_1, \{(2,2)\}, p_2> \\
p^* &= <p_{1,2}, \{(5,5)\}>
\end{align*}
\]

This equation sequence clearly provides all the information needed to construct $p^*$ from $p_1$ and $p_2$; moreover it is a more concise notation than the graphical representations in Figures 4(b) and 4(c).

**Example 4 (A BSC Protocol):** Figure 5 shows a version of the BSC protocol [18,22] modeled as a composite network that consists of five phases (namely one call setup phase, two data transfer phases, and two call clear phases). The constituent phases are those defined in Section 3; they are represented in Figure 5 by their initial and exit nodes only. The dash lines identify the individual phases. The bold lines show how the phases are connected. Machine M models the primary station, and machine N models the secondary station of BSC.

An equation sequence that specifies the construction sequence of this version of BSC is as follows:

\[
\begin{align*}
p_1 &= (M_1, N_1) \\
p_2 &= (M_2, N_2) \\
p_3 &= (N_2, M_2) \\
p_4 &= (N_3, M_3) \\
p_5 &= (M_3, N_3) \\
p_{1,2} &= <p_1, C_{1,2}> \\
p_{1,2,3} &= <p_{1,2}, C_{2,3}> \\
p_{1,2,3,4} &= <p_{1,2,3}, C_{3,4}> \\
p_{1,2,3,4,5} &= <p_{1,2,3,4}, C_{4,5}> \\
p &= <p_{1,2,3,4,5}, C_5>
\end{align*}
\]

where $M_1$ and $N_1$ are defined in Figure 1

$M_2$ and $N_2$ are defined in Figure 2.
$M_3$ and $N_3$ are defined in Figure 3

$C_1=\{(6,9) \text{ in } p_1, (7,7) \text{ in } p_1\}$

$C_2=\{(5,5) \text{ in } p_1\}$

$C_3=\{(9,9) \text{ in } p_2, (10,10) \text{ in } p_2, (11,11) \text{ in } p_2, (12,12) \text{ in } p_2\}$

$C_4=\{(9,9) \text{ in } p_3, (10,10) \text{ in } p_3, (11,11) \text{ in } p_3, (12,12) \text{ in } p_3\}$

$C_5=\{(2,2) \text{ in } p_4, (2,2) \text{ in } p_5\}$

where $(i,j) \text{ in } p_k$ = the node pair $(i,j)$ in phase $p_k$.

Notice that this equation sequence is not unique for constructing the BSC protocol in Figure 5.

From Theorems 1 and 2, the communication of the composite network $(M,N)$ in Figure 5 is free from deadlocks and unspecified receptions. In the next section, we shall show that this communication is also bounded.

6. Boundedness of Multiphase Networks

In this section we present a sufficient condition for the communication of a composite network to be bounded provided that all its constituent phases are bounded. Before doing so, some definitions are in order.

Let $(M,N)$ be a composite network, and let $M_i$ be a constituent machine in $M$. A final node in $M_i$ is called a plus node iff all its incoming edges are receiving edges. A final node in $M_i$ is called a minus node iff all its incoming edges are sending edges. A final node in $M_i$ is called a zero node iff its incoming edges include both sending and receiving edges.

Let $(M,N)$ be a composite network, and assume that machine $M$ consists of $r$ ($r \geq 1$) constituent machines $M_1,M_2,...,M_r$. The abstract machine $\tilde{M}$ of $M$ is a directed labelled graph constructed from $M$ as follows:

i. For each constituent machine $M_i$ in $M$, add a node labelled $M_i$ to $\tilde{M}$.

ii. If only plus nodes of a constituent machine $M_i$ are joined with the initial node of some constituent machine $M_j$ (which may be the same as $M_i$), then
add a directed edge, labelled +, from node $M_i$ to node $M_j$ in $\tilde{M}$.

iii. If only minus nodes of a constituent machine $M_i$ are joined with the initial node of some constituent machine $M_j$ (which may be the same as $M_i$), then add a directed edge, labelled -, from node $M_i$ to node $M_j$ in $\tilde{M}$.

iv. If the nodes of a constituent machine $M_i$, joined with the initial node of some constituent machine $M_j$ (which may be the same as $M_i$), include one zero node or include both plus nodes and minus nodes, then add two directed edges, one labelled -, the other labelled +, from node $M_i$ to node $M_j$ in $\tilde{M}$.

A directed edge labelled + (or -) in $\tilde{M}$ is called a plus (or minus) edge.

As an example, Figure 6 shows the abstract machine of the communicating machine $M$ in Figure 5.

**Theorem 3:** Let $(M,N)$ be a composite network whose constituent phases are all bounded; and let $\tilde{M}$ be the abstract machine of $M$. If each directed cycle in $\tilde{M}$ has at least one plus edge and one minus edge, then the communication of $(M,N)$ is bounded.

$\square$

Notice that $\tilde{M}$ satisfies this condition iff $\tilde{N}$ satisfies the same condition; hence checking one abstract machine is sufficient.

Assume that the communication of $(M,N)$ is known to be bounded after checking the condition in Theorem 3. From the proof of Theorem 3 (in Appendix II), the communication of $(M,N)$ is bounded by $K = \sum_{i=1}^{r} K_i$ where $K_1,...,K_r$ are the communication bounds for the $r$ constituent phases of $(M,N)$. $K$ is not necessarily a tight communication bound of $(M,N)$; it is merely an upper bound. A tighter upper bound can be obtained by executing the following four steps on the abstract machine $\tilde{M}$ of $M$:

i. Label each node $M_i$ in $\tilde{M}$ with the communication bound of its corresponding
constituent phase.

ii. Remove all plus edges from \( \tilde{M} \). (The resulting graph is acyclic by Theorem 3.) An upper bound \( m \) for the number of messages in the output channel of \( M \) is the length of the longest directed path in the modified \( \tilde{M} \), where the length of a path is computed by adding all the labels of its nodes.

iii. From the original \( \tilde{M} \), remove all minus edges from \( \tilde{M} \). (The resulting graph is acyclic by Theorem 3.) An upper bound \( n \) for the number of messages in the output channel of \( N \) is the length of the longest directed path in the modified \( \tilde{M} \), where the length of a path is computed by adding all the labels of its nodes.

iv. The communication bound of \((M,N) \leq \max(m,n)\).

A correctness proof that the above four steps give a communication bound for \((M,N)\) is similar to that of Theorem 3.

Executing these steps on the abstract machine \( \tilde{M} \) in Figure 6, we found that the communication of the BSC protocol in Figure 5 is bounded by 5. The actual bound for this network is 3; therefore the upper bound computed by the above four steps is still not very tight.

7. A Token Ring Protocol Example

The method of phases and the theorems in this paper can be extended in a straightforward manner to networks with \( n \) (\( n \geq 2 \)) communicating finite state machines. For example, a high-level session control protocol modeled after one in IBM's Systems Network Architecture[11] can be constructed as a multiphase network of three machines[9]. This method can also be extended to networks whose topology is characterized by one or more parameters. As an example, we construct in this section a token ring protocol as a multiphase network of \( n \) machines, where \( n \) is a parameter.

Consider a ring network of \( n \) communicating finite state machines \( M_0, M_1, \ldots, M_{n-1} \) which communicate via \( n \) unidirectional channels as shown in Figure 7. Clearly, each
machine $M_i$ receives messages only from its upstream neighbor $M_{(i-1) \mod n}$ and sends messages only to its downstream neighbor $M_{(i+1) \mod n}$. The communication protocol can be defined as follows:

i. When a machine has the token, it can send its data messages, one by one, downstream.

ii. When a machine $M_j$ receives a data message (generated by $M_i$, $i \neq j$) from its upstream neighbor, it examines the message to decide whether it wants to keep a copy of it. The message is then sent to its downstream neighbor.

iii. A data message generated by $M_i$ is subsequently removed by $M_i$ after the message has travelled once around the ring.

iv. When $M_i$ has removed all its data messages from the ring, and has no more data message to send, it sends the token to its downstream neighbor.

This protocol can be viewed as consisting of $n$ phases $p_0, p_1, \ldots, p_{n-1}$, where phase $p_i$ defines the communication among the $n$ machines when machine $M_i$ has the token. For example, the $n$ machines $M_0^0, M_1^0, \ldots, M_{n-1}^0$ in phase $p_0$ can be defined as shown in Figure 8(a), where the messages have the following meanings:

\begin{align*}
D & \text{ denotes a data message.} \\
T & \text{ denotes the token.}
\end{align*}

Similarly, the $n$ machines in phase $p_1$ are shown in Figure 8(b).

Two comments concerning $p_0$ are in order:

i. To prove that network $p_0$ is indeed a phase whose exit set is $\{[3,6,7,7,\ldots,7]\}$, one can use induction over $n$. Moreover, since each of the networks $p_1, p_2, \ldots,$ and $p_{n-1}$ is identical to $p_0$ (except for the order of the machines in the network), this same inductive proof shows that each of $p_1, p_2, \ldots,$ and $p_{n-1}$ is also a phase.

ii. Phase $p_0$ has one "exit tuple" namely $[3,6,7,7,\ldots,7]$, where nodes labelled 7 are not final nodes; rather they are receiving nodes. This requires a slight extension to the definition of exit node pairs (or tuples). For this extension to be valid (i.e., for the results in Theorems 1 and 2 to continue to hold), the receiving nodes in an exit tuple can be joined only with initial nodes that are receiving nodes when we construct composite networks.
To construct the composite network \((M_0, M_1, \ldots, M_{n-1})\) from these phases, we need to join all the \(M_0^i\) machines to form \(M_0\), join all the \(M_1^i\) machines to form \(M_1\), and so on. For example, Figure 9(a) shows the construction of \(M_0\) from its constituent machines \(M_0^i\), \(i=0,\ldots,n-1\). (In Figure 9(a), an undirected edge between two nodes means that these two nodes should be joined into one.) The resulting \(M_0\), after joining nodes, is shown in Figure 9(b). Similarly, the resulting \(M_j\) \((j=1,\ldots,n-1)\), after joining nodes, is shown in Figure 9(c). We have assumed, without any loss of generality, that initially \(M_0\) has the token.

8. Achieving Modularity

This paper presents a method for the modular construction of protocols. First, individual phases are constructed. Each phase is verified to satisfy certain desirable properties. Second, phases are connected together using the method described in Section 4. The resulting protocol is guaranteed to terminate properly and to be free from deadlocks and unspecified receptions. Under some additional conditions, the resulting protocol is also bounded.

Advantages of this construction methodology are as follows.

i. **Ease of construction and reasoning:** The methodology allows us to focus on one phase of a complex protocol at a time. By ensuring that each phase satisfies some desirable properties, we are guaranteed that the phases can be later connected together to form a multiphase protocol with the same desirable properties.

ii. **Parallel construction and verification:** Construction and verification of the different phases of a protocol can proceed independently and hopefully in parallel.

iii. **Flexibility for modifying a phase:** After constructing a protocol by connecting a number of phases together, it is possible to modify one of the phases without affecting the others. This is done by preserving the exit set in the modified phase. As an example, consider the data transfer phase in Figure 2. Assume that each sending edge labelled \(-D\) in \(M_2\) is replaced by the structure in Figure 10(a) whose message labels have the following meanings:

SOH denotes a "start-of-header" message.
H denotes a byte in the header.
STX denotes a "start-of-text" message.
T denotes a text byte.
ETX denotes an "end-of-text" message followed by a check sum.

Assume also that each receiving edge labelled +D in N₂ is replaced by the structure in Figure 10(b). The resulting network (M₂',N₂') is a phase, and has the same exit set as (M₂,N₂). Therefore, (M₂',N₂') can replace (M₂,N₂) in any composite protocol. (Notice however that the communication of (M₂',N₂') is unbounded and so the resulting composite protocol is unbounded.)

iv. Flexibility for rearranging phases: After constructing a protocol by connecting a number of phases together, it is possible to add more copies of the existing phases and rearrange the connections between phases to make the protocol satisfy some additional desirable properties (fairness, robustness, etc.). For example the BSC protocol in Figure 5 is unfair. This is because whenever the primary M and the secondary N compete to become the sender, the primary M always wins. This unfairness is intentional in the original BSC protocol [18,22]. It is possible to make this protocol fair by adding one call setup phase, four data transfer phases, and four call clear phases and rearranging the phase connections to obtain the protocol (Mₙ,Nₙ) as shown in Figure 11. (In Figure 11, each rectangular node labelled Mᵢ represents machine Mᵢ in some phase. Also, each directed edge from node Mᵢ to Mₖ means that all the exit nodes of Mᵢ are joined with the initial node of Mₖ. This convention, however, is not followed in the case of M₁ and N₁ where each of their final nodes is joined with the initial node of a different machine.)

Notice that initially M₁ behaves as a primary and N₁ behaves as a secondary. They change roles each time after they compete to become the sender and the primary wins.

v. Efficient validation: As demonstrated by the BSC protocol example, many copies of the same phase may be used in a protocol. The method of phases requires such a phase to be validated only once regardless of how many copies of it are used in the protocol. Table 1 shows the number of generated states and the required execution time to validate the BSC protocol (in Figure 5) and the modified BSC protocol (in Figure 11). By using the method of phases, the number of generated states is reduced by factors of 1.5 and 4.0 respectively, and the execution time is reduced by factors of 1.4 and 4.8 respectively. (Notice that these gains are accomplished without relying on any parallel validation discussed in ii.)
9. Concluding Remarks

We have described a methodology for constructing large multiphase communication protocols, and demonstrated that this methodology can be used to construct (and understand) some realistic protocols. The protocols constructed are guaranteed to terminate properly and to be free from deadlocks and unspecified receptions. In addition to the examples presented in this paper, we have also shown that a session management protocol modeled after one in IBM's System Network Architecture [9] and the call establishment/clear protocol of X.25 [10] can be constructed as multiphase protocols.

Although the multiphase concept and our construction methodology have been developed using the model of communicating finite state machines, it should be straightforward to extend the results herein to facilitate protocol construction using other models as well.

Our methodology can be viewed as a bottom-up approach to the protocol construction problem. A top-down approach for protocol construction was recently proposed by Gouda [14]. Both approaches need to be examined and compared so that the protocol construction problem can be better understood. An integrated approach, which employs both bottom-up and top-down strategies, for protocol construction seems attractive.

In the BSC data transfer phase, we have demonstrated the use of virtual messages to model message losses and nonpremature timeouts. This technique seems promising, and using it we have managed to specify several protocols to our satisfaction. In reality, protocol systems are prone to other types of errors (e.g., reordering of messages), as well as premature timeouts and crashes. Further research is needed to develop specification and verification techniques for these problems.

We are currently developing an interactive protocol design system to support, among other things, the multiphase construction methodology herein and the projection method in [21]. The system is called PROSPEC. It is implemented in C, runs on a
SUN workstation, and has a multi-window graphical interface.

After writing this paper, it came to our attention that Raymond Miller and Tat Y. Choi at Georgia Institute of Technology had independently obtained results [7,8] similar to those reported in this paper. They have also applied the multiphase concept to simplify the analysis of certain protocols. The reader can find in their work another interesting multiphase protocol example, namely the call establishment/clear protocol of X.21.

APPENDIX I: USING CLOSED COVERS TO PROVE PHASES

The technique of closed covers has been proposed in [13] to prove that the communication of a network \((M,N)\), where \(M\) and \(N\) have no final nodes, is free from deadlocks and unspecified receptions. Here we extend this technique to prove that a network \((M,N)\), where \(M\) and \(N\) may have final nodes, is a phase.

Let \(M\) and \(N\) be two communicating finite state machines possibly with final nodes. A closed cover \(C\) of network \((M,N)\) is a finite set of state schemas \([v_1,w_1,x_1,y_1],..., [v_r,w_r,x_r,y_r]\) such that the following three conditions are satisfied.

i. For each state schema \([v,w,x,y]\) in \(C\), \(v\) is a node in \(M\), \(w\) is a node in \(N\), and \(x\) and \(y\) are two sets of (possibly infinite) message sequences. Each state schema can be viewed as a set of network states. A state \([v,w,x,y]\) is in some state schema \([v,w,x,y]\) of \(C\) iff the message sequences \(x\) and \(y\) are in sets \(X\) and \(Y\) respectively.

ii. The initial state of \((M,N)\) is in some state schema of \(C\).

iii. For every state \(s\) in some state schema of \(C\), there exist two states \(s'\) and \(s''\) such that

- \(s''\) is in a state schema of \(C\), and
- either \((s'\) follows over an edge in \(M\) and \(s''\) follows \(s'\) over an edge in \(N\)) or \((s'\) follows over an edge in \(N\) and \(s''\) follows \(s'\) over an edge in \(M\)).

iv. if \([v,w,x,y]\) is in \(C\) where \(v\) (w) is a final node, then \(w\) (v) is a final node and \(X=Y=E\) (the empty string).
Theorem A1: If a network \((M,N)\) has a closed cover \(C\), then \((M,N)\) is a safe network. (The proof is in Appendix II.)

From Theorem A1, the existence of a closed cover \(C\) for a network \((M,N)\) guarantees that the communication of \((M,N)\) terminates properly and is free from deadlocks and unspecified receptions. In order to guarantee that \((M,N)\) is a phase, an additional condition on its closed cover \(C\) is needed.

Theorem A2: Let \((M,N)\) be a network whose closed cover \(C\) satisfies the following condition. For any final node \(v\) \((w)\) in \(M\) \((N)\), there exists exactly one final node \(w\) \((v)\) in \(N\) \((M)\) such that \([v,w,E,E]\) is in \(C\). Then, \((M,N)\) is a phase. (The proof is in Appendix II.)

Example A: Consider the two communicating finite state machines \(M\) and \(N\) in Figure A1; they model a full-duplex data transfer procedure with flow control. The exchanged messages have the following meanings:

- \(F\) denotes a "flow control" message.
- \(D\) denotes a data message.
- \(A\) denotes an "acknowledgment" message.
- \(D\) denotes a "disconnect" message.

The following set \(C\) of state schemas of network \((M,N)\) is an infinite closed cover for \((M,N)\):

\[
C = \{[1,1,F^n,F^n],[2,1,F^{n+1},F^nD],[1,2,F^nD,F^{n+1}],[2,3,E,E],[3,2,E,E],
[2,2,F^nD,F^nD],[2,2,A,A],[5,6,E,E],[6,5,E,E]\}.
\]

The first state schema \([1,1,F^n,F^n]\) represents the following infinite set of states \([1,1,E,E],[1,1,F,F],[1,1,F^2,F^2],\ldots\) where \(E\) denotes the empty string, and each state has an equal number of \(F\) messages in its two channels. The other state schemas should be interpreted in the same way; in particular, the schema \([2,3,E,E]\) represents the set \([2,3,E,E]\). It is straightforward to check that the set \(C\) satisfies the conditions of a closed cover; hence \(C\) is a closed cover for network \((M,N)\). Moreover, this closed cover
satisfies the condition of Theorem A2; hence \((M,N)\) is a phase. (Notice that the communication of \((M,N)\) is unbounded, and so network \((M,N)\) cannot be proven to be a phase using state exploration.)

\[
\square
\]

**APPENDIX II: PROOFS OF THEOREMS**

**Proof of Theorem 1:**

Let \(p_1=(M_1,N_1)\), \(p_2=(M_2,N_2)\), and \((M,N)\)=\(<p_1,C,p_2>\). To prove that \((M,N)\) is a phase, we first prove that \((M,N)\) is a safe network; i.e., it terminates properly and is free from deadlocks and unspecified receptions; then, we prove that \((S_1 - C) \cup S_2\) is its exit set where every final node of \(M\) or \(N\) appears exactly once. We begin by proving that \((M,N)\) terminates properly. (The proof is by contradiction. The proofs for freedom of deadlocks and unspecified receptions are similar.)

Assume that \(s=[v,w,x,y]\) is a reachable improper terminating state of \((M,N)\). Then there are four cases to consider:

i. \(v\) is a final node and \(x \neq E\), where \(E\) is the empty string.

ii. \(w\) is a final node and \(y \neq E\).

iii. \(v\) is a final node and \(x = E\), but there does not exist a directed path of all receiving edges from node \(w\) to a final node \(w'\) in \(N\) where the string \(y\) is received.

iv. \(w\) is a final node and \(y = E\), but there does not exist a directed path of all receiving edges from node \(v\) to a final node \(v'\) in \(N\) where the string \(x\) is received.

We prove that case i leads to a contradiction. (Similar proofs can show that cases ii, iii, and iv also lead to contradictions.)

Let \(\alpha\) and \(\beta\) be the two directed paths in \(M\) and \(N\) respectively that lead \((M,N)\) from its initial state to state \([v,w,x,y]\). There are four cases to consider.

**Case i** (All edges of path \(\alpha\) are in \(M_1\), and all edges of path \(\beta\) are in \(N_1\)).

This implies that \((M_1,N_1)\) does not terminate properly; hence it contradicts the assumption that \((M_1,N_1)\) is a phase.
Case ii (Path $\alpha$ ends at a final node $v$ in $M_2$ and path $\beta$ ends at node $w$ in $N_2$).

Let $(v,w')$ be an exit node pair of $(M_2,N_2)$. One of the nodes in $\alpha$ must be a final node $v^*$ of $M_1$, which appears in exactly one exit node pair $(v^*,w^*)$ in $C$. Assume that node $w \neq w^*$. (The proof for the case $w = w^*$ is similar.) The network $(M,N)$ must have reached state $[v^*,w^*,v^*,w^*] \rightarrow [v,w,x,y]$. Since $(M_1,N_1)$ terminates properly, $x^* = E$ and machine $N_1$ can only follow a directed path of all receiving edges from node $w^*$ via node $w$ to node $w^*$, where $y^*$ is received. It implies that the input channel of $M$ is kept empty from state $[v^*,w^*,v^*,y^*]$ to state $[v,w,x,y]$, in other words $x = E$. Therefore in machine $M$, the path from node $v^*$ (for that matter the initial node of $M_2$) to node $v$ must be a path of all sending edges. Let $y_2$ be the string sent by $M_2$ from node $v^*$ to node $v$. Since $(M_2,N_2)$ terminates properly, there exists a directed path of all receiving edges from the initial node of $N_2$ to node $w'$, where string $y_2$ is received. Let $y = y_1 y_2$. It is easy to see that there exists a directed path of all receiving edges from node $w$ via node $w^*$ to node $w'$ in $N$, where string $y$ is received. This contradicts the assumption that condition i is true.

Case iii (All edges of path $\alpha$ are in $M_1'$, and path $\beta$ ends at node $w$ in $N_2$).

Using a similar argument to that of case ii, it can be shown that this case also leads to a contradiction.

Case iv (Path $\alpha$ ends at a final node $v$ in $M_2$, and path $\beta$ ends at a node $w$ in $N_2$).

Path $\alpha$ should have a node $v^*$ and path $\beta$ should have a node $w^*$, where $(v^*,w^*)$ is an exit node pair of $(M_1,N_1)$. By the definition of proper termination, each message sent along path $\alpha$ before node $v^*$ should be received along path $\beta$ before node $w^*$, and each message sent along path $\beta$ before node $w^*$ should be received along path $\alpha$ before node $v^*$. Since the exit node pair $(v^*,w^*)$ of $(M_1,N_1)$ joins the initial node pair $(v_0,w_0)$ of $(M_2,N_2)$, the state $[v,w,x,y]$ can be reached from the state $[v_0,w_0,E,E]$ of $(M_2,N_2)$. This implies that $(M_2,N_2)$ does not terminate properly; hence, it contradicts the assumption that $(M_2,N_2)$ is a phase.
This completes the proof that \((M,N)\) terminates properly. Similar arguments can be used to prove that \((M,N)\) is free from deadlocks and unspecified receptions. Therefore, \((M,N)\) is a safe network.

To show that \((S_1 - C) \cup S_2\) is the exit set of \((M,N)\), it is necessary and sufficient to show that (i) each node pair in \((S_1 - C) \cup S_2\) is an exit node pair of \((M,N)\), and (ii) each exit node pair of \((M,N)\) is in \((S_1 - C) \cup S_2\).

i. Each node pair in \((S_1 - C) \cup S_2\) is an exit node pair of \((M,N)\).

For each node pair \((v,w)\) in \(S_1\), \([v,w,E,E]\) of \((M,N)\) is reachable, since \([v,w,E,E]\) of \((M_1,N_1)\) is reachable and \((M_1,N_1)\) is a subgraph of \((M,N)\). Let \((v^*,w^*)\) be an exit node pair in \(C\). \([v^*,w^*,E,E]\) of \((M,N)\) is reachable, since \(C\) is a subset of \(S_1\). Since \((v^*,w^*)\) joins the initial node pair \((i_1,i_2)\) of \((M_2,N_2)\), \([i_1,i_2,E,E]\) and \([v^*,w^*,E,E]\) are the same states. Therefore, \([i_1,i_2,E,E]\) of \((M,N)\) is reachable. For any node pair \((v,w)\) in \(S_2\), \([v,w,E,E]\) of \((M,N)\) is reachable from \([i_1,i_2,E,E]\) because \((M_2,N_2)\) is a subgraph of \((M,N)\). Since \([i_1,i_2,E,E]\) is reachable, \([v,w,E,E]\) is reachable. Therefore, by definition of exit node pair, the node pairs of \((S_1 - C) \cup S_2\) are exit node pairs of \((M,N)\).

ii. Each exit node pair \((v,w)\) of \((M,N)\) is in \((S_1 - C) \cup S_2\).

If both \(v\) and \(w\) are in subgraph \((M_1,N_1)\) of \((M,N)\), \((v,w)\) must be in \(S_1 - C\); otherwise, it contradicts the assumption that \((M_1,N_1)\) is a phase. For the same reasons, if \(v\) and \(w\) are both in subgraph \((M_2,N_2)\) of \((M,N)\), then \((v,w)\) must be in \(S_2\). It is impossible that one node of \(v\) and \(w\) is in \((M_1,N_1)\) and the other node is in \((M_2,N_2)\), since it implies that \((M_1,N_1)\) and \((M_2,N_2)\) do not terminate properly. Therefore no other node pair \((v,w)\) of \((M,N)\) can be an exit node pair of \((M,N)\).

This complete the proof that \((S_1 - C) \cup S_2\) is the exit set of the safe network \((M,N)\). Clearly, every final node in \(M\) or \(N\) appears in exactly one exit node pair in the exit set of \((M,N)\). Therefore \((M,N)\) is a phase.

\(\square\)

Proof of Theorem 2:

Let \(p=(M,N)\) and \(<p,C>=(M^*,N^*)\). As illustrated in Figure A2, \((M^*,N^*)\) is equivalent to an infinite chain of identical phases, \(p_1,p_2,...\), connected by joining the exit node pairs of \(C_i\) to the initial node pair of \(p_{i+1}\), \((i \geq 1)\), such that:
i. each phase \( p_i \) is isomorphic to phase \((M,N)\), and

ii. each \( C_i \) is isomorphic to \( C \).

Notice that the first \( n \) phases in this chain, along with the connections \( C_{1 \ldots} \ldots C_{n-1} \) constitute a phase (by theorem 1.) We denote this phase \((M^n,N^n)\).

Now, to show that \((M^*,N^*)\) is a phase, we first prove, by contradiction, that \((M^*,N^*)\) terminates properly. Assume that \((M^*,N^*)\) does not terminate properly; i.e., it can reach a state \( s \) which satisfies condition i, ii, iii, or iv in the proof of theorem 1. Let paths \( \alpha \) and \( \beta \) be the two paths that lead the execution of \((M^*,N^*)\) from initial state to state \( s \). Assume that \( \alpha \) ends at a node in machine \( M_i \), and that \( \beta \) ends at a node in machine \( N_j \); without loss of generality, assume that \( i \) is greater than or equal to \( j \). Since both paths \( \alpha \) and \( \beta \) are in \((M^i,N^i)\), the network \((M^i,N^i)\) can reach state \( s \). This contradicts the fact that \((M^i,N^i)\) is a phase. Therefore \((M^*,N^*)\) must terminate properly. By similar arguments, we can prove that \((M^*,N^*)\) is free from deadlocks and unspecified receptions. This completes the proof that \((M^*,N^*)\) is a safe network.

We now prove that \( S - C \) is the exit set of \((M^*,N^*)\). Referring to the infinite chain of phases in Figure A2, let \( S_1, S_2, \ldots \) be the exit sets for phases \( p_1, p_2, \ldots \) respectively. Since each \( p_i \) is isomorphic to \((M,N)\), then select each \( S_i \) to be isomorphic to the exit set \( S \) of \((M,N)\) and select each \( C_i \) to be isomorphic to \( C \). From Theorem 1, \( (S_1 - C_1) \cup (S_2 - C_2) \cup \ldots \cup (S_n - C_n) \) is the exit set for \((M^n,N^n)\), for any \( n \). By folding the infinite chain into \((M^*,N^*)\), the exit set of \((M^*,N^*)\) becomes \( S - C \).

It is easy to see that every final node in \( M^* \) or \( N^* \) appears in exactly one exit node pair in the exit set of \((M^*,N^*)\). Therefore \((M^*,N^*)\) is a phase. \( \square \)

**Proof of Theorem 3:**

Let \( K_1, \ldots, K_r \) be the communication bounds for the constituent phases of \((M,N)\), and assume that each directed cycle in \( \tilde{M} \) has at least one plus edge and one minus
edge. We prove, by contradiction, that the communication of \((M,N)\) is bounded by \(K = \sum_{i=1}^{r} K_i\).

Assume that there exists a reachable state \([v,w,x,y]\) where \(|y| = K+1\); i.e., 
\(y = y_1\ldots y_{K+1}\). Assume that the messages in \(y\) are sent during the execution of a sequence of phases, \(p_1,\ldots,p_n\). Since \(|y| > \sum_{i=1}^{r} K_i\), at least two of these phases in the phase sequence must be the same. Without loss of generality, assume that phase \(p_1\) occurs twice and the phase sequence becomes \(p_1,\ldots,p_m,p_1,\ldots,p_n\). Let \(p_i = (M_i,N_i), i=1,\ldots,n\). Since phase \(p_1\) occurs twice in the phase sequence, it implies that there exists a directed cycle in \(\widetilde{M}\). Let \(f_i\) be the last directed edge that is executed by machine \(M_i, i=1,\ldots,m\).

According to the assumption that each directed cycle of \(\widetilde{M}\) has at least one plus edge and one minus edge, one edge in the edge sequence, \(f_1,\ldots,f_m\) must be a receiving edge. (See Figure A3 for the relationship between string \(y\), the phase sequence and the edge sequence.) By the definition of proper termination, the following proposition is true.

Let \(p = (M,N)\) be a phase. If the last executed edge \(f\) of a machine \(M\) is a receiving edge, then all the messages sent by machine \(M\) in phase \(p\) must have already been received by machine \(N\) when the receiving edge \(f\) is executed.

Based on this proposition, if edge \(f_i\) is a receiving edge, then all the messages in \(y\), which have been sent out before \(f_i\) is executed, must have already been received by machine \(N\) when \(f_i\) is executed. Therefore, \(|y| \neq K+1\) since some of the messages in \(y\) must have already been received; contradiction.

\(\square\)

**Proof of Theorem A1:**

Let \((M,N)\) be a network with a closed cover \(C\). From [13], the communication of \((M,N)\) is free from deadlocks and unspecified receptions. It remains now to show that the communication of \((M,N)\) terminates properly. The proof is by contradiction.

Assume that \(s = [v,w,x,y]\) is a reachable improper terminating state of \((M,N)\). Then there are four cases to consider:
i. $v$ is a final node and $x \not\in E$.

ii. $v$ is a final node and $y \not\in E$.

iii. $v$ is a final node and $x = E$, but there does not exist a directed path of all receiving edges from node $w$ to a final node $w'$ in $N$ where the string $y$ is received.

iv. $w$ is a final node and $y = E$, but there does not exist a directed path of all receiving edges from node $v$ to a final node $v'$ in $N$ where the string $x$ is received.

We prove that case i leads to a contradiction. (Similar proofs can show that cases ii, iii, and iv also lead to contradictions.)

Since $s = [v, w, x, y]$ is reachable, there is a sequence $s_0, s_1, \ldots, s_r$ of reachable states of $(M, N)$ such that $s_0$ is the initial state, $s_r = s$, and for $i = 0, 1, \ldots, r-1$, $s_{i+1}$ follows $s_i$. This state sequence corresponds to two directed paths $\alpha$ and $\beta$ in machines $M$ and $N$ respectively, such that the following condition is satisfied: Path $\alpha$ ($\beta$) starts with the initial node $v_0$ ($w_0$) and ends with node $v$ ($w$) in $M$ ($N$).

Let the nodes in path $\alpha$ referenced in some state in the closed cover $C$ be $v_0, v_1, \ldots, v_m$, and let the nodes in path $\beta$ referenced in some state in $C$ be $w_0, w_1, \ldots, w_n$. There are two cases to consider:

1. $m \leq n$.
2. $m > n$.

We prove that case 1 leads to a contradiction. (A similar proof can show that case 2 also leads to a contradiction.)

From conditions i, ii, iii and iv of closed covers, the closed cover $C$ must have the states $s_0 = [v_0, w_0, x_0, y_0]$, $s_1 = [v_1, w_1, x_1, y_1]$, $\ldots$, $s_m = [v_m, w_m, x_m, y_m]$, where $x_0 = y_0 = E$, and for $i = 1, 2, \ldots, m$, $x_i (y_i)$ is the string $x_{i-1} (y_{i-1})$ after adding to its right side the string of sent messages along path $\beta$ from node $w_{i-1}$ ($v_{i-1}$) to $w_i$ ($v_i$), and after removing from its left side the string of received messages along path $\alpha$ ($\beta$) from node $v_{i-1}$ ($w_{i-1}$) to $v_i$ ($w_i$).

If $v_m = v$, then by condition iv of closed covers, $w_m$ is a final node and
In this case, the network cannot reach any other state after reaching the state \( s_m = [v_m, w_m, x_m, y_m] \). Since \( s = [v, w, x, y] \) is not reached before \( s_m \), then \( s = s_m \) and \( x = x_m = E \), contradicting the assumption that \( x \neq E \).

On the other hand, if \( v_m \neq v \), then the network \((M,N)\) starting at the state \( s_m = [v_m, w_m, x_m, y_m] \) must reach a state \( s' = [v', w', x', y'] \), where \( x' \neq E \), after which no other state is reachable. However state \( s' \) cannot be in \( C \) since \( x' \neq E \) contradicting condition iii of closed covers.

\[\square\]

**Proof of Theorem A2:**

Let \((M,N)\) be a network whose closed cover \( C \) satisfies the following condition. For any final node \( v \) (\( w \)) in \( M \) (\( N \)), there exists exactly one final node \( w \) (\( v \)) in \( N \) (\( M \)) such that \([v, w, E, E]\) is in \( C \).

From Theorem A1, \((M,N)\) is a safe network. To show that it is a phase, it is sufficient to prove that for any exit node pair \((v, w)\) of \((M,N)\), there is a state \([v, w, E, E]\) in \( C \).

Let \((v, w)\) be any exit node pair of \((M,N)\). This implies that the state \( s = [v, w, E, E] \) of \((M,N)\) is reachable, i.e. there exists a sequence \( s_0, s_1, \ldots, s_r \) of reachable states of \((M,N)\) such that \( s_0 \) is the initial state, \( s = s_r \), and for \( i = 0, 1, \ldots, r-1 \), \( s_{i+1} \) follows \( s_i \). This sequence corresponds to two directed paths \( \alpha \) and \( \beta \) in \( M \) and \( N \) respectively such that the following condition is satisfied: Path \( \alpha \) (\( \beta \)) starts with the initial node \( v_0 \) (\( w_0 \)) and ends with node \( v \) (\( w \)) in \( M \) (\( N \)).

Let the nodes in path \( \alpha \) referenced in some state in the closed cover \( C \) be \( v_0, v_1, \ldots, v_m \); and let the nodes in path \( \beta \) referenced in some state in \( C \) be \( w_0, w_1, \ldots, w_n \). There are three cases to consider:

1. \( m < n \).
2. \( m = n \).
3. \( m > n \).

Using the definition of closed covers and using a similar argument to that of Theorem A1, it straightforward to show that cases 1 and 3 lead to contradictions. It also straightforward to show that case 2 leads to the fact that state \([v,w,E,E]\) must be in C. This completes the proof that \((M,N)\) is a phase.
REFERENCES


<table>
<thead>
<tr>
<th>Protocols analyzed</th>
<th># of states generated in validation</th>
<th>Exec. time of validation†</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSC (Figure 5)</td>
<td>107</td>
<td>.760</td>
</tr>
<tr>
<td>modified BSC (Figure 11)</td>
<td>292</td>
<td>2.581</td>
</tr>
<tr>
<td>Call setup phase (Figure 1)</td>
<td>32</td>
<td>.228*</td>
</tr>
<tr>
<td>Data transfer phase (Figure 2)</td>
<td>40</td>
<td>.313*</td>
</tr>
<tr>
<td>BSC using phases</td>
<td>72</td>
<td>.541*</td>
</tr>
<tr>
<td>modified BSC using phases</td>
<td>72</td>
<td>.541*</td>
</tr>
</tbody>
</table>

†: Execution time is measured by Cyber TM seconds.
*: includes checking of the exit set condition.
List of Figures

Figure 1. A Call Setup Phase.
Figure 2. A Data Transfer Phase.
Figure 3. A Call Clear Phase.
Figure 4. An example for constructing multiphase networks.
Figure 5. A BSC Protocol.
Figure 6. Abstract Machine of Machine M in Figure 5.
Figure 7. A ring network.
Figure 8. Two phases in the token ring protocol.
Figure 9. Constructing a multiphase token ring protocol.
Figure 10. Modification of the data transfer phase in Figure 2.
Figure 11. Abstract machines for a modified BSC protocol.
Figure A1. Network (M,N).
Figure A2. Proof of Theorem 2.
Figure A3. Proof of Theorem 3.
Figure 1. A Call Setup Phase.
Figure 2. A Data Transfer Phase.
Figure 3. A Call Clear Phase.
Figure 4. An example for constructing multiphase networks.
Figure 5. A BSC Protocol.
Figure 6: Abstract Machine of Machine M in Figure 5.
Figure 7. A ring network.
Figure 8. Two phases in the token ring protocol.
Figure 9. Constructing a multiphase token ring protocol.
Figure 10. Modification of the data transfer phase in Figure 2.
Figure 11. Abstract machines for a modified BSC protocol.
Figure A1. Network (M,N).
(a) Network \((M^*, N^*)\)

\[ \begin{array}{ccc}
M & \rightarrow & N \\
\downarrow & & \downarrow \\
C & & C
\end{array} \]

\[ M_1 \rightarrow N_1 \quad \text{phase } p_1 \]

\[ \quad \downarrow \quad \quad \downarrow \\
M_2 \rightarrow N_2 \quad \text{phase } p_2 \]

\[ \quad \quad \downarrow \quad \quad \downarrow \\
M_3 \rightarrow N_3 \quad \text{phase } p_3 \]

\[ \quad \quad \quad \downarrow \quad \quad \quad \downarrow \\
M_4 \rightarrow N_4 \quad \text{phase } p_4 \]

\[ \vdots \quad \vdots \]

(b) Infinite Chain

Figure A2. Proof of Theorem 2.
message sequence \( y = y_1 \ldots y_{K+1} \)

phase sequence \( p_1 \ p_2 \ldots p_m \ p_1 \ldots \ p_n \)

edge sequence \( e_1 \ldots f_1 \ldots f_2 \ldots f_m \ldots \ e_1 \)

where \( y_1 \) is the message label of sending edge \( e_1 \),

\( y_{K+1} \) is the message label of sending edge \( e_1 \),

phase \( p_{\cdot i} = (M_{\cdot i}, N_{\cdot i}) \), \( i=1,\ldots,n \), and

\( f_{\cdot i} \) is the last executed edge in \( M_{\cdot i} \), \( i=1,\ldots,m \).

Figure A3. Proof of Theorem 3.