An Improved Method for Constructing Multiphase Communications Protocols

Huai-An Lin, Member, IEEE, and Chao-Li Tarng

Abstract—Research has shown that many communications protocols exhibit multiple phases of behavior, performing a distinct function in each phase. A systematic method has been proposed by Chow, Gouda, and Lam for building multiphase protocols. By connecting several simpler protocols modeling the specific phases in a disciplined way, the newly constructed multiphase protocol enjoys the same correctness properties as the individual phases. The inherent modularity of the resultant protocol makes it easier to understand and analyze. However, the applicability of the existing method is subject to two rather stringent restrictions: the inability to handle message corruption or loss during phase transitions, and a rigid requirement on the selection of the points that connect different phases. This paper describes an improved method that either relaxes or eliminates the above restrictions. The construction of the Normal Response Mode of HDLC (High-level Data-Link Control) is presented to illustrate the use of this new method.

Index Terms—Communicating finite-state machines, communications protocols, design, HDLC, multiphase protocols, safety properties, validation.

I. INTRODUCTION

Divide-and-Conquer has always been one of the most effective ways to solve complex problems. In the domain of communications protocols, Choi and Miller [1] proposed a general scheme for partitioning protocols as the means to deal with complexity that often hampers the design and analysis process. For special classes of protocols, specific techniques have been developed. For example, Chow, Gouda, and Lam [2], [3] proposed a method for designing multiphase protocols—protocols that exhibit multiple phases of behavior, performing a distinct function in each phase. Another method [4] has been devised for building protocols that perform several functions simultaneously. More recently, a technique [5] has been developed for protocols that have the ability to perform several functions but are limited to execute one function at a time.

All these methods allow a large, complex protocol to be constructed as a composite of several simpler protocols that are much easier to design and analyze. Common to these techniques is a two-step building procedure:

1) Design a component protocol for each individual aspect (i.e., a single function or a single phase) of the target protocol.
2) Combine the component protocols into the target protocol in a disciplined manner.

An important advantage of this compositional approach is modularity, which makes a complex protocol easier to understand and validate. Combining the component protocols in a disciplined fashion assures that if each of them satisfies certain correctness properties, the composed protocol possesses the same properties.

This paper presents an extension to the existing method [2], [3] for building multiphase protocols. One of the limitations of the former method is a rather stringent restriction on the validity of phases. Because of it, one has to assume that the messages transmitted immediately before a phase transition can never be damaged or lost. Consequently, the protocols built by this method must run in a noiseless environment or rely on other protocols for reliable transport. Another restriction is a rigid requirement on the selection of the points that connect different phases, further limiting the applicability of the method.

The method presented in this paper either minimizes or eliminates the above restrictions. It enables a multiphase protocol to deal with message corruption and loss, and provides a less restrictive way for phase connection. In the mean time, the new method maintains the advantages of the existing method in allowing a multiphase protocol to be constructed in an easy and modular way.

The rest of the paper is organized as follows. The existing method is reviewed in Section II. The improved method and the correctness criteria for multiphase protocols are discussed in Section III. Section IV presents the application of the new method to a simplified version of NRM (Normal Response Mode) of HDLC (High-level Data-Link Control). Section V provides some concluding remarks on some possible further extensions to the current work.

II. REVIEW OF THE EXISTING METHOD

This section reviews the existing method [2], [3] for constructing multiphase protocols. A protocol is modeled as a pair of finite automata (FA). Each FA consists of a finite set of states and a set of transitions occurring upon symbols that represent exchange of messages. A symbol \(-g\) in one FA represents the action of transmitting a message \(g\) to the other FA. A symbol \(+g\), on the other hand, expresses the reception of \(g\). Because messages are exchanged, the finite automata in a protocol are also called communicating finite-state machines (CFSM's).
Pictorially, a CFSM can be depicted by a directed graph in which states are represented by nodes, and each transition is indicated by an arc $xy$ drawn from the node representing $x$ to the node representing $y$. In discussing CFSM's, we shall call $x$ and $y$ the starting and terminal states, respectively, of the transition represented by $xy$, and say that this transition is incident from and incident to $x$ and $y$, respectively. In each CFSM, one of the states is identified as the initial state, in which the CFSM starts. A state from which no transition is incident (i.e., the state is not the starting state of any transition) is a final state.

An example protocol is given in Fig. 1. State 1 is the initial state of both CFSM's $M$ and $N$. There is no final state in either CFSM.

In the rest of this paper, we use an ordered pair $(M, N)$ to denote a protocol consisting of CFSM's $M$ and $N$. The global state of $(M, N)$ is represented by a vector $[v, w, x, y]$, where $v$ and $w$ are the current states of $M$ and $N$, respectively, and $x$ and $y$ are the FIFO (first-in-first-out) strings of messages being transmitted to $M$ and $N$, respectively. Note that the state of a CFSM changes (so does the corresponding input message string) whenever the CFSM sends or receives a message. Consequently, the global state of $(M, N)$ changes whenever a transition takes place in either $M$ or $N$. Initially, the global state of $(M, N)$ is $[v_0, w_0, A, A]$, where $v_0$ and $w_0$ are the initial states of $M$ and $N$, respectively, and $A$ denotes an empty message string.

If the global state of $(M, N)$ changes from $S_1$ to $S_2$ due to the occurrence of a transition (in either $M$ or $N$), we say that $S_2$ is a successor state of $S_1$. A global state $S_0$ is said to be reachable from another state $S_a$ if either $S_a = S_0$ or there is a series of states $S_1, \ldots, S_n$ such that $S_a = S_1$, $S_b = S_n$, and for $i = 1, \ldots, n - 1$, $S_{i+1}$ is a successor state of $S_i$. If a global state $S$ is reachable from the initial global state, $S$ is simply called reachable.

In Fig. 1, the initial global state of the protocol is $[1, 1, A, A]$. The transmission of $-g_1$ by $M$ results in a new global state $[2, 1, A, A]$. The receiving of $g_1$ leads to another reachable state $[2, 2, A, A]$. From here the protocol can move on to either $[2, 3, g_2, A]$ or $[4, 2, A, g_3]$, depending on which message ($g_2$ or $g_3$) gets transmitted. From $[2, 3, g_2, A]$, the successor state may be either $[3, 3, A, A]$ if $g_2$ is received first, or $[4, 3, g_2, g_3]$ if $g_3$ gets transmitted first. Two possible successor states of $[4, 2, A, g_3]$ are $[4, 1, A, A]$ and $[4, 3, g_2, g_3]$. The reachable state $[4, 3, g_2, g_3]$ represents an error known as unspecified reception. This is because the reception of message $g_2$ is not specified at state 4 in $M$. Similarly, the reception of $g_3$ at state 3 in $N$ is not given. More formally, a global state $[v, w, x, y]$ of protocol $(M, N)$ is an unspecified reception state if and only if any of the following conditions hold:

1) The message string $x$ is not empty, and none of the transitions incident from $v$ has the label $+g$, where $g$ is the head (i.e., the first message to be received) of $x$.
2) The message string $y$ is not empty, and none of the transitions incident from $w$ has the label $+g$, where $g$ is the head of $y$.

The reachable state $[4, 1, A, A]$ in the example protocol represents another type of error—deadlock. In this state, both CFSM's can only receive messages, but their input message strings are empty. Formally, a global state $[v, w, x, y]$ of $(M, N)$ is a deadlock state if and only if both of the following conditions are satisfied:

1) $x = y = \lambda$.
2) If $v$ or $w$ is not a final state, then none of the transitions incident from this state is a message transmission transition. However, if both $v$ and $w$ are final states, then $[v, w, \lambda, A]$ represents proper termination of $(M, N)$ instead of a deadlock.

If none of the reachable global states of a protocol is an unspecified reception or deadlock state, then the protocol is said to be safe. Unexecutable transitions and state ambiguities [1], [6] are also viewed by some as errors with respect to safety. However, the safety in this paper is restricted to unspecified receptions and deadlocks because these two types of behavior are more widely accepted as protocol design errors.

Given two protocols $(M_1, N_1)$ and $(M_2, N_2)$, a multiphase protocol is created by connecting them in the following sense: When the execution of $(M_1, N_1)$ reaches some point of connection, this "leading phase" is terminated while $(M_2, N_2)$, the "trailing phase," is activated. The most challenging task is to define appropriate connections to ensure that the product is safe.

In the existing state of the art, the connection method is quite simple. Let $(M, N)$ denote the multiphase protocol built from $(M_1, N_1)$ and $(M_2, N_2)$. $M$ is constructed by joining some selected final states in $M_1$ with the initial state of $M_2$. Similarly, $N$ is built by combining the selected final states in $N_1$ with the initial state of $N_2$. The initial states of $M_1$ and $N_1$ become the initial states of $M$ and $N$, respectively.

An example is given in Fig. 2 to illustrate this method for multiphase protocols. There are two final states in $M_1$ and $N_1$, as shown in Fig. 2(a), but only state 2 is selected for phase connection. By joining this state to the initial states in $M_2$ and $N_2$, we derive the multiphase protocol in Fig. 2(b).

This method can be easily extended to use a multiphase protocol as a phase in a larger protocol. By repeating the connection procedure, one can build protocols with more than two phases.

Despite its simplicity, this construction method suffers from several significant limitations. First, it requires final states in $M_1$ and $N_1$. In building real multiphase protocols, we often
find that the leading phase has no final states for two reasons. One is the need for recovering from message corruptions and losses. The existence of final states implies that the message transmitted by a CFSM immediately prior to the activation of the trailing phase must always be correctly delivered. If the message is lost or damaged, the other CFSM will never get to the matching final state and thus the execution of the protocol is stalled.

The other rationale for not having final states is that one phase often gets “sandwiched” by another phase in a connection-oriented protocol such as Transmission Control Protocol (TCP). In TCP, connection setup and closing are captured in a single protocol instead of separate phases. Data transfer conducted between the two functions is modeled as an ESTABLISHED state in the connection setup/closing protocol. The problem is that this state is not a final state and thus cannot be used as a connection point to link the protocol modeling the data transfer phase. The HDLC to be discussed in Section IV also illustrates the need to lift the requirement for final states.

An additional limitation of the existing method is that the safety of the multiphase protocol $\langle M, N \rangle$ is contingent upon the following conditions: [2], [3]

1) Both $(M_1, N_1)$ and $(M_2, N_2)$ are safe.

2) There is an one-to-one relationship between the final states in $M_1$ and $N_1$ as follows. If the execution of one CFSM reaches a final state, then the other CFSM will eventually reach the matching final state. And when that happens, there should be no in-transit message between the CFSM’s.

To express this restriction in a formal way, we use $v$ and $w$ to denote such a pair of matching states in $M_1$ and $N_1$, respectively. Then if the execution of $(M_1, N_1)$ reaches a global state $[v, \pi, \Lambda, y_1]$, where $\pi \neq w$, then inevitably the global state $[v, w, \Lambda, \Lambda]$ will be reached. This means that there is a series of states $\pi_1, \pi_2, \ldots, \pi_n$ such that $\pi = \pi_1$, $w = \pi_n$, and for $i = 1, \ldots, n - 1$, $[v, \pi_{i+1}, \Lambda, y_{i+1}]$ is the only possible successor state of $[v, \pi_i, \Lambda, y_i]$, where $y = y_1$ and $y_{i+1}$ is the string obtained by removing the head of $y_i$. Similarly, a global state $[\mu, w, \Lambda, \Lambda]$, where $\mu \neq v$, inevitably leads to $[v, w, \Lambda, \Lambda]$.

Note that if state $v$ is paired with state $w$, then $v$ cannot be paired with any other state in $N_1$. In other words, if $w'$ is a state in $N_1$, then $[v, w', \Lambda, \Lambda]$ cannot be a reachable global state unless $w' = w$. Correspondingly, $w$ cannot be paired with any state in $M_1$ except $v$.

3) The matching final states must be both selected for phase connection. This is, if a final state $v$ in $M_1$ is selected, then the corresponding final state in $N_1$ must also be selected. In the same way, if a state $w$ in $N_1$ is selected, the corresponding state in $M_1$ must be selected.

In the next section, we will discuss a new method for building multiphase protocols that eliminates the need for final states. In the new technique, the CFSM’s can actually “reactivate” the leading phase even after the trailing phase has been started. Through reactivations, the CFSM’s are able to detect and correct errors that occur during the phase transition. The new technique also loosens the requirement on state pairing. Instead of enforcing the strict one-to-one relationship, the new method permits a more flexible way for pairing states for phase connection. We will demonstrate these relaxations in building the HDLC protocol in Section IV.

III. THE IMPROVED METHOD

A problem that must be dealt with in relaxing the aforementioned restrictions is the possibility that the CFSM’s may take inconsistent courses of action—a phenomenon we call collision. In this section, we first discuss the sources and ramifications of collisions in the context of multiphase...
protocols. Then a mechanism for resolving collisions based on phase prioritization is described and illustrated. Two conditions that guarantee the safety of multiphase protocols with the built-in collision resolution capability are discussed through informal arguments. A rigorous proof for the sufficiency of these conditions is excluded from this section due to space limitations. Interested readers are referred to an accompanying technical report [9].

As before, we use \((M_1, N_1)\) and \((M_2, N_2)\) to denote the leading and trailing phases, respectively, and \((M, N)\) the resultant multiphase protocol. As in the previous method, \(M\) and \(N\) are essentially built by joining selected states in \(M_2\) and \(N_2\) with the initial states in \(M\) and \(N\), respectively. However, the states selected for phase connection are no longer restricted to final states. In addition, a state selected in one CFSM can be paired with more than one state in the other CFSM. When the phases are connected, several conditions are tested for potential collisions. If such conditions exist, then certain transitions are added to either \(M\) or \(N\) to resolve collisions.

A. Collisions

A possible consequence of allowing reactivation of the leading phase is collisions. Consider the protocols in Fig. 3. Two phases in Fig. 3(a) are connected to form the protocol in Fig. 3(b). Both phases are safe. However, the following sequence of successor global states indicates that an error may occur in the multiphase protocol \((M, N)\):

\[
[1, 1, \lambda, \lambda][2, 1, \lambda, g_1][2, 2, \lambda, \lambda][2, 4, 6, g_2, \lambda][5, 6, 4, 6, \lambda, \lambda]
[7, 4, 6, \lambda, h_1][7, 1, g_5, h_3].
\]

The CFSM's \(M\) and \(N\) first proceed to the connection states 5.6 and 4.6, respectively (represented by the reachable global state \([5, 6, 4, 6, \lambda, \lambda]\) in the sequence). Then \(M\) activates the trailing phase by sending a message \(h_1\) to \(N\) (the new global state is \([7, 4, 6, \lambda, h_1]\)). Before this message is received, \(N\) reactivates the leading phase by returning to state 1.2

The last global state in the sequence, \([7, 1, g_5, h_1]\), is a case of unspecified reception. The transition incident from state 7 in \(M\) is not designated for receiving \(g_5\), and no transition incident from state 1 in \(N\) is for receiving \(h_1\). This error is a consequence of the collision between activating the trailing phase and reactivating the leading phase.

Collisions that arise inside a phase may also cause errors in a multiphase protocol. We illustrate this problem in Fig. 4. The multiphase protocol in Fig. 4(b) is built from two safe phases in Fig. 4(a). When operating in the leading phase, the CFSM's \(M\) and \(N\) collide with each other by sending messages \(g_2\) and \(g_1\), respectively, at their initial states. Without the connection to the trailing phase, \(N\) will eventually receive \(g_2\) and be forced to move to state 2. On the other hand, \(M\) remains in state 2 when receiving \(g_1\). This collision is thus resolved in favor of \(M\). However, when the two phases are connected, the collision resolution mechanism in the leading phase loses its power, as depicted by the following sequence of successor global states:

\[
[1, 1, \lambda, \lambda][1, 3, 4, g_1, \lambda][2, 3, 4, g_1, g_2][2, 3, 4, \lambda, g_2]
[2, 5, h_1, g_2].
\]

The global state is \([2, 3, 4, g_1, g_2]\) immediately after the occurrence of the aforementioned collision. Before receiving message \(g_2\), \(N\) activates the trailing phase by sending \(h_1\) to \(M\). Overlooking the collision, \(N\) is no longer ready to receive \(g_1\). Similarly, after receiving \(g_1\), \(M\) cannot handle the incoming message \(h_1\) while operating in the leading phase. This example shows that collisions that go unresolved within a phase may lead to erroneous behavior in a multiphase protocol.

B. Resolving Collisions

In the following, we describe a scheme for resolving collisions in multiphase protocols. Our approach is based on prioritizing the constituent phases: When a collision arises, the execution of the trailing phase is aborted. There is no
special logic in giving the higher priority to the leading phase; however, this choice seems to work in most multiphase protocols.

Before discussing the scheme, we introduce some definitions helpful for the presentation as follows. When connecting two phases \((M_1, N_1)\) and \((M_2, N_2)\) into a multiphase protocol \((M, N)\), we say that \(M\) inherits a state \(S\) (or a transition \(t\)) from \(M_i (i = 1, 2)\) if \(S\) (or \(t\)) is a part of \(M_i\). For instance, in Fig. 3(b), \(M\) inherits states 1, 2, 3, and 4 from \(M_1\). \(M\) also inherits state 7 from \(M_2\), and the joint state 5.6 from both \(M_1\) and \(M_2\). In addition, \(M\) inherits the transitions for transmitting/receiving \(g\) and \(h\) messages from \(M_1\) and \(M_2\), respectively. Also for convenience, we say that \(M\) inherits a path\(^3\) from \(M\) if all the transitions on the path are inherited from \(M_i\). In the same way, we say that \(N\) inherits a state (transition or path) from \(N_i (i = 1, 2)\) if the state (transition or path) is a part of \(N_i\).

The constructions of \(M\) and \(N\) are symmetric to each other; therefore, only that of \(M\) will be explicitly discussed here. The initial step of building \(M\), as described earlier, is to combine the selected states in \(M_1\) with the initial state in \(M_2\). In the following, we use \(v\) and \(u_0\) to denote a selected state in \(M_1\) and the initial state in \(M_2\), respectively. We also use \(w\) and \(v_0\) to represent a state in \(N_1\) selected for connection (to \(N_2\)) and the initial state of \(N_2\), respectively. For simplicity, we assume that the messages in the leading phase are completely different from those in the trailing phase. This assumption ensures that the CFM's always unmistakably recognize messages used in different phases. This is almost always true in real cases.

After the initial step of constructing \(M\), the following four tests are applied. The first three tests are concerned with resolving collisions due to phase reactivation. The last one handles unresolved collisions within the leading phase in the context of the multiphase protocol \((M, N)\).

**Test 1:** If \(M\) inherits from \(M_1\) a sending transition \(t\) incident from the joint state \(v_0u_0\), then add a sending transition incident from each state inherited from \(M_2\). Each newly added transition has the same terminal state and label as \(t\).

The sending transition \(t\) represents the action of reactivating the leading phase. By adding the new sending transitions to the states in \(M_2\), we in effect allow \(M\) to take this action at any state in \(M_2\). However, the additions may be restricted to any subset of the states in \(M_2\). In either case (adding the transitions to all states or selected states in \(M_2\)), the resulting multiphase protocol is safe, provided the safety conditions to be discussed later are satisfied.

**Test 2:** If \(M\) inherits from \(M_1\) a receiving transition \(t\) incident from the joint state \(v_0u_0\), then add a receiving transition incident from each state inherited from \(M_2\). Each newly added transition has the same terminal state and label as \(t\).

This test is symmetric to Test 1. The receiving transition \(t\) is an indication that the leading phase can be reactivated by the other CFM \(N\). The leading phase has the higher priority; therefore, when seeing the message for phase reactivation, \(M\) leaves the trailing phase and returns to the leading phase. This process of responding to phase reactivation is implemented by adding the new transitions to the states in \(M_2\).

**Test 3:** If \(M\) inherits a receiving transition \(t\) from \(M_2\) and a sending path (a path that is entirely composed of sending transitions) \(p\) from \(M_1\), starting from joint state \(v_0u_0\), then add a loop\(^4\) to each state on \(p\). Each newly added loop has the same label as \(t\).

The first transition on the sending path \(p\) represents an action of reactivating the leading phase by \(M\). The receiving transition \(t\) is used to receive a message from \(N\) in the trailing phase. If \(M\) indeed reactivates the leading phase while \(N\) sends the message in the trailing phase, then \(M\) may receive the message at any state on \(p\), which is in the leading phase. This message should be ignored (discarded) because of the priority of the leading phase. This process of discarding the message is implemented by adding a receiving loop to each state on the path.

**Test 4:** If the following two statements are true:

1) \(N\) inherits a sending path \(p\) from \(N_2\), starting from the joint state \(w_v_0\) (as mentioned, \(w\) and \(v_0\) are a selected state in \(N_1\) and the initial state in \(N_2\), respectively), and

---

\(^3\)In a CFM, a path is a finite sequence of transitions in which the terminal state of each transition coincides with the initial state of the following transition.

\(^4\)In a CFM, a loop is a transition that has the identical starting and terminal states.
2) (Condition K) \(N\) may reach \(w\) while \(M\) is in a non-
connection state \(\mu\), meanwhile the input message string
for \(M\) is empty whereas the string for \(N\) is not. In
other words, \([\mu, w, A, y]\) is a reachable global state of
\((M_1, N_1)\), where \(y\) is an arbitrary nonempty message
string,\(^5\)
then for each transition on \(p\), add a receiving loop to state \(\mu\)
in Condition K and each state on any sending path \(q\) that
is inherited from \(M_1\) and starts from \(\mu\). Each newly added loop
is for discarding one of the messages transmitted by \(N\) from
the states on \(p\).

This test can be best explained in light of the example in
Fig. 4. As shown before, \([2, 3, 4, A, g_2]\) is reachable global state
in this protocol. At state 3.4 (corresponding to state \(w\) in
Condition K), \(N\) may activate the trailing phase \((M_2, N_2)\).
\(M_2\), on the other hand, is not ready to enter \((M_2, N_2)\) at state
2 (corresponding to the nonempty input string to \(N\)). The empty input string of \(M\) embodies the assumption that the collision inside \((M_1, N_1)\) should be resolved in favor of \(M\). If \(N\) indeed activates the trailing phase, it may transmit one or more messages on the sending path \(p\) in \(N_2\) before detecting the collision. These messages should be discarded by \(M\). Because these messages can be received by \(M\) at \(w\) or any states on the sending path \(q\) inherited from \(M_1\), we add the receiving loops to these states to implement the process of discarding the messages in question.

### C. Two Examples

Fig. 5 depicts the multiphase protocol obtained by applying
the above collision resolution mechanism to the protocol in
Fig. 3(b). As before, state 5 in \(M_1\) and state 4 in \(N_1\) are
selected for connection. The following transitions (highlighted
in Fig. 5) have been added:

1) In \(M\), a receiving transition labeled \(+g_5\) is added from
state 7 to state 1 according to Test 2. This is because \(M\)
inherits from \(M_1\), a receiving transition that is labeled
\(+g_5\) and incident from the joint state 5.6 and to state 1.

2) In \(N\), a sending transition labeled \(-g_5\) is added from
state 7 to state 1 according to Test 1. This is because \(N\)
inherits from \(N_1\) a sending transition labeled \(-g_5\) and incident
from the joint state 4.6 and to state 1.

3) Also in \(N\), two receiving loops labeled \(+h_1\) are added to
states 1 and 3 according to Test 3, owing to the receiving
edge labeled \(+h_1\) that \(N\) inherits from \(N_2\), the sending
path, starting from the joint state 4.6, that \(N\) inherits
from \(N_1\), and the fact that states 1 and 3 are on this
path.

Note that the reachable global state \([7, 1, g_5, h_1]\), used to
be an unspecified reception state in the original protocol [Fig.
3(b)], is no longer an error in Fig. 5. State 7 in \(M\) now
is able to process the incoming message \(g_5\), and state 1 in \(N\)
can handle \(h_1\).

\(^5\)The reachability of such a state can be verified using a class of techniques
known as reachability analysis [10]–[14].
selected for phase connection (called exit state pair) must be stable, in the sense that when the CFSM's reach the respective states in the pair, there should be no in-transit messages. This requirement eliminates the possibility of “unfinished business” when both CFSM’s are ready to enter the trailing phase. If \((v, w)\) is an exit state pair, then this stability requirement can be expressed as follows:

1) The global state \([v, w, A, A]\) is reachable in \((M_1, N_1)\).
2) If a global state \([v, w, x, y]\) is reachable, then \(x = y = A\).

Note that in the previous method by Chow et al., each state pair selected for phase connection must also satisfy the stability requirement. In addition, if \((v, w)\) is such a pair, and \(M\) first reaches state \(v\) (or \(N\) first reaches \(w\)), then it is necessary for the other CFSM to reach the matching state. Also, each selected state must appear in exactly one exit state pair. These two additional restrictions are no longer needed in the new method for building multiphase protocols.

Having stability in exit state pairs is not enough, as illustrated by the counter example in Fig. 7(b). Both state pairs \((3,3)\) and \((4,3)\) are stable in the leading phase [Fig. 7(a)], but only the former is selected as the exit state pair. Now consider the following two sequences of successor global states:

- **S1**: \([1, 1, A, A][2, 1, A, g_1][2, 2, A, A][2, 3, 4, g_3, A]\)
  \([3, 5, 3, 4, A, A][4, 5, 5, h, A]\)
- **S2**: \([1, 1, A, A][2, 1, A, g_1][2, 2, A, A][2, 3, 4, g_2, A]\)
  \([4, 3, 4, A, A][4, 5, 5, h, A]\).

No error occurs in S1. However, S2 stops at an unspecified reception global state \([4, 5, 5, h, A]\). In both sequences, \(N\) gets to the state for phase connection (i.e., state 3). The difference is that \(M\) also reaches the connection state (state 3) in S1 but not in S2. The error in S2 can be avoided by also selecting state 4 in \(M_1\) (as state 3) for connection to \(M_2\). That is, if either \((3,3)\) or \((4,3)\) is selected as a point for phase connection, then the other should also be selected.

More generally, a safety condition for the multiphase protocol \((M, N)\) is that if any stable state pair \((v, w)\) in \((M_1, N_1)\) is selected as an exit state pair, then any other stable state pair of the form \((v, w')\) or \((v', w)\) should also be selected.

Formally, if \(C\) denotes a set of stable state pairs in \((M_1, N_1)\), we say that \(C\) is a *closure* if it satisfies the following constraints:

1) If \((v, w)\) is in \(C\), then any stable state pair \((v, w')\) is also in \(C\).
2) If \((v, w)\) is in \(C\), then any stable state pair \((v', w)\) is also in \(C\).

This safety condition is summarized as follows:

**SC 1**: A multiphase protocol \((M, N)\) constructed from two phases \((M_1, N_1)\) and \((M_2, N_2)\) using the method described in Section III-B is safe if both \((M_1, N_1)\) and \((M_2, N_2)\) are safe and all the exit state pairs in \((M_1, N_1)\) form a closure.

In many real protocols, the trailing phase can be activated by only one of the CFSM’s. In such cases, a less restrictive safety condition can be found.

Let \(C\) be a set of the stable state pairs in \((M_1, N_1)\). We call \(C\) a *half closure* if any one of the aforementioned closure restrictions is satisfied. For ease of distinction, we say \(C\) is a *left closure* or *right closure* if the above restriction 1) or 2) holds, respectively.

Now another safety condition can be stated:

**SC 2**: A multiphase protocol \((M, N)\) constructed from two phases \((M_1, N_1)\) and \((M_2, N_2)\) using the method described in Section III-B is safe if both \((M_1, N_1)\) and \((M_2, N_2)\) are safe...
and any of the following conditions holds:

1) The exit state pairs in \((M_1, N_1)\) form a left closure, and only \(N\) can activate the trailing phase.

2) The exit state pairs in \((M_1, N_1)\) form a right closure, and only \(M\) can activate the trailing phase.

We argue the validity of SC 2 informally using the example in Fig. 7. Consider Fig. 7(b) again. Only \(N\) can activate the trailing phase, since state 5 is not the starting state of any sending transition in \(M_2\). However, the set of state pair \(\{(3,3)\}\) is not a left closure (as mentioned earlier in this section, the state pair \((4,3)\) is also stable). The safety condition SC 2 is not satisfied.

Now let the transitions incident from the joint states 3.5 and 3.4 be swapped as shown in Fig. 7(c). The global state \([4,3,4,\Lambda,\Lambda]\) (in S2) is still reachable in the modified protocol. Note that \(\{(3,3)\}\) is a right closure, and \(N\) can no longer activate the trailing phase. Thus SC 2 is met. In the new protocol, the erroneous global state \([4,5,\Lambda,\Lambda]\) (reachable in S2) cannot be reached. Instead, after getting to state 4, \(M\) will transmit a message \(g_4\), bringing \(N\) back to state 2. This process may be repeated indefinitely until \(N\) sends a \(g_3\) to \(M\). Only then can the CFSM’s enter the trailing phase.

IV. CONSTRUCTING THE NORMAL RESPONSE MODE OF HDLC AS A MULTIPHASE PROTOCOL

One interesting application of the new method for building multiphase protocols is the International Standards Organization’s HDLC [15]–[18]. HDLC provides a number of options for its implementations, supporting both half-duplex and full-duplex transmission, point-to-point and multipoint configurations, as well as switched or nonswitched channels. The stations are allowed to communicate in one of the three operational modes: Normal Response Mode (NRM), Asynchronous Response Mode (ARM), and Asynchronous Balanced Mode (ABM). Each of these modes can be constructed as a multiphase protocol. The building of NRM is demonstrated in this section.

Under the normal conditions, the operation of NRM can be divided into three stages: link setup, data transfer, and disconnection. However, it will become apparent that NRM does not conform to the clean, “water-fall” type of multiphase behavior as characterized in Section II, and thus cannot be constructed using the previous method.

Due to the space limitation, we will consider only the fundamental features of NRM. Various failure recovery features (e.g., reset) and other peripheral functions (e.g., exchange of station identification) are excluded from the discussion; however, all these “extras” can be added without much difficulty.

The rest of this section is organized as follows. The protocols that model the link setup/disconnection and the data transfer phases are described in Sections IV-A and IV-B, respectively. The construction of NRM as a multiphase protocol is discussed in Section IV-C.

A. Link Setup and Disconnection

In NRM, a primary station and one or more secondary stations communicate in a point-to-point or multipoint configuration. Again, for brevity, only the point-to-point communication between a primary station and a secondary station will be considered.

The link setup/disconnection protocol \((P_i, S_i)\) is shown in Fig. 8, where \(P_i\) and \(S_i\) model the primary and secondary stations, respectively.

The meaning of the messages exchanged in this phase are explained as follows:

- **SNRM** denotes a “set NRM” command; it is transmitted from the primary station \(P_i\) to place the secondary station \(S\) in the NRM.
- **UA** denotes an “unnumbered acknowledgment” response; it is used by \(S\) to acknowledge a link setup (i.e., SNRM) or disconnection (i.e., DISC) command.
- **DM** denotes a “disconnected mode” response; it is used by \(S\) to indicate that \(S\) is inoperative.
- **DISC** denotes a “disconnection” command; it is used by \(P_i\) to place \(S\) in the disconnected state.
- **ER\textsubscript{SNRM}** denotes a virtual message which represents the corruption or loss of SNRM.
- **ER\textsubscript{UA}** denotes a virtual message which represents the corruption or loss of UA.
- **ER\textsubscript{DISC}** denotes a virtual message which represents the corruption or loss of DM.
corruption or loss of DISC.

$T_m$ denotes a virtual message which represents an occurrence of timeout.

We use virtual messages to model the effects of message loss or corruption. The loss/corruption of $SNRM$ ($UA$ or $DISC$) is modeled by the transmission and reception of a virtual message $ET_{SNRM}$ ($ET_{UA}$ or $ET_{DISC}$) in $P_1$ and $S_1$, respectively. Similarly, the loss/corruption of $DM$ is modeled by the transmission and reception of a virtual message $ET_{DM}$ in $S_1$ and $P_1$, respectively. The occurrence of timeout is modeled by a virtual message $T_m$ transmitted from $S_1$ to $P_1$. The operation of the link setup/disconnection phase $(P_1, S_1)$ can be described as follows:

1) $P_1$ initializes a link by sending a $SNRM$ command to $S_1$. If $S_1$ is ready for communication, it responds with an acknowledgment $UA$, and $P_1$ and $S_1$ enter the data phase (states 3 and 9, respectively). Otherwise, $S_1$ responds with a $DM$, and both stations return to their initial states.

2) The request $SNRM$ is corrupted or lost (modeled by $P_1$ sending a virtual message $ET_{SNRM}$ to $S_1$). Upon timeout (modeled by $S_1$ sending a virtual message $T_m$ to $P_1$), $P_1$ retransmits a $SNRM$.

3) The returned $UA$ or $DM$ is corrupted or lost (modeled by the virtual messages $ET_{UA}$ or $ET_{DM}$). Then a $SNRM$ (or $ET_{SNRM}$) will be retransmitted; this process of retransmission will be repeated until a correct acknowledgment $UA$ (or disconnected mode $DM$) is received.

4) When $P_1$ and $S_1$ have reached the data transfer phase (at states 3 and 9, respectively), $P_1$ may disconnect the link at any time by sending a $DISC$. $S_1$ then responds with an acknowledgment $UA$, and both $P_1$ and $S_1$ return to the initial states.

5) The command $DISC$ is corrupted or lost (modeled by the virtual message $ET_{DISC}$). Upon timeout (modeled by the virtual message $T_m$), $P_1$ retransmits a $DISC$.

6) The acknowledgment $UA$ for the $DISC$ request is corrupted or lost (modeled by the virtual message $ET_{UA}$). Then a $DISC$ will be retransmitted; this process of retransmission will be repeated until a correct $UA$ is received.

For simplicity, we have assumed that a timeout occurs only after a message has actually been lost or corrupted. In other words, only the nonpremature timeouts are considered.

The previous method, as described in Section II, does not allow the protocol $(P_1, S_1)$ to be connected by another protocol, since $P_1$ and $S_1$ have no final states. However, state 3 in $P_1$ and state 9 in $S_1$ represent the logical places where the data transfer phase (Section IV-B) can be activated. However, two sending transitions, labeled $-DISC$ and $-ET_{DISC}$, are incident from state 3 in $P_1$, and two receiving transitions, labeled $+DISC$ and $+ET_{DISC}$, are incident from state 9 in $S_1$. These transitions allow the primary station $P_1$ to skip the data transfer phase and disconnect the link immediately after it has been set up.

Even if $P_1$ is not allowed to disconnect the link, there is another problem—the secondary station $S_1$ still has no final state. As shown in Fig. 8, state 9 is the starting state of two receiving transitions labeled $+SNRM$ and $+ET_{SNRM}$. These transitions exist because the message $UA$ transmitted by $S_1$ before entering state 9 may get lost or damaged, and the primary station $P_1$ may retransmit the connection request $SNRM$. The issue of message corruption/loss during phase transitions is not addressed in the previous method, where control messages (such as $SNRM$ and $DISC$) are simply assumed to be uncorruptable.

The phase $(P_1, S_1)$ has been proved, via reachability analysis, to be safe [9]. Ten state pairs—(1,7), (2,8), (3,9), (5,13), (6,9), (5,13), (6,9), (2,10), (4,7), (4,9), and (2,11)—are found to be stable. The state pair (3,9) is selected as the exit state pair for connection to the data transfer phase. Note that $(3,9)$ is a right closure.

### B. Data Transfer

During the data transfer phase, the secondary station must receive explicit permission (i.e., poll) from the primary station before transmitting messages. After receiving a poll, the secondary may transmit one or more information frames to the primary station. After the last frame is transmitted, the secondary waits again for a new poll.

To simplify the model of the data transfer phase, we specify it in two parts:

- Two CFSM’s that provide a general control flow of the primary station $P_s$ and the secondary station $S_s$ (Fig. 9).
- Two tables that provide a more detailed description of transitions in $P_s$ and $S_s$ (Tables I and II).

Each state in the model is labeled by a 5-tuple $(VS(i), VA(j), VR(m), \text{poll})$.

- $VS(i)$ denotes “expect to send data with sequence number $i$ (modulo 8)”.
- $VA(j)$ denotes “expect to receive acknowledgment with sequence number $j$ (modulo 8)”.
- $VR(m)$ denotes “expect to receive data with sequence number $m$ (modulo 8)”.
- $\text{poll}$ denotes the polling condition.

$T$ denotes the timeout condition.

In $P_s$, $\text{poll}$ is set to 1 when the poll-bit has been set in an outstanding (unacknowledged) information frame; otherwise, $\text{poll}$ is set to 0. In $S_s$, $\text{poll}$ is set to 1 when a poll has just arrived, after responding to the poll, $\text{poll}$ is set to 0.

In $P_s$, $T$ is set to 1 only after a timeout (i.e., arrival of a virtual message $T_m$) and before a retransmission. In $S_s$, this value is set to 1 only after the arrival of a corrupted/lost message and before the transmission of a $T_m$.

For example, a state labeled $(VS(i), VA(j), VR(m), 0, 0)$ in $P_s$ means that the next information frame to be sent will carry the sequence number $i$, all the received information frames with sequence numbers up to $j - 1$ have been acknowledged, and the next information frame to be received is expected to have sequence number $m$. No poll is outstanding and no timeout has occurred.

Each information frame is labeled by a sequence of $1$ to $3$ entities which have the following meanings:
TABLE I

<table>
<thead>
<tr>
<th>incident from</th>
<th>incident to</th>
<th>label</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(j), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 0, 0$</td>
<td>$D(i{A(j)p$</td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_A(j)$, $V_R(m), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 0, 0$</td>
<td>$-D(i{A(j)p$</td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_A(j)$, $V_R(m), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 1, 0$</td>
<td>$-D(i{A(j)f$</td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_A(j)$, $V_R(m), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 1, 0$</td>
<td>$-D(i{A(j)f$</td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_A(j)$, $V_R(m), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 1, 0$</td>
<td>$-D(i{A(j)p$</td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_A(j)$, $V_R(m), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 1, 0$</td>
<td>$-D(i{A(j)f$</td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_A(j)$, $V_R(m), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 1, 0$</td>
<td>$-D(i{A(j)f$</td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_A(j)$, $V_R(m), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 1, 0$</td>
<td>$-D(i{A(j)f$</td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_A(j)$, $V_R(m), 0, 0$</td>
<td>$V_S(i + 1), V_A(j), V_R(m), 1, 0$</td>
<td>$-D(i{A(j)f$</td>
</tr>
</tbody>
</table>

TABLE II

<table>
<thead>
<tr>
<th>incident from</th>
<th>incident to</th>
<th>label</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 1, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 0, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 1, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 0, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 1, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 0, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 1, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 0, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 1, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 0, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 1, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 0, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 1, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 0, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
<tr>
<td>$V_S(i), V_A(j), V_R(m), 0, 0$</td>
<td>$V_S(i), V_A(k), V_R(m + 1), 1, 0$</td>
<td>$+D(m{A(k)p$</td>
<td></td>
</tr>
</tbody>
</table>

$D(i)$ denotes a "data frame with sequence number $i,$" where $0 \leq i \leq 7$

$A(j)$ denotes an "acknowledgment with sequence number $j,$" where $0 \leq j \leq 7$

$RR$ denotes a "receiver ready" message; $RR$ is used by $P_d$ to transmit an unpiggybacked acknowledgment.

$p$ denotes a set poll-bit.

$f$ denotes a final-bit; it is used by $S_d$ to respond to a poll received from $P_d$

$Er$ denotes a "corrupted or lost information frame" virtual message.

$Tm$ denotes a "timeout" virtual message.

For example, an information frame labeled $D(i\{A(j)p$ represents a frame which carries sequence number $i,$ acknowledgment number $j,$ and a set poll-bit. The operation of $(P_d, S_d)$ follows the go back $n \{n = 7\}$ ARQ (automatic repeat request). Discussion on this technique can be found elsewhere [19], [20] and is not repeated here. For simplicity, the selective repeat capability of HDLC is not modeled. The data transfer phase $(P_d, S_d)$ has been shown to be safe [9].

C. The Construction of NRM

Now we connect the two phases $(P_1, S_1)$ and $(P_d, S_d)$ to form a multiphase protocol. The exit state pair $(3, 9)$ in $(P_1, S_1)$ is selected for phase connection. The resulting protocol $(P, S)$ is depicted in Fig. 10. To avoid unnecessarily cluttering the figure, we only show the states and transitions related to phase reactivation and collision resolution. The dashed lines...
identify the boundary between the leading phase \((P_l, S_l)\) and the trailing phase \((P_d, S_d)\).

In the primary station \(P\) (the CFSM constructed from \(P_l\) and \(P_d\)), the following transitions are added, according to Tests 1 and 3 of the collision resolution scheme discussed in Section III-B, for reactivating the link setup/disconnection phase \((P_l, S_l)\) and for discarding messages received from the data transfer phase \((P_d, S_d)\):

\[-DISC\] due to Test 1
\[-\bar{E}\bar{DISC}\] due to Test 1
\[+E\bar{r}\] due to Test 3
\[+D(m)\bar{A}(k)\] due to Test 3
\[+D(m)\bar{A}(k)f\] due to Test 3
\[+T_m\] due to Test 3.

In the secondary station \(S\) (the CFSM constructed from \(S_l\) and \(S_d\)), the following transitions are added, according to Test 2 of the scheme, for resolving collisions in favor of \(P_l\):

\[+SNRM\]
\[+DISC\]
\[+E\bar{SNRM}\]
\[+E\bar{DISC}\].

As said, both phases \((P_l, S_l)\) and \((P_d, S_d)\) are safe. Moreover, the set of the exit state pair \((3, 9)\) is a right closure, and only \(P\) can activate the data transfer phase. From safety condition SC 2, we conclude that the multiphase protocol \((P, S)\) is safe.

V. CONCLUSION

We have presented an improved method for building multiphase communications protocols. Like the previous one [2], [3], the new method allows a multiphase protocol to be built as a composition of several component protocols modeling the individual phases. When these components are combined in a disciplined manner, the resultant multiphase protocol retains the correctness properties of the individual ones. In this way, the process of designing a multiphase protocol can be broken into processes of designing component protocols. The components are normally much easier to validate for correctness; consequently, the validation of the end protocol is simplified. The inherent modularity also makes the multiphase protocol easier to understand and modify. Furthermore, the compositional approach to protocol design advocated by these methods promote the reuse of existing protocols. Once a protocol has been designed and analyzed for correctness, its components can be reused in other protocols.

The proposed new method extends the previous one in an important dimension—the ability to handle message loss and corruption during phase transitions. In building the NRM of HDLC, we demonstrated the need for this ability. The new state of the art allows a less restrictive sufficient condition for safety than the one in safety condition SC 2 has been discussed and applied to the ARM (Asynchronous Response Mode) of HDLC in a technical report [9]. The method may also be extended to handle failures of the communicating finite-state machines. The work that needs to be done to complete this extension includes the development of a special phase which models machine failures and the definition of the relationship between this “failure” phase and the other “normal” phases in the protocol. Finally, our method has been developed based on the model of CFSM that has its limitations, particularly in the area of expressiveness. It is desirable to extend the method to make it functional in other more powerful specification models.

ACKNOWLEDGMENT

We are indebted to Prof. C.-H. Chow of the University of Colorado at Colorado Spring and Profs. M. Gouda and S. Lam of the University of Texas at Austin, whose pioneer work inspired ours. We are also thankful to our colleagues D. Cohen, J. Giacopelli, Y.-J. Lin, and C. Schahczenski at Bell Communications Research; their careful reviews of the manuscript of this paper led to many improvements.

REFERENCES


Huai-An Lin (S'81-M'83) received the B.S.E.E. degree from National Taiwan University, Taipei, Taiwan, in 1977, and the M.S. and Ph.D. degrees in computer and information science from the Ohio State University, Columbus, in 1981 and 1983, respectively.

He worked at Gould Research Center, Rolling Meadows, Ill., from 1983 to 1985, and at the University of Washington, Seattle, from 1985 to 1988. He joined Bell Communications Research in 1988, where his current research activities include broadband networks, protocol processing efficiency, performance modeling and analysis of communications systems, distributed computing, and formal techniques for communications protocols and telecommunications services.

Chao-Li Tarng received the B.S. degree in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 1982, the M.S. degree in electrical and computer engineering from Oklahoma State University, Stillwater, in 1985, and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 1990.

He joined Teknekron Communications Systems, Inc., in June 1990, working in the Network Management Division, where his activities include managed information base specification and design. His areas of technical interests are open systems network management, protocol design, and broadband switching systems.

Dr. Tarng is a member of Phi Kappa Phi and Eta Kappa Nu.