(Proc. IEEE Military Comm. Conf. (MILCOM), Atlantic City, NJ, Oct 1999)

USING SEMICONTROLLABLE INTERFACES IN TESTING ARMY COMMUNICATIONS PROTOCOLS: APPLICATION TO MIL-STD 188-220B *

Mariusz A. Fecko[1], M. Ümit Uyar[2], Paul D. Amer[1], Adarshpal S. Sethi[1]

[1] Computer and Information Science Department
University of Delaware, Newark, DE

[2] Electrical Engineering Department
The City College of the City University of New York, NY

Abstract

Testing Army communications protocols is considered for a testing environment where tester has limited degrees of controllability on applying inputs to an Implementation Under Test. The inputs fall into three categories: directly controllable, semicontrollable, or uncontrollable. A graph conversion algorithm is presented to utilize the semicontrollable inputs, thereby increasing the number of testable transitions. The research was motivated while generating tests for MIL-STD 188-220B. The number of testable transitions for 188-220B Class A–Type 1 Datalink Service module is approximately 200 without utilizing semicontrollable inputs. These 200 account for only 30% of the transitions defined in the protocol specification. The presented methodology makes it possible to increase the number of testable transitions to over 700. Combined with our previous work on testing protocols with timing constraints, the methodology allows us to generate tests free of interruptions due to timeouts, and covering more than 95% of the defined transitions in 188-220B's Type 1 Datalink Layer.

1 Introduction

Testing protocol implementations for conformance to their specifications has been an active research area [1]–[10]. In a testing environment, a tester's control over an Implementation Under Test (IUT) may be limited. This problem is likely to make certain protocol features untestable. In an ideal situation, it should be possible to apply to the IUT every possible input that is defined in the protocol specification. In reality, testers may not have a direct access to all of an IUT's interfaces. In a testing framework containing an (N)-layer IUT, the (N-1)-Service Provider [11], and the upper layer, the IUT's interfaces with the upper layer or the peer entities (such as timers, etc.), typically are not directly accessible (with an exposed interface only between the IUT and the (N-1)-Service Provider). If this is the case, the interactions that involve these 'not directly controllable' interfaces leave certain portions of the IUT model untestable.

There are many real-life protocols that possess not directly controllable inputs due to a tester's limited control over the interactions between the IUT and other communicating entities. For example, for MIL-STD 188-220B (herein 188-220B) [12] Datalink layer service, over 70% of the transitions cannot be directly controlled in CECOM's testing facility without access from the Network layer. Similar controllability problems are present in the IEEE 802.2 LLC Connection Component protocol [13].

This paper presents a methodology that utilizes an IUT's semicontrollable interfaces. An algorithm is presented to modify an IUT's directed graph representation such that the semicontrollable portions of the IUT become directly controllable, where possible. This algorithm is being applied to 188-220B to generate conformance tests for use at CECOM's test facility. Initial results are promising: the number of testable transitions increased from 200 to over 700 for the Class A–Type 1 Datalink Service module [14].

This paper is organized as follows. Section 2 illustrates the *controllability problem* based on 188-220B. The problem is then formally defined in Section 3. Section 4 describes a system model for a testing environment with multiple interfaces of varying controllability, with practical issues introduced into this model in Section 5. An algorithm to modify an IUT's graph so that semicontrollable interfaces can be fully utilized is sketched in Section 6. In Section 7, the application of the technique to testing 188-220B is presented.

2 Practical motivation–188-220B

As motivation for solving the controllability problem, a real protocol is considered where an SUT's (N+1)-layer must be utilized indirectly to test certain transitions within the (N)-layer IUT.

188-220B is a military standard for interoperability of command, control, communications, computers, and intelligence over Combat Net Radios. 188-220B focuses on three layers: Physical, Datalink, and Network. An SUT contains the (N)-layer IUT implemented in the Datalink layer, and the Intranet sublayer, which is part of the (N+1)-layer (Network), as shown in Figure 1.

In the environment used to test 188-220B (located at CECOM, Ft. Monmouth, NJ), the upper layers cannot be directly controlled. Table 1 shows examples of three 188-220B's implementations, developed by different manufacturers for specific areas of use. Although it is technically feasible to directly use inputs from the upper layers for one of them (Internet Controller), extending direct testing approach to other implementations is either impossible, or requires costly, implementation-specific changes

*Prepared through collaborative participation in the Advanced Telecommunication Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002.

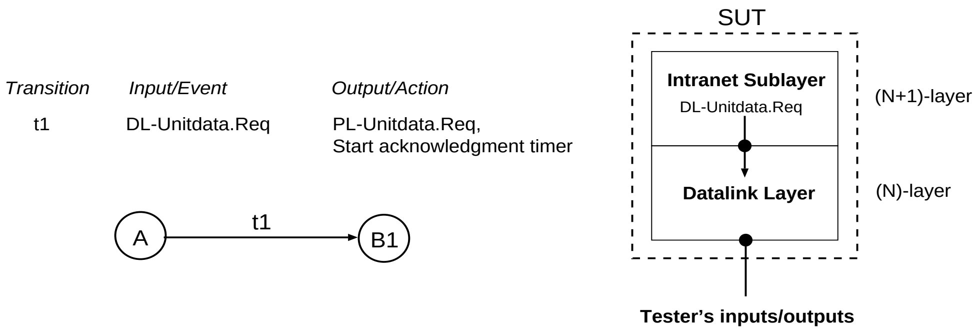

Figure 1: MIL-STD 188-220B: Example of the controllability problem

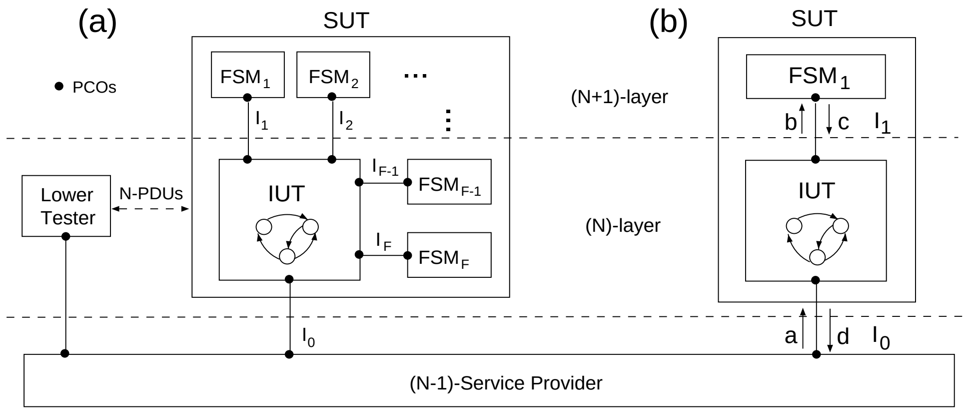

Figure 2: (a) Testing IUT with multiple interfaces; (b) Testing (N)-layer IUT with an (N+1)-layer semicontrollable interface.

Platform	*Implementer*	*Area of use*
Internet Controller	ITT	Command & Control
Tactical Communication Interface Modem	Hughes/ Raytheon	Fire Support
Improved Data Modem	ICI	Airborne

Table 1: 188-220B implementations tested at CECOM.

in the conformance tester's software. Moreover, each implementation would require its own test suite. The technique presented in this paper provides a means to obtain a generic, implementation-independent test suite for the protocol.

The above restrictions make the IUT's transitions that are triggered by the inputs coming from the Network layer not directly testable. An example SUT transition that causes a controllability problem is transition $t1$ from the Class A–Type 1 Service Datalink module [12, 14] (Figure 1). The *input/event* field for this transition requires a *DL_Unitdata_Req* from the (N+1)-layer. Unfortunately, the interface between the IUT and the (N+1)-layer is not directly accessible for generating this input.

In fact, 70% of the transitions are based on not directly controllable inputs. Without indirect testing, test coverage would be seriously reduced. However, by applying the technique introduced in this paper, almost all (>95%) defined transitions can be tested (the number of testable transitions rose to over 700 from approximately 200 for the Class A–Type 1 Datalink Service module.

3 Definition of controllability problem

Consider a testing environment shown in Figure 2 (a). The System Under Test (SUT) contains an IUT, which interacts with F FSMs. $FSM_1, \cdots, FSM_F$, implemented inside the SUT, interact with the IUT through interfaces $I_1, \cdots, I_F$. The points at which a testing system can apply inputs to and observe outputs from the IUT are called *points of control and observation* (PCOs) [11]. Each IUT's interface is associated with a full-duplex PCO through which inputs and outputs can be exchanged. As introduced in [15], each input can be one of three different types: (1) **directly controllable:** a tester can directly apply the input to the IUT through the PCO; (2) **semicontrollable:** a tester cannot directly apply the input to the IUT through the PCO. However, it is possible to utilize one of the FSMs interacting with the IUT to supply this input indirectly; and (3) **uncontrollable:** the input may be supplied through a PCO without any explicit action of the tester, i.e., the input may be generated in the testing system without the tester's control.

The inputs at a given PCO can belong to one or more of these three types. If a PCO has any semicontrollable inputs and does not have any uncontrollable inputs, we say that its associated interface is semicontrollable. If there are no semicontrollable or uncontrollable inputs, the interface is called directly controllable. A typical example of a directly controllable interface is a Lower Tester FSM [11]. A timer FSM, whose only inputs come from an IUT (e.g., start, restart, and stop the timer), is a typical semicontrollable interface.

This paper addresses the problem of generating optimal realizable test sequences in an environment with multiple semicontrollable interfaces. This problem will be referred to as the *controllability problem.*

In the testing framework of Figure 2 (a), the tester is unable to supply inputs directly to the IUT through interfaces $I_1, \cdots, I_F$. Therefore, the interfaces $I_1, \cdots, I_F$ are only semicontrollable, provided that $FSM_1, \cdots, FSM_F$ can be utilized to supply inputs to the IUT. On the other hand, the tester can apply inputs to the IUT directly by using a Lower Tester (LT), which exchanges N-PDUs with the IUT by using the (N-1)-Service Provider. The interface I_0 between the LT and the IUT is therefore directly controllable (I_0 can be considered an interface between the LT and the IUT, because the (N-1)-Service Provider acts as a pass-through that does not alter inputs or outputs).

A simplified test framework with one semicontrollable interface is shown in Figure 2 (b). To test the IUT's transitions triggered by the inputs from I_1, the tester must use the directly controllable interface (by applying message *a* at I_0) to force the IUT to generate outputs to I_1 (message *b*). These outputs are applied to FSM_i at I_1's PCO. As response to these outputs, FSM_i will send back inputs to the IUT through I_1 (message *c*). These inputs will trigger the appropriate transitions in the IUT.

For testing 188-220B's transition $t1$ (described in Section 2), message *a* corresponds to *PL-Unitdata.Ind* that contains an intranet layer message telling the (N+1)-layer to relay the frame to a different network node, message *b* is *DL-Unitdata.Ind*, message *c* corresponds *DL-Unitdata.Req* needed to trigger $t1$, and message *d* is *PL-Unitdata.Req* observable by the lower tester.

4 FSM-based model of test system with semicontrollable interfaces

Let us consider an IUT (modeled by a Finite State Machine (FSM) [6]) interacting with multiple semicontrollable interfaces. The goal of test generation in this environment is *to derive a set of tests exercising each transition in an IUT's FSM at least once.* Specifically, given a graph G representing an IUT's FSM, we want to find a minimum-cost tour of G such that each transition is covered at least once.

Let $A_i = \{a_{i,1}, \ldots, a_{i,c_i}\}$ and $O_i = \{o_{i,1}, \ldots, o_{i,m_i}\}$. A_i is a set of semicontrollable inputs at I_i. O_i is a set of outputs of the IUT that force inputs in A_i to be buffered at I_i. There may be several outputs in set O_i that force input $a_{i,j}$ to be buffered at I_i–$o_{i,j}$ denotes any such output.

Given the graph $G(V, E)$ representing an FSM model of an IUT with multiple semicontrollable interfaces, let us define the following parameters:

- F–number of semicontrollable interfaces interacting with the IUT
- $T_{i,j} \subset E$–subset of edges in G triggered by input $a_{i,j} \in A_i$ at I_i
- b_i–buffer size (maximum number of inputs buffered) at the i-th semicontrollable interface I_i
- c_i–number of different inputs $a_{i,j} \in A_i$ that trigger transitions at I_i
- $U_{i,j} \subset E$–set of IUT's transitions with output $o_{i,j}$

4.1 Modeling buffers as FIFO-type queues

For now, let us assume that there exists a separate FIFO buffer in the semicontrollable interface between the IUT and each interacting FSM. During testing, a buffer may be empty or store an arbitrary sequence of inputs (referred to as a buffer state) to the IUT generated indirectly through the i-th semicontrollable interface.

Let us model the system as an FSM, represented by $G^{'}(V^{'}, E^{'})$. Each vertex in $V^{'}$ is a tuple consisting of the original vertex in V and F buffer states. The maximum number of nodes $|V^{'}|_{max}$ is $|V^{'}|_{max} = |V| * \prod_{i=1}^{F} B(i)$, where $|V|$ is the number of nodes in G, and $B(i)$ is the maximum number of states of the i-th buffer defined as follows:

$$B(i) = \begin{cases} (1 - c_i^{1+b_i})/(1 - c_i) & \text{if } c_i > 1 \\ 1 + b_i & \text{if } c_i = 1 \end{cases} \quad (1)$$

In the general case, the maximum number of nodes in $G^{'}$ grows exponentially with the number of semicontrollable interfaces F and the buffer size b.

5 Model refinement based on practical constraints

Although the model presented in Section 4 assumes that a semicontrollable interface consists of FIFO-type buffers, in practice this assumption may not be true for all implementations. Test sequences generated for an IUT with only FIFO-type buffers become non-deterministic for other IUTs using different types of interfaces [15]. To avoid this type of non-determinism during testing, the model presented in Section 4 will be used to generate tests with the restriction that, **(1)** the IUT model should have a buffer size $b_i = 1$, and, **(2)** at any time, only one of the IUT's semicontrollable interfaces can buffer an input. In such case, the maximum number of nodes is $|V^{'}|_{max} = |V| * (1 + \sum_{i=1}^{F} c_i)$, which indicates a linear growth. For small F and c_i, the size of $G^{'}$ is only a small multiplicant of G.

The introduced restrictions on the buffer size and on the number of buffered inputs help avoid non-deterministic behavior of the SUT during testing. Although the test sequence length is increased by these restrictions, the tests become applicable to many different SUT interface types.

6 Graph conversion algorithm

In this section, an algorithm sketch for converting $G(V,E)$ to $G^{'}(V^{'},E^{'})$ is presented (a detailed description of the algorithm along with its pseudocode is available in [16]). The algorithm proceeds as follows:

1. $r^{'}$, the root of $G^{'}$, is initialized as $(r, \emptyset, \cdots, \emptyset)$–the root of G and the configuration of empty buffers
2. $E^{'}$ is initialized as empty set, and $V^{'}$ as $\{r^{'}\}$
3. unexplored vertices are queued in Q (initialized to $V^{'}$)
4. repeat in a loop until Q is empty
   (a) $v^{'} = (v_{start}, \hat{B}_1, \ldots, \hat{B}_F)$ is dequeued from Q
   (b) for each outgoing edge $e = (v_{start}, v_{end}) \in E$
      i. determine k, index of e's class
      ii. given $(\hat{B}_1, \ldots, \hat{B}_F)$ and *Class k*, construct:
         - new configuration $(B_1, \ldots, B_F)$
         - new vertex $v^{'}_{new} = (v_{end}, B_1, \ldots, B_F) \in V^{'}$
         - new edge $e^{'}_{new} = (v^{'}, v^{'}_{new}) \in E^{'}$
   (c) add new edges to $E^{'}$ iff inputs in $(\hat{B}_1, \ldots, \hat{B}_F)$ cannot trigger other edges outgoing from v_{start}
   (d) end vertices $v^{'}_{new} \in V^{'}$ of new edges in $E^{'}$ are appended to Q
5. prune $G^{'}$ to a Strongly Connected Component

Let B_i denote a sequence of inputs buffered at the i-th semicontrollable interface. Each state $v^{'} \in V^{'}$ has two components: the original state $v \in V$, and the current configuration of F buffers, i.e., $v^{'} = (v, \hat{B}_1, \ldots, \hat{B}_F)$. The algorithm constructs all possible buffer configurations with up to b_i inputs buffered at I_i. Given an original edge $e = (v_{start}, v_{end}) \in E$, and the current vertex $v^{'} = (v_{start}, \hat{B}_1, \ldots, \hat{B}_F)$, a new vertex $v^{'}_{new}$ is created based on a new configuration $B_1, \ldots, B_F$, i.e., $v^{'}_{new} = (v_{end}, B_1, \ldots, B_F)$. The edge $e = (v_{start}, v_{end}) \in E$ is included in $E^{'}$ as $e^{'}_{new} = (v^{'}, v^{'}_{new})$. The edge $e \in E$ belongs to one of the four classes depicted in Figure 3:

- *Class 1:* e is triggered by an input from and generates output(s) to an LT.
- *Class 2:* e is triggered by an input from an LT and generates an output $o_{q,l}$ (buffered in B_q to create a new configuration) at I_q.
- *Class 3:* e is triggered by $a_{p,k}$ (extracted from B_p to create a new configuration) from I_p and generates output(s) to an LT.
- *Class 4:* e is triggered by an input $a_{p,k}$ from I_p and generates an output $o_{q,l}$ at I_q.

It can be shown that $G^{'}(V^{'},E^{'})$ built by the presented algorithm is a *minimal valid* representation of the system as an FSM. Based

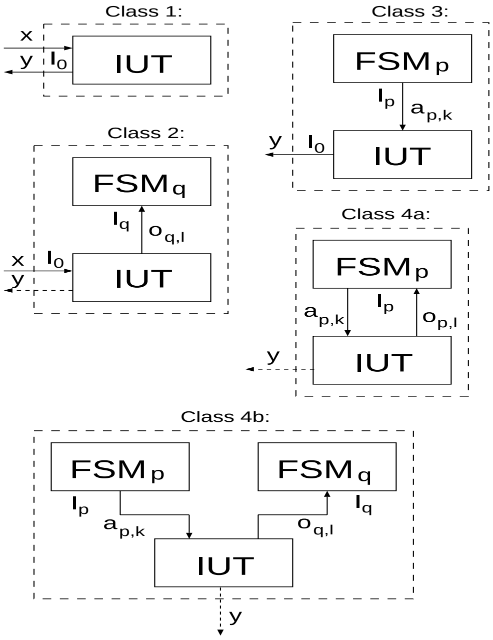

Figure 3: Classes of edges in $G^{'}$ (dashed-lined outputs are optional).

on the practical considerations discussed in Section 5, the algorithm can be refined so that at any given point in time there could be a single input buffered in only one of the buffers B_i, which yields a linear running time of $RT_{ref} = O(c * F * |E|)$.

7 Generating tests for practical testing framework–188-220B

7.1 Rural Chinese Postman Problem formulation

Given graphs G and $G^{'}$, our goal is to find a minimum-cost tour of $G^{'}$ such that each edge in E is covered at least once. $G^{'}$ will likely contain multiple appearances of certain edges from the original graph G. Let $E^{'}_c \subseteq E^{'}$ be the subset of edges in $G^{'}$ such that each original edge in E is represented by at least one copy in $E^{'}_c$. To build a minimum-cost tour of $G^{'}$ covering each original edge in E is equivalent to finding a minimum-cost tour of $G^{'}$ that includes each transition in $E^{'}_c$ (the set of *mandatory* edges) at least once, and each transition in $(E^{'} - E^{'}_c)$ (the set of *optional* edges) zero or more times. This problem is known as the Rural Chinese Postman Problem [17], with a solution presented in [3].

The practical concern of making diagnostics easier [15] suggests the following guideline: "Test as many transitions as possible without interactions at semicontrollable interfaces." Therefore, each transition in E should be represented in $E^{'}_c$ by a copy corresponding to no inputs buffered at the semicontrollable interfaces.

Edge name	*Input from*	*Output to*
t1	$FSM_1?a_{1,1}$	$LT!y_1$
u1	$LT?x_1$	$FSM_1!o_{1,1}$
u2	$LT?x_1$	$FSM_1!o_{1,1}$
u3	$LT?x_1$	$FSM_1!o_{1,1}$
u4	$LT?x_1$	$FSM_1!o_{1,1}$
u5	$LT?x_1$	$FSM_1!o_{1,1}$
e1	$LT?x_2$	$LT!y_2$
e2	$LT?x_3$	$LT!y_3$
e3	$LT?x_4$	$LT!y_4$
e4	$LT?x_5$	$LT!y_2$
e5	$LT?x_4$	$LT!y_5$
e6	$LT?x_3$	$LT!y_3$
e7	$LT?x_2$	$LT!y_5$
e8	$LT?x_3$	$LT!y_3$
e9	$LT?x_5$	$LT!y_2$

Table 2: Inputs and outputs for Figure 4 (a). $A?x$ denotes receiving input x from A; $B!y$ sending output y to B.

7.2 Application to 188-220B Class A–Type 1 Datalink Layer [12, 14]

The FSM in Figure 4 (a) models the part of Class A–Type 1 Datalink Layer (DLL) of 188-220B that handles retransmissions and coupled acknowledgments. The Datalink layer IUT interacts with one semicontrollable interface at the Intranet layer. The Intranet layer must be used as an "implicit" upper tester to test all of the IUT's transitions.

To keep the example simple enough, the following assumptions have been made:

- there is only one semicontrollable interface between the Intranet and the Datalink layers;
- the semicontrollable interface's buffer size $b = 1$;
- the maximum allowable number of retransmissions is 1;
- frames are sent to one destination (188-220B allows up to 16 data link addresses for multidestination frames);
- all frames require a coupled acknowledgment.

The IUT maintains two timers: an acknowledgment timer, whose expiration causes retransmission of an unacknowledged frame or passing an acknowledgment failure to the Intranet layer, and a TP timer that in the running state prevents the IUT from transmitting frames.

The graph G representing the IUT consists of 4 states:

- *A*–the initial state, there are no outstanding frames, acknowledgment timer off;
- *B0, B1*–one outstanding frame, acknowledgment timer on, number of retransmissions left equal to 0 and 1, respectively;
- *C0*–TP timer running, the outstanding frame buffered with one retransmission left;
- *C*–TP timer running, no outstanding frames.

G has 15 transitions:

- *t1*–input $a_{1,1}$: the Intranet layer sends a packet with a TOS requiring a coupled acknowledgment, output y_1: the Datalink layer transmits a frame with a P/F=1 and starts acknowledgment timer;
- *u1-5*–input x_1: tester sends a frame from the Physical Layer containing a packet to be relayed, output $o_{1,1}$: the Datalink layer passes the frame up to the Intranet layer which shall relay the packet and send back an input triggering t1;
- *e1*–input x_2: TP timer times out; output y_2: null;
- *e2, e8*–input x_3: tester sends a URR Command with a P/F=1 and the destination address other than the IUT's address, output y_3: null; TP timer started, acknowledgment timer stopped if running;
- *e3*–input x_4: acknowledgment timer times out, output y_4: acknowledgment failure to the Intranet Layer;
- *e4, e9*–input x_5: destination acknowledges the frame; output y_2: null;
- *e5*–input x_4: acknowledgment timer times out; output y_5: frame retransmitted, acknowledgment timer restarted;
- *e6*–same as e2 and e8 except that an outstanding frame is buffered;
- *e7*–input x_2: TP timer times out; output y_5: frame retransmitted, acknowledgment timer started.

In this example $F = 1$, $c_1 = 1$, $T_1 = T_{1,1} = \{t1\}$, $U_{1,1} = \{u1, u2, u3, u4, u5\}$, $A_1 = \{a_{1,1}\}$, and $O_1 = \{o_{1,1}\}$. Transition $t1$ is the one that needs to be triggered indirectly, as described in Section 2. Graph G contains several invalid paths. Any test sequence including those paths will not be realizable in a test laboratory. For example, consider the following paths:

- **t1, e5, e9**–t1 cannot be triggered without prior traversal of u1, u2, u3, u4, or u5;
- **u5, t1, u1, e3, e2, e1**–after executing e3, t1 will trigger because of an input buffered as a result of traversing u1. Therefore, e2 cannot follow e3 in the path before a buffered input is consumed by t1.

After applying the refined version of the algorithm from Section 6 to G (Figure 4 (b)), each state S is replaced with its two copies in G': $S.0$ for the empty buffer, and $S.1$ for the buffer containing an input from the Intranet Layer. Since G' is a minimal valid graph for the SUT, it no longer contains invalid paths.

The sets E, E', and E'_c are as follows:

$$
\begin{aligned}
E &= \{e1, \ldots, e9, u1, \ldots, u5, t1\} \\
E' &= \{e1.0, e1.1, e2.0, e3.0, e3.1, e7.0, e7.1, \\
&\quad\; e8.0, e8.1, e9.0, e9.1, u1.0, \ldots, u5.0, t1.0\} \\
E'_c &= \{e1.0, e2.0, e3.0, e7.0, e8.0, \\
&\quad\; e9.0, u1.0, \ldots, u5.0, t1.0\}
\end{aligned}
$$

Given sets E' and E'_c, the Aho et al. optimization technique gives the following minimum cost test sequence (the suffixes of edges are dropped):

$$
\begin{aligned}
&u5, t1, e6, u2, e7, e3, t1, e6, e7, e9, u5, t1, u1, e3, t1, \\
&e5, e8, u3, e1, t1, e5, u4, t1, e5, e3, u5, t1, e4, e2, e1 \qquad (2)
\end{aligned}
$$

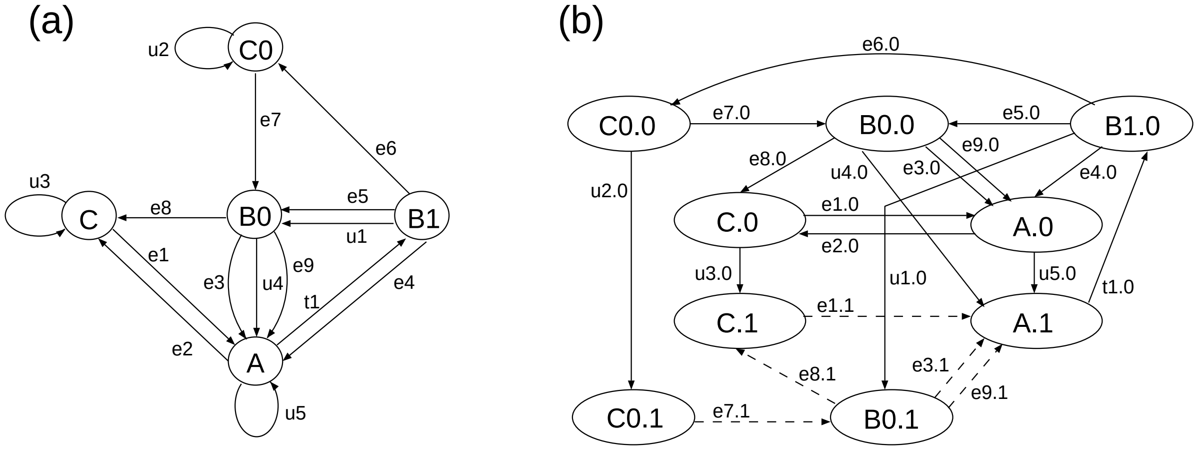

Figure 4: 188-220B Datalink layer: Graphs G (a) and G' (b) representing an FSM for the IUT.

8 Conclusion

This paper considers testing Army communications protocols in a testing environment with limited controllability. A tester has varying degrees of controllability on the inputs needed from multiple entities communicating with the IUT. The inputs fall into three categories: directly controllable, semicontrollable, or uncontrollable. Most test generation techniques are limited to directly controllable inputs. The presented graph conversion algorithm [15, 16] is capable of also utilizing the semicontrollable inputs, which helps increase the number of testable transitions. Practical considerations allow to avoid an exponential growth of a test suite, resulting in a linear test sequence length.

The research was motivated by work to generate tests for MIL-STD 188-220B. The number of testable transitions for 188-220B Class A–Type 1 Datalink Service module is approximately 200 without utilizing the semicontrollable inputs, which accounts for only 30% of all transitions defined in the protocol specification. The presented methodology makes it possible to increase the number of testable transitions to over 700. Combined with our previous work on testing protocols with timing constraints [18, 19], the methodology allows to generate tests free of interruptions due to timeouts and a coverage of more than 95% of the transitions defined in the specification.

References

[1] H. Ural, “Formal methods for test sequence generation,” *Computer Communications*, vol. 15, pp. 311–325, Jun 1992.

[2] R. E. Miller and S. Paul, “Structural analysis of protocol specifications and generation of maximal fault coverage conformance test sequences,” *IEEE/ACM Trans. on Networking*, vol. 2, pp. 457–470, Oct 1994.

[3] A. V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar, “An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours,” *IEEE Trans. on Communications*, vol. 39, pp. 1604–1615, Nov 1991.

[4] J. Tretmans, “Conformance testing with labelled transitions systems: Implementation relations and test generation,” *Computer Networks and ISDN Systems*, vol. 29, no. 1, pp. 49–79, 1996.

[5] H. Ural and B. Yang, “A test sequence selection method for protocol testing,” *IEEE Trans. on Communications*, vol. 39, no. 4, 1991.

[6] R. J. Linn and M. U. Uyar, *Conformance Testing Methodologies and Architectures for OSI Protocols*. Los Alamitos, CA: IEEE Comp. Soc. Press, 1994.

[7] E. Brinksma, “A theory for the derivation of tests,” in *Proc. IFIP Protocol Specification, Testing, and Verification (PSTV)*, Amsterdam: North-Holland, 1988.

[8] D. Lee and M. Yannakakis, “Principles and methods of testing finite state machines—a survey,” *Proc. of the IEEE*, vol. 84, pp. 1090–1123, Aug 1996.

[9] W. Y. L. Chan and S. T. Vuong, “An improved protocol test generation procedure based on UIOs,” in *Proc. ACM SIGCOMM*, Sep 1989.

[10] B. Sarikaya, G. von Bochmann, and E. Cerny, “A test design methodology for protocol testing,” *IEEE Trans. on Software Engineering*, vol. 13, pp. 518–531, May 1987.

[11] ISO, Information Technology—OSI, Geneva, Switzerland, *ISO International Standard 9646: Conformance Testing Methodology and Framework*, 1991.

[12] DoD, *Military Standard—Interoperability Standard for Digital Message Device Subsystems (MIL-STD 188-220B)*, Jan 1998.

[13] ISO/IEC, *International Standard ISO/IEC 8802-2, ANSI/IEEE Std. 802.2*, 2nd ed., Dec 1994.

[14] M. A. Fecko, P. D. Amer, A. S. Sethi, M. U. Uyar, T. Dzik, R. Menell, and M. McMahon, “Formal design and testing of MIL-STD 188-220A based on Estelle,” in *Proc. IEEE Military Comm. Conf. (MILCOM)*, (Monterey, CA), Nov 1997.

[15] M. A. Fecko, M. U. Uyar, A. S. Sethi, and P. D. Amer, “Issues in conformance testing: Multiple semicontrollable interfaces,” in *Proc. IFIP Joint Int'l Conf. FORTE/PSTV*, (Paris, France), pp. 111–126, Nov 1998.

[16] M. A. Fecko, M. U. Uyar, A. S. Sethi, and P. D. Amer, “Conformance testing in systems with semicontrollable interfaces,” Tech. Rep. TR-98-18, CIS Dept., University of Delaware, Newark, DE, 1998. (submitted for journal publication).

[17] J. K. Lenstra and A. H. G. Rinnooy Kan, “On general routing problems,” *Networks*, vol. 6, pp. 273–280, 1976.

[18] M. U. Uyar, M. A. Fecko, A. S. Sethi, and P. D. Amer, “Generation of realizable conformance tests under timing constraints,” in *Proc. IEEE Military Comm. Conf. (MILCOM)*, (Bedford, MA), Oct 1998.

[19] M. U. Uyar, M. A. Fecko, A. S. Sethi, and P. D. Amer, “Testing protocols modeled as FSMs with timing parameters,” *Computer Networks*, vol. 31, pp. 1967–1988, Sep 1999.