FORMAL DESIGN AND TESTING OF MIL-STD 188-220A BASED ON ESTELLE

Mariusz Fecko, Paul D. Amer, Adarshpal S. Sethi
Computer and Information Science Department
University of Delaware, Newark, DE

M. Ümit Uyar
Department of Electrical Engineering
City College of New York, New York, NY

Ted Dzik, Raymond Menell
US Army CECOM, Fort Monmouth, NJ 07703

Mike McMahon
ARINC, Inc., Shrewsbury, NJ 07702

Abstract

This paper describes the Estelle specification of MIL-STD 188-220A [7] Intranet Layer as well as a methodology for generating test sequences for checking the conformance of a protocol implementation to its specification. The methodology for deriving test cases from Estelle specification, which serves as input to test generation techniques, is presented. A Chinese postman tour *[1] is used to determine a minimum-cost tour of the transition graph for various transition types. Finally, the paper discusses several controllability and optimization issues that need to be addressed in test cases generation for intranet and datalink layers of MIL-STD 188-220A.*

Key words: *Estelle, protocol specification, conformance testing, test cases generation, protocol testing.*

1 Introduction

One of the University of Delaware's (UD) major contributions to the ATIRP is the study of formal specification languages in the specification and testing of Army communication protocols. In 1989, Estelle was approved as one of two ISO International Standard Formal Description Techniques (FDT) for the specification of computer communication protocols [5, 6]. Based on communicating extended finite state machines, Estelle has a formal, mathematical, implementation-independent semantics. It is an expressive, well-defined, well-structured language that is capable of specifying distributed, concurrent information processing systems in a complete, consistent, concise and unambiguous manner. An Estelle specification aims at discovering and resolving ambiguities in the original English document that would cause interpretation problems for implementors. The Estelle specification as a model of a communication protocol then can be used as input to conformance test generation techniques. Since Estelle makes it possible to create a complete and unambiguous protocol model, the test cases generated from it achieve high fault coverage.

2 Specifying MIL-STD 188-220A Data Link and Intranet Layers

For the past ten years, faculty and students at UD's Protocol Engineering Lab have been researching problems and developing tools that facilitate the general use of Estelle in designing and bringing communication network protocols to market. Most recently, we have been investigating one specific suite of protocols: MIL-STD 188-220A [7].

Originally designated 188-220 [8], this protocol suite was a joint services interoperability standard for digital message transfer device subsystems. It evolved into 188-220A to become the standard for interoperability of command, control, communications, computers, and intelligence (C4I) over Combat Net Radio (CNR). 188-220A is a key component of the Army Technical Architecture (ATA) for the digitized battlefield, and will likely become so

*"This work supported, in part, by the US Army Research Office Scientific Services Program administered by Battelle (DAAL03-91-C-0034), by the US Army Research Office (DAAL03-91-G-0086), and through collaborative participation in the Advanced Telecommunications/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." Dr. Uyar, a Research Professor with CCNY, is presently Visiting Associate Professor at University of Delaware.

in the Joint Technical Architecture (JTA). There are several synergistic efforts to design 188-220A to be complete, correct, unambiguous, and performing suitably well.

To ensure that the 188-220A standard is free from ambiguities which might cause implementation problems, we used Estelle to create an unambiguous specification of the Data Link Layer as specified in the 27Jul95 version of the standard [3]. This specification effort was divided into three subprojects specifying: Type 1: Connectionless (CL) Operation (unack'ed and coupled ack'ed); Type 2 Connection-mode Operation, and Type 4: Decoupled Ack'ed Connectionless Operation.

Most recently, UD's Protocol Engineering Lab has been working on specifying 188-220A's Intranet Layer which resides above Data Link and below IP. All radios tuned to the same radio channel make up an Intranet. Conceptually when any one radio transmits, its signal is physically broadcast to all of the others. However, due to geographic (or other) obstacles, some radios within an Intranet may be unable to communicate with others. These 'down' links and the Intranet topology in general may be temporary, particularly during highly dynamic battlefield situations. As a result, routing issues exist both within an Intranet and between Intranets (i.e., IP).

In the process of developing the Estelle specifications of the Data Link layer and, most recently, the Intranet Layer, a large number of problems (> 50) in the original English specification have been documented. These problems have been reported back to the CNR Implementation Working Group, the official group that meets roughly bi-monthly near Ft. Monmouth, NJ and is responsible for the evolving 188-220A document.

Examples range from wording ambiguities such as:

- "... a station shall wait for some period of time *bounded by the probability* of the remote ack time expiration."
- " ... RR command is sent to a destination when the k value at the originating station reaches half of the k value for that connection."[1]

to more serious concerns such as:

- Intranet routing was originally defined based on spanning trees of the Intranet topology. However the draft standard's examples did not comply with the mathematical definition of a spanning tree.
- The Intranet Layer allows a station to enter *Quiet mode* whereas the Data Link layer refers to a station being in *response mode off*. It was unclear how these two terms differ, if at all.

3 Estelle Specification of the Intranet layer

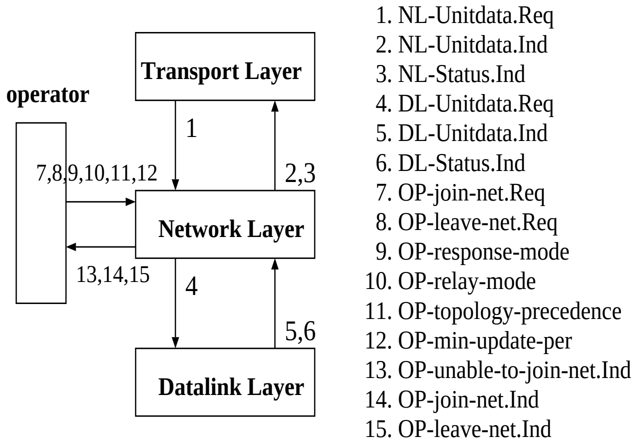

Figure 1: Network Layer Interface

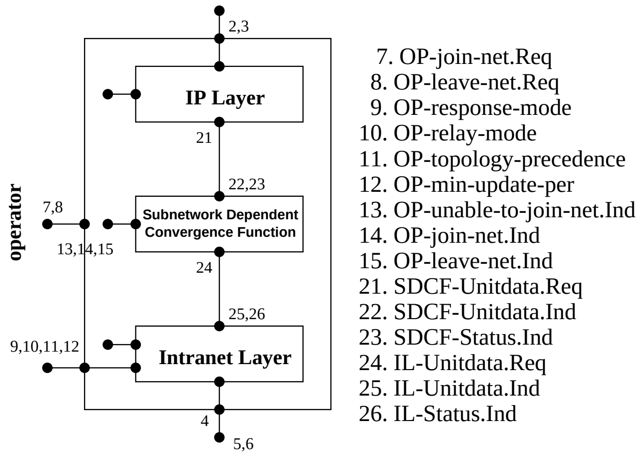

Figure 2: Network Layer Architecture

Due to page limitations, we are unable to include actual full Estelle specifications in this paper. For the more detailed description of the semantics of Estelle specification components (communication channels, interactions, etc), the interested reader may consult our paper on Datalink Layer specification [3]. We instead present a brief overview of Net-

[1] Both references to k refer to the same variable.

work Layer architecture with a focus on the Topology Update function of the Intranet layer.

Figures 1 and 2 show the interface and general architecture of the Network Layer, which consists of Internet (IP) Layer, Subnetwork Dependent Convergence Function, and Intranet Layer. This represents the protocol stack at a single station, as well as an interface with "operator module" which can interact with several different layers in the stack. The operator module abstracts the link layer's interactions with both a human operator and a system management process.

3.1 Intranet Layer Architecture

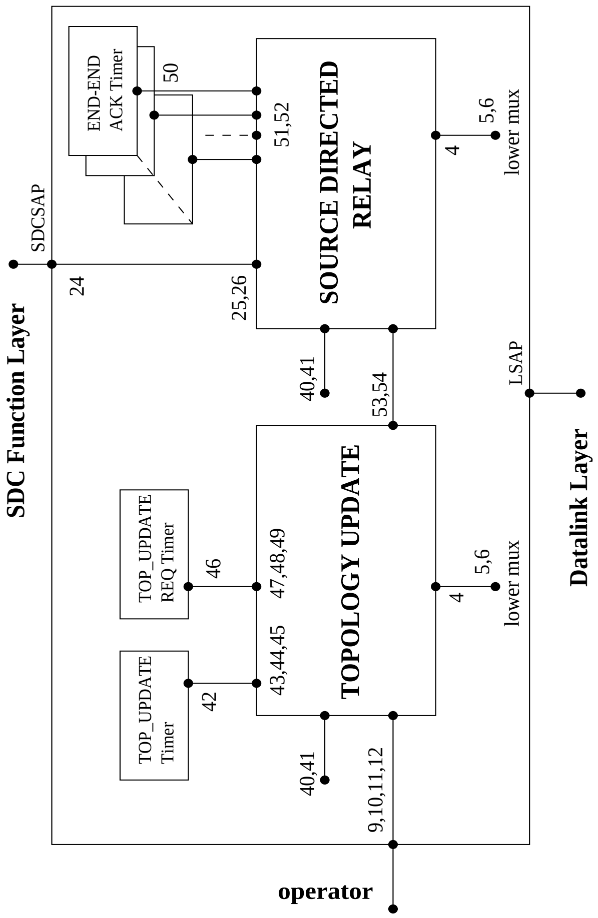

Figure 3: Intranet Layer Architecture

Figure 3 shows the internal structure of the Intranet Layer. The two main Intranet Layer functionalities: Source Directed Relay (SDR) and Topology Update exchange (TU), were encapsulated in two separate component modules of the Intranet Layer module. Not only does this simplify the design of the FSMs that model the entire layer, it also allows for generating test cases for each functionality separately.

The SDR module receives *IL_Unitdata_Req* messages through *SDCSAP* interaction point. It interacts with a varying number of *END_END_ACK* timers, one for each IP packet that has been sent but not yet acknowledged. The TU module interacts with the SDR module by notifying it of any topology changes that take place dynamically. The TU module communicates with two timers: *Topology_Update Timer* and *Topology_Update_Request Timer*. The former is started after a topology update message is sent by the station. According to 188-220A, a station is not allowed to send another topology update message until the timer timeouts. The latter performs the same role for topology update request messages.

Both SDR and TU modules can send and receive messages from the datalink layer through their *lower_mux* interaction points - the messages from the two modules are multiplexed by the parent Intranet Layer module.

3.2 State Diagram and Transition Table for Topology Update

Each module in an Estelle specification is modeled as a communicating EFSM (Communicating Extended Finite State Machine) after careful analysis of its behavior. Since in section 4 we apply our test generation techniques to the TU module, in this paper we restrict ourselves to describing the corresponding EFSM for this module (see figure 4).

There are five states in the EFSM. The four active states are defined based on the status of the *Topology_Update Timer* and *Topology_Update_Request Timer*. Each timer may or may not be running at a given point in time - this gives four possible configurations. The timers' status determines the I/O behavior of the TU module, because a running timer prevents it from sending certain interactions. For example, when the topology information changes, the station sends topology update message only if the *Topology_Update Timer* is not running.

The final EFSM for the TU module consists of 5 states and 86 transitions.

4 MIL-STD 188-220A Test Case Generation

The Army Communications-Electronics Command (CECOM) Distributed Integrated Laboratory (DIL)

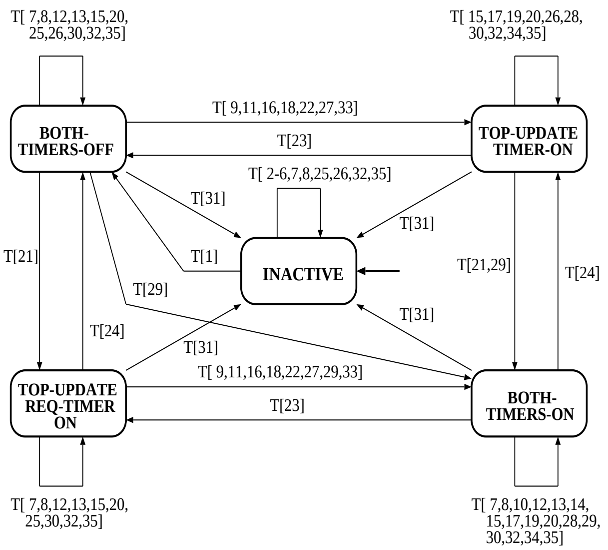

Figure 4: Extended FSM for Topology Update

is responsible for certifying that all systems participating in Task Force XXI are interoperable. To perform this responsibility for systems communicating over CNR, the DIL needs 188-220A protocol test tools. The CECOM Software Engineering Directorate is developing a Conformance Tester that automatically evaluates a 188-220A implementation identifying where it differs from the standard. This information can be used as a first step in the DIL's certification process, or to objectively categorize a 188-220A implementation to guide future implementations and standard evolution. While the Conformance Tester will be used for Task Force XXI, the capability will serve the Army and the Army's joint requirements for years to come.

In support of this task, the UD's Protocol Engineering Lab is developing test scripts to be used by the 188-220A Conformance Tester. The test scripts specify a logical sequence of test steps that must be performed by the Conformance Tester to individually test the Data Link Layer (Types 1, 2 and 4 procedures) and Intranet Layer.

The test scripts will be used as input to the Conformance Tester which in turn will stimulate a Implementation Under Test (IUT), and assess responses to determine if the implementation has correctly implemented the protocols. Since it is known to be theoretically impossible to exhaustively test an implementation, checks should be made on those events that affect state/transition, boundary conditions, and stress points. Further, since there are no hooks into the IUT, behavior is observed externally (black box testing). The test scripts are to be structured as independent modular components to facilitate modifying and adding to the scripts in response to the continuing evolution of 188-220A.

4.1 Black Box Testing

In conformance testing, the implementation under test (IUT) is viewed as a *black box* whose behavior is characterized by a set of observable actions, called *outputs*. Outputs are generated by applying a set of externally controllable inputs. A black-box model for a protocol is represented as a pure finite-state machine (Pure FSM) [2].

4.2 Transition types and testing order

For testing purposes, transitions in EFSM are didvided into three groups:

- *valid transitions:* defined for the "normal" (or expected) behavior of the tested entity;
- *inopportune transitions:* inputs are semantically and syntactically correct, but arrive unexpectedly (or out of sequence) in a given state;
- *illegal input transitions:* inputs do not meet the syntax requirements and signal a faulty behavior from the far-end entity or from the transmission channel, e.g., invalid version number in intranet header.

We want to test each class of transitions separately. In *valid transitions* testing, in any given state each distinct input to an IUT corresponds to a state transition in the corresponding EFSM model. *Inopportune transitions* testing is performed after all valid transitions have been tested successfully. The transitions in our specification may model all or a subset of possible inopportune transitions.

Similarly, *illegal input transitions* testing is performed after valid transitions testing is successful. Since there are infinitely many illegal inputs, we test the subset that includes only the most likely and/or the most important ones. This may involve some subjectivity on the part of 188-220A experts.

Single transition testing consists of three steps:

- for a transition $v_i \longrightarrow v_j$, put UIT into state v_i;
- apply required input and compare the output generated with the one defined in the specification;
- verify that the new state of the EFSM is v_j (optional).

The entire methodology proposed for obtaining test cases consists of four phases.

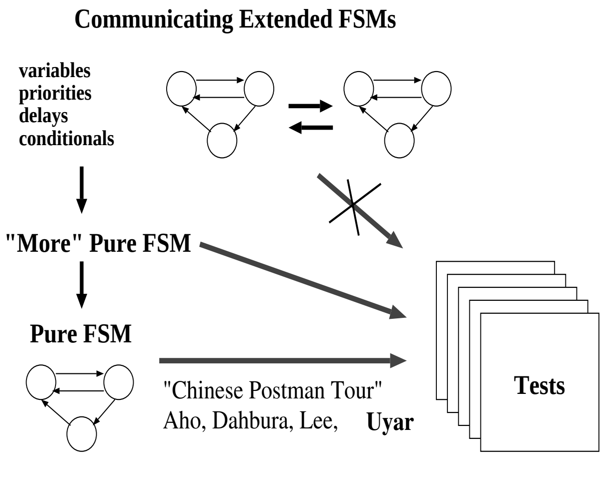

Figure 5: Test generation from Extended FSM

4.3 Phase 1 - EFSM expansion

To apply existing test generation techniques to the system modeled as an EFSM, the first step in our approach is conversion of the Estelle specifications (which are EFSMs) into more pure FSMs (figure 5).

Ideally, the original EFSM of 5 states and 86 transitions should be converted to pure FSM. Suppose the EFSM contains a set of variables $VAR = \{v_1, v_2, ..., v_n\}$. Each variable has a corresponding set of values. The set of pairs $(variable, value)$ with one entry for each variable is called a *configuration*.

Given the set VAR, each state s from the original EFSM is replaced by a set of pairs $(s, configuration)$ for all possible configurations. Similarly, each original transition is replaced by a set of transitions according to the way an original transition changed/retained the value of each variable.

This brute-force approach often leads to the well-known state and transition explosion problem. The number of states and transitions in the expanded pure FSM may be infeasibly large. Therefore, based on our knowledge of the protocol, we compromize and consider only variables that are directly involved in Estelle $PROVIDED$ clauses and that influence the output generated by IUT.

This procedure converted the original EFSM to a new one with 20 states and 648 transitions. After detecting and removing unreachable states and transitions between them, we ended up with the EFSM of 17 states and 612 transitions. The new EFSM is not a pure FSM, it is a closer approximation of a pure FSM than the original one. And fortunately this closer approximation does allow for some test case generation.

With complete testing theoretically not possible, initial practical concerns defined the test criterion that we need to satisfy as *"test each transition in the expanded EFSM exactly once."*

4.4 Phase 2 - valid transitions testing

The transitions in the expanded EFSM were divided into 525 valid transitions and 87 inopportune transitions. Each transition has a corresponding cost (e.g., transitions that involve heavy computations and take longer are assigned higher cost).

The test method that we then use is a *Chinese Postman Tour* that includes all valid transitions [1]. The method was developed at AT&T Bell Labs and applied successfully to testing various other communications protocols including ISDN switches, PBXs and terminals. The method produces a minimum-cost transition tour that starts and ends in the initial state, and that includes each valid transition at least once. The tour length for our TU module is 815 transitions without state verification, and $\approx 2,000$ with state verification.

4.5 Phase 3 - inopportune transition testing

All inopportune transitions are assumed to be *self loops* (i.e. start and end state are the same). To test each of them, we need to build a tour that visits each state at least once and tests a number of inopportune transitions while the IUT is in this state.

One of the problems that may occur during transition tour traversal, and that we are currently researching, is the inability to execute all self-loop transitions of a given state during one visit to this state. Since the internal structure of an IUT is unknown to the testers, the EFSM model cannot possibly capture all implementation details such as internal timers. Therefore, during testing it may not be possible to stay in a given state for the time necessary to execute all self-loop transitions (e.g., due to premature timeouts). We designed an algorithm that builds a minimum-cost transition tour that solves this problem[2].

The expected tour length is 100 transitions without state verification; ≈ 200 with state verification.

4.6 Phase 4 - illegal input testing

Our current assumption is that all illegal messages are self loops. In reality, some implementors may

[2]paper in progress

choose a different approach. For example, an IUT may go to an error state after detecting an illegal input.

As with inopportune transition testing, we need to visit each state at least once and test a number of illegal transitions while an IUT is in this state. We encounter the same theoretical problem due to timeouts - since it is likely that we will test many illegal messages in a given state, we may need to visit some states more than once.

5 Ongoing research issues

5.1 Observability and controllability

Ideally, testers should be able to generate every possible input message that is defined in the FSM for an IUT. Similarly, the output messages generated by an IUT should be observable by the testers.

The basic framework for conformance testing is known as the *local method* (figure 6). A basic assumption of the local method is that exposed interfaces exist above and below the IUT. These interfaces serve as *points of control and observation* (PCOs); i.e., points at which a real system can control inputs to and observe outputs from an IUT. In practice, testers may not have direct access to the interface(s) between the lower/upper tester and an IUT. There are three other testing frameworks that address this issue [4].

Based on the feedback that we got from CECOM's 188-220A's Conformance Tester developers, we concluded that an explicit upper tester is unlikely to exist in the testing framework. This may make transitions that are fired by inputs coming from upper layers untestable. The same issue exists in testing of 188-220A datalink layer. However, the implementation will contain both intranet and datalink layers. The part of datalink layer will constitute the IUT that contains several transitions fired by inputs coming from the intranet layer. We are currently investigating the possibility of using the implemented intranet layer as an implicit upper tester for the IUT. The generated transition tour for the IUT may have transitions that generate inputs for intranet layer. Using those inputs as stimuli to intranet layer, we may be able to force it to generate outputs that are necessary to fire the IUT transitions that we want to test.

An example of controllability problems may be a valid transition with the following *input/event* column in the transition table:

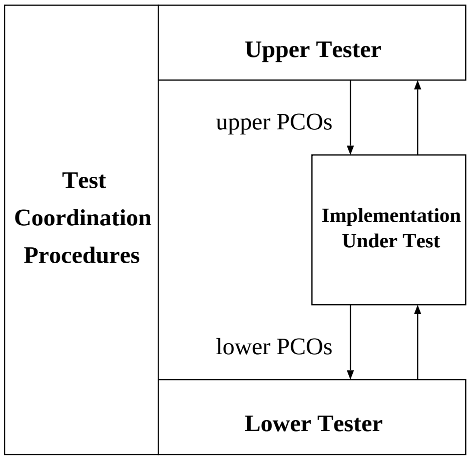

Figure 6: Local method

```
receive: DL-Unitdata.Ind and
(intranet message type is topology update) and
(topology update is not obsolete) and
(receiver's topology information modified)
```

To test this transition, the tester must be able to perform the following steps:

- generate required input - the *DL_Unitdata_Ind* from datalink layer;
- specify message type - *TOPOLOGY_UPDATE*;
- specify the *topology update id* and the sender's datalink address;
- specify the *topology* in a message;
- must know the current topology information of the IUT;
- observe messages sent by IUT to datalink layer;
- optionally verify if EFSM remained in its state.

5.2 Limiting subtour length

A **subtour** is a sequence of transitions from a full transition tour that starts and ends in the initial state.

For practical testing reasons, it is important that the subtour length be limited. Because of limited controllability of the IUT, testers may not be able to execute an arbitrarily long transition sequences without resetting the IUT to the initial state. On the other hand, because setting up a new test is costly, we do not want short subtours that consist of one or two transitions.

Currently, the *Chinese Postman Tour* is generated without addressing this problem. Therefore, some subtours are long (over 200 transitions), whereas several others are only one or two transitions long. We are investigating this problem with a view to generating a transition tour whose subtour lengths are more uniformly distributed.

6 Conclusion

In the Army effort to develop MIL-STD 188-220A, it is essential that the implementations of the protocol conform to its specification. In the paper we have described the technique for deriving test cases for conformance testing from the Estelle specification of the protocol. We used a powerful and efficient method of *Chinese postman tour* that allows automatic generation of minimum-cost test sequences.

We have also identified several research issues that are being addressed with FY97 support. These issues arise from the limited controllability and observability of an IUT and practical restrictions on the transition tour length and form.

Initially, the test generation method was applied to the Intranet Layer of MIL-STD 188-220A. Our immediate goal, which is consistent with suggestions by CECOM's Conformance Tester developers, is to produce test cases for the Data Link Layer Types 1-4.

"The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government."

References

[1] A.V. Aho, A.T. Dahbura, D. Lee, M.U. Uyar. *An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours.* IEEE Transactions on Communications, 39(11), Nov 1991.

[2] M.H. Sherif, M.U. Uyar. *Protocol Modeling for Conformance Testing: Case Study for the ISDN Protocol.* AT&T Technical Journal, Jan/Feb 1990.

[3] P. Amer, G. Burch, A. Sethi, D. Zhu, T. Dzik, R. Menell, M. McMahon. *Estelle Specification of MIL-STD 188-220A DLL.* Proc MILCOM 96, Oct 1996.

[4] R.J. Linn. *Conformance Testing for OSI protocols.* Computer Networks and ISDN Systems, 18, 1989/1990.

[5] S. Budkowski, P. Dembinski. *An Introduction to Estelle: A Specification Language for Distributed Systems.* Computer Networks and ISDN Systems, 14(1), 1987.

[6] *ISO International Standard 9074: Estelle - A Formal Description Technique Based on an Extended State Transition Model.* Information Processing Systems - Open System Interconnection, 1989.

[7] *Military Standard - Interoperability Standard for Digital Message Device Subsystems (MIL-STD 188-220A), 27Jul96.*

[8] *Military Standard - Interoperability Standard for Digital Message Device Subsystems (MIL-STD 188-220), 7May93.*