ISSN 2250-3153

Survey of Network Protocol Verification Techniques

Ms. Veena Bharti, Dr. Sachin Kumar

Research Scholar, IFTM Moradabad
Professor (CS Deptt.) AKG Engineering College, Gzb, India

Abstract- In the world of designing network protocols, verification is a crucial step to eliminate weaknesses and inaccuracies of effective network protocols. There are many models and tools to verify network protocols, including, Finite State Machines (FSM), Colored Petri Nets (CP-Nets), Temporal Logic, Predicate Logic, Estelle Specification, Path based Approach etc. This paper presents a survey of various techniques for verifying correctness properties of communications protocol designs.

Index Terms- protocol verification, validation, survey methods, petri nets.

I. INTRODUCTION

The advance in protocol specification using Formal Description Techniques (FDTs)[1] in recent years has opened up a new horizon for protocol verification. Three FDTs – SDL (Recommendation Z.100, 1987), Estelle (ISO/IEC 9074, 1989) and LOTOS (ISO/IEC 8807, 1989) – have been standardized. One of the main aims in developing FDTs is to have a formal specification acting as a sound basis for such verification. Academic research has recently made significant advances in the generation of test sequences from formal specifications and in the development of computer-aided tools, with the aim of improving the effectiveness of verification.

There is not much progress in the techniques for practical verification of communication networks. There is a big gap between verification practice and research results published in journals and reported at conference. This big gap between academic and industrial practices is the fact that academia has not been addressing the verification issues and problems account for the fact that academic verification methods are seldom used in industry. To help the industrialization of academic techniques, empirical studies and a new orientation towards real issues need to be given higher priority by researchers and field experience need to be more frequently reported. This paper presents a literature survey of communication protocol verification. The conclusion section summarizes our findings for each of these topics.

II. BACKGROUND, CONCEPTS AND TERMINOLOGY

This section describes some concepts and terminology related to communications protocols. An understanding of these concepts is necessary for understanding the various techniques described in the paper.

A. Protocol Specification, Verification and Validation

In layer-structured communication networks, protocols are defined as a set of rules which govern the exchange of messages through interactions of partners or processes to achieve two goals. One is to provide a particular set of services to its local protocol layers above. The other goal is to furnish a set of logical rules or protocols to its remote peer partners on other machines. In the area of specification, the former is called service specifications; the latter is called protocol specifications. Protocols can be verified against their design and implementation. In other words, protocol can be verified either during the design phase before the system is implemented, or during the testing and simulation phase after the system has been implemented. Since design verification can detect design errors at the early stage, unnecessary or incorrect implementation can be avoided. Therefore, design verification has the potential to significantly reduce the cost of protocol development and testing. With respect to design verification, the work can be divided into two tasks: service-specification verification, and protocol specification verification. Since service specifications vary from system to system, different procedures are required for different protocols. Therefore, the protocol-specification verification has been the focus of most of researchers; and is discussed in the remainder of this paper. In summary, protocol verification is a task which attempts to detect the existence of logic errors in the protocol design specification at the early development stage. Some papers[l7,18] refer to this logic verification as "validation". Since this is a survey paper, no distinction is made between these two. No matter which term is used, we are concerned primarily with the logical structure or syntax of protocols as opposed to their semantics or intended functions.

III. WHAT TO VERIFY – TYPES OF PROTOCOL ERRORS

There are six general types of protocol syntax errors. They are defined as follows:

1) ***Unspecified Reception:*** An unspecified reception is a reception that is executable but not specified in the design. Since the required reception is not specified in the design, the occurrence of it means the subsequent behavior of the system interaction is unpredictable.
2) ***Nonexecutable Interaction:*** **A** nonexecutable interaction is a transmission or reception that is specified in the design but never be executed. It is oftenly called dead code.
3) ***Deadlock:*** A state deadlock occurs when each and every processes can only remain indefinitely in the same state. In other words, a deadlock is a global state reachable from the initial global state, with all channels empty, in which no transmissions are possible. Deadlocks must be avoided, since once the system has arrived in a deadlock state, it is blocked forever.
4) ***Livelock:*** A livelock is a situation where several processes keep exchanging messages but without any effective work being accomplished toward performing

its services. It is sometimes called "dynamic deadlock". Livelock has to be avoided, since once it occurs, the system is locked forever into a small set of global states. It is sometimes called Temp o-blocking .

5) ***State Ambiguity:*** A state ambiguity exists when a state in one process can coexist stably (i.e. reachable with channels empty) with more than one state in another process. In practice, processes within a communication network often execute in highly synchronous manner. Thus, although state ambiguities do not necessary represent errors, they should be more carefully treated.
6) ***Overflow:*** An overflow is a channel state such that the number of messages in the channel is not bounded by a predefined positive integer number. In other words, any communications channel has a storage capacity that limits the amount of information it can carry at any instant. An attempt to exceed this capacity may result in loss of data being transferred among processes. Overflow may occur in two different situations: finite overflow, or infinite overflow. Finite overflow indicates that channels are bounded by one number, although not the predefined one; whereas infinite overflow indicates that channels are never bounded. While the former may be dynamically solved; the latter gives a serious sign of existence of potential design errors. In addition, a protocol is "well-formed" *iff* it contains no unspecified reception and no nonexecutable interaction. A protocol is "live" *iff* it is free from deadlocks and unspecified receptions. In other words, the protocol will progress indefinitely if it is live. Finally, a protocol is "safe" *iff* it is live and always terminates properly.

IV. SURVEY METHODS

In 1988 an **Algorithmic procedure** to check whether a protocol provides a desired service is developed.This method describes protocols as a collection of interacting processes. For modeling the interaction between processes, the synchronous interprocess input/output operations proposed by Hoare in his language of communicating sequential The safety behavior of the processes is described using FSM's whereas the liveness behavior is described using propositional temporal logic. The reason for choosing FSM's for the safety behavior is that it typically consists of simple actions in response to numerous events such as user commands, message arrivals, and time-outs **[3].** On the other hand, the liveness properties of many processes, such as the communication medium, require the description of infinitely long sequences of messages, which is hard to do using FSM's, but is easy to do with temporal logic.

In 1990 Petri nets [7] are widely used for modeling and analysis of concurrent processing systems. In order to describe certain fundamental properties of concurrent systems, such as eventuality and fairness, in Petri net models, Suzuki [lo] introduced temporal Petri nets in which temporal constraints of a given net are represented by formulas containing temporal operators, such as 0 (eventually) and ***0*** (henceforth). Temporal Petri nets are Petri nets in which certain restrictions on the firings of transitions are represented by formulas containing temporal operators. The temporal Petri net which models the protocol is analyzed formally by using the existing theory of w-regular expressions and Buchi-automata.

In 1993 the formal **Communication Finite State Machine (CFSM) model[4]** is developed, a communication protocol consists of several communicating entities which can be represented in some CFSMs. Global state reachability analysis is one of the most straightforward ways to automatically detect logical errors in a communication protocol specified in the CFSM model. Global state reachability analysis generates all of the reachable global states and checks the correctness one by one. Even though communication protocols are error free, global state reachability analysis still needs to be executed completely. Thus backward protocol verification method is proposed, to detect logical errors. By analyzing the properties of deadlock error, unspecified reception error, and channel overflow error, some candidate erroneous global states are generated. Then, each candidate global state is checked whether there is a path, i.e., a global state sequence, connects to the original initial global state. If there is a path, then the candidate global state is really an erroneous global state and the communication protocol does have some logical errors. Otherwise, if there is no candidate global state or none of the candidate global state has a path, then the communication protocol is error free.

In 1994 A number of protocol verification reduction techniques were proposed in the past. Most of these techniques are suitable for verifying communicating protocols specified in the Communicating Finite State Machine (CFSM) model. However, it is impossible to formally specify communicating protocols with predicate and variables using the CFSM model. The **Extended Communicating Finite State Machine (ECFSM) model[5]**, which incorporates the mechanism for representing variables and predicates, can formally model communicating protocols with variables and predicates. To have more efficient verification for ECFSM-specified protocols, an integrated ECFSM-based global state reduction technique is given. This method is based on two techniques: the dead variables analysis which can reduce the number of global states, and the ECFSM-based maximal progress state exploration which can speed up the global state reachability analysis. Using ECFSM-based method, the maximal progress protocol verification can be directly applied to the Formal Description Techniques (FDTs) which are based on the extended state transition model, i.e., **ISO's** Estelle and CCITT's SDL.

In 1995 Estelle specifications involve translating the specifications into another form, such as finite state machines or Petri nets for which verification tools have been implemented. All of the Estelle verification tools impose some restrictions on the specifications to be verified: they use a subset of Estelle or restrict the complexity of the specifications that can be verified; or the specifications need to be in a variant of Estelle, rather than standard Estelle. In particular, dynamic behaviours and exported variables of an Estelle specification are not handled. Thus developed an approach, to verifying standard Estelle specifications by translating them into Numerical Petri Net (NPN)[10] specifications. The merits of this approach are that all dynamic behaviours, exported variables and most Estelle statements can be handled. The deficiencies are that *priority* and *delay* clauses and the *systemprocess* attribute can not be modelled.

In 2000 the concept of **concurrent path** is used in concurrent program testing [5]. Assume a concurrent system is composed of concurrent modules. The concurrent path is a partial-order representation, a combination of modules execution behavior, i.e., an ordered set of module paths. The full set of concurrent paths is included in the Cartesian product of the path sets of all modules, which are easily generated. The analysis to determine whether a member in the product is a concurrent path is performed independently. Hence, the memory requirement to generate the concurrent path depends on the complexity of individual module and individual concurrent path rather than that of the whole system. Therefore, it will concentrate on formalizing the concurrent path of a concurrent system to perform the verification. The new approach is called the ***path-based approach***[6]

In 2009 A method is proposed, that uses the **process algebra [13]** for protocol specification, and transformation rules for a translation of the specification into a Petri net while preserving the semantics of the specification. Petri nets are well-known formal method for their analytical power to deal with a problem of protocol verification: invariant, reachability, deadlock and liveness analysis. Elements of theory behind the method are sketched in a short way. The elegance of protocol specification by using the process algebra [10] and a powerful analysis by means of Petri nets [5] are main reasons for such the integration,

In 2010 a **Colored Petri nets** (CP-nets) [18] based method to integrate functionality correctness verification and performance analysis for network protocols is given. The central idea is that CP-nets based functional models for a protocol is constructed for correctness validation, and then this model is slightly modified by adding performance related temporal constrains and data monitor units, and finally simulation based performance analysis executes. CP-nets models used in both analysis processes are coessential, that is, every occurrence sequence in the performance model corresponds to an occurrence sequence in the functional model. So if functional models are validated and successfully translated to performance models, both of them would satisfy the functionality requirements of the protocol. Furthermore, CP-nets and its modeling and analysis tool, CPN Tools [3], have many strong capabilities used to facilitate modeling and analysis network protocols, e.g., flexible descriptive capability, visual feedback simulation, and effective analysis techniques. Adopting CP-nets as basic models to integrate functional verification and performance analysis for network protocols is feasible and effective.

V. CONCLUSION AND DISCUSSION

As we confirmed earlier, the field of verification and analysis needs more investigation, but as time goes on, a lot of research at different universities can be done to facilitate its correctness and improve its weaknesses. This paper has presented a survey on communication protocol verification. The aim is to provide pointers to practitioners on reports that could be useful and relevant to practical communication protocol verification.
We have focused our survey on the following topics:

Method	Year	Researcher
Algorithmic method	1988	Krishan Sabnani
Petri net	1990	Ichim *Suzuki*
CFSM	1993	Chung-Ming Huang and Duen-Tay Huang
ECFSM	1994	Chung-Ming Huang and Jenq-Muh Hsu
Estelle	1995	Ajin Jirachiefpattana and Richard Lai
Path based approach	2000	W.-C. Liu, C.-G. Chung
Process algebra	2009	Slavomır Simonak, Stefan Hudak and Stefan Korecko
Coloured Petri nets	2010	Jing LIU, Xinming YE, Jun ZHANG, Jun LI, Yi SUN

REFERENCES

[1] Yuang,M.C.; "Survey of protocol verification techniques based on finite state machinemodels",Computer Networking Symposium, 1988., Proceedings of the 11-13 April 1988 Page(s):164 – 172.

[2] Krishnan Sabnani, "An Algorithmic Technique for Protocol Verification", IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 36, NO. 8, AUGUST 1988,

[3] Lee, T.T.; Lai, M.-Y.; "A relational algebraic approach to protocol verification Software Engineering, IEEE Transactions on Volume 14Feb. 1988 Page(s):184 – 193.

[4] Chung-Ming Huang; Duen-Tay Huang,"A backward protocol verification method",TENCON'93 Proceedings. Computer, Communication, Control and Power Engineering.1993 IEEE Region 10 Conference on, 19-21 Oct. 1993 Page(s):515 - 518 vol.1

[5] Chung-Ming Huang; Jenq-Muh Hsu; Huei-Yang Lai; Jao-Chiang Pong; Duen-Tay Huang, "An incremental protocol verification method for ECFSM-based protocols", Proceedings of the Eighth Annual Conference on 14-17 June 1993 Page(s):87 – 97.

[6] Wen-Chien Liu; Chyan-Goei Chung;"Protocol verification using concurrent paths", Communications, Fifth Asia-Pacific Conference on ... and Fourth Optoelectronics and Communications Conference, Volume 2, 18-22 Oct. 1999 Page(s):1188 -1191.

[7] Ajin Jirachiefpattana and Richard Lai, "An Estelle-NPN Based System for Protocol Verification", 0-7803-2680-6/95/$4.00 , 1995 IEEE.

[8] EI maati Chabbar,Mohammed Bouhdadi "On Verification of Communicating Finite State Machines using Residual Languages" Proceedings of the first Asia International Conference on modeling and simulation(AMS,07), 2007.

[9] Edgar pek, Nikol Bogunovic "Predicate Abstraction in Protocol Verification", 8th International Conference on Telecommunication conTEL 2005, June 15-17, Zagreb, Croatia.

[10] Bhaskar Sardar, Debashis Saha," A Survey Of TCP Enhancements For Last-Hop Wireless Networks" 3rd Quarter 2006, Volume 8, No. 3, www.comsoc.org/pubs/surveys.

[11] Ulle Endriss " Temporal logic for representing agent communication protocol.

[12] W.-C. Liu, C.-G. Chung, "Path-based protocol verification approach", Information and Software Technology 42 (2000) 229–244 , www.elsevier.nl/locate/infsof.

[13] Wen-Chien Liu, Chyan-Goei Chung, "Symbolic path-based protocol verification", Information and Software Technology 42 (2000) 245–255, www.elsevier.nl/locate/infsof.

[14] Horatiu Cirstea, Pierre-Etienne Moreau, Antoine Reilles, "Rule-based Programming in Java For Protocol Verification", Electronic Notes in Theoretical Computer Science 117 (2005) 209–227, www.elsevier.com/locate/entcs.

[15] Imtiaz Ahmad, Faridah M. Ali, "A. Shoba Das, Synthesis of finite state machines for improved state verification", Electronic Notes in Theoretical Computer Science 117 (2005) 209–227, www.elsevier.com/locate/entcs.

[16] Slavomır Simonak, Stefan Hudak and Stefan Korecko, "Protocol Specification and Verification Using Process Algebra and Petri Nets", 2009 International Conference on Computational Intelligence, Modelling and Simulation.

[17] Vijay Gehlot and Carmen Nigro, "Colored Petri Net Model of the Session InitiationProtocol (SIP)", 978-1-4244-5226-2/10/$26.00 ©2010 IEEE.

[18] Meng Xiao-jing, Wang Li-Ji, "Analysis and Verification of Colored Petri Net in PPPoE Protocol", 2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE).

[19] Xinming YE, Jun ZHANG, Jun LI, Yi SUN, "Integrating Functional Verification and Performance Analysis for Network Protocols using CP-nets", 978-1-4244-7755-5/10/$26.00 ©2010 IEEE.

AUTHORS

First Author – Ms Veena Bharti, (MCA, M.Tech.), bharti.veena@gmail.com

Second Author – Mr. Sachin Kumar (M.Tech,P.hd) , AKGEC,Ghaziabad, imsachingupta@rediffmail.com