Protocol Engineering: An Historical Perspective

Gregor v. Bochmann

University of Ottawa
Bochmann@site.uottawa.ca

Abstract: During the 1970ies and '80ies, the first computer communication networks were designed and implemented in the research and commercial sectors. Many of the protocols developed during that time are still in use today. This paper starts by giving an overview of these developments. Then it concentrates on the development of protocol engineering, that is, the methods for the specification of communication protocols and services, the verification of protocols and their implementation and testing. After personal views of the developments in the 1970ies, the basic concepts developed at that time are explained. The standardization of Formal Description Techniques in the 1980ies is discussed in the following section. The purpose of the paper is to show the long way we have come and to suggest that many of the basic concepts have not changed too much during these years, although more detailed aspects have evolved and given rise to new technological developments.

1. Introduction

In the 1960ies, most computers were used in a batch processing mode. But this was also the time when the first time sharing operating systems were developed, allowing the use of a single computer by several interactive users. However, the transmission quality over the communication lines was not perfect, and it was common to experience transmission errors leading to wrong characters being received. Therefore, this was also the time when the first communication protocols were designed for obtaining reliable data transmission over unreliable communication links.

The next decade was the time when the first computer networks were developed, including the precursors of the Internet. This required a number of different communication protocols for different purposes, e.g. routing, and it became customary to introduce some form of hierarchical layering in order to organize the complexity of the protocols involved. The first protocol standard for computer networks was introduced by the International Telecommunications Union (ITU) under the name X.25, and many computer manufacturers developed their own proprietary communication protocols for their line of computers and terminals.

The 1980ies brought further technological developments, including personal computers and local area networks. But this time was also characterized by an effort to come up with global standards for computer communications, not only at the link and network layer, but also at the higher layers involving the applications. The OSI standardization effort, which had the goal to allow for the interworking between heterogeneous systems from different manufacturers, was however not successful. Finally, in the early 1990ies, the protocols of the Internet were largely adopted in practice, including the newly developed Web protocols HTML and HTTP and rudimentary Internet e-mail protocols developed in the 1970ies. Since then, many more protocols have been developed, e.g. for wireless networks and for various applications running over the Internet.

This paper does not so much deal with these protocols, but it gives a perspective of how the engineering methods that are used to design these complex protocols have been developed over the years. These engineering methods, sometimes called "protocol engineering", have to deal with the distributed nature of the protocols, and must include methods for (a) specifying the protocols, (b) verifying that a protocol works correctly and provides some desired communication service, (c) implementing the protocol in software or hardware, and (d) for testing that a given implementation conforms to the requirements of the protocol definition.

The need for such protocol engineering methods was felt during the early times of protocol development in the 1970ies, and much work was done during the 1980ies in the context of OSI standardization. The purpose of this paper is to describe the early developments of protocol engineering in the 1970ies and '80ies from the perspective of my own involvement in this field[1].

The paper is structured as follows: In Section 2, we give an overview of the development of computer communications protocols in the 1970ies and '80ies. In Section 3, we first provide some comments on our own research path during the 1970ies, and then explain the basic concepts of protocol engineering that were explored and defined by the research community during those years. In Section 4, we give a short overview of the development of Formal Description Techniques (FDTs) for communication protocols and services, which took place during the 1980ies in the context of the standardization of protocols for Open Systems Interworking (OSI). In the conclusions, we discuss the

[1] This paper is based on a draft from 2009, which was also used for [Boch 11]. Compared with the latter, this paper does not include the contribution by Colin West about his work on SNA and OSI standardization, nor the section on standardization of conformance testing contributed by Dave Rayner.

impact of the early work on protocol engineering on the current methods and tools that are used for developing distributed applications for data processing and communications.

2. Computer communications in the 1970-80ies

2.1. The first computer network protocols

The 1970ies and '80ies were stimulating times for people working in the area of computer communications. In the late '60ies, the first link-level protocols were introduced for remote access, over leased lines, from terminals to host computers. In 1968, Lynch described a protocol with positive and negative acknowledgments which was not fully reliable, as pointed out by Bartlett et al. in 1969 [Bart 69]. During the same time, IBM had introduced the Bisync protocol which was used for long time (a superficial look at its definition seems to indicate that it had similar flaws as the protocol of Lynch).

Then in the early '70ies, a new generation of link protocols was introduced with bit-oriented framing and sequence numbering (initially represented by 3 bits with values from 0 to 7). IBM's SDLC strongly influenced the international standard HDLC developed within the International Standardization Organization (ISO). This protocol was used in 1976 by the International Telecommunications Union (ITU) for their standard network access protocol, called X.25. Note: IBM, at that time, had a dominating market position as computer and software manufacturer, similar to Microsoft in the software domain nowadays.

In the late 1960ies, several research projects had been established to experiment with the concept of packet switching. The first two operational packet switched networks were the ARPANET in the USA and the NPL network at the National Physical Laboratory in the UK. Larry Roberts led the ARPANET team that had the first nodes of a wide area network operating by December 1969. Donald Davies led the NPL team that had the world's first local area network operational by 1970. Donald Davies coined the term "packet switching" in 1966 and co-invented the concept along with Paul Baran of the original ARPANET team. ARPANET is seen as being the forerunner of today's Internet. The French Cyclade network project, led by Louis Pouzin, introduced the concept of an unreliable datagram service at the Network layer over which an end-to-end Transport layer protocol would build a reliable connection-oriented service, sometimes called "virtual circuits" [Pouzin 1973]. These concepts were later adopted for the IP/TCP protocol hierarchy of the Internet as we know it today.

During this time, it also became apparent that one needs some standards for interconnecting different packet-switched networks. Protocols that could be used for this purpose were called Internet protocols. An international group of experts was formed in 1972 under the name INWG which became affiliated with the Technical Committee for Communications Systems (TC-6) of the International Federation of Information Processing (IFIP), as Subgroup 6.1. This group met quite frequently and elaborated a set of Internet protocols which included precursors of IP and TCP, a virtual terminal protocol and others. I participated in some of these meetings and had the impression that Vint Cerf from ARPAnet and Louis Pouzin from Cyclade were the principal leaders of the group. Towards the end of the 1970ies, these IP/TCP protocols were implemented for the interconnection of Ethernets and satellite networks on an international basis, and the original protocols of the ARPAnet disappeared[2].

2.2. Commercial computer networks

At the same time, the common carrier companies and governmental PTT ("post-telephone-telegraph") organizations had realized that standards for commercial packet-switched networks were required and started to discuss possible approaches within the ITU. Following their tradition of telephone network technology, they opted for a network service of reliable virtual circuits. This meant that at the interface between a host computer and the network, there was the requirement of supporting a large number of such circuits with separate flow control. Therefore the interface standard X.25 that was adopted in 1976 was much more complex than what would be required for providing a datagram service.

During the mid-70ies, there was much polemics concerning the question whether a datagram or a virtual circuit network service would be more appropriate for packet-switched networks. The INWG group had submitted their proposal for datagram and end-to-end Transport protocols to the ITU and tried to convince the PTT experts that their approach was preferable over the virtual circuit approach. However, the ITU stuck to the virtual circuit approach and

[2] An account of these developments and the emergence of the Internet as the general networking infrastructure in the 1990ies is given in [Hauben].

produced in X.25 a standard network access protocol supporting several multiplexed virtual circuits. Later, a similar standard, called X.75, was defined for the interconnection of different packet-switched networks.

Although the datagram approach had not be adopted by ITU for the network access protocol standard, discussion continued whether this technology would be advantageous to be deployed within a network, internally. The Datapac network developed in Canada by Bell-Northern-Research did use internally datagrams with alternative routing. On the other extreme was the French Transpac network which used virtual circuits internally - where each circuit, when established, was allocated to a fixed route. Arguments were related to resilience against different types of hardware faults and resource requirements in terms of memory requirements in the switches (in case of Transpac) and transmission overhead due to lengthy packet headers (in case of datagrams, plus the Transport protocol required for reliability).

In the meantime, different computer manufacturers had also started developing proprietary protocols for computer communications. The previous decade had seen batch-oriented computer systems evolve towards interactive systems. A systematic approach to providing shared communications facilities that permitted large numbers of terminals to access a variety of applications running on large host machines was needed. The Systems Network Architecture (SNA) defined by IBM in 1974 was much deployed to enable users to access data processing services and devices over distance. SNA was defined as a layered architecture including data link control, routing, flow control, management of shared resources and presentation services.

2.4. Application layer protocols

Given that proprietary computer networking protocols had been developed by different manufacturers – besides SNA, similar protocols had been developed by Digital Equipment Corporation, Honeywell and others – interworking between systems provided by different manufacturers was difficult. In order to simplify the interworking between such systems, the standardization project on Open Systems Interworking (OSI) was initiated within ISO in 1977. It was mainly aimed at standards for distributed applications that would use the X.25 standard as the basic communication service over which the application protocols would run (see [Green 80] for the situation around 1980).

The work on OSI extended over more than 10 years until the early 1990ies. As usual, not all parties participating in the standardization meetings were really interested in successful standards, and there were a variety of different interest, which led to relatively complex standard proposals that had options for all kinds of situations. For instance, much discussion took place about the Transport protocol, which is not really required over X.25, but which is required if the datagram service is used in the network layer. Because of the input from the Internet community, a Transport protocol Class 4 was introduced, quite similar to TCP. In fact, there were altogether 4 classes of Transport protocols.

We note that the ARPAnet (later called Internet) was continuously in service for the university and research sector within the USA and also in other countries since the mid 70ies. Later these protocols were supported by the Berkeley Unix software which was available free of charge within the research community and became widely used in this context. This continuous use of the Internet is the main reason why it finally became the de-facto standard for the current global Internet – much helped by the Web application (originally developed at CERN, Geneva) in the early 90ies.

To finish this section, let me mention just two other interesting protocol developments in the 1980ies: In 1980, the ITU had defined the Teletext standard. It may be called the Web standard of the ‘80ies. Like the current Web service over the Internet, Teletext was based on the Hypertext concept with its pages, links and anchors. The service was implemented, for instance, in France and Germany, and the Canadian version included graphic images. The main problem was the lack of suitable terminals to display the pages (since desktop computers were not very common at that time). Special Videotext terminals were manufactured and could be rented. Together with my colleague Jan Gecsei and other people, I was involved in several research projects related to Videotext (see e.g. [Tompa 81]).

Within the OSI framework, many application-layer protocols were developed. For the description of these protocols, aspects related to the data structures of the parameters of service primitives and protocol messages are usually much more important than the temporal ordering of interactions specified by state machine models. To describe these data structures, the ASN.1 notation was developed and was used for most OSI application protocols. (Note: this was probably influenced by an earlier protocol for Remote Procedure Calls (RPC) developed at the Xerox Parc research center). ASN.1 also includes standards for encoding the structured information. Several encoding standards were developed, based on octet-oriented encoding of tags and lengths indicators. In the mid-‘80ies, several tools were developed for automatically generating encoding and decoding routines for protocol messages defined using this description method. Nowadays, XML is often used for the same purpose, and associated encoding and decoding tools are available from various sources.

3. Development of protocol engineering methods in the 1970ies

3.1. A personal view of early developments

In the early 1970ies, I had been working on compiler design, semantic attributes and methods for defining the semantics of programming languages. However, I was looking for a younger field of investigation with more open problems. In 1973 or '74, I attended a conference on computer communications where I met Louis Pouzin who was talking about the need for standard communication protocols to allow the interworking between different computer networks developed in different parts of the world. Pouzin told me that good methods for precisely defining the communication protocols where needed and also methods for verifying that they provide the communication service that is expected. The question how these protocols and services could be formally defined was not known. His comments motivated me to work in this area, and I experimented with different specification paradigms for specifying protocols.

On the one hand, I explored the use of finite state machines, inspired by the paper by Bartlett et al. [Bart 69] on the alternating bit protocol which used a state machine formalism[3]. In this context, I developed the method of protocol verification by reachability analysis which I also applied to the X.25 protocol standard [Boch 78i]. On the other hand, I also tried a specification style using a programming language and assertions such that properties of specifications could be proven using program proving techniques for concurrent systems [Boch 75c]. A year later, a journal publication by Stenning appeared presenting a very similar protocol verification. It was clear that certain protocol features, such as sequence numbering, cannot be adequately described by state machines, although the state machine approach appears to be natural for many aspects of protocols. Therefore I proposed in 1977 an approach combining both methods in the form of an extended state machine, adding state variables, interaction parameters and assignment statements [Boch 77c], very similar to today's UML state diagrams.

When I presented this paper at the IFIP Congress 1977 in Toronto, I met Pitro Zafiropulo who showed me some work he had done independently with Harry Rudin and Colin West at the IBM Research Lab in Zurich. They had developed a protocol verification technique very similar to mine [Rudin 78, Zafi 80]. Colin West had also built some automated tool for performing verifications which he applied to several protocols, including the ITU standard X.21 [West 78b]. (More details about the work of this group in the context of SNA and OSI standardization can be found in [Boch 11]).

The Computer Network Protocols Symposium [Danthine 1978] organized in February 1978 by André Danthine in Liège was the first international conference dedicated to the design of communication protocols. At this conference, for the first time, I met many people interested in similar questions, and I have kept contact with many of them over long time. There were many papers on methods for defining and verifying protocols, and also many papers related to particular protocols, in particular to the newly adopted ITU standard X.25 for computer communications. Several of the papers presented at this conference were later published in enhanced versions in Computer Network. This is also where I met Carl Sunshine who had already written some paper surveying different methods for protocol specification and verification, and with whom I wrote the survey paper that appeared in the IEEE Transactions [Boch 80a].

3.2. Some important concepts

Many of the important concepts for protocol engineering were elaborated during the 1970ies. In the following, I will high-light some of these developments. More references for these topics can be found in [Boch 11].

3.2.1. Protocol layering – communication service

The concept of protocol layering became well-known through the OSI standardization effort which defined seven protocol layers. These ideas were first developed within the French Cyclade network in the early 1970ies, distinguishing

[3] After my discussion with Louis Pouzin, I found the paper by Bartlett et al. on the alternating bit protocol and I tried to develop a formal model of this protocol and to prove that it provides a reliable communication service. The paper by Bartlett included already a description of the protocol using a state machine model. The result of my work was written down in a working document in 1974 of which only a summary was included in my 1975 conference presentation [Boch 75d] because of lack of space. Later I applied the same method to the X.25 protocol; these results were included, together with the details of the 1974 document, in my presentation at the Liège conference [Dant 78].

between an unreliable datagram network service and a reliable end-to-end transport service with corresponding protocols (see for instance [Pouzin 1973]). A protocol layer is characterized by its protocol (defined by the behavior of the protocol entities) and the communication service it provides. The importance of both, protocol and service descriptions, was clearly explained in our survey with Sunshine in 1980 [Boch 80a]. The architectural diagrams are shown in Figure 1 (a) and (b). The first formal model of a communication service was perhaps the (very simple) service of the alternating bit protocol shown in Figure 1(c) (taken from [Boch 78i] – by the New primitive one user submits data, which is received by the other user through the Use primitive). The formalized description of more complex communication services was discussed in [Boch 80e], where I made the distinction between local properties (related to the interactions at a single service access point – and therefore locally observed by each protocol entity) and global properties (relating interactions at different service access points – the properties of the communication service "proper"). This concept was generalized in my paper with Michel Raynal in 1983 [Boch 83a] to arbitrary multi-component architectures.

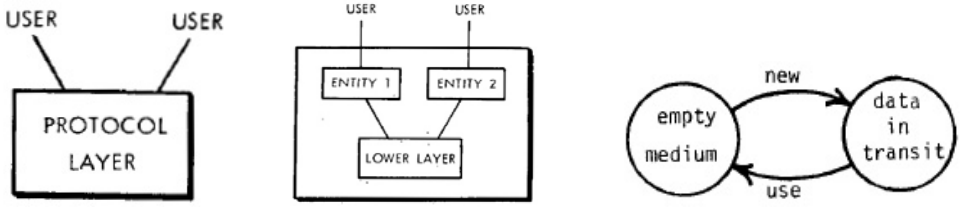

Figure 1: (a) service of a protocol layer, (b) protocol architecture, (c) behavioral specification of a simple service

3.2.2. Interworking between heterogeneous networks

With the development of experimental packet-switched networks in several countries during the first half of the 1970ies, and the development of commercial packet-switched networks in the second half, the interconnection of different networks and their interworking became a major issue (see for instance [Green 86]). The first paper I remember on the principles of interworking between heterogeneous networks was written by Gien and Zimmermann in 1979 [Gien 79]. They pointed out that the interconnection must be realized at the service level; it requires compatible services within the two interconnected networks. Much of the later work did not take this into account. My papers from 1990 [Boch 90b] tried to put these principles into a formal setting.

3.3.3. The role of protocol specifications

Towards the end of the 1970ies, it became clear that a precise protocol specification is important because it should be (a) the starting point for the verification of the correctness of the protocol design (against the service specification), (b) the basis for the development of implementations on different computers that would be compatible with one another, and (c) the reference for developing test cases by which a given implementation could be checked for conformance with the protocol specification. This fact is reflected in the title of the IFIP conference series on Protocol Specification, Testing and Verification (PSTV) which started in 1981, and also led to the formation of a working group on Formal Description Techniques (FDT) for OSI Protocols and Services within the OSI standardization effort of ISO (see below). At the 1978 workshop in Liège, John Day talked about the need for precise protocol specifications in the context of the OSI standardization, which had just started the year before.

One important aspect of protocol specifications is their relative abstractness. In particular, the interfaces of a protocol entity with the layer below and with the user in the layer above must be specified in an abstract manner in order to allow different realizations of these interfaces for different implementations. This led to the concept of "service primitives" which was used to define the OSI layer services. Abstractness is also required for the description of the dynamic behavior of protocol entities; for instance, often different acknowledgement schemes are allowed for a protocol and could be implemented in different manners.

The specification of a protocol consists of defining the dynamic behavior of the two protocol entities (see Figure 1(b)). As an example, Figure 2(b) shows the state machine models of the two entities of the alternating bit protocol, the sender on the left, and the receiver on the right. Figure 2(a) shows the system architecture including the service primitives New and Use and the interactions with the underlying message transmission layer ("s" means send, "r" means receive, "D" means data packet, "A" means acknowledgement packet, "E" means a received message with error, and the value of x represents the alternative bit – with value 0 or 1).

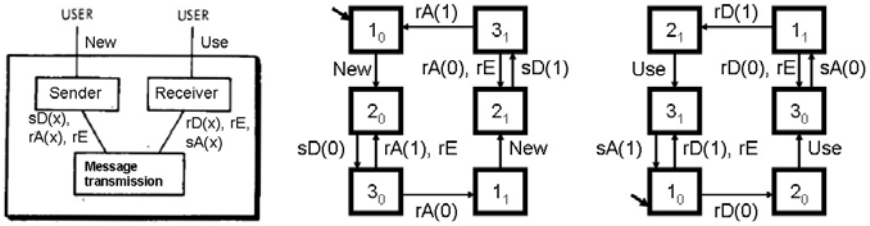

Figure 2 : (a) architectural view, (b) protocol specification (dynamic behavior of the two protocol entities)

3.2.3. Protocol verification and reachability analysis

It soon was apparent that one has to be very careful when designing a new communication protocol. Already the paper describing the alternating bit protocol in 1969 [Bart 69] pointed out a bug in the design of an earlier bit-based protocol. Sunshine's surveys on protocol description and verification methods (and his collection of articles [Sunshine 1981]) show the many different approaches that were used to specify the properties of the dynamic behavior of protocol entities and related methods to show that their communications would lead to an overall desired behavior.

This work led to the distinction of general properties that each protocol should satisfy and specific properties that are required by the communication service that is intended to be provided by the protocol [Boch 80a]. The general properties state the absence of certain flaws, including well-known concepts such as deadlocks, but also the concept of "unspecified reception", first described by the IBM Zürich team. The first proofs that some (very simple) specific properties of the communication service are satisfied were included in my papers of 1975.

The term "reachability analysis" has been used to describe a verification approach where one considers a global system consisting of the protocol entities and the underlying communication medium and where the protocol specification is used to determine the possible state transitions that may occur in the global system. Reachability analysis then consists of exploring all possible global states that can be reached from the initial state of the system. As an example, Figure 3 shows the global reachability graph of the alternating bit protocol, assuming that no message is completely lost. Each state of the graph is characterized by the state of the two protocol entities (e.g. " 2_0 1_0 " means that the sender is in state 2_0 and the receiver in state 1_0 ; any message in transit is also indicated.

By analyzing this global reachability graph, one can determine whether the general properties and certain service requirements are satisfied. When I worked on these problems, I noted that such ideas had been proposed in a more general context. In the context of communication protocols, the state transitions are usually associated with the sending or receiving of protocol messages or with the loss or error in the transmission medium. This led to certain models for the communication medium that were incorporated in certain verification tools, and specific types of errors, such as "unspecified reception". For instance, the graph of Figure 3 shows that the service primitives New and Use are executed alternatively, as required by the service definition in Figure 1(c) - unless all messages are received in error.

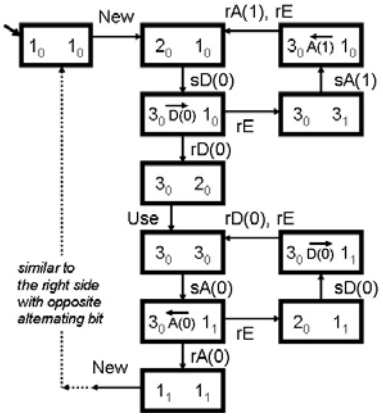

Figure 3 : Reachability graph of the alternating bit protocol

While the reachability analysis for certain simple protocols have been done "by hand", any real-life application of this method requires automated tools because of the large number of reachable global states. As mentioned above, the first tools for this purpose were developed within IBM Zurich. Later many other tools have been developed, not only for

finite state machine models, but also for many other specification formalisms. Holzmann's SPIN tool [Holz 91], initially developed in the early 1980ies and continuously improved, introduced the so-called bit-state representation where, through hashing, a global state is represented by a single bit; this allows the analysis of very large state spaces. Many of these tools have been extended to the realm of model checking, where in addition to the detection of deadlocks and unspecified receptions in the model, the designer can specify specific properties in temporal logic that will then be checked by the tool.

3.2.4. Constructive methods for protocol design

While protocol verification is intended for checking whether a given specification of a protocol has all the desired general properties and satisfies the requirements of the desired communication service, some researchers explored constructive methods for protocol design. The objective was to find methods for constructing automatically, or semi-automatically, a protocol specification that has the given desired properties. The obtained protocol designs would then be "correct by construction". We can mention a semi-automatic approach described in [Zafi 80], and the "submodule construction" approach proposed [Boch 80d], however, both methods require the user to input major parts of the protocol specification. The latter approach was later found to be useful for finding the behavior of a protocol converter, for finding a controller in discrete-event control applications, for embedded testing, and possibly for component re-use.

I think that the following two methods proposed in the 1980ies are more interesting for protocol construction. At the first PSTV workshop in 1981, Gouda and Yu presented an interesting method for designing communication protocols without deadlocks and with bounded communication channels (see journal version [Gouda 84]). The method starts with the specification of one protocol entity, and ignores the communication service. Its main contribution is to show how conflicting initiatives from the different protocol entities can be resolved.

Another protocol derivation approach was proposed in my paper with Reinhard Gotzhein in 1986 [Boch 86g]. Here it is assumed that the specification of the communication service is given, including two or more service access points. The method allows the derivation of the behavior of the protocol entities for each of the service access points, including the exchange of protocol messages between the different entities for coordinating the temporal order of service interactions at the different service access points. A reliable underlying message transmission service is assumed. Over the years, this method has been adapted to various specification paradigms, and it can be used for deriving component behaviors for distributed applications concerned with business processes, workflow and other systems where the dynamic behavior may be described by collaborations performed in some specific temporal order [Boch 08b].

3.3.6. Tools for protocol implementation

A verified protocol specification is clearly the starting point for protocol implementation. While finite state machine models are easy to implement in hardware and software, extended state models and the Formal Description Techniques (FDT) developed in the 1980ies require more complex implementation structures, especially related to the concurrency of the model specifications. Automated tools (compilers) for generating implementation code have been developed for these specification languages since the early 1980ies. These developments later led to commercial modeling tools, for instance for the SDL language, which now has been integrated into the UML modeling framework.

3.3.7. Specification-based testing

In the context of protocol standardization, conformance testing is an important issue. The question is how to test a protocol implementation to determine whether it conforms to the protocol standard. It is clear, therefore, that the test cases to be applied to the protocol implementation should be based on the protocol specification that defines the standard. Since the early 1980ies, much research was aimed at test suite development, that is, developing methods for finding a set of test cases, based on a given protocol specification, that would find most faults that may exist in any implementation. The first publication on this topic was probably Sarikaya's paper at the 1982 PSTV conference, in which he adapts the so-called W-method, originally developed for testing software components, to protocol testing based on FSM models. In the subsequent journal paper [Sari 84], he also discussed the problems related to the synchronization between different interfaces, a problem specific to protocol testing.

Since then a large body or research results have been established by researchers around the world[4]. On the one hand, tests for specific faults were considered, such as output and transition faults for state machines; on the other hand, test

[4] An invited paper at the 1994 International Symposium on Software Testing and Analysis, gave a review of these methods and discussed their relevance to software testing in general [Boch 94].

development methods were adapted to extended state machine models and other specification languages, such as Input/Output Automata and process algebras. While most methods provide a fixed set of test cases, other methods consider adaptive testing where the next test input depends on previous interactions.

3.4. Conferences on protocol specification, verification and testing

The first international conference concentrating on protocol specification, verification and testing was organized by Dave Rayner 1981 in Teddington (UK) under the title "Protocol Testing – Towards Proof ?". Sponsored by Working Group 6 of IFIP, this conference started the series of conferences under the name "Protocol Specification, Testing and Verification" (PSTV). This became the principal conference for the discussion of formal description techniques, and methods and tools for protocol verification, implementation and testing. In 1988, when the ISO FDTs had been standardized, Ken Turner started a second, parallel conference series called Formal Description Techniques (FORTE) which concentrated more on the FDTs, their tools and practical applications. Later in 1996, when the general interest in these FDTs had decreased, these two conference series were combined into a single conference under the name "Formal Techniques for Networked and Distributed Systems" (FORTE).

In 1988, also a conference series on protocol conformance testing was started by Son Vuong in Vancouver (Canada) under sponsorship from IFIP. Initially called International Workshop on Protocol Test Systems (IWPTS), it was later called International Conference on Testing Communicating Systems (TestCom) and became more recently a joint conference with the International Workshop on Formal Approaches to Testing of Software (FATES). Since that time, most of the research results on protocol testing have been presented at this specialized conference.

The bibliographical information and the tables of content for most of these conferences can be found on the Internet [conferences].

4. Standardization of formal description techniques for communication protocols and services

As noted above, the people involved in the standardization of OSI protocols and services were looking for description techniques that could be used to defined protocols precisely and that would be associated with tools that would be helpful for verification, implementation and conformance testing. In 1980 an "FDT Group" was formed within ISO under the leadership of John Day. Initially, the group concentrated on writing guidelines about how to informally describe communication protocols and services, which were used by the various OSI subgroups for writing protocol and service descriptions. Then the group attacked the problem of defining formal techniques for which tool support would be possible to develop. The experts participating in the group had varying interests and therefore various approaches to protocol specification were proposed. This led finally to the formation of three subgroups A, B and C with subgroups B and C developing different techniques. Subgroup B developed an extended FSM formalism, later called Estelle, in which I participated, and subgroup C developed a formalism based on rendezvous communication, later called LOTOS. The early stage of development of these languages is described in a paper by Vissers et al. [Vissers 83]. During the development of these FDTs, the OSI Transport protocol Class 2 was often used as an example to show the effectiveness of the FDT. Later, in order to provide some comparison between the different FDTs, I published example specifications of this protocol in the three FDTs (Estelle, LOTOS and SDL) [Boch 90a].

At this time, the ITU had already defined the System Description Language (SDL) which had been developed over several 4-year study periods. This was in fact an FDT based on the extended FSM model, and therefore the question came up whether Estelle and SDL could be developed jointly into a single language. This gave rise to the first joint working meeting between ISO and ITU experts, which was held 1984 in Ottawa, and which I had the honor to chair. (Note that up to that time, the development of OSI protocols had been done by ISO and ITU in separate meetings.) It was decided that the FDT experts from OSI and ITU should get together and draft a proposal for a joint FDT as a candidate to be discussed afterwards by the respective parent organizations. However, the resulting document, called X.250 within ITU, was not accepted, and the development of these two languages continued independently.

During that time, the OSI experts were waiting for an FDT that they could use for describing the new OSI protocols. However, the development of the FDT's was not complete, and the question of which FDT should be used was very sensitive. The subgroup working on guidelines for conformance testing, led by Dave Rayner, expressed a strong need for an FDT, since they were looking for a language to define conformance test cases. Since the testing experts did not want to select one of the proposed FDTs, they finally decided to develop yet another language, called TTCN (acronym of

Tree-Table-Combined-Notation, not related to any FDT), which has since then been used in the area of conformance testing.

Estelle and LOTOS became ISO standards around 1986, but were never much used. SDL was further developed within ITU and was used for the description of ITU protocol standards. An overview of the state of the art of protocol specifications using informal and formal techniques at that time is given in [Boch 90g]. Looking back to this time of FDT development, one has to conclude that these description methods and the related tools were not much used for the purpose they were developed (except for SDL). One can give different reasons for explaining this situation:

1. The community was not ready to use formal modeling approaches to define requirements for protocol implementations.
2. The implicit competition between the three different languages made it difficult for practitioners to make the choice.
3. The OSI protocols, for which the FDTs were originally intended, did not have much success themselves.

A further reason was that the specification techniques in use for OSI protocols were already quite good and extremely useful. For example, the OSI Session Layer was specified in terms of tabular finite state machines. In 1985 Colin West found it quite easy to transform these into an executable model that he validated using the reachability analysis techniques that he had earlier used for SNA [West 86]. A substantial number of errors were found.

6. Conclusions

This historical retrospective indicates that the basic concepts related to the specification of communication protocols and services, the verification of protocols and their implementation and testing originally developed in the 1970ies and ‘80ies are still valid today. Many things have remained the same, although over the years, the more detailed aspects have evolved and given rise to new technological developments. The IFIP conferences in the field of protocol engineering, namely PSTV, FORTE and TestCom, have played an important role in maintaining the tradition in this field and providing a forum for the discussions of new developments in these areas.

During the first half of the 1980ies, there was some form of hype on Formal Description Techniques (FDT). Industry had high hopes that formal specifications of protocol standards, especially in the context of OSI, would lead to less ambiguous definitions of protocols, and would lead, through the use of automated tools, to simpler methods for verification, implementation and conformance testing. However, these hopes did not materialize, and in addition, the push for OSI protocols slowed down during the second half of the decade. This was a big disappointment for the FDT community.

We note, however, that the results of the work on protocol engineering, formal description techniques and the related tools for protocol verification, implementation and testing had an important impact by being carried over to the current state of the art, and having been used for many practical applications in different fields, as explained in the following paragraphs.

Layered protocol architecture: The conceptual layered architecture for communication protocols, as developed during the 1970ies, and standardized as the OSI Reference Model in the early ‘80ies, are nowadays part of each good text book on computer communications.

Model checking: In a sense, the reachability analysis tool of Colin West was the first model checker – able to check a model of several protocol entities for general types of flaws, such as deadlocks and unspecified receptions. The invention of “model checking” by Clark, Emmerson and Sifakis in 1981 provided, in addition, for the checking of specific properties specified in temporal logic. During my visit at Havard University in 1982, Edmund Clark was excited about the fact that his model checking algorithm could not only check properties of a given state machine model, but also properties of protocols and provided communication services – by applying his algorithm to the global state model obtained by reachability analysis. The SPIN tool developed by Gerald Holzmann since the early 1980ies is an example of a successful model checking tool that started out as a tool for reachability analysis and, over the years, incorporated many advances in the field, including checking temporal logic properties [Holz 91]. This tool has been used for many practical applications of hardware and software verification. The community of people using the tool has been growing over the years and meets at yearly SPIN workshops.

UML tools: As mentioned above, the ITU had developed over the years (since the late ‘70ies) a standard FDT, called System Description Language (SDL). It was used for describing many communication protocol standards and other industrial systems, and its commercial tools, developed in the 1990ies, have been used for the development of commercial protocol implementations, for instance in the wireless telephony sector. The most successful tool was produced by the Swedish company Telelogic (recently bought by IBM); it includes code generation for model simulation

and implementation, as well as advanced reachability analysis functions. Recently, SDL has been integrated into UML-2 as a profile, and the tools are adapted to this new context.

Model-driven development: This term has recently become a fashionable concept. Protocol engineering has used this approach since its beginning. The reason for this is the fact that a protocol specification should be relatively abstract (in order to allow for different implementations and APIs) and precise (in order to guarantee compatibility of different implementations). The protocol specification is therefore an abstract model of the final implementation, and protocol verification is done at the model level. In fact, the FDTs SDL and Estelle, as well as Harel's State Charts of 1987 are languages based on the principle of extended finite state machines from the 1970ies, and they can be considered to be ancestors of the State Diagram notation now part of UML.

References

[Bart 69] K. A. Bartlett, R. A. Scantlebury and P. T. Wilkinson, A note on reliable full-duplex transmission over half-duplex links, ACM Communications, Vol.12, No.5, May 1969, pp.260-265.

[Boch 08b] G. v. Bochmann, Deriving component designs from global requirements, Proc. Intern. Workshop on Model Based Architecting and Construction of Embedded Systems (ACES), Toulouse, Sept. 2008.

[Boch 11] G. v. Bochmann, D. Rayner and C. H. West, Some notes on the history of protocol engineering, Computer Networks Journal, 54 (2010), pp. 3197–3209.

[Boch 75c] G. v. Bochmann, Logical verification and implementation of protocols, Proc. 4th Data Communication Symposium (ACM-IEEE), pp. 7-15 to 7-20, Oct. 1975, reprinted in "Communication Protocol Modeling", edited by C. Sunshine, Artech House Publ., 1981.

[Boch 75d] G. v. Bochmann, Communication protocols and error recovery procedures, Proceedings ACM Symposium on Interprocess Communication, SIGOPS Review, 9, No.3, 45-50 (1975).

[Boch 77c] G. v. Bochmann and J. Gecsei, A unified method for the specification and verification of protocols, Proc. IFIP Congress 1977, pp. 229-234.

[Boch 78i] G. v. Bochmann, Finite State Description of Communication Protocols, Computer Networks, Vol. 2 (1978), pp. 361-372.

[Boch 80a] G. v. Bochmann and C. A. Sunshine, Formal methods in communication protocol design, (invited paper) IEEE Tr. COM-28, No. 4 (April 1980), pp. 624-631, reprinted in "Communication Protocol Modeling", edited by C. Sunshine, Artech House Publ., 1981.

[Boch 80d] G. v. Bochmann and P. M. Merlin, On the construction of communication protocols, ICCC, 1980, pp.371-378, reprinted in "Communication Protocol Modeling", edited by C. Sunshine, Artech House Publ., 1981.

[Boch 80e] G. v. Bochmann, A General Transition Model for Protocols and Communication Services, IEEE Trans. Comm., COM-28, 4 (April 1980), pp. 643-650, reprinted in "Communication Protocol Modeling", edited by C. Sunshine, Artech House Publ., 1981.

[Boch 83a] G. v. Bochmann and M. Raynal, Structured specification of communicating systems, IEEE Trans. Computers C-32, 2(Febr. 1983), pp. 120-133.

[Boch 86g] G. v. Bochmann and R. Gotzhein, Deriving protocol specifications from service specifications, Proc. ACM SIGCOMM Symposium, 1986, pp. 148-156.

[Boch 90a] G. v. Bochmann, Specifications of a simplified Transport protocol using different formal description techniques, Computer Networks and ISDN Systems, Vol. 18, no.5, June 1990, pp. 335-377.

[Boch 90b] G. v. Bochmann and P. Mondain-Monval, Design principles for communication gateways, IEEE Tr. on Selected Areas in Communications, Vol. 8, 1 (Jan. 1990), pp. 12-21; russian translation: Express Information (overview of western publications), Information Transfer, 1991.

[Boch 90g] Bochmann, G. v., Protocol specification for OSI, Computer Networks and ISDN Systems 18 (April 1990), pp.167-184.

[Boch 94d] G. v. Bochmann and A. Petrenko, Protocol testing: Review of methods and relevance for software testing (invited paper), ACM International Symposium on Software Testing and Analysis (ISSTA'94), Seattle, USA, 1994, pp 109-124.

[Conferences] Computer Science Conferences and Workshops, see http://www.informatik.uni-trier.de/~ley/db/conf/index-f.html

[Danthine 1978] Computer Network Protocols Symposium, 13-15 Febr. 1978, organized by André Danthine, Université de Liège, Belgium. See also Computer Networks, Vol.2, Nr. 4/5 (1978).

[Gien 79] M. Gien and H. Zimmermann, Design principles for network interconnetion, in Proc. VI Data Communication Symp. (IEEE/ACM), Nov. 1979, pp. 109-119.

[Goud 84] M. G. Gouda and Y.-T. Yu, Synthesis of communicating Finite State Machines with guaranteed progress, IEEE Trans. on Comm., vol. Com-32, No. 7, July 1984, pp. 779-788.

[Green 80] P. E. Green, ed., IEEE Transactions Special Issue on "Communications on Computer Network Architectures and Protocols" (April 1980).

[Green 86] P. E. Green, Protocol conversion, IEEE Trans. Commun. P. E. Green, Vol. COM-34. pp. 257-268. Mar. 1986.

[Hauben] Ronda Hauben, Communicating across the boundaries of dissimilar networks, http://www.columbia.edu/~rh120/other/misc/finland416.txt

[Holz 91] G. J. Holzmann, Design and Validation of Computer Protocols, Prentice Hall, 1991.

[Pouzin 73] L. Pouzin, Presentation and Major Design Aspects of the Cyclades Computer Network, Proc. Third Data Communications Symposium, St. Petersburg, Florida.

[Rudin 78] H. Rudin, C.H. West, P. Zafiropulo, Automated protocol validation: One chain of development, Computer Networks, Vol. 2, No 4/5, pp. 373-380, 1978.

[Sari 84a] B. Sarikaya and G. v. Bochmann, Synchronization and specification issues in protocol testing, IEEE Trans. on Comm., COM-32, No.4 (April 1984), pp. 389-395.

[Sunshine 81] C. Sunshine (ed.), Communication Protocol Modeling, Artech House, 1981.

[Tomp 81b] F. W. Tompa, J. Gecsei and G. v. Bochmann, Data structuring facilities for interactive videotex systems, IEEE Computer, Vol.14, No.8, August 1981, pp.72-81.

[West 78c] C. West, General technique for communications protocol validation, IBM Journal for Reseach and Development, Vol. 22, No. 4, 1978, pp. 393-404.

[West 86] C. H. West, A validation of the OSI session layer protocol, Computer Networks ISDN Systems 11 (1986) 173-182.

[Viss 83a] C. A. Vissers, G. v. Bochmann and R. L. Tenney, Formal description techniques, Proceedings of the IEEE, vol. 71, 12, pp. 1356-1364, Dec. 1983; translated into russian.

[Zafi 80] P. Zafiropulo, C. H. West, H. Rudin and D. D. Cowan, Towards analyzing and synthesizing protocols, IEEE Tr. Comm. COM-28, 4 (April 1980), pp. 651-660.