Testing Quality-of-Service Aspects in Multimedia Applications

Jens Grabowski[a] and Thomas Walter[b]

[a]University of Berne, Computer Science Department, Neubrückstrasse 10, 3012 Bern, Switzerland, e-mail: grabowsk@iam.unibe.ch

[b]Swiss Federal Institute of Technology Zürich, Computer Engineering and Networks Laboratory, ETH Zürich/ETZ Building, 8092 Zürich, Switzerland, e-mail: walter@tik.ee.ethz.ch

Abstract

Assuring and proofing the quality of multimedia applications and services by means of testing will be a great challenge for manufacturers and service providers. Standardized methods and tools for conformance testing are applicable to traditional protocols (i.e., a single data stream, no timing requirements) only. In this paper we discuss a testing methodology and framework for testing multimedia applications. We started to develop and implement a new *TeleCom*munication *T*est *S*pecification and implementation *L*anguage (TelCom TSL). TelCom TSL is meant to be a tool for specifying and implementing test cases for (distributed) multimedia applications. TelCom TSL defines a novel testing architecture. Its formal syntax and semantics definition with real-time extensions makes TelCom TSL applicable for testing multimedia applications. The contributions of this paper are an analysis of different QoS semantics in the context of multimedia applications, a definition of QoS testing and the TelCom TSL testing architecture.

Keywords: Distributed multimedia applications, quality of service, conformance testing, testing architectures, specification languages

1 Introduction

Due to the widespread dissemination of new hard- and software technologies multimedia begins to play an important factor on the commercial market. In the near future different manufacturers and service providers will compete with comparable multimedia products. Besides the price the quality of a multimedia product will be a major argument for convincing customers to purchase a specific product. This implies that manufacturers and suppliers of multimedia products are faced with the problem of proofing the correctness and assuring the quality of their products.

Our aim is to investigate, elaborate and provide methods and tools for proofing, measuring and assessing the quality of multimedia products and services. We restrict ourselves to distributed multimedia applications, e.g., applications and services for tele-teaching, multimedia archiving and retrieval, cooperate working in different locations, or video-on-demand services. Furthermore, we focus on testing because for *traditional* protocols and protocol implementations it is common practice to proof and assure quality by means of *conformance testing* [13]. We assume that conformance testing will be similarly important for distributed multimedia applications and services.

The term *traditional* denotes protocols and protocol implementations which handle one data stream and, only to a limited extend, impose timing constraints. *Conformance testing* as understood by ISO and ITU-T [13] is *functional black-box testing* of OSI protocol implementations. An *implementation under test* (IUT) is meant to be a black box and its observable behavior is compared with the observable behavior as derived from a protocol specification.

Compared with traditional protocols and protocol implementations the term *distributed multimedia application* refers to applications utilizing more than one data stream, e.g., video and audio at the same time, with functional properties (as tackled by conformance testing) for each data stream, and, furthermore, *non-functional* properties, e.g., timing constraints (as in real-time applications), quality-of-service (QoS) aspects, synchronization of different data streams (audio, video, textual information) [21]. It is known that methods standardized for conformance testing are not able to deal with the upcoming new requirements of distributed multimedia applications.

Methods and tools for conformance testing are defined in the international ISO/IEC multipart standard IS 9646 *'OSI Conformance Testing Methodology and Framework'*. Part III of ISO/IEC 9646 [15, 16] defines the *Tree and Tabular Combined Notation* (TTCN) as the main tool for specifying test cases for conformance testing. TTCN is a notation and not a language since it has a standardized syntax but no formal semantics definition.

Strengths and lacks of TTCN are well known and have been discussed thoroughly [1]. To overcome some lacks several TTCN extensions concerning parallel test components [16] and modularization [17, 18] are in the process of standardization. But, none of these extensions tackle the problem specifying test cases for checking the mentioned multimedia specific functional and non-functional requirements. This is also due to the fact that some requirements, e.g., QoS aspects and timing constraints, are still research topics [5, 6, 7].

To close this gap, in 1994 the University of Berne and the ETH Zürich started a cooperation with the goal to define and implement TelCom TSL, a new *TeleCommunication Test Specification and implementation Language*. TelCom TSL shall be general enough to be used for testing *traditional* protocols and new multimedia applications. Currently, we are working on the requirements that are to be met by the new test specification language and the language definition. In this paper we present first results of our work focusing on a test system architecture for QoS testing.

Section 2 discusses a typical multimedia application scenario. We identify the QoS requirements that are specific to the application and that are to be supported by the system running the application. Since it is not common knowledge what is to be assessed during QoS testing, Section 3 is an introduction to QoS semantics, i.e., the level of support of systems in providing QoS values. Based on the analysis of QoS semantics we define QoS testing as the process of assessing the behavior of an IUT performing QoS maintenance. Particularly interesting is that QoS maintenance need not be observed directly. Varying specific QoS parameter suffice since, if the negotiated QoS values cannot further be guaranteed by the IUT, the IUT should behave as defined by the QoS semantics supported, e.g., the connection is aborted (Section 4). In Section 5, we propose the TelCom TSL testing architecture and introduce its main features. The proposed testing architecture is an extension of an existing one [13, 14]. The extensions are that we can deal with several multimedia data streams and with distributed IUTs. From the discussion of our testing architecture we derive the features that the specification language TelCom TSL has to have: a notation for the defini-

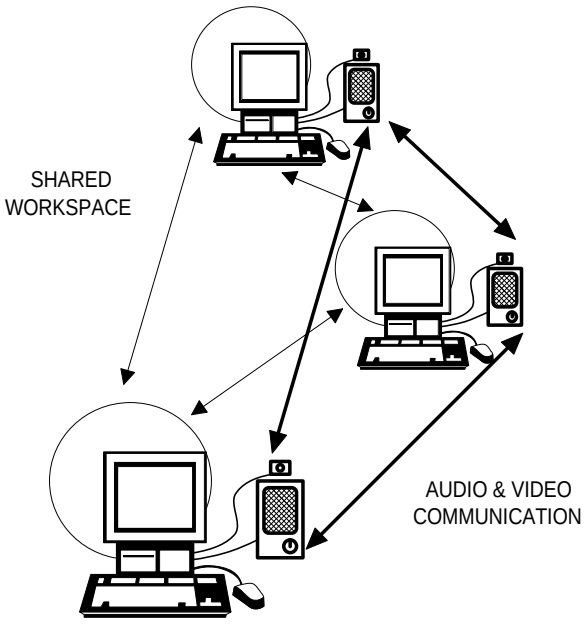

Figure 1: The teleseminar scenario

tion of functional behavior and non-functional behavior, e.g., timing constraints at any level of detail (execution time of actions, delay of message transfer, etc.). We conclude with a summary and we identify further developments of QoS testing.

2 Teleseminar - A multimedia application scenario

Teleseminar is a multimedia application that uses multimedia workstations distributed over a wide area network [2]. Each workstation acts as a communication unit that transmits, receives, and processes video, audio and data streams. Fig. 1 describes the scenario schematically.

In this scenario video and audio are used to give each participant a feeling like attending a face-to-face meeting. On each workstation video images of all participants are displayed, and, like in all face-to-face meetings, all participants are able to talk at the same time.

For sharing textual information an *application sharing tool* is used. This tool allows to display the actual working document on all workstations and provides a tele-pointer for each participant. The different tele-pointers are distinguished by an annotation which identifies the owner. The application sharing tool also allows to make changes in the actual working document by all participants. The teleseminar application implies a number of stringent requirements on the distribution of video and audio streams as well as on the distribution of data, like text or tele-pointer.

In most cases, for each stream there exist some metric for describing the quality of service (QoS). E.g., for a video stream we may distinguish high-definition-television (HDTV) quality, (PAL) color quality, and black-and-white quality. According to different application scenarios we may define different qualities as acceptable. E.g., for attending a Rolling stones concert via multimedia workstation the acceptable QoS value for video data may be black-and-white,

QoS value	video before audio	audio before video
optimal	0 – 20 ms	–
good	20 – 40 ms	0 – 20 ms
acceptable	40 – 80 ms	20 – 80 ms
not acceptable	> 80 ms	> 80 ms

Table 1: Possible assignment of QoS values for synchronization

whereas for attending a David Copperfield show we may accept HDTV quality only.

Besides QoS values for individual data streams there also exist QoS values describing the quality of synchronization between different streams. According to [21] in most cases the synchronization of data streams in multimedia scenarios is *soft synchronization*. This means that the synchronizing action can be done within some time interval. For instance, video and audio should be lip synchronized. Similarly, tele-pointer and audio should be synchronized. The extent to which the synchronization should be achieved depends on the combination of data streams: A good approximation for the synchronization of video and audio is about 80 ms whereas 150 ms are sufficient for audio and tele-pointer which means the audio should be at most 150 ms ahead the tele-pointer (i.e., the text to be pointed to should be displayed at most 150 ms after the tele-pointer is pointing to a specific location on a participant's screen) or at most 150 ms behind the tele-pointer.

Coming back to the synchronization between video and audio streams, for the time interval for lip synchronization (80 ms) we may distinguish several degrees, i.e., QoS values, of synchronization. Possible QoS values are *optimal*, *good*, *acceptable*, and *not acceptable*. Relating time intervals to these QoS values may be influenced by the preferences of the users. E.g., field trials have shown that video before audio is more accepted than the other way round [21]. Table 1 shows an example of how QoS values for synchronization may be related to time intervals.

3 QoS semantics

As explained informally in the previous section quality-of-Service (QoS) refers to a set of parameters that characterize a connection between communication entities across a network. Typical QoS parameters are [4, 5]: throughput, delay, jitter (*performance* related parameters), or residual error rates, connection establishment failure probability (*reliability* related properties), or presentation coding and security requirements (*miscellaneous* properties).

The negotiation of QoS parameters takes place between calling and called service users (e.g. multimedia application) and service provider. With service provider we refer to an entity supporting distributed applications. Particularly, the service provider is capable handling several data streams. The QoS semantics define the way how QoS parameter values are negotiated and handled during a connection. We distinguish between *best effort*, *guaranteed*, *compulsory*, *threshold*, and *mixed compulsory and threshold* QoS values.

- **Best effort QoS values.** In this scenario, the calling user requests QoS values that are considered as suggested values, i.e., the service provider has the freedom of lowering

the requested QoS value. Similarly, the called service user may also further weaken the QoS value. At the end of QoS negotiation all partners involved have the same QoS values. But this does not imply that the service provider has any obligation for taking preconditions in order to assure that the QoS is maintained for the lifetime of the connection. If the QoS becomes worse the service provider is not even expected to indicate this to the service users. Particularly, no monitoring of the negotiated QoS values is required.

- **Guaranteed QoS values.** In this QoS semantics, the calling user requests a QoS value which is to be regarded as a minimal acceptable value. The service provider has the possibility to strengthen the value or to reject the request if it cannot provide the degree of QoS requested. However, if the request is accepted and the connection is established then the service provider has the obligation for maintaining the agreed QoS values for the lifetime of the connection. In order to achieve this guarantee the permanent availability of resources allocated to the connection is required. This may imply that further connection requests are rejected since the newly requested QoS values may interfere with the QoS values of already established connections.

One can think of levels of QoS support in between *best-effort* and *guaranteed* QoS. These are *compulsory* and *threshold* QoS semantics [5, 6].

- **Compulsory QoS values.** The value for a QoS parameter to be negotiated may only be strengthened by the service provider and the called service user. When the service provider analyzes a request then the service provider may decide to reject the service request since available network resources are not sufficient to satisfy the desired QoS. However, in the case that the connection is established the QoS of the connection has to be monitored. If the service provider detects that the QoS is not longer provided as agreed during QoS negotiation, then the connection is aborted.

- **Threshold QoS values.** The negotiation of a threshold QoS value follows the same procedure as for a compulsory QoS value. Particularly, any modification to a QoS value is only allowed if it strengthen the QoS value. However, if the QoS value cannot be supported by the service provider then the calling service user gets a feedback on the weakening of the QoS value. Furthermore, whenever the service provider detects a violation of the negotiated QoS value (by monitoring the QoS values of the connection) the service users are informed about this degradation of QoS but the connection is not aborted.

- **Mixed threshold and compulsory QoS values.** A quite interesting possibility is the combination of a compulsory and a threshold QoS value. In such a case, the threshold QoS value must be stronger than the compulsory QoS value. If a connection is established then the negotiated QoS value is greater than or equal to the threshold QoS value. So, if during data transfer the monitored QoS value degrades then first the threshold QoS value is violated which results in a feedback to the service users that QoS of the connections becomes worse and possibly a connection abort may be experienced in the future.

Guaranteed QoS implies the highest degree of commitment for a service provider of maintaining the QoS of a connection. Particularly, the service provider has to take any necessary precautions that under any conceivable circumstances the negotiated QoS values are supported. For threshold and compulsory QoS the obligation of the service provider is to monitor the QoS values and to inform the service users as soon as a violation of the negotiated QoS values is detected (threshold QoS values) or to abort the connection in the case that the QoS has become worse than the negotiated ones (compulsory QoS values).

Although we have stated at the beginning of this section that *'QoS refers to a set of parameters that characterize a connection between communication entities across a network'*, this has to be refined in the context of multimedia applications. As has been explained in Section 2, multimedia applications, generally, consist of several data streams. The negotiation of QoS values should be possible for an individual data stream but should also be possible for several streams, e.g., in the case of synchronization of audio and video, QoS values might be specified by an application that defines a delay of the audio stream relative to the video stream [20]. The previously discussed QoS semantics may remain unchanged but QoS maintenance is to be applied to several data streams.

4 QoS testing issues

From the previous discussion we conclude that different QoS semantics have different impacts of QoS testing. We do not consider the negotiation of QoS values since negotiation of QoS values is a functional property of a protocol specification which can be tested using methods developed for OSI conformance testing [13]. Furthermore, we exclude best-effort QoS semantics from our consideration since no particular constraints on the behavior of a service provider are imposed by this semantics. We argue that OSI conformance testing suffice in this case.

Threshold, compulsory and guaranteed QoS semantics all require that a service provider or, more generally, a multimedia system, besides implementing the usual protocol functions, is also requested to implement additional functions for QoS maintenance, e.g., monitoring of multimedia data streams in order to determine actual QoS values or, if the synchronization of data streams is concerned, mechanism for synchronization as specified by the application.

> *QoS testing, to our understanding, refers to assessing the behavior of a protocol implementation performing QoS maintenance.*

However, it is not necessary to control and observe the behavior of an implementation directly. It suffice if the tester can eventually observe the specific behavior defined for the agreed QoS semantics. In order to provoke this behavior varying the actual QoS values is sufficient.

As an example we consider the teleseminar scenario (Section 2) but restrict ourself to video and audio streams which have to be synchronized (Fig. 2). In order to enable the synchronization of audio and video data we assume that so-called *event stamps* [20] are introduced in the data streams. Synchronization is performed relatively to these event stamps, i.e., the audio stream should be at most 80 ms ahead or after the video stream (Fig. 3).

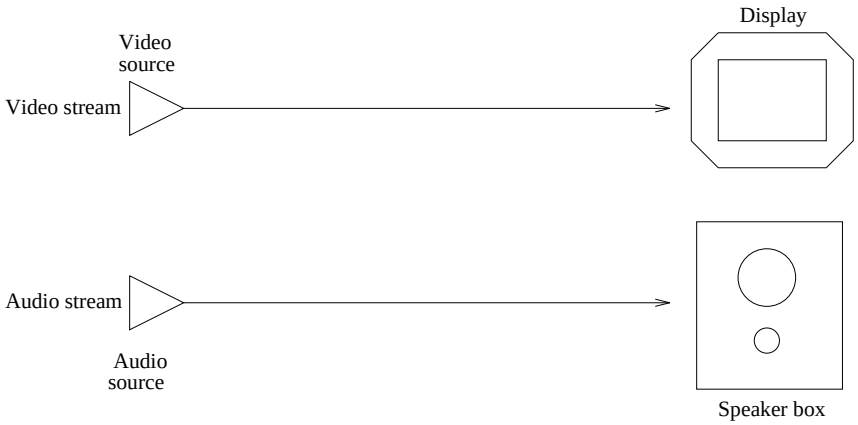

Figure 2: Video and audio multimedia application

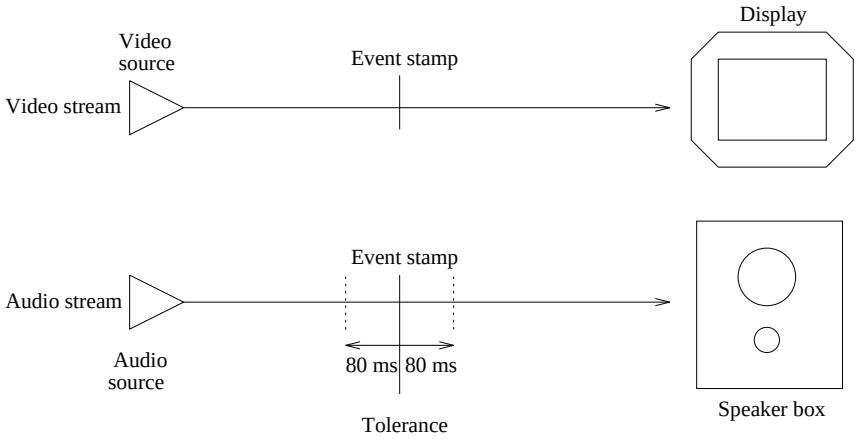

Figure 3: Synchronization of video and audio

The question is what should happen if the audio data is, for instance, delayed for more than 80 ms (not acceptable QoS (Table 1))? If we assume that the QoS semantics agreed between the multimedia applications involved is supporting *compulsory* QoS values then the users of the applications should receive an indication that the negotiated QoS values are violated and, according to the QoS semantics, the audio and video connection should be aborted.

From a testing point of view QoS assessment should give evidence that an implementation behaves as prescribed by the QoS semantics agreed. Consequently, it is required that a tester should force an implementation through a test case so that the expected (with respect to the QoS semantics) behavior of an implementation is observed.

5 TelCom TSL - A Framework for QoS Testing

The definition of *TelCom TSL* has been influenced by our work on the formal definition of TTCN, concurrent TTCN [23, 24], QoS specification and verification [22, 19], and the specification and generation of TTCN test cases based on SDL and MSC [8, 9, 10, 11]. TelCom TSL aims at defining a QoS testing architecture and a test specification language.

In this paper we concentrate on a QoS testing architecture and identify the requirements that are to be met by such an architecture to be applicable in a scenario as described above.

5.1 A QoS Testing Architecture

In our opinion QoS testing extends protocol conformance testing approaches [14, 12] in several directions:

1. The implementation under test (IUT) is distributed. Maintenance of QoS values is performed by the service provider. The service provider (in OSI terminology) consists of all those components providing the requested services to multimedia applications with the properties specified by the applications in terms of QoS parameters. Parts of such a service provider run on different systems, however, all of them have to be controlled and observed during QoS assessment.

2. Multimedia applications generally make use of several data streams. As for the parts of a distributed service provider the different data streams have to be generated and processed individually during testing. The situation becomes even worse since

3. The specification of QoS parameters (e.g., throughput, delay) may be given per data stream (e.g., for a video stream a throughput of 500 kBits/s may be request) or for several streams (e.g., synchronization requirements as previously discussed).

Figure 4 presents our ideas of a test architecture for QoS testing. The test architecture consists of *test components*, an *IUT*, and a *network facility*. The test components access the IUT via *service access points*. IUT and network facility are connected via *network interfaces*. Test components and test components and network facility are interconnected by *communication links*.

Since the IUT is distributed it is quite natural that control and observation of parts of the IUT is done by more then one test component. In our approach on each site at least one test component is running. Generally, we have one test component for each data stream. Test components are responsible for generating and processing a multimedia data stream. Besides this a test component also contains part of the test case specification, i.e., the sequence of test events to be executed in order to achieve a particular test purpose.

The network facility provides an underlying network service which is necessary for the IUT in order to provide its service. The network facility might be a real network or just a network simulator. In order to force an IUT into situations where negotiated QoS values are not guaranteed anymore, test components may interact with the network facility in a controlled manner. Such an interaction may be advising the network facility to introduce additional packet losses or to introduce additional network load. For instance, assessing that the IUT aborts a connection if the synchronization of video and audio streams is lost may be performed by instructing the network facility to delay an audio packet (or a number of audio packets) for more the 80 ms.

The above example has shown that information is transferred between test components and network facility. However, exchange of information may also be performed between test components. Those test components that control and observe video and audio streams

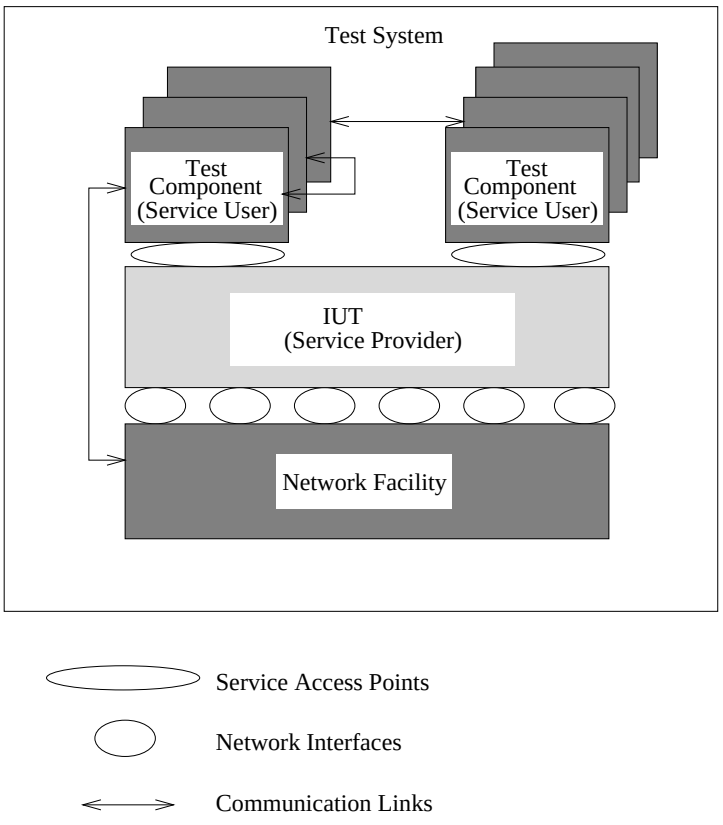

Figure 4: A testing architecture for QoS testing

may exchange synchronization information for the purpose of validating that the IUT indeed supplies video and audio as specified by the synchronization requirements.

Similar to other testing architectures [14, 12] we assume that an IUT may be a small part of a bigger software system. But unlike to existing conformance testing architectures we presuppose that the IUT is driven only via service access points. In our model the parts in which the IUT might be embedded is hidden in the network facility. The network facility abstracts from the network that supports the communication between parts of the IUT. The network facility should be configurable in the sense that various parameters that determine the communication behavior of the network facility can be controlled. This enables us to change traffic characteristics like error rates or delays.

Referring to our discussion of the synchronization of video and audio streams (Section 4) a specific QoS testing architecture configuration is shown in Fig. 5. Note, screen and speaker box have been substituted by test components. Furthermore, video and audio sources are implemented by test components too. In such a configuration the generated and transmitted video and audio information is processed in the test components to the right of Fig. 5 so that the behavior of the service provider in maintaining the synchronization of video and audio can be assessed.

Unfortunately, the latter statement is not completely true. The problem is that in the configuration shown test components *video sink* and *audio sink* do not have the information which is necessary for assessing that the video and audio stream are delivered timely. This

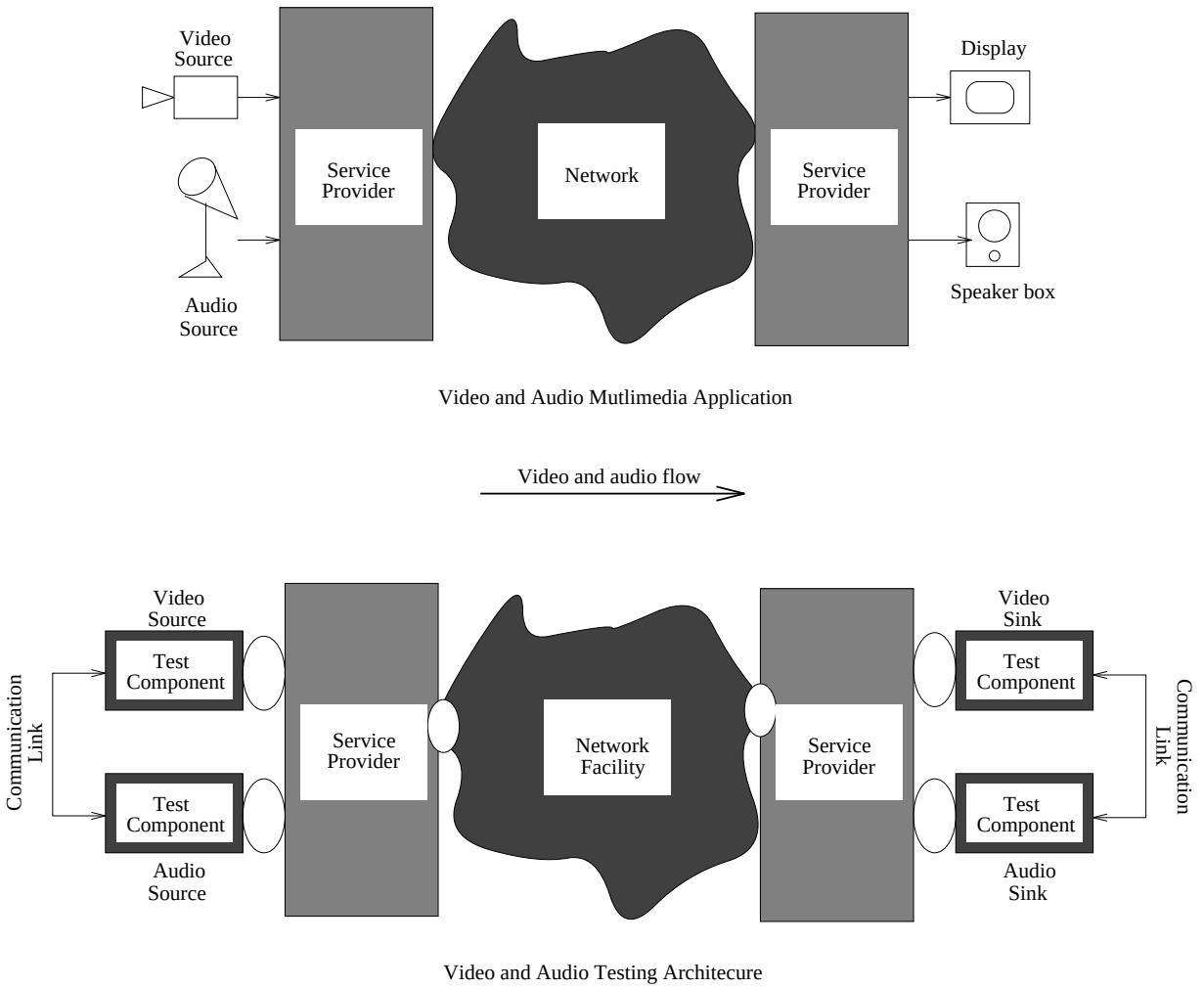

Figure 5: A testing architecture for testing synchronization of video and audio

simply is because the event stamp coded in the video and audio stream for synchronization purposes have been removed in the service provider. The streams which are leaving the service provider are plain video and audio data. A solution to this problem is moving the service access points which give control over the IUT into the service provider itself. Although this approach is possible theoretically, it is infeasible practically since, in general, we cannot assume that an IUT is providing the required access.

In our testing architecture test components can communicate over communication channels. These channels are dedicated to the exchange of control information between test components. They can be used to communicate synchronization information from test components *Video source* and *Audio source* to test components *Video sink* and *Audio sink* (Fig. 6). The synchronization information may consist of a time stamp of a video or audio packet send event. From this information and an estimation of packet transfer time and clock drifts [3] the receiving test component can compute the expected arrival time of the video/audio packet. If the video or audio packet does not arrive as predicted (with the given tolerance of 80 ms) then it can be assumed that synchronization of video and audio has been lost.

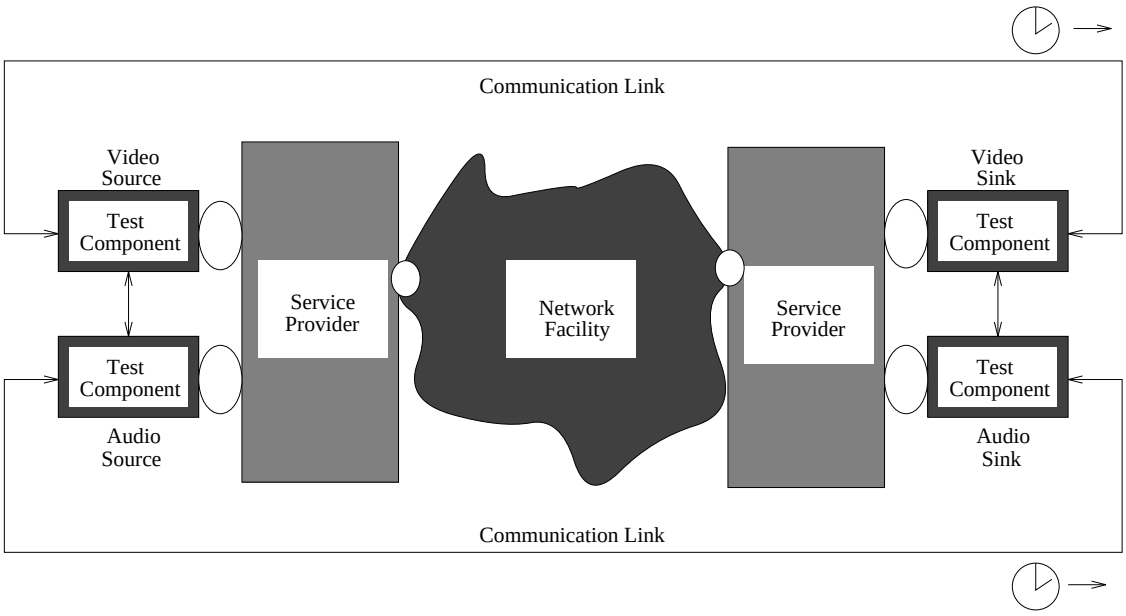

Figure 6: A testing architecture for testing synchronization of video and audio (revisited)

5.2 TelCom TSL - Features of a new Test Specification Language

TelCom TSL should meet two requirements. First, a test case specifier should define the functional behavior of a test case independently of any timing constraints that may apply. Referring back to the previous example the functional behavior of a test case for video and audio synchronization consists of a sequence of send and receive events for video and audio data. Second, the test case specifier should be enabled to add timing requirements at any time in the test case design and validation process. For instance, the variation in the delay of the audio (or video) stream is, thus, specified independently of the functional behavior of the test case.

In order to cope with the first requirement TelCom TSL provides the means for the description of the functional behavior of test systems (Figure 4) except for the IUT and network facility. The IUT is assumed to be a black box whose external behavior is visible only. The network facility is also assumed to be given, but unlike the IUT, the network facility is under the control of a test operator. Particularly, the test operator has the possibility to change the configuration of the network facility with respect to QoS characteristics. TelCom TSL structures a test system into a hierarchically ordered set of entities. We distinguish *test components*, *link processes* and the *test system* itself.

- Test components are active entities running in parallel. Test components have assigned a behavior specification that describe their behavior during execution of a test case.
- To make the distribution of test components over real systems explicit, we have introduced *test modules* which combine all test components located on one site. Test components located on the same system communicate synchronously.
- Communication between test components on different systems is supported by unidirectional link processes. Interaction between test components and link processes

is synchronous. Link processes may also be used for the coordination among test components and between test components and network facility.

IUT and test components communicate through *service access points*. IUT and network facility use *network interfaces* for communication purposes. The realization of these interfaces is not constraint by TelCom TSL. The only requirement imposed is that communication between IUT, test components, and network facility is not arbitrarily delayed but that the delay is fixed and known. This constraint stems from the requirements that QoS testing imposes stringent timing constraints and therefore a certain knowledge of the timing behavior of system components is needed (refer also to the discussion at the end of the previous section).

In order to determine the timing behavior of a test system, the internal organization of the real system executing test components and link processes may also have to be considered. If a multiprocessor system supports the assignment of test processes and (if necessary) link processes to processors, we can assume that these processes are executed in parallel. The execution of processes sharing the same processor is modeled by an arbitrary interleaving of actions of the processes involved. Based on the knowledge of the timing behavior of all components of a test real system we are able to make predictions whether a given test case with given timing constraints can be executed correctly on a specific system, i.e., whether the intended result can be achieved.

The internal organization of the test system architecture and the timing constraints of link processes, interfaces and test components are to be seen as external parameters that need not be known while specifying the functional behavior of a test case for QoS testing. If actual values for these parameters are known (later in the test case design process) then validation of the timing behavior of the test case against the QoS timing requirements becomes possible.

6 Summary and outlook

Since standardized methods and tools for protocol conformance testing are not able to cope with the specific requirements of distributed multimedia applications we have developed a method for multimedia conformance testing. We have presented our work on testing QoS aspects. We discussed the teleseminar scenario as a typical multimedia application where QoS aspects play an important role. Moreover, QoS values are not restricted to one data stream only, but QoS values may also be defined for describing the synchronization between different data streams. We have introduced QoS semantics in the context of multimedia applications and discussed the resulting QoS testing issues: QoS testing is defined as the process of assessing the behavior of an implementation performing QoS maintenance. We have argued that it is not necessary to control and to observe an implementation performing QoS maintenance directly but that it is sufficient to observe the behavior of an implementation in cases where the negotiated QoS values are violated. Based on this discussion we have presented the TelCom TSL testing architecture and have listed some features of TelCom TSL.

The development and implementation of TelCom TSL is a research cooperation between the ETH Zürich and the University of Berne which started in 1994. A funding of this research by the Priority Programme Informatics of the Swiss National Fund is currently under consideration. Our research comprises the following research tasks:

1. *Requirements analysis:* The requirements that are to be met by the new test case specification language will be analyzed.

2. *Language definition:* The syntax and semantics of the language will be defined.

3. *Validation and simulation:* Appropriate theories for the validation of test cases with respect to functional and non-functional properties will be defined (or adapted if existing approaches are sufficient).

4. *Tools:* We intend to provide tools supporting every activity in the test life cycle.

5. *Implementation:* The test case implementation will be supported by a compiler and runtime libraries.

This paper has presented initial work on Task 1 and first ideas on Task 2. We plan to start with syntax and semantics definition of TelCom TSL in early 1996.

References

[1] B. Baumgarten. *Open Issues in Conformance Test Specification*, Proceedings PTS VII, 1994.

[2] BETEUS Consortium. *Broadband Exchange for Trans-European USage - Functional Specifications*, BETEUS M1010, Deliverable 2, 1994.

[3] G. Coulouris, J. Dollimore, T. Kindberg. *Distributed Systems - Concepts and Design Second Edition*, Addison-Wesley, 1995.

[4] CCITT/ITU, *Data Communication Networks Open Systems Interconnection (OSI) Model and Notation, Service Definition*, X.200 - X.219, 1988.

[5] A. Danthine, Y. Baguette, G. Leduc, Léonard, *The OSI 95 Connection-Mode Transport Service - The Enhanced QoS*, A. Dantine, O. Spaniol (Editors), *High Performance Networking*, IFIP, 1992.

[6] A. Danthine, O. Bonaventure. *From Best Effort to Enhanced QoS*, CIO Deliverable, No. R2060/ULg/CIO/DS/P/004/b1, 1993.

[7] R. Gopalakrishan, G. Parulkar. *Application Level Protocol Implementations to Provide QoS Guarantees at Endsystems*, IEEE *Ninth Annual Workshop on Computer Communication*, 1994.

[8] J. Grabowski, D. Hogrefe, R. Nahm. *Test Case Generation with Test Purpose Specification by MSCs.* In O. Faergemand and A. Sarma, editors, *SDL'93 - Using Objects.* North-Holland, October 1993.

[9] J. Grabowski, D. Hogrefe, R. Nahm, A. Spichiger. *Die* SAMSTAG *Methode und ihre Rolle im Konformitätstesten.* In *Praxis der Informationsverarbeitung und Kommunikation (PIK) Nr. 4/94*, K.G. Saur Verlag, München, December 1994.

[10] J. Grabowski, D. Hogrefe, I. Nussbaumer, A. Spichiger. *Improving the Quality of Test Suites for Conformance Tests by using Message Sequence Charts.* In *Proceedings of the 'Fourth European Conference on Software Quality' in Basel (Switzerland)*, October 1994.

[11] J. Grabowski, D. Hogrefe, I. Nussbaumer, A. Spichiger. *Combining MSCs and Data Descriptions in order to Generate Executable Test Cases for ISDN Systems.* In *Proceeding of the XV International Switching Symposium (ISS'95) - World Telecommunications Congress*, Berlin, April 95.

[12] D. Hogrefe *Conformance Testing Based on Formal Methods*, FORTE 90, North-Holland, 1990.

[13] ISO/IEC. *Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 1: General Concepts*, ISO/IEC 9646-1, 1994.

[14] ISO/IEC. *Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 2: Abstract Test Suite Specification*, ISO/IEC 9646-2, 1994.

[15] ISO/IEC. *Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN)*, ISO/IEC 9646-3, 1992.

[16] ISO/IEC. *Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN): Amendment 1: TTCN Extensions*, ISO/IEC 9646-3 DAM 1, 1993.

[17] ETSI TC MTS. *Methods for testing and specification (MTS); Partial and multi-party Abstract Test Suites (ATS)*, ETSI DTR/MTS-00021, 1994.

[18] ETSI TC MTS. *Draft Technical Corrigendum on defect report 9646-3/002 Modular TTCN*, Contribution from H. Afsharazad and A. Flodin (both from Telelogic AB, Sweden) to ETSI TC MTS, July 1995.

[19] T. Plagemann, B. Plattner, M. Vogt, T. Walter. *A Model for Dynamic Configuration of Light-Weight Protocols*, Proceedings IEEE Third Workshop on Future Trends of Distributed Computing Systems, IEEE, 1992.

[20] R. Steinmetz. *Synchronization Properties in Multimedia Systems*, IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, April 1990.

[21] R. Steinmetz. *Multimedia-Technologie Einführung und Grundlagen (in German)*, Springer-Verlag, 1993.

[22] J. Montiel, E. Rudolph, J. Burmeister (Editors), *Methods for QoS Verification and Protocol Conformance Testing in IBC - Application Guidelines* -, TOPIC, 1993.

[23] T. Walter, J. Ellsberger, F. Kristoffersen, P.v.D. Merkhof. *A Common Semantics Representation for SDL and TTCN*, Proceedings PSTV XII, North-Holland, 1992.

[24] T. Walter, B. Plattner. *An Operational Semantics for Concurrent TTCN*, Proceedings PTS V, North-Holland, 1992.