A Computational and Engineering View on Open Distributed Real-Time Multimedia Exchange

Peter Leydekkers[1,3], Valérie Gay[2] and Leonard Franken[3]

[1]TINA - Core Team, 331, Newman Spring Rd, Bellcore, Red bank, NJ 07701, USA
[2]Université Paris VI, Laboratoire MASI, 4, place Jussieu, 75252 Paris Cedex 05 - France
[3]PTT Research, P.O. Box 15000, 9700 CD Groningen - The Netherlands

Abstract. *An important requirement for distributed multimedia applications is the support of real-time communication and the means to specify real-time aspects. The aim of this paper is to extend RM-ODP and TINA-C computational and engineering views on distributed systems for the specification and support of real-time communication. It is expected that these bodies have a major impact in the area of distributed processing. However, concepts and mechanisms to support real-time communication are not yet fully included or detailed in these standards. In particular this paper addresses Quality of Service (QoS) specifications for continuous dataflows. These QoS specifications are described from the ODP computational and engineering viewpoint and the repercussions of these QoS specifications for functions located in both the computing and telecommunications environment are discussed.*

Keywords. Distributed Multimedia Architectures, QoS, ODP, TINA-DPE

1. Introduction

In the near future, distributed multimedia applications such as video-on-demand and multimedia conferencing services will operate in a Distributed Processing Environment (DPE). The DPE is a distributed platform that offers important properties such as heterogeneity and distribution transparency. In a distributed environment heterogeneity may include: equipment heterogeneity due to a multi-vendor environment, operating system heterogeneity due to different operational contexts (office, factory), and authority heterogeneity (e.g. co-operation between separate network providers).

In a distributed multimedia context, an important requirement for the DPE is the provision of real-time communication and the means to specify real-time aspects for distributed applications. This paper addresses this real-time communication aspect from both the ODP computational and engineering viewpoint. It is closely aligned to RM-ODP, TINA and OMG since it is expected that these 'standardisation' bodies will have a major impact in the area of distributed processing. However, in the area of multimedia communication these 'standardisation' bodies have some weak points:

– They do not specify a complete language to specify real-time interaction at the computational level. OMG defines a language which is used as a basis by TINA-C and RM-ODP but it does not incorporate the streams concept that is used to model continuous dataflows which is essential for real-time multimedia

communication. They also do not provide a language to specify the non-functional properties of objects. In particular QoS specifications are required for stream interfaces to guarantee real-time multimedia communication.
- They do not provide (complete) mapping rules to relate a computational real-time specification to an engineering configuration which can be executed.
- Concerning the engineering configuration, even if their architecture is flexible enough to integrate the functionalities required by the real-time application, there is still the need for further research to define and design the functions required for an open real-time platform.

This paper addresses these issues using the TINA DPE platform as a basis and proposes solutions for real-time multimedia communication based on concepts defined in RM-ODP.

2. TINA-DPE and RM-ODP

The Telecommunications Information Networking Architecture Consortium (TINA-C) is a world-wide initiative that consists of members from the computer industry, telecommunication network operators and telecommunication vendors. TINA-C is defining an open architecture which will enable the rapid deployment of telecommunication services. The TINA-architecture is based on techniques such as object-orientation and distributed computing and it uses as much as possible concepts and standards from both the telecommunication and computing area. One of the main goals of TINA-C is to define a platform that supports the execution of distributed telecommunication applications. This platform is called the TINA Distributed Processing Environment (TINA-DPE).

In this paper we present two views on the TINA-DPE as shown in Figure 1, i.e. a computational view and engineering view. These viewpoints are derived from RM-ODP [1] and adopted by TINA. RM-ODP identifies five viewpoints from which a distributed system can be described. However, the computational, engineering and technology viewpoint are of primary interest for the specification of real-time multimedia exchange in a distributed environment.

The *ODP computational view* on the TINA-DPE (upper plane in Figure 1) specifies a distributed application in terms of computational objects that interact with each other in a distribution transparent way. A computational object provides a set of services that can be used by other objects. To enable other objects to access a service, an object offers a computational interface which is the only means by which other objects can use the service thereby providing data encapsulation. Complexities introduced by the distribution can be hidden from applications in this view on the TINA-DPE. This can be done using transparencies which are used to hide aspects of systems that arise through their distribution. The applications using the DPE platform may select those transparencies they need and handle other aspects of distribution characteristics as they want. The DPE platform supports, for instance, location transparency which implies that applications are not concerned with communication aspects and on which nodes the (computational) objects are located. This implies for example that applications that interact with each other do not need to be aware of the physical locations of each other.

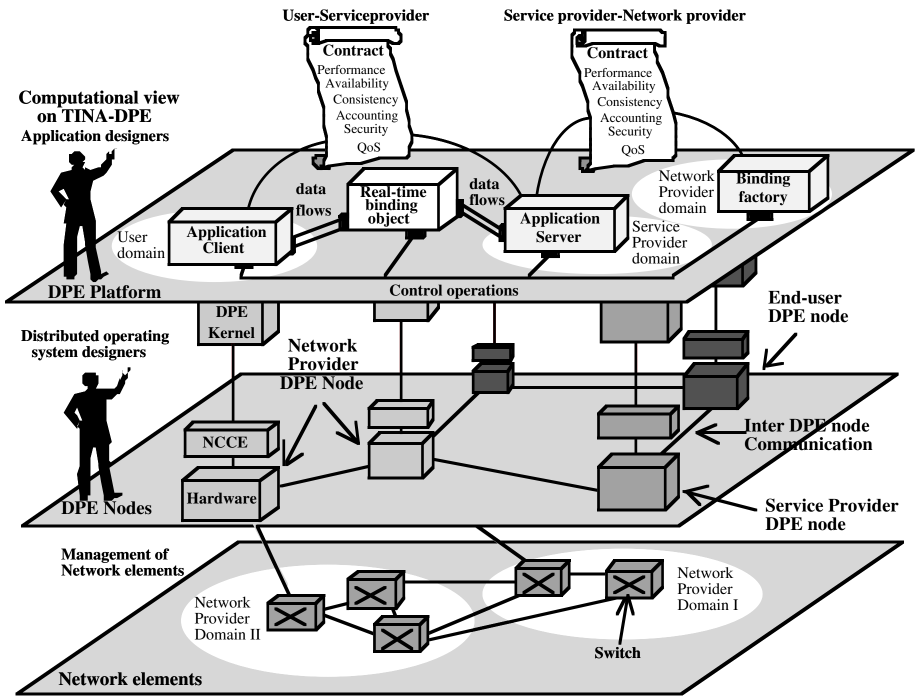

Fig.1. TINA Distributed Processing Environment

For computational objects to interact in a meaningful way the specification of well-defined interfaces in TINA and ***contracts*** are important. A contract can be specified between objects describing the agreed non-functional properties which should be observed in order for the application to operate properly. Several contract classes exist addressing different issues (e.g. security or accounting) but in this paper we focus on the ***QoS contract*** in relation with real-time interaction for continuous flows such as audio and video. A QoS contract provides a specification of the service provided and 'level of service' agreed between the involved computational objects.

The ***ODP engineering view*** and corresponding engineering language describe the mechanisms and functions required to support distributed execution and interaction between (computational) objects. The ODP engineering view on the TINA-DPE (middle plane in Figure 1) shows a collection of ***DPE nodes***, a user node, service provider node and a network provider node each having different characteristics. A DPE node provides the mechanisms to support distribution transparency as assumed in the computational view and hides the native computing and communication environment from the application designer. For applications to be capable of interoperating with applications on the other DPE nodes there should be a (minimum) agreed set of functionality that is available on each DPE node.

A DPE node is composed of three parts, i.e. the DPE kernel, the Native Computing and Communications Environment (NCCE) and Hardware. The *DPE kernel* is a software layer that is available on each DPE node. It is a software layer between the local operating system and application components. It provides a set of basic functions such as communication, storage and processing capabilities. The *NCCE* describes the local operating system and basic communication functions that are used by the DPE kernel. The *Hardware* describes the devices and hardware resources available on a DPE node. The description of the hardware and NCCE is part of the ODP technology view on a system and is often vendor specific.

The network provider provides communication services (bottom plane in Figure 1) to interconnect the user and service providers that are geographically distributed. The network provider has access to and controls the network elements for the purpose of set-up, release and maintenance of network connections. Network elements represent the resources in the telecommunication network and consist of elements such as switches, cross-connects and video-bridges.

It is worth noting that the elements in the DPE platform plane and DPE node plane in Figure 1 need not to have a one-to-one relation. Computational objects in the DPE platform plane may correspond to one or multiple (engineering) objects in the DPE node plane and may also be distributed over multiple DPE nodes.

3. Computational view on real-time exchange

Focusing on the computational view on the DPE (upper plane of Figure 1) real-time aspects are mainly reflected in the ODP computational concepts of ***stream interface***, ***explicit binding object*** and ***environment contract*** [1]. Stream interfaces are used for objects to exchange real-time continuous data (e.g. voice, video). For real-time exchange it is important to be able to specify QoS as part of the binding between computational objects (interfaces). QoS properties can be specified in a static way by means of declarations attached to the object and its stream interfaces (expressed in an environment contract).

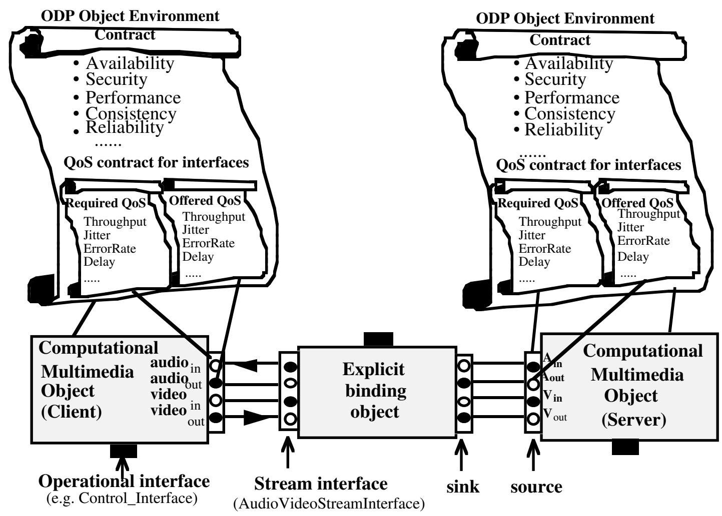

Fig. 2. Object environment contract, stream interface and binding object

In a dynamic way QoS properties are handled by the ***explicit stream binding*** process. Figure 2 highlights and relates the computational concepts of concern for real-time specifications.

3.1 Static specification of real-time exchange

The stream interface concept in RM-ODP can be used to describe continuous flows that have tight real-time constraints. Stream interfaces represent a communication end-point that may be a source or sink of continuous flows. Stream interfaces are described in detail in [9] using the TINA-ODL (Object Definition Language) specification language. TINA-ODL is an extension of OMG-IDL (Interface Definition Language) [11] that enables to express telecommunication and multimedia oriented computational specifications. Based on the syntax template for stream interfaces as proposed in [9], a simplified TINA-ODL specification for a multimedia computational object is shown in Table 1 and the related stream interface specification in Table 2.

A ***computational object template*** [1] comprises a set of computational interface templates which the object can instantiate, a behaviour specification and an environment contract specification.

The QoS specification is inspired on [4] which allows the separate specification of the guarantee level required for each QoS parameter. In general, three classes of guarantee level are identified, i.e., deterministic, statistical reliable and best effort. For each guarantee level this results in different commitment specification of the QoS parameter.

```
object template  Computational_MultiMedia_Object;
typedef enum guarantee { Deterministic, StatisticalReliable, BestEffort};

supported interfaces /* interfaces that are offered by the object. */
        AudioVideoStreamInterface;
        ControlInterface;

/* Describes non-functional aspects that apply to the whole object inclusive interfaces. */
environment  contract
 struct QoSContract {
        union Performance switch (guarantee) {/*expressed in response time in seconds*/
             case Deterministic:        real Peak;
             case StatisticalReliable:  real Mean;
             case BestEffort:           struct interval {real min, max;} Bound;
        };
        union Availability switch(guarantee){/*expressed in mean time between failure*/
             case Deterministic:        real Peak;
             case StatisticalReliable:  real Mean;
        };
        /* similar descriptions can be made for Reliability, Consistency, Security etc. */
 }      multimediacontract;

behaviour /* describes the behaviour of the object */
```

Table 1. TINA-ODL specification of a MultiMedia terminal object template

A ***computational stream interface template*** consists of a finite ***set of action templates***, one for each flowtype in the stream interface. Each action template for a flow contains the name of the flow, the information type of the flow and an indication of causality for the flow since flows are unidirectional (producer or consumer but not both), a ***behaviour*** specification and an ***environment contract*** specification.

```
interface template  AudioVideoStreamInterface;
typedef struct VideoFlowType {...};  /*A VideoFlowType describes the characteristics
of video and attributes are e.g. codingtype, resolution, colordepth. */
typedef struct AudioFlowType {...};/*AudioFlowType  characteristics are e.g
                                       compressiontype, samples/s. See [9] for details. */

typedef enum guarantee { Deterministic, StatisticalReliable, BestEffort};
environment  contract
  struct StreamQoS {
      union Throughput switch (guarantee) /* expressed in samples/s or frames/s */
          case Deterministic:        real Peak;
          case StatisticalReliable:  real Mean;
          case BestEffort:           struct  interval {real min, max;} Bound;
      };
      union Jitter  switch (guarantee) /* expressed in samples/s or frames/s */
          case Deterministic:        real Peak;
          case StatisticalReliable:  real Mean;
      };
      /* similar definitions for Delay and Errorrate */
  } requiredVideoQoS, requiredAudioQoS, offeredVideoQoS, offeredAudioQoS;

sink   display       VideoFlowType with requiredVideoQoS;
sink   speaker       AudioFlowType with requiredAudioQoS;
source camera        VideoFlowType with offeredVideoQoS;
source microphone    AudioFlowType with offeredAudioQoS;

behaviour /* This part describes the behaviour of this AudioVideoStreamInterface */
```

Table 2. TINA-ODL specification of a Audio video stream interface template

In RM-ODP the ***environment contract*** concept can be used to specify the QoS attributes of a computational object and its interfaces (Table 1, 2). The values in an environment contract are specified between a set of objects which are possibly located in different domains (Figure 1). Such a contract provides a specification of the service provided and of the 'level of service' agreed between the involved objects. It expresses the requirements and obligations which have to be fulfilled and it addresses issues such as the performance, availability, and security aspects of the objects and indications of behaviour which invalidates the contract (e.g. severe QoS-degradation). An environment contract between computational objects is determined during the set-up phase. QoS negotiation is possible during this phase and once agreement is achieved the QoS values can be monitored by a QoS manager.

Based on this contract, specific object environment contracts can be determined for each involved computational object (Figure 2). For each interface a contract is specified. We distinguish between ***offeredQoS*** contract (i.e. for outgoing flows) and

requiredQoS contract (i.e. for incoming flows) for stream interfaces since the QoS requirements may be different for each flow [13].

The offeredQoS specifies the values by which the stream interface will transfer its output to other objects. These parameters are described in the QoScontract of the AudioVideoStreamInterface (Stream_QoS in Table 2). The requiredQoS specification is similar to the offeredQoS specification and represents how the object expects frames or packets to be delivered at its input.

The guarantee level agreed should be obeyed by the object. Three classes of service commitment are distinguished, deterministic, statistical and best effort. If the object can not maintain the agreed service commitment the involved objects should be informed.

3.2 Dynamic specification of real-time exchange

In addition to the specification of stream interfaces and QoS contracts, it is important to check the compatibility of stream interfaces prior to data exchange and the ability to express some real-time control on the set of stream interfaces that are bound.

The ***binding object*** is a suitable ODP concept to describe these aspects of real-time exchange. It represents an end-to-end association between two or more computational objects. The binding object is used to abstract over end-to-end connections and is responsible for compatibility checks between the stream interfaces (in particular the non-functional aspects such as QoS). It provides several operational control interfaces for monitor and control purposes during the exchange of continuous media. These control interfaces may be used by other computational objects such as the QoS manager (Figure 3).

A binding object is of a certain type that defines its configuration possibilities and its characteristics. Its type influences the engineering configuration and in [2] an example is described of a ‘multiparty’ binding object. In this paper, we consider a ***real-time stream binding object***.

Figure 3 shows an example of an explicit binding action to create a real-time binding between three computational objects similar to the configuration shown in Figure 1. In our example, the server object interacts with the binding factory object. It asks for the creation of a binding object of type ‘real-time’ to link and control three stream interfaces, each of them having non-functional parameters specified. The binding factory creates the real-time stream binding object to enable the data exchange necessary for the binding. The binding object is an abstraction of the connection(s) set-up in the telecommunication network (lowest plane in Figure 1). The binding factory functionalities are compatible with those supported by Communication Session Manager as described in TINA and Eurescom P.103 [12].

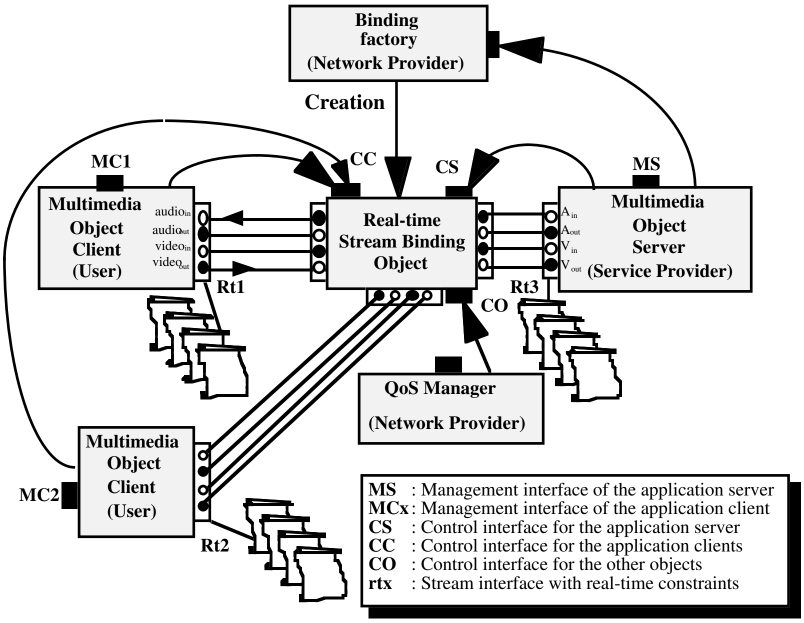

Fig. 3. Binding process

Client and server stream interfaces are linked and controlled by a real-time stream binding object and exchange of data is now possible. The binding object has several control interfaces used by the application server (CS), the application clients (CC) and other computational objects (CO). The CO interface will be used for example by the QoS manager (owned by a network provider) to monitor the network QoS and to initiate operations (e.g. QoS-renegotiation) in case of QoS contract violations. The binding object may also invoke signal operations on the clients and servers to notify particular events (e.g. synchronisation events). The binding object should provide the QoS guarantees between the interacting interfaces as described in the QoS environment contract.

4. Engineering view on real-time exchange

In the engineering view (Figure 1 middle plane) functions and mechanisms are identified which are required to support real-time stream communication. Many functions of a DPE node are derived from RM-ODP. RM-ODP contains general descriptions of functions fundamental to the construction of ODP systems (like the TINA DPE), but it does not define detailed mechanisms for these functions. The functions defined in RM-ODP are management, co-ordination, repository and security functions [1].

In general, each DPE ***node*** contains a ***nucleus*** that co-ordinates processing, storage and communication functions for use by other engineering objects within the node to which it belongs (Figure 4). Engineering objects (e.g. Video object, QoS manager) are grouped into a ***capsule*** which is a configuration of objects forming a single unit for processing and storage. Functionalities such as capsule creation, deletion and checkpointing are supported by the capsule manager. The ODP ***channel*** concept [1] provides engineering mechanisms that enable communication between engineering objects that are either located in the same DPE node or on a remote DPE node.

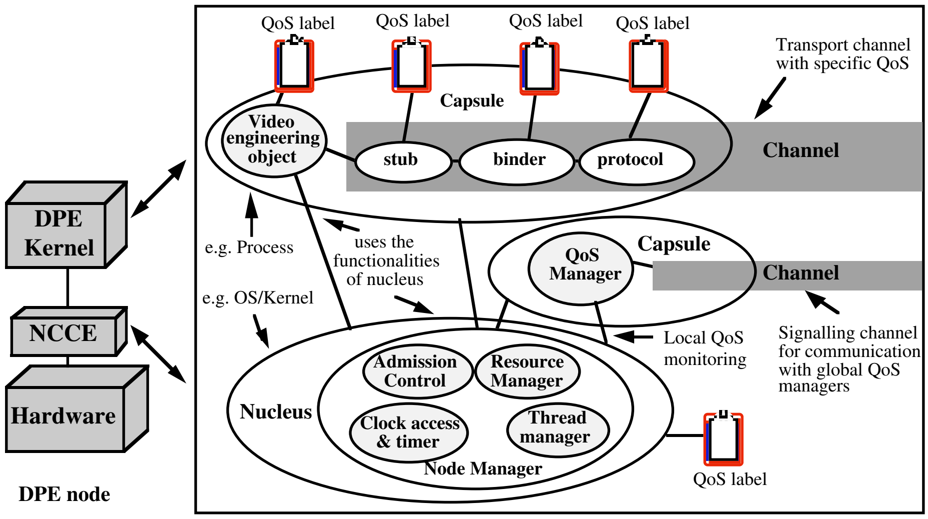

Fig. 4. RM-ODP engineering view on a DPE node

For the support of real-time multimedia exchange, specialisation is required (e.g. thread manager) and additional functions are needed to be included such as QoS management and resource management (section 4.2).

4.1 Computational QoS contract related to engineering QoS labels

The computational QoS specification provides an abstract expression of QoS which does not make any reference to engineering or technology mechanisms. A translation has to be made onto the engineering configuration that realises the QoS aspects as specified in the (computational) contracts. A computational QoS specification covers both the end-to-end aspects as well as the QoS aspects of the end systems DPE node. To check if the suitable amount of resources from the nucleus are allocated to the capsules we introduce ***QoS labels*** to guide the allocation process. The labels describe the QoS aspects of the engineering components. The values of the QoS labels specify the available capacity of the nucleus, and the required capacity by the capsules. The QoS labels associated to the channel objects guide the allocation of communication resources in order to realise the required throughput, delay and jitter [6].

For example, the specification of throughput expressed by the number of frames/sec in the computational QoS contract will be translated to throughput in Mbit/s for the protocol object and compression/decompression speed of the stub object in the engineering viewpoint. Furthermore, the video engineering object needs per invocation (for the processing of a frame) a number of milliseconds on a certain CPU type located in the nucleus. It is important to realise that a computational QoS parameter may influence several QoS parameters in the engineering viewpoint. The computational throughput can be derived from the combination of engineering objects using queuing theory or operations research [7], [8].

The check if the engineering objects can fulfil the required throughput depends on the capacities available of the hardware components such as CPU, memory, storage and communications. These descriptions should be expressed in a technology viewpoint description of the system.

The approach described above is in its initial stage and subject to future work in TINA-C.

4.2 Functions required for real-time support

The DPE node that deals with real-time dataflows accounts for several functions that can monitor and control the resources available on a DPE node.

The ***resource manager*** should be available on each DPE node and has a view of the available resources on a particular DPE node (CPU, memory, etc.). In case of a network provider DPE node it has an overview of the network resources available that are located in his domain. Resource allocation is based on special mechanisms which might be different for each DPE node. In general resource allocation for the object is based on the object's ***QoS labels*** and the computational QoS contract. Especially for real-time services it is important to have strict resource management policies so that real-time QoS contracts can be not only met at service instantiation but also during the life-time of the service.

The ***QoS manager*** checks if the DPE node can and, will fulfil the computational contracts as agreed. The QoS manager uses monitoring and control functions to perform its task. It requests the resource manager for certain resources (e.g. buffer, CPU-time, Audio-device etc.) that can be allocated to a particular application. The QoS manager in the DPE node uses the QoS labels to check if the specified contract can be realised. Global QoS managers have the task to check if the computational contracts between user and service provider can be met in relation to the network provider.

The ***Node management*** function manages the threads in a DPE node. Several threads will be active to deal with real-time flows on a DPE node. (In general for each flow a separate thread will be running since the devices are located in different capsules [9]). Due to the timeliness of these flows, threads have to be scheduled in some manner to satisfy the timeliness constraint. Customised ***thread scheduling*** algorithms can be used for real-time exchange. Pre-emptive scheduling is most appropriate for real-time flows, where a running thread can be stopped and another thread allowed to run. Different pre-emptive policies can be applied such as First In First Out, Round Robin or Time-sharing [14].

Admission control functions are needed to decide whether new (engineering) objects can be executed without offending the existing QoS contracts that are already established for other objects running on the DPE node. The admission control functions have a different scope for each type of DPE Node. In case of an end-user DPE node the admission control checks the CPU, memory and devices that are in use and makes decisions upon this workload. For a network provider DPE node the admission control function checks the workload of the network elements, available bandwidth etc. The allocation policy of network resources should be in accordance to the guarantee level that is requested for the binding (i.e. statistical, best effort or deterministic).

As indicated above different engineering configurations will exist depending on the concerned operating environment. For instance, the DPE kernel of an end-user DPE node will have different mechanisms described for its QoS managers than the network provider DPE node.

5. Conclusion

From the computational viewpoint, this paper describes how the RM-ODP concept of environment contract can be used to specify real-time requirements for objects and interfaces. It also shows how real-time requirements are handled by a real-time stream binding object. The approach described in this paper to assign an environment contract to each flow can also be applied to operational interfaces. We suggest to extend OMG-IDL with the concept of ***stream interface*** and ***environment contract*** since they are important for the specification of applications operating in a telecommunications environment. An outline is given of QoS specifications using the computational environment contracts. Further work is needed to complete the specifications and to provide ‘standard’ contract environment descriptions which can then be used to specify the non-functional aspects of distributed applications.

A lot of open issues still remain and future work is needed how to make the appropriate QoS translation from a computational to an engineering specification. This translation is very complex since it also depends on the engineering configuration of objects that represent a computational object. With respect to QoS this paper presents the first ideas of relating computational QoS specifications to engineering objects and associated engineering QoS labels. In order to enable the mapping, QoS labels have to be refined for all engineering objects and functions.

In the engineering view, we added several engineering functions necessary for DPE nodes that need to deal with real-time multimedia applications. It shows how they fit in the RM-ODP and TINA-C engineering view on distributed systems. Future work remains to be done to describe the functions in the context of ODP standardisation. This paper may serve as an input for the RM-ODP standardisation group on the work item ‘QoS in Open Distributed Processing’.

6. References

1. Basic Reference Model of Open Distributed Processing:
‘Part 2: Foundations (IS)’, ITU/T X.902-ISO 107046-2.
‘Part 3: Architecture (IS)’, ITU/T X.903-ISO 107046-3. February 1995.

2. V. Gay, P. Leydekkers, R. Huis in ‘t Veld, *‘Specification of Multiparty Audio and Video Interaction Based on the Reference Model of Open Distributed Processing’*, Computer Networks and ISDN Systems - Special issue on RM-ODP, 1995.

3. K. Nahrstedt, R. Steinmetz, *‘Resource Management in Multimedia Networked Systems’*, submitted to IEEE Computer, Special issue on Multimedia, April ‘95.

4. A. Campbell, G. Coulson, D. Hutchison, *‘A Quality of Service Architecture’*, ACM SIGCOMM, Computer Communication Review, June 1994.

5. R. Gopalakrishna, G. Parulkar, *‘Efficient QoS Support in Multimedia Computer Operating Systems’*, Washington university report WUCS_TM_94_04, August 94.

6. L.J.N. Franken and B.R.H.M. Haverkort, *‘The Performability Manager’*, IEEE Network: The Magazine of Computer Communications, Special Issue on Distributed Systems for Telecommunications, volume 8, 1994

7. L.J.N. Franken and R.H. Pijpers and B.R. Haverkort, *‘Modelling Aspects of Model Based Dynamic QoS Management by the Performability Manager’*, In Lecture Notes in Computer Science, Springer-Verlag, Volume 794, Proceedings of the 7th International Conference on Computer Performance Evaluation. Modelling Techniques and Tools, Vienna, Austria, 1994

8. L.J.N. Franken, P. Janssens, B.R.H.M. Haverkort and E.P.M. Van Liempd, ‘*Quality of Service Management in Distributed Systems using Dynamic Routation’,* Proceedings of ICODP`95, Brisbane Australia, February 1995.

9. A.T. van Halteren, P. Leydekkers, H.B. Korte*, ‘Specification and Realisation of Stream interfaces for the TINA-DPE’*, Proceedings of TINA’95 Conference, Melbourne Australia, p.299-312, February 1995.

10. M. Jacobs, P. Leydekkers, ‘*Specification of Synchronisation in Multimedia Conferencing Services using the TINA Life-Cycle Model*, Proceedings of SDNE’95, Whistler Canada, June 1995.

11. The Common Object Request Broker: Architecture and Specification, OMG Document Number 91.12.1, Draft edition, December 1991.

12. M. Zweiacker (editor), *‘Network Resource Model’*, Eurescom Project P103, Technical report 7, December 1994.

13. J.B. Stefani, *‘Computational Aspects of QoS in an object-based, distributed systems architecture’*, 3rd International Workshop on Responsive Computer Systems, Lincoln, NH, USA, September 93.

14. M. Kudela, K. MacKinnon*, ‘Engineering Modelling Concepts (DPE Kernel Specification)’*, TINA document TR_KMK.001_1.1_94, restricted distribution, November 1994.