An Experimental Real-Time Object-Oriented Database Architecture for Intelligent Networks

Juha Taina and Kimmo Raatikainen

University of Helsinki, Department of Computer Science
P.O.Box 26 (Teollisuuskatu 23), FIN-00014 University of Helsinki, Finland
e-mail: {juha.taina,kimmo.raatikainen}@cs.helsinki.fi
fax: + 358 0 708 44441

Abstract

Future telecommunication services will extensively exploit database technology. The persistent and temporal information needed in operations and management of the telecommunication networks and services will be in databases. The current IN Recommendations of ITU-T (Q.1200 Series) imply that real-time transaction processing capabilities should be provided. We present an experimental real-time object-oriented database architecture that can be integrated with teleoperator's service provision architecture.

1. Introduction

Databases will, already in the near future, have an important role in telecommunication. The databases will contain the persistent and temporal information needed in operations and management of the telecommunication networks and services. Such databases exhibit stringent reliability, availability, and performance requirements. Thousands of retrievals must be delivered in a second. Allowed downtime is only a few seconds in a year. The globalization of services implies that databases managed by different operators must cooperate. The interoperability requires that the database systems must be open, must use standardized protocols, must be protected against misuse and intruders, and must have a common logical view of information.

One of the major areas where database will heavily be used is Intelligent Network (IN) services. In the Darfin[1] project we have identified the needs that can be found in the ITU-T recommendations for IN database services [1,2]. The most important requirements are object orientation, real-time access to data, and fault tolerance. Starting from those requirements we have designed a database architecture. The architecture is especially designed for IN but can also be exploited in Telecommunications Management Network (TMN) as well as in other telecommunication architectures.

The architecture is a real-time object-oriented database architecture that can be used to implement database services for telecommunications. The architecture is still evolving; during the design and implementation of a prototype we have identified several areas calling for further studies. In particular, combining high-speed real-time transactions with object orientation needs further enhancements.

[1] Darfin is the acronym of the project name *Database Architecture for Intelligent Networks.*

The rest of this paper is organized as follows. In Section 2 we summarize related work carried out elsewhere. In Section 3 we briefly describe telecommunication architectures that pose requirements for the database architecture. In Section 4 we present our database architecture and the current stage of prototype implementation.

2. Related work

Our work in progress can be divided into three subtasks: requirements analysis for telecommunications, object-oriented databases, and real-time databases. Each of them, particularly object-oriented and real-time databases, is currently popular research topics. Below we briefly summarize recent related work.

Requirements analysis for telecommunications. Work related to the requirement analysis of databases needs in telecommunications derives from the international standardization done in ITU-T. Databases have a major role in Intelligent Networks (IN) and Telecommunications Management Networks (TMN). From database management point of view most interesting part of IN is the IN *Distributed Functional Plane* (DFP) [3], in particular the recommendations in the *Capability Set 1* (IN CS-1) [4]. In the recommendations the database requirements are given as functional description of the element called *Service Data Function* (SDF). Database requirements for telecommunications based on IN CS-1 have been studied in Eurescom research projects [5,6]. Based on the ODP [7,8] engineering viewpoint the database requirements have also been examined the in [9].

Database issues in networks have lately gained growing interest; see [10,11]. Currently relational databases still dominate in telecommunications. The relational data model has been the basis for most implementations of IN databases. This has often led to a very complex front-end architecture of the database management system. Primarily due to this complexity object-orientation has recently gained growing interest. The theme has been to taken an object-oriented architecture and to choose an object model that supports telecommunications applications.

There are a couple of solutions both from commercial vendors and from research groups proposed for the database needs of IN:

— AT&T has developed an architecture called *data cycle* based on special purpose [12].

— Norwegian Telecom Research develops a relational database system that is a shared-nothing parallel system using an ATM network [13,14].

— Deutsche Telecom has an IN database architecture based on the OBJbase commercial object-oriented database management system.

— Nokia Telecommunications has a database architecture that is embedded into their Service Control Point (SCP) software [15].

— Hewlett Packard develops a scaleable main-memory database management system for high-performance applications that uses the relational model [16].

Object-oriented databases. The object-oriented database research had two different starting points. The first one was the need of persistent objects in object-oriented software. The second one was the need of a data model for databases that can more easily support complex applications than a pure relational model can. Now the consequences from the different starting points have almost vanished which can be seen in the first object-oriented standards and recommendations based on the *object-oriented database system manifesto* [17]. The most notable object models include the ODMG-93 object database model [18] and the OMG object

model with OMG Persistent Object Service [19-21]. A comprehensive survey of the current object-oriented databases is given in [22].

Real-time database systems. Real-time database systems (RT-DBS) support both real-time transactions and data integrity and consistency. The essential feature of RT-DBS is the timing constraint: The transaction must or should be completed before the given deadline. In real-time database systems the transactions are classified into *soft*, *firm*, and *hard* transactions according to the consequences of missing the deadline. Soft transactions are finished at a lower priority after the deadline has expired. Firm transactions are terminated when the deadline is reached. Hard deadlines must always be met. Detailed description of real-time database systems can be found in [23-27].

The main difficulty in an RT-DBS is to combine concurrency control and real-time scheduling. Several real-time concurrency control and scheduling algorithms have been proposed in the literature [28-36]. An excellent survey is given in [36]. In real-time database systems the relational model has been the most popular data model. However, object orientation has recently received increasing attention in the real-time community. In real-time databases an object-oriented approach is still quite undeveloped. One of the first attempts to combine object orientation and real time was done in the University of Rhode Island [37,38].

3. Design requirements from telecommunication architectures

In telecommunications there are currently several architectures that must be taken into account when a database architecture to be used in telecommunications is to be designed. In addition to the current ITU-T Recommendations on Intelligent Networks (Q.1200-series), related architectures such as Telecommunications Management Network (M.3000-series), Telecommunications Information Networking Architecture (TINA-C), and the third generation mobile systems (UMTS and FPLMTS/IMT-2000) create design constraints to database architectures.

3.1 Intelligent Network

Intelligent Network (IN) is a design platform for real-time telecommunication service design, implementation, and usage. The IN reference model has four planes of which the distributed functional plane is the most important one in database design. It defines the functional entities that constitute the IN functional architecture. The functional entities and their relationships that are of interest in the database design are depicted in Figure 1. The Service Control Function (SCF) is responsible of IN service control. The Service Data Function (SDF) is responsible of IN data management. The Service Management Function (SMF) is responsible of IN management functions. All the elements are related to each other. The IN Capability Set 1 (CS-1) gives functional definitions and reference architectures of all the entities. However, except for SCF the reference architectures are very broad and vague. [4]

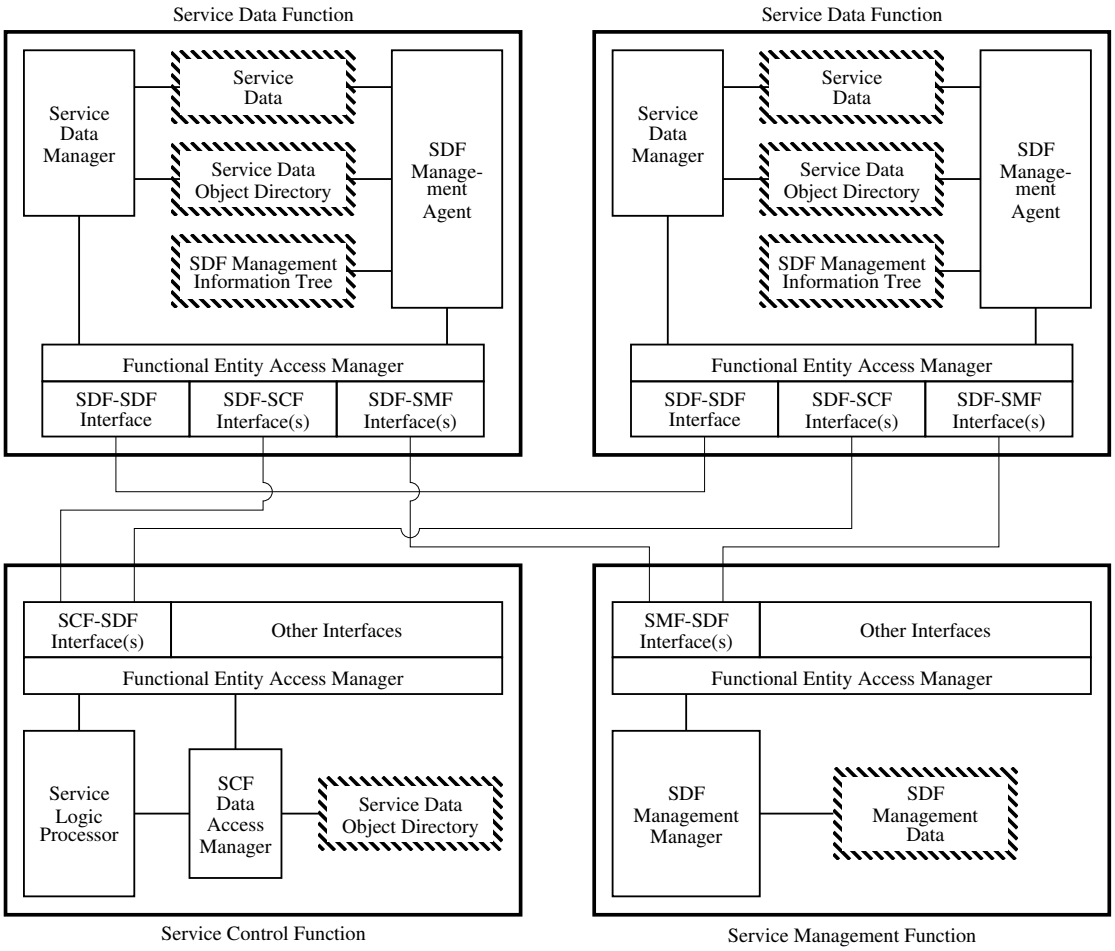

Figure 1: Elements of database servers and clients in the IN functional architecture

The Service Data Function is responsible for real-time access and maintenance of global network related data. The SDF contains and manages the data that is related to service logic processing programs and accessed in the execution of the programs. Therefore, the SDF is a data server for SCFs.

The SCF architecture model has two data managers: Resource Manager and SCF Data Access Manager. The Resource Manager controls the local SCF resources and provides access to network resources. Therefore, it sets requirements to the SCF Data Access Manager but not directly to an IN database architecture. The SCF Data Access Manager provides the functionality needed in storing, managing, and accessing shared and persistent information locally in the SCF. It also provides the functionality needed to access remote information in SDFs. This implies that the SCF Data Access Manager is both a client of one or several SDFs and a database manager for SCF local data. In the SCF two structures containing SCF data have been identified: the Service Data Object Directory and the IN Network-wide Resource Data. The Service Data Object Directory provides the means to locate Service Data Objects from the local SCF and from SDFs.

The CS-1 SDF model has three managers: Functional Entity Access Manager, SDF Data Manager, and Exclusive Control Manager. The interesting managers are the SDF Data Manager and the Exclusive Control Manager. The Functional Entity Access Manager only provides an interface to other functional entities. It is defined that the SDF Data Manager

provides the functionality needed for storing, managing, and accessing information in the SDF. Thus it offers regular data management services, not necessarily database management services. It is also stated that the SDF Data Manager may handle a database accessing language if the data is stored in a database. Therefore, an actual implementation of SDF can be anything from a single file operating manager to a complex database management system. The Exclusive Control Manager provides the functionality needed to provide data control. However, the Exclusive Control Manager is left for further study in CS-1. The recommendation gives examples of possible control manager use, such as locking control and integrity constraints.

The following five basic types of database operations needed in the IN CS-1 were identified in [1]:

1. *Retrieval of structured objects from persistent subscriber data objects*. Some of the retrievals trigger a later update.
2. *User management actions that modify persistent subscriber data objects*. These actions must be protected against unauthorized use. Some updates may also have to be verified against other criteria.
3. *Verification of Personal Identification Number*. For security reasons the PIN verification must be done in the database system when services are globalized.
4. *Writing sequential log records*.
5. *Mass calling and Televoting*. They need high-volume small updates that must be serialized. A special aspect of these updates is that they can be done in large blocks.

The most common operations are retrieving structured objects and writing sequential log records. Security operations, such as PIN verification, are also quite common. Management updates are not as common as the previous classes. Finally, mass calling and televoting occur only on special occasions. They may also need SCFs that are specifically tailored for that use.

The list above and our short analysis imply that most IN CS-1 database operations are reads or simple writes (for example writing sequential logs). Thus the CS-1 implies that a good database architecture is tailored for fast data access in the cost of random object updates. It should also be noted that CS-1 defines that data distribution is hidden from SCF. This implies that every database functional entity in the IN functional plane is a member of a large distributed database. Detailed description of database access in Intelligent Networks can be found in [39].

3.2 Telecommunications Management Network

Telecommunications Management Network (TMN) [40] is an architecture for supporting management activities associated with telecommunication networks. A TMN provides management functions for telecommunication networks and services and offers communications between itself and the managed network. Conceptually TMN is a separate network that communicates with the managed network to control its operations.

In TMN there are three basic aspects of the architecture that can be dealt with separately:

1. TMN functional architecture that describes the functionality within TMN.
2. TMN information architecture that describes the information and object modeling techniques used in TMN.
3. TMN physical architecture that describes the TMN network design on physical level.

Of the three TMN architectures the TMN information architecture is the most interesting for the database design. The TMN information network is based on OSI X.700 Management

model [41]. The fundamental idea is the separation of the information knowledge base from the functional modules that perform management actions.

The OSI Management is based on interactions between management applications that can take the roles of *manager* and *agent*. Agents and managers interact with each other in terms of *management operations* and *notifications*. Management activities manipulate *Managed Objects* (MOs) that are abstractions of the managed objects in the real world. An agent manages MOs within its local system environment. It performs management operations on MOs that a manager has requested. An agent may also forward notifications from MOs to a manager. The agent maintains a part of the *Management Information Tree* (MIT) that is a special kind of a distributed hierarchical object database.

Database requirements in TMN are somewhat different from those in IN. IN is based on fast enough execution of services while TMN is based on reliable management operations. In IN database architecture the most important aspects are fault tolerance and ability to answer queries before their predefined deadlines. In TMN the overall throughput is usually considered to be more important than finishing a certain request before its deadline.

The TMN management will be especially important to the service management IN. Currently the IN Service Management Function is still under further study in ITU-T. It is highly probable that the IN service management will be based on TMN. Therefore, the principles of OSI Management and TMN require that the database system contains the functionality of an OSI management agent.

3.3 Telecommunications Information Networking Architecture

The Telecommunications Information Networking Architecture Consortium (TINA-C) is a worldwide consortium aiming to define and validate an open architecture for future telecommunications services. The architecture is based on distributed computing, object orientation, and other standards and recommendations in telecommunications and distributed processing fields, especially Open Distributed Processing (ODP), Intelligent Networks (IN), Telecommunication Management Networks (TMN), Asynchronous Transfer Mode (ATM), and Common Object Request Broker Architecture (CORBA) [42].

TINA architecture defines a common background for future telecommunications networks. Distributed data management in TINA is based on ODP [8,9]and CORBA [20,43]. Both ODP and CORBA are object-oriented models and related to each other. ODP defines five viewpoints that describe an ODP system: enterprise viewpoint, information viewpoint, computational viewpoint, engineering viewpoint, and technology viewpoint. As presented in [44] the ODP viewpoints may be used to separate the distributed data management issues and to identify the TINA concerns at each level. CORBA defines an object information highway that can be used to deliver ODP and TINA objects. The implications are that the database system must support the object invocation mechanisms of CORBA and communication through an ODP channel.

So far three different IN/TINA coexistence possibilities have been identified [45]: 1) Interworking of IN and TINA service architectures: a TINA compliant network can request services/management/etc. from an IN network and vice versa; 2) Mapping of IN services on TINA architecture, allowing a TINA compliant network to “run” IN services; and 3) Evolution of IN towards TINA architecture; subsequent capability sets of IN converge towards TINA. Thus we assume that in the future IN and TINA must either be compatible or IN must be an

integral part of TINA. In either way, the important aspects of TINA must also be available in an IN database architecture.

3.4 Third generation mobile networks

Currently the third generation mobile networks are under development. They rely on IN services for multimedia architecture. Such systems are the Future Public Land Mobile Telecommunication Systems[2] (FPLMTS) [46] that is under standardization in ITU-R and ITU-T, and Universal Mobile Telecommunication System (UMTS) [47] that is under standardization in ETSI. UMTS and FPLMTS are closely aligned.

Universal Mobile Telecommunication System (UMTS) provides wireless communication services to users on the move. The services will include high speed data services and video services like mobile videophone [48]. UMTS will use IN services for achieving the result.

The mobility specific control functions that UMTS needs from IN should be available in CS-3. The IN CS-2 will already support some areas required for UMTS but full control should be available in CS-3. While IN CS-1 services use mostly read operations on IN databases, UMTS functions also frequently update the database. Access delay to the data is of crucial importance and the number of accesses per user will be very high. The data required to support a certain user might not be stored near the location of where it is used [48]. Future mobile systems, such as UMTS, set following requirements [49]:

— *Traffic projections*: 7 to 8 K real-time query transactions/second/regional calling area, 7 to 8K real-time updates/second/regional calling area.

— *Database system performance requirements*: maximum 150 ms response time for more than 98 percent of queries, response times on the order of seconds for updates, and high availability (7 days/week and 24 hours/day availability, at most 3 minutes/year downtime).

The main functions of UMTS database services are [48]:

° the basic functions of providing read and write access to the data,

° to locate the requested data within the database,

° to move data to locations close to the user when the user has roamed to provide fast access to the data where it is frequently being used,

° to provide mechanisms that ensure the integrity and reliability of the data even when it is copied between locations and possibly duplicated, and

° to ensure that data is only accessed by entities that have been authorized to do so.

3.5 Summary of requirements

If a database architecture was designed only for CS-1 service logic programs, a good design choice would be a distributed relational database that is fault tolerant and that is mostly tailored for reads and writing sequential log records. Such an architecture can answer the current SDF requirements that are listed in ITU-T recommendations. However, the future trends give the architecture new functional requirements. There are basically three future areas: object orientation, network management, and mobility.

[2] FPLMTS is also known as IMT-2000 (*International Mobile Telecommunications 2000*).

Object orientation is the current trend in most telecommunication architectures, such as TINA-C and TMN. Either when IN architecture and services will be used with TMN and TINA-C, or when the architectures are combined into a single telecommunications architecture, the IN architecture must follow the object oriented standards.

A possible IN long term database architecture does not have to be object oriented. However, in order it to be compatible with the rest of the IN network the internal model must be flexible enough to support an object oriented front end. In the long run it is better to build a full object-oriented database that can be easily modified to future IN needs.

Network management brings new requirements to the IN database architecture. Currently CS-1 does not define management aspects. They should be available in CS-2, and then it is most probable that the management will be based on TMN management model. TMN is based on X.700 OSI management model that is object oriented. This implies object orientation also in IN management.

The requirements that management functions set to a database management system differ from IN requirements. The IN database services are mostly based on fast data access and writing sequential log records. TMN needs reliable operations and data integrity. The requirements conflict with each other.

Finally, mobility aspects bring new requirements. The assumption that most database operations are reads is no longer true. The mobility support implies that the database must maintain location information. This results IN databases to have several magnitudes more writes than today. If the mobile networks used only existing mobile network technology then the mobility aspects could be isolated into separate tailored database nodes. However, the third generation mobile systems, such as UMTS, will use IN provided services.

Our conclusion is that currently it is not possible to build a database architecture that fully answers all three request groups: IN service execution requests, TMN management requests, and third generation mobile system requests. However, it is already possible to build a database architecture that fulfills current and future IN requirements and mobility aspects. Then a good alternative for a global database architecture is to use both IN database architecture and TMN database architecture.

4. Darfin database architecture

In the Darfin-project we have designed a database management system, called *Darfin DataBase Management System* (Darfin DBMS) to be used in telecommunications. The architecture has three primary levels: global architecture, node architecture, and object-oriented database management system.

4.1 Global architecture

The Darfin DBMS architecture is a *real-time object-oriented database management system architecture*. It consists of a set of autonomous database nodes that interact with each other. Each node may communicate with one or more applications, and an application may communicate with one or more database nodes. Each node is replicated into a mirror node (Figure 2).

An application communicates with the database node through a run-time database interface module that is linked to the application. The module takes care of all communication between the node and the application. It also supports fault-tolerance by making the recovery process details invisible to applications.

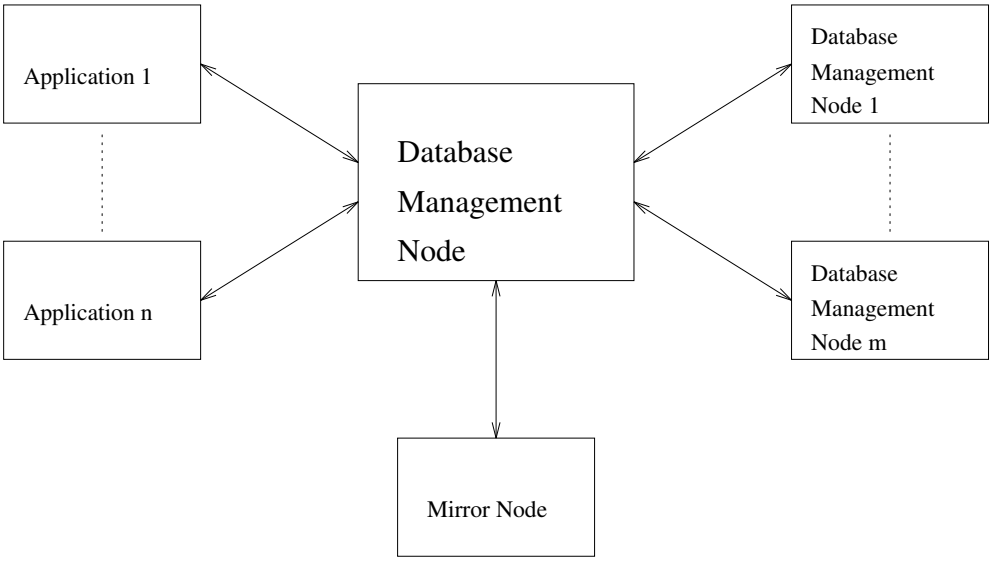

Figure 2: Overview of the Darfin database architecture

Applications, databases, and mirror nodes may be geographically distributed. In our architecture a network connection is equal to a disk connection or to a terminal connection. This is equal to UNIX network stream equality to other UNIX data streams.

Although we support distribution, we do not call our architecture a distributed database architecture. Our architecture is based on the assumption that most operations are local to an autonomous node. Usually a telecommunications application refers to data in its primary database node or knows where data resides. The distributed operations are mostly updates and usually affect a few objects. Currently we have not planned to support complex distributed operations. Instead, we support location transparency. The data location transparency method follows the requirements of the data access in Intelligent Networks. A requesting application need not know where data resides. Each database node can forward a request to other nodes.

In complex distributed operations the application itself is responsible of being the leader. The autonomous databases understand two phase commit that is directed from the application. However, will implement true two-phase commit where any of the database nodes may be the leader if really needed in telecommunication applications. We have also plans to X.860 recommendations for distributed operations.

Every autonomous database node has a mirror node that is used in fault-tolerance and recovery. When a node fails, its mirror node takes control of database management during the recovery process. Local database managers in autonomous databases decide what data is mirrored to achieve quick recovery and what data is protected against failures through traditional logging. Mirroring everything is not a good alternative as data quantity in telecommunications databases may grow to terabytes in size. The costs of complete mirroring will overdo the benefits. In principle the mirroring decision leads to a situation where not all data is continuously available. However, with a good mirroring policy it is possible to build a data schema that lets important data to be continuously available while other data goes through a slower recovery process.

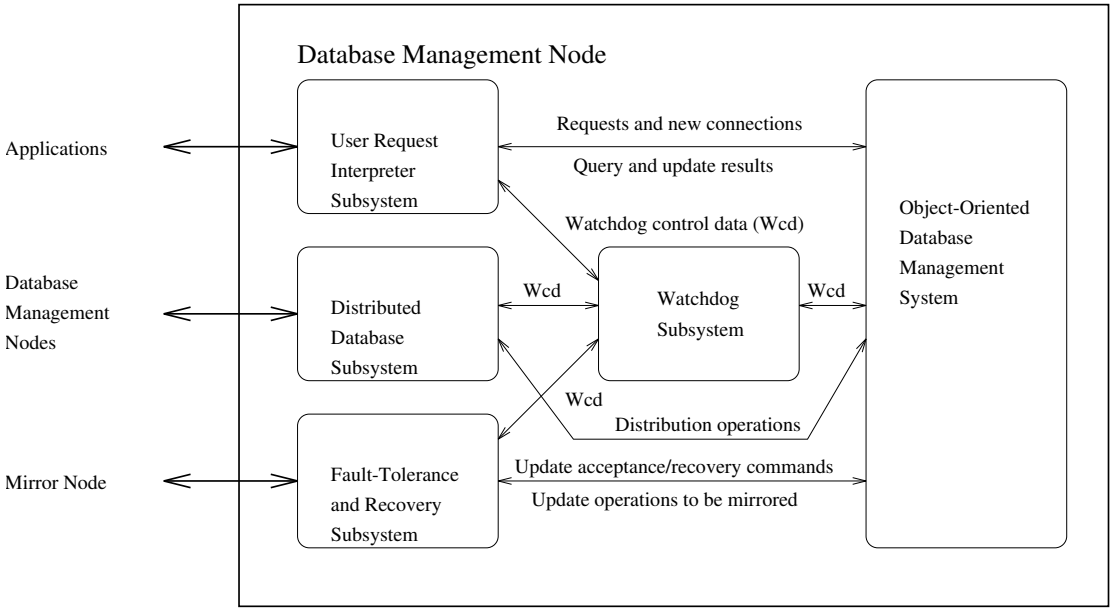

Figure 3: Darfin database management node

4.2 Node architecture

A Darfin DBMS node consists of a set of subsystems that communicate with each other. The subsystems are: User Request Interpreter Subsystem, Distributed Database Subsystem, Fault-Tolerance and Recovery Subsystem, Watchdog Subsystem, and Object-Oriented Database Management System (Figure 3). In addition, the node architecture includes the run-time database interface module that is the main communication interface between applications and Interpreter Subsystems and Watchdog Subsystems. The application does not see the Watchdog Subsystem. It is only used internally by the run-time database interface module to support application-level fault tolerance (Figure 4).

User Request Interpreter Subsystem. A database management system for telecommunication use must support several interfaces to the database. Currently IN CS-1 defines Intelligent Network Application Protocol (INAP) interface between functional elements. It includes an interface definition to database services. The next IN database interface may be based on X.500 OSI Directory recommendation. [50] Other interfaces that used be supported include the X.700 OSI Management interface and the TINA object model interface. The User Request Interpreter Subsystem translates various interfaces into a common connection language used by the database management system. Darfin DBMS architecture supports several simultaneous Interpreters, each of which is tailored to interpret one type of requests. An interpreter may need to do a model translation between an application and the database management system. For instance, X.700 supports dynamic attribute packages where the set of attributes may vary depending on the object state. However, our object model (as well as most object models do) supports only static attribute sets.

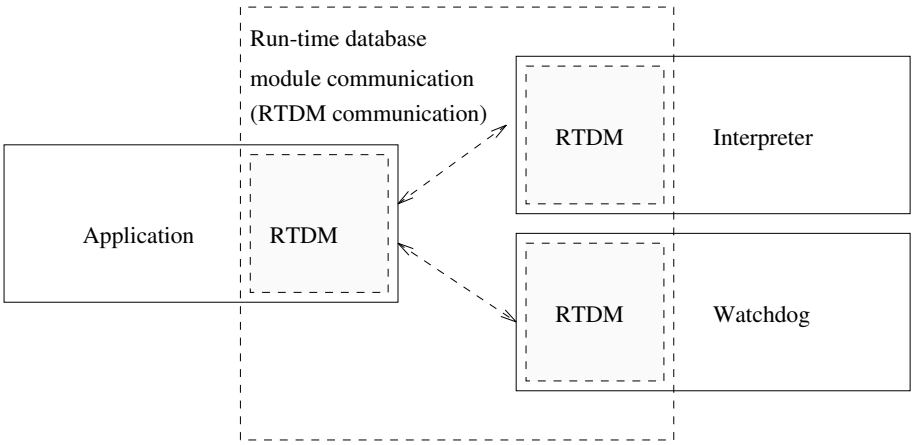

Figure 4: Run-time database interface to a database node

Distributed Database Subsystem. A Darfin DBMS node may either be used alone or in cooperation with the other autonomous database nodes. The database cooperation management in the node is left to the Distributed Database Subsystem. The use of the subsystem frees the actual database management system from distribution control. When the database management system receives a request that it can not resolve, it forwards it to the Distributed Database Subsystem. Similarly, when the Distributed Database Subsystem receives a request that belongs to this node, it forwards it to the database management system. Thus, the Distributed Database Subsystem is the only subsystem that is aware of other autonomous databases.

The Distributed Database Subsystem forwards a distributed query to a set of nodes. Each node in the set tries to resolve the query. If the query is a read, each node sends a reply (value or indication of not found) to the requesting node. If all replies are 'not found,' then the reply 'not found' is send to the application. Now the application is responsible to start a 'location search' procedure. This is clearly a restriction to the location transparency. However, we have done the restriction on purpose. If the query can not be completed in one step of 'distribution transparency,' then it is highly probable that the deadline can not be met. In the case of an update procedure, the two phase commit procedure is always initiated. Each node returns update status (commit[3] or non-commit) and the application decides if the commits are accepted. The operation is automatically accepted by the run-time database interface module if all results are commits.

Fault-Tolerance and Recovery Subsystem. The subsystem controls communication between the main database node and the mirror node. It also cooperates with the Watchdog Subsystem to support fault-tolerance.

Watchdog Subsystem. The subsystem watches over the other systems in the database management node. It communicates with the other subsystems and with the applications to state the current node status. Upon a failure it restores the node. The Watchdog Subsystem notices when a system is down and starts a recovery process. Different recovery processes go as follows:

° *Object-Oriented Database Management System fails*. The Fault-Tolerance and Recovery Subsystem informs the mirror node to take control. When acknowledged by the mirror node, the subsystem informs the other subsystems that they should forward requests to the

[3] If a node receives an update of a data item it does not have, the reply is commit.

mirror node. After that it informs the database node to start recovery process and controls data traffic between the mirror node and the recovering database node. In addition, the Watchdog Subsystem watches over the recovery process in case the Fault-Tolerance and Recovery Subsystem fails.

- *Interpreter Subsystem fails*. The Watchdog Subsystem restores the failed Interpreter Subsystem but connections are lost for a moment. If the restoring process is too slow for the applications, then we will use mirror Interpreter processes that take control. The run-time database interface module in applications is aware of the changes in the node control and makes the connection invisible to the actual application. It asks the Watchdog Subsystem when a suitable interpreter is available and possibly reconnects to a mirror Interpreter Subsystem. The database management system accepts connections from any Interpreters so it needs no special procedure for an Interpreter failure. In case of an unexpected connection loss it aborts the running transaction from the failed Interpreter and waits for a new one from any Interpreter Subsystem.
- *Distributed Database Subsystem or Fault-Tolerance and Recovery Subsystem fails*. The Watchdog Subsystem restores the failed subsystem. If the restoring process turns out to be too slow, then we will support a mirror subsystem that the Watchdog starts. Connections to the mirror node or the Distributed Database Subsystems in other nodes are restored after the recovery process.
- *Watchdog Subsystem fails*. The Fault-Tolerance and Recovery Subsystem restores the Watchdog Subsystem. The Watchdog Subsystem is very simple in structure and so it will not take long to restore it. A mirrored Watchdog Subsystem is not necessary.
- *Several subsystems fail or the whole node fails*. In most situations the recovery procedure is similar to situations when one subsystem fails. However, in situations when the Watchdog Subsystem and the Interpreter fail, or when the whole node is down, the procedure is different. If the run-time database interface module notices that it cannot connect to an Interpreter Subsystem, it queries the Watchdog Subsystem of the system state. If the Watchdog Subsystem does not reply, then the module expects the node to be down and forwards the queries to the mirror node. The mirror node takes full control of the data management and also accepts queries from other nodes.

Object-Oriented Database Management System. This is the main component of a Darfin DBMS node. It maintains databases, real-time constraints, integrity, and concurrency control (see Section 4.3 below).

4.3 Object-Oriented Database Management System

The Object-Oriented Database Management System (OO-DBMS) consists of a set of database processes that use database services to resolve requests from other subsystems, and a set of manager services that implement database functionality. The functional structure of Darfin OO-DMMS is shown in Figure 5.

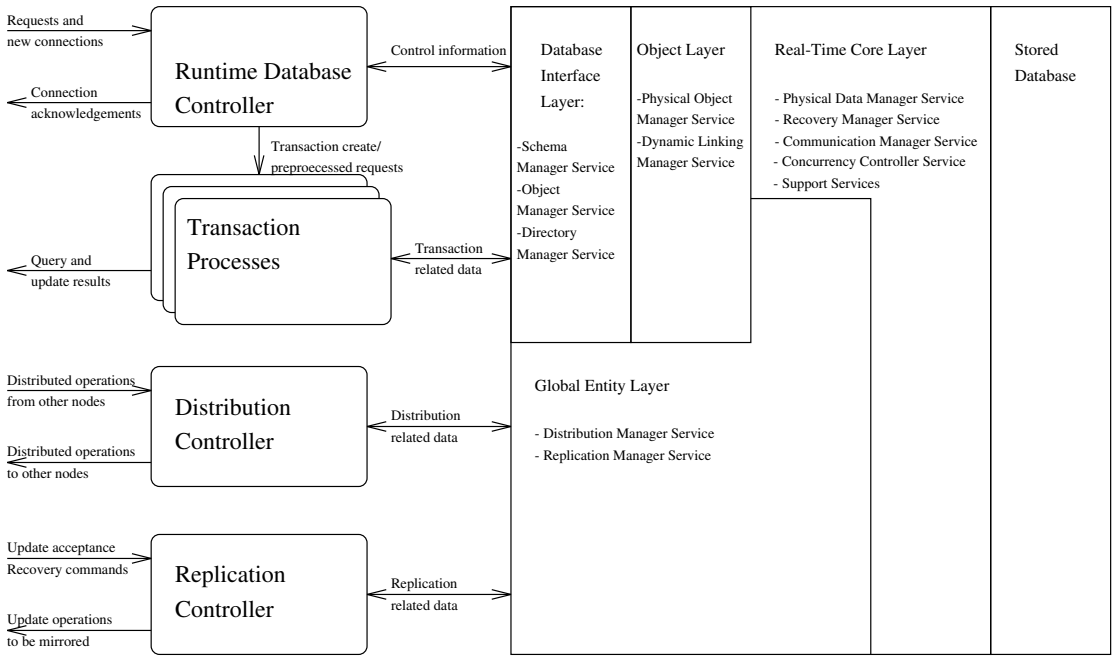

Figure 5: Processes and database services in Darfin OO-DBMS subsystem

4.3.1 Database manager services and service layers

The manager services are divided into four layers based on their nature. A service in a layer uses services in the next lower layer. Some services in the lowest layer are used by all other layers. The layers are *Database Interface Layer* that is the visible layer to common database processes, *Global Entity Layer* that is the visible interface layer to distribution and replication processing in the OO-DBMS, *Object Layer* that implements object functionality, *and Real-Time Core Layer* that implements low level database functionality such as real-time and physical data control.

Database Interface Layer. This layer is the interface to the database processes of the database services. It offers Schema information services for the use of metadata and for maintenance, of object information services of object oriented interface, and for directory services of directory maintenance.

The database architecture should be flexible due to different kinds of needs present in telecommunications. We believe that the required functionality is easiest to implement with an object oriented approach. An open question is if a full object-oriented architecture is needed. Currently we believe that it is better to use a reduced set of object features than to use a fully object-oriented database when we are developing a real-time database to be used in telecommunications.

We have chosen the ODMG-93 object database architecture as the basis of our object model. It follows the general object-oriented database requirements that are defined in [17]. A detailed description of the model is given in [18]. An analysis and comparison of the ODMG-93 model to other telecommunications related object models can be found in [51].

We have expanded the object model to support real-time transactions. The new object model is a superset of the ODMG-93 model. It is possible to use the object model without the real-time additions and thus, in principle, it is also possible to replace the object database management system with another system that follows ODMG-93. However, most future telecommunications applications will use the real-time features of the model.

The principal item in Darfin real-time object model is an object. It can be either a regular object or a real-time object. Objects have identity and they belong to a known type. A real-time object has predefined attributes for isolation level and access type. Different isolation levels allow transactions to reduce serializability when the objects accessed allow it. Access type restrictions allow transactions to be optimized for certain types of objects.

Real-time objects are referenced by real-time transactions. A real-time transaction [36] that has an explicit deadline is really needed in telecommunications. We want to support two kinds of real-time transactions: 1) *soft transactions* that may continue their execution after deadline but with a reduced lower priority and 2) *firm transactions* that are terminated when their deadline expires. We do not believe that *hard transactions* are needed. Detailed discussion can be found in [52].

A real-time transaction is also a real-time object, either persistent or transient. It has a deadline and transaction type (soft, firm, hard). A transaction may suggest its isolation level to other transactions. In case of isolation conflict the more stringent isolation level is chosen. In this way transactions that want serializability are allowed to execute. In addition, isolation level between objects is supported. Again, the most stringent isolation level is chosen. For instance, a transaction that runs at a weak isolation level does not care if an object is accessed by another transaction. However, it will not get access to the object if a transaction that wants more stringent isolation is accessing it. Similarly, if two low isolation level transactions want to access an object that has a more stringent isolation level, the latter transaction must wait the first transaction to release the object.

Real-time objects consist of real-time attributes and real-time operations. A real-time attribute defines attribute access time that can be used to support different serializability levels. A real-time operation defines an estimate of the operation execution time that can be used for transaction scheduling. Details in our real-time object model are still evolving

Object Layer. The layer offers the actual object implementation services: Physical Object Manager Service and Dynamic Linking Manager Service. The Database Interface Layer managers use the services to implement the visible object model. On this layer the objects are all alike and the actual object model is no longer visible.

The Physical Object Manager Service is the core manager service for objects and object operations. It offers services for object creation, object deletion, object access, object attribute access, and object operation access. It uses the global entity layer when a replicated operation occurs or when it cannot resolve an object request on the local database. It uses the real-time core layer services to implement object functionality. The physical object manager supports the object oriented database system manifesto as defined in [17].

The Dynamic Linking Manager Service offers services to dynamically link new object operations into the object database. It uses the services of the Real-Time Core Layer to implement object operations.

Global Entity Layer. The layer offers services to other layers and to special controller processes in order to support distribution and replication in the OO-DBMS. It consists of Distribution Manager Service and Replication Manager Service.

The Distribution Manager Service is used by the Distribution Controller process when it needs database services. It is also used by other services when an unresolved database operation occurs. An unresolved database operation triggers the service that then informs the Distribution Controller to take control of the operations.

The Replication Manager Service is used by the Replication Controller process when it needs database services. It is also used by other database services when a database update operation that should be mirrored occurs.

Real-Time Core Layer. The layer implements the core operations of real-time database management system. This lowest layer consists of Physical Data Manager Service, Recovery Manger Service, Communication Manager Service, Concurrency Controller Service, and Support Services. Of the services the Communication Manager Service and Support Services are used on all other layers.

The Physical Data Manager service is the only interface to the physical data in the database. It is the only manager that has knowledge of physical data locations in the database. It offers services for data storage, retrieval, and access estimates.

Data in Darfin database may belong to one of three heat groups: hot, warm, or cold data. Hot data is always in main memory. All updates are done in main memory and a transaction log is maintained to keep the database in a consistent state. Warm data is always in main memory when read. However, the update and write operations are done directly to a disk. Cold data is always accessed from disks or disk buffers. Thus, we use a hybrid data management method that is a combination of a main memory database and a disk based database.

The Recovery Manager Service is the first service to gain control when the Fault-Tolerance and Recovery Subsystem has started OO-DBMS recovery operations. It restarts database processes and offers services to the Replication Manager Service o return data integrity and lost data. The Communication Manager Service offers services to the other managers to maintain communication channels to the other subsystems. The Concurrency Controller Service allows transactions to run in parallel. It offers control services to other database managers and to the Real-Time Scheduler. It may also awake the Real-Time Scheduler when a conflict occurs.

In real-time applications concurrent transactions do not necessarily need serializability to produce a correct result. In many situations the result of a transaction can be regarded as correct as long as the result corresponds to the real-world situation presented in the database even if the transaction does not serialize with other transactions. Our concurrency controller supports different serialization and isolation levels as described in the real-time object model. An interesting correctness criteria, which we call τ-*serializability* [53], can be used to reduce the number of serialization conflicts. In τ-*serializability* a transaction is allowed to read old data as long as the update is not older than a predefined time.

4.3.2 Database processes

In the Darfin database architecture the OO-DBMS processes are either interacting or non-interacting. An interacting process receives requests from a database subsystem. A non-interacting process is a control process for the interacting processes. In Darfin DBMS the

interacting database processes are Runtime Database Controller, Transaction Processes, Distribution Controller, and Replication Controller. The non-interacting process is the Real-Time Scheduler.

The Runtime Database Controller is the main controller of the OO-DBMS. It accepts new connections from Interpreter Subsystems, creates new Transaction Processes, and forwards request information to appropriate transaction processes. The Transaction Processes are the main elements of transaction processing in the OO-DBMS. They process the requests that the Interpreter Subsystems have sent and send results back.

The Distribution Controller is a special database process that maintains the connection to the Distributed Database Subsystem. It hides distribution management in the OO-DBMS from the other database processes. It is triggered when transaction processes cause a database operation that belongs to a different database node. The Replication Controller is a special database process that maintains the connection to the Fault-Tolerance and Recovery Subsystem. It receives and processes recovery commands from the subsystem, and forwards update commands to it. It is triggered when other processes cause a database operation that should be mirrored.

Real-Time Scheduler schedules the other database processes (Figure 6). Every database process has a priority. The Real-Time Scheduler process has the highest possible priority. It executes either periodically or when a concurrency control occurs. The second highest priority belongs to the Runtime Database Controller. It executes only when Requests or new connections arrive from the Interpreter Subsystems. The Distribution Controller and Replication Controller processes have the third highest priority. They execute either when the appropriate subsystem sends a request or when a database operation triggers them. Finally, the transaction processes have the lowest priorities. The transaction priorities depend on the chosen real-time scheduling policy.

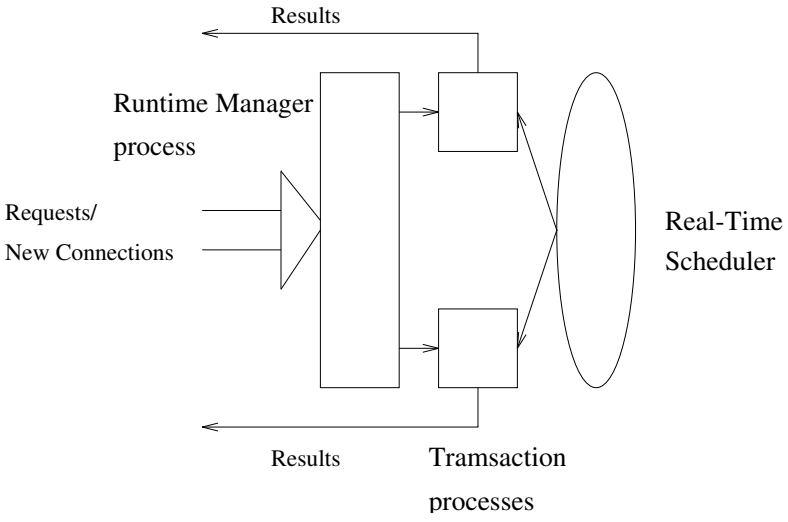

Figure 6: Process control in database management prototype

Both the Concurrency Controller and the Real-Time Scheduler may be changed to support different concurrency control and scheduling policies. Currently we use a pessimistic concurrency control method that also supports different isolation levels. The chosen scheduling

policy is the Earliest Deadline First [29]. The database architecture allows more sophisticated scheduling policies than the pure EDF.

4.4 Implementation status

In the Darfin project and in its follower RODAIN[4] project we have primarily concentrated on specifications of the architecture. Currently we are planing an implementation of the architecture with our industry partners[5]. So far our experiences from the implementation procedure consist of a sample database management system prototype and a detailed query of possible real-time operating systems upon which we could implement the final product.

The objectives of the prototype developed in the Darfin project were to find out:

1. How a telecommunications application interacts with a real-time object-based database system?
2. How object orientation and real-time constraints are related to each other?
3. How much the implementation platform affects the design and implementation process?

Starting from the requirements we built a test application, a sample interpreter, and a very simple database management system. In the prototype we concentrated on the most important aspects from the application's point of view. The test application designed and implemented is an X.700 CMIS agent that uses the XI protocol developed and used in Telecom Finland. The XI protocol is an implementation of the CMIP protocol and the X.700 information model.

The test application consists of the following four layers (see Figure 7): User Interface Layer, XI Generic Layer, XI Server Specific Layer, and Database Interface Layer. The User Interface Layer is a character based XI message sender and receiver. The XI Generic Layer consists of the XI interface. It understands a large subset of X.700 OSI Management messages. The XI Server Specific Layer converts the used application model to the real network resource. In the test application it also converts XI operations to object database operations. The Database Interface Layer hides the communication channel between the application and the prototype database management system. It communicates with a User Interpreter Subsystem in a database management node over the network. The application and the database management system may be geographically distributed.

Connecting the test application to the prototype of our database architecture turned out to be straightforward. The connection was based on a subset of low level calls in the ODMG-93 object model [18]. The experiences we got suggest that the application designer can first define the interface and later ask the database maintenance staff to implement a suitable Interpreter Subsystem for the application. Most Interpreters are straightforward to build from predefined modules.

Due to our desire of having a fast prototype implementation we put our main focus to the real-time constraints of the application instead of various aspects in object orientation. During the design phase of the prototype system this choice turned out to be fruitful. In its simplest form a real-time database model is rather straightforward and easy to implement. On the other hand, an object-oriented database model is very complex with late binding, object methods, and inheritance.

[4] RODAIN is the acronym of the project name *Real-Time Object-Oriented Database Architecture for Intelligent Networks*.

[5] The industrial partners in RODIN are Nokia Telecommunications, Solid Information Technology, Systems Software Partners, and Telecom Finland.

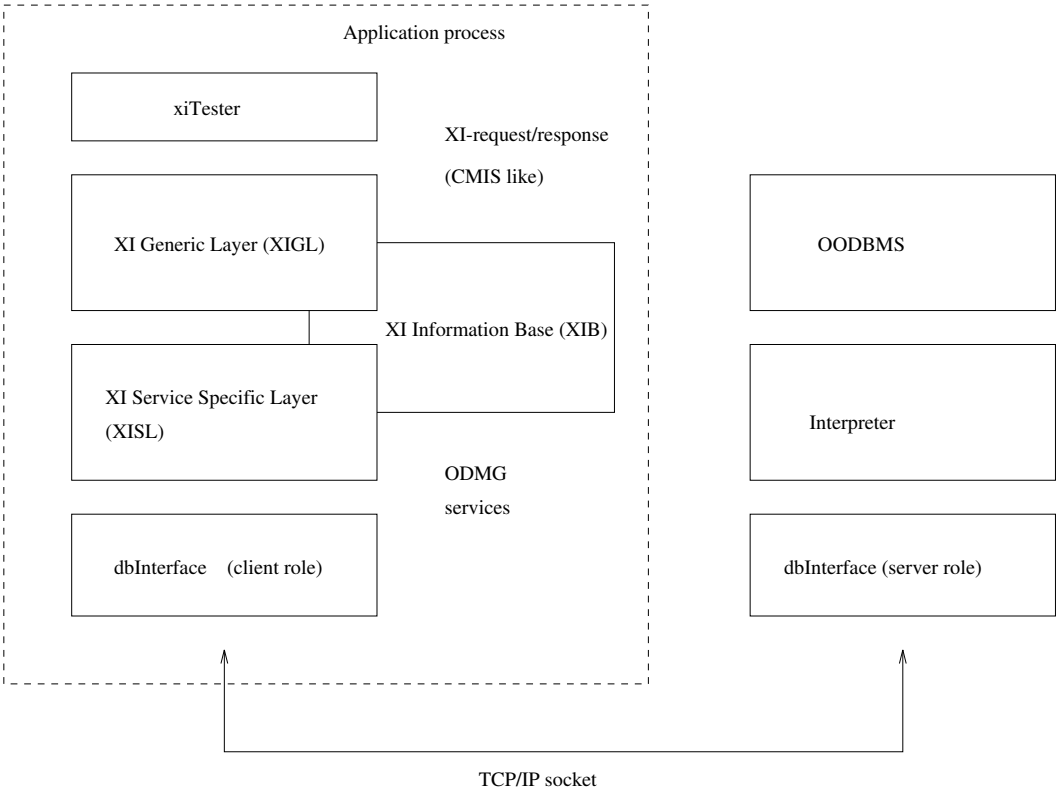

Figure 7: Layout of the test application

We first implemented a simple prototype of a real-time database management system with the earliest deadline first scheduling policy. Later we inserted necessary features of object orientation. The features we inserted were object identity, object directories, naming of objects, lookup and fetch operations, and a simplified OQL. However, it should be noted that our prototype is not a fully object-oriented database management system. Although the Darfin real-time object model defines a full ODMG-93 compliant object model, we believe that an object-oriented model useful in telecommunications is a subset of the ODMG-93 object model. Some features such as late binding and multiple inheritance will create problems in estimating resource consumption.

The primary lesson we have learned is that the needs of telecommunication can be fulfilled when we progress from real-time towards object orientation. The object-oriented databases currently available — such as O_2 [54] and EXODUS [55] — are so complex that it will be hard to achieve the transaction times and rates needed in telecommunications.

5. Summary

We have introduced a real-time object-oriented database architecture that is designed to be used in telecommunications, particularly to provide the database service for the Intelligent Network. The basis of the architecture is the extensive requirements analysis carried out in the Rodain project. The needs of Intelligent Network (IN), Telecommunications Management Network, (TMN), Telecommunications Information Networking Architecture (TINA), and the third generation mobile system like UMTS and FPLMTS/IMT-2000 are quite different. However, the architecture introduced in this article is flexible and general enough to provide a common

basis for particular implementations that take into account the differences between the database usage in IN, TMN, TINA, UMTS, and FPLMTS.

The combination of real-time and object-orientation is still a new area in database research. The specific characteristics of telecommunications suggest that in short term useful results can more easily be achieved when a real-time database is expanded with object-oriented features than when a full object-oriented database is incorporated with real-time features.

Acknowledgments

This article is based on the work done the Darfin and RODAIN projects. The former was funded by Telecom Finland and the latter by the Finnish Technology Development Center (TEKES) together with Nokia Telecommunications, Solid Information Technology, System Software Partners and Telecom Finland. In addition, the preparation of the article was supported by the Research and Education Foundation of Telecom Finland.

The authors want to thank the Darfin and RODAIN teams, in particular Tapani Karttunen, Mika Rautila and Harri Töhönen from Telecom Finland and Petri Elovaara and Tiina Niklander from University of Helsinki. The comments and suggestions made by the Guest Editor Jari Veijalainen and the two anonymous referees greatly helped to improve the article.

References

1 **Raatikainen, K E E** *Information Aspects of Services and Service Features in Intelligent Network Capability Set 1*. Technical report C-1994-46, University of Helsinki, Department of Computer Science (1994).

2 **Taina, J** *Requirements Analysis for Database Services in Telecommunications*. Technical report C-1995-17, University of Helsinki, Department of Computer Science (1995)

3 **ITU-T** *Intelligent Network Distributed Functional Plane Architecture*. Recommendation Q.1204 (1993)

4 **ITU-T** *Distributed Functional Plane for Intelligent Network CS-1*. Recommendation Q.1214 (1993)

5 **Eurescom** *Fact-finding Study on Requirements on Databases for Telecom Services*. Technical report (1993)

6 **Eurescom** *TMN Management of Pan-European IN Based Freephone Service*. Technical report (1994)

7 **ITU-T** *Open Distributed Processing - Reference Model: Part 1 - Overview*. (Draft) Recommendation X.901 (1995)

8 **ITU-T** *Open Distributed Processing - Reference Model: Part 1 - Architecture*. Recommendation X.903 (1995)

9 **Kerboul, R; Pageot J-M; and Robin, V** 'Database Requirements for Intelligent Network: How to Customize Mechanisms to Implement Policies', *Proc.TINA'93 Workshop*, Volume 2 (1993) pp 35-46

10 **Jagadish, H V** 'Databases for Networks', *ACM SIGMOD Conference Proceedings,* ACM (1994) p 522

11 **Ahn, I** 'Database Issues in Telecommunications Network Management', *ACM SIGMOD Conference Proceedings,* ACM (1994) pp 37-43

12 **Bowen, T; Gopal, G; Herman, G; and Mansfield, W** 'A Scale Database Architecture for Network Services', *IEEE Communications Magazine* Vol 29 No 1 (January 1991) pp 52-59

13 **Hvasshovd, S-O and Torbjornsen, O** *A Software Architecture for a Continuously Available Shared-Nothing Parallel DBMS Based on ATM Technology*. Technical report, Sintef Delab (1993)

14 **Torbjornsen, O and Hvasshovd, S-O** 'An Abstract Machine Supporting Fault-Tolerance in a Parallel Database Server', *Proc. Int. Symp. on Fault Tolerant Systems* (1993)

15 **Lehtinen, P and Warsta ,A** 'Nokia's IN solution for fixed and cellular networks', *Intelligent Networks*, Harju, J; Karttunen, T and Martikainen, O (eds), Chapman & Hall (1995) pp 61-67

16 **Heytens, M; Listgarten, S; Neimat, M-A; and Wilkinson, K** *Smallbase: A Main-Memory DBMS for High-Performance Applications* (Release 3.1). Technical report, Hewlett Packard Labs (1994)

17 **Atkinson, M; Bancilhon, F; DeWitt, D; Dittrich, K; Maier, D; and Zdonik, S** 'The Object-Oriented Database System Manifesto', *Building an Object-Oriented Database System—the Story of O_2*, Bancilhon, F; Delobel, C and Kannellakis, P (eds), Morgan Kaufman Publishers (1992) pp 3-20

18 **Cattel, R G G (ed)** *Object Database Standard: ODMG-93, Release 1.1*. Morgan Kaufman Publishers (1994)

19 **OMG** *Object Management Architecture Guide — Revision 2.0*, 2nd ed. Object Management Group (1992)

20 **OMG** *The Common Object Request Broker: Architecture and Specification*. Object Management Group (1991)

21 **OMG** *Common Object Services Specification* - Chapter 6: Persistent Object Service Specification. Addendum to Common Object Services Specification, Object Management Group (1996)

22 **Zand, M; Collins, V; and Caviness, D** 'A Survey of Current Object-Oriented Databases', *Data Base Advances* Vol 26 No 1 (1995) pp 14-29

23 **Stankovic, J A and Ramamritham, K** *Advances in Real-Time Systems*. IEEE Computer Society Press (1993)

24 **Son, S H** *Advances in Real-Time Systems.* Prentice-Hall (1995)

25 **Son, S H** 'Real-Time Database Systems: Present and Future',. *Proc. Real-Time Computing Systems and Applications Workshop*, IEEE (1995) pp 50-52

26 **Graham, M H** 'Issues in Real-Time Data Management', *J. of Real-Time Systems* Vol 4 No 3 (1992) pp 185-202

27 **Ulusoy, Ö** 'Current Research in Real-Time Databases, *ACM SIGMOD Record* Vol 21 No 4 (1992) pp 16-21

28 **Kung, H T and Robinson, J T** 'On Optimistic Methods for Concurrency Control', *ACM Trans. on Database Systems* Vol 6 No 2 (1981) pp. 213-226

29 **Jensen, D; Locke, D and Tokuda, H** 'A Time-Driven Scheduling Model for Real-Time Operating Systems', *Proc. Real-Time Systems Symposium*, IEEE (1985) pp. 112-122

30 **Abbott, R and Garcia-Molina**, H 'Scheduling Real-time Transactions', *ACM SIGMOD Record* Vol 17 No 1 (1988) pp 71-81

31 **Huang, J; Stankovic, J A; Towsley, D and Ramamritham, K** 'Experimental Evaluation of Real-Time Transaction Processing' *Proc. Real-Time Systems Symposium*, IEEE (1989) pp. 144-153

32 **Sha, L; Rajkumar, R and Lehoczky, J** 'Priority Inheritance Protocols: An Approach to Real-Time Synchronization', *IEEE Trans. on Computers* Vol 39 No 9 (1990) pp 1175-1185

33 **Huang, J; Stankovic, J A; Ramamritham, K and Towsley, D** 'On Using Priority Inheritance in Real-Time Databases', *Proc. Real-Time Systems Symposium*, IEEE (1991) pp 210-221

34 **Haritsa, J R; Livny, M and Carey, M J** 'Earliest Deadline Scheduling for Real-Time Database Systems', *Proc. Real-Time Systems Symposium*, IEEE (1991) pp 232-242

35 **Sha, L; Rajkumar, R; Son, S H and Chang, C-H '**A Real-Time Locking Protocol', *IEEE Trans. on Computers* Vol 40 No 7 (1991) pp 793-800

36 **Yu, P S; Wu; K-L; Lin; K-J and Son, S H** 'On Real-Time Databases: Concurrency Control and Scheduling', *Proceedings of the IEEE* Vol 82 No 1 (1994) pp 140-157

37 **Cingiser DiPippo, L B and Wolfe, V F** 'Object-based Semantic Real-time Concurrency Control', *Proc. Real-Time Systems Symposium*, IEEE (1993) pp. 87-96

38 **Prichard, J J; Cingiser DiPippo, L B; Peckham, J and Wolfe, V F** *RTSORAC: A Real-Time Object-Oriented Database Model*. Technical report, University of Rhode Island (1994)

39 **Raatikainen, K E E** 'Database Access in Intelligent Networks', *Intelligent Networks*, Harju, J; Karttunen, T and Martikainen, O (eds) Chapman \& Hall (1995) pp 173-193

40 **CCITT** *Principles for a Telecommunications Management Network* Recommendation M.3010 (1992)

41 **CCITT** *System Management Overview* Recommendation X.701 (1992)

42 **Nilsson, G; Dupuy, F and Chapman, M** 'An Overview of the Telecommunications Information Networking Architecture', *Proc. TINA 95 Conference*, Vol 1, TINA-C (1995) pp 1-11

43 **Mowbray, T J and Zahavi, R** *The Essential CORBA*. OMG (1995)

44 **Dupuy, F; Lepetit, Y and Nicolas, B** 'Data Management in an ODP-Conformant Information Networking Architecture', *Proc. TINA 93 Workshop*, Vol II (1993) pp 3-19

45 **Demounem, L and Zuidweg, H** 'On the Coexistence of IN and TINA', *Proc. TINA 95 Conference*, Vol 1(1995) pp 131-148

46 **Callendar, M H** 'Future Public Land Mobile Telecommunications System', *IEEE Personal Communications* Vol 1 No 4 (1994) pp 18-22

47 Special Issue on The European Path Towards UMTS. *IEEE Personal Communications* Vol 2 No 1 (1995)

48 **Mitts, H** 'Use of INtelligent networks in the Universal Mobile Telecommunication System (UMTS), *Intelligent Network*, Harju, J; Karttunen, T; and Martikainen, O (eds) Chapman & Hall (1995) pp 236-245

49 **Wirth, P E** 'Teletraffic Implications of Database Architectures in Mobile and Personal Communications', *IEEE Communications Magazine* Vol 33 No 6 (1995) pp 54-65

50 **Chatras, B and Gallant, F** 'Protocols for Remote Data Management in Intelligent Networks CS1', *Proc. Intelligent Network '94 Workshop*, Vol 1, IEEE (1994)

51 **Taina, J** 'Problem Classes in Intelligent Network Database Design', *Intelligent Networks*, Harju, J; Karttunen, T and Martikainen, O (eds) Chapman & Hall, (1995) pp 194-207

52 **Raatikainen, K E E and Taina, J** 'Design Issues in Database Systems for Telecommunication Services', *Proc. of IFIP IN'95 Working Conference*, IFIP (1995)

53 **Raatikainen, K; Karttunen, T; Martikainen, O and Taina, J** 'Evaluation of Database Architectures for Intelligent Networks', *Proc. TeleCom9'5 - Technology Summit*, Vol 2, ITU (1995) pp 549-553

54 **Deux, O** 'The O_2 System', *Communications of the ACM* Vol 34 No 10 (1991) pp 34-48

55 **Carey, M** 'The EXODUS Extensible DBMS Project: An Overview', *Readings in Object-Oriented Databases*, Morgan Kaufman Publishers (1989)