Appendix

Appendix A

THE UML DOCUMENTATION METHOD

The simulation classes discussed in Appendix B are illustrated using concepts and notation defined by the Unified Modelling Language (UML) (UML,1997). UML is a general-purpose visual modelling language that is designed to specify, visualise, construct and document the artefacts of a software system. UML static class diagrams and object sequence diagrams are used to document and illustrate class interaction.

It is assumed the reader is familiar with the key concepts of object-oriented programming and system modelling. A brief discussion on the UML notation follows. A more in depth discussion on the UML can be found in the UML Notation and UML Semantics documents (UML,1997).

A.1 Static Structure Diagrams

In the UML there are two forms of static structure diagrams, the class diagram and the object diagram. Class diagrams show the static structure of the model, in particular entities that exist, their internal structure and their relationships to other entities and things. An object diagram shows instances of object classes at a particular point in time.

Object diagrams are mainly used to show examples of data structures while class diagrams illustrate object classes and their relationship to one another.

A.1.1 Static Class Diagrams

A model is made up of classes which represent derived model elements. A class represents a set of objects that have similar structure, behaviour and relationships. Class diagrams show the structure and behaviour of a class. The structure of a class is described by its set of attributes. Attributes are data definitions (object types) that each instance of a class holds. The behaviour of a class is represented by its set of operations. An operation (method) is a service that affects the behaviour of the class.

The interaction (relationship) between classes describes a model. Relationships provide a pathway for communication between objects. Relationships commonly found in static class diagrams are:

> Association - This is the most simple type of relationship representing a bidirectional link between classes.

Aggregation - A special form of an association illustrating the relationship between a whole and its parts.

Composition - A form of aggregation with strong ownership and coincident lifetime of the part with the whole.

Multiplicity - The number of instances of one class related to another class.

Navigation - Restricts the direction of the association.

Inheritance – One class shares the structure and/or behaviour of another class.

Sequence diagrams determine what links need to exist between objects in order to accomplish the desired behaviour.

A.1.1.1 Static Class Diagram Notation

Class Representation

A class is drawn as a rectangle with three compartments separated by horizontal lines. The top compartment holds the class name; the middle list compartment holds a list of attributes; while the bottom list compartment holds a list of operations.

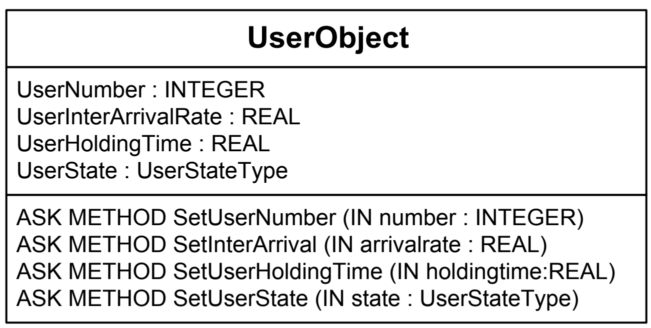

Figure A1 Example of the UML notation for representing a class

Attribute Compartment Notation

The syntax used to describe attributes is:

visibility name : *type-expression* = *initial value*

where *visibility* (optional) is one of the following:

+	public visibility
#	protected visibility
-	private visibility

where name is an identifier string representing the name of the attribute;

where *type expression* is the implementation type of the attribute;

where *initial value* (optional) is the value given to a newly created object;

The visibility marker may be suppressed. An attribute is instance scope unless the name is underlined which represents a class scope attribute.

Multiplicity shows the range of allowable cardinalities that a set may assume. Multiplicity is shown by placing a multiplicity indicator in brackets after the attribute name, for example:

SubscriberCloudUsers [1..*] : UserObject
Points [2..*] : Point

No multiplicity indicator indicates that the attribute holds one value. A multiplicity of 0..* provides the possibility of a null value.

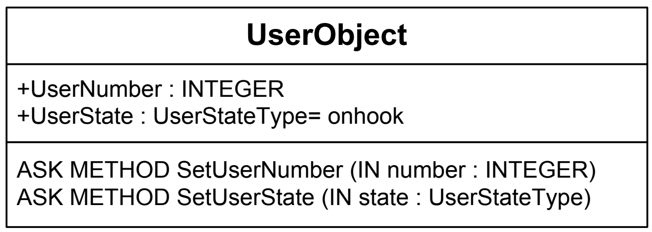

Figure A2 Example of UML attribute compartment syntax declarations

Operator Compartment Notation

The syntax used to describe an operator is:

visibility name (parameter list) : return-*type-expression*

where *visibility* (optional) is one of the following:

+ public visibility
# protected visibility
- private visibility

where *name* is an identifier string;

where return-*type-expression* is the returned implementation type of the operator;

where *parameter list* is list separated by commas of parameters, each using the syntax:

kind name : type-expression = default-value

where kind is IN, OUT or INOUT (the default is IN);

where name is the name of the parameter;

where type-expression is a type expression;

where default-value is an optional value expression for the parameter;

The visibility marker may be suppressed. A class-scope operation is shown by underlining the name and type expression string; otherwise the operation is instance-scope.

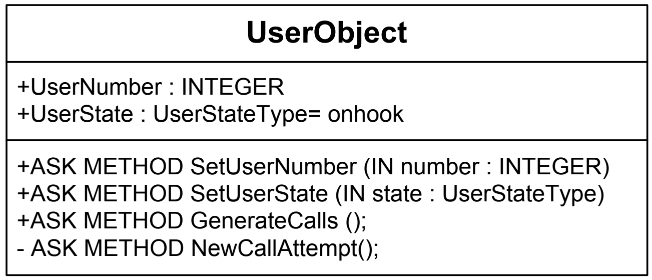

Figure A3 Example of UML operation compartment syntax

A.1.1.2 Relationship Notations

Association Relationship Notation

A binary association is shown as a solid-line connecting two classes. An association end is the end of an association where it connects to a class; forming part of the association. Each association has two or more ends. Adornments at the end of an association represent various types of relationships. The different kinds of adornments are:

> aggregation – A hollow diamond is attached to the end of the path. The diamond is attached to the class that is the aggregate.
>
> composition – Similar to the aggregation adornment except the diamond is filled.
>
> multiplicity – Used to show multiplicity.
>
> navigation – An arrow is attached to the end of the path to show that navigation is supported toward the class attached to the arrow.

Inheritance Relationship Notation

An inheritance relationship is shown as a solid-line from the specific element (sub-class) to the general element (parent class), with a large hollow triangle at the end of the path where it meets the general element (parent class).

A.1.2 Example

This example shows how a telecommunications access network user connected to a concentration terminal (CT) might be represented using classes. UserObj represents a general user in the network while CTUserObject represents a user connected to a CT. CTUserObject inherits the functionality of the UserObject class. The CTUserObject class has an association with the CTObject (representing a CT), because the user is connected to the CT. One SubscriberCloudObject class can contain multiple CTUserObject classes.

The attribute and operation compartments sections in the class blocks have been omitted for diagram simplicity.

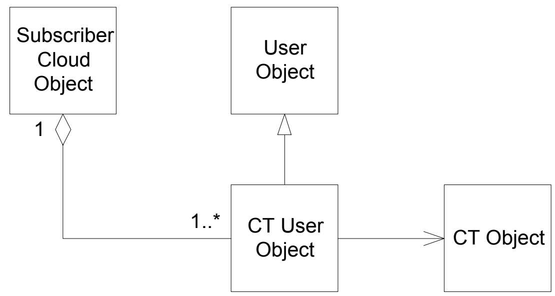

Figure A4 UML class diagram relationship example

A.2 Object Sequence Diagrams

A sequence diagram shows the interaction action between objects over time. In particular, the sequence diagram shows objects participating in the interaction by their lifelines and the messages they exchange arranged in time sequence. Sequence diagrams do not show the association among objects.

A.2.1 Notation

A sequence diagram has two dimensions. The vertical dimension represents time and the horizontal dimension represents different objects. Time proceeds down the page while the ordering of objects across the page is arbitrary. Usually only *time sequences are important*, however real-time applications can have a metric timescale. Various labels (timing marks, descriptions of actions etc.) pertaining to the transitions or activations can be shown in the margin or near the transition or activation.

A.2.1.1 Object Lifeline Notation

An object's lifeline is shown as a dashed vertical line and represents the existence of the object at a particular time. If an object is created or destroyed during the period of time shown on the diagram, then the lifeline starts and stops and the appropriate time points.

An object symbol is drawn at the head of the lifeline. If the object is created during the period shown in the diagram, then the message that created the object is shown with its arrowhead on the object symbol. Objects destroyed during the period shown in the sequence diagram are marked with a large "X", either at the message that causes the destruction or (in the case of self-destruction) at the final return message from the destroyed object. An object that exists before the transaction starts is shown at the top of the sequence diagram. Objects that exist after the transaction finishes has its lifeline continued after the final message.

The lifeline may be split into two or more concurrent lifelines to show conditionality. Each separate lifeline then corresponds to a conditional branch in the message flow. Multiple lifelines may merge at some subsequent point.

A.2.1.2 Activation Notation

An activation shows the period when an object is *performing an action* directly or through a subordinate procedure. It represents the duration of the action in time and the control relationship between the activation and its callers.

An activation is drawn as a tall thin rectangle with the top of the rectangle aligned to the initiation time and bottom of the rectangle aligned to the completion time. In concurrent systems, the activation shows the duration when each object is performing an operation.

A.2.1.3 Message Passing Notation

A message is drawn between two object lifelines as a horizontal solid arrow. A message may start and finish on the same object symbol representing a message sent to itself. The arrow is labelled with the name of the message (operation) and its arguments. *A procedure call is drawn as a solid arrow. A return is shown as a dashed arrow.*

In a concurrent system, wait semantic messages are represented as a full arrowhead while no-wait semantics are represented with half an arrowhead.

Normally messages are drawn horizontally and nothing happens during the message transmission. If the message has a delay in transmission i.e. it requires time to arrive, then the arrow is slanted so that the arrowhead is below the arrow tail.

A.2.2 Example

Examples illustrating object interactions are shown in Appendix B.

Appendix B

DEFINITION OF ISAN SIMULATION CLASSES

Classes developed in order to simulate the traffic characteristics of ISAN networks are presented in this chapter. The classes described in this chapter can be used to simulate various network topologies such as ring, mesh, bus and star networks. Appendix C and Appendix D give results of simulations based on ring architectures.

Classes created to form a simulator can be broken up into the following three categories: classes developed to generate traffic, classed developed to represent components in an ISAN which affect traffic, and classes developed to control the simulation.

B.1 Classes Needed To Generate Traffic

The class **CTUserObj** represents a user connected to a concentration terminal (CT) that generates traffic. The **CTUserObj** inherits the functionality of the generic class **UserObj.** **UserObj** has an originating and terminating traffic model, of type **UserTrafficModelObj**. **UserTrafficModelObj** has fields that describe traffic parameters such as mean holding time, mean interarrival time and the type of traffic (telephone, payphone, Internet-dialup). The relationship between the **CTUserObj** class and other classes necessary to perform the traffic generation function is shown in Figure B1.

Users connected to each CT are divided into groups of a particular traffic type. The class **SubscriberCloudObj** represents such a group. The inclusion of a random number generator object is necessary in order to generate random call arrivals and holding times. A description of these classes' fields and methods is now briefly discussed.

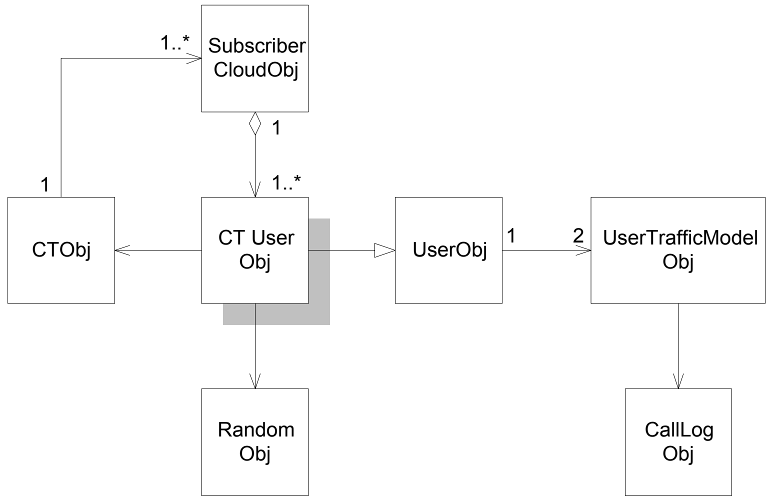

Figure B1 Relationship between CTUserObj and other objects used to generate call attempts

B.1.1 The CallLogObj class

The **CallLogObj** class is used to keep a record of the number of calls generated and lost. It also outputs the interarrival times of user attempts and the holding times of successful attempts to a file. This file can then be viewed using third party software to form opinions on the data such as trends and statistical distribution fit. *An object of type CallLogObj, logs all calls generated and lost only once initial transients have died down*. The **CallLogObj** class definition is shown in Figure B2.

CallLogObj
LogName : STRING CallDuration : REAL; NumberOfCallsGenerated : INTEGER NumberOfCallsLost : INTEGER HoldTimeStream : StreamObj InterStream : StreamObj
ASK METHOD ObjInit ASK METHOD SetupLog (IN name : STRING) ASK METHOD IncCallGeneratedCounter : BOOLEAN ASK METHOD IncCallLostCounter ASK METHOD SetCallDuration (IN duration : REAL) ASK METHOD ResetVariables

Figure B2 CallLogObj class definition

B.1.2 The UserTrafficModelObj class

The **UserTrafficModelObj** contains fields that describe the mean interarrival time, the mean holding time and the traffic type (telephone, payphone, Internet dial-up) of call attempts. Each object instantiation of the **UserTrafficModelObj** has a field of type **CallLogObj**, to keep record of the object's call activity. The **UserTrafficModelObj** class definition is shown in Figure B3.

UserTrafficModelObj
InterArrival : REAL HoldingTime : REAL TrafficType : UserTrafficType TrafficLog : CallLogObj
ASK METHOD SetTrafficLogPointer (IN logpointer : CallLogObj) ASK METHOD SetModelParameters (IN traffictype : UserTrafficType; IN MeanInterArrival :REAL; IN MeanHoldingTime : REAL)

Figure B3 UserTrafficModelObj class definition

UserTrafficType is a user defined type and can be one of the following: telephone, payphone or Internet dial-up.

B.1.3 The UserObj class

The **UserObj** class models a user in the access network and contains a field that describes the user's state i.e. onhook or offhook, and a field containing the user phone number. **UserObj** also contains two traffic variables of type **UserTrafficModelObj** that contain fields that describe the originating and terminating traffic parameters of the **UserObj**. The **UserObj** class definition is shown in Figure B4.

UserObj
UserNumber : INTEGER UserState : UserStateType OriginatingTrafficModel : UserTrafficModelObj TerminatingTrafficModel : UserTrafficModelObj
ASK METHOD SetUserNumber (IN number : INTEGER) ASK METHOD SetUserState (IN state : UserStateType) ASK METHOD SetOriginatingTrafficPointer (IN OriginatingModelPointer : UserTrafficModelObj) ASK METHOD SetTerminatingTrafficPointer (IN TerminatingModelPointer : UserTrafficModelObj)

Figure B4 UserObj class definition

B.1.4 The CTUserObj class

UserObj is a generic class that models user characteristics. The **CTUserObj** represents a user connected to a CT. **CTUserObj** inherits the functionality of the **UserObj** class, and contains additional fields that describes which CT the user is connected to, and a field that can generate random numbers. The **CTUserObj** class definition is shown in Figure B5. Inherited object class operations and attributes from **UserObj** are not shown.

CTUserObj
RandomNumber : RandomObj CT : CTObj
ASK METHOD SetRandomNumberPointer (IN random : RandomObj) ASK METHOD SetPointerToCT (IN ct : CTObj) TELL METHOD GenerateAttempts (IN callorigin : SlotOriginType) TELL METHOD NewTelephoneAttempt (IN origin : SlotOriginType) TELL METHOD NewInternetAttempt (IN origin : SlotOriginType)

Figure B5 CTUserObj class definition

B.1.5 The RandomObj class

The **RandomObj** class generates random numbers based on the congruential method. Three *mod* generators are used to obtain ‘good’ random numbers, and generate random streams of sufficient length. The **RandomObj**’s methods can generate random variables that are uniformly distributed, negative exponentially distributed or lognormal distributed. The **RandomObj** class definition is shown in Figure B6.

RandomObj
IX : INTEGER IY : INTEGER IZ : INTEGER
ASK METHOD GetRandomNumber : REAL ASK METHOD Exponential (IN mean : REAL) : REAL ASK METHOD LogNormal (IN mean : REAL) : REAL ASK METHOD UniformInt (IN low : INTEGER; IN high : INTEGER) : INTEGER ASK METHOD SetSeed (IN ix:INTEGER; IN iy:INTEGER ; IN iz:INTEGER) ASK METHOD ObjInit

Figure B6 RandomObj class definition

B.1.6 The SubscriberCloudObj class

The **SubscriberCloudObj** class contains one or more objects of type **CTUserObj**. The subscriber cloud is a logical grouping of users connected to a CT that have the same call generation and termination characteristics. The **SubscriberCloudObj** class does not use a random number generator object. Each instantiation of **SubscriberCloudObj** however contains an object of type **RandomObj** that generates random numbers. This saves on memory requirements since all users in the subscriber cloud use the same random number generator. Each instantiation of **RandomObj** is seeded with different seeds. Similarly, all the users in a subscriber cloud use traffic models set up for the subscriber cloud. The traffic models then give the number of calls generated and lost for each subscriber cloud. The **SubscriberCloudObj** class definition is shown in Figure B7.

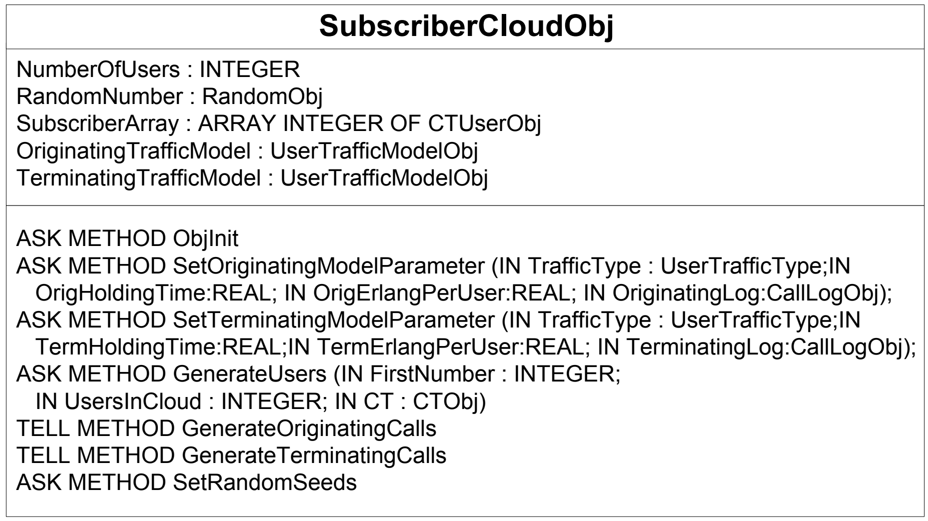

Figure B7 SubscriberCloudObj class definition

B.2 Classes Created That Affect Traffic

The focus of the simulation study in this work is on traffic issues such as blocking and network utilisation. Components in an access network that affect blocking are the slots in a frame that is transported using links, which connect nodes. The class **LinkObj** represents a link in an access network while the class **SlotObj** represents a slot in a frame travelling on a link.

The **LinkObj** class contains a group of slots that represent a frame travelling on the link. **LinkObj** also contains fields that describe the overall number of slots in the frame on the link, the number of slots being used, and the status of the link i.e. link active or link failed. The relationship between the **LinkObj** and **SlotObj** class is shown in Figure B8.

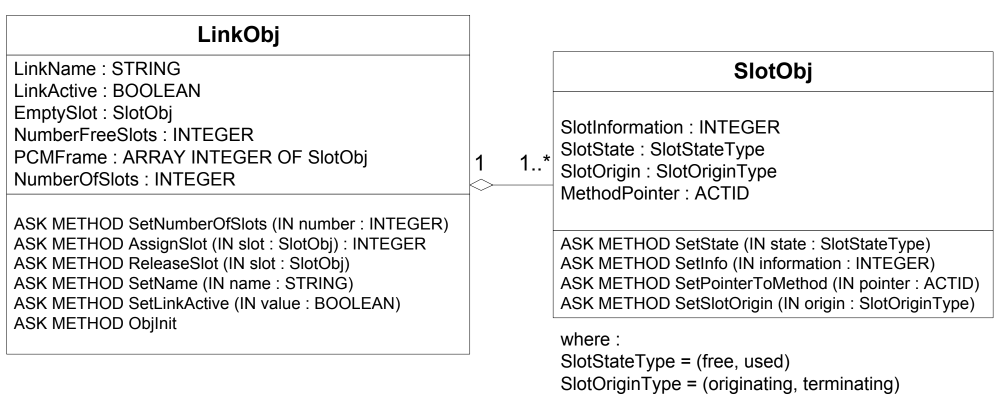

Figure B8 Diagram showing the relationship between LinkObj and SlotObj

The number of slots in a link is not fixed and can be set at any integer value e.g. a link containing 30 slots would represent an E1 (2 Mbit/s) link with 30 bearer channels.

The **SlotObj** class contains fields that describe the state of the slot i.e. free or used, the slot origin and the number of the party in the access network who is one party in the call process. In the case of a failure on a link, the **LinkObj** link status field is set to 'failed', and each slot on the link is cleared. The object that created the slot is notified that the slot has been cleared.

B.3 Classes Created To Control The Simulation

The **CTUserObj** class models users connected to a CT. Users of the same traffic type are grouped and represented by the **SubscriberCloudObj** class. Any number of subscriber clouds can be logically connected to a CT. The **CTObj** class represents a CT in the ISAN. **CTObj** has a field that identifies the subscriber clouds attached by means of a pointer, and fields that define the primary, secondary and backup routing strategies for the CT. The **CTObj** class definition is shown in Figure B9.

Every **CTObj** instantiation creates an instance of an originating and terminating traffic log to keep a record of calls generated to and from the CT. Each **CTUserObj**'s originating and terminating calls are monitored by these traffic logs.

CTObj
SubscriberCloudArray : ARRAY INTEGER OF SubscriberCloudObj OriginatingLog : CallLogObj TerminatingLog : CallLogObj Controller : ControlObj PossibleRoutes : ARRAY RouteDirection, INTEGER OF LinkObj BackupRoutes : ARRAY RouteDirection, INTEGER OF LinkObj CTnumber : INTEGER
ASK METHOD SetControllerPointer (IN controller : ControlObj) ASK METHOD ShowRoutes ASK METHOD RouteCall (IN slot : SlotObj) : INTEGER ASK METHOD ReleaseCall (IN slot : SlotObj) ASK METHOD ResetCallStatisitics ASK METHOD ObjInit

Figure B9 CTObj class definition

RouteDirection is a user defined type and can be one of the following: primary or secondary.

The **ControlObj** class controls the simulation. **ControlObj** is responsible for setting up the access network links, and routing and clearing down subscriber attempts. **ControlObj** contains a field with a list of pointers to all links in the network, and can generate a link failure on any link in the network. Routing strategies are based on fields defined in each respective **CTObj** that requests a route to be established. The **ControlObj** class definition is shown in Figure B10.

ControlObj

LinkArray : ARRAY INTEGER OF LinkObj
NumberOfLinks : INTEGER
FirstLinkStream : StreamObj
LastLinkStream : StreamObj
BackupLinkStream : StreamObj
CallsGenerated : INTEGER

+ASK METHOD LogSlotsUsed (IN link : INTEGER; IN stream : StreamObj)
+ASK METHOD SetupLinks (IN NumberofLinks : INTEGER)
+ASK METHOD GetPointerToLinkObject (IN linkname : STRING) : LinkObj
+ASK METHOD RouteCall (IN slot : SlotObj; IN node : CTObj) : Result
+ASK METHOD ReleaseCall (IN slot : SlotObj)
+TELL METHOD GenerateLinkFailure (IN whichlink : INTEGER; IN FailureTime : REAL; IN FailureDuration : REAL)
-ASK METHOD CheckForAvailableSlots (IN Node : CTObj; IN Route : RouteDirection) : BOOLEAN;

Figure B10 ControlObj class definition

B.4 A Ring Simulator

Two ISAN ring network case study results are presented in Appendix C and Appendix D. Simulation of the ring is simplified by modelling a class **RingObj**, which is responsible for the setting up of network nodes (**CTObj**) and a controller to control the ring network (**ControlObj**). The relationship of **CTObj** with other classes used to simulate a ring type network is shown in Figure B11.

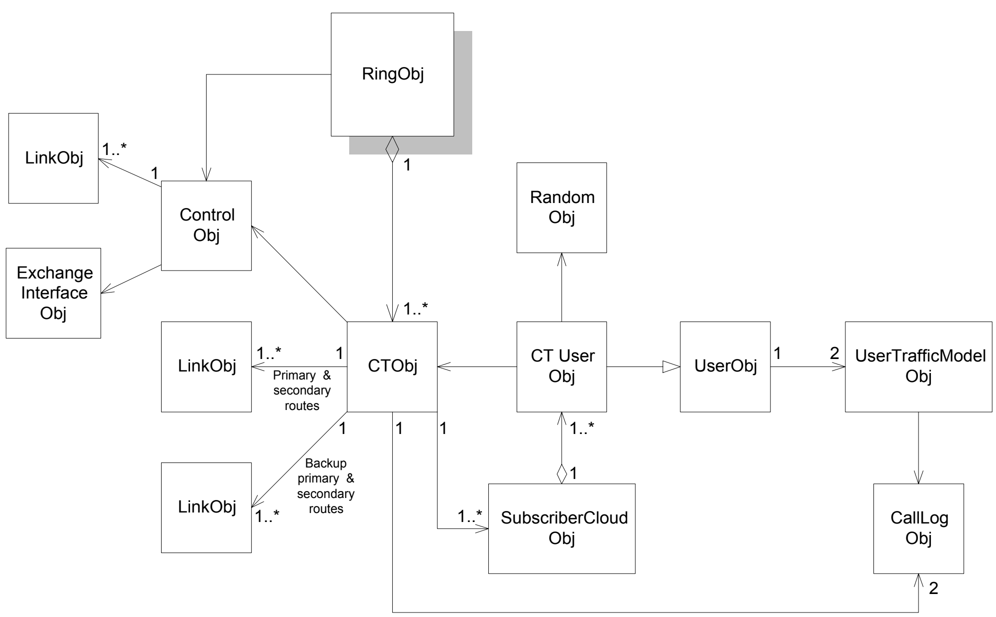

Figure B11 CTObj class relationship diagram

In Appendix C, simulation results of the DCR-300 access network (with an R2 interface to the local exchange) are presented. In the case where an access network and the exchange interface are simulated, the **ControlObj** contains a pointer to an exchange interface. In the DCR-300 simulation case, this would be the **R2InterfaceObj**. The **R2InterfaceObj** class definition as used in the DCR-300 simulation is shown in Figure B12.

R2InterfaceObj
OriginatingCallsLink : LinkObj TerminatingCallsLink : LinkObj <u>InterfaceNumber : INTEGER</u>
ASK METHOD AssignSlot (IN slot : SlotObj) : INTEGER ASK METHOD ReleaseSlot (IN slot : SlotObj) ASK METHOD ObjInit

Figure B12 R2InterfaceObj class definition

B.5 Object Sequence Diagrams

The object sequence diagrams presented in this section apply to traffic simulation of ISAN networks such as ring, mesh, bus, star and point-to-point architectures.

Three object interaction diagrams that describe the fundamental operation of the simulation are the generation of a call attempt and subsequent events, the routing of a call, and the generation of a link failure.

Important UML object sequence concepts from Appendix A are:

- An activation shows the period when an object is performing an action and is drawn as a thin rectangle.
- A procedure call is drawn as a solid arrow. A return is shown as a dashed arrow.
- The vertical dimension represents time and the horizontal dimension represents different objects.
- The object sequence diagrams show time sequences.

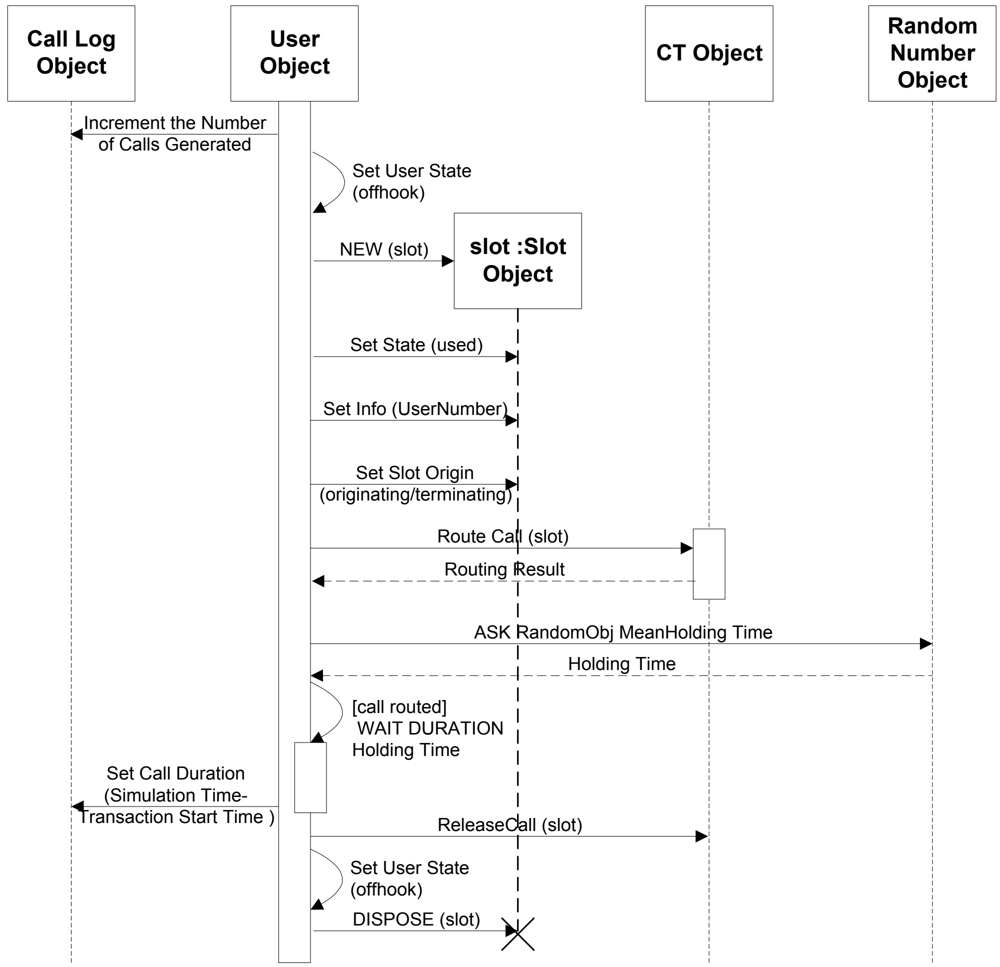

Figure B13 Object interaction during the generation of a call attempt

The two activations in Figure B13 represent the routing a call through each link (using the routing strategy of the CT), and the user in a conversation state. The routing activation is shown in Figure B14. Figure B13 is drawn with the routing result being successful. If the routing result were unsuccessful, there would be no conversation state.

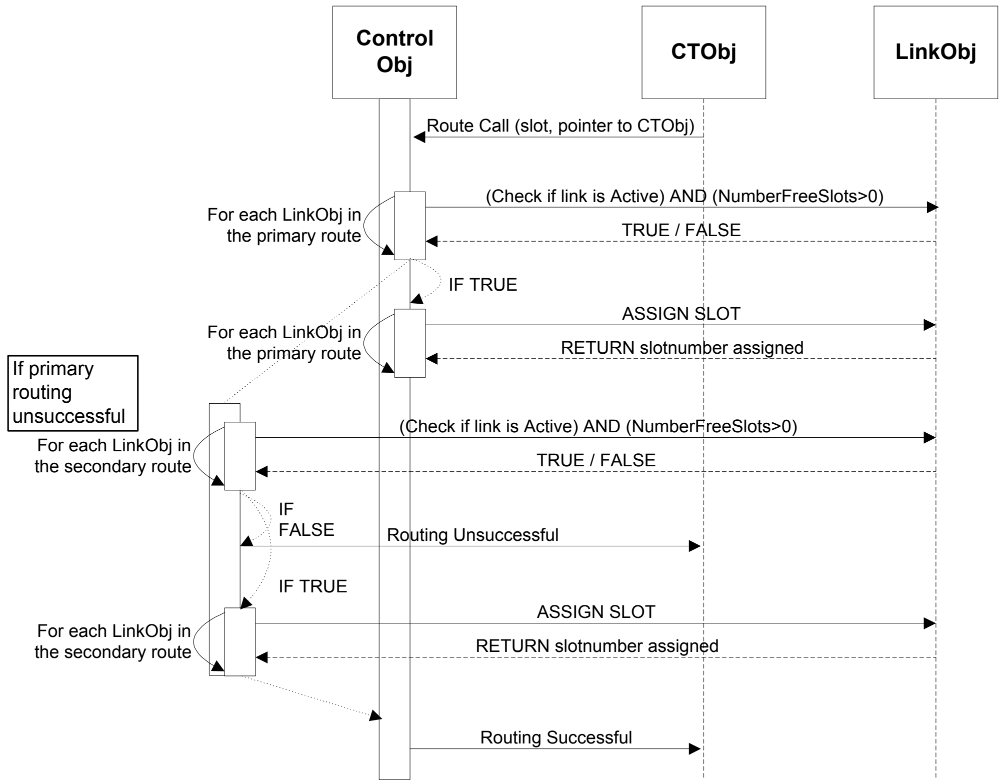

Figure B14 Object sequence diagram showing the interaction of objects instances when routing a call

Routing a call through the ISAN network is based on the following principle:

- Check each link in the primary routing strategy of the respective **CTObj** for an available bearer channel/slot.
- If every link in the primary routing strategy has a bearer channel free, then assign the slot created by the user in the ISAN to each link. The primary routing strategy is then successful and control is passed back to the CT object with a routing successful result. If one link in the primary routing strategy does not have a free bearer channel, then check each link in the secondary routing strategy for an available free bearer channel/slot.
- If every link in the secondary routing strategy has a bearer channel free, then assign the slot created by the user in the ISAN to each link. The secondary routing strategy is then successful and control is passed back to the CT object with a routing successful result. If one link in the secondary routing strategy does not have a free bearer channel, then control is passed back to the CT object with a routing unsuccessful result.

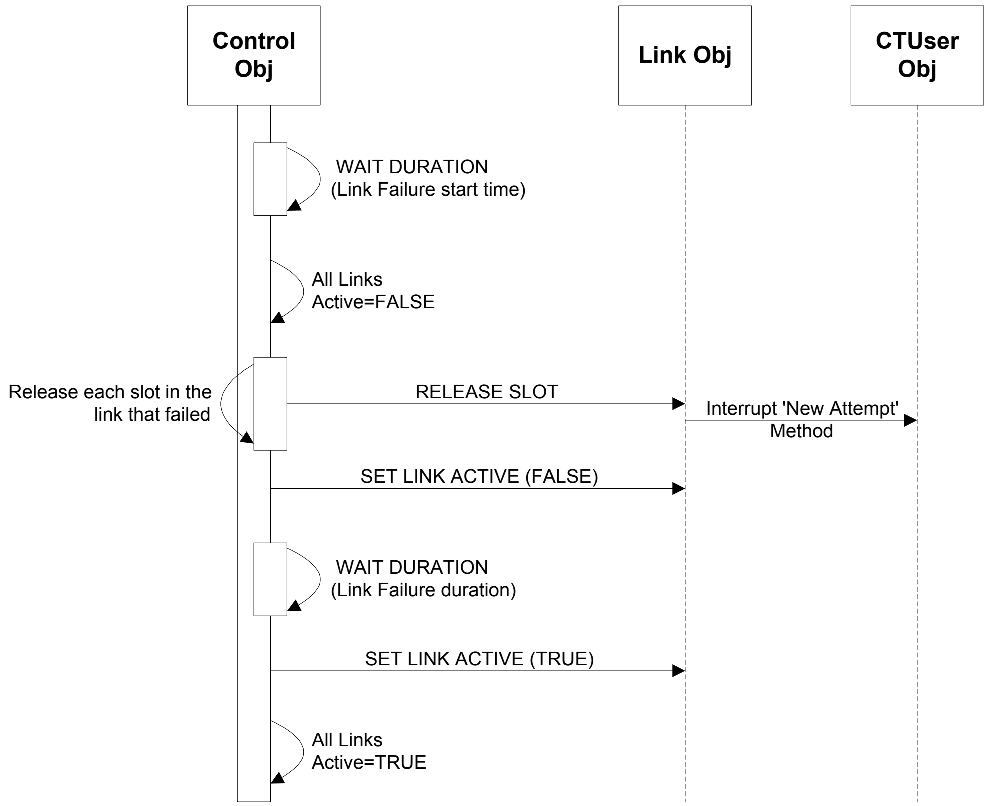

Figure B15 Interaction of objects when generating a link failure

Figure B15 shows the sequence of events to generate a link failure. **ControlObj** has a pointer to each link in the ISAN and can generate a failure on the link at a particular point in time and for a certain duration. Figure B15 starts at the beginning of the simulation (i.e. *simulation time = 0*) and waits until a defined failure time to generate a failure on a particular link in the ISAN. When a failure is generated, each slot in the frame on the link is lost and the call is terminated by interrupting the user party in the ISAN that created the slot object. A link failure duration is then passed and the link is restored to active.

Appendix C

DCR-300 CASE STUDY

A three concentration terminal (CT) DCR-300 network simulation is run with 330 users in the ring. Each CT has 110 users with each user generating 45 mErl (originating and terminating) *telephone traffic*. The mean holding time used is 180 seconds. The estimate blocking probabilities for each CT from 10 simulations runs of 500 000 seconds each is shown in Table C1.

Sim. Run	CT1	CT2	CT3	Total
1	0.002936	0.003628	0.003489	
2	0.003231	0.003231	0.002824	
3	0.003560	0.002306	0.002995	
4	0.003870	0.003443	0.002998	
5	0.004201	0.003347	0.004251	
6	0.003066	0.003201	0.004718	
7	0.004542	0.003796	0.004511	
8	0.003596	0.003773	0.003693	
9	0.003309	0.003246	0.003764	
10	0.002055	0.002931	0.004438	
Mean	0.003437	0.003290	0.003768	0.003498
Std. Dev	0.000698	0.000440	0.000691	0.000610

Table C1 Simulation runs showing estimate blocking probability in the DCR-300

A detailed analysis of the second simulation run is given in the tables below. Table C2 shows the originating and terminating holding times for each CT for the ninth simulation run.

Holding Times

CT	Orig	Term	Average
1	183.8838	179.4463	181.6651
2	181.5949	181.5024	181.5487
3	180.8974	181.1775	181.0375

Table C2 DCR-300 simulated originating and terminating holding times

Table C3 shows the originating Erlang per user in the DCR-300 for the ninth simulation run.

Originating InterArrival rate

Expected	Actual
72.7273	78.0581
72.7273	75.2415
72.7273	77.3861

Originating Erlang traffic

Erl/CT	Users/CT	Erl/User
2.355730	110	0.021416
2.413494	110	0.021941
2.337596	110	0.021251
		0.021536

Table C3 DCR-300 simulated originating Erlang per user

Table C4 shows the terminating Erlang per user in the DCR-300 for the ninth simulation run.

Terminating InterArrival rate

Expected	Actual
72.7273	75.8668
72.7273	75.7059
72.7273	76.2624

Terminating Erlang traffic

Erl/CT	Users/CT	Erl/User
2.365281	110	0.021503
2.397467	110	0.021795
2.375712	110	0.021597
		0.021632

Table C4 DCR-300 simulated terminating Erlang per user

The mean number of slots used on the unidirectional ring for the ninth simulation run were 14.22 slots out of 30 possible slots, with 27 maximum slots used in the unidirectional ring network at any one time.

Appendix D

RING NETWORK CASE STUDY

The simulation parameters, routing strategy and simulation results in the study of a 7-CT ring network with 30-channel bidirectional links are presented in this chapter.

The ratio of users on each node to total ring users for this ring network case study for CTs/nodes 1 to 7 is: 30%, 12%, 8%, 12%, 8%, 20% and 10% respectively.

D.1 Simulation Of Non-Uniformly Distributed Users Generating Telephone Traffic

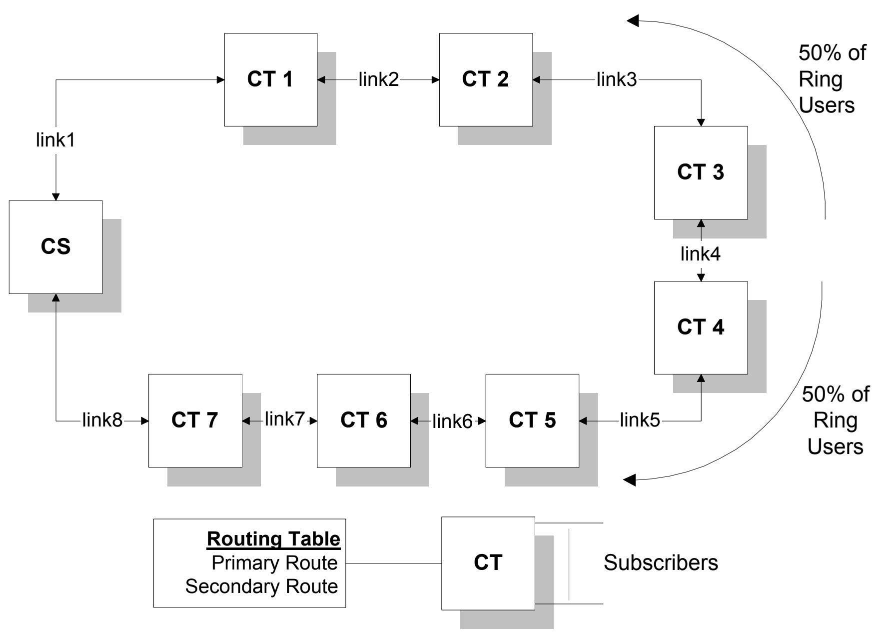

Figure D1 Seven-CT ring network

Traffic in the ring is made up of originating and terminating telephone traffic totalling 45mErl per user. The average telephone holding time used was 180 seconds. Table D1 shows the number of telephone users connected to each CT, and the primary and secondary routing strategy for each CT.

	Users connected	Routing Strategy (link numbers shown)	
		Primary Route	Secondary Route
CT 1	300	1	2 3 4 5 6 7 8
CT 2	120	2 1	3 4 5 6 7 8
CT 3	80	3 2 1	4 5 6 7 8
CT 4	120	5 6 7 8	4 3 2 1
CT 5	80	6 7 8	5 4 3 2 1
CT 6	200	7 8	6 5 4 3 2 1
CT 7	100	8	7 6 5 4 3 2 1
Total	1000		

Table D1 Routing strategy used in the simulation of telephone traffic in the ring network

Table D2 shows the estimate blocking probability for each CT from 10 simulation runs of 500 000 seconds each.

Sim.Run	CT1	CT2	CT3	CT4	CT5	CT6	CT7	Total
1	0.001810	0.002177	0.002298	0.000981	0.002441	0.001835	0.001614	
2	0.002361	0.001742	0.003292	0.002213	0.002740	0.002010	0.002961	
3	0.002875	0.003210	0.002995	0.002155	0.002212	0.002985	0.003761	
4	0.002595	0.002092	0.001991	0.003118	0.002542	0.002481	0.002437	
5	0.002563	0.002698	0.002010	0.002946	0.002510	0.002518	0.002465	
6	0.002338	0.001529	0.002629	0.003520	0.002499	0.002660	0.002182	
7	0.002446	0.001796	0.001540	0.002495	0.001883	0.002720	0.002171	
8	0.002722	0.003482	0.003369	0.002908	0.002621	0.002576	0.003306	
9	0.002548	0.002026	0.002390	0.002994	0.001851	0.002766	0.003612	
10	0.002292	0.002229	0.001972	0.002728	0.002556	0.002028	0.002986	
Mean	0.002455	0.002298	0.002449	0.002606	0.002386	0.002458	0.002749	0.002486
Std.Dev	0.000289	0.000640	0.000612	0.000706	0.000304	0.000376	0.000691	0.000517

Table D2 Simulation runs showing the estimate blocking probability for each CT in a ring network with users generating telephone traffic

A detailed analysis of the sixth simulation run is given in the following tables. Table D3 shows the holding times for originating and terminating telephone traffic in the ring. Table D4 shows the originating and terminating Erlang per user in the ring network.

Holding Time

CT	Orig	Term	Average
1	182.841	178.9847	180.9129
2	181.0135	182.4722	181.7429
3	179.9636	180.0918	180.0277
4	181.9416	178.1949	180.0683
5	182.3424	178.6124	180.4774
6	178.2176	178.9487	178.5832
7	181.9043	181.002	181.4532
		Total	180.4665

Table D3 Simulated holding times of telephone traffic in the ring

InterArrival Rate (O&T)

Actual	Expected
13.9170	13.3333
34.7385	33.3333
52.5822	50.0000
34.5120	33.3333
52.0598	50.0000
20.7788	20.0000
41.9611	40.0000

Erlang (O&T) traffic

Erl/CT	Users/CT	Erl/User
12.999414	300	0.043331
5.231741	120	0.043598
3.423738	80	0.042797
5.217555	120	0.043480
3.466732	80	0.043334
8.594488	200	0.042972
4.324318	100	0.043243
		0.043251

Table D4 Originating and terminating telephone traffic Erlang per user in the ring

The mean number of slots used on the first and last links in the ring for the sixth simulation run were 21.6281 slots and 21.5291 slots respectively, out of 30 possible slots for each link.

D.2 Simulation Of Non-Uniformly Distributed Users Generating Mixed Traffic

A seven-CT ring network is simulated with telephone, payphone and Internet dial-up originating and terminating (O&T) traffic. Each CT generates 80% telephone, 10% payphone and 10% Internet dial-up traffic (O&T). A total of 500 users are distributed unevenly throughout the ring network using the distribution shown in Table D5.

CT	Number of Users	Telephone Users	Payphone Users	Internet Users	Primary Route	Secondary Route
1	150	120	15	15	1	2 3 4 5 6 7 8
2	60	48	6	6	2 1	3 4 5 6 7 8
3	40	32	4	4	3 2 1	4 5 6 7 8
4	60	48	6	6	5 6 7 8	4 3 2 1
5	40	32	4	4	6 7 8	5 4 3 2 1
6	100	80	10	10	7 8	6 5 4 3 2 1
7	50	40	5	5	8	7 6 5 4 3 2 1

Table D5 Routing strategy for mixed traffic in a non-uniformly distributed ring network

10 simulation runs of 500 000 seconds each showing the estimate blocking probability for each CT in the ring for this case is shown in Table D6. Analysis of the first simulation run in the table reveals that the slot occupancies for the fist and last link in the ring were 21.6944 and 21.7110 slots respectively, out of 30 possible slots for each link.

Sim.Run	CT1	CT2	CT3	CT4	CT5	CT6	CT7	Total
1	0.001640	0.001778	0.003088	0.001196	0.002394	0.001615	0.001201	
2	0.001772	0.001199	0.002284	0.001905	0.001324	0.002109	0.002062	
3	0.001757	0.001906	0.002108	0.001680	0.002258	0.001965	0.001321	
4	0.001398	0.001297	0.001511	0.001808	0.002097	0.002224	0.001571	
5	0.001198	0.001582	0.001196	0.001688	0.002235	0.001753	0.002389	
6	0.002061	0.002586	0.002129	0.002222	0.002093	0.002764	0.002135	
7	0.002186	0.002277	0.002579	0.002791	0.003598	0.002627	0.002744	
8	0.002132	0.001592	0.001987	0.00182	0.001814	0.002237	0.001795	
9	0.001234	0.001387	0.002558	0.000894	0.001493	0.002341	0.001824	
10	0.001847	0.002089	0.002118	0.001984	0.001484	0.001855	0.001437	
Mean	0.001723	0.001769	0.002156	0.001799	0.002079	0.002149	0.001848	0.001932
Std.Dev	0.000356	0.000448	0.000537	0.000518	0.000650	0.000368	0.000491	0.000481

Table D6 Simulation runs showing the estimate blocking probability for each CT in a ring network with users generating mixed traffic

D.3 Simulation Of Link Failure In A Ring With Non-Uniformly Distributed Users Generating Mixed Traffic

Simulation in this case is on the effect of link failure in a 7-CT ring network. Users generating originating and terminating telephone, payphone and Internet dial-up traffic in the ring are distributed using the distribution given in the beginning of the chapter, and summarised in Table D5. Link failure of worst case scenarios such as failure of the first and last links in the ring were simulated.

D.3.1 Seven-CT ring network simulation with the backup link connected to CT3

A 7-CT ring network with a backup link connected to the third node (i.e. CT3) in the ring is shown in Figure D2.

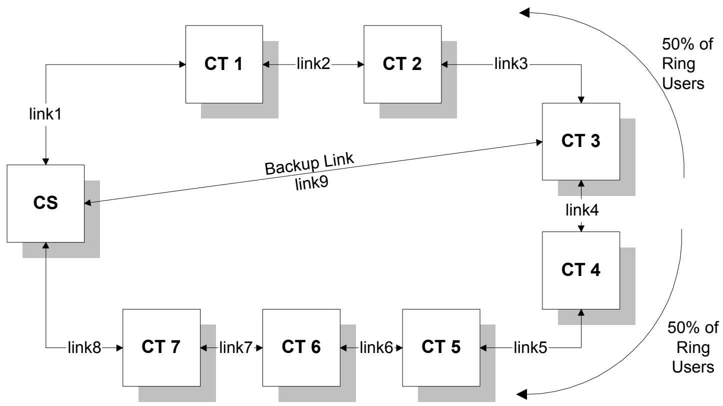

Figure D2 Seven-CT ring network with the backup link connected to CT3

D.3.1.1 Link 1 failing

The primary and secondary routing strategy for a 7-CT ring network with users distributed in the ring as described in Table D5 is shown in Table D7. The backup primary and secondary routing strategy is shown if the first link in the ring fails and the backup link is connected to the third CT in the ring.

	Primary Route	Secondary Route	Backup Primary Route	Backup Secondary Route
CT 1	1	2 3 4 5 6 7 8	2 3 9	2 3 4 5 6 7 8
CT 2	2 1	3 4 5 6 7 8	3 9	3 4 5 6 7 8
CT 3	3 2 1	4 5 6 7 8	9	4 5 6 7 8
CT 4	5 6 7 8	4 3 2 1	5 6 7 8	4 9
CT 5	6 7 8	5 4 3 2 1	6 7 8	5 4 9
CT 6	7 8	6 5 4 3 2 1	7 8	6 5 4 9
CT 7	8	7 6 5 4 3 2 1	8	7 6 5 4 9

Table D7 Backup routing strategy if first link fails (backup link on CT3)

10 simulation runs of 500 000 seconds each showing the estimate blocking probability for each CT in the ring for this case is shown in Table D8. Analysis of the first simulation run shows that the average slot occupancies for the last link and backup link were 21.7210 slots and 21.7225 slots respectively, out of 30 possible slots for each link. The first link slot occupancy is 0 slots since this link failed.

Sim.Run	CT1	CT2	CT3	CT4	CT5	CT6	CT7	Total
1	0.003554	0.003498	0.002701	0.001889	0.001936	0.002480	0.001929	
2	0.004309	0.003650	0.002716	0.002700	0.003424	0.002154	0.002342	
3	0.002873	0.002704	0.002392	0.001389	0.002367	0.001659	0.001435	
4	0.003322	0.003670	0.001192	0.001420	0.001499	0.001438	0.001772	
5	0.004424	0.004034	0.001502	0.001976	0.002120	0.002393	0.002737	
6	0.003307	0.003232	0.002522	0.001469	0.002083	0.001477	0.001685	
7	0.003715	0.003518	0.002961	0.001597	0.001190	0.001853	0.001659	
8	0.004353	0.003658	0.003147	0.001618	0.002413	0.001730	0.001302	
9	0.004082	0.003744	0.002985	0.002469	0.002113	0.002575	0.002016	
10	0.003122	0.004491	0.002370	0.001797	0.001513	0.002324	0.001564	
Mean	0.003706	0.003620	0.002449	0.001832	0.002066	0.002008	0.001844	0.002504
Std.Dev	0.000558	0.000467	0.000638	0.000445	0.000621	0.000427	0.000433	0.000513

Table D8 Simulation runs showing the estimate blocking probability in a ring when the first link fails and the backup link is connected to CT3

D.3.1.2 Link 8 failing

The primary and secondary routing strategy for a 7-CT ring network with users distributed in the ring as described in Table D5 is shown in Table D9. The backup primary and secondary routing strategy is shown if the last link in the ring fails and the backup link is connected to the third CT in the ring.

	Primary Route	**Secondary Route**	**Backup Primary Route**	**Backup Secondary Route**
CT 1	1	2 3 4 5 6 7 8	1	2 3 9
CT 2	2 1	3 4 5 6 7 8	2 1	3 9
CT 3	3 2 1	4 5 6 7 8	3 2 1	9
CT 4	5 6 7 8	4 3 2 1	4 9	4 3 2 1
CT 5	6 7 8	5 4 3 2 1	5 4 9	5 4 3 2 1
CT 6	7 8	6 5 4 3 2 1	6 5 4 9	6 5 4 3 2 1
CT 7	8	7 6 5 4 3 2 1	7 6 5 4 9	7 6 5 4 3 2 1

Table D9 Backup routing strategy if last link fails (backup link on CT3)

10 simulation runs of 500 000 seconds each showing the estimate blocking probability for each CT in the ring for this case is shown in Table D10. Analysis of the tenth simulation run shows that the average slot occupancies for the first link and backup link were 21.7382 slots and 21.6949 slots respectively, out of 30 possible slots for each link. The last link slot occupancy is 0 slots since this link failed.

Sim.Run	CT1	CT2	CT3	CT4	CT5	CT6	CT7	Total
1	0.001079	0.000795	0.001949	0.018474	0.017014	0.015757	0.013647	
2	0.001161	0.000603	0.001364	0.014286	0.012838	0.016453	0.013792	
3	0.000441	0.000698	0.000451	0.015014	0.014421	0.015080	0.015516	
4	0.000919	0.001111	0.001493	0.016973	0.016997	0.016500	0.018210	
5	0.001154	0.000805	0.000746	0.014144	0.014432	0.015814	0.014796	
6	0.001160	0.000907	0.000441	0.013941	0.015373	0.011960	0.014125	
7	0.001166	0.001791	0.001881	0.014008	0.013307	0.013779	0.014822	
8	0.001473	0.001474	0.000589	0.015696	0.015479	0.014924	0.015037	
9	0.001683	0.001609	0.001030	0.018200	0.017630	0.018427	0.014835	
10	0.000872	0.001295	0.000602	0.014451	0.015477	0.015473	0.014427	
Mean	0.001111	0.001109	0.001055	0.015519	0.015297	0.015417	0.014921	0.009204
Std.Dev	0.000335	0.000414	0.000580	0.001753	0.001594	0.001718	0.001291	0.001098

Table D10 Simulation runs showing the estimate blocking probability in a ring when the last link fails and the backup link is connected to CT3

D.3.2 Seven-CT ring network simulation with the backup link connected to CT4

A 7-CT ring network with a backup link connected to the fourth node (i.e. CT4) in the ring is shown in Figure D3.

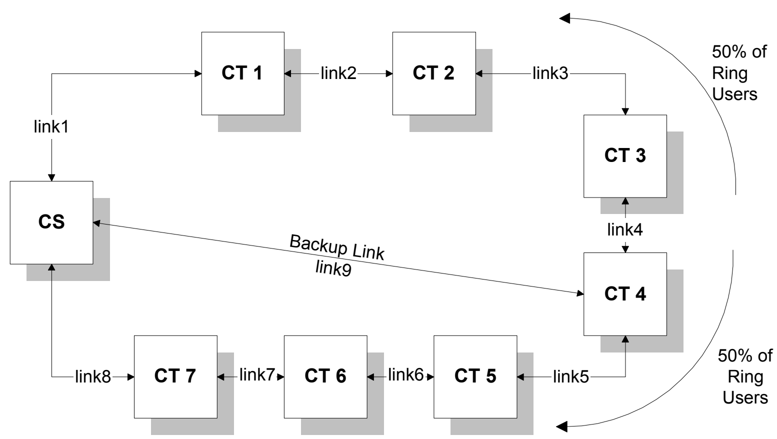

Figure D3 Seven-CT ring network with the backup link connected to CT4

D.3.2.1 Link 1 failing

The primary and secondary routing strategy for a 7-CT ring network with users distributed in the ring as described in Table D5 is shown in Table D11. The backup primary and secondary routing strategy is shown if the first link in the ring fails and the backup link is connected to the fourth CT in the ring.

	Primary Route	**Secondary Route**	**Backup Primary Route**	**Backup Secondary Route**
CT 1	1	2 3 4 5 6 7 8	2 3 4 9	2 3 4 5 6 7 8
CT 2	2 1	3 4 5 6 7 8	3 4 9	3 4 5 6 7 8
CT 3	3 2 1	4 5 6 7 8	4 9	4 5 6 7 8
CT 4	5 6 7 8	4 3 2 1	5 6 7 8	9
CT 5	6 7 8	5 4 3 2 1	6 7 8	5 9
CT 6	7 8	6 5 4 3 2 1	7 8	6 5 9
CT 7	8	7 6 5 4 3 2 1	8	7 6 5 9

Table D11 Backup routing strategy if first link fails (backup link on CT4)

10 simulation runs of 500 000 seconds each showing the estimate blocking probability for each CT in the ring for this case is shown in Table D12. Analysis of the third simulation run shows that the average slot occupancies for the last link and backup link were 21.5854 slots and 21.8204 slots respectively, out of 30 possible slots for each link. The first link slot occupancy is 0 slots since this link failed.

Sim.Run	CT1	CT2	CT3	CT4	CT5	CT6	CT7	Total
1	0.013736	0.014460	0.011789	0.000907	0.000150	0.000420	0.000964	
2	0.016466	0.015382	0.016574	0.000704	0.000894	0.001066	0.000725	
3	0.019318	0.019764	0.017802	0.001011	0.000451	0.001309	0.000599	
4	0.019416	0.017774	0.016172	0.000908	0.001351	0.001151	0.001059	
5	0.016080	0.013930	0.015768	0.000892	0.000602	0.001020	0.000722	
6	0.015879	0.017426	0.014636	0.000594	0.000298	0.000359	0.000606	
7	0.014585	0.015206	0.011708	0.001104	0.001335	0.000665	0.000970	
8	0.014071	0.013293	0.014463	0.001206	0.001663	0.001139	0.000703	
9	0.015235	0.015954	0.017115	0.001283	0.000450	0.000843	0.001210	
10	0.013095	0.013370	0.015965	0.001605	0.001322	0.001616	0.000970	
Mean	0.015788	0.015656	0.015199	0.001021	0.000851	0.000959	0.000853	0.007190
Std.Dev	0.002169	0.002110	0.002078	0.000294	0.000532	0.000393	0.000208	0.001112

Table D12 Simulation runs showing the estimate blocking probability in a ring when the first link fails and the backup link is connected to CT4

D.3.2.2 Link 8 failing

The primary and secondary routing strategy for a 7-CT ring network with users distributed in the ring as described in Table D5 is shown in Table D13. The backup primary and secondary routing strategy is shown if the first link in the ring fails and the backup link is connected to the fourth CT in the ring.

	Primary Route	Secondary Route	Backup Primary Route	Backup Secondary Route
CT 1	1	2 3 4 5 6 7 8	1	2 3 4 9
CT 2	2 1	3 4 5 6 7 8	2 1	3 4 9
CT 3	3 2 1	4 5 6 7 8	3 2 1	4 9
CT 4	5 6 7 8	4 3 2 1	9	4 3 2 1
CT 5	6 7 8	5 4 3 2 1	5 9	5 4 3 2 1
CT 6	7 8	6 5 4 3 2 1	6 5 9	6 5 4 3 2 1
CT 7	8	7 6 5 4 3 2 1	7 6 5 9	7 6 5 4 3 2 1

Table D13 Backup routing strategy if last link fails (backup link on CT4)

10 simulation runs of 500 000 seconds each showing the estimate blocking probability for each CT in the ring for this case is shown in Table D14. Analysis of the second simulation run shows that the average slot occupancies for the first link and backup link were 21.7726 slots and 21.7001 slots respectively, out of 30 possible slots for each link. The last link slot occupancy is 0 slots since this link failed.

Sim.Run	CT1	CT2	CT3	CT4	CT5	CT6	CT7	Total
1	0.001947	0.002400	0.002273	0.002910	0.002857	0.002693	0.003423	
2	0.002193	0.002766	0.000602	0.002533	0.003194	0.002832	0.003046	
3	0.002433	0.002557	0.002833	0.002513	0.003003	0.002679	0.003225	
4	0.001291	0.000696	0.001668	0.001604	0.002275	0.001398	0.002381	
5	0.002034	0.002010	0.001205	0.002079	0.003186	0.003395	0.003432	
6	0.002513	0.003279	0.002684	0.003306	0.002994	0.003606	0.002179	
7	0.002536	0.001888	0.001195	0.003006	0.001933	0.002457	0.002180	
8	0.002267	0.002755	0.002417	0.002970	0.003843	0.003694	0.002762	
9	0.001794	0.001309	0.001213	0.002277	0.002119	0.002363	0.002237	
10	0.002133	0.002180	0.001523	0.003330	0.001958	0.002469	0.002183	
Mean	0.002114	0.002184	0.001761	0.002653	0.002736	0.002759	0.002705	0.002416
Std.Dev	0.000378	0.000756	0.000748	0.000556	0.000635	0.000683	0.000535	0.000613

Table D14 Simulation runs showing the estimate blocking probability in a ring when the last link fails and the backup link is connected to CT4

Appendix E

COMPACT DISC GUIDE

The compact disc (CD) attached on the back of this dissertation contains the simulation programs used to simulate the access network cases presented in Chapter 6 and Chapter 7.

E.1 CD Directory Structure

The CD attached to the back of this dissertation has the directory structure shown in Figure E1.

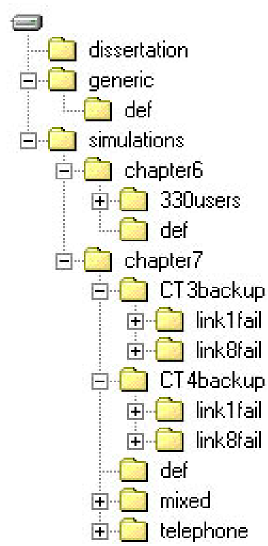

Figure E1 CD Directory Structure

The contents of the directories on the CD are:
\dissertation: contains the dissertation and appendix documents written with Microsoft Word 97
\generic: contains an executable that can simulate any type of access network topology
\chapter7: contains the simulation executables and definition (object class) modules used to simulate the 7-CT ring network presented in Chapter 7
\chapter6: contains the simulation executable and definition (object class) modules used to simulate the DCR-300 ring network presented in Chapter 6

The definition and implementation MODSIM modules used to compile the simulation code into an executable are contained in the **\def** directory of the respective simulation directories.

Simulation generated output files from all simulation runs of network cases presented in Chapter 6 and Chapter 7 are included on the CD. Microsoft Excel 97 spreadsheets showing the estimate blocking probabilities for the network cases presented in Chapter 6 and Chapter7 can be found in the **\chapter6** and **\chapter7** directories on the CD.