INTEGRATED COMMUNICATIONS MANAGEMENT OF BROADBAND NETWORKS

Crete University Press, Heraklio, Greece
ISBN 960 524 006 8

Edited by David Griffin

Copyright © The ICM consortium, Crete University Press 1996

First published in 1996 by
CRETE UNIVERSITY PRESS
Foundation for Research and Technology
P.O. Box 1527, Heraklio, Crete, Greece 711 10
Tel: +30 81 394235, Fax: +30 81 394236
email: pek@iesl.forth.gr

Copyright © The ICM consortium, CUP 1996

The ICM consortium consists of the following companies:

Alcatel ISR, France
Alpha SAI, Greece
Ascom Monetel, France
Ascom Tech, Switzerland
Centro de Estudos de Telecommunicações, Portugal
Cray Communications Ltd., United Kingdom (Prime contractor)
Danish Electronics, Light & Acoustics, Denmark
De Nouvelles Architectures pour les Communications, France
Foundation for Research and Technology - Hellas, Institute of Computer Science, Greece
GN Nettest AS, Denmark
National Technical University of Athens, Greece
Nokia Corporation, Finland
Queen Mary and Westfield College, United Kingdom
Unipro Ltd., United Kingdom
University College London, United Kingdom
University of Durham, United Kingdom
VTT - Technical Research Centre of Finland

Chapter 11

The ICM ATM network simulator

Editor: Panos Georgatsos
Authors: Matthew Bocci, Panos Georgatsos, Michael Hansen,
Eric Scharf, Jim Swift, Jakob Thomsen, Kyriakos Varvaressos

The ICM project has developed cell rate network simulators for use in the development and testing of the management functions/systems developed in the course of the project. The simulator is an integral part of the ICM TMN testbed. It is used as a substitute for a physical ATM network, enabling management applications to be tested in a controlled environment before interfacing with real networks. In addition, the simulator is able to provide a number of major enhancements over real network, thus enabling more extensive and thorough evaluation of the management functionality to be deployed in real networks.

This chapter describes the functionality of the ICM ATM network simulator and the underlying modelling principles. It highlights the functional aspects of suitable generic reusable simulation components that allow the modelling and simulation of complex network configurations ranging from the simulation of a single ATM-based network domain to the simulation of multiple network domains including Customer Premises Networks and the modelling of Virtual Private Networks. The interaction of the simulator with the management system is emphasised as well as the facilities of the simulation components to aid this interaction.

11.1 Introduction

11.1.1 Overview

The dramatic developments in the last two decades in telecommunications have opened the door to many new user services. Powerful network management tools are needed to support the provision of these new services and to maintain the performance of the increasingly complex networks in acceptable levels. Recognising this need, the ICM project was initiated to produce a prototype of an integrated communications management system tailored to the management of IBC networks and services, conforming to the TMN standards [11.13]. The aim of the ICM project is to produce a TMN testbed, where selected management functions could be tested and validated. Apart from the management system (management platform and applications), the ICM TMN testbed includes an ATM network simulator as well as interfaces to real ATM networks such as the RACE Exploit ATM Test Bed (ETB).

The following sections highlights the role of network simulation tools in management studies, as well as the ICM requirements to simulation, justifying the choice of ICM to include a network simulator in its TMN testbed.

This rest of this chapter is organised as follows: Section 11.2 presents the functional architecture of the simulator, identifying its main functional components. Section 11.3 presents the functional decomposition of the simulator, describing the main aspects of traffic and network function modelling. The functional architecture of the identified simulation components is also presented. Section 11.4 describes the interfaces of the simulator, with the simulator user and the management system. Finally, Section 11.5 presents the summary and the conclusions.

11.1.2 Role of network simulation in management studies

Traditionally the main application of simulation has been in ATM research and the design of networks and equipment. More recently, simulation has found application in the development of management systems such as those of CEC RACE projects MIME [11.8], NEMESYS [11.9] and ICM [11.10].

Large scale implementations of broadband ISDN networks carrying high volumes of traffic of an ever changing variety of types require comprehensive and flexible management. This is particularly true when one considers the capital investment in the network infrastructure: efficient resource management becomes paramount. This implies the use of sophisticated, interacting algorithms in the management plane. Thus, there is a major requirement not only to test, characterise and verify algorithms before implementation on real networks, but also to test the actual management software before commitment on real networks and to support staff training. As real networks only provide very restricted access (if at all) for such functions, the need for a simulation alternative to the use of the real networks has major commercial benefits for all these pre-deployment activities.

Simulation has a number of distinct advantages over the use of real networks. Large ATM networks are not yet available for experimentation and development work, and in any case, access to a revenue-earning network is generally restricted. Additionally, such networks would be difficult and costly to instrument in a manner that allows them to host realistic experiments. Unlike with a simulator, access to the parameters for configuring a real network and its traffic would be limited. Experiments run on a simulator can also be frozen in time and are potentially repeatable. Therefore, meaningful statistical results can be drawn and analysed. For evaluating new management systems, simulation reduces the time and cost, especially when the network set-up phase is taken into consideration.

11.1.3 General requirements for supporting TMN experiments

In the ICM project, the simulator is regarded as both a substitute for, and a complement to, a real network for exercising the ICM TMN testbed in an ATM network environment. The ICM simulator thus supports both the development of the ICM TMN testbed, and of the applications for exercising the testbed. Moreover, the simulator is used to provide extended testing capabilities of the developed management applications.

The simulator provides the following value added (on top of real networks) features which are required to support the ICM TMN experiments:

- *Scaling:* The simulator can simulate large high-speed ATM networks. Present day demonstration ATM networks are usually small scale and laboratory based.
- *Flexibility:* Compared to a real commercial network, the simulator can offer flexibility in the following respects:
  - *Functionality:* A variety of network functions, technologies and traffic types can be supported by the simulator.
  - *Measurement:* The simulator provides access to a variety of network data and parameters which may be difficult or impossible to obtain in real systems.
  - *Scenarios:* The simulator can simulate a variety of traffic scenarios as well as eventualities such as node and link failures and buffer overflows.
- *Portability:* The simulator is a software product written in ANSI C and designed to be ported easily between different computing platforms. This means that the ICM TMN testbed can be exercised at many different experimentation sites with a minimum of specialist equipment.
- *TMN Interface:* The simulator provides a standard TMN interface.
- *Simulation Speed:* Ideally, the simulator should run at speeds comparable to those of the real network in order to enable results to be gathered rapidly and also because the simulator should appear as a real network to the management system. Hence the ICM simulator employs cell rate (burst level) modelling techniques in preference to cell level modelling. In cell rate modelling each event represents a change in the rate of flow of ATM cells [11.4], speed-up over cell level modelling being achieved by the consequent reduction in the number events which have to be processed. Speed-up can be achieved while maintaining an accuracy sufficient for the purposes of exercising the ICM TMN testbed. Additionally, the simulator/management system has a mechanism to ensure a common perception of time.

11.1.4 Requirements specific to the ICM project

In ICM the Performance Management, Configuration Management, and Monitoring functions of a TMN compliant management system are being exercised through three main case studies which the simulator must be able to support:

- Network Monitoring,
- Virtual Path Connection and Routing Management (VPCM),
- Intermediate Virtual Private Network Management (IVPNM).

The network monitoring case study was a prerequisite for the VPCM case study, which was in turn needed for the IVPNM case study. In the network monitoring case study, the main network statistics of concern were link or VPC cell traffic counters and link or VPC bandwidth utilisation. Alarms generated by buffer overflow and equipment malfunction were also of interest.

The Virtual Path Connection and Routing Management case study was concerned with the development of a performance management system following the TMN conceptual framework. It requires the following functions in the simulator:

- CAC (connection admission control) algorithms,
- UPC (usage parameter control) algorithms,
- Routing algorithms,
- A CC (call control) mechanism to co-ordinate all three of the above,
- A management interface for receiving the management actions regarding VPCs (creation, deletion, bandwidth modification) and routes (creation, deletion).

In addition the TMN operator must be able to change the bandwidth and routing of virtual channels and paths.

The third ICM case study involves the management of Virtual Private Networks, in the iVPN (intermediate VPN) framework (see Chapter 6). The environment to be simulated is shown in Figure 11.1 and consists of a number of interconnected ATM networks. Network traffic is generated from a number of network users accessing the network at the network access nodes; network users may belong to CPN (customer premises network) sites of specific organisations. Network traffic is carried between source-destination access nodes and/or between CPN sites of the same organisation, through the establishment of switched connections and/or through leased lines. In addition to the VPCM case study requirements, the IVPNM case study imposes the following functional requirements:

- simulation of multiple interconnected ATM networks,
- simulation of multiple CPN sites for one or more organisations,
- ability for requesting CPN connectivity within each organisation,
- ability to configure end-to-end leased lines as far as CPNs, over multiple interconnected networks,
- ability to route CPN calls over the established leased lines (VPN).

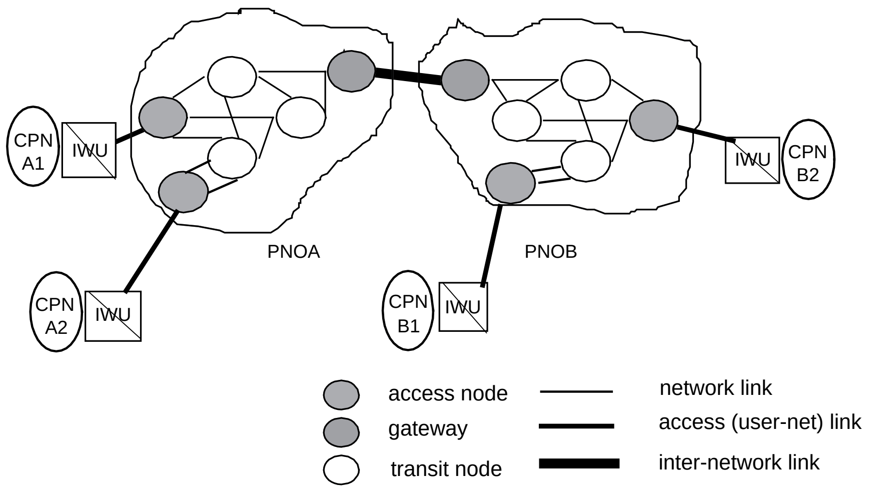


Figure 11.1 Simulated environment

11.2 Architecture

To fulfil the requirements of the ICM case studies, a network simulation tool has been developed that allows users to view the system as an instantiation of a real network through a set of well defined interfaces.

The architecture (Figure 11.2) of the network simulator consists of the following two sets of components.

- *Simulator Interface Components*. These enable interaction with the network simulation system and comprise a:
  - *Man-Machine interface (MMI)*, offering a graphical interface, through the menu-driven facilities of which users can configure, initiate, control and monitor simulation runs.
  - *Q-adaptor (QA)*, providing a Q3 interface to the TMN, through which the TMN can (a) retrieve the required network data and (b) send the necessary management actions to the network.
- *Simulation Engine (SE)*. This receives network configuration information from the MMI and the QA interface components, runs the simulation and produces the required network data for the TMN and the MMI. The SE contains a real-time, distributed Kernel which controls the simulation and handles the flow of data in the simulator system. The SE contains all the models required to simulate the operation of the network, consisting of the following components:
  - The *Kernel*, responsible (a) for handling the flow of messages exchanged between the different components in the Simulation Engine and (b) for the overall time synchronisation of the simulation.
  - The *User Model (UM)* component, simulating (a) the generation of network traffic comprising call requests and the generation of the cell streams associ-

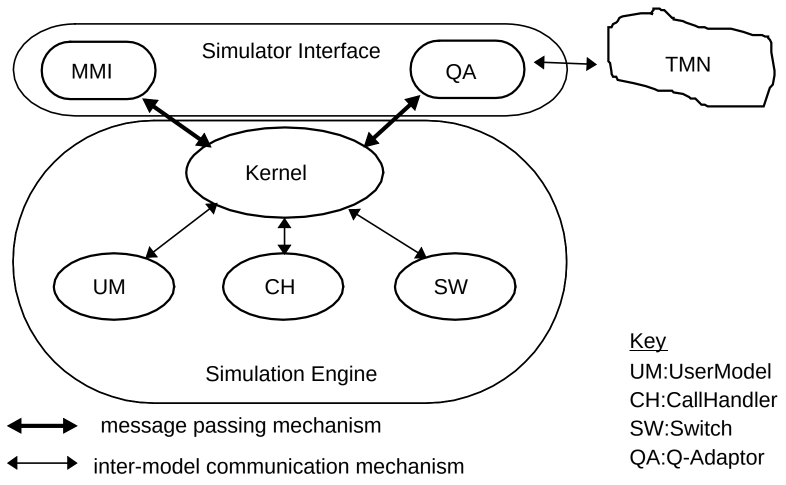


Figure 11.2 Simulator's architecture

ated with particular services, and (b) the Usage Parameter Control (UPC) mechanism.

- The *Call Handler (CH)* component, simulating the Call Control (CC) part of an ATM switch or interworking unit (IWU); this includes the Connection Admission Control (CAC) and Routing functions.
- The *Switch (SW)* component, simulating the cell switching and transmission functions of an ATM switch or a VP cross-connect.

The above components include measurement functions for calculating the required network performance data. The CH and SW components make up a transit network node or an ATM-to-ATM gateway node. The UM, CH and SW components make up a network access node or a CPN site including its interworking unit. The number and configuration of these components can be defined at the start of a simulation run to suit the particular simulation task.

Time advances in the simulator on an *event driven* basis. Components within the simulation engine communicate using Kernel packets that are delivered to each component in time order by the Kernel. These packets can be used to represent, for example, cell rate changes or signalling messages, and their processing in a component can result in the generation of further packets. Simulated time is propagated outside the simulation engine through the simulator interface. This allows, for example, a TMN system to obtain knowledge of the simulator's perception of time. Time synchronisation across the whole simulation is maintained using a synchronous time-stepping scheme [11.1] that enables the simulator to be ported easily between sequential and parallel computing platforms.

Communication between the simulation engine components and the simulator interface components is done by exchanging messages of an agreed format; the payload of these messages is not interpreted by the Kernel. In order to guarantee the correct chronological order in packet processing, communication between the components inside the simulation engine or between the simulation engine and simulator interface components is allowed only via the Kernel.

The following sections describe the main functionality of the architectural components of the whole simulator, presenting the main modelling aspects.

11.3 Functional description

11.3.1 Traffic modelling

The ICM simulator is an event-driven simulator where the basic unit of traffic is a *burst*. A burst is defined as a group of cells with constant cell rate i.e. a burst lasts for a particular period of time during which the inter-cell times do not differ. The start of a burst marks the end of the previous one and is treated as a simulation event. This differs from the cell level modelling where an event marks a cell arrival. In this sense, in burst level simulation the traffic events are far less than the traffic events in cell level simulation. Bursts are characterised by: burst rate, burst delay and burst delay rate.

Burst level simulation is much faster than cell level simulation, as in the burst level simulation the simulator needs to process one event (a burst) instead of many events (corresponding to the cells within the burst) as it would be the case in the cell level simulation. However, cell level simulation is more accurate than burst level simulation; the accuracy depending on the type of traffic being modelled. But for the purposes of ICM the accuracy obtained by burst simulation is thought to be sufficient.

Traffic sources in a burst (cell rate) level simulator need to produce a series of bursts of cells. The approach taken is to model traffic sources as a group of states, each state representing a specified fixed cell rate. A source remains in a state for time duration described by a probability distribution. The scheme for transition between states depends on the model chosen. For example, transition (with specified probabilities) from any state to any other may be possible, or the transition may cycle deterministically through a sequence of states. The burst generation models supported by the simulator are described in Section 11.3.3.3.2.

Traffic modelling has been inspired from the RACE I projects MIME [11.8] and NEMESYS [11.9].

Traffic generation in the simulator is built up in layers. The principle is described in Figure 11.3.

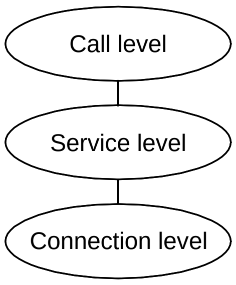


Figure 11.3 The layering of traffic generation

The *call level* describes the behaviour of the user. This level entails the characteristics of the calls, such as call duration time, call request frequency and the type of call in terms of the requested service (see below).

The *service level* describes the different services offered to the users. Different services may be provided: such as PCM telephone, TASI coded telephone, various data services, video broadcast/video conferencing and multimedia services. The approach taken is that a service is made up of a number of connections (see below). The service level combines connection level traffic into services, and gives the relation between them. The service level converts the calls for services requested by the users into actual connection requests to the network.

The *connection level* provides the actual connections that are supported by the network. The UPC, CAC and routing functions of the network operate on the basis of the supported connections. A connection is regarded as a unidirectional flow of information between two network users. Different *connection types* (such as picture connections, VBR, CBR, etc.) may be defined. In this level the characteristics of the traffic (cell stream) of each connection type are given.

Different connection and service types may be defined by the simulator user through the MMI. The definition of connection and service types is an independent procedure from the actual configuration of the network.

The definition of a connection type involves the definition of:

- mean and peak bandwidth requirements,
- burst generation model and its parameters according to which the bursts will be generated (Section 11.3.3.3.2),
- maximum tolerable blocking probability,
- maximum tolerable cell loss ratio,
- maximum tolerable cell delay,
- maximum tolerable cell delay variation.

The first two parameters are used by the CC (call control) functions of the network; the latter four parameters are not required by the functionality in the simulator, but they are provided for consistency with the management system.

Service types are defined in terms of:

- connection type, and,
- direction (forward, return).

Up to four connections may be defined in a service.

Apart from the notions of calls, services and connections, traffic generation in the simulator involves the following notions as well:

The traffic that each node generates, is defined in terms of *service user groups*. A service user group or user group for short, at a specific access node, corresponds to a population of users that request a specific network service from this node. All users defined in a user group follow the same *service pattern*, characterising the demand and usage of the service. A service pattern specifies the frequency that a user (in a user group) will be generating service calls and the time that the user will be connected when the call has been accepted by the network.

A user group is thus a collection of users sharing similar (within statistical range) traffic aspects. A user group could be used to model 20 data terminals sitting in the same CPN, who are able to contact another CPN. The group concept primarily serves to minimise the effort of specifying the initialisation data. In other words, it will have the same effect if you create 20 user groups with one user or one group with 20 users, providing that the traffic parameters are the same.

For a specific service, the definition of a service pattern includes the specification of the following parameters:

- mean call duration time,
- call duration distribution (exponential or deterministic),
- mean silence time (time between two successive call requests),
- silence time distribution (exponential or deterministic),
- start time (time the pattern will be in effect),
- end time (time the pattern will end).

For a given access node and a particular service, the definition of a user group requires the specification of the following information:

- service pattern,
- destination node,
- number of users.

Each user in a user group is simulated independently. This means that there is no correlation between the traffic from different users within a group. Each user will be generating a new call request every silence time seconds (within the silence probability distribution) after the release of a successful or unsuccessful (rejected) call request. Once a call request gets accepted in the network, the call release time will be after mean call duration time seconds (within the call duration probability distribution).

The definition of service patterns for each defined network service and user groups for each access node should be done by the simulator user as a part of the network configuration procedure. The defined connection types, services, service patterns and user groups are passed as initialisation information to the simulator. Based on this information, traffic is generated as described previously.

Figure 11.4 gives the general outline of the functions involved with traffic generation. As it can be seen, traffic generation is split into call generation and burst generation. All the traffic related functions shown in the figure are implemented within the UM component, apart from the Sink function -that treats destined cells to produce VCC statistics- which is implemented in the SW component.

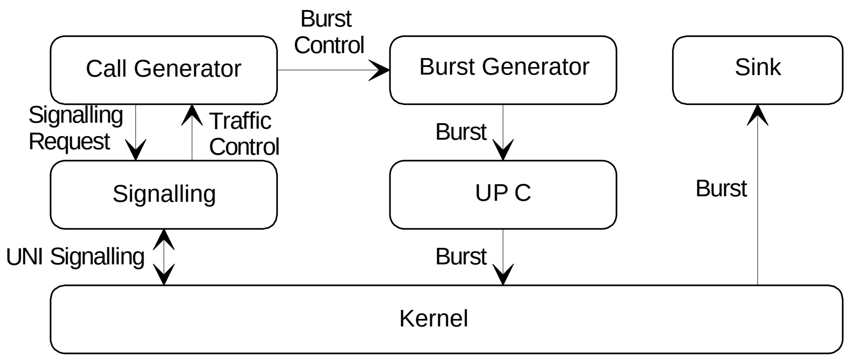


Figure 11.4 Overview of traffic generations functions

11.3.2 Network functions modelling

The functionality of the simulator has been designed following the distinction between the management and control plane in the operation of telecommunications networks [11.14] [11.15]. According to this viewpoint, VPCs and routes are created on a semi-permanent basis, through management activities, whereas VCCs are created dynamically by the control plane of the network through signalling. Following this directive, the simulator is concerned with the simulation of the operation of the network control functions and the transmission of flow of information (cells) within the network resources, producing appropriate results. On the other hand, the simulator offers the appropriate management hooks to allow management systems to interact with the (simulated) network resources, for the purpose to supply/modify the operational parameters of the network functions, for configuring network resources and for retrieving the desired raw data.

In the following sections, the modelling of the major network (control plane) functions and their implementation in the simulator is described.

11.3.2.1 Signalling

The signalling implemented in the simulator is based on a subset of the access signalling protocol specified in ITU-T Q.2931 as described both by RACE project R2081 TRIBUNE [11.11] and in recommendations from the ATM Forum [11.12]. This signalling closely resembles that used in real networks. In the simulator this protocol is used at both the User-Network Interface (UNI) also denoted Tb, and the Network-Network Interface (NNI), i.e. the access signalling protocol is actually used as signalling means over the Network-Network Interface.

The following subset of Q.2931 messages have been used in the simulator:

- Set-Up,
- Call_Proceeding,
- Connect,
- Connect_Acknowledge,
- Release,
- Release_Complete.

No alerting is performed in the simulator, and no timers are used in the signalling. As a consequence only the access signalling states shown in Table 11.1 have been implemented in the simulator. Basically the Uxx states are used on the user side of the UNI and Nxx states are used on network side. However, since the Q.2931 signalling has been used to implement the NNI signalling, Uxx states also occur throughout the network, on the calling side of the NNI.

Figure 11.5 provides a message sequence chart illustrating the signalling for a successful call set-up through a simple three-node network, from Tb to Tb.

The signalling functionality is implemented within the UM and CH simulation components. The UM component includes the user side signalling over UNI and the CH component includes the network side signalling over UNI and the signalling over NNI.

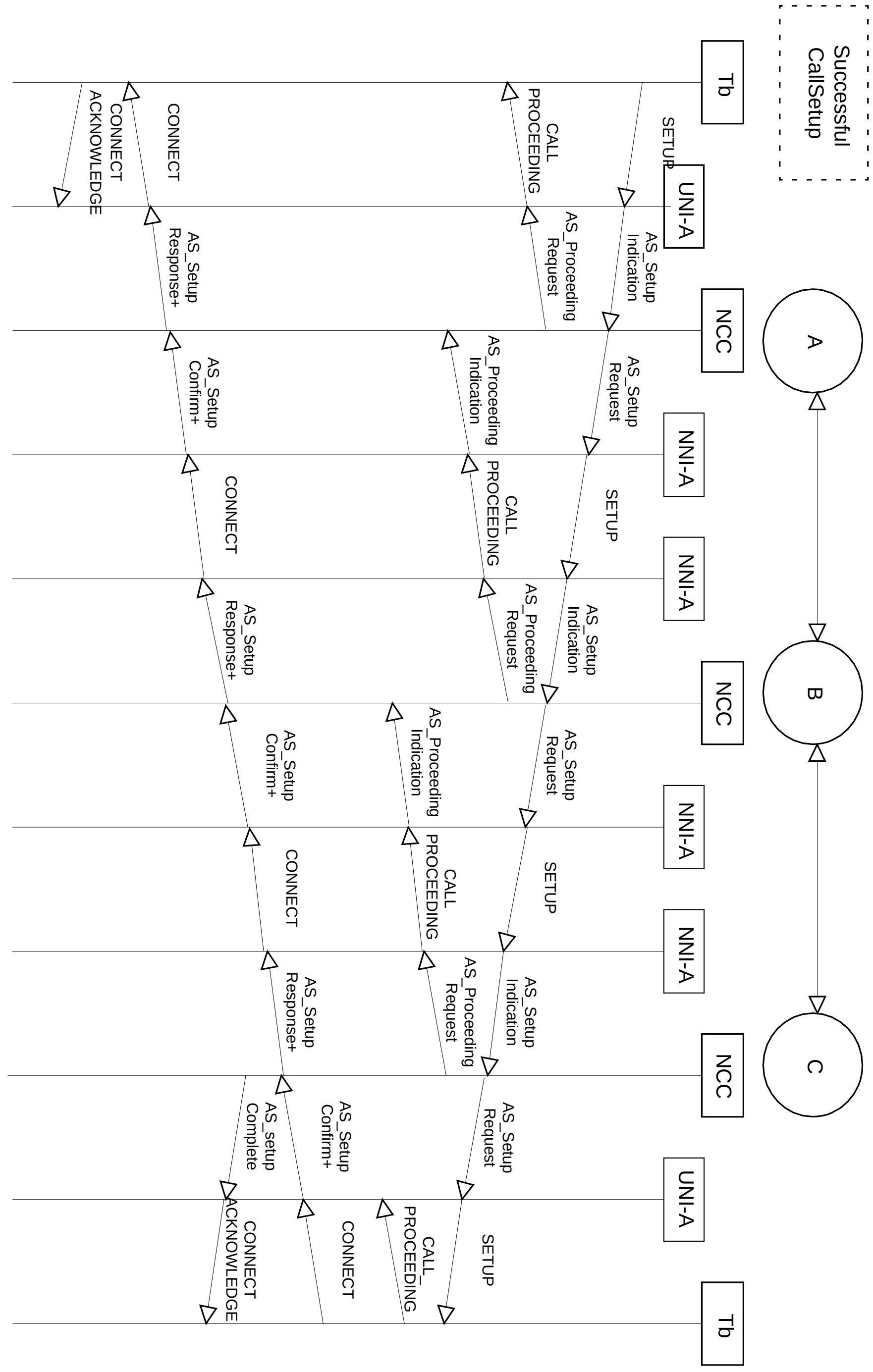


Figure 11.5 Signalling sequence of a successful call set-up procedure

Name	State	Description
Null	U0	No call exists
Call Initiated	U1	This state exists for an outgoing call, when the user requests call establishment from the network
Outgoing Call Proceeding	U3	This state exists for an outgoing call when the user has received acknowledgement that the network has received all call information necessary to effect call establishment
Call Present	U6	This state exists for an incoming call when the user has received a call establishment request but has not yet responded
Connect Request	U8	This state exists for an incoming call when the user has answered the call and is waiting to be awarded the call
Incoming Call Proceeding	U9	This state exists for an incoming call when the user has sent an acknowledgement that sufficient call information has been received to effect call establishment
Active	U10	This state exists for an incoming call when the user has received an acknowledgement from the network that the user has been awarded the call. This state exists for an outgoing call when the user has received an indication that the remote user has answered the call.
Release Request	U11	This state exists when the user requests the network to clear the end-to-end connection (if any) and is waiting for a response
Release Indication	U12	This state exists when the user has received an invitation to disconnect because the network has disconnected the end-to-end connection (if any)
Null	N0	No call exists
Call Initiated	N1	This state exists for an outgoing call, when the network has received a call establishment request but has not yet responded
Outgoing Call Proceeding	N3	This state exists for an outgoing call when the network has sent acknowledge that the network has received all call information necessary to effect call establishment

Table 11.1 Signalling states

Name	State	Description
Call Present	N6	This state exists for an incoming call when the network has sent a call establishment request but not yet received a satisfactory response
Connect Request	N8	This state exists for an incoming call when the network has received an answer but the network has not yet awarded the call
Incoming Call Proceeding	N9	This state exists for an incoming call when the network has received acknowledgement that the user has received all call information necessary to effect call establishment
Active	N10	This state exists for an incoming call when the network has awarded the call to the called user, This state exists for an outgoing call when the network has indicated that the remote user has answered the call.
Release Request	N11	This state exists when the network has received a request from the user to clear the end-to-end connection (if any)
Release Indica-tion	N12	This state exists when the network has disconnected the end-to-end connection (if any) and has sent an invitation to disconnect the user-network connection.

Table 11.1 Signalling states

The addressing scheme used in the simulator caters for multiple network domains and the addressing of CPNs; note that a number of CPNs may be connected at a given network access node. The addressing method is independent of the underlying simulator Kernel numbering scheme. The addressing scheme addresses network nodes and CPNs with a triplet of the form:

- network number,
- node number, and
- UNI number.

11.3.2.2 Routing

As in ATM networks, routing in the simulator is based on a defined set of VPCs. A route is a sequence of VPCs. For a given source-destination (s-d) pair, different routes may be defined for different connection types.

It should be stressed that the simulator does not provide any management functionality; rather it provides the required management hooks. Adopting the distinction between control and management planes [11.14] [11.15], the simulator is concerned with the simulation of the control plane network functions. Therefore, the simulator does not offer any functionality for deriving the appropriate set of VPCs and routes per (s-d) pair and connection type. Rather it provides functionality for receiving the appropriate routing information, defined through management activities, and accomplishing the required routing procedures during VCC establishment, based on this information.

The simulator supports *distributed*, or *source node*, *multiple-route adaptive routing*.

Distributed (hop-by-hop) routing means that routing decisions are taken at the network switches along the route from the source to the destination node. Under distributed routing, the entire route from the source to the destination node is not known at any switch; each switch along the route takes its routing decisions determining the next VPC in the route. In source node routing, the source access node decides on the entire route to the destination node. In this case, end-to-end VPCs between network access nodes must have been defined. Both types of routing are supported by the simulator, provided that it has been given the necessary routing information.

Multiple-route routing means that there may be several routes available for a given connection type and a source-destination pair.

Adaptive routing means that the routes and their parameters as well as the VPCs based on which routes have been defined, may change dynamically by the management system.

For accomplishing routing, the simulator provides the required switching and call control functionality. Specifically, the simulator provides the functionalities of:

- Route selection,
- CAC, and,
- VP/VC switching.

The first two functions, route selection and CAC, are part of the call control functionality in the ATM switches and are modelled within the CH component. The latter function, VP/VC switching, is the switching functionality offered by the network VP and VC switches and is modelled within the SW component. The CAC and VP/VC functionalities as supported by the simulator are described in the following sections. The rest of the section focuses on the route selection functionality as implemented in the simulator and the required routing information.

Route selection in the simulator is accomplished by means of a route selection algorithm of deterministic type, run in the network VC switches, whereby, the selection of a particular route for a given destination and connection type is made on the basis of a priority. The route with the highest priority is selected first and the CAC algorithm applies. If the CAC algorithm rejects the call on the selected route, the route with the next higher priority is then tried. This procedure is repeated until a route able to accommodate the call is found, or until all possible routes are exhausted. In the latter case the call is rejected.

The modular and open (in terms of clearly defined interfaces between the simulation components and between the functional blocks inside each simulation component) design of the simulator allows the incorporation of alternative route selection algorithms (e.g. random or locally adaptive route selection algorithms [11.6], [11.7]). The route selection functionality is part of the CH component functionality, having clear interfaces with the other CH functional blocks. Therefore, the incorporation of a new route selection algorithm does not affect the other functional blocks as long as the interfaces remain the same. Moreover, the existing simulator’s design can easily be expanded to include the feature of route backtracking. According to this feature, when, at a switch, a call cannot be accommodated in any of its alternative routes, the previous switch is signalled to attempt its alternative routes. If all the routes at the access node have been exhausted, then the call is rejected.

The routing information required by the simulator for the operation of the route selection algorithms, is in the form of route selection entries associating a particular destination and connection type with an outgoing VPC with a defined priority. In particular, the route selection entries include the following information:

- destination network,
- destination node,
- destination UNI,
- connection type,
- out port,
- out VPI,
- priority.

This information is kept locally at each simulated node (in the CH component) as in the real ATM switches; based on this information, the route selection algorithms operate in the way described previously.

Route selection entries are downloaded in the simulator either by the MMI, through initialisation files, or by the management system, through the QA component.

With respect to routing information, the simulator offers the following management capabilities:

- creation/deletion of route selection entries,
- modification of priority of an existing route selection entry,
- creation/deletion of a VPC (its TPs - termination points - and its cross-connections).

The above management actions are received by the QA component which translates them to corresponding M interface messages which are then forwarded to the simulation engine.

11.3.2.3 CAC

In ATM networks, CAC (connection admission control) is required in order for the VC switch to determine whether a connection can be accommodated on the VPC selected by the route selection algorithm. The CAC algorithm is associated with each VPC starting from a VC switch and controls the loading on the VPC in an admissible region i.e. in a region where the buffer overflow is within the bound of a predefined probability. This probability is referred to by the term *cell loss target* of the CAC.

The CAC functionality is implemented within the CH simulation component. The simulator supports alternative CAC algorithms; the selection of a particular one is offered as a run time configuration parameter. The following CAC algorithms are supported:

- A simple linear CAC algorithm.

  This algorithm is based on peak bandwidth. The total VPC load is just the sum of the peak bandwidths of the existing connections on the VPC. A new connection is accepted when the sum of its peak bandwidth and the current total VPC load does not exceed the VPC bandwidth; otherwise it is rejected. This algorithm does not operate with a specific cell loss target.

- A 2-level convolution CAC algorithm, taken from the RACE project EXPLOIT [11.16].

  The first level is a linear algorithm based on peaks and operates at the time scale of connection requests. The second level operates in a larger time scale and based on the existing connections accepted on a VPC, applies a convolution method, based on an on-off description of traffic sources, to determine the total VPC load. The calculation is done so that to preserve a specified cell loss target. However, its full implementation is prohibitive due to the memory requirements and slow speed of the convolution algorithm.
- A non-linear CAC algorithm based on effective bandwidth, taken from the CEC COST 242 project [11.17].

  According to this algorithm, the decision as to whether or not to accept a connection on a VPC is dependent on an estimation of the total effective bandwidth of all the existing connections on the VPC plus that of the new connection. If this sum exceeds the available bandwidth then the connection is rejected, else it is accepted. The calculation of the effective bandwidth of a connection of a given type is done by means of a simple algorithm with takes into account the bandwidth characteristics of the connection type as well as a specific cell loss target, without resort to convolution.

The cell loss target associated with a CAC algorithm is set by the MMI, through initialisation files, or by the management system, through the QA component.

With regard to the CAC algorithm, the simulator offers the following management capabilities:

- set of the cell loss target,
- creation/deletion of VPC TPs (termination points),
- modification of VPC allocated bandwidth,
- VPC effective bandwidth (total VPC load as perceived by CAC) statistics.

The above management actions are received by the QA component which translates them to corresponding M interface messages which are then forwarded to the simulation engine.

11.3.2.4 VP/VC switching

As in real ATM networks, so in the simulator, switching is done at two levels: at VP cross-connect level where bursts (cells) are switched within a VPC based on the VPI value; at VC switch level where burst (cells) of a particular VCC are switched between VPCs based on their VCI value.

Switching is accomplished within the SW component, by means of a direct access procedure. For this purpose, the SW component maintains VP-cross-connection and VC-switching tables per input port.

For a given input port, the VP-cross-connection table contains entries of the following form:

- input VPI,
- output port,
- output VPI.

For a given input port, the VC-switching table contains entries of the following form:

- input VPI,
- input VCI,
- output port,
- output VPI,
- output VCI.

Upon burst arrival, the VP-cross-connection table of the port the burst came in, is directly accessed first using the VPI information in the burst header. If an appropriate entry is not found, then the VC-switching table is accessed using the VPI and VCI information in the burst header. If an entry is not found, then a error message is sent to MMI. Upon switching, the burst's VPI and VCI values change to the corresponding output values (indicated by the switching table entry), for the burst to be switched properly in the next switch.

The VP-cross-connection table entries are created on a semi-permanent basis either from MMI, through initialisation files, or from the management system, through the QA component. Apart from the SW component, VP-cross-connection tables are stored in the CH component as well for routing the NNI signals. Both the SW and CH components allow for dynamic creation/deletion of VP-cross-connect table entries.

The VC-switching tables are kept within the SW component. The VC-switching entries are updated accordingly at connection acceptance/release events, through the exchange of appropriate messages from the CH to SW component.

11.3.2.5 Call establishment

As already outlined in Section 11.3.1, service calls in the simulator, as in real networks, are decomposed into a number of unidirectional connections. Each connection belongs to one of the specified connection types that the (simulated) network supports. With respect to its direction, a connection may be either *forward* or *return*.

The establishment of a call involves the establishment of VCCs for all of its constituent connections. As in real networks, so in the simulator, VCCs are established by the control plane via UNI and NNI signalling and by means of the CC (call control) functions provided in the VC switches. The VPCs and the routing information required by the CC functionality is supplied in the simulator through its interface components MMI and QA. This section outlines the call establishment procedure in the simulator.

As it will become apparent, the VCC establishment procedure in the simulator requires the existence of a *reverse VPC* for each VPC defined. For a given VPC going from node A to node B, its reverse VPC goes from node B to node A, spanning through the exactly reverse VPLs and must have the same VPI values for each VPL with the given one. The existence of reverse VPCs is also required for pure signalling reasons - for the return signals to reach back the source node.

The call establishment procedure depends on the type of the call. A call may be one of the following types:

- *Bidirectional* or *symmetric*: is a call that is made up of the same number of connections of the same type in each direction (e.g. telephone calls), or,
- *Asymmetric*: is a call that is made up of an arbitrary number of connections of different types in each directions (e.g. data-transfer or multi-media calls).

To reduce call set-up time and the time of provisionally reserved resources, both types of calls are established in one go, namely forward and return connections are established in one pass and not in different passes (i.e. first the forward connections from source to destination and then the return connections from destination to source). In particular:

- Bidirectional calls are established through signalling from source to destination nodes, and in each intermediate switch the CC functions perform:
  - Route selection for the forward connections for the given destination,
  - CAC on the selected VPC (for the forward connections),
  - CAC on the reverse VPC (for the return connections).
- Asymmetric calls are established through signalling from source to destination nodes, and in each intermediate switch the CC functions perform:
  - for the forward connections:
    - Route selection for the given destination, and,
    - CAC on the selected VPC.
  - for the return connection:
    - CAC on the VPC the signal came in -reverse VPC of the VPC selected in the previous switch (this function is not performed in the source switch), and,
    - Route selection for the given destination.

The call establishment functionality is carried out in the UM and CH simulation components. UNI signalling is split in the UM and CH; NNI signalling and CC functions are implemented within the CH component.

Once the call has been established, the generation of the cell streams, modelled by bursts, commences for each of the connections of the call (call active phase). The bursts are generated by the UM component and are passed to the SW component which models the functions of cell switching and transmission. During the call establishment procedure the VC-switching entries required for the switching of the actual traffic within each established VCC are created. These entries are created within the SW component through appropriate messages from the CH component which convey the required port/VPI/VCI information.

Figure 11.6 shows the role of the simulation components and their interfaces during the call establishment and call active phases.

11.3.2.6 UPC

UPC (user parameter control) or source policing is considered to be a part of CC (call control) functionality but it is dealt with by the UM component in the simulator.

UPC is the function that regulates the amount of traffic injected onto the network by a source. This involves two main operations: the monitoring of the traffic rate, and the discarding of cells that violate the agreed traffic rate parameters. UPC can be applied on a VPC or VCC level.

The simulator supports a UPC algorithm based on peak/mean bandwidths, modified to work in cell rate (burst) traffic:

The UPC function surveys a traffic stream, suppose a VCC. If the cell rate in a burst exceeds the negotiated cell rate (which is the peak rate), then the exceeding rate is cut

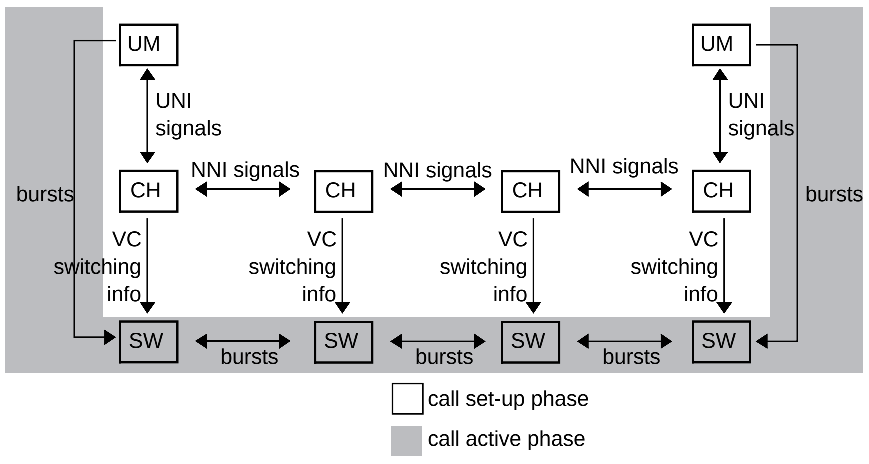


Figure 11.6 Role and interfaces of simulation components during call set-up and call active phases

off and the burst is sent with the peak rate. At connection release instances, an alarm to MMI is sent if the mean source transmission rate over the whole connection duration is violated (compared to the mean connection rate).

It should be noted that the modular and open design of the simulator facilitates the incorporation of alternative UPC schemes (e.g. of leaky bucket type).

11.3.3 Simulator component descriptions

11.3.3.1 The CH component

11.3.3.1.1 Overview

The CH (call handler) component is responsible for the set-up of connections through the network, implementing the functions of:

- signalling, and,
- CC (call control) of ATM switches including route selection and CAC.

The CH component receives connection requests via UNI from the UM component. Then, signalling proceeds to other CHs in nodes along a connection's route via NNIs (Network-Network Interfaces) until the destination node is reached, where the UM component is signalled via UNI.

When a call is set-up or cleared, the CH component sends messages to the SW model via the message passing interface, to enable the SW component to update its VC-switching tables.

The signalling implemented by the CH, encompasses the network side UNI signalling and NNI signalling and it is based on a subset of the access signalling protocol specified in ITU-TS Q.2931 as described both by RACE project R2081 TRIBUNE [11.11] and in recommendations from the ATM Forum [11.12]. This signalling closely resembles that used in real networks. In the simulator this protocol is used at both the

UNI and the NNI. Aspects of the signalling protocol implemented have been described in Section 11.3.2.1.

The CC functionality implemented in the CH comprises route selection and CAC; these functions have been described in Sections 11.3.2.2 and 11.3.2.3 respectively.

The operation of the CH component may be supervised by a management system through the QA component, making it possible to alter and tune the logical topology (VPCs and routes) of the network.

When simulating CPNs or more than one network, the CH component can be configured to operate as a limited IWU (inter-working unit) or a gateway node respectively.

At initialisation the CH component reads initialisation data from a file generated by the MMI. As a minimum, this file contains the physical topology of the network local to the node associated with the CH component. The logical topology, comprising routes selection entries and VPCs, can either be defined by the MMI or by the management system and downloaded to the CHs through initialisation files or M interface messages respectively. As the CH component reads the contents of the initialisation file, it builds data structures to hold the information herein. These data structures are organised in a hierarchy assembled in a ‘node-data’ structure, which is passed to the CH component main function, each time the CH component shall handle an event. The ‘node-data’ structure holds all data needed by the component to operate.

When the initialisation has taken place, the component can be viewed as an event-handler, simulating the call-handling parts of a VC-switch, by processing the events passed to it by the Kernel.

11.3.3.1.2 Functional decomposition

The architecture of the CH component is shown in Figure 11.7.

The *Call Control Layer* selects routes, performs CAC, administers VPC bandwidth and sends information to the MMI and the QA components. A variety of CAC algorithms can be used, depending on the selection made via the MMI. The Call Control Layer sends notifications for connection set-up, rejections and release at VPC or end-to-end levels to the interface components as well as scheduled notifications of statistics of VPC usage.

The *Access Signalling Layer*, lying below the Call Control Layer, is responsible for carrying out the signal handling functionality. It includes signalling both at the UNIs and NNIs.

The *CallHandler Lower interface* operates as a message dispatcher. Depending upon the message kind, it passes the message on to the Access Signalling Layer, the QA Interface or the MMI Interface.

The *QA Interface* and *MMI Interface* are responsible for accepting/sending messages to the QA and MMI interface components respectively. Through the QA Interface component, the M interface messages are received and processed.

The *SW Interface* is responsible for sending messages to the SW component, at connection acceptance/release epochs, conveying VC-switching information.

The *CAC* function performs the CAC algorithm.

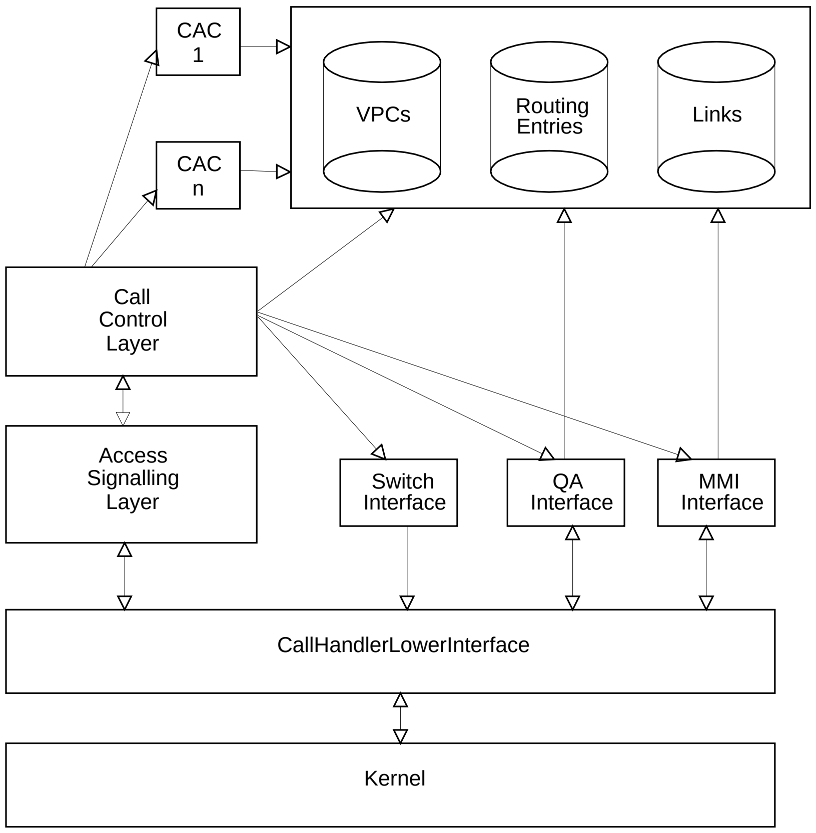


Figure 11.7 CH functional architecture

Finally, the *VPCs*, *Routing Entries* and *Links* blocks are the repository of information regarding the defined VPCs, route selection entries and links, respectively.

11.3.3.1.3 Management capabilities

The CH component offers the following capabilities for aiding interactions with management systems:

- creation/deletion of VP-cross-connections,
- creation/deletion of VPC source or sink TPs (termination points),
- modification of VPC bandwidth,
- modification of CAC performance targets,
- creation/deletion of route selection entries,
- modification of route selection priorities,
- raw data:
  - VPC effective bandwidth (total VPC load as perceived by CAC),
  - notifications on connection establishment/release/rejection on end-to-end level.

11.3.3.2 The SW component

11.3.3.2.1 Overview

The SW (switch) component simulates:

- VP/VC switching, and,
- cell transmission,

of a VC-switch or VP-cross-connect, IWU or gateway node.

Single-stage non-blocking switching architectures are modelled; these include:

- a single buffer per outgoing link, and,
- a buffer shared between all outgoing links.

The SW component interfaces, through the message passing interface, with the CH component - for receiving VC-switching information - and with the UM component - for receiving bursts. At initialisation the SW component reads initialisation data from a file generated by the MMI. As a minimum, this file contains the physical topology of the network local to the node associated with the SW component. Information about VPCs can either be defined by the MMI or by the management system and downloaded to the SWs through initialisation files or M interface messages respectively.

The VP/VC switching functionality was discussed in Section 11.3.2.4. The following section outlines the cell transmission and buffering aspects of the SW functionality.

11.3.3.2.2 Cell transmission and buffering

The switch is modelled at the burst level using a fluid-flow model [11.2] [11.3]. In this model, the rates of the flow into and out of the buffer(s) are the combined rates of all cell streams arriving at, or departing from, the switch. In cell rate simulation the traffic rates remain constant between successive cell rate change events (bursts), and hence the model allows the calculation of link/buffer/cell level statistics as well as the prediction of certain events/alarms (e.g. buffer full events). The model operates on burst arrival events (coming from other SWs), connection establishment/release events (coming from CHs) and link queue-empty/busy events and buffer full/not full events which are local events scheduled by the model itself.

A generic part of the model is a *buffer* (used for resolving contention in the switches). The basic concepts of a single server buffer model are outlined next.

A buffer is described by two parameters: the maximum number of cells that it can store (buffer capacity) and the constant rate at which cells are served (exiting the buffer). The state of a buffer at any time is determined by the combination of the input rates of all connections (VCCs), the current buffer occupancy and the queue parameter values.

The flow of traffic through a buffer is described by input, output, queuing (filling) and loss cell rates. Over any time period, all cells input to the buffer must be accounted for; they are either served, queued or lost. Thus, at any time the cell rates for any connection or for all connections must balance:

input rate = output rate + queuing rate + loss rate

The effect of a change to the input rate of a connection i.e. an event at the input to the buffer, is not immediately apparent on the output, if there are bursts stored. At the time of the input event, only the queuing (buffer filling) or loss rates change. The change appears on the output after the bursts which are currently in the buffer have been served. At the time of this event, only the queuing and output rates change.

The evolution of the buffer occupancy is as follows:

If the instantaneous cell rates of all connections on the input to an empty buffer combine to less than the cell service rate (output link capacity), then the buffer occupancy is zero and the connections' output cell rate equals their input rates. Under the same condition (i.e. input rate less than service rate), if the buffer was not empty, its occupancy would be decreasing (queuing rate negative) until it becomes empty. While the buffer is emptying, its total output rate is its server capacity; when it becomes empty, its output rate equals the total input rate.

If the total input rate is greater than the server capacity, then the buffer occupancy begins to increase and the server capacity is shared among the present connections in proportion to their input cell rates. The queuing rate is the excess of input over output cell rates. While the buffer is not empty, the total output cell rate is always equal to the server capacity. If the buffer becomes full, then the total queuing rate is zero and the excess of total input over output cell rates is lost. The cell loss rate is calculated individually for each connection.

If there are changes to the input rates of the connections while the buffer is full, these changes must propagate through to the buffer output. While these changes are propagating through the queue, the individual queuing rate of each connection temporarily goes positive or negative depending on whether the connection's proportion of the total input rate has increased or decreased as a result of the change; note though, that the combined queuing rates must still total zero. This propagation of connection input rate changes also happens if the buffer is only partly full; in this case, however, the overall queuing rate is not restricted to zero.

The ability of a buffer to be of zero size, yet have connections of non-zero cell rate, is because of the restriction placed on the VCC traffic: during a burst of cells, the intercell rate does not vary. Under the same conditions, with a single traffic source as input, a buffer modelled at the cell level would also be of zero size, because each arriving cell would enter service before the next cell arrival. In the case of more than one source being multiplexed through the buffer (again, under the same conditions), queuing occurs if cells from different sources arrive simultaneously. This situation is not modelled by the cell rate technique, but is obviously reproduced in cell level modelling.

From the above, it becomes apparent that besides the burst arriving events, the model requires the information of the times when the buffer will become empty, busy or full. The times of these events are calculated by the model itself and appropriate local events are scheduled. These events are processed, as the 'normal' burst events (coming from other SW components), to determine any traffic changes in the output rates of the existing connections. Should such changes occur, new bursts are sent.

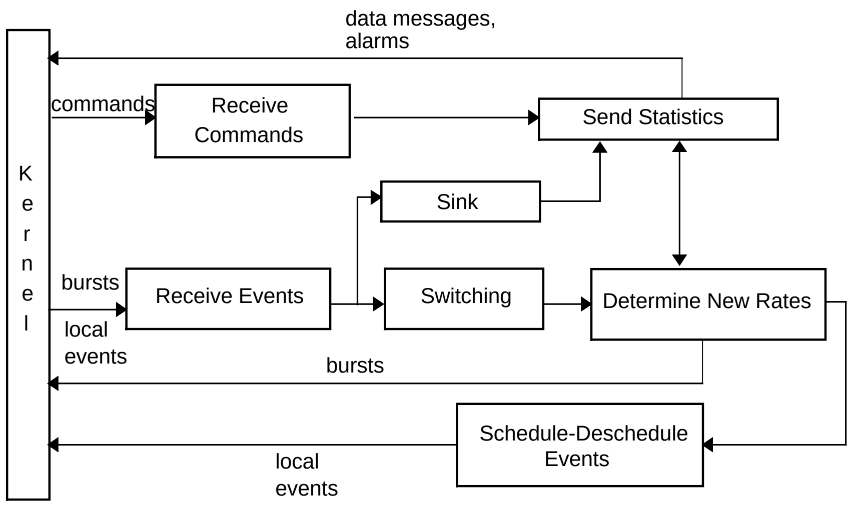


Figure 11.8 SW functional architecture

11.3.3.2.3 Functional decomposition

Figure 11.8 shows the functional decomposition of the SW component.

The *Receive Events* function receives and identifies incoming traffic events and updates performance data associated with input links, and VP links. Simultaneous bursts (cell rate changes) are processed together.

The *Receive Commands* function receives messages from the interface components (MMI/QA).

The *Switching* function switches the incoming bursts to the appropriate output link.

The *Determine New Rates* function determines the status of link queues and buffers for all outgoing links and buffers for which an event has occurred. New cell rate changes are generated on output links accordingly and output cell traffic counters are appropriately updated.

The *Schedule-Deschedule Events* function is responsible for the management of the local events. For all output links and buffers for which an event has occurred, any previously scheduled local events (link-queue empty, buffer full) are cancelled and the new times when these events are to occur are calculated.

The *Send Statistics* function retrieves required performance data from the appropriate parts of the switch model and sends them to the interface components (MMI, QA). Statistics are sent to the MMI at the time of their occurrence in the form of ‘alarms’ corresponding to the events link-queue empty/busy and buffer full/not full. Sample statistics concerning links and buffers may also be sent to the MMI at the end of the simulation run. Statistics are sent on request to the QA and include data associated with links, buffers and VPLs (virtual path links).

The *Sink* function treats cells that reached their destination and calculates cell level statistics. These statistics are sent to the MMI component, as soon as after the connection has been released.

11.3.3.2.4 Management capabilities

The SW component offers the following capabilities for aiding interactions with management systems:

- creation/deletion of VP-cross-connections,
- creation/deletion of VPC source or sink TPs (termination points),
- raw data:
  - transmitted cells counter for link or VPL (virtual path link) source TPs,
  - lost cells counter for link or VPL source TPs,
  - received cells counter for link or VPL sink TPs,
  - lost cells counter for link or VPL sink TPs.

11.3.3.3 The UM component

11.3.3.3.1 Overview

The UM (user model) component is responsible for generating the network traffic, performing the functions of:

- call generation,
- burst generation,
- user access signalling,
- UPC.

The UM component generates calls (corresponding to a user population) using predefined service call generation patterns. Successful service calls generate appropriate cell streams according to the network connection types (bearer service) they are using. Cell level traffic is modelled as series of rate changes, or bursts.

At initialisation, the UM component requires information regarding the supported connection types, the offered network services and the call generation parameters. It interfaces with the CH component for exchanging the required signals for call establishment and release and with the SW component for passing the generated bursts.

The UM component can operate in one of two modes: either as a component simulating the behaviour of specific user populations at a network access node, or as a generator of the network traffic produced by a Customer Premises Network (CPN) interworking with an ATM public network.

The modelling principles of traffic generation in the simulator have been presented in Section 11.3.1. The implemented user access signalling has been discussed in Section 11.3.2.1 and the UPC functionality has been presented in Section 11.3.2.6. The following sections describe the burst generation process in the UM component and the modelling of CPN traffic.

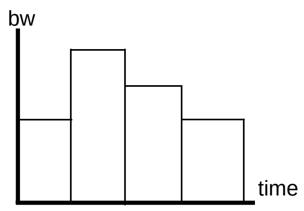
11.3.3.3.2 Burst generation

Bursts in the UM component are generated based on pre-specified *burst models*. Different burst models produce different burst sequences and different bursts in terms of length and rate. The definition of a connection type specifies the model according to which bursts will be produced. A burst model is defined in terms of the following parameters.

- *Silence time*: characterises the inter-burst time,
- *Burst Time 1*: characterises the burst length,
- *Burst Time 2*: characterises the burst length,
- *Peak BW 1*: characterises the burst rate,
- *Peak BW 2*: characterises the burst rate.

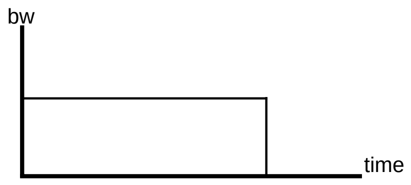
Four burst models are supported which are described next.

- *Burst model 0*. This model corresponds to a variable bit rate (VBR) connection. It produces variable length bursts continuously, as shown in Figure 11.9.
- *Burst model 1*. This model corresponds to a constant bit rate (CBR) connection. It produces a single burst, as shown in Figure 11.10.
- *Burst model 2*. This model corresponds to a variable bit rate (VBR) connection. It produces bursts of the same rate interspersed with silent periods, as shown in Figure 11.11.
- *Burst model 3*. This model corresponds to a variable bit rate (VBR) connection. It produces bursts with different rates sifting between two states, as shown in Figure 11.12.



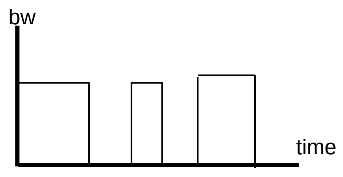
Silence time	Not used.
Burst Time 1	The length of a burst. Each burst has the same length.
Burst Time 2	Not used.
Peak BW 1	The mean of the burst rate distribution. Burst rates are drawn from a truncated exponential distribution.
Peak BW 2	The upper limit of the truncated exponential distribution. The maximum burst rate.

Figure 11.9 Burst generation model 0



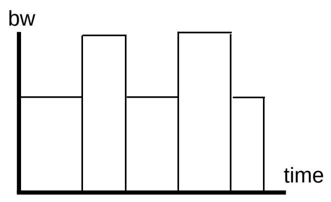
Silence time	Not used.
Burst Time 1	The length of the single burst.
Burst Time 2	Not used.
Peak BW 1	The rate of the burst.
Peak BW 2	Not used.

Figure 11.10 Burst generation model 1



Silence time	The mean inter-burst time. Drawn from an exponential distribution.
Burst Time 1	The mean burst length. Burst lengths are exponentially distributed.
Burst Time 2	Not used.
Peak BW 1	The rate of the bursts. All bursts have the same rate.
Peak BW 2	Not used.

Figure 11.11 Burst generation model 2



Silence time	Not used.
Burst Time 1	The mean burst length at state 1. These are exponentially distributed.
Burst Time 2	The mean burst length at state 2. These are exponentially distributed.
Peak BW 1	The burst rate of the burst at state 1. All bursts at state 1 have the same rate.
Peak BW 2	The burst rate of the burst at state 2. All bursts at state 2 have the same rate.

Figure 11.12 Burst generation model 3

11.3.3.3.3 Customer Premises Network (CPN)

As no specific requirements were imposed by the IVPN case study, (see Section 11.1.4), in order to facilitate the reuse of code, it has been chosen to use ATM as the LAN technology for the CPNs. The services in the LAN environment are the ones offered for the public network users.

The traffic between terminals within a CPN is not specifically modelled. The traffic from one CPN to the other CPN is modelled by simulating the traffic from each terminal and combining the traffic. These traffic sources compete for the available bandwidth between the CPNs. The sum of the traffic coming from a CPN should fit to the capacity of the link from the CPN to the network; hence UPC algorithm, on a VPC level, should apply.

Traffic between CPNs is exchanged through the establishment of switched connections (via normal signalling procedures) or through the establishment of a leased line. In the latter case, as long as the capacity of the leased line is sufficient to accommodate the generated traffic, the calls can be set-up through simple agreement by the two CPNs. The establishment of the leased line as well as the modification of its capacity is done through management activities and it is not of concern to the simulator.

11.3.3.3.4 Functional decomposition

Figure 11.13 shows the functional entities making up the UM component.

The UM component functional entities are described in the following.

The ***User Application*** **entity** is responsible for the generation of calls and the underlying cell streams (for each connection making up the call). As already mentioned (see Section 11.3.1), calls are generated according to service user groups which specify a number of users sharing the same service type. The concept of user groups makes it easy to specify a large number of users in a simple way. The connection type definition specifies a burst model type and a number of corresponding parameters, such as burst rate, burst time, silence time etc. for the generation of the burst sequence. The User Applications entity is entirely proprietary, and specially designed to cater for the burst nature of the simulator. Figure 11.14 shows the functional decomposition of this functional entity.

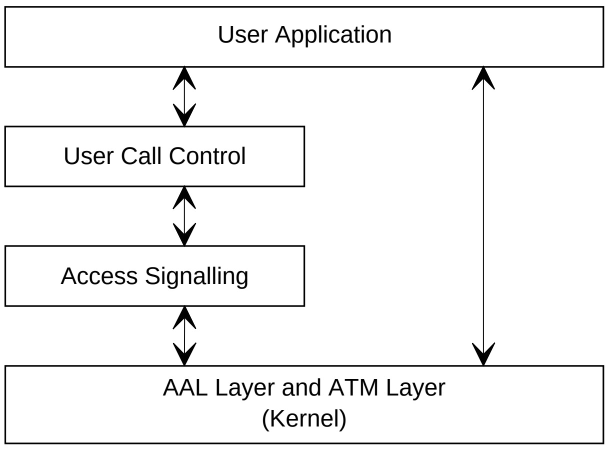


Figure 11.13 Overview of the UM functional architecture

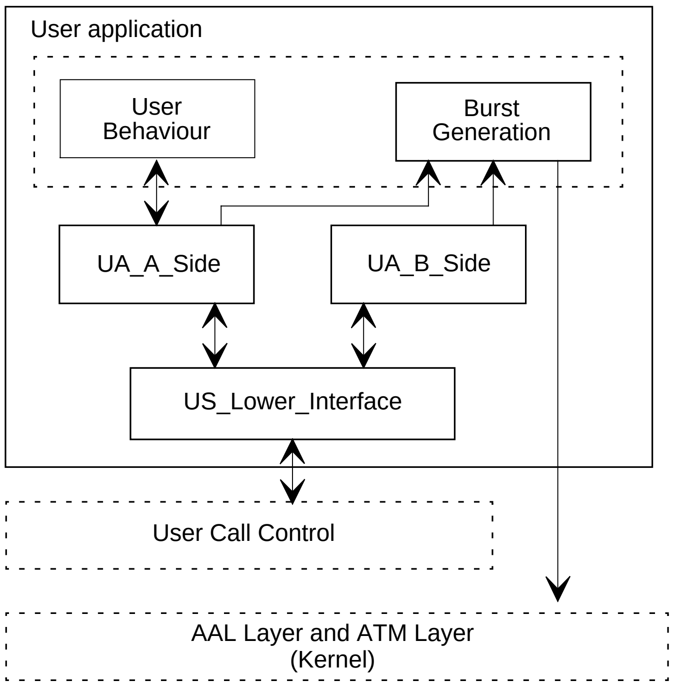


Figure 11.14 Overview of the User Application functional entity

The *User Behaviour* function contains the description of the user behaviour (read through initialisation files) and is responsible for determining the call set-up and release times.

The *UA_A_Side* function takes care of the calls initiated by the users. The function forwards call set-up and release signals. The burst generation is started when a Connect signal is received from the called user. The burst generation is cleared when a Release or Release_Complete signal is received.

The *UA_B_Side* function takes care of calls initiated by other users. The function forwards set-up and release signals. The burst generation is started when a Set-Up signal is received from the called user and in this case a Connect signal is returned. The burst generation is cleared when a Release or Release_Complete signal is received.

The *Burst Generation* function performs the actual generation of the bursts.

The *UA_Lower_Interface* function takes care of the communication with the User Call Control functional entity. Messages received from the User Call Control entity are given to a UA_A_Side function if the call has been originated from this node or to a UA_B_Side function if another user originated the call.

The ***User Call Control* entity** performs the interface between the User Application and the User Access Signalling functional entities. In the User Application entity a call is defined as a collection of connections. The User Access Signalling entity is only able to deal with single connection calls. The User Call Control entity keeps record of the relationship between a user call and the resulting network connections. The User Call Control entity is built to cater for the special needs of the user applications. Inspiration has been found in the RACE project R2029 ESSAI. The User Call Control entity keeps information about the calls including added parties. Figure 11.15 shows the functional decomposition of User Call Control functional entity.

The *CC_Upper Interface* takes care of the communication with the User Application functional entity.

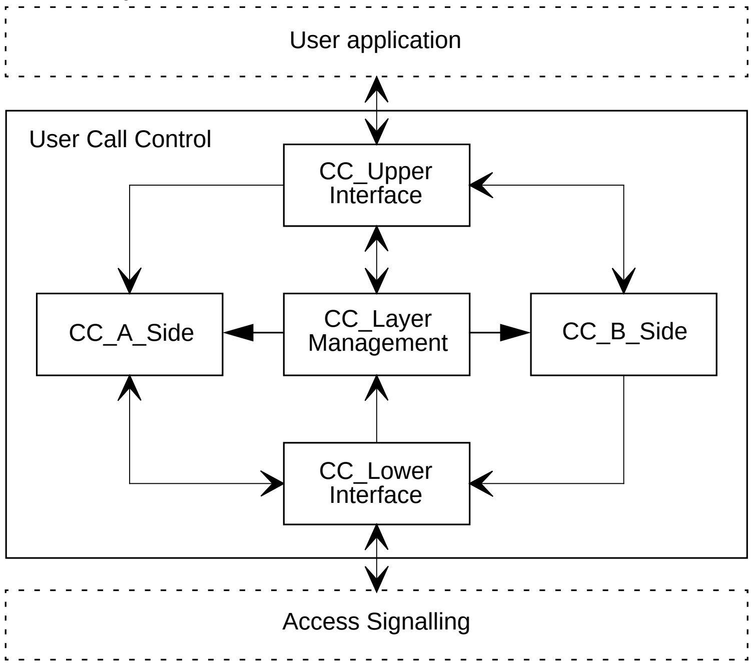


Figure 11.15 Overview of the User Call Control entity

The *CC_Lower Interface* takes care of the communication with the User Access Signalling functional entity.

The *CC_Layer Management* function instantiates new CC_A_Side or CC_B_Side functions when new calls are being set-up, and delete them when calls are being released. It can be called either from the CC_Upper Interface or the CC_Lower Interface.

The *CC_A_Side* function contains state machines for calls originated from this node. This function also keeps track of parties added to the call.

The *CC_B_Side* function contains state machines for calls originated from other nodes. This entity also keeps track of parties added to the call.

The ***User Access Signalling* entity** implements the user side of the UNI signalling according to the ITU-TS specification Q.2931 as described by the RACE project R2081 TRIBUNE [11.11]. The implemented signalling protocol has been discussed in Section 11.3.2.1. It is also the intention to comply to the ATM forum UNI specification version 3.0 [11.12]. The TRIBUNE project has created a complete SDL specification for the access signalling; this specification has been used as far as possible. Figure 11.16 shows the functional decomposition of the User Access Signalling entity.

The *AS_Upper Interface* takes care of the communication with the User Call Control entity. It also invokes the Layer Management function when calls are set-up or released.

The *AS_Lower Interface* takes care of the communication with the lower layers. In our case the lower layers consists of the interface to the simulator's Kernel; in another

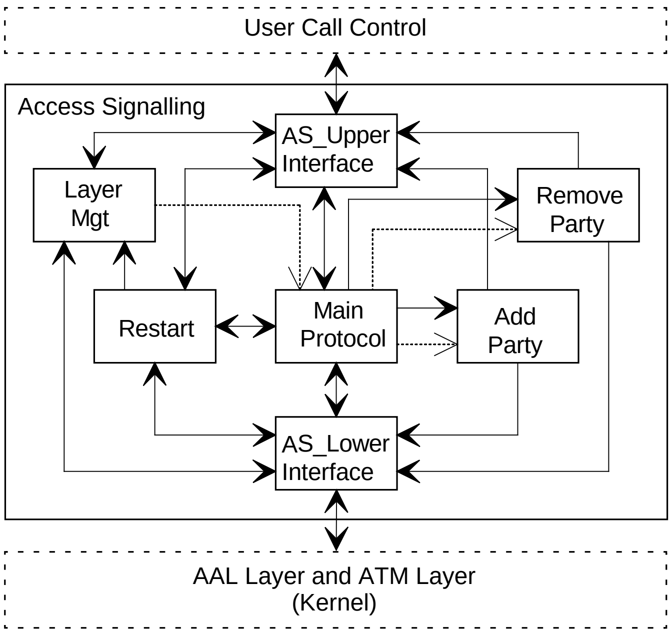


Figure 11.16 Overview of the User Access Signalling entity

case it would be the AAL, if explicit simulation of the AAL was required. It also invokes the Layer Management function when calls are set-up or released.

The *Layer Management* function creates new AS_Main_Protocol functions when new calls are being set-up and delete them when calls are being released. It can be called either from the AS_Upper Interface or the AS_Lower Interface.

The *AS_Main Protocol* function contains state machines for the initiated calls. A call can be in a number of different states (see Table 11.1 on page 320). In the active state it is possible to instantiate the Add Party and Remove Party functions.

An *Add Party* function contains a state machine for a call that is in the process of adding a party to the call. The adding of parties can be done in parallel, also in parallel with removing other parties.

A *Remove Party* function contains a state machine for a call that is in the process of removing a party from the call. The removing of parties can be done in parallel, also in parallel with adding other parties.

The *Restart* function is used to return a virtual channel, all virtual channels in a virtual path or all virtual channels controlled by the signalling virtual channel to the Null state. The procedure is usually invoked when the other side of the interface does not respond to other call control messages or a failure has occurred.

11.3.3.4 The Kernel

11.3.3.4.1 Aim of the Kernel

The simulator Kernel's aim is to provide:

- the environment for the execution of the component models that are used to simulate the network under study, and,
- the necessary communications to the world external to the simulator such as the MMI and TMN compliant management systems.

In real networks, all the equipment and software executes in parallel, so one of the major functions of the Kernel is to provide the mapping of the parallel execution of the network hardware to the number of processors to be used in the simulation (often a single processor). For network simulation, the fundamental requirement for any equipment simulation component is to be able to process and exchange information with adjacent components, the processing being triggered by the arrival of information from an adjacent component or a scheduled event.

The simulation of complex equipment and protocols is a non-trivial task, requiring significant development effort. It is therefore important that such simulation components should be as independent as possible of the computer hardware or operating systems that they execute on. The Kernel therefore is responsible for providing a complete environment for the simulator components. All operating system and hardware dependencies are restricted to the Kernel, allowing the investment in simulation component development to be retained if the simulator is moved to different computing platforms. Example platforms considered for the simulator at the outset were a single processor UNIX workstation and a Transputer array. Both the hardware and operating system were completely different in these two extreme examples.

To meet the above requirements, the Kernel provides an environment for the simulation component code close to that provided by an event-driven real time operating system. The Kernel understands the network topology and the location of the various elements in the network. The simulation component developer has only to consider the operation of a single instance of his/hers component, the parallelisation aspects are all dealt with by the Kernel. The simulation component developer is not aware if the simulator is run on a single processor or multiprocessor computing platform.

11.3.3.4.2 Kernel organisation

The relationship of the Kernel to the other components within the simulator can be shown in Figure 11.2 on page 314.

The Kernel knows about the physical links between the real nodes in the network to be simulated. It also knows about their speeds and delays. On a component basis, links are simply numbered 0 to n-1.

The queue configuration for each component within the simulator, maintained by the Kernel is presented in Figure 11.17. These queues are used for storing packets and synchronising the progression of the component code. Here, stream relate to the physical link in the real network. Each stream may correspond to one link or trunk. For each component, the Kernel sorts incoming packets, and scheduled requests to form a single, time ordered list to be past to the individual component. As well as arrival time, the Kernel also takes into account the time on the real equipment that would be required for the processing of each packet.

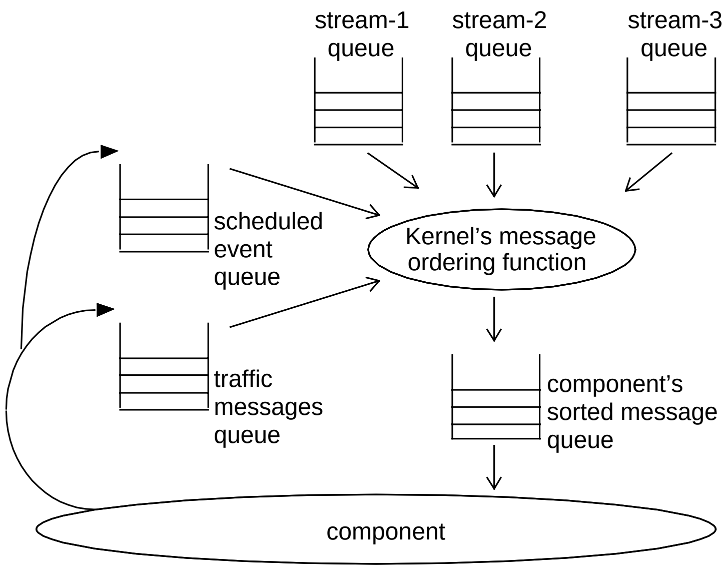


Figure 11.17 Simulation component queue configuration

11.3.3.4.3 Initialisation

At initialisation, the Kernel is first initialised by reading a file which describes the network topology and the Kernel visible characteristics of the components and links. This includes the number of links per component, their speed and delay etc. In addition, the file also contains the name of the initialisation routine to be called to set-up each component, along with the file to configure the particular component instance.

Each component is then initialised by the Kernel calling its initialisation routine and supplying the data from its own instances initialisation file. When the component instance has completed its initialisation, it returns to the Kernel the addresses of all its interface routines. These include the interfaces for receiving packets, calls to its traffic generator, MMI and management system communication.

The simulator Kernel then establishes communication with the MMI and QA interface components, via sockets, and then commences its simulation run.

11.3.3.4.4 Run-time behaviour

The Kernel operates using a *time-stepping* approach to allow for simple synchronisation on multiprocessor systems. It has two sizes of time-step, and the simulation components can request that the smaller time-step is used. If no small time-step requests have been received by the Kernel for a given number of small time-steps, then the Kernel switches back to the large time-step. It is important to note that time-steps are used for synchronisation, and are not the time resolution: the Kernel deals in actual event times. Normally, for operational efficiency, the large time-step would be set to the typical transmission delay for inter node information.

Each simulation component then processes events as they are presented to it by the Kernel. These may be events the component has scheduled, or those caused by the arrival of a packet at that component. During event processing, the component can send packets to other components and schedule events for itself at a later time.

As the Kernel progresses simulated time, it communicates the current time to both the MMI and the management system (via the QA component).

As well as inter-component communication, the Kernel also provides a communication mechanism with the management system and the simulator user via the QA and MMI interfaces. The flow of information is bidirectional, and can thus accommodate both statistics gathering by the MMI and TMN from components as well as real time configuration changes.

11.4 Interface description

11.4.1 The MMI

The MMI component is the front-end of the simulation system. Through its menu-driven and mouse-driven facilities, users can configure, initiate, control and monitor simulation runs. Moreover, MMI offers the necessary facilities to aid interaction with the management system, by launching the TMN processes (OSs) and by supplying the necessary initialisation information with respect to network configuration.

MMI operates in four main functional modes:

- *Configuration Mode:* When operating in this mode, simulator users can define connection and service types and can configure CPNs and single or interconnected networks. Network configuration includes the specification of the network topology, definition of VPCs, routes, and network traffic.

  Network interconnection is achieved through the definition of gateway nodes at each network boundary and the links between the networks. Interconnected networks can be treated as separate networks as well as parts of a super network. The traffic and routes between interconnected networks can also be defined.

  Simulator users can define multiple CPN sites belonging to one or more organisations. The configuration of each CPN site involves the definition of the traffic to be generated. Traffic can only be defined between the CPN sites of the same organisation. Each CPN site is associated with an IWU which in turn is connected to a network access node.
- *Running Mode:* MMI enters this mode after the users have finished with the configuration work. In this mode, users are able to prepare, initiate and view simulation runs of the configured networks.

  While in Running Mode, users can see on-line alarms indicating the occurrence of network events such as acceptance/release/reject events at connection level at CPNs, nodes or VPCs, as well as buffer full/empty and cell loss indications. The MMI produces the necessary initialisation files required by the simulation engine. Moreover, it produces initialisation files for the TMN processes regarding network configuration information.

  Simulation runs are initiated in a non-blocking fashion by means of a client-server networking architecture in which the MMI acts as a client. These runs may be distributed on any computer connected with the machine hosting the MMI, which also allows users to activate the *QA* interface and the other management processes.
- *Play-back mode*: The MMI enters this mode when users wish to re-play simulation runs.

  When in this mode, users can select pre-run simulation runs and play them back. Play-back is meant from the point of view of seeing the history of the simulation run in terms of the alarms and events produced by the simulator. The events that had happened during simulation are displayed in ‘slow-motion’, at a user defined pace. This is achieved by offering to the user the ability to define the number of events to be displayed in a specific time interval. This mode is very useful for the testing and validation of the simulator.
- *Result Mode:* After the completion of a simulation run, MMI may enter this mode, allowing users to view statistics for network entities such as nodes, buffers, links, and VPCs. The type of statistics to be displayed are defined before each simulation run.

MMI also offers facilities common to the Configuration, Running and Result modes for providing different views of the network (e.g. physical, logical, super-network and CPN connectivity) and for querying configuration information.

Moreover, MMI offers to the users facilities for defining their working environment and file-handling facilities for handling their defined connection types, service types, networks and experiments. Simulator user administration facilities are also offered.

More information on the graphical design and implementation aspects of the simulator's MMI can be found in Chapter 8.

11.4.2 The TMN interface

11.4.2.1 Overview

The ability to transparently interchange between the real network and simulator as far as the management system is concerned is the cornerstone of the approach taken by the ICM project. As the project selected OSIMIS as its platform (see Chapter 10), the interface between the simulator and TMN compliant management system was built using the high level facilities OSIMIS provides as a framework.

For the communication between the TMN system and the individual network elements, such as switches, there are two basic alternatives. The first, and the most elegant is that the devices themselves directly understand and interpret the CMIP protocol. However, the protocol itself has many facilities and features, and needs more computing resources than are often found in the interface processors in typical network equipment. Additionally, other protocols may be used by the equipment manufacturers, such as SNMP. In such cases, the CMIP based TMN system accesses the real devices via a Q3 adaptor function (QAF), which at one side provides the CMIP interface for the device, and at the other, communicates directly with the device in whatever proprietary protocol the device itself uses (the M interface). Thus for the TMN to connect to the simulator, a QA (Q3 adaptation) process can be used, just as with actual network equipment.

11.4.2.2 Design of the QA component

As the computing requirements to implement the full protocol in each simulation component instance in the simulation of a large network would be impractical, the approach taken was to implement a QAF running on a UNIX workstation, and then use a proprietary protocol for the M interface between the QAF and the simulator. This functionality is implemented within the QA interface component.

As a simulator can consume considerable computing resources, the option was required to have the simulator on non-UNIX machines, such as a Transputer Array. This meant that the protocol used for the M interface should be usable across different processor architectures without high encoding and decoding costs. To minimise such hardware dependencies, it was decided to use a simple protocol based on ASCII strings. This protocol would then use a transmission method to the simulator appropriate to the hardware that the simulator was on. For a UNIX-based simulator, the simulator Kernel would communicate with the QA via sockets. For a Transputer based simulator, INMOS links could be used. While it may initially appear inefficient to use ASCII strings, it has to be borne in mind that messages in the management plane (as opposed to the control plane) are on a time scale of minutes rather than milliseconds, so such a simple solution is capable of providing adequate performance.

The major difference between using a management system with a real network and the simulator is that of time. Management systems normally use the clock on the processor they are running on, to schedule events and calculate rates of change. However, a simulator will not normally run in real time. Depending on the size of network being simulated, the nature of the traffic, the granularity of the simulation and the power of the computing platform it is running on, it may run faster or slower than real time. It is therefore necessary to ensure that the management system runs against simulated rather than real time. Bearing in mind that one of the aims of the simulator was to test the actual management software itself, the project extended the facilities of OSIMIS so that applications transparently run in real or simulated time, depending on whether time was broadcast from the QA or not. Thus a management process binary could be run with the simulator or the real network without change.

The M interface to simulator was designed to support any number of QA components, allowing complex TMN configuration to be used with the simulator. As it was not felt necessary to simulate in detail the flow of the TMN messages in the simulated network, a pseudo TMN network was implemented within the simulator Kernel to deliver and transmit the messages to and from the individual models.

For a given type of network equipment to be simulated, the managed objects within the TMN side of the QA component must be identical to the objects presented by the equipment been simulated (or its QAF). With OSIMIS, if these are described for the real equipment in GDMO terms, the same GDMO definition can be compiled to build the TMN side of the simulator QAF for that equipment. As OSIMIS compiles managed objects into C++ objects, the main hand-crafted parts of the QA consist of mapping the object changes into the M-interface protocol and visa versa, transmission of the messages from the QA to the simulator, and the receiving of the messages from the simulator and their delivery to the appropriate managed objects. The QA component as currently implemented for the simulator uses the same GDMO definition as the NELM for the ETB network; and this is the standard network element information model that the ICM management system relies on.

11.4.2.3 The M interface

Within the simulator Kernel, the simulation components (CH, SW, UM) are simply numbered with integer number starting from zero. Each component may have a higher level name, or be part of a high level numbering scheme, but this is transparent to the simulator Kernel. Internally, the Kernel must use a non look-up scheme for efficiency because of the very large number of interactions between the simulation components themselves and external agents it has to deal with. Typically, the simulator will have to deal with many orders of magnitude more 'messages' than the management system.

Because of this, one of the main functions of the QA component is to map between the TMN visible name of a node, and the internal Kernel number for each simulation component used to make up the node. It is reminded that each network node is modelled by typically three simulation components (CH, SW and UM). There is a simple relationship between the number of a node and the numbers of the components (understood by Kernel) that make up this node. To accomplish name to number mapping, at start-up the QA reads a file which relates the visible TMN node names with node number and therefore the Kernel's internal component numbers. A node is numbered using a global numbering scheme as well as a scheme local to the network that belongs. This file has to be consistent with the simulator initialisation files. Normally, all these files will have been produced by the MMI from its own network description, so consistency is guaranteed.

When the TMN accesses a managed object in the QA component that requires information from a real resource, the QA functionality maps the access to the required number of low level M interface messages, sending these via a socket (in a totally UNIX configuration) to the simulator Kernel. Also the QA functionality provides the translation to the absolute component number. On receipt of the message, the Kernel delivers it to the appropriate component within the simulator, using the entry point returned by the model to the Kernel during the simulators initialisation process.

In the reverse direction, a simulation component calls the Kernel routine to send a message to the QA. The Kernel sends this message to the QA, which by means of its functionality determines the MO to receive it, and then calls the appropriate method. To deduce the relevant MO, it first resolves the TMN node name. It does this by either using the global node number, or the local node number (relative to the PNO) depending on which the component supplied. It then processes the message type to deduce the actual MOs within the particular node. If required by the MOs method, higher levels of the TMN are then notified using the standard features of OSIMIS. This method is used for both solicited and unsolicited messages. This allows for components to delay their reply to a later simulated time if required.

This communication method is also used to communicate with the MMI.

11.5 Conclusions

Despite the increasing number of real ATM networks, network simulation continues to play an important role in TMN research and development. Simulators act as both a substitute for, and an extension to the scale and facilities obtainable with, real networks and enable management functions/systems to be tested in a controlled environment at low cost, avoiding the need for experimentation on revenue earning networks. Recognising this need, the ICM project has undergone the development of an ATM network simulator, considering it as an integral part of the developed ICM TMN test bed.

This chapter has described the architecture, modelling aspects and basic features of the ICM ATM network simulator and showed how the simulator interworks with the management system. Summarising, the ICM ATM network simulator offers the following capabilities:

- The simulator simulates the operation of ATM networks with a given number of nodes and links.
- The simulator may support a configurable number of traffic sources generating the required network traffic. The simulated traffic sources generate calls (corresponding to a user population) using predefined service call generation patterns.

Successful calls generate appropriate cell streams according to the network bearer service they are using. Cell level traffic is modelled as series of bursts.

- The simulator is able to support a wide range of services. Each network service is defined in terms of connection types. Each connection type (corresponding to a network bearer service) is defined in terms of its bandwidth requirements (mean, peak) and its performance characteristics (cell loss, call blocking, delay and jitter).
- The following network control functions are supported:
  - Distributed call set-up with pretty-close to standards UNI and NNI signalling.
  - VP/VC switching.
  - Routing (route selection algorithm), based on connection type and destination node information. Routes are defined in terms of VPCs. Different routes may be defined for different connection types and multiple routes may have been defined for the same connection type.
  - CAC algorithm, operating at a VPC level and at a predefined cell loss target.
  - UPC algorithm (based on peaks).
  - The simulator simulates the transfer of cells from node to node within an established connection. The simulated network nodes (switches, cross-connects) apart from VP/VC switching simulate cell buffering and cell transmission over their output links. The simulated nodes are modelled by means of a fluid-flow model. The switching architectures supported are: one output buffer per outgoing link, shared output buffer. No input buffers or special buffer sharing schemes are considered.
  - The simulator is capable of producing various statistics at a call, connection and cell level.
- A graphical MMI allows users to configure connection types, network services and networks. The configuration of the network includes the specification of the following information:
  - topology,
  - definition of VPCs (optional),
  - definition of Routes per Connection Type (optional),
  - definition of network traffic.
- Moreover, through the MMI the users can:
  - see different views of the network,
  - on-line alarms indicating the occurrence of various network events,
  - various statistics per network entity (node, VPC etc.).

  The MMI produces the necessary initialisation files required by the simulation engine as well as the files required for the initialisation of the TMN components. The users through the MMI can start up a simulation run (in a non blocking fashion). Moreover, the users through the MMI can activate the TMN processes.
- The simulator provides a Q3 management interface. Through this interface, the TMN can:
  - create/delete/modify VPCs and route selection entries,
  - modify CAC cell loss target,

    - retrieve data; the following data are provided: number of active, accepted, rejected connections and VPC effective bandwidth (total load).
- The simulator supports the modelling of Virtual Private Networks, offering the following capabilities:
    - simulation of multiple interconnected ATM networks,
    - simulation of multiple CPN sites for one or more organisations,
    - ability for requesting CPN connectivity within each organisation,
    - ability to configure end-to-end leased lines as far as CPNs, over multiple interconnected networks,
    - ability to route CPN calls over the established leased lines (VPN).

By making use of generic and reusable modelling components, the management system can be exercised in a wide variety of network scenarios operating on multiple network domains. The use of the cell rate (burst) modelling technique in the simulator maximises the simulation speed whilst still allowing cell level traffic statistics to be obtained. The inherent portability and openness in the simulator design gives potential for further enhancing its functional capabilities in terms of alternative network technologies and functions as well as for enhancing its speed-up by enabling the simulator to be ported to a multiprocessor environment.

11.6 References

[11.1] M. Bocci, J. M. Pitts, E. M. Scharf, “Performance of Time Stepping Mechanism for Parallel Cell Rate Simulation of ATM Networks,” 11th IEE UK Teletraffic Symposium, April 1994.

[11.2] B. Meister, N. Karatzas, P. Georgatsos, “Modelling of ATM networks,” 5th RACE TMN Conference, London, 1991.

[11.3] B. Meister, P. Georgatsos, N. Karatzas, “Switch models for TMN applications,” 6th RACE TMN Conference, Madeira, 1992.

[11.4] J. M. Pitts, Z. Sum, “Burst Level Teletraffic Modelling and Simulation of Broadband Multiservice Networks,” 7th IEE UK Teletraffic Symposium, April 1990.

[11.5] K. E. Mourelatou, D. Griffin, P. Georgatsos, G. Mykoniatis, “ATM VPN services: Provisioning, Operational and Management Aspects,” IFIP TC6, 3rd Workshop on Performance Modelling and Evaluation of ATM Networks, West Yorkshire, UK, July 1995.

[11.6] P. Georgatsos, D. Griffin, “Load Balancing in Broadband Multi-Service Networks: A Management Perspective,” Proceedings of the Third Workshop on Performance Modelling and Evaluation of ATM Networks,” July 1995.

[11.7] P. Georgatsos, D. Griffin, “A management system for load balancing through adaptive routing in multi-service ATM networks,” accepted for presentation in INFOCOM’96.

[11.8] RACE R1084 MIME, Deliverable 26, “Final Experiment Reports - Recommendation on Definition of a Common TMN Testbed,” R1084/AL/ISD/DS/C/026/B1, Feb.1993.

[11.9] RACE R1005 NEMESYS, Deliverable 11, “Conclusions of Project NEMESYS,” 05/KT/AS/DS/B/028/b1, Dec.1992.

[11.10] RACE R2059 ICM Deliverable 3, “Selection of Networks, Network Interfaces and Definition of Testbed Laboratory,” R2059/VTT/TEL/DS/R/004/b1, Jim Reilly, Juha Koivisto, editors, December 1992

[11.11] RACE R2081 TRIBUNE, Deliverable 22, “Final TRIBUNE B-UNI Specification,” 2081/BT/DP/DS/S/022/b1, Dec.1994.

[11.12] ATM Forum, “ATM User-Network Interface Specification,” Version 3, Sept.1990.

[11.13] ITU-T Recommendation M.3010 “Principles of a telecommunications management network.”

[11.14] ITU-T Recommendation I.320, “ISDN protocol reference model.”

[11.15] ITU-T Recommendation I.321, “B-ISDN asynchronous transfer mode functional characteristics.”

[11.16] RACE R1022, Deliverable 126-Draft B, “Final Recommendation on Connection Acceptance Control, Usage Monitoring and Validation of Control Schemes,” 22/TGIII&IV/11/DS/P/126/B1, Dec.1992.

[11.17] COST 242, “Multi-Rate Models for Dimensioning and Performance Evaluation of ATM Networks,” Interim Report, June 1994.