Chapter 4

Dynamic Modeling and Simulation

Chapter 4: Dynamic Modeling and Simulation

This chapter gives topological models for cell flow and characteristics of the modeled ATM network. Initially, simple models will be studied with a gradual evolution of more complex topology. It also provides the specification of the component blocks that will be used for dynamic simulation. Based on the specifications, suitable algorithms were designed.

4.1 Topological Models for Cell Flow

4.1.1 Single Hop Topology

The single hop topology has one switch that connects two pairs of source-destination. Cells are flowing from input port to output port directly.

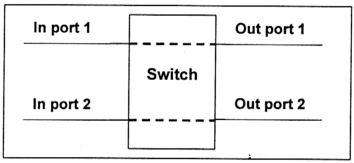

Figure 4.1 Port 1 to Port 1, Port 2 to Port 2

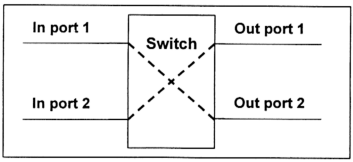

Figure 4.2 Port 1 to Port 1, Port 2 to Port 2

4.1.2 Multi Hop Topology

For multi hop topology, the cells flow through two switches that are connected to each other.

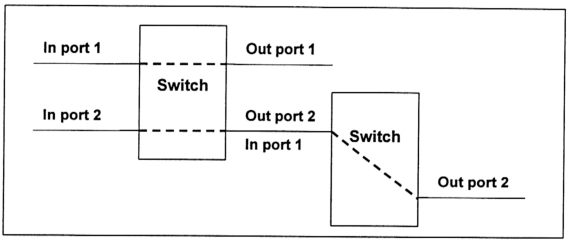

Figure 4.3 Two switches with one connection

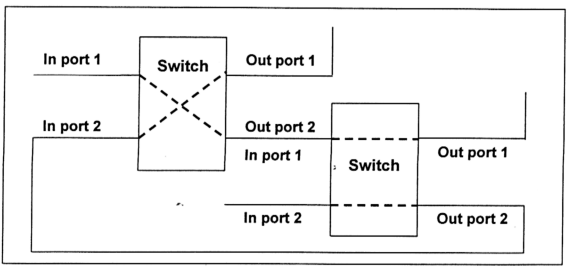

Figure 4.4 Two switches with connection to each other

4.1.3 Ring Topology

Ring topology consists of three switches that connected to each other, three types of traffic flow through switches.

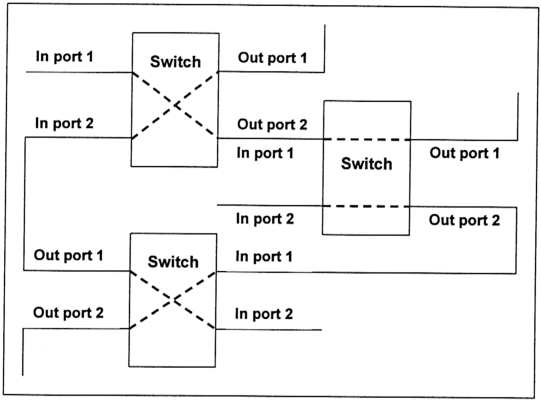

Figure 4.5 Three switches connected to each other

4.1.4 Mesh Topology

Mesh topology consists of four switches that connected to each other, two types of traffic flow through switches.

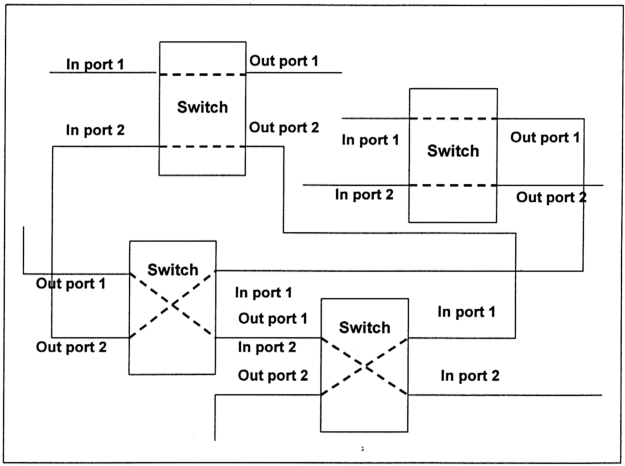

Figure 4.6 Four switches connected to each other

4.2 Characteristics of Modeled ATM Network

1. The ATM network simulation will be a discrete time-driven simulation. The system state will be changed every unit sample time (ticks).

2. ATM network components will be represented by block in Simulink. A model can be constructed by putting various blocks together under Simulink environment. There must be connection between blocks in order to interacting blocks. The simulation can be executed only after the model has been built.

3. The connection simulated is a permanent virtual connection, which means the connection is unchangeable for a particular network model.

4. The switch simulated is 2x2. Input buffering is applied in switch.

5. Cell format: |Source|Traffic_type|Destination|. The cell fields will be multiplexed before sending, and demultiplex as soon as they are received by the receiver. This is caused by the inability of Simulink to pass a combination of signals. Simulink will only pass numeric signal, which means that all the data (signal) will have to be in numeric form.

6. Three types of traffic source are simulated: CBR (represented by 1), VBR (represented by 2), and ABR (represented by 3).

All the ATM component blocks are built using S-function. This is because the existing Simulink blocks are not suited for the ATM network simulation environment.

4.3 Implementation Issues

An S-function can be written as C-MEX file or M-file. C-MEX file is written using C language and M-file is written using MATLAB. C-MEX file S-function is adopted in creating ATM component blocks.

4.3.1 Overview of C MEX S-Function Routines

All S-functions must conform to a particular format as shown below:

```
#define S_FUNCTION_NAME sfunction_name
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

static void mdlInitializeSizes (SimStruct *S)
{
}

<additional S-function routines/code>

static void mdlTerminate (SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE      /* Is this file being compiled as a MEX-file? */
#include "simulink.c"       /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"        /* Code generation registration function */
#endif
```

Figure 4.7 General Format of a C MEX S-function *source: [SFCN98]*

The data structure, *SimStruct* is for Simulink to maintain information about the S-function. Macros that are provided to access the *SimStruct* will be discussed later. There are several *mdl** routines within an S-function and different S-functions can be implemented using routines of the same name but different contents. These routines are called in certain sequence as shown below:

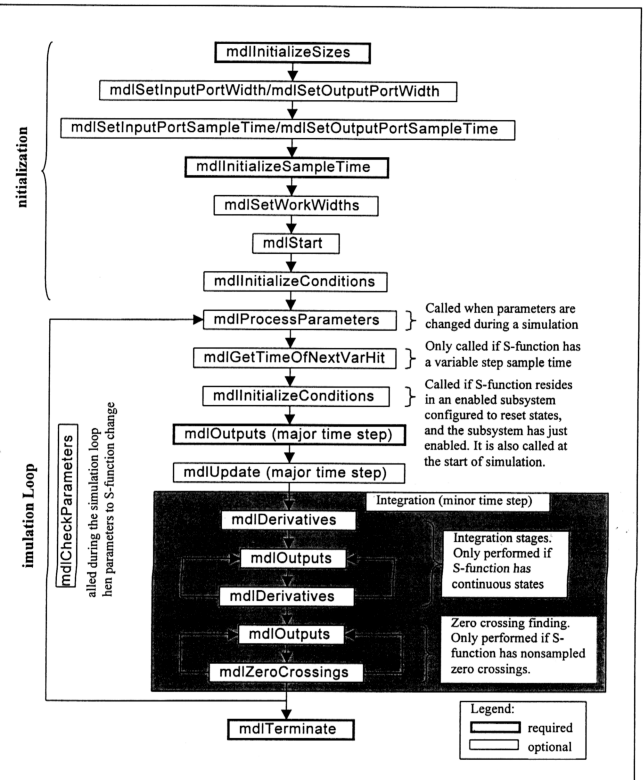

Figure 4.8 The Calling Sequence of S-functions *source: [SFCN98]*

4.3.2 Data View of S-Functions

S-functions blocks have input and output signals, parameters, internal states and other general work areas. Generally, block inputs and outputs are written to, and read from a block I/O vector. Parameters can be passed to S-function blocks using the S-function block dialog box. The following picture shows the general mapping between these various types of data:

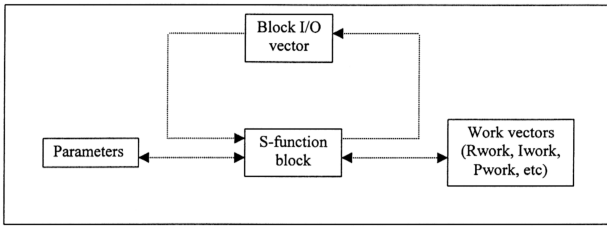

Figure 4.9 The Relationships Among S-Functions and S-Function Data

The length of the various signals and vectors can be configured in the *mdlInitializeSizes* routine. The signals as well as their length can be accessed in S-function routines that are called during the simulation loop.

4.3.3 S-Function Parameters

Parameters can be provided to S-functions using S-function parameters field of the block's dialog box. The order and number of the parameters must be determined when creating S-function. The order of parameters when specified in block's dialog box must be identical to the order determined in S-function. Number of parameters can be defined in the *mdlInitializeSizes* routine using *ssSetNumSFcnParams* macro. The parameter values can be accessed using *ssGetSFcnParam* macro.

4.3.4 Defining S-Function Block Characteristics

The *sizes* structure in the *SimStruct* stores essential size information about the S-function block including number of input ports, output ports, work vectors, and other block characteristics. The *sizes* structure is initialized in the *mdlInitializeSizes* routine. Values for the structure fields are set by various macros supplied. If the value is not specified, it is initialized to zero.

4.3.5 Setting Sample Time

Simulink supports blocks that execute at different rates. There are two methods by which rates can be specified:

- Block-based sample time – more than one sample time can be specified, but the inputs and outputs will be processed at the fastest rate specified
- Port-based sample time – the sample time is specified for each individual input and output port, and will be used if the application requires unequal sample rates for input and output execution

4.3.6 Configuring Input and Output Port Properties

The port width needs to be specified as a positive nonzero integer. The port width means number of signals for a particular port. It is set after the number of ports is specified in the *mdlInitializeSizes* routine. If the sample time is port-based, then the port sample time needs to be specified also.

4.3.7 Configuring Work Vectors

S-function work vectors are used instead of static or global variables if the S-function requires persistent memory storage. Static and global variables are used by multiple instances of an S-

function. This occurs when there are multiple S-function blocks in a Simulink model and the same S-function C MEX-file has been specified. This will cause the S-function to produce undesired outcomes. By using work vectors, Simulink manages persistent storage locations for an S-function. The number of elements in each vector can be specified dynamically as a function of the number of inputs to the S-function. There are several advantages of work vector:

- Instance specific storage for block variables
- Supports integer, real, pointer, and general data types
- Elimination of static and global variables and the associated multiple instance problems

4.3.8 Memory Allocation

It is possible that available work vectors do not provide enough capability when creating an S-function. In this case, memory needs to be allocated for each instance of the S-function. For C MEX S-functions, memory allocating is done by the *stdlib.h – calloc()* for allocating memory and *free()* to free the memory.

4.3.9 SimStruct Macros in use

Table 4.1 SimStruct Macros and Descriptions

Marco	Description
SsSetNumInputPorts	Set the number of input ports to a nonnegative integer.
SsSetInputPortWidth	Specify width for each input port index starting at 0.
ssSetNumOutputPorts	Set the number of output ports to a nonnegative integer.
ssSetOutputPortWidth	Specify width for each output port index starting at 0.
ssGetInputPortWidth	Get the width of an input port.
ssGetInputPortRealSignalPtrs	Access to an input port signal.
ssGetOutputPortWidth	Get the width of an output port.
ssGetOutputPortRealSignal	Access to an output port signal.

ssSetNumSFcnParams	Set the number of S-function parameters.
ssGetNumSFcnParamsCount	Get the number of parameters entered in the block dialog box.
ssGetSFcnParam	Access to a parameter entered in the block dialog box.
ssSetNumSampleTimes	Set the number of S-function sample times that is greater 0.
ssSetNumRWork	Specify the number of real_T work vector elements.
ssSetNumIWork	Specify the number of int_T work vector elements.
ssSetNumPWork	Specify the number of pointer (void *) work vector elements.
ssGetRWork	Get the real_T work vector.
ssGetIWork	Get the int_T work vector.
ssGetPWork	Get the pointer (void *) work vector.

4.3.10 Buffer/Queue

The buffers are implemented using integer type arrays. The array size is the same as the *buffersize* parameter entered by the user measured in cell. The buffer adopted FIFO (First-In-First-Out) in its queueing system. A counter is used to maintain the current buffer filled size. For each incoming cell, the counter will increase by 1 and decrease by 1 for each outgoing cell.

4.3.11 Link/Bandwidth

The links are implemented using output port width of an S-function. The output port width is the same as the *outlink* parameter entered by the user measured in cell in which each unit of output port width represents one link slot. At the beginning of each sample time, all the units of output port width are initialized to zero value indicates that all the link slots are empty. Then, cells from buffers will start to fill the link slot as long as there are link slots available.

4.4 ATM Component Blocks

The list below provides the specification of the custom designed components by using S-function.

I. cbrsource - ATM application that generate constant bit rate traffic

- one cbrsource will be paired with one destination only
- a cbrsource are attached to one BTE only
- three parameters are required from the user:
  1) Bit rate measured in Mbits per second
  2) Total bits to be sent are measured in Mbits
  3) Name of destination BTE – must be a number between 0-99

II. vbrsource - ATM application that generate variable bit rate traffic

- one vbrsource will be paired with one destination only
- a vbrsource are attached to one BTE only
- three parameters are required from the user:
  1) Maximum bit rate measured in Mbits per second
  2) Total bits to be sent are measured in Mbits
  3) Name of destination BTE – must be a number between 0-99
- one input from Random Number block by which the mean and variance of variable bit rate can be determined by the user

III. abrsource - ATM application that generate available bit rate traffic

- one abrsource will be paired with one destination only
- an abrsource are attached to one BTE only
- three parameters are required from the user:
  1) Bit rate measured in Mbits per second
  2) Total bits to be sent are measured in Mbits

3) Name of destination BTE – must be a number between 0-99

IV. destination - ATM application that receive traffic from source

- one destination will be paired with one source (either CBR, VBR, or ABR) only
- one destination are attached to one BTE only
- no parameters are required from the user
- two parameters can be displayed on screen:
  1) Name of source BTE where cells are received from
  2) Total number of cells received

V. bte - Broadband Terminal Equipment

- one BTE can only be attached to one source (either CBR, VBR, or ABR) at input side
- the output side can be attached to a switch or destination
- three parameters are required from the user:
  1) Name of BTE
     - if BTE input side is attached to a source then the name must be number between 500-599
     - if BTE input side is attached to a destination then the name must be a number between 0-99
  2) Buffer size of queues are measured in number of cells
  3) Bandwidth for output link measured in Mbits per second
- three parameters can be displayed on screen:
  1) Output link capacity and utilization of the link at that particular time measured in number of cells
  2) Number of cells dropped
  3) Number of cells received from source or switch
- 2 queues are maintained: CBR/VBR queue and ABR queue

VI. atmswitch - ATM 2x2 switch

- the switch has two input ports and two output ports
- the input ports can either be attached to source BTEs or switches
- the output ports can either be attached to destination BTEs or switches
- five parameters are required from the user:
  1) Name of switch – must be a number between 800-899
  2) Name of component attached to first output port, either a destination BTE or another switch
  3) Name of component attached to second output port, either a destination BTE or another switch
  4) Buffer size of queues measured in number of cells
  5) Bandwidth for output link measured in Mbits per second

- five parameters can be displayed on screen:
  1) Output link capacity and utilization of the link at that particular time measured in number of cells
  2) Number of cells dropped at first output port
  3) Number of cells received at first output port
  4) Number of cells dropped at second output port
  5) Number of cells received at second output port

- two queues are maintained for each output port: CBR/VBR queue and ABR queue

VII. link - physical link

- no parameters are required from the user
- two parameters can be displayed on screen:
  1) Link capacity measured in cell
  2) Link utilization at a particular time

4.5 Algorithms

The algorithm adopted is based on the specification of the component as given in section 4.4.

I. cbrsource

- the outport width will be determined by bit rate parameter, where

  *output port width = (Mbits persecond * 1024) / (48 * 8)*

  , which is actually a cell rate value
- total cells to be sent will be calculated by

  *total cells to be sent = (total bits to be sent * 1024) / (48 * 8)*
- except the last sending, for each sample time (which is set to 1), all the output ports will be filled with cells which means that the cells are sent in a constant rate
- for last sending, the output port may not full with cells, depending on the number of cell left to be sent which is normally less than the cell rate
- for output port that is not sending any cell, a zero value will be filled

II. vbrsource

- the outport width will be determined by maximum bit rate parameter, where

  *output port width = (Max. Mbits persecond * 1024) / (48 * 8)*

  , which is actually the maximum cell rate value
- total cells to be sent will be calculated by

  *total cells to be sent = (total bits to be sent * 1024) / (48 * 8)*
- for each sample time (which is set to 1), the number of cells to be sent will be calculated based on the mean and variance value of variable bit rate, but it must not exceed the maximum cell rate
- for output port that is not sending any cell, a zero value will be filled

III. abrsource

- the outport width will be determined by bit rate parameter, where

  *output port width = (Mbits persecond * 1024) / (48 * 8)*

  , which is actually a cell rate value
- total cells to be sent will be calculated by

  *total cells to be sent = (total bits to be sent * 1024) / (48 * 8)*
- except the last sending, for each sample time (which is set to 1), all the output port will be filled with cells which means that the cell are sent in a constant rate
- for last sending, the output port may not full with cells, depending on the number of cell left to be sent which is normally less than the cell rate
- for output port that is not sending any cell, a zero value will be filled

IV. destination

- accumulate cells received from BTE

V. bte

- the outport width will be determined by output link parameter, where

  *output port width = (Output link * 1024) / (48 * 8)*

  , which is actually output link capacity in cell rate
- for each incoming cell,
  - if buffer is full, then drop the cell
  - else if buffer is not full, check the traffic type,
    - if it is cbr (=1) or vbr (=2), then go to first queue
    - else if it is abr (=3), then go to second queue
- upon sending outgoing cells, the first queue will be given priority, only when the first queue is emptied, the cells in second queue will be sent
- amount of cells can be sent per sample time are based on the output link capacity
- for output port that is not sending any cell, a zero value will be filled

VI. atmswitch

- the outport width will be determined by output link parameter, where

  *output port width = (Output link * 1024) / (48 * 8)*

  , which is actually output link capacity in cell rate
- for each incoming cell at first input port,
  - if buffer is full, then drop the cell
  - else if buffer is not full, check the traffic type,
    - if it is cbr (=1) or vbr (=2), then go to first queue of first port
    - else if it is abr (=3), then go to second queue of first port
- for each incoming cell at second input port,
  - if buffer is full, then drop the cell
  - else if buffer is not full, check the traffic type,
    - if it is cbr (=1) or vbr (=2), then go to first queue of second port
    - else if it is abr (=3), then go to second queue of second port
- upon sending outgoing cells,
  - for first switch output port, the priority will be
    - cbr/vbr queue at first port
    - vbr queue at first port
    - cbr/vbr queue at second port
    - vbr queue at second port
  - for second switch output port, the priority will be
    - cbr/vbr queue at second port
    - vbr queue at second port
    - cbr/vbr queue at first port
    - vbr queue at first port
- four possibilities are considered:

  1) both switch outport are attached to destination BTE, all cells will be checking their destination at first destination, then at second destination

2) first switch output port is attached to a destination BTE, second switch output port is attached to another switch, all the cells will be checking their destination first before being switch to next switch
3) first switch output port is attached to another switch, second switch output port is attached to a destination BTE, all the cells will be checking their destination first before being switch to next switch
4) both switch outport are attached to another switches, all cells will be switched to first switch, if the first switch is full they will go to second switch

- amount of cells for each switch output port that can be sent per sample time are based on the output link capacity
- for output port that is not sending any cell, a zero value will be filled

VII. link

- get the link capacity from output port width of component which link is attached to its output side
- get the link utilization by calculating number of output ports which carry non-zero value