.- Chapter 4: STATE OF THE ART

4.1.- Functional Blocks

The next figure 4.1. shows the Functional Blocks (FB) developed into OPNET simulation software. Each FB is related to a specific block of the Phoenix basic chain:

1) The **Source Terminal** is the source of the video bit-stream encoded and eventually ciphered and protected by UEP. Moreover, it implements the JSCC application layer controller.

2) The **IP Network** FB simulates the effect in terms of delay and loss on IP packets by an IP cloud (the Internet). Also, this FB allows to simulate the effect of a bottleneck in the wired part on the packet flow. Depending on the simulation scenario, an IP cloud can be both at the sender side (from Source Terminal to the radio transmitter) and at the receiver side (from the radio receiver to the destination terminal).

3) The **TX Radio** and the **RX Radio** FBs represent the wireless transmitter and the wireless receiver respectively. From the Opnet point of view, these FBs have been modeled with the same FSM (considering the upstream packet flow a TX Radio is also a wireless receiver while a RX radio is also a wireless transmitter). TX/RX node is able to simulate both a WiFi interface and an UMTS interface. The FSM will be a wireless receiver in case of an incoming packet the wireless interface while it will be a wireless transmitter in case of an incoming packet from a wired interface (to be forwarded to the wireless interface). They implement the physical wireless functionality (both WiFi and UMTS technology) and the JSCC physical layer controller.

4) The **Receiver Terminal** is the destination of the video flow and implements the JSCC application layer controller.

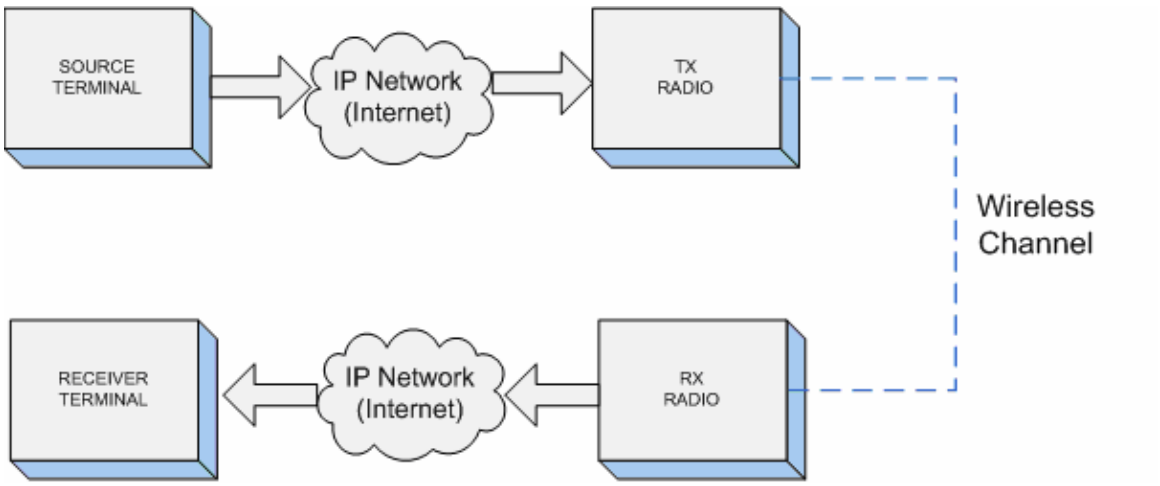

Figure 4.1 : Functional Blocks

In the following paragraphs we will explain in detail how these nodes have been modeled into OPNET simulator. A description of the state machine, the parameters-mask and the collected simulation statistics will be given for every node. Some parameters-mask can be not strictly related to a specific FB node but can be general for all the systems. These parameters can be defined in a custom utility node named

CONFIG F.B.(Figure 4.2) It is a simple configuration node that allows sharing general parameters between FBs.

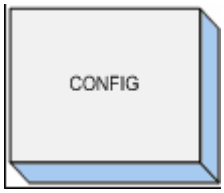

Figure 4.2 : CONFIG FB

4.2.-Config F.B.

The CONFIG FB is a repository for all the general parameters that need to be shared with all FBs.

4.2.1 Node Model

The node model of the configuration FB (Figure 4.3) is very simple and it is composed by a single process called CONFIG_PROC.

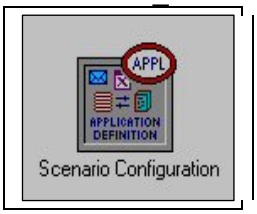

Figure 4.3 : CONFIG Node Model

4.2.2 Parameters

Table 4.1 contains the list of the parameters that can be configured into the CONFIG_PROC model.

Control/signaling information can be transported in several ways: by a well-know protocol (e.g. ICMP v6, RTP/RTCP), by an ad-hoc protocol, as well as encapsulated in IP packets (e.g. data payload and extension header). The first and the second ways are referred as out-of-band signaling while the last way as in-band signaling. In case of control/signaling information generated by the video source node, the control/signaling encapsulation could be carry out inside the related video data packet or inside the previous one in order to provide to the concerned nodes the control/signaling information in advance. Moreover, In the case of a well-know or an ad-hoc protocol, the whole control/signaling information related to a video packet could be carry out by a single control/signaling packet or by multiple packets (e.g. a packet for each SSI). The config FB defines also the format of control/signaling information in terms of bit-size and in terms of frequency (e.g. send signaling every packet or every defined timeout). Table 4.1. contains the default value of size and frequency.

In the following simulations we have used the schemes that have been selected from the previous simulations because it gives us the best performance. They are the following:

SSI e SRI: These two signals have a very similar nature, so we have considered them together. Both signals must be sent synchronized with the video frame at which they refer to. So it is clear that they must to be encapsulated in the same frame that

contains the multimedia flow. Since SSI and SRI is used in different points of the Network (particularly in the Channel Coder/Decoder and in the Destination Node), the more suitable Header Extension is the hop-by-hop extension. The added overhead is related to video code (average rate), in the sense of each packet is sent with this information.

CSI: This information is sent periodically by a programmed timer. There are not especial requirements about synchronization so the possible mechanisms are ICMPv6 and IP ad-hoc. There is not a flow to encapsulate this information because it is a feedback information, so we have decided to use ICMPv6 that offers a good result with a reduced overhead

NSI: This information is periodically created as well, therefore ICMPv6 packets are a good mechanism. If time requirements are high, we can use RTP/RTCP streams, which can assure QoS inside the IP Network. But thinking about the overhead, is better ICMPv6.

SAI: This type of information is created in video Decoder at Destination Node and analyzed at Radio Channel Receiver. Despite it travels in opposite direction from video flow, is close related to received and decoded frames. This information has the same frequency and dimensions as received video, so it is thought to generate a new IP flow from the Destination Node to the Radio Receiver. It can be used an ICMPv6 flow too, or encapsulate the information with IP hop-by-hop extension if an opposite data flow existed, but it would be problems with MTU dimensions.

DRI: DRI information is related to "fuzzy" decoding or "soft" type, and contains information about the certain value given by the Channel Decoder to a particular bit. This information is the completion or the substitution of the bit value. The flow must be matched to the transmitting video frame. Since the encapsulation is not possible due to the dimensions of this information, it has been thought the possibility of insert an ICMPv6 flow, in which insert also the Timestamp or the frame index of the information that is referring to. We must control the impact on the efficiency between Radio Receiver and Video Decoder that it can cause because of its dimension. If Receiver and Destination agree, the problems will be minor.

Table 4.1 : Parameters mask of the CONFIG node

Parameter Name	Description	Default Values
Control/signaling Scheme	Define how the signaling will be sent out for each control/signaling information: - Standard Protocol (e.g. ICMPv6) - Ad-hoc protocol (e.g. RTP/RTCP) - IP packet (payload) IP extension headers - hop-by-hop	**SSI:** -scheme:IPv6 header encap hop-by-hop -clustering:0 -pre-encapsulation: current data packet -code_len(bits): 3 -size_len(bits): 13 **SRI:** -scheme: IPv6 header encap hop-by-hop

	Destination	-clustering: 0 -pre-encapsulation: current data packet -size_len: 16 **SAI(priori):** -scheme:promoted -clustering: promoted -pre-encapsulation: current data packet -size: 4 **CSI:** -scheme:stdICMPv6 -clustering: single packet -pre-encapsulation: current data packet -size: 64 -timer: 1 (sec) **DRI:** -scheme:std ICMPv6 -clustering: promoted -pre-encapsulation: current data packet -size: 4 -dri_ovrhead: 4 **NSI:** -scheme: stdICMPv6 -clustering: single packet -pre-encapsulation: current data packet -size:256 -timer:1 sec
Clustering of control/signaling packets	Define if the control/signaling information related to a video packet will be sent by a single packet or by multiple packets. No valid if the control/signaling information is encapsulated into video packet.	Single packet Multiple packets
Control/signaling pre-encapsulation	Enable to send the control/signaling information related to the n-nth packet inside the (n-1)-nth packet. . Only valid if the control/signaling information is encapsulated into video packet.	No

Control/signaling information size for each type	Size in bytes of each control/signaling information type	SSI: size 16 bit (3 code, 13 size) frequency (every video packet) CSI: size 4byte,frequency 200msec
code_rate	A list of compound attributes, where each compound attribute has two fields (*code* and *rate*), one defining the compression identifying code and the other the channel code rate.	4/5
header_compress_rates	A list of compound attributes, where each compound attribute has two fields (*code* and *rate*), one defining the compression identifying code and the other the header compression rate.	NO
umts_mac_header_size	Size in bytes of UMTS MAC frame header	40
header_compression_activate	Defines if the IPv6 header compression is activated at the RX/TX level	NO
Scenario configurator.node_desc	A description of JSCC messages timeout (in terms of packets or seconds) and node type for all nodes present in the scenario. Each row represents a node in the scenario. At each row node type values are coded as follows: • 0= JSCC source node • 1= JSCC destination node • 2= non JSCC source node • 3= non JSCC destination node • 4=RX node • 5=TX node	0
Scenario configurator.SSI/channel coding	Associate at each SSI value a coding rate. At present we have only two coding rate values	4/5

	used in the simulation of the wireless interface are : 1/3 and 4/5	
Use App Controller	Indicates if the Application Controller is Enabled or Disabled	Enabled
App Cntr GOV	Assigns an initial GOV value. This parameter is used to get the correct input file	15
App cntr state	Differs between the 7 possible states of the Application Controller. It is used to get the correct input file too.	2
Appl cntr cycle	It is the period of time in which are applied all parameters calculated by the Application Controller at the beginning of it.	1000 (msec)

4.2.3 Statistics

No statistics will be captured on this node.

4.2.4 Finite State Machine

Also the FSM of the configuration node is very simple. It consists in a single INIT state as depicted in Figure 4.4. Simulation enters in the INIT state only one time, at simulation start.

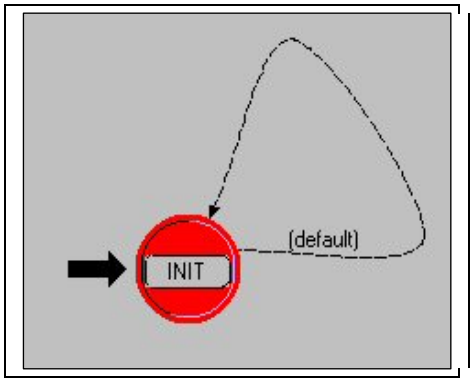

Figure 4.4 : CONFIG FB FSM

The state performs the following tasks:

- **INIT**: load into the related global variables the values defined by the parameters mask.

4.3 Simple Source F.B.

From the point of view of data traffic this FB is responsible to generate data packets sent on the network. Two source models have been developed in the OPNET simulator: a simple source and a JSCC compliant source.

The simple source realizes a traditional video source with the only feature to import a video trace file and generate the related flow. The simple source doesn't receive any upstream traffic.

The JSCC compliant source generates a data stream according to the information extracted from the above described *Video Stream Input File.* This process generates the video flow and also the JSCC signaling, especially the SSI and SRI information. Moreover, the JSCC source has to be able to receive and process the incoming JSCC signaling. In the following, a detailed description of the two source model is given.

4.3.1 Node Model

Figure 4.5. shows the node model of the Source Terminal node. It is composed by a Source Coder process linked to an OPNET standard transmitter process (source_tx) and to a standard receiver process (source_rx).

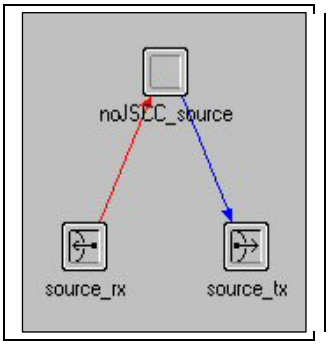

Figure 4.5 : Simple Source Terminal Node Model

4.3.2 Parameter mask

Every node in the network (not the CONFIG node) needs to be univocally identified in order to send and receive correctly both data and control packets. A parameter mask defines the node identifier (ID) parameter that simulates the IP address of a real network. When a node generates an IP packet, it stores its ID into the IP packet header (IP source address) and the ID of the destination node into the IP destination address. Intermediate nodes (like router in the IP cloud or wireless transmitter connected to multiple receivers) can therefore route packets basing on their routing table. The simple source can be configured to import any available video trace file formatted with a packet defined for every line. Any entry in the video trace file must include the timestamp and packet size information. A parameter mask allows the user to set the video trace file name that need to be imported by the simple source.

The destination node of the video flow is a simple destination model. The source selects its destination by a parameter mask indicating the destination ID of the target node.

The source defines also its allocation on the wireless channel. For Wi-Fi it uses a QoS level (as type of service) while for UMTS the source chooses the CDMA code rate defined in kbps.

Table 4.2 summarizes the parameters mask.

Table 4.2 : Parameters mask of the Simple Source node

Parameter Name	Description	Default
IP source int	ID of this node	
Video Trace file	Name of the video trace file to be imported	
IP destination int	ID of the destination node	
Wireless Service Option	Define the QoS required by this source on a wireless link: in case of Wi-Fi this means a QoS level in a shared channel while in case of UMTS this means to set a WCDMA coding rate on the channel to this source	Wi-Fi: 0 = best effort UMTS: 64kbps

N.B. This node has not been used in the simulations, because of this, the parameter masks have not been allocated.

4.3.3 Statistics

Statistics on this node are related to the video data flow generated at IP and application layer.

Table 4.3 : Simple Source Statistics

Statistic Name	Description
IPv6 Packet Sent (pck/sec)	Rate of the IPv6 packet sent
IPv6 Traffic Sent (byte/sec)	Throughput of the IPv6 packet sent
Application Packet Sent (pck/sec)	Rate of the application packet sent
Application Traffic sent (byte/sec)	Throughput of the application data sent

4.3.4 Finite State Machine

The FSM of the Simple source is composed by two states: an INIT state processed only at simulation start and an IDLE state processed every time an event happen on this FSM.

- **INIT**: initialization of the Simple Source process:
  - Loading parameter mask values into state variable
  - Loading video trace file into a memory structure suitable for the processing
  - Initialization of the statistics
  - Scheduling of the first event that will happen on this FSM (the fist packet that will be sent according to the first timestamp)
- **IDLE**: this state is the core of the Simple Source. When a packet needs to be sent out according to the timestamp of the loaded video trace file, the FSM creates an IPv6 packet encapsulating a video data packet with the indicated size.

In the IPv6 header, the IPv6 destination and source address are then updated in function to the source and destination node ID

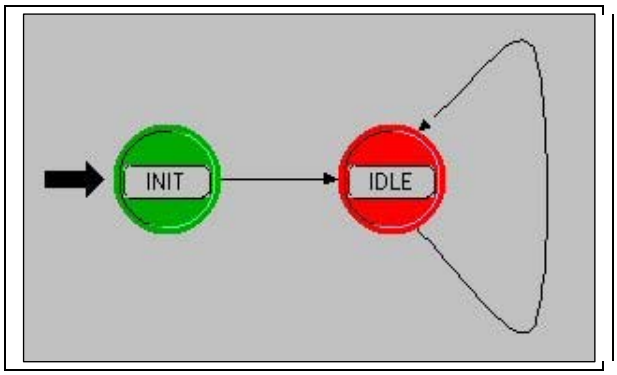

Figure 4.6 : Simple Source FSM

4.4 .- JSCC/D Source F.B.

This node includes the joint code system and the Application Controller. In the following paragraphs we will explain in more details this complex node.

4.4.1.- Node Model

Figure 4.7. shows the node model of the JSCC/D Source terminal node. Like previous Simple Source, it is composed by a Source Coder process linked to a standard transmitter process (TX) and to a standard receiver process (RX).

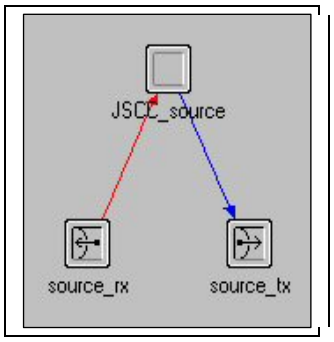

Figure 4.7 : JSCC/D Node Model

4.4.2 Parameter mask

We have a set of *Video Stream Input File* related to different codec and rate. We need therefore a mask parameter to select the Video file at simulation start. A possible way to implement this could be to have a structure that ties the codec/rate specification with the Video Stream Input File. The structure has been realized in order to have the possibility to set a lot of video parameter: video codec (MPEG4, H264...), video size (QCIF and CIF), frame rate (fps) and codec rate (kbps).
The video file could change during the simulation due to the received control/signaling information like for example an incoming CSI informing of a change in the channel condition. The JSCC Source needs therefore a list of all available video file classified yet in terms of the above video parameters.

The file contains timestamp information used to generate the packets. It could be useful to set the source to begin the packet delivery after a predefined interval time. A parameter mask has been introduced to set the start-time. The start-time will be the simulation time correspondent to the dispatching of the first video packet. If start-time is set to 0 then the start time of the packet will be exactly that specified for the first packet of the video input trace file.

In the JSCC/D chain, the source is responsible also to cipher the source data. For the simulation scope, the ciphering can be expressed in terms of overhead percentage added to the packet according to realistic value of existing ciphering algorithm. If the source ciphering is applied at application layer, it could affect only a part of the data (for example according to the SSI values). Hence, it is useful a parameter that indicates the level of ciphering: it has been defined as an integer value indicating the last size field that will be ciphered (e.g.: a value of N means that size1….sizeN will be ciphered and sizeN+1….. will not be ciphered).

Similar to the ciphering case, a UEP overhead can be specified according to realistic value of existing UEP algorithm.

Since the video streaming could be unicast or multicast, a parameter indicating the list of destination node allows to set also more than one destinations.

Finally, similar to the Simple Source process, the JSCC source defines its allocation on the wireless channel. For Wi-Fi it uses a QoS level (as type of service) while for UMTS the source chooses the CDMA code rate defined in kbps.

Table 4.1. shows all the parameters mask of the JSCC/D source.

Table 4.4 : Parameters mask of the JSCC Source Terminal FB

Parameter Name	Description	Default Value
IP source int	ID of this node	1
Codec name	The name of the source video codec	MPEG4
Video size	Size of the source video	CIF
Frame rate	Rate of the frame sent in fps	30
Codec rate	Rate of the source video in kbps	370
Codec Choice Threshold	Mapping between the channel state (CSI) and the codec to be used	
Start-time	The time the first video packet will be sent	0
Ciphering overhead	Added overhead percentage to realize the ciphering	20%
Level of ciphering	Indicate the size field that are ciphered	Significance 4
UEP overhead for significance	Added overhead percentage due to the UEP for each SSI field	UEP mode Off
Destination(s)	Receiver or list of receivers of the video stream	192.168.20.1
Wireless Service Option	Define the QoS required by this source on a wireless link: in case of Wi-Fi this means a QoS	Wi-Fi: 0 = best effort UMTS: 64kbps

	level in a shared channel while in case of UMTS this means to set a WCDMA coding rate on the channel to this source	
BER Threshold	Maximum value for BER (Bit Error Rate)	0.3
PLR Threshold	Maximum value for PLR (Packet Loss Ratio)	0.05
MTU	Maximum Size of the Transfer Unit, in bytes.	1500

4.4.3 Statistics

The JSCC source collects throughputs at IP layer and at application layer related to the video traffic flow sent on the network and control/signaling traffic sent and received (signaling overhead). Table 4.5. shows all the simulation statistics collected by this process.

Table 4.5 : JSCC/D Source Statistics

Statistic Name	Description
Application Current File Index	Level of codification for the Application Controller. Possible values from 1(minimum)to 6 (maximum)
Application ciphering overhead	Overhead added because of the ciphering
Application UEP overhead	Overhead added because of UEP
CSI BER value	BER value received in CSI packets along the time
CSI packet received (pck/sec)	Rate of the CSI packet received
Payload size before uep/ciphering	Size of the payload before applying uep/ciphering
Payload size after uep/ciphering	Size of the payload after applying uep/ciphering
IPv6 Traffic Sent (byte/sec)	Throughput sent at IPv6 layer
IPv6 Packet Sent (pck/sec)	Rate of the IPv6 packet sent
IPv6 Packet Received (pck/sec)	Rate of the IPv6 packet received
IPv6 Traffic Receive (byte/sec)	Throughput received at IPv6 layer
Application Packet Sent (pck/sec)	Rate of the Application packet sent
Application Traffic Sent (byte/sec)	Throughput sent at application layer
Total Control/Signaling Traffic Received (byte/sec)	Total Throughput of control/signaling information received
Total Control/Signaling Packet Received (pck/sec)	Total packet rate of control/signaling information received
Total Control/Signaling Traffic Sent (byte/sec)	Total Throughput of control/signaling information sent
Total Control/Signaling Packet Sent (pck/sec)	Total packet rate of control/signaling information received
SSI/SRI Traffic Sent (byte/sec)	SSI/SRI Throughput sent

CSI/NSI Traffic Received (byte/sec)	CSI/NSI Throughput received
Control/signaling information overhead	Relative overhead of the control/signaling information with respect to the sent data (counting the generated control/signaling only, or also the received one)
PSNR estimated	Value of PSNR calculated in Application Controller Algorithm, based on PER, CSI, BER
PLR received	Estimation of PLR based on NSI packets received

4.4.4 Finite State Machine

Figure 4.8. shows the JSCC/D Source Terminal node FSM. The FSM has mainly two procedural flows: the first one related to an outgoing packet and the second one related to an incoming packet (control/signaling feedback). When a video packet needs to be sent, the FSM creates the IPv6 packet with the encapsulated video information. The source sends also (if any) the JSCC/D control/signaling information according to the configured scheme. On the other side, when the source gets an incoming control/signaling packet from the network, it manages the packet according to the current JSCC/D application layer controller policies.

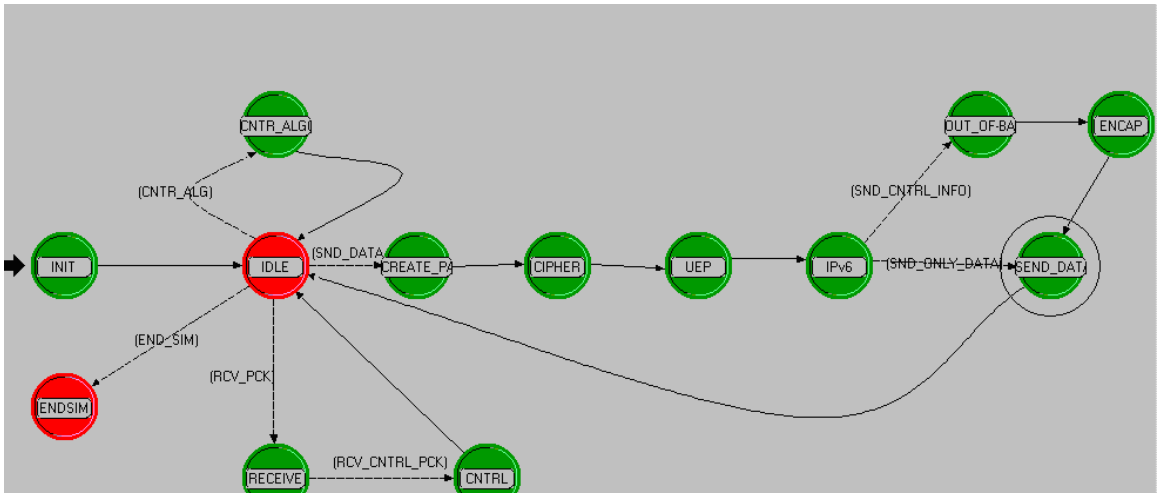

Figure 4.8 : FSM of the JSCC/D Source

In the following bullets, an exhaustive description of the task performed by each state is given.

- **INIT**: the init state initializes all the data structure that will be used during the simulation into the FSM. Especially the following actions been carry out:
  - Load the data from the parameter mask into internal static variables
  - Initialize the simulation statistics
  - Open the related Video Streaming Input file according to the value set in the parameter mask
  - Generate the first event (a self event).The first event of the Source Terminal will be to send the first Video Packet according to the Video

Stream Input File (timestamp, field and related size). The start time parameter mask, if not zero, can delay this event.

- **IDLE**: in the idle state the source does nothing. It waits for an event and when event happen, it moves to the next state. The next state depends on the event type. If the event is an incoming packet the FSM moves to the RECEIVE state while if the event is a transmission of a video packet the FSM moves to the CREATE_PAYLOAD state.
- **CREATE_PAYLOAD**: this state is triggered every time the source terminal has to send a data packet (when the simulation time is equal to the next packet timestamp) or when a signaling packet must be sent. According to the size of the packet, a payload packet is generated. The state moves then to the CIPHER state.
- **CIPHER**: the cipher state applies the ciphering to the just created payload packet according to the value stored the parameter mask. After the ciphering, a ciphered packet is generated with a size evaluated according to the different significance in the packet and the ciphering overhead in the parameter mask. The state moves then to the UEP state.
- **UEP**: this state simulates the behaviour of the UEP mechanism. According to the significance of the current packet and to the value stored in the UEP parameter mask, a packet is generated with the evaluated size.
- **IPv6**: the payload packet has been created, eventually ciphered and protected by UEP. Now it can be encapsulated into an IPv6 packet. The IPv6 state creates an IPv6 packet according to the IPv6 protocol specification and encapsulates the payload packet. If some JSCC/D control/signaling information is related to the current video packet, the state moves to the OUT_OF_BAND state else it moves to the SEND_DATA state.
- **OUT_OF_BAND**: this state checks the parameter mask of the CONFIG node to know if the control/signaling information that need to be carried (related to the current packet) uses an out-of-band mechanism (like sent in an ICMPv6 packet or as payload in a new IPv6 packet). If this is the case, then the new packet is created. The state moves then to the ENCAP state.
- **ENCAP**: similar to the previous state but in this case the state check if the control/signaling information need to be carried by an in-band mechanism (like IPv6 hop-by-hop and destination option header). The state moves then to the SEND_DATA state
- **SEND_DATA**: this state can be triggered both in the case that a control/signaling information has to be carried related to the current video packet and in the case that only the video data packet needs to be sent out. A check on which packets are present at the moment allow to send out all the information. The state moves then to the IDLE state.
- **CNTRL_ALGORITM**: This state does the Application Controller Algorithm. If Application Controller is activated, then this will be executed each control cycle (secs). For more information see the paragraph before [3.6]
- **RECEIVE**: the FSM moves in this state in case of an incoming packet. The JSCC/D source can receive control/signaling information, mainly the CSI (or reduced CSI) and the NSI. After the reception of the packet, the state checks the destination IP address to report eventually some problem on the delivery of the packet. Then it checks if the packet contains control/signaling information. If this is the case, the state moves to the CNTRL state.

- **CNTRL**: this state checks the type of the control/signaling information (CSI or NSI) then extracts the information according to the used mechanism (in-band or out-of-band). The CNTRL state implements also the JSCC/D application Controller functionalities. Basing on the received control/signaling information and the value of some parameters mask (like the codec choice threshold), it can for example change the source video codec or the codec rate. For the simulation this means to load the video source data from a different input file. The states moves then to the IDLE state

4.5.- IP Network F.B.

The IPv6 network is analytically modeled by a configurable number of N router, each one introducing a variable **delay** and a **loss probability**; reasonably, no bit errors or re-ordering occur. For the latest thanks to the minimum allocated resources. In further steps, such an assumption can be relaxed taking into account of possible network failure that imply routing table re-calculation; in this case a further index in the output file should be inserted to uniquely identify a single packet and hence its position in the initially generated sequence. The network receives a data (with possibly control information inside) or a signaling packet from one side and transmits the same entity on the other side, introducing delay and loss; possible replications of the packets can happen in case of multicasting. Congestion on the wired network needs to be simulated: a bottleneck can be represented by two IP clouds concatenated by a link. A FIFO queue manages the transmission on the bottleneck link.

4.5.1 Node Model

Figure 4.9. shows the node model of the IP Network node. It is composed by an IP Network process that elaborates the packets traversing this node. Multiple transmitter and receiver couples represent multiple interfaces.

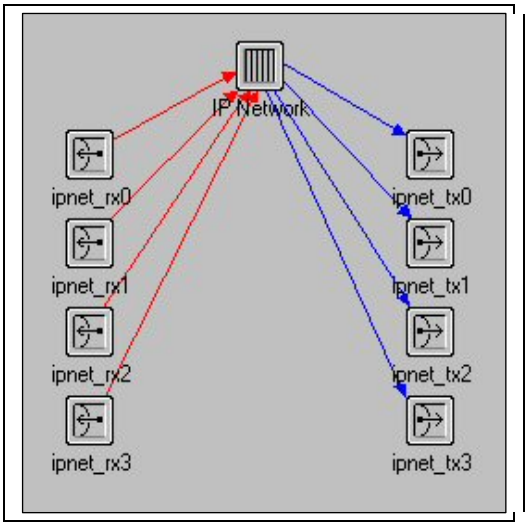

Figure 4.9 : IP Network Node Model

4.5.2 Parameters Mask

The number of the router, the delay and the loss that each one introduces are the main parameters that the user can define on the IP Network Node. The probability loss has been defined in terms of average packet loss on each router while the delay in terms

of average following a defined distribution. The one-way Internet delay is modeled using a shifted gamma distribution with scale parameter α=1. The lower the average delay and loss, the higher the QoS on the network. To consider QoS on the wired network, two default settings are available_ without QoS and with QoS. If an incoming packet carries control/signaling information inside the ipv6 hop-by-hop option header, a further delay due to the processing on each router has to be evaluated.

One parameter indicates if packets traversing the IP Network need tunnelling or not (it depends on the applied IPSec mode).

Another parameter is used to consider the ciphering and deciphering overhead introduced by the network.

To simulate a bottleneck in the wired network a FIFO queue length and a bottleneck rate have been introduced. These attributes effects the simulation only if a bottleneck between two IP network nodes have been defined. An attribute allows also to enable or to disable the FIFO behaviour (that is, enabling or disabling the bottleneck).

Finally, an IP network node can be linked to more than two FBs: for example in the case more sources or destinations (both JSCC and non JSCC). A routing mechanism has to be implemented on the IP network node in order to forward packets to the right port. A static routing table allows mapping destination IP addresses (node IDs) into the right forwarding port.

Table 4.6 : IP Network Parameters mask

Parameter Name	Description	Default Values
Number of Router	Number of router in the IP Network.	10
Delay	Distribution, average and variance of the delay introduced on the forwarded packet by each router	Without QoS - mean 10msec - max 50msec With QoS - mean 3msec - max 10msec
Loss	Probability that a router loss a packet	With QoS = 0.01% Without QoS = 1%
IPv6 option processing delay	A constant defining the processing delay on each router to process the control/signaling information inside the IPv6 hop-by-hop option header (if any).	Uniform (0,0)
Tunneled	Specification of the working mode of Ipsec (tunneled or transport)	Transport
Network ciphering	Overhead due to the ciphering on the network	0%
Network deciphering	Overhead due to the deciphering on the network	0%
FIFO bottleneck	Enable or disable the bottleneck effect on the IP	Enabled

	network	
FIFO queue len (pck)	Maximum capacity of the FIFO queue length	0
Bottleneck Rate (bps)	Available throughput on the bottleneck	2Mbps(%50free)
Static Route Table	A static routing table to forward incoming packets on the right outgoing port. A number of port is associated to a int IP(IP address int,Port)	(1,1) (2,0) (3,0) (4,0) (5,3) (6,2) (7,0)
Subqueue	Capacity of the subqueue	400.000 (bits) Infinity (packets)

4.5.3 Statistics

The relevant statistics are related to the delay and the loss introduced by the network. These statistics are useful also to validate the model behaviour.

Table 4.7 : IP Network Statistics

Statistic Name	Description
Loss Rate (pck/sec)	Loss packet Rate
Traffic loss (byte/sec)	Traffic loss in byte/sec
Packet forwarded (pck/sec)	Forwarded packets Rate (packet that traverse the whole IP Network)
Traffic forwarded (byte/sec)	Forwarded traffic rate in byte/sec
Average Delay (sec)	Average delay of forwarded packets
Average queue length (pck)	Average packet number in the FIFO queue
Average delay in the queue (sec)	Average packet delay in the FIFO queue

4.5.4 Finite State Machine

Figure 4.10. shows the FSM of the IP Network node. The FSM behaviour is different if a bottleneck is active or not on the IP cloud. If it is not active, the FSM apply the loss and the delay to the packet according to parameters mask configuration. On the other hand, if bottleneck is active, the FSM inserts packets (forwarded to the bottleneck link) into the FIFO queue.

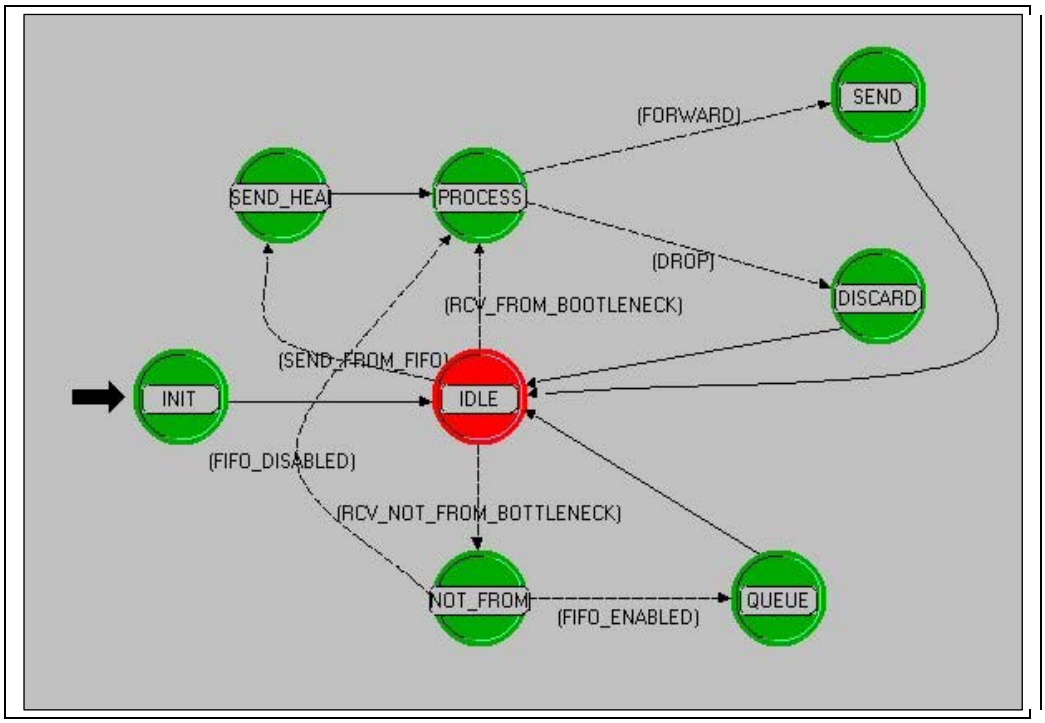

Figure 4.10 : IP Network FSM

The tasks performed by each state are:

- **INIT**: the init state initializes all the data structure that will be used during the simulation into this FSM. Especially the following action will be carry out:
  - Load the data from the parameter mask into static variables
  - Initialize the simulation statistics
- **IDLE:** the idle state wait for an incoming packet. When an incoming packet event occurs, the FSM moves to the PROCESS state if the packet has been received from the bottleneck link or it moves to the NOT_FROM_BOTTLENECK in the other case.
- **PROCESS:** if the packet has been received from the bottleneck, the IP network first checks if the packet has to be dropped. If this is the case the state moves to the DISCARD state. If this is not the case, then the state computes the delay to apply to the packet and schedules the packet transmission. A FIFO scheme allows the packet ordering of delivered packets. The state moves then to the SEND state.
- **DISCARD:** the packet has to be dropped. The state updates statistics and destroys the packet.
- **SEND:** the packet has to be forwarded. The state checks in the right routing table the outgoing port number and schedules the transmission of the current packet according to the evaluated delay.
- **NOT_FROM_BOTTLENECK**: In this case the packet comes from a not bottleneck link. If the outgoing link of the packet is a bottleneck link and the FIFO behaviour has been enabled then the states move in the QUEUE state, otherwise it moves to PROCESS state
- **QUEUE**: insert the packet in the tail of the FIFO queue and schedule the transmission of the packet basing on the bottleneck rate and the packet ready in the queue.

- **SEND_HEAD:** this state is triggered every time a packet has to be removed from the FIFO queue. After removing, the states move to the PROCESS state in order to apply network delay and loss to the packet.

4.6.-TX/RX RADIO F.B.

The TX/RX Radio FB binds the traditional IP network (fixed network) to the wireless network (such as UMTS, Wi-Fi, Bluetooth). Moreover, it prepares the packet to the transmission on a wireless link (channel coding and modulation). The main goal of this node is therefore to forward IP packet from one side to the other and vice-versa. Since TX and RX Radio modules have similar functionalities and mechanisms like **channel coding** and **modulation**, we chose to integrate TX and RX Radio in a single functional block. This choice is also driven by the goal to reduce the effort in case of enhancement or modification to the code of these similar features. The similar features include:

- Packet forwarding
- Introduce the packet delay due to
    - channel coding/decoding processing
    - packet queuing and transmission
    - IPv6 option header processing in the case the control/signalling information is carried on the IPv6 option header.
    - ad-hoc protocol processing in the case the control/signalling information is carried out in an ad-hoc protocol
    - adopted coding scheme and interleaving, if present

An important difference between TX and RX Radio is due to the fact that we will simulate video streaming application running between a source and one or multiple destinations in case of multicast. Therefore, the video data will be delivered only in a single way and the TX Radio should have the following added functionalities:

- Multicast transmission on the wireless link (duplication of packets to multiple RXs)
- Resize of traversing video data packet according to channel coding based on SSI

In case of multicast transmission the TX radio has to able to manage different wireless link at the same time. The TX radio should maintain the association between TX and RX couples to manage the signalling control information.

On the other hand, the RX Radio should be able to:

- Resize the arriving packets from the wireless interface to their original size (de-coding)
- Model the wireless channel introducing errors into packets according to the following factors:
    - wireless technology
    - modulation scheme
    - channel coding scheme
    - mobility
    - interleaving, if present
    - channel condition
    - ARQ mechanism, if present
- Generate the control/signalling information like CSI and NSI.

A set of input files (a file for each combination of these factors) should be provided in order to properly model the wireless channel. An item of the said sets is represented by a couple of files of bits, as the transmitted and received sequence of bits, for the corresponding combination of factors. The transmitted and received sequence of bits should be related to the sequence of bits before and after the channel coding and channel de-coding respectively.

For properly read the bit error files, the channel coding rates and IPv6 header compression rates must be chosen according to those used while obtaining the channel bit error files.

The IPv6 header compression algorithm used for the simulation was ROHC(Robust header compression).

A mapping between SSI codes and channel coding rates was done in order to adapt channel coding and decoding according to the SSI information. The channel coding rates values chosen were those used during the modelling of the wireless channel:

- 1/3
- 4/5

A way to get these files could be to concatenate the bits of the packet at data-link layer at the transmitting and receiving side (MAC and LLC header included). The correct item is selected depending on the analysed scenario. A way to exploit each item is to consider a subsequence of bits of the same length of the transmitted part of the data packet (i.e. different items are in general picked up for the transmission of a single packet, for example for different values of the SSI). The item will be accessed in a sequential way (e.g. two sequent fields SSI1 of size size1 and size2 respectively will access sequent bit stream of size size1 and size2 respectively both in the bound tx file and in rx file) and in a cyclic manner (to ensure the continuity of the stream).

The result of the comparison among the transmitted and the received bits sequences of a packet are stored into the packet itself to provide the *Video Stream Output file*.

The right item is also chosen according to the transported control information into the packets and the configuration decisions taken by the JSCC/D controller. For example, if DRI is provided, multiple bits for the data payload are generated; in this case, more items must be available and selected for the same part of the transmitted packet (several IP packets could be created for a single one, some way associated; for example with the same packet identifier).

The channel conditions can be configured a priori by a given sequence of the form

<channel condition, validity interval>.

In order to investigate a representative set of cases, the following options will be considered:

- wireless technology: at least 3 different radio technology should be evaluated (UMTS, Wi-Fi and Bluetooth)
- modulation scheme: at least 3 different types of modulations
- channel coding scheme: at least 3 different types
- mobility: fixed or mobile Rx
- interleaving: applied or not
- channel condition: 4 conditions good, fair, poor and very poor

- ARQ: applied or not (if possible indicating also the maximum number of retransmission attempts).

Choosing a suitable nomenclature can be helpful, e.g.:

technology_modulation_coding_mobility_interleaving_condition_ARQ

4.6.1 Node Model

Figure.4.11. shows the node model of the TX/RX Radio FB. It is composed by a TX/RX Process that elaborates the packets traversing the node. The node model has two or more interfaces: the first one is the wired interface while the second is the wireless interface.

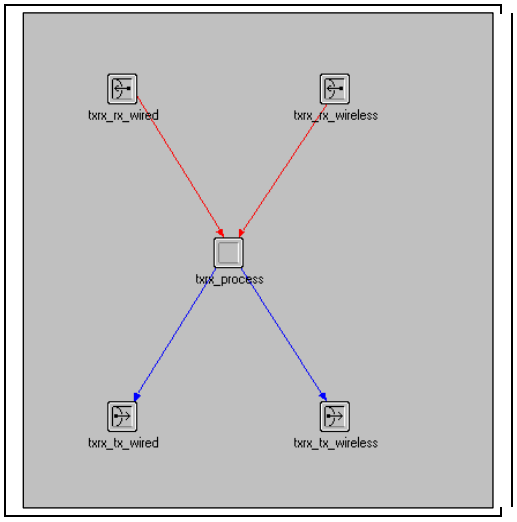

Figure 4.11 : TX/RX Radio node model

4.6.2 TX/RX process Parameter Mask

The parameters mask includes all the parameters needed to evaluate the delay introduced in the packet by the TX/RX Radio node: packet queuing, IPv6 option header processing, Ad-hoc processing, channel coding processing.

A compound parameter is used to define the delay due to different coding schemes. For each coding scheme the input/output size for coded block is given together to the delay needed to code a single block. A similar compound attribute is used also for interleaving parameters: the block size and the delay for a single block have to be specified.

One parameter allows setting the coding scheme for header field (IP header, UDP header etc….). Two options are available: header coded with the strongest coding scheme used for the current data transmission or coded with a defined coding scheme.

Two compound parameters allow the selection of right files at RX side in order to evaluate the transmission delay and the bit error respectively. The compound attributes should be composed by an attribute for each of the above listed factors (excluding the channels and modulation schemes that are run-time selected by the JSCC/D controller).

A parameter defines the channel condition (good, fair poor or very poor) during the simulation in the form of a list of <condition, interval> couples.

Table 4.8 :TX/RX Parameter Mask

Parameter Name	Description	Default Values
Header Coding Scheme	Define coding scheme used to encode protocol headers	- Strongest data coding scheme
Transmission Delay File	Compound attribute that define which file will be loaded in order to compute the transmission delay on the related wireless link It contains the specification of the wireless technology, modulation scheme, channel coding scheme, mobility, interleaving and ARQ.	
Bit Error File	Compound attribute that define which file will be loaded in order to compute the bit error on the received packets. It contains the specification of the wireless technology, modulation scheme, channel coding scheme, mobility, interleaving, and ARQ.	
Channel Conditions	Compound attribute that defines the channel conditions during the simulation at specific interval time.	0-20(sec) good 20-30(sec)poor 30-40(sec)good 40-50(sec)good 50-60(sec)very poor
IP address int	A integer value that permit to identify univocally the node in the network	3 TX Node 4 RX Node
FIFO activate	Permit to enable or disable the FIFO queue attached to the node	ENABLED
Header compress rates	A compound attribute which map each header compression rate to a channel status	30 20 10
DRIs	Compound attribute that specifies if the DRI is generated and the related overhead	Do not generate DRIs
bit_error_pattern_pos	Defines by where the cyclic error bit file must be read for the next packet arriving	NO

	at the wire-less interface	
bit_error_pattern_length	Defines the length of the error bit file.	NO

4.6.3 Statistics

Statistics on TX/RX Radio FB refers to the traffic received and traffic forwarded, packet-by-packet or average delay introduced by queuing, processing, coding, interleaving and transmission, number of errors on bits of different fields. This last should be registered both in the TX side (on packet received from the wired link) and in RX side (on packets received from wireless link). In fact, it could be useful to compare the number of errors on bits between the TX Radio and the RX radio on the concerned air interface in the case the source-to-destination path contains two or more wireless links.

Table 4.9 : TX/RX Statistics

Statistic Name	Description
Packet received (pck/sec)	Received packets Rate at IP level
Traffic received (byte/sec)	Received traffic rate in byte/sec at IP level
Packet forwarded (pck/sec)	Forwarded packets Rate at IP level
Traffic forwarded (byte/sec)	Forwarded traffic rate in byte/sec at IP level
Queuing Delay (sec)	Packet-by-packet and average Delay due to queuing
IPv6 Option Header Delay (sec)	Packet-by-packet and average Delay due to IPv6 Option Header processing
Ad-hoc Delay (sec)	Packet-by-packet and average Delay due to Ad-hoc control/signalling protocol
Encoding Delay (sec)	Packet-by-packet and average Delay due to the encoding process
Decoding Delay (sec)	Packet-by-packet and average Delay due to the decoding process
Transmission Delay (sec)	Packet-by-packet and average Delay due to the transmission over the wireless link (it includes the delay related to the ARQ, if deployed).
total Delay (sec)	Total packet-by-packet and average delay of forwarded packets
Errors on bits	Number of bit errors. This statistic is customized for every field in the packet (header and in the payload) and for the whole packet as well
BER on channel	Bit Error Rate in the Channel, this is given by the CSI packets
BER on payload	Bit Error Rate on the Payload of the packet received.
BER value on CSI	Bit Error Rate sent on CSI packet
Channel Status Value	Level of the channel condition : Good,

	fair , poor, very poor.
IPv6 Packet Received	IPv6 packet rate received (pck/sec)
IPv6 Traffic Received	IPv6 packet rate received (byte/sec)
Errors on payload	Number of bit errors on payload
Payload size(bits)	Size of payload (bits)
Wireless to Wired Delay (sec)	Delay from wireless to wired interface
Total Bit Errors on Payload	
Total Bit errors on Packet	
Total Bit errors on Header	
TX/RX Total control/signalling trafficSent (byte/sec)	
TX/RX SSI_Traffic Sent (byte/sec)	
TX/RX SRI_Traffic Sent (byte/sec)	
TX/RX SAI_Traffic Sent (byte/sec)	
TX/RX NSI_Traffic Sent (byte/sec)	
Physical Traffic Received	
Packet Error Rate	
TX/RX JSCC Traffic Received	Traffic Received of JSCC information (byte/sec)
TX/RX NO_JSCC Traffic Received	Traffic Received No JSCC (byte/sec)
TX/RX CSI_Traffic Sent (byte/sec)	
TX/RX DRI_Traffic Sent (byte/sec)	

4.6.4 Finite State Machine

Figure 4.12 shows the FSM of the TX/RX Radio process.

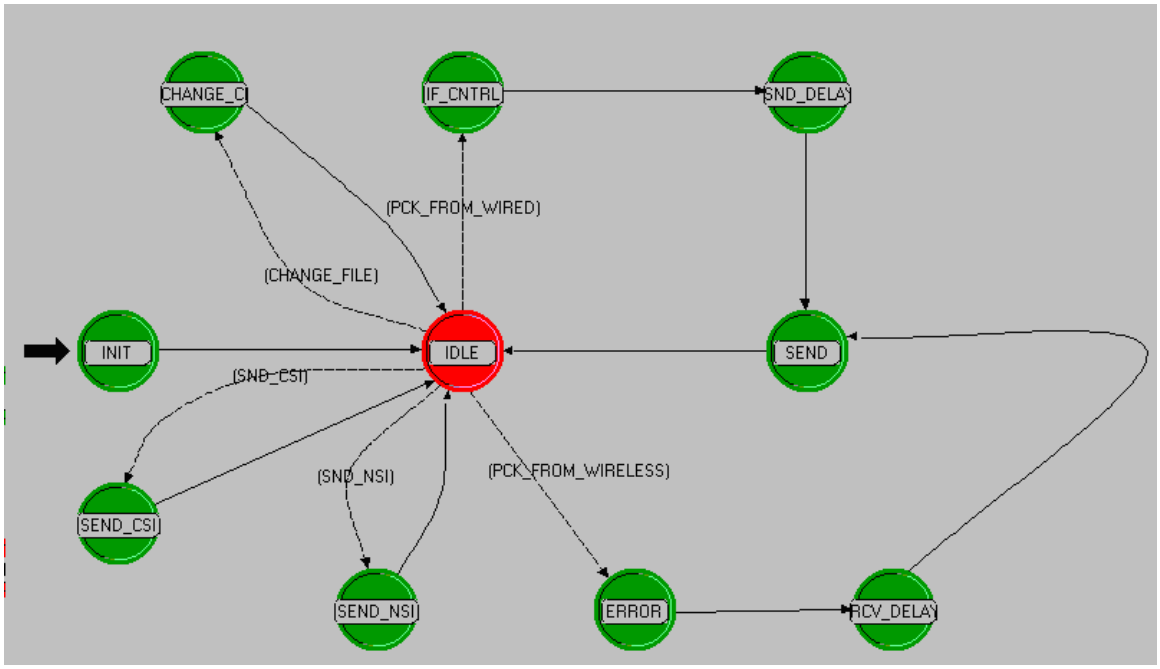

Figure 4.12 : TX/RX FSM

The tasks performed by each state are:

- **INIT**: the init state initializes the static variables importing the value from related parameters mask, load the bit error file and set-up the local statistics. Then it moves to the IDLE state.
- **IDLE**: this state waits for an incoming interrupt (event). In the TX/RX radio process the incoming interrupt can be of five types: an incoming packet from a wired interface, an incoming packet from a wireless interface, the sending of a packet stored in the local queue, the generation of a control/signalling packet (such as the CSI) and finally the change of the channel status. If the incoming packet arrives from the radio interface (PCK_FROM_WIRELESS), then the FSM moves to the ERROR state. If the incoming packet arrives from the wired interface (PCK_FROM_WIRED), then the FSM moves to the IF_CNTRL state. If a control/signalling packet needs to be created the FSM moves to the SEND_NSI or SEND_CSI states. If the wireless channel status changes, the FSM moves to the CHANGE_CH_STATUS state. The IDLE state updates the statistics related to incoming traffic.
- **CHANGE_CH_STATUS**: this state updates the wireless channel condition according to the related parameter mask value. The FSM enters in this state every time the wireless channel state is defined by a different **error pattern file** than the current one. Note that the state of the wireless channel can take the following values:
    - Good
    - Fair
    - Poor
    - Very Poor

  It loads the new error pattern file if the channel status changes.

- **IF_CNTRL**: in this state the incoming packet from the wired link is checked in order to capture control/signalling information (if any) that could be useful for the joint controller of the physical level. From the wired link a TX/RX node can receive two type of control/signalling information, either **SSI and SRI** (source a priori information) from the source controller or SAI (Source A-posteriori Information) from the destination controller. Since control/signalling information can be carried by different schemes (Standard Protocol like ICMPv6, Ad-hoc protocol like RTP/RTCP, payload of IP packets or IP extension headers), this state checks the presence of control/signalling information according to the scheme selected for the specific scenario. If the check is positive then the control/signalling information is extracted from the packet in order to allow the joint controller to take the right action. The FSM moves therefore then to the SND DELAY state.
- **SND_DELAY**: this state applies the delay to the packets according to parameters mask value (queuing, IPv6 processing, ad-hoc processing) and delay just evaluated in ENCODE state due to encode processing. The state updates also the related statistics.
- **SEND_CSI**: In this state we send a packet with CSI information, and then , the next interrupt to send this type of information is scheduled, based on the CSI timer. Then, we turn to IDLE state.
- **SEND_NSI**: In this state we send a packet with NSI information, and then , the next interrupt to send this type of information is scheduled, based on the NSI timer. Then, we turn to IDLE state.

- **ERROR**: This state checks if the packet is received in TX node or is received in RX node. If it is received on RX side, it adds errors according to the bit error pattern and SSI info, while if it is received on TX side, it writes statistics about delay and traffic received. Then, the FSM moves through the RCV_DELAY state, that do not do anything.
- **RCV_DELAY:** this state computes and applies the delay to the packets at receiver side. This delay includes the de-coding (hence, also the interleaving) and transmission delay. The last is reading the packet size-delay association from the right file. If the packet-size doesn't mach exactly with any entry of the input file, a delay interpolation will be conducted between the two nearest (higher and lower) packet-size entry. The state update also the statistics related to all the computed delays. The FSM moves then to the SEND state.
- **SEND**: this state forwards the packet to the right interface. The state updates the traffic forwarding statistics. There is a case where the forwarding is not necessary: if the packet is an SSI coming from the wireless interface in an ad-hoc or standard protocol it has not to be forwarded to the wired interface but it is useful only to the joint controller (this is not true for SAIs, source a priori information). In case of multicast transmission the packets have to be duplicated and delivered to all the RX interfaces.

4.7.- Simple Destination F.B.

The destination terminal FB is the final destination of the video data flow. Similar to the case of the Source model, we have implemented two destination terminal types: a Simple Destination node that implements a traditional sink and a JSCC/D compliant node able to manage incoming video data packets and to send back the control/signalling SAI information. Moreover, it has to be able to manage control/signalling information like DRI, SAI (source a priori information), SAI (Source A posteriori Information) and NSI (Network State Information).

4.7.1 Node Model

Figure 4.13. depicts the node model of the Destination Terminal FB. Like the previously described Source Node Model, it has a standard transmitter (TX) and a receiver (RX) process. The Destination process (over the TX and RX node) receives from the RX process the video data packets and the control/signalling information (DRI and SAI – source a priori information) as well as transmits to the TX process the control/signalling information (NSI and SAI – Source A posteriori Information) backward. **Especially, the SAI will be delivered to a wireless channel decoder (a TX/RX node) while the NSI to the SOURCE terminal node.**

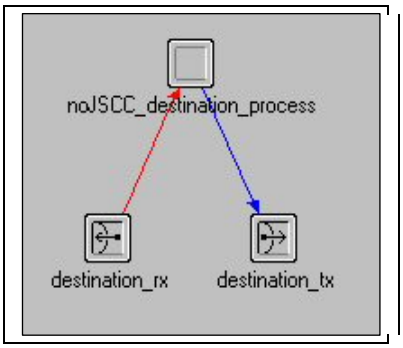

Figure 4.13 : Destination Terminal Node Model

4.7.2 Parameter Mask

The only relevant attribute of the simple destination node is its IP address (node ID). Setting this parameter the destination can receive data and control/signalling traffic and send control/signalling information.

Table 4.10 : Simple Destination Parameter Mask

Parameter Name	Description	Default Values
IP destination int	ID of the destination node	

4.7.3 Statistics

Table 4.11 shows collected statistics of noJSCC destination terminal. They are mainly the throughput and the delay received by this node.

Table 4.11 : Simple Destination Statistics

Statistic Name	Description
Application Packet received (pck/sec)	Received video packets rate at application layer
Application Traffic received (byte/sec)	Throughput received at application layer
IPv6 Packet received (pck/sec)	Received packets at IP layer
IPv6 Traffic received (pck/sec)	Throughput received at IP layer
IPv6 Packet sent (pck/sec)	Sent packets at IP layer
IPv6 Traffic sent (pck/sec)	Throughput sent at IP layer
End-to-end data delay	End 2 end delay of data packet
End-to-End packet loss	End-to-End loss of data packets

4.7.4 Finite State Machine

Figure 4.14. shows the FSM of the Destination process.

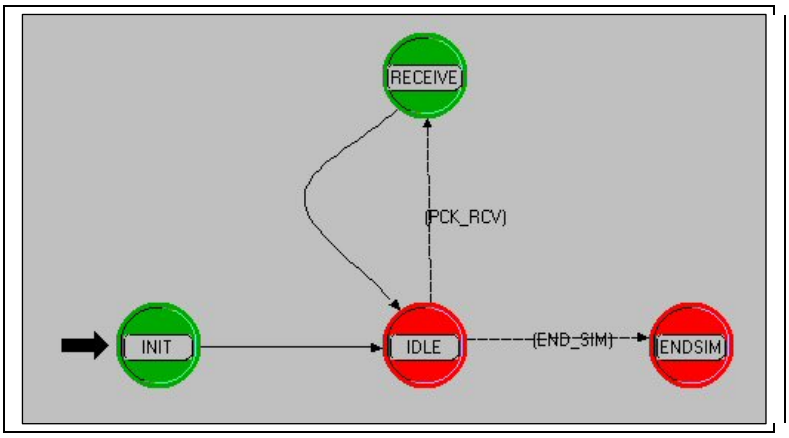

Figure 4.14 : Simple Destination FSM

The tasks performed by each state are:

- **INIT**: the init state initializes the static parameters according to the parameters mask. Moreover, it registers the statistics in order to collect simulation results. The FSM moves then to the IDLE state.
- **IDLE**: this state waits for an incoming interrupt (event). The event could be related to the reception of a video data packet. In this case the FSM moves to the RECEIVE state
- **RECEIVE**: this state processes the just received video data packet and update the statistics. Since the FSM behaviour is similar to a sink process, the packet is then destroyed.

4.8.-JSCC/D Destination F.B.

4.8.1 Node Model

The node model is similar to the node model of the Simple Destination and is depicted in Figure 4.15. It is composed by a transmitter (destination_tx) and receiver (destination_rx) and a process that implement the JSCC/D FSM (JSCC_destination_process).

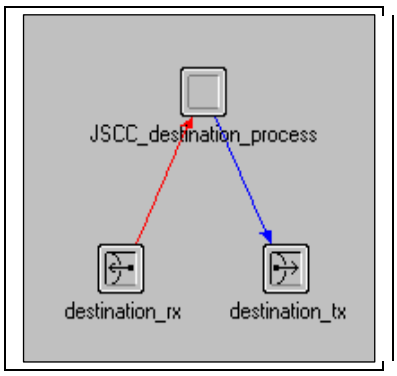

Figure 4.15 : JSCC/D Destination Node Model

4.8.2 Parameter Mask

Unlike the Simple Destination, the JSCC/D Destination Node has to manage received and transmitted control/signalling information. The control/signalling parameter are stored in the CONFIG node so, like the Simple Destination, the only relevant parameter is the IP destination address (node ID).

Table 4.12 : JSCC/D Destination Parameters Mask

Parameter Name	Description	Default Values
IP destination int	ID of the destination node	2
MTU	Maximum Transfer Unit	1500

4.8.3 Statistics

Data and control/signalling statistics are collected by the JSCC/D Destination Node.

Table 4.14 : JSCC/D Destination Statistics

Statistic Name	Description
Application Packet received (pck/sec)	Received video packets rate at application layer
Application Traffic received (byte/sec)	Throughput received at application layer
IPv6 Packet received (pck/sec)	Received packets at IP layer
IPv6 Traffic received (pck/sec)	Throughput received at IP layer
End-to-end data delay	End 2 end delay of data packet
End-to-End packet loss	End-to-End loss of data packets
Control/signalling packet received (pck/sec)	Received control/signalling packets rate at IP level
Control/signalling traffic received (byte/sec)	Received control/signalling traffic rate in byte/sec at IP level
Control/signalling packet sent (pck/sec)	Control/signalling packet sent
Control/signalling traffic sent (byte/sec)	Control/signalling traffic sent in byte/sec
Control/signalling information overhead	Relative overhead of the control/signalling information with respect to the received data (counting the received control/signalling only, or also the generated one)
SSI Packet Received (pck/sec)	SSI packet rate Received by this node
SRI Packet Received (pck/sec)	SRI packet rate Received by this node
DRI Packet Received (pck/sec)	DRI packet rate Received by this node
QoS Level Indicator	SAI Traffic Sent (byte/sec)

4.8.4 Finite State Machine

Figure 4.16. depicts the FSM of the JSCC/D Destination node.

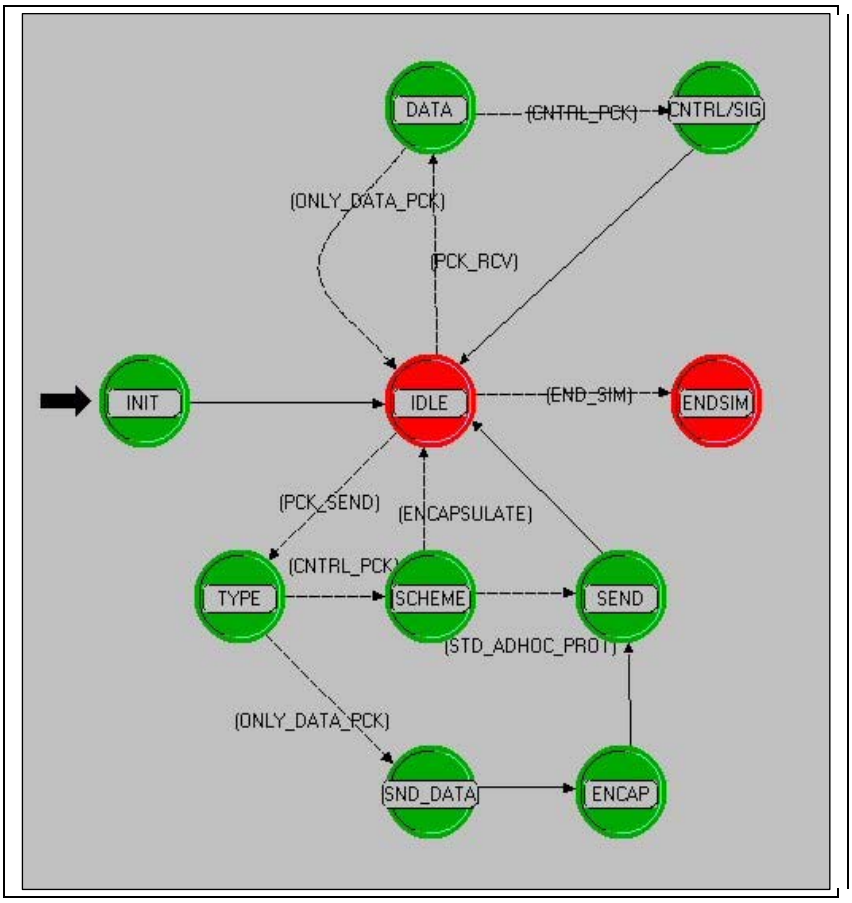

Figure 4.16 : JSCC/D Destination FSM

The tasks performed by each state are:

- **INIT**: initialization of the state variable and statistics
- **IDLE**: this state wait for an incoming interrupt like the packet reception or the packet delivery.
- **DATA**: the received packet can contain only video data, control/signalling information (in the case of out-of-band scheme) or video and control/signalling information (in-band case). This state check for the packet content. In case of video data update the related statistics while, in case of control signalling information the state moves to the CNTRL/SIGN state. If the packet contains only video data the state moves to the IDLE state
- **CNTRL/SIGN**: the packet contains control/signalling information. The state extracts and processes this information. Finally it updates the statistics and moves to the IDLE state.
- **TYPE**: the destination has to send out a packet. It can be both a data packet (for example RTCP report) or control/signalling information (SAI). This state check what type of information will be sent out. If it is control/signalling then it moves to the SCHEME state, else if it is a data packet it moves to the SND_DATA state
- **SCHEME**: a control/signalling packet need to be sent out. If the scheme is in-band, the states moves to the IDLE (the control/signalling information will be encapsulated into the next data packet that will be delivered). If the scheme is out-of-band the state creates the packet and moves to the SEND state.

- **SND_DATA**: this state creates the data packet that will be sent out then it moves to the ENCAP state.
- **ENCAP**: This state check if there is some pending control/signalling information to encapsulate into data packet. Then it moves to the SEND state.
- **SEND**: send the just created packet out.

N.B: Here , we must say that SAI and DRI information have not been implemented yet, in further scenarios and development it will be made