...
...

Process Model Attributes			
attribute	*value*	*type*	*default value*
DATA_IAT	promoted	double	10 (sec)

Header Block

```
#define   PK_ARR       op_intrpt_type () == OPC_INTRPT_STRM
#define   SELF_INT1    op_intrpt_type () == OPC_INTRPT_SELF  && op_intrpt_code () == 0
#define END_SIM        op_intrpt_type () ==  OPC_INTRPT_ENDSIM
#define DATA_REC_STAT   0

#define MAX_NO_NETS     24
#define MAX_NO_NODES  1

#define UDP_PROTOCOL_ID     20

#define ON                      1
#define OFF                     0

#define VL                      if(OFF)
#define UADL                    if(OFF)

typedef struct {
      int NetNum;
      int NodeNum;
}IPAddress;

typedef struct {
      char Message[500];
}App_Data;
```

State Variable Block

```
Objid           \my_id, \ip_objid;

double          \data_rec,\data_send, \data_iat, \print_time;

Distribution* \net_dist,\node_dist;

IPAddress \LocalAddress;
```

Temporary Variable Block

```
Packet*  pkptr;
int in_strm,da1,da2,da3;
FILE*  fil;
```

forced state **init**			
attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

...
...

enter execs **init**

```
my_id = op_id_self();

ip_objid = op_topo_assoc (my_id, OPC_TOPO_ASSOC_OUT, OPC_OBJMTYPE_MODULE, 0);

/* Use the object id to get the address */
op_ima_obj_attr_get (ip_objid, "net_num0", &LocalAddress.NetNum );
op_ima_obj_attr_get (ip_objid, "node_num0", &LocalAddress.NodeNum );

data_rec = 0;
data_send = 0;
print_time = 0;

op_ima_obj_attr_get(my_id,"DATA_IAT", &data_iat);

/*time when he joins initially*/
net_dist = op_dist_load("uniform_int", 1, MAX_NO_NETS );
node_dist = op_dist_load("uniform_int", 1, MAX_NO_NODES );

op_intrpt_schedule_self((op_sim_time() + data_iat + 35),0);
```

transition **init -> idle**

attribute	*value*	*type*	*default value*
name	tr_0	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

unforced state **idle**

attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **idle -> pk_arr**

attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	PK_ARR	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> data**

attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition	SELF_INT1	string	
executive		string	

color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> END**			
attribute	*value*	*type*	*default value*
name	tr_9	string	tr
condition	END_SIM	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state **pk_arr**			
attribute	*value*	*type*	*default value*
name	pk_arr	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **pk_arr**

```
/*
** This is a port for all the traffic that was destinated for this host. The
** state upon  receiving the packet updates the counter which keeps track of
** the number of packets received by this host. Right now it also prints the
** number of packets  received for each 20 second  interval. It destroys the
** packets that are received.
*/

pkptr  = op_pk_get (op_intrpt_strm ());

VL  UADL
{
      App_Data DataRecv;
      int Src_Net, Src_Node;
      Packet* RecivedAppPkptr;


      op_pk_nfd_get(pkptr, "src_net", &Src_Net );
      op_pk_nfd_get(pkptr, "src_node", &Src_Node );
      op_pk_nfd_get(pkptr, "data", &RecivedAppPkptr );
      op_pk_nfd_get(RecivedAppPkptr, "UniData", &DataRecv );

      printf("(%d.%d) Recieved data message from %d.%d at the time %f\n",
                  LocalAddress.NetNum, LocalAddress.NodeNum, Src_Net, Src_Node, op_sim_time() );
      printf(" %s \n", &DataRecv.Message );

      op_pk_destroy (RecivedAppPkptr);
}

data_rec++;
op_stat_local_write( DATA_REC_STAT, data_rec/op_sim_time() );

if( op_sim_time() > print_time + 35.95 )
{
      printf("(%d.%d) Received %f No. of Messages\n",
```

```
                LocalAddress.NetNum, LocalAddress.NodeNum, data_rec );
        print_time = op_sim_time();
    }

40  op_pk_destroy (pkptr);
```

transition **pk_arr -> idle**

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	default	string	
executive		string	
color	RGB203	color	RGB333
drawing style	spline	toggle	spline

forced state **data**

attribute	*value*	*type*	*default value*
name	data	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **data**

```
    App_Data DataSent;
    Packet* ip_pkptr;
    int dest_net, dest_node, data_len;

 5  dest_net = op_dist_outcome( net_dist );
    dest_node = op_dist_outcome( node_dist );

    sprintf( DataSent.Message, "This was data sending to the host %d.%d by %d.%d \n",
            dest_net, dest_node, LocalAddress.NetNum, LocalAddress.NodeNum );
10  pkptr = op_pk_create_fmt("Unicast_Appl");

    op_pk_nfd_set(pkptr, "UniData", &DataSent, op_prg_mem_alloc,
                op_prg_mem_free, sizeof(App_Data) );

15  data_len = op_pk_total_size_get( pkptr )/8;

    ip_pkptr = op_pk_create_fmt("ip_dgram");
    op_pk_nfd_set(ip_pkptr, "data", pkptr );
    op_pk_nfd_set(ip_pkptr, "prot_id", UDP_PROTOCOL_ID );
20  op_pk_bulk_size_set(ip_pkptr, data_len*8 );

    op_pk_nfd_set(ip_pkptr, "dest_net", dest_net );
    op_pk_nfd_set(ip_pkptr, "dest_node", dest_node );
    op_pk_nfd_set(ip_pkptr, "src_net", LocalAddress.NetNum );
25  op_pk_nfd_set(ip_pkptr, "src_node", LocalAddress.NodeNum );

    op_pk_nfd_set(ip_pkptr, "orig_len", data_len );
    op_pk_nfd_set(ip_pkptr, "frag_len", data_len );
    op_pk_nfd_set(ip_pkptr, "ttl", 2 );
30  op_pk_nfd_set(ip_pkptr, "frag", 0 );
```

...
...

```
VL  if(OFF)  printf("Sending a data pk to %d.%d. PACKET ID = %d\n",
                    dest_net, dest_node, op_pk_id(ip_pkptr) );
op_pk_send(ip_pkptr,0);
op_intrpt_schedule_self((op_sim_time() + (data_iat)),0);

data_send++;
```

transition **data -> idle**

attribute	*value*	*type*	*default value*
name	tr_8	string	tr
condition	default	string	
executive		string	
color	RGB203	color	RGB333
drawing style	spline	toggle	spline

unforced state **END**

attribute	*value*	*type*	*default value*
name	END	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

enter execs **END**

```
fil = fopen("OP","a+");
fprintf(fil,"\nAve DATA recieved : %f pks/sec",data_rec/op_sim_time());
fclose(fil);

printf("(%d.%d) Number of packets received by me is %f \n",
        LocalAddress.NetNum, LocalAddress.NodeNum, data_rec );
printf("(%d.%d) Number of packets sended by me is %f \n",
        LocalAddress.NetNum, LocalAddress.NodeNum, data_send );
```