The ATM Application Program Interface for the Intel Paragon

Revision 5
February 1, 1996

GigaNet Incorporated
Stow MA
(508) 461−0402

INTRODUCTION

The ATM Application Program Interface (API) for the Intel Paragon provides ATM network access to Paragon applications. The API allows an application to create an ATM access point, to bind and unbind ATM Virtual Circuit Identifiers (VCI) to the access point, to transmit frames to the ATM access point, and to receive frames from the ATM access point.

This API supports a low overhead interface to the ATM network from any Paragon node. The Paragon operating system is used to create and destroy the ATM access point. After the ATM access point is created, all data is transferred directly between the application address space and the Paragon mesh. Several data structures located in the application address space and shared with the Paragon message processor are used to control data transfers. The transmit queue is used to transmit frames to the ATM network. The receive free queue and the receive ready queues are used to receive frames from the ATM network.

Since the Paragon operating system is not involved in data transfers, this API provides a polled interface to the application.

ATM ACCESS POINT

An ATM access point is a connection between an application and the ATM driver software and ATM hardware. The ATM library, in conjunction with the Paragon operating system, creates the ATM access point. After the ATM access point is created, the application may transmit and receive frames.

ATM MESH PROTOCOL

The ATM mesh protocol is a lightweight protocol for the Paragon mesh that is used to carry data between Paragon applications and the ATM network interface driver software and hardware. At the application node, the ATM mesh protocol supports direct data transfers between the application address space and the Paragon mesh. At the ATM node, the ATM mesh protocol supports direct data transfers between the Paragon mesh and the ATM network interface hardware.

TRANSMIT QUEUE

The transmit queue contains requests to transmit a frame to the ATM network. When the application is ready to transmit a frame, the application enqueues the transmit request in the transmit queue. The transmit request contains the ATM control and buffer pointers necessary to create an ATM AAL5 frame. This information is used by the Paragon message processor to gather data from the application address space and transfer the data over the Paragon mesh to the ATM node.

RECEIVE FREE QUEUE

The receive free queue contains buffers that may be used when a frame is received from the ATM network. When the application expects to receive frames from the ATM network, the application should add free buffers to the receive free queue.

RECEIVE READY QUEUE

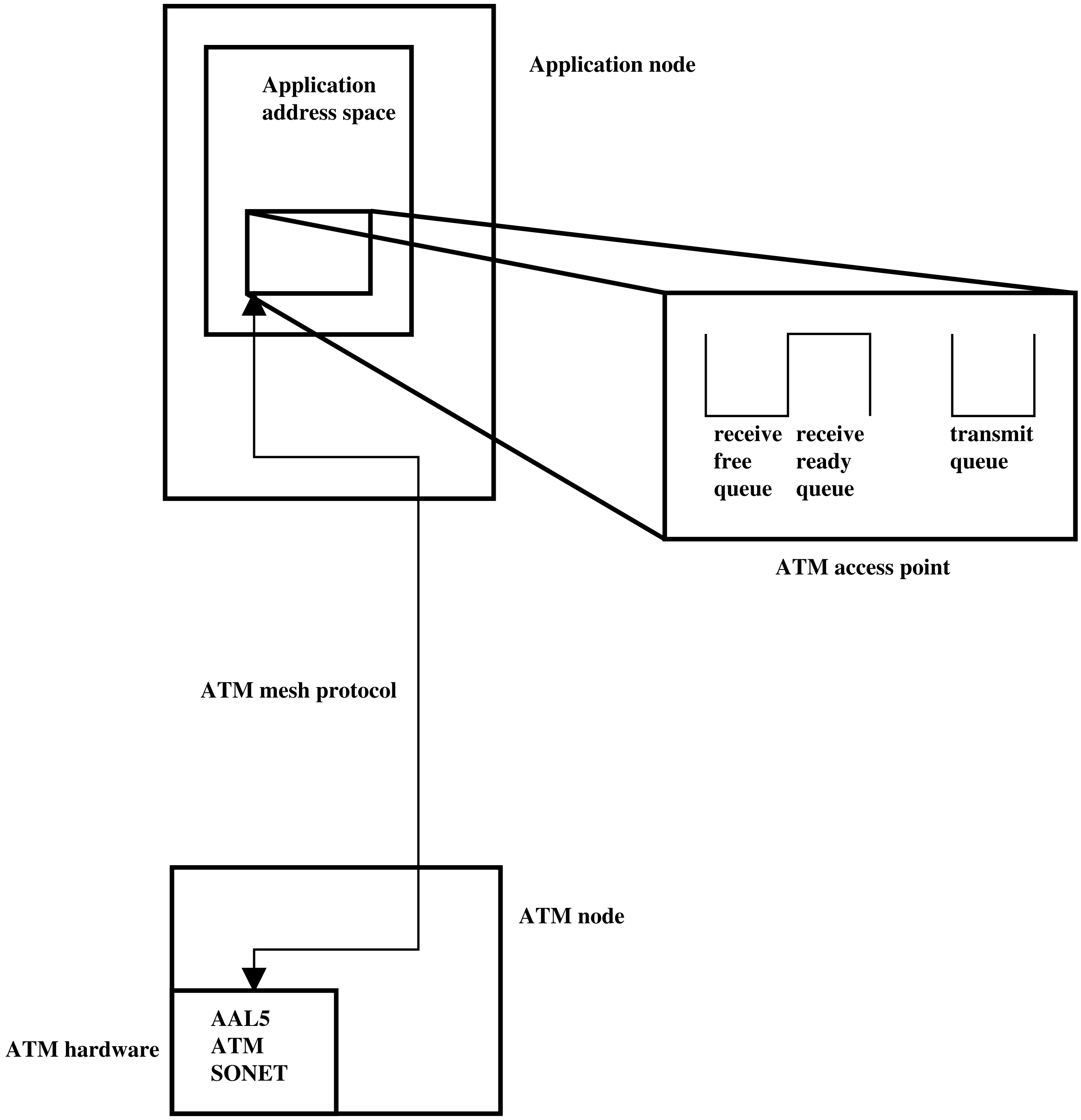
The receive ready queue contains frames already received from the ATM network. ATM frames received from the ATM network are placed into free buffers. When the complete frame has been received, the ATM control information and the buffer pointer are placed into the receive ready queue. The application may then receive the frame from the receive ready queue.

LIBRARIES

libatm.a

INCLUDE FILES

#include <atm/atm_sap_mesh.h>


Application node
Application
address space
receive
free
queue
receive
ready
queue
transmit
queue
ATM access point
ATM mesh protocol
ATM node
AAL5
ATM
SONET
ATM hardware

SUMMARY

SAP Creation and Destruction

atm_sap_mesh_new – create a new ATM SAP

atm_sap_mesh_delete – delete an ATM SAP

ATM VCI Binding and Unbinding

atm_sap_mesh_bind_vci – bind aVCI to an ATM SAP

atm_sap_mesh_unbind_vci – unbind a VCI from an ATM SAP

Transmitting to the ATM Network

atm_sap_mesh_xmit_raw_cells – transmit ATM cells

atm_sap_mesh_xmit_atm_cells – make and transmit ATM cells

atm_sap_mesh_xmit_aal5 – make and transmit an AAL5 frame

atm_sap_mesh_xmit_done – determine if a transmit request is completed

atm_sap_mesh_xmit_empty – determine if the transmit queue is empty

Receiving from the ATM Network

atm_sap_mesh_nfree – return the number of buffers in the free queue

atm_sap_mesh_addfree – add a buffer to the free queue

atm_sap_mesh_addfree_buffer – add a user supplied buffer to the free queue

atm_sap_mesh_nready – return the number of entries on the receive ready queue

atm_sap_mesh_recv – retrieve and remove the first entry from the receive ready queue

NAME

`atm_sap_mesh_new` – create a new ATM access point

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

atm_sap_mesh_t *atm_sap_mesh_new(const char *server)
```

DESCRIPTION

The atm_sap_mesh_new function create a new ATM access point. Once created, ATM VCIs may be dynamically bound and unbound to the access point. Once bound, ATM frames may be transmitted to and received from the ATM VCIs.

An ATM access point consists of the transmit queue, the receive free queue, and the receive ready queue allocated in the application address space, as well as a Paragon mesh connection between the application address space and the ATM device node.

An ATM access point is attached to the ATM interface board identified by server. For example, /dev/atmN is a name in the UNIX filesystem space that returns a unique ATM access point to the Nth ATM interface board when opened.

RETURNS

`atm_sap_mesh_new` returns a pointer to a `atm_sap_mesh_t` if a new ATM access point can be created. Otherwise zero is returned.

BUGS

The clone open feature in Paragon Release 1.3 does not work correctly for remote devices. Therefore, the users must perform name space management. Names with format /dev/atm/N.M exist in the file system name space. An open of /dev/atm/N.M creates an ATM access point on ATM hardware interface N device M.

NAME

`atm_sap_mesh_delete` – delete an existing ATM service access point

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

void atm_sap_mesh_delete(
            atm_sap_mesh_t *sap, void (*reclaim)(void *, int))
```

DESCRIPTION

Deleting an ATM access point disassociates the ATM access point from the ATM network. All of the ATM VCIs bound to the ATM access point are unbound. The transmit and receive queues are deallocated.

If the `reclaim` function is provided, then the `reclaim` function is called for all receive buffers on the receive ready queue or on the receive free queue.

RETURNS

None.

BUGS

The reclaim feature is not yet implemented.

NAME

`atm_sap_mesh_bind_vci` – bind a vci to this ATM access point

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_bind_vci(atm_sap_mesh_t *sap, int vci)
```

DESCRIPTION

Bind this vci to this ATM access point. The binding allows the application to transmit frmes to the ATM network using this `vci`. Frames received from the network with this `vci` are directed to this ATM access point.

Multiple ATM VCIs may be bound to the same ATM access point.

RETURNS

Returns non−zero if the VCI was successfully bound to the ATM access point. Otherwise returns zero.

NAME

`atm_sap_mesh_unbind_vci` – unbind a vci from this ATM access point

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_unbind_vci(atm_sap_mesh_t *sap, int vci)
```

DESCRIPTION

Unbind this `vci` from this ATM access point.

RETURNS

Returns non–zero if the VCI was successfully unbound from the ATM access point. Otherwise returns zero.

NAME

`atm_sap_mesh_xmit_raw_cells` – transmit ATM cells

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_xmit_raw_cells(
            atm_sap_mesh_t *sap, void *a, int al, void *b, int bl);
```

DESCRIPTION

The `atm_sap_mesh_xmit_raw_cells` function causes ATM cells to be transmitted . The concatenated buffer denoted by the a and b pointers is the source of a sequence of ATM cells. Each cell is formatted in GigaNet ATM cell format.

RETURNS

Returns less than zero if the transmit request could not be put into the transmit queue. Otherwise returns the index in the transmit queue of the transmit request. The index is a number greater than or equal to zero

BUGS

The software currently does not verify that the VCI in the ATM cell headers are valid for this ATM access point.

NAME

atm_sap_mesh_xmit_atm_cells – make and transmit ATM cells.

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_xmit_atm_cells(
            atm_sap_mesh_t *sap,
            atm_cell_header_user_t *header,
            void *a, int al, void *b, int bl);
```

DESCRIPTION

The`atm_sap_mesh_xmit_atm_cells` function causes the ATM hardware to make and transmit ATM cells . The concatenated buffer denoted by the a and b pointers is the source of a sequence of ATM cell payloads. The ATM cell headers is supplied by `header`.

RETURNS

Returns less than zero if the transmit request could not be put into the transmit queue. Otherwise returns the index in the transmit queue of the transmit request. The index is a number greater than or equal to zero

NAME

`atm_sap_mesh_xmit_aal5` – transmit data to the ATM network using AAL5 encapsulation

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_xmit_aal5(
            atm_ap_mesh_t *sap,
            atm_cell_header_user_t *cell,
            aal5_trailer_user_t *trailer,
            void *a, int al, void *b, int bl)
```

DESCRIPTION

The `atm_sap_mesh_xmit_aal5` function encapsulates the payload contained within the concatenated buffer denoted by the pointers a and b in an ATM Adaptation Layer 5 frame.

The payload of the AAL5 frame consists of the concatenation of the data in buffer `a` for length `al` and buffer `b` for length `bl`. The pad of the AAL5 frame is automatically added.

The AAL5 trailer consists of the 4 byte `trailer` supplied by the application and the AAL5 CRC computed by the ATM network interface. If `trailer` is zero, then a default AAL5 trailer is used.

The ATM network interface segments the AAL5 frame into ATM cells using the ATM cell header supplied by the user. If `header` is zero, then a default ATM cell header is used.

RETURNS

Returns less than zero if the transmit request could not be put into the transmit queue. Otherwise returns the index in the transmit queue of the transmit request. The index is a number greater than or equal to zero

NAME

`atm_sap_mesh_xmit_done` – determine if a transmit request is finished

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_xmit_done(atm_sap_mesh_t *sap, unsigned xi)
```

DESCRIPTION

This functions determines if the transmit request denoted by `xi` has been transmitted.

RETURNS

Returns zero if the transmit request is queued but not transmitted. Otherwise returns non–zero.

NAME

`atm_sap_mesh_xmit_empty` – determine if the transmit queue is empty

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_xmit_empty(atm_sap_mesh_t *sap)
```

DESCRIPTION

The transmit queue is a data structure located in the application address space that is used by the application to transmit frames to the ATM network. The transmit queue is empty when all of the frames that have been queued for transmission have actually been transmitted.

RETURNS

Returns non–zero if the transmit queue is empty. Otherwise returns zero.

NAME

`atm_sap_mesh_nfree` – compute the number of free buffers in the receive free queue

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

unsigned atm_sap_mesh_nfree(atm_sap_mesh_t *sap)
```

DESCRIPTION

The receive free queue is a queue of free buffers that will be used when frames are received from the ATM network. `atm_sap_mesh_nfree` computes the number of buffers in the receive free queue.

RETURNS

Returns the number of buffers in the receive free queue.

NAME

`atm_sap_mesh_addfree` – add a buffer to the receive free queue

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_addfree(atm_sap_mesh_t *sap, int nbytes)
```

DESCRIPTION

`atm_sap_mesh_addfree` allocates a buffer of size `nbytes` and places the address and size of the buffer in the receive free queue. The buffer that is allocated is paged into the application address space, but not wired into memory.

RETURNS

Returns non−zero if a buffer was successfully placed into the receive free queue. Otherwise returns zero.

NAME

`atm_sap_mesh_addfree_buffer` – add a buffer to the receive free queue

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_addfree_buffer(
            atm_sap_mesh_t *sap, void *addr, int nbytes)
```

DESCRIPTION

`atm_sap_mesh_addfree_buffer` adds a buffer with base address `addr` and maximum size `nbytes` to the receive free queue. The buffer should be paged into the application space. The buffer may be wired into memory.

If the buffer is not resident in memory when a frame is received from the ATM network, then the frame will be discarded.

RETURNS

Return non–zero if the buffer was successfully placed into the receive free queue. If the queue is full, then zero is returned.

NAME

`atm_sap_mesh_nready` – return the number of received frames in the receive ready queue

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

unsigned atm_sap_mesh_nready(atm_sap_mesh_t *sap)
```

DESCRIPTION

The receive ready queue is a queue of frames received from the ATM network. This function computes the number of received frames.

RETURNS

Returns the number of received frames in the receive ready queue.

NAME

`atm_sap_mesh_recv` – retrieve a received frame from the receive ready queue

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

int atm_sap_mesh_recv(
            atm_sap_mesh_t *sap,
            atm_cell_header_user_t *cell,
            aal5_trailer_user_t *trailer,
            void **addr,
            unsigned *nbytes,
            unsigned *rcrc,
            unsigned *checksum)
```

DESCRIPTION

The receive ready queue contains frames that have been received from the ATM network but not yet consumed by the application. The `atm_sap_mesh_recv` function retrieves the first frame from the receive ready queue. The ATM cell header is copied into `cell`. The ATM AAL5 trailer is copied into `trailer`. The pointer to the frame is stored into `addr`. The length of the received frame, including the AAL5 pad and trailer, is stored into `nbytes`. The AAL5 CRC computed by the receive hardware is stored into `rcrc`. The Internet checksum computed over the AAL5 payload including the pad is stored into `checksum`.

RETURNS

Returns a number less than zero if the receive ready queue is empty. Returns zero if an AAL5 frame was received with no errors.

Returns a number greater than zero if data has been received in error. If the received data is not a valid AAL5 frame, then the data has been received in error. If the receive buffer was not resident in memory or is too small, then the data has been received in error.

NAME

`atm_sap_mesh_cell_header` – return a pointer to an ATM cell header

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

atm_cell_header_user_t *
atm_sap_mesh_cell_header(atm_sap_mesh_t *sap)
```

DESCRIPTION

Each `atm_sap_mesh_t` has an ATM cell header structure located in it. This function retuns a pointer to that structure.

RETURNS

Returns a pointer to an ATM cell header.

NAME

`atm_sap_mesh_aal5_trailer` – return a pointer to an AAL5 trailer

SYNOPSIS

```
#include <atm/atm_sap_mesh.h>

aal5_trailer_user_t *
atm_sap_mesh_aal5_trailer(atm_sap_mesh_t *sap)
```

DESCRIPTION

Each `atm_sap_mesh_t` has an AAL5 trailer structure located in it. This function retuns a pointer to that structure.

RETURNS

Returns a pointer to an AAL5 trailer.

EXAMPLE

```
#include <atm/atm_sap_mesh.h>

int
atm_rpc(int vci, void *to, int tolen, void **from, int *fromlen)
{
    atm_sap_mesh_t *sap;
    atm_cell_header_user_t cell;
    aal5_trailer_user_t trailer;
    unsigned rcrc, sum;
    unsigned xi;
    int r;

    /* create the ATM access point */
    sap = atm_sap_mesh_new("atm1");
    if (sap == 0)
        return -1;

    /* bind vci */
    if (!atm_sap_mesh_bind_vci(sap, vci)) {
        atm_sap_mesh_delete(sap, 0);
        return -2;
    }

    cell = *atm_sap_mesh_atm_header(sap);
    cell.vci = vci;

    /* put a receive buffer in the free queue */
    if (!atm_sap_mesh_addfree(sap, *fromlen)) {
        atm_sap_mesh_delete(sap, 0);
        return -3;
    }

    /* transmit the request */
    xi = atm_sap_mesh_xmit_aal5(sap, &cell, 0, to, tolen, 0, 0);
    if (xi < 0) {
        atm_sap_mesh_delete(sap, 0);
        return -4;
    }

    / optionally wait for the transmit to finish */
    while (!atm_sap_mesh_xmit_done(sap, xi))
        ;

    /* receive the reply */
    for (;;) {
        r = atm_sap_mesh_recv(sap,
                        &cell, &trailer, from, fromlen, &rcrc,&sum);
        if (r >= 0)
            break;
    }

    /* truncate the AAL5 pad and trailer */
    *fromlen = trailer.length;

    atm_sap_mesh_delete(sap, 0);
    return 1;
}
```

NAME

`atm_cell_header_network_encode` – encode an ATM cell header

SYNOPSIS

```
#include <protocols/atm/cell_giganet.h>

void atm_cell_header_network_encode(
            unsigned char *header,
            atm_cell_header_user_t *from)
```

DESCRIPTION

The `atm_cell_header_network_encode` function encodes an ATM cell header in native ATM (big endian) format. The ATM cell header in native format is specified by `from`. The encode buffer is specified by `header`.

NAME

atm_cell_header_network_encode_list – encode an ATM cell header

SYNOPSIS

```
#include <protocols/atm/cell_giganet.h>

void atm_cell_header_network_encode_list(
            unsigned char *header,
            unsigned gfc,
            unsigned vpi,
            unsigned vci,
            unsigned pt,
            unsigned clp)
```

DESCRIPTION

The atm_cell_header_network_encode_list function encodes an ATM cell header in native ATM (big endian) format. The encode buffer is specified by header.

NAME

`atm_cell_header_network_decode` – decode an ATM cell header

SYNOPSIS

```
#include <protocols/atm/cell_giganet.h>

void atm_cell_header_network_decode(
            unsigned char *header,
            atm_cell_header_user_t *user)
```

DESCRIPTION

An ATM cell header in native ATM format specified by `header` is converted into local machine format and stored in `user`.

NAME

`atm_cell_header_hec_compute` – compute the ATM cell header check byte

SYNOPSIS

```
#include <protocols/atm/cell_giganet.h>

unsigned char atm_cell_header_hec_compute(unsigned char *header)
```

DESCRIPTION

The `atm_cell_header_hec_compute` function computes the ATM header check function on the first four bytes of the ATM cell header.

RETURNS

The result of the ATM header check computation is returned. This result may be placed into the hec field of the ATM cell header.

NAME

`atm_cell_header_hec_check` – check the ATM cell header check byte

SYNOPSIS

```
#include <protocols/atm/cell_giganet.h>
```

```
unsigned char atm_cell_header_hec_check(unsigned char *header)
```

DESCRIPTION

The atm_cell_header_hec_check function computed the ATM header check function on the first five bytes of the ATM cell header

RETURNS

The result of the ATM header check function is returned.