MPCP State Machines

Yinghua Ye, Chunsheng Xin, Nokia
Ariel Maislos, Passave
Outline

• General Description

• Main Function of State Machines

• The Detailed State Machine Diagrams
General Description

• The MPCP, which is implemented in MAC control sublayer, specifies a control mechanism between an OLT and ONUs connected to a Point-to-Multi-Point (P2MP) segment to allow efficient transmission of data.

• The operation of the protocol is controlled by a number of state machines, each of which performs a distinct function.

• Since OLT and ONU have different requirements for each function, MPCP state machines further consists of the transmit, receive, and auto-discovery state machines in ONU and OLT respectively.
Function of State Machines

- **Transmit State Machine**
  - It handles the transmission of control or data packet from the layer above MAC sublayer.

- **Receive State Machine**
  - The opcode-independent MAC control sublayer Receive State Machine accepts and parses valid frames received from the MAC sublayer.

- **Auto-discovery State Machine**
  - It is responsible for harmonizing new ONU's into EPON system.
Transmission_in_progress=false

INITIALIZE TX
Transmission_in_progress=false

TRANSMIT READY
transmitEnabled=true

HALT TX
transmitEnabled=false

SEND CONTROL FRAME
Transmission_in_progress=true,
Timestamp\leftarrow\text{Local\_Clock\_Counter}
TransmitFrame (DA, SA, 802.3\_MAC\_Control,
Opcode | indication\_operandlist)

SEND DATA FRAME
Transmission_in_progress=true
TransmitFrame (DA, SA, m\_sdu)

MA\_CONTROL.request (DA, SA, 802.3\_MAC\_Control,
Opcode | indication\_operandlist)

\text{MA\_DATA.request}(DA, \ m\_sdu,
\text{service\_class})^*

\text{MA\_CONTROL.request}(DA, \ SA,
802.3\_MAC\_Control, \ Opcode,
\text{indicaton\_operandlist} )

\text{UCT}

\text{UCT}
Receive State Machine @ OLT

BEGIN

WAIT FOR ENABLE

ReceiveEnabled=true

RX READY

ReceiveFrame (DA,SA, LengthOrType, data): ReceiveStatus

LengthOrType=802.3_MAC_CONTROL

CHECK OPCODE

Opcode=data[1:16]

Opcode={supported code}

PASS TO CLIENT

MA_DATA.indication (DA,SA, LengthOrType |data, ReceiveStatus)

LengthOrType≠802.3_MAC_CONTROL

INITIATE MAC CONTROL

Timestamp=data[17:48]

RTT_register⇐⇐⇐⇐Local_Clock_Counter-Timestamp

UCT

PASS TO CONTROL CLIENT

indication_operandlist=Data

MA_CONTROL.indication (Opcode, indication_operandlist)

UCT
**Auto-Discovery State Machine @ OLT**

**BEGIN**

- **INITIATE DISCOVERY**
  - Disc_W_Start=false
  - DA=multicast_address
  - MA_CONTROL.request (DA, GATE_command, Dis_GRANT | OLT capabilities)

- **SEND DISCOVERY GATE**
  - Disc_W_Start=true
  - DA=multicast_address
  - MA_CONTROL.request (DA, GATE_command, Dis_GRANT | OLT capabilities)

  MA_CONTROL.indication (REGISTER_REQUEST_command, PHY_ID capabilities | ONU capabilities | echo of ONU capabilities)

- **SEND REGISTER AND GATE FUNCTIONS**
  - RecordPDU Selected PHY_ID_list DA=ONU_MAC_address
  - MA_CONTROL.request (DA, REGISTER_command, PHY_ID_list | echo of ONU capabilities)
  - MA_CONTROL_request (DA, GATE_command, GRANT)

<table>
<thead>
<tr>
<th>ONU UNREGISTERED</th>
<th>ONU REGISTERED</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCT</td>
<td>UCT</td>
</tr>
</tbody>
</table>

MA_CONTROL.indication (REGISTER_ACK_command, echo of registered PHY ID) *(Slot_Start_register + Slot_Length_register < Local_Clock_Counter)* + MA_CONTROL.indication (REGISTER_REQUEST_command, PHY_ID capabilities | ONU capabilities | echo of ONU capabilities)
transmitEnabled=false * Local_Clock_Counter < Slot_Start_Register + Slot_Length_Register

BEGIN

TRANSMIT READY

transmitEnabled=true

INITIALIZE TX

transmitEnabled=false

HALT TX

transmitEnabled=true

SEND CONTROL FRAME

transmitEnabled=false

SEND DATA FRAME

transmitEnabled=false

READ GRANT QUEUES

transmitEnabled=true
Receive State Machine @ONU

BEGIN

WAIT FOR ENABLE

ReceiveEnabled=true

RX READY

ReceiveFrame (DA,SA, LengthOrType, data): ReceiveStatus

LengthOrType=802.3_MAC_CONTROL

Opcode=data[1:16]

LengthOrType≠802.3_MAC_CONTROL

Pass to Client

INITIATE MAC FUNCTION

Timestamp=data[17:48]

Local_Clock_Counter←Timestamp

Write to Grant Queue

NumberofGrant=data[49:55] Start=data[57:88]

WriteQueue (GrantQueue, data[57:57+NumberofGrant*48])

Start≥Slot_Start_register+Slot_Length_register

Write Slot Related Registers

Start= Read (GrantQueue), Length=Read (GrantQueue)

Slot_Start_register←Start, Slot_Length_register←Length

Start<Slot_Start_register+Slot_Length_register

Pass to Client

_opcode=[supported code] UCT

Opcode=02

Start<Slot_Start_register+Slot_Length_register

Pass to Client

Opcode=02 UCT

Opcode≠02

UCT

PASS TO CLIENT

indication_operandlist=Data

MA_CONTROL.indication (Opcode, indication_operandlist)

Opcode≠02 UCT

Start<Slot_Start_register+Slot_Length_register

Pass to Client

Opcode≠02

UCT

PASS TO CLIENT

indication_operandlist=Data

MA_CONTROL.indication (Opcode, indication_operandlist)
BEGIN

**WAIT FOR DISCOVERY GATE**

- DiscoveryGate=true
- UCT

**SEND REGISTER REQUEST**

DiscoveryGate=false,
MA_CONTROL.request (DA, REGISTER_REQUEST_command, PHY ID capabilities | ONU capabilities | echo of OLT capabilities)

- DiscoveryGate=true*
- Contention_Scheme=2

**RECEIVE REGISTER**

- Local_PHY_ID_list ← PHY ID list

**SEND REGISTER_ACK**

MA_CONTROL.request (DA, REGISTER_ACK_command, echo of registered PHY ID)

- UCT

**DISCOVERY DONE**

- DiscoveryGate=true
- *Skip_times≠0

**DELAY PROCESS 2**

DELAY(Rand2()%Disc_Window_Size)

- UCT

**DELAY PROCESS 1**

- Skip_Times=Rand1()
- UCT

**SKIP DISCOVERY WINDOWS**

- Skip_Times=Skip_Times-1

- DiscoveryGate=false

**DISCOVERY DONE**

- DiscoveryGate=true
- *Skip_times≠0

**END**