Contention State Machine for High Speed Point to Multipoint Networks

Subir Varma

Aperto Networks
1637 South Main Street
Milpitas, CA 95035

1.0 Current State of the Art

Upstream transmissions in a point-to-multipoint network are subject to contention, since several CPE devices share a common channel. Hence the MAC layer in these networks incorporates a contention resolution protocol, in which CPEs with upstream data send a short REQ message in contention mode, followed by the data packet in reserved mode. One of the objectives of the MAC protocol is to minimize the amount of contention traffic in the upstream channel. This objective is important since it leads to a reduction in the latency required to access the upstream channel.

In the current state of the art, the reduction in the amount of upstream contention traffic is achieved by technique known as piggybacking, which works as follows: Whenever a CPE sends a data packet upstream, it attaches the size of the current backlog to one of the header fields in the packet. As a result, as long as the backlog is non-zero, the BSC is able to give grants to the CPE, without the CPE having to send any more contention requests.

2.0 Shortcomings of Current Art

The piggybacking scheme is effective only if upstream traffic from the CPE is of the bursty type, so that the CPE needs to make only one contention request per burst. However, there are important traffic sources that are non-bursty. Examples include the traffic generated by online games and voice sources. Even ordinary Internet traffic running over TCP becomes non bursty when the link is in heavy traffic. In the presence of non-bursty traffic, the piggyback scheme breaks down, and an excessive number of contention requests are sent upstream.

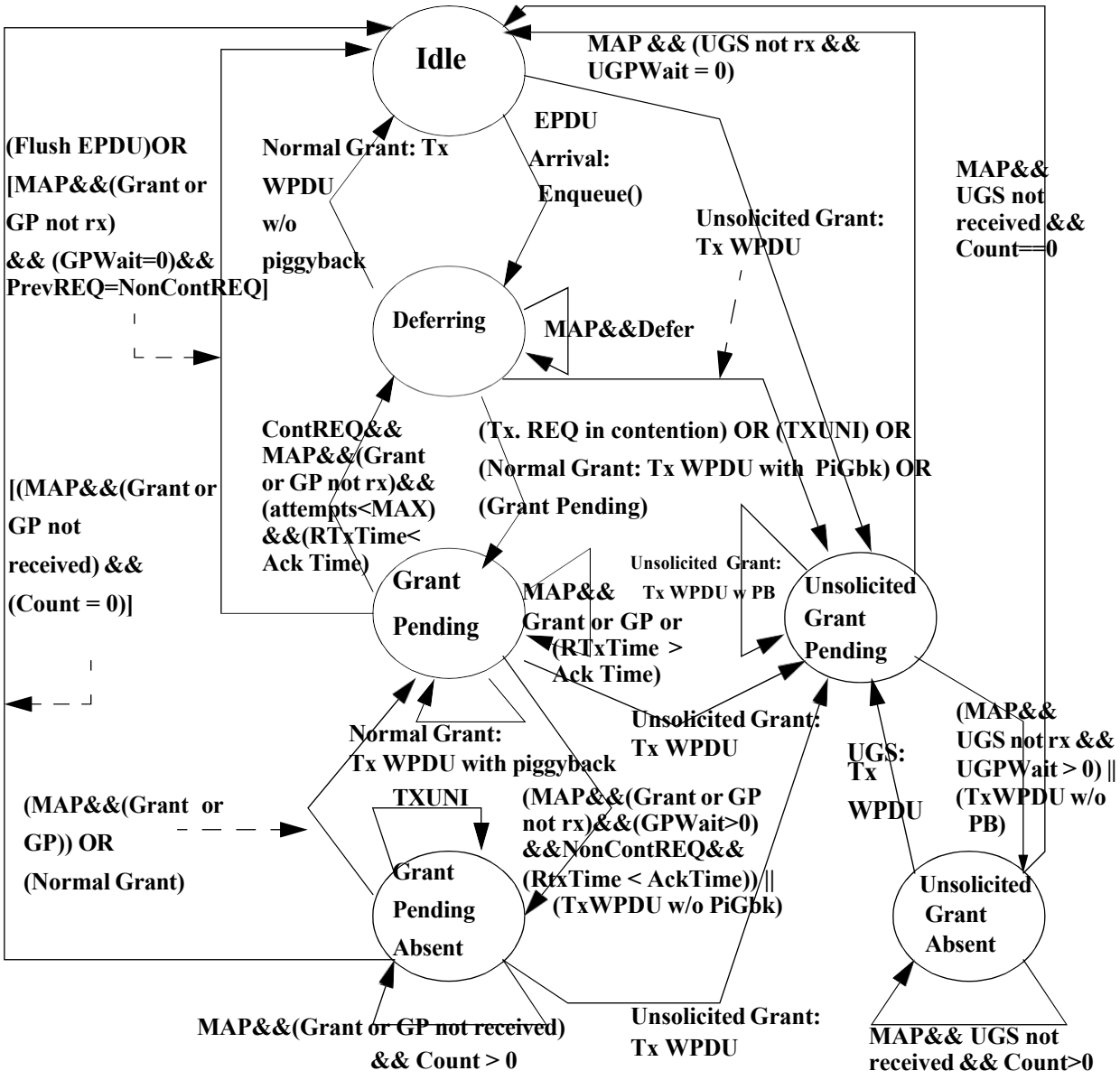
3.0 How the Proposed Invention Extends the Current Art

We propose a new State Machine that governs the transmission of upstream requests. This State Machine solves the problem mentioned above, ie, it reduces the amount of conten-

tion traffic generated by non-bursty data sources. It does so by doing the following: Whenever the CPE runs out of upstream data, then instead of going back to the IDLE state, it goes to the GRANT_PENDING_ABSENT state. It waits in the GRANT_PENDING_ABSENT state for additional upstream traffic. Meanwhile the BSC periodically polls the CPE with Unicast non-contention based request slots. If new upstream traffic arrives while the CPE is in the GRANT_PENDING_ABSENT state, then it makes a non-contention based upstream request by utilizing one of the unicast request grants from the BSC. As a result of this design, the CPE is able to avoid making a contention based request for the new burst, and thus reduce the amoung of upstream contention request traffic.

Pseudocode

4.0 FSM for SIDs with Best Effort and Committed Information Rate Traffic



4.1 State: Idle

ContentionWindow = 0;
Wait for !QueueEmpty;
/* The CPE may get an unicast REQ slot in the idle state. In this case it returns the

```
   current reqWin value */
 if (unicast REQ SID == mySID)      /* Polling case */
 {
  Transmit REQ in reservation;
  Tx_slot = slot;
 PrevREQ = NonContREQ;
 }
if (NormalGrantId == mySID)
 Utilize Normal Grant();
else if (UnsolicitedGrantId == mySID)
 {
  Utilize  Unsolicited Grant();
  Go to State Unsolicited Grant Pending;
 }

/* EPDU Arrives */

Enqueue();
CalculateDefer();
Go to State Deferring
```

4.2 State: Deferring

```
if (UnsolicitedGrantId == mySID)           /* Unsolicited Grant Service */
 {
  Utilize Unsolicited Grant();
  Go to State Unsolicited Grant Pending;
 }
else if (NormalGrantId == mySID)
  Utilize Normal Grant();
else if (unicast REQ SID == mySID)      /* Polling case */
 {
  Transmit REQ in reservation;
  Tx_slot = slot;
  Go to Grant Pending;
  PrevREQ = NonContREQ;
 }
else
 {
  for (REQ Transmit Opportunity)  /* Contention based REQ transmission */
   {
    if (Defer != 0)
     Defer = Defer - 1;
   else                          /* Defer = 0 */
    {
      if (Number of SIDs in CPE, with Defer = 0 is greater than 1)
```

```
        choose one SID at random;
      if (my SID chosen)
       {
       Transmit REQ in contention;
       Tx_slot = slot;
       RTxTime = time_now;
       PrevREQ = ContREQ;
       Go to Grant Pending;
      }
     }
   }
 }
```

4.3 State: Grant Pending

```
Wait for next MAP;

Move ACK pointer as per ACK field in MAP;
The next byte to transmit is set as per ACK/NACK flag and
Sequence Number in the ACK
if (Flush EPDU field set)
 {
  Flush HOL EPDU;
  Go to Idle;
 }

 if (unicast REQ SID == mySID)      /* Polling case */
 {
  Transmit REQ in reservation;
  Tx_slot = slot;
  PrevREQ = NonContREQ;
 }

if  (Normal GrantId == mySID)
 Utilize Normal Grant();
else if (Unsolicited GrantId == mySID)
 {
  Utilize Unsolicited Grant();
  Go to State Unsolicited Grant Pending;
 }
else if  (implicit collision indication received)
 Retry();
else        /* Error Condition: BSC did not give grant that CPE is expecting */
  Go to Idle;
```

4.4 State: Grant Pending Absent

```
If (First Time Entering State)
Count  =  GrantPendingWait;
else
  --Count;

 if (unicast REQ SID == mySID)      /* Polling case */
 {
   Transmit REQ in reservation;
   Tx_slot = slot;
  PrevREQ = NonContREQ;
 }
```

4.5 State: Unsolicited Grant Pending

```
 if (unicast REQ SID == mySID)      /* Polling case */
 {
   Transmit REQ in reservation;
   Tx_slot = slot;
   PrevREQ = NonContREQ;
 }

if (Unsolicited GrantId == mySID)
 {
   Utilize Unsolicited Grant();
   Remain in State Unsolicited Grant Pending;
 }

if (Last Unsolicited Grant)
 Go to state Idle;
```

4.6 State: Unsolicited Grant Absent

```
If (First Time Entering State)
  Count = UnsolicitedGrantPendingWait;
else
  --Count;

 if (unicast REQ SID == mySID)      /* Polling case */
 {
   Transmit REQ in reservation;
   Tx_slot = slot;
  PrevREQ = NonContREQ;
```

```
}
```

4.7 Function: CalculateDefer()

```
If (ContentionWindow < Start)
 Window = Start;

If (ContentionWindow > End)
 Window = End;

Defer = Random[2^ContentionWindow];
```

4.8 Function: Utilize Normal Grant()

```
If (Grant Size == 0)     /* Scheduler not able to make grant during this frame */
 Go to Grant Pending;
else                          /* Grant Size > 0 */
 {
  while (GrantSID == mySID)      /* Multiple Grants in MAP */
   {
   Extract Indicated number of bytes from SID queue;
   Confirm that these bytes fit in the tick space allocated;
   piggyback size = RequestWindow;
   Transmit WPDU with Sequence Number Field set as per MAP and
   Piggyback field set as above;
   }


  if (piggyback size > 0)
  {
   Go to Grant Pending;
   RTxTime = time_now;
   PrevREQ = NonCOntREQ;
  }
  else                           /* No more bytes left in SID queue */
   Go to GrantPendingWait;
 }
```

4.9 Function: Utilize Unsolicited Grant()

```
while (GrantSID == mySID)      /* Multiple Grants in MAP */
   {
   Extract Indicated number of bytes from SID queue;
   if (#bytes == 0)
     Tx only WPDU header;
```

```
    else
     {
     Confirm that these bytes fit in the tick space allocated;
      piggyback size = RequestWindow;
     Transmit WPDU with Sequence Number Field set as per MAC and
     Piggyback field set as above;
     }
   }
```

4.10 Function: Retry()

```
Retries = Retries + 1;
if (Retries > 16)
 {
  Discard HOL EPDU;
  Go to Idle;
 }
ContentionWindow = ContentionWindow + 1;
CalcDefer();
Go to Deferring;
```

4.11 Function: Enqueue()

```
Enqueue EPDU to tail of queue;
RequestWindow = RequestWIndow + Size of EPDU;
```