Realization of 802.11n and 802.11e models

M. Banchi S. Bracciali T. Pecorella

Dpt. di Elettronica e Telecomunicazioni
Università di Firenze, Italy.

NS3 Workshop, March 2 2009, Rome.

SIEMENS

Outline

SIEMENS

Standard 802.11e

The development of some aspects in the 11e Standard is mandatory for the 11n Standard. The new features can be summarized as follows:

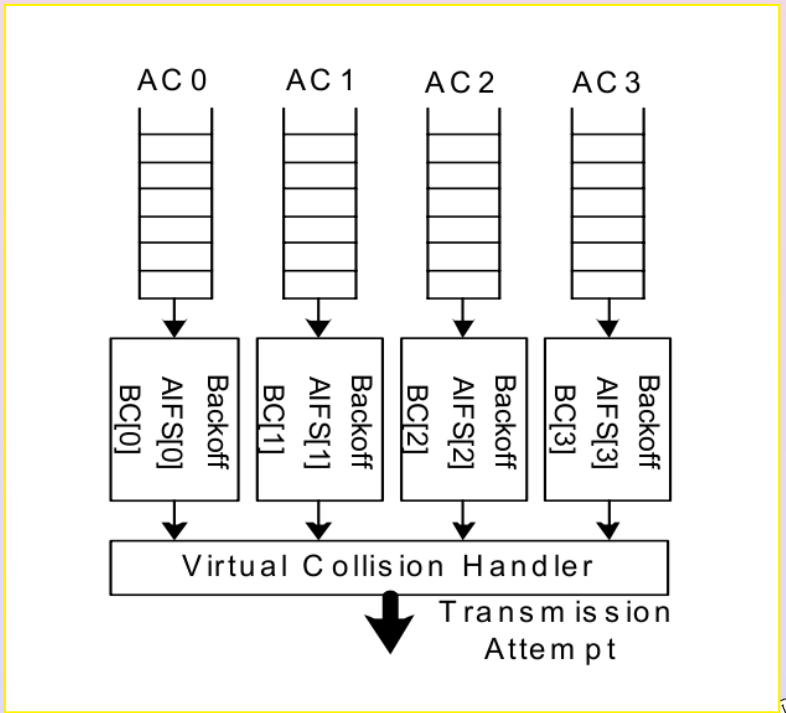
- AC queues mapping QoS traffic
- EDCA mechanism
- Block Ack (Added in 11e but enhanced in 11n)
- Multiple frame transmission in a Transmission Opportunity (optional)

The development work focuses as a pre-requisite for the 11n standard

SIEMENS

EDCA

EDCA deals with QoS traffic internal collision resolution.



- 4 Different Access Class (AC_BK, AC_BE, AC_VI, AC_VO)
- Each AC uses different parameters
- Collision procedure similar to DCF

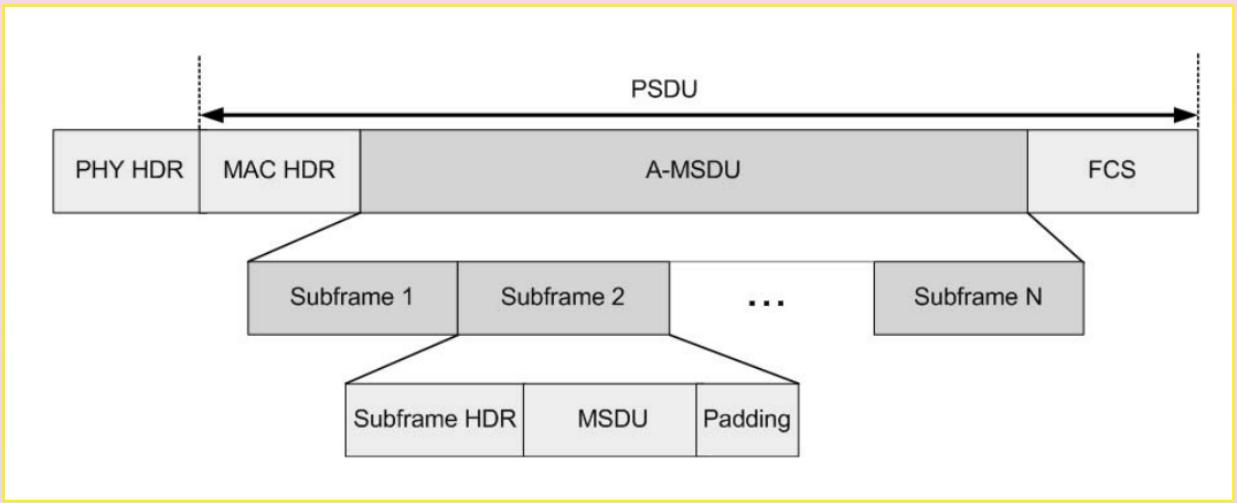
Standard 802.11n

Although is a draft standard, pre-n AP are already in the market. Currently the standard provides:

- Speeds up to 600 Mbit/sec
- OFDM modulation with MIMO technology
- Optional 40 Mhz channel
- Both 2.4 Ghz & 5 Ghz bandwidths
- Protection modes for backward compatibility
- New MAC feature: Frame Aggregation
- Optional LDPC coding
- Beamforming
- High Throughput terminals with Greenfield preamble
- Reverse Direction Protocol

A-MSDU

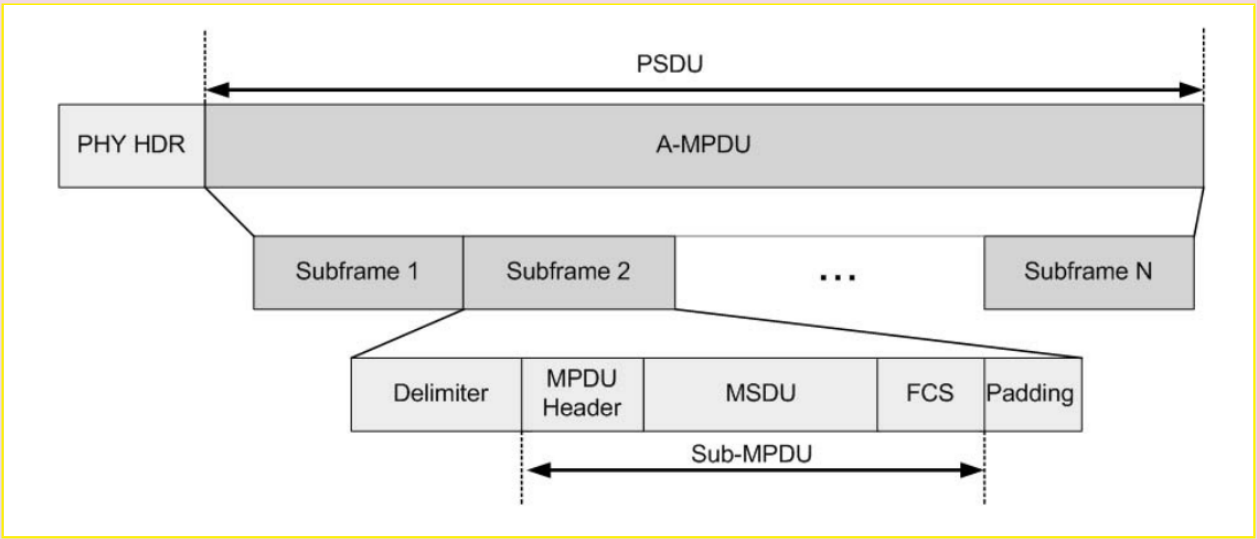
Aggregation on top of MAC level. If subframe failure occurs all the packets are discarded.



The A-MSDU maximum length is 7935 octets.

A-MPDU

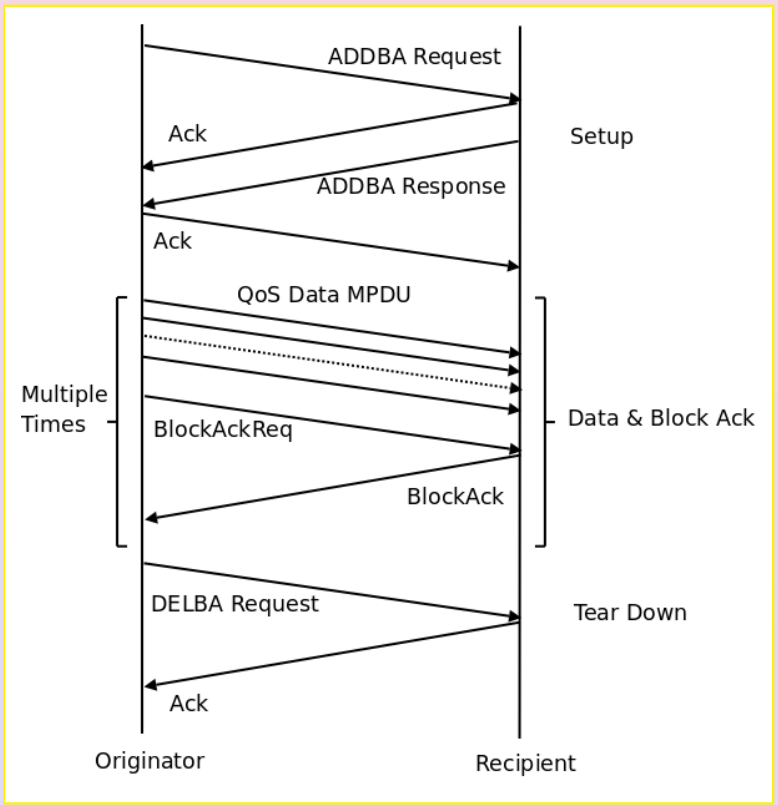
Aggregation on bottom of MAC level, each subframe can be recovered indipendently.



The A-MPDU maximum length is 65535 octets.

A-MSDU and A-MPDU can be merged, but in this case the A-MSDU can not exceeds 4065 octets.

Block Ack



Phases

- Setup
- Data & Block Ack
- Tear Down

Options

- Delayed or Immediate
- Multi Tid or Single Tid
- Normal or Compressed

NS3

Done

- Frame aggregation (A-MSDU)
- Basic Block Ack
- Compressed Block Ack

Lacks

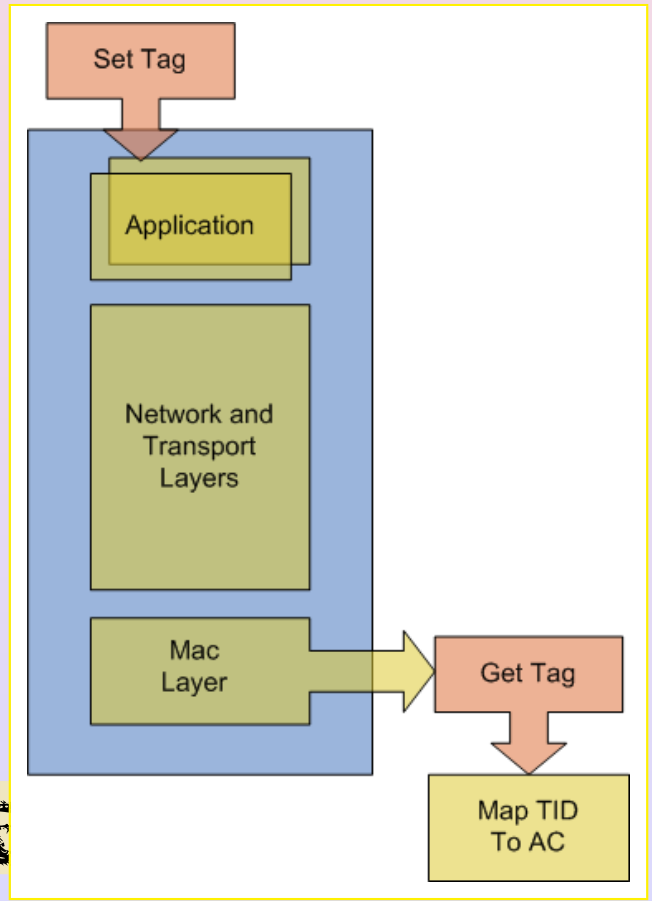
- Delayed Block Ack
- Multi Tid Block Ack

Next step

- A-MPDU

QoS Tag

Each application needs to add a QoS Tag that acts as a traffic marker:




```
void QosTag::Set (uint8_t tid);
```

```
uint8_t QosTag::Get (void) const
```

Transparent for Network and Transport Layers

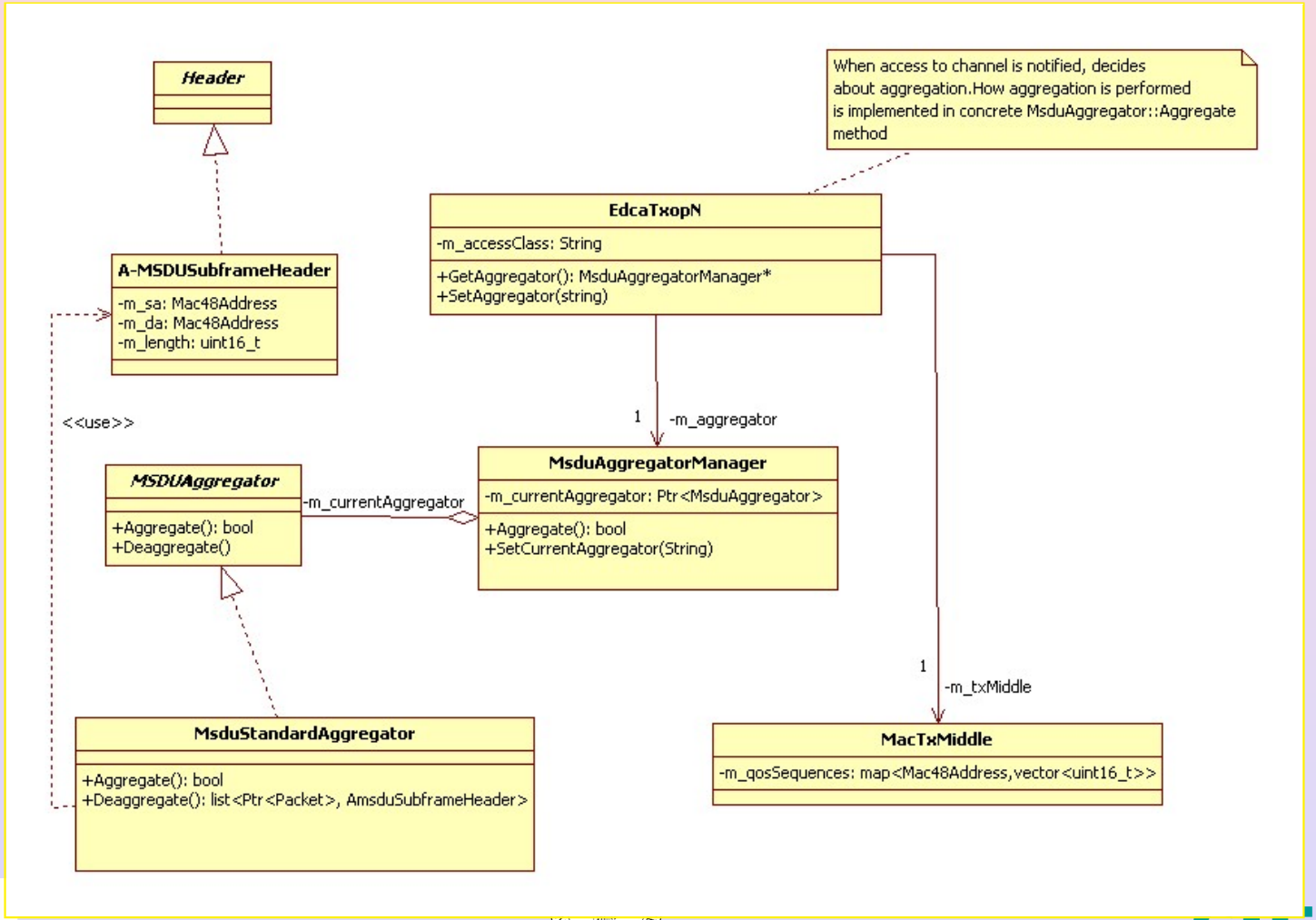
4 *ACs* map 8 *TIDs*

Additions

- Virtual internal collision management already present in NS-3 (ns3::DcfManager)
- New EdcaTxopN queue
- MacTxMiddle now assigns different sequence numbers per TID, per unique Receiver Address.
- Multiple queues:
  - 4 AC queues for data traffic
  - A management queue for AP beacon
- ns3::EdcaTxopN achieves aggregation (A-MSDU) and Block Ack negotation
- ns3::QstaWifiMac and ns3::QapWifiMac

SIEMENS

Frame aggregation (A-MSDU)



Frame Aggregation (A-MSDU)

- Every concrete aggregator has to inherit from abstract class ns3::MsduAggregator

```
class MsduAggregator :  public Object {
public:
virtual bool Aggregate (Ptr<const Packet> packet,
                        Ptr<Packet> aggregatedPkt,
                        Mac48Address src,
                        Mac48Address dest) = 0;
}
```

- Possibility of switching among different algorithms.

SIEMENS

Changes and additions in ns3::WifiMacQueue

- Methods to search in queue packets by addresses.

```
Ptr<const Packet> Dequeue (WifiMacHeader *hdr,
                           uint8_t addrIndex,
                           Mac48Address addr);
Ptr<const Packet> Peek (WifiMacHeader *hdr,
                        uint8_t addrIndex,
                        Mac48Address addr);
```

- Method to push packets in front of queue

```
void PushFront (Ptr<const Packet> packet,
                WifiMacHeader const &hdr);
```

SIEMENS

Frame aggregation (A-MSDU) steps

- When channel access is notified, EdcaTxopN creates A-MSDU.

```
m_aggregator.Aggregate (packet,aggrPkt,src,dest);
```

- A-MSDU is trasmitted as a normal packet.
- A-MSDU is deaggreated in ns3::QstaWifiMac::Receive or ns3::QapWifiMac::Receive.

```
if (hdr.IsQosAmsdu ()) {
   DeaggregateAmsduAndForward (packet, &hdr);
}
```

SIEMENS

A-MSDU Example

```
▷ Frame 38 (2182 bytes on wire, 2182 bytes captured)
▽ IEEE 802.11 QoS Data, Flags: o.....F..
    Type/Subtype: QoS Data (0x28)
  ▷ Frame Control: 0x8288 (Normal)
    Duration: 60
    Destination address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    BSS Id: 00:00:00_00:00:03 (00:00:00:00:00:03)
    Source address: 00:00:00_00:00:03 (00:00:00:00:00:03)
    Fragment number: 0
    Sequence number: 0
  ▷ Frame check sequence: 0x00000000 [incorrect, should be 0x684a7c6c]
  ▽ QoS Control
        Priority: 7 (Network Control) (Voice)
        ...0 .... = EOSP: Service period
        Ack Policy: Normal Ack (0x00)
        Payload Type: A-MSDU
        QAP PS Buffer State: 0x0
▽ IEEE 802.11 Aggregate MSDU
  ▷ A-MSDU Subframe #1
  ▷ A-MSDU Subframe #2
```

SIEMENS

Block Ack

Block Ack mechanism operates as follows:

1. Block Ack enabled when a queue lenght reaches X packets having same receiver as first packet dequeued.
2. Setup frame exchange
3. Single Tid data transmission
4. Packet is stored in a buffer
5. Other data transmission with EDCA
6. Block Ack frame exchange when X packets have been transmitted
7. Tear down frame exchange

SIEMENS

Block ack setup (1)

ADDBARequest/ADDBAResponse exchange is needed.

Originator

- ADDBARequest is sent by ns3::EdcaTxopN

```
void EdcaTxopN::SendADDBARequest (Mac48Address recipient, uint8_t tid,
                                  uint16_t startSeq, uint16_t bufferSize = 0,
                                  uint16_t timeout = 0, bool immediateBAck = true);
```

- ADDBAResponse is handled in ns3::MacLow which notifies reception of the response to ns3::EdcaTxopN.

```
void TransmissionListener::GotADDBAResponse(MgtADDBAResponseHeader const *respHdr,

                                            Mac48Address recipient);
```

SIEMENS

Block ack setup (2)

Recipient

- ADDBARequest is handled by ns3::QstaWifiMac or ns3::QapWifiMac in Receive method.

```
MgtActionFrameHeader actHdr;
packet->PeekHeader (actHdr);
if (actHdr.IsADDBARequest) {
...
    SendADDBAResponse (&reqHdr, hdr->GetAddr2 ());

}
```

- ADDBAResponse is created by ns3::QstaWifiMac or by ns3::QapWifiMac.

```
void QstaWifiMac::SendADDBAResponse (MgtADDBARequestHeader const *reqHdr,

                                     Mac48Address originator);
```

Block ack exchange

- Block ack request is created by ns3::EdcaTxopN.

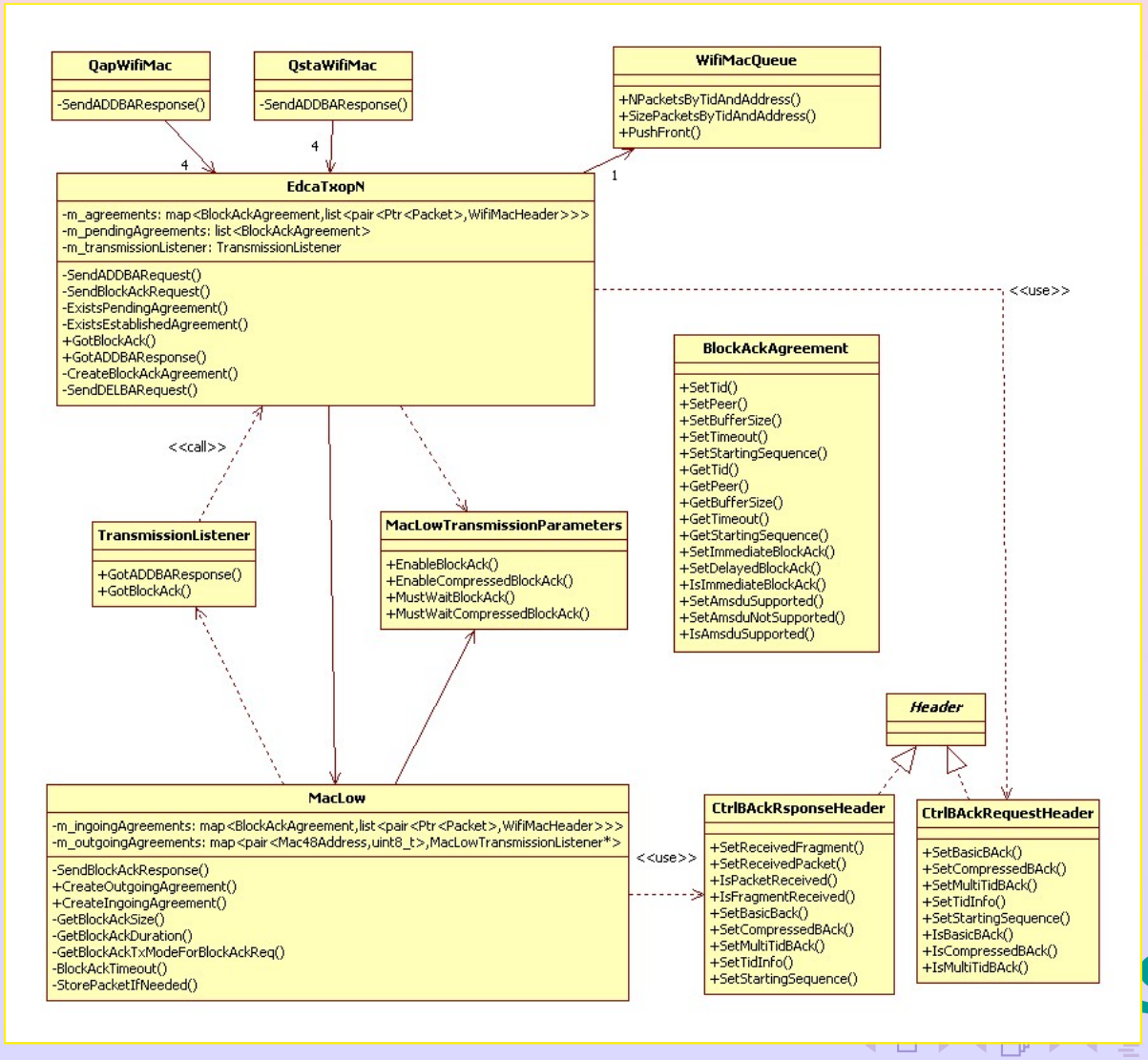
```
void EdcaTxopN::SendBlockAckRequest (Mac48Address dest, uint8_t tid, string type);
```

- Block ack in created by ns3::MacLow which is in charge of buffer all packets under a block ack agreement

```
void MacLow::SendBlockAckResponse (CtrlBAckRequestHeader const reqHdr,
                                   Mac48Address dest, Time duration, WifiMode mode);
```

SIEMENS

Class Diagram



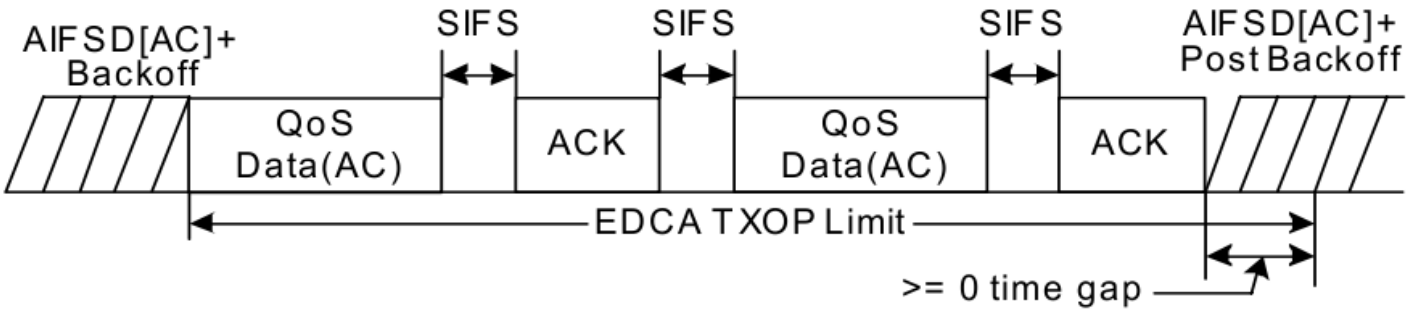
Block Ack Example

```
▷ Frame 34 (32 bytes on wire, 32 bytes captured)
▽ IEEE 802.11 802.11 Block Ack, Flags: o........
    Type/Subtype: 802.11 Block Ack (0x19)
  ▷ Frame Control: 0x8094 (Normal)
    Duration: 0
    Receiver address: 00:00:00_00:00:03 (00:00:00:00:00:03)
    Transmitter address: 00:00:00_00:00:02 (00:00:00:00:00:02)
    Block Ack Request Type: Compressed Block (0x02)
  ▽ Block Ack (BA) Control: 0x0004
      .... .... .... ...0 = BAR Ack Policy: Sender Does Not Require Immediate Acknowledgement
      .... .... .... ..0. = Multi-TID: False
      .... .... .... .1.. = Compressed Bitmap: True
      .... 0000 0000 0... = Reserved: 0x0000
      0000 .... .... .... = TID for which a Basic BlockAck frame is requested: 0x0000
  ▽ Block Ack Starting Sequence Control (SSC): 0x0010
      .... .... .... 0000 = Fragment: 0
      0000 0000 0001 .... = Starting Sequence Number: 1
    Block Ack Bitmap
  ▷ Frame check sequence: 0x00000000 [incorrect, should be 0xca0c5734]
```

```
0000  94 80 00 00 00 00 00 00  00 03 00 00 00 00 00 02   ........ ........
0010  04 00 10 00 07 00 00 00  00 00 00 00 00 00 00 00   ....???? ????....
```

Multiple frame Transmission

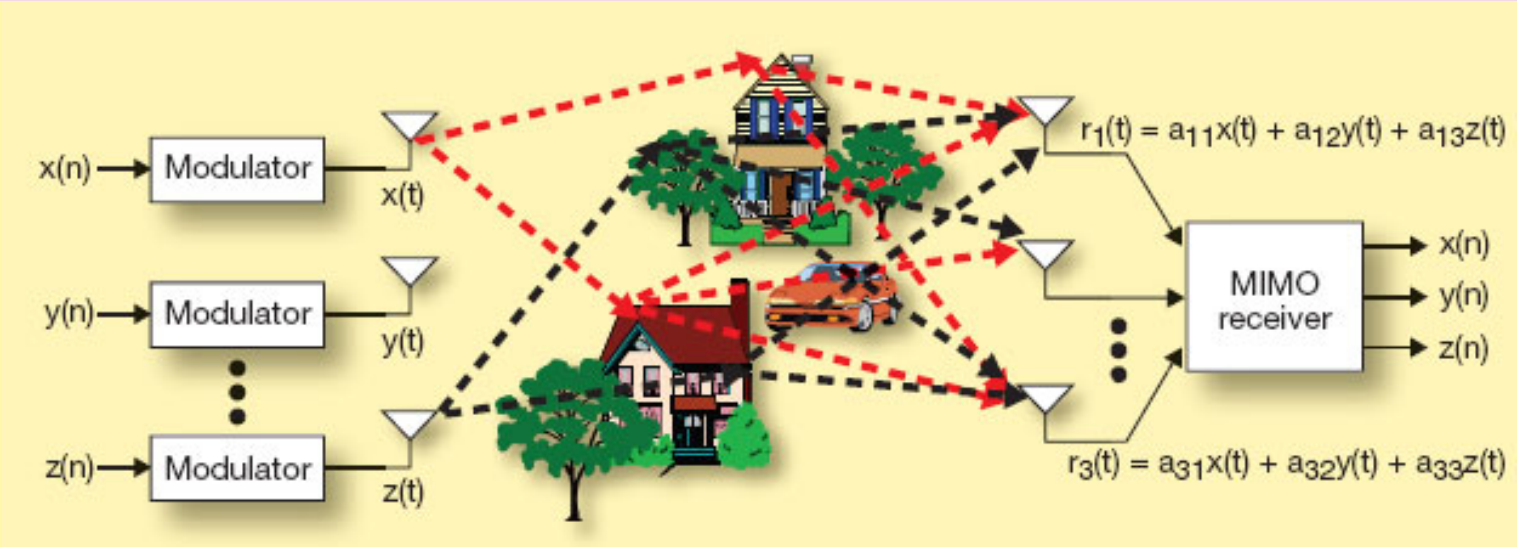
Initiation of the TXOP occurs when the EDCA rules permit access to the medium.
Each frame wait a SIFS time before being transmitted.



Implementation needs a correct NAV setting and changes to the Backoff procedure

MIMO

802.16 <-> 802.11



- Multiple channel transmission (good also for multiple SSID simulations)
- STBC channel error model

Support for 20/40 Mhz Transmission

Throughput enhancement with HT terminal needs:

Standard

- Greenfield preamble

Code development

- New MAC header

SIEMENS

Support for 20/40 Mhz Transmission

Throughput enhancement with HT terminal needs:

Standard

- Greenfield preamble
- Protection modes

Code development

- New MAC header
- Co-channel a/b/g/n terminals

Support for 20/40 Mhz Transmission

Throughput enhancement with HT terminal needs:

Standard

- Greenfield preamble
- Protection modes
- 20/40 Mhz bandwidth

Code development

- New MAC header
- Co-channel a/b/g/n terminals
- Dynamic channel

Support for 20/40 Mhz Transmission

Throughput enhancement with HT terminal needs:

Standard

- Greenfield preamble
- Protection modes
- 20/40 Mhz bandwidth
- Up to 600$Mbit/s$

Code development

- New MAC header
- Co-channel a/b/g/n terminals
- Dynamic channel
- New rates/channel model

SIEMENS

END

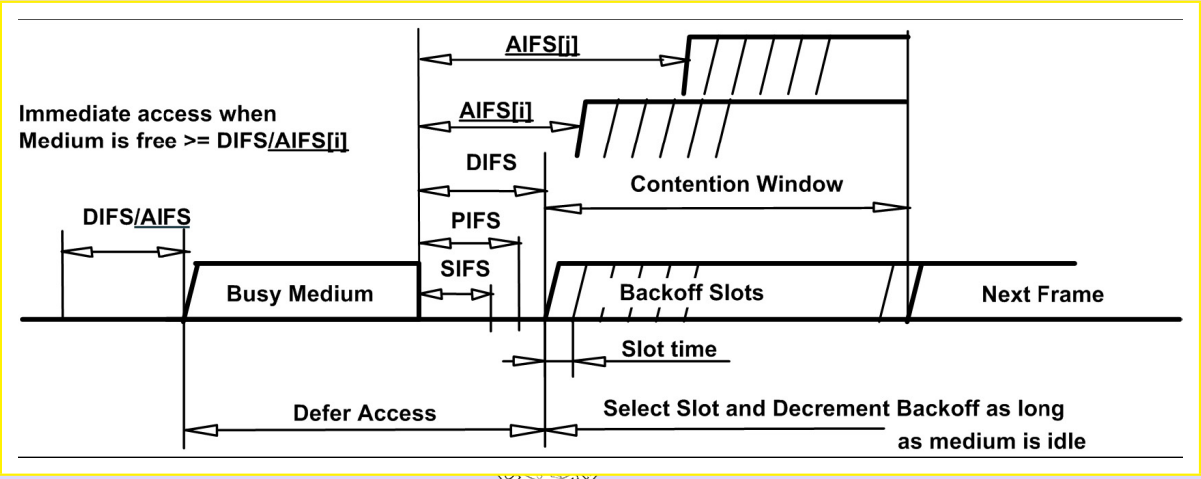
Thanks for the attention !

SIEMENS

EDCA

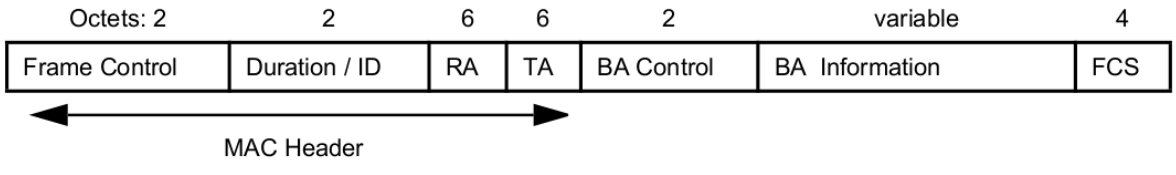
EDCA is the evolution of DCF, with CSMA/CA channel access and represents the Contention part of the Hybrid Contention Function. New features:

- AC queues
- DCF parameters are different for each AC
- Transmission Opportunity

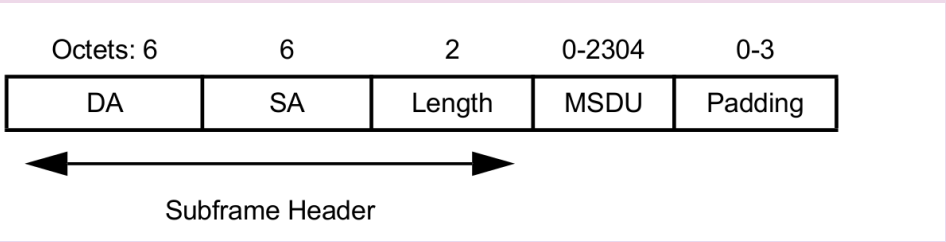


Frames

BlockAck



A-MSDU



A-MPDU