9 MAC throughput enhancements

Early on in the 802.11n standardization process it was recognized that even with significantly higher data rates in the PHY the fixed overhead in the MAC protocol was such that little of that gain would be experienced above the MAC. It was clear, as this chapter will show, that without throughput enhancements in the MAC the end user would benefit little from the improved PHY performance.

The throughput enhancements introduced in 802.11n are reused in 802.11ac. Some aspects of 802.11n, particularly A-MPDU and HT-immediate block ack, that were challenging to implement at the time 802.11n was developed are now widely adopted.

A station that implements the 802.11n features is called a high throughput (HT) STA. A station that implements the 802.11ac features is called a very high throughput (VHT) STA. Since 802.11ac is built on 802.11n, a VHT STA is also an HT STA.

9.1 Reasons for change

Since the original 802.11 specification was completed, a number of amendments have introduced new PHY capabilities and with them enhanced performance. In addition, the 802.11e amendment, which primarily added QoS features, also enhanced MAC performance with the introduction of the TXOP concept and block acknowledgement. However, these MAC performance improvements were only slight, and with the potential for significantly higher PHY performance it was soon realized that the existing MAC protocol did not scale well with PHY data rate.

9.1.1 Throughput without MAC changes

The poor scaling of throughput above the MAC with PHY data rate is illustrated in Figure 9.1 where the theoretical throughput is given for unicast data sent from one station to another assuming a 3 ms TXOP limit, block ack protocol, and a 10% packet error rate (PER). As the PHY data rate is increased beyond the 54 Mbps peak data rate of 802.11a/g, throughput begins to level off. A 40 MHz 2 × 2 system with a 270 Mbps PHY data rate only achieves 92 Mbps above the MAC. Worse, a 40 MHz 4 × 4 system with a 540 Mbps PHY data rate achieves almost exactly the same throughput.

The drop in efficiency (Figure 9.2) with increasing data rate results from the fixed overhead in the preamble and inter-frame space. Not only does this overhead account for

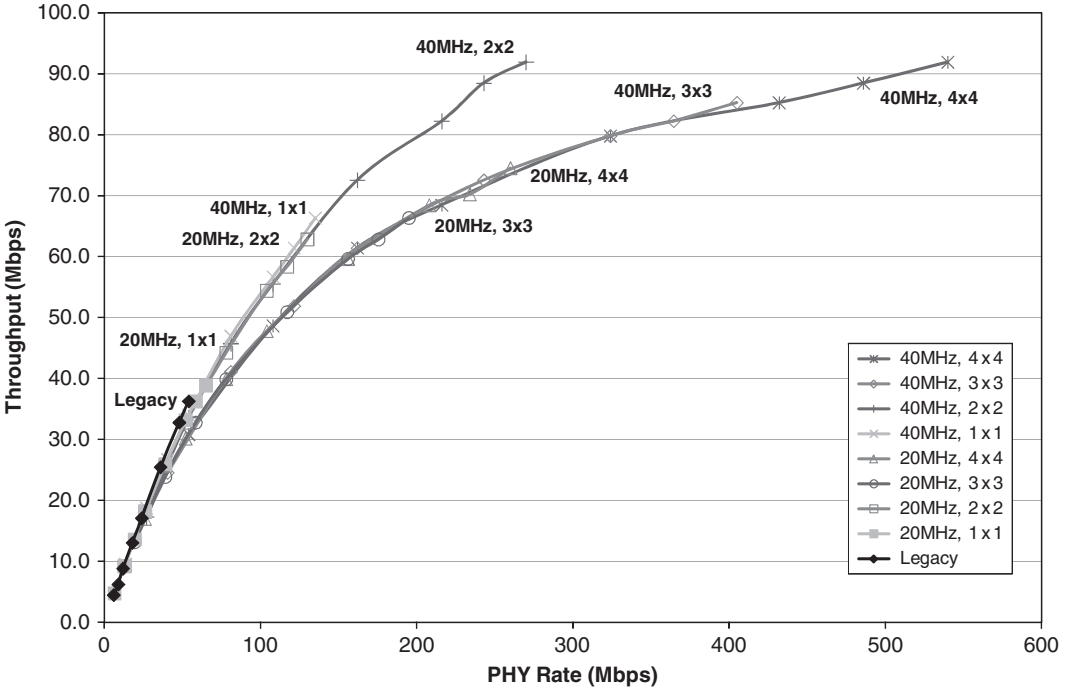

Figure 9.1 MAC throughput with HT PHY assuming no MAC changes (3 ms TXOP limit, block ack, 10% PER).

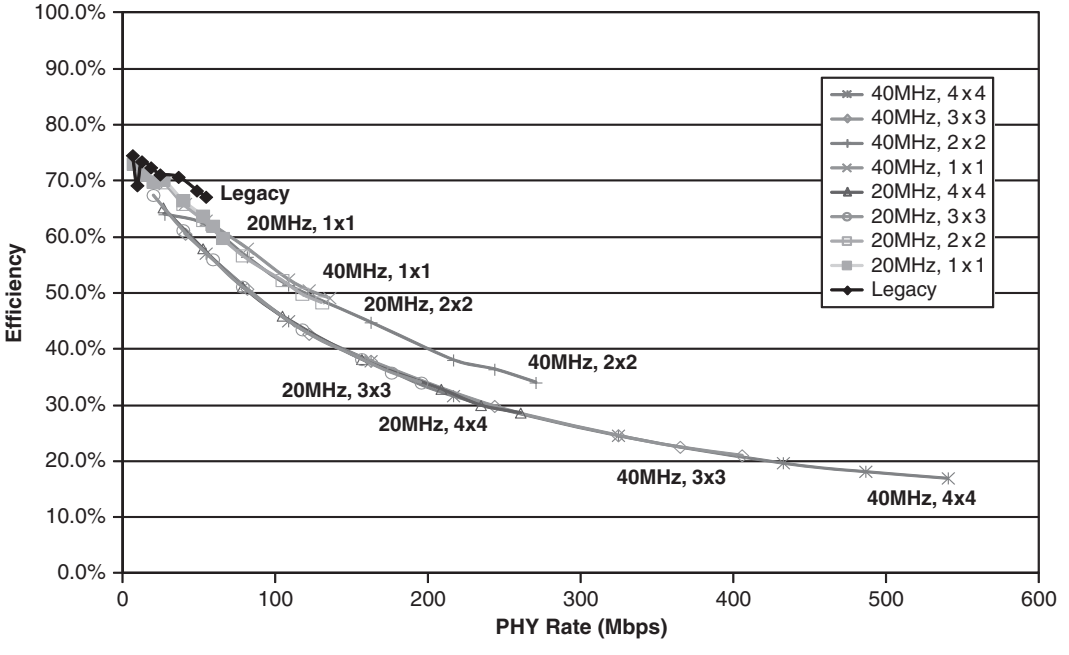

Figure 9.2 MAC efficiency with HT PHY assuming no MAC changes (3 ms TXOP limit, block ack, 10% PER).

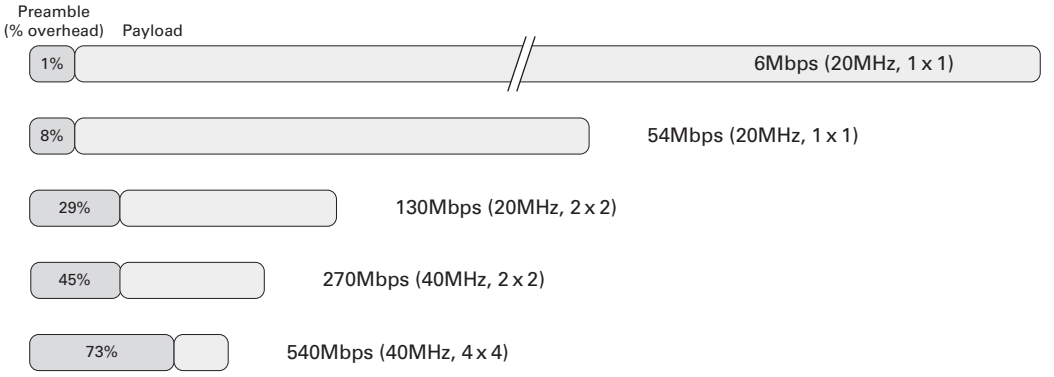

Figure 9.3 Relative preamble overhead for a 1500 byte frame at different PHY data rates.

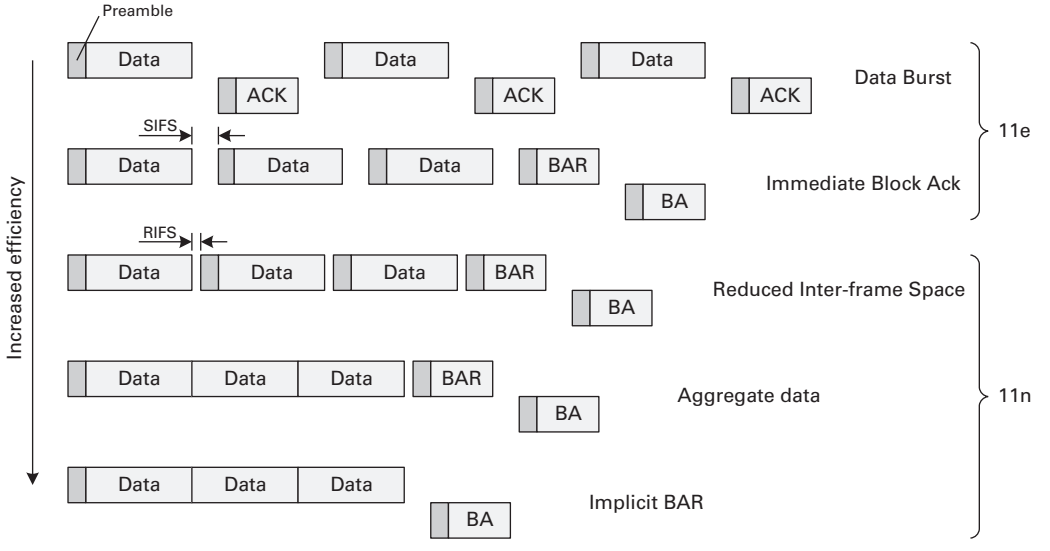

Figure 9.4 Basic throughput enhancements to the 802.11 MAC.

a greater percentage of the air time as the data payload gets shorter in duration, but the preamble itself needs to be longer to support the multiple spatial streams at the higher data rates introducing additional fixed overhead. The relative preamble overhead for a typical 1500 byte data frame is illustrated in Figure 9.3 for a select set of data rates.

As the payload gets shorter in duration and the preamble length increases in duration we see diminishing returns reducing the air time occupied by a single frame. Clearly changes are needed to improve efficiency if applications are to see any significant gain in throughput.

9.1.2 MAC throughput enhancements

The 802.11n amendment developed a number of simple enhancements to the 802.11e MAC that significantly improved efficiency. The most important of these are summarized in Figure 9.4.

The first two sequences in Figure 9.4 show data bursting within a TXOP, features that are supported with the 802.11e amendment. The first of these sequences shows data bursting using normal ack and the second using immediate block ack. The block ack protocol allows data frames to be grouped together and is the key to further efficiency improvements introduced in 802.11n.

An easy enhancement under block ack is to reduce the inter-frame space for back-to-back transmissions during the data burst. Since the station remains in transmit mode for the duration of the burst, there is no need for the long SIFS between frames, a duration that was originally designed to accommodate receive to transmit switching. With back-to-back transmissions the inter-frame space need only be long enough for the receiver to re-arm for acquisition of the new signal.

Taking it a step further, one could eliminate the inter-frame space and preamble altogether and concatenate data frames in a single transmission. In 802.11n this is referred to as aggregation and is the key throughput enhancing feature introduced in the 802.11n MAC.

An additional enhancement that was considered was to concatenate the BAR frame with the data frames, improving efficiency slightly. This, however, reduces robustness since the BAR frame is transmitted at the end of the aggregate transmission and at the data rate rather than the more robust MCS used for control frames. Instead, it was recognized that one function of the BAR frame – soliciting a BA frame – could be performed with a single bit piggybacked on each of the data frames making up the aggregate. This change is both more efficient and more robust since it eliminates the single point of failure in the BAR frame being sent at the end of the aggregate transmission and at the data rate. As long as one of the data frames making up the aggregate gets through, the recipient will respond with a BA frame.

Recognizing that fragmentation has little benefit at high data rates, particularly when data aggregation is being performed, it is also possible to reduce the size of the BA frame. Compressing the BA frame so that it only acknowledges MSDUs and not MSDU fragments further enhances efficiency.

All these techniques and more have been adopted with the 802.11n amendment and will be described in detail in this and the following chapters.

9.1.3 Throughput with MAC efficiency enhancements

With these basic MAC efficiency enhancements, the 802.11n system performs as shown in Figure 9.5. Notice that throughput now scales near linearly with PHY data rate primarily as a result of aggregation, which allows for long data transmissions bounded by the TXOP limit. Implicit BAR and the use of the compressed BA frame format improve efficiency further.

Throughput of 100 Mbps at the top of the MAC, a target set in the 802.11n PAR (IEEE, 2006), is now easily reached with a PHY data rate of roughly 130 Mbps, achievable in 20 MHz bandwidth with two spatial streams or in 40 MHz with one spatial stream.

The improved efficiency is shown in Figure 9.6. MAC efficiencies of between 70% and 80% are maintained over the full PHY data rate range. In fact, efficiency is

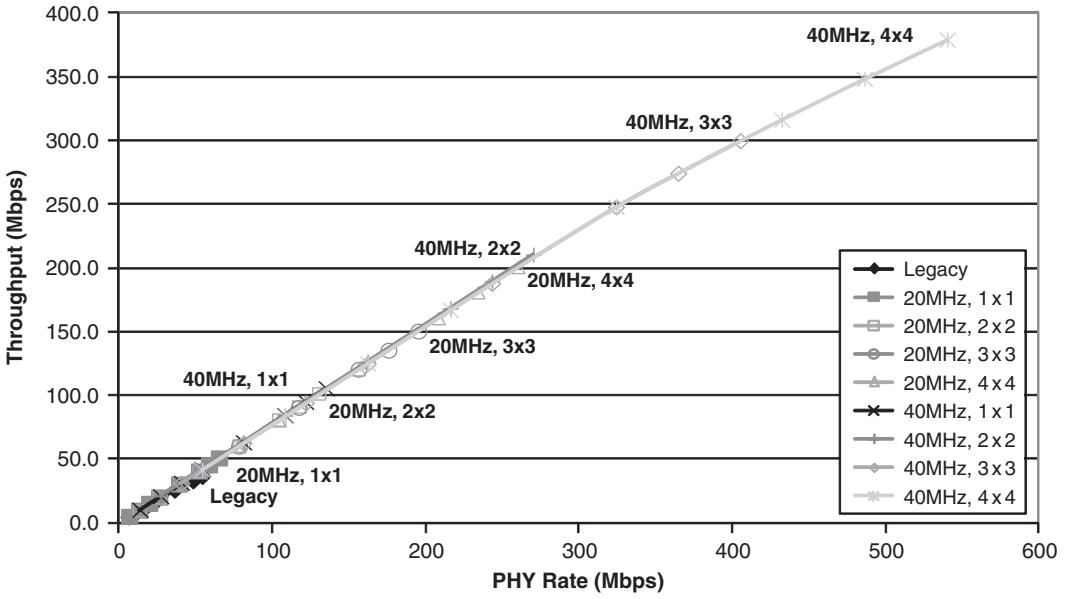

Figure 9.5 MAC throughput with HT PHY and MAC enhancements (3 ms TXOP limit, block ack, 10% PER).

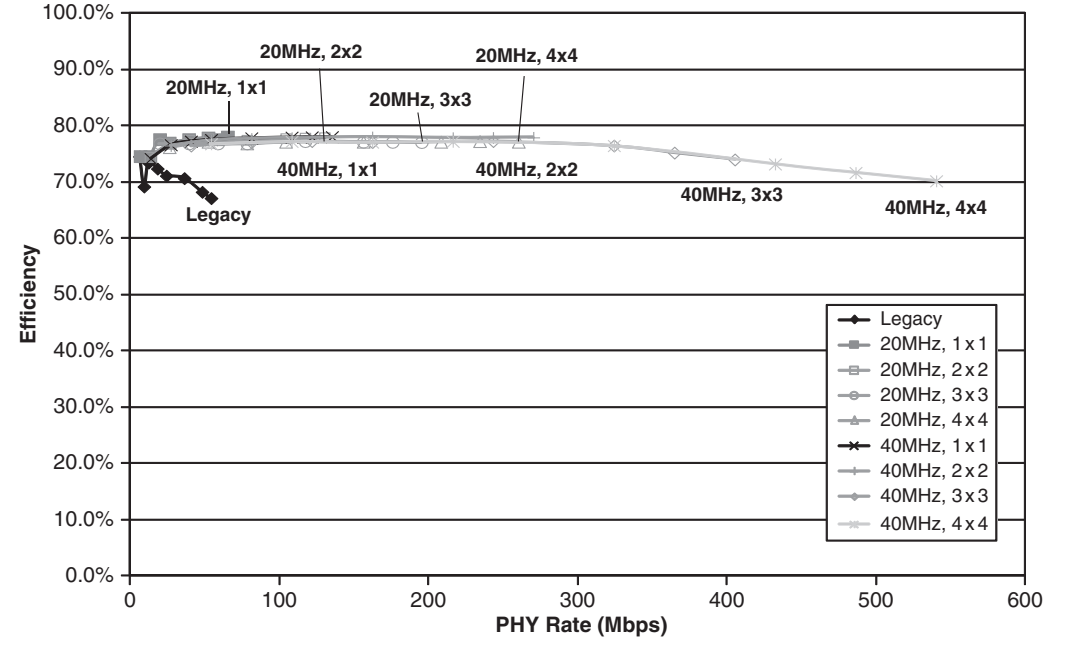

Figure 9.6 Efficiency vs PHY data rate with basic MAC enhancements (3 ms TXOP limit, block ack, 10% PER).

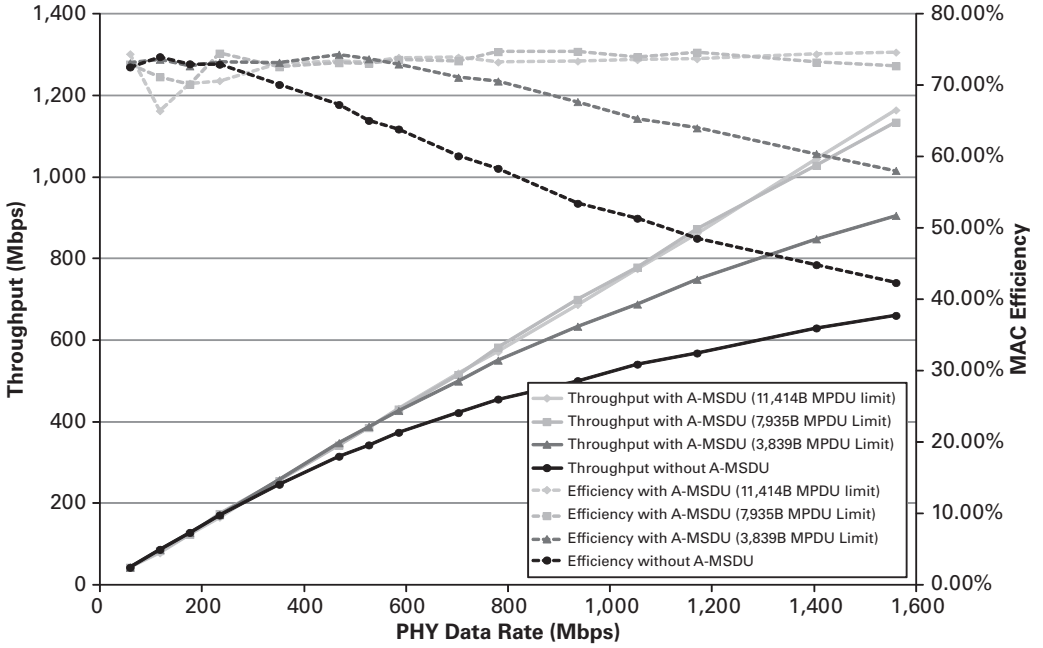

Figure 9.7 MAC throughput and efficiency with VHT PHY using A-MPDU and A-MSDU (1.5 ms TXOP limit, block ack, 10% PER).

improved over 802.11e even at legacy data rates (54 Mbps and below). The legacy plot reflects 802.11a/g PHY data rates with the 802.11e protocol.

The slight roll off in efficiency at very high PHY data rates is the result of hitting the block ack window limit of 64 MSDUs, which starts to limit the aggregate transmission length. This can be overcome using a combination of A-MSDU and A-MPDU aggregation techniques as described later in this chapter. In fact, the use of both A-MSDU and A-MPDU aggregation is necessary to take advantage of the higher PHY data rates provided by the VHT PHY as shown in Figure 9.7.

9.2 Aggregation

Early on in the 802.11n standardization process it was recognized that some form of aggregation would be required. Three techniques were proposed of which two were ultimately adopted in the standard. These two types of aggregation logically reside at the top and bottom of the MAC as illustrated in Figure 9.8.

The third aggregation technique, which was ultimately not adopted, logically resides at the top of the PHY. This is discussed briefly later in this section as it has some interesting characteristics.

At the top of the MAC is MSDU aggregation (or A-MSDU), which in the egress direction aggregates MSDUs as the first step in forming an MPDU. At the bottom of the

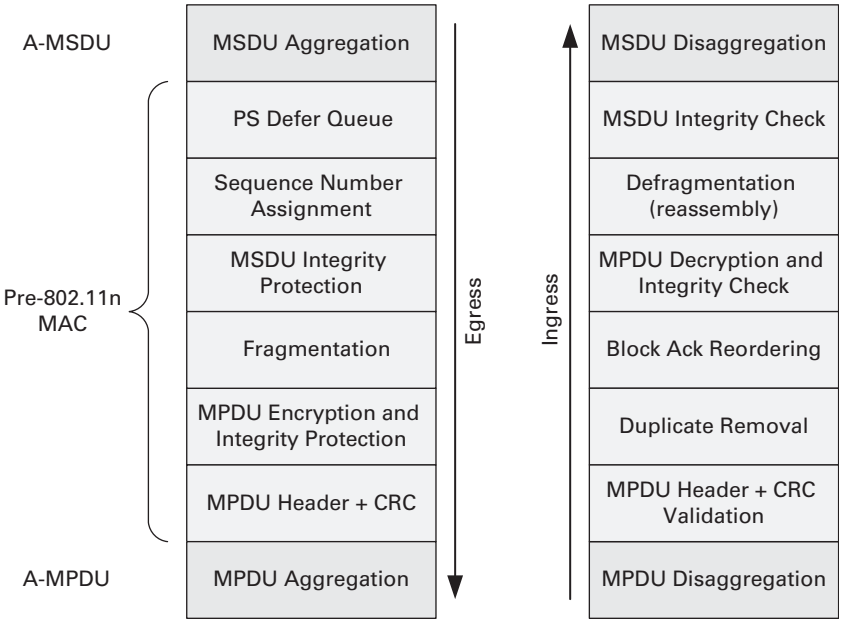

Figure 9.8 Two aggregation layers in relation to other MAC functions.

MAC is MPDU aggregation (or A-MPDU), which in the egress direction aggregates multiple MPDUs to form the PSDU that is passed to the PHY to form the payload of a single transmission. The reverse functions, MPDU and MSDU disaggregation, reside in the same logical positions in the ingress direction.

9.2.1 Aggregate MSDU (A-MSDU)

With A-MSDU, MAC service data units (MSDUs) received from the LLC and destined for the same receiver and of the same service category (same traffic identifier or TID) may be accumulated and encapsulated in a single MAC protocol data unit (MPDU). The encapsulation is shown in Figure 9.9.

The MSDU as received from the LLC is prefixed with a 14 byte subframe header consisting of the destination address (DA), source address (SA), and a length field giving the length of the SDU in bytes. The header together with the SDU is padded with 0 to 3 bytes to round the subframe to a 32-bit word boundary. Multiple such subframes may be concatenated together to form the payload of the QoS Data frame, provided the total length of the data frame does not exceed the maximum MPDU size.

Support for A-MSDU is mandatory at the receiver under Normal Ack policy. Support is negotiated for use under Block Ack policy during the block ack establishment handshake. The maximum length A-MSDU that a station can receive in an HT PPDU is declared in its HT Capabilities information element as either 3839 bytes

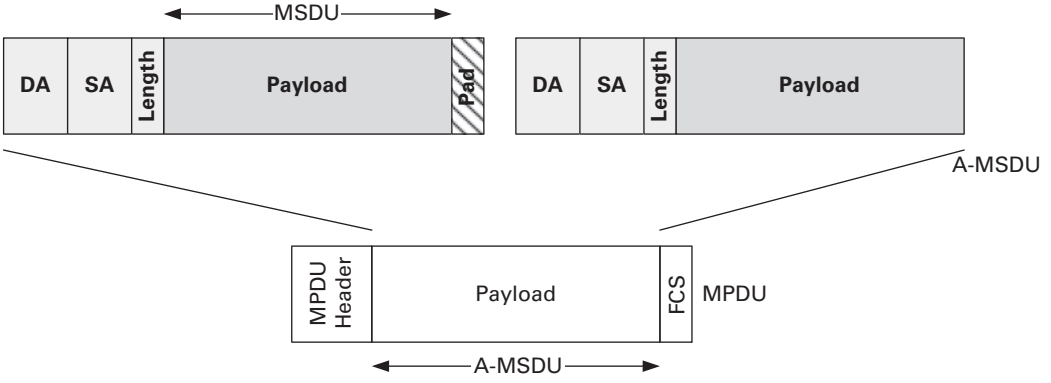

Figure 9.9 A-MSDU encapsulation.

or 7935 bytes. These limits are derived from implementation considerations. Some early 802.11n draft 2.0 systems were only capable of allocating a single 4 kB buffer per A-MSDU and, allowing for some implementation-specific use of the buffer (257 bytes), one arrives at the lower limit figure. The upper limit is arrived at by assuming an 8 kB receive buffer size and again allowing for some implementation specific use.

The maximum length A-MSDU that a station can receive in a VHT PPDU is determined by the maximum MPDU length declared in its VHT Capabilities element. Possible values for the maximum MPDU length are 3895 bytes, 7991 bytes and 11 454 bytes. The lower two values are arrived at by taking the HT PPDU A-MSDU length limits and adding MAC header overhead. The 11 454 byte limit was chosen because it is the maximum frame length for which the 32-bit FCS detects all three, two, or one bit errors (Jain, 1990).

The channel access rules for a QoS Data MPDU carrying an A-MSDU are the same as a data MPDU carrying an MSDU of the same TID. The maximum lifetime of an A-MSDU is the lifetime of the oldest constituent MSDU.

9.2.2 Aggregate MPDU (A-MPDU)

With A-MPDU, fully formed MAC PDUs are logically aggregated at the bottom of the MAC. A short MPDU delimiter is prepended to each MPDU and the aggregate presented to the PHY as the PSDU for transmission in a single PPDU. A-MPDU encapsulation is illustrated in Figure 9.10.

The MPDU delimiter is 32 bits in length and consists of an EOF field, a 3-bit reserved field, a 12-bit MPDU length field, an 8-bit CRC field, and an 8-bit signature field. The 8-bit CRC covers the 4-bit reserved and 12-bit length fields and validates the integrity of the header. The signature byte is set to the ASCII character "N" and was intended to aid software implemented disaggregation. The MPDU, unless it is the last in the A-MPDU, is padded with 0–3 bytes so that the subsequent A-MPDU delimiter is aligned on a 32-bit word boundary.

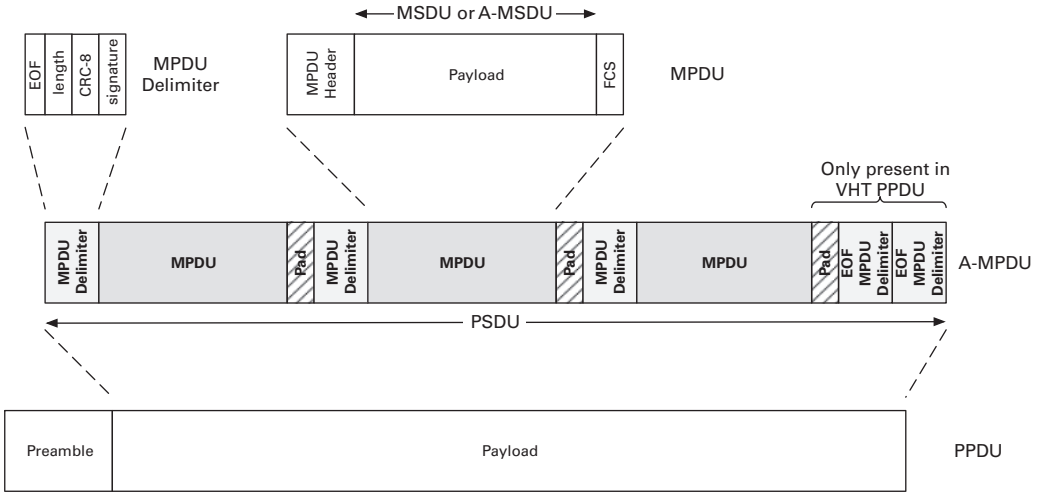

Figure 9.10 A-MPDU encapsulation.

A receiver implementation parses the A-MPDU framing structure by using the length in each delimiter to extract the following MPDU. If a delimiter is corrupt (has an out of range length value, or an invalid signature or CRC-8) then the receiver can scan forward looking for the next valid delimiter on a 32-bit boundary. The receiver should re-sync on an actual delimiter (as opposed to a random 32-bit word that appears to be a delimiter) with high probability and be able to extract the MPDU and subsequent MPDUs. In this sense the A-MPDU framing structure is robust since it is possible to recover MPDUs following a corrupt delimiter.

All the MPDUs in an A-MPDU are addressed to the same receiver and all are of the same service category (same TID). The Duration/ID field in the MAC header of all MPDUs in an A-MPDU is set to the same value.

A-MSDU aggregation may be used together with A-MPDU aggregation if negotiated through the block acknowledgement handshake. This typically offers little advantage over straight A-MPDU aggregation at low PHY data rates, but becomes essential for maintaining MAC efficiency at high PHY data rates (greater than around 400 Mbps). This is because at high data rates the PPDU length is limited by the block ack window limit of 64 outstanding MSDUs.

9.2.2.1 A-MPDU contents

All MPDUs within an A-MPDU are addressed to the same receiver address. This restriction simplifies the frame sequences that would need to be supported but it also means that a station can read the MAC header of the first MPDU in the aggregate and immediately shut down receive processing for the remainder of the PPDU to conserve power if it is not addressed to it.

The Duration/ID fields in the MAC headers of all MPDUs in an A-MPDU carry the same value. There are also restrictions on the types of MPDUs that can be carried in an A-MPDU.

Under HT-immediate block ack, an A-MPDU may carry:

- a single BA as the first MPDU in the aggregate
- QoS Data MPDUs belonging to the same TID and subject to the constraints of the block ack protocol
- any management MPDU of subtype Action No Ack.

Under HT-delayed block ack, an A-MPDU may carry:

- one or more BA MPDUs with the BA Ack Policy field set to No Ack
- QoS Data MPDUs belonging to the same TID and subject to the constraints of the block ack protocol
- any management MPDU of subtype Action No Ack
- BAR MPDUs with the BA Ack Policy field set to No Ack.

Additional MPDU types are allowed with PSMP and these are detailed in Section 10.4.

The fragmentation of QoS Data MPDUs is not permitted in an A-MPDU.

9.2.2.2 A-MPDU length and MPDU spacing constraints

A station advertises the maximum A-MPDU length that it can receive in an HT PPDU in its HT Capabilities element. The advertised maximum length may be one of the following: 8191, 16 383, 32 767, or 65 535 octets. A station advertises the maximum A-MPDU length that it can receive in a VHT PPDU in its VHT Capabilities element. The advertised maximum length may be one of the following: 8 191, 16 383, 32 767, 65 535, 131 071, 262 143, 524 287, or 1 048 575 octets. The HT PPDU limit is set to the same value as the VHT PPDU limit for values below 65 535 and to 65 535 otherwise. The sending station must not send an A-MPDU of greater length.

Some implementations have MPDU processing limitations and could be overwhelmed by sequences of short data frames. To prevent data loss through buffer overrun, a station may advertise a minimum MPDU start spacing such that MPDUs in an aggregate do not arrive at a rate faster than the implementation can process them. In its HT Capabilities element, such a station would indicate the minimum start spacing between MPDUs (see Section 12.3.2.2). This minimum start spacing constraints applies to both HT and VHT PPDUs.

A transmitting station ensures that when forming an aggregate it does not violate that spacing constraint. If MPDUs packed into an aggregate will violate the spacing constraint then the transmitter may send the MPDUs in separate transmissions, use A-MSDU to create larger MPDUs, or insert null MPDU delimiters to increase the spacing. A null MPDU delimiter contains a length field indicating a zero length MPDU.

9.2.3 Aggregate PSDU (A-PSDU)

During the proposal phase of the 802.11n development an aggregation scheme was proposed (Hansen and Edwards, 2004) that logically resides at the top of the PHY. This technique is illustrated conceptually in Figure 9.11. Essentially the proposal called for a single training sequence at the front of the PPDU followed by a framing structure

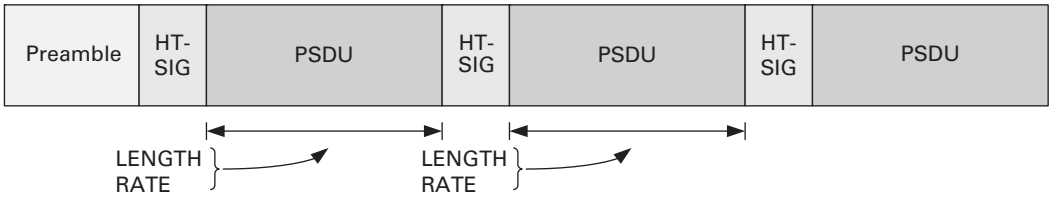

Figure 9.11 Early 802.11n proposal for A-PSDU encapsulation.

that consists of the PHY signaling field (HT-SIG) delimiting one or more PSDUs. The LENGTH and MCS fields in the PHY signaling field give the duration of the PSDU that followed. A bit in the HT-SIG would indicate the last HT-SIG in the aggregate.

The A-PSDU concept had the appealing characteristic that the data rate could be changed for each PSDU making up the aggregate. This would support multi-rate aggregation allowing the aggregation scheme to be used to efficiently send data to multiple receivers in the same burst.

Ultimately this scheme was rejected, as the far more common use case where data is aggregated to a single station is more efficiently handled using A-MPDU aggregation. For this usage scenario the A-PSDU technique had higher overhead (each HT-SIG field occupied at least one 4 μs symbol) and was less robust since a single error in one of the HT-SIG fields would prevent the remainder of the aggregate from being demodulated.

9.2.4 A-MPDU in VHT PPDUs

A VHT PPDU always carries an A-MPDU. Unlike HT PPDUs, a VHT PPDU does not have a length field in the PHY header to indicate the length of the MAC data contained in the PPDU. MPDUs are delineated using the length field in the MPDU delimiter, even if only one MPDU is carried in the PPDU.

In addition, the A-MPDU occupies all available octets in the VHT PPDU payload. The receiving PHY will forward all payload octets to the MAC, which must then be able to correctly parse the data stream.

The procedure by which the sending MAC creates an A-MPDU for a VHT PPDU is as follows. The A-MPDU is created by adding MPDUs prefixed by MPDU delimiters. Null MPDU delimiters are also inserted if needed to meet the recipient's MPDU start spacing requirements. The number of symbols required to transmit the A-MPDU is then calculated. The number of symbols is then translated back into a count of the octets available in the payload of the PPDU. End of frame (EOF) MPDU delimiters are then repeatedly appended to the A-MPDU while there are at least 4 octets available in the payload. An EOF MPDU delimiter is a delimiter with 0 in the length field and 1 in the EOF field. Finally, 0–3 arbitrary octets are appended to fill out the payload. The content of these last octets does not matter since they will not create a complete A-MPDU delimiter.

The EOF field in the EOF MPDU delimiters indicates to the recipient MAC that all MPDUs have been received and that receive processing can cease. This is not of great value in SU PPDUs, since only a few octets in the last symbol are padding, but is of more

use in MU PPDUs where multiple A-MPDUs are present in the PPDU, some of which may have multiple symbols worth of padding (see Section 14.5.1).

9.2.5 VHT single MPDU

An HT PPDU either carries a single MPDU as a non-aggregate or one or more MPDUs in an A-MPDU as an aggregate. Different rules may apply depending on whether the MPDU is carried in an aggregate or non-aggregate, for example the rules for generating an ACK or a BA frame response to a QoS Data frame with Normal Ack policy (see Section 9.4.1). A VHT PPDU always carries an A-MPDU. A method is thus required to emulate the behavior that applies to non-aggregate HT PPDUs. The method adopted by 802.11ac to do this is the VHT Single MPDU.

A VHT single MPDU is an MPDU that is carried as the only MPDU in an A-MPDU and that has the EOF field in the MPDU delimiter set to 1. A VHT single MPDU follows the same rules as a non-aggregate MPDU.

The reuse of the EOF field to indicate VHT single MPDU does not conflict with its use to indicate end of frame, since in this case the MPDU delimiter length field is always non-zero.

9.3 Block acknowledgement

The block acknowledgement mechanism was introduced in the 802.11e amendment to improve efficiency by allowing for the transfer of a block of data frames that are acknowledged with a single Block Acknowledgement (BA) frame instead of an ACK frame for each of the individual data frames. The station with data to send is referred to as the originator and the receiver of the data as the recipient.

Two flavors of block ack were originally defined in the 802.11e amendment: immediate block ack and delayed block ack. Both flavors are enhanced in the 802.11n amendment to improve efficiency and take advantage of aggregation and the higher data rates. The enhanced mechanisms are referred to as HT-immediate block ack and HT-delayed block ack. All four flavors may be supported by HT stations, although the original mechanisms would only be used for interoperability with legacy stations. No changes were made to the block ack mechanism in 802.11ac and the requirements for HT stations also apply to VHT stations.

In this section, the original 802.11e block ack mechanisms are described and then the HT variants. It should be noted that the original mechanisms were modified slightly to support HT stations, for example signaling A-MSDU support in the ADDBA Request and Response. However, these modifications were minor and the major changes were made in the HT variants.

9.3.1 Immediate and delayed block ack

While the normal acknowledgement mechanism is always in operation and in fact forms part of the underlying DCF, the block ack mechanism needs to be enabled by establishing

a block ack session through the exchange of an ADDBA Request and Response. The block ack session is established between two stations for a particular traffic identifier (TID) and for data transfer in one direction, originator to responder.

Following a successful ADDBA exchange the data transfer phase is entered. The originator will send a block of data followed by a BAR, to which the responder will respond with a BA. The BA acknowledges correctly received data frames from the previous block. The originator requeues data frames which were not correctly received and may send them in the subsequent block.

The originator or recipient may tear down the block ack session by sending a DELBA Request which, if correctly received, is acknowledged with an ACK.

Immediate block ack and delayed block ack differ in the handling of the BAR and BA frames during the data transfer phase. With immediate block ack the BAR solicits an immediate BA response, while with delayed block ack correct reception of the BAR frame itself is acknowledged with an ACK and the BA is returned in a separate channel access and acknowledged with another ACK. Immediate and delayed block ack sessions are illustrated in Figure 9.12 and described in more detail below.

9.3.2 Block ack session initiation

Stations indicate their ability to support block ack by setting the Immediate Block Ack and/or Delayed Block Ack capability bits in the Capability Information field in

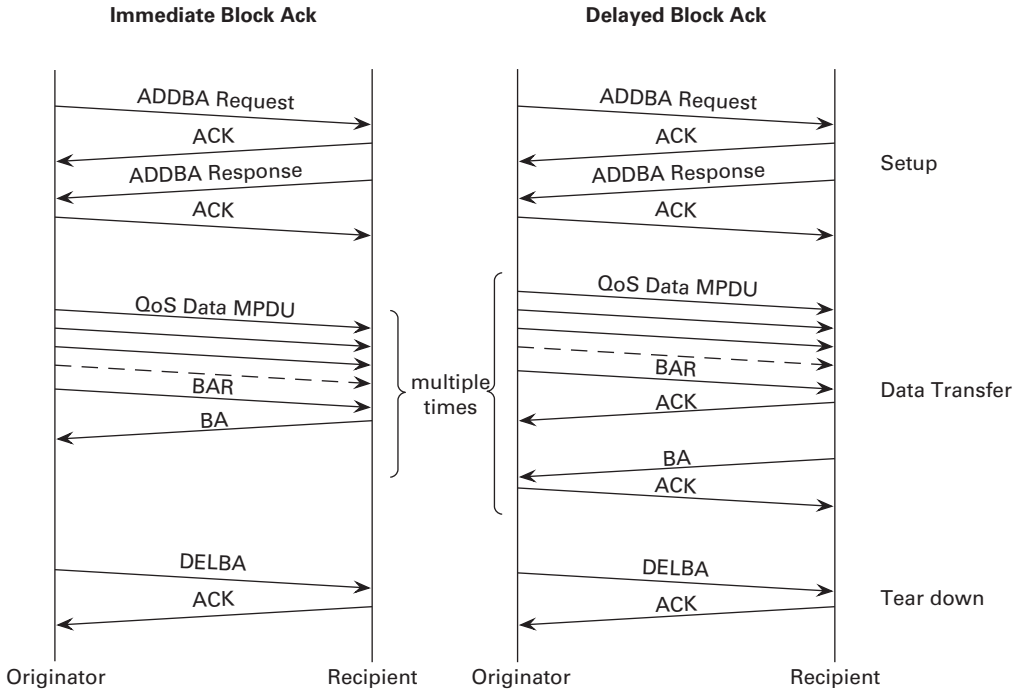

Figure 9.12 Immediate and delayed block ack sessions.

their Beacon, Association/Reassociation Request, and Response frames. If a station advertises that it supports one or both flavors of block ack then a peer station may establish a compatible block ack session for a particular traffic class with that station.

The block ack session is initiated by the originator sending an ADDBA Request frame. In response to a correctly received ADDBA Request frame, the responder sends an ACK. After further processing the responder sends an ADDBA Response frame to which the originator responds with an ACK if correctly received. The originator and responder will retransmit the ADDBA Request/Response if the expected ACK is not received. An inactivity timeout at the originator will detect a failed session setup.

The ADDBA Request and Response frames include the following fields:

- **Block Ack Policy**. This field indicates whether the session is an immediate block ack session or delayed block ack session. If one of the stations is a non-HT station, then the value set by the originator in the ADDBA Request is advisory and the value returned by the responder indicates the type of session should the originator continue with the session. If both stations are HT stations then the block ack policy established by the originator must either be accepted or rejected by the recipient.
- **TID**. This field gives the identifier for the traffic class or traffic stream for the session.
- **Buffer Size**. This field indicates the number of frame buffers the responder has available for reordering frames. The value, if set by the originator, is the desired value; the value set by the responder is binding. The originator must not have more than this number of MPDUs outstanding before soliciting a block ack.
- **A-MSDU Supported**. This field is set in the ADDBA Request to indicate that the originator may send A-MSDUs under this session and is set in the ADDBA Response if the recipient is capable of receiving A-MSDUs in this session. If the field is not set in the response then the originator will not send A-MSDUs.
- **Block Ack Timeout Value**. This field gives the duration after which the block ack session is terminated when there are no frame exchanges in this session.
- **Start Sequence Number (SSN)**. This is the sequence number of the first data frame from the originator.

The responder may reject a block ack session from an originator by sending a DELBA frame to the initiator after acknowledging receipt of the ADDBA Request.

9.3.3 Block ack session data transfer

During the data transfer phase, the originator may transmit a block of QoS Data frames, either as a burst, separated by SIFS or RIFS, or as part of an A-MPDU. Each QoS Data frame in the block has its Ack Policy field set to Block Acknowledgement. The recipient maintains a scoreboard to track which MPDUs have been received correctly. The data block may be wholly contained within a single TXOP or it may straddle multiple TXOPs. The data block and TXOP are not coupled in any way.

After transferring the data block, the originator sends a BAR frame. This frame includes a starting sequence number (SSN), which is the sequence number of the

oldest MSDU in the block for which an acknowledgement is needed. On receiving the BAR, the recipient performs two functions. First, it prepares a BA response using the scoreboard for that session. The scoreboard is converted into a bitmap where the first bit represents the MPDU with the same sequence number as the SSN from the BAR frame and subsequent bits indicate successive sequence numbers. The bitmap thus forms an array indexed by sequence number with the SSN as starting reference.

Second, it examines its reorder buffer for MPDUs with sequence numbers that precede the SSN value. These MPDUs are either reassembled into complete MSDUs and forwarded to the higher layers or discarded if complete MSDUs cannot be created.

The primary difference between immediate and delayed block ack is in the timing with which the recipient responds to the BAR. Under immediate block ack, the recipient responds to the BAR with a BA frame after SIFS. Under delayed block ack, the recipient responds to the BAR with an ACK. Later, in a separate channel access, the recipient generates a BA frame and sends it to the originator. The originator responds to the delayed BA with an ACK.

Immediate block ack provides better performance while delayed block ack was defined for ease of implementation. With delayed block ack, the recipient has more time to process the BAR and is suited to implementations where the bulk of the BA processing is performed in software on the host system.

On receiving the BA, the originator releases MPDUs that are acknowledged and requeues MPDUs that were not acknowledged for retransmission provided their time to live has not been exceeded.

9.3.4 Block ack session tear down

When the originator has no additional data to send and the final block ack exchange has completed, it may disable the block ack session by sending a DELBA frame to the recipient. The recipient sends an ACK in response and releases any resources allocated for the block ack session.

The block ack session may also be torn down by the originator or recipient if either does not receive a BA, BAR, or QoS Data frame belonging to that session within the duration of the block ack timeout value.

9.3.5 Normal ack policy in a non-aggregate

Small efficiency gains are possible using normal ack in a block ack session. Many traffic patterns are bursty and frequently have short periods where only a single frame needs to be sent. If the last block transfer has completed and all frames have been acknowledged to that point then it is more efficient to send the data frame using the normal ack procedure than to perform a BAR/BA exchange. In this case the QoS Data frame has its Ack Policy field set to Normal Ack and is sent in a non-aggregate PHY transmission. If correctly received the responder responds with an ACK. The frame is marked as correctly received in the block ack session scoreboard.

9.3.6 Reorder buffer operation

When the recipient receives a QoS Data frame for which a block ack session is in place, the recipient will buffer the MPDU. If the arriving MPDU completes the MSDU at the head of the reorder buffer, then the recipient forwards the complete MSDU and subsequent complete MSDUs in the reorder buffer in sequence to the higher layers until an incomplete MSDU forming a hole in the sequence space is encountered. If, when the MPDU arrives, there are incomplete preceding MSDUs in the reorder buffer then the MSDU is held until those preceding MSDUs are complete.

If an MPDU arrives and the reorder buffer is full then the first MSDU in the reorder buffer is discarded (since it is incomplete) to make room. This may also result in the release of complete subsequent MSDUs to the higher layers.

If a BAR frame is received, all complete MSDUs with a lower sequence number than the starting sequence number of the BAR are forwarded to the higher layers and all incomplete MSDUs with a lower sequence number are discarded. The BAR frame thus has a dual role. In addition to soliciting a block ack response, it provides the originator with a mechanism for flushing the recipient's reorder buffer of incomplete MSDUs or holes representing MSDUs whose retransmit lifetime has expired. If the originator discards one or more MPDUs due to lifetime expiry it must send a BAR to flush the recipient reorder buffer so that subsequent MSDUs are not needlessly held up waiting for the sequence to be completed.

Figure 9.13 gives an example of reorder buffer behavior. A block of QoS Data frames composed of fragmented MSDUs is sent. In the diagram, the QoS Data MPDUs are numbered with the number before the decimal point being the MSDU sequence number and the number after the decimal point being the MSDU fragment number. MSDU 1 is received completely, reassembled and forwarded to the higher layers. The second fragment of MSDU 2 is lost and thus the received fragment is stored until MSDU 2 can be completed. This holds up subsequent MSDUs even if they are complete.

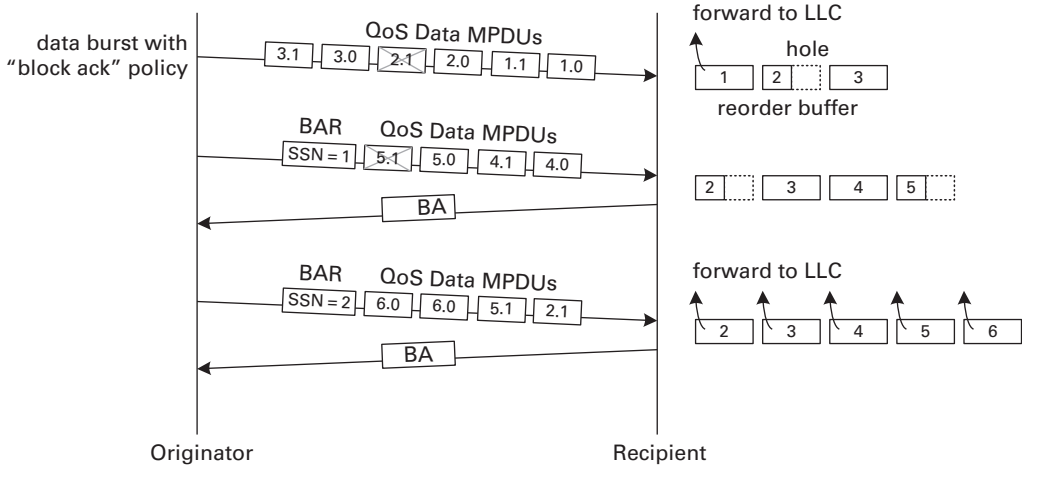

Figure 9.13 Reorder buffer behavior with fragmented MSDUs.

After a BAR/BA exchange, the originator learns of the lost fragments and retransmits them together with additional MSDUs that are available and that will fit in the reorder buffer. All MSDUs are now complete and forwarded in sequence to the higher layers. The originator learns that all MSDUs have been successfully transferred with another BAR/BA exchange.

9.4 HT-immediate block ack

HT-immediate block ack is a significant modification to the immediate block ack protocol and stands as a separate protocol for the purposes of backward compatibility with legacy devices. An HT station that wishes to establish a block ack session with a non-HT station must use the original immediate or delayed block ack protocol. An HT station that establishes a block ack session with another HT station uses the HT-immediate or HT-delayed block ack protocol. There is a lot of commonality between the HT variants and the original protocol, which eases implementation.

All HT stations (and VHT stations, since VHT stations are also HT stations) are required to support HT-immediate block ack as a recipient.

9.4.1 Normal Ack policy in an aggregate

The block ack protocol was introduced in the 802.11e amendment prior to the introduction of aggregation in the 802.11n amendment. Since an aggregate is a single PHY transmission containing multiple MPDUs it was thought that a mechanism analogous to the original Data/ACK mechanism might be possible. The mechanism introduced in the 802.11n amendment to support this changes the meaning of the Normal Ack policy when present in QoS Data frames in an aggregate transmission. If one or more of the QoS Data MPDUs in an aggregate have their Ack Policy field set to Normal Ack then the responder will return a BA in response to the aggregate. The two response mechanisms to Normal Ack policy are illustrated in Figure 9.14.

The use of Normal Ack policy to solicit a BA does not eliminate the need for the BAR. Recall that the BAR frame performs two functions: it solicits a BA response and it flushes the MSDUs in the reorder buffer that are held up due to an earlier incomplete MSDU. If the originator does not receive an acknowledgement for an MSDU whose lifetime has

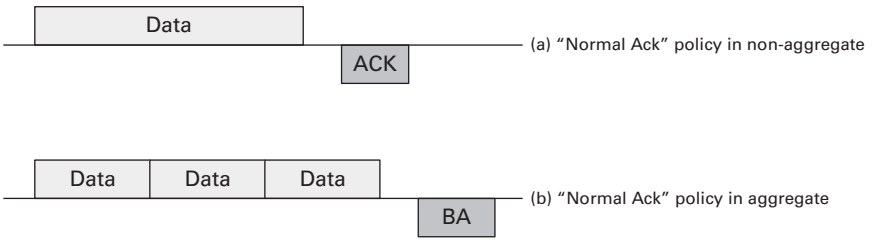

Figure 9.14 Normal Ack policy in (a) non-aggregate and (b) aggregate.

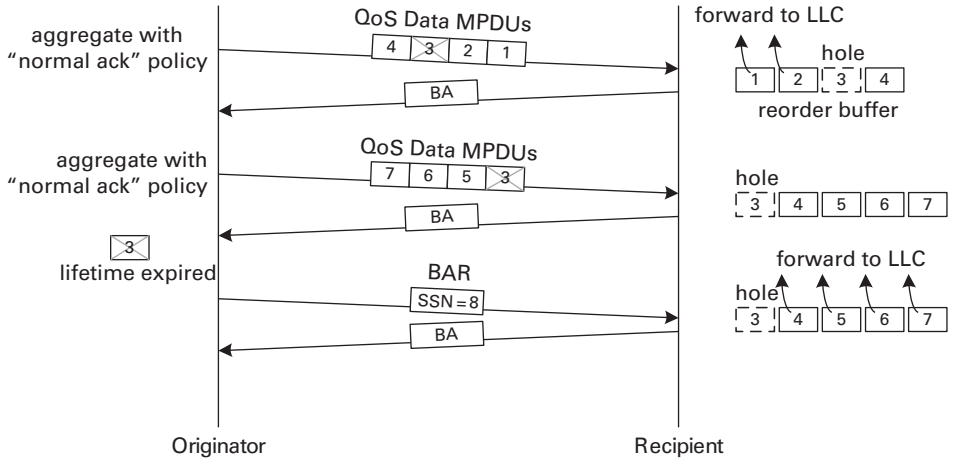

Figure 9.15 Using BAR to flush the reorder buffer.

expired then the originator must send a BAR to flush the recipient's reorder buffer of complete MSDUs subsequent to the MSDU that will never make it across. This is illustrated in the example given in Figure 9.15.

In this example, MSDU 3 is not successfully received by the recipient after a number of retries. On lifetime expiry the originator discards MSDU 3. To eliminate the hole in the recipient's reorder buffer, the originator must send a BAR with a SSN that is greater than the sequence number of the discarded MSDU. In practice, the SSN is set to the sequence number of the next MSDU to be transmitted (although that MSDU may not be available for transmission yet) which in this case is 8 since all MSDUs with lower sequence numbers have been acknowledged and discarded by the originator.

9.4.2 Compressed block ack

At the higher data rates, fragmentation does not provide much benefit. The original BA frame was defined with a 1024-bit scoreboard (128 octets) to support 64 MSDUs, each of which can be fragmented with up to 16 fragments. 802.11n introduces a compressed BA variant (see Section 12.2.1.6) that does away with the 16 bits per MSDU for fragmentation, resulting in a 64-bit scoreboard (8 octets). This reduces both the on-air overhead and the memory requirements in the recipient.

9.4.3 Full state and partial state block ack

The block ack mechanism defined in the 802.11e amendment is referred to as full state block ack to distinguish it from partial state block ack, introduced in the 802.11n amendment. Partial state block ack is backward compatible with full state block ack in the sense that an originator using partial state rules will operate correctly with a recipient implementing full state operation. The distinction between full state and partial state operation is discussed in more detail in the next section.

9.4.3.1 Full state block ack operation

Under full state block ack, the recipient maintains an ack state scoreboard for each block ack session. The scoreboard records the ack state of up to 64 MSDUs. When fragmentation is used each MSDU may be fragmented in up to 16 fragments, thus the scoreboard is an up to 64 entry by 16-bit array. A recipient implementation with limited memory may constrain the extent of the array by setting the BufferSize parameter in the ADDBA Response.

The MSDU sequence number is a 12-bit value, thus the scoreboard represents a window in the sequence number space of 4096 values. The scoreboard window is defined by a beginning sequence number WinStart, an ending sequence number WinEnd, and an extent WinSize. With the establishment of the block ack session the scoreboard is initialized with WinStart set to the starting sequence number provided in the ADDBA request.

When a QoS Data frame arrives, if the sequence number falls within the space represented by the scoreboard then the recipient will index the scoreboard using the data frame's sequence number (SN) and record its correct receipt. If the SN is outside the space represented by the scoreboard but within the range WinEnd to WinStart + 2^{11} (half the sequence number space) then the recipient will shift the scoreboard to the right until it includes the new sequence number on the rightmost edge of its window.

When a BAR arrives, the scoreboard window is shifted to the right so that WinStart is equal to the SSN provided in the BAR frame and a BA response is returned with the contents of the scoreboard.

9.4.3.2 Motivation for partial state block ack

With the original block ack mechanisms, it is required that the scoreboard state persist for the duration of the block ack session. This burdens the recipient implementation with the need to maintain state for all active block ack sessions and, in practice, with the low latency required to produce a BA in response to a BAR, this means using expensive on-chip memory. With the 802.11n amendment the rules are relaxed so that on-chip memory reserved for block ack state may be reused by different block ack sessions. The state memory effectively serves as a cache, storing the state of the most recently active block ack session. The newer rules are referred to as partial state block ack and are fully backward compatible with the original full state block ack rules.

To understand the motivation for the change consider a typical implementation illustrated in Figure 9.16. Under immediate BA, when the recipient receives a BAR or an aggregate frame containing QoS Data frames with Normal Ack policy, the recipient must transmit a BA Response frame SIFS after receiving the BAR or aggregate QoS Data frame. With decode latencies on the receive path and encode latencies on the transmit path, there is little time available to locate the appropriate state information and form the BA Response. This largely necessitates on-chip storage of the block ack scoreboard that will be returned in the BA Response.

The other major function in the block ack mechanism is the reassembly and reorder function, which reassembles complete MSDUs and forwards them in order to the higher

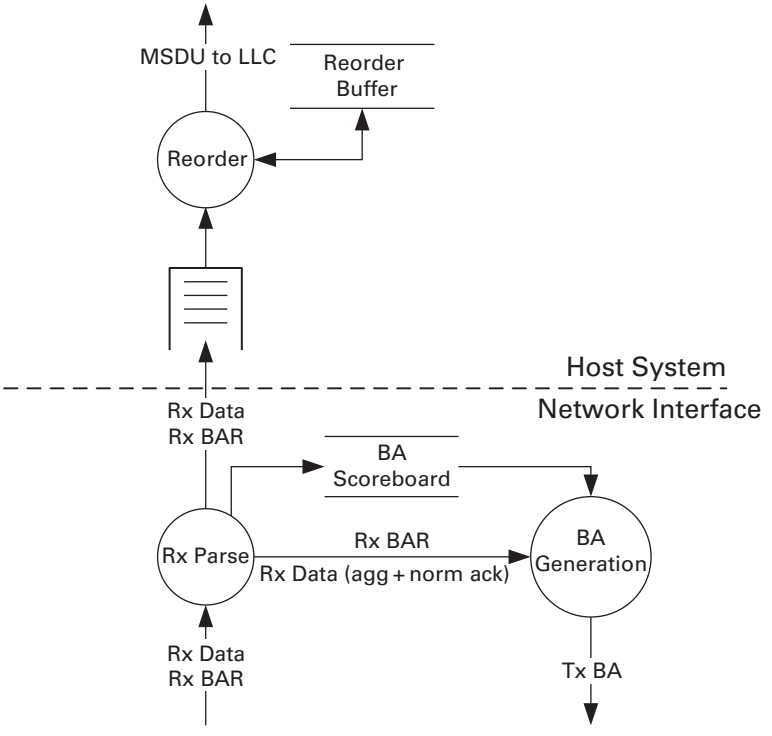

Figure 9.16 Typical functional split for immediate block ack implementation in recipient station.

layers. This function is not time critical and requires a large buffer for storing packets during the reorder and reassembly process and as such is typically implemented in the system hosting the network interface.

The new partial state rules do not affect the reorder and reassembly process, but reduce the resource requirements in the network interface for storing the block ack scoreboard. A reorder buffer is still required for each block ack session, but since this is typically stored in the host memory it is relatively cheap. Much less memory is required on-chip to store the block ack scoreboard since the same memory can be reused for multiple block ack sessions.

9.4.3.3 Partial state block ack operation

On receiving a QoS Data frame with sequence number SN, the recipient checks to see if it has a record of the block ack scoreboard for that block ack session, where the session is identified by the transmit address (TA) and TID. If not, then it creates a scoreboard for that session with WinEnd = SN and WinStart = WinEnd − WinStart + 1, perhaps reusing memory from another session. The correct reception of the data frame is recorded by setting a 1 in the position representing SN, i.e. WinEnd.

For each data frame thereafter:

- If SN is within the current scoreboard window, i.e. WinStart ≤ SN ≤ WinEnd, then the scoreboard records receipt at the offset represented by SN.

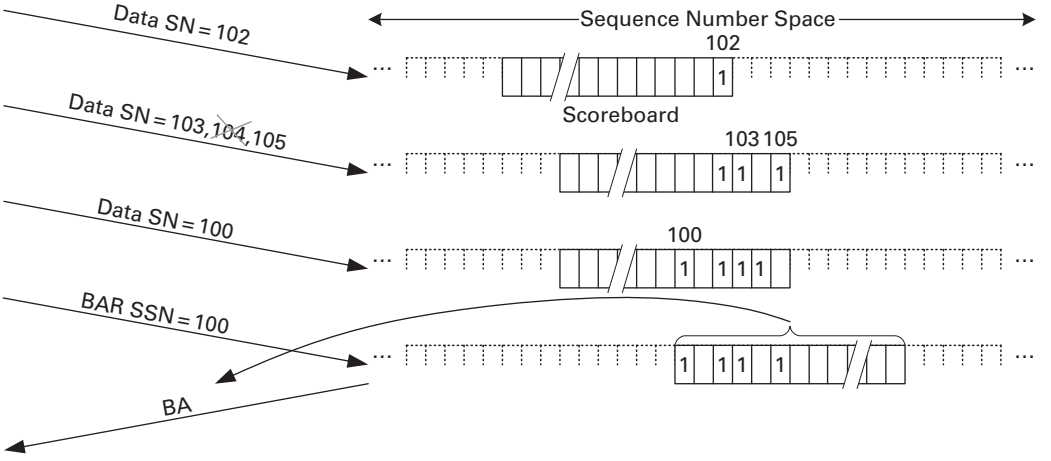

Figure 9.17 Scoreboard operation.

- If SN is outside the current scoreboard window, but within half sequence space range, i.e. WinEnd < SN < WinStart + 2^{11}, then the scoreboard is shifted right to accommodate SN.
- If SN is more than half the sequence space beyond the window, i.e. WinStart + 2^{11} < SN < WinStart, then no change is made.

The scoreboard operation is illustrated in Figure 9.17. Here a QoS Data frame with sequence number 102 is received and the recipient creates a new scoreboard with a mark for 102 at its rightmost edge. With the next aggregate, QoS Data frames 103 and 105 are received correctly and the scoreboard is shifted right to accommodate the new entries. QoS Data frame 100 falls within the scoreboard sequence number range and is simply marked up. Note that in this sequence the QoS Data frames have their Ack Policy field set to Block Ack.

When the BAR is received, the recipient shifts the scoreboard to the right so that WinStart = *SSN* from the BAR (100 in this case) and returns a BA frame with the contents of the scoreboard.

Note that the sequence numbers shown are for illustration only. In practice, the sequence number is a 16-bit value composed of a 12-bit MSDU number and a 4-bit fragment number.

The major difference between partial state and full state block ack operation is the transient nature of the scoreboard retained by the recipient. Under partial state block ack the originator should ensure that it retrieves the ack state with high probability before another station has a chance to send data to the recipient and potentially erase the session's block ack scoreboard. In practice this means that the originator should attempt to retrieve the block ack scoreboard before the end of each TXOP. If occasionally the block ack scoreboard is not retrieved during the TXOP (perhaps the BA frame is received in error) then there is still a good chance that a subsequent immediate channel access by the same station will occur before data belonging to another block ack session is received

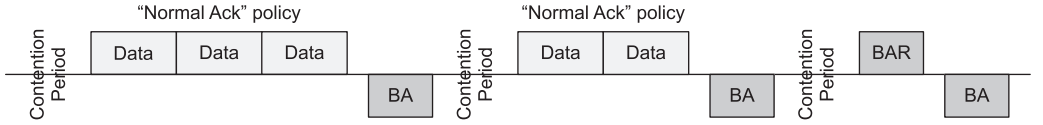

Figure 9.18 Typical HT-immediate TXOP sequences.

by the recipient, erasing the session's block ack scoreboard. The originator can thus simply use a BAR frame in a subsequent channel access to retrieve the ack state. In the unlikely event that the block ack scoreboard is no longer available, the BA will show all zeros and the originator will need to retransmit the MSDUs. Alternatively, if the originator sent a single aggregate consisting of QoS Data frame with Normal Ack policy and no BA was received, then the originator may assume that none of the QoS Data frames made it through and simply retransmit the QoS Data frames.

While more complex behavior is permitted, most implementations will conform to partial state rules by sending a single aggregate in each TXOP, soliciting a BA by setting the ack policy of the QoS Data frames making up the aggregate to Normal Ack. On the rare occasion when an MSDU lifetime expires before it is acknowledged, the originator will send a BAR frame to flush the holes in the reorder buffer. An originator implementing this behavior would work with both a full state and partial state implementation in the recipient.

9.4.4 HT-immediate block ack TXOP sequences

TXOP sequences under HT-immediate block ack are shown in Figure 9.18. The TXOP typically contains an A-MPDU carrying multiple QoS Data MPDUs each with "Normal Ack" policy followed by an immediate BA response. If a lot of frame errors are encountered and an MPDU is discarded after numerous retransmissions, a TXOP with a BAR/BA exchange may occur.

Also, in some environments, it may be beneficial to prefix each aggregate-BA exchange with an RTS/CTS exchange. This short frame exchange acts as a low overhead collision detect mechanism when a lot of stations are competing for channel access and as a protection mechanism against hidden nodes.

9.5 HT-delayed block ack

HT-delayed block ack is an extension to the delayed block ack protocol and differs from it in the manner in which BAR and BA frames are acknowledged. Support for HT-delayed block ack is optional and a station advertises support by setting the HT-delayed Block Ack capability bit in the HT Capabilities element (see Section 12.3.2.2). A peer station may establish a HT-delayed block ack session with a station that advertises itself as HT-delayed block ack capable.

Under HT-delayed block ack the BAR and BA frames carry a BAR Ack Policy and BA Ack Policy field, respectively. If set to 1 this indicates that the receiver of the frame should not return an ACK response.

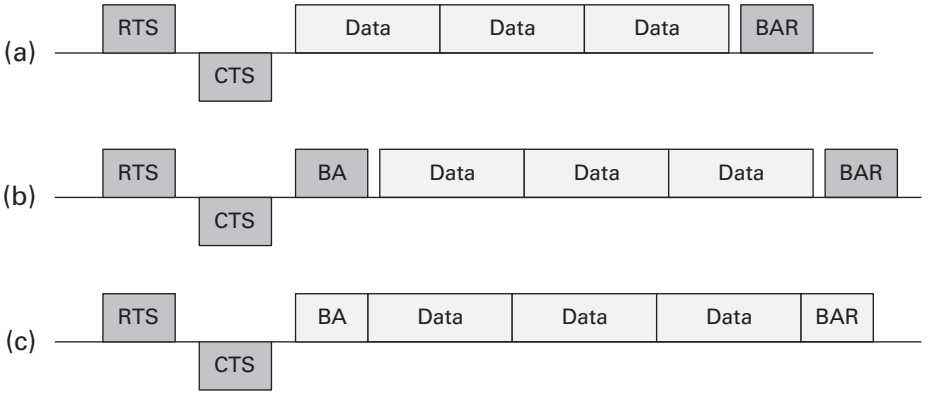

Figure 9.19 Typical HT-delayed TXOP sequences.

9.5.1 HT-delayed block ack TXOP sequences

Typical TXOP sequences under HT-delayed block ack are shown in Figure 9.19. The TXOP begins with a short frame exchange which could be RTS/CTS (as shown) or Data/ACK. HT-delayed block allows the TXOP to be fully utilized by the originator since no immediate response is required from the responder. Sequence (a) shows the originator sending an aggregate with ack policy set to Block Ack followed by a BAR. When data flows both ways, the BA may be combined with data in the reverse direction. This is shown in sequence (b).

It is possible, with reduced robustness, to aggregate the BA and BAR frames together with the data. This is shown in sequence (c).

References

Hansen, C. and Edwards, B. (2004). *WWiSE Proposal: High Throughput Extension to the 802.11 Standard*, IEEE 802.11-04/0886r6.

IEEE (2006). IEEE *802.11n Project authorization request, 26 May 2006*, available at: http://standards.ieee.org/board/nes/projects/802-11n.pdf.

Jain, R. (1990). Error characteristics of fiber distributed data interface (FDDI), *IEEE Transactions on Communications*, **38**(8), 1244–52.