Timeout-aware Out-of-order Delivery Queues for Establishing Delay-Reliability Tradeoffs

Date: March 4, 2025

Authors:

Name	Affiliations	Address	Phone	email
Behnam Dezfouli	Nokia	520 Almanor, Sunnyvale, California		behnam.dezfouli@nokia.com
Alvin Lee				
Klaus Doppler				
Salvatore Talarico				
Kerstin Johnsson				

Introduction

- In 802.11 networks, **frame loss** is a common phenomenon caused by factors such as interference, rate adaptation algorithm, mobility, etc.

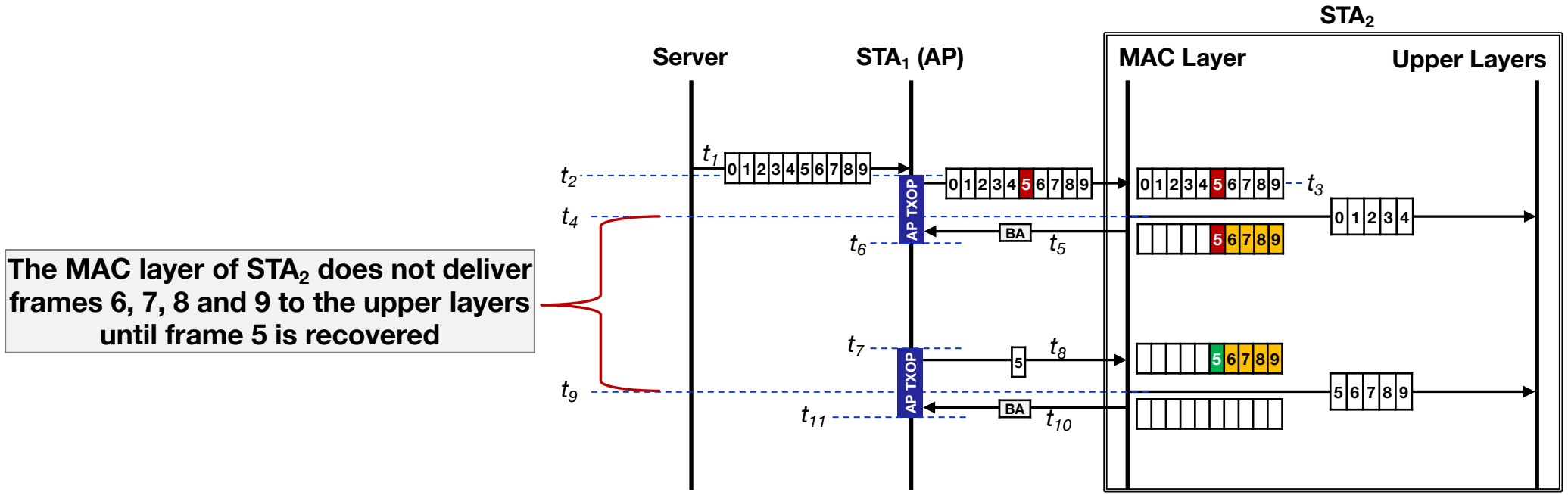
- **When a frame loss occurs, its subsequent frames are received out-of-order (OOO)**
- **The MAC layer of the receiver does not deliver the OOO-received frames until the missing frame is recovered**
- **This operation delays the delivery of frames to the upper layers, thereby causing a drop in application performance [11-23/0069] [11-24/463r2] [11-24/2123r1]**
- A proposed solution to this problem is dedicating a TID to OOO frame delivery [11-24/463] [11-24/2123r1]
- In this contribution, we show that this approach causes two main problems: **(1)** applications cannot establish delay-reliability tradeoffs for their traffic streams, and **(2)** cause unnecessary frame retransmissions

- We propose **timeout-aware OOO queues, which enable applications to map their traffic streams to various queues, based on their delay and reliability requirements**

The Head of Line (HoL) Blocking Problem

This example shows the delay caused by the MAC layer before delivering frames to upper layers

- t_1: AP receives frames 0 through 9 from the Server
- t_2: AP acquires a TXOP
- t_3: AP sends frames 0 through 9 to STA_2; **frame 5 is lost**
- t_4: **MAC of STA_2 delivers in-order (IO) received frames to upper layer**
- t_8: AP retransmits frame 5 to STA_2
- t_9: MAC of STA_2 delivers frames 5, 6, 7, 8 and 9 to upper layers



Out-of-order (OOO) Queues

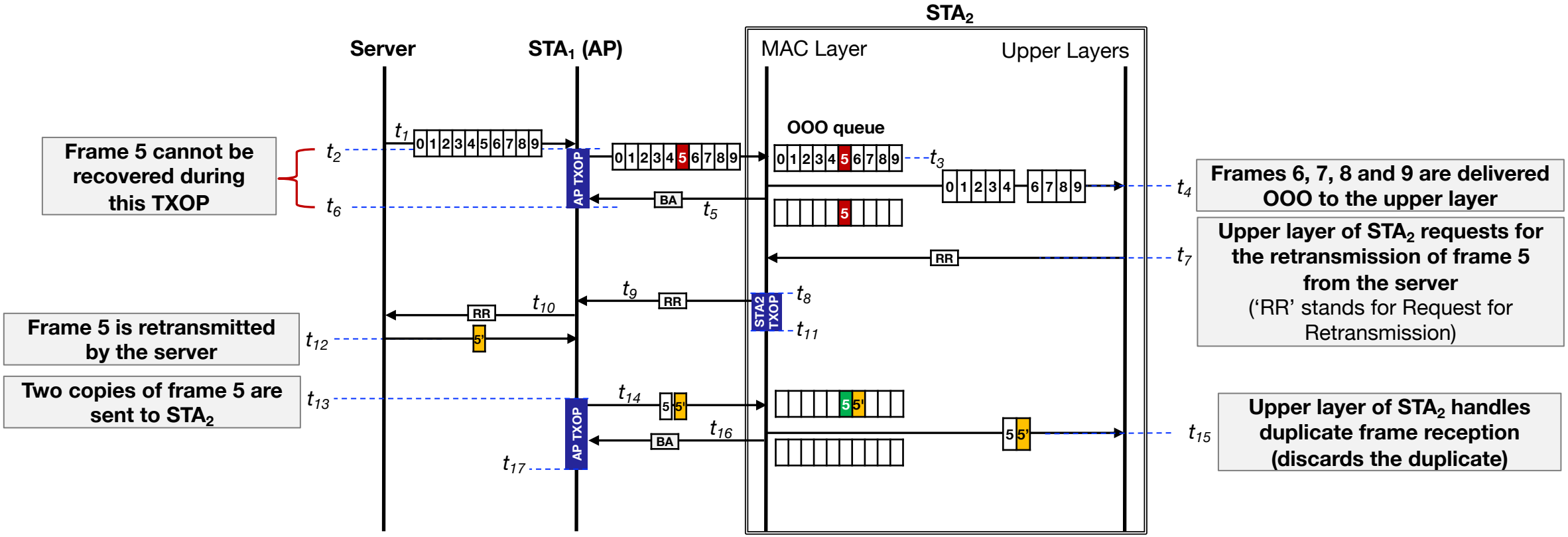
- **A proposed solution for mitigating the impact of the blocking time is to dedicate a TID to OOO frame delivery [11-24/463]**
- **Using this OOO queue, OOO received frames are delivered to the upper layers immediately**

- **Shortcomings of this approach:**
  - OOO frame delivery to the upper layers, such as TCP, may results in unnecessary upper layer (TCP, QUIC) retransmissions, higher delay, and overhead of recovery
  - Reduces the efficiency of layer-2 frame recovery
    - Does not provide the MAC layer of the receiver enough time to wait before deciding about OOO delivery to upper layers
  - Does not allow applications to establish delay-reliability tradeoffs for their traffic streams

❑ Out-of-order (OOO) Queues

The following example shows the unnecessary end-to-end retransmission when using OOO queues

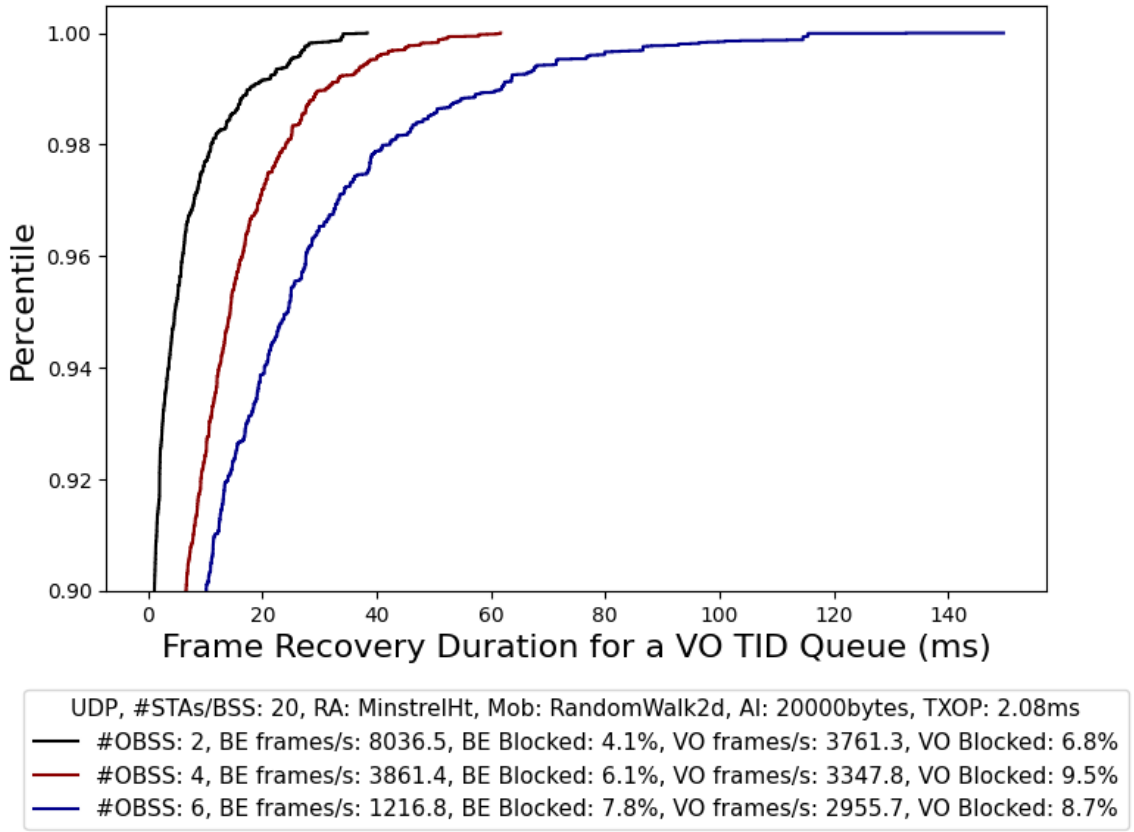
- Now we consider the case where the MAC layer provides an OOO queue
- The upper layer of STA_2 needs frame 5; however, it is unaware that this frame is buffered at the AP and will be retransmitted again



Timeout-aware Out-of-order (OOO) Queues

- **We propose timeout-aware OOO queues: a missing frame is delivered to the upper layer if it is not recovered by the expiry of the timeout**
- **Depending on the recovery duration of OOO frames, various timeout values are assigned to the OOO queues**
- Applications can choose from these OOO queues when receiving or sending frames to establish their delay and reliability tradeoffs

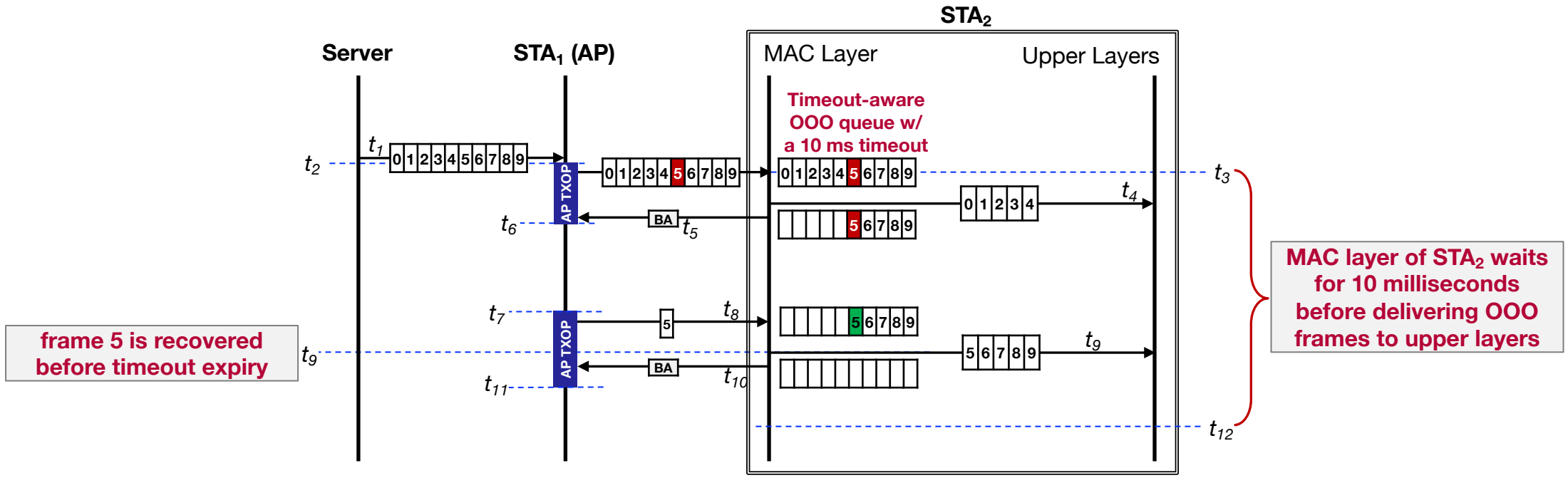
Recovery time of AC_VO frames in presence of 2 or more OBSSs



Please see the appendix for more results related to frame recovery delay

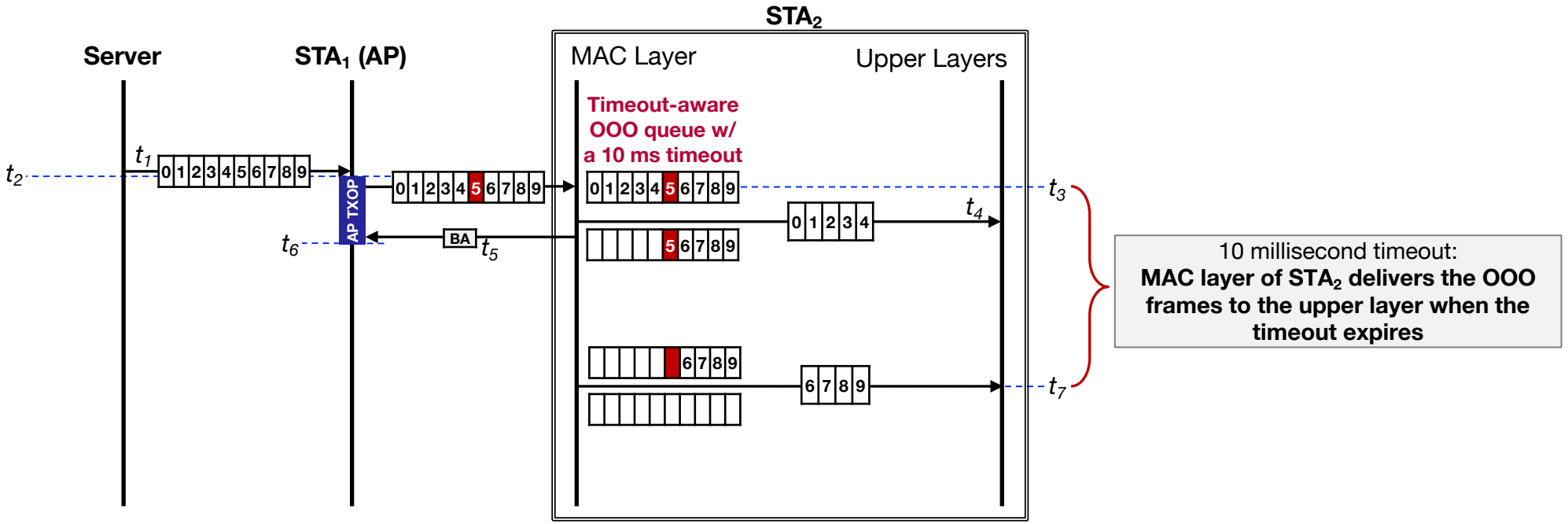
❑ Timeout-aware Out-of-order (OOO) Queues

- In the following example, since frame 5 is **recovered before the expiry of the timeout of the OOO queue**, the MAC layer of STA_2 delivers frames 5, 6, 7, 8, and 9 **in-order** to the upper layer



❑ Timeout-aware Out-of-order (OOO) Queues

- In the following example, since frame 5 **is not recovered before the expiry of the timeout** of the OOO queue, the MAC layer of STA_2 delivers frames 6, 7, 8, and 9 to the upper layer in an **out-of-order** fashion

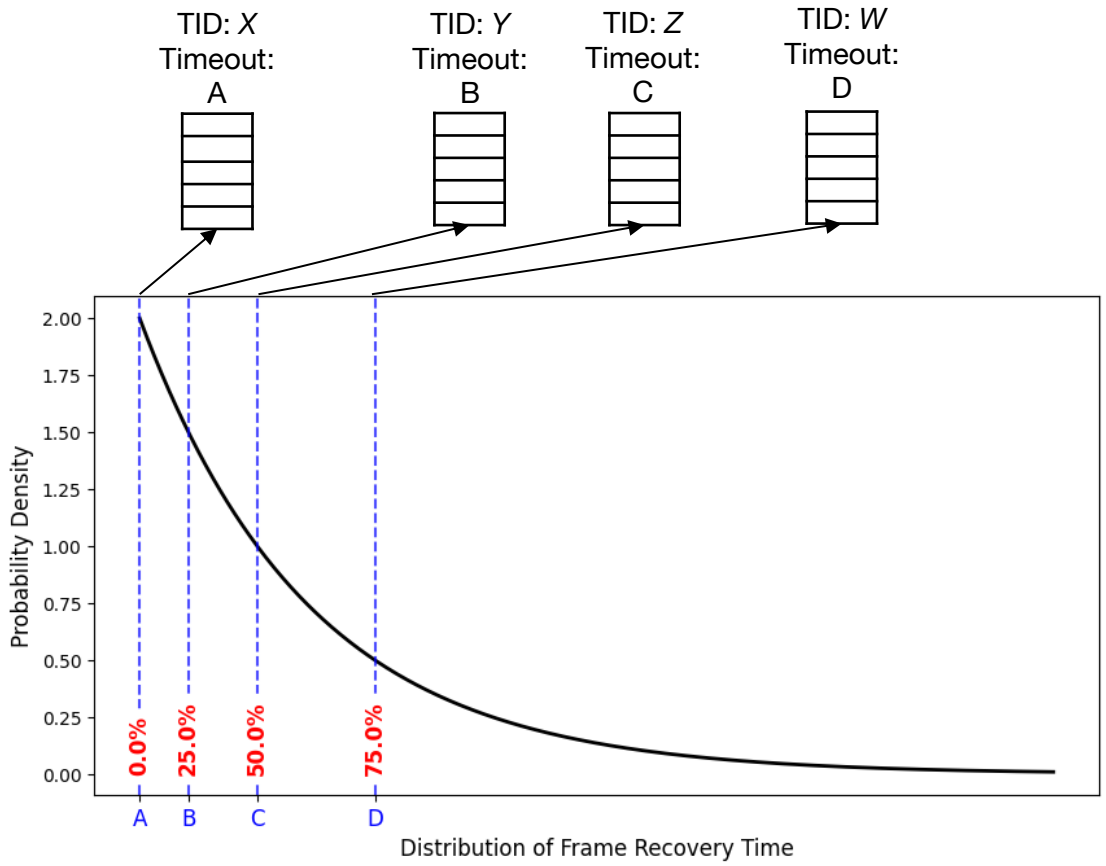


Determining Timeout Values for OOO Queues

- One approach to determine the mapping of timeout values to OOO queues is via measuring frame recovery times and creating the distribution of recovery times

- In this example, we assume there are **four OOO queues**
- The distribution of frame recovery duration is split into four equal-density intervals
- The line separating these four intervals are used to determine the timeout value of the four OOO TID queues

- **An OOO queue is assigned to a stream based on its timeliness and/or reliability requirements**



Conclusion

- Frame loss is common in 802.11 networks and negatively affects application performance
- The recovery time of frames depends on various factors such as the number of BSSs, TXOP duration, mobility, rate adaption, etc.
- **While some applications can tolerate a certain extent of frame loss, layer-2 frame recovery is preferred to enhance user experience**

- **By using timeout-aware OOO queues, the MAC layer waits for certain duration before delivery of OOO frames to upper layers**
  - Allows layer-2 frame recovery before sending OOO frames to the upper layers

Mapping a Stream to a OOO Queue

- When a STA assigns timeout values to OOO queues, **the STA establishes a SCS agreement with the sender to assign the TID of the OOO queue to the stream.**
  - This enforces future frames sent by the sender to have OOO TID field value that maps the received frames to its corresponding OOO queue

- **Other considerations:**
  - **Enhanced relaying operation:** To enhance end-to-end QoS when two non-AP STAs communicate through the AP, a method similar to [11-23/1885] can be used by a non-AP STA to inform the AP and the other non-AP STA about the TID to queue mapping
  - **Dynamic SCS profiles:** Similar to [11-24/660], a STA may dynamically switch between various TIDs to establish its desired delay-reliability tradeoff over time

Straw Poll

- Do you agree that introducing two or more out-of-order (OOO) mapped TID queues in 11bn allows applications to establish delay-reliability tradeoffs more efficiently?

YES/NO/ABSTAIN

Appendix

Evaluation of Frame Recovery Delays (ns3 simulation)

❑ Evaluation of Frame Recovery Delays: Simulation Parameters

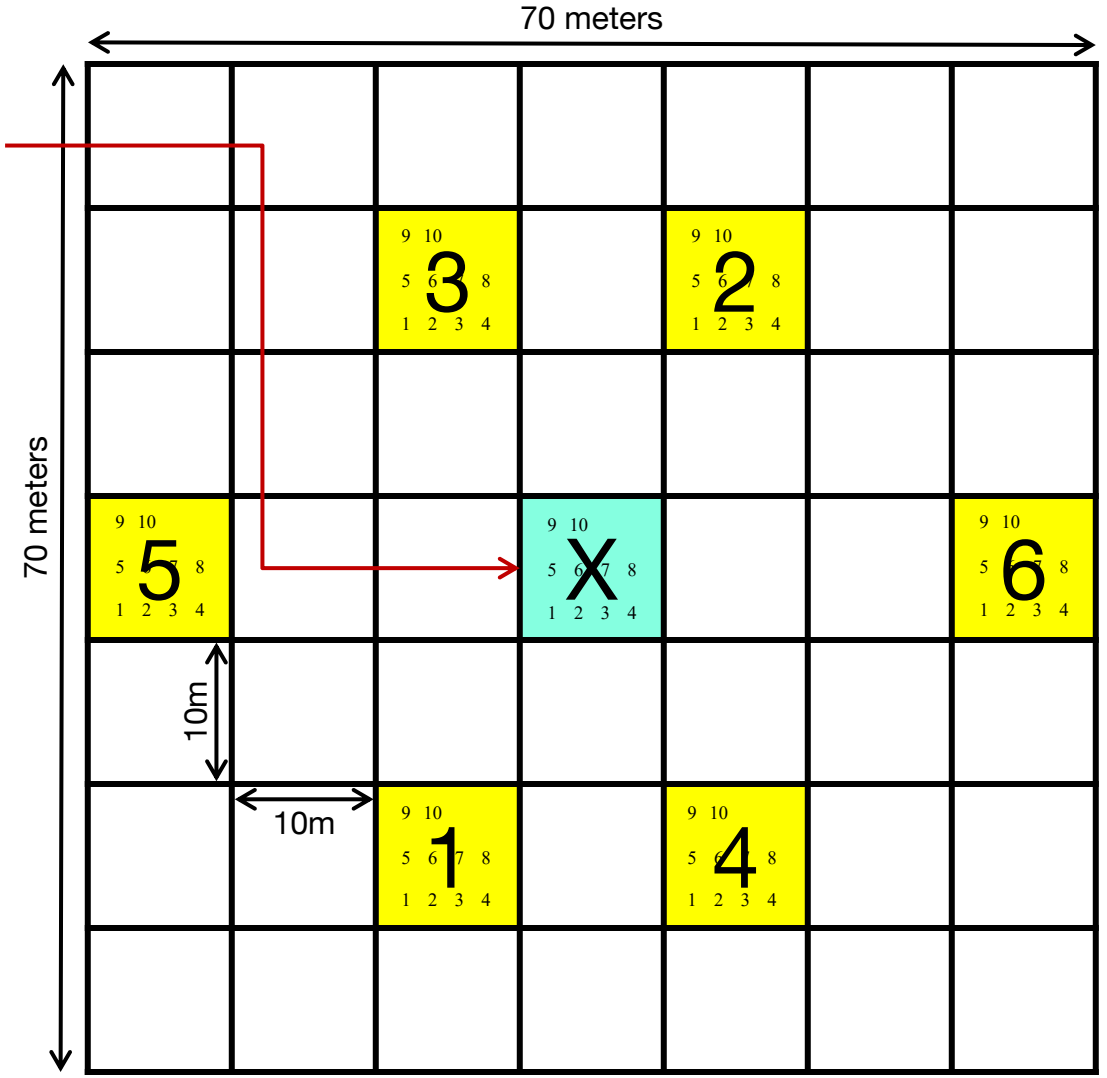
- **Network**
  - Each BSS includes an AP and 20 non-AP STAs
  - UL UDP flow from each non-AP STA to the AP
  - Default parameters of 40MHz, MCS-6, NSS-2
  - Rate Adaptation Algorithm = {None, Minstrel-Ht}
  - TXOP Duration for BE = 2.528ms, VO = {0.520, 2.080}ms [802.11-2020 Table 9-155]
  - Aggregation Intensity (AI) = {3500, 20000} bytes
  - Non-AP STA Mobility = {Stationary, RandomWalk2d}
  - #OBSS = {2, 4, 6}, each with 20 non-AP STAs
  - AP TX Power = 23 dBm, non-AP STAs TX Power = 17 dBm
- **Traffic**
  - Constant UDP data-rate; MSS Size = 1460
  - BE traffic rate = full buffer
  - Awake = 50 ms, Dormant = 50 ms, VO traffic rate when awake = 7.7 Mbps/STA
- **Scenarios**
  - **Scenario 1: Each BSS includes 80% BE and 20% VO STAs**
  - **Scenario 2: Each BSS includes all BE STAs**

❑ Evaluation of Frame Recovery Delays: Simulation Parameters

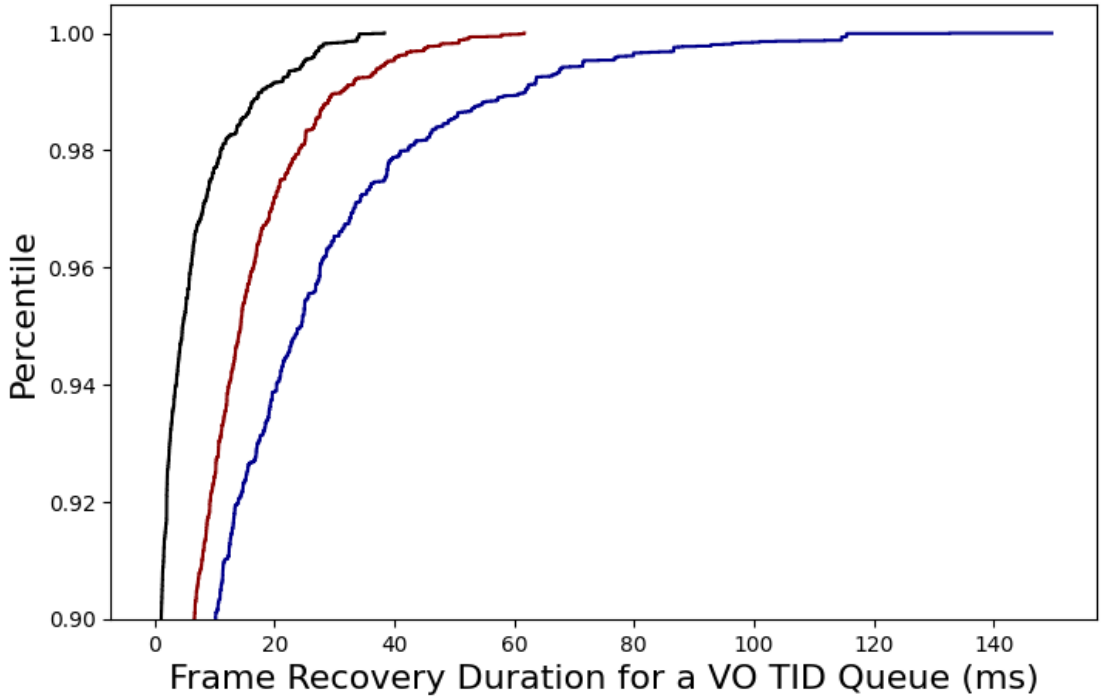
- OBSSs are placed in rooms around the BSS under test

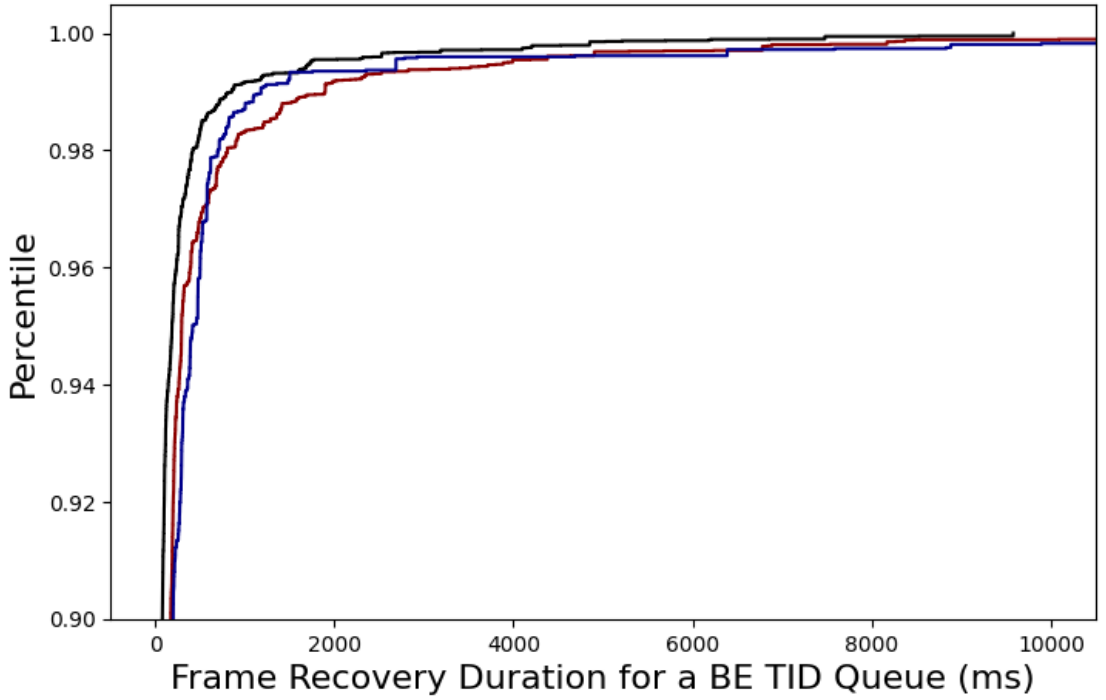
- Placed in order of increasing label number
- Example: with #OBSS=2, rooms 1 and 2 have BSSs in them

- AP is placed in the middle of the room assigned to the BSS
- STAs are placed in a grid formation within the room
- STAs never walk out of their room



❑ Scenario 1: 80% BE and 20% VO STAs: **Impact of the Number of OBSSs**



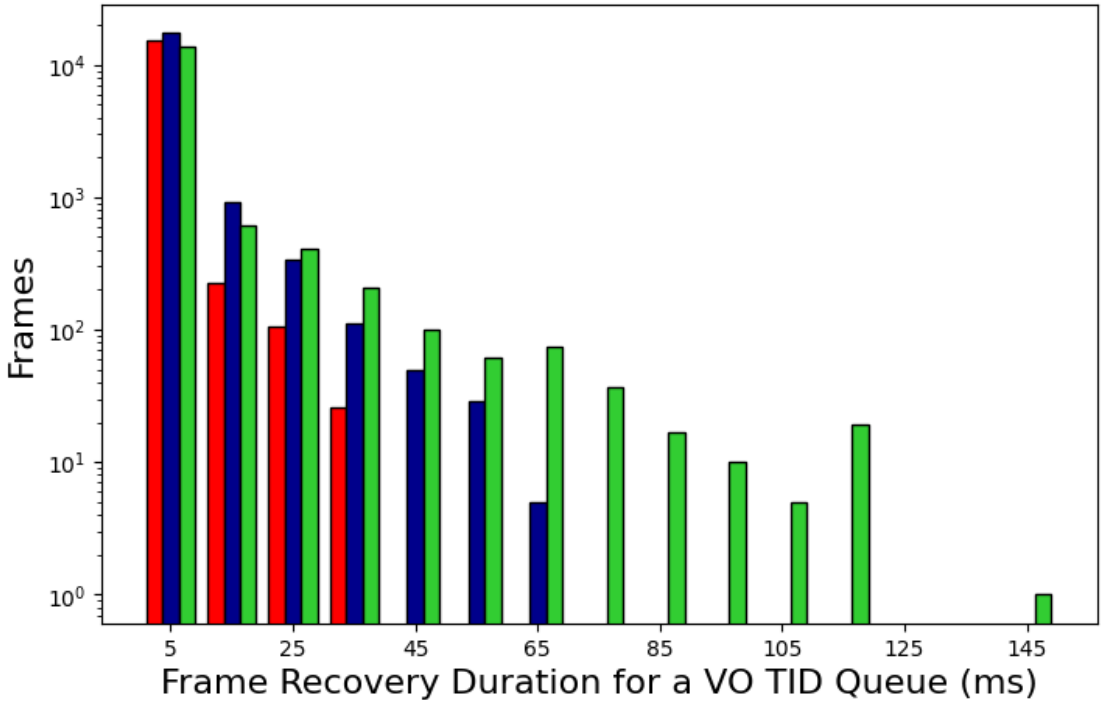


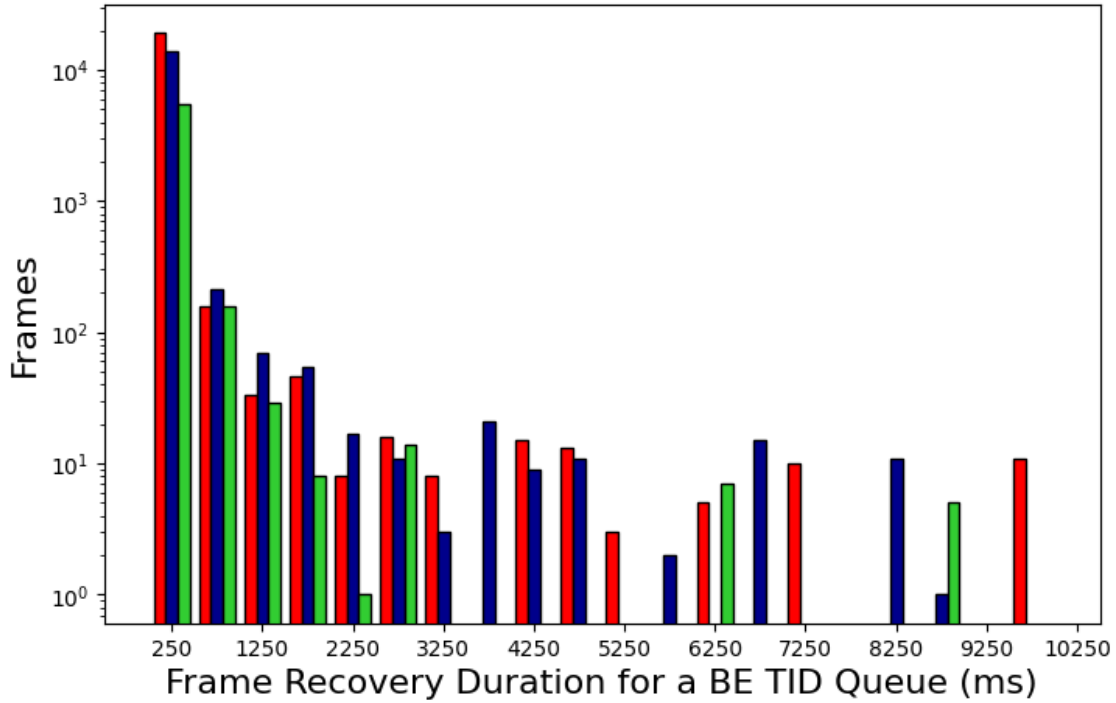
UDP, #STAs/BSS: 20, RA: MinstrelHt, Mob: RandomWalk2d, AI: 20000bytes, TXOP: 2.08ms
#OBSS: 2, BE frames/s: 8036.5, BE Blocked: 4.1%, VO frames/s: 3761.3, VO Blocked: 6.8%
#OBSS: 4, BE frames/s: 3861.4, BE Blocked: 6.1%, VO frames/s: 3347.8, VO Blocked: 9.5%
#OBSS: 6, BE frames/s: 1216.8, BE Blocked: 7.8%, VO frames/s: 2955.7, VO Blocked: 8.7%

UDP, #STAs/BSS: 20, RA: MinstrelHt, Mob: RandomWalk2d, AI: 20000bytes, TXOP: 2.08ms
#OBSS: 2, BE frames/s: 8036.5, BE Blocked: 4.1%, VO frames/s: 3761.3, VO Blocked: 6.8%
#OBSS: 4, BE frames/s: 3861.4, BE Blocked: 6.1%, VO frames/s: 3347.8, VO Blocked: 9.5%
#OBSS: 6, BE frames/s: 1216.8, BE Blocked: 7.8%, VO frames/s: 2955.7, VO Blocked: 8.7%

- As the number of OBSSs increase, the blocking frequency and blocking duration increases for VO STAs
- Caused by the increase in transmission error rate with the number of OBSS
- BE stations experience significantly longer tail times compared to VO stations

❑ Scenario 1: 80% BE and 20% VO STAs: **Impact of the Number of OBSSs**

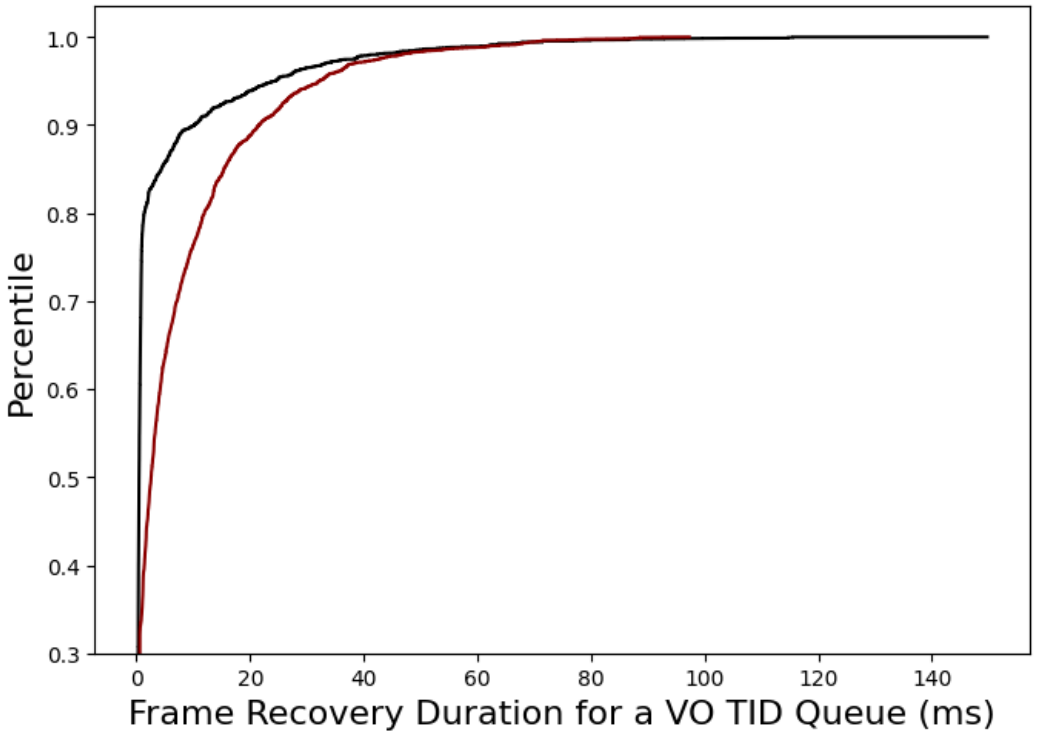


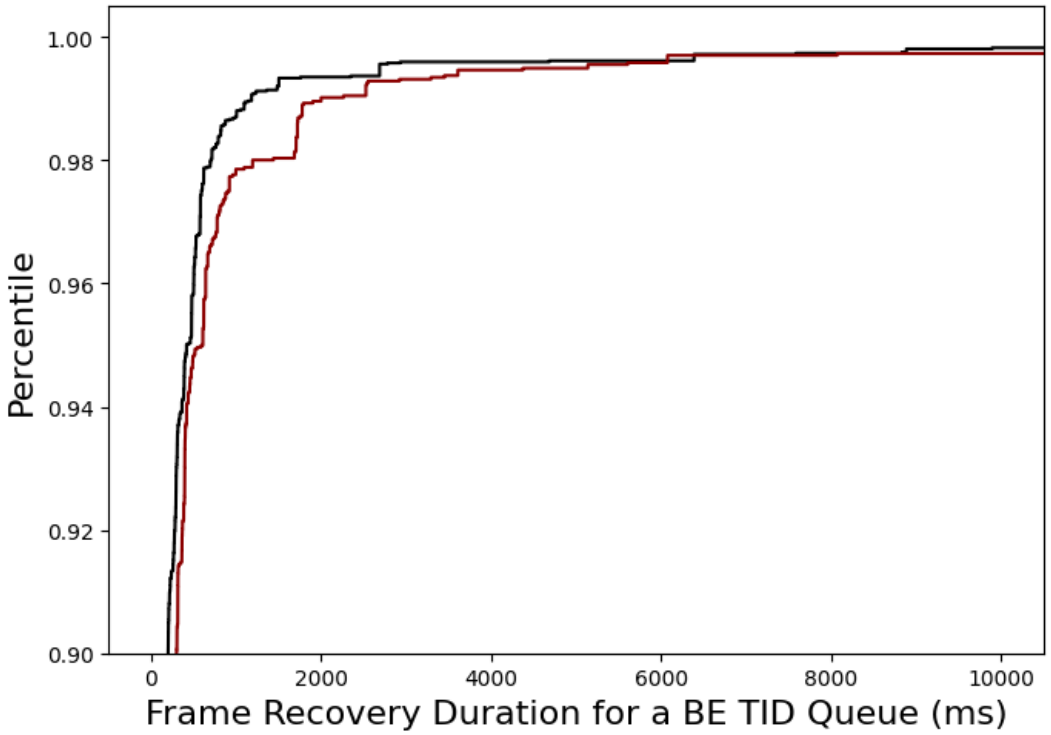


UDP, #STAs/BSS: 20, RA: MinstrelHt, Mob: RandomWalk2d, AI: 20000bytes, TXOP: 2.08ms
#OBSS: 2, BE frames/s: 8036.5, BE Blocked: 4.1%, VO frames/s: 3761.3, VO Blocked: 6.8%
#OBSS: 4, BE frames/s: 3861.4, BE Blocked: 6.1%, VO frames/s: 3347.8, VO Blocked: 9.5%
#OBSS: 6, BE frames/s: 1216.8, BE Blocked: 7.8%, VO frames/s: 2955.7, VO Blocked: 8.7%

UDP, #STAs/BSS: 20, RA: MinstrelHt, Mob: RandomWalk2d, AI: 20000bytes, TXOP: 2.08ms
#OBSS: 2, BE frames/s: 8036.5, BE Blocked: 4.1%, VO frames/s: 3761.3, VO Blocked: 6.8%
#OBSS: 4, BE frames/s: 3861.4, BE Blocked: 6.1%, VO frames/s: 3347.8, VO Blocked: 9.5%
#OBSS: 6, BE frames/s: 1216.8, BE Blocked: 7.8%, VO frames/s: 2955.7, VO Blocked: 8.7%

❑ Scenario 1: 80% BE and 20% VO STAs: **Impact of TXOP Duration**



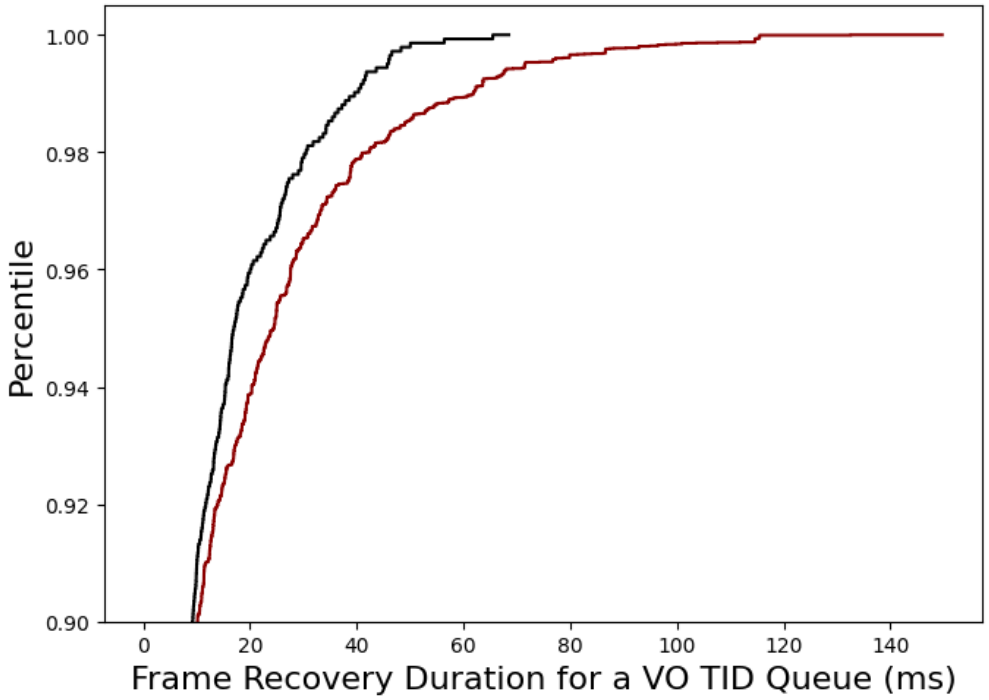


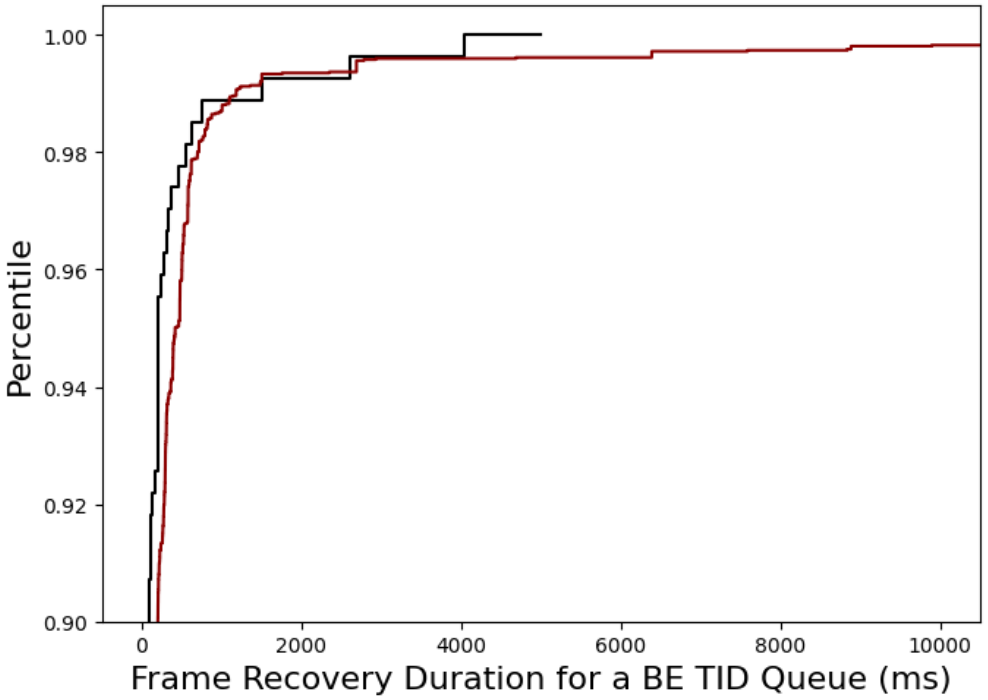
UDP, #OBSS: 6, #STAs/BSS: 20, RA: MinstrelHt, Mob: RandomWalk2d, AI: 20000bytes
TXOP: 2.08ms, BE frames/s: 1216.8, BE Blocked: 7.8%, VO frames/s: 2955.7, VO Blocked: 8.7%
TXOP: 0.52ms, BE frames/s: 750.6, BE Blocked: 7.5%, VO frames/s: 2501.9, VO Blocked: 4.0%

UDP, #OBSS: 6, #STAs/BSS: 20, RA: MinstrelHt, Mob: RandomWalk2d, AI: 20000bytes
TXOP: 2.08ms, BE frames/s: 1216.8, BE Blocked: 7.8%, VO frames/s: 2955.7, VO Blocked: 8.7%
TXOP: 0.52ms, BE frames/s: 750.6, BE Blocked: 7.5%, VO frames/s: 2501.9, VO Blocked: 4.0%

- A shorter TXOP causes a longer average blocking time

❑ Scenario 1: 80% BE and 20% VO STAs: **Impact of Aggregation Intensity (AI)**



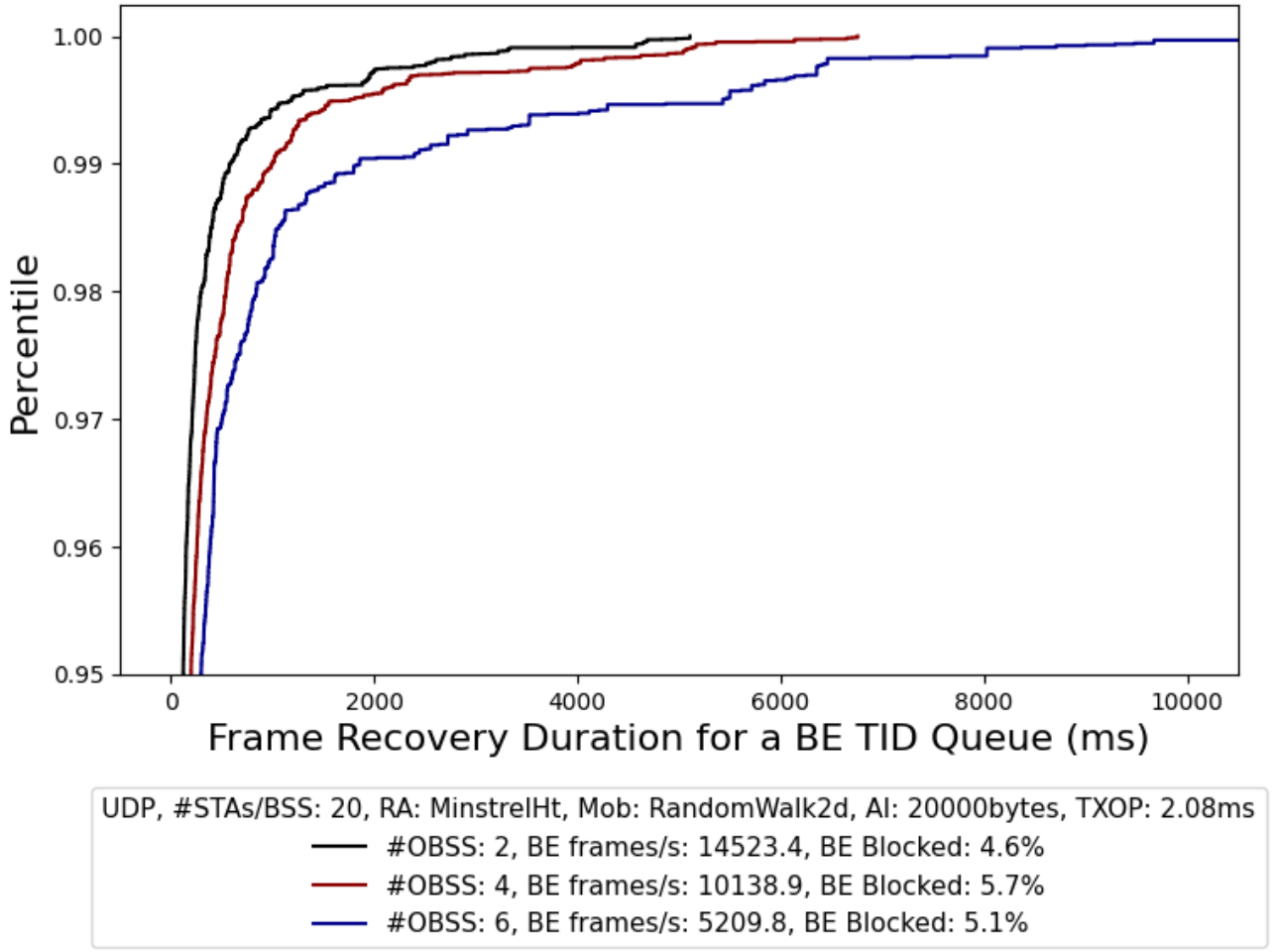


UDP, #OBSS: 6, #STAs/BSS: 20, RA: MinstrelHt, Mob: RandomWalk2d, TXOP: 2.08ms
AI: 3500bytes, BE frames/s: 407.9, BE Blocked: 1.1%, VO frames/s: 1515.5, VO Blocked: 1.6%
AI: 20000bytes, BE frames/s: 1216.8, BE Blocked: 7.8%, VO frames/s: 2955.7, VO Blocked: 8.7%

UDP, #OBSS: 6, #STAs/BSS: 20, RA: MinstrelHt, Mob: RandomWalk2d, TXOP: 2.08ms
AI: 3500bytes, BE frames/s: 407.9, BE Blocked: 1.1%, VO frames/s: 1515.5, VO Blocked: 1.6%
AI: 20000bytes, BE frames/s: 1216.8, BE Blocked: 7.8%, VO frames/s: 2955.7, VO Blocked: 8.7%

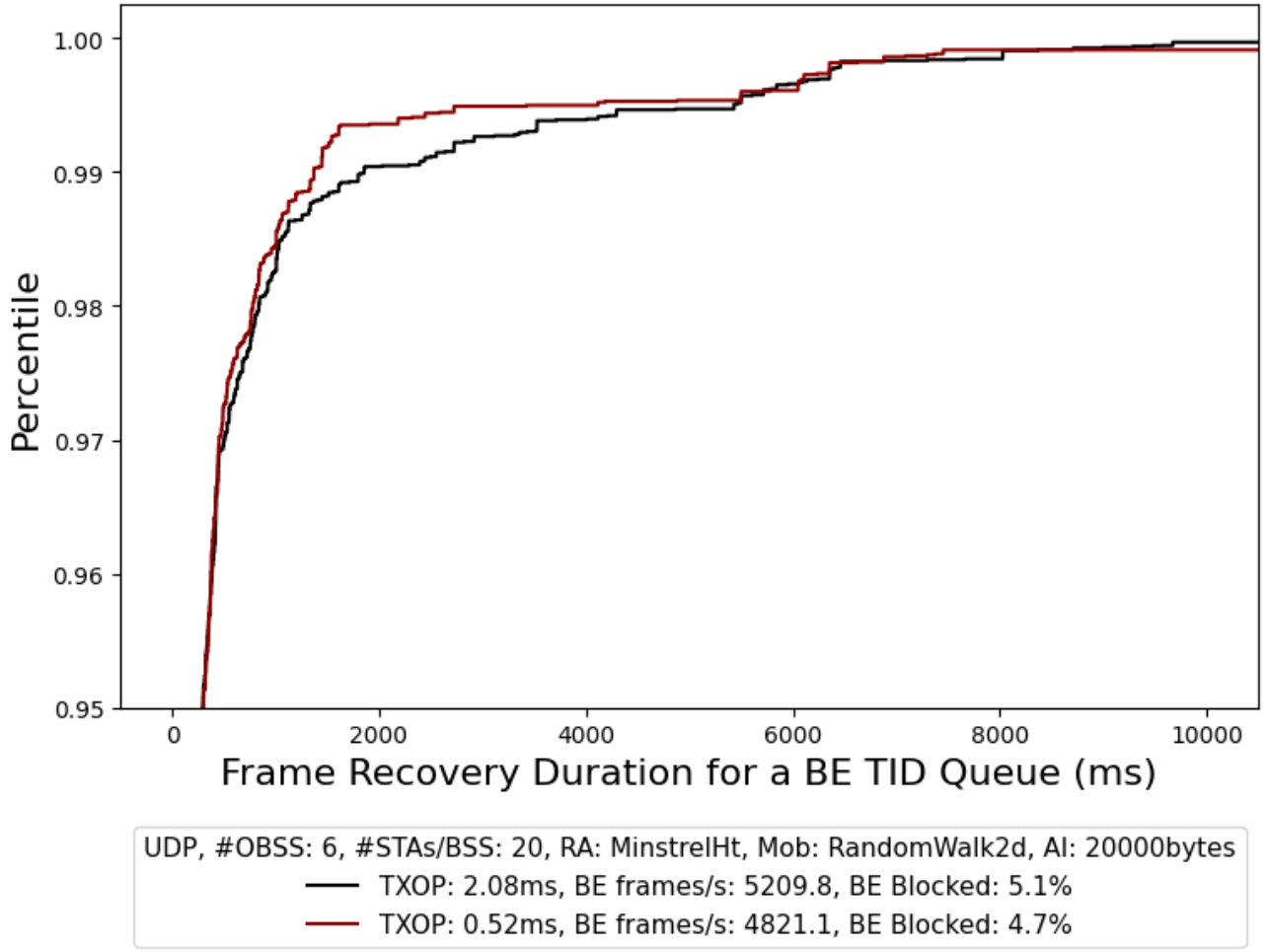
- A larger Aggregation Intensity (AI) causes a longer average blocking time
- A larger Aggregation Intensity (AI) significantly increases the percentage of frames blocked

❑ Scenario 2: 100% BE STAs: **Impact of the Number of OBSSs**



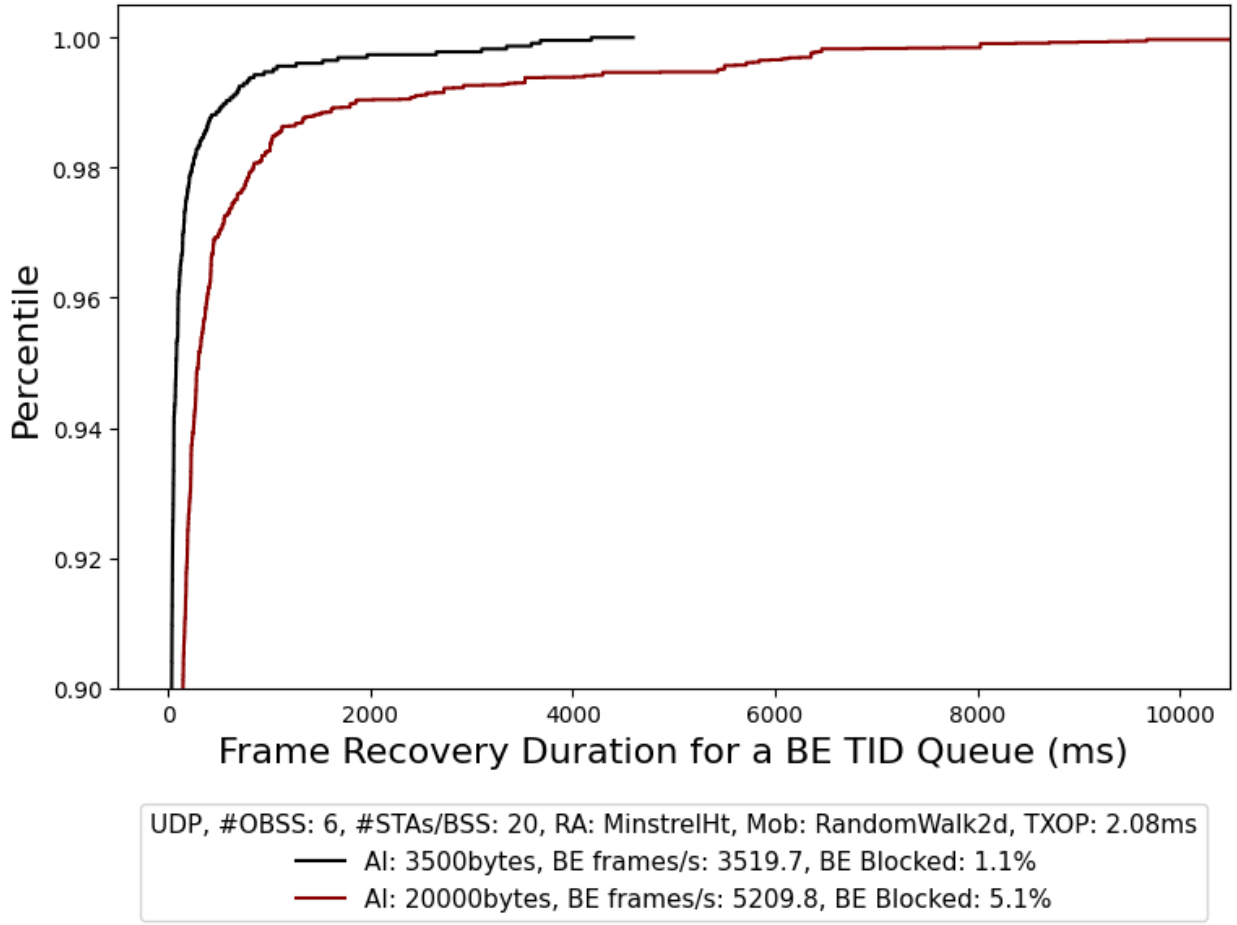
- As the number of OBSSs increases, the blocking frequency and blocking duration increase as well
- Caused by the increase in transmission error rate

❑ Scenario 2: 100% BE STAs: **Impact of TXOP Duration**



- Longer TXOPs have more blocking longer average blocking times

❑ Scenario 2: 100% BE STAs: **Impact of Aggregation Intensity (AI)**



- A larger Aggregation Intensity (AI) causes a longer average blocking time
- A larger Aggregation Intensity (AI) also significantly increases the percentage of frames blocked

Appendix

Other Considerations Related to SCS and QoS Characteristics

QoS Characteristics's 'Delay Bound'

- [11be draft 7.0]: The **Delay Bound** field contains an unsigned integer that specifies the maximum amount of time, in microseconds, targeted to transport an MSDU or A-MSDU belonging to the traffic flow described by this element
- [11be draft 7.0]: Measured between the time marking the arrival of the MSDU, or the first MSDU of the MSDUs constituting an A-MSDU, at the local MAC sublayer from the local MAC SAP and the time of completion of the successful (re)transmission of the MPDU containing the MSDU to the destination

- **Relevance to this contribution**
  - When a STA specifies a Delay Bound for a TID, the sender may drop the frames whose deadline has passed
  - Assuming that there is no timeout-aware OOO queue:
    - Using an IO TID queue: the receiver does not know if the sender has dropped the frame, and therefore, the delivery of OOO frames to upper layers may be unnecessarily delayed
    - Using an OOO TID queue: the receiver may send the OOO frames to the upper layers while the sender is trying to resend those frames
  - **Using a timeout-aware OOO TID queue allows the receiver to implement a wait time that is the same or similar to that of the sender's Delay Bound**

SCS Drop Eligibility

- **The use of timeout-aware OOO TID queues works well with the Drop Eligibility feature of SCS**
  - SCS's Drop Eligibility: A STA should discard MSDUs or A-MSDUs belonging to a TS with the **Drop Eligibility (DEI) subfield set to 1**, prioritizing their removal over those where the subfield is **set to 0**
  - Both sender and receiver may drop a frame whose DEI =1 before other frames

- **Relevance to this contribution**
  - Reliability of sending a stream's frames may get affected by the lack of having enough resources on the sender (e.g., buffer size)
  - A receiving STA may use shorter timeout values for those streams whose frames are marked with DEI

Appendix

Optional enhancements:

Requesting or Sharing Timeout Values of OOO Queues

❑ Requesting or Sharing Timeout Values of OOO Queues

- Each STA can **locally** calculate the distribution of frame recovery times **for both UL and DL**
  - **For DL:** Measuring the interval between the recovery of DL frames
  - **For UL:** Measuring interval between receiving ACK for retransmitted UL frames

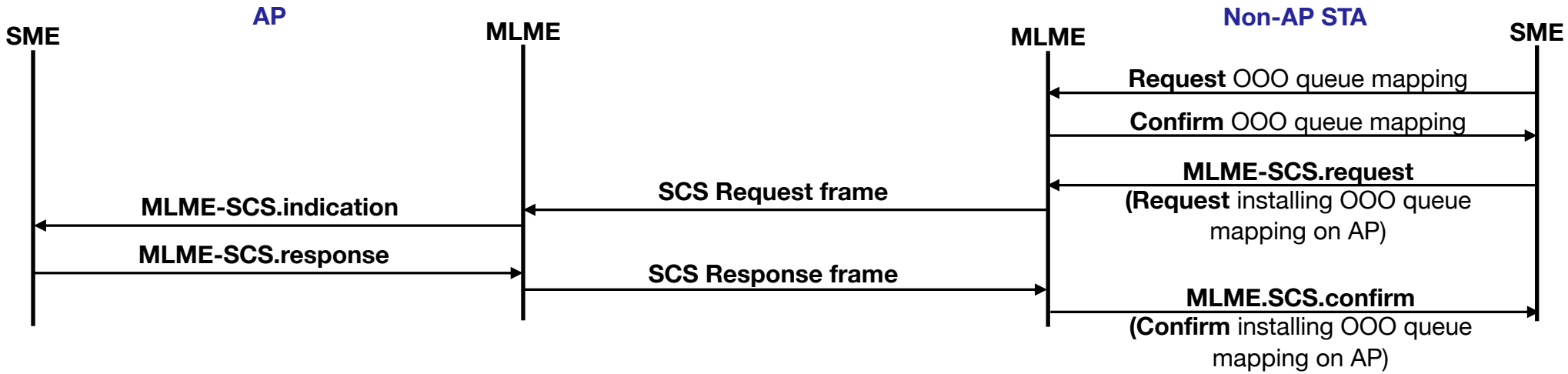
- **In general**, for communication between a STA_X and STA_Y, the delay distribution and mapping of delays can be calculated by one STA and then shared with the other STA

- A STA may request the timeout values of OOO queues from another STA
  - For instance, when a non-AP STA does not have enough resources (e.g., IoT device) to perform timeout to OOO queue mapping, it may request the AP to perform and share the mapping
  - **Another example is when the other STA, such as the AP, does not maintain a per-link OOO TID queues;** in this case, other STAs may request the AP to share its timeout to OOO TID queues mapping when sending their UL traffic

❑ Requesting or Sharing Timeout Values of OOO Queues

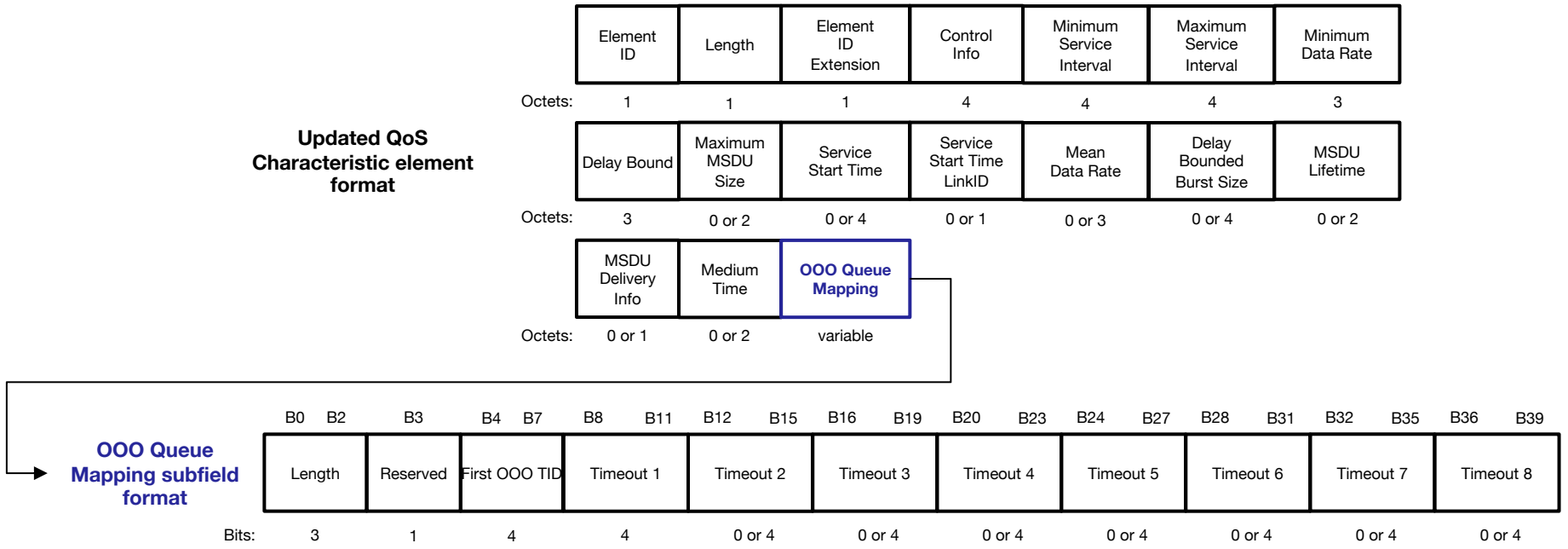
- A non-AP STA may use MLME to enable timeout-based OOO queues, configure the number and parameters of OOO queues (e.g., number of queues/TIDs), or get the current mapping of timeout values to OOO queues
- A non-AP STA may use SCS to request the AP to perform a particular timeout to OOO queue mapping for this STA

- In the following example, we assume the SME of a non-AP STA requests the MAC layer to enable timeout-aware OOO queues and return the established mapping (timeout of OOO queues) to SME
- The SME of the non-AP STA then makes a SCS request to install the same mapping on the AP



❑ Requesting or Sharing Timeout Values of OOO Queues

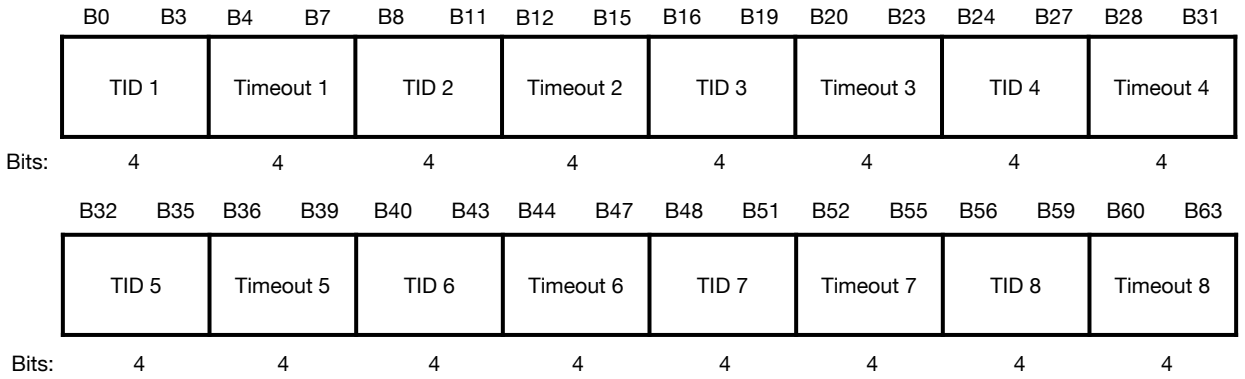
- **Requesting OOO mapping:** A STA can **request OOO queues mappings** from another STA through a **SCS Request with a unique Robust Action value *x* (in 6-255)**
- **Sharing and enforcing QoS mapping:** A STA_X can use a Robust Action value *y* (in 6-255) and the QoS Characteristics field to share its OOO queue mapping with another STA_Y and ask STA_Y to use the shared mapping when receiving frames from STA_X



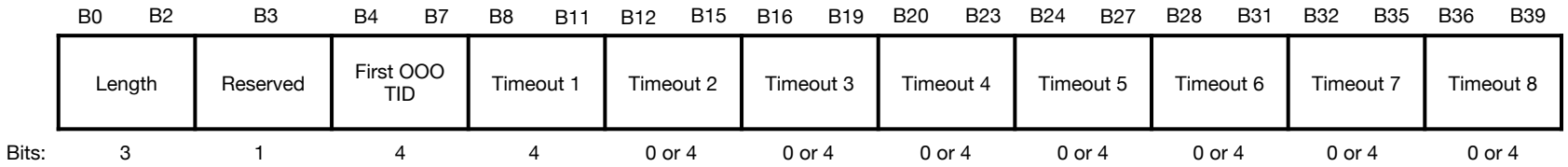
❑ Requesting or Sharing Timeout Values of OOO Queues

- Two sample encoding formats to share the mapping of timeout-aware TID queues

Sample Encoding 1: Sharing timeout-aware OOO queues



Sample Encoding 2: Sharing timeout-aware OOO queues



❑ Requesting or Sharing Timeout Values of OOO Queues

- If the Robust Action field in SCS agreements is not used, the type of transaction can be encoded using the **Control Info field of QoS Characteristic elements**
- The example below shows sharing and enforcing to use the shared mapping on another STA