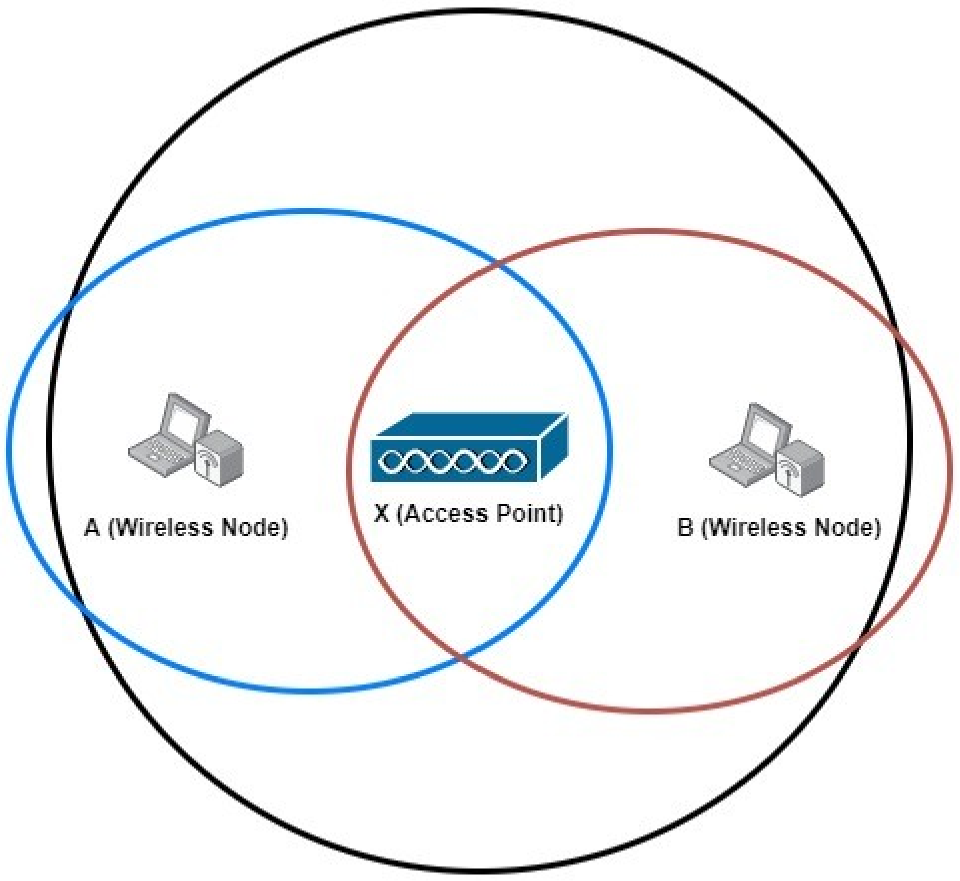
COMP 2211 Network Summative

Part 2

Question 1



We want to show that

- There are 3 nodes
- A and B are wireless nodes
- X is an Access Point
- A and B cannot hear each other
- X can hear both A and B

Question 2

Transmission Situation

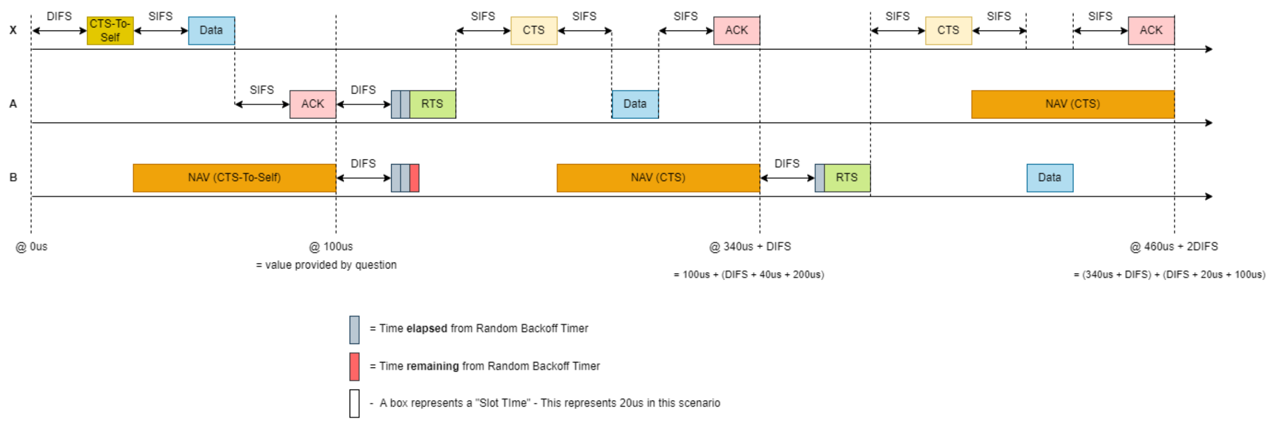
Due to the presence of an AP, this is a wireless network that is likely using the 802.11 Infrastructure mode. A network like this can employ CSMA/CA through DCF w/ RTS/CTS/ACK for larger transmissions. Since the logical topology suffers from the hidden terminal problem, RTS/CTS helps mitigate this.

Transmission Procedure

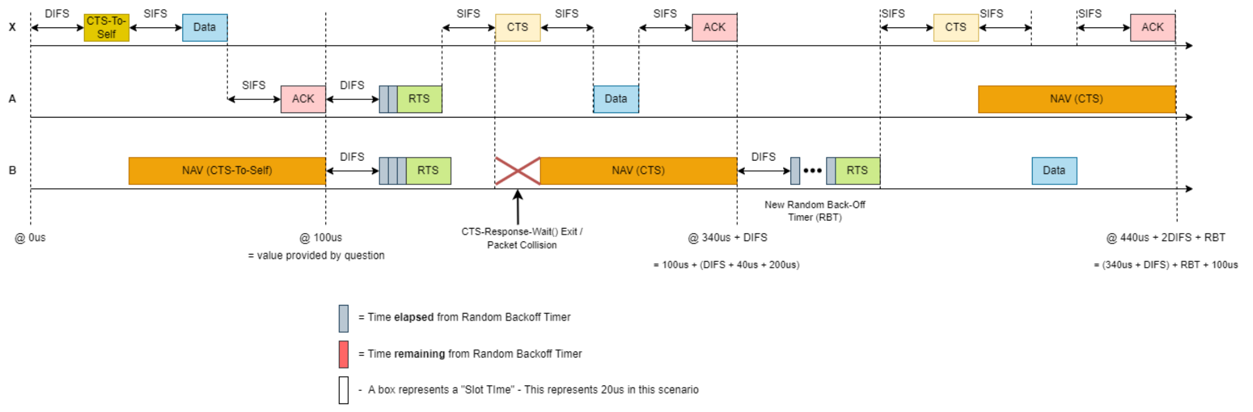
There are a range of configurations and variables that effect timings in the above transmission procedure, which is why the times provided in this answer will be relative. There are a few generalizations and assumptions made including:

- The channel was idle before $0\mu s$ and there were no previous frames sent – Hence why after the first DIFS, there is no back-off timer shown
- A single packet can be encapsulated into multiple frames – However, since number of frames hasn’t been specified and to keep it simple, this has been generalized to “Data” in the diagram

If Slot Time (ST) > SIFS, then the transmission follows this diagram:


X
A
B
DIFS
CTS-To-Self
SIFS
Data
SIFS
ACK
DIFS
RTS
SIFS
CTS
SIFS
Data
SIFS
ACK
NAV (CTS-To-Self)
DIFS
NAV (CTS)
DIFS
RTS
SIFS
CTS
SIFS
SIFS
ACK
NAV (CTS)
Data
@ 0us
@ 100us
= value provided by question
@ 340us + DIFS
= 100us + (DIFS + 40us + 200us)
@ 460us + 2DIFS
= (340us + DIFS) + (DIFS + 20us + 100us)
= Time elapsed from Random Backoff Timer
= Time remaining from Random Backoff Timer
- A box represents a "Slot TIme" - This represents 20us in this scenario

Otherwise, Slot Time (ST) < SIFS, then the transmission follows this diagram:


X
A
B
DIFS
CTS-To-Self
SIFS
Data
SIFS
ACK
DIFS
RTS
SIFS
CTS
SIFS
Data
SIFS
ACK
NAV (CTS-To-Self)
DIFS
RTS
NAV (CTS)
DIFS
RTS
New Random Back-Off Timer (RBT)
CTS-Response-Wait() Exit / Packet Collision
SIFS
CTS
SIFS
SIFS
ACK
NAV (CTS)
Data
@ 0us
@ 100us
= value provided by question
@ 340us + DIFS
= 100us + (DIFS + 40us + 200us)
@ 440us + 2DIFS + RBT
= (340us + DIFS) + RBT + 100us
= Time elapsed from Random Backoff Timer
= Time remaining from Random Backoff Timer
- A box represents a "Slot TIme" - This represents 20us in this scenario

I believe that these two diagrams are sufficient to answer the question – However, there is a written out explanation that explains the procedure and provides more detail below if necessary.

<u>**Transmission 1:** “X” -> “A”</u> **(0μs)**

- Consider time takes for X to prepare packet = Frame Creation Time (FCT) (0μs)
- “X” performs carrier sense for the duration of DIFS – returns Idle (0μs + FCT)
    - “X” performs physical carrier sense – returns Idle
    - “X” performs virtual carrier sense – returns Idle
- “X” sends a “CTS-To-Self” control frame (FCT + DIFS)
    - All other connected nodes receive this when performing carrier sense and updates their NAV timers to the duration specified in the “CTS-To-Self” control frame
- “X” sends the data through an unspecified number of frames (FCT + DIFS + SIFS)
    - “X” will set the destination MAC address on the frames amongst other fields.
    - All other nodes will drop the frames, but “A” will read and process it.
- After receiving the data, “A” sends an “ACK” frame to “X” (FCT + DIFS + DATA + 2SIFS)
- NOTE: The question states that this transmission takes 100μs – therefore it ends at 100μs.

<u>Other Occurrences During Transmission 1</u>

- “A” is ready to transmit to “X” (30μs)
    - “A” performs virtual carrier sense – Returns Busy
    - “A” counts down the NAV Timer
- “B” is ready to transmit to “X” (70μs)
    - “B” performs virtual carrier sense – Returns Busy
    - “B” counts down the NAV Timer

<u>**Transmission 2**: “A” -> “X”</u> **(100μs)**

- “A” and “B” simultaneously perform carrier sense for the duration of DIFS – Returns Idle (100μs)
- “A” and “B” perform carrier sense until their random-backoff timers to finish (100 μs + DIFS)
    - A finishes its timer and sends an RTS packet (100 μs + DIFS + 40μs)

  <u>Assuming (Slot Time > SIFS)</u>

    - “X” replies with a CTS packet with address of “A” (140μs + DIFS + SIFS)
    - “B“ hears the CTS packet **before** the random-backoff finishes (140μs + DIFS + SIFS)

  <u>Assuming (Slot Time < SIFS)</u>

    - “B” finishes its backoff timer and sends an RTS packet (140μs + DIFS + 20 μs)
    - “X” replies with a CTS packet with address of “A” (140μs + DIFS + SIFS)

  - "B" hears the CTS packet **after** its random-backoff finishes (160μs + DIFS + SIFS)
  - "X" ignores "B"'s RTS packet (160μs + DIFS + SIFS)

Continuing with Transmission 2

- "B" sets it's NAV timer using the CTS packet's duration field (140μs + DIFS + SIFS)
- "A" sends the data to the destination "X" (140μs + DIFS + SIFS)
- "X" returns an ACK packet to "X" (140μs + DIFS + DATA + SIFS)
- NOTE: The question states this transmission takes 200μs – Assuming this is from the NAV (RTS)
  - Start of Transmission (140μs + DIFS) + Transmission Duration (200μs) = **340μs + DIFS**

Transmission 3 "B" -> "X" **(340μs + DIFS)**

- "B" performs carrier sense for DIFS time **(340μs + 2DIFS)**

  Assuming (Slot Time > SIFS)

  - "B" performs carrier sense for the remaining backoff timer (20μs¿ – Returns Idle
    (340μs + 2DIFS + 20μs)

  Assuming (Slot Time < SIFS)

  - "B" performs carrier sense for a new random backoff timer duration (RBT) – Returns Idle (340μs + 2DIFS + RBT)

Continuing with Transmission 3 1. **(ST > SIFS),** 2. **(ST < SIFS)**

- "B" sends an RTS packet to "X" (360μs + 2DIFS) | (340μs + 2DIFS + RBT)
  - "A" cannot hear this so doesn't set it's NAV timer
- "X" responds with a CTS packet (360μs + 2DIFS + SIFS) | (340μs + 2DIFS + RBT + SIFS)
  - "A" hears this and so sets it's NAV timer
- "B" sends the data to "X" (360μs + 2DIFS + 2SIFS) | (340μs + 2DIFS + RBT + 2SIFS)
- "X" replies with an ACK frame to "B" (360μs + 2DIFS + 2SIFS + DATA) | (340μs + 2DIFS + RBT + 2SIFS + DATA)
- NOTE: The question states this transmission takes 100μs - Assuming this is from the NAV (RTS):
  - Start of Transmission + Transmission Duration = End Time
  - (ST > SIFS) → (360μs + 2DIFS) + (100μs) **= 460μs + 2DIFS**
  - (ST < SIFS) → (340μs + 2DIFS + RBT) + (100μs) = **440μs + 2DIFS + RBT**

DIFS is the time that should be used when performing carrier sense before deciding whether to send

SIFS is the time for a wireless interface to process a frame and respond

Slot is the longest time for any transmission to be sent in the network

<u>Question 3</u>

<u>Previous Protocol Issues</u>

As a network grows past 4 or 5 nodes, more contention method problems manifest or become more prevalent regardless of whether in Ad-Hoc mode or Infrastructure mode. The protocol that was used in the previous question isn't perfect and inhibits successful transmissions in a wide range of scenarios. These are displayed in figure 1 with corresponding explanations:

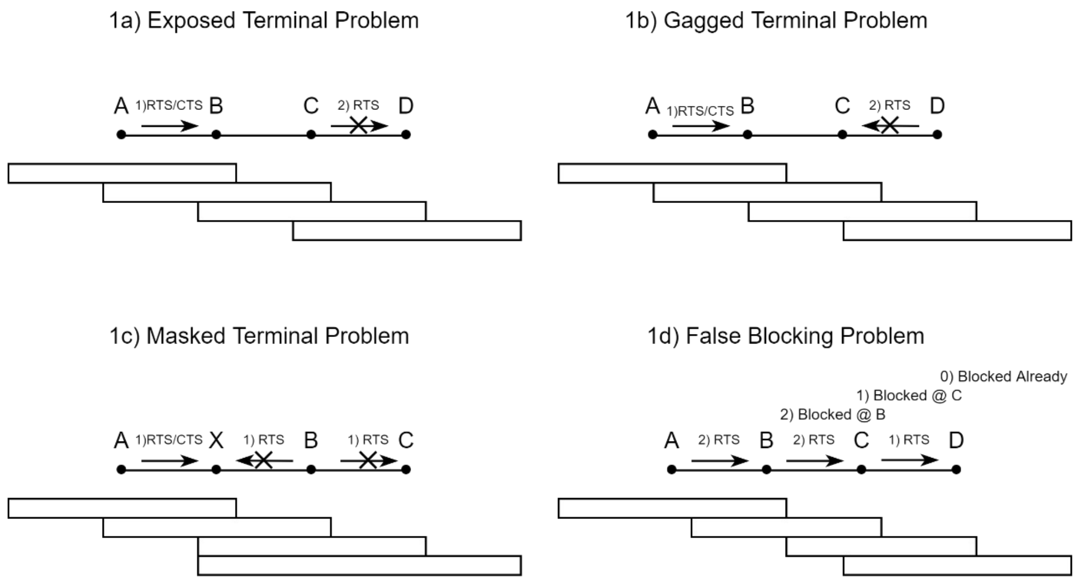


Figure 1

- *Exposed Terminal Problem* – When a node that doesn't participate in a transmission is blocked by the sender (via an RTS frame), even though the receiver is out of the blocked node's range. (*Fig 1a)*
- *Gagged Terminal Problem* – When a node that doesn't participate in a transmission is blocked by the recipient (via a CTS frame), even though the sender is out of the blocked node's range. (*Fig 1b)*
- *Masked Terminal Problem* – When a hidden node "B" sends an RTS frame to "X" just before receiving a CTS frame from "X" destined for hidden node "A". "B"'s RTS frame causes a subset of its neighbourhood that is hidden from A to be *masked* from "X"'s CTS broadcast to A. (*Fig 1c)*
- *False Blocking Problem* – When a falsely blocking node can't reply to neighbouring RTS/CTS frames, causing blocking issues to propagate across the network via neighbours. (*Fig 1d)*

The protocol used in the previous question will be modified in order to mitigate and reduce most of the above issues. These issues arise from stations being unable to process control frames after being blocked from previous control frames. The proposed solution will continue to use RTS/CTS/ACK frames, but the Virtual Carrier Sense will work in an asynchronous manner in order to continuously process these control frames to determine what type of activity the station is permitted to perform.

In the previous answer under the scenario where Slot Time (ST) < SIFS, we induced a *masked terminal problem*, (where the big red cross is at). Depending on how we decided to deal with it, this could have resulted in the later node constantly resending RTS packets until it successfully observes a successful handshake (either through itself or a neighbour). However, our proposed solution is intended to mitigate that (hence the "CTS-Response-Wait End()" label on the diagram.

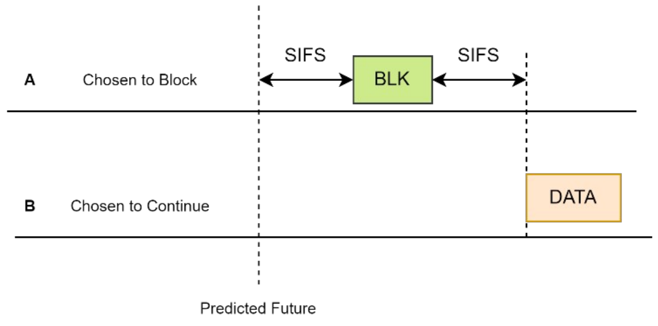
<u>Proposed Solution</u>

Laying out foundations

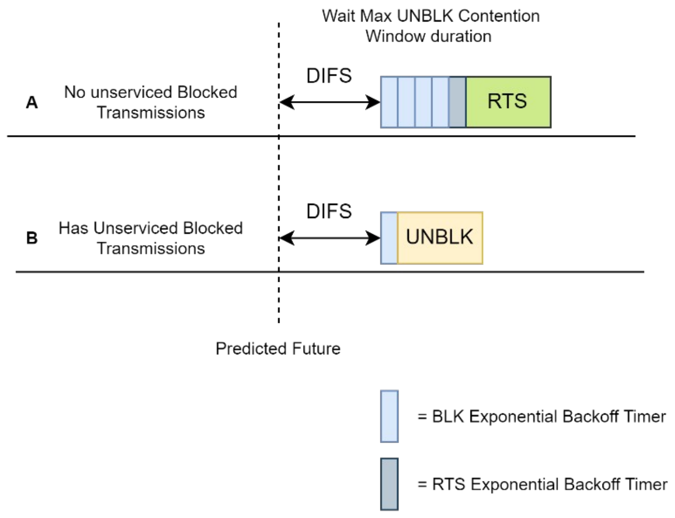
Each node now has a range of modes that determines a node's current permitted operation set, which is determined by the current transmission situation: 1) *Control Mode* – The node is only allowed to send and process "Control" and "Management" Frames (every node uses this mode all the time), 2) *Read Data Only Mode* – The node is in *control mode* and can only perform *read data* operations, *3) Send Only Mode* – The node is in *control mode* and can only perform *send data* operations, 4) *Send + Read Data Mode* – The node is in *control mode* and can perform *send data* and *read data* operations.

A distinction of neighbours is now made 1) *Visible Neighbours* – these are neighbours that a node discovers by listening to broadcasts where the frame's source address is a neighbour 2) *Aware Neighbours* – these are neighbours that a node has never heard from (or haven't heard in a while), but overhears frames where the destination address is this neighbour. (NOTE: Visible Neighbours are a subset of Aware Neighbours, although the data structures holding them are mutually exclusive to avoid redundancy)

Two new frames are introduced called *BLOCK* (BLCK) and *UNBLOCK* (UBLCK) in order to handle stations moving and roaming. When two simultaneously transmitting aware neighbours become visible to each other, depending on some condition (e.g. SRTF), one of the stations pauses their transmission and broadcasts a *BLOCK* Frame, while the other also pauses for a short duration before continuing transmitting. (Depending on implementation method, it’s likely to require more than just firmware modifications).



The receiver (AP) pauses this transmission and sends an ACK frame to notify all nodes of the free medium. After each successful ACK, any previously *BLOCK*ed nodes that are now available, can send an *UNBLOCK* frame to the AP. The exponential backoff timer for RTS and *UNBLOCK* are modified so that *UNBLOCK* window is first, and RTS window is second. The AP will then broadcast a CTS to this previously blocked node.


Wait Max UNBLK Contention Window duration
A
No unserviced Blocked Transmissions
DIFS
RTS
B
Has Unserviced Blocked Transmissions
DIFS
UNBLK
Predicted Future
= BLK Exponential Backoff Timer
= RTS Exponential Backoff Timer

Modified DCF and Modified Virtual Carrier Sense

Each node applies modes on a per-neighbour basis - by continuously listening for broadcasts, a station constantly updates the *mode* that it is permitted to operate under in relation to a neighbour. Unlike the previous protocol that completely blocks the node using NAV timers, *Control-Mode* permits the node to operate under assigned modes which allow for transmissions that would have otherwise been inhibited, which increases network throughput.

Stations are less likely to experience blocking problems induced by *gagged terminal problem, exposed terminal problem,* and *masked terminal problem*. In scenarios where stations suffer this problem, the almost continuous monitoring reduces the negative effects significantly. The mitigations to these problems additionally mitigates the *false blocking problem* from propagating through the network.

Stations are also somewhat able to determine (and maybe even predict) whether a collision has occurred by evaluating the sequence, volume, and type of control frames in a transmission scenario. If a node has determined a collision has occurred, instead of sending control frames or resending data like other approaches do, the node could use the largest CTS.duration timer with an exponential backoff timer before sending frames, in order to avoid exacerbating the problem

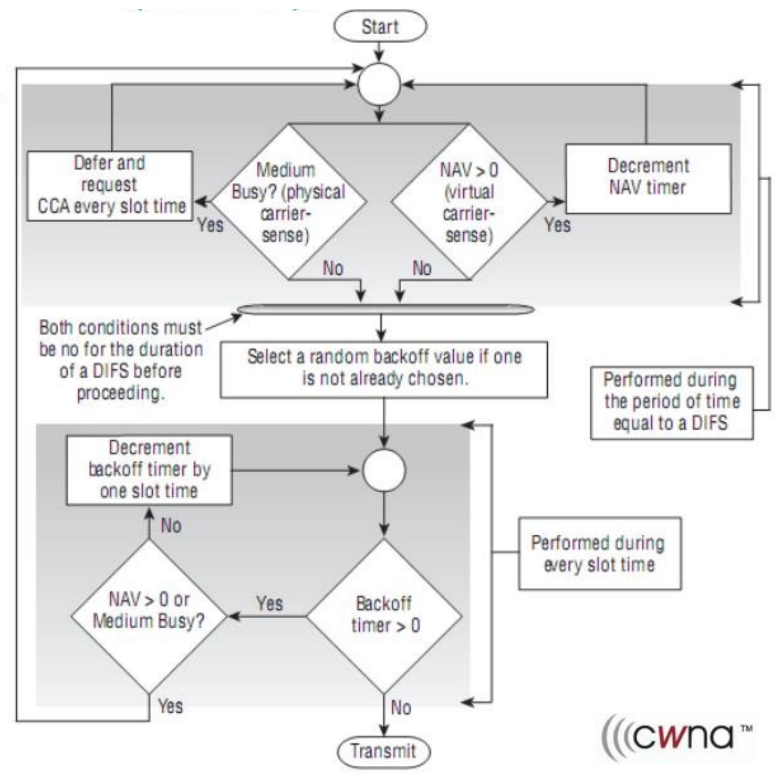
In order to enable the above system to work, we have removed the NAV timer and replaced it with the Mode Allocation Vector (MAV) timer. When a sender node is performing a RTS/CTS handshake, it sets the MAV timer with the duration of the handshake (2SIFS).

There are a few important things to understand before attempting to understand the diagrams explaining this protocol:

- The “buffers” described below are sets of data structures that each station constructs on a per-neighbour basis that enables it to determine its own mode and additional actions.
- Some of the control flow has been rearranged into short flows of logic in order to more easily comprehend the fundamental idea that this protocol works on
- In order to avoid redundant or complex implementation details, certain paths have been omitted for the sake of ease of comprehension
- 802.11 Frames have space for 4 addresses, below, they will be accessed using dot notations (e.g. Frame.source, Frame.destination, Frame.ap (Since we are looking at Infrastructure Mode). Dot notation may also be used to access different fields in a frame.
- BLK and UNBLK are control frames and will be designated some predetermined available constant type/subtype values.

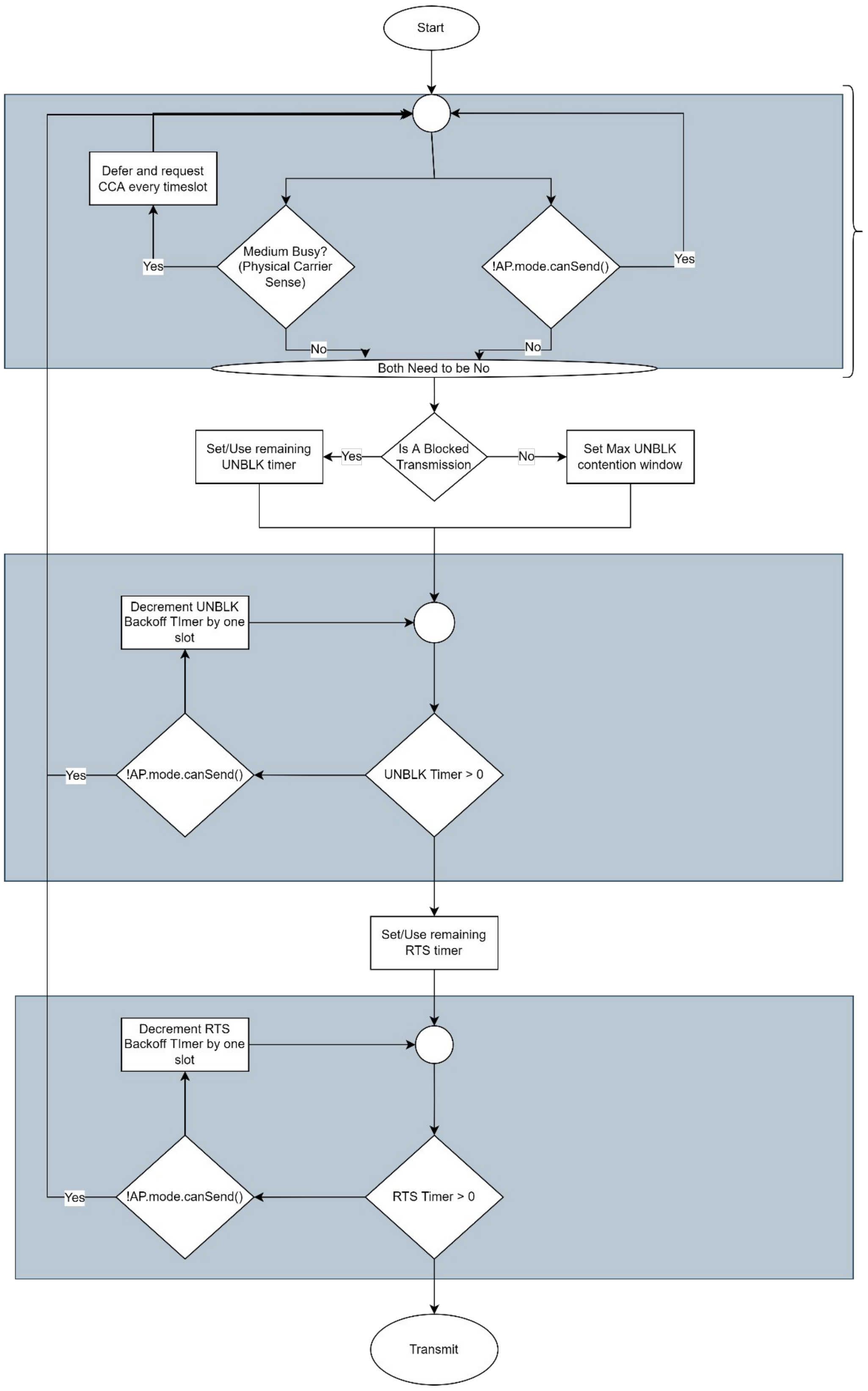
Original DCF

I will be using this high-level flowchart diagram explaining DCF by CWNA as a basis in order to show how the proposed modification differ from the original.


Start
Defer and request CCA every slot time
Yes
Medium Busy? (physical carrier-sense)
No
NAV > 0 (virtual carrier-sense)
No
Yes
Decrement NAV timer
Both conditions must be no for the duration of a DIFS before proceeding.
Select a random backoff value if one is not already chosen.
Performed during the period of time equal to a DIFS
Decrement backoff timer by one slot time
No
NAV > 0 or Medium Busy?
Yes
Backoff timer > 0
Yes
No
Performed during every slot time
Transmit
cwna

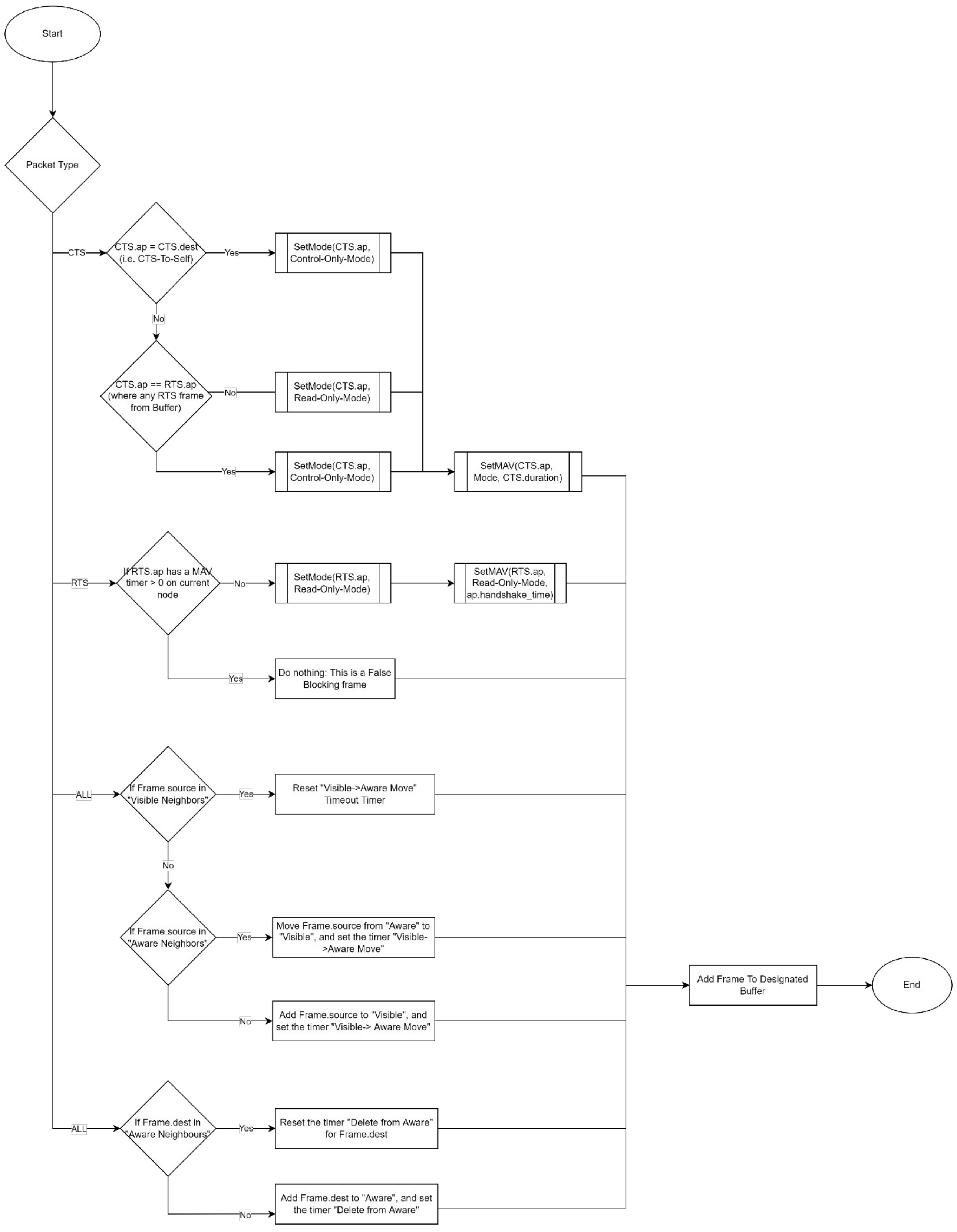
Modified DCF

A diagram of the modified DCF is provided below. Unlike the original method, virtual carrier sense is performed whenever possible in order to build up an understanding of its neighbourhood. There is also the addition of the UNBLCK exponential timer.


Start
Defer and request CCA every timeslot
Yes
Medium Busy? (Physical Carrier Sense)
!AP.mode.canSend()
Yes
No
No
Both Need to be No
Set/Use remaining UNBLK timer
Yes
Is A Blocked Transmission
No
Set Max UNBLK contention window
Decrement UNBLK Backoff TImer by one slot
Yes
!AP.mode.canSend()
UNBLK Timer > 0
Set/Use remaining RTS timer
Decrement RTS Backoff TImer by one slot
Yes
!AP.mode.canSend()
RTS Timer > 0
Transmit

Modified Virtual Carrier Sense (Pt 1)

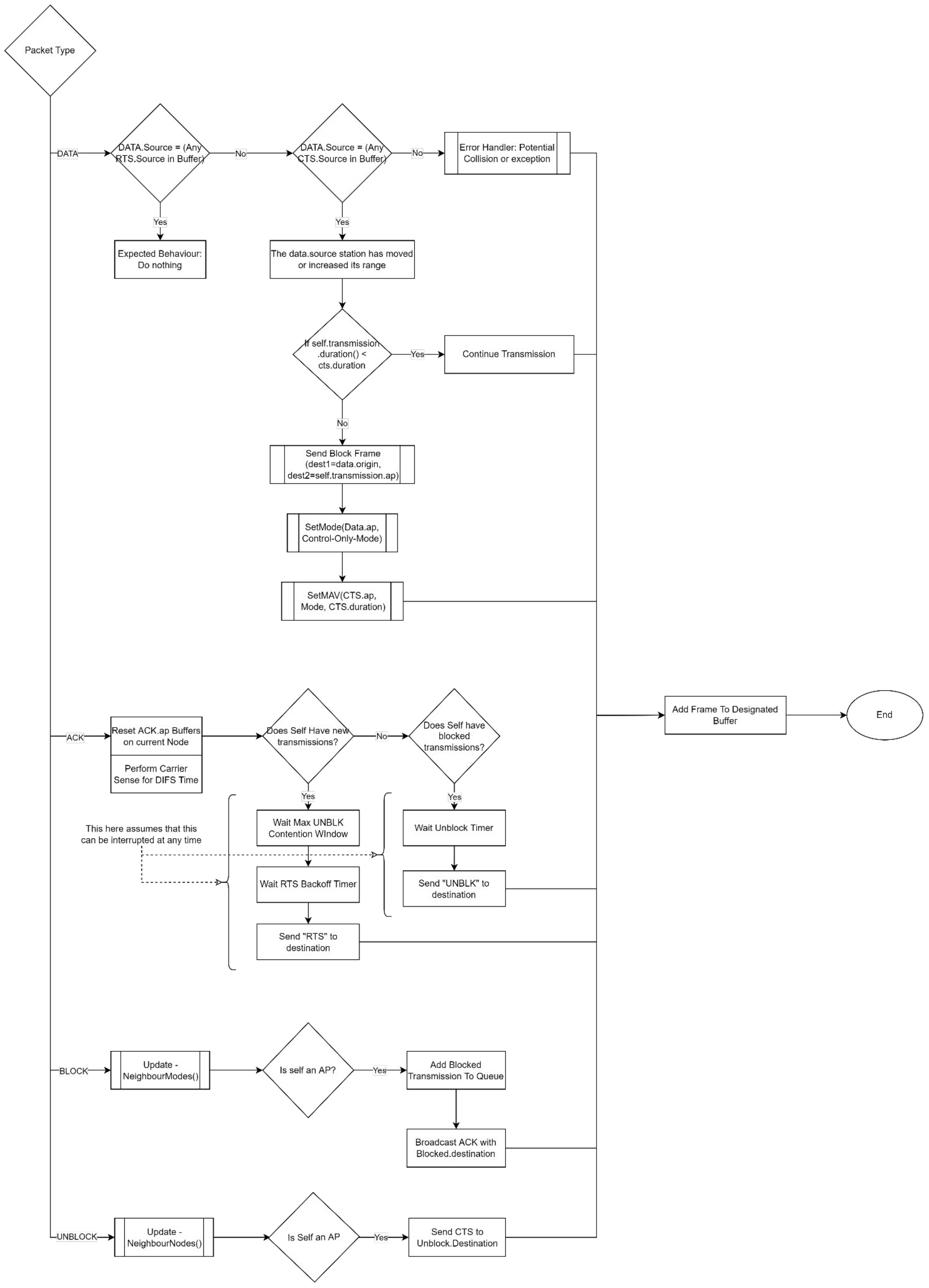
This is the second part of the virtual carrier sense to explain how this modified mechanism processes transmissions.



Start
Packet Type
CTS
CTS.ap = CTS.dest (i.e. CTS-To-Self)
Yes
SetMode(CTS.ap, Control-Only-Mode)
No
CTS.ap == RTS.ap (where any RTS frame from Buffer)
No
SetMode(CTS.ap, Read-Only-Mode)
Yes
SetMode(CTS.ap, Control-Only-Mode)
SetMAV(CTS.ap, Mode, CTS.duration)
RTS
If RTS.ap has a MAV timer > 0 on current node
No
SetMode(RTS.ap, Read-Only-Mode)
SetMAV(RTS.ap, Read-Only-Mode, ap.handshake_time)
Yes
Do nothing: This is a False Blocking frame
ALL
If Frame.source in "Visible Neighbors"
Yes
Reset "Visible->Aware Move" Timeout Timer
No
If Frame.source in "Aware Neighbors"
Yes
Move Frame.source from "Aware" to "Visible", and set the timer "Visible->Aware Move"
No
Add Frame.source to "Visible", and set the timer "Visible-> Aware Move"
ALL
If Frame.dest in "Aware Neighbours"
Yes
Reset the timer "Delete from Aware" for Frame.dest
No
Add Frame.dest to "Aware", and set the timer "Delete from Aware"
Add Frame To Designated Buffer
End

Modified Virtual Carrier Sense (Pt 2)

This is the second part of the virtual carrier sense to explain how this modified mechanism processes transmissions


Packet Type
DATA
DATA.Source = (Any RTS.Source in Buffer)
No
DATA.Source = (Any CTS.Source in Buffer)
No
Error Handler: Potential Collision or exception
Yes
Expected Behaviour: Do nothing
Yes
The data.source station has moved or increased its range
If self.transmission .duration() < cts.duration
Yes
Continue Transmission
No
Send Block Frame (dest1=data.origin, dest2=self.transmission.ap)
SetMode(Data.ap, Control-Only-Mode)
SetMAV(CTS.ap, Mode, CTS.duration)
Add Frame To Designated Buffer
End
ACK
Reset ACK.ap Buffers on current Node
Perform Carrier Sense for DIFS Time
Does Self Have new transmissions?
No
Does Self have blocked transmissions?
Yes
Wait Max UNBLK Contention WIndow
Yes
Wait Unblock Timer
This here assumes that this can be interrupted at any time
Wait RTS Backoff Timer
Send "UNBLK" to destination
Send "RTS" to destination
BLOCK
Update - NeighbourModes()
Is self an AP?
Yes
Add Blocked Transmission To Queue
Broadcast ACK with Blocked.destination
UNBLOCK
Update - NeighbourNodes()
Is Self an AP
Yes
Send CTS to Unblock.Destination

Conclusion

Most of the functionality of the components not mentioned is very likely to remain the same or be tweaked only to accommodate the new features. At the cost of energy consumption, this solution theoretically mitigates the problems mentioned at the start of this answer with minimal overhead. However, whether this solution definitely works in a practical environment can only be determined with implementation and testing.