Networking on WARP

Chris Hunter
Rice University

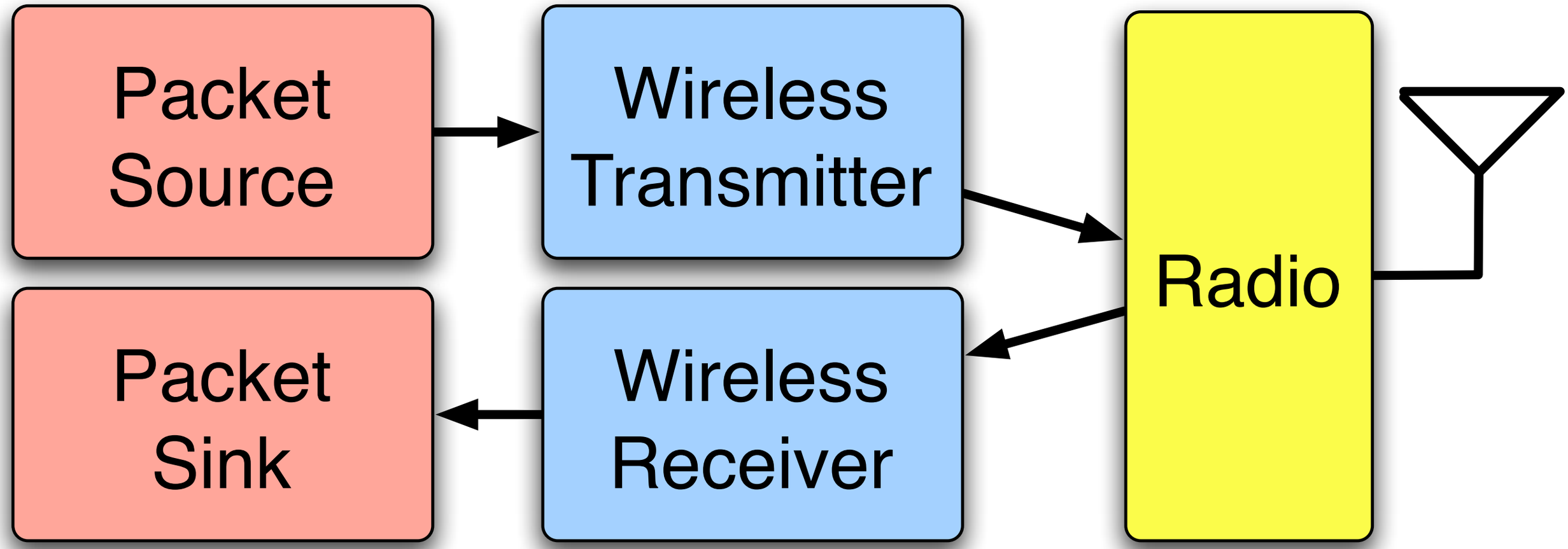
WARP Workshop at Rice University
November 15, 2008

warp.rice.edu

Today's Agenda

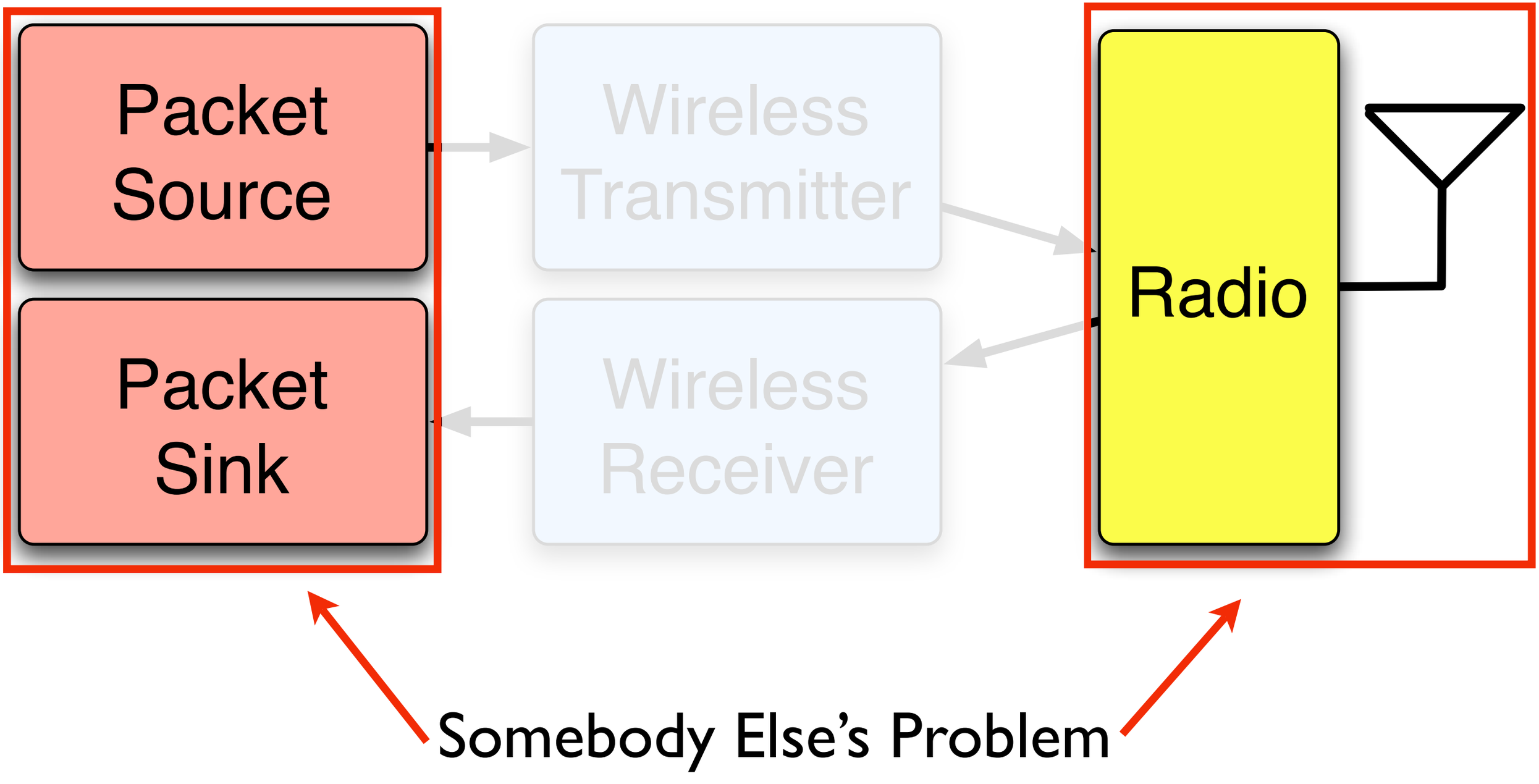
- Outstanding questions?
- Networking on WARP lecture
- Labs 4, 5, and 6
- Workshop wrap-up

Simple Wireless Node



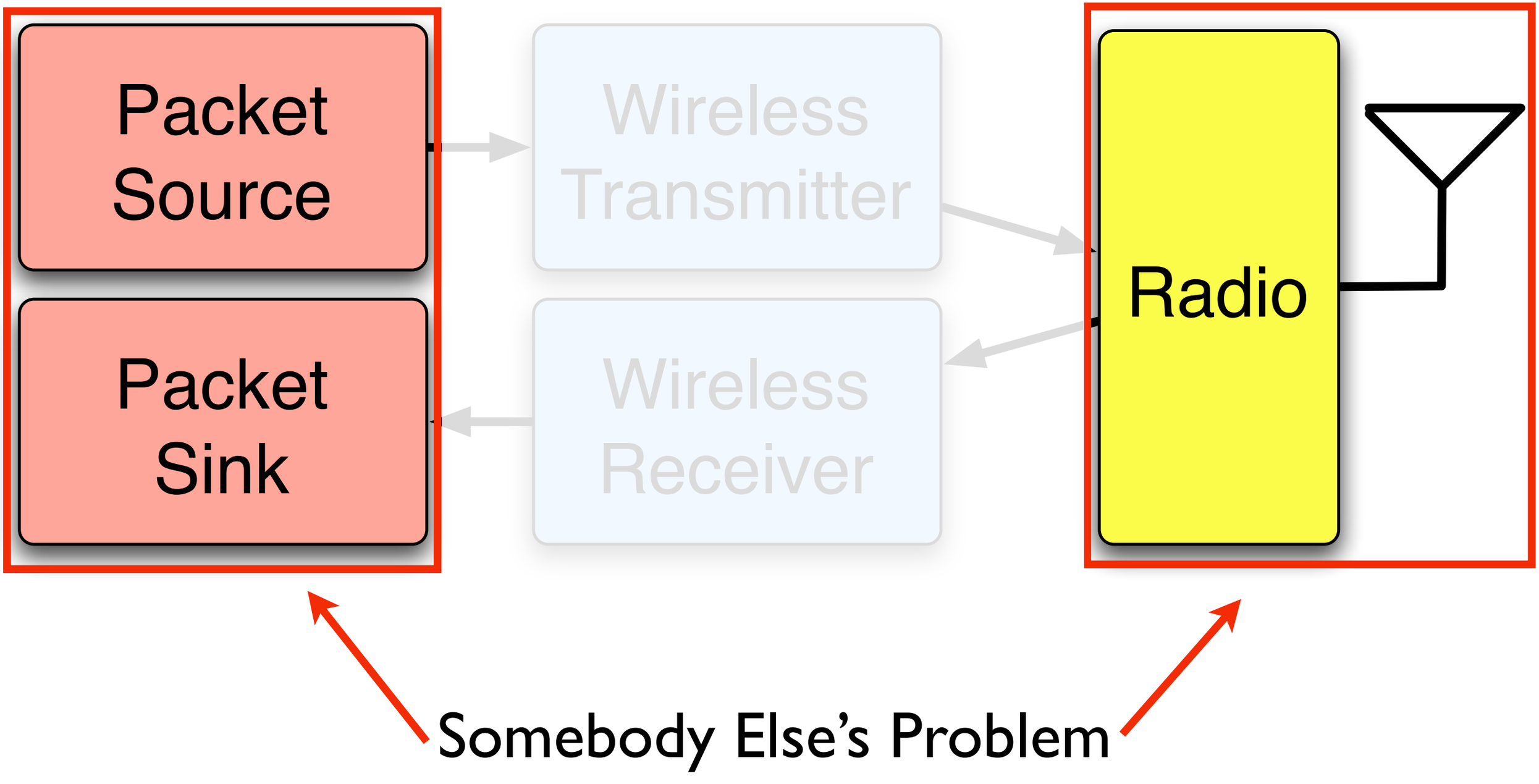
Physical Layer Basics

Simple Wireless Node



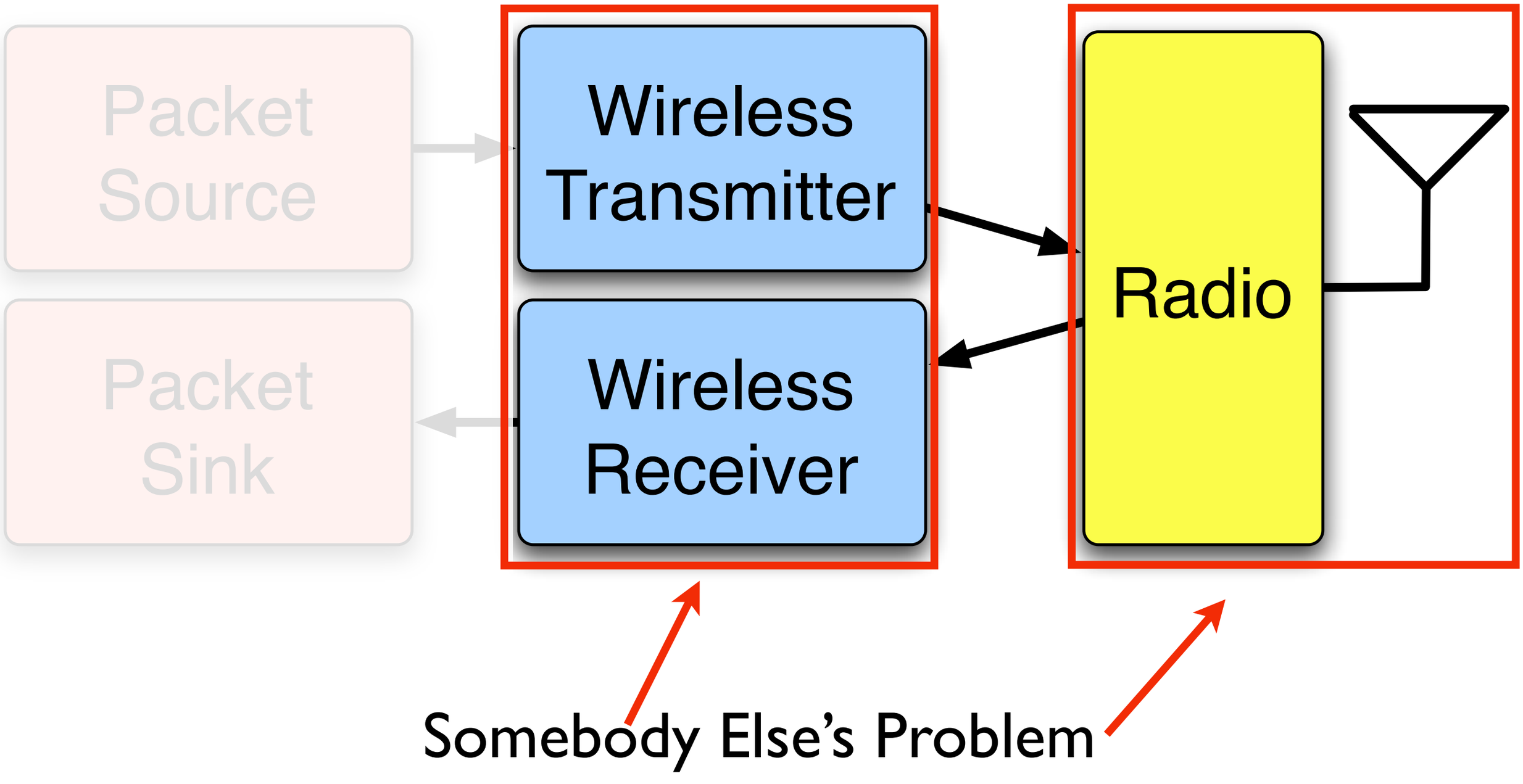
Network Layer Basics

Simple Wireless Node



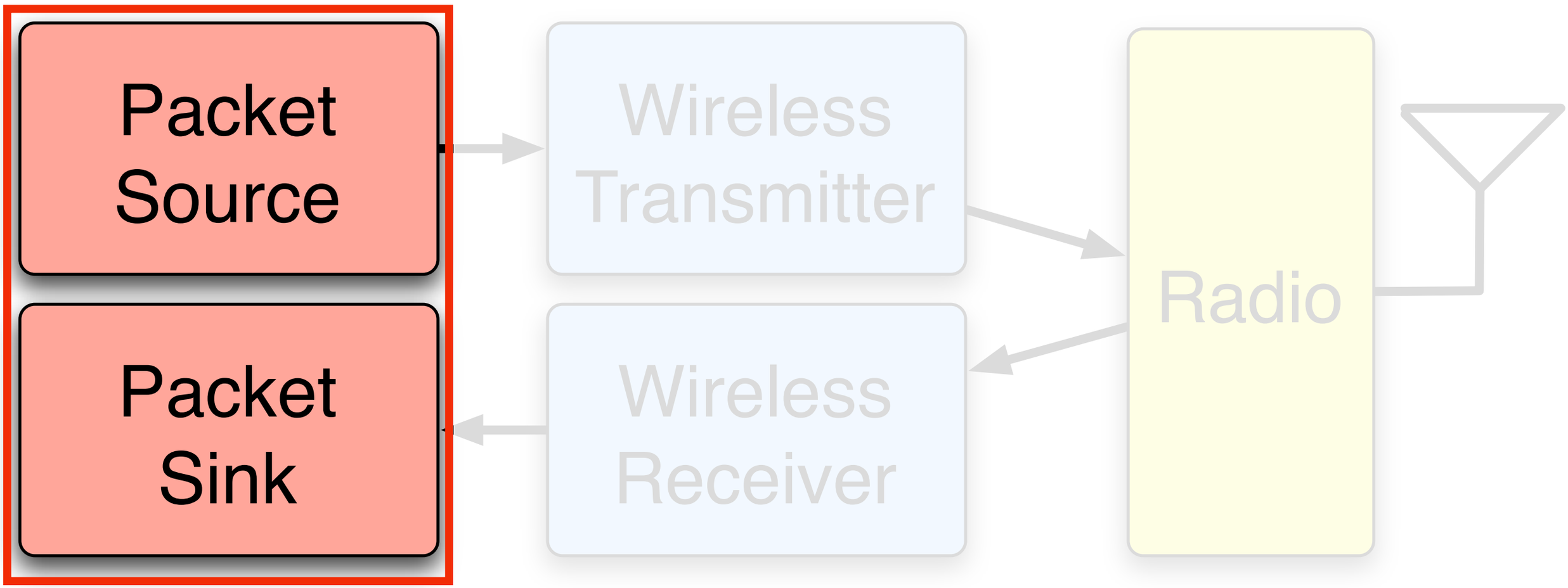
Network Layer Basics

Simple Wireless Node



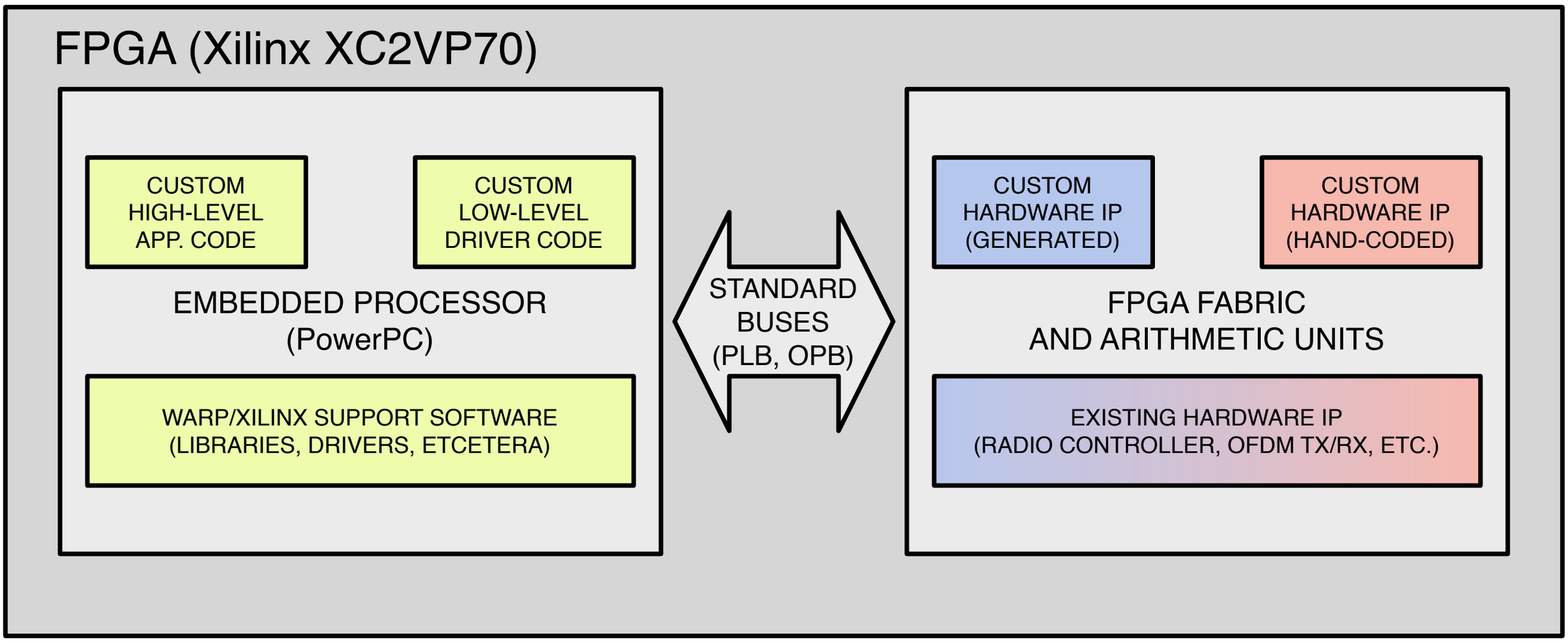
Network Layer Basics

Simple Wireless Node

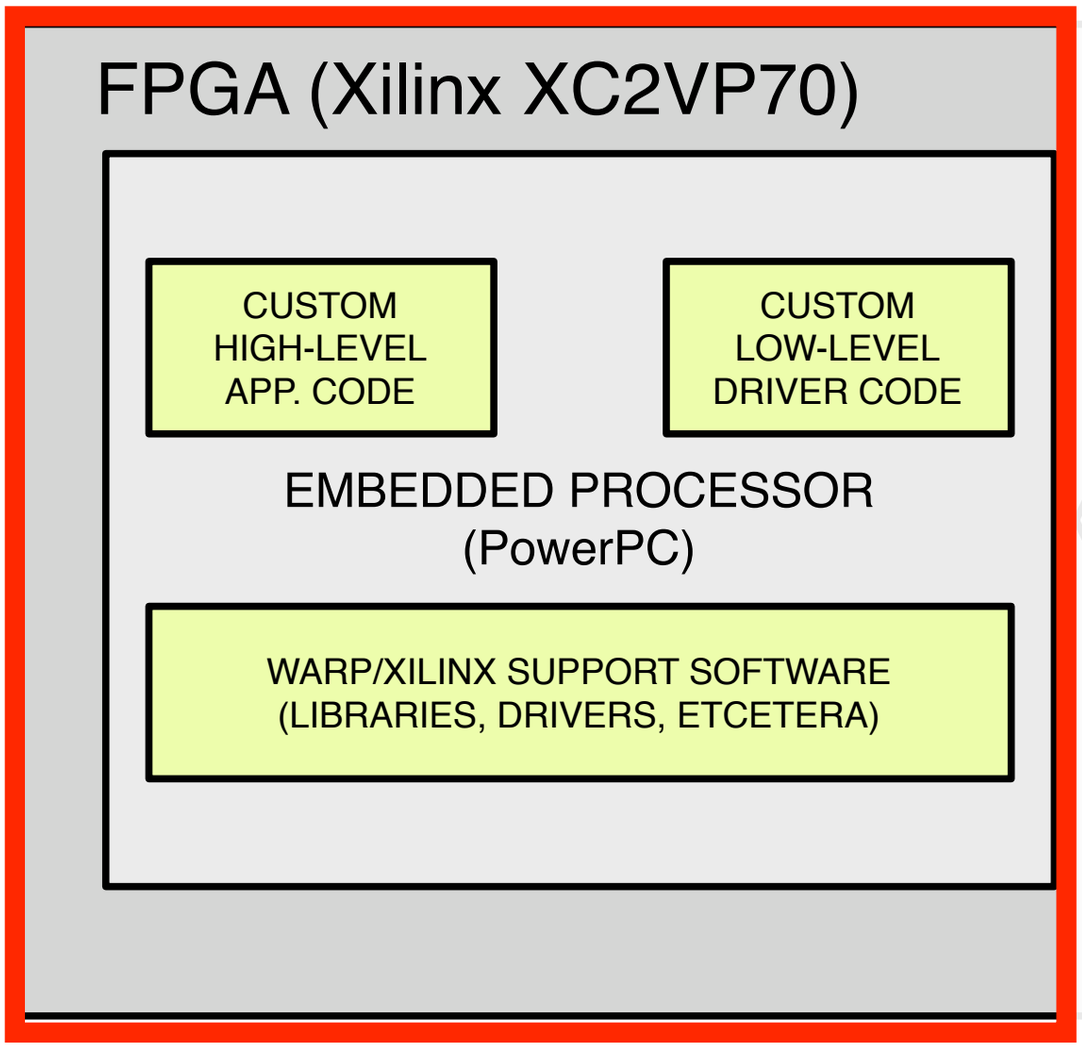


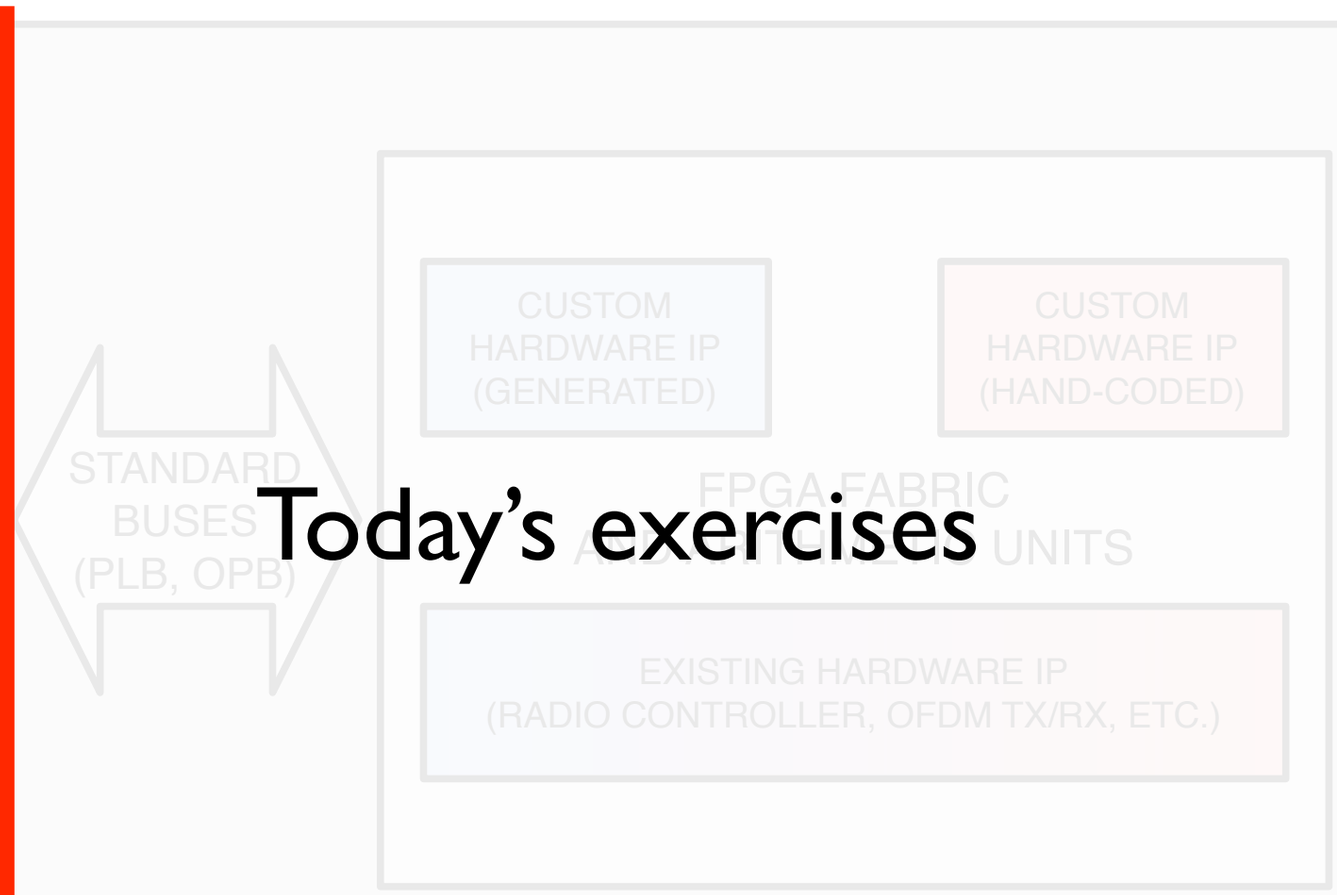
Targeting WARP Hardware

(Understanding the Development Environment)



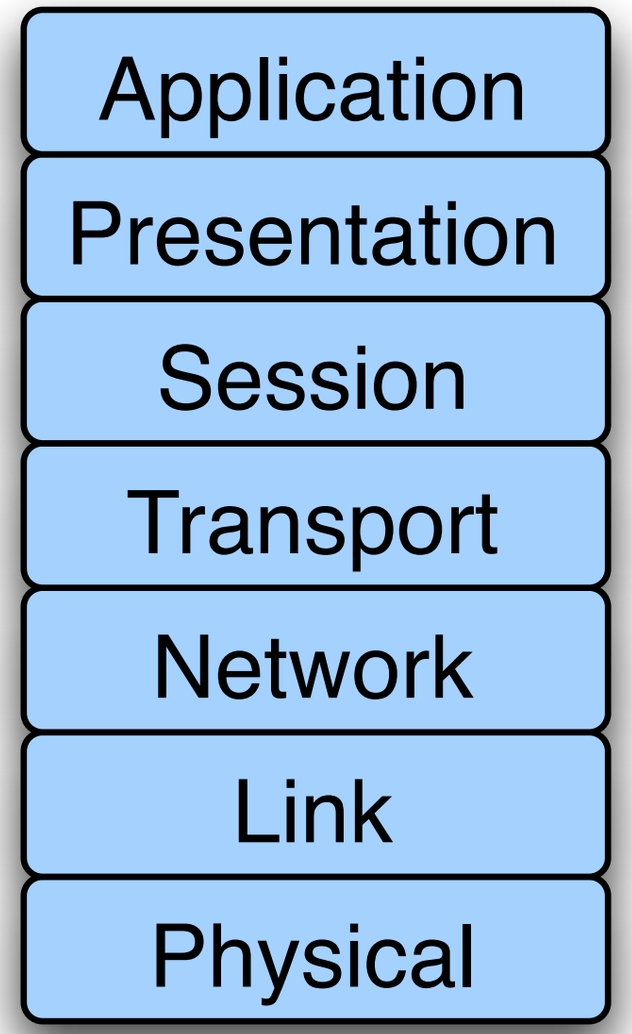
Targeting WARP Hardware
(Understanding the Development Environment)



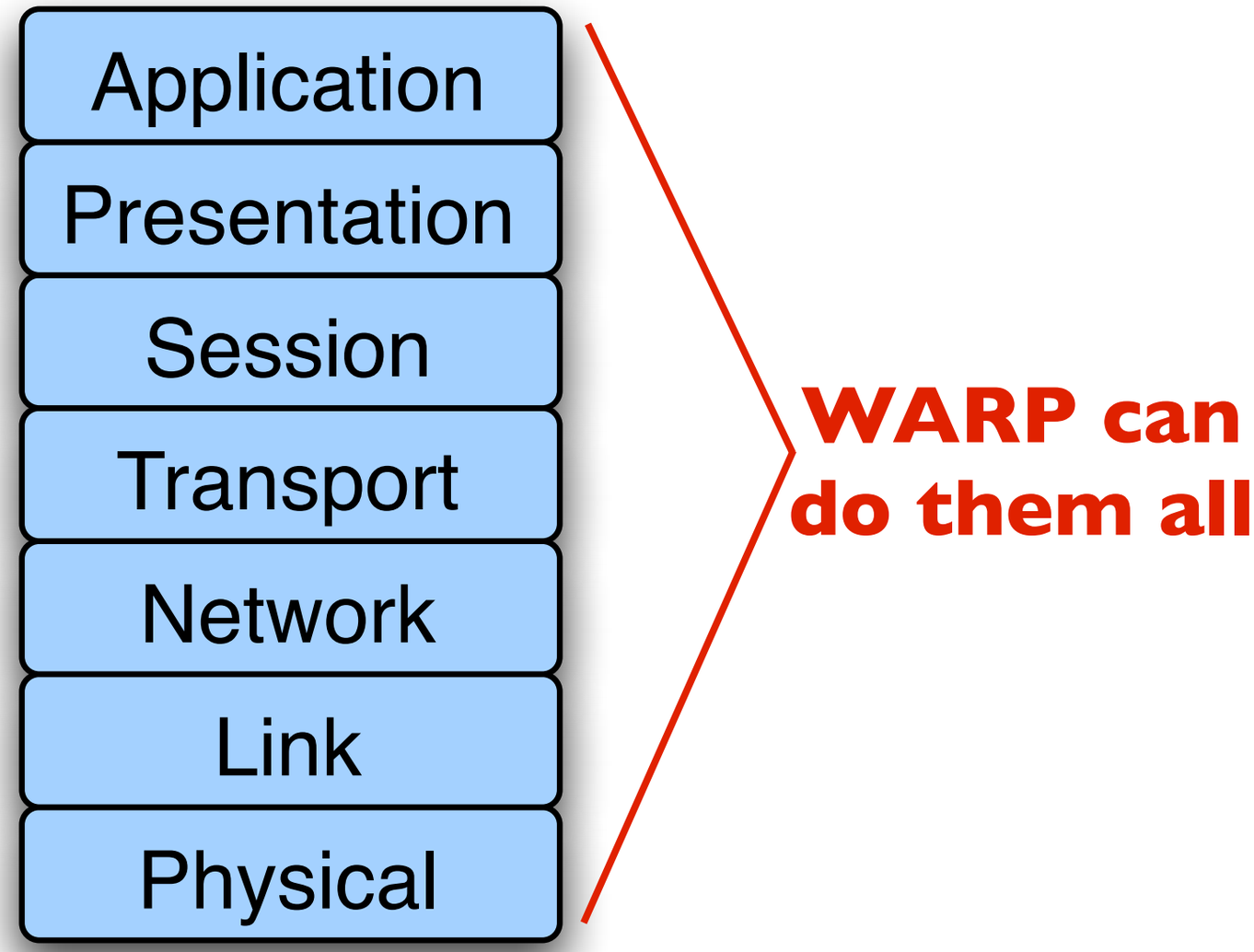


Today's exercises

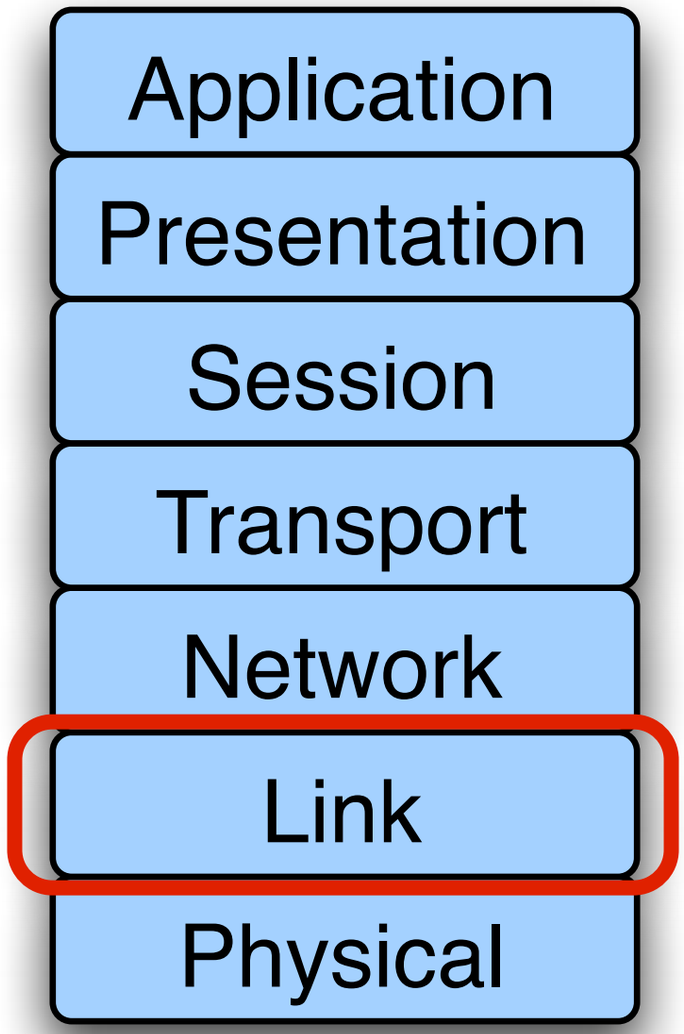
The OSI Model



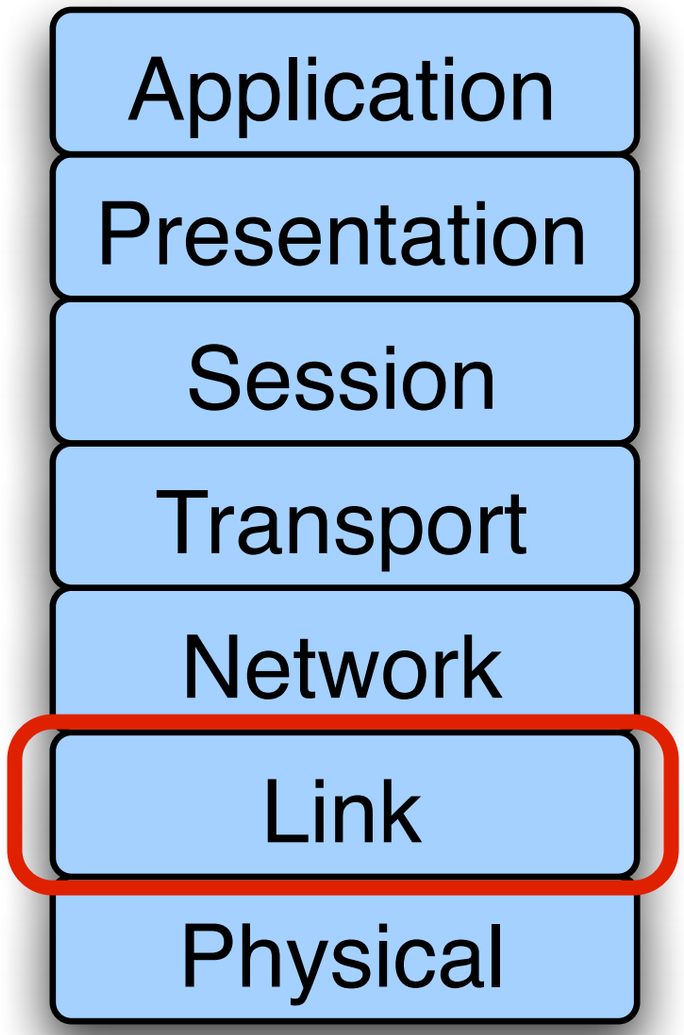
The OSI Model



The OSI Model

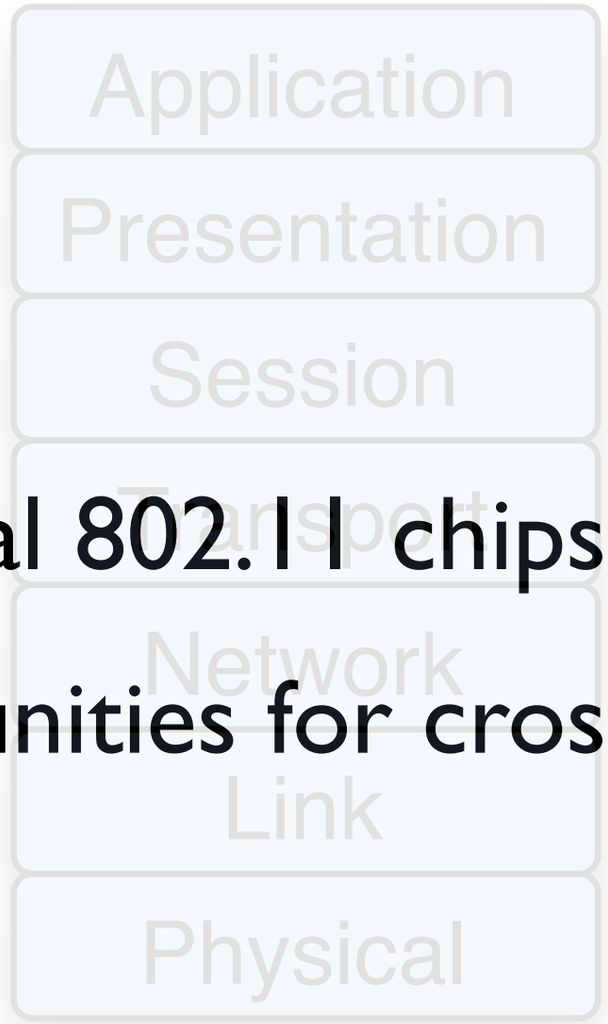


The OSI Model



Our Focus: Medium Access Control

The OSI Model



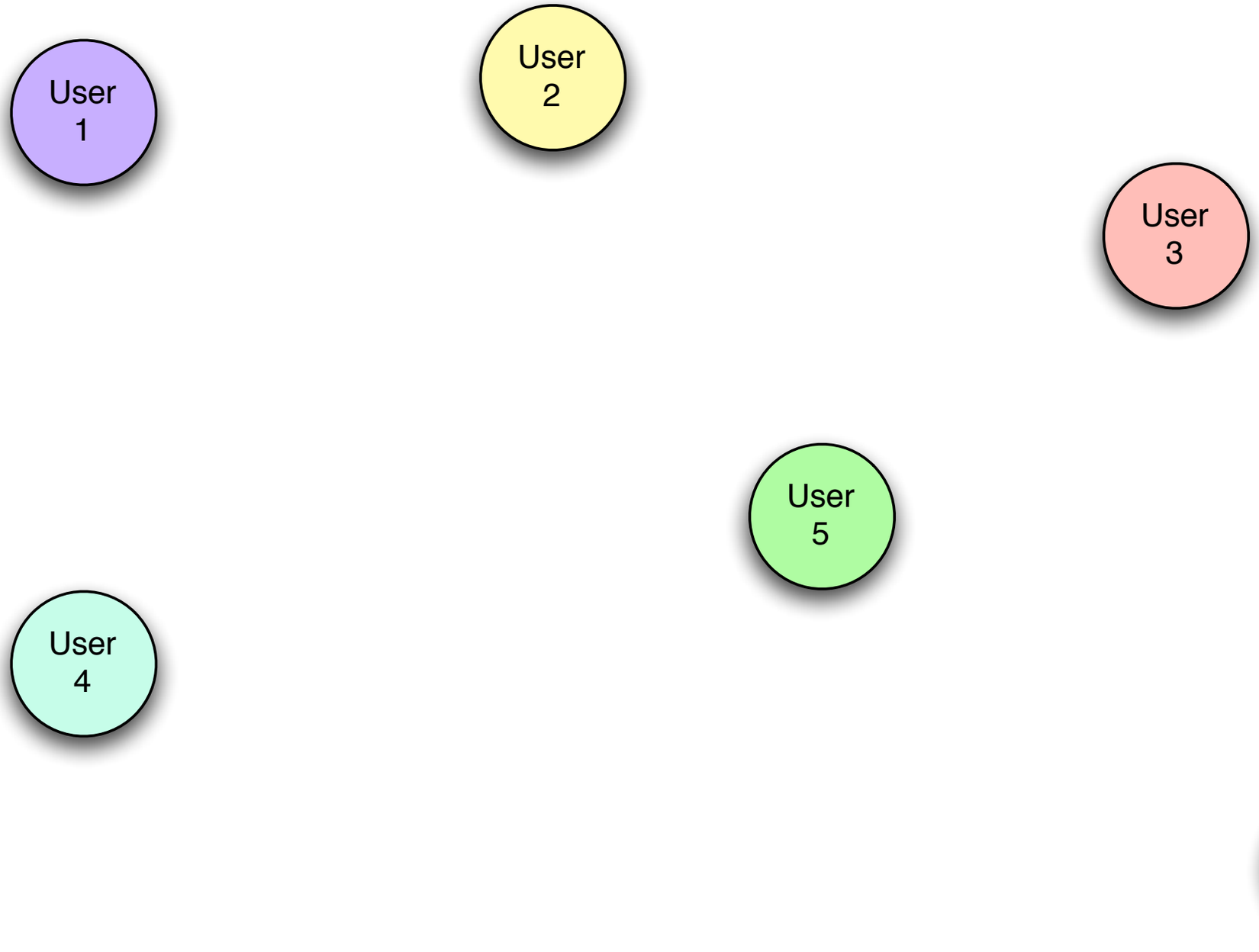
- Why?
- All commercial 802.11 chipsets are closed
- Many opportunities for cross-layer research

Outline

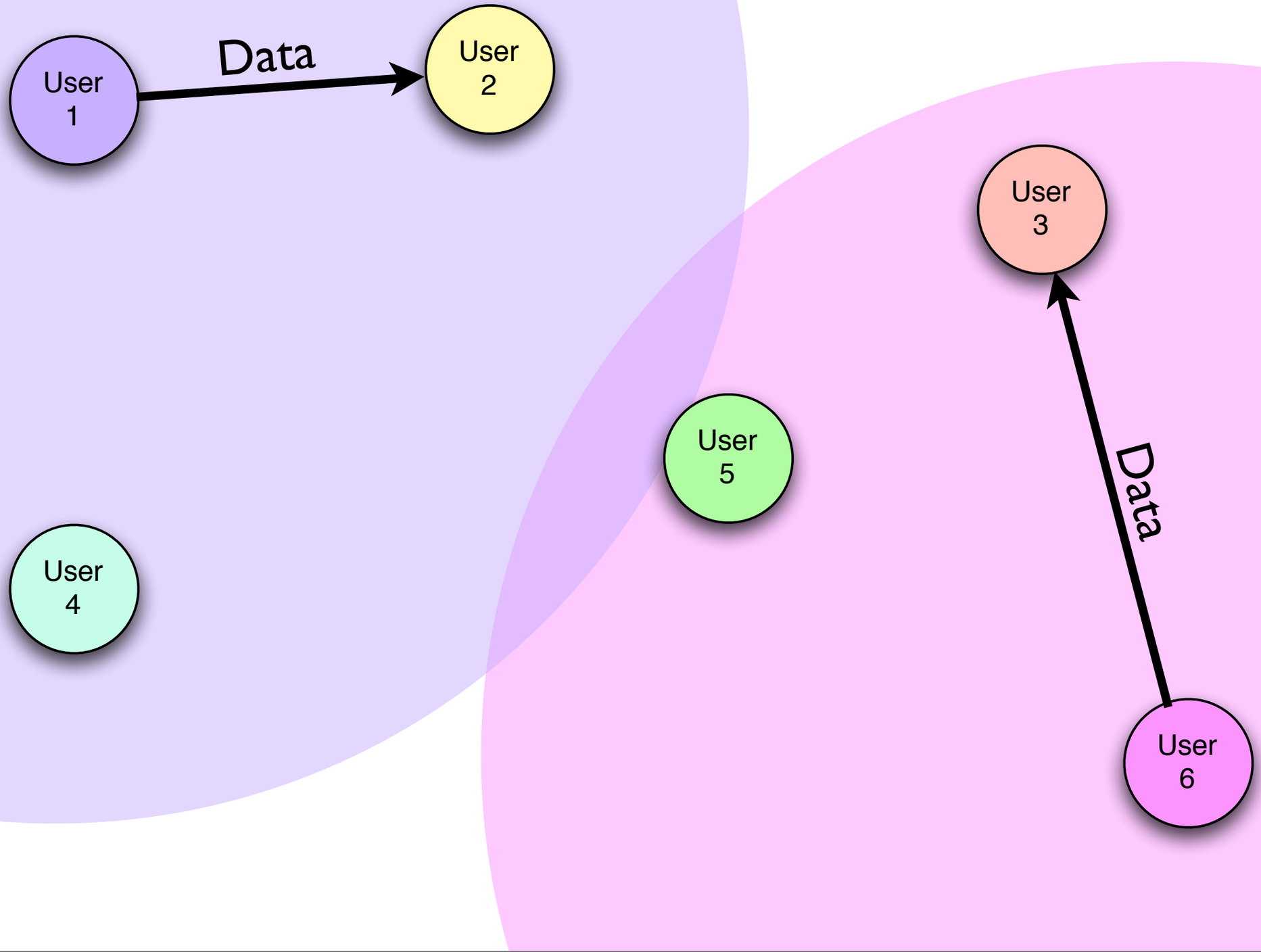
- Overview of Medium Access Control
- Design Realization
- Example
- Lab Exercises

Medium Access Control Overview

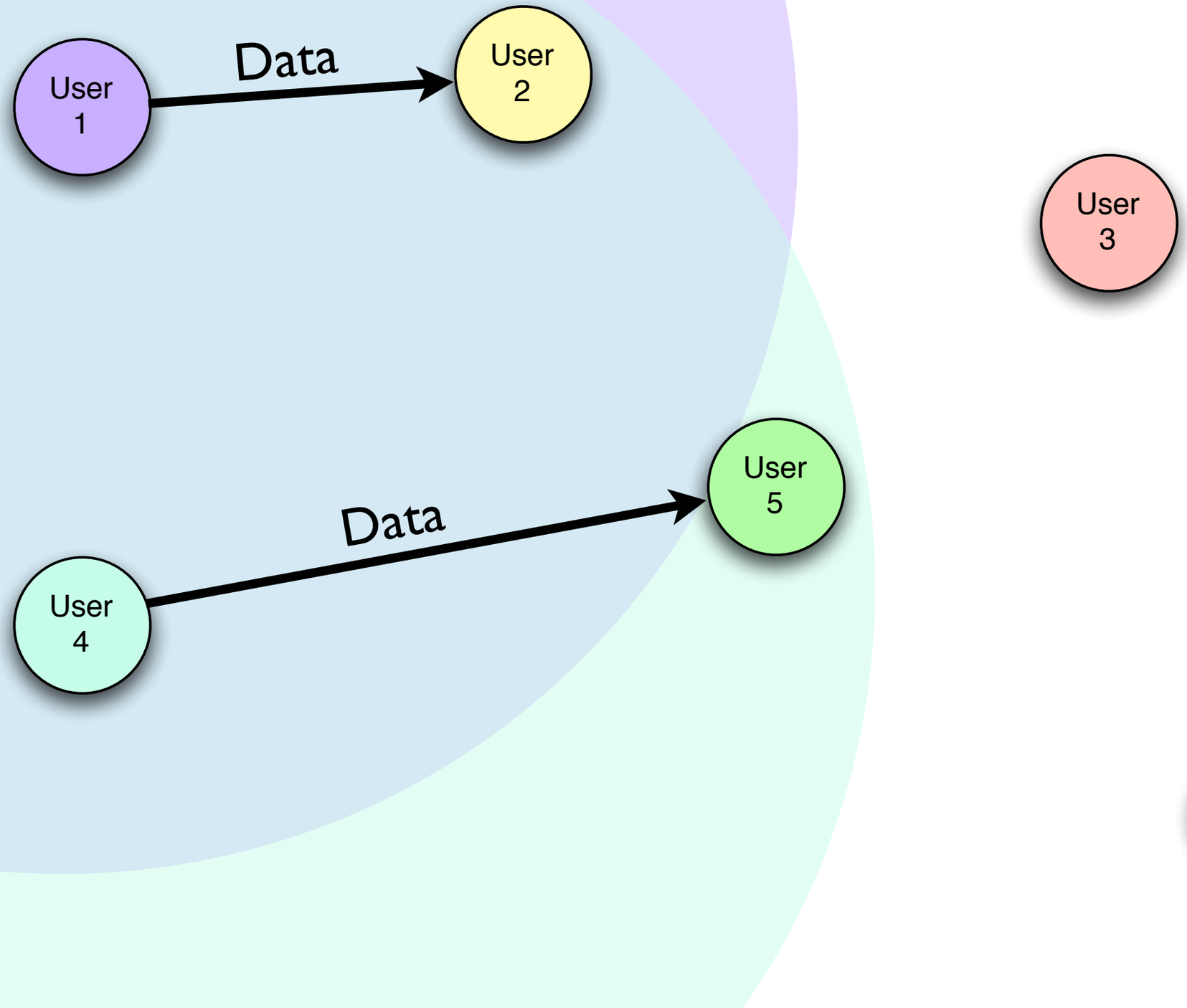
What is a MAC?



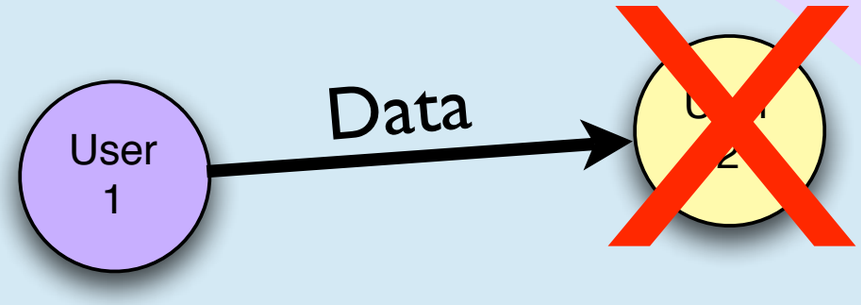
What is a MAC?

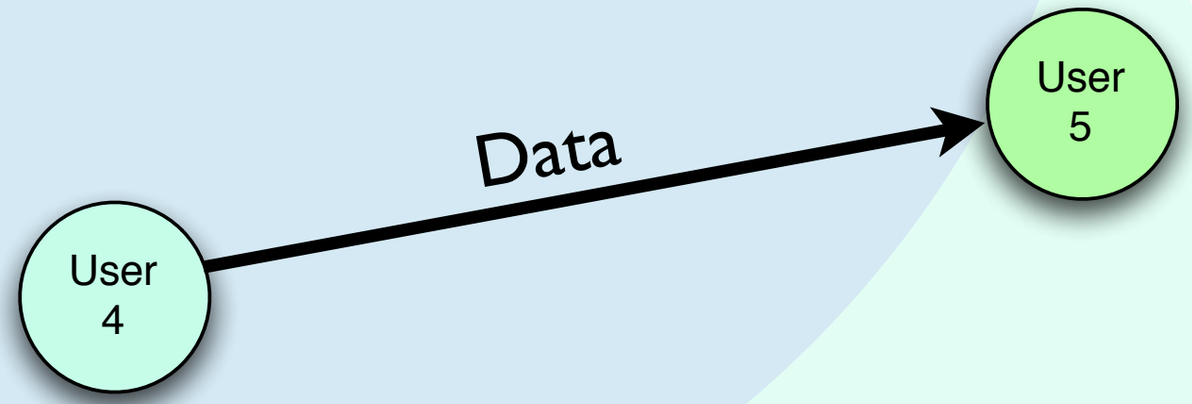


What is a MAC?

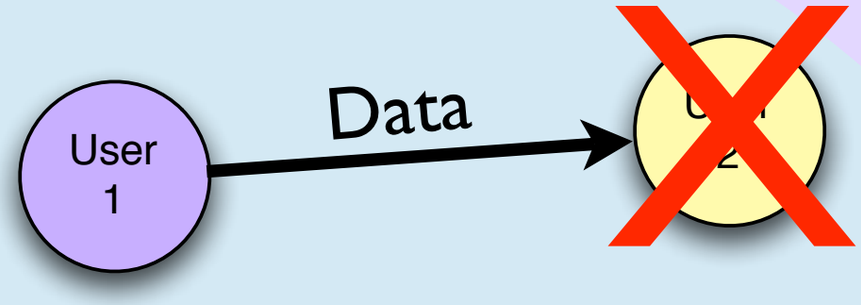


What is a MAC?

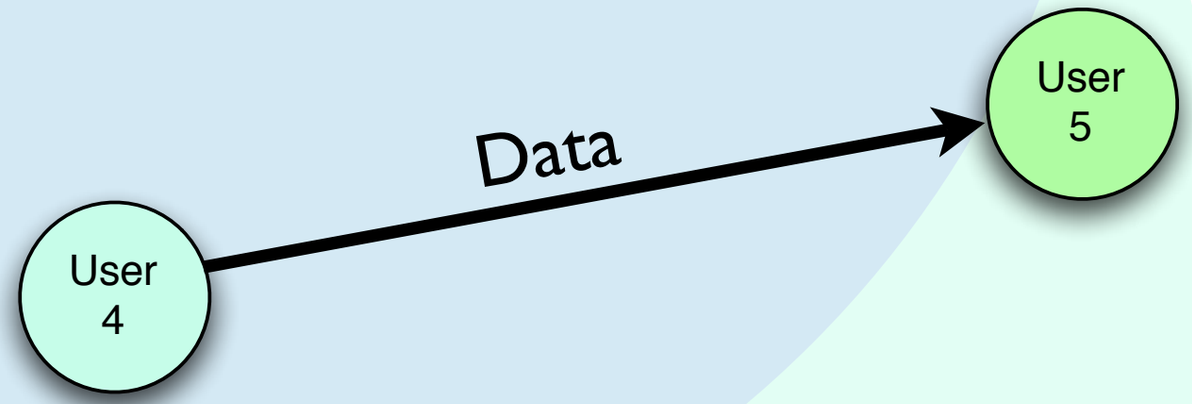




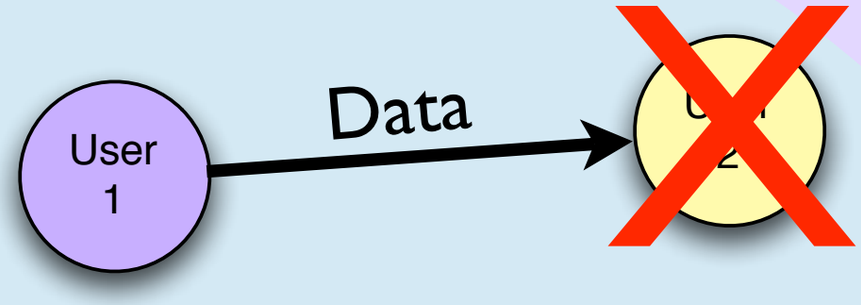
What is a MAC?



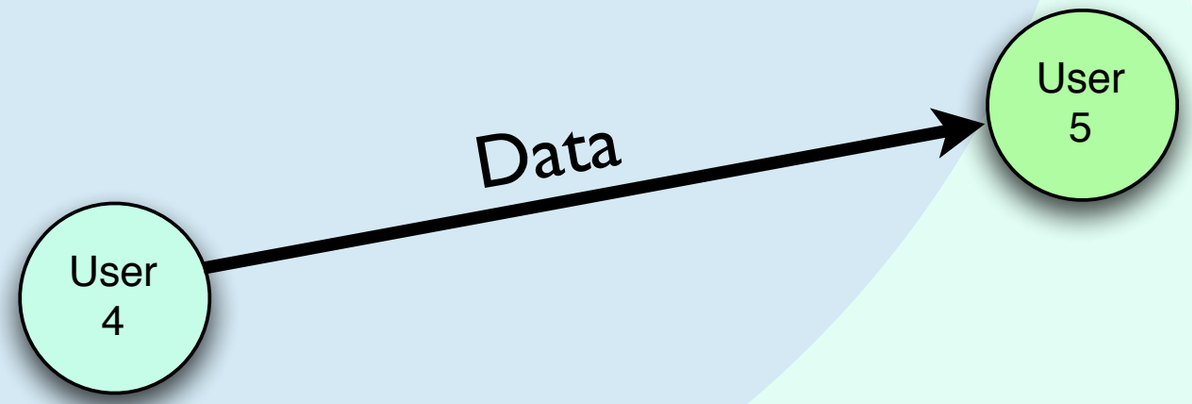
Received a jumbled packet... infer a packet collision



What is a MAC?

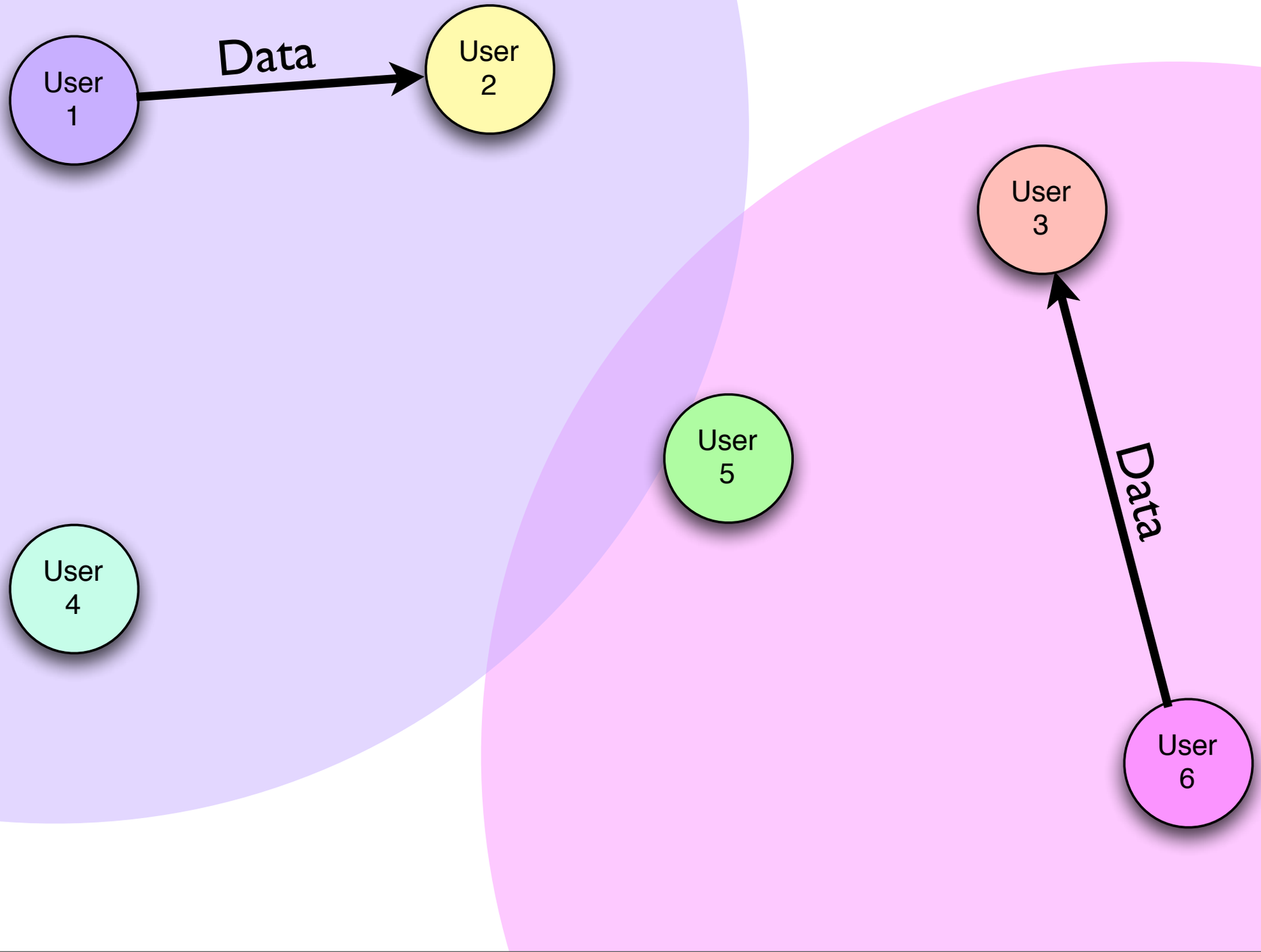


Received a jumbled packet... infer a packet collision

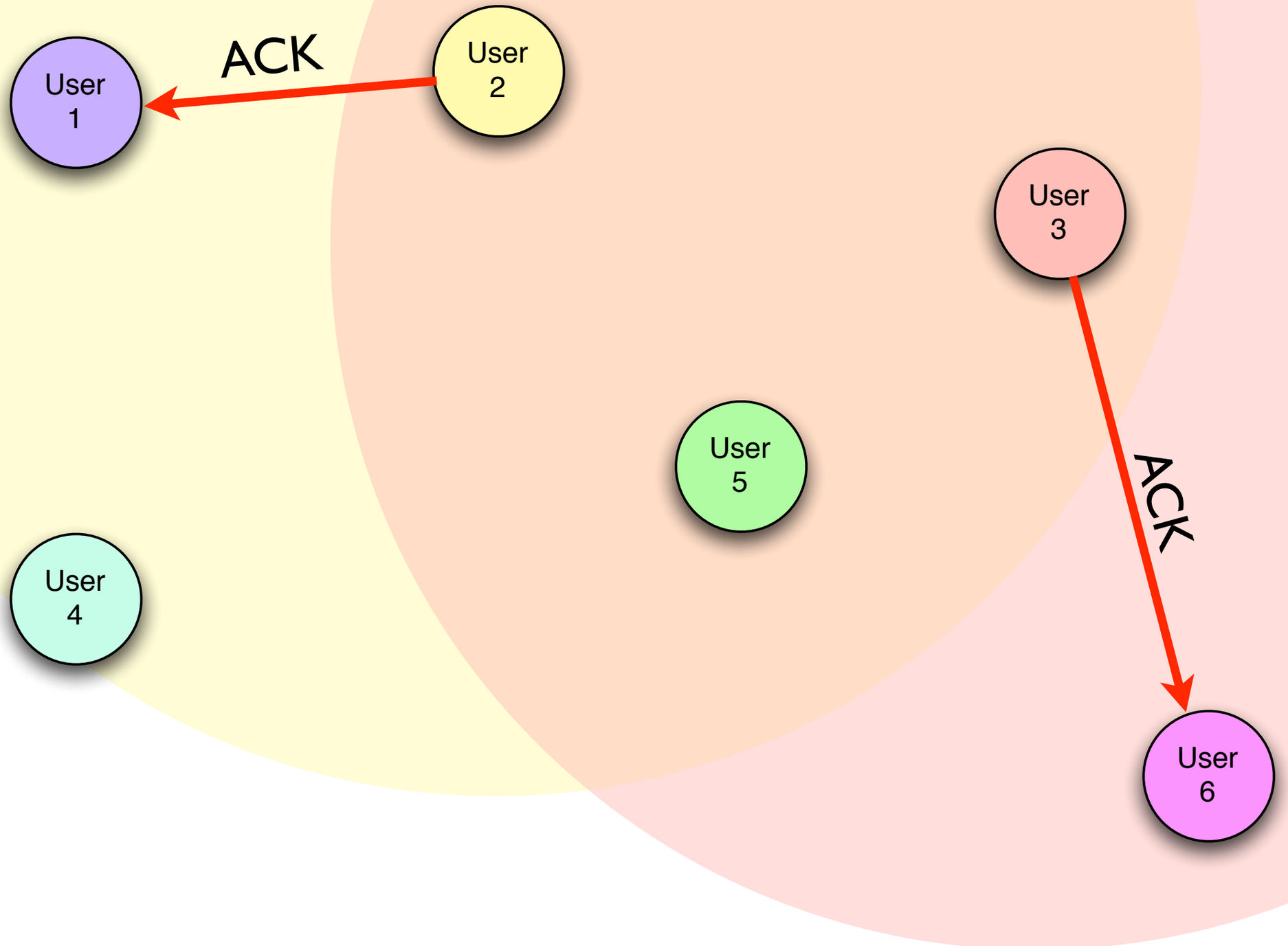


What if we ACK every transmit, and retransmit when we receive no ACK?

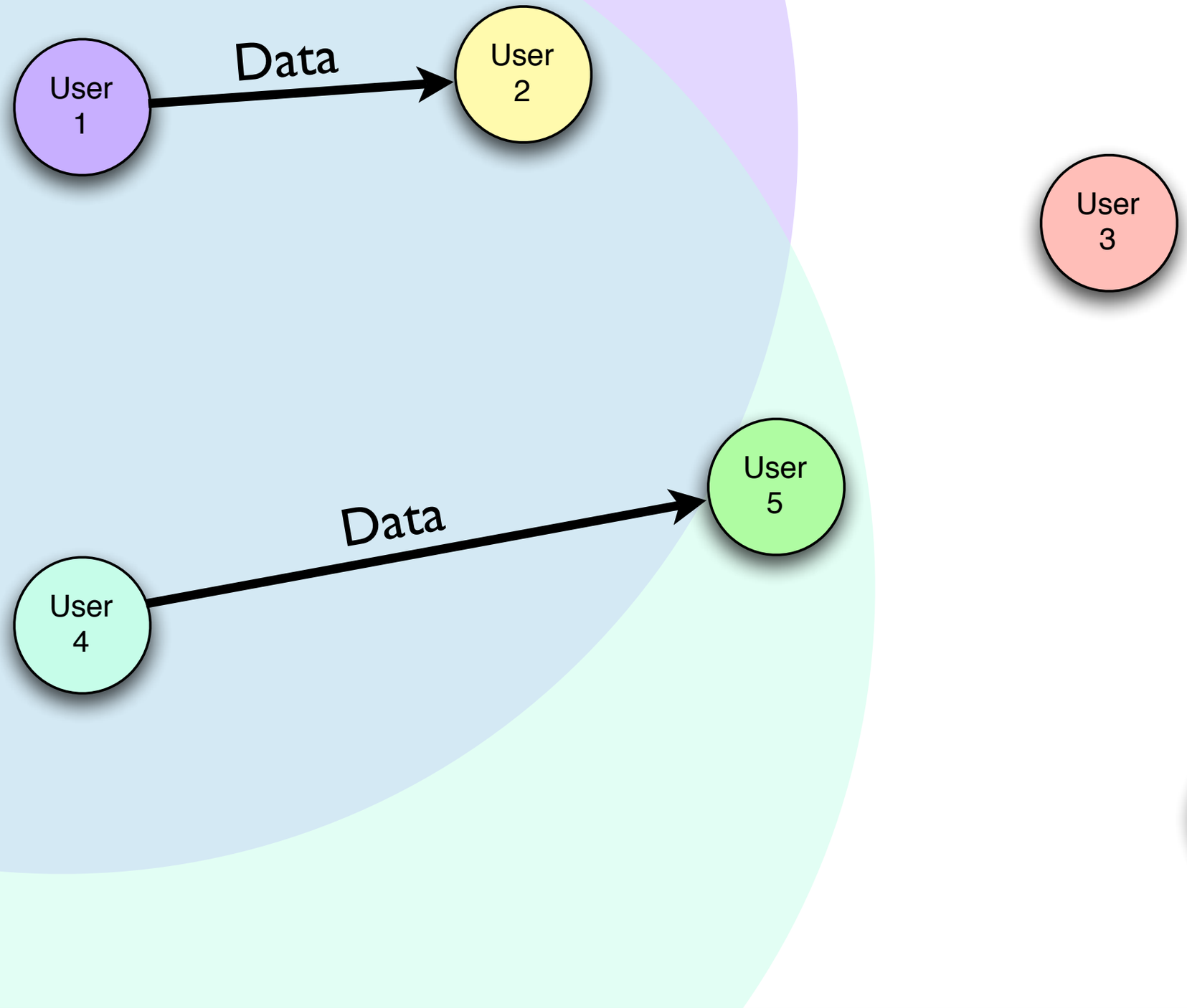
What is a MAC?



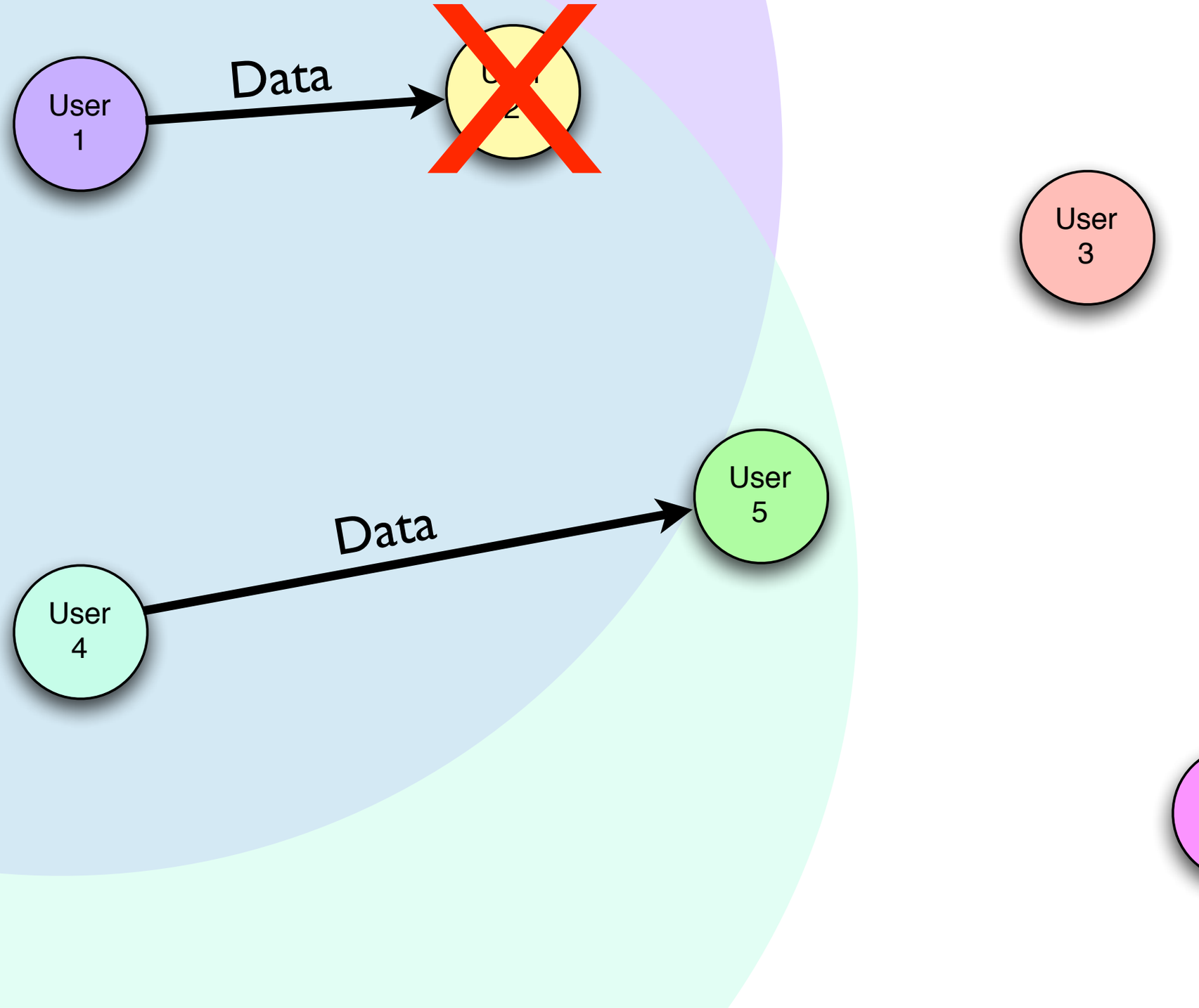
What is a MAC?



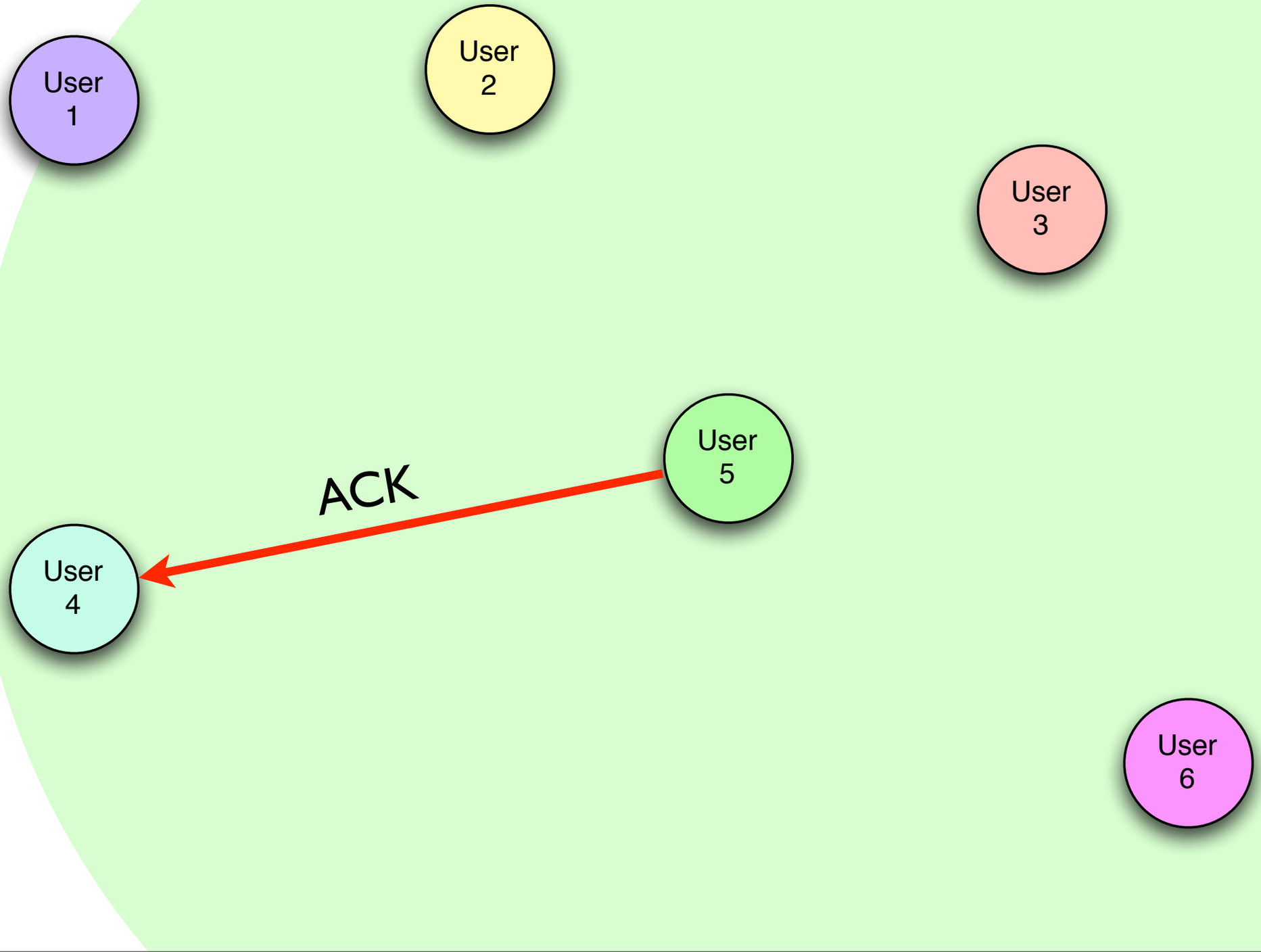
What is a MAC?



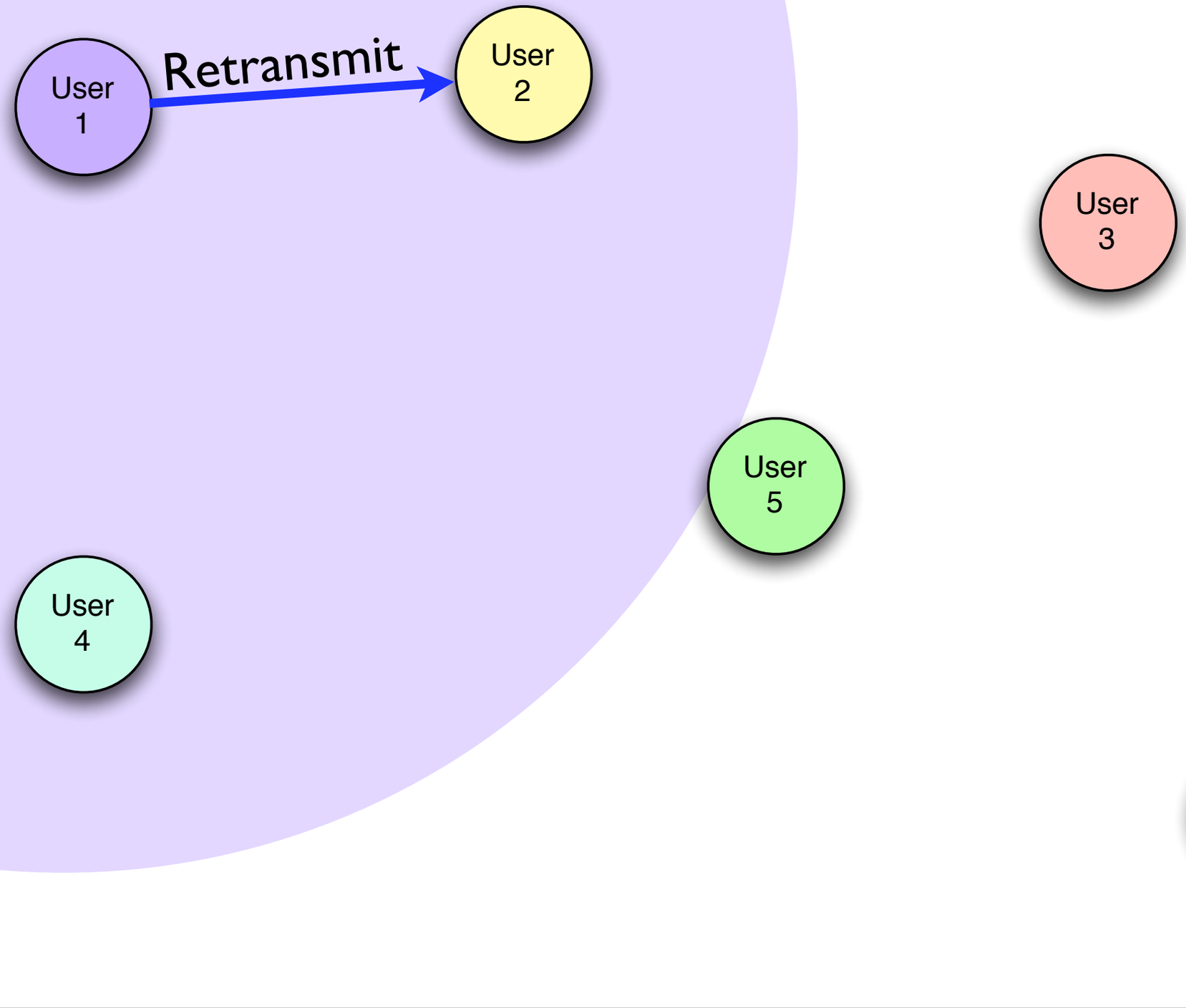
What is a MAC?



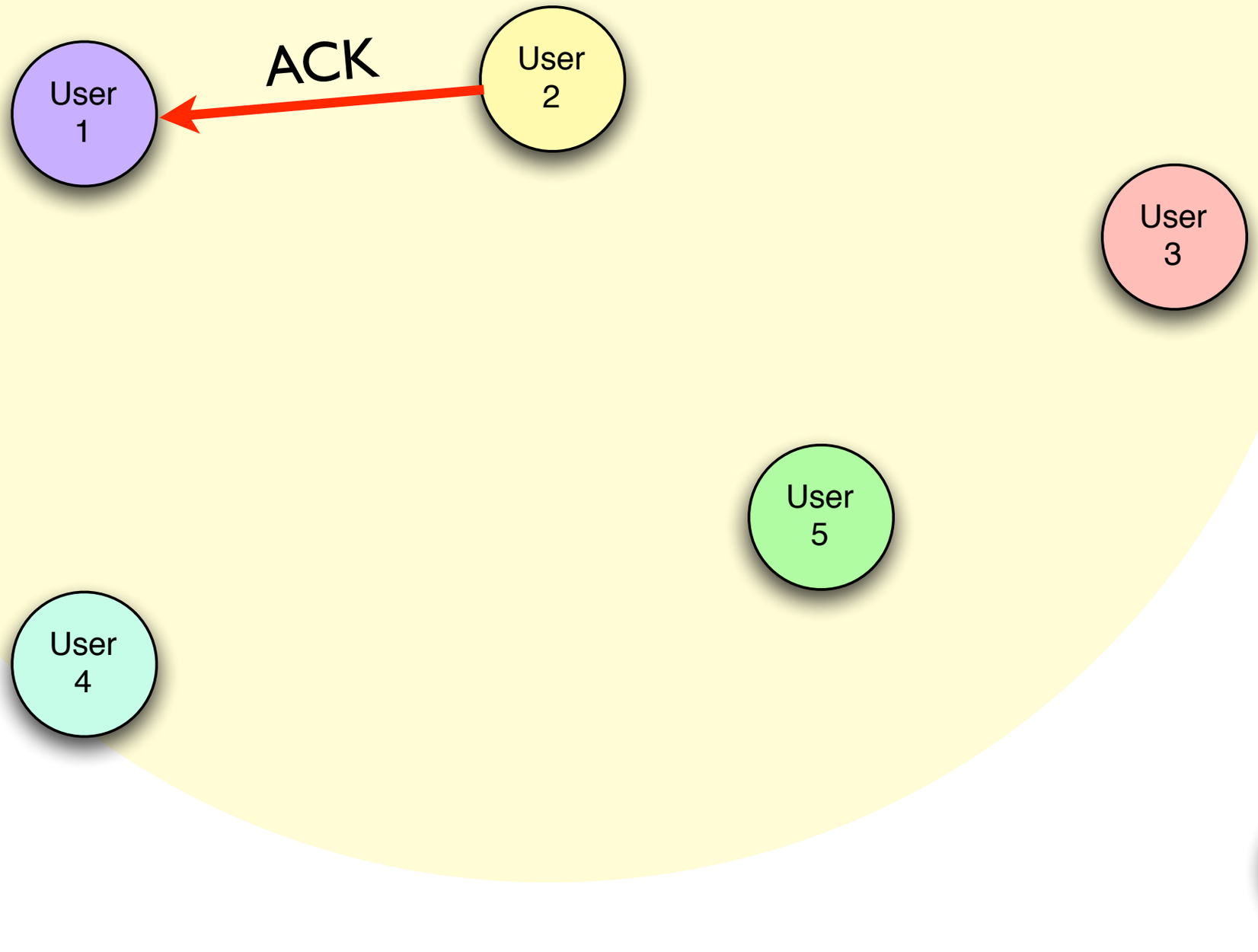
What is a MAC?



What is a MAC?



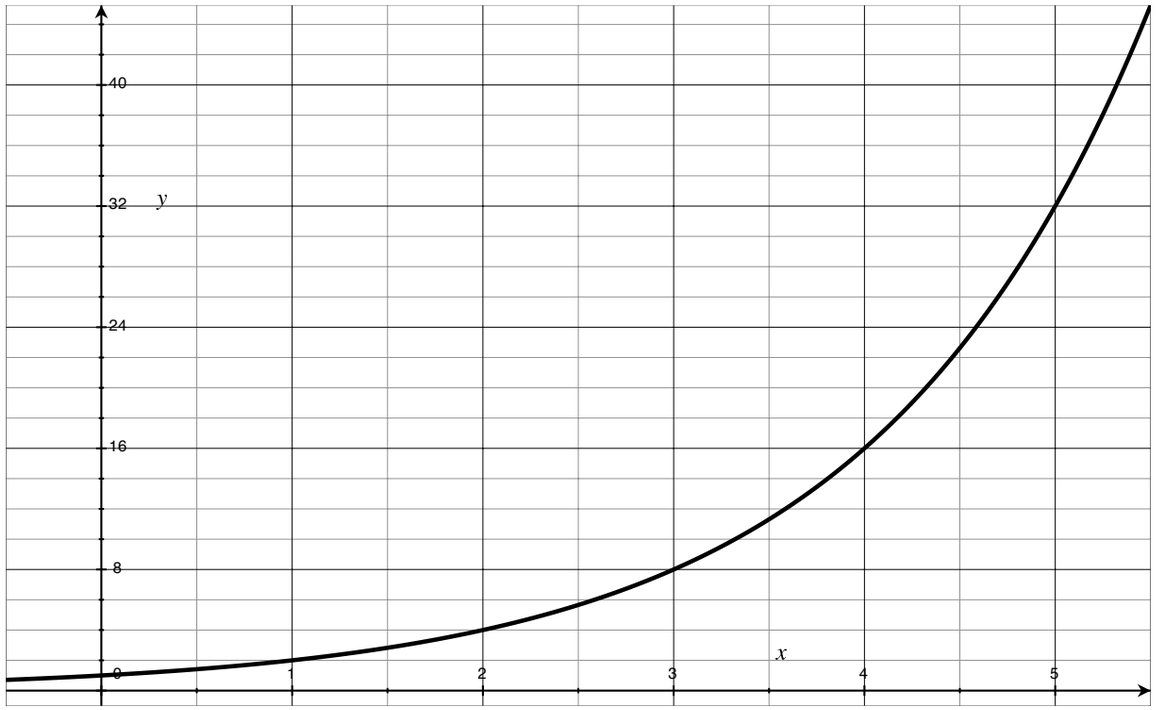
What is a MAC?



Random Backoffs

- **PROBLEM:** Retransmissions can collide *ad infinitum!*

- **SOLUTION:** Wait a random amount of time before a retransmit



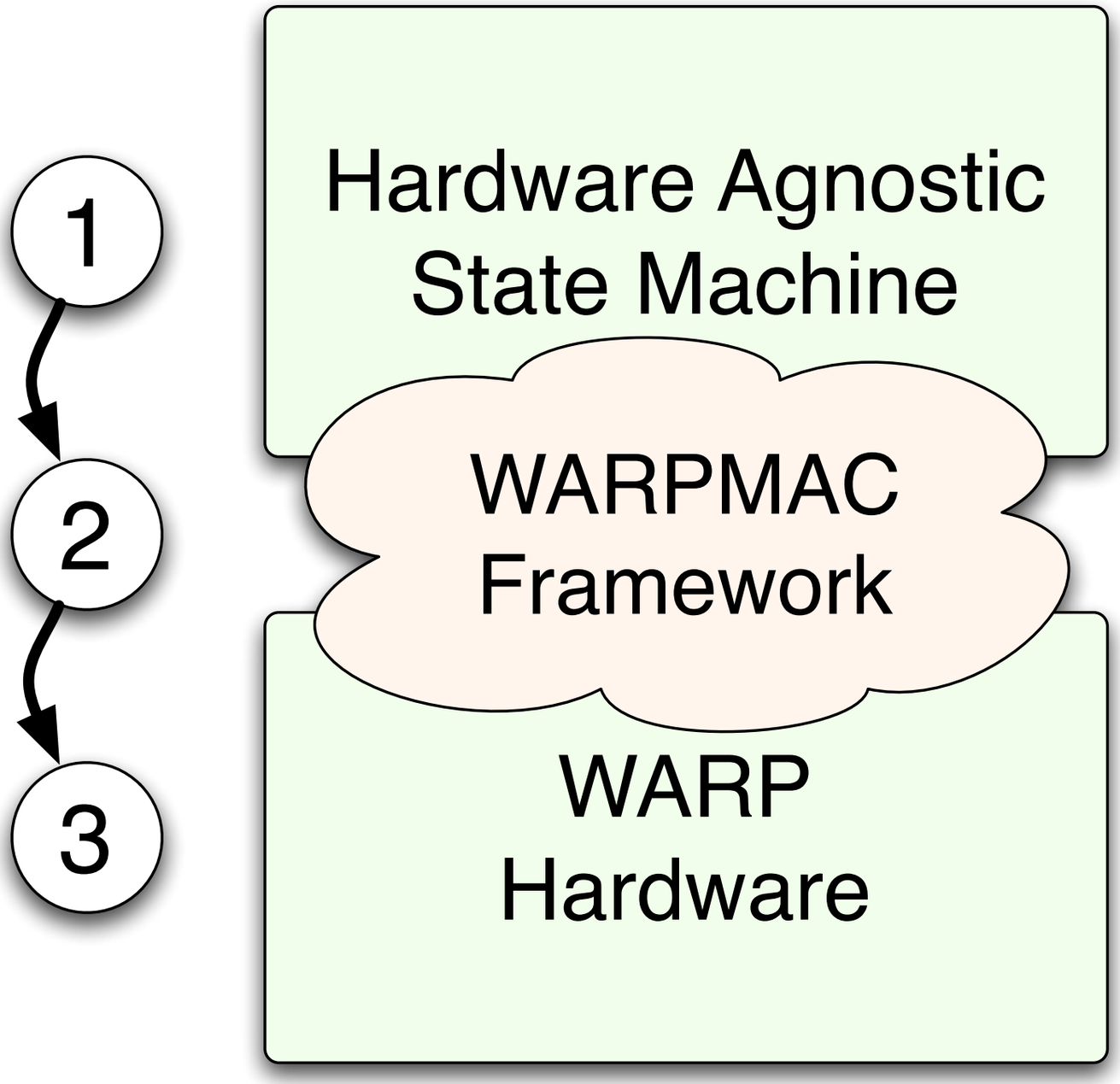
Contention Window increases over time

Important Extensions

- Carrier Sense Multiple Access (CSMA)
  - Listen to the medium before sending
- Request to Send / Clear to Send (RTS/CTS)
  - “Reserve” the medium with a short packet before sending a long one

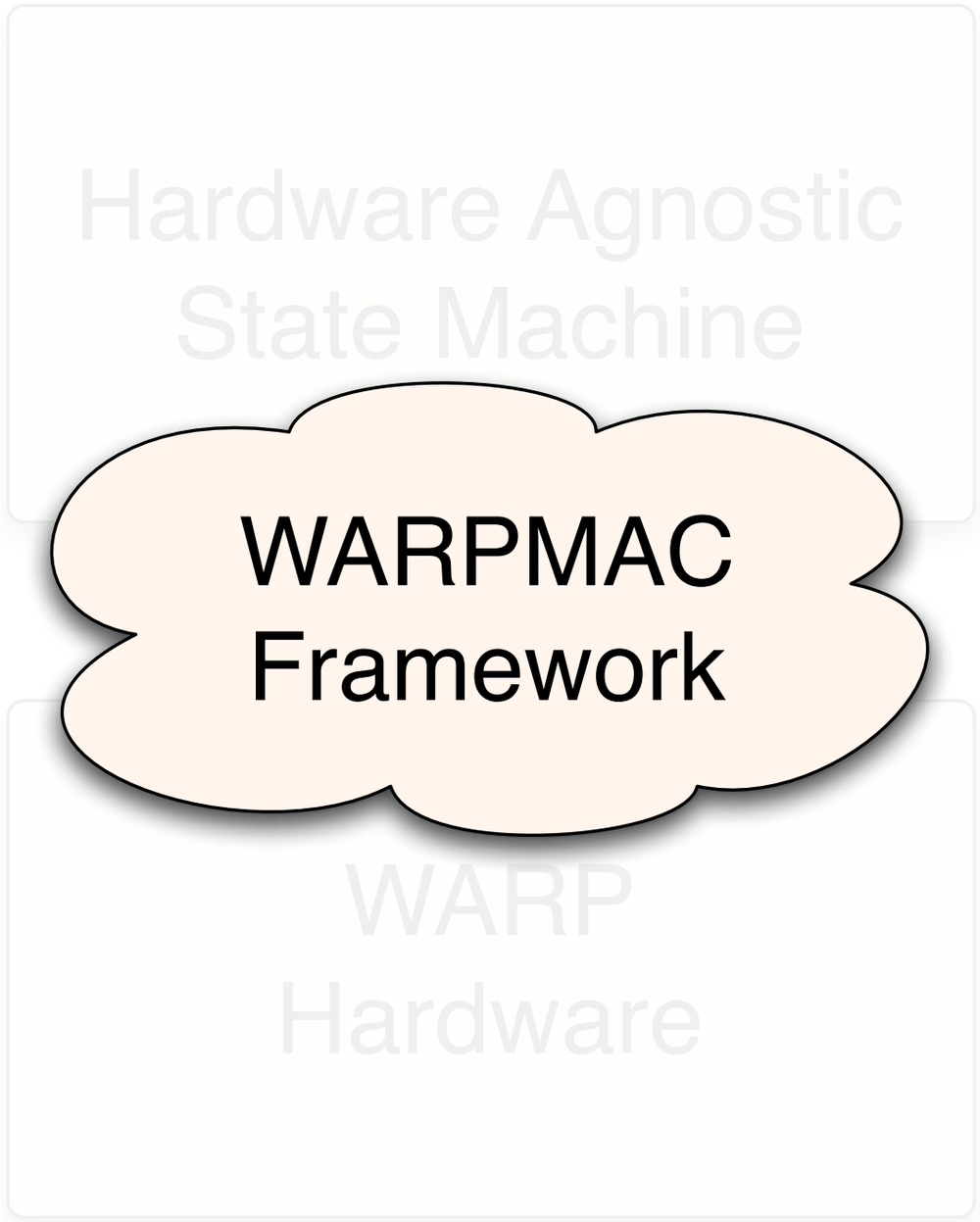
Design Realization

Design Realization



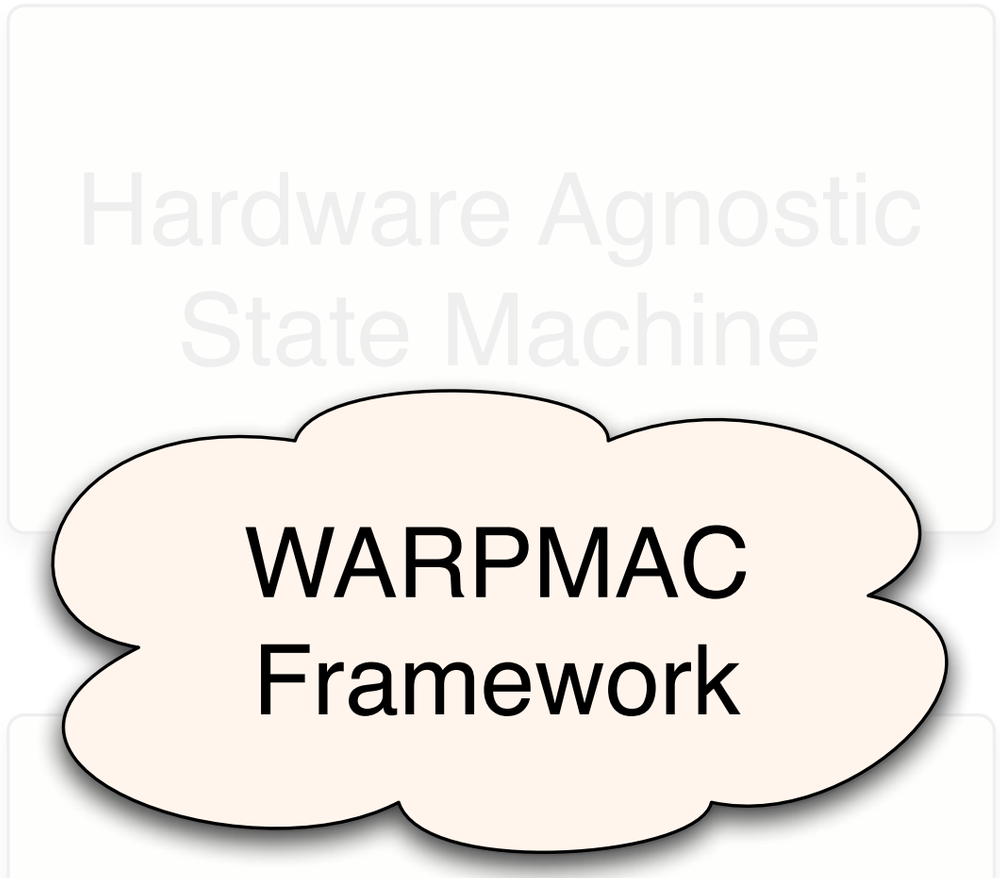
- Program high-level MAC behavior independent of hardware
- Use the WARPMAC framework to stitch the MAC to hardware

Design Realization



- No way to “lock” the framework and have it support all possible future MAC layers

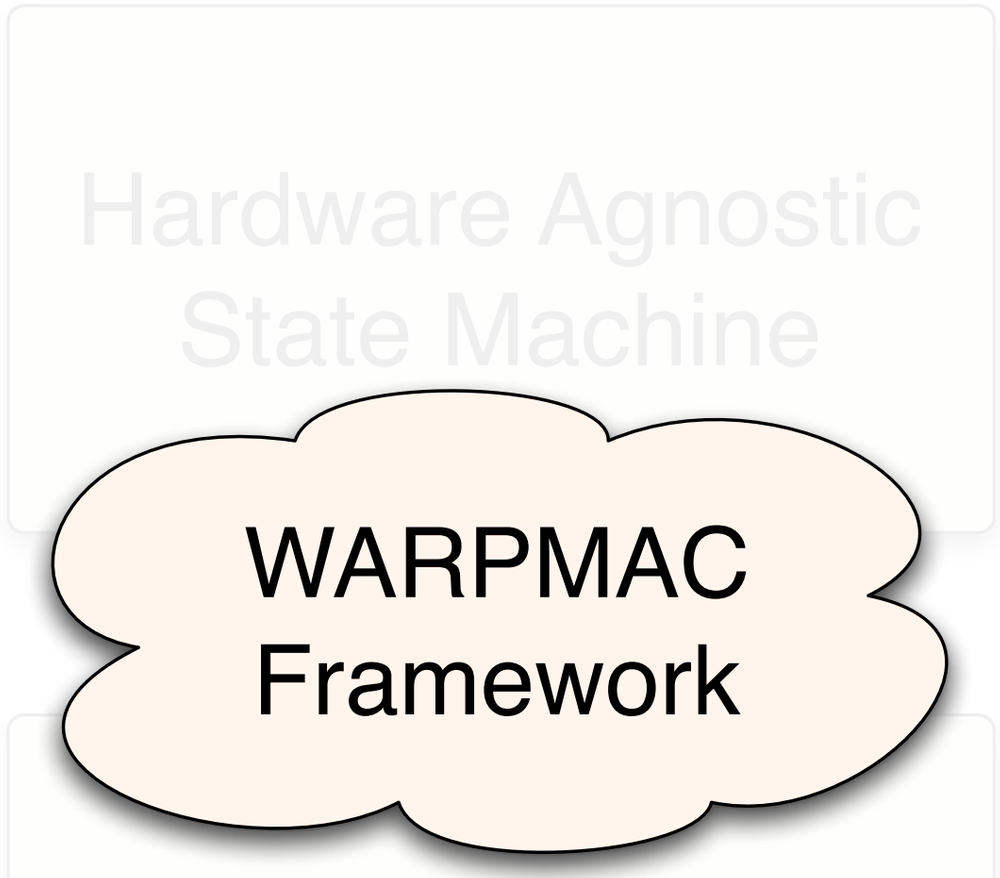
Design Realization



- No way to "lock" the framework and have it support all possible future MAC layers

Solution: *WARPMAC must grow with new algorithms*

Design Realization



- No way to "lock" the framework and have it support all possible future MAC layers

Solution: *WARPMAC must grow with new algorithms*

Problem: *How do we maintain sync between designs?*

Reference Designs

Reference Designs

Reference Designs

- Snapshots of the WARP repository

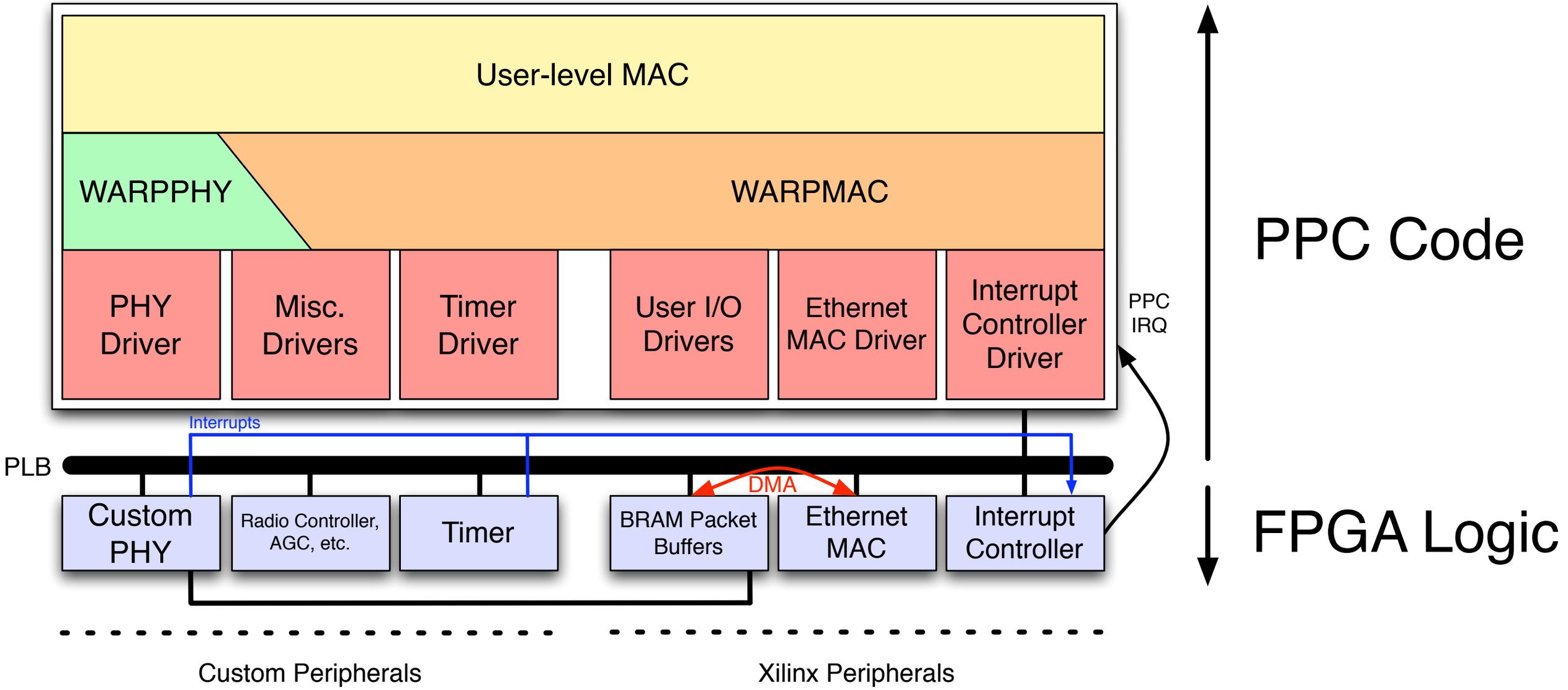
Reference Designs

- Snapshots of the WARP repository
- Free, open-source releases at regular intervals
  - Today's exercises are on nearly-released Reference Design v12
  - Keeps pace with Xilinx design tools

Reference Designs

- Snapshots of the WARP repository
- Free, open-source releases at regular intervals
  - Today's exercises are on nearly-released Reference Design v12
  - Keeps pace with Xilinx design tools
- Reference design is an example of:
  - a working PHY
  - a working MAC
  - the way all the pieces fit together

Reference Designs




User Code
WARPMAC
WARPPHY
Drivers


User Code
WARPMAC
WARPPHY
Drivers


User Code
WARPMAC
WARPPHY
Drivers

PHY Driver:

- Configure very low-level parameters
  - Correlation thresholds
  - FFT scaling parameters
  - Filter coefficients
  - Etc.


User Code
WARPMAC
WARPPHY
Drivers


User Code
WARPMAC
WARPPHY
Drivers

Radio Controller Driver:

- Set center frequency
- Switch from Rx to Tx mode and vice versa


User Code
WARPMAC
WARPPHY
Drivers


User Code
WARPMAC
WARPPHY
Drivers

PHY Control:

- Provides control over PHY commonalities
  - General initialization command
  - Configure constellation order
  - “Start” and “Stop” the PHY


User Code
WARPMAC
WARPPHY
Drivers


User Code
WARPMAC
WARPPHY
Drivers

Mostly PHY agnostic

Completely PHY dependent


User Code
WARPMAC
WARPPHY
Drivers

MAC Control:

- Provides control over MAC commonalities
  - Timers for timeouts, backoffs, etc.
  - Carrier-sensing functions
  - Register user callbacks to ISRs
  - Etc.


User Code
WARPMAC
WARPPHY
Drivers


User Code
WARPMAC
WARPPHY
Drivers

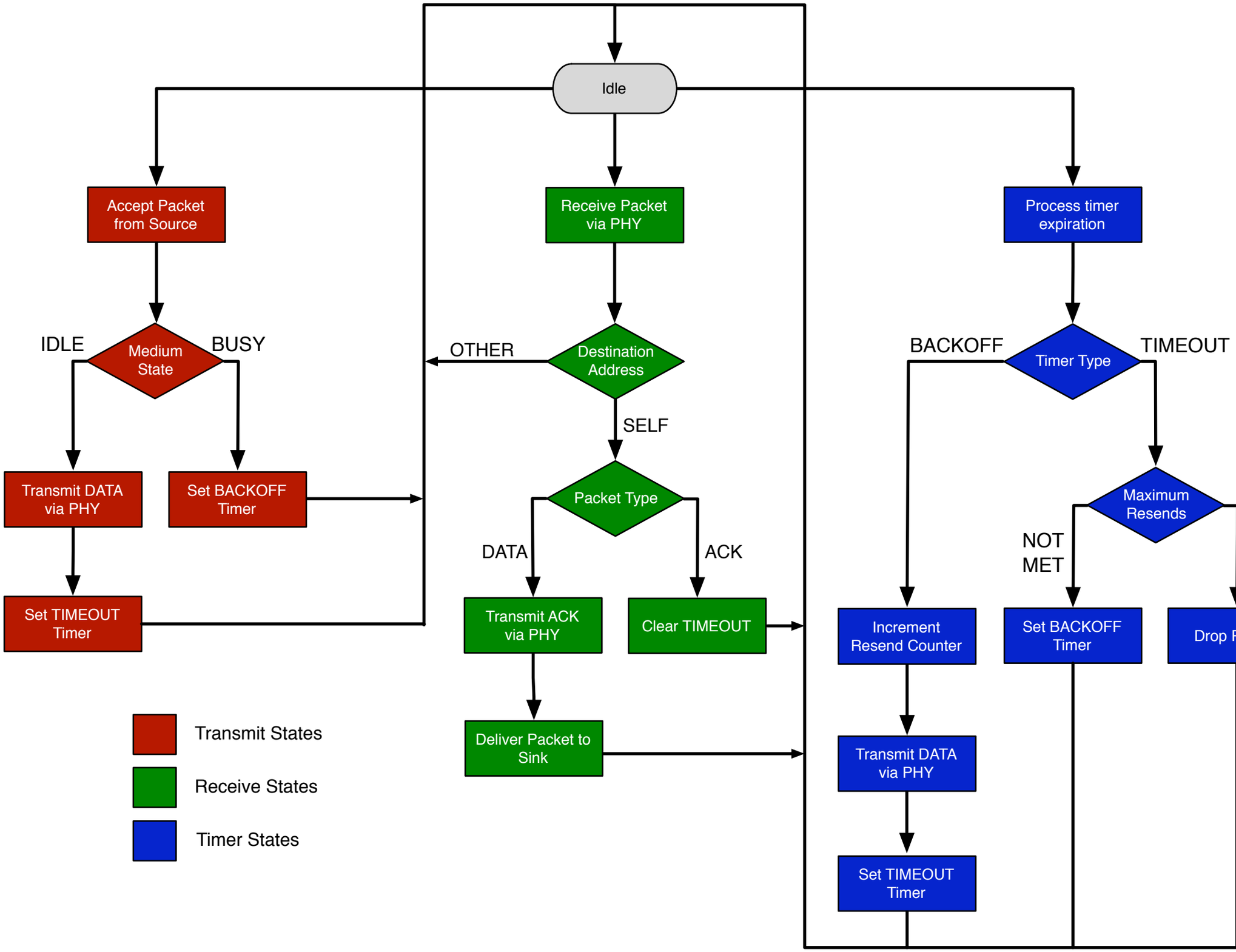
User-level MAC Algorithms:

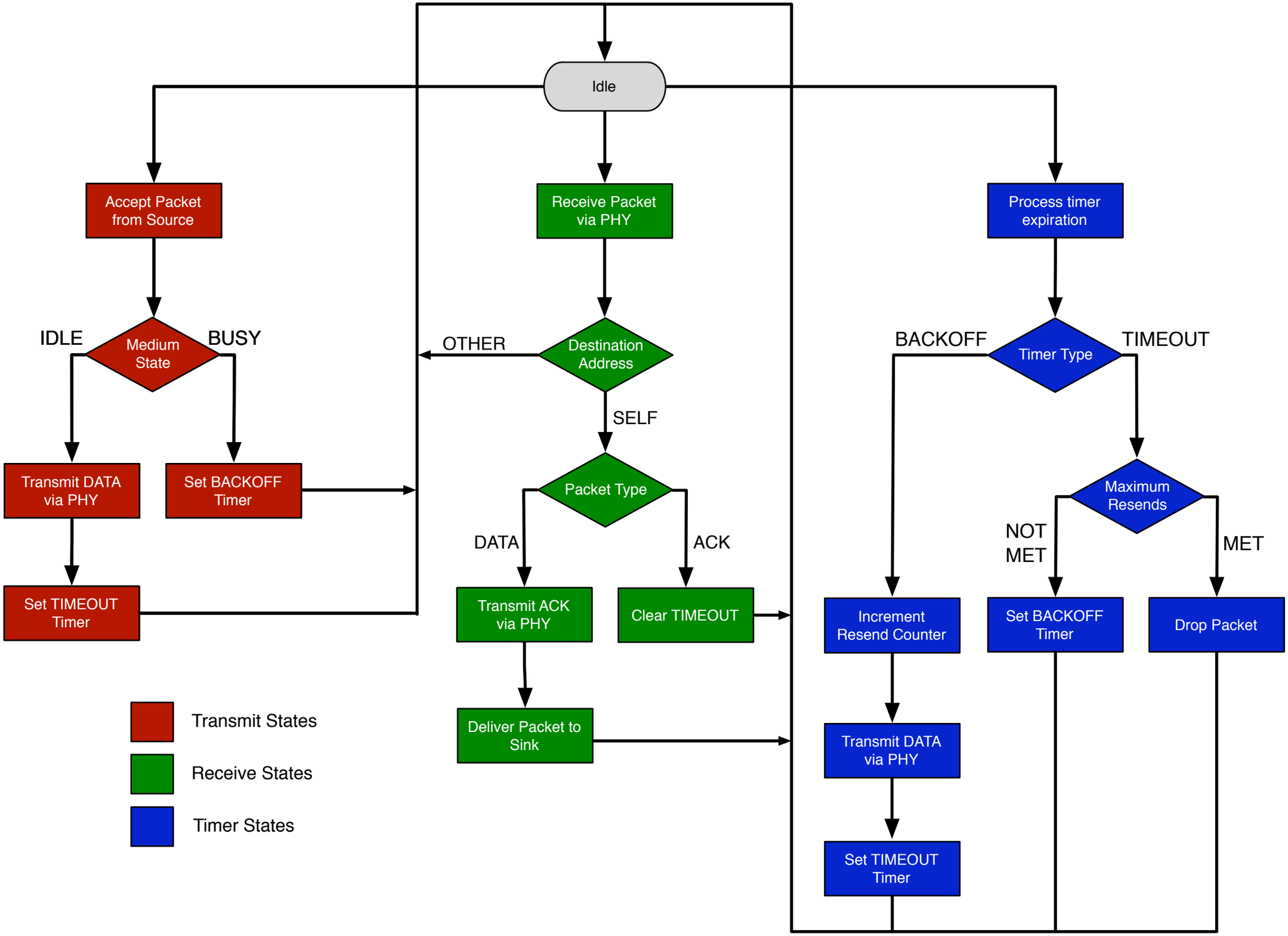
- High-level MAC algorithms
- Some examples so far:
  - Aloha
  - Carrier-sensing MAC
  - Opportunistic Auto-Rate (OAR)
  - MAC Workshop Exercises

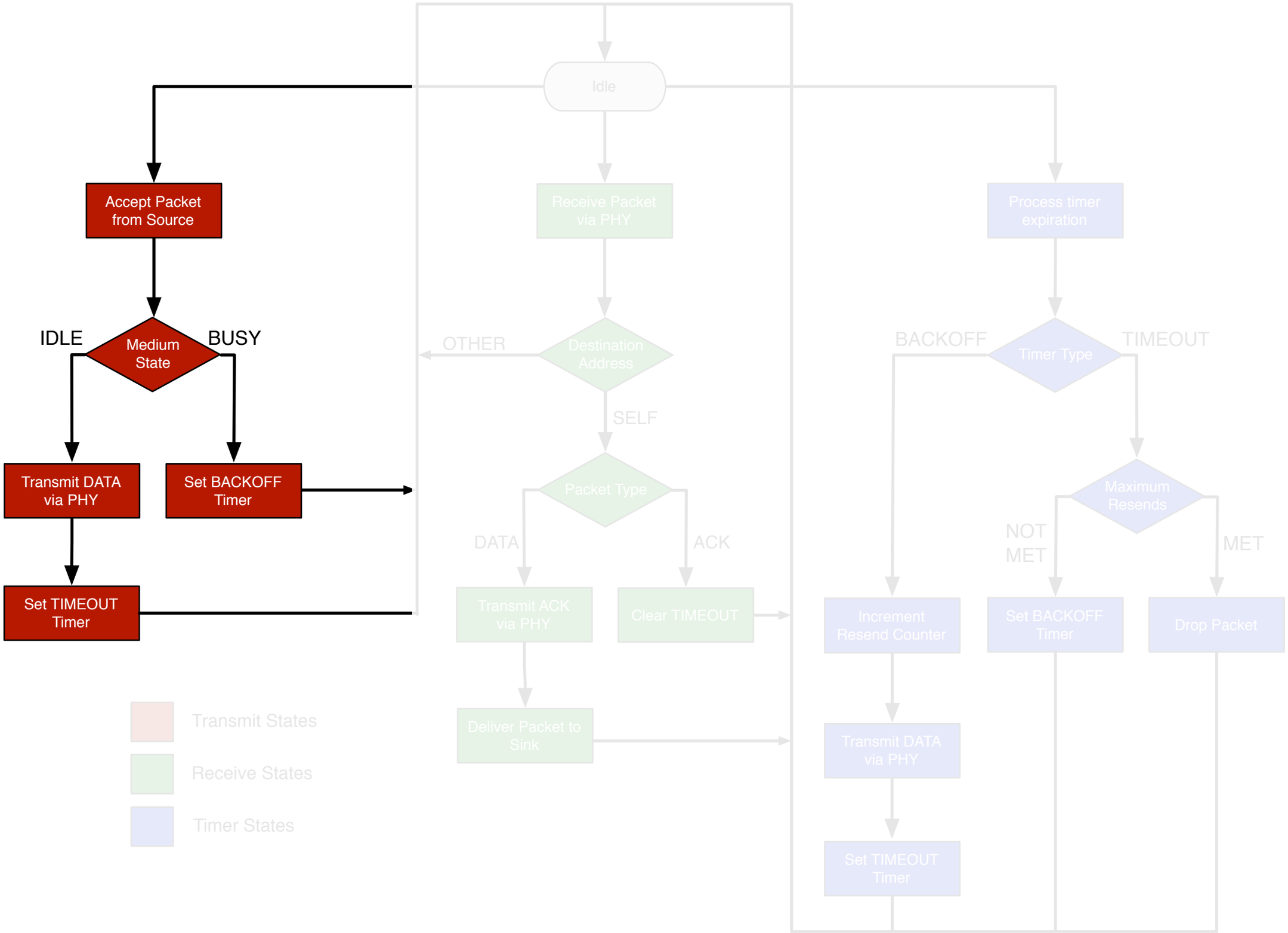

User Code
WARPMAC
WARPPHY
Drivers

An example: CSMA

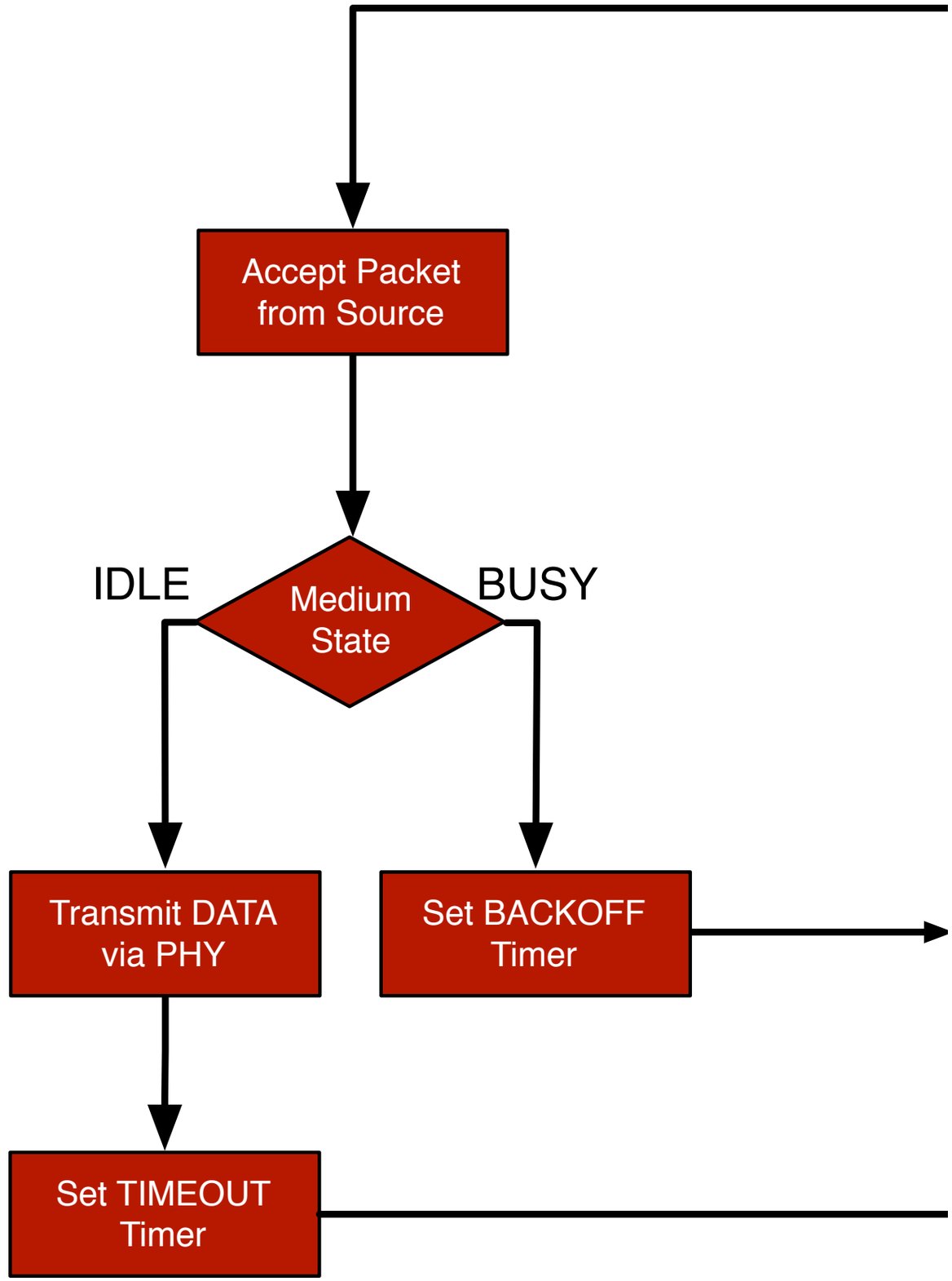
- Carrier-sensing Multiple Access
- Serves as a foundation for a large class of other random access protocols
- Fairly simple algorithm


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
SELF
BACKOFF
Timer Type
TIMEOUT
Transmit DATA via PHY
Set BACKOFF Timer
Packet Type
Maximum Resends
NOT MET
MET
Set TIMEOUT Timer
DATA
ACK
Transmit ACK via PHY
Clear TIMEOUT
Increment Resend Counter
Set BACKOFF Timer
Drop Packet
Deliver Packet to Sink
Transmit DATA via PHY
Set TIMEOUT Timer
Transmit States
Receive States
Timer States


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
SELF
BACKOFF
Timer Type
TIMEOUT
Transmit DATA via PHY
Set BACKOFF Timer
Packet Type
DATA
ACK
Maximum Resends
NOT MET
MET
Set TIMEOUT Timer
Transmit ACK via PHY
Clear TIMEOUT
Increment Resend Counter
Set BACKOFF Timer
Drop Packet
Deliver Packet to Sink
Transmit DATA via PHY
Set TIMEOUT Timer
Transmit States
Receive States
Timer States


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
SELF
BACKOFF
Timer Type
TIMEOUT
Transmit DATA via PHY
Set BACKOFF Timer
Packet Type
Maximum Resends
DATA
ACK
NOT MET
MET
Set TIMEOUT Timer
Transmit ACK via PHY
Clear TIMEOUT
Increment Resend Counter
Set BACKOFF Timer
Drop Packet
Deliver Packet to Sink
Transmit DATA via PHY
Set TIMEOUT Timer
Transmit States
Receive States
Timer States

Transmit States



Transmit States

emacRx_int_handler

- Starts DMA transfer from EMAC to PHY

emacRx_callback

- Constructs Macframe header for data packet

If medium is idle {

warpmac_prepPhyForXmit

- Configures PHY
- Copies Macframe header into PHY's buffer

warpmac_startPhyXmit

- Disables packet detection
- Starts radio controller's transmit state machine

warpmac_finishPhyXmit

- Polls PHY and waits for it to complete
- Enables packet detection and radio reception

- Starts a timeout timer

}

If medium is busy {

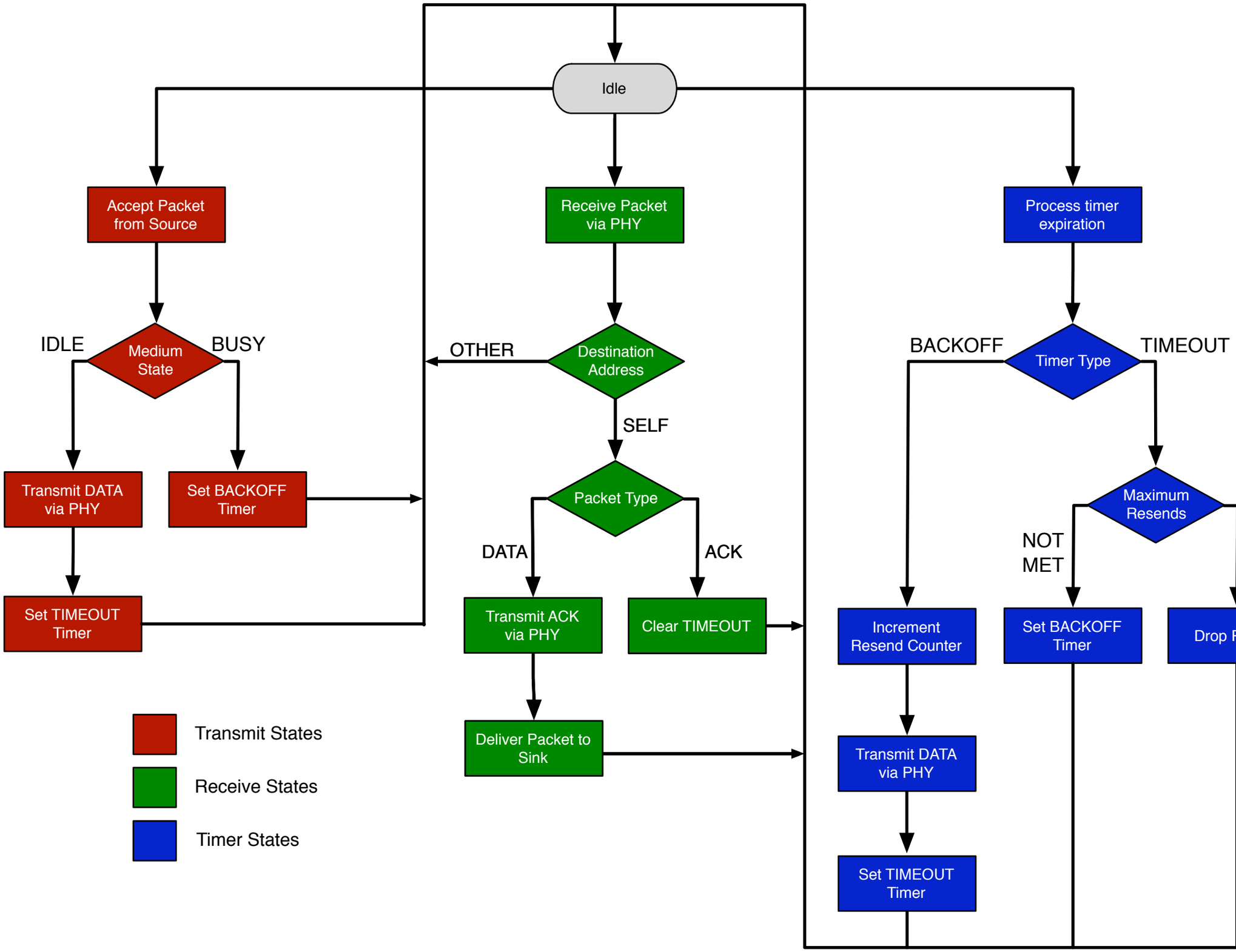
- Starts a backoff timer

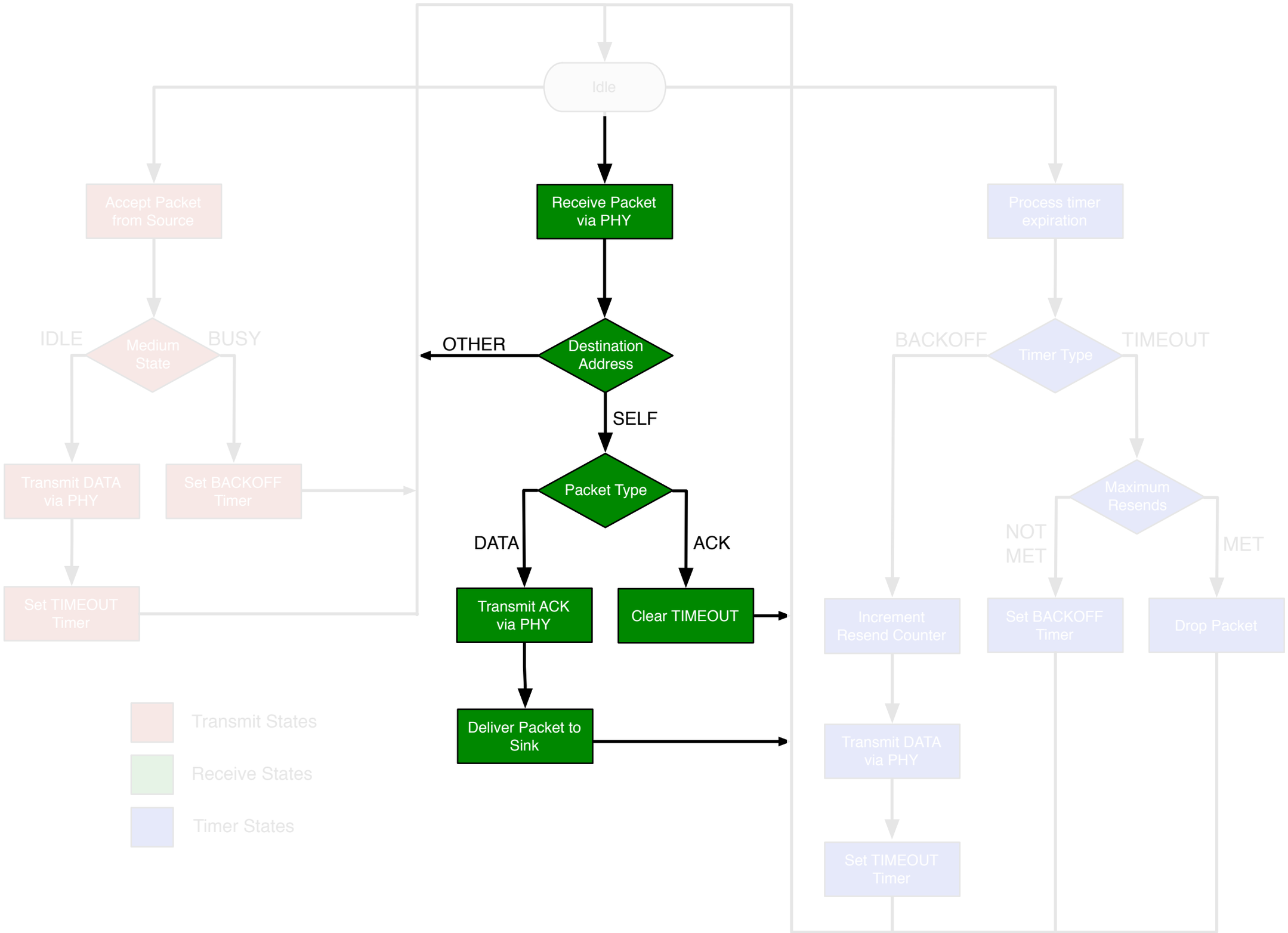
}

- Clears EMAC and interrupt controller

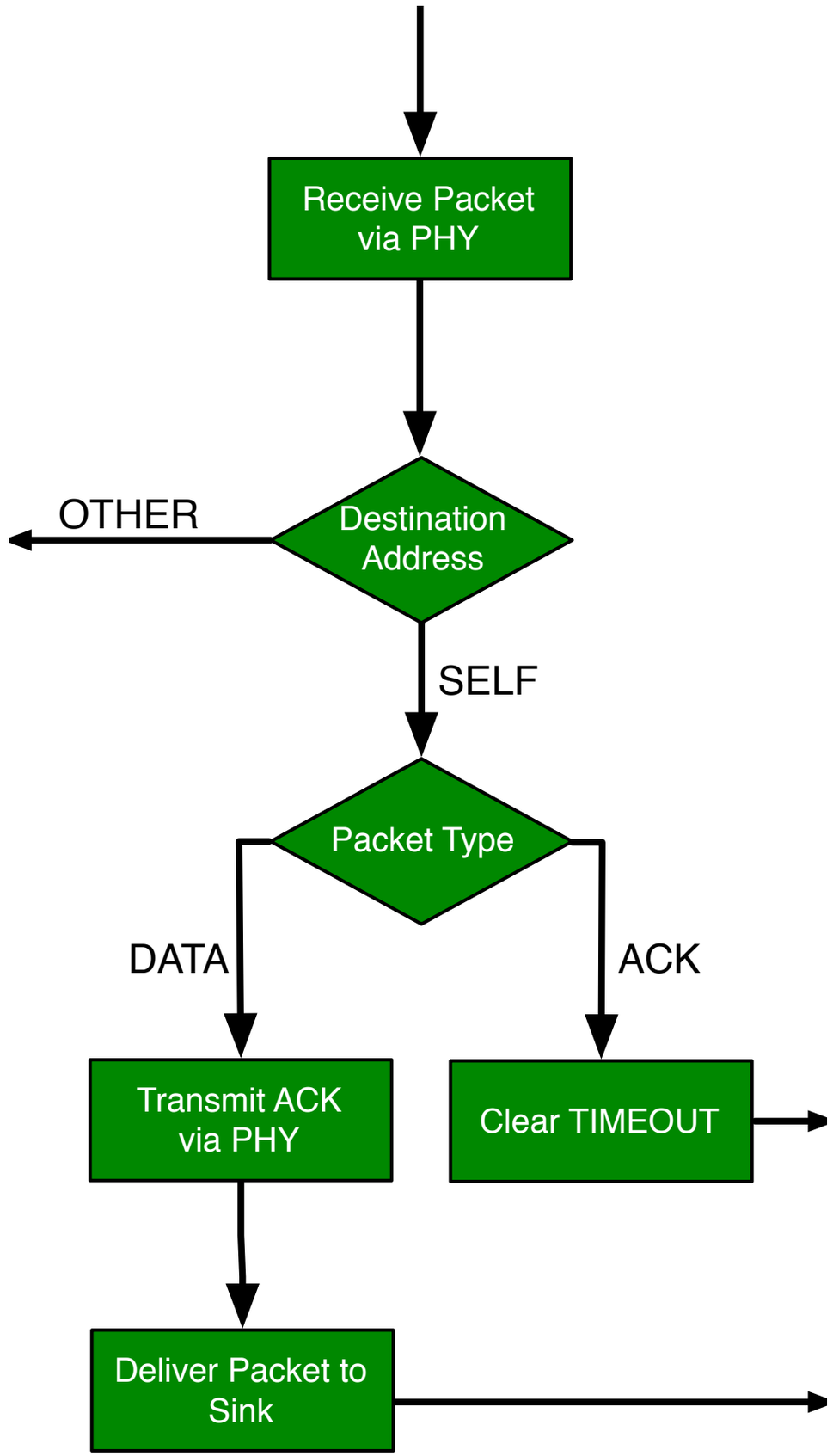
WARPMAC

User-Code


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
BACKOFF
Timer Type
TIMEOUT
SELF
Transmit DATA via PHY
Set BACKOFF Timer
Packet Type
Maximum Resends
NOT MET
MET
DATA
ACK
Set TIMEOUT Timer
Transmit ACK via PHY
Clear TIMEOUT
Increment Resend Counter
Set BACKOFF Timer
Drop Packet
Deliver Packet to Sink
Transmit DATA via PHY
Set TIMEOUT Timer
Transmit States
Receive States
Timer States


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
BACKOFF
Timer Type
TIMEOUT
Transmit DATA via PHY
Set BACKOFF Timer
SELF
Maximum Resends
NOT MET
MET
Packet Type
DATA
ACK
Set TIMEOUT Timer
Transmit ACK via PHY
Clear TIMEOUT
Increment Resend Counter
Set BACKOFF Timer
Drop Packet
Deliver Packet to Sink
Transmit DATA via PHY
Set TIMEOUT Timer
Transmit States
Receive States
Timer States

Receive States



Receive States

phyRx_goodHeader_int_handler

- Copies header into Macframe

phyRx_goodHeader_callback

- Checks address/type fields of Macframe header

If data {

- Constructs an ACK Macframe to prepare for a complete Rx packet

warpmac_prepPhyForXmit

- Configures PHY
- Copies Macframe header into PHY's buffer

- Polls PHY receiver and waits for a "Good" or "Bad" state

If Good {

warpmac_startPhyXmit

- Disables packet detection
- Starts radio controller's transmit state machine

warpmac_prepEmacForXmit

- Starts DMA transfer from PHY to EMAC

warpmac_finishPhyXmit

- Polls PHY and waits for it to complete
- Enables packet detection and radio reception

warpmac_startEmacXmit

- Polls DMA and waits for it to complete
- Starts EMAC transmission

}

}

If acknowledgment {

- Clears timeout timer

}

- Clears interrupts in PHY and interrupt controller

WARPMAC

User-Code

Receive States

phyRx_goodHeader_int_handler

- Copies header into Macframe

phyRx_goodHeader_callback

- Checks address/type fields of Macframe header

If data {

- Constructs an ACK Macframe to prepare for a complete Rx packet

warpmac_prepPhyForXmit

- Configures PHY
- Copies Macframe header into PHY's buffer

- Polls PHY receiver and waits for a "Good" or "Bad" state

If Good {

warpmac_startPhyXmit

- Disables packet detection
- Starts radio controller's transmit state machine

warpmac_prepEmacForXmit

- Starts DMA transfer from PHY to EMAC

warpmac_finishPhyXmit

- Polls PHY and waits for it to complete
- Enables packet detection and radio reception

warpmac_startEmacXmit

- Polls DMA and waits for it to complete
- Starts EMAC transmission

}

}

If acknowledgment {

- Clears timeout timer

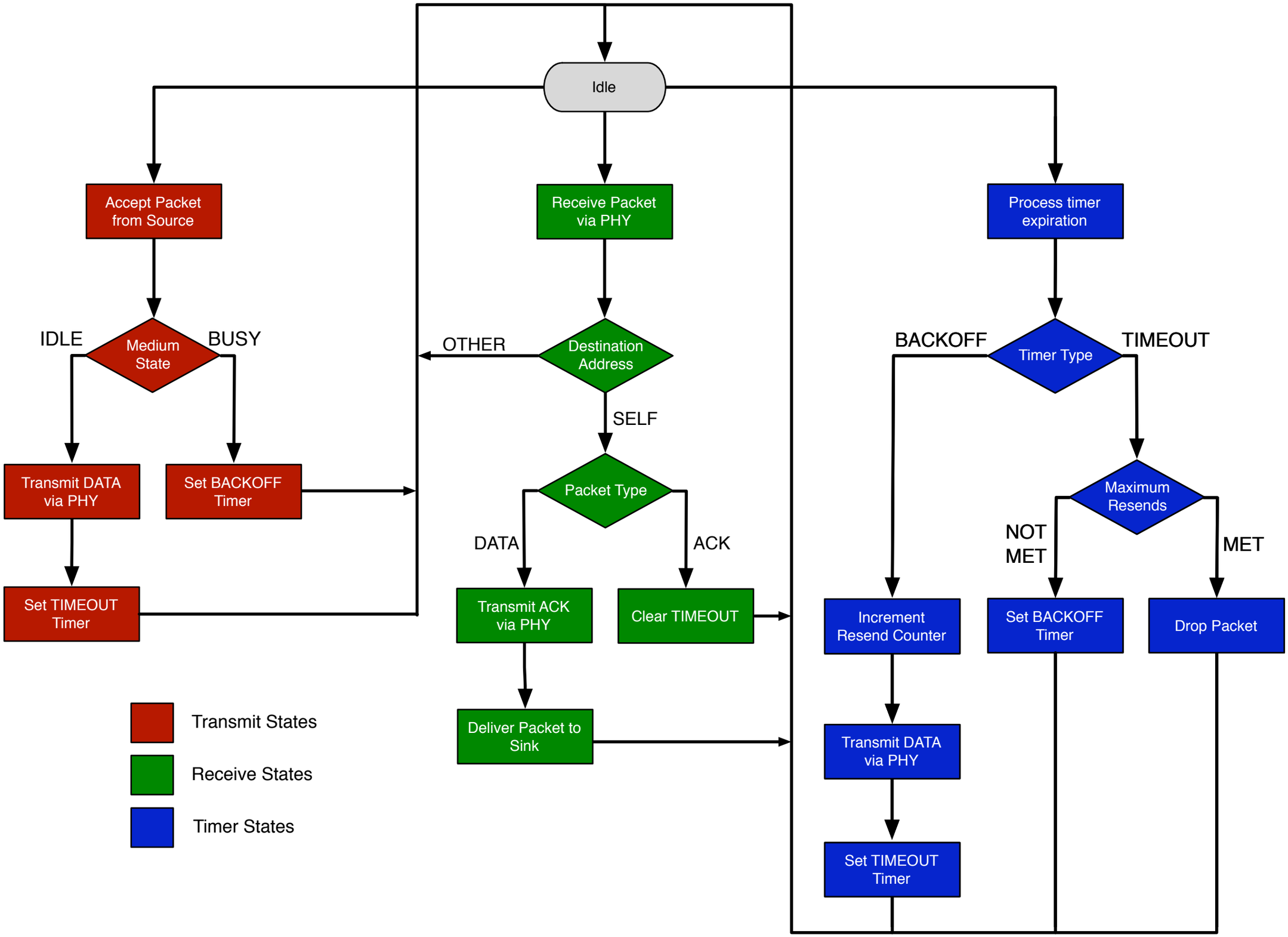
}

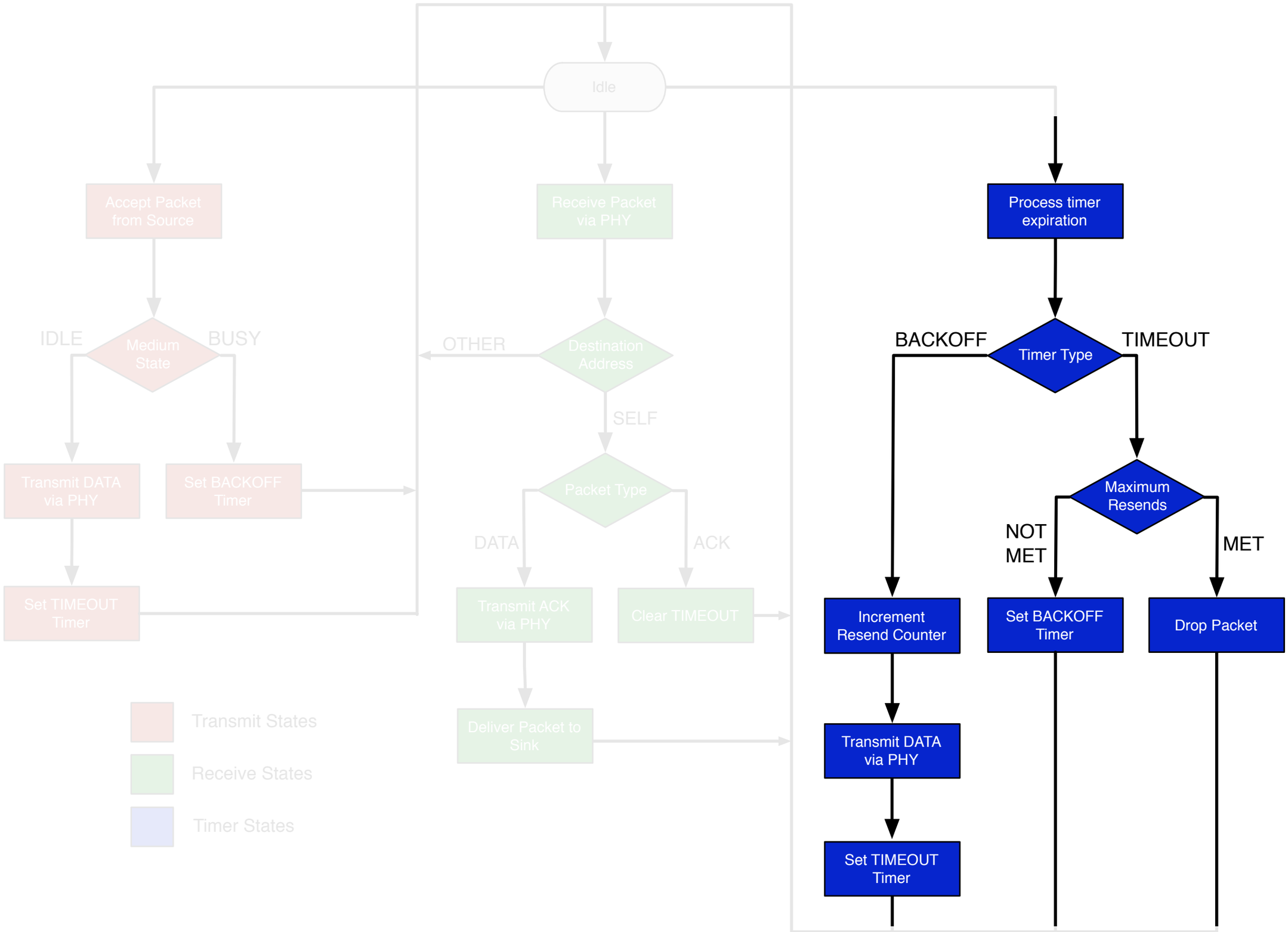
- Clears interrupts in PHY and interrupt controller

Fast Turn-Around Time (TAT)

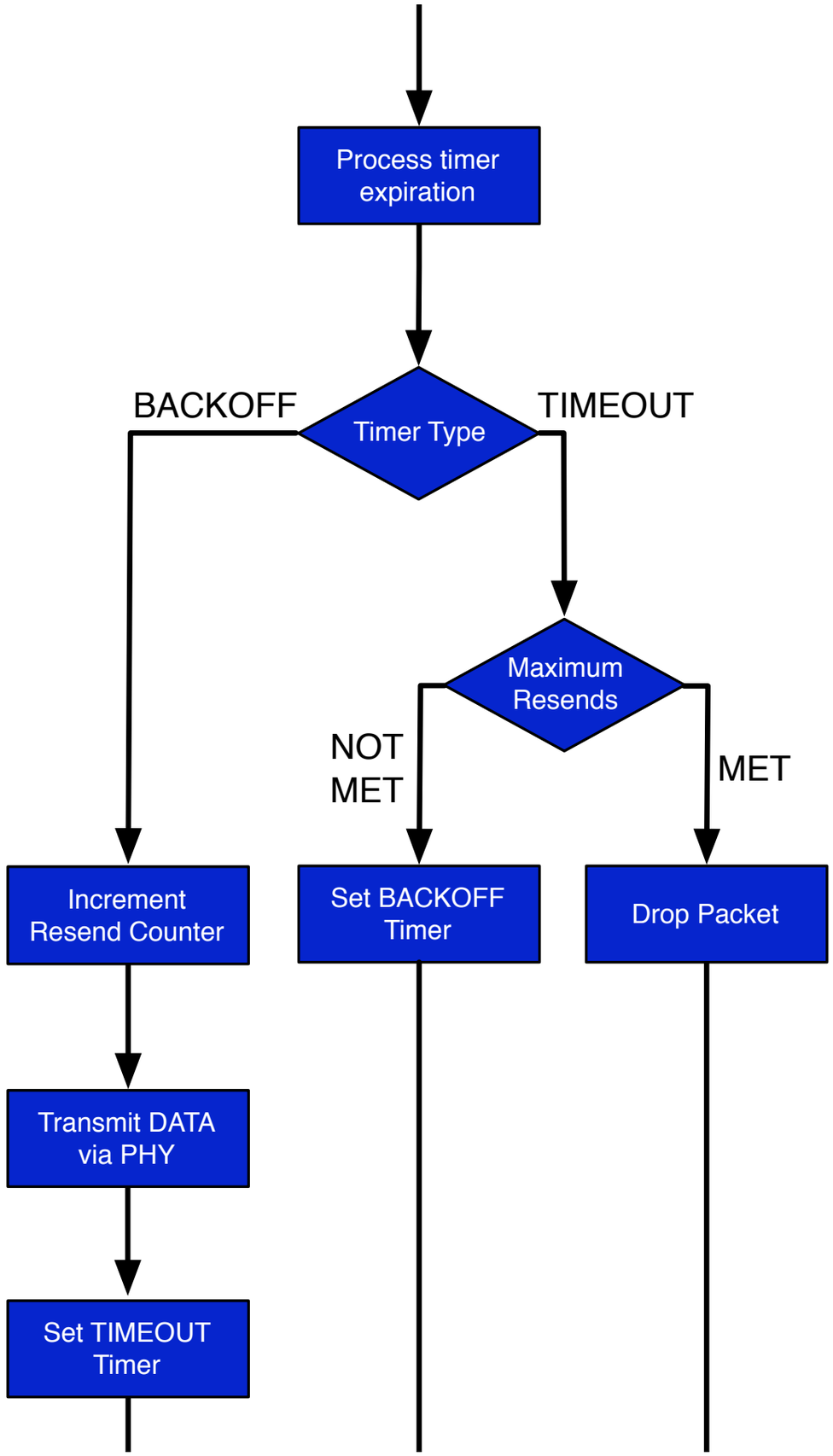
WARPMAC

User-Code


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
SELF
BACKOFF
Timer Type
TIMEOUT
Transmit DATA via PHY
Set BACKOFF Timer
Packet Type
DATA
ACK
Maximum Resends
NOT MET
MET
Set TIMEOUT Timer
Transmit ACK via PHY
Clear TIMEOUT
Increment Resend Counter
Set BACKOFF Timer
Drop Packet
Deliver Packet to Sink
Transmit DATA via PHY
Set TIMEOUT Timer
Transmit States
Receive States
Timer States


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
BACKOFF
Timer Type
TIMEOUT
SELF
Transmit DATA via PHY
Set BACKOFF Timer
Packet Type
Maximum Resends
NOT MET
MET
DATA
ACK
Set TIMEOUT Timer
Transmit ACK via PHY
Clear TIMEOUT
Increment Resend Counter
Set BACKOFF Timer
Drop Packet
Deliver Packet to Sink
Transmit DATA via PHY
Set TIMEOUT Timer
Transmit States
Receive States
Timer States

Timer States



Timer States

timer_int_handler

timer_callback

- Checks timer type

If timeout {

- Increments retransmission counter
- If maximum retransmissions reached, returns
- If not, starts a backoff timer

}

If backoff {

warpmac_prepPhyForXmit

- Configures PHY
- Copies Macframe header into PHY's buffer

warpmac_startPhyXmit

- Disables packet detection
- Starts radio controller's transmit state machine

warpmac_finishPhyXmit

- Polls PHY and waits for it to complete
- Enables packet detection and radio reception

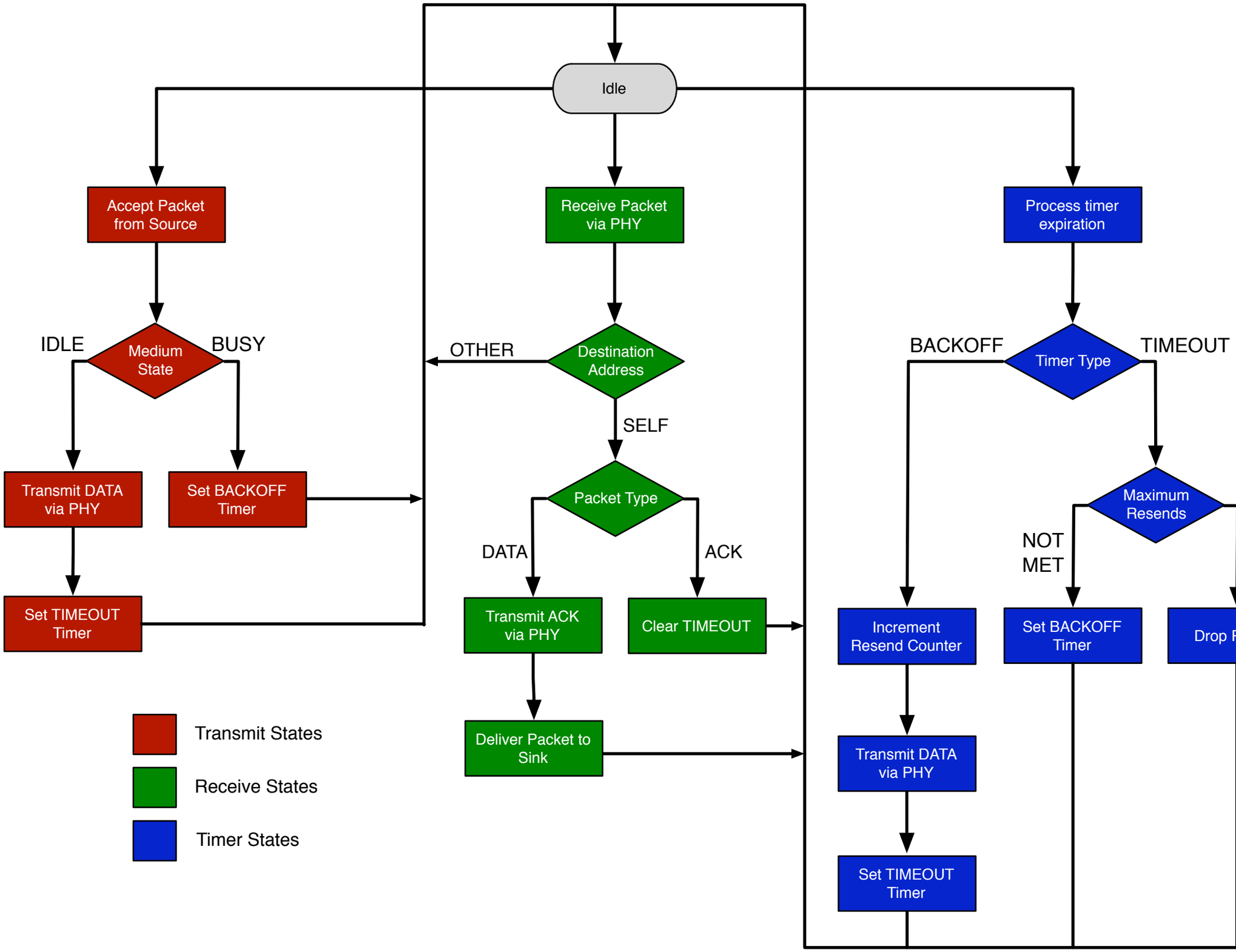
- Starts a timeout timer

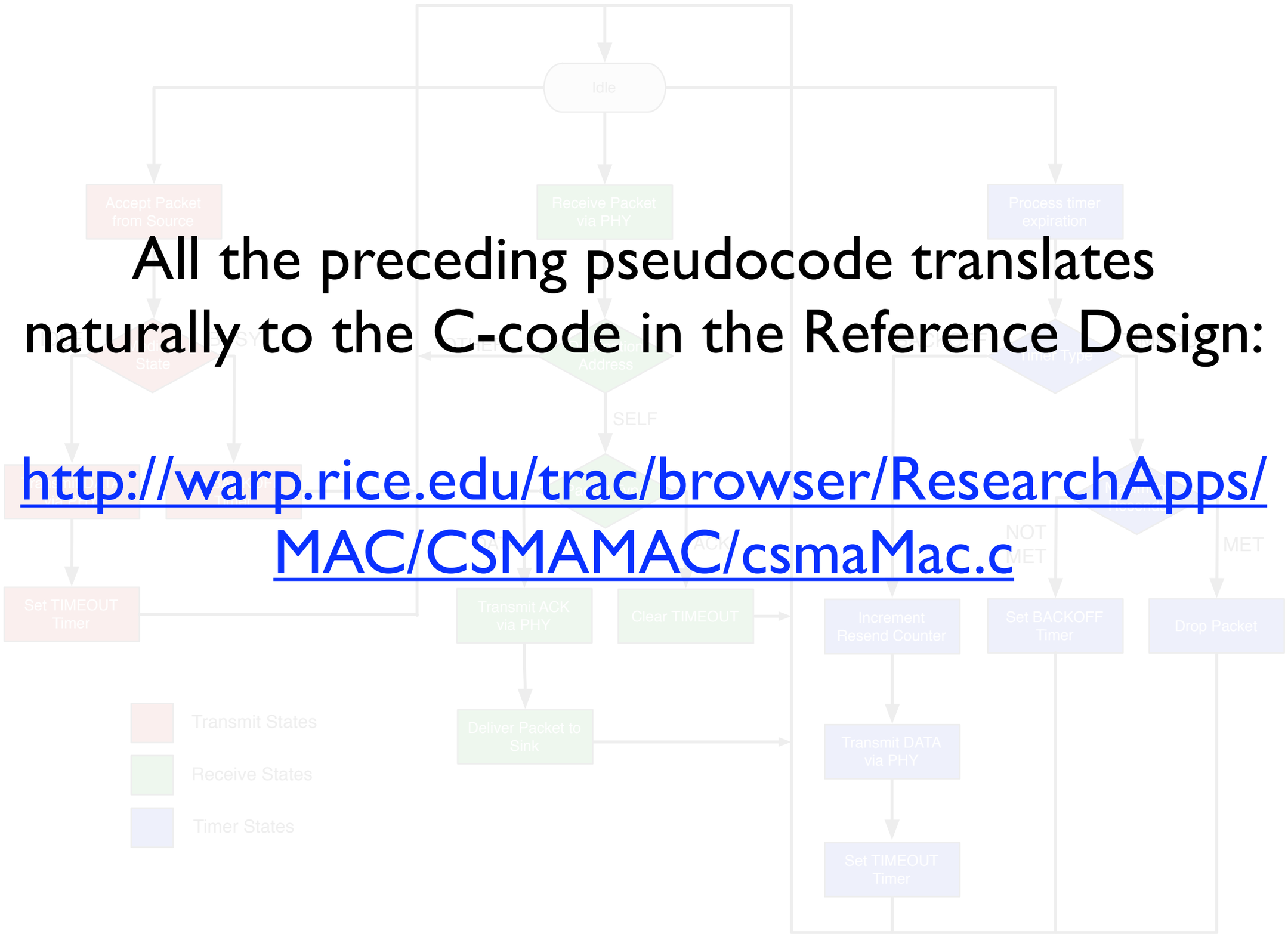
}

- Clears interrupts in timer and interrupt controller

WARPMAC

User-Code


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
SELF
BACKOFF
Timer Type
TIMEOUT
Transmit DATA via PHY
Set BACKOFF Timer
Packet Type
Maximum Resends
NOT MET
MET
Set TIMEOUT Timer
DATA
ACK
Transmit ACK via PHY
Clear TIMEOUT
Increment Resend Counter
Set BACKOFF Timer
Drop Packet
Deliver Packet to Sink
Transmit DATA via PHY
Set TIMEOUT Timer
Transmit States
Receive States
Timer States



All the preceding pseudocode translates naturally to the C-code in the Reference Design:

http://warp.rice.edu/trac/browser/ResearchApps/MAC/CSMAMAC/csmaMac.c

WARPMAC Structs

Macframe:

phyHeader header /* *Another struct* */
unsigned char isNew /* *Flag for new packets* */

Fully documented in API (http://warp.rice.edu/WARP_API)

WARPMAC Structs

phyHeader:

```
unsigned char fullRate; /* Payload modulation rate */
unsigned char codeRate; /* Coding rate */
unsigned short int length; /* Payload length */
unsigned char pktType; /* Packet type */
unsigned char destAddr[6]; /* Destination address */
unsigned char srcAddr[6]; /* Source address */
unsigned char currReSend; /* Re-send count */
unsigned char reserved1; /* Unused */
unsigned char reserved2; /* Unused */
unsigned char reserved3; /* Unused */
unsigned short int checksum; /* CRC placeholder */
```

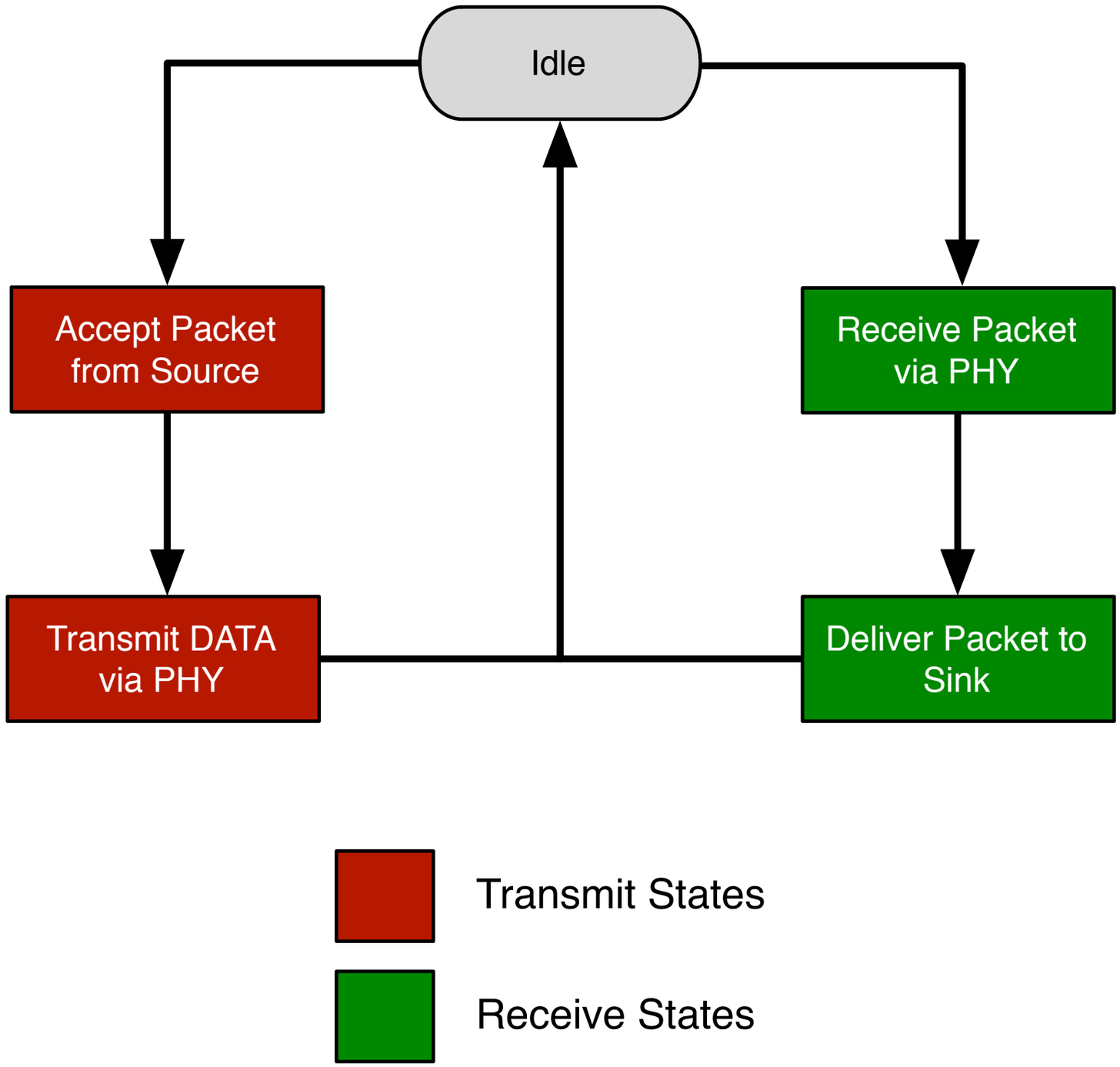
Fully documented in API (http://warp.rice.edu/WARP_API)

Questions?

Lab Exercises

- Lab 4 - noMAC: Too simple to be a MAC
- Lab 5 - halfMAC: Receive-half of a MAC
- Lab 6 - hopMAC: Channel-hopping extension

noMac



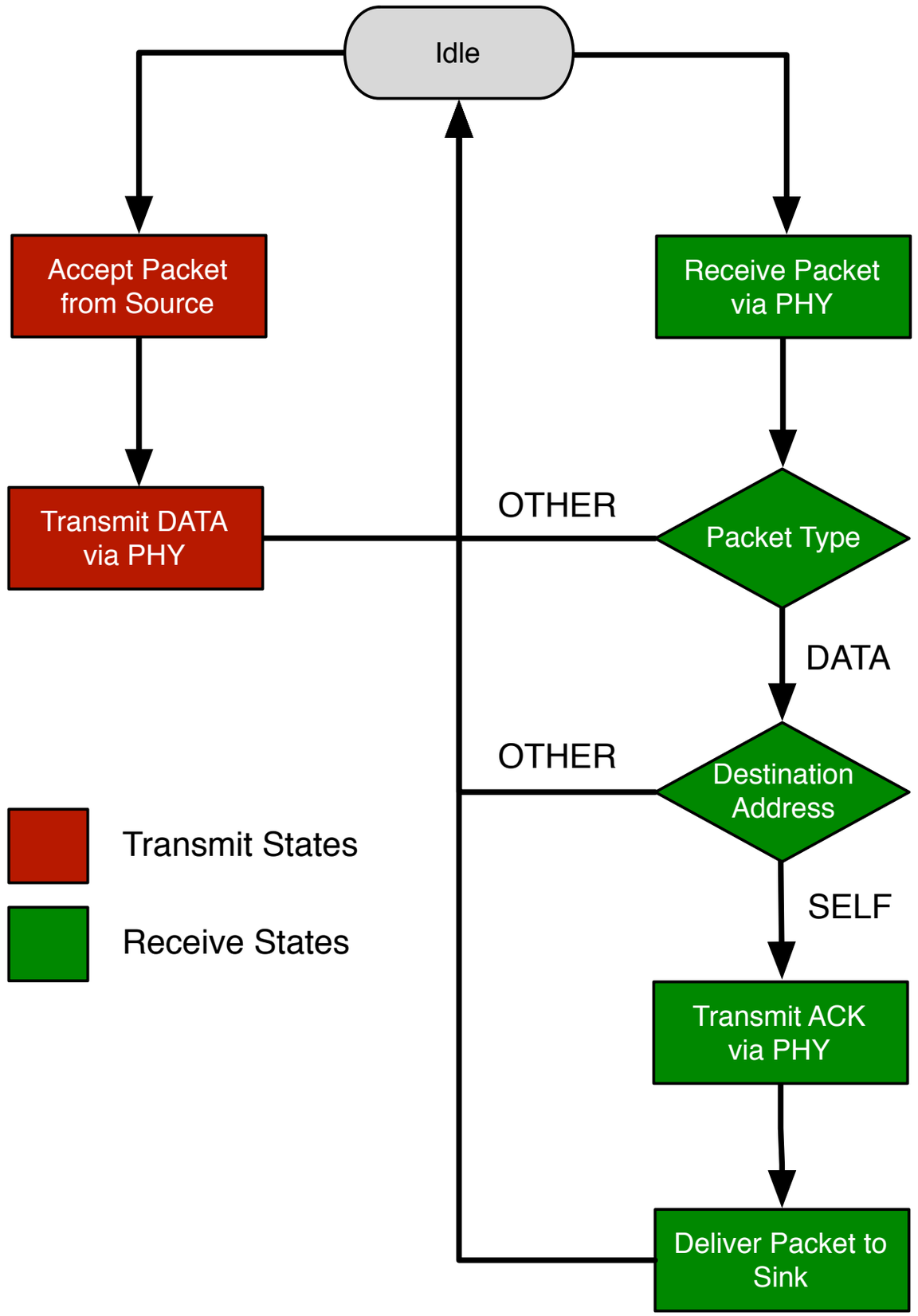
To test your noMac code, ping 10.0.0.20

Questions?

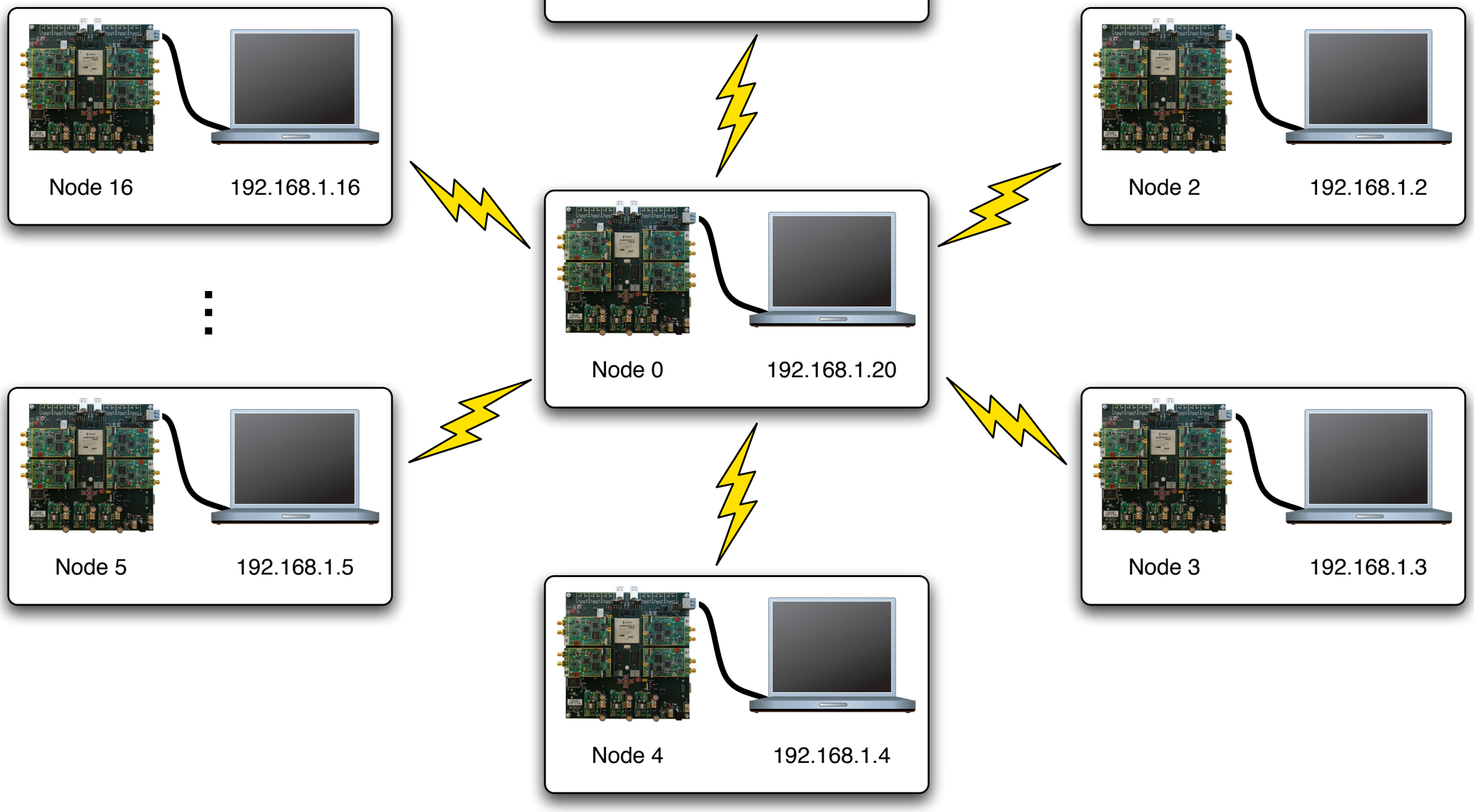
Remember to use the API:

http://warp.rice.edu/WARP_API

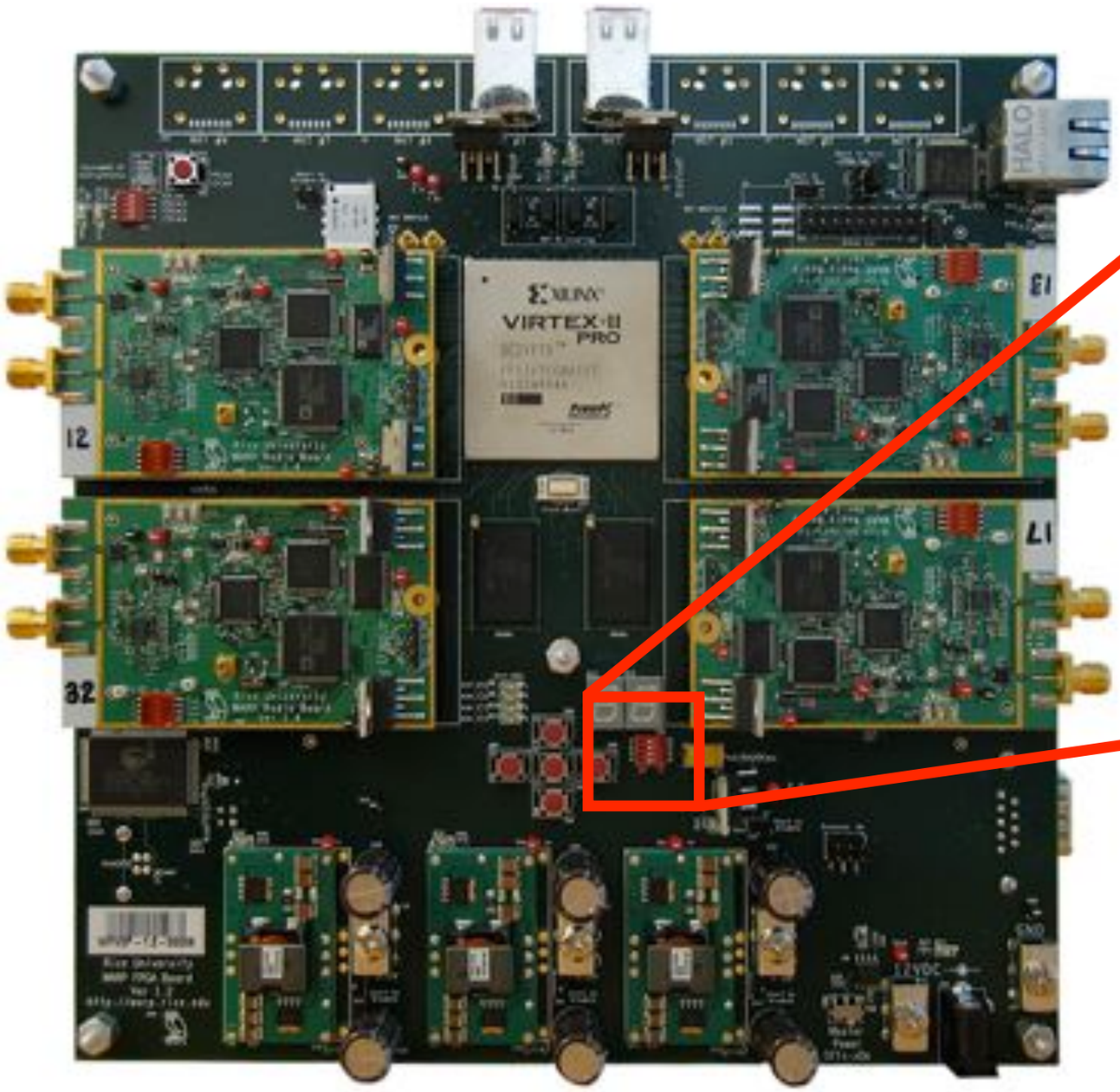
halfMac

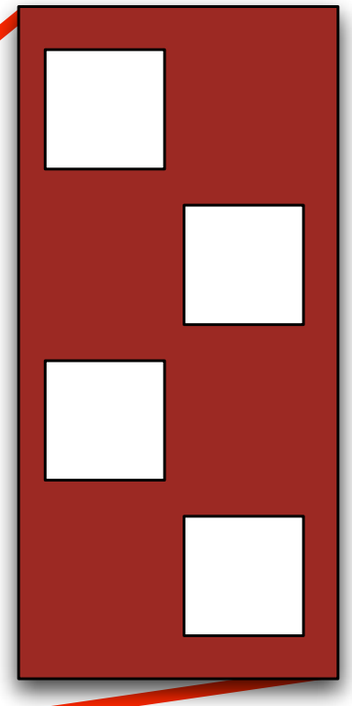

Idle
Accept Packet from Source
Receive Packet via PHY
Transmit DATA via PHY
OTHER
Packet Type
DATA
OTHER
Destination Address
SELF
Transmit ACK via PHY
Deliver Packet to Sink
Transmit States
Receive States

halfMac


Node 1
192.168.1.1
Node 16
192.168.1.16
Node 2
192.168.1.2
Node 0
192.168.1.20
Node 5
192.168.1.5
Node 3
192.168.1.3
Node 4
192.168.1.4

halfMac





Most Significant Bit (MSB)

Least Significant Bit (LSB)

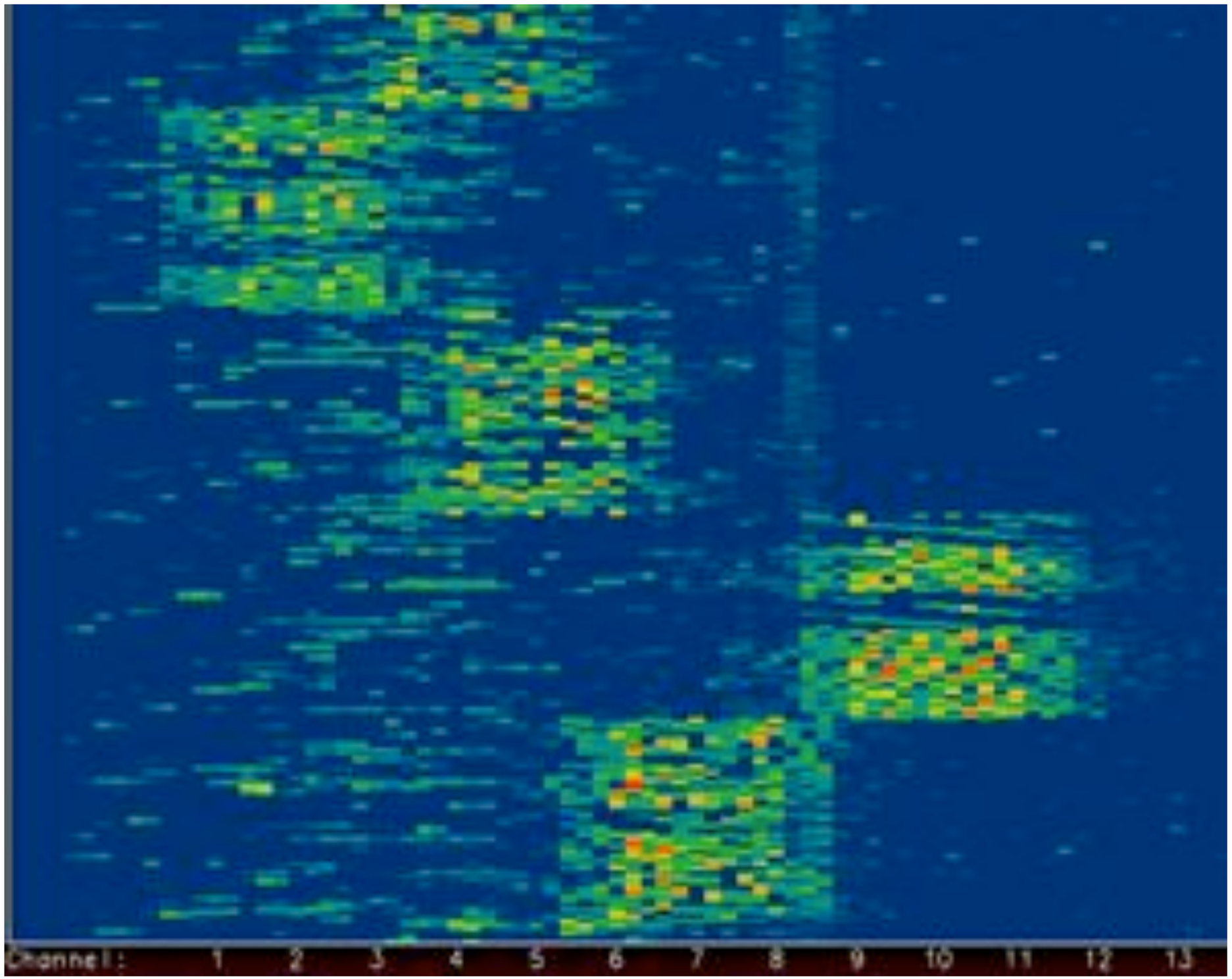
Node 5

Questions?

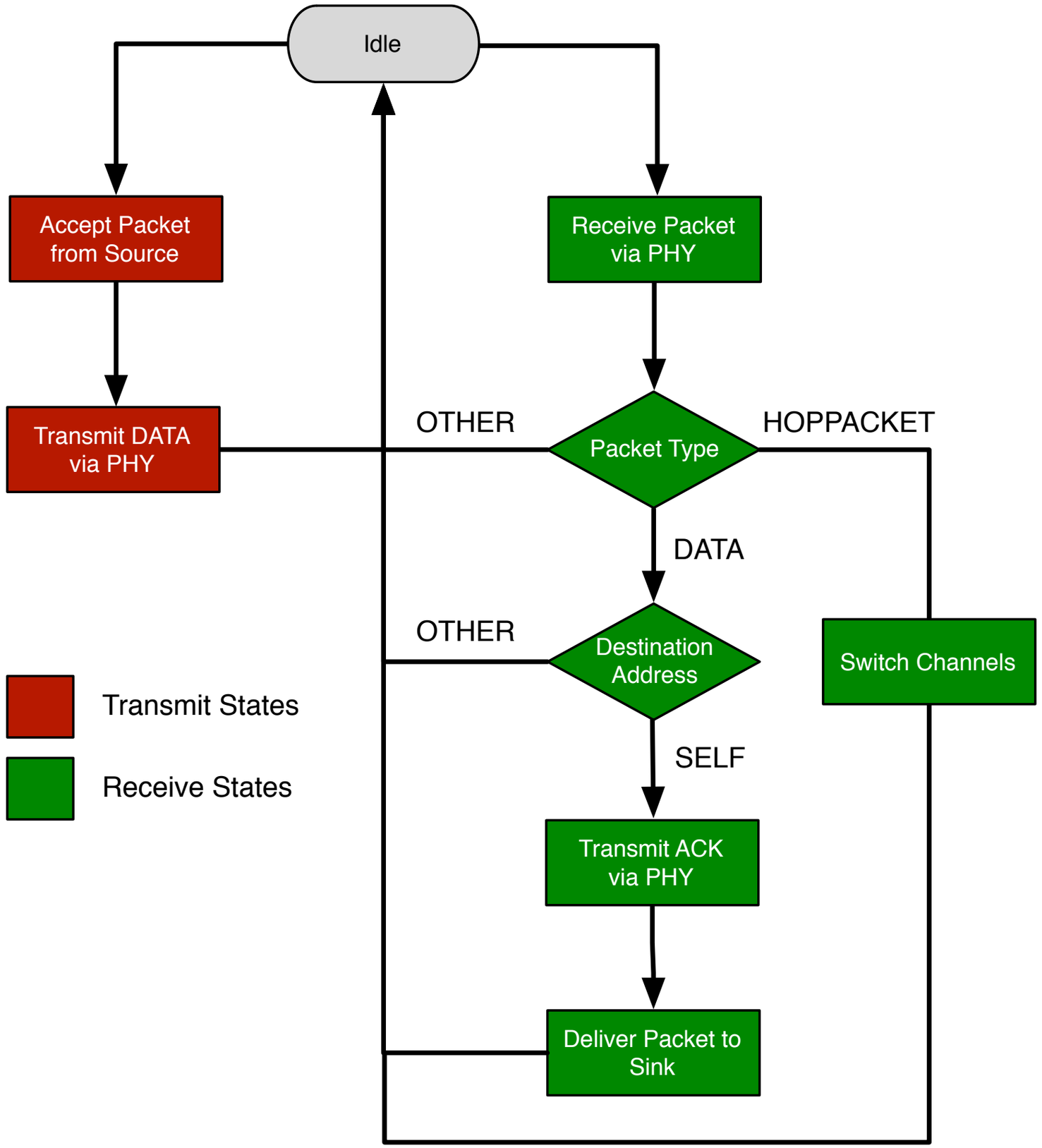
Remember to use the API:

http://warp.rice.edu/WARP_API

hopMac



hopMac


Idle
Accept Packet from Source
Receive Packet via PHY
Transmit DATA via PHY
OTHER
Packet Type
HOPPACKET
DATA
OTHER
Destination Address
Switch Channels
SELF
Transmit ACK via PHY
Deliver Packet to Sink
Transmit States
Receive States

Questions?

Remember to use the API:

http://warp.rice.edu/WARP_API

Logistics

- Contacting us
  - Support & technical questions
    - http://warp.rice.edu/forums/
- Hardware sales
  - Mango Communications (http://mangocomm.com/)