

# Networking on WARP

Chris Hunter  
Rice University

WARP Workshop at Rice University  
July 15, 2008

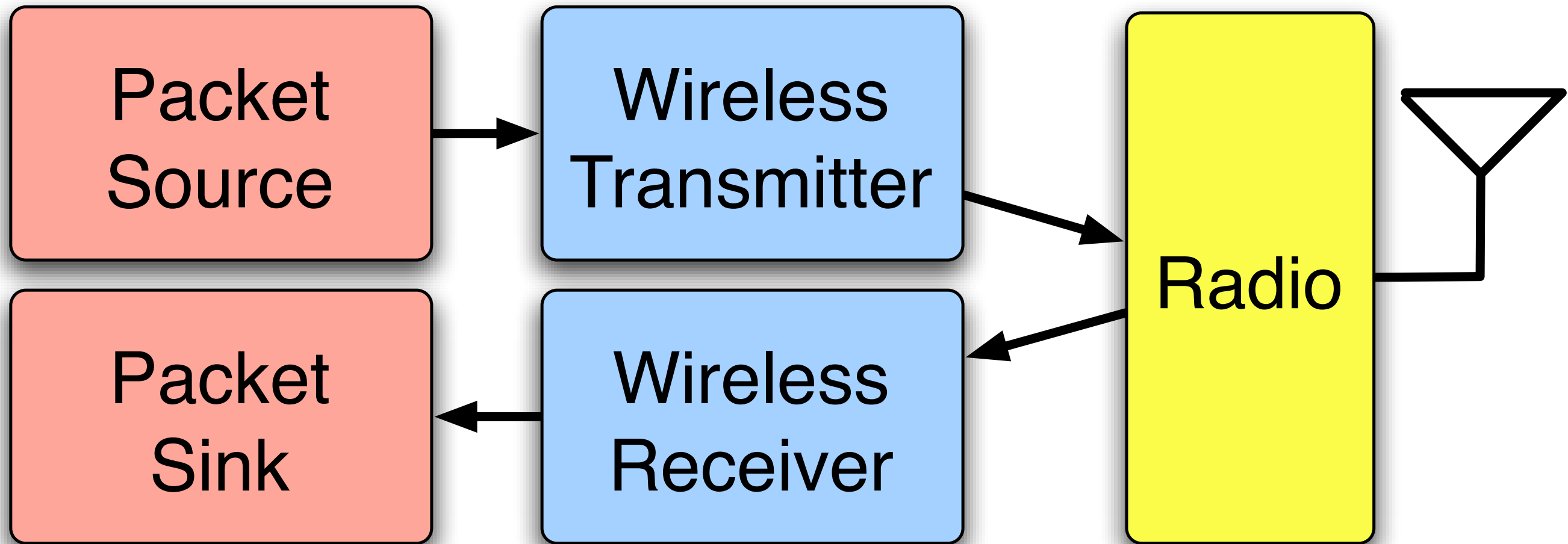


# Today's Agenda

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

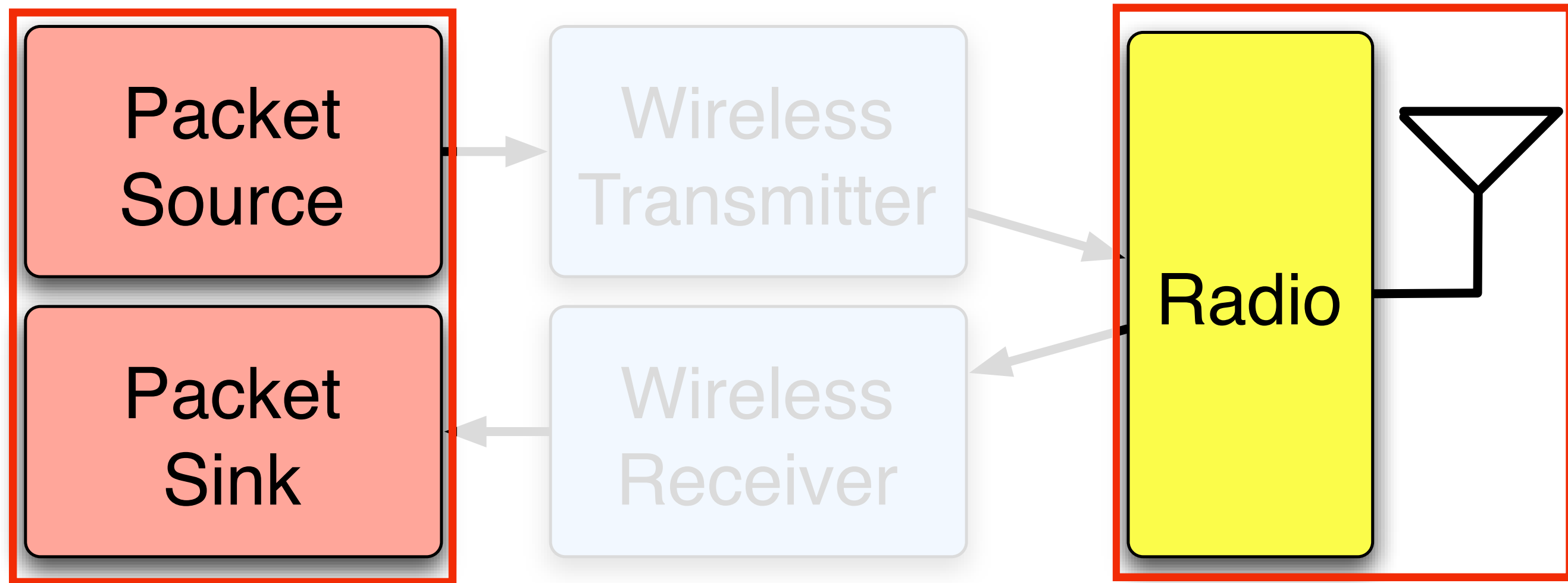
# Physical Layer Basics

*Simple Wireless Node*



# Physical Layer Basics

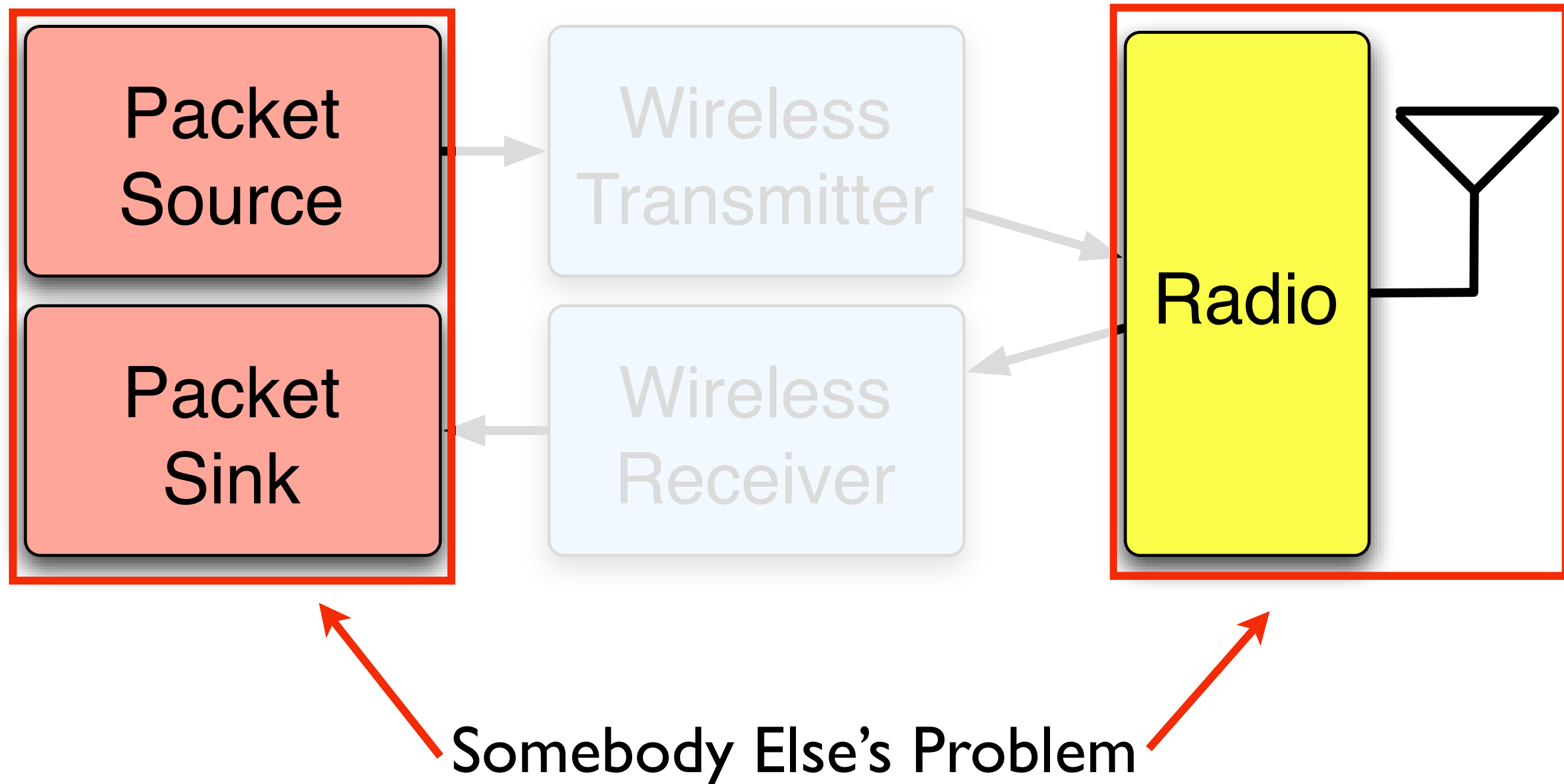
## *Simple Wireless Node*



Somebody Else's Problem

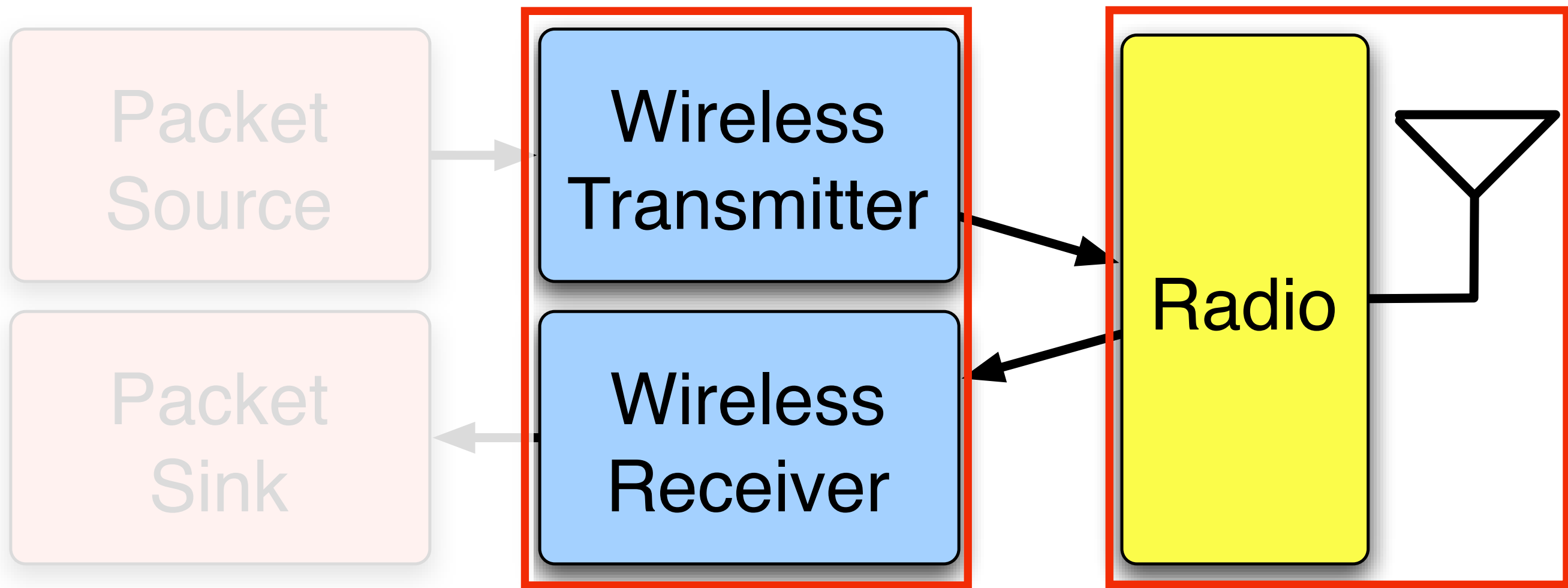
# Network Layer Basics

## *Simple Wireless Node*



# Network Layer Basics

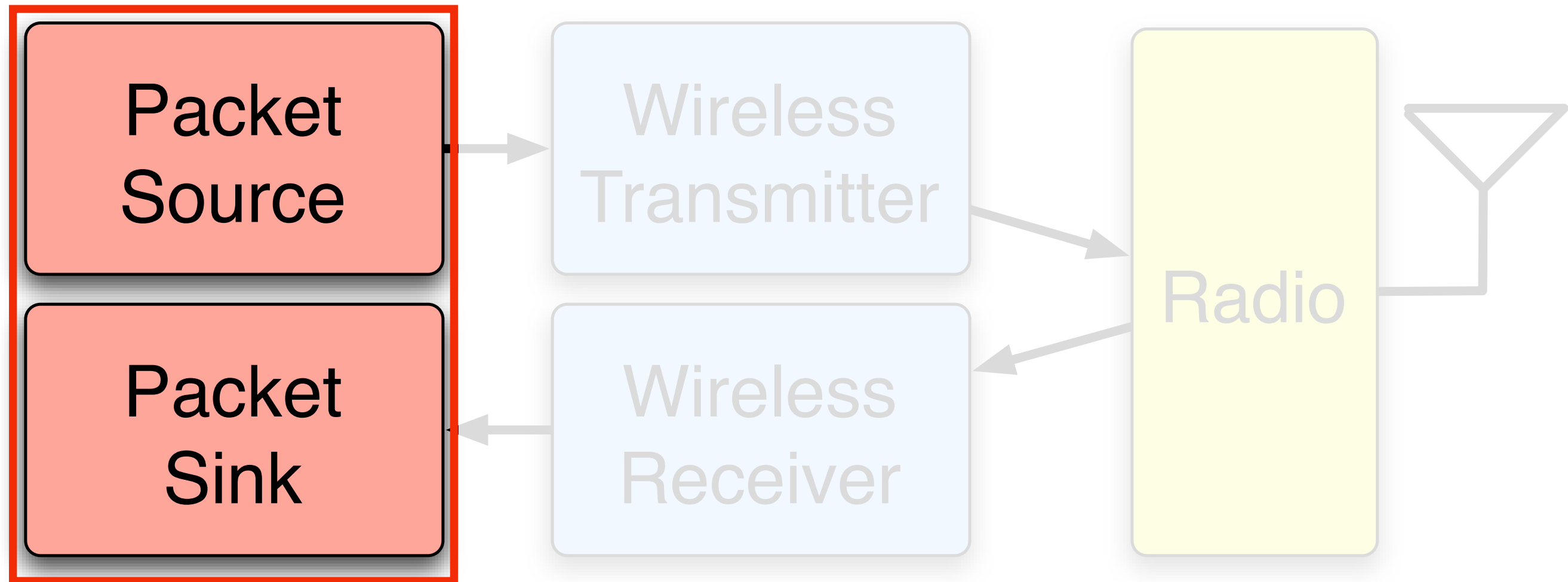
## *Simple Wireless Node*



Somebody Else's Problem

# Network Layer Basics

## *Simple Wireless Node*



# Targeting WARP Hardware

(Understanding the Development Environment)

## FPGA (Xilinx XC2VP70)

CUSTOM  
HIGH-LEVEL  
APP. CODE

CUSTOM  
LOW-LEVEL  
DRIVER CODE

EMBEDDED PROCESSOR  
(PowerPC)

WARP/XILINX SUPPORT SOFTWARE  
(LIBRARIES, DRIVERS, ETCETERA)

STANDARD  
BUSES  
(PLB, OPB)

CUSTOM  
HARDWARE IP  
(GENERATED)

CUSTOM  
HARDWARE IP  
(HAND-CODED)

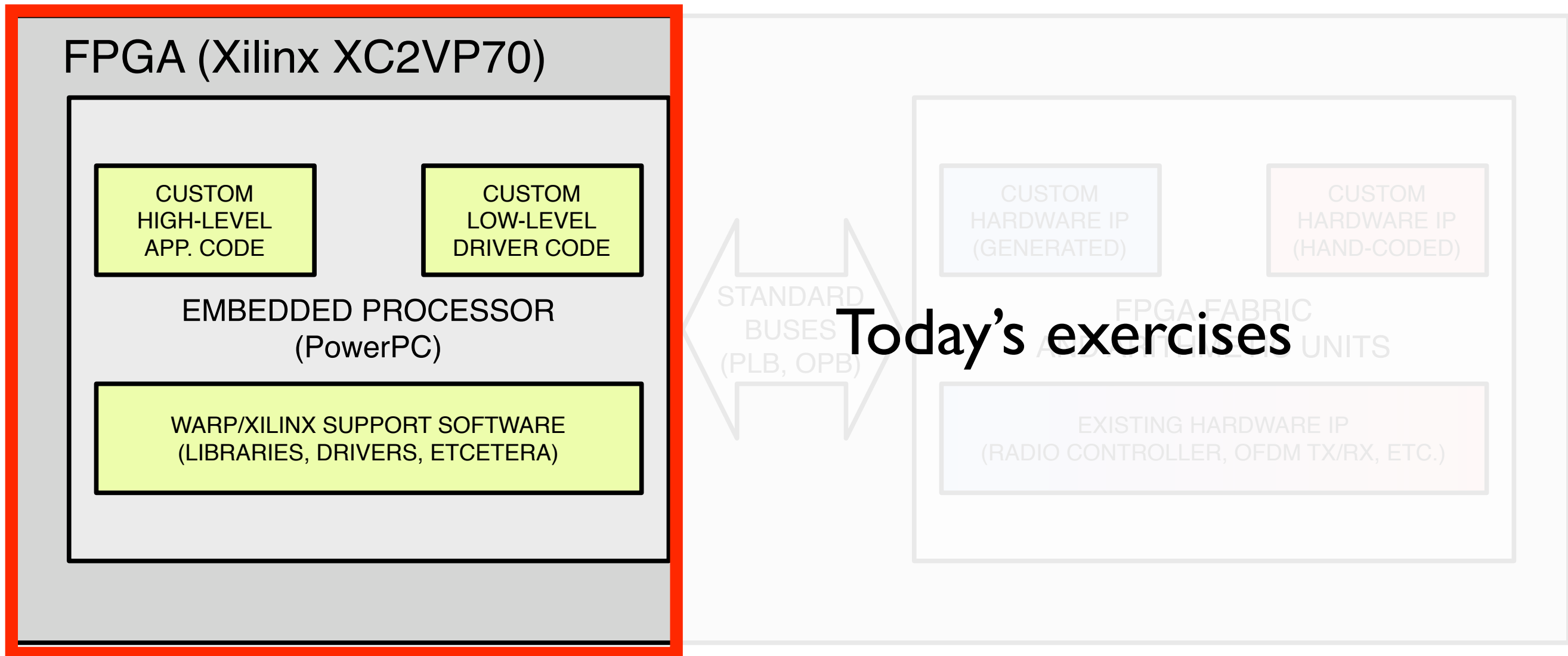
FPGA FABRIC  
AND ARITHMETIC UNITS

EXISTING HARDWARE IP  
(RADIO CONTROLLER, OFDM TX/RX, ETC.)

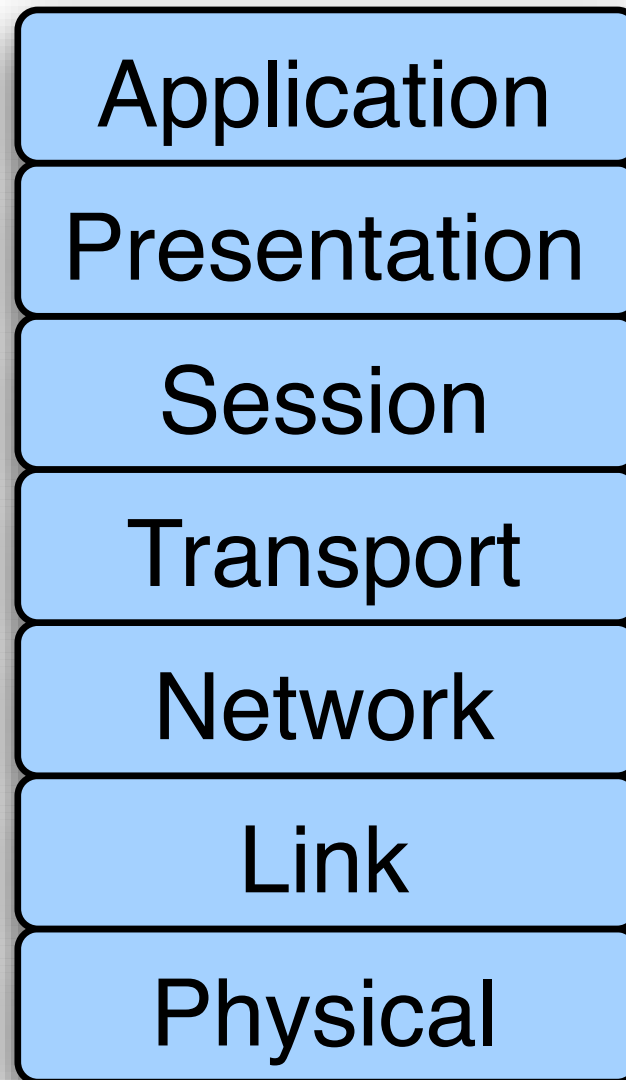


# Targeting WARP Hardware

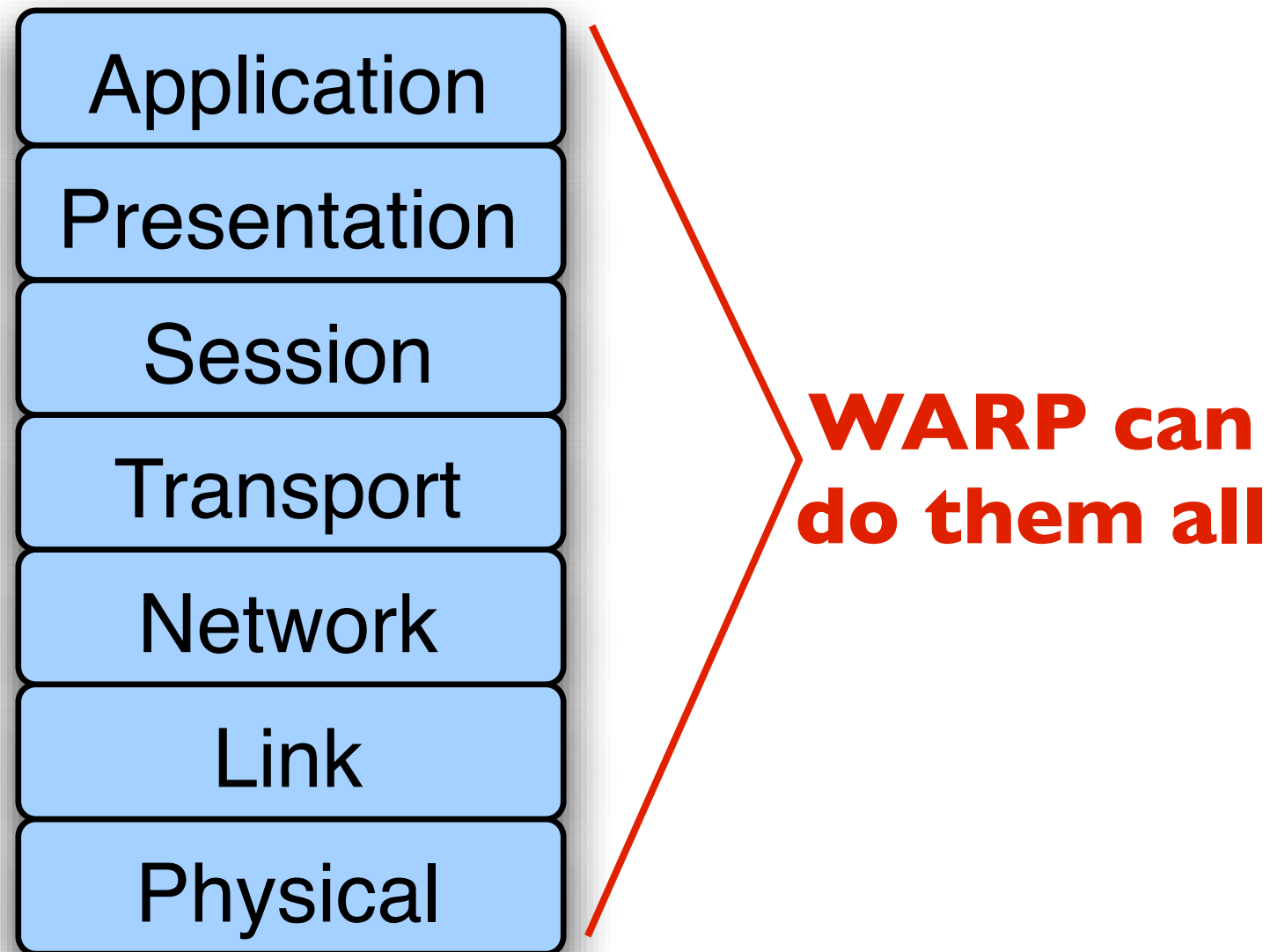
(Understanding the Development Environment)



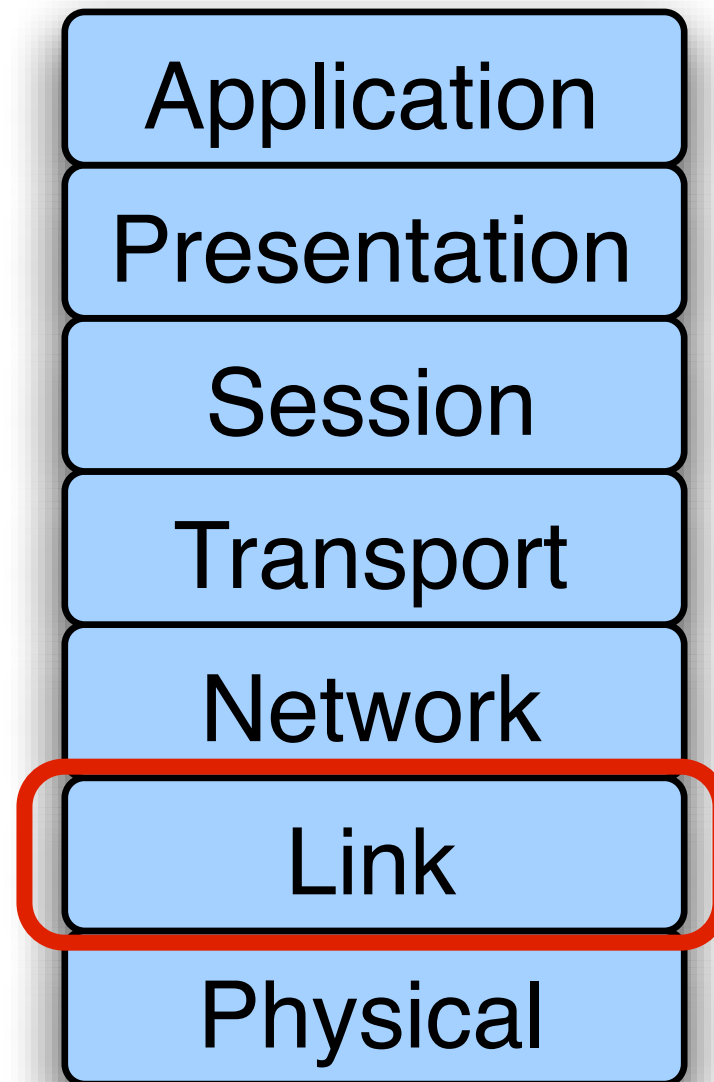
# Some Perspective - The OSI Model



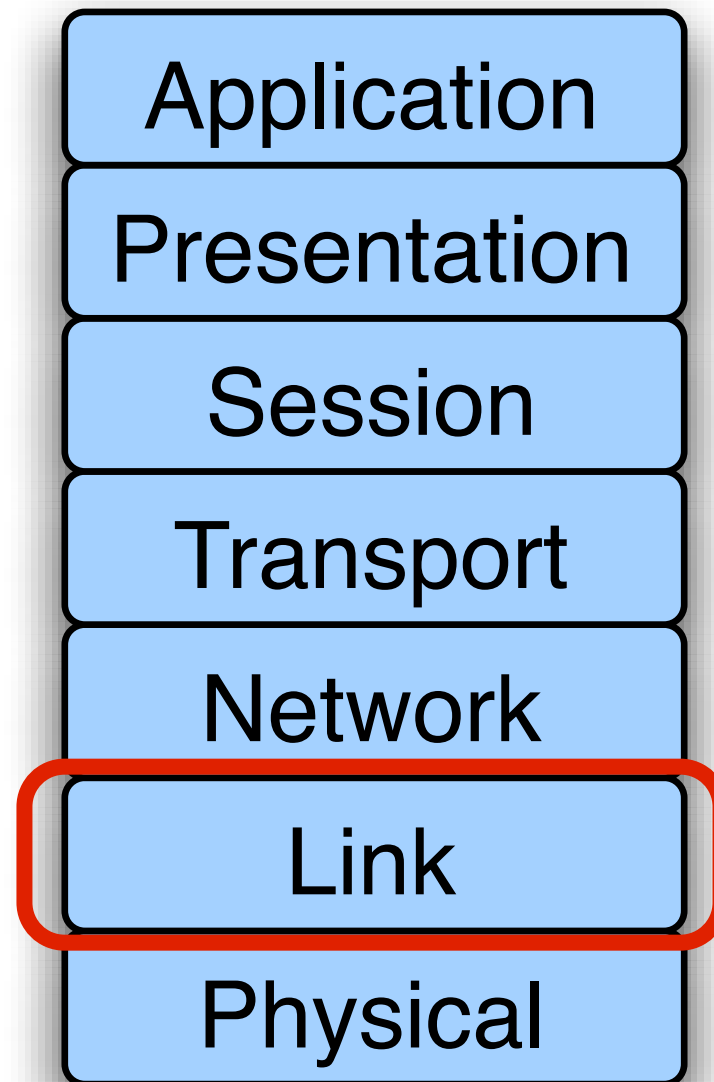
# Some Perspective - The OSI Model



# The OSI Model

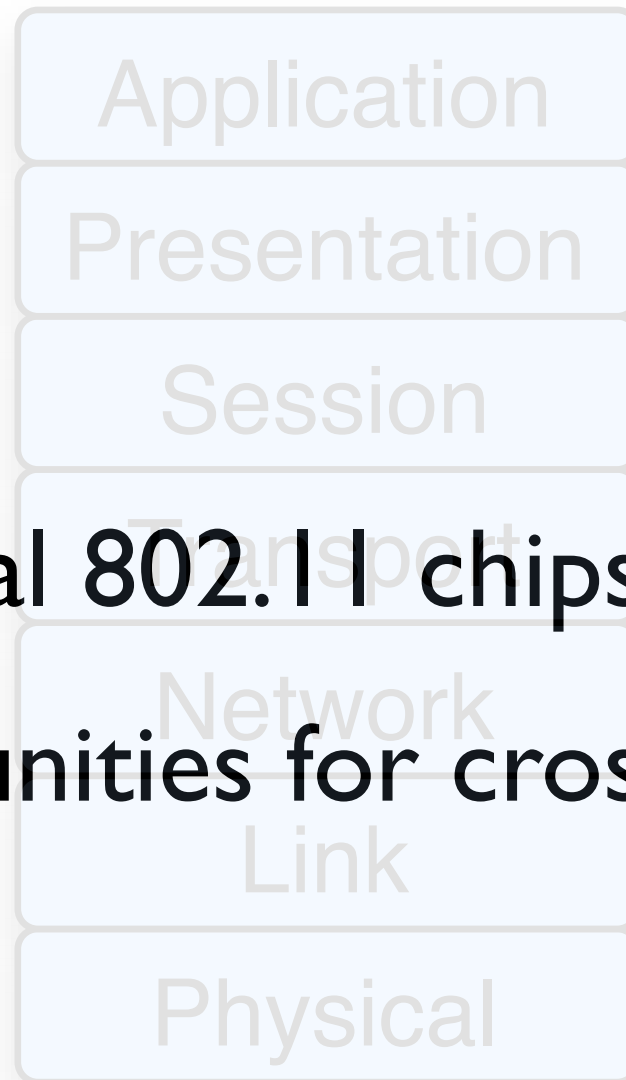


# The OSI Model



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?

User  
1

User  
2

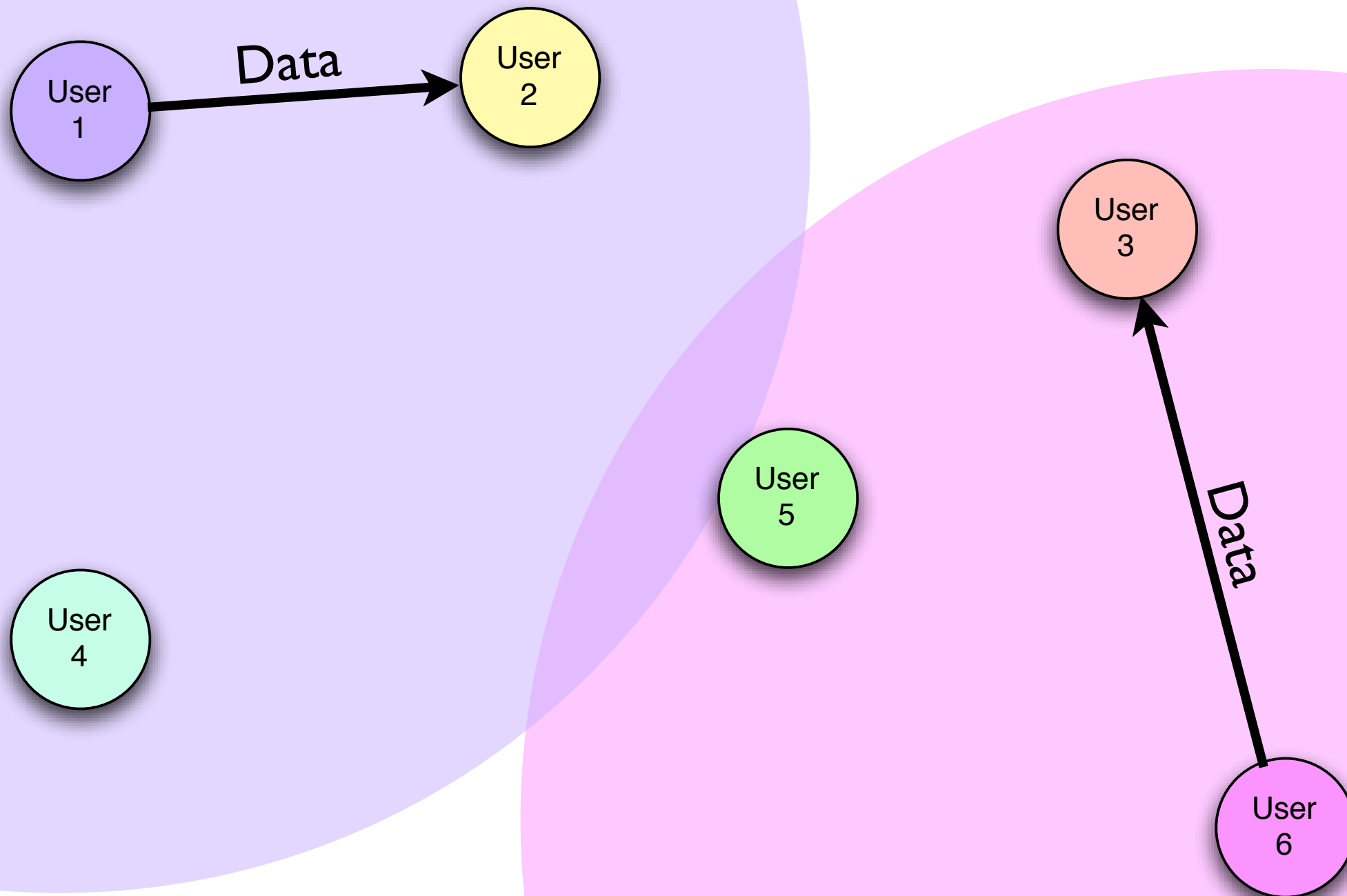
User  
3

User  
5

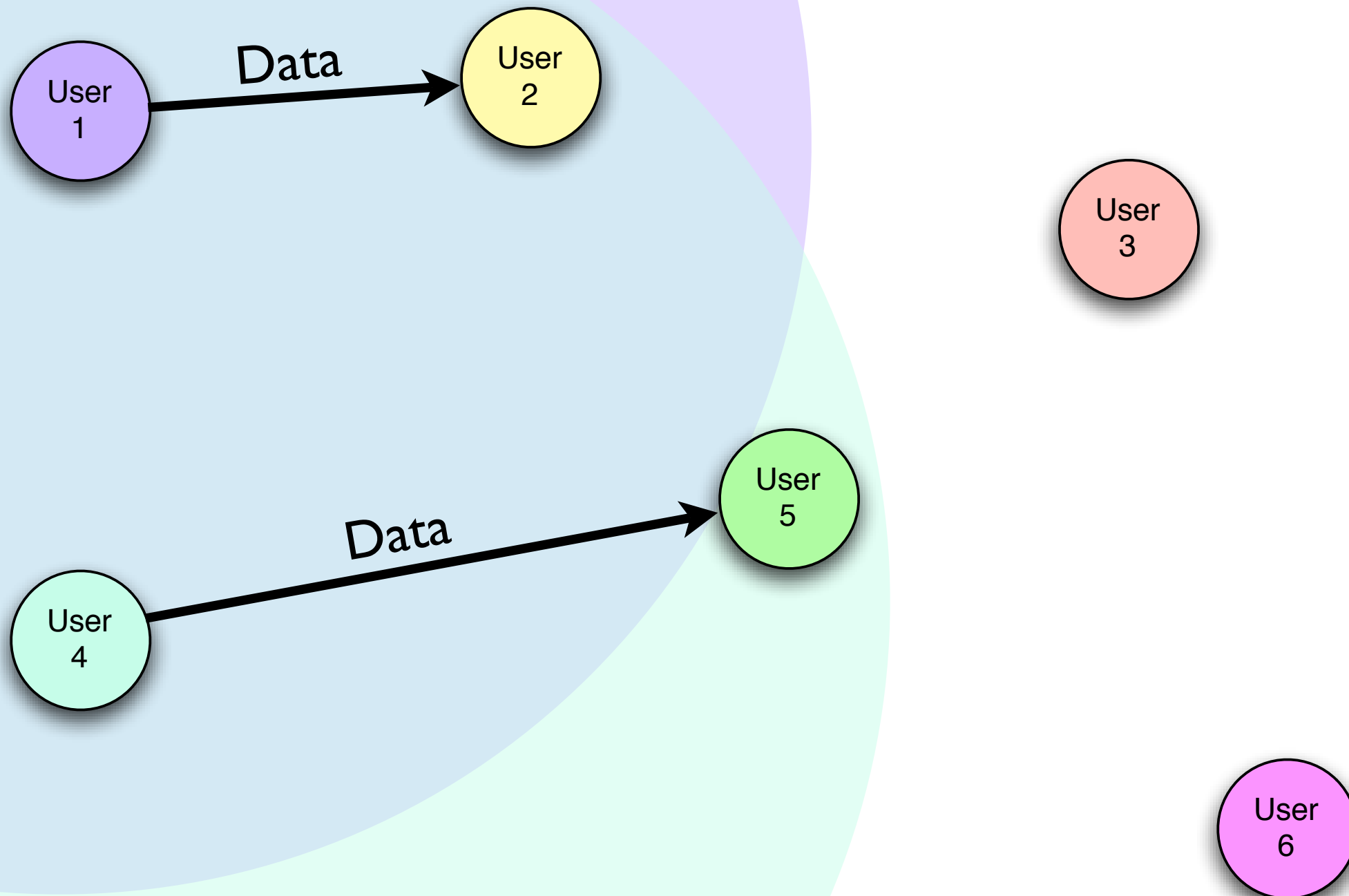
User  
4

User  
6

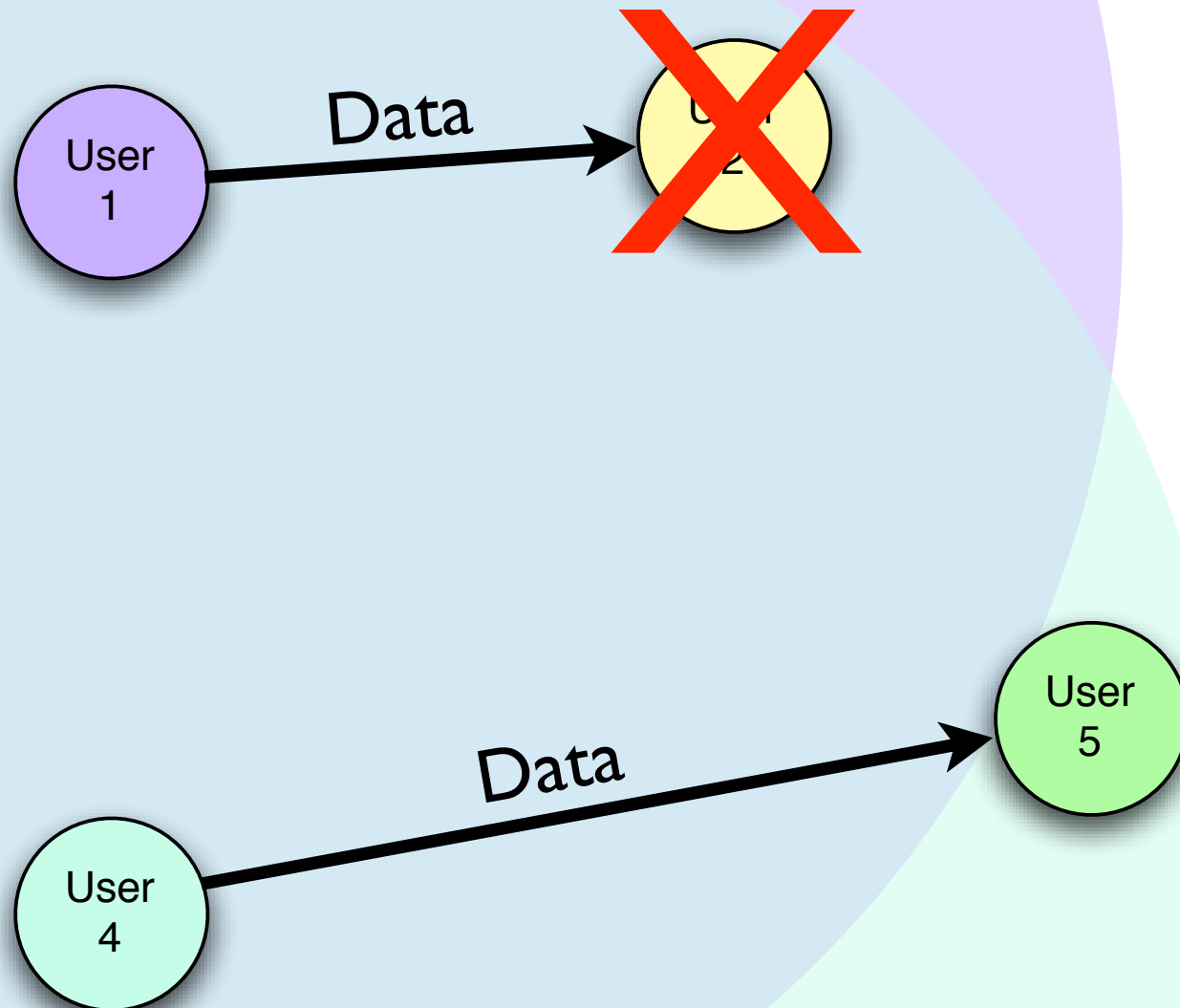
# What is a MAC?



# What is a MAC?

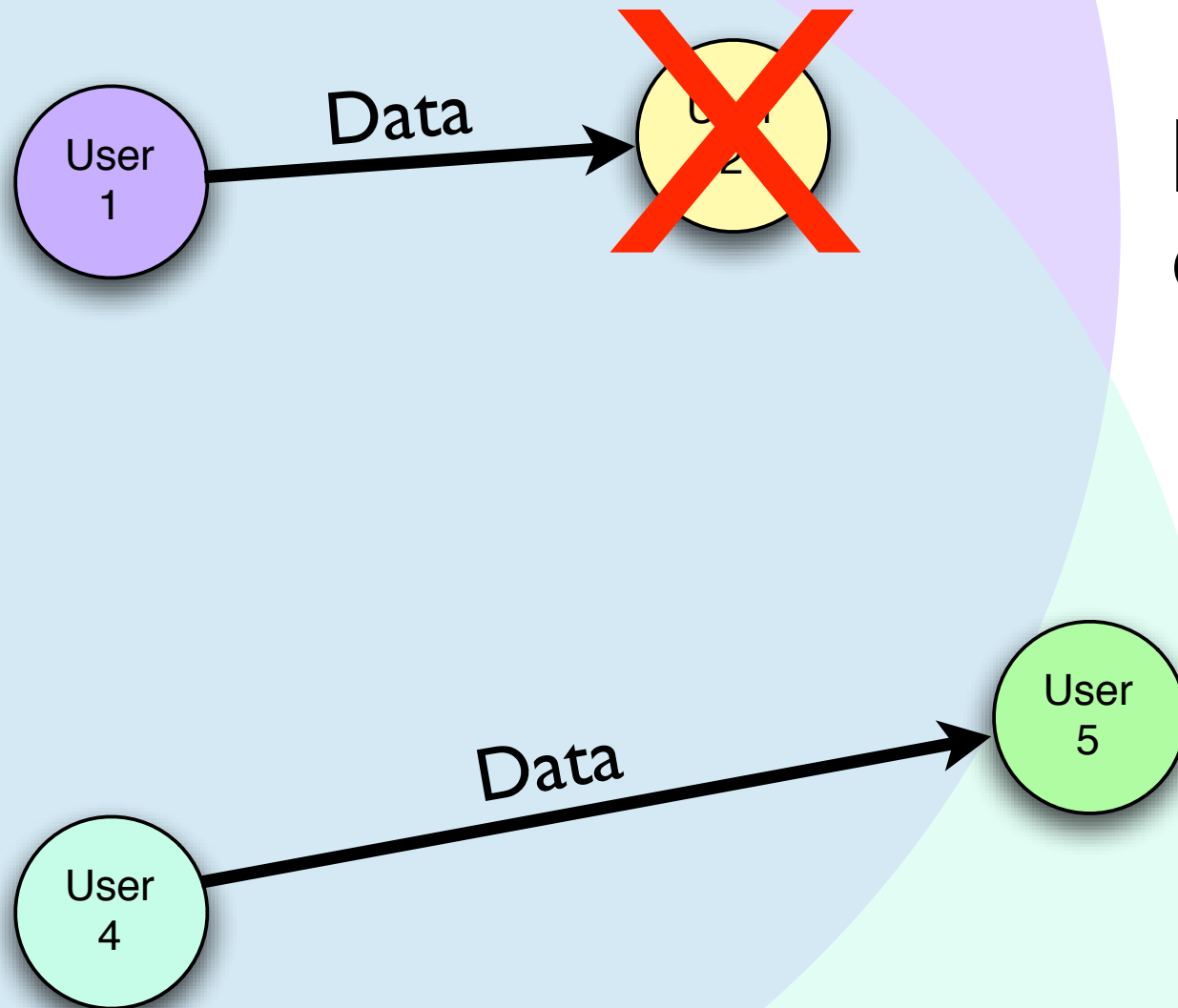


# What is a MAC?



# What is a MAC?

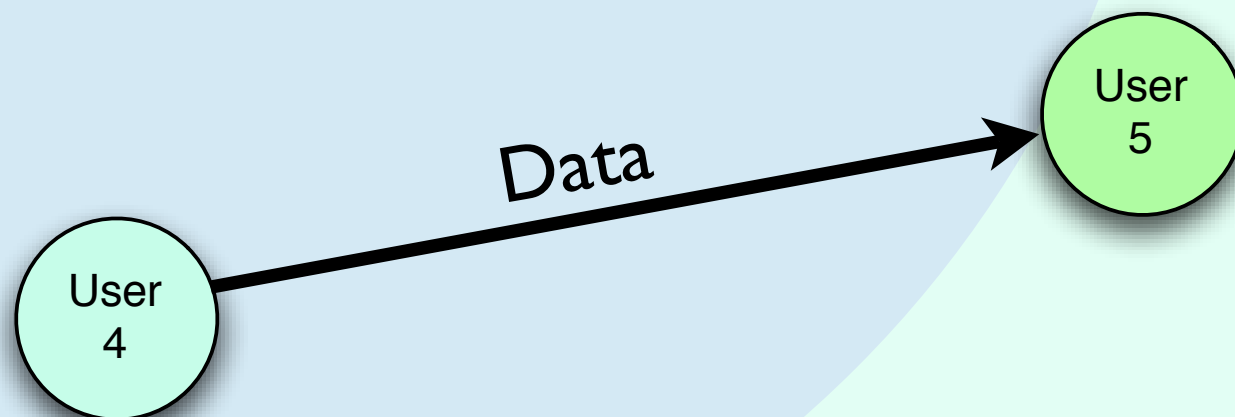
Received a jumbled packet... infer a packet collision



# What is a MAC?

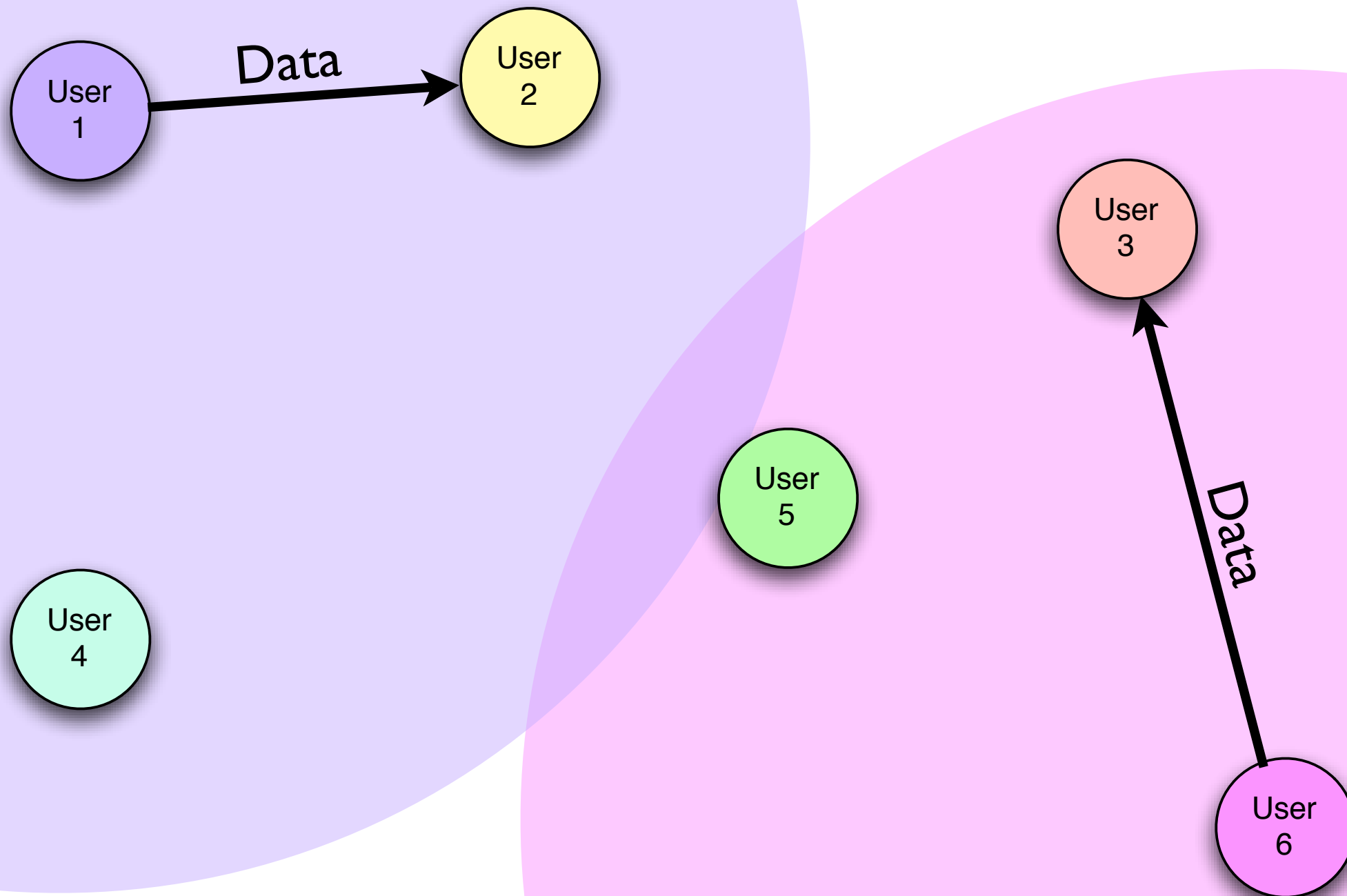


Received a jumbled packet... infer a packet collision

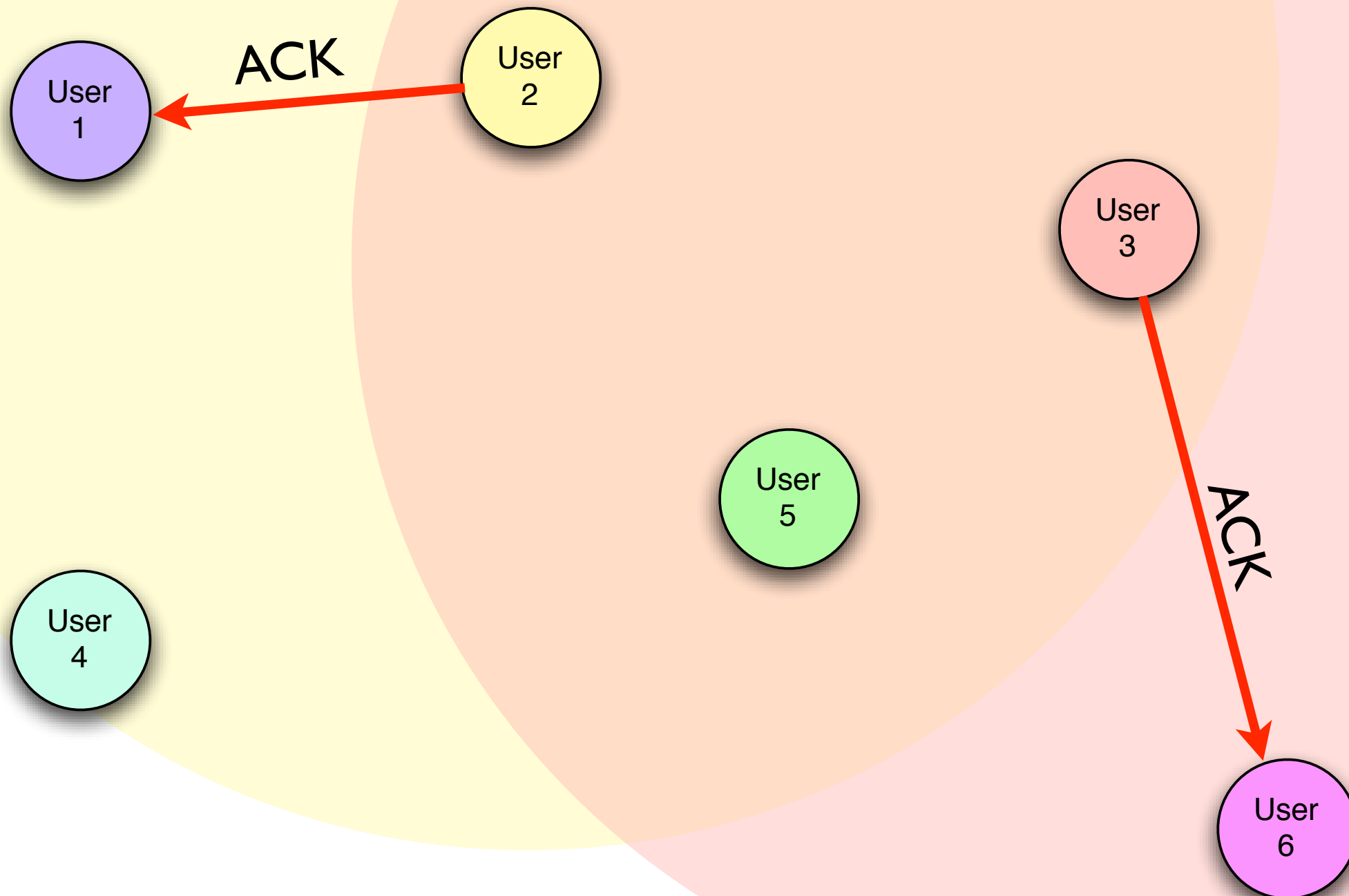


What if we ACK every transmit, and retransmit when we receive no ACK?

# What is a MAC?

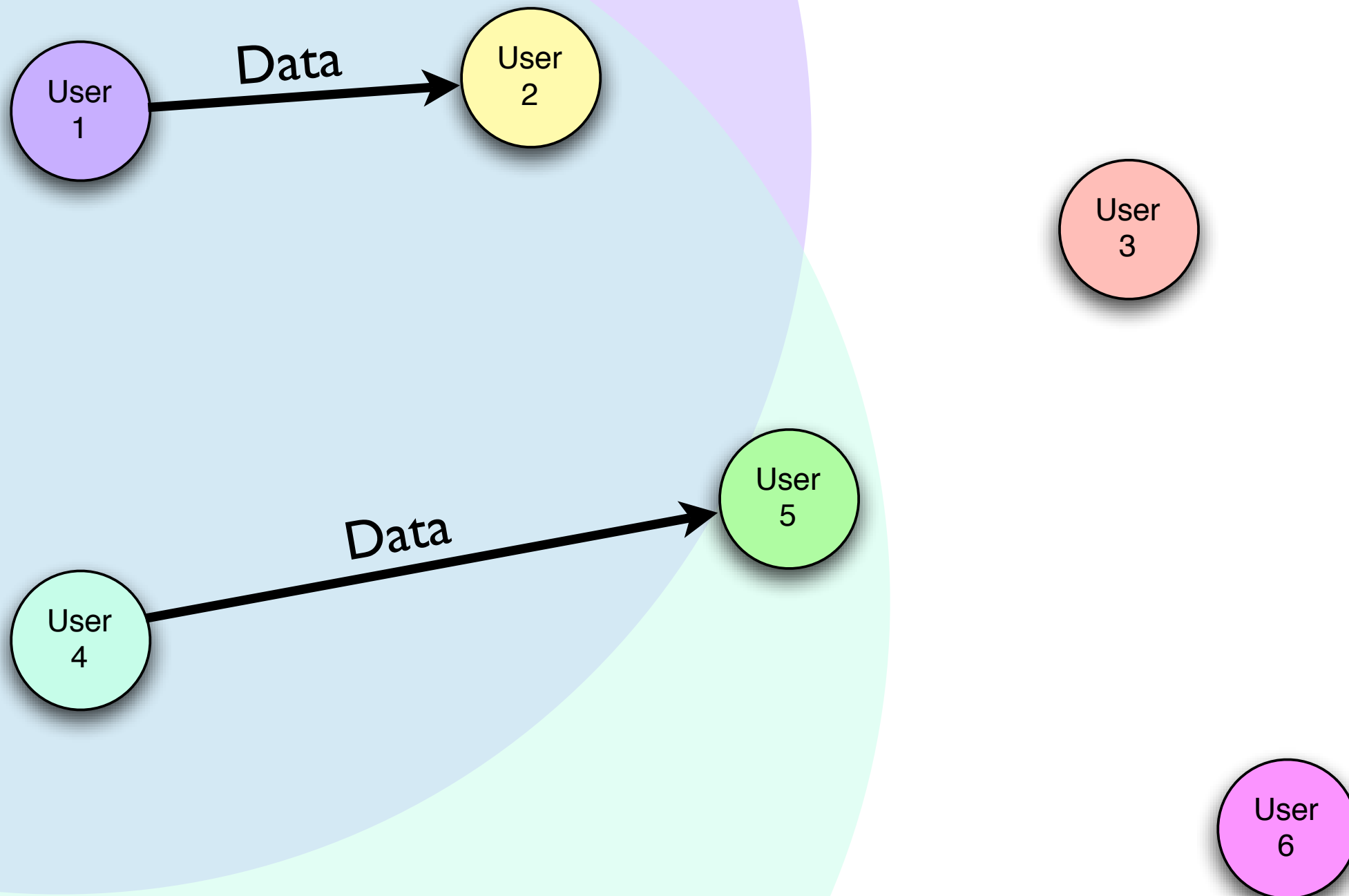


# What is a MAC?

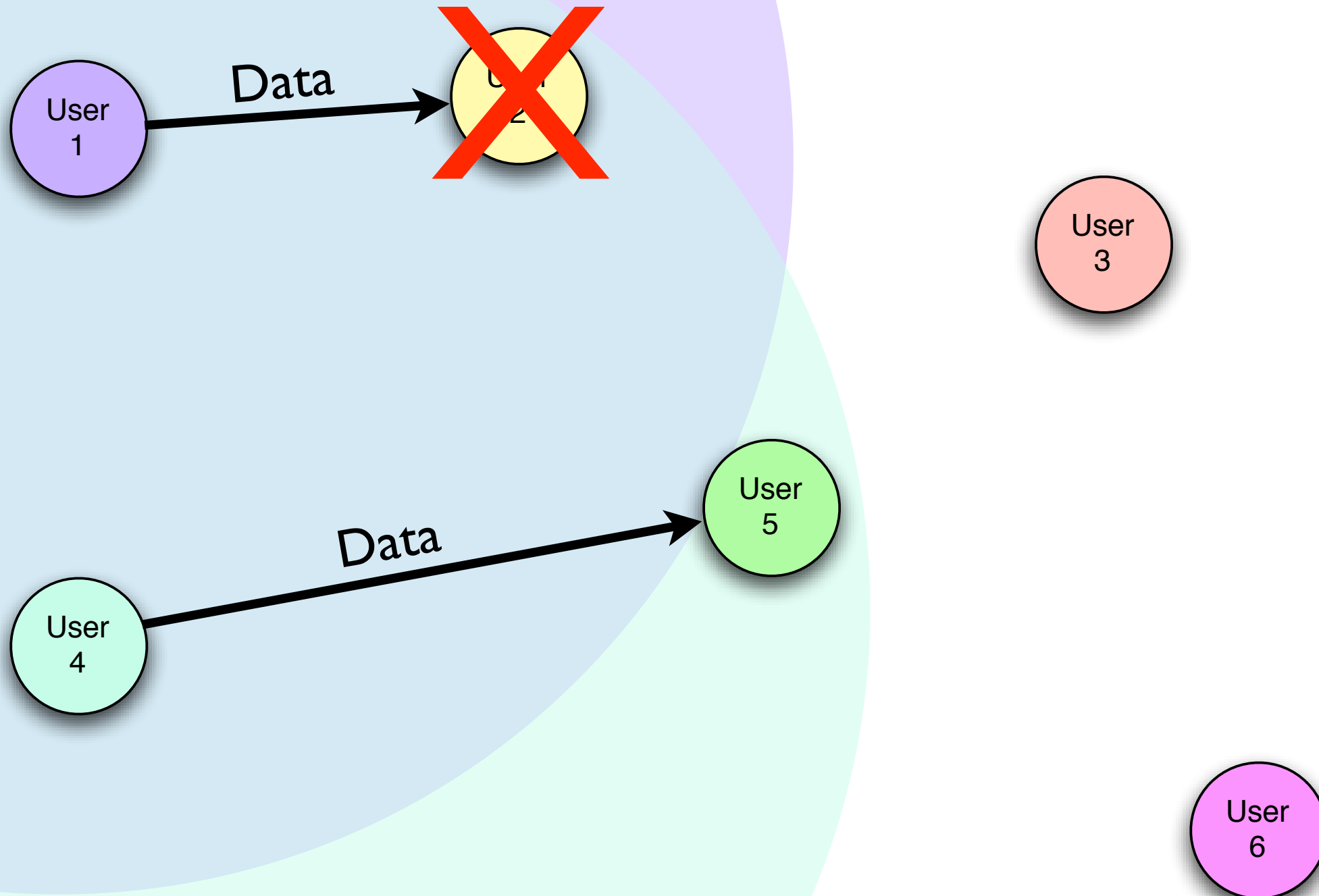




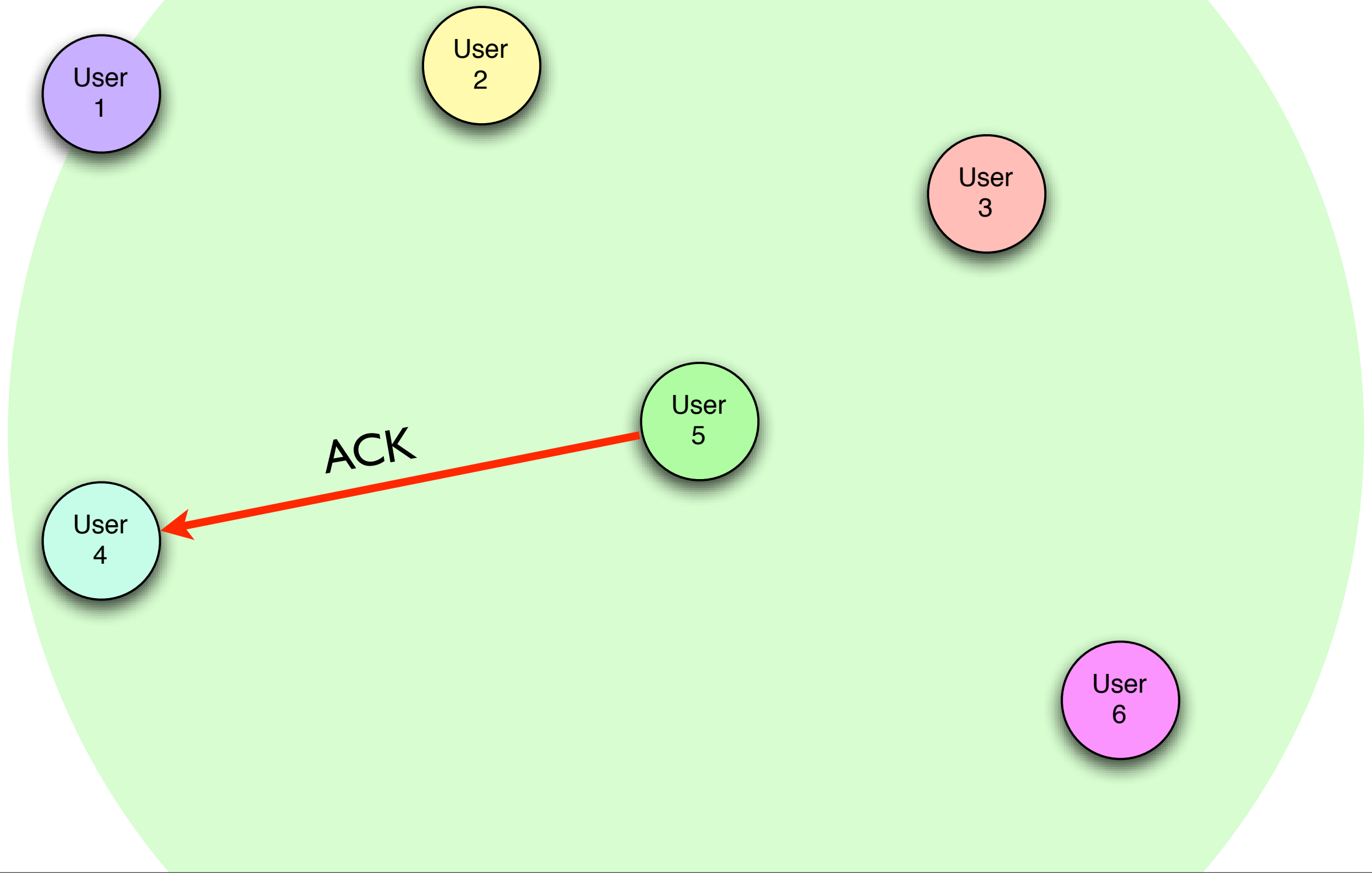
# What is a MAC?



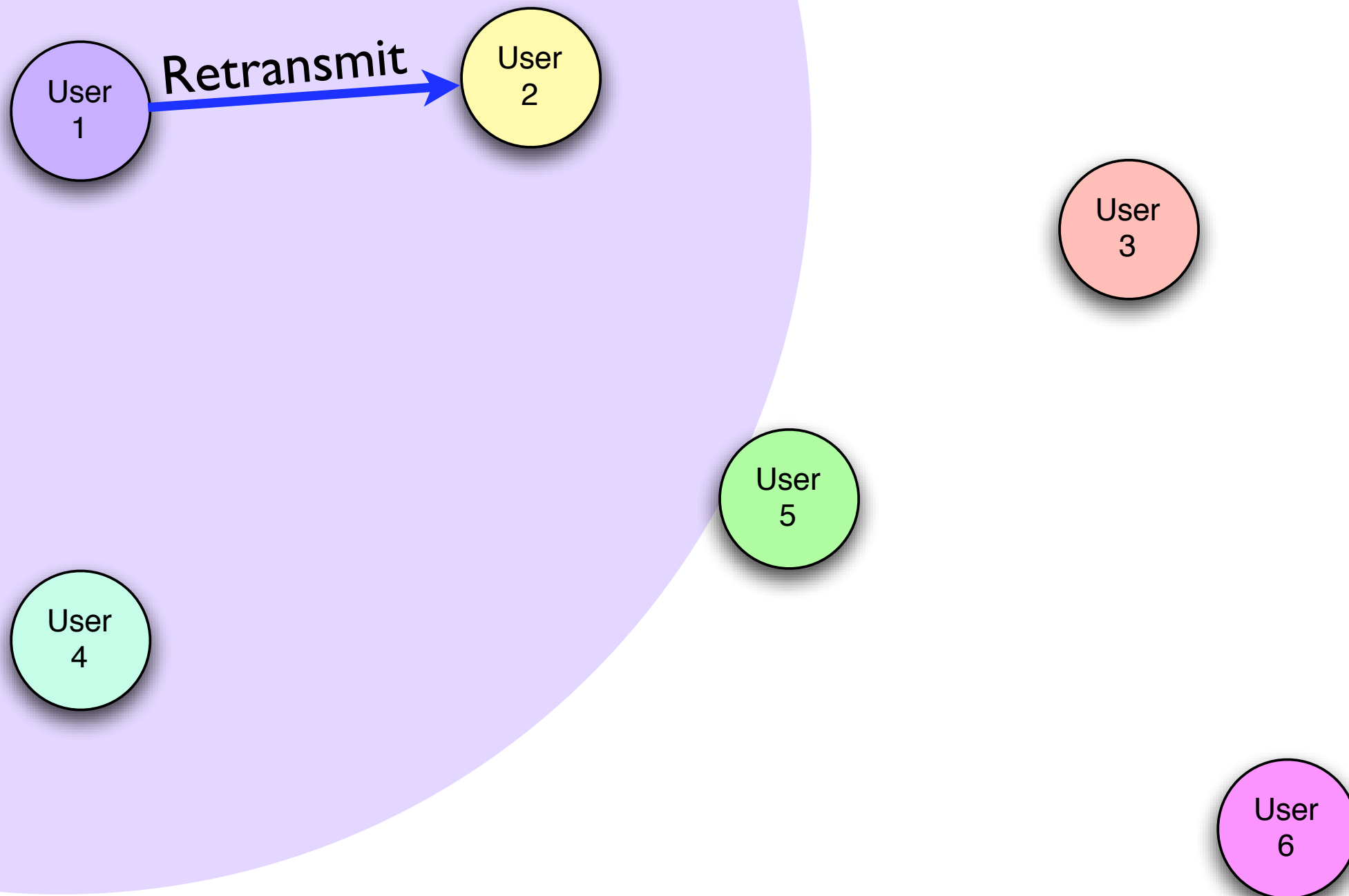
# What is a MAC?



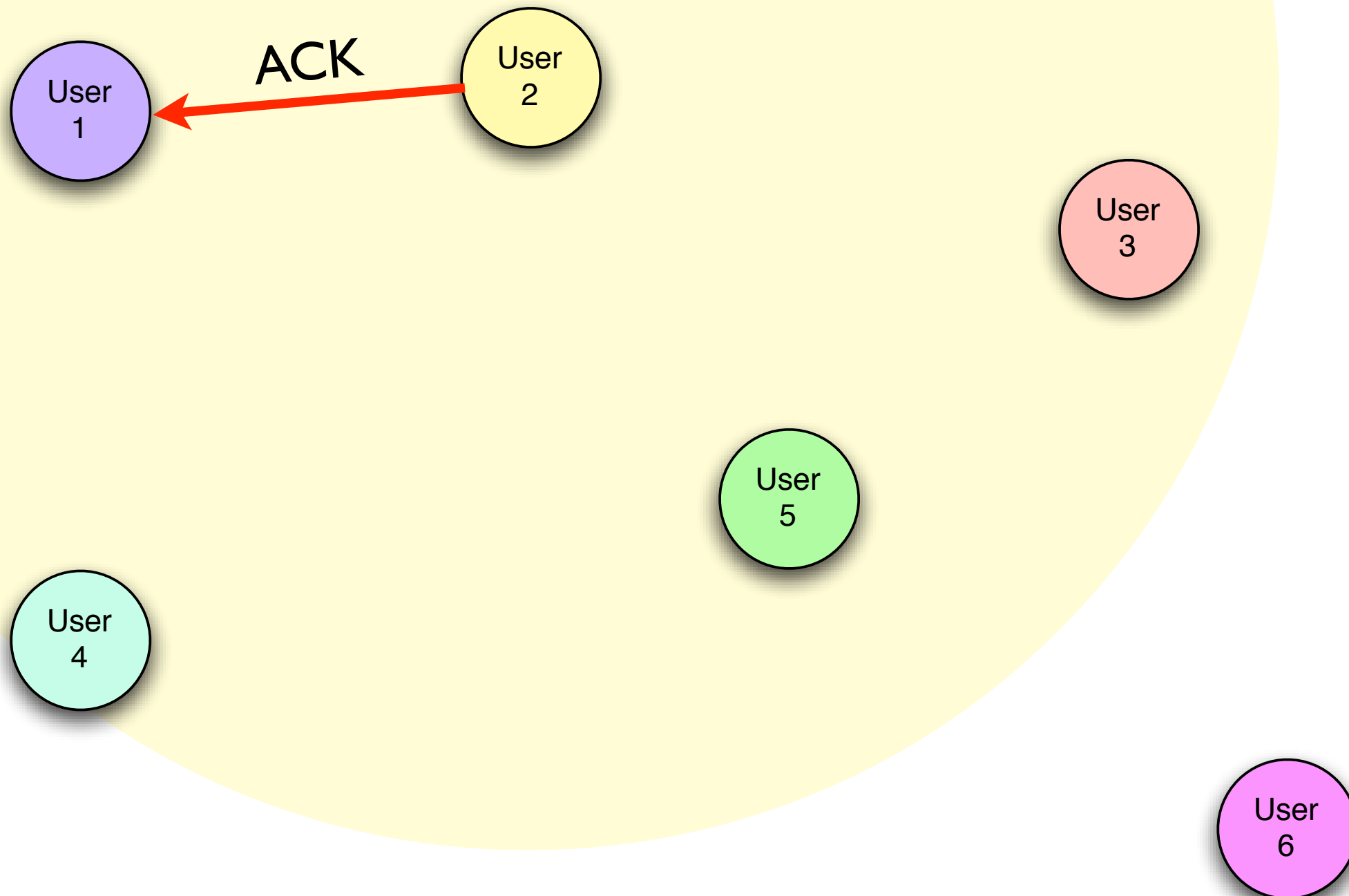
# What is a MAC?



# What is a MAC?

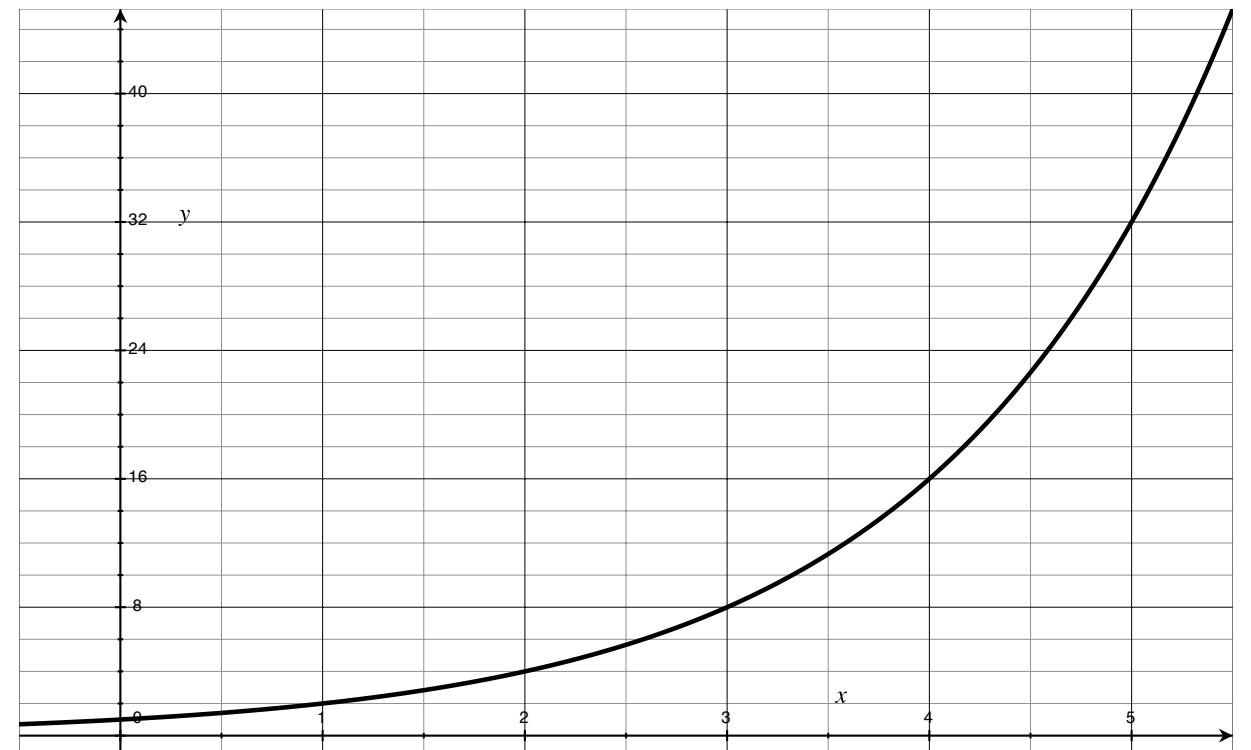


# What is a MAC?



# Random Backoffs

- **PROBLEM:**  
Retransmissions can collide *ad infinitum!*
- **SOLUTION:** Wait a random amount of time before a retransmit



Contention Window  
increases over time

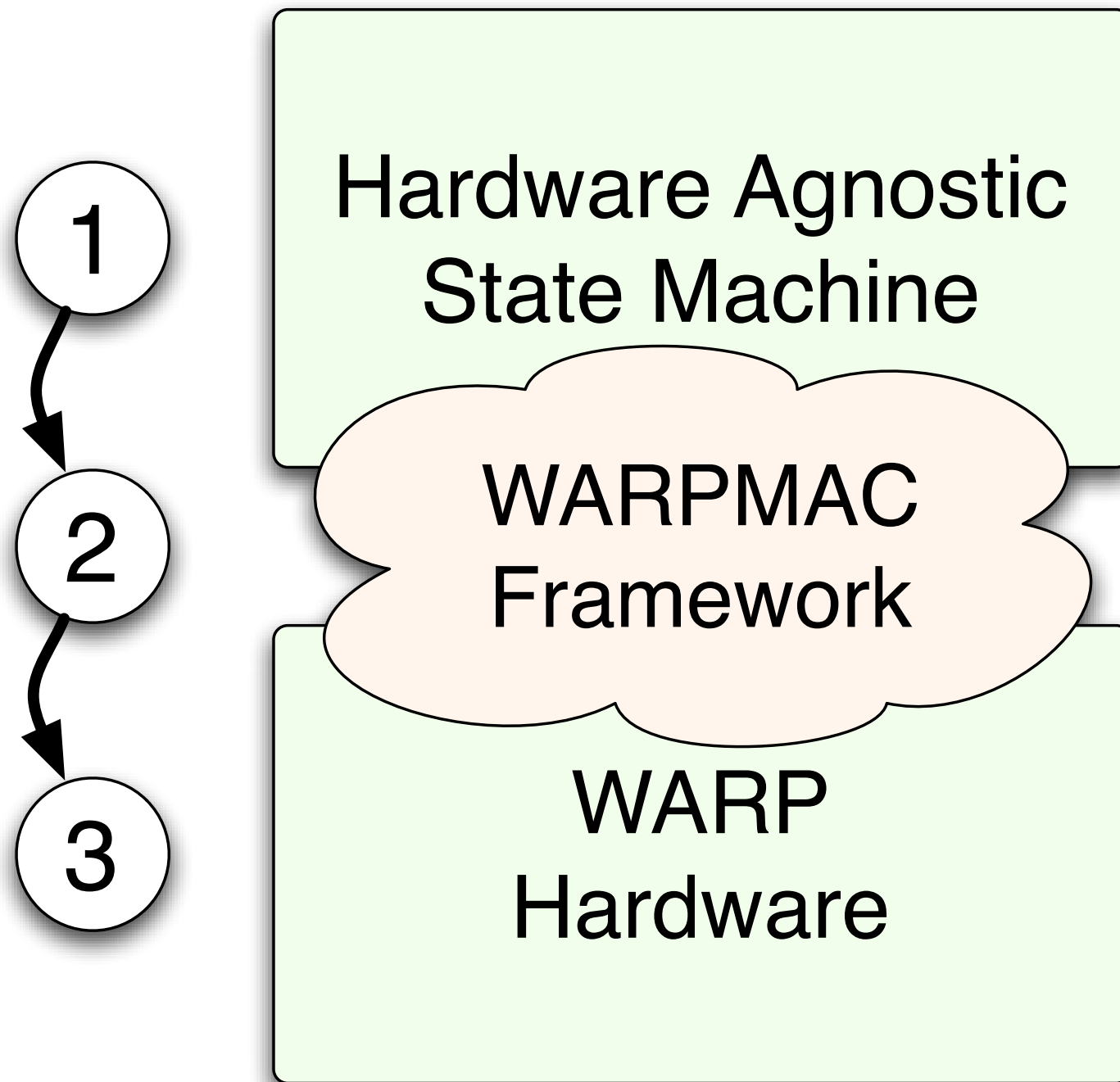
# Important Extensions

- 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

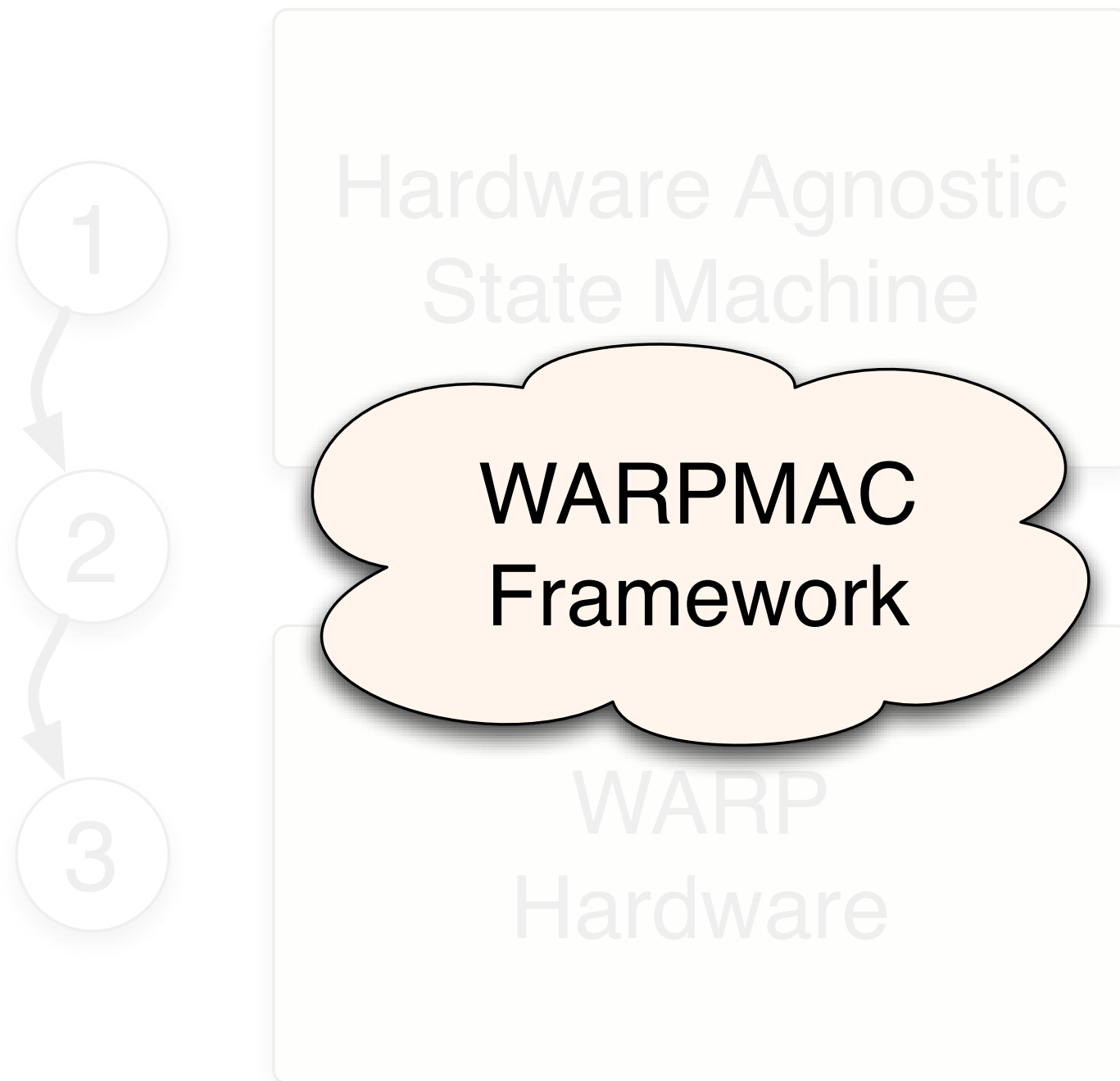


# Design Realization



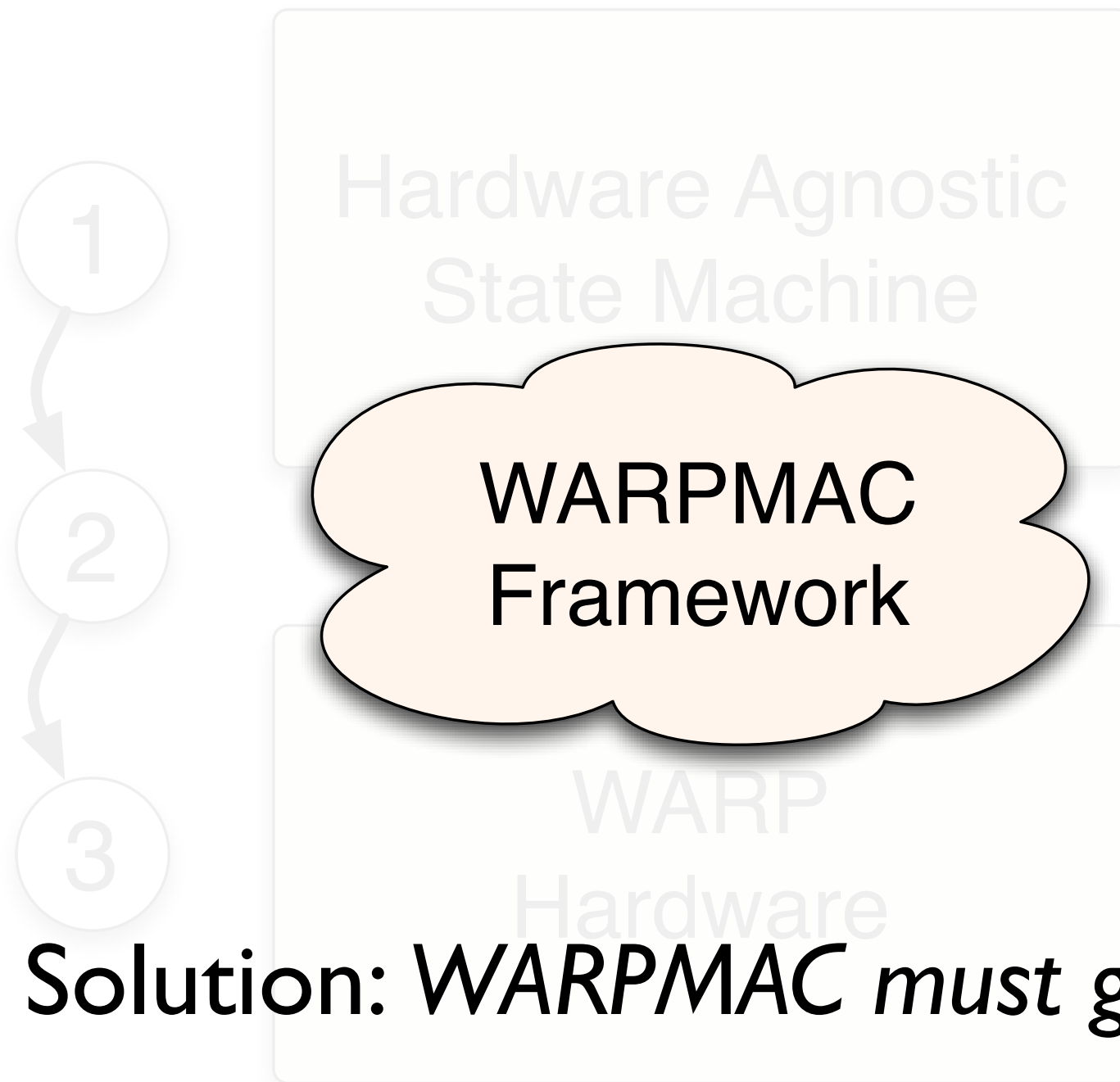
- Program high-level MAC behavior independent of hardware
- Use the WARPMAC framework to stitch the MAC to hardware

# Design Realization



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

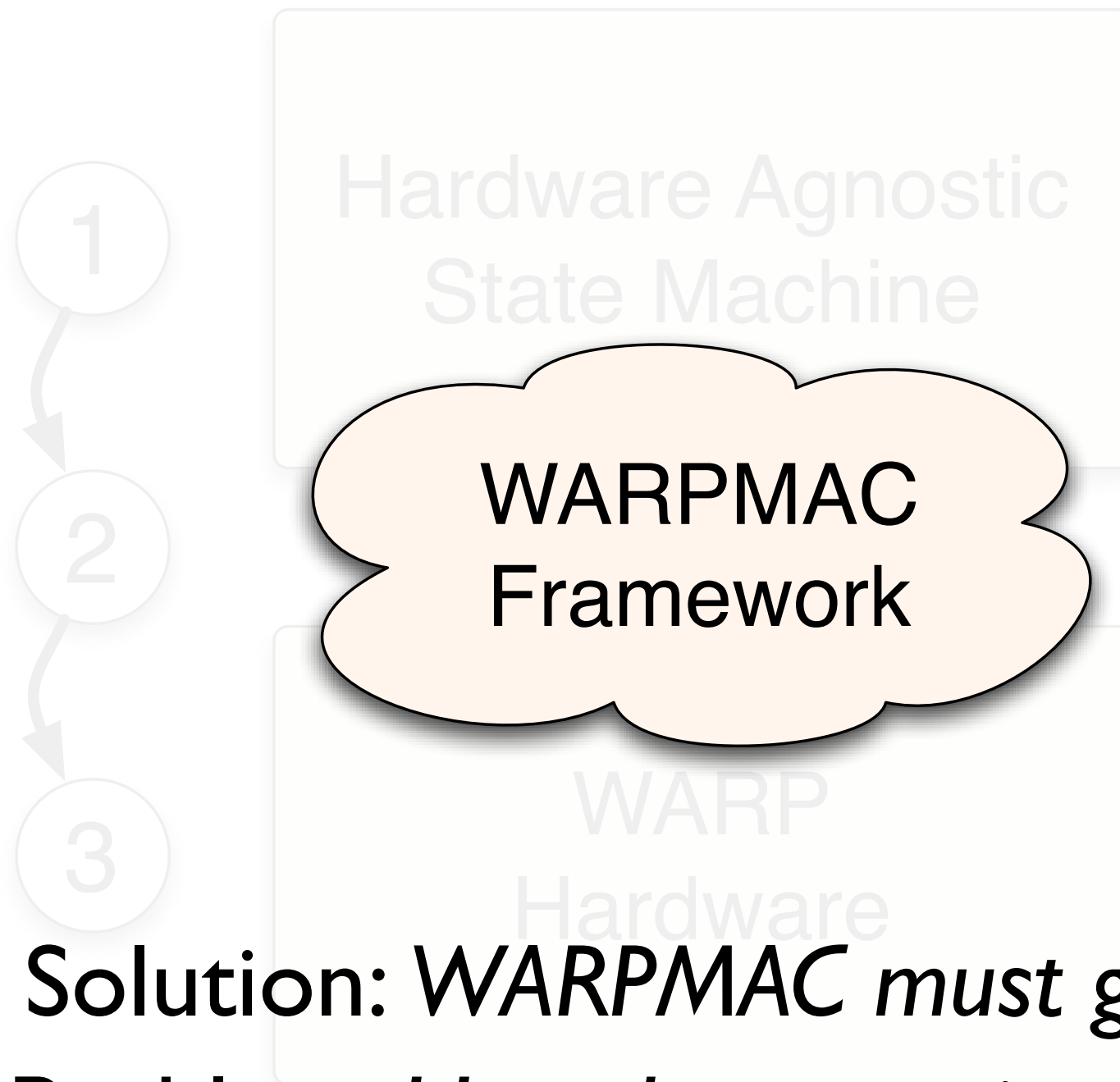
# Design Realization



- 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 v10



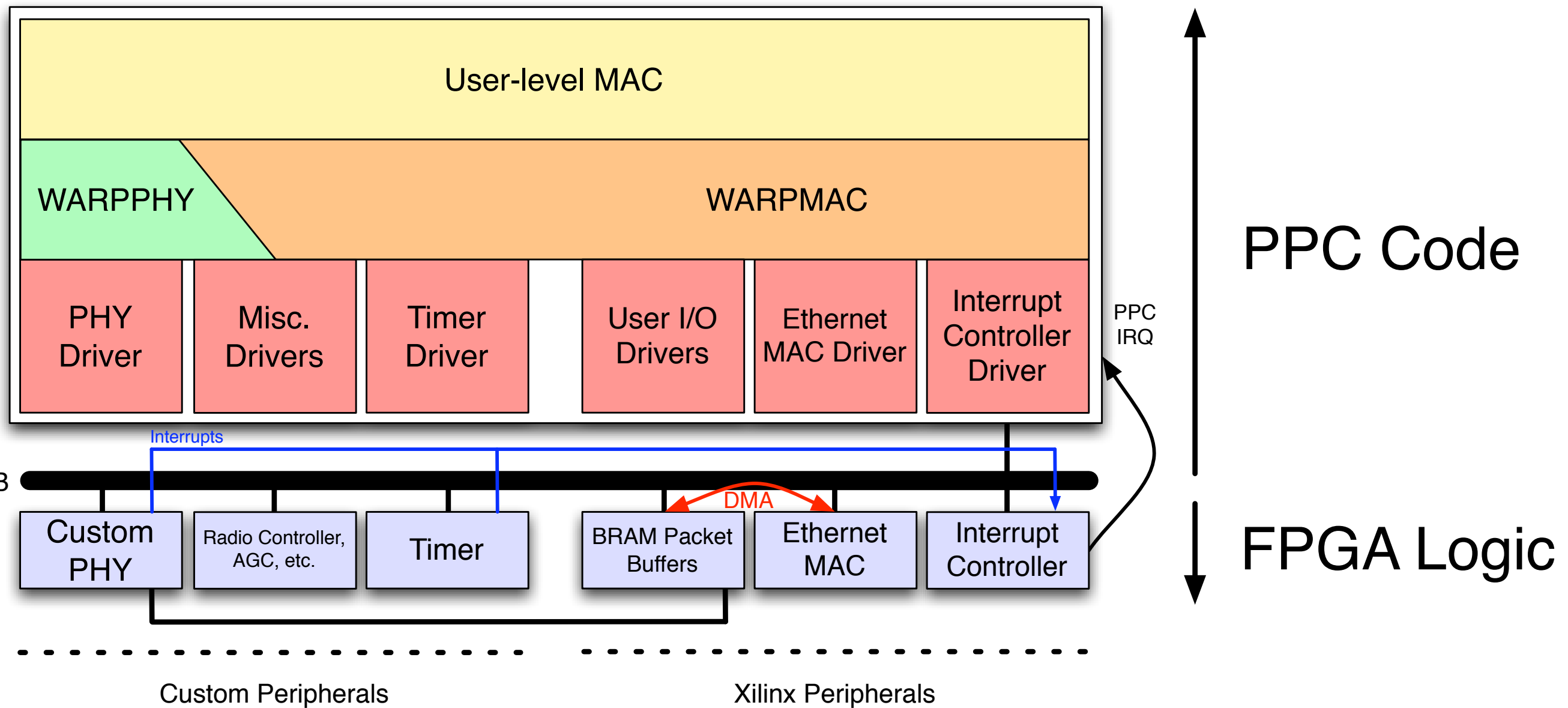
# Reference Designs

- Snapshots of the WARP repository
- Free, open-source releases at regular intervals
- Today's exercises are on nearly-released Reference Design v10
- 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 v10
- 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





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



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



User Code

WARPMAC

WARPPHY

Drivers



**Mostly PHY  
agnostic**

User Code

WARPMAC

**Completely PHY  
dependent**

WARPPHY

Drivers





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 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



User Code

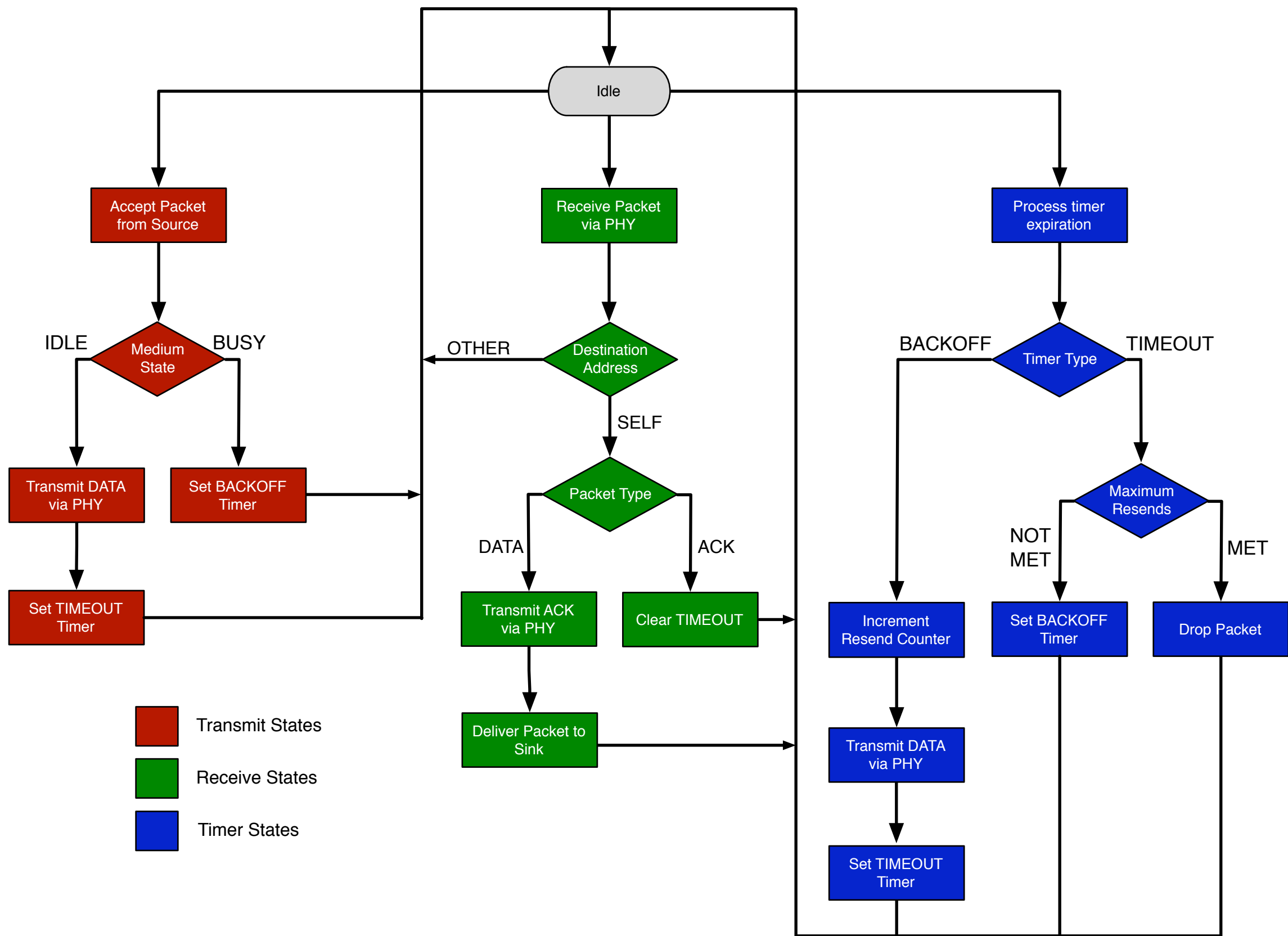
WARPMAC

WARPPHY

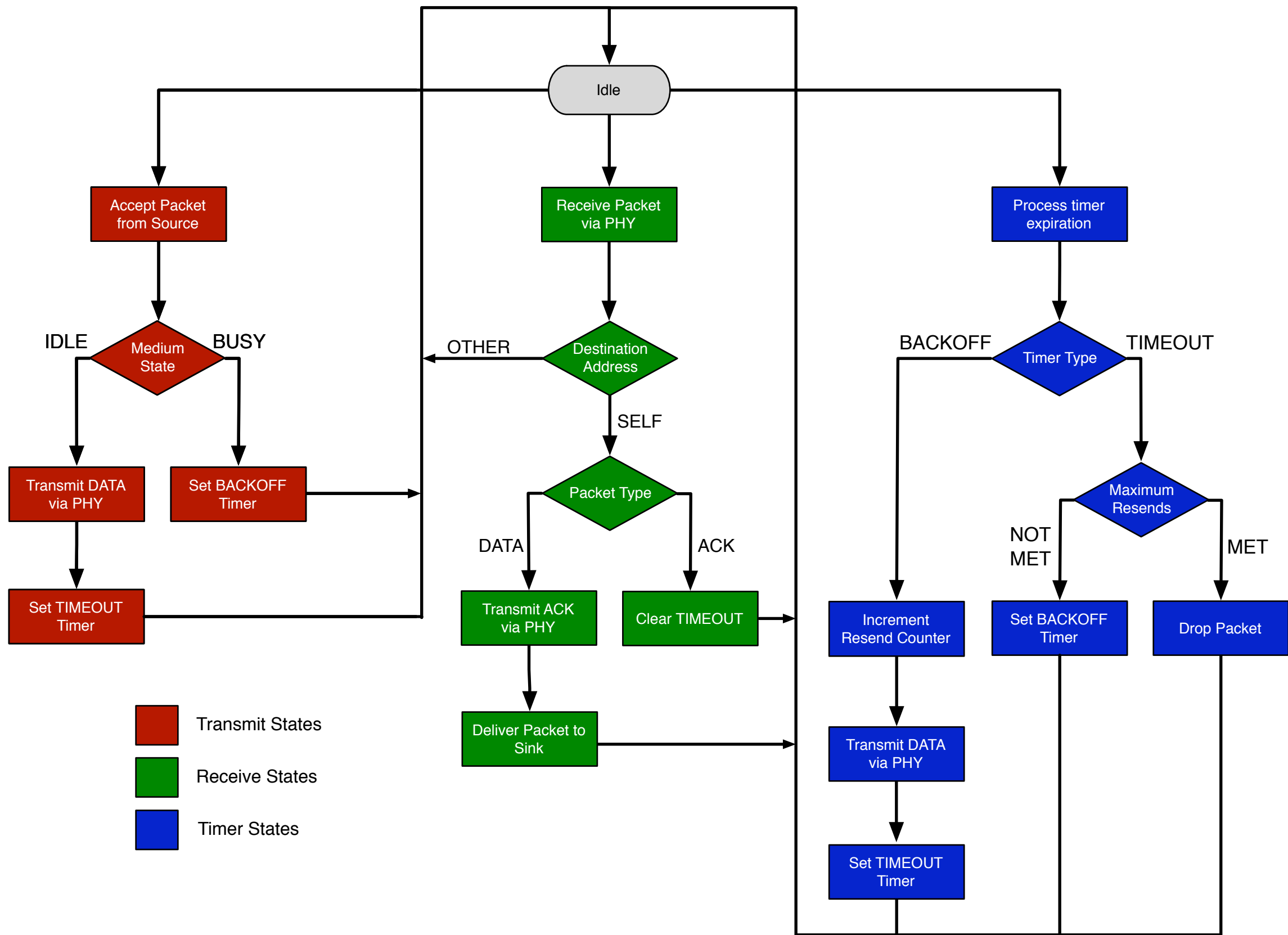
Drivers

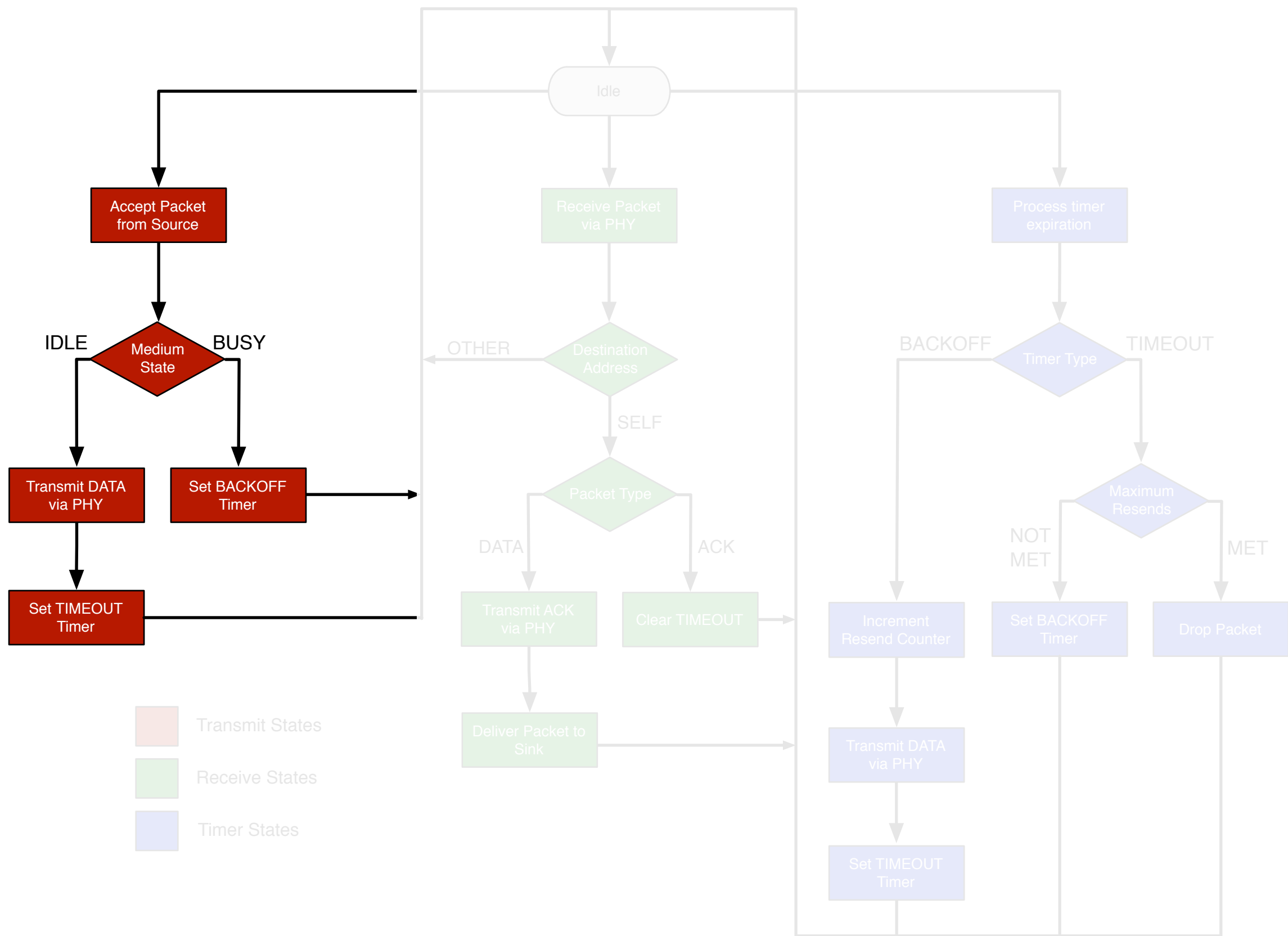
# An example: CSMA

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

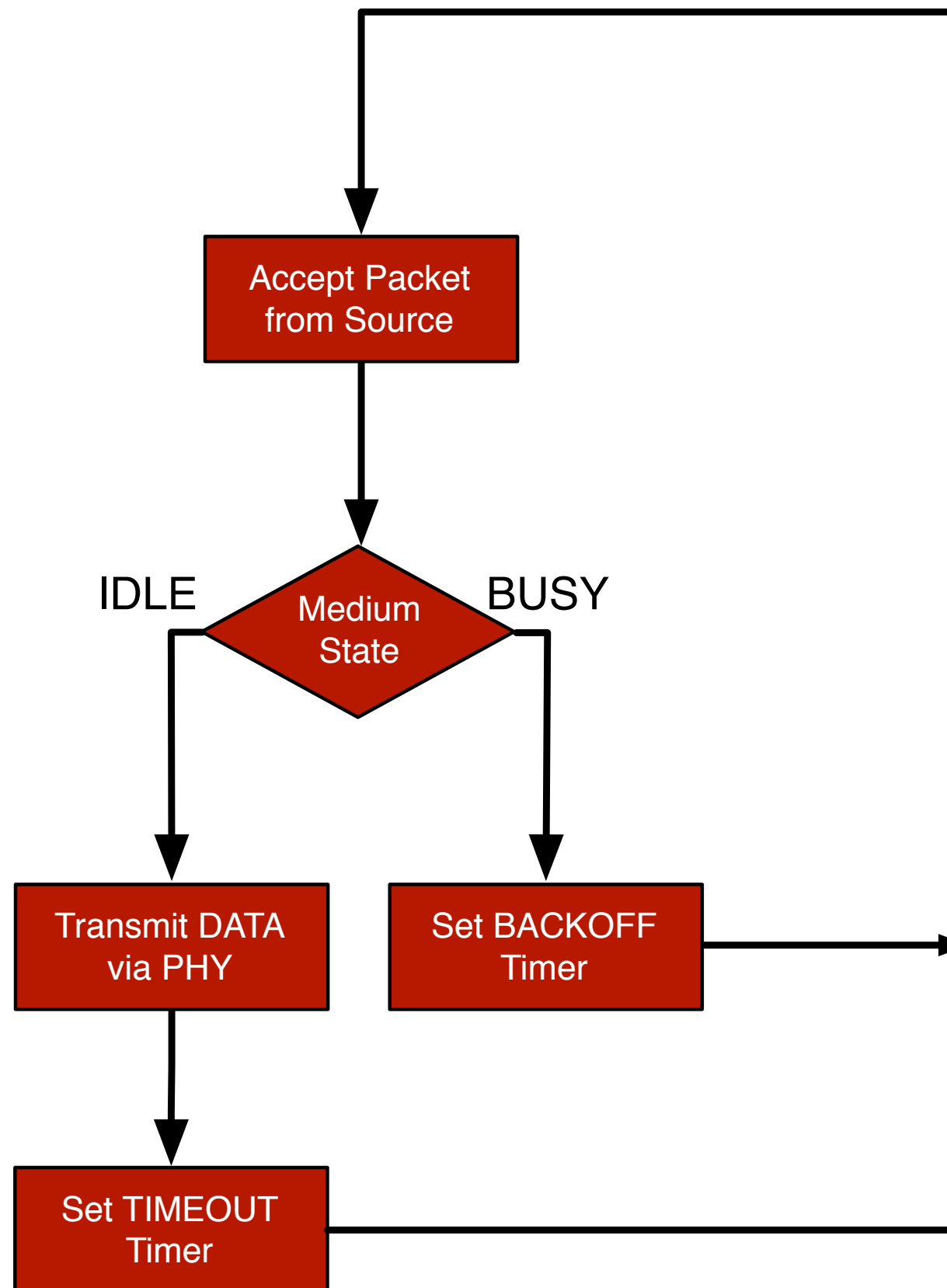








# Transmit States



# 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

}

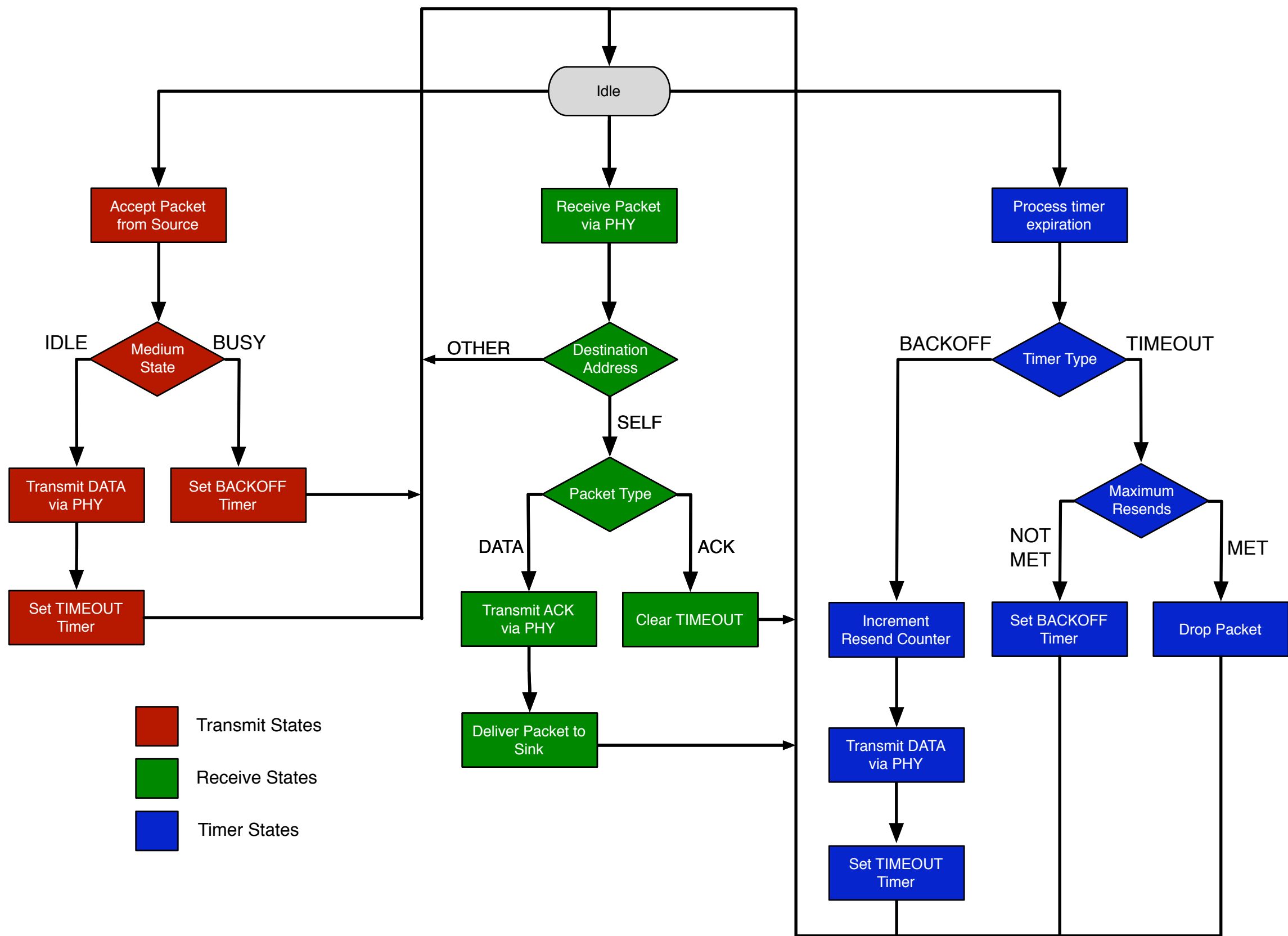
***If medium is busy {***

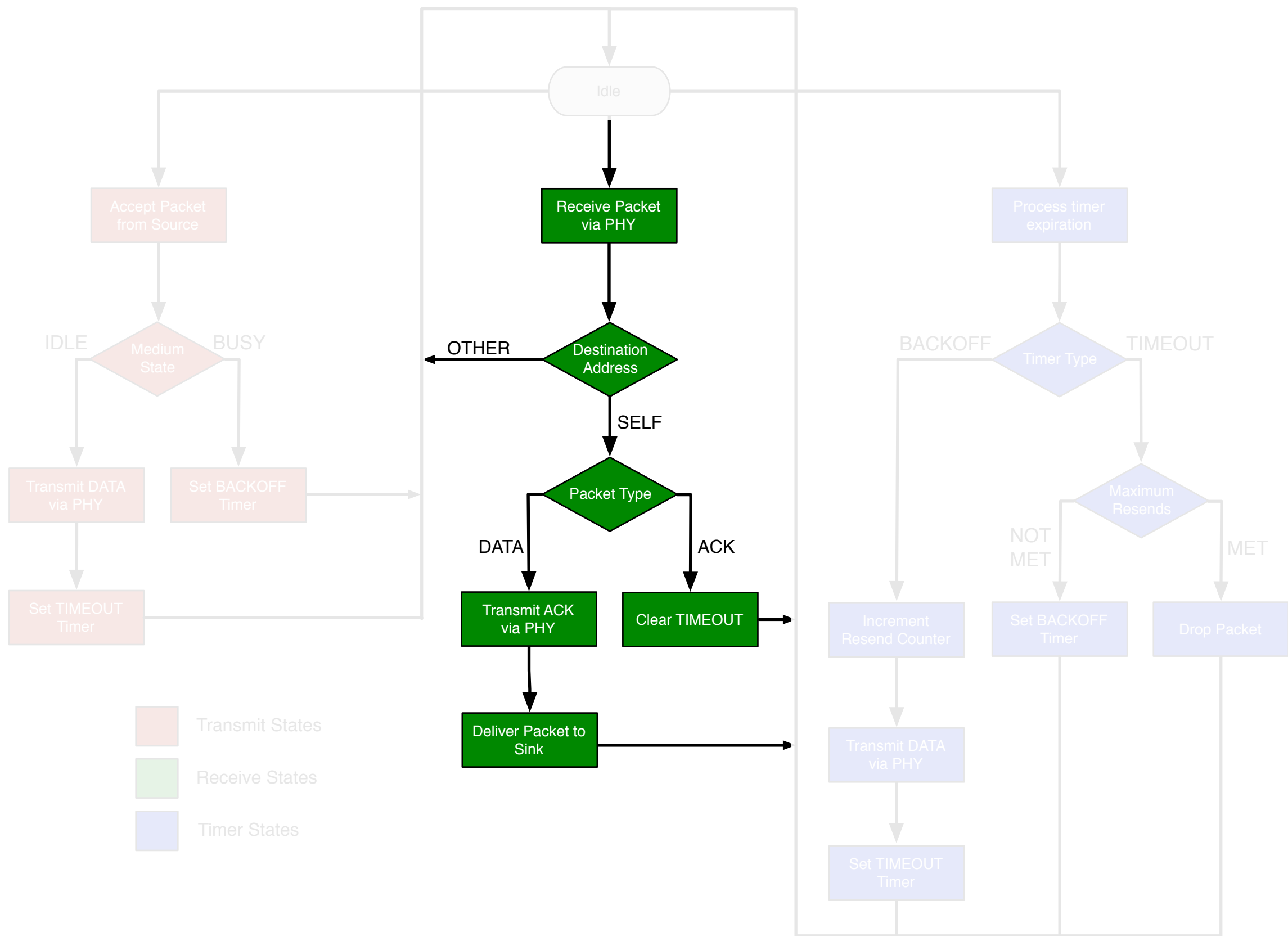
- Starts a backoff timer

}

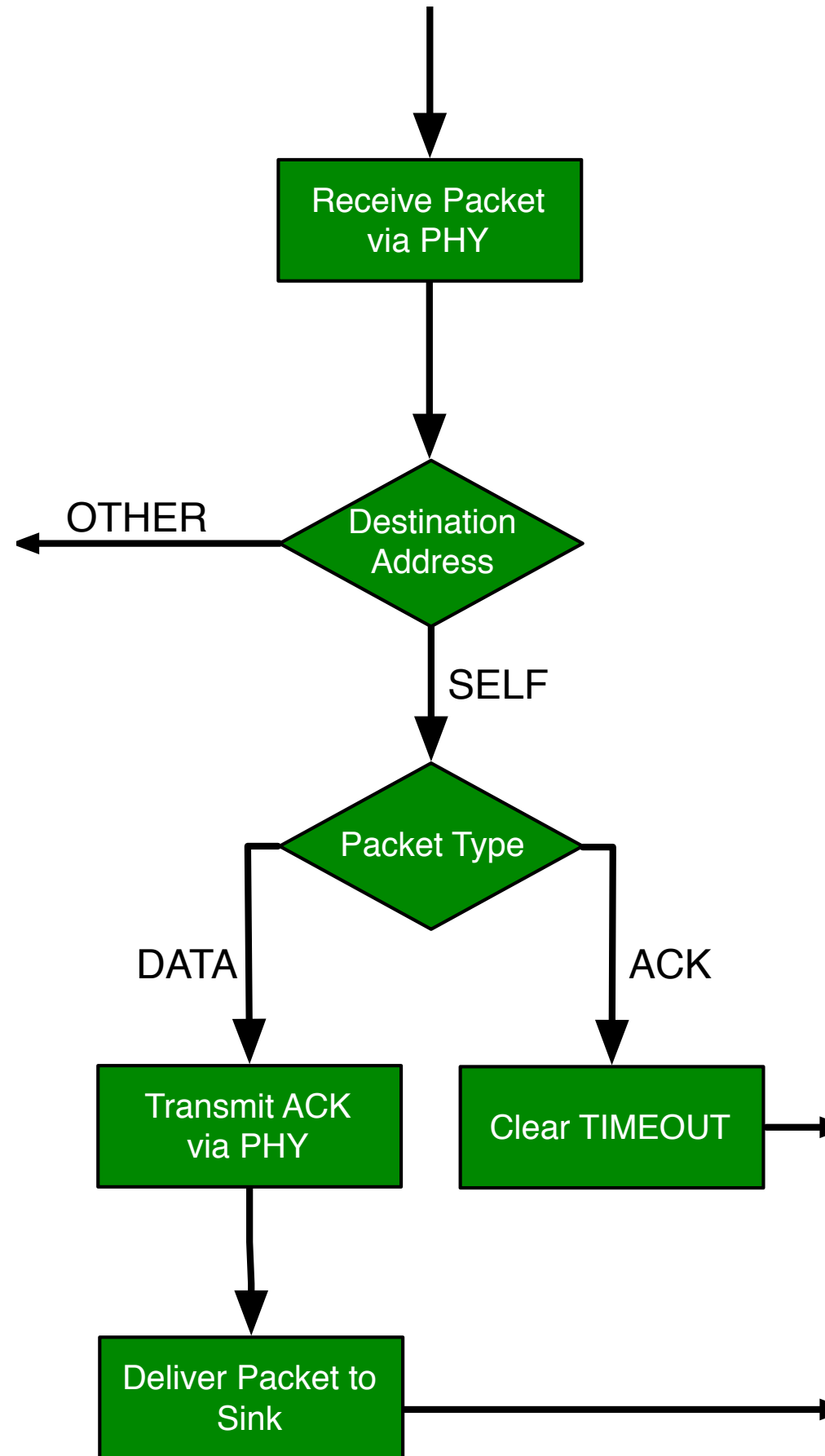
- Clears EMAC and interrupt controller







# 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





# 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

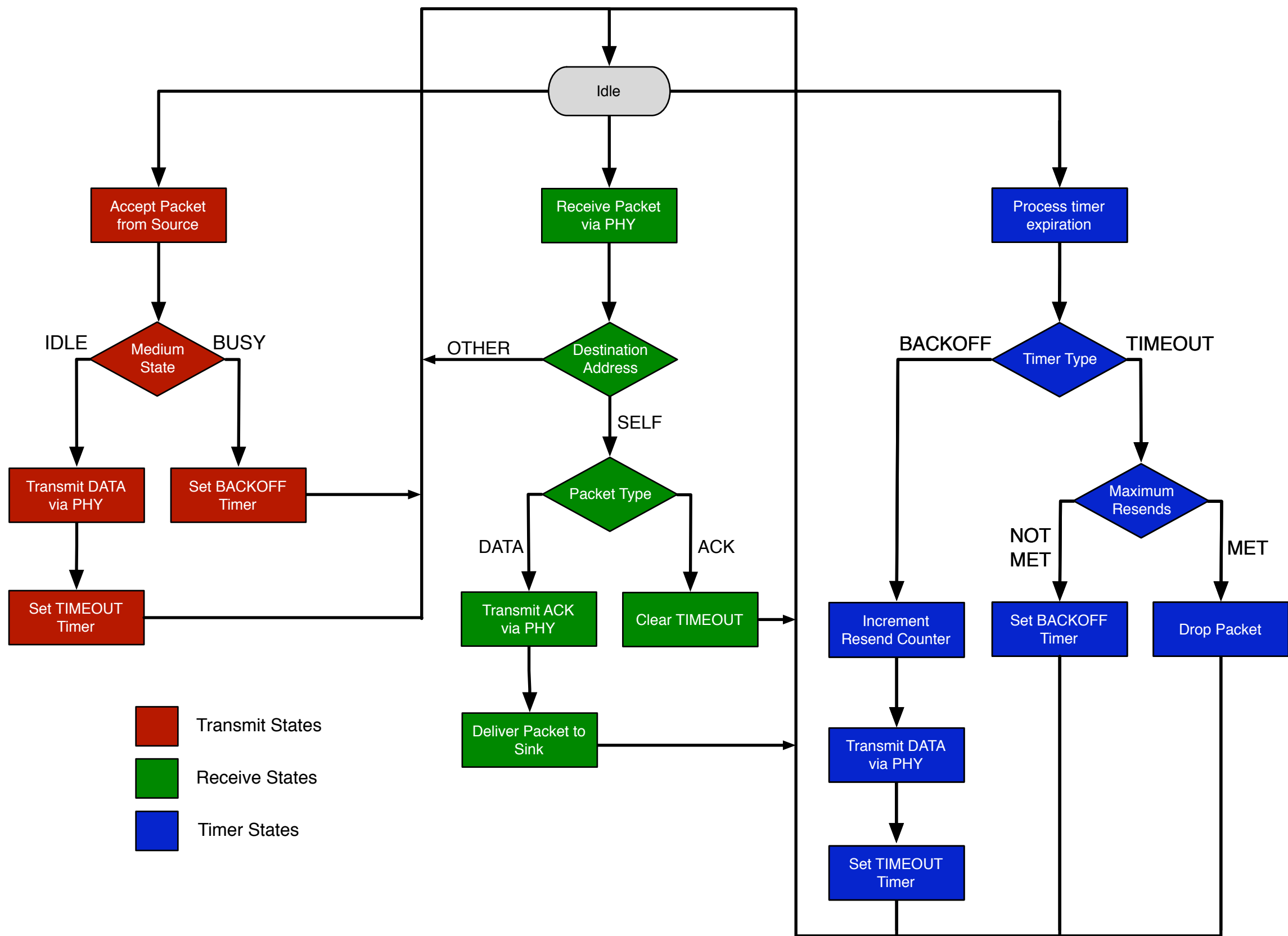
}

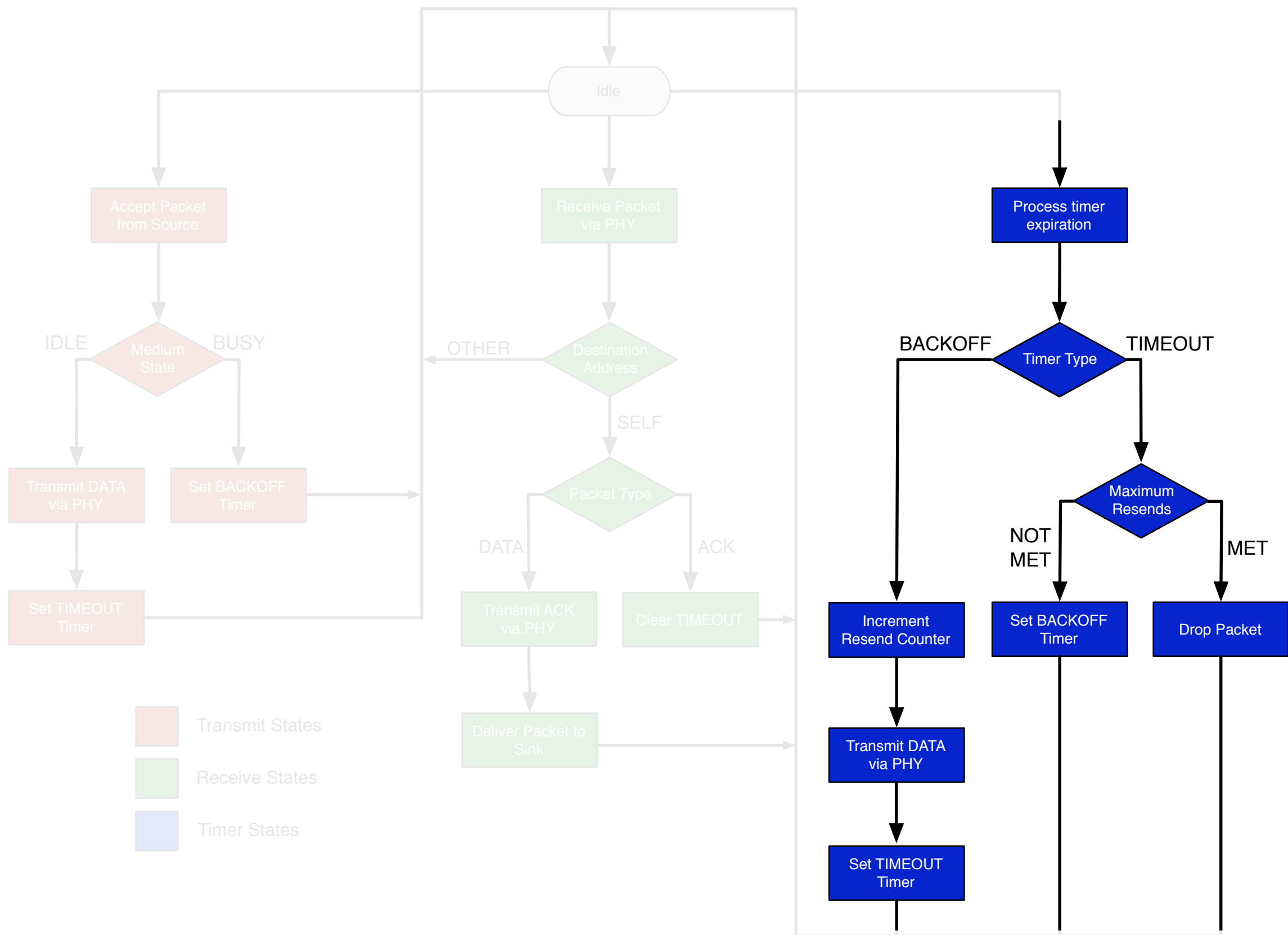
- Clears interrupts in PHY and interrupt controller

Fast Turn-Around Time  
(TAT)

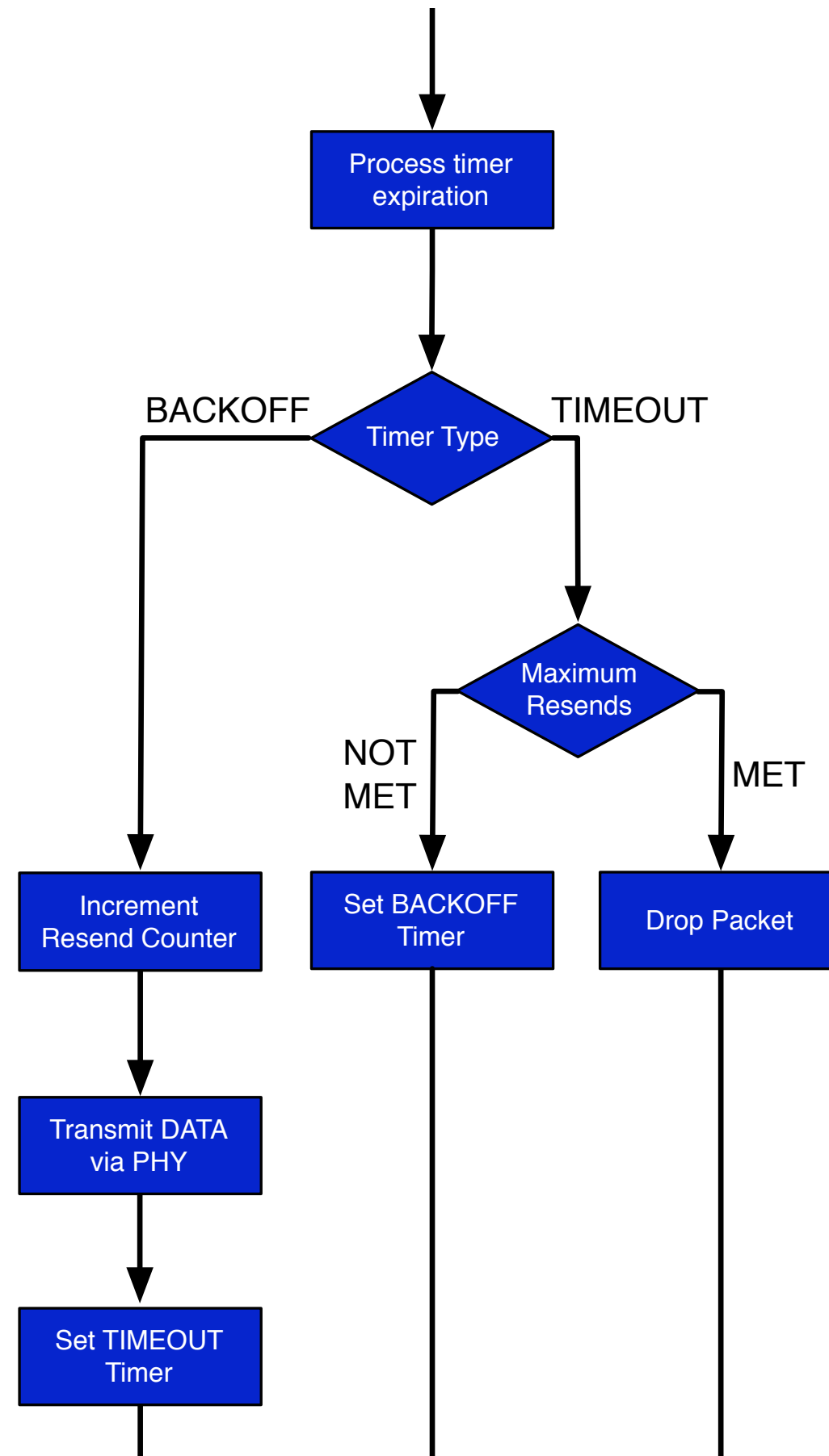
 WARPMAC

 User-Code





# 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

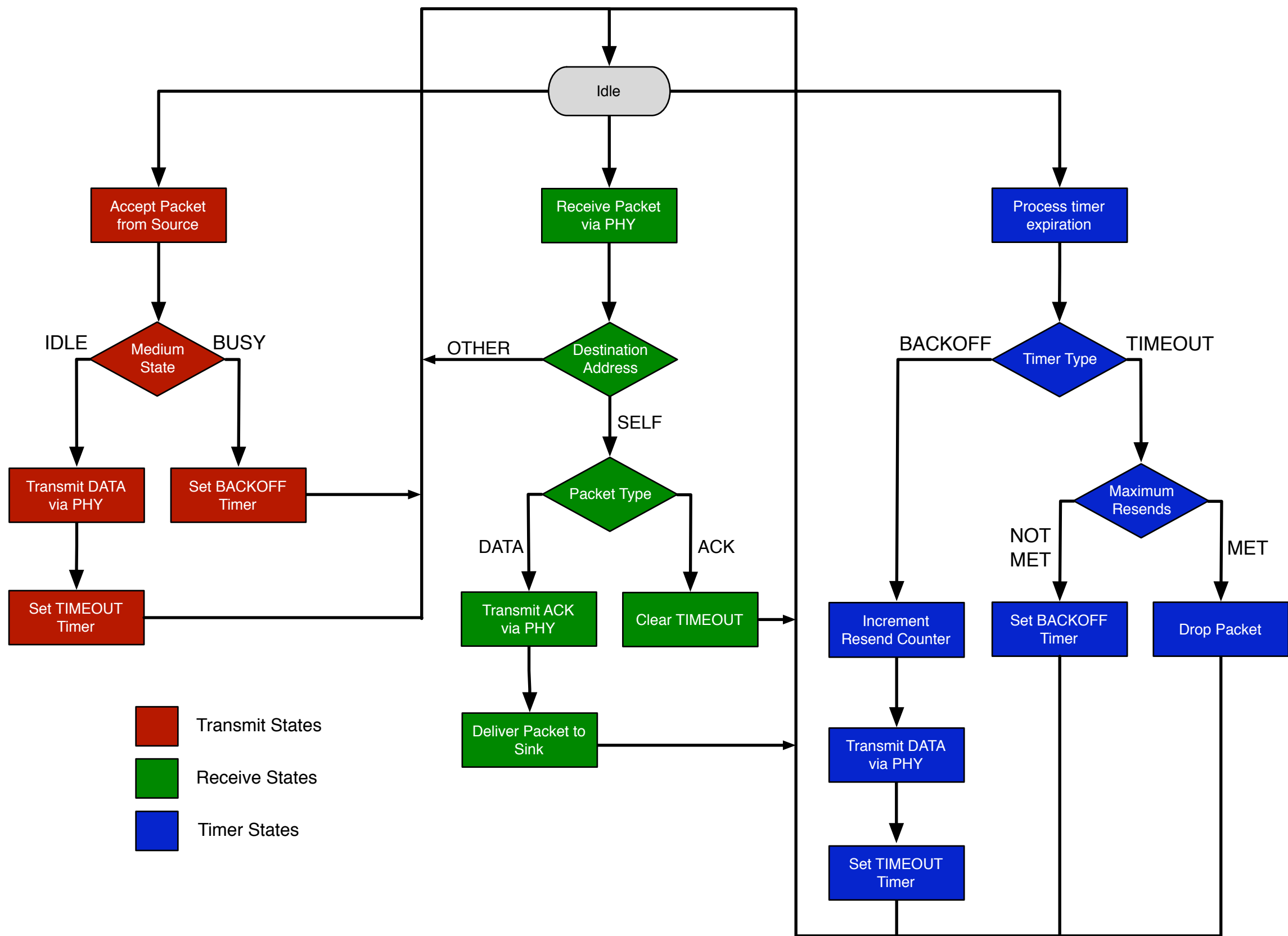
- Starts a timeout timer

}

- Clears interrupts in timer and interrupt controller

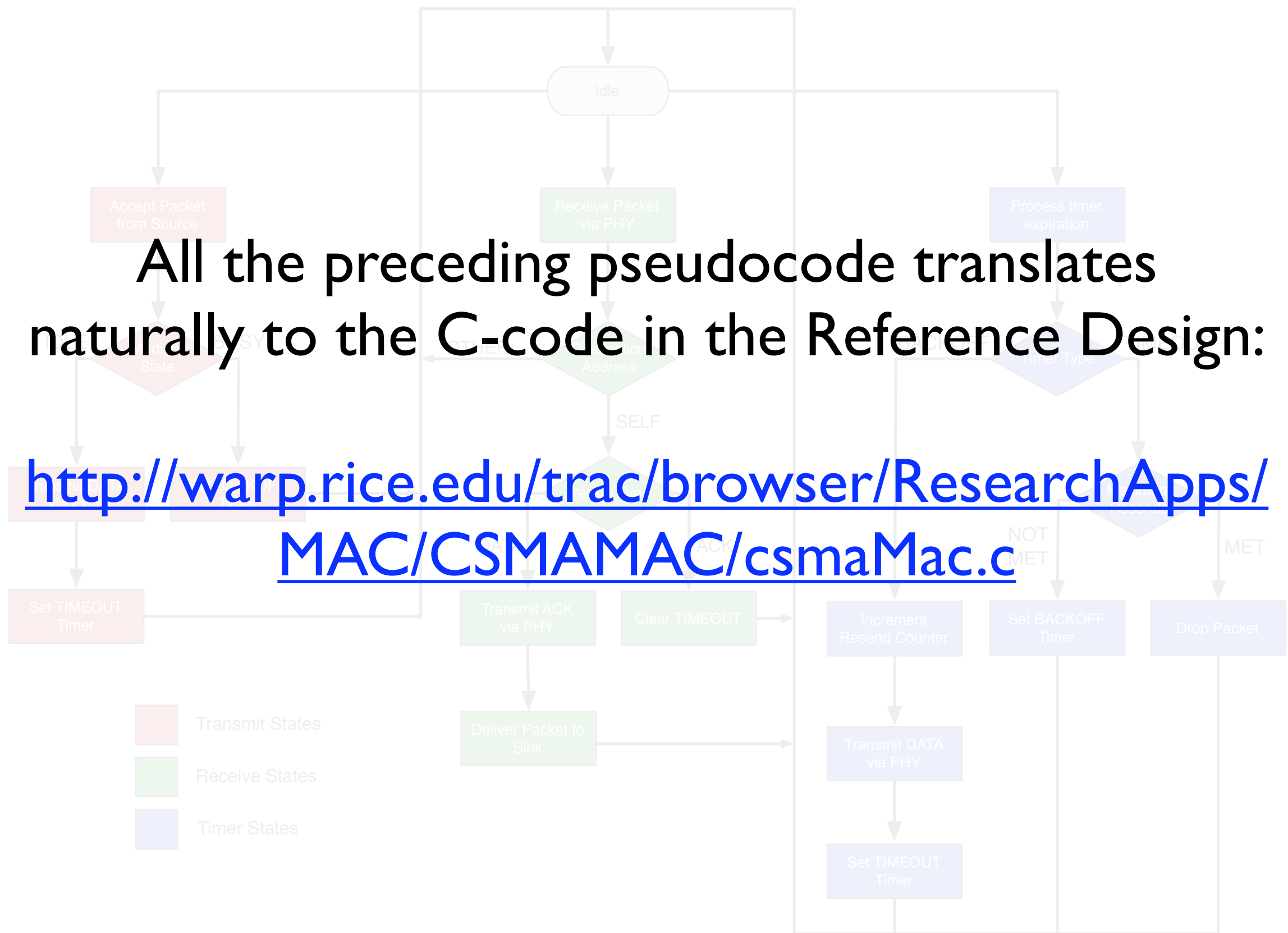
 WARPMAC

 User-Code



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 \*/*



# WARPMAC Structs

phyHeader:

```
unsigned char fullRate; /* Payload modulation rate */
unsigned char reserved4; /* Unused */
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 */
```

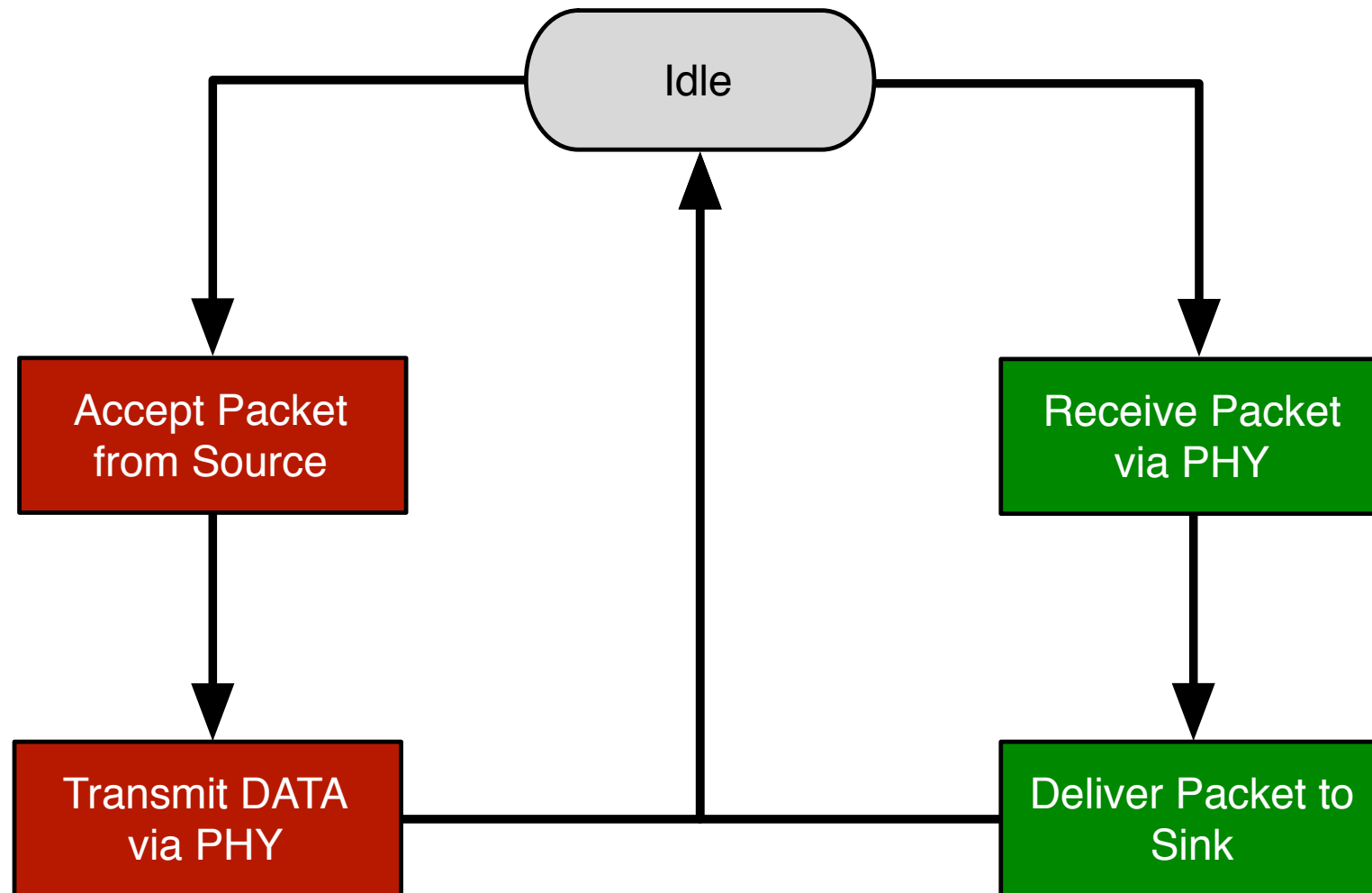
Fully documented in API ([http://warp.rice.edu/WARP\\_API](http://warp.rice.edu/WARP_API))

**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



Transmit States



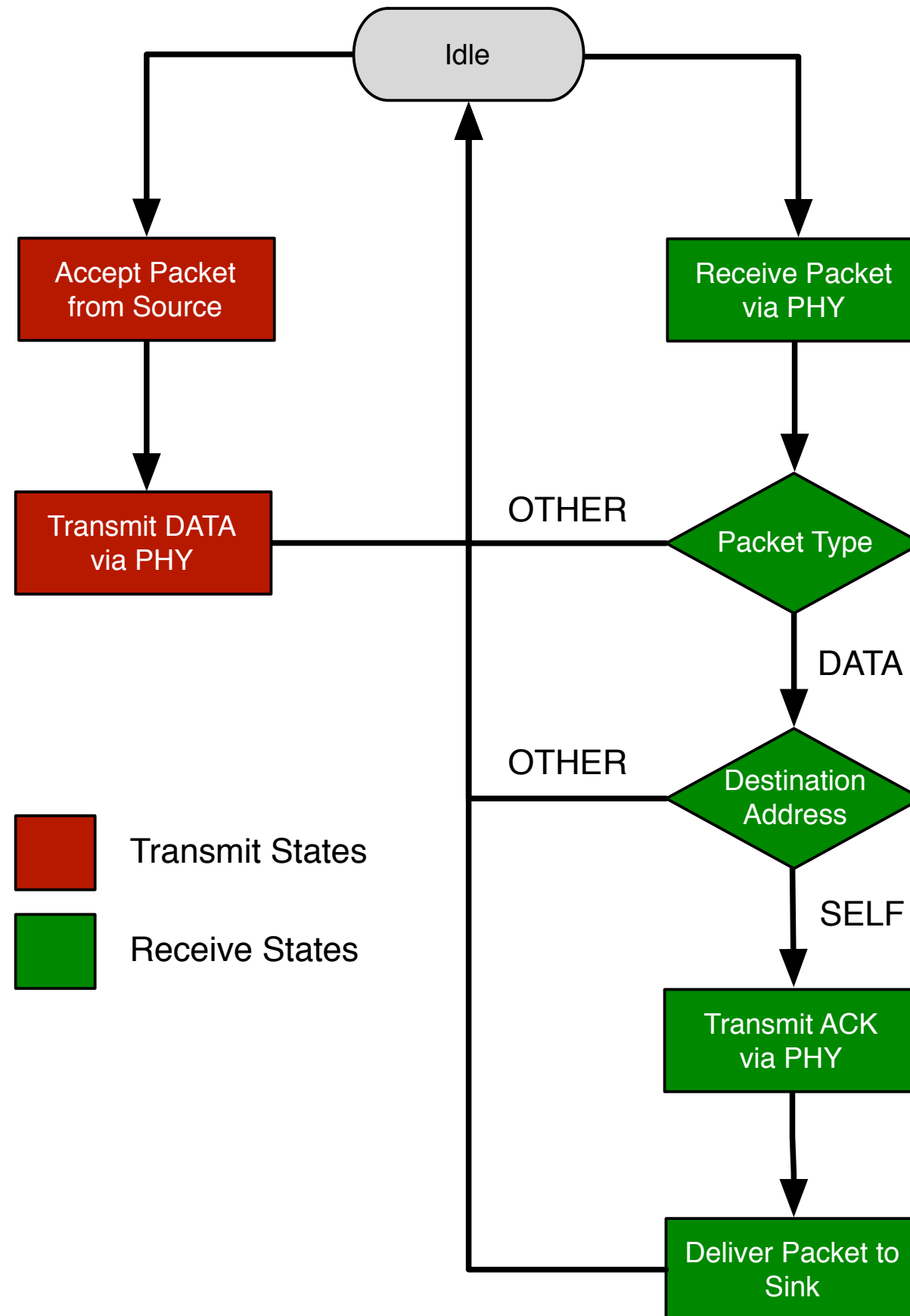
Receive States

# Questions?

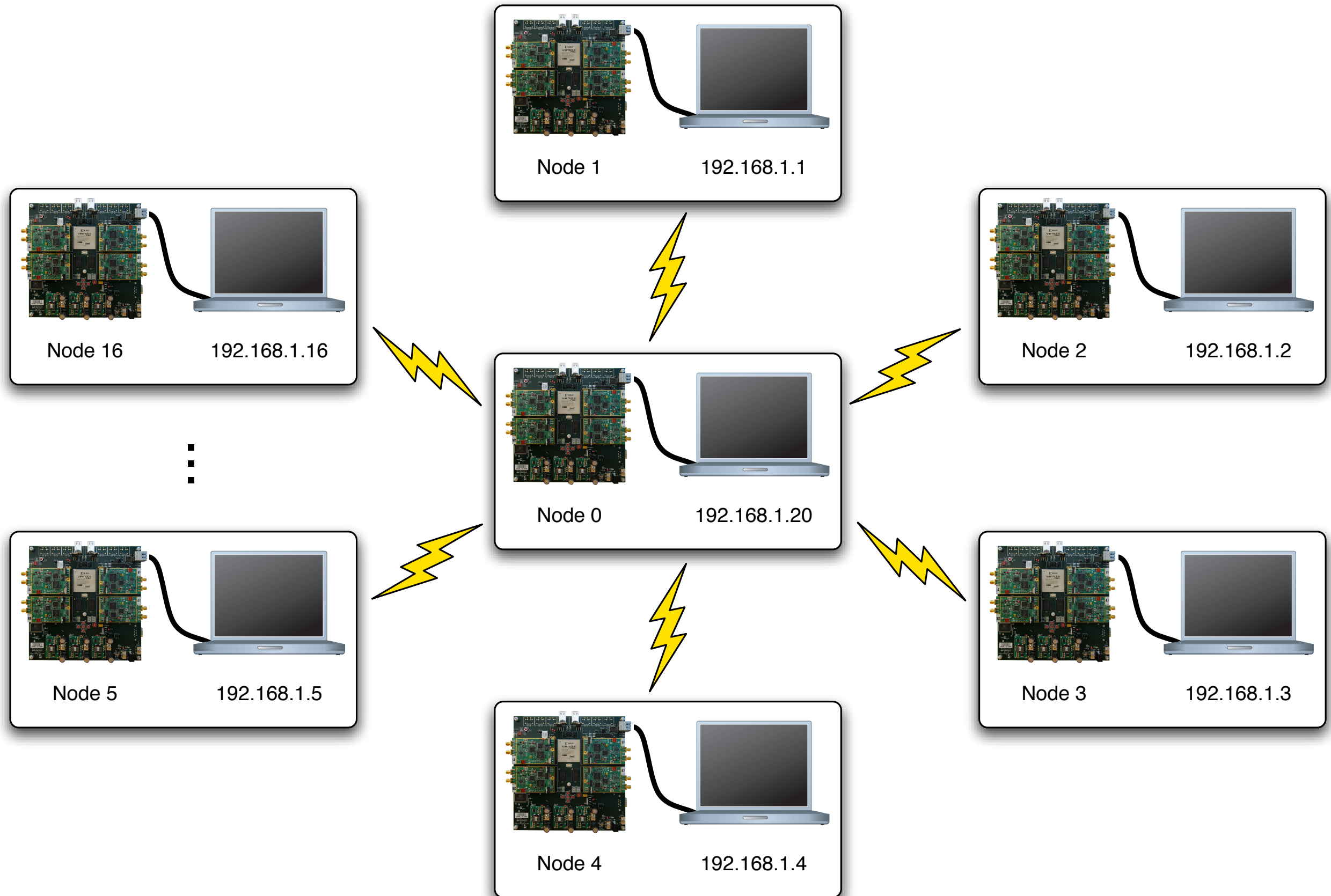
Remember to use the API:

[http://warp.rice.edu/WARP\\_API](http://warp.rice.edu/WARP_API)

# halfMac

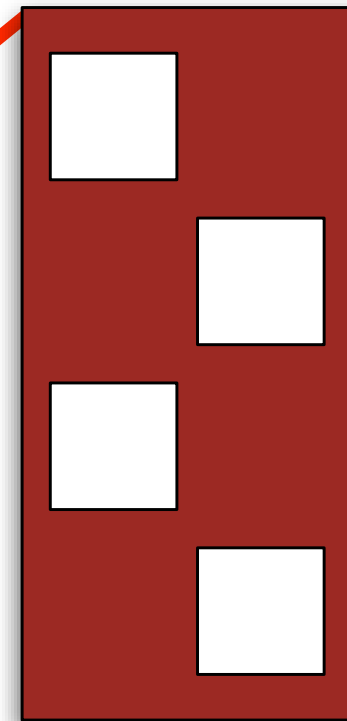
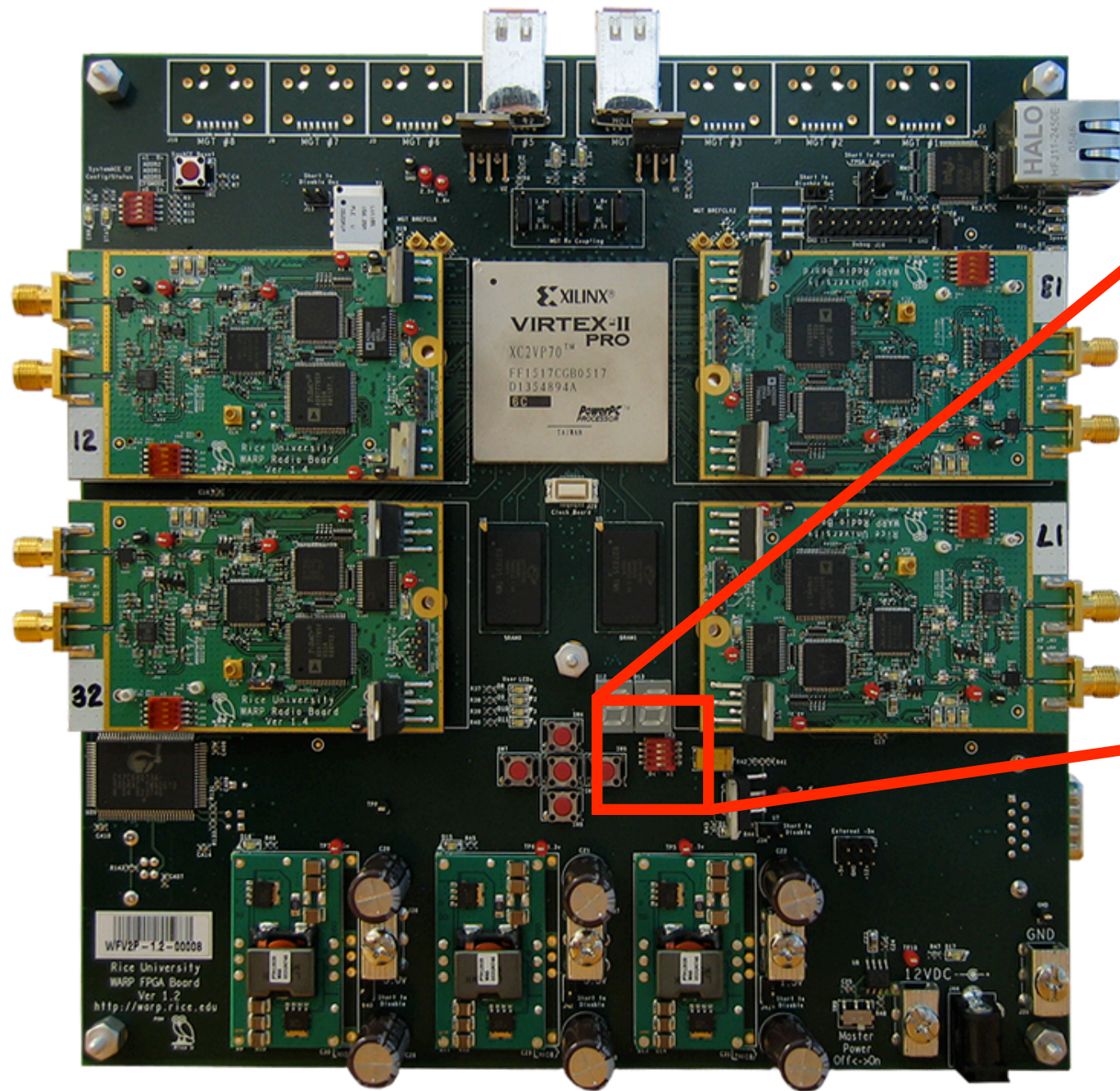


# halfMac





# halfMac



Most Significant Bit (MSB)

Least Significant Bit (LSB)

Node 5

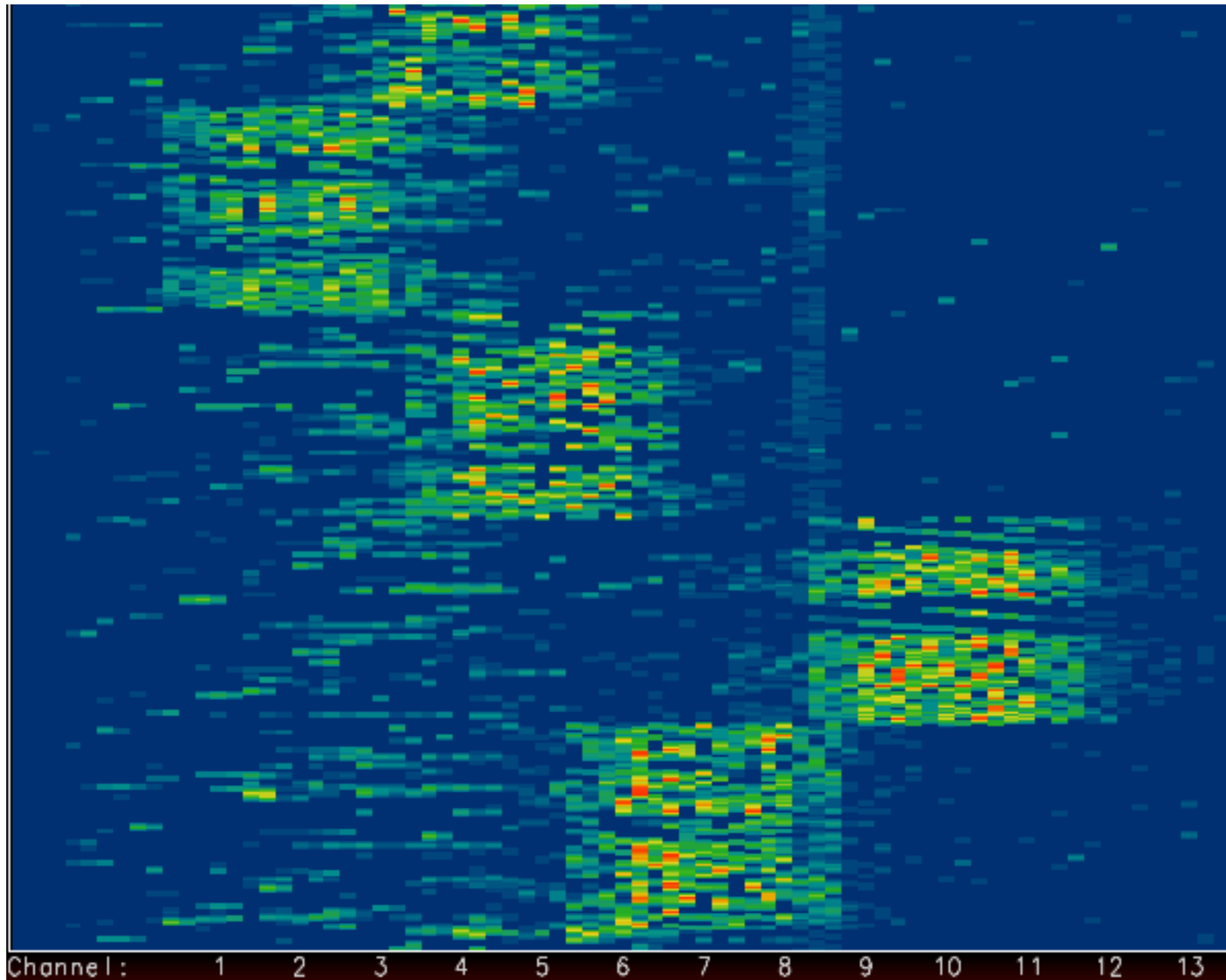


# Questions?

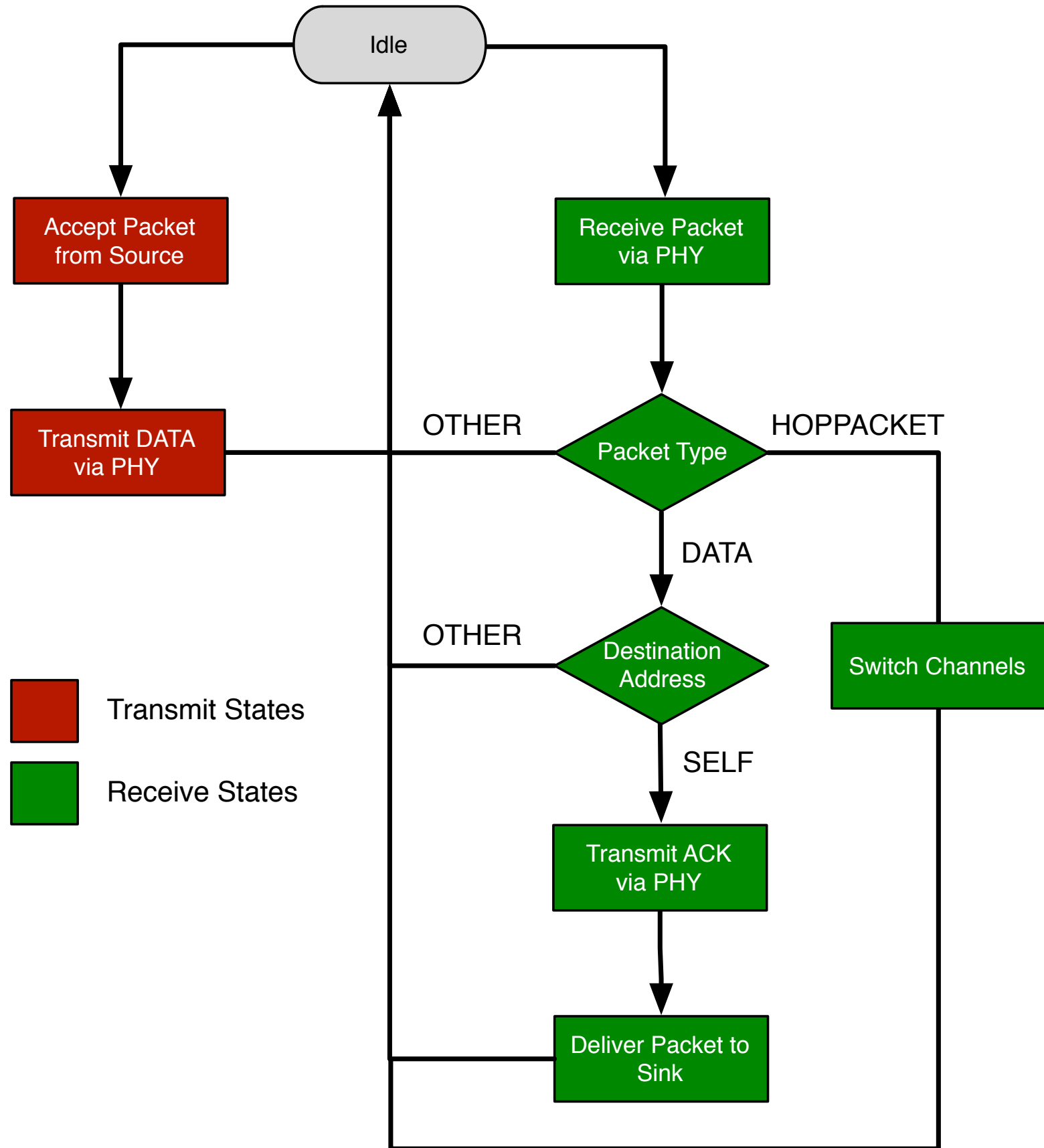
Remember to use the API:

[http://warp.rice.edu/WARP\\_API](http://warp.rice.edu/WARP_API)

# hopMac



# hopMac



# Questions?

Remember to use the API:

[http://warp.rice.edu/WARP\\_API](http://warp.rice.edu/WARP_API)

# Logistics

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