Networking on WARP

Chris Hunter
Rice University

WARP Workshop at Rice University
March 30, 2010

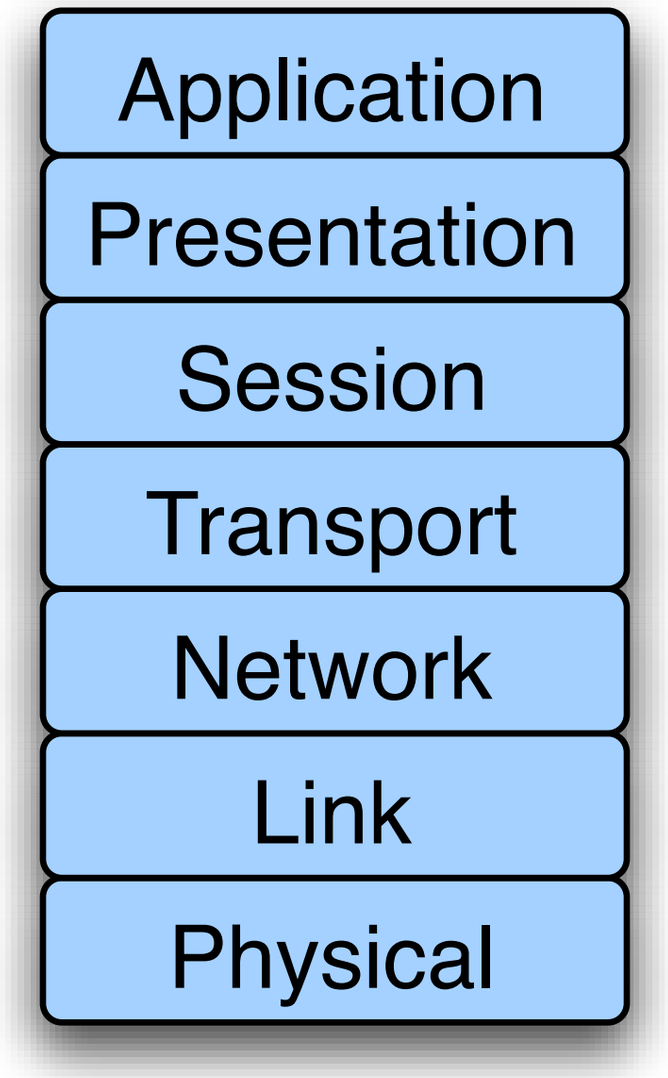
WARP

warp.rice.edu

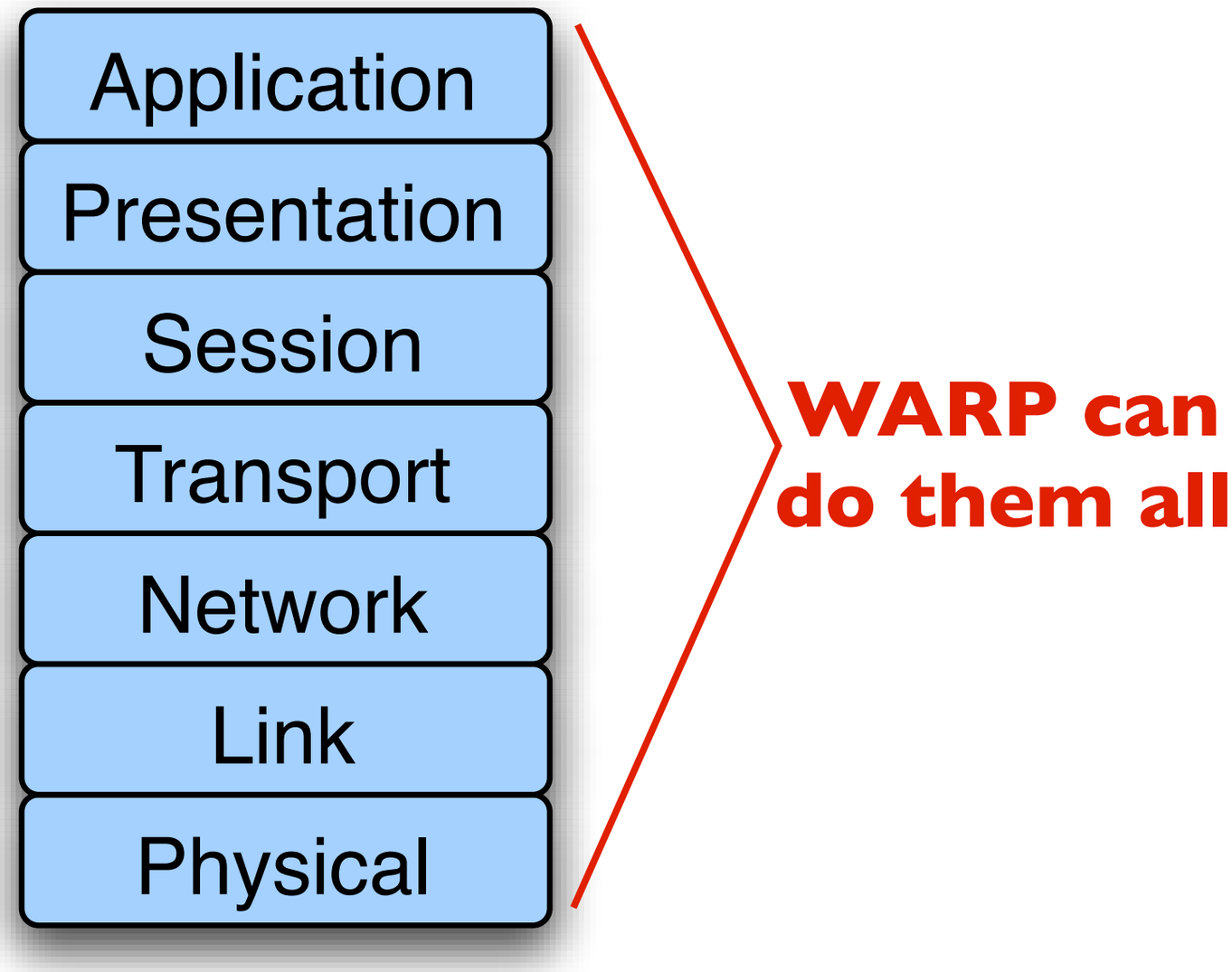
Today's Agenda

- Outstanding questions?
- Networking on WARP lecture
- Labs 4, 5
- Lunch
- Advanced MAC/PHY Design: A Case Study
- Lab 6+
- Workshop wrap-up
- UCI Demo

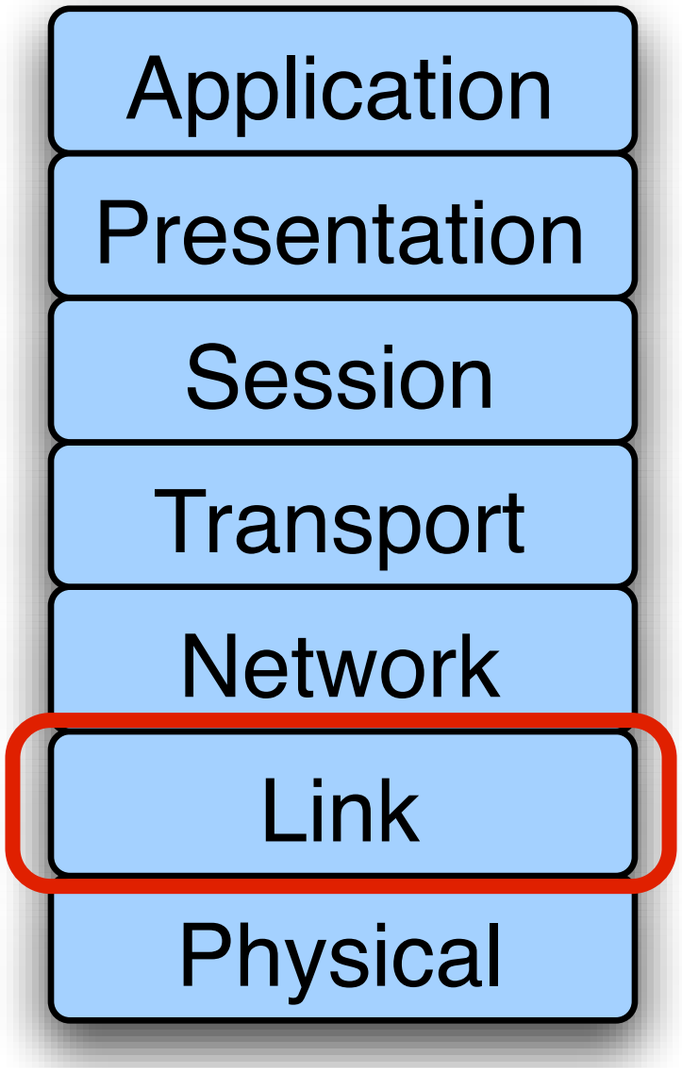
The OSI Model



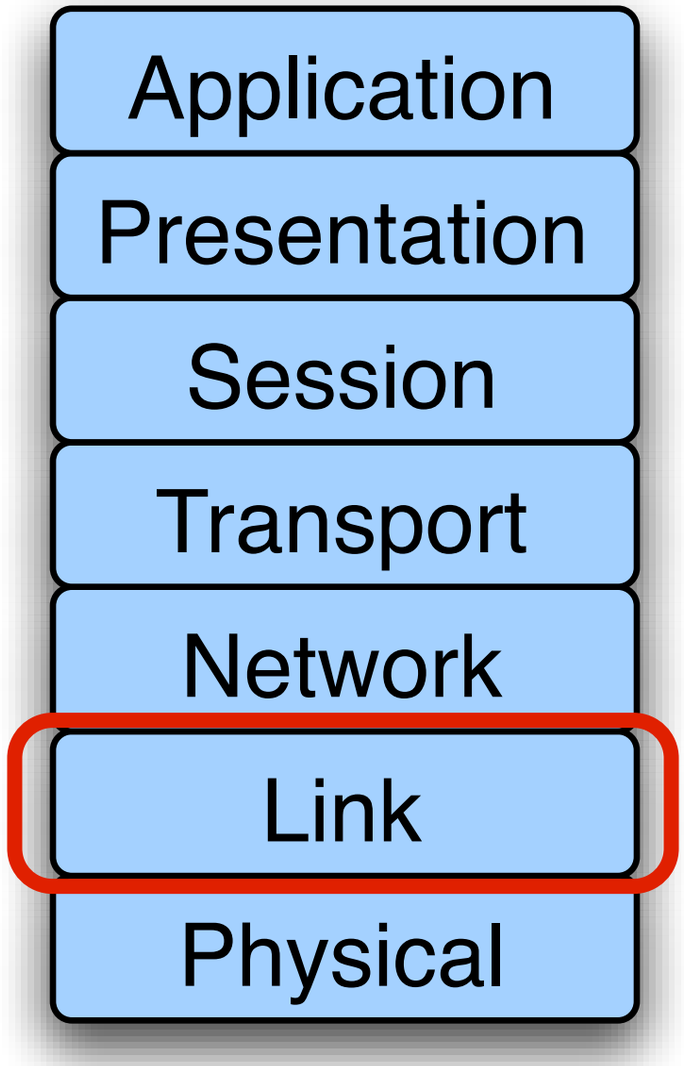
The OSI Model



The OSI Model

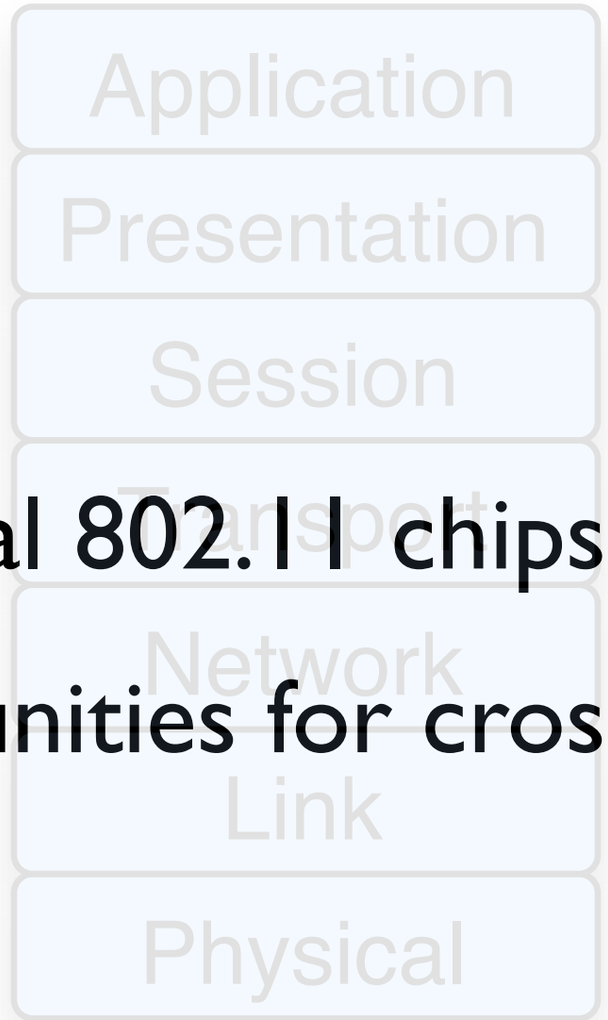


The OSI Model



Our Focus: Medium Access Control

The OSI Model



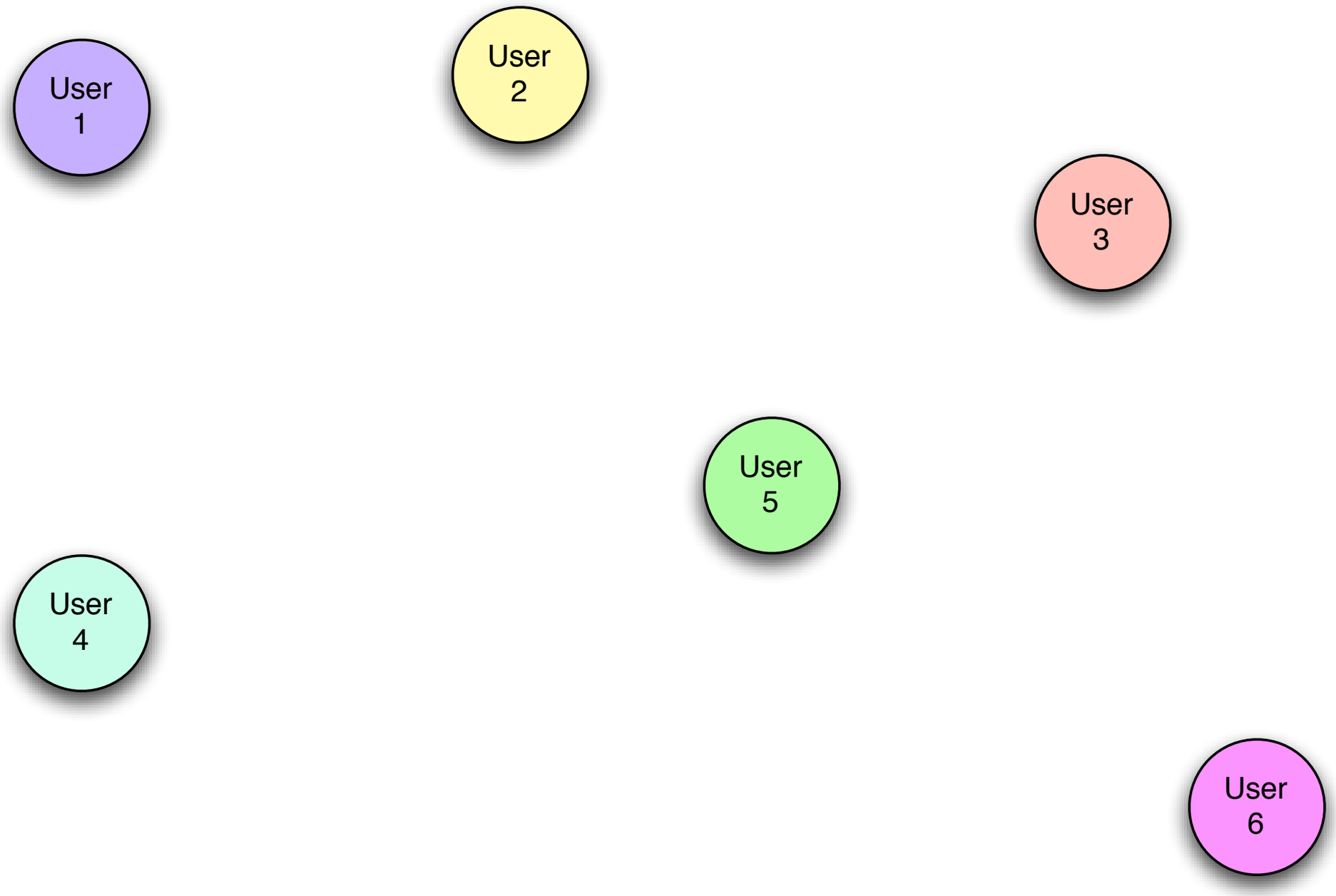
- Why?
  - All commercial 802.11 chipsets are closed
  - Many opportunities for cross-layer research

Outline

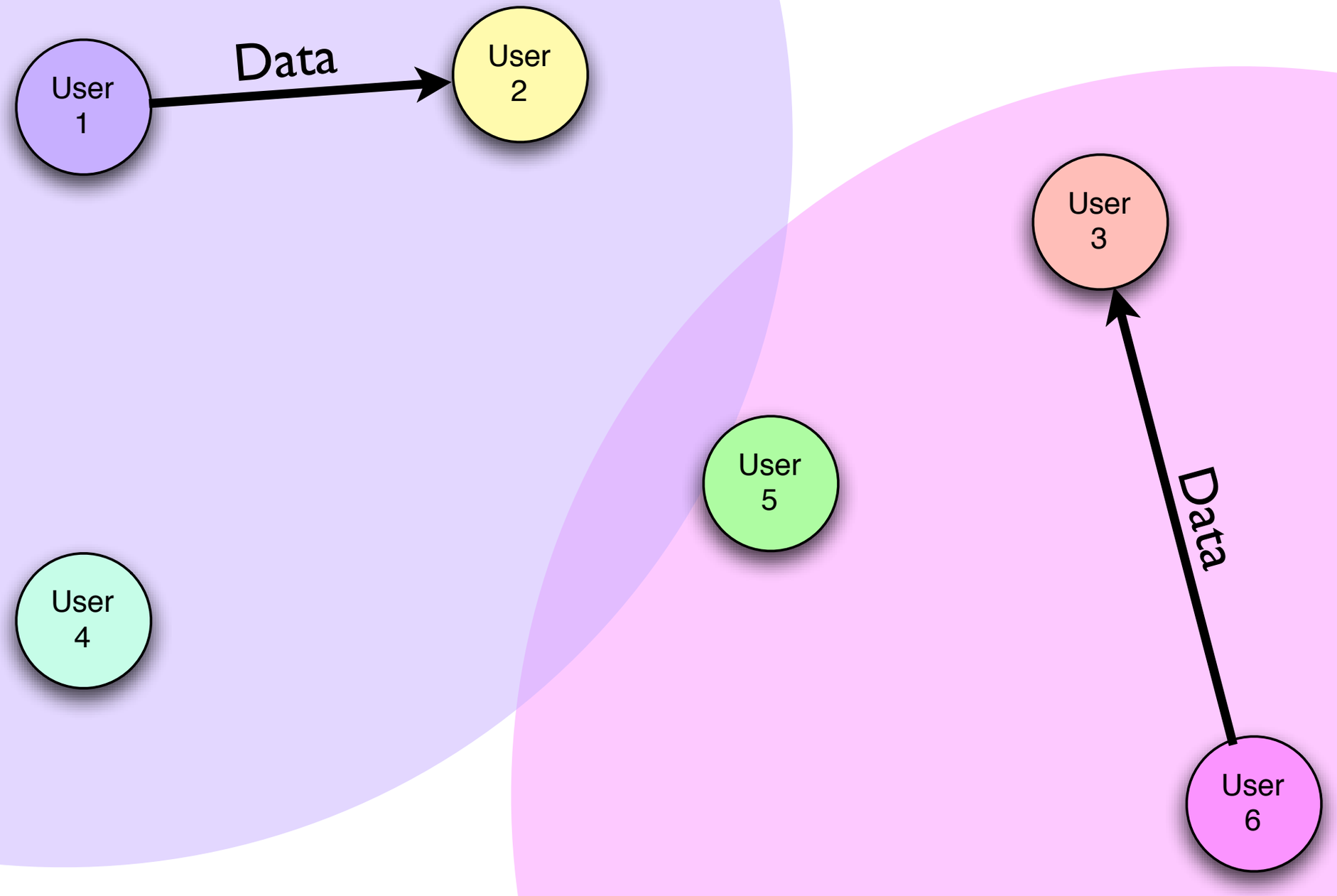
- Overview of Medium Access Control
- Design Realization
- Example
- Lab Exercises

Medium Access Control Overview

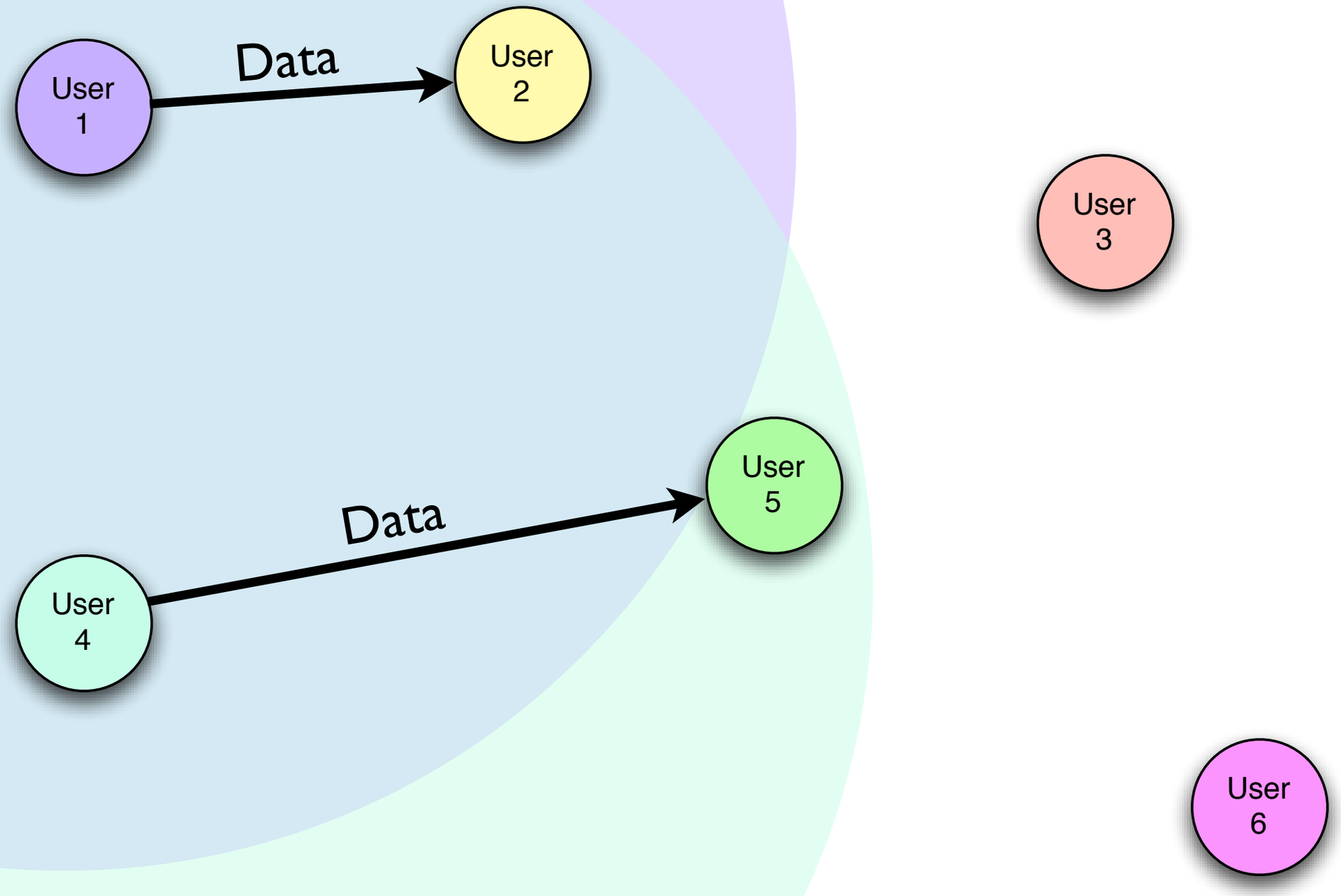
What is a MAC?



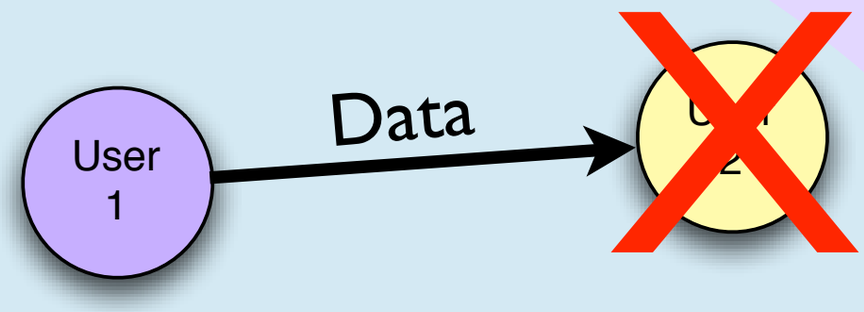
What is a MAC?

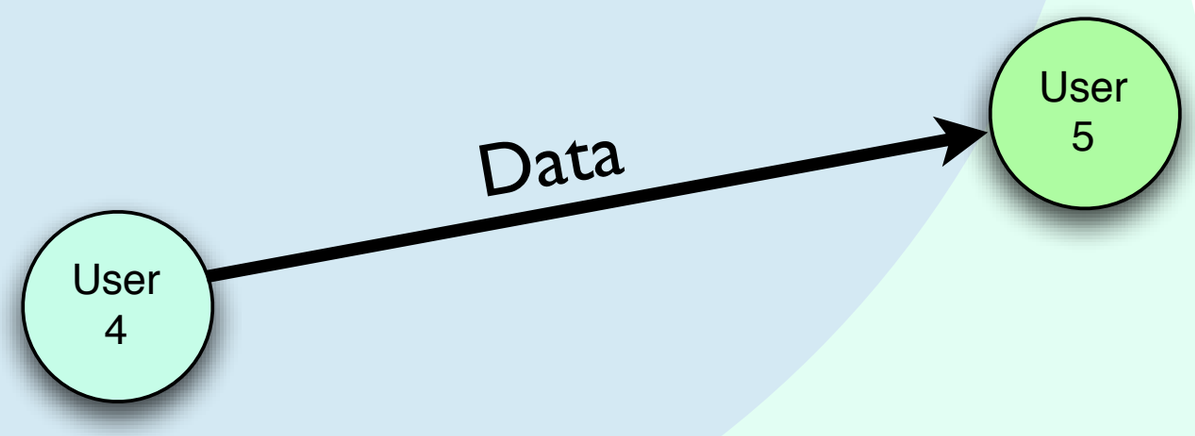


What is a MAC?

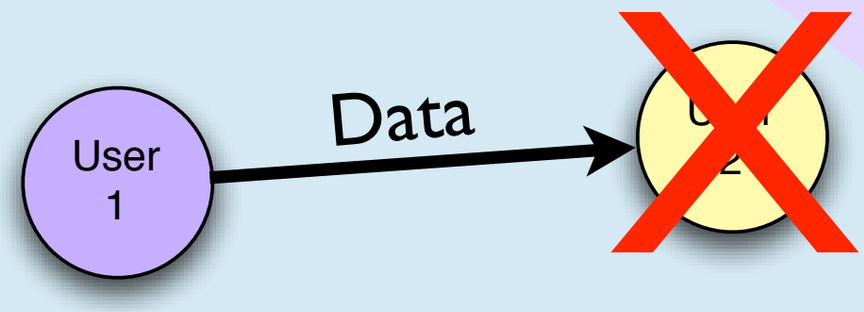


What is a MAC?

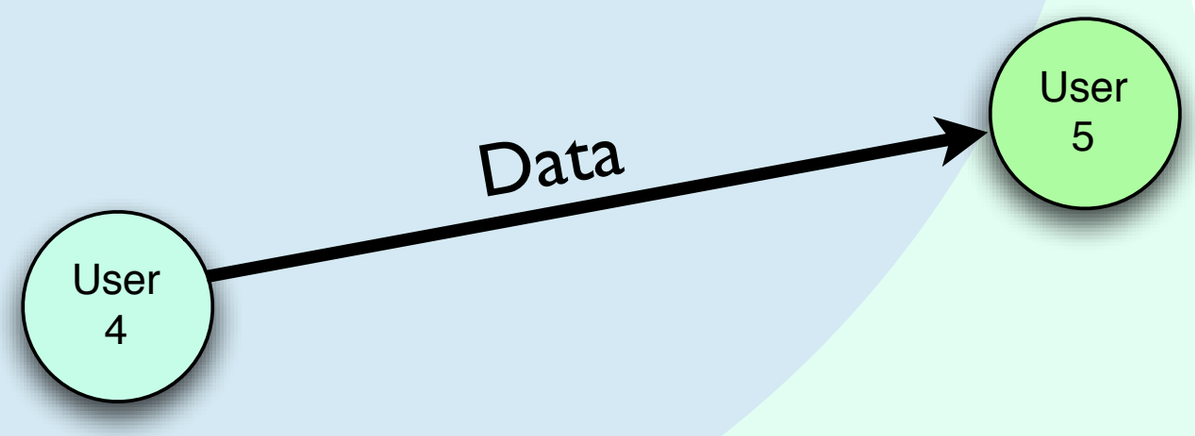




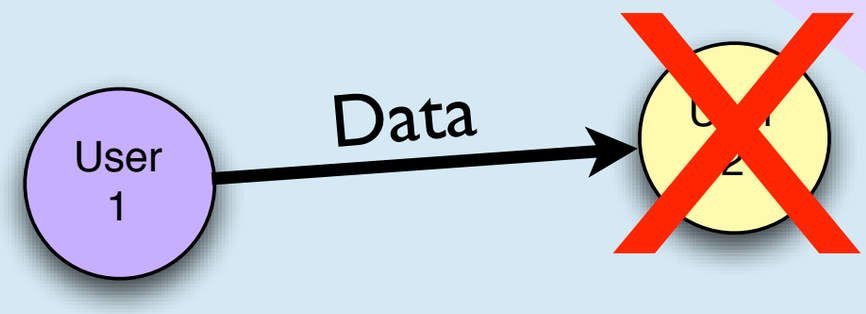
What is a MAC?



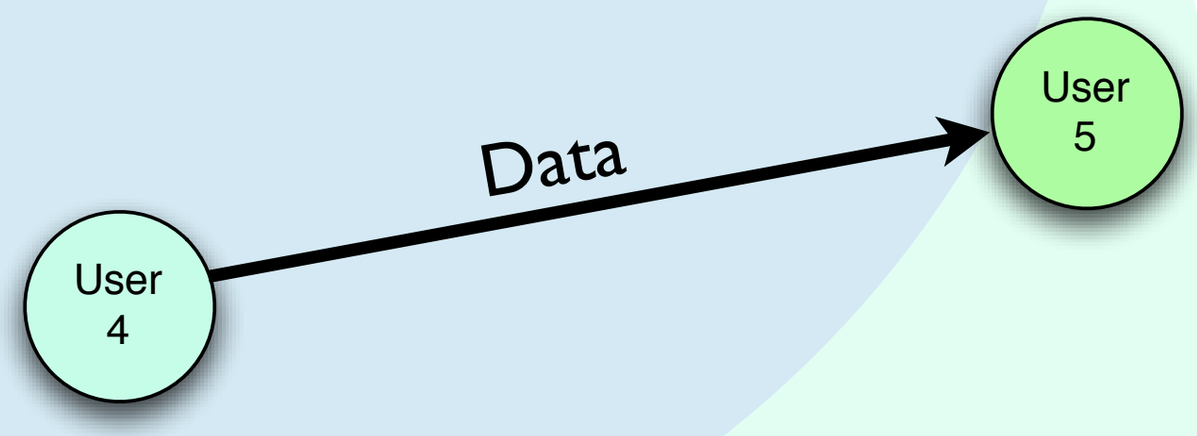
Received a jumbled packet... infer a packet collision



What is a MAC?

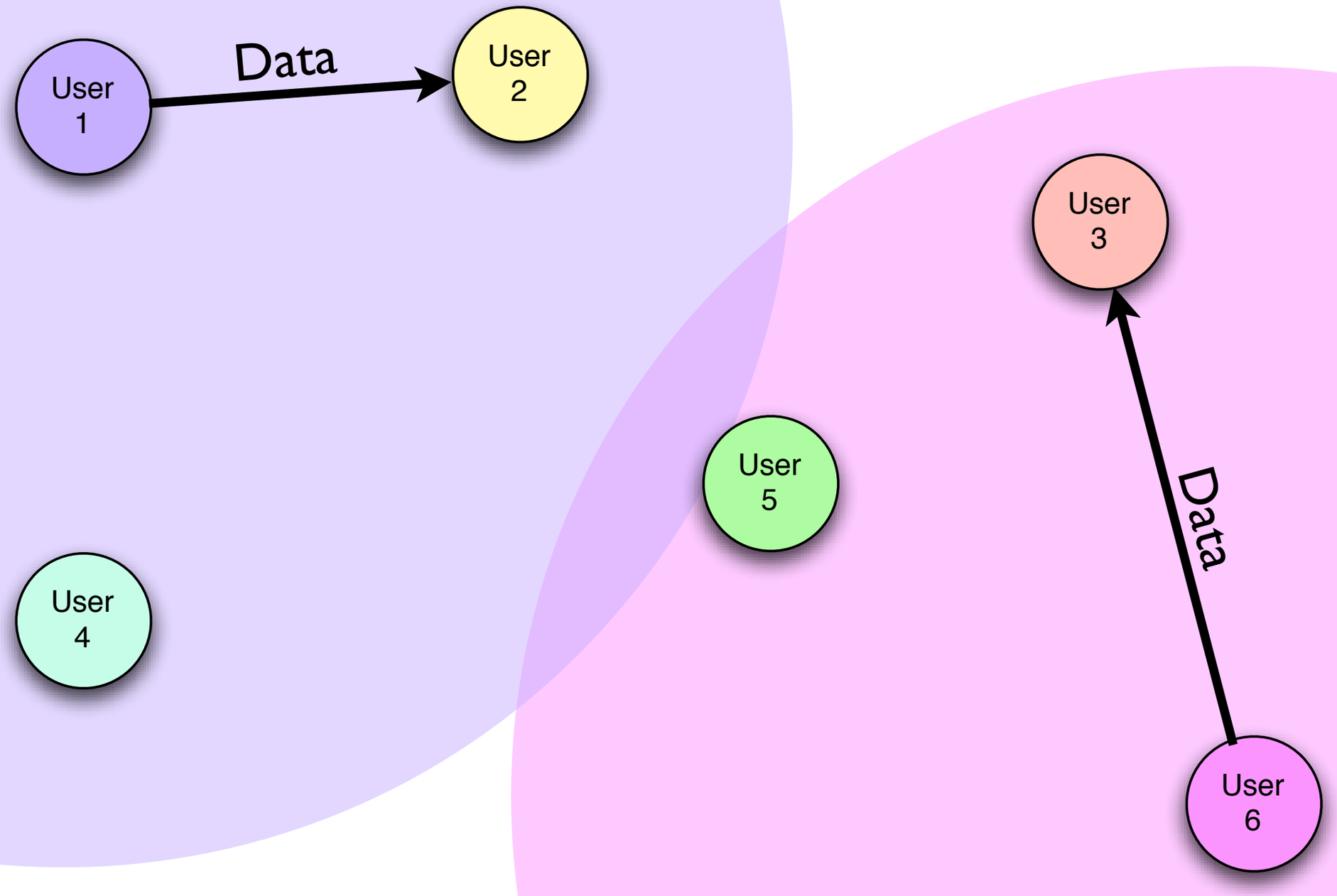


Received a jumbled packet... infer a packet collision

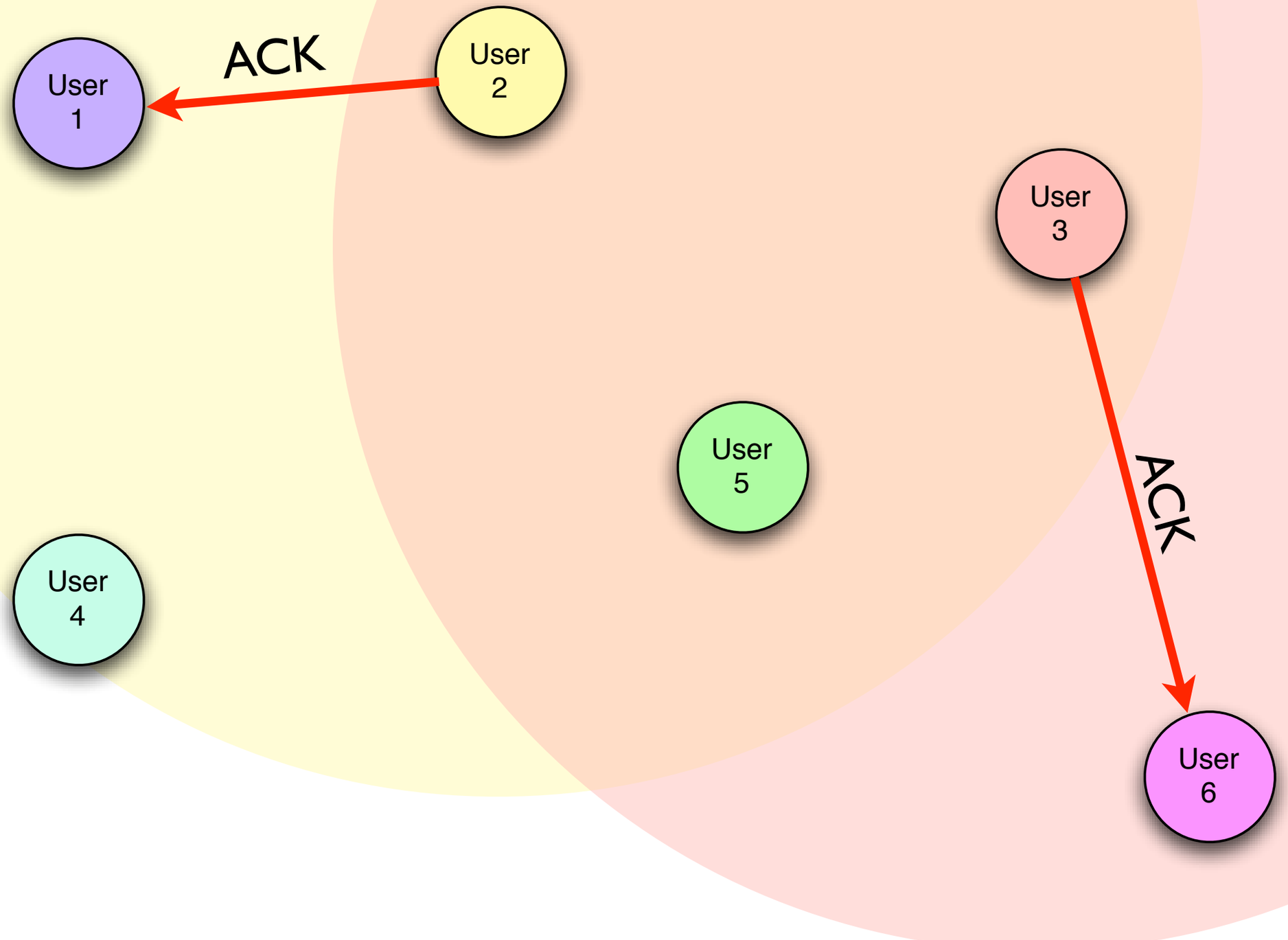


What if we ACK every transmit, and retransmit when we receive no ACK?

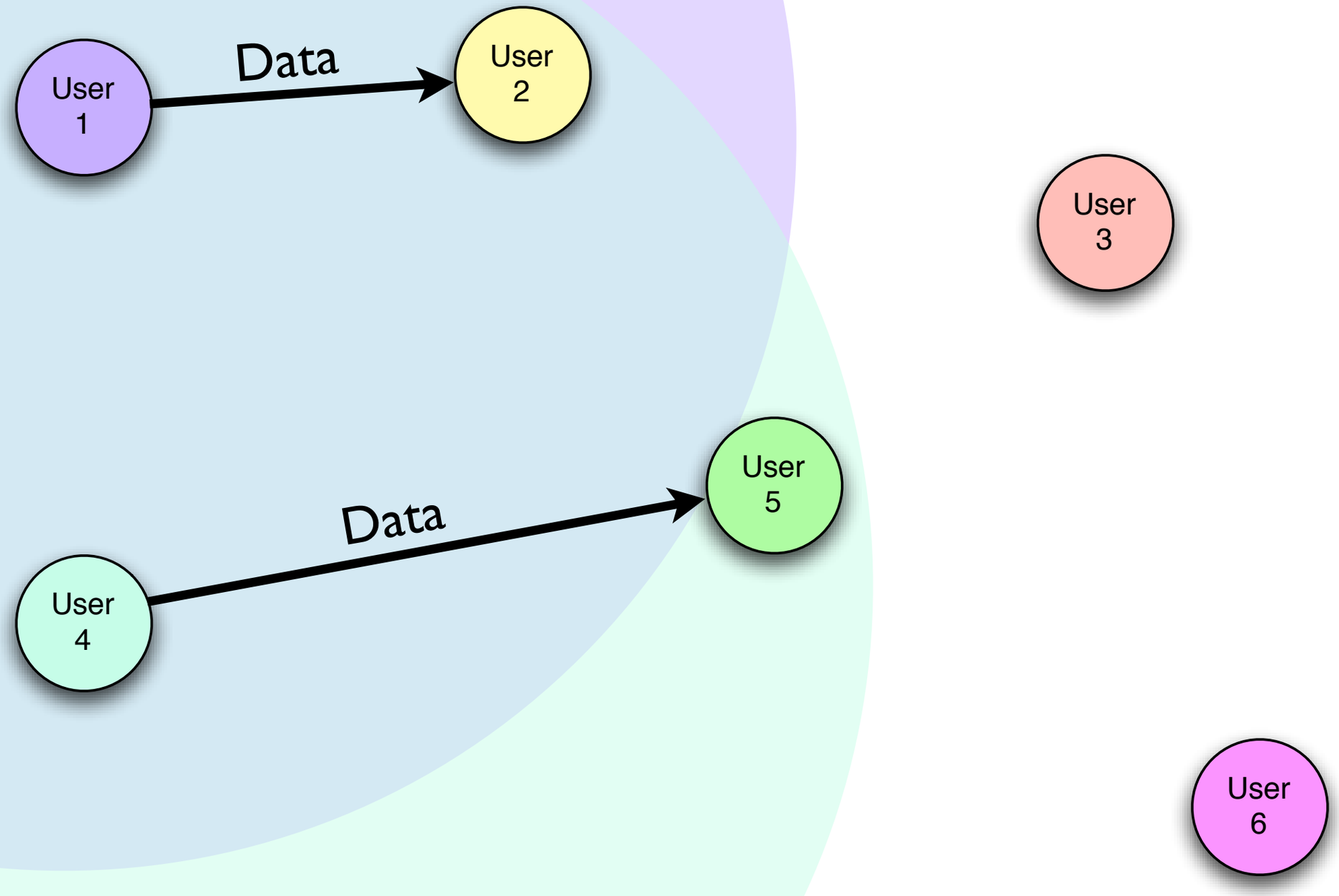
What is a MAC?



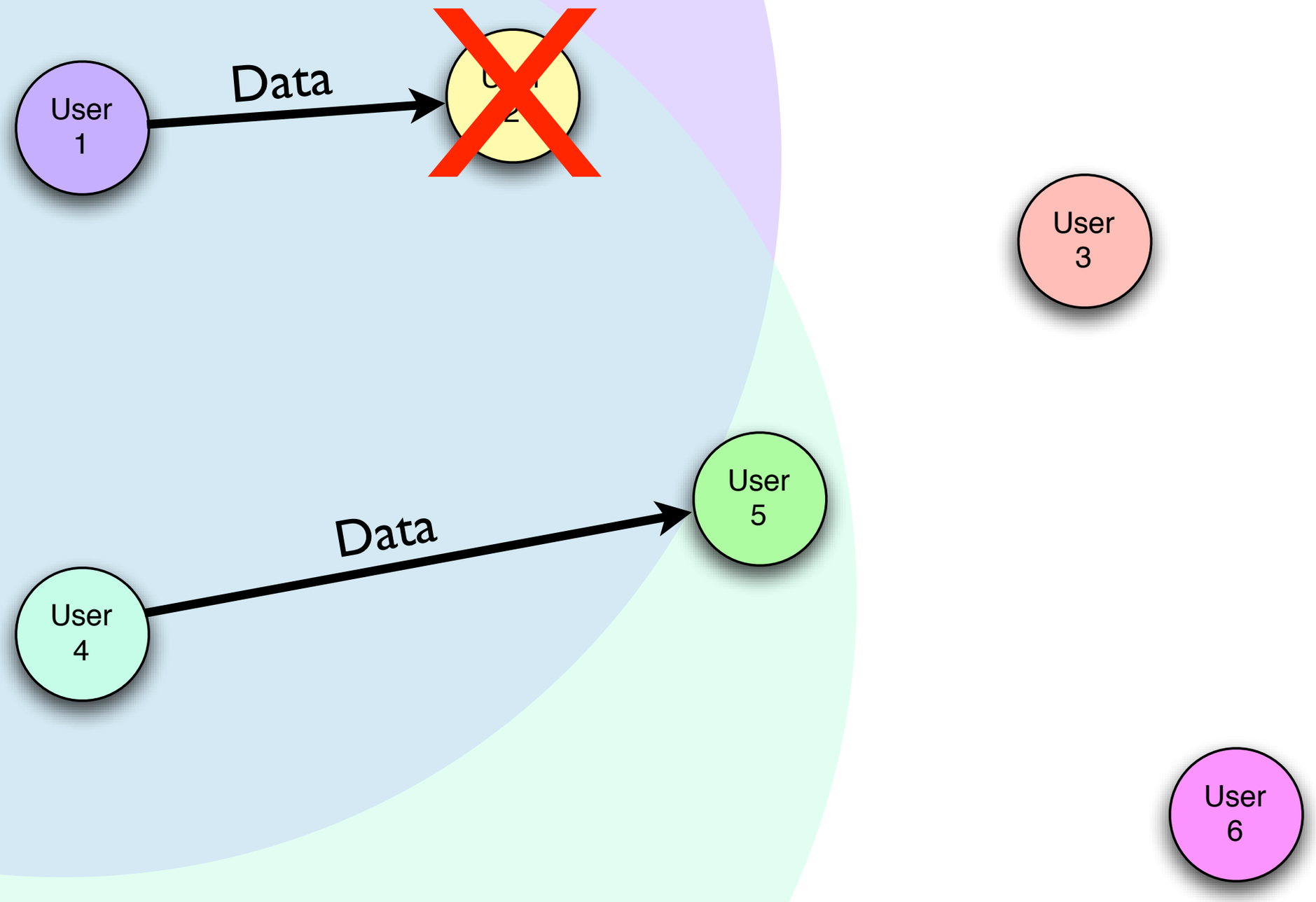
What is a MAC?



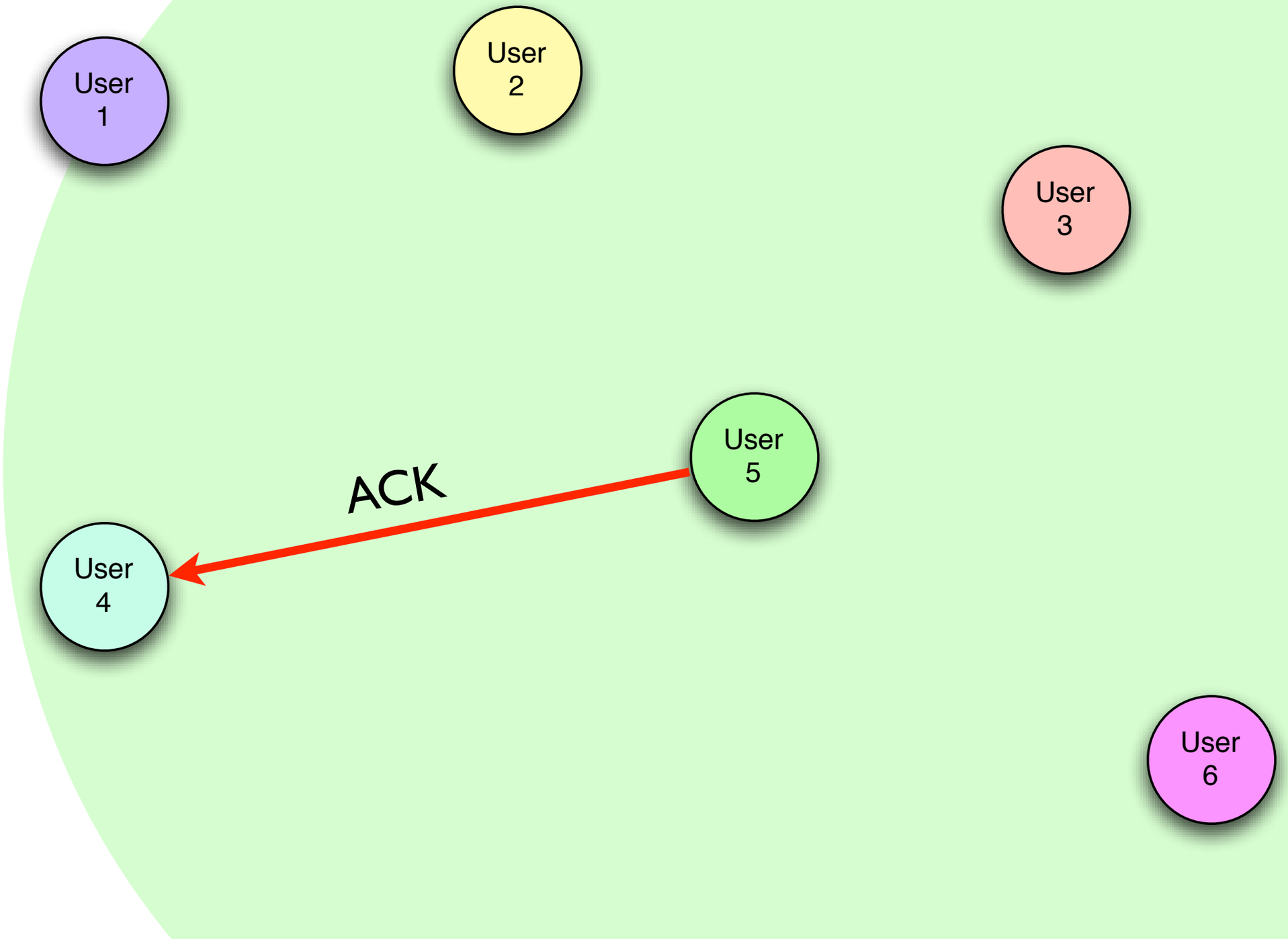
What is a MAC?



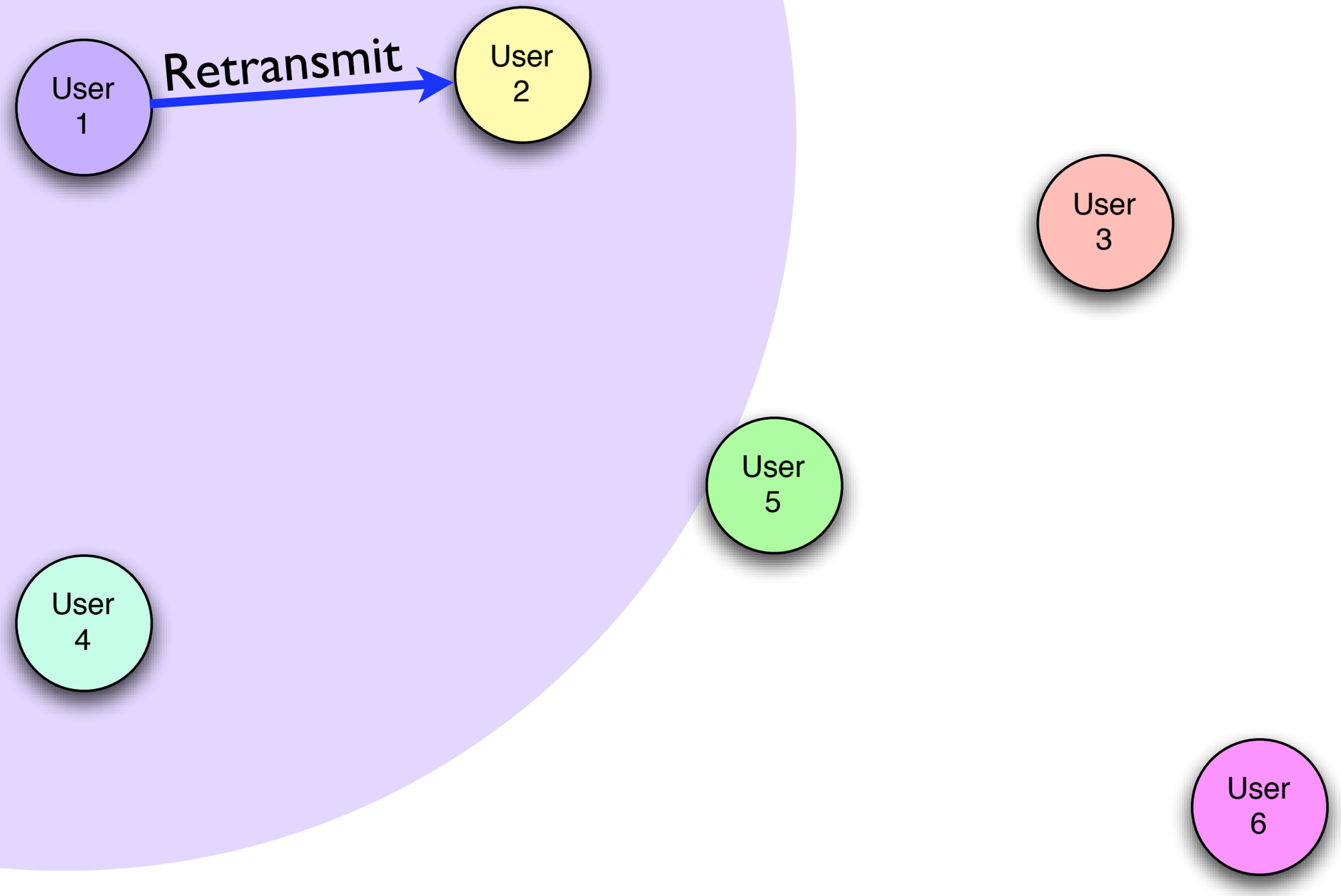
What is a MAC?



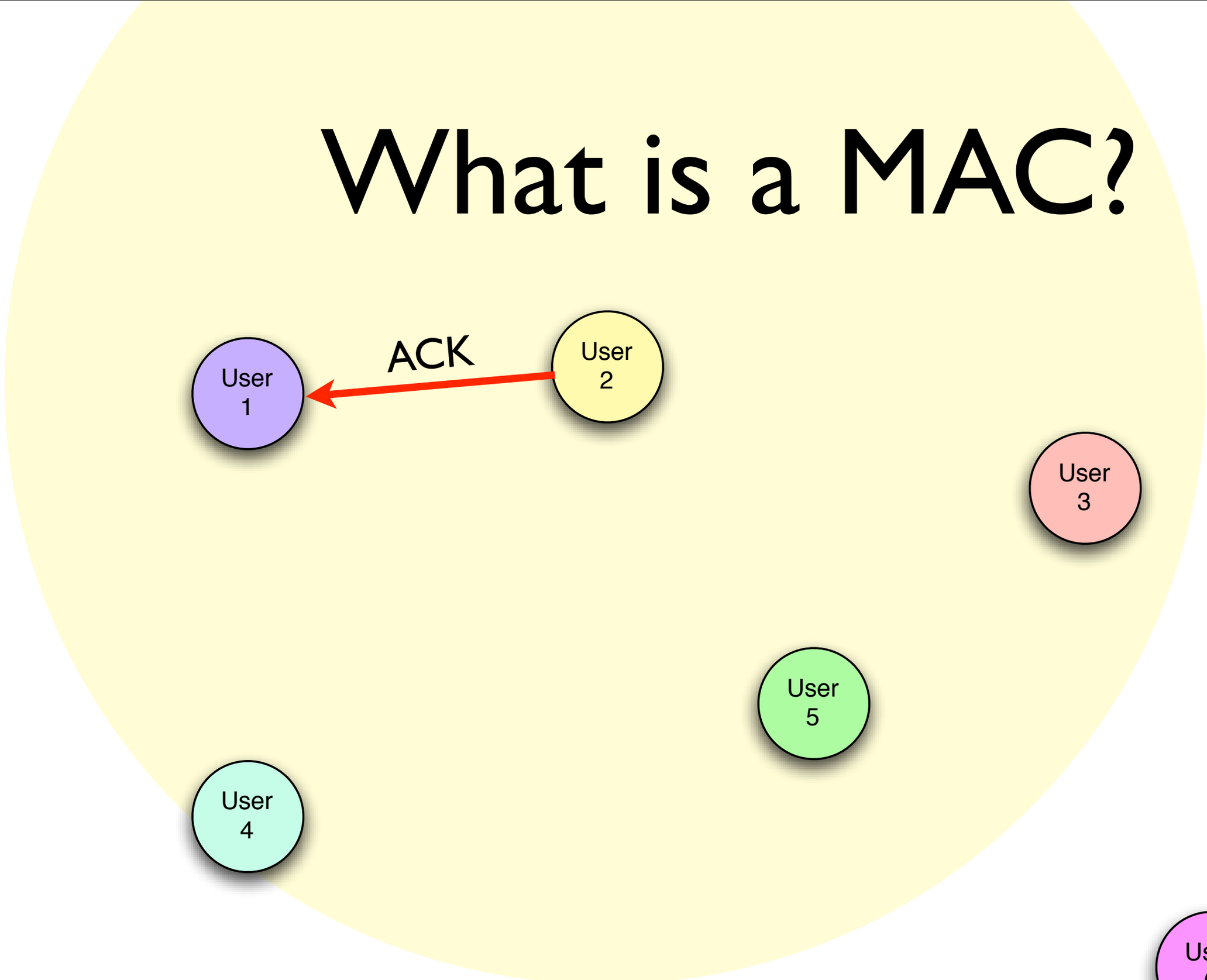
What is a MAC?



What is a MAC?



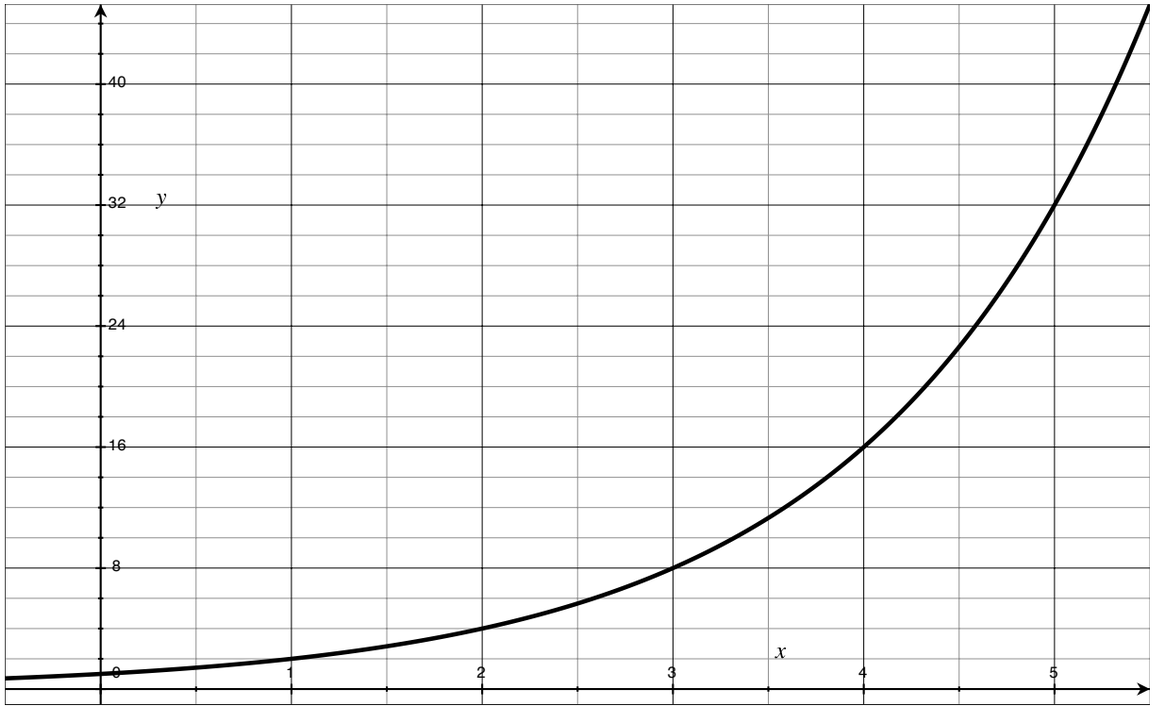
What is a MAC?



Random Backoffs

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

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



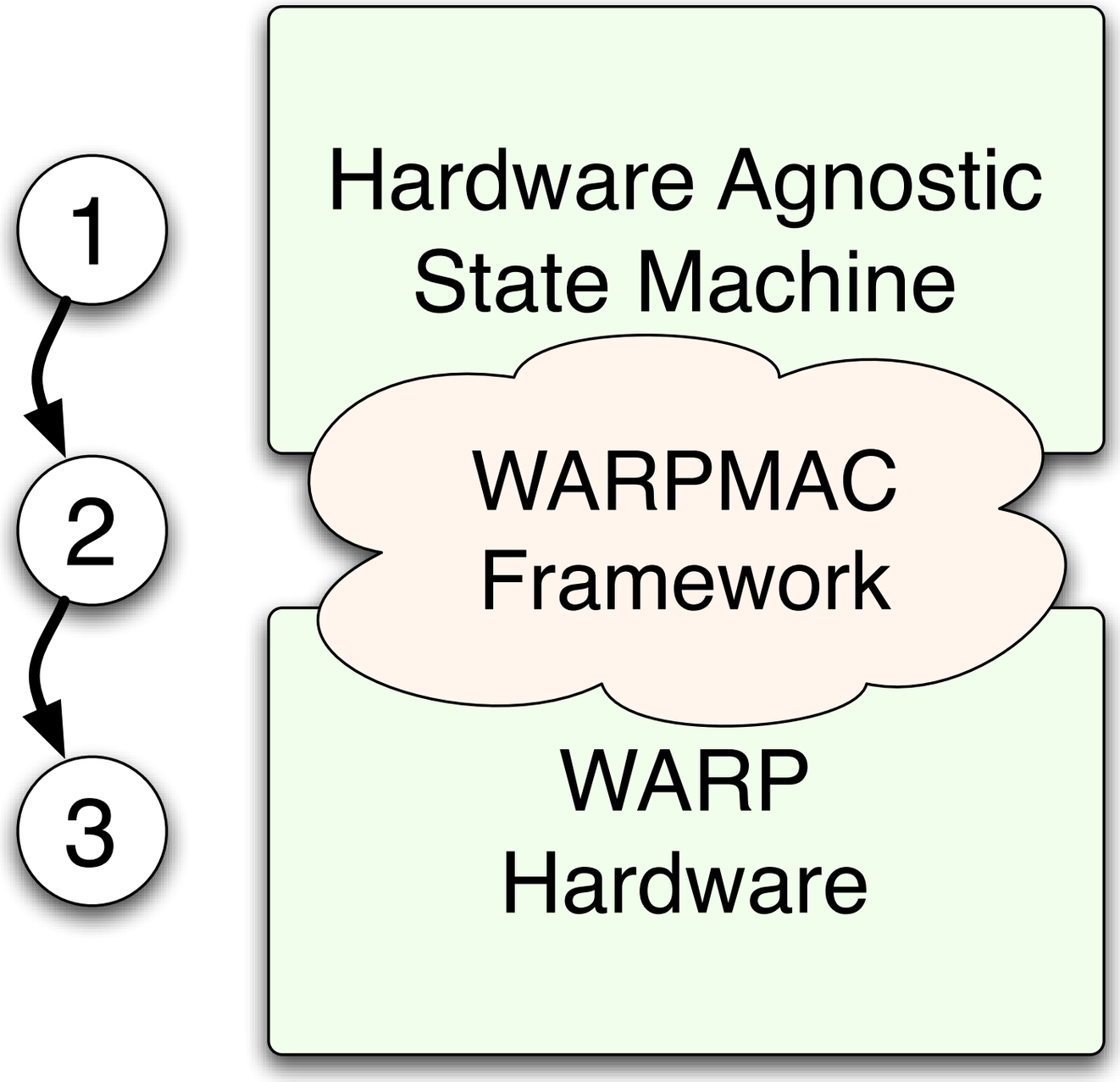
Contention Window increases over time

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

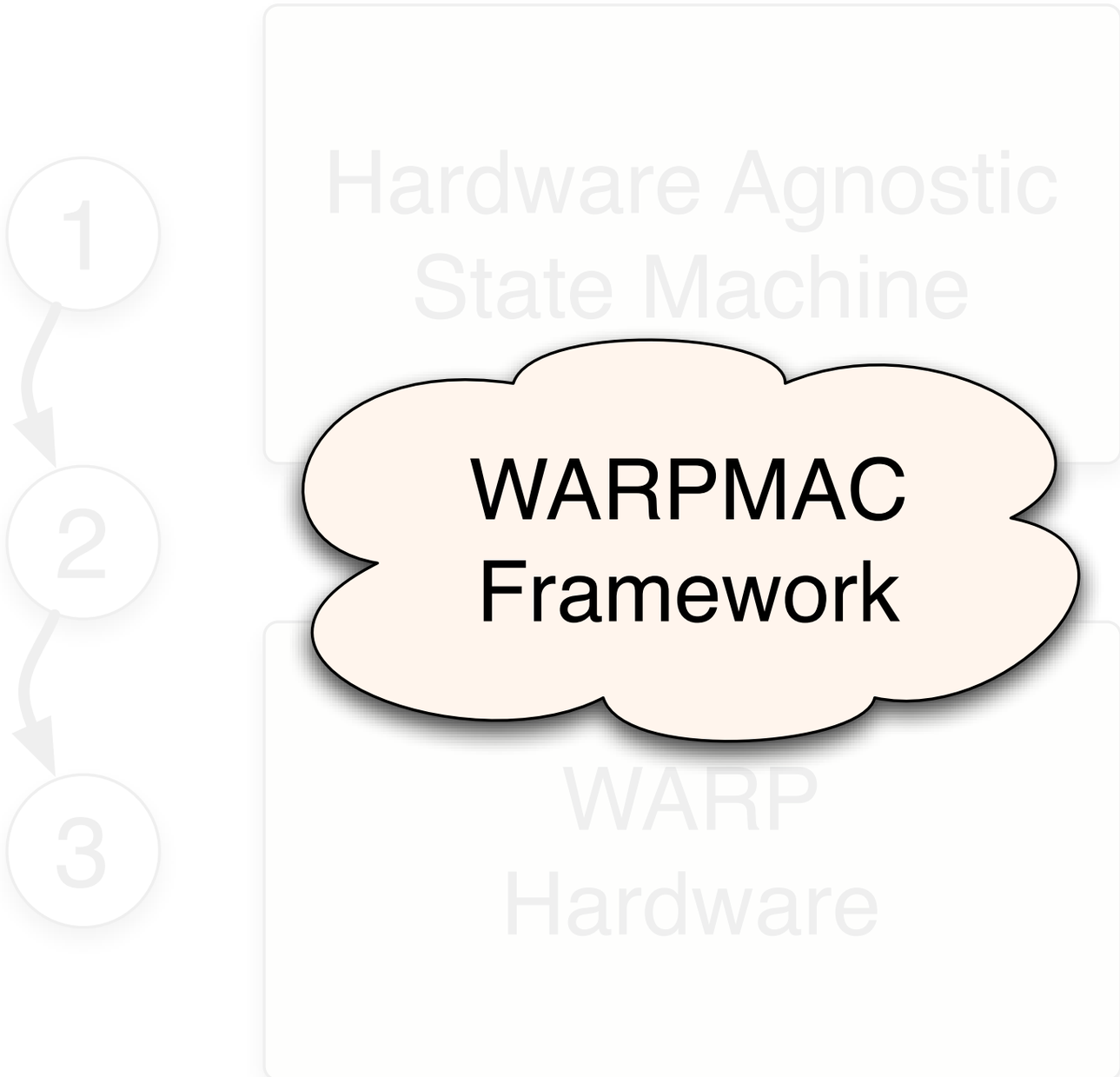
Design Realization

Design Realization



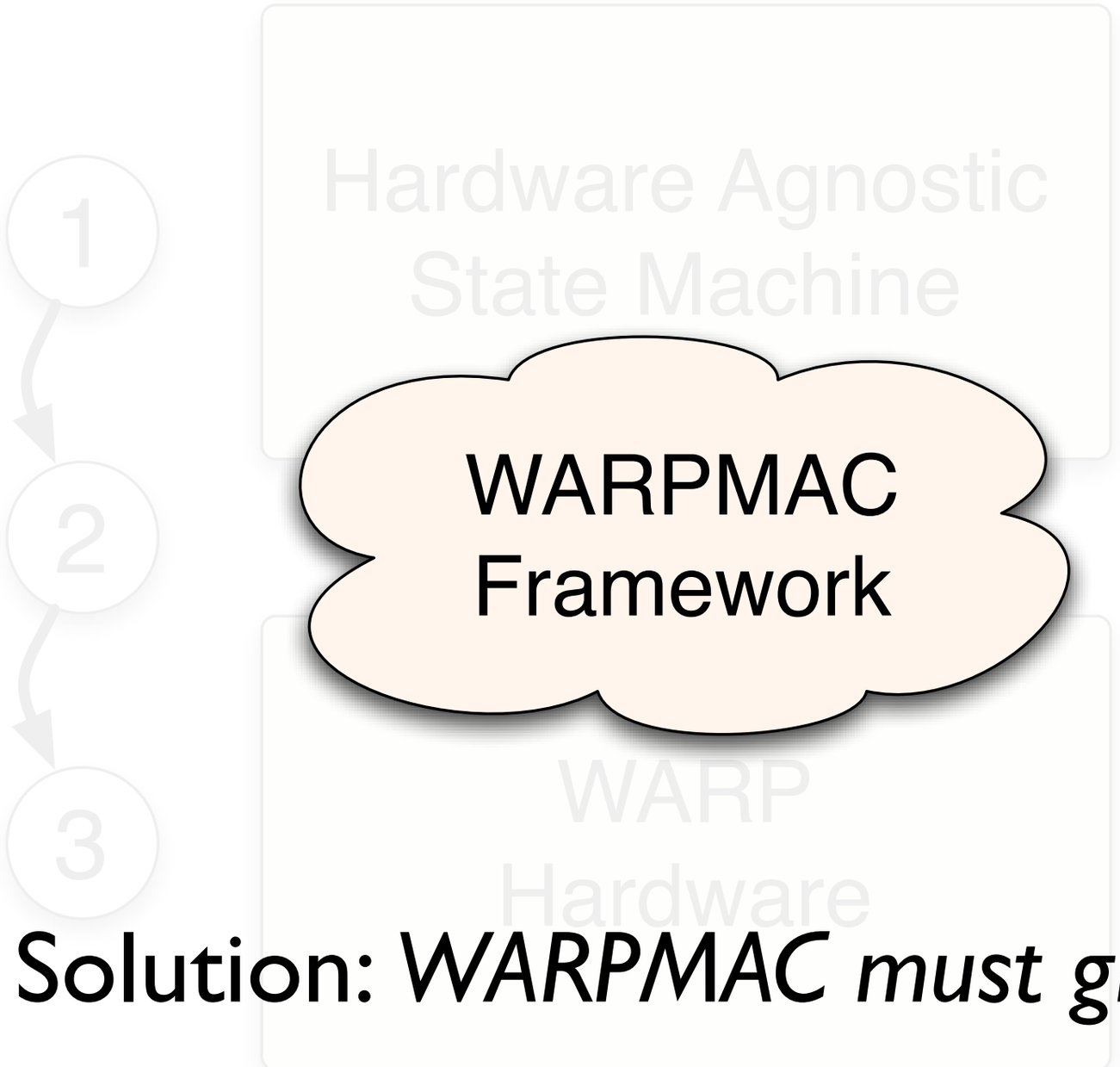
- Program high-level MAC behavior independent of hardware
- Use the WARPMAC framework to stitch the MAC to hardware

Design Realization



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

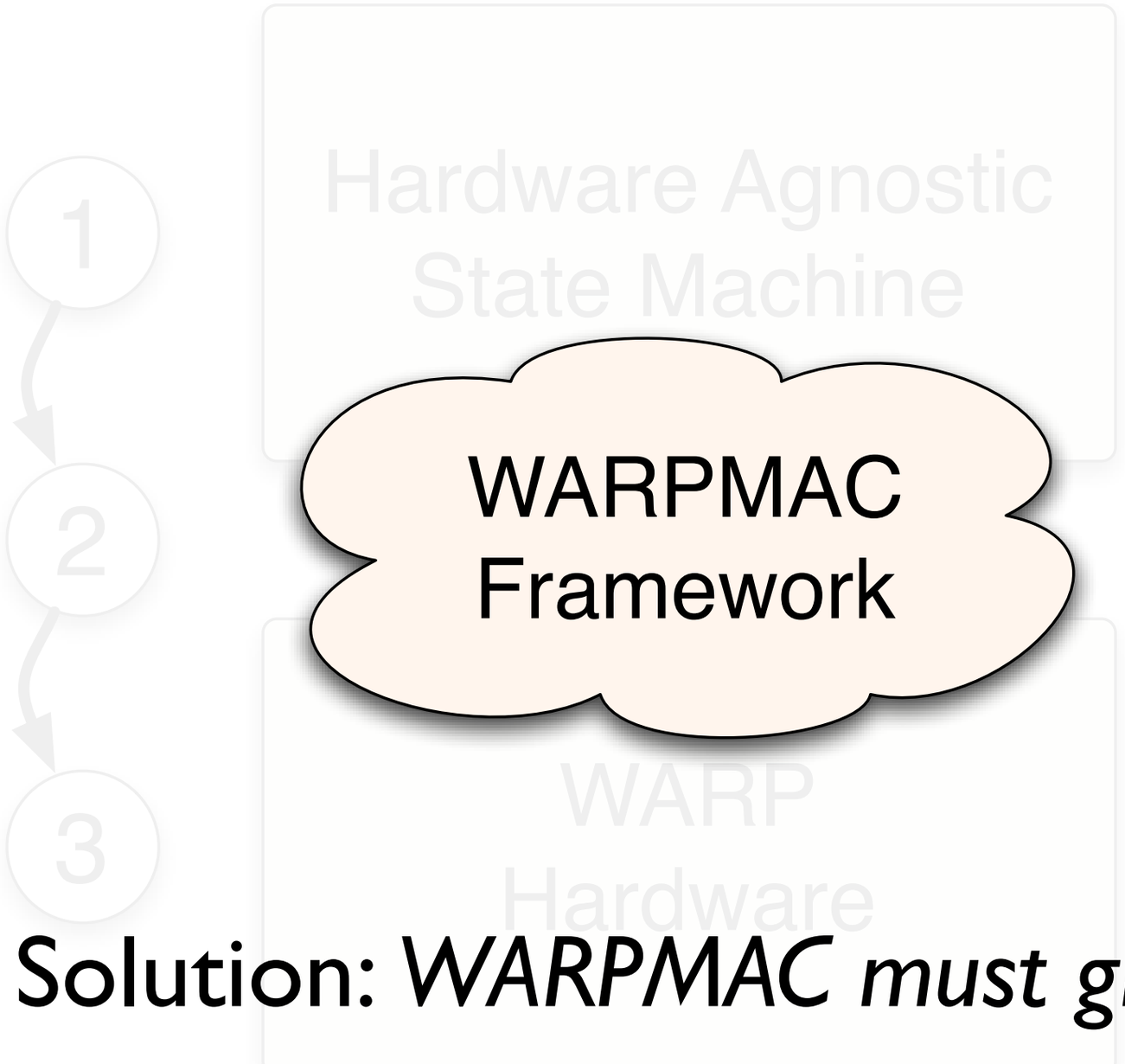
Design Realization



- 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

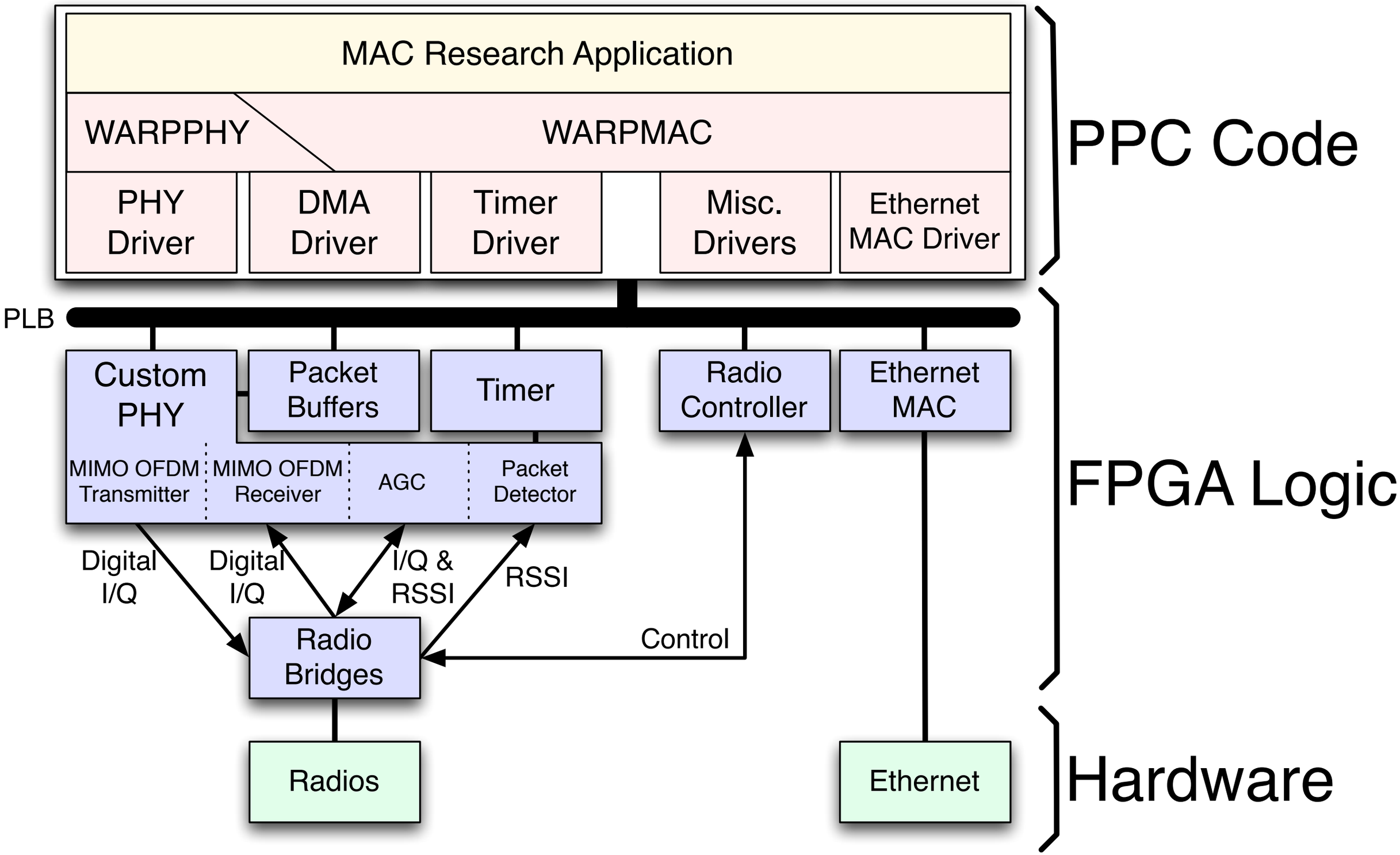
Reference Designs

- Snapshots of the WARP repository
- Free, open-source releases at regular intervals
  - Today's exercises are Reference Design v14.1
  - Keeps pace with Xilinx design tools

Reference Designs

- Snapshots of the WARP repository
- Free, open-source releases at regular intervals
  - Today's exercises are Reference Design v14.1
  - 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 for event-driven operation
  - Etc.


User Code
WARPMAC
WARPPHY
Drivers


User Code
WARPMAC
WARPPHY
Drivers

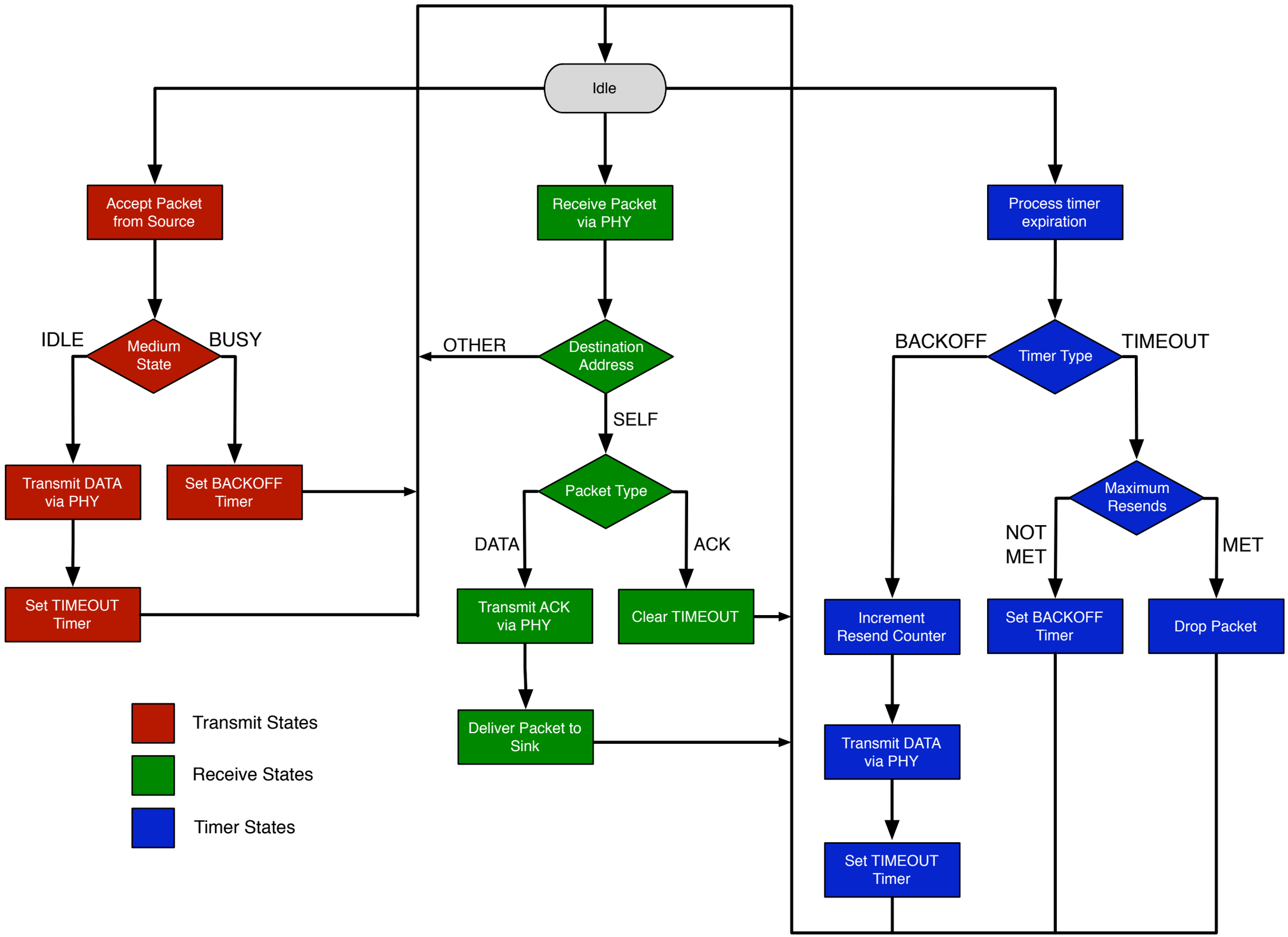
User-level MAC Algorithms:

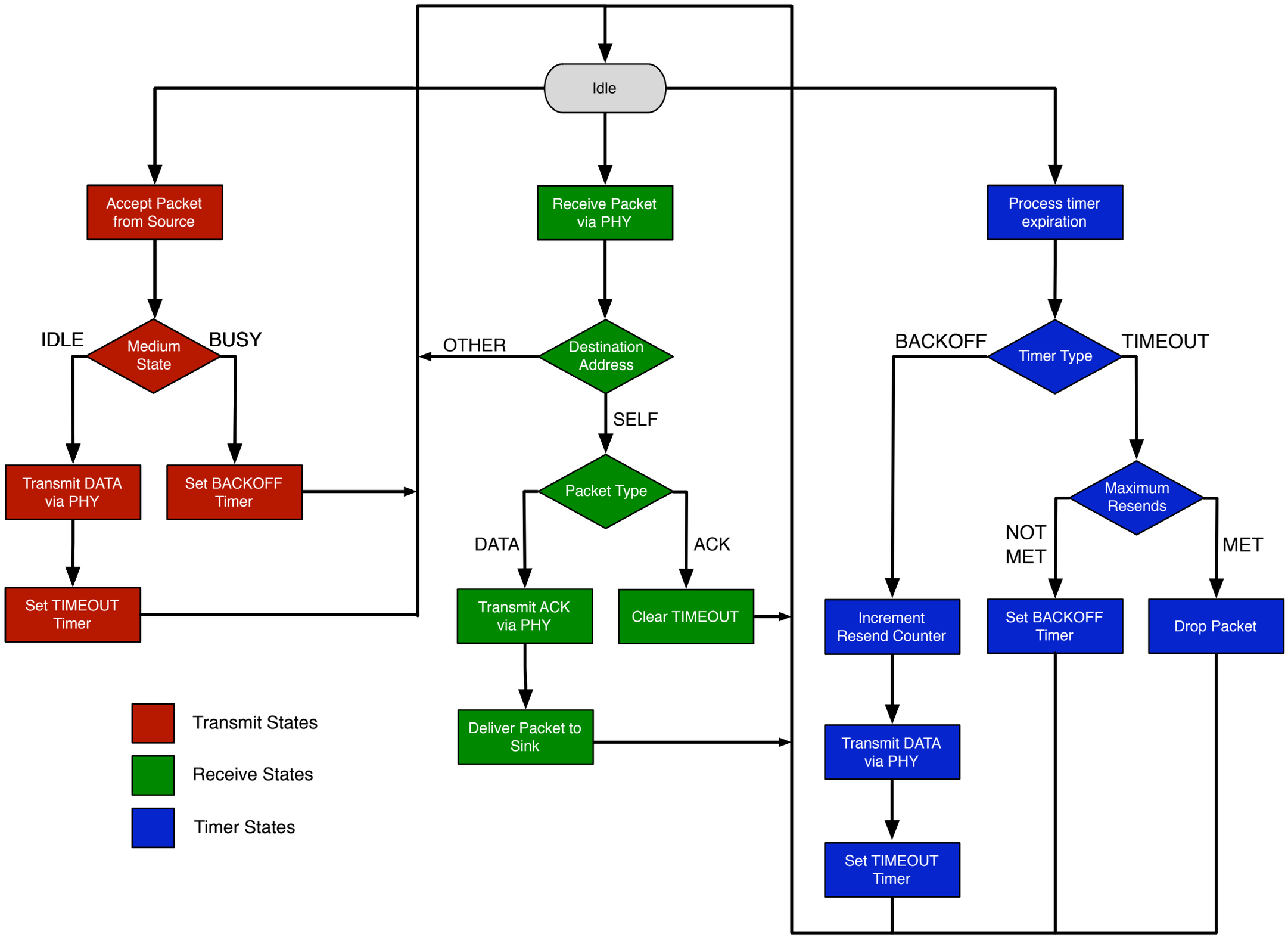
- High-level MAC algorithms
- Some examples so far:
  - Aloha
  - Carrier-sensing MAC
  - Opportunistic Auto-Rate (OAR)
  - MAC Workshop Exercises
  - Distributed On-demand Cooperation (DOC)

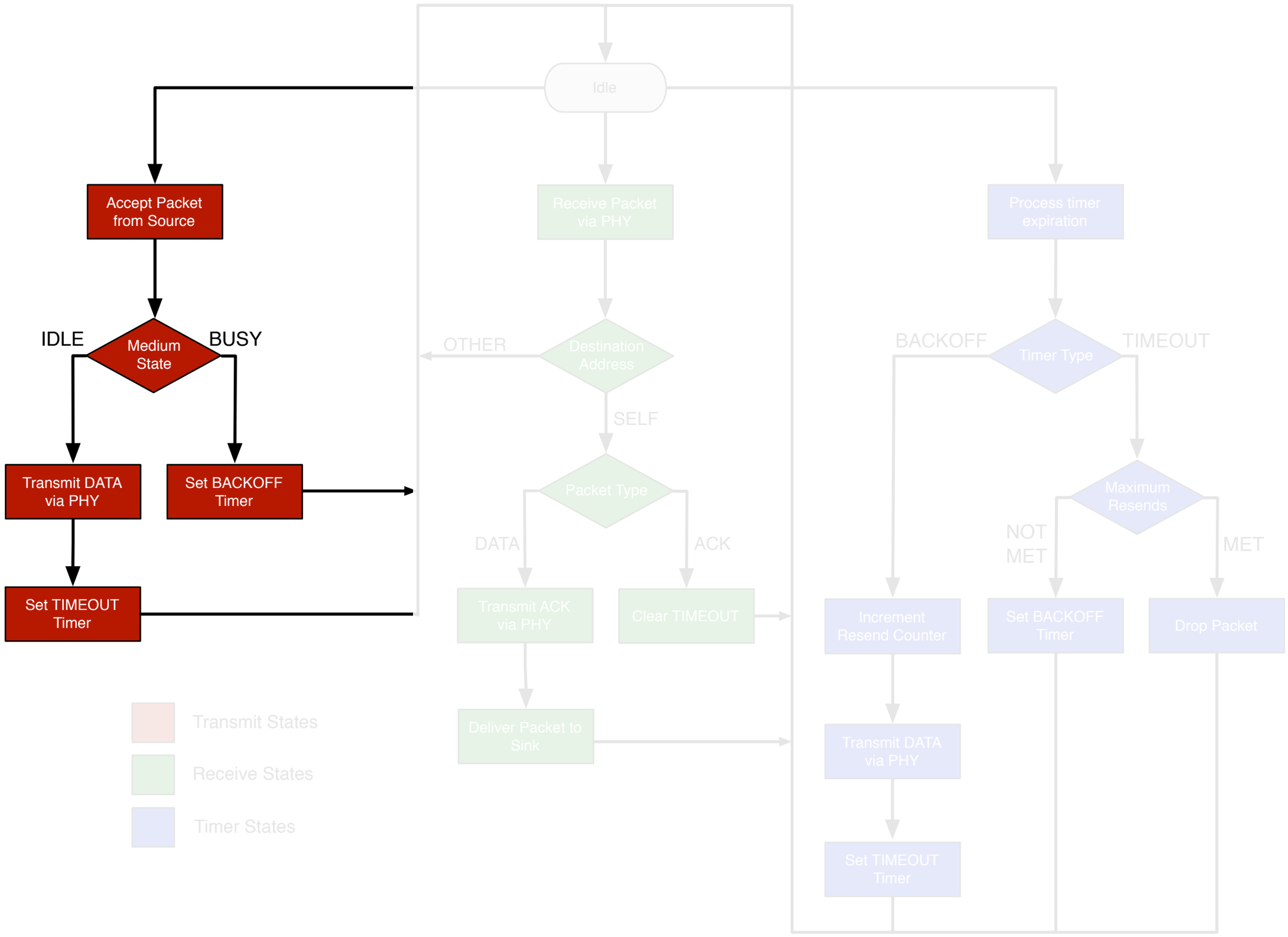

User Code
WARPMAC
WARPPHY
Drivers

An example: CSMA

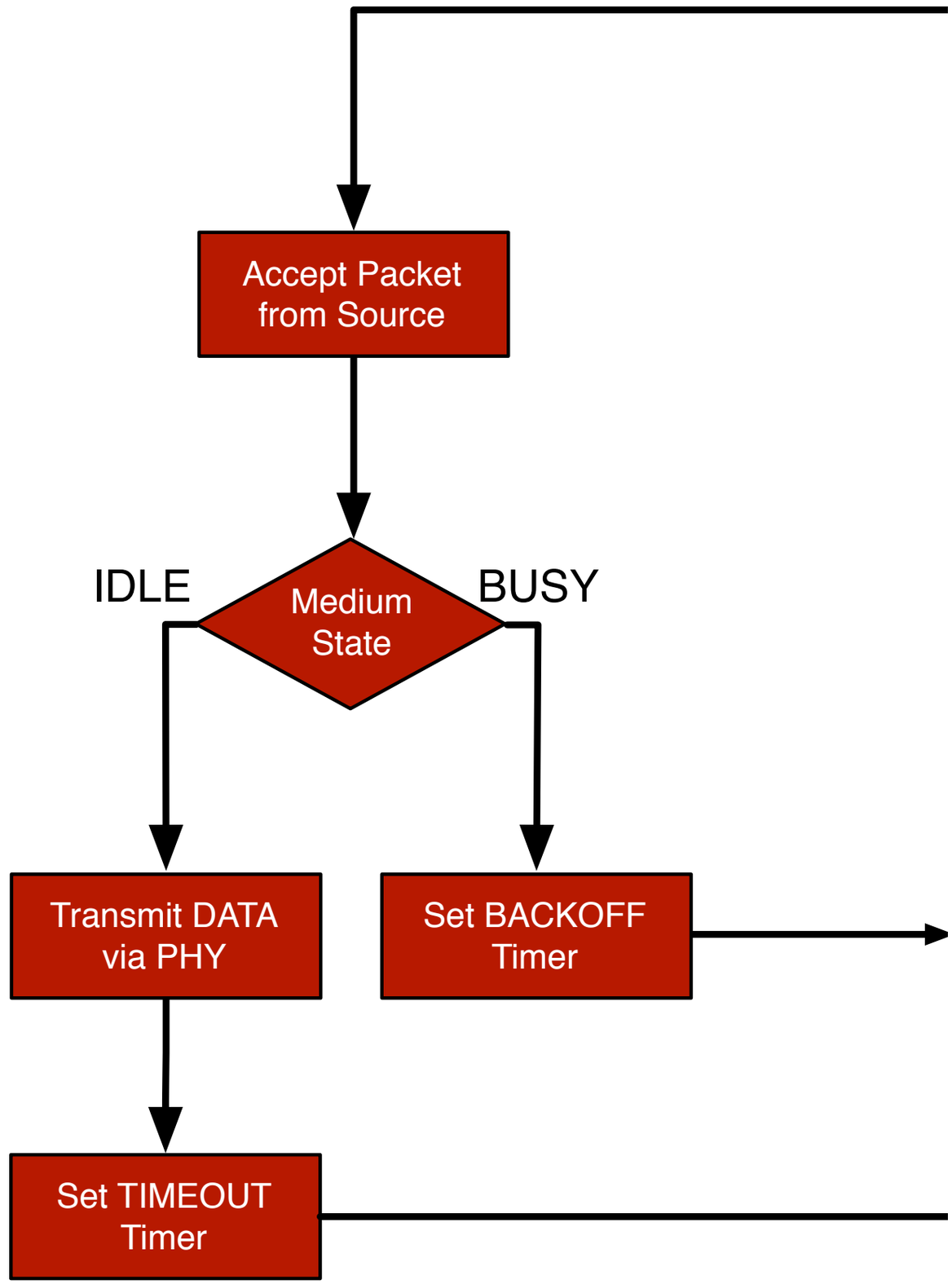
- Carrier-sensing Multiple Access
- Serves as a foundation for a large class of other random access protocols
- Fairly simple algorithm


Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
BACKOFF
Timer Type
TIMEOUT
SELF
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


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
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


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

Transmit States



Transmit States

warpmac_emacRx_handler

- Starts DMA transfer from EMAC to PHY

dataFromNetworkLayer_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
- Decrements remaining resend counter

}

If medium is busy {

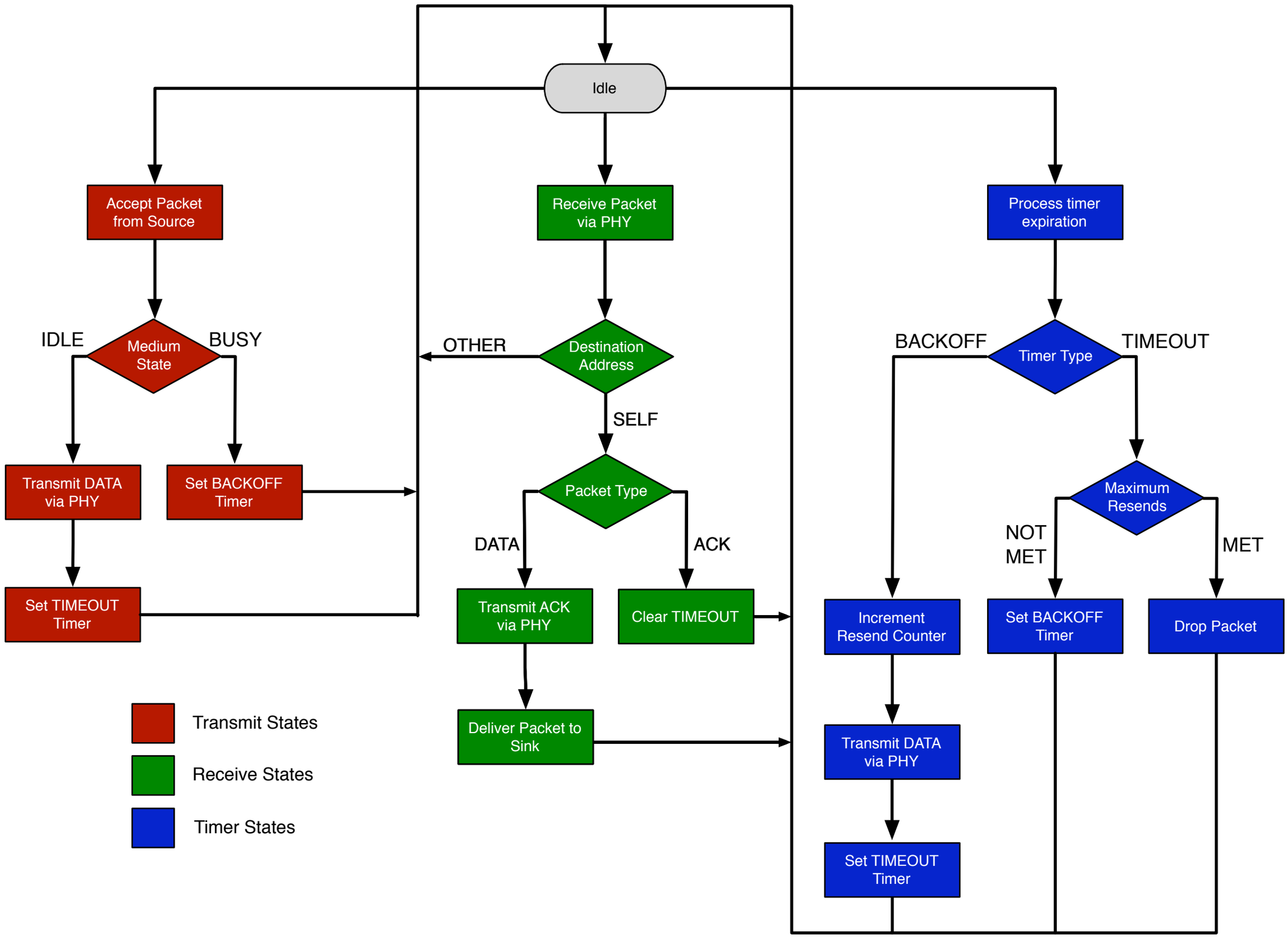
- Starts a backoff timer

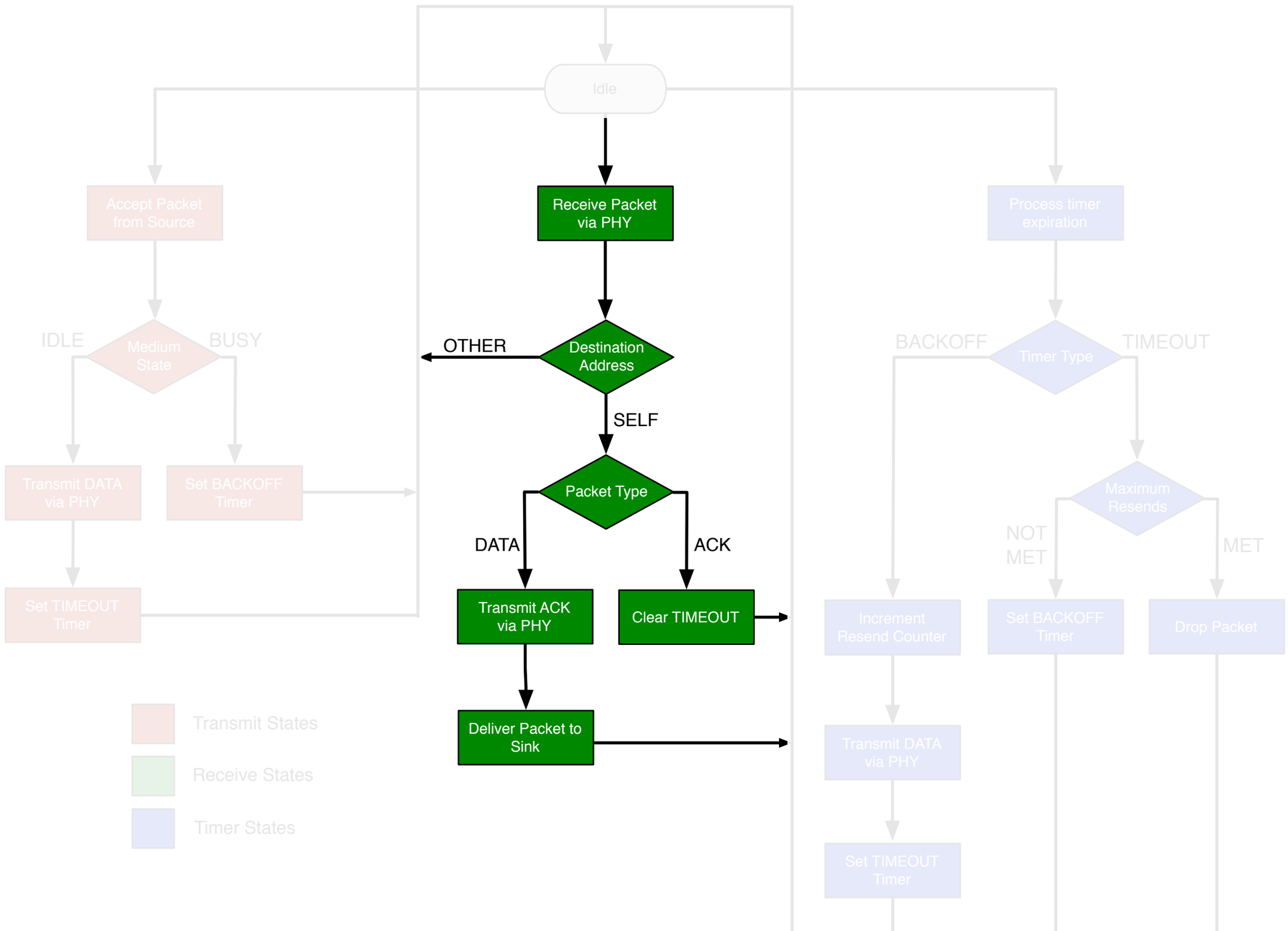
}

- Clears EMAC

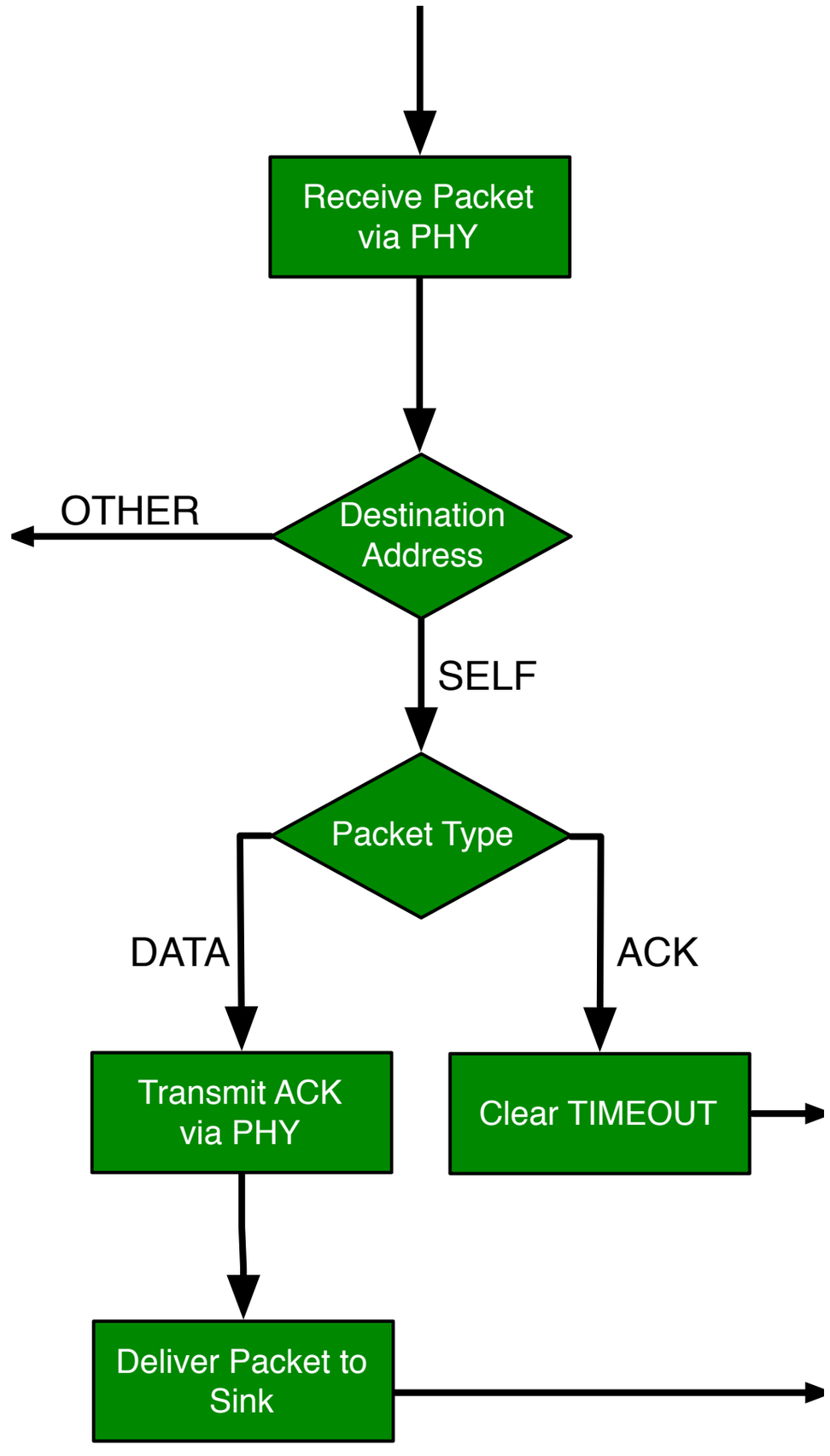
WARPMAC

User-Code

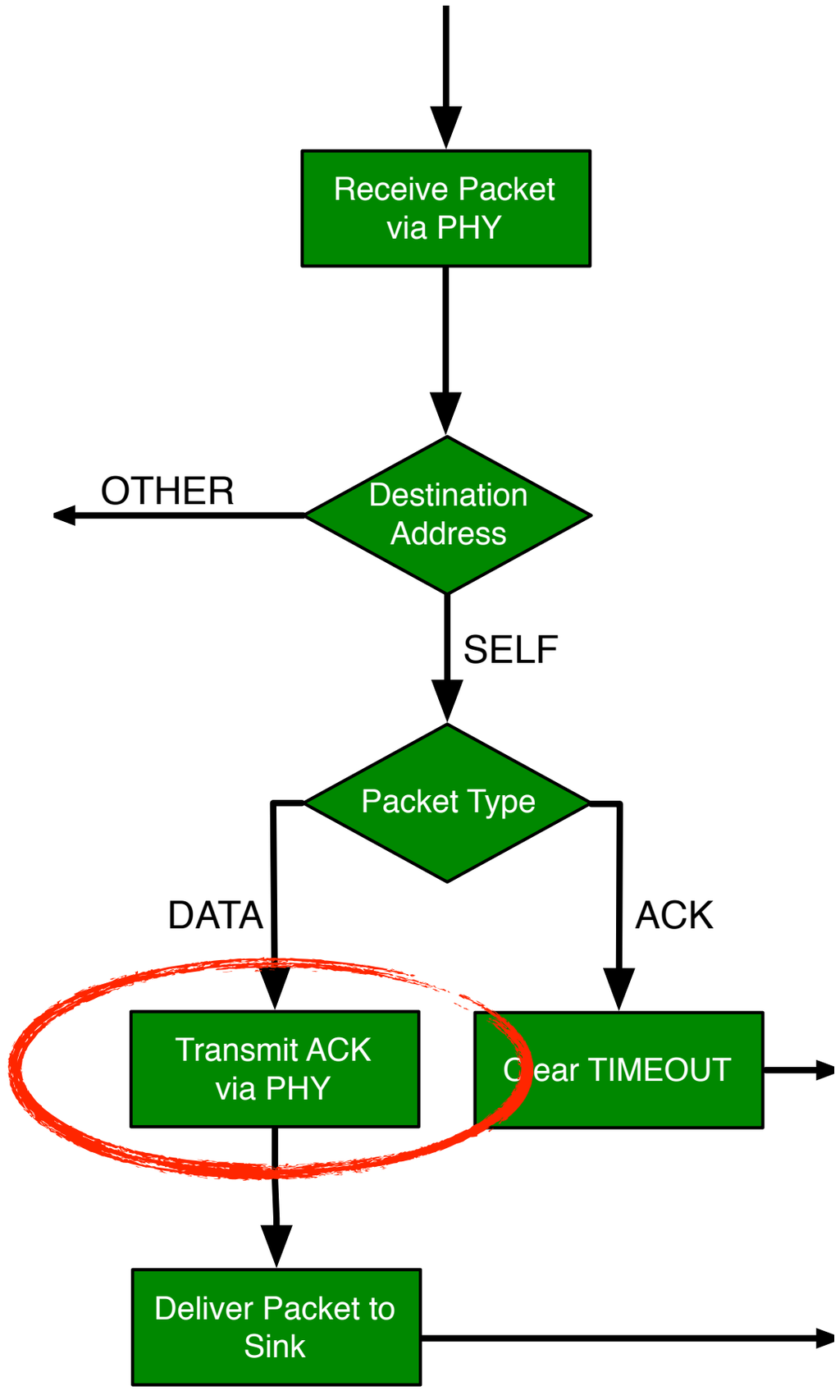

Idle
Accept Packet from Source
Receive Packet via PHY
Process timer expiration
IDLE
Medium State
BUSY
OTHER
Destination Address
BACKOFF
Timer Type
TIMEOUT
SELF
Transmit DATA via PHY
Set BACKOFF Timer
Packet Type
Maximum Resends
DATA
ACK
NOT MET
MET
Set TIMEOUT Timer
Transmit ACK via PHY
Clear TIMEOUT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


Idle
Accept Packet from Source
Process timer expiration
Receive Packet via PHY
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
Transmit States
Deliver Packet to Sink
Transmit DATA via PHY
Receive States
Timer States
Set TIMEOUT Timer

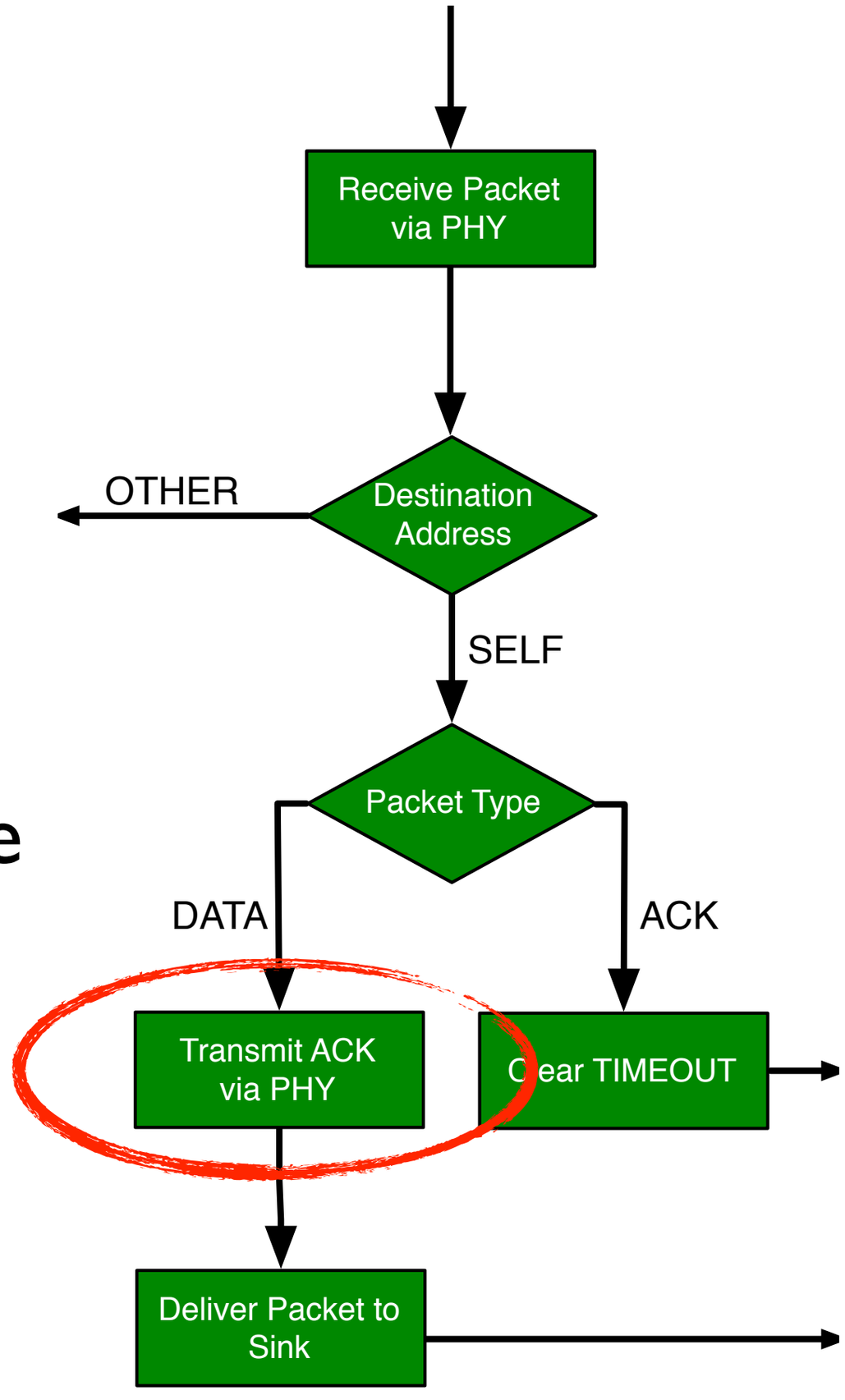
Receive States



Receive States



Receive States



How can we use WARPMAC to keep turn-around-time (TAT) small?

Receive States

warpmac_pollPhy

- Copies header into Macframe

phyRx_goodHeader_callback

- Checks address/type fields of Macframe header

If data {

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

If Good {

Send acknowledgement

warpmac_prepPktToNetwork

- Starts DMA transfer from PHY to EMAC

warpmac_finishPhyXmit

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

warpmac_startPktToNetwork

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

}

}

If acknowledgment {

- Clears timeout timer

}

- Resets PHY

WARPMAC

User-Code

Receive States

warpmac_pollPhy

- Copies header into Macframe

phyRx_goodHeader_callback

- Checks address/type fields of Macframe header

If data {

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

If Good {

Send acknowledgement

warpmac_prepPktToNetwork

- Starts DMA transfer from PHY to EMAC

warpmac_finishPhyXmit

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

warpmac_startPktToNetwork

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

}

}

If acknowledgment {

- Clears timeout timer

}

- Resets PHY

Fast Turn-Around Time (TAT)

WARPMAC

User-Code

Receive States

warpmac_pollPhy

- Copies header into Macframe

phyRx_goodHeader_callback

- Checks address/type fields of Macframe header

If data {

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

If Good {

Send acknowledgement

warpmac_prepPktToNetwork

- Starts DMA transfer from PHY to EMAC

warpmac_finishPhyXmit

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

warpmac_startPktToNetwork

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

}

}

If acknowledgment {

- Clears timeout timer

}

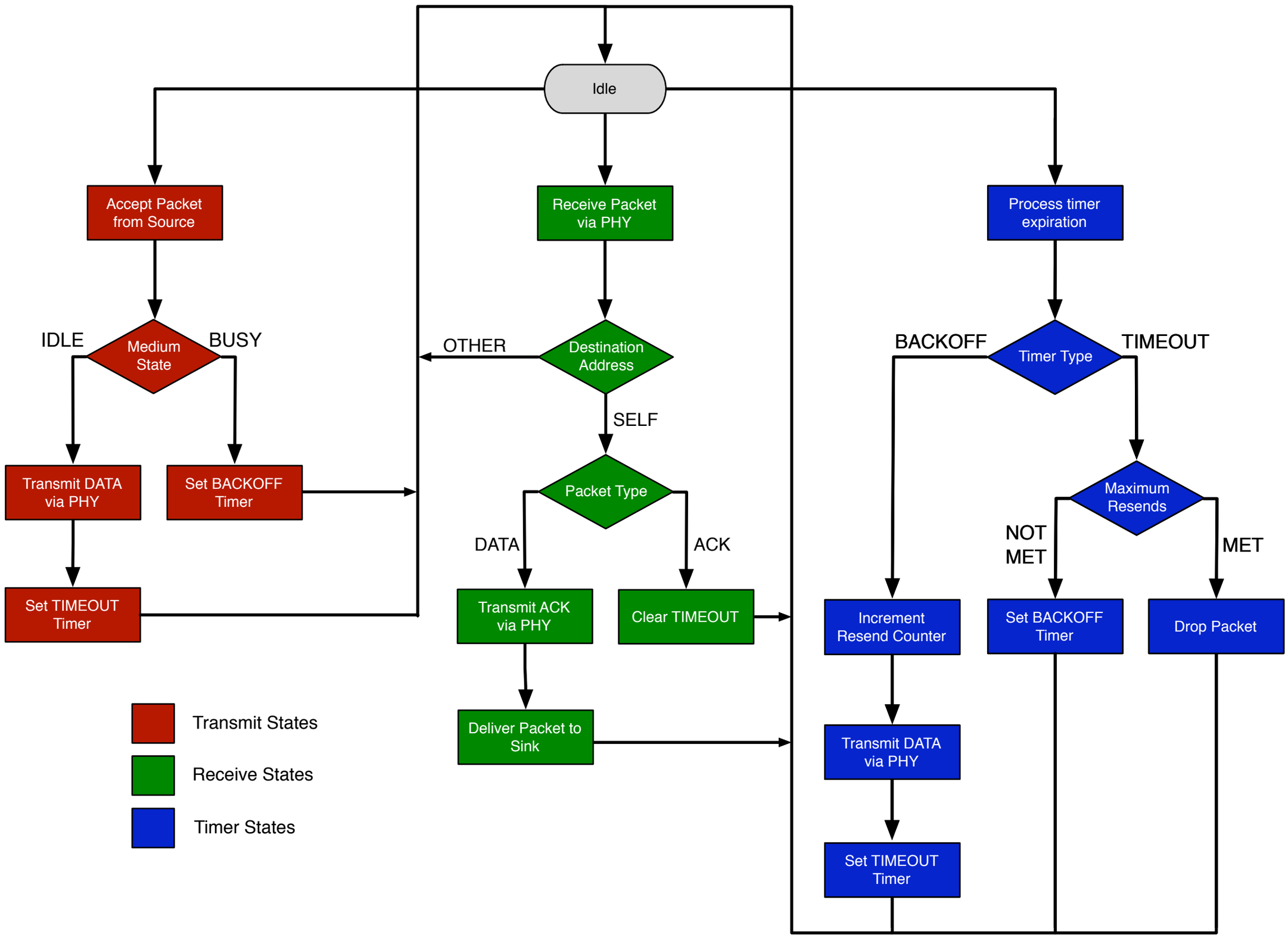
- Resets PHY

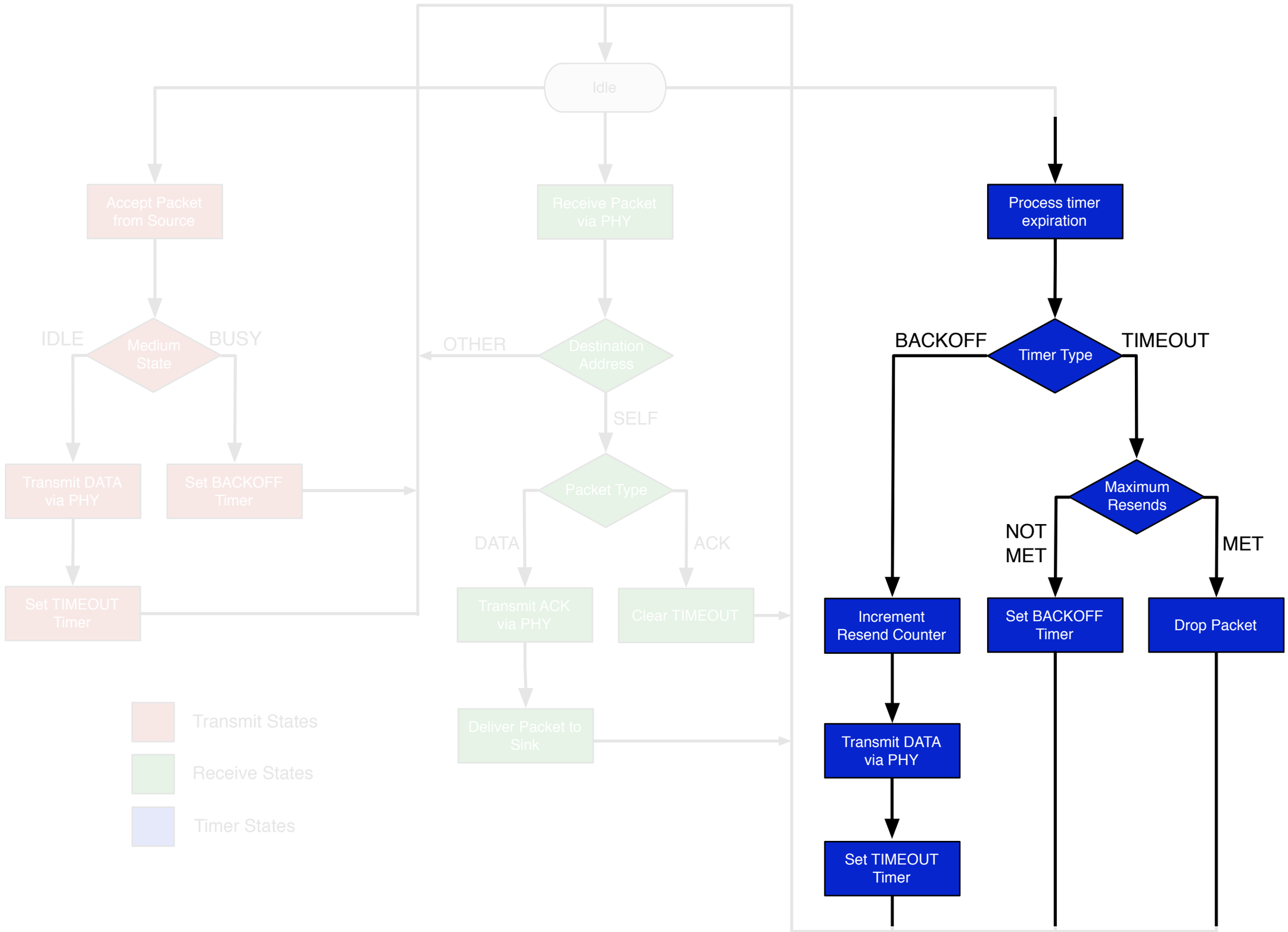
Fast Turn-Around Time (TAT)

1) Software calls
2) Hardened "autoresponder"

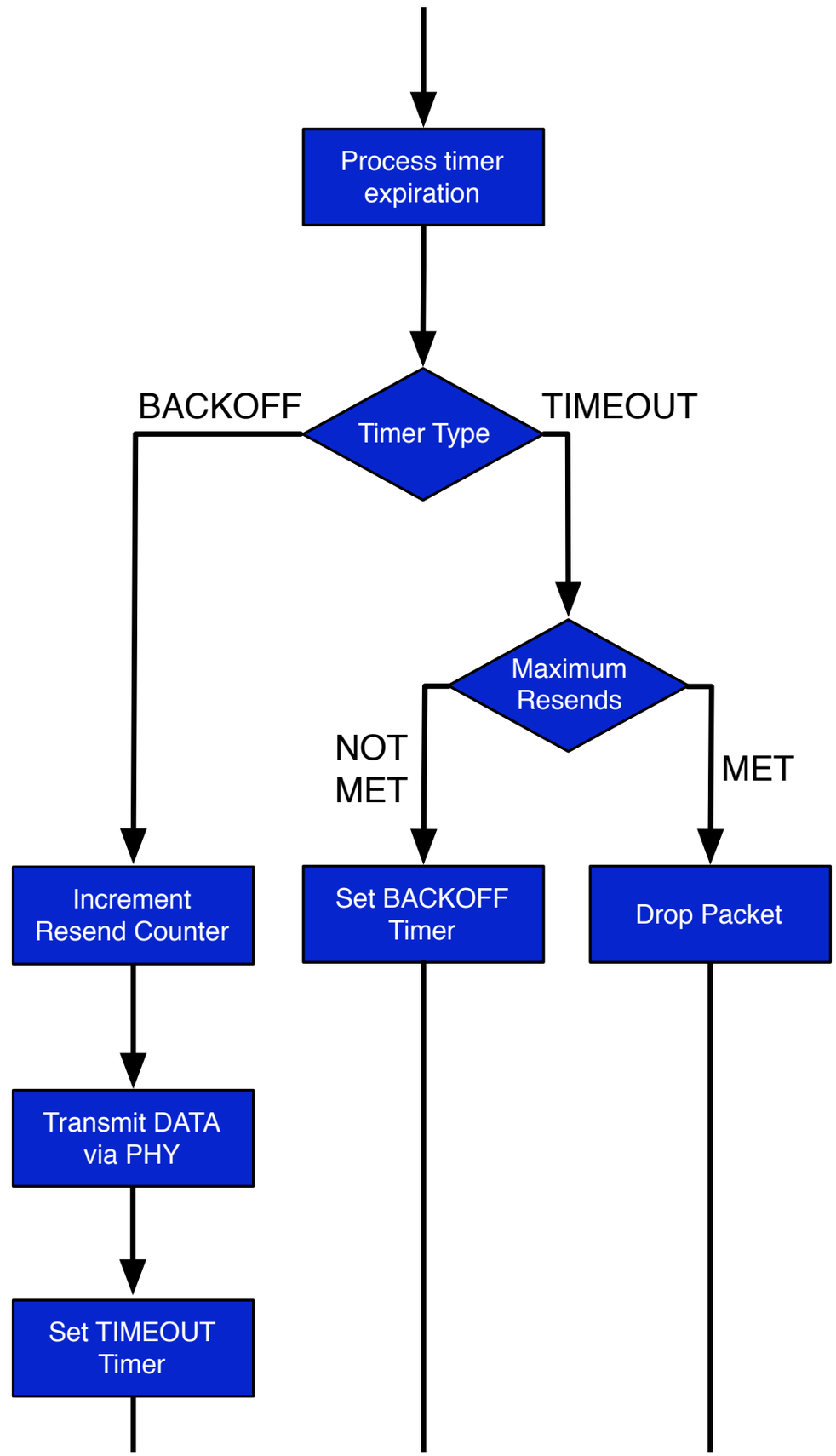
WARPMAC

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
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
Transmit States
Deliver Packet to Sink
Transmit DATA via PHY
Receive States
Timer States
Set TIMEOUT Timer

Timer States



Timer States

warpmac_pollTimer

- Checks each timer status and calls relevant callbacks

timer_callback

- Checks timer type

If timeout {

- 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

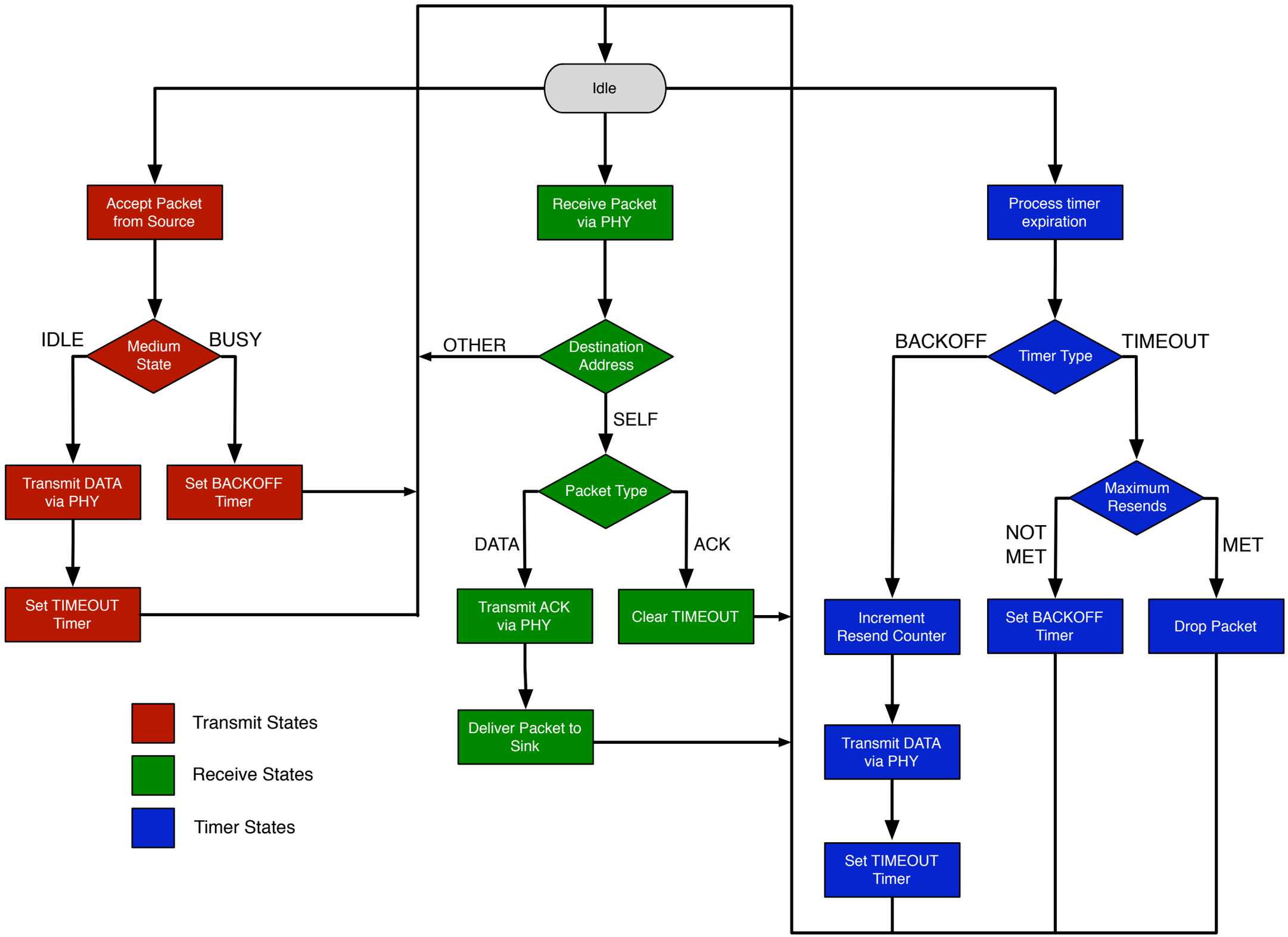
- Starts a timeout timer
- Decrements remaining resend counter

}

- Clears timers

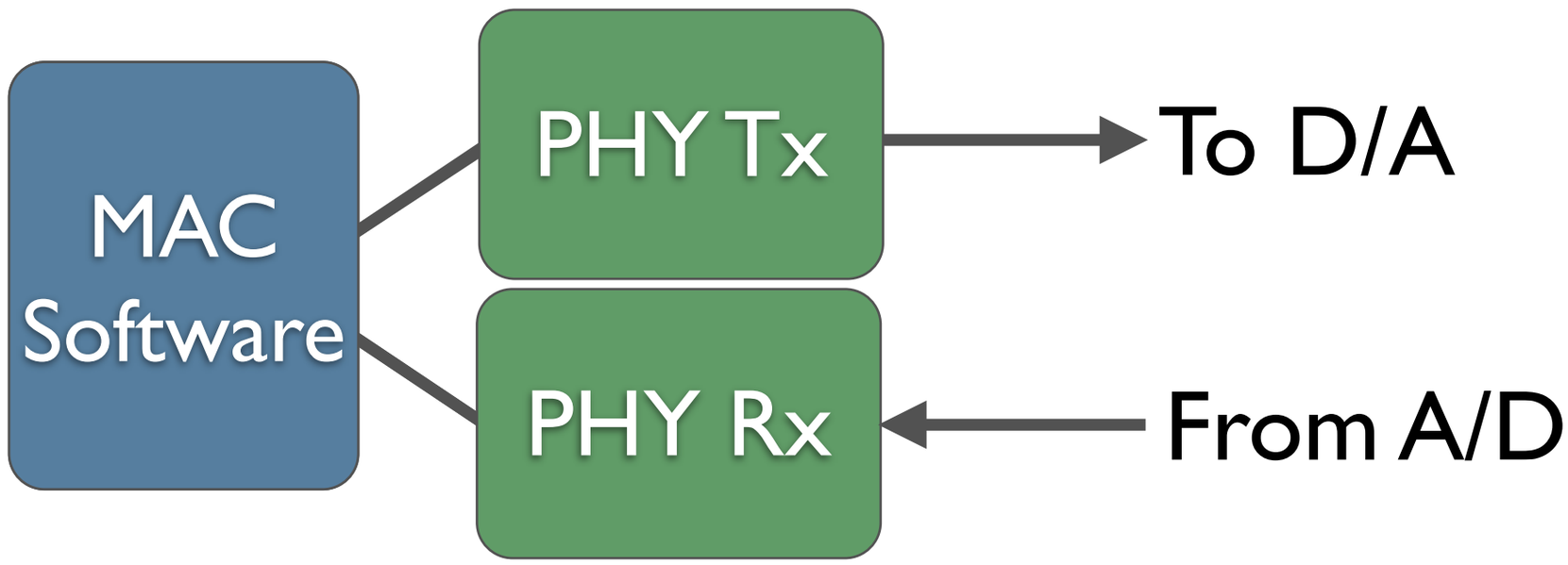
WARPMAC

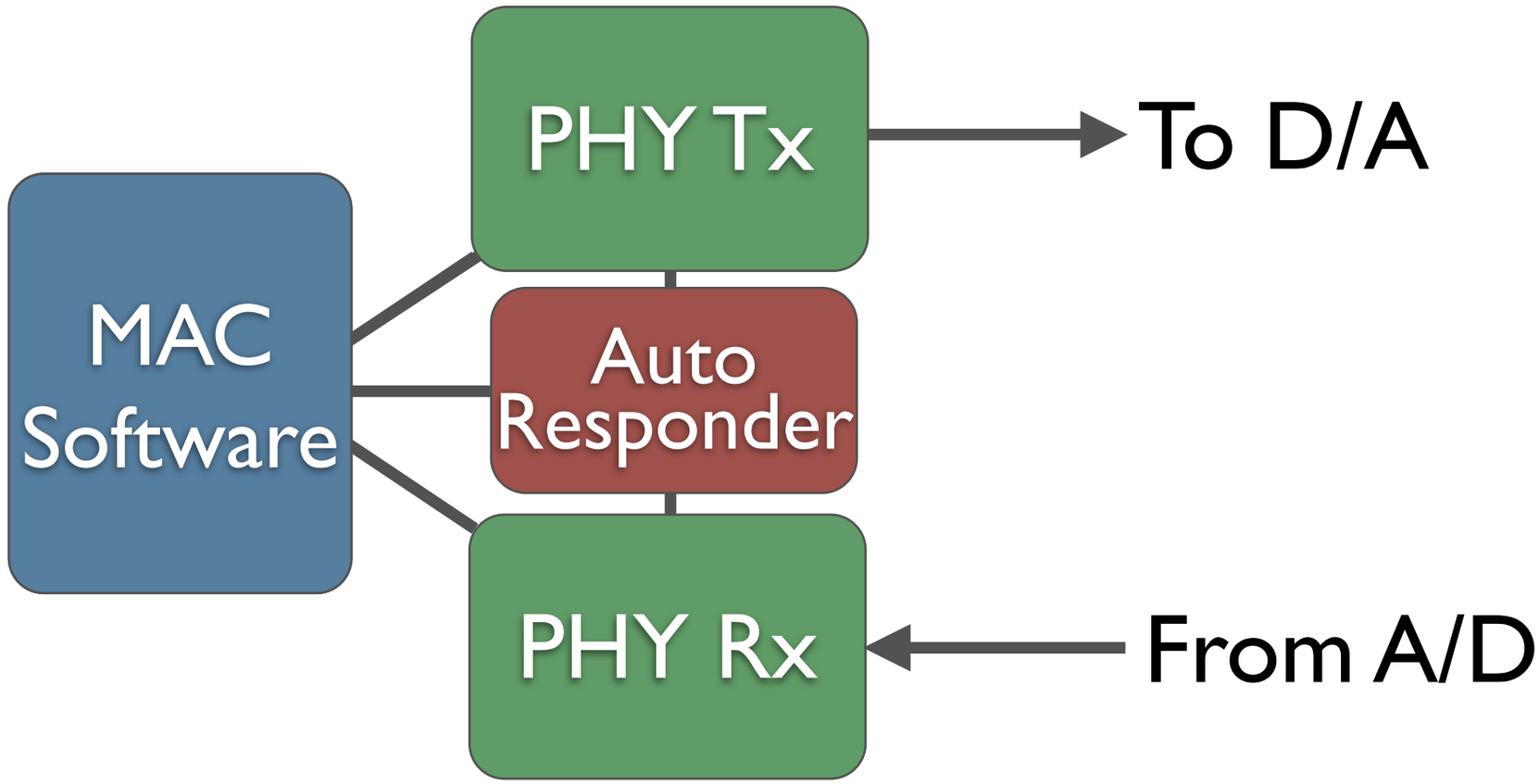
User-Code


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
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

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

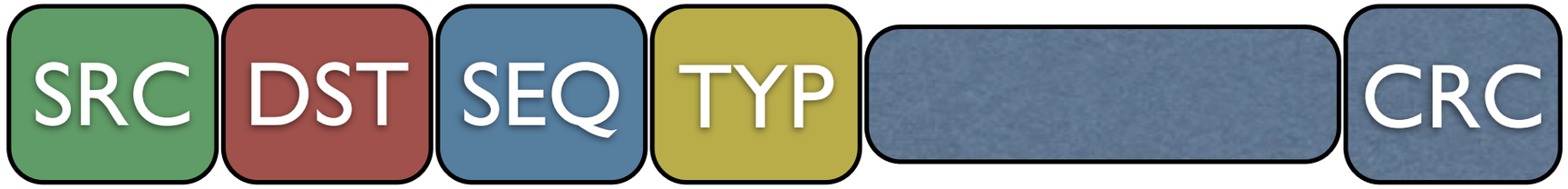
http://warp.rice.edu/trac/browser/ResearchApps/MAC/CSMAMAC/csmaMac.c

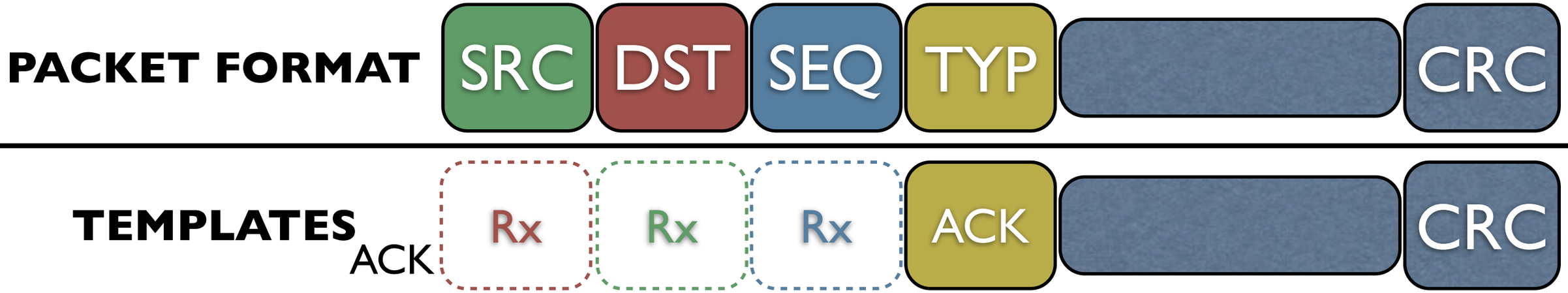

MAC
Software
PHY Tx
PHY Rx
To D/A
From A/D



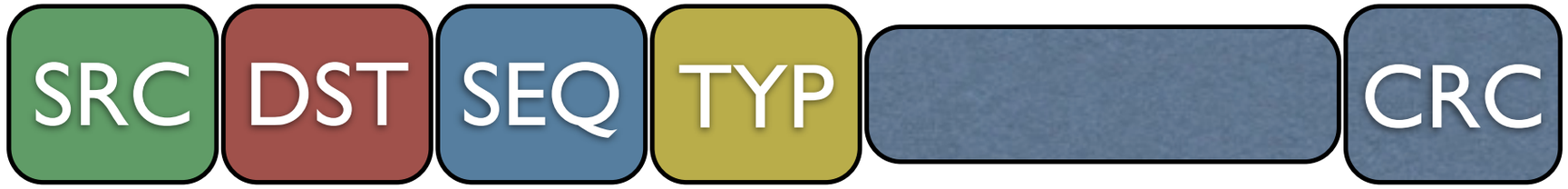
MAC specifies packet **templates**, Rx packet **conditions** and Tx header **substitution**. PHY initiates transmission automatically.

PACKET FORMAT

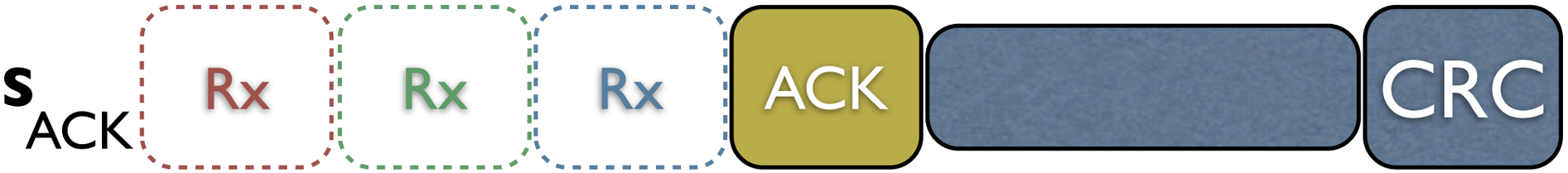



PACKET FORMAT
SRC
DST
SEQ
TYP
CRC
TEMPLATES
ACK
Rx
Rx
Rx
ACK
CRC

PACKET FORMAT



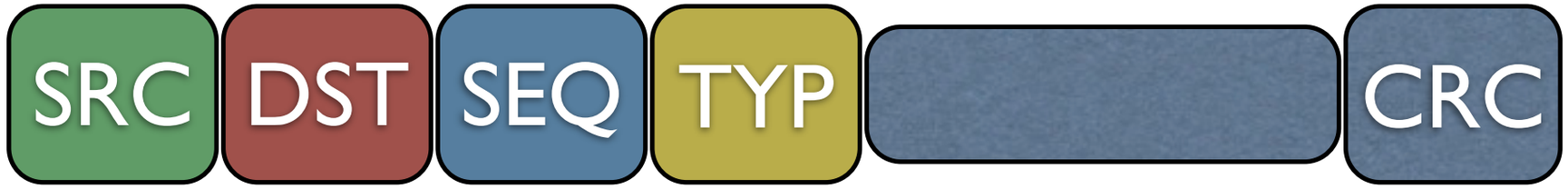
TEMPLATES



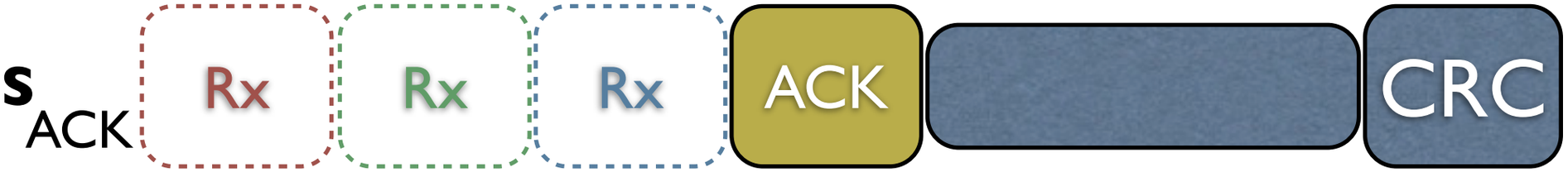
CONDITIONS

Template	Addressed to Node 1	Good Payload	Bad Payload	Packet Type
ACK	X	X		DATA

PACKET FORMAT



TEMPLATES

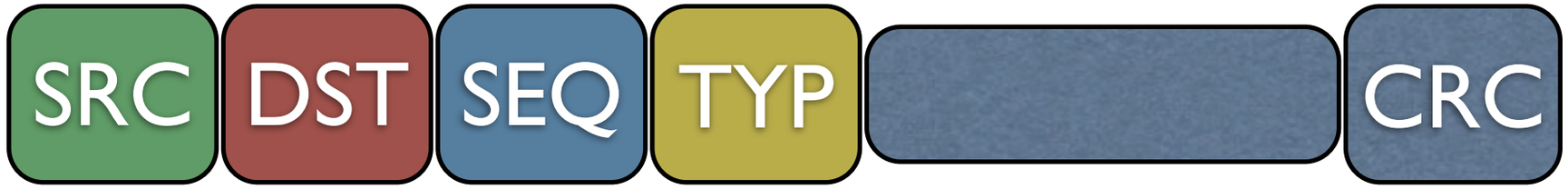


CONDITIONS

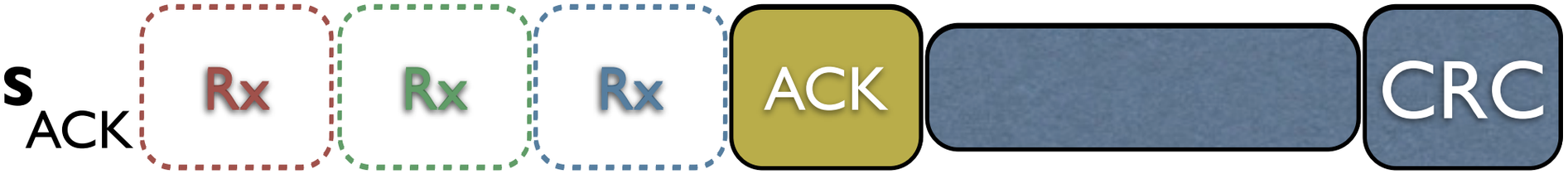
Template	Addressed to Node 1	Good Payload	Bad Payload	Packet Type
ACK	X	X		DATA

Rx PACKET

PACKET FORMAT



TEMPLATES



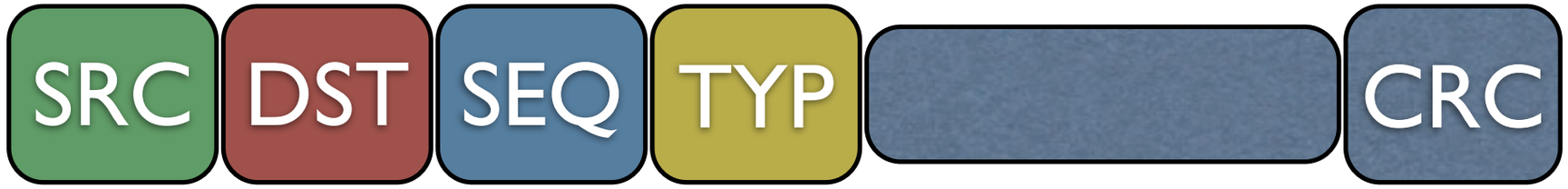
CONDITIONS

Template	Addressed to Node 1	Good Payload	Bad Payload	Packet Type
ACK	X	X		DATA

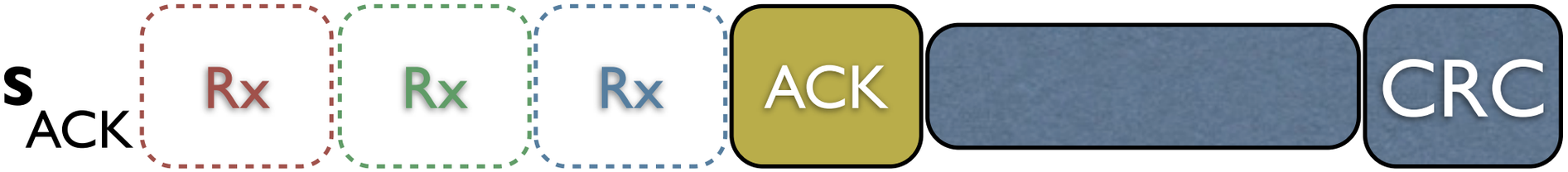
Rx PACKET

Tx PACKET

PACKET FORMAT



TEMPLATES



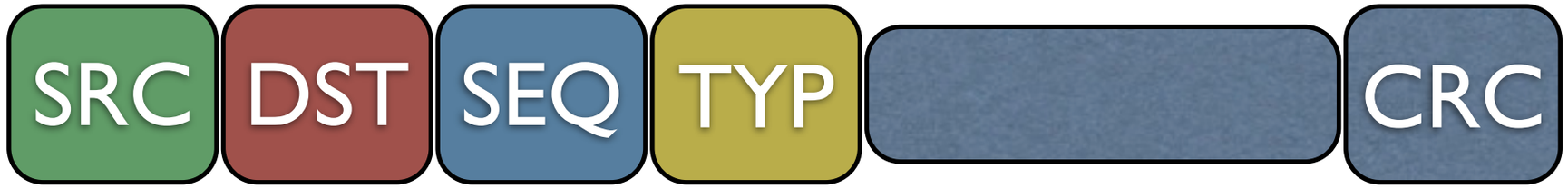
CONDITIONS

Template	Addressed to Node 1	Good Payload	Bad Payload	Packet Type
ACK	X	X		DATA

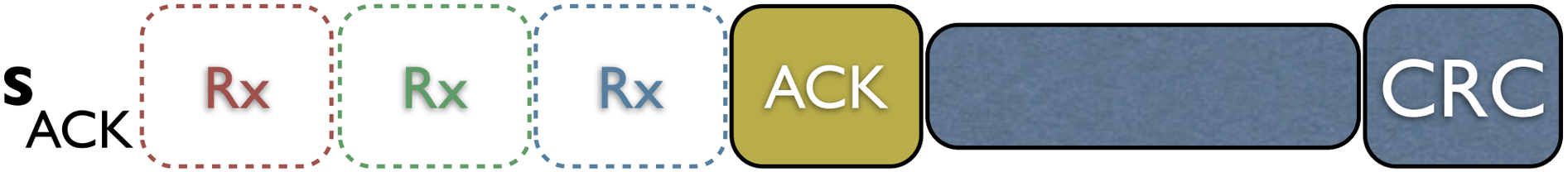
Rx PACKET

Tx PACKET

PACKET FORMAT



TEMPLATES



CONDITIONS

Template	Addressed to Node 1	Good Payload	Bad Payload	Packet Type
ACK	X	X		DATA

Rx PACKET

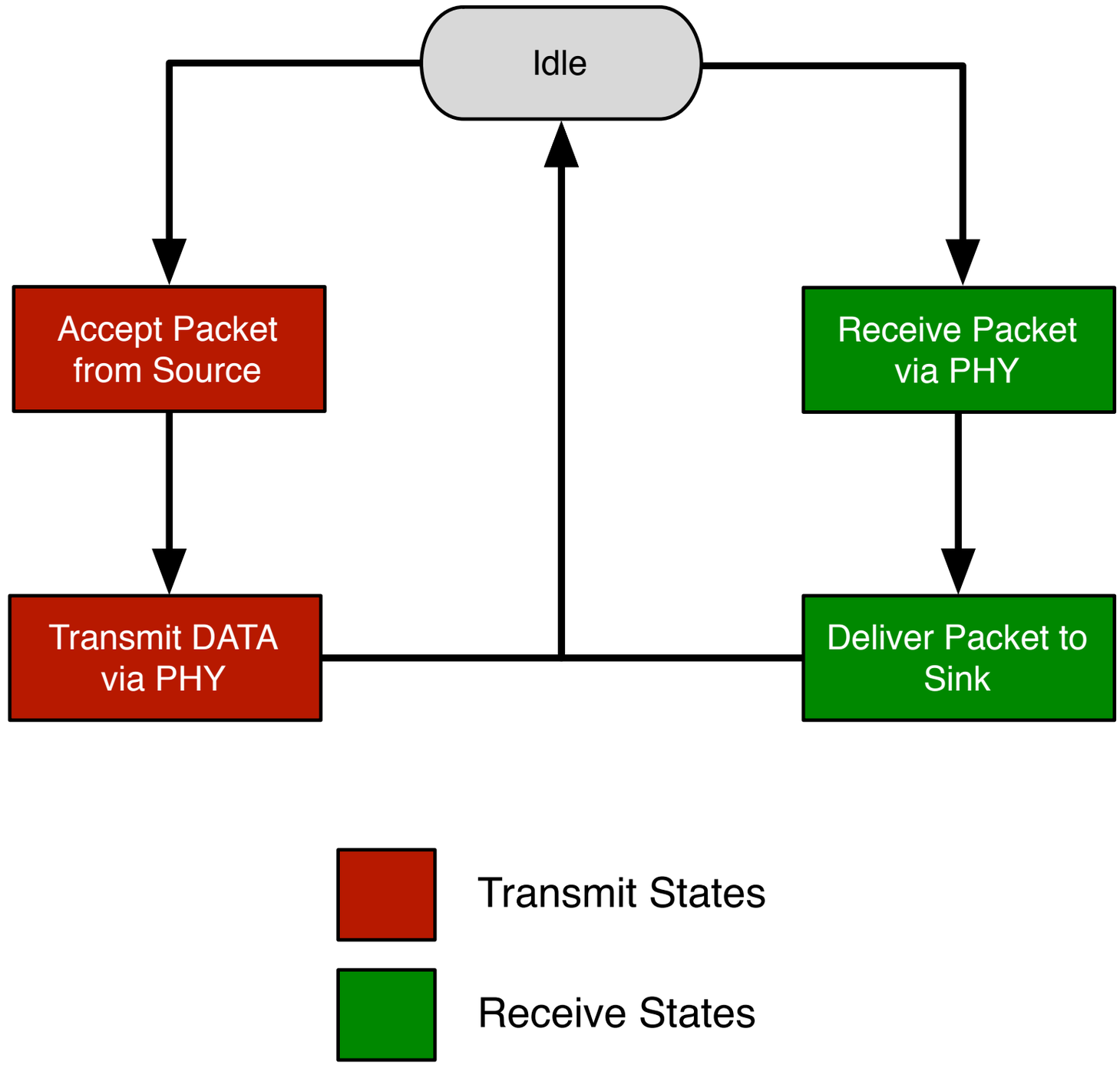
Tx PACKET

Questions?

Lab Exercises

Lab 4	noMAC	Too simple to be a MAC
Lab 5	halfMAC SW	Reception-half of a MAC (using software calls for ACK generation
Lab 6	halfMAC HW	Reception-half of a MAC (using autoresponder for ACK generation
Lab 6+	fullMAC	Add transmission half to previous lab

noMac



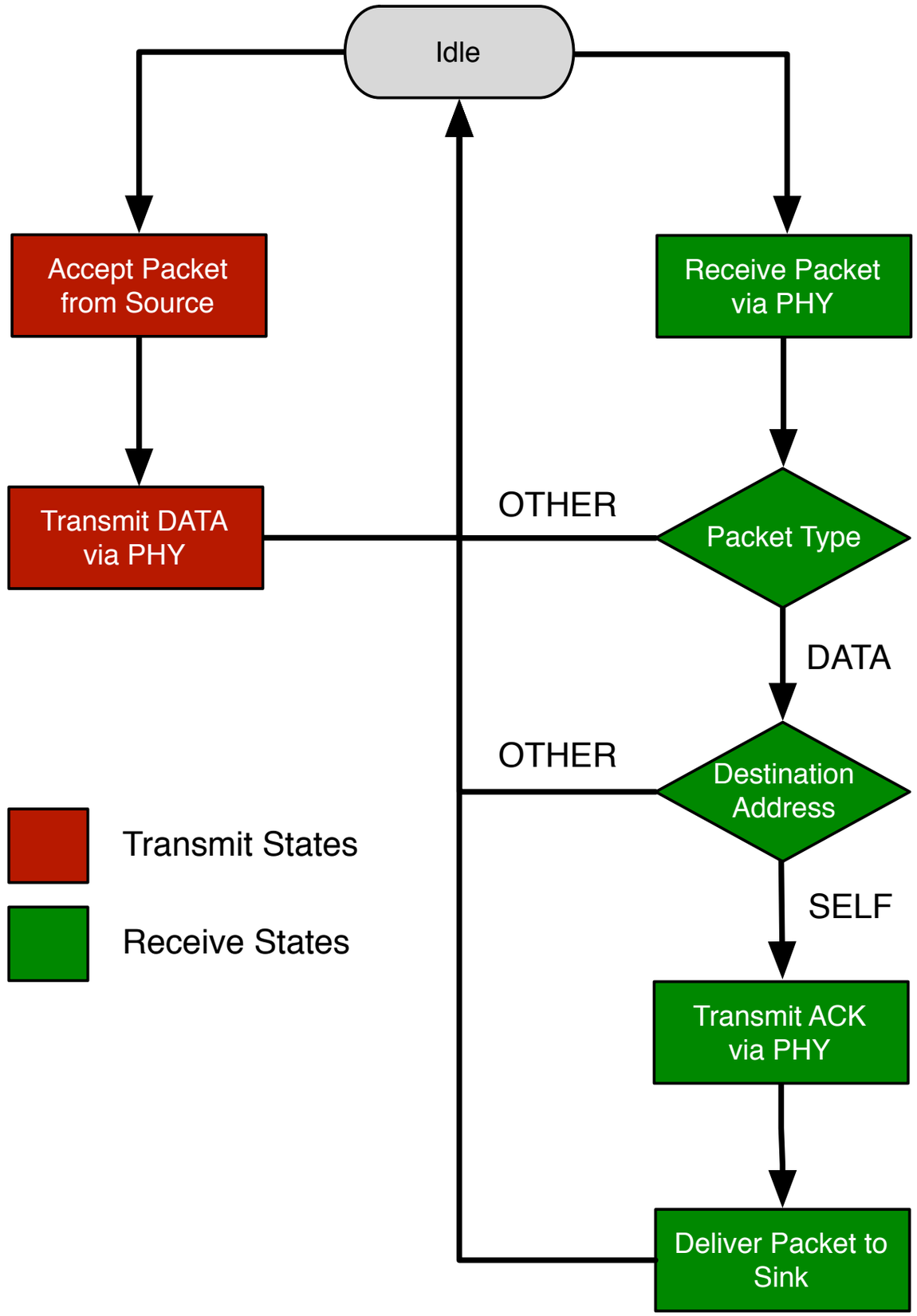
To test your noMac code, ping 10.0.0.20

Questions?

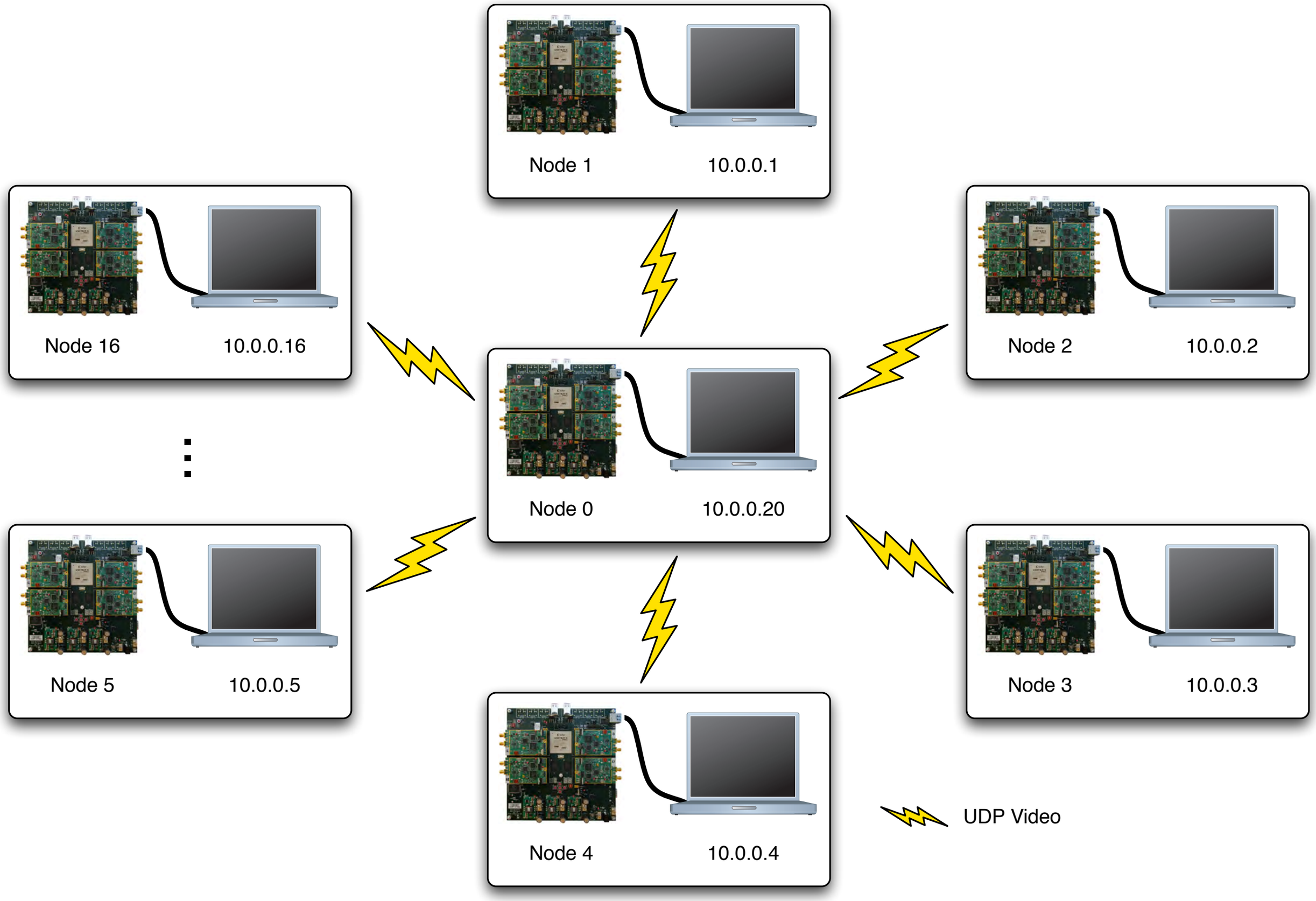
Remember to use the API:

http://warp.rice.edu/WARP_API

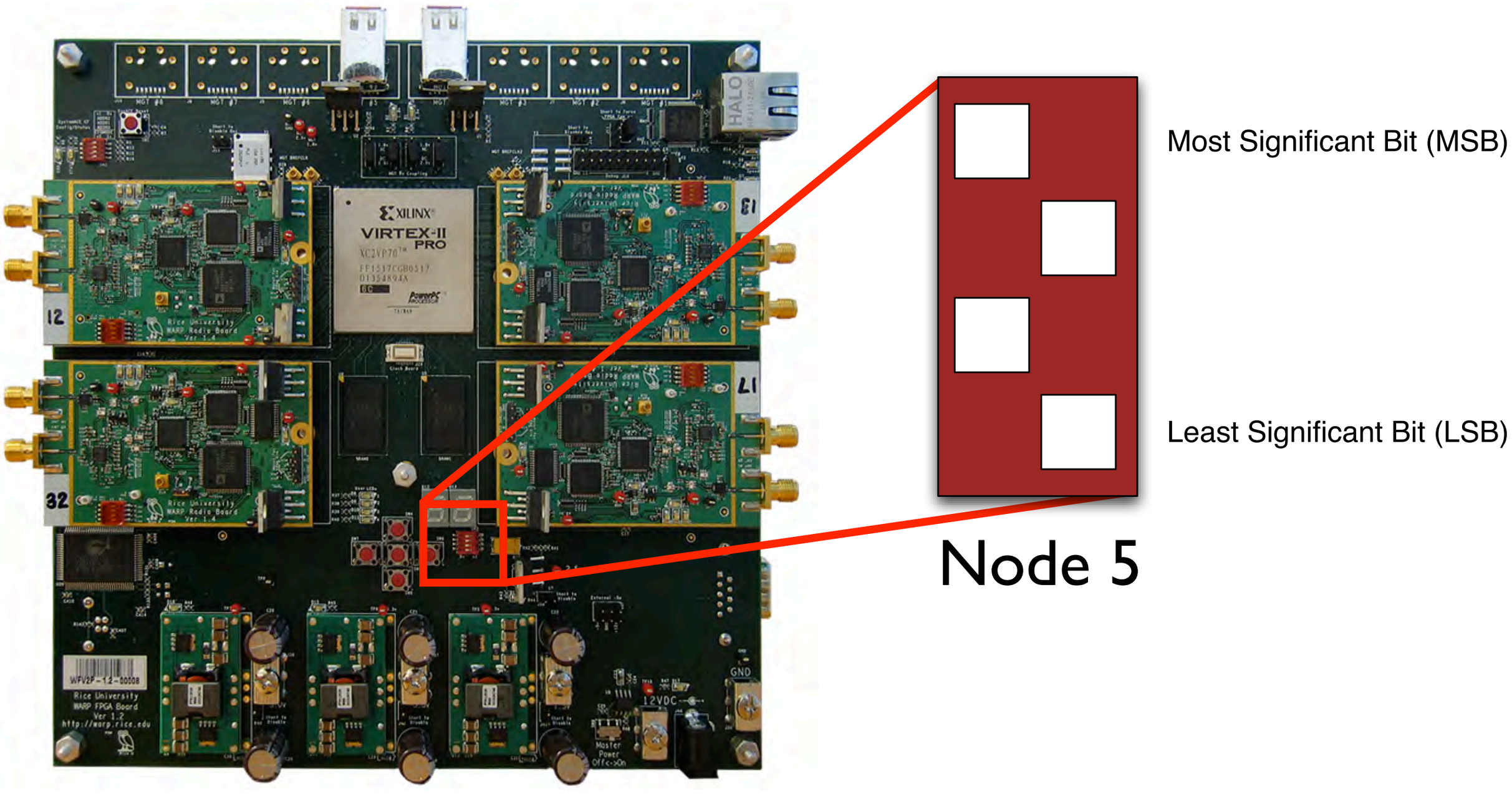
halfMac


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
10.0.0.1
Node 16
10.0.0.16
Node 2
10.0.0.2
Node 0
10.0.0.20
Node 5
10.0.0.5
Node 3
10.0.0.3
Node 4
10.0.0.4
UDP Video

halfMac


XILINX
VIRTEX-II PRO
Rice University
WARP Radio Board
Ver 1.4
WFV2P-1.2-00008
Rice University
WARP FPGA Board
Ver 1.2
http://warp.rice.edu
12VDC
GND
Master Power Off<->On
Most Significant Bit (MSB)
Least Significant Bit (LSB)
Node 5

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/)