CENG797 Ad-Hoc Networks
ALOHANet Implementation on USRP-AHC

Berkay Demirören

Wireless Systems, Networks and Cybersecurity Laboratory
Department of Computer Engineering
Middle East Technical University
Ankara Turkey

Outline of the Presentation

Agenda

Introduction

- Ad-Hoc Networks
- ALOHANet

Introduction

In this scope of study, I will try to explain and analyze how ALOHANet (Additive Links On-line Hawaii Area) mac-layer protocol will be implemented on USRP-AHC infrastructure

This study uses ahc python module as a software layer. On the bottom of that OFDM-USRP Component employed as a hardware layer. While, physical, linking and application layers are directly employed by AHC module MAC layer which is ALOHA in this case designed and implemented by myself.

Agenda

Ad-Hoc Networks

Ad-Hoc networks are decentralized type of wireless networks. It does not rely on a pre-existing infrastructure, such as routers or access points in wireless networks. Instead, each node participates in routing by forwarding data for other nodes, so the decision of which nodes forward data is made dynamically on the basis of network connectivity and the routing algorithm in use.

ALOHANet

ALOHANet, also known as the ALOHA System, was a pioneering computer networking system developed at the University of Hawaii. ALOHAnet became operational in June 1971, providing the first public demonstration of a wireless packet data network.ALOHA originally stood for Additive Links On-line Hawaii Area.

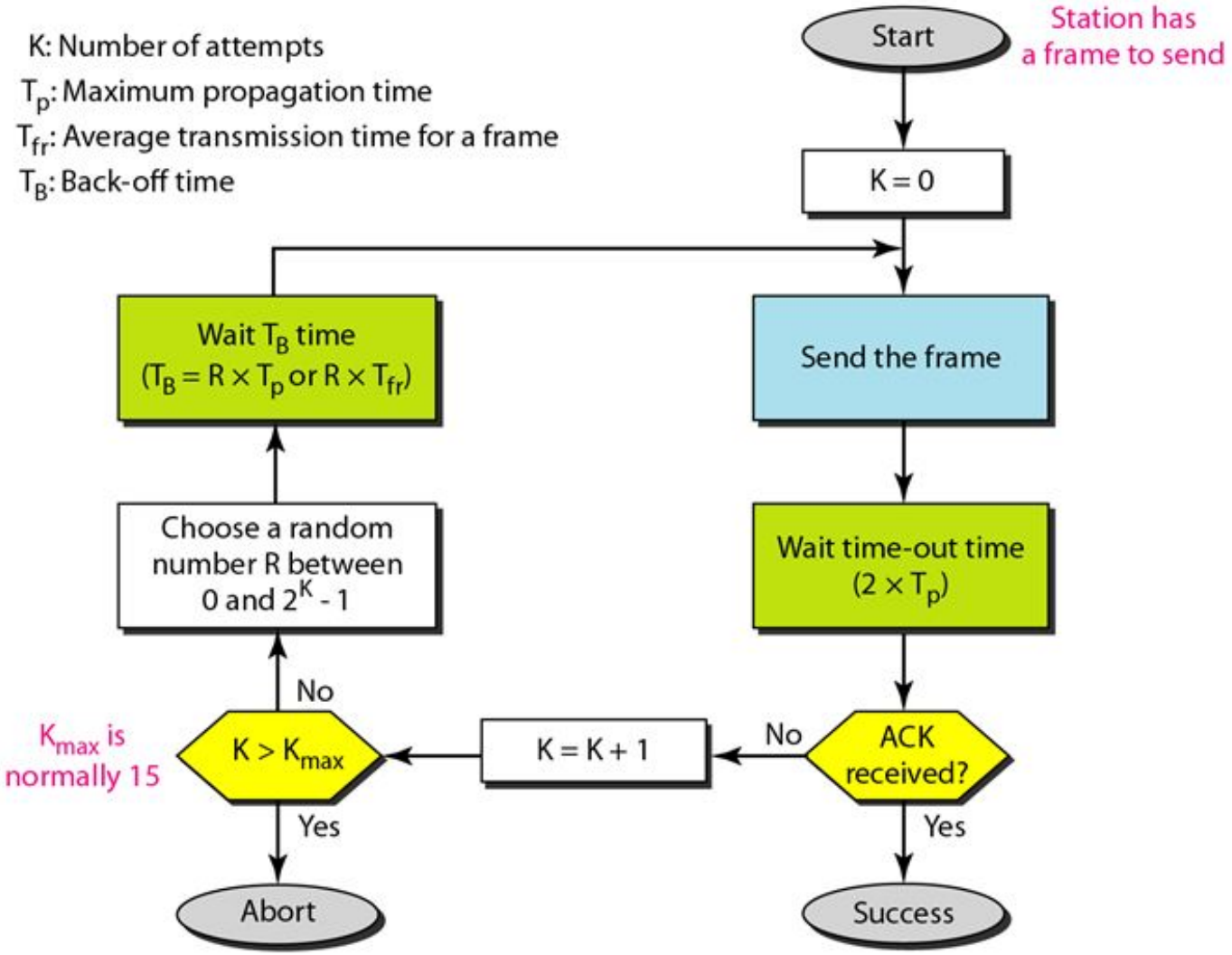
ALOHANet

To sum up pure Aloha protocol

- If you have data to send, send the data
- If, while you are transmitting data, you receive any data from another station, there has been a message collision. All transmitting stations will need to try resending later.

Agenda

Implementation



Implementation

```
class AlohaEventTypes(Enum):
    ACK = "ACK"
    DATA = "DATA"


class PacketUnit:
    K_MAX = 15

    def __init__(self, event: Event):
        self.event = event
        self.K = 0
        self.send_time = time.time() * 1000

    def get_back_off_time(self):
        R = randint(0, 2 ** self.K - 1)
        return T_P ** R

    def is_expired(self):
        return self.K > self.K_MAX

    def should_resend(self):
        return time.time() * 1000 - self.send_time > T_P
```

Implementation

```
class AlohaNode(GenericMac):
    def __init__(self, componentname, componentinstancenumber, context=None,
                 configurationparameters=None,
                 num_worker_threads=1, topology=None, sdr=None):
        super().__init__(componentname, componentinstancenumber, context, configurationparameters, num_worker_threads,
                         topology, sdr)

        self.not_acked_packets: [PacketUnit] = []

    def update_not_acked_packets(self):
        for packet in self.not_acked_packets:
            if packet.should_resend:
                packet.K = packet.K + 1
                packet.send_time = time.time() * 1000

    def on_ack(self, ack: Event):
        filter(lambda packet: not (packet.event.eventid == ack.eventcontent.payload or packet.is_expired()),
               self.not_acked_packets)
```

Implementation

```
def on_message_from_bottom(self, eventobj: Event):
    """ Message from link physical layer message goes into inbox """
    if eventobj.eventcontent.header.messagetype == AlohaEventTypes.ACK:
        self.on_ack(ack=eventobj)
    else:
        if not self.not_acked_packets:
            self.send_ack(eventobj)
            self.send_up(eventobj)

def send_ack(self, eventobj: Event):
    evt = Event(self, EventTypes.MFRT, eventobj.eventcontent)
    evt.eventcontent.header.messagetype = AlohaEventTypes.ACK
    evt.eventcontent.header.messageto = eventobj.eventcontent.header.messagefrom
    evt.eventcontent.header.messagefrom = self.componentinstancenumber
    evt.eventcontent.payload = eventobj.eventid
    self.send_down(evt)  # Send the ACK

def on_message_from_top(self, eventobj: Event):
    """ Message from link link layer message goes into outer """
    packet = PacketUnit(event=eventobj)
    self.not_acked_packets.append(packet)
    self.update_not_acked_packets()
    head = self.not_acked_packets[0]
    self.send_down(head.event)
```

Agenda

Conclusions

- Ad-Hoc Networks
- ALOHANet

Questions

THANK YOU

CENG797 Ad-Hoc Networks
ALOHANet Implementation on USRP-AHC

presented by Berkay Demirören