COMPUTER NETWORKS

SAKSHAM KUMAR

COMPUTER SCIENCE AND ENGINEERING
ID - 2022UCP1700
SECTION- A4

ASSIGNMENT – 7

Task 1:

Theoretical Understanding of CSMA/CA Protocol

1. Research and summarize the key principles of the CSMA/CA protocol.
2. Explain how CSMA/CA differs from other MAC protocols, such as CSMA/CD
(Carrier Sense Multiple Access with Collision Detection).
3. Discuss the advantages and disadvantages of CSMA/CA in wireless communication scenarios.

1. Key Principles of CSMA/CA Protocol:

CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) is a medium access control protocol used in wireless communication networks to manage access to the shared communication medium. Its key principles are:

a. **Carrier Sensing:** Nodes listen to the medium before transmitting to check for ongoing transmissions. If the medium is sensed as busy, the node defers its transmission to avoid collisions.

b. **Collision Avoidance:** Unlike CSMA/CD, which deals with collisions after they occur, CSMA/CA aims to prevent collisions by employing a mechanism called the "virtual carrier sensing." Nodes inform others about their intention to transmit by sending a Request-to-Send (RTS) packet. If the medium remains idle for a predetermined time after RTS, the node transmits a Clear-to-Send (CTS) packet. Other nodes receiving CTS update their network allocation vector (NAV), which represents the time duration during which they should refrain from transmitting to avoid collisions.

c. **Random Backoff:** In case multiple nodes attempt to transmit simultaneously or after the NAV period, nodes initiate a random backoff process to choose a random time slot to reattempt transmission. This random backoff helps in reducing collisions further.

2. CSMA/CA and CSMA/CD have different approaches to handling collisions:

- **Collision Detection (CD)**: In CSMA/CD, if two nodes transmit simultaneously and a collision occurs, both nodes detect the collision and stop transmitting. After a random backoff time, they reattempt transmission.

- **Collision Avoidance (CA)**: In CSMA/CA, nodes try to avoid collisions altogether by first requesting permission to transmit (RTS) and waiting for acknowledgment (CTS) before actual transmission. Additionally, CSMA/CA utilizes NAV to inform other nodes about ongoing transmissions, reducing the likelihood of collisions.

3. Advantages and Disadvantages of CSMA/CA:

Advantages:

- **Reduced Collisions**: CSMA/CA's collision avoidance mechanism minimizes the occurrence of collisions, enhancing overall network efficiency.

- **Suitable for Wireless Networks**: CSMA/CA is particularly suitable for wireless networks where collisions are more common due to factors like signal interference and hidden node problem.

- **Scalability**: It can scale well to accommodate a large number of nodes contending for the medium.

Disadvantages:

- **Increased Overhead**: The process of RTS/CTS exchange adds overhead to the network, reducing available bandwidth for actual data transmission.

- **Latency**: The additional steps involved in collision avoidance may introduce latency in data transmission, impacting real-time communication applications.

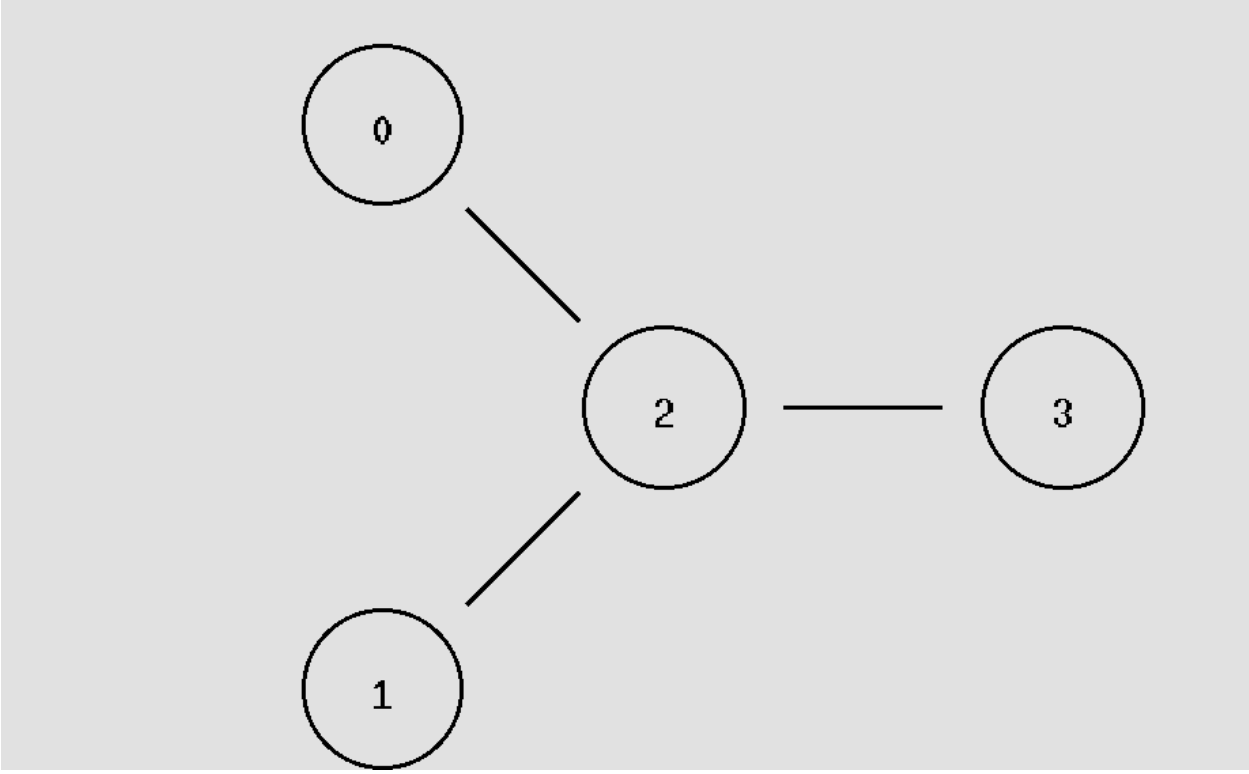
- **Complexity**: Implementing CSMA/CA requires more sophisticated hardware and software compared to simpler protocols, potentially increasing the cost and complexity of network devices.

Task 2: Simulation Setup

1. Open your NS2 environment and create a new simulation script file (e.g.,csma_ca.tcl).
2. Define simulation parameters such as network topology, number of nodes, and communication range.
3. Choose a wireless channel and set relevant parameters.
4. Configure the traffic model, specifying the type and quantity of traffic to be Generated.

```
set ns [new Simulator]
set nf [open csma_ca.nam w]
$ns namtrace-all $nf

proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam csma_ca.nam &
        exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 10Mb 10ms DropTail
$ns duplex-link $n1 $n2 10Mb 10ms DropTail
$ns duplex-link $n2 $n3 10Mb 10ms DropTail

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

$ns at 1.0 "finish"
```

$ns run




```
set val(chan)       Channel/WirelessChannel    ;
set val(prop)       Propagation/TwoRayGround   ;
set val(netif)      Phy/WirelessPhy            ;
set val(mac)        Mac/802_11                 ;
set val(ifq)        Queue/DropTail/PriQueue    ;
set val(ll)         LL                         ;
set val(ant)        Antenna/OmniAntenna        ;
set val(ifqlen)     50                         ;
set val(nn)         10                         ;
set val(stop)       10                         ;

set ns [new Simulator]
set tracefd [open csma_ca.tr w]
$ns trace-all $tracefd
```

```
$ns node-config -adhocRouting OFF       \
                -llType $val(ll)      \
                -macType $val(mac)    \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen)   \
                -antType $val(ant)       \
                -propType $val(prop)   \
                -phyType $val(netif)   \
                -channelType $val(chan) \
                -topoInstance $topo   \
                -agentTrace ON           \
                -routerTrace ON          \
                -macTrace ON             \
                -movementTrace ON

set udp [new Agent/UDP]
$ns attach-agent $n(0) $udp

set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 500
$cbr set interval_ 0.005
$cbr attach-agent $udp
$ns connect $udp $null
$ns at 2.0 "$n(0) setdest 300.0 300.0 50.0"

$ns run
```

Task 3: CSMA/CA Implementation

1.Write NS2 code to implement CSMA/CA protocol.
2.Define the backoff mechanism and other relevant parameters.
3. Ensure that the implementation considers the RTS/CTS (Request to Send/Clear to Send) mechanism in CSMA/CA.
4. Include mechanisms for acknowledging successful data transmission and handling collisions.

```
#initialize the variables
set val(chan)       Channel/WirelessChannel    ;   #Channel Type
set val(prop)       Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)      Phy/WirelessPhy            ;        # network interface type WAVELAN
DSSS 2.4GHz
set val(mac)        Mac/802_11                 ;        # MAC type
set val(ifq)        Queue/DropTail/PriQueue    ;     # interface queue type
set val(ll)         LL                         ;                # link layer type
set val(ant)        Antenna/OmniAntenna        ;     # antenna model
set val(ifqlen)     50                         ;                # max packet in ifq

set val(nn)         6                          ;                # number of mobilenodes
set val(rp)         AODV                       ;                # routing protocol
set val(x)          500                        ;                # in metres
set val(y)          500                        ;                # in metres

# CSMA/CA parameters
set cw_min          4                          ;     # minimum contention window size
set cw_max          1024                       ; # maximum contention window size
set retry_limit     7                          ;     # maximum number of                transmission
attempts

set ns [new Simulator]

#creation of Trace and namfile
set tracefile [open wireless2.tr w]
$ns trace-all $tracefile

#Creation of Network Animation file
set namfile [open wireless2.nam w]
$ns namtrace-all-wireless $namfile $val(x) $val(y)

#create topography
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
```

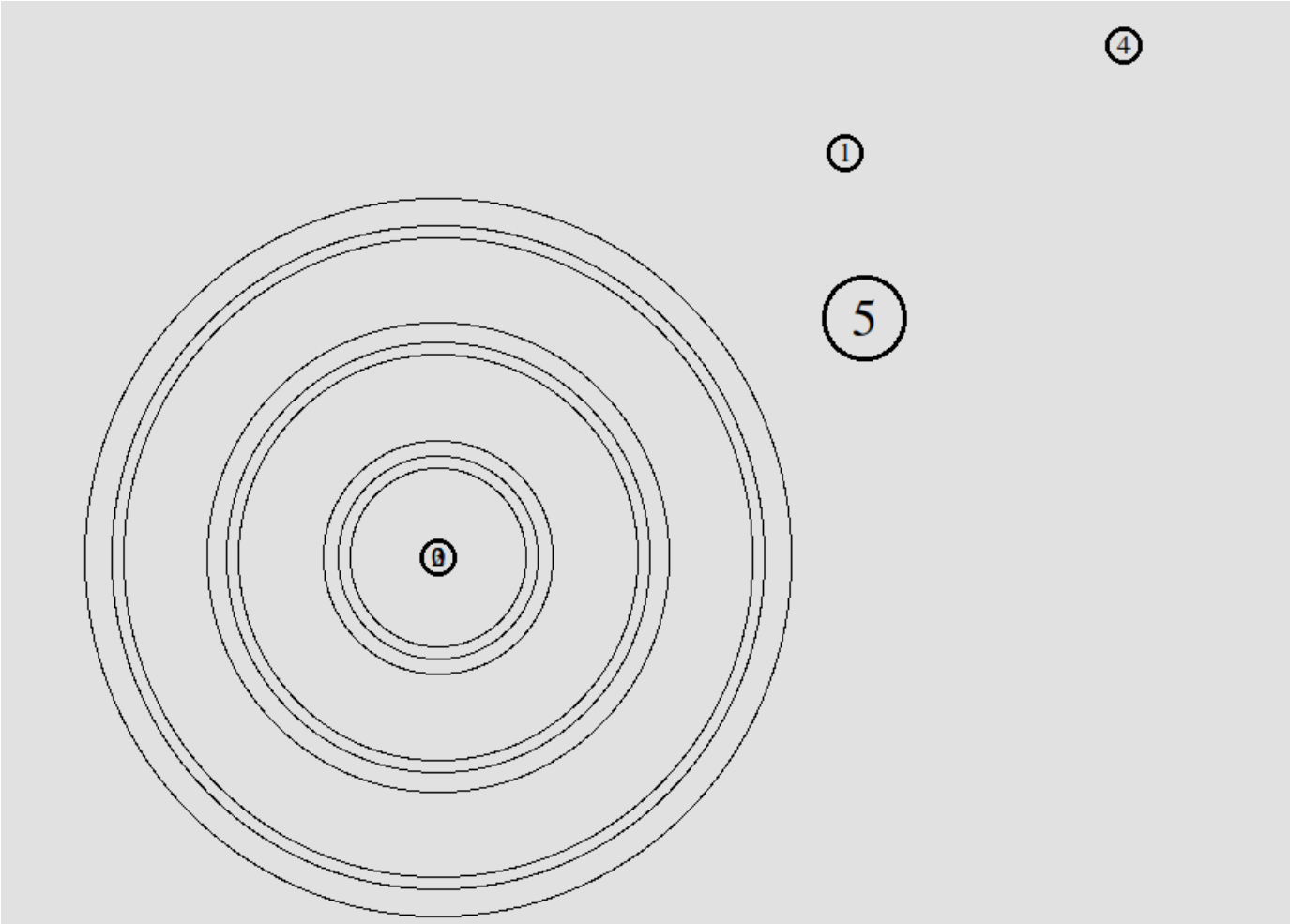
```
#GOD Creation - General Operations Director
create-god $val(nn)

set channel1 [new $val(chan)]
set channel2 [new $val(chan)]
set channel3 [new $val(chan)]

#configure the node
$ns node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
  -macType $val(mac) \
  -ifqType $val(ifq) \
  -ifqLen $val(ifqlen) \
  -antType $val(ant) \
  -propType $val(prop) \
  -phyType $val(netif) \
  -topoInstance $topo \
  -agentTrace ON \
  -macTrace ON \
  -routerTrace ON \
  -movementTrace ON \
  -channel $channel1

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$n0 random-motion 0
$n1 random-motion 0
$n2 random-motion 0
$n3 random-motion 0
$n4 random-motion 0
$n5 random-motion 0
```

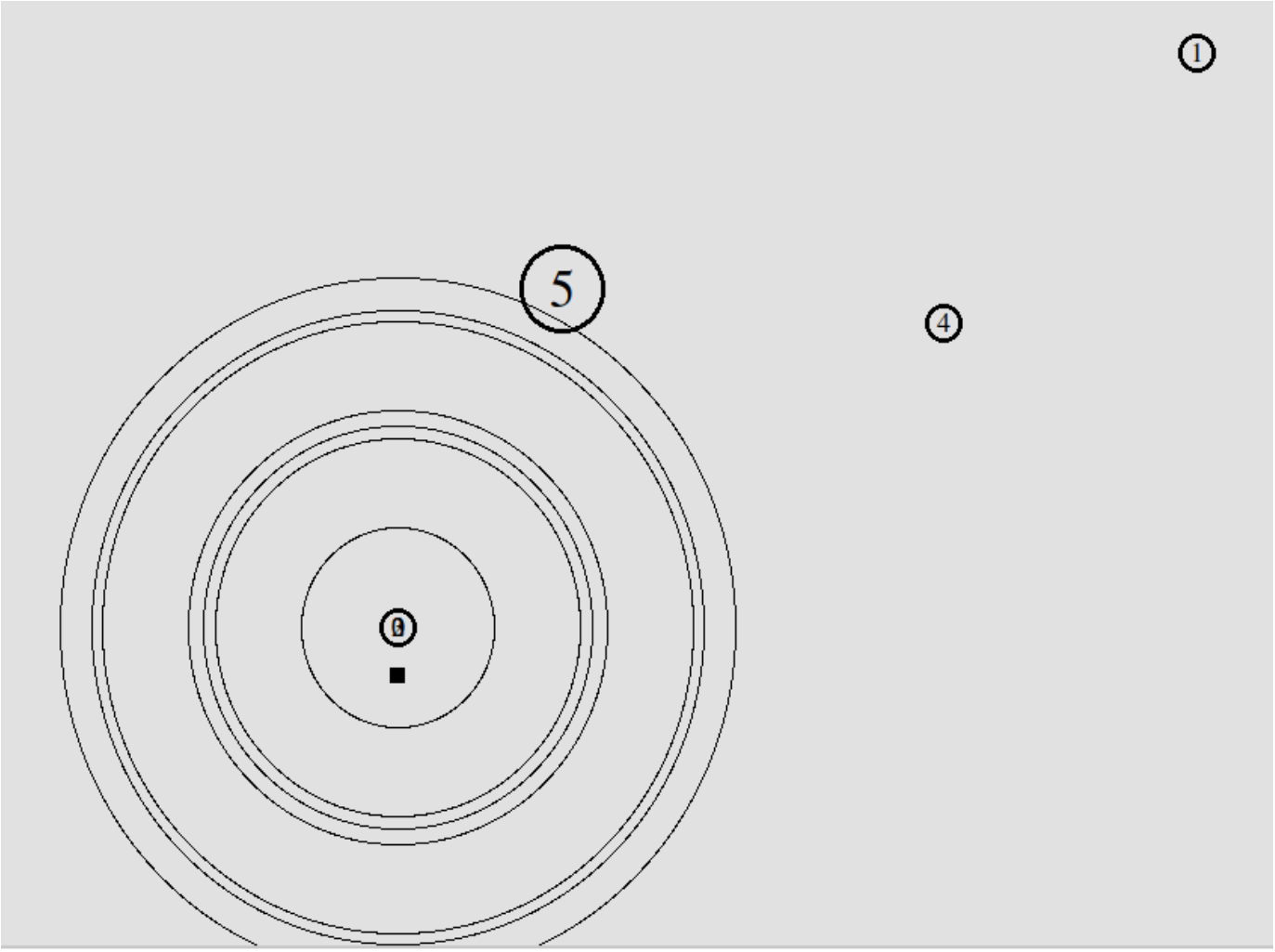
```
$ns initial_node_pos $n0 20
$ns initial_node_pos $n1 20
$ns initial_node_pos $n2 20
$ns initial_node_pos $n3 20
$ns initial_node_pos $n4 20
$ns initial_node_pos $n5 50

#initial coordinates of the nodes
$n0 set X_ 10.0
$n0 set Y_ 20.0
$n0 set Z_ 0.0

$n1 set X_ 210.0
$n1 set Y_ 230.0
$n1 set Z_ 0.0

$n2 set X_ 100.0
$n2 set Y_ 200.0
$n2 set Z_ 0.0

$n3 set X_ 150.0
$n3 set Y_ 230.0
$n3 set Z_ 0.0

$n4 set X_ 430.0
$n4 set Y_ 320.0
$n4 set Z_ 0.0

$n5 set X_ 270.0
$n5 set Y_ 120.0
$n5 set Z_ 0.0

#mobility of the nodes
#At what Time? Which node? Where to? at What Speed?
$ns at 1.0 "$n1 setdest 490.0 340.0 25.0"
$ns at 1.0 "$n4 setdest 300.0 130.0 5.0"
```

```
$ns at 1.0 "$n5 setdest 190.0 440.0 15.0"
#the nodes can move any number of times at any location during the simulation
(runtime)
$ns at 20.0 "$n5 setdest 100.0 200.0 30.0"

# CSMA/CA function to handle backoff
proc csma_ca_backoff {node} {
    set backoff [expr int(rand() * $cw_min)]
    $node set backoff $backoff
}

# Function to handle RTS/CTS
proc handle_rts_cts {sender receiver} {
    if {[rand] < 0.5} {
        # Channel is busy, defer transmission
        csma_ca_backoff $sender
    } else {
        # Channel is clear, send RTS
        $sender send_rts $receiver
        # Wait for CTS or defer if collision detected
        if {collision_detected} {
            csma_ca_backoff $sender
        } else {
            # Start data transmission
            $sender send_data
        }
    }
}

# Function to handle successful transmission
proc handle_successful_transmission {node} {
    # Send ACK
    $node send_ack
    # Update performance metrics
}

# Function to handle collision
```

```
# Function to handle collision
proc handle_collision {sender receiver} {
    # Implement collision handling
    # Adjust contention window
    set cw [$sender get_cw]
    set new_cw [expr {min($cw * 2, $cw_max)}] ;# Double contention window but limit to
max
    $sender set_cw $new_cw

    # Retry transmission
    if {[$sender get_attempts] < $retry_limit} {
        $sender retry_transmission
    } else {
        # Exceeded retry limit, give up
        $sender reset_attempts
        # Optionally: log or handle the fact that transmission failed
    }
}


# Traffic agents
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.0 "$ftp start"

set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n2 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
```

```
$ns at 1.0 "$cbr start"

# Run the simulation
puts "Starting Simulation"
$ns run

# Finish procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exit 0
}
```





Task 4: Simulation Execution and Analysis

1. Run the simulation using the script created in Task 2.
2. Monitor the simulation output for key performance metrics such as throughput, packet loss, and delay.
3. Analyze the impact of varying parameters, such as contention window size and data rates, on the overall performance of CSMA/CA.

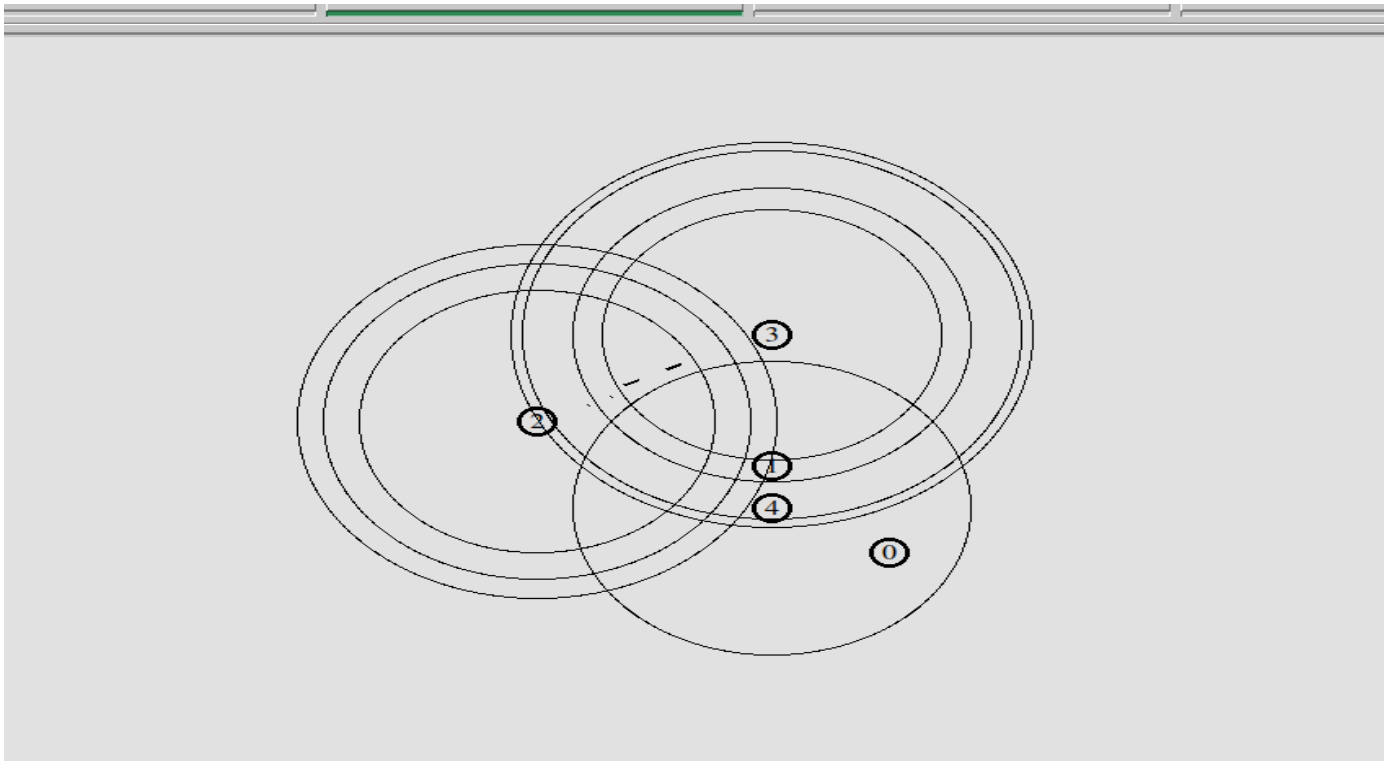
```
proc finish {} {
global ns tracefile namfile
$ns flush-trace
```

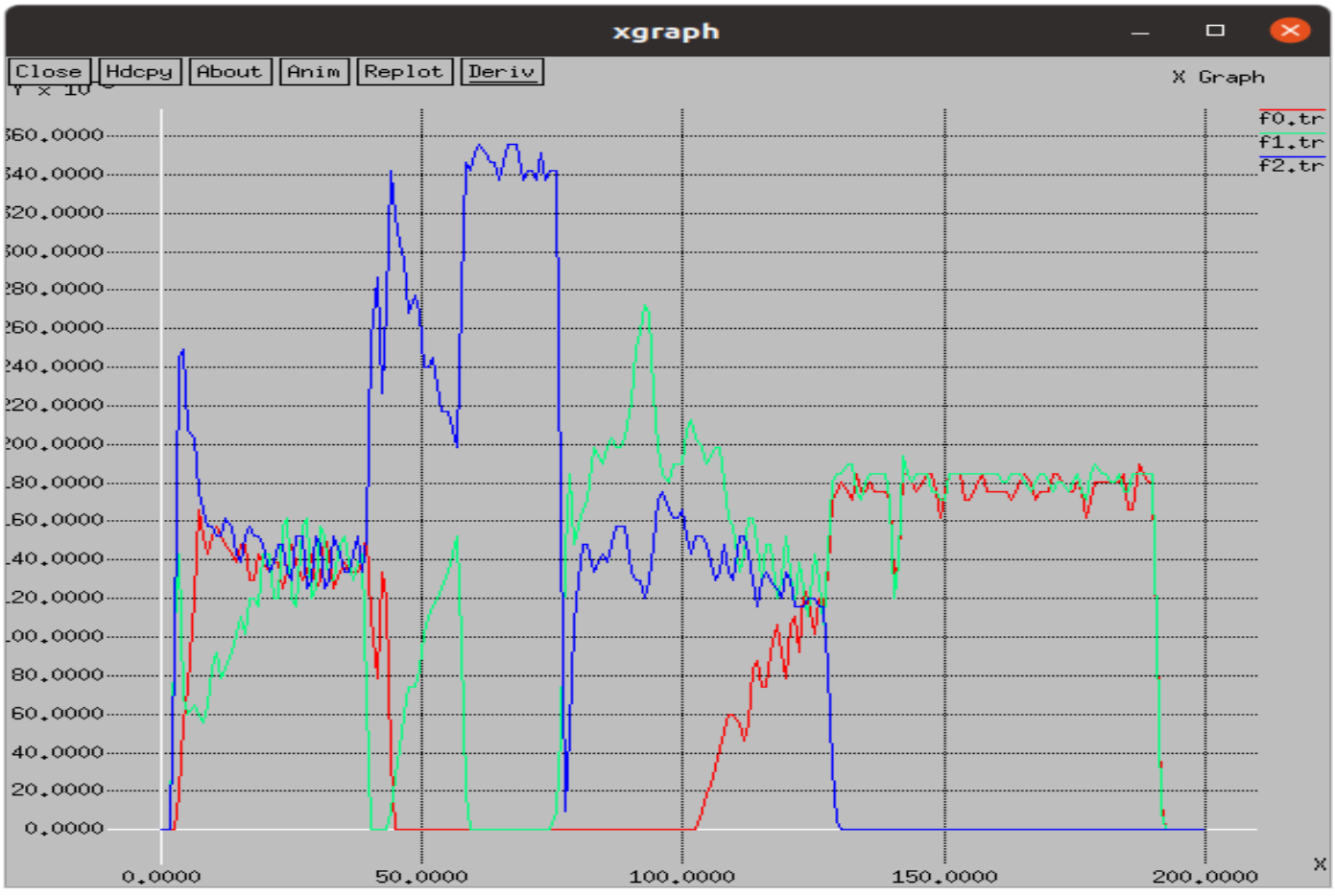
```
close $tracefile
close $namfile
exec xgraph f0.tr f1.tr f2.tr &
exec nam csma_ca.nam &
exit 0
}
set val(chan) Channel/WirelessChannel
set val(iqtype) Queue/DropTail/PriQueue


set val(mac) Mac/802_11
set val(netif) Phy/WirelessPhy
set val(X) 500
set val(Y) 500
set val(nn) 5
set val(ifqlen) 100


set val(rp) AODV
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(prop) Propagation/TwoRayGround

set ns [new Simulator]
set tracefile [open csma_ca.tr w]
$ns trace-all $tracefile
set namfile [open csma_ca.nam w]
$ns namtrace-all-wireless $namfile $val(X) $val(Y)
set topo [new Topography]
$topo load_flatgrid $val(X) $val(Y)
create-god $val(nn)
$ns node-config -adhocRouting $val(rp) -channelType $val(chan) -propType $val(prop)
-llType $val(ll) -macType $val(mac) -ifqLen $val(ifqlen) -antType $val(ant) -ifqType
$val(iqtype) -phyType $val(netif) -topoInstance $topo -agentTrace ON -routerTrace ON
-macTrace ON -movementTrace ON

set n0 [$ns node]
```

```
$n0 set X_ 400
$n0 set Y_ 500
set n1 [$ns node]
$n1 set X_ 300
$n1 set Y_ 600
set n2 [$ns node]
$n2 set X_ 100
$n2 set Y_ 650
set n3 [$ns node]
$n3 set X_ 300
$n3 set Y_ 750
set n4 [$ns node]
$n4 set X_ 300
$n4 set Y_ 550


$ns at [ expr 15+round(rand()*60) ] "$n0 setdest [ expr 10+round(rand()*480) ] [ expr
10+round(rand()*380) ] [ expr 2+round(rand()*15) ]"
$ns at [ expr 15+round(rand()*60) ] "$n1 setdest [ expr 10+round(rand()*480) ] [ expr
10+round(rand()*380) ] [ expr 2+round(rand()*15) ]"
$ns at [ expr 15+round(rand()*60) ] "$n2 setdest [ expr 10+round(rand()*480) ] [ expr
10+round(rand()*380) ] [ expr 2+round(rand()*15) ]"
$ns at [ expr 15+round(rand()*60) ] "$n3 setdest [ expr 10+round(rand()*480) ] [ expr
10+round(rand()*380) ] [ expr 2+round(rand()*15) ]"
$ns at [ expr 15+round(rand()*60) ] "$n4 setdest [ expr 10+round(rand()*480) ] [ expr
10+round(rand()*380) ] [ expr 2+round(rand()*15) ]"

$n0 random-motion 1
$n1 random-motion 1
$n2 random-motion 1
$n3 random-motion 1
$n4 random-motion 1

set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
set tcp1 [new Agent/TCP]
set sink1 [new Agent/TCPSink]
```

```
set tcp2 [new Agent/TCP]
set sink2 [new Agent/TCPSink]

set tcp3 [new Agent/TCP]
set sink3 [new Agent/TCPSink]

$ns attach-agent $n0 $tcp
$ns attach-agent $n3 $sink
$ns attach-agent $n1 $tcp1
$ns attach-agent $n3 $sink1
$ns attach-agent $n2 $tcp2
$ns attach-agent $n3 $sink2
$ns attach-agent $n4 $tcp3
$ns attach-agent $n3 $sink3
$ns connect $tcp $sink
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
$ns connect $tcp3 $sink3
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
set f0 [open f0.tr w]
set f1 [open f1.tr w]
set f2 [open f2.tr w]

set hr1 0
set hr2 0
set hr3 0

proc record {} {
    global sink sink1 sink2 f0 f1 f2 hr1 hr2 hr3
    set ns [Simulator instance]
```

```
    set time 0.9
    set bw0 [$sink set bytes_]
    set bw1 [$sink1 set bytes_]
    set bw2 [$sink2 set bytes_]
    set now [$ns now]
    # Record the Bit Rate in Trace Files
    puts $f0 "$now [expr (($bw0+$hr1)*8)/(2*$time*1000000)]"
    puts $f1 "$now [expr (($bw1+$hr2)*8)/(2*$time*1000000)]"
    puts $f2 "$now [expr (($bw2+$hr3)*8)/(2*$time*1000000)]"
    $sink set bytes_ 0
    $sink1 set bytes_ 0
    $sink2 set bytes_ 0
    set hr1 $bw0
    set hr2 $bw1
    set hr3 $bw2
    $ns at [expr $now+$time] "record"
}

$ns at 0 "record"
$ns at 2.0 "$ftp start"
$ns at 2.0 "$ftp1 start"
$ns at 2.0 "$ftp2 start"
$ns at 2.0 "$ftp3 start"
$ns initial_node_pos $n0 30
$ns initial_node_pos $n1 30
$ns initial_node_pos $n2 30
$ns initial_node_pos $n3 30
$ns initial_node_pos $n4 30
$ns at 190.0 "$ftp stop"
$ns at 190.0 "$ftp1 stop"
$ns at 190.0 "$ftp2 stop"
$ns at 190.0 "$ftp3 stop"
$ns at 200.0 "finish"
$ns run
```


3
2
1
4
0


xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
X Graph
f0.tr
f1.tr
f2.tr
360.0000
340.0000
320.0000
300.0000
280.0000
260.0000
240.0000
220.0000
200.0000
180.0000
160.0000
140.0000
120.0000
100.0000
80.0000
60.0000
40.0000
20.0000
0.0000
0.0000
50.0000
100.0000
150.0000
200.0000
X

Task 5: Results Presentation and Report

CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) is a protocol used in wireless networks to regulate access to the shared transmission medium to avoid collisions. Optimizing its performance depends on the specific scenario and requirements. Here are some recommendations for optimizing CSMA/CA performance in different scenarios:

1. **Adjust Contention Window (CW) Parameters**: CW determines the range of backoff values before reattempting transmission. In scenarios with high contention, increasing CW can reduce collisions but may also increase latency. Conversely, decreasing CW reduces latency but may lead to more collisions. Fine-tune CW parameters based on network load and latency requirements.
2. **Utilize RTS/CTS Mechanism**: Request-to-Send (RTS) and Clear-to-Send (CTS) frames help avoid collisions by reserving the channel for the duration of the data exchange. In scenarios with large data packets or high interference, enabling RTS/CTS can improve performance by reducing collisions.
3. **Optimize Interframe Spacing (IFS)**: IFS determines the time gap between frames. Shorter IFS values allow faster access to the medium but increase the likelihood of collisions. In scenarios with high traffic, reducing IFS can improve performance, but this may also lead to more collisions.
4. **Dynamic Adjustment of Parameters**: Implement algorithms that dynamically adjust CW, IFS, and other parameters based on network conditions. Adaptive algorithms can optimize performance in varying traffic conditions and adapt to changes in interference or network load.
5. **Prioritize Traffic**: Differentiate between traffic types based on priority levels and allocate resources accordingly. For example, prioritize real-time traffic such as voice or video over non-real-time traffic like file transfers.
6. **Channel Quality Awareness**: Implement mechanisms to detect channel quality and adjust CSMA/CA parameters accordingly. This can involve techniques such as Clear Channel Assessment (CCA) to sense channel occupancy or dynamic modulation and coding schemes based on channel conditions.
7. **Power Management**: Optimize power management strategies to reduce contention and interference. For example, devices can coordinate their wake-up times to minimize collisions during contention periods.
8. **Traffic Engineering**: Design the network layout and topology to minimize collisions and contention. This may involve deploying multiple access points

strategically, using directional antennas to focus transmissions, or segmenting the network to reduce interference.

9. **Avoid Hidden Node Problem**: Address the hidden node problem by employing mechanisms such as virtual carrier sensing or using directional antennas to improve spatial reuse and minimize collisions caused by hidden nodes.
10. **Performance Monitoring and Optimization**: Continuously monitor network performance metrics such as throughput, latency, and packet loss rates. Use this data to identify bottlenecks and fine-tune CSMA/CA parameters for optimal performance.

By carefully considering these recommendations and tailoring them to specific scenarios, you can effectively optimize CSMA/CA performance in wireless networks.

THE END