```
# Utama.tcl

Tracefile set debug_ 0    ;# to avoid some dummy warning
TBF set debug_ 0          ;# to avoid some dummy warning

# ======================================================================
# System parameters
# ======================================================================
set opt(nam)          1            ;# output ns2 nam wireless trace
set opt(trc)          1            ;# output ns2 wireless trace
set opt(alive_time)   10.0         ;# ping ns2 each 10 seconds
set opt(out)          /dev/null    ;# statistics output filename
set opt(run)          1            ;# run identifier
set opt(warm)         20.0         ;# warm-up period duration
set opt(endtime)      100.0        ;# simulation end time
set opt(basetrace)    "output"     ;# base trace filename
set opt(seed)         1            ;# RNG seed

# ======================================================================
# Physical parameters & trace features
# ======================================================================
set val(chan)      Channel/WirelessChannel   ;# channel type
set val(prop)      Propagation/TwoRayGround  ;# radio-propagation model
set val(netif)     Phy/WirelessPhy           ;# network interface type
set val(mac)       Mac/802_11/802_11e        ;# 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(rp)            NOAH                  ;# routing protocol
set val(mactrace)      ON                    ;# MAC trace ON/OFF
set val(agenttrace)    ON                    ;# Agent trace ON/OFF
set val(movementtrace) ON                    ;# Movement trace ON/OFF
set val(routertrace)   ON                    ;# Router trace ON/OFF
set val(verbosity)     0                     ;# verbosity level
set val(n)             2                     ;# number of wireless nodes
set val(lambda)        100                   ;# area length
set val(radius)        20                    ;# QAP distance
set val(phy)           "dsss802.11b"         ;# PHY layer
set val(cwmin)         31                    ;# CWMin
set val(cwmax)         1023                  ;# CWMax
set val(rtsthresh)     4096                  ;# RTS threshold
set val(srl)           7                     ;# short retry limit
set val(lrl)           4                     ;# long retry limit
set val(max_sdu_size)  2132                  ;# max SDU size
set val(schedmap)      "reference"
set val(hcca_scheduler_qap)  "periodic"      ;# HCCA QAP scheduler
set val(hcca_scheduler_qsta) "oneflow"       ;# HCCA QSTA scheduler
set val(oneflow_qlen)        "infinite"      ;# HCCA queue length at QSTAs
set val(uniform_error_rate)  0               ;# packet error rate
set val(qack)                0               ;# qack bit
set val(piggyback)           0               ;# 1 -> piggybacking active
set val(ref_vbr_MaxMSDU)     "off"   ;# use the Maximum MSDU Size
                          ;# instead of the Nominal MSDU Size for computing
                          ;# the TXOP length with 'reference' if 'on'
# ======================================================================
```

(Lanjutan)

```
# Default Physical & MAC parameters (IEEE802.11b DSSS PHY)
Simulator set EotTrace_ OFF

Phy/WirelessPhy set CPThresh_       1000000.0     ;# capture threshold
                                    ;# no capture (default = 10.0)

proc finish {} {
   global ns opt      ;# input

   # flush ns traces
      if { $opt(nam) || $opt(trc) } {
            $ns flush-trace
      }

   # Close the trace file(s)
      if { $opt(nam) } { close $opt(namtr) }
      close $opt(trace)

      $ns stat print

   mylog "simulation ended"
   exit 0
}

proc open_traces {} {
   global opt val defaultRNG    ;# input
   global ns                    ;# output

      if { $opt(nam) } { set opt(namtr) [open "$opt(basetrace).nam" w] }
      set opt(trace) [open "$opt(basetrace).trc" w]
   mylog "trace descr: $opt(trace)"

   set ns [new Simulator]
      $defaultRNG seed 1

   # initialize statistics collection
      $ns run-identifier $opt(run)
      $ns stat file "$opt(out)"
      $ns at $opt(warm) "$ns stat on"
      $ns at $opt(endtime) "finish"

   $ns use-newtrace
   if { $opt(nam) } {
            $ns namtrace-all-wireless $opt(namtr) $val(lambda)
$val(lambda) }
      $ns trace-all $opt(trace)
}

proc create_topology {} {
   global ns opt val difs     ;# input
   global topo qsta qap       ;# output

# ============================================================================
# set PHY parameters
# ============================================================================
      if { $val(phy) == "fhss802.11" } {
            Mac/802_11 set SlotTime_          0.000050    ;# 50us
```

(Lanjutan)

```
            Mac/802_11 set SIFS_                0.000028    ;# 28us
            Mac/802_11 set PreambleLength_      0           ;# no preamble
            Mac/802_11 set PLCPHeaderLength_    128         ;# 128 bits
            Mac/802_11 set PLCPDataRate_        1.0e6       ;# 1Mbps
            Mac/802_11 set dataRate_            1.0e6       ;# 11Mbps
            Mac/802_11 set basicRate_           1.0e6       ;# 1Mbps
      } elseif { $val(phy) == "dsss802.11b" } {
            Mac/802_11 set SlotTime_            0.000020    ;# 20us
            Mac/802_11 set SIFS_                0.000010    ;# 10us
            Mac/802_11 set PreambleLength_      144         ;# 144 bit
            Mac/802_11 set PLCPHeaderLength_    48          ;# 48 bits
            Mac/802_11 set PLCPDataRate_        1.0e6       ;# 1Mbps
            Mac/802_11 set dataRate_            11.0e6      ;# 11Mbps
            Mac/802_11 set basicRate_           1.0e6       ;# 1Mbps
      } elseif { $val(phy) == "dsss802.11" } {
            Mac/802_11 set SlotTime_            0.000020    ;# 20us
            Mac/802_11 set SIFS_                0.000010    ;# 10us
            Mac/802_11 set PreambleLength_      144         ;# 144 bit
            Mac/802_11 set PLCPHeaderLength_    48          ;# 48 bits
            Mac/802_11 set PLCPDataRate_        1.0e6       ;# 1Mbps
            Mac/802_11 set dataRate_            2.0e6       ;# 2Mbps
            Mac/802_11 set basicRate_           1.0e6       ;# 1Mbps
      } else {
            puts "not using a well-known PHY parameters set"
      }
            Mac/802_11 set CWMin_               $val(cwmin)
            Mac/802_11 set CWMax_               $val(cwmax)
            Mac/802_11 set RTSThreshold_        $val(rtsthresh)
            Mac/802_11 set ShortRetryLimit_     $val(srl)
            Mac/802_11 set LongRetryLimit_      $val(lrl)
            Mac/802_11 set MaxSDUSize_          $val(max_sdu_size)

# ==================================================================
# Create topography
# ==================================================================
set topo [new Topography]
$topo load_flatgrid $val(lambda) $val(lambda)
mylog "topography created"

# ==================================================================
# Create General Operations Director
# ==================================================================
create-god [expr $val(n) + 1]
mylog "god created"

# ==================================================================
# Create channel
# ==================================================================
set chan_1 [new $val(chan)]
mylog "wireless channel created"

# ==================================================================
# Color
# ==================================================================
$ns color 0 red
$ns color 1 blue
```

(Lanjutan)

```
$ns color 2 yellow
$ns color 3 green

# ==================================================================
# Create Access Point
# ==================================================================
$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 $val(agenttrace) \
                 -routerTrace $val(routertrace) \
                 -macTrace $val(mactrace) \
                 -movementTrace $val(movementtrace) \
                 -channel $chan_1

set qap [$ns node]
$qap random-motion 0
# AP in the center of a circle of radius $val(radius)
$qap set X_ [expr $val(lambda) / 2.0]
$qap set Y_ [expr $val(lambda) / 2.0]
$qap set Z_ 0
$ns initial_node_pos $qap 5

# set verbosity level
[$qap getMac 0] verbosity $val(verbosity)
if { $opt(hcca) == 0 } {
      mylog "AP node 0 created"
} else {
      mylog "QAP node 0 created"
}

# ==================================================================
# creating mobile nodes
# ==================================================================
for {set i 0} {$i < $val(n)} {incr i} {
   set qsta($i) [$ns node]
   $qsta($i) random-motion 0    ;# disable random motion
   set angle [expr (2 * acos(-1.0) / $val(n)) * $i]
   $qsta($i) set X_ [expr $val(lambda) / 2.0 + $val(radius) * sin($angle)]
   $qsta($i) set Y_ [expr $val(lambda) / 2.0 - $val(radius) * cos($angle)]
   $qsta($i) set Z_ 0
   $ns initial_node_pos $qsta($i) 5
   if { $opt(hcca) == 0 } {
         mylog "STA node [expr $i+1] created"
   }
}

# ==================================================================
# make $qap the QAP for QSTAs in the BSS
# ==================================================================
set qap_mac [$qap getMac 0]
```

(Lanjutan)

```
set qap_macaddr [$qap_mac id]
$qap_mac bss_id $qap_macaddr
if { $val(hcca_scheduler_qap) == "periodic" } {
      if { $val(schedmap) == "reference" } {
            $qap_mac scheduler periodic $val(schedmap)
            $qap_mac scheduler set_opt ref_vbr_MaxMSDU
            $val(ref_vbr_MaxMSDU)
      }
} else {
      $qap_mac scheduler $val(hcca_scheduler_qap)
}

$qap_mac uniform_error_rate $val(uniform_error_rate)
$qap_mac piggyback $val(piggyback)
for {set i 0} {$i < $val(n)} {incr i} {
       set qsta_mac [$qsta($i) getMac 0]
       $qsta_mac qack $val(qack)
       $qsta_mac piggyback $val(piggyback)
       $qap_mac qack $val(qack) $i
       $qsta_mac verbosity $val(verbosity)
       $qsta_mac bss_id $qap_macaddr
       $qsta_mac scheduler $val(hcca_scheduler_qsta)
       $qsta_mac uniform_error_rate $val(uniform_error_rate)
       if { $val(hcca_scheduler_qsta) == "oneflow" } {
             $qsta_mac oneflow $val(oneflow_qlen)
       }
   }
}

proc macDebug {} {
      global ns qap qsta val ;       # input

      [$qap getMac 0] debugvars
      for {set i 0} {$i < $val(n)} {incr i} {
            [$qsta($i) getMac 0] debugvars
      }
}

proc status {} {
      global qap opt

      [$qap getMac 0] scheduler status
}

proc init {} {
      global ns opt val argc argv    ;# input
      global defaultRNG              ;# input

      $defaultRNG seed $opt(seed)

   open_traces
   create_topology
      do_routing_stuff
   create_connections

      $ns at $opt(alive_time) "alive"
   $ns at $opt(endtime) "finish"
```

(Lanjutan)

```
        mylog "simulation started"
}

proc mylog { s } {
        set now [ exec date ]
        puts stdout "$now $s"
}

proc alive {} {
   global ns opt

   set now_r [exec date]
   set now_s [$ns now]
   puts "$now_r $now_s"
   $ns at [expr $now_s + $opt(alive_time)] "alive"
}

proc hcca-start {} {
   global ns qap qsta val opt     ;# input

   [$qap getMac 0] start-hcca
   for {set i 0} {$i < $val(n)} {incr i} {
      [$qsta($i) getMac 0] start-hcca
   }
}

proc hcca-stop {} {
   global ns qap qsta val opt     ;# input

   [$qap getMac 0] stop-hcca
   for {set i 0} {$i < $val(n)} {incr i} {
      [$qsta($i) getMac 0] stop-hcca
   }
}

proc do_routing_stuff {} {
   global ns opt val qsta qap      ;# input

   for {set i 0} {$i < $val(n)} {incr i} {

      for {set k 0} {$k < 2} {incr k} {
         set j [expr $k * $val(n) + $i]

         set application($j) [new Application/Traffic/CBR]
         set agtsrc($j)      [new Agent/UDP]
         set agtsink($j)     [new Agent/UDP]

         $application($j)  set random_     0
         $application($j)  set packetSize_ 1
         $application($j)  set rate_       1
         $application($j)  set maxpkts_    1

         $agtsrc($j)       set packetSize_ 1
         $agtsrc($j)       set prio_       0
         $agtsrc($j)       set class_      0
      }
```

(Lanjutan)

```
        ;# downlink
        $ns attach-agent $qsta($i) $agtsrc($i)
        $ns attach-agent $qap $agtsink($i)
        $ns connect      $agtsrc($i) $agtsink($i)
        $application($i) attach-agent $agtsrc($i)
        $ns at 0 "$application($i) start"

        ;# uplink
        set j [expr $val(n) + $i]
        $ns attach-agent $qap $agtsrc($j)
        $ns attach-agent $qsta($i) $agtsink($j)
        $ns connect $agtsrc($j) $agtsink($j)
        $application($j) attach-agent $agtsrc($j)

        $ns at "$application($j) start"
    }
}

proc getopt {argc argv} {
    global opt

    for {set i 0} {$i < $argc} {incr i} {
        set arg [lindex $argv $i]
        if {[string range $arg 0 0] != "-"} continue

              set name [string range $arg 1 end]
              set opt($name) [lindex $argv [expr $i+1]]
    }
}
```

```
# Skenario1_audio.tcl

# ======================================================================
# load standard scenario running script
# ======================================================================
source Utama.tcl
set val(agenttrace)     ON

# default argument options
set opt(start)        20.0          ;# traffic start time
set opt(n)            3             ;# number of real-time stations
set opt(hcca)         1             ;# HCCA is on

getopt $argc $argv

set val(n)        [expr 1 + $opt(n)]

# ======================================================================
# create connections
# ======================================================================
proc create_connections {} {
   global ns opt val qsta qap defaultRNG    ;# input

# ======================================================================
# real-time stations
# ======================================================================
for { set i 0 } { $i < $val(n) } { incr i } {
      for { set j 0 } { $j < 2 } { incr j } {
         set k [expr 2 * $i + $j]

         set agt_dst($k)      [new Agent/Null]
         set agt_src($k)      [new Agent/UDP]
         $agt_src($k) set packetSize_ 160
         $agt_src($k) set prio_       0
         $agt_src($k) set fid_        [expr $k + 1]

         set app($k)          [new Application/Traffic/CBR]
         $app($k) set packetSize_ 160
         $app($k) set random_     0
         $app($k) set rate_       64000

         if { $j == 0 } {  ;# downlink
            $ns attach-agent $qap  $agt_src($k)
            $ns attach-agent $qsta($i) $agt_dst($k)

            if { $opt(hcca) == 1 } {
                  [$qap getMac 0] tclas [expr $k + 1] 1 [expr $i + 1]
                  [$qap getMac 0] tspec 1 [expr $i + 1] \
                        downlink 160 160 0 0 .020 0 64000 0 0 0 0
            }
         } else {          ;# uplink
            $ns attach-agent $qsta($i) $agt_src($k)
            $ns attach-agent $qap  $agt_dst($k)

            if { $opt(hcca) == 1 } {
                  [$qsta($i) getMac 0] tclas [expr $k + 1] 1
```

(Lanjutan)

```
                    [$qsta($i) getMac 0] tspec 1 [expr $i + 1] \
                          uplink 0 0 0 0 0 0 0 0 .040 0 1
                    [$qap getMac 0] tspec 1 [expr $i + 1] \
                          uplink 160 160 0 0 .040 0 64000 0 0 0 0
               }
          }

          $ns connect $agt_src($k) $agt_dst($k)
          $app($k) attach-agent $agt_src($k)
                    $ns at $opt(start) "$app($k) start"
       }
    }
}

# ======================================================================
# run simulation
# ======================================================================
init
if { $opt(hcca) } {
       $ns at 0.0 status
       $ns at [expr $opt(start) - 0.0001] "hcca-start"
}
$ns run
```