```
function[NewEvents] = action3(event, log_file)

%suppose a node has these status:
%    0 - idle
%    1 - idle & just sent an RTS, waiting for CTS (for a potential transmitter)
%    2 - idle & just replied a CTS after receiving a RTS(for a potential receiver)
%    10 - primary sending
%    11 - primary sending and just replied a CTS3(OPPPCTS)
%    12 - primary sending and just replied a CTS(OPCTS)
%    20 - primary receiving
%    21 - primary receiving & just sent an RTSii(to another node)
%    22 - primary receiving & just sent an RTS(to its opposite node)
%    30 - secondary sending;
%    40 - secondary receiving;
%    50 - S-linking
%
%
%



%node(i,1) = x coordinate
%node(i,2) = y coordinate
%node(i,3) = power
%node(i,4) = state
%node(i,5) = k, where k is the node sending to this node.
%event =
%{
%    ifdelete = {0,1};
%    instant = t;
%    node = i;
%    pkt.rv = j;
%    pkt.type = {unknown,rts,rtsii,cts,cts1,cts2,cts3,ctsn1,ctsn2,ctsn3,data,ack};
%    pkt.id = new_id(i,j);
%    pkt.nav
%    type = {.....};
%
%}



%event.pkt.type: unknown, rts, rtsii, jamming, cts, .....data, ack

global Event_list;
NewEvents = [];

if event.ifdelete == 1
    return;
end

switch event.type
    case 'send_mac'
        t = event.instant;
        i = event.node;
        j = event.pkt.rv;
        event.pkt.id = new_id(i,j); % from i to j
        event.pkt.type = 'unknown'; %????????????????????
        event.pkt.nav = SIFS + cts_tx_time + SIFS + tx_time(event.pkt) + SIFS +
ack_tx_time;%?????????????????????????????nav???????????
        if ~isempty(mac_queue(i,j).list)|mac_status(i) %mac queue is not empty or
one pkt is being transmitted.
            mac_queue(i,j).list = [mac_queue(i,j).list; event];
        else %mac queue is already empty
            mac_status(i) = 1;
            newevent = event;
            newevent.instant = t;
            newevent.type = 'wait_for_channel';
```

```
                newevent.node = i;
                NewEvents = [NewEvents; newevent];
                clear newevent;
            end
        %send mac is done: generate a new packet and put it into mac_queue or prepare to send it by "wait_for_channel".

        case 'wait_for_channel'
            t = event.instant;
            i = event.node;
            j = event.pkt.rv; %get the information of the packet that will be delivered.
            if node(i,4) == 0 & ifnav(i) == 0; %this node is idle & the sender is not keeping an NAV
            %backoff
                if backoff_counter(i) > 0 %resume the backoff
                    newevent = event;
                    newevent.instant = t + slot_time;
                    newevent.type = 'backoff';
                    newevent.node = i;
                    NewEvents = [NewEvents; newevent];
                    clear newevent;
                else
                    newevent = event;
                    newevent.instant = t + DIFS;
                    newevent.type = 'backoff_start';
                    newevent.node = i;
                    NewEvents = [NewEvents; newevent];
                    clear newevent;
                end
            elseif node(i,4) == 20 & ifnav(i) == 0; %.....the sender is not keeping an NAV, this node is receiving
                newevent = event;
                newevent.instant = t;
                newevent.type = 'send_phy'%????????????????????????????????????????????send directly
                if node(i,5) == j
                    newevent.pkt.type = 'rts';
                else
                    newevent.pkt.type = 'rtsii';
                newevent.node = i;
                NewEvents = [NewEvents; newevent];
                clear newevent;
            elseif ifnav(i) == 1
                newevent = event;
                newevent.instant = max(nav(i,:).end);%????????????????????????????????need modifying
                newevent.type = 'wait_for_channel';
                newevent.node = i;
                %newevent.navnode = ??????????????????
                NewEvents = [NewEvents; newevent];
                clear newevent;
            end
        %wait_for_channel is done: if the node is idle and not keep an NAV, then backoff; if the node is receiving and not keeping an NAV, then 'send_phy' %immediately; if the node is keeping an NAV, then continue waiting until NAV is expired, or until an ACK invalidated the NAV.

        case 'send_jamming'
            t = event.instant;
            i = event.node;
            if node(i,5) ~= 0 %if some node is sending to this node
                for k = 1:n
                    if k == i | k == node(i,5) %except for this node and the sending node
                        continue;
                    else
                        if (node(k,4) == 1 | node(k,4) == 2 | node(k,4) == 21 | node(k,4)
```

```
== 22) & sig_detect(k,i)
                    newevent = event;
                    newevent.instant = t + signature_time*3;
                    newevent.node = k;
                    newevent.type = 'abort_transmission';
                    NewEvents = [NewEvents, newevent];
                    clear newevent;
                else (node(k,4) == 20 | node(k,4) == 21 | node(k,4) == 22 |
node(k,4) == 30 | node(k,4) == 40 | node(k,5) == 50) & sig_detect(k,i)
                    delay_recv(k, 7.5);
                end
            end
        end
    end
    %send_jamming is done: for each node hearing the jamming signal, if it is about to
send, it will "abort_transmission". If it is receiving, it will delay the reception.

    case 'abort_transmission'
        t = event.instant;
        i = event.node;
        j = event.pkt.rv;
        %search the transmission events and delete them from the Event_list, i.e.,
send_phy_finish, and change the node status.
        for i = 1:length(Event_list)
        if node == Event_list(i).node & strcmp(event_type,
Event_list(i).type)%.....
        %here: node of different status should delete different event_type. Need to
modify it.
            Event_list(i).ifdelete = 1;
        end
        if node(i,4) == 1 | node(i,4) == 2
            node(i,4) = 0;
        elseif node(i,4) == 21 | node(i,4) == 22
            node(i,4) = 20;
        end

    case 'backoff_start' %after DIFS
        t = event.instant;
        i = event.node;
        j = event.pkt.rv;
        if node(i,4) == 0 & ifnav(i) == 0
            backoff_attempt(i) = 0;
            temp = min(backoff_attempt(i) + CW_min, CW_max);
            backoff_counter(i) = floor((2^temp - 1)*rand);
            newevent = event;
            newevent.instant = t + slot_time;
            newevent.type = 'backoff';
            newevent.node = i;
            NewEvents = [NewEvents; newevent];
            clear newevent;
        else
            newevent = event;
            newevent.instant = t;
            newevent.type = 'wait_for_channel';
            newevent.node = i;
            NewEvents = [NewEvents; newevent];
            clear newevent;
        end
    %backoff_start is done: if node is idle and not keeping NAV, then start...; else wait
for channel again.

    case 'backoff'
        t = event.instant;
        i = event.node;
        j = event.pkt.rv;
        if node(i,4) == 0 & ifnav(i) == 0
```

```
            if backoff_counter(i) > 1;
                backoff_counter(i) = backoff_counter(i) - 1;
                newevent = event;
                newevent.instant = t + slot_time;
                newevent.type = 'backoff';
                newevent.node = i;
                NewEvents = [NewEvents; newevent];
                clear newevent;
            else %ready to send the packet
                backoff_counter(i) = 0; %reset counter for next use
**********?
                newevent = event;
                newevent.instant = t;
                newevent.type = 'send_phy';
                newevent.pkt.type = 'rts';
                newevent.node = i;
                NewEvents = [NewEvents; newevent];
                clear newevent;
        else %ifnav(i) becomes TRUE.
            if backoff_counter(i) > 1
                backoff_counter(i) = backoff_counter(i) - 1;
            else
                %start a new backoff counter when count-down is zero
                backoff_attempt(i) = backoff_attempt(i) + 1;
                temp = min(backoff_attempt(i)+CW_min, CW_max);
                backoff_counter(i) = floor((2^temp - 1)*rand);
            end
            newevent = event;
            newevent.instant = t;
            newevent.type = 'wait_for_channel';
            newevent.node = i;
            NewEvents = [NewEvents; newevent];
            clear newevent;
                end
            %backoff is NOT DONE: if the node is idle and not keeping NAVs, then continue
backoff. Else, .....


        case 'send_phy' %assume the packet type has been set.
            t = event.instant;
            i = event.node;
            j = event.pkt.rv;
            node(i,3) = event.pkt.power;
            if event.pkt.type == 'data'
                %txtime
= %**************************************************check table
            else
                txtime = tx_time(event.pkt);
            end
            if node(i,4) == 0 & ifnav(i) == 0 %idle and no nav
                if strcmp(event.pkt.type, 'rts')
                    node(i,4) = 1;
                elseif strcmp(event.pkt.type, 'cts')
                    node(i,4) = 2;
                end
            elseif node(i,4) == 10 & ifnav(i) == 0
                if strcmp(event.pkt.type, 'cts3')
                    node(i,4) = 11;
                elseif strcmp(event.pkt.type, 'cts')
                    node(i,4) = 50;
                end
            elseif node(i,4) == 20 & ifnav(i) == 0
                if strcmp(event.pkt.type, 'rtsii')
                    node(i,4) = 21;
                elseif strcmp(event.pkt.type, 'rts')
                    node(i,4) = 22;
```

```
        end
    elseif node(i,4) == 50 & ifnav(i) == 0;
        node(i,4) = 10;
    end

    if j~= 0 %unicast from i to j
        newevent = event;
        newevent.instant = t + txtime; % ????????????????????????
        newevent.type = 'recv_phy';
        newevent.node = j;
        NewEvents = [NewEvents; newevent];
        clear newevent;
        for k = 1:n
            if k == i | k == j
                continue;
            end
            elseif node(k,4) == 20 | node(k,4) == 21 | node(k,4) == 22 |
node(k,4) == 30 | node(k,4) == 40 | node(k,4) == 50 %对接收有干扰
                if sig_detect
                    %delay_recv();
                end
            t1 = rv_threshold_delt;
            % 这个 rv_threshold 是 packet 的
            if snr >= (rv_threshold+t1)
                probability_receive = 1;
            elseif snr < (rv_threshold-t1)
                probability_receive = 0;
            elseif rand <= (snr-(rv_threshold-t1))/(t1+t1)
                probability_receive = 1;
            else
                probability_receive = 0;
            end
            if probability_receive
                newevent = event;
                newevent.instant = t + txtime; % ????????????????????????
                newevent.type = 'recv_phy';
                newevent.node = j;
                NewEvents = [NewEvents; newevent];
                clear newevent;
            end
        end
        %。。。。。。。。。。。。。。。这一段还得要
        if strcmp(event.pkt.type, 'rts')
            %set timeout timer for RTS
            newevent = event;
            newevent.instant = t + (txtime + SIFS + cts_tx_time) *
2; %**************************??????
            newevent.type = 'timeout_rts';
            newevent.node = i;
            NewEvents = [Newevents; newevent];
            clear newevent;
            pending_id(i) = event.pkt.id;
        elseif strcmp(event.pkt.type, 'cts')
            newevent = event;
            newevent.instant = t + (txtime +SIFS + cts_tx_time) * 2;
            newevent.type = 'timeout_cts';
            newevent.node = i;
            NewEvents = [Newevents; newevent];
            clear newevent;
            pending_id(i) = event.pkt.id;
        elseif strcmp(event.pkt.type, 'data') & j ~= 0
            %set timeout timer for DATA
            newevent = event;
            newevent.instant = t + (txtime + SIFS + ack_tx_time) *
2; %****************??????????????????
            newevent.type = 'timeout_data';
```

```
                newevent.node = i;
                NewEvents = [NewEvents; newevent]; clear newevent;
                pending_id(i) = event.pkt.id;
            end
        end

    case 'recv_phy'
        t = event.instant;
        i = event.pkt.tx;
        j = event.node;
        %node(j,4) = 0;
        if node(j,4) == 0
            [pr snr] = recv_phy(i,j,rmodel);
            t1 = rv_threshold_delta;
            if snr >= (rv_threshold + t1);
                probability_receive = 1;
            elseif snr < (rv_threshold - t1);
                probability_receive = 0;
            elseif rand <= (snr-(threshold-t1))/(t1+t1)
                probability_receive = 1;
            else
                probability_receive = 0;
            end
            if probability_receive
                if event.pkt.rv == j        %broadcast or unicast to j
                    newevent = event;
                    newevent.instant = t;
                    newevent.type = 'recv_mac';
                    newevent.node = j;
                    NewEvents = [NewEvents; newevent];
                    clear newevent;
                elseif event.pkt.nav > 0 % this packet is not for j, but use its
nav.....改。。。。。。。。。。。。。。。。。
                    if nav(i,j).start < t
                        nav(i,j).start = t;
                    end
                    if nav(i,j).end < (t + event.pkt.nav)
                        nav(i,j).end = t + event.pkt.nav;
                    end
                end
            end
        end

    case 'recv_mac'
        t = event.instant;
        i = event.pkt.tx;
        j = event.node;
        %if adebug, disp(['recv_mac @ node ' num2str(j)]); end
        %if event.pkt.rv == 0 & strcmp(event.pkt.type, 'data') == 0
            % broadcast but not data packet
        %       error(['recv_mac: node ' num2str(j) ' receives a broadcast packet with a
wrong type: ' event.pkt.type]);
        %end
        %if j == i
        %       % I myself sent this packet, no action
        %       return;
        %end
        switch event.pkt.type
            case 'rts'
                if node(j,4) == 0 & ifnav(j) == 0 & %.....(有给 i 的包)
                    % send back a CTS with two Omega & one P(t), attempting for
full-duplex
                    newevent = event;
                    newevent.instant = t + SIFS + 3*%.....;
                    newevent.type = 'send_phy';
                    newevent.node = j;
```

```
                % keep the data size, rate, and id as RTS packet
                newevent.pkt.type = 'cts';
                newevent.pkt.tx=j;
                newevent.pkt.rv=i;
                newevent.pkt.nav=event.pkt.nav - SIFS - cts_tx_time;
                NewEvents = [NewEvents; newevent]; clear newevent;
            elseif node(j,4) == 0 & ifnav(j) == 0 & %.....(没有给 i 的包）
                % send back a CTS with one Omega & two P(t), attempting for
full-duplex
                newevent = event;
                newevent.instant = t + SIFS + 3*%.....;
                newevent.type = 'send_phy';
                newevent.node = j;
                % keep the data size, rate, and id as RTS packet
                newevent.pkt.type = 'cts';
                newevent.pkt.tx=j;
                newevent.pkt.rv=i;
                newevent.pkt.nav=event.pkt.nav - SIFS - cts_tx_time;
                NewEvents = [NewEvents; newevent]; clear newevent;

            elseif node(j,4) == 0 & ifnav(j) ~= 0
                % send back a CTS with two Omega & one P(t), always
attempting for full-duplex.
                newevent = event;
                newevent.instant = t + SIFS + 3*%.....;
                newevent.type = 'send_phy';
                newevent.node = j;
                % keep the data size, rate, and id as RTS packet
                newevent.pkt.type = 'cts';
                newevent.pkt.tx=j;
                newevent.pkt.rv=i;
                newevent.pkt.nav=event.pkt.nav - SIFS - cts_tx_time;
                NewEvents = [NewEvents; newevent]; clear newevent;

            elseif node(j,4) == 1
                %send back a CTS with one Omega & three P(t), transit to
secondary transmitting mode
                newevent = event;
                newevent.instant = t + SIFS + 4*%.....;
                newevent.type = 'send_phy';
                newevent.node = j;
                % keep the data size, rate, and id as RTS packet
                newevent.pkt.type = 'cts';
                newevent.pkt.tx=j;
                newevent.pkt.rv=i;
                newevent.pkt.nav=event.pkt.nav - SIFS - cts_tx_time;
                NewEvents = [NewEvents; newevent]; clear newevent;

                %search the sending packet event and change its
time.**********************************************

            end

        case 'rtsii'
            if node(j,4) == 0
                %send back a CTS with one Omega and one P(t)
                newevent = event;
                newevent.instant = t + SIFS + 2*%.....;
                newevent.type = 'send_phy';
                newevent.node = j;
                % keep the data size, rate, and id as RTS packet
                newevent.pkt.type = 'cts';
                newevent.pkt.tx=j;
                newevent.pkt.rv=i;
                newevent.pkt.nav=event.pkt.nav - SIFS - cts_tx_time;
                NewEvents = [NewEvents; newevent]; clear newevent;
```

```
            end

        case 'cts'
            % remove pending id for RTS
            if pending_id(j) ~= event.pkt.id
                if ddebug, disp(['the received CTS id ' num2str(event.pkt.id) '
does not match the pending RTS id ' num2str(pending_id(j))]); end
                % probably this CTS is in response to an earlier RTS,
                % but I have retransmitted a new RTS which is replied
                % already or I have retransmitted so many times and given up
                % so we just ignore this CTS.
                return;
            end
            pending_id(j) = 0;
            retransmit(j) = 0;
            % send cts/ctsn
            newevent = event;
            newevent.instant = t + SIFS;
            newevent.type = 'send_phy';
            newevent.node = j;
            % keep the data size and rate as before
            % newevent.pkt.ttl = 1;
            newevent.pkt.type = 'data';
            newevent.pkt.tx=j;
            newevent.pkt.rv=i;
            % creat a new id for the data packet
            newevent.pkt.id = new_id(j);
            newevent.pkt.nav = 0; % not necessary because RTS already did so
            NewEvents = [NewEvents; newevent]; clear newevent;


        case 'ctsf'
            if    ..............exposed    node    和    不    exposed    的    区
别？？？？？？？？？？？？？？？？？？、

        case 'ctsn'


        case 'data'
            % should check that this is not a duplicated or out-of-order packet
            if event.pkt.rv ~= 0        % send ACK if not broadcast
                % send back an ACK
                newevent = event;
                newevent.instant = t + SIFS;
                newevent.type = 'send_phy';
                newevent.node = j;
                % keep the data size, rate, and id the same as DATA packet
                newevent.pkt.type = 'ack';
                newevent.pkt.tx=j;
                newevent.pkt.rv=i;
                newevent.pkt.nav=0; % not necessary because CTS already did
so

                NewEvents = [NewEvents; newevent]; clear newevent;
            end
            % send data up to network layer
            newevent = event;
            % Make sure the ACK is sent out before processing this data packet
in

            % the upper layers because the upper layers may immediately
            % send more packets upon receiving this data packet.
            if event.pkt.rv ~= 0,
                newevent.instant = t + SIFS + ack_tx_time + 2*eps;
            else
                newevent.instant = t + 2*eps;
            end
            newevent.type = 'recv_net';
```

```
                newevent.node = j;
                NewEvents = [NewEvents; newevent]; clear newevent;
            case 'ack' %*****************************************************导致 event list 需要更新
                % make sure the acknowledged packet is the just sent DATA packet
                if pending_id(j) ~= event.pkt.id
                    if ddebug, disp(['the received ACK id=' num2str(event.pkt.id) ' does not match the pending DATA id=' num2str(pending_id(j))]); end
                    % probably this is a duplicated ACK (same reason as the above CTS case)
                    return;
                end
                % remove pending id for DATA
                pending_id(j) = 0;
                retransmit(j) = 0;
                if ~isempty(mac_queue(j).list)
                    % more packets are waiting to be sent
                    % newevent.instant = t + turnaround_time; % switch from receive to transmit
                    % if ddebug, disp('recv_mac: after receiving ACK, take the next packet from mac_queue'); end
                    mac_status(j) = 1;
                    newevent = mac_queue(j).list(1);
                    mac_queue(j).list(1) = [];
                    newevent.instant = t + cca_time;
                    newevent.type = 'wait_for_channel';
                    newevent.node = j;
                    % the packet setup is already done in 'send_mac'
                    NewEvents = [NewEvents; newevent]; clear newevent;
                else
                    mac_status(j) = 0;
                end
            otherwise
                disp(['recv_mac: Undefined mac packet type: ' event.pkt.type]);
        end

    case 'timeout_rts'
        t = event.instant;
        i = event.node;
        j = event.pkt.rv;
        if adebug, disp(['timeout_rts @ node ' num2str(i)]); end
        if pending_id(i) == event.pkt.id % not acknowledged yet, retransmit
            if cdebug, disp(['timeout_rts: node ' num2str(i) ' pending_id=' num2str(pending_id(i)) ' event_id=' num2str(event.pkt.id)]); end
            retransmit(i) = retransmit(i) + 1;
            if retransmit(i) > max_retries
                % so many retries, drop the packet
                if cdebug, disp(['timeout_rts: node ' num2str(i) ' has retried so many times to transmit RTS']); end
                retransmit(i) = 0;
                pending_id(i) = 0;
                % question: what if there are waiting packets in mac_queue?
                % answer: should send them anyway as if the current packet is done.
                % similar to the the operation when ACK is received
                if ~isempty(mac_queue(i).list)
                    % more packets are waiting to be sent
                    % newevent.instant = t + turnaround_time; % switch from receive to transmit
                    mac_status(i) = 1;
                    newevent = mac_queue(i).list(1);
                    mac_queue(i).list(1) = [];
                    newevent.instant = t + cca_time;       % question: cca_time or other
                    newevent.type = 'wait_for_channel';
```

```
                    newevent.node = i;
                    % packet setup is already done in 'send_mac' before put into
the mac_queue
                    NewEvents = [NewEvents; newevent]; clear newevent;
                else
                    % cannot send RTS successfully, reset MAC layer
                    mac_status(i) = 0;
                end
                return;
            end
            if adebug, disp(['timeout_rts: node ' num2str(i) ' to retransmit RTS']);
end
            % retransmit the RTS
            newevent = event;
            newevent.instant = t + cca_time;        % check channel status
            newevent.type = 'wait_for_channel';
            newevent.node = i;
            NewEvents = [NewEvents; newevent]; clear newevent;
        else
            % if pending_id(i) ~= 0 & ddebug, disp(['timeout_rts at node ' num2str(i)
' pending id=' num2str(pending_id(i)) ' does not match the waiting RTS id='
num2str(event.pkt.id)]); end
        end
    case 'timeout_data'
        t = event.instant;
        i = event.node;
        j = event.pkt.rv;
        if adebug, disp(['timeout_data @ node ' num2str(i)]); end
%           if pending_id(i) == event.pkt.id % not acknowledged yet
%               if adebug, disp(['timeout_data: node ' num2str(i) ' failed to transmit
DATA, go back to transmit RTS']); end
%               % remove the pending id for DATA
%               pending_id(i) = 0;
%               retransmit(i) = 0;
%               % go back to send RTS
%               newevent = event;
%               newevent.instant = t + cca_time;        % check channel status
%               newevent.type = 'wait_for_channel';
%               newevent.node = i;
%               newevent.pkt.type = 'data';
%               newevent.pkt.nav = SIFS + cts_tx_time + SIFS + tx_time(newevent.pkt)
+ SIFS + ack_tx_time;
%               newevent.pkt.type = 'rts';
%               % create a new id for the new RTS
%               newevent.pkt.id = new_id(i);
%               NewEvents = [NewEvents; newevent]; clear newevent;
%           end
        if pending_id(i) == event.pkt.id % not acknowledged yet
            if cdebug, disp(['timeout_data: node ' num2str(i) ' pending_id='
num2str(pending_id(i)) ' event_id=' num2str(event.pkt.id)]); end
            retransmit(i) = retransmit(i) + 1;
            if retransmit(i) > max_retries
                % so many retries, drop the data packet
                if cdebug, disp(['timeout_data: node ' num2str(i) ' has retried so
many times to transmit DATA']); end
                retransmit(i) = 0;
                pending_id(i) = 0;
                if ~isempty(mac_queue(i).list)
                    % more packets are waiting to be sent
                    mac_status(i) = 1;
                    newevent = mac_queue(i).list(1);
                    mac_queue(i).list(1) = [];
                    newevent.instant = t + cca_time;        % question: cca_time or
other
                    newevent.type = 'wait_for_channel';
                    newevent.node = i;
```

```
                    NewEvents = [NewEvents; newevent]; clear newevent;
                else
                    % Cannot send DATA successfully, reset MAC layer
                    mac_status(i) = 0;
                end
                return;
            end
            if adebug, disp(['timeout_data: node ' num2str(i) ' to retransmit DATA']);
end
            % retransmit the DATA
            newevent = event;
            newevent.instant = t + cca_time;        % check channel status
            newevent.type = 'wait_for_channel';
            newevent.node = i;
            % newevent.pkt.type = 'data';
            newevent.pkt.nav = SIFS + ack_tx_time; % necessary for retransmission
because the initial DATA has NAV=0
            NewEvents = [NewEvents; newevent]; clear newevent;
        end


    case 'send_phy_finish'
```