1. Modified Tree identify (submitted by Takayuki Nyu)

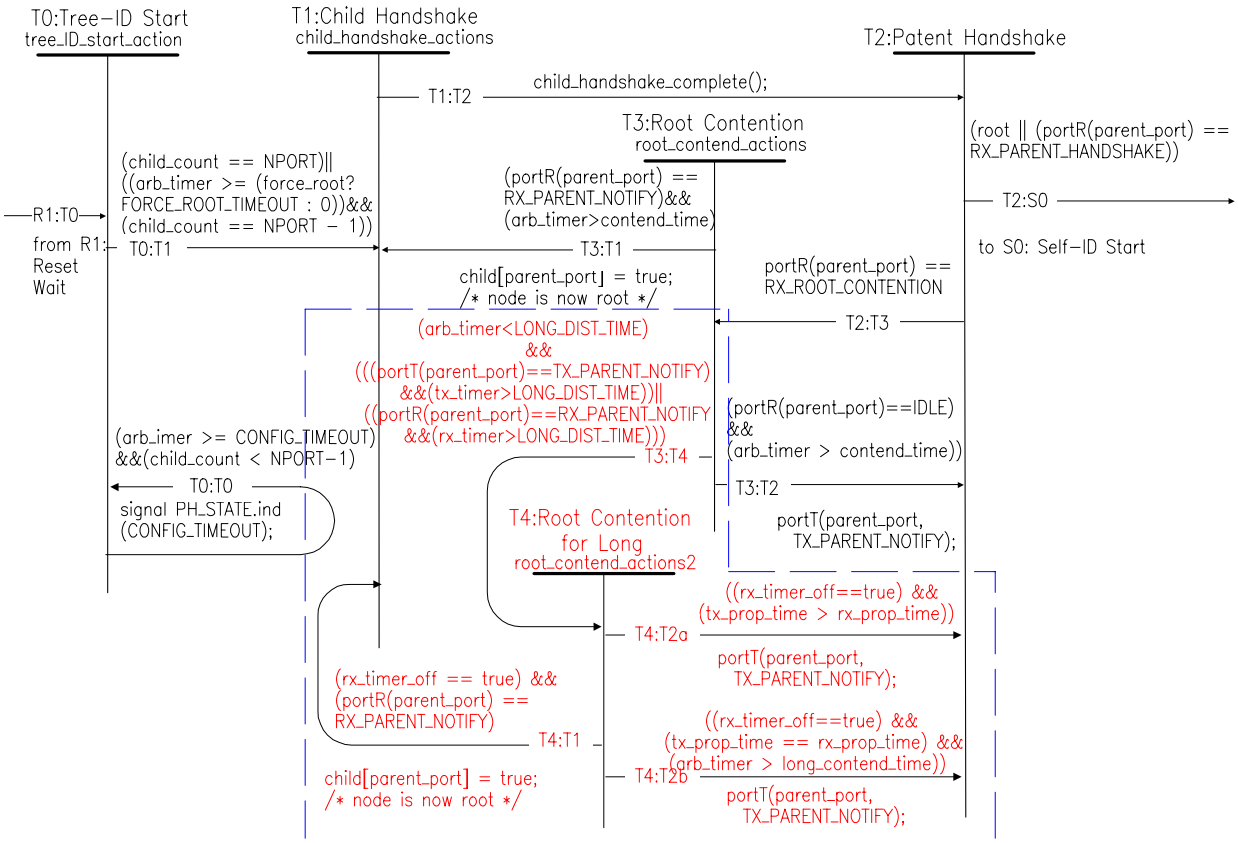
1. Additional PHY timing constants

Timing constant	Minimum	Maximum	Comment
LONG_DIST_TIME		0.15μs	Transition timing from the traditional re resolution method of root contention to the another method for long distance
BACK_OFF_TIME	0.04μs		Time to wait in state T4 before transition to state T2

2. Cable PHY code definition

```
timer  tx_timer;      // timer for measurement the duration of tx_parent_notify
timer  rx_timer;      // timer for measurement the duration of rx_parent_notify
int  tx_prop_timer;   // the duration of tx_parent_notify
int  rx_prop_timer    // the duration of rx_parent_notify
boolean  rx_timer_off  // set when reception of rx_parent_notify is complete
baserate_time  long_contend_time  // amount of time to wait for re-start tx_parent_notify
```

3. *Tree-ID state machine*



F

igure 1 : Modified Tree-ID state machine

The parts enclosed by blue dashed lines is added.

1.Tree-ID state machine notes

State T4 : Root Contention for long cable. At this point, both nodes compare the duration of tx_parent_notify and the duration of rx_parent_notify. If the duration of tx_parent_notify is longer than the duration of rx_parent_notify, the node will be child node. If the duration of rx_parent_notify is longer than the duration of tx_parent_notify, the node will be parent node, that is, root.

Transition T3 : T4. When tx_timer or rx_timer is more than LONG_DIST_TIME, the resolution method for long cable is applied.

Transition T4 : T1. If the duration of rx_parent_notify is longer than the duration of tx_parent_notify and the node receives the rx_parent_notify, the node take on the role of bus root.

Transition T4 : T2a. If the duration of tx_parent_notify is longer than the duration of rx_parent_notify, the node once again sends the tx_parent_notify signal.

Transition T4 : T2b. If the duration of tx_parent_notify is equal to the duration of rx_parent_notify, the nodes wait for long_contend_time in state T4, then once again send the tx_parent_notify signal.

2.Tree-ID actions and conditions

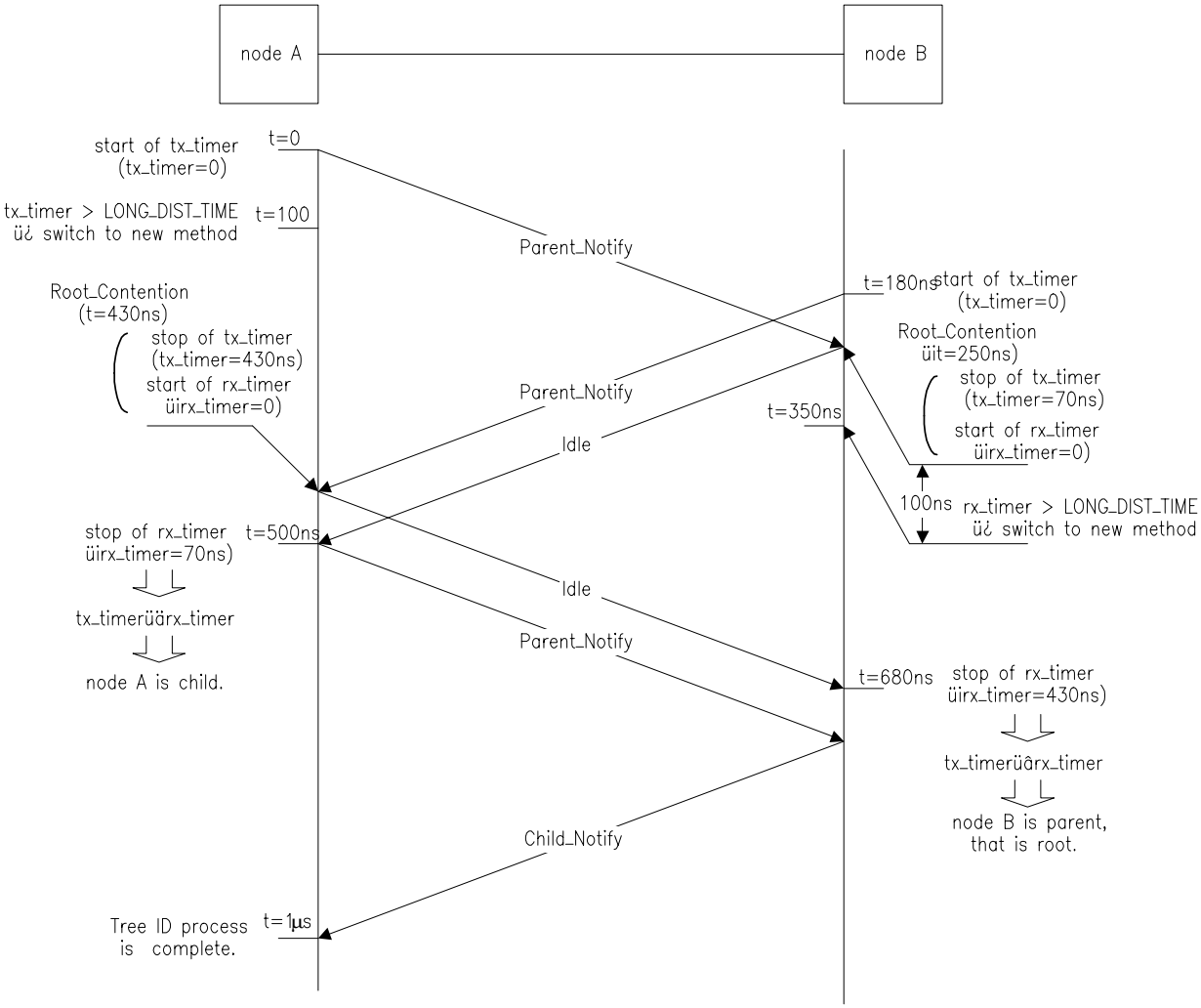
Table 1 : modified IEEE 1394-1995 Table 4-45

```
void tree_ID_start_actions() {
int i, temp_count;
arb_timer = 0;                    // start timer
while(true) {                     // loop forever
   temp_count = 0;                // temporary child counter
   for (i = 0; i < NPORT; i++)
          if (connected[i] || portR(i) == RX_PARENT_NOTIFY) {
                    // when unconnected or receiving "you are my
parent"
        child[i] = true;          // set child flag
        temp_count++;             // and increment counter
   child_count = temp_count;      // set current child count
    }                             // end of forever loop
}
void child_handshake_actions() {
int i;
root = true;                      // root will stay true if all ports are child ports
for (i = 0; i < NPORT; i++) {
   if (connected[i] && child[i])
      portT(i, TX_CHILD_NOTIFY);  // you are my child
   else if (connected[i]) {
      portT(i, TX_PARENT_NOTIFY); // you are my parent
      parent_port = i;            // there is at most one port with child==false
      root = false;               // cannot be root since there is a parent
      tx_timer=0;         // start transmission timer
   }
}
boolean child_handshake_complete() {   // true id all active children in "Start Self_ID"
int i;
for (i = 0; i < NPORT; i++)
   if (child[i] && connected[i] && (portR(i) != RX_CHILD_HANDSHAKE)
```

```
        return false;                // active child not giving "you are my parent"
return true;                         // will also be true if there are no active
children
}
void root_contend_actions() {
int i;
contend_time = (random_bool() ? CONTEND_SLOW : CONTEND_FAST);
tx_prop_time = tx_timer;     // set transmission time
for (i = 0; i < NPORT; i++) {
   if (child[i])
        portT(i, TX_CHILD_NOTIFY);    // you are my child
   else
        portT(i, IDLE);              // abandon "you are my parent" request
   }
arb_timer = 0;                       // start arbitration timer
rx_timer = 0;               // start reception timer
}

void root_contend_actions2( ) {
long_contend_time = ( random_bool( ) ? 0 : BACK_OFF_TIME );
while ( portR(parent_port) == RX_PARENT_NOTIFY ) {
      rx_timer_off = false;
}
if ( portR(parent_port) == IDLE ) {
      rx_timer_off = true;
      rx_prop_time = rx_timer;
      arb_timer = 0;       // start arbitration timer
}
```

Example : The distance between nodes is 50m


node A
node B
start of tx_timer
(tx_timer=0)
t=0
tx_timer > LONG_DIST_TIME
ü¿ switch to new method
t=100
Parent_Notify
t=180ns
start of tx_timer
(tx_timer=0)
Root_Contention
(t=430ns)
stop of tx_timer
(tx_timer=430ns)
start of rx_timer
üirx_timer=0)
Root_Contention
üit=250ns)
stop of tx_timer
(tx_timer=70ns)
start of rx_timer
üirx_timer=0)
Parent_Notify
t=350ns
Idle
100ns
rx_timer > LONG_DIST_TIME
ü¿ switch to new method
stop of rx_timer
üirx_timer=70ns)
t=500ns
tx_timerüärx_timer
node A is child.
Idle
Parent_Notify
t=680ns
stop of rx_timer
üirx_timer=430ns)
tx_timerüârx_timer
node B is parent,
that is root.
Child_Notify
Tree ID process
is complete.
t=1µs