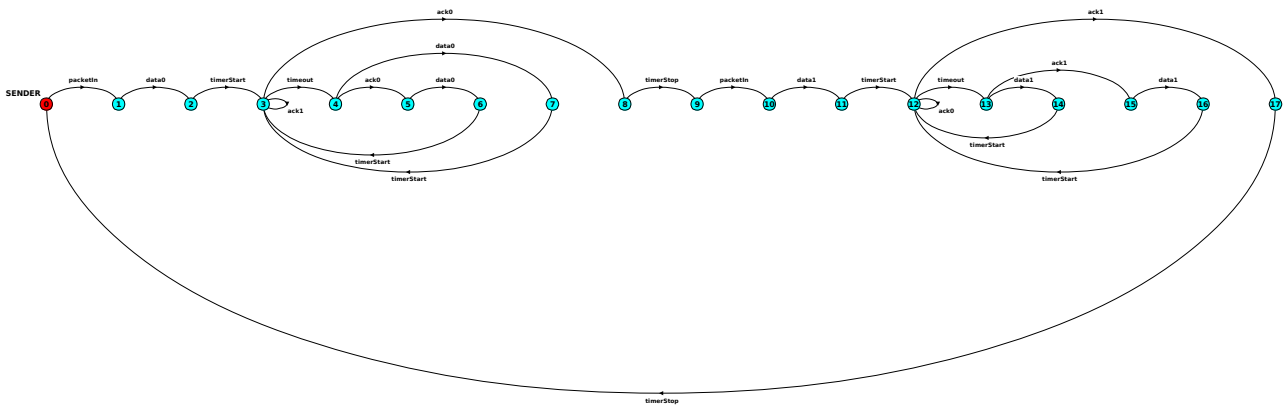
ALTERNATING BIT PROTOCOL

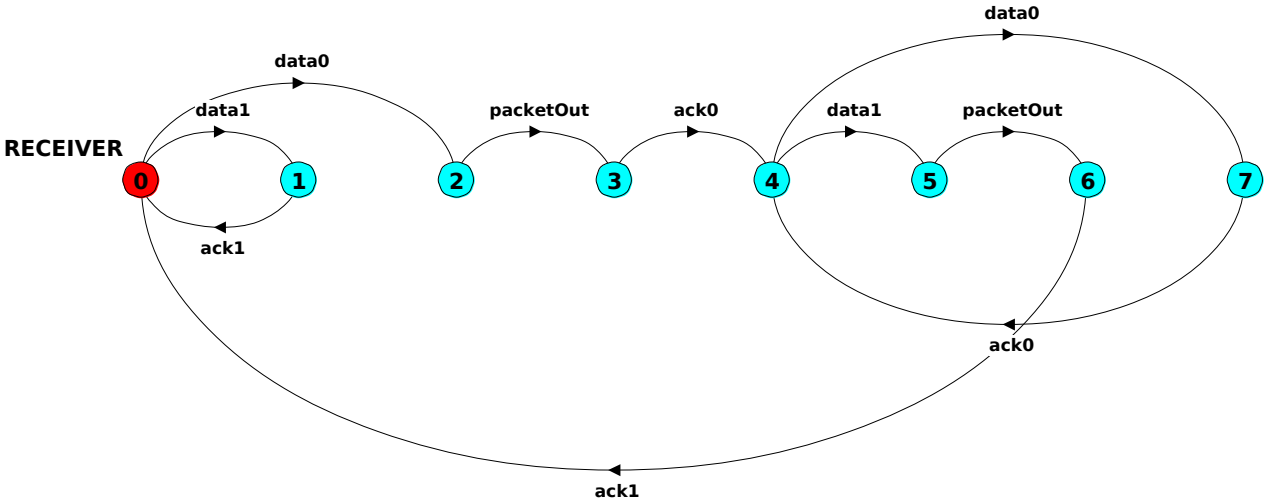
(Robert Meolic, 2008, 2013)

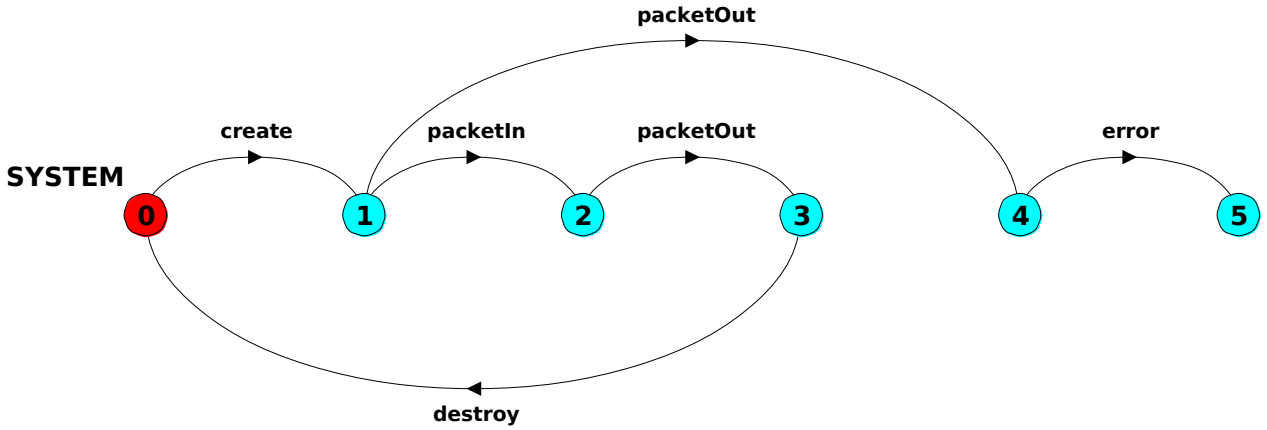
File **abp-simple.ccs** is an implementation of alternating bit protocol. In this version, data are never lost or corrupted. This is correct implementation.

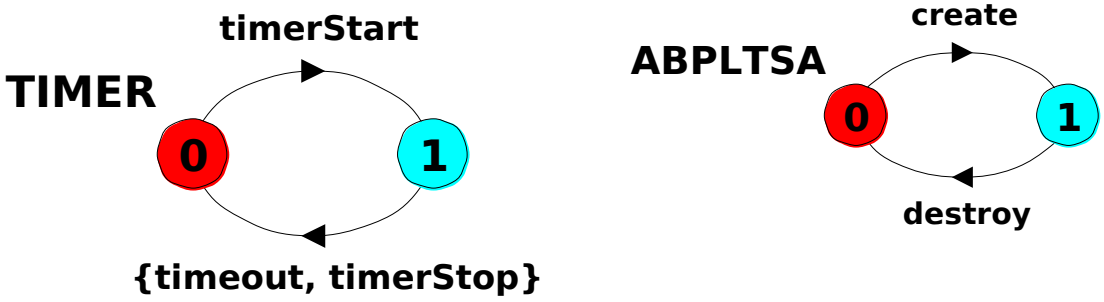
```
agent SENDER = Ready0
Ready0 = ?packetIn.!data0.!timerStart.Wait0
Wait0 = ?ack0.!timerStop.Ready1 + ?ack1.Wait0 +
         ?timeout.(!data0.!timerStart +
                   ?ack0.!data0.!timerStart).Wait0
Ready1 = ?packetIn.!data1.!timerStart.Wait1
Wait1 = ?ack1.!timerStop.Ready0 + ?ack0.Wait1 +
         ?timeout.(!data1.!timerStart +
                   ?ack1.!data1.!timerStart).Wait1

agent RECEIVER = Get0
Get0 = ?data0.!packetOut.!ack0.Get1 + ?data1.!ack1.Get0
Get1 = ?data1.!packetOut.!ack1.Get0 + ?data0.!ack0.Get1

agent TIMER = TimerIdle
TimerIdle = ?timerStart.(?timerStop + !timeout).TimerIdle

agent SYSTEM = SystemIdle
SystemIdle = ?create.(?packetOut.!error.STOP +
                      !packetIn.?packetOut.!destroy.SystemIdle)

raw composition ABP =
           |(SENDER,RECEIVER,TIMER,SYSTEM)
            \packetIn\packetOut
            \data0\data1\ack0\ack1
            \timerStart\timerStop\timeout
```

Here are LTSs used for specification and the composed LTS minimised with LTSA V3.0.









The following ACTLW formulae are used to verify the correctness:

```
/* Safety: is there a reachable error */
property F1 == EEF {!error};

/* Liveness: is there a reachable deadlock state */
property F2 == AAX {false} OR EEF AAX {false};

/* Liveness: is there a reachable divergent state */
property F3 == (EEX {true} AND EEG {tau} EEX {true}) OR
          (EEF (EEX {true} AND EEG {tau} EEX {true}));
```

Error and deadlock state are not reachable. Divergent state is reachable because of unfair paths. EST does not support fairness constraints, yet.

Here is the log from EST:

```
Efficient Symbolic Tools, 2nd Edition, Copyright (C) 2003-2013 UM-FERI
This is free software, and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute EST; type "license" for details.

Running on i686 (Linux, 3.2.0-38-generic-pae) with tcl 8.5.11 and tk 8.5.11.

Initialization of GUI package... OK
Initialization of BDD package... OK
Initialization of Process_Algebra package... OK
Initialization of Versis package... OK
Initialization of Model checking package... OK
Initialization of Strucval package... OK
Initialization of CCS package... OK
Ready.

>ccs_read /home/meolic/est/est-2ed/data/abp/abp-simple.ccs
Reading file: /home/meolic/est/est-2ed/data/abp/abp-simple.ccs
Alternating bit protocol, Robert Meolic, 2008
In this version, data are never lost or corrupted
This is correct implementation.
  Process SENDER ... OK
  Process Ready0 ... OK
  Process Wait0 ... OK
  Process Ready1 ... OK
  Process Wait1 ... OK
  Process RECEIVER ... OK
  Process Get0 ... OK
  Process Get1 ... OK
  Process TIMER ... OK
  Process TimerIdle ... OK
  Process SYSTEM ... OK
  Process SystemIdle ... OK
  Parallel composition ABP ...OK
  Property F1
    F1 = EEF {!error};
  Property F2
    F2 = AAX {false} OR EEF AAX {false};
  Property F3
    F3 = (EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true}));

>mc_check_actl 1 ABP F1
ACTL/ACTLW model checking on composition ABP
EEF {!error} ==> FALSE

>mc_check_actl 1 ABP F2
ACTL/ACTLW model checking on composition ABP
AAX {false} OR EEF AAX {false} ==> FALSE

>mc_check_actl 1 ABP F3 [expr $mc_diagnostic | $mc_explain | $mc_tracepath]
ACTL/ACTLW model checking on composition ABP
(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> TRUE
@@ Diagnostics
@@ #0:(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})):T:
((ready04<Ready0>),(get08<Get0>),(timeridle5<TimerIdle>),(systemidle9<SystemIdle>))
@@ Formula #0 is valid in state ((ready04<Ready0>),(get08<Get0>),(timeridle5<TimerIdle>),
(systemidle9<SystemIdle>)) because:
@@ 1. formula #1 is not valid in it,
@@ 2. formula #5 is valid in it.
@@ #5:EEF (EEX {true} AND EEG {TAU} EEX {true}):T:((ready04<Ready0>),(get08<Get0>),
(timeridle5<TimerIdle>),(systemidle9<SystemIdle>))
@@ There exist a path satisfying formula #5: ((ready04<Ready0>),(get08<Get0>),
(timeridle5<TimerIdle>),(systemidle9<SystemIdle>))--create?->((ready04<Ready0>),(get08<Get0>),
(timeridle5<TimerIdle>),(systemidle8<SystemIdle>))
@@ #6:EEX {true} AND EEG {TAU} EEX {true}:T:((ready04<Ready0>),(get08<Get0>),
(timeridle5<TimerIdle>),(systemidle8<SystemIdle>))
@@ Formula #6 is valid in state ((ready04<Ready0>),(get08<Get0>),(timeridle5<TimerIdle>),
(systemidle8<SystemIdle>)) because:
@@ 1. formula #7 is valid in it,
@@ 2. formula #9 is valid in it.
WARNING (operator AND): Cannot explain both part at once.
@@ Witness: (create?)
Spin trail generated and saved into file wc.out
```

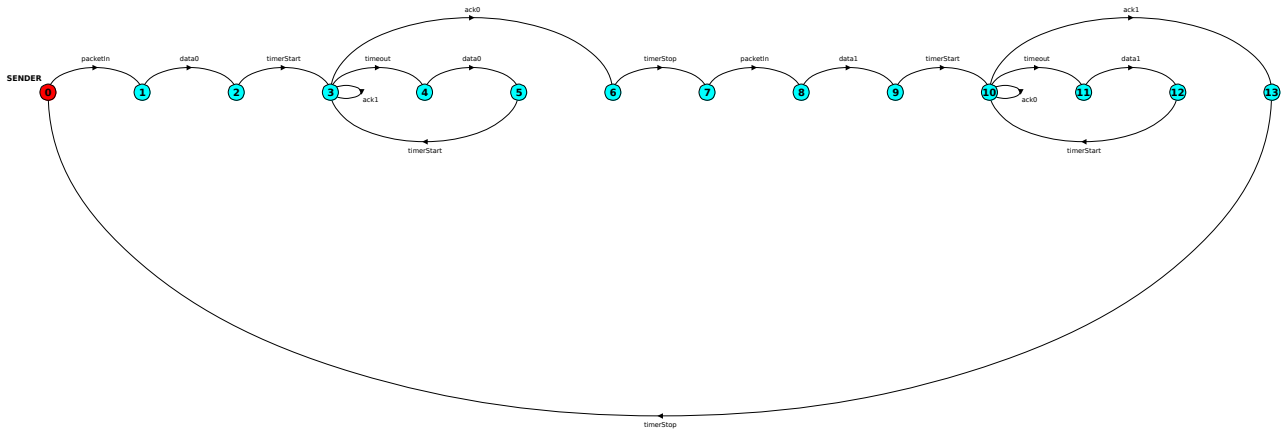
```
MSC generated and saved into file wc.msc
@@ End Of Diagnostic
```

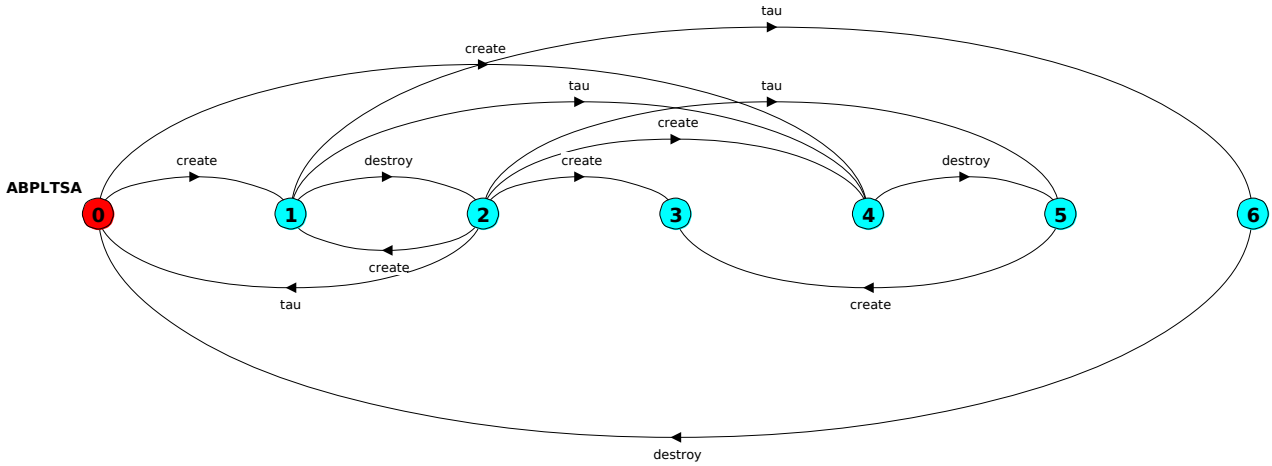
File **abp-simple-error.ccs** is another implementation of simplified alternating bit protocol. This implementation has deadlock due to synchronisation problem in SENDER (there are no channels!). Processes RECEIVER, TIMER, and SYSTEM are the same as in **abp-simple.ccs**.

```
agent SENDER = Ready0
Ready0 = ?packetIn.!data0.!timerStart.Wait0
Wait0 = ?ack0.!timerStop.Ready1 + ?ack1.Wait0 +
        ?timeout.!data0.!timerStart.Wait0
Ready1 = ?packetIn.!data1.!timerStart.Wait1
Wait1 = ?ack1.!timerStop.Ready0 + ?ack0.Wait1 +
        ?timeout.!data1.!timerStart.Wait1
```

For this system, error and divergent state are not reachable. Deadlock state is reachable.

Here is LTS used for specification of SENDER and the composed LTS minimised with LTSA V3.0.

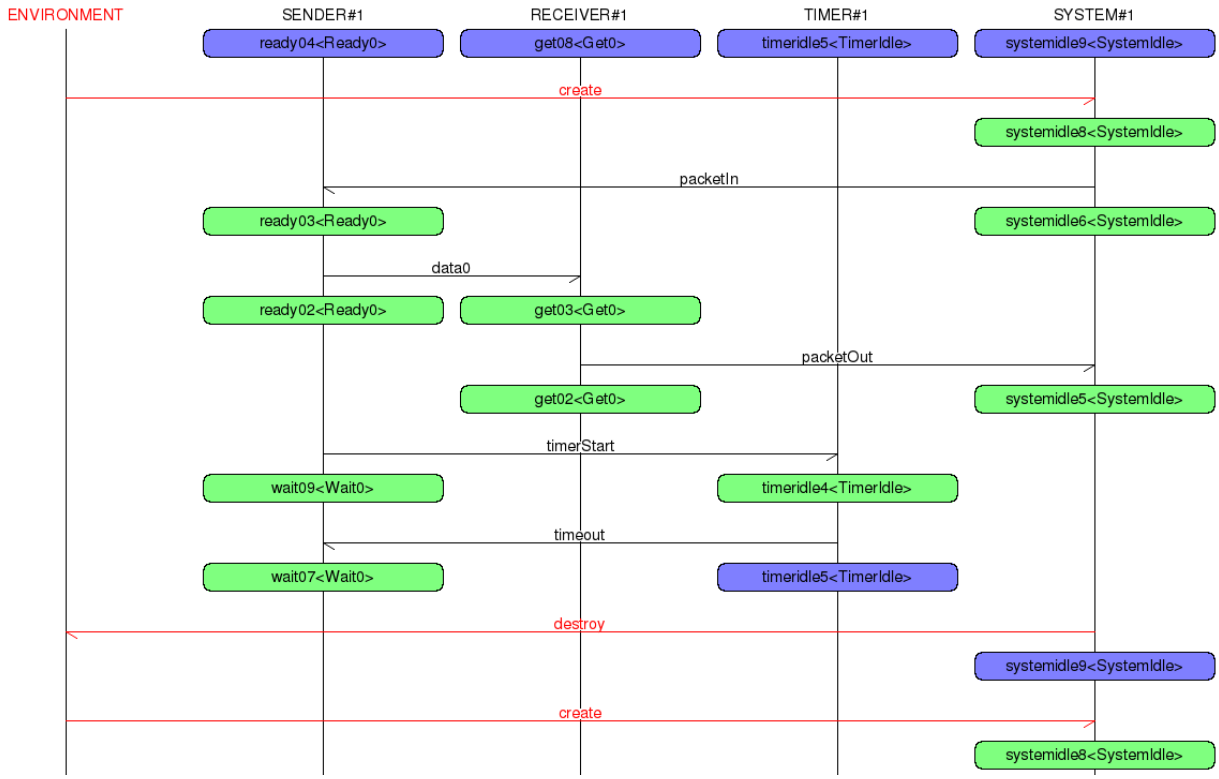




Here is the log from EST:

```
>mc_check_actl 1 ABP F2 [expr $mc_diagnostic | $mc_explain | $mc_tracepath]
ACTL/ACTLW model checking on composition ABP
AAX {false} OR EEF AAX {false} ==> TRUE
@@ Diagnostics
@@ #0:AAX {false} OR EEF AAX {false}:T:((ready04<Ready0>),(get08<Get0>),(timeridle5<TimerIdle>),
(systemidle9<SystemIdle>))
@@ Formula #0 is valid in state ((ready04<Ready0>),(get08<Get0>),(timeridle5<TimerIdle>),
(systemidle9<SystemIdle>)) because:
@@ 1. formula #1 is not valid in it,
@@ 2. formula #2 is valid in it.
@@ #2:EEF AAX {false}:T:((ready04<Ready0>),(get08<Get0>),(timeridle5<TimerIdle>),
(systemidle9<SystemIdle>))
@@ There exist a path satisfying formula #2: ((ready04<Ready0>),(get08<Get0>),
(timeridle5<TimerIdle>),(systemidle9<SystemIdle>))--create?->((ready04<Ready0>),(get08<Get0>),
(timeridle5<TimerIdle>),(systemidle8<SystemIdle>))--TAU->((ready03<Ready0>),(get08<Get0>),
(timeridle5<TimerIdle>),(systemidle6<SystemIdle>))--TAU->((ready02<Ready0>),(get03<Get0>),
(timeridle5<TimerIdle>),(systemidle6<SystemIdle>))--TAU->((ready02<Ready0>),(get02<Get0>),
(timeridle5<TimerIdle>),(systemidle5<SystemIdle>))--TAU->((wait09<Wait0>),(get02<Get0>),
(timeridle4<TimerIdle>),(systemidle5<SystemIdle>))--TAU->((wait07<Wait0>),(get02<Get0>),
(timeridle5<TimerIdle>),(systemidle5<SystemIdle>))--destroy!->((wait07<Wait0>),(get02<Get0>),
(timeridle5<TimerIdle>),(systemidle9<SystemIdle>))--create?->((wait07<Wait0>),(get02<Get0>),
(timeridle5<TimerIdle>),(systemidle8<SystemIdle>))
@@ #3:AAX {false}:T:((wait07<Wait0>),(get02<Get0>),(timeridle5<TimerIdle>),
(systemidle8<SystemIdle>))
@@ All transitions satisfy formula #3.
@@ Witness: (create?)(TAU)(TAU)(TAU)(TAU)(TAU)(destroy!)(create?)
Spin trail generated and saved into file wc.out
MSC generated and saved into file wc.msc
@@ End Of Diagnostic
```

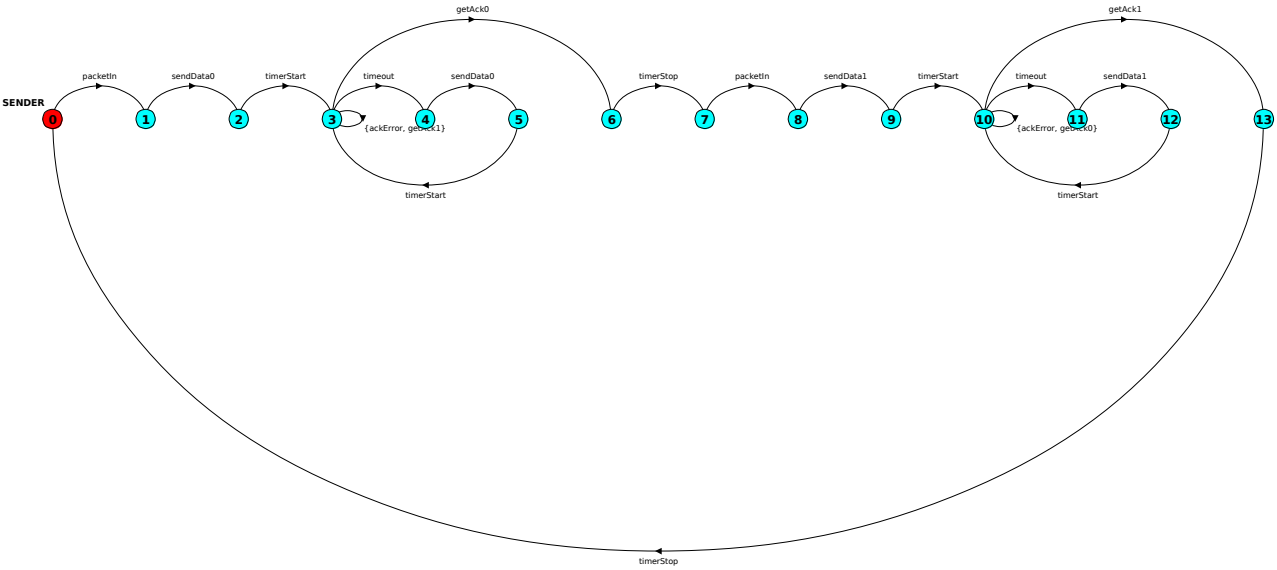
Here is the MSC produced by EST and visualised with mscgen v0.20.

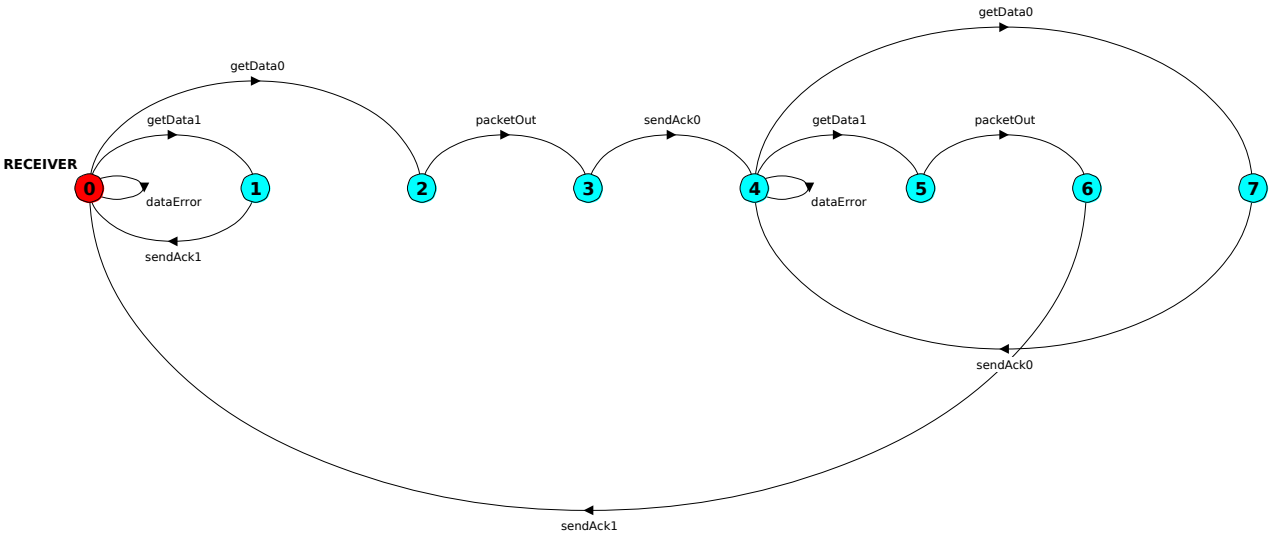


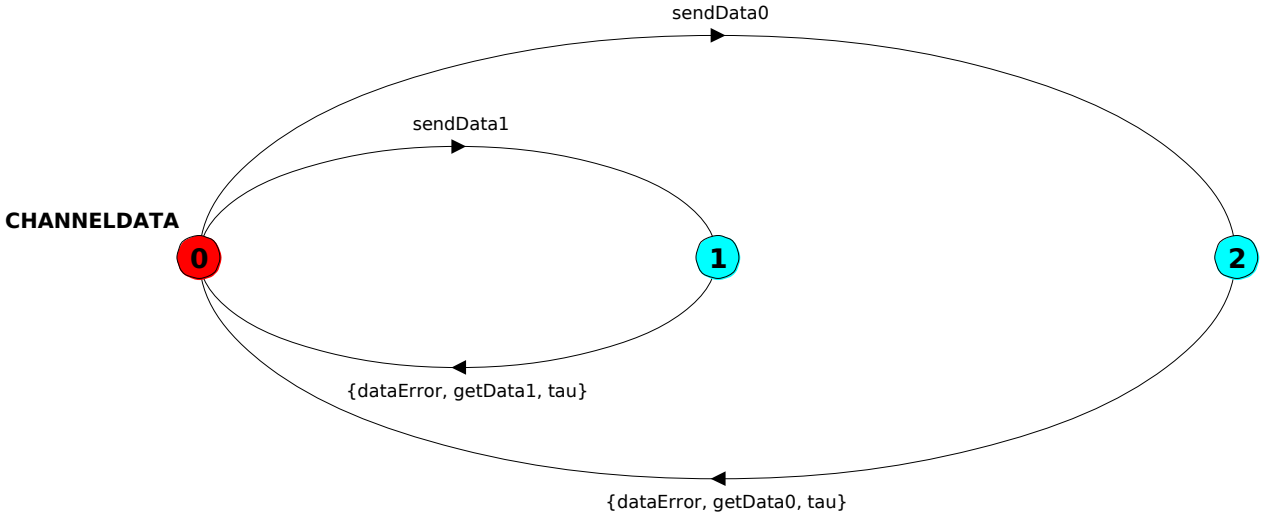
File **abp.ccs** is more complete implementation of alternating bit protocol. In this version, data can be lost or corrupted. This is correct implementation. Processes TIMER and SYSTEM are the same as in **abp-simple.ccs**.

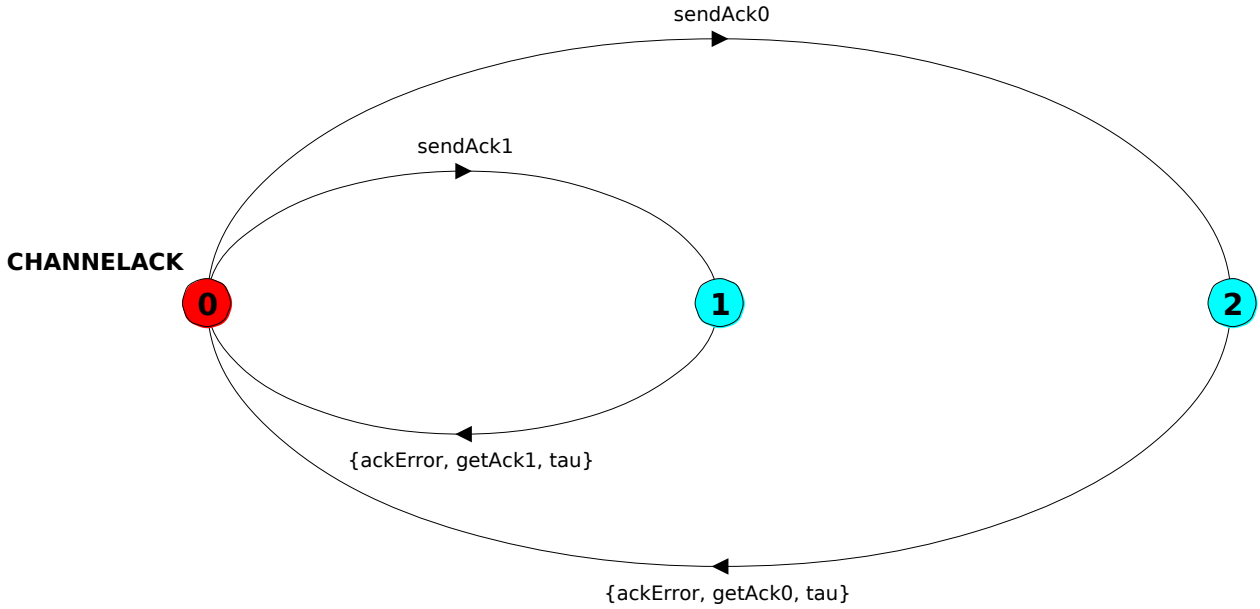
```
agent SENDER = Ready0
Ready0 = ?packetIn.!sendData0.!timerStart.Wait0
Wait0 = ?getAck0.!timerStop.Ready1 +
        ?getAck1.Wait0 + ?ackError.Wait0 +
        ?timeout.!sendData0.!timerStart.Wait0
Ready1 = ?packetIn.!sendData1.!timerStart.Wait1
Wait1 = ?getAck1.!timerStop.Ready0 +
        ?getAck0.Wait1 + ?ackError.Wait1 +
        ?timeout.!sendData1.!timerStart.Wait1

agent RECEIVER = Get0
Get0 = ?getData0.!packetOut.!sendAck0.Get1 +
       ?getData1.!sendAck1.Get0 + ?dataError.Get0
Get1 = ?getData1.!packetOut.!sendAck1.Get0 +
       ?getData0.!sendAck0.Get1 + ?dataError.Get1

agent CHANNELDATA = ChannelDataIdle
ChannelDataIdle =?sendData0.(!getData0 + !dataError + tau).
                 ChannelDataIdle +
                 ?sendData1.(!getData1 + !dataError + tau).
                 ChannelDataIdle

agent CHANNELACK = ChannelAckIdle
ChannelAckIdle = ?sendAck0.(!getAck0 + !ackError + tau).
                 ChannelAckIdle +
                 ?sendAck1.(!getAck1 + !ackError + tau).
                 ChannelAckIdle

raw composition ABP =
           |(SENDER,RECEIVER,TIMER,CHANNELDATA,CHANNELACK,SYSTEM)
           \packetIn\packetOut
           \sendData0\getData0\sendData1\getData1
           \sendAck0\getAck0\sendAck1\getAck1
           \dataError\ackError
           \timerStart\timerStop\timeout
```

Here are LTSs used for specification.









For this system, error and deadlock state are not reachable. Divergent state is reachable. Here is the log from EST:

```
>outline [pa_comp_state_number ABP]
430
>outline [pa_comp_transition_number ABP]
1220
>outline [pa_comp_transition_visible ABP]
216

>mc_check_actl 1 ABP F3 [expr $mc_diagnostic | $mc_explain | $mc_tracepath]
ACTL/ACTLW model checking on composition ABP
(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> TRUE
@@ Diagnostics
@@ #0:(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})):T:
((ready04<Ready0>),(get09<Get0>),(timeridle5<TimerIdle>),(channeldataidle12<ChannelDataIdle>),
(channelackidle12<ChannelAckIdle>),(systemidle9<SystemIdle>))
@@ Formula #0 is valid in state ((ready04<Ready0>),(get09<Get0>),(timeridle5<TimerIdle>),
(channeldataidle12<ChannelDataIdle>),(channelackidle12<ChannelAckIdle>),(systemidle9<SystemIdle>))
because:
@@ 1. formula #1 is not valid in it,
@@ 2. formula #5 is valid in it.
@@ #5:EEF (EEX {true} AND EEG {TAU} EEX {true}):T:((ready04<Ready0>),(get09<Get0>),
(timeridle5<TimerIdle>),(channeldataidle12<ChannelDataIdle>),(channelackidle12<ChannelAckIdle>),
(systemidle9<SystemIdle>))
@@ There exist a path satisfying formula #5: ((ready04<Ready0>),(get09<Get0>),
(timeridle5<TimerIdle>),(channeldataidle12<ChannelDataIdle>),(channelackidle12<ChannelAckIdle>),
(systemidle9<SystemIdle>))--create?->((ready04<Ready0>),(get09<Get0>),(timeridle5<TimerIdle>),
(channeldataidle12<ChannelDataIdle>),(channelackidle12<ChannelAckIdle>),(systemidle8<SystemIdle>))
@@ #6:EEX {true} AND EEG {TAU} EEX {true}:T:((ready04<Ready0>),(get09<Get0>),
(timeridle5<TimerIdle>),(channeldataidle12<ChannelDataIdle>),(channelackidle12<ChannelAckIdle>),
(systemidle8<SystemIdle>))
@@ Formula #6 is valid in state ((ready04<Ready0>),(get09<Get0>),(timeridle5<TimerIdle>),
(channeldataidle12<ChannelDataIdle>),(channelackidle12<ChannelAckIdle>),(systemidle8<SystemIdle>))
because:
@@ 1. formula #7 is valid in it,
@@ 2. formula #9 is valid in it.
WARNING (operator AND): Cannot explain both part at once.
@@ Witness: (create?)
Spin trail generated and saved into file wc.out
MSC generated and saved into file wc.msc
@@ End Of Diagnostic
```