Fra harde µ-sekunder via mjuke sekunder til forte år: sann tid i industrien

Øyvind Teig

senior utviklingsingeniør
Autronica Fire and Security, «a UTC company»

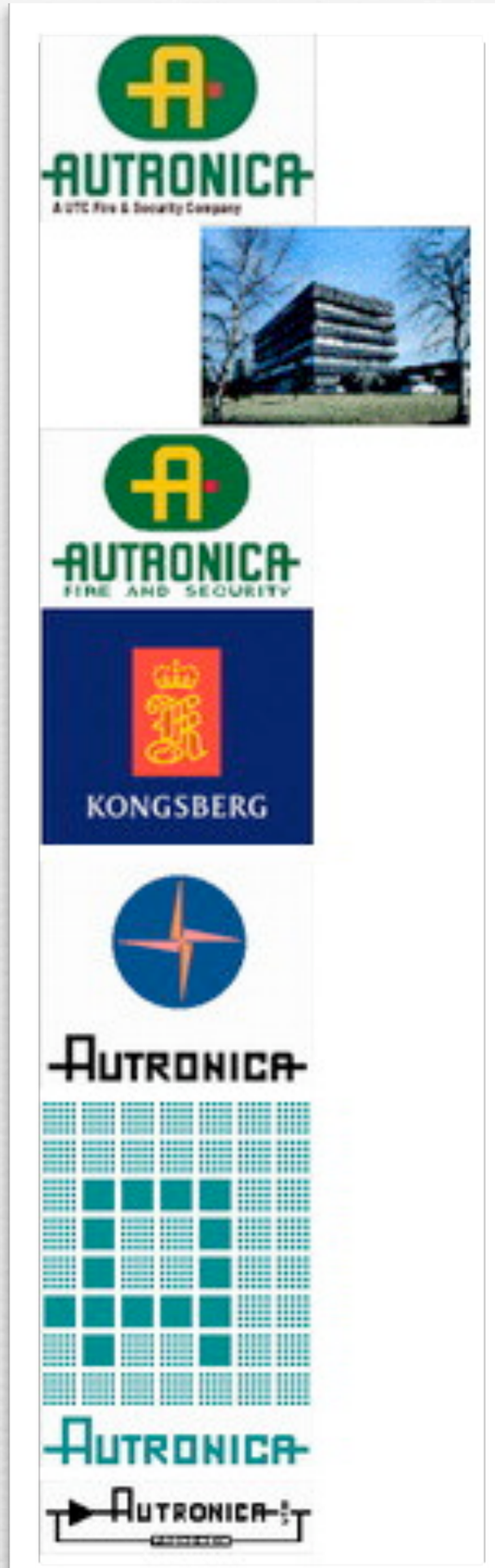
Gjesteforelesning, 23. nov. 2010
NTNU, fag TTK 4145 Sanntidsprogrammering (Real-Time Programming)

Autronica Fire and Security (AFS)

- ✓ Ca.400 ansatte med ca.40 på utvikling
- ✓ 6-700 mill NOK omsetning
- ✓ På Lade, her i Trondheim
- ✓ Samme sted, og for meg - seks logoer
- ✓ Begynte med "gründerne" i 1957
- ✓ Børs og forskjellige eiere

Autronica Fire and Security (AFS)
Siden 2005

- "A UTC Fire & Security Company" - United Technologies Corporation, 210000 ansatte med selskap som bla.a. Otis, Pratt & Whitney, Hamilton Sundstrand, Carrier og tidligere Kidde (nå UTC Fire & Security, som vi er en del av)
- Så vi har nå søsterselskap å jobbe sammen med. Det øker staben og produktmangfoldet



Punktene i denne forelesningen står for egen regning!

CV

- ✓ NTH, 1975
- ✓ Autronica, 1976-nå (+studentjobbing der fra 1972)
- ✓ HW og SW (mest SW)
- ✓ Har jobbet med embeddedsystemer hele tida
- ✓ Publisert en del - relativt uvanlig i industrien
- ✓ Hjemmesiden min: http://www.teigfam.net/oyvind

CV

- ✓ NTH, 1975
- ✓ Autronica, 1976-nå (+studentjobbing der fra 1972)
- ✓ HW og SW (mest SW)
- ✓ Har jobbet med embeddedsystemer hele tida
- ✓ Publisert en del - relativt uvanlig i industrien
- ✓ Hjemmesiden min: http://www.teigfam.net/oyvind

«Pappesken» (SW)

- ✓ HW/SW for 30+ år i den pappesken
- ✓ Diverse assembler (78-80)
- ✓ PL/M (80-90)
- ✓ Modula-2 (88-90)
- ✓ MPP-Pascal (82-88)
- ✓ occam (90-01)
- ✓ C (02-nå)
- ✓ Java (97-00)
- ✓ Perl (02)

«Pappesken» (SW)

Øverst til høyre (fra diplomen 1975)

Den papirtapen inneholder fremdeles programmet!

Sann tid
går alt for fort!

«Pappesken» (SW)

Øverst til høyre (fra diplomen 1975)

Den papirtapen inneholder
fremdeles programmet!

«Pappesken» (mimre)

- ✓ Ikke *en* kjedelig dag på mer enn 34 år!
- ✓ Meningsfylt!
- ✓ Sanntidsprogrammering er gøy!
- ✓ Og det kommer bare til å bli mer og mer av det!

«Pappesken» (sanntidsmetodikk & produkt)

- ✓ 1978: Ikke noe kjøresystem, ren assembler
  Dieselaggregat start/stopp nødstrøm
- ✓ 1979: MPP Pascal med prosessbegrep
  Protokollkonvertering, nivåmåling og brannvarsling
- ✓ 1980: PL/M med NTH-utviklet kjøresystem
  Maskinromsovervåking
- ✓ 1982: Assembler med jobbsnekret kjøresystem
  Brannvarsling
- ✓ 1988: PL/M med jobbsnekret kjøresystem
  Brannvarsling

«Pappesken» (sanntidsmetodikk & produkt)

- ✓ 1988: Modula-2 med kjøpt kjøresystem
  Brannvarsling
- ✓ 1990: occam med prosessbegrep
  Motormålinger og effektberegninger
- ✓ 1995: C med VxWorks opsys/kjøresystem
  Brannvarsling
- ✓ 2003-8: C med CSP kanaler oppå SDL kjøresystem
  Brannvarsling
- ✓ 2009: "ChanSched" stand-alone kjøresystem, med kollega..
  Brannvarsling
- ✓ 2010: ..brukt i patentert enhet (AutroKeeper) i nytt cruiseskip med det nye internasjonale kravet "Safe Return to Port"
  Brannvarsling

Lange år

- ✓ OO er fra sekstitallet
  - ✤ I bruk nå, men lite i embedded (blir mer og mer!)
- ✓ Sanntidsmekanismene er fra 60-70+ tallet
  - ✤ Semafor etc. i bruk siden da
- ✓ CSP etc. fra 70-tallet. Lite i bruk (ennå?)
- ✓ UML 2.0 er stort og uten aksjonsspråk
- ✓ Java -"synchronized" - det første allment brukte språket som har concurrency-tankegang innebygd..
  - ✤ ..og som er "livsfarlig" å bruke!
  - ✤ ..men som det går an å bygge CSP oppå!

Lange år

- ✓ OO er fra sekstitallet
  - ✤ I bruk nå, men lite i embedded (blir mer og mer!)
- ✓ Sanntidsmekanismene er fra 60-70 tallet
  - ✤ Semafor etc. i bruk siden da
- ✓ CSP etc. fra 70-tallet. Lite i bruk (ennå?)
- ✓ UML 2.0 er stort og uten aksjonsspråk
- ✓ Java -"synchronized" - det første allment brukte språket som hadde concurrency-tankegang innebygd..
  - ✤ ..og som er "livsfarlig" å bruke!
  - ✤ ..men som det går an å bygge CSP oppå!

Dette går da ikke fort!

- ✓ OO er fra sekstitallet
  - ✤ I bruk nå, men lite i embedded (blir mer og mer!)
- ✓ Sanntidsmekanismene er fra 60-70+ tallet
  - ✤ Semafor etc. i bruk siden da
- ✓ CSP etc. fra 70-tallet. Lite i bruk (ennå?)

men..

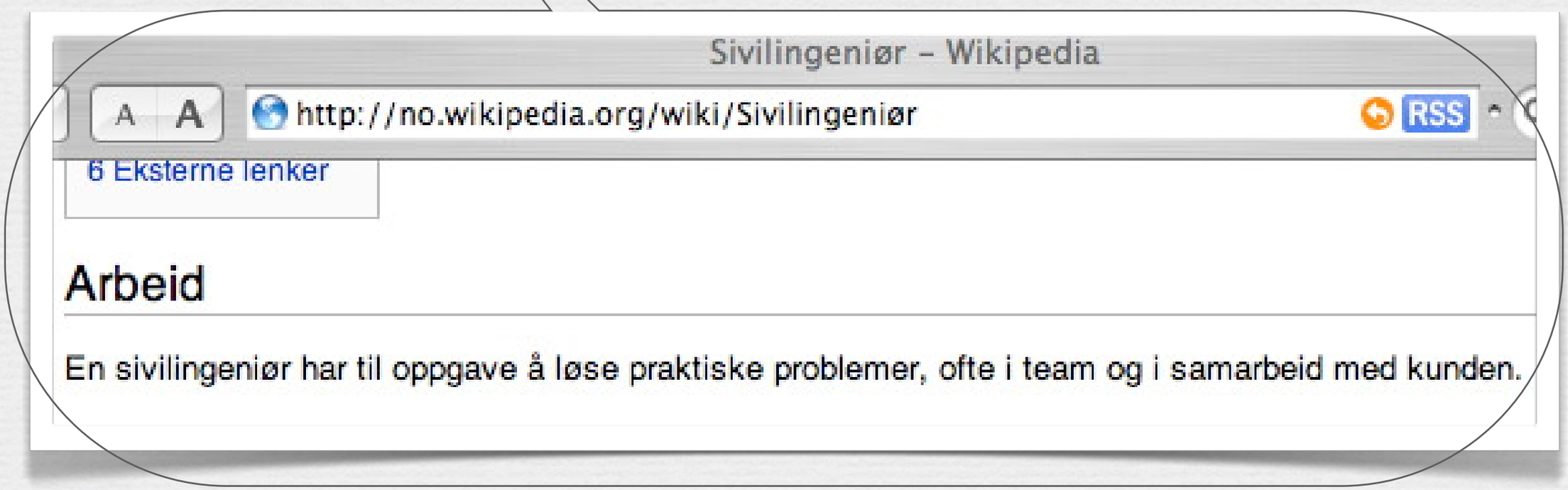
- ✓ UML 2.0 er stort og uten aksjonsspråk
- ✓ Java -"synchronized" - det første allment brukte språket som har concurrency-tankegang innebygd..
  - ✤ ..og som er "livsfarlig" å bruke!
  - ✤ ..men som det går an å bygge CSP oppå!

21 år etter occam (2009)

- ✓ **Go** fra Google plukket med seg litt occam!
- ✓ chan, go, select, eksplisitt typekonvertering, array-oppdeling, ikke pekerartitmetikk
- ✓ Schedulering av prosesser på prosessorer i Go som KRoCs kjøresystem for bla.a. occam-pi (fra tråd-basseng) (http://www.cs.kent.ac.uk/pubs/2009/2928/index.html)
- ✓ Ikke "sikkert" (no «parallel usage rules»)

Etter forelesningen ønsker jeg

- ✓ At dere skal huske at å jobbe med (s) embedded-systemer, er gøy
  - ➡ Sa jeg (sivile)?
- ✓ "En sivilingeniør har til oppgave å løse praktiske problemer, ofte i team og i samarbeid med kunden". (Wikipedia 2007)
- ✓ At dere skal huske å fortelle om det dere har lært i dette faget når dere havner i slike team!



Små embeddedsystemer i industrien

- ✓ Vil nok forholde seg til C en god stund til!
- ✓ Vil nok ha prosjekt- ledere og -deltakere som (bare) kan asynkrone system en god stund til!
- ✓ Ikke overta deres vertøykasse uten å legge synkrone kanaler og tette prosesser oppi!
- ✓ Lær av oss!

Utviklingsavdelingen på AFS

For små (AVR/ARM32) system i ANSI C

1. Begynte vi med et asynkront **SDL** kjøresystem (brukes mye!)
2. La så et synkront kanal-lag oppå **CHAN_CSP** (brukes!)
3. Har nå lagd et nakent synkront kjøresystem **ChanSched** (nå tatt i bruk!)

Kjøreystem

1988-90

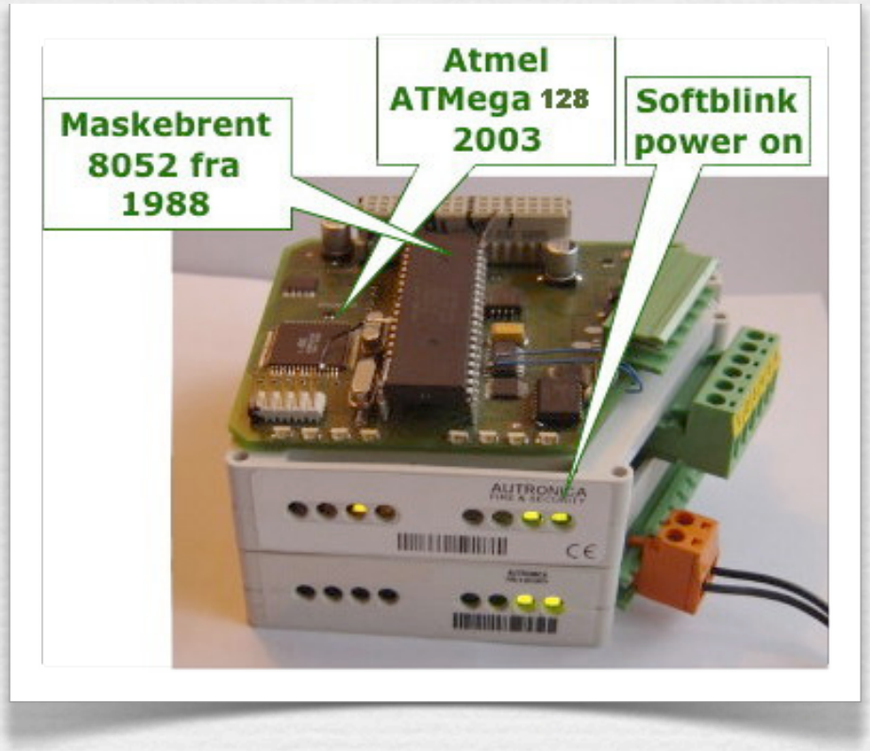
- ✓ PL/M-51
- ✓ Egenutviklet mailboksbasert kjøresystem
- ✓ Flere occam-system med transputer (90-95)
- ✓ Flere occam-system med SPoC og Texas DSP (95-2000)

2005-2008

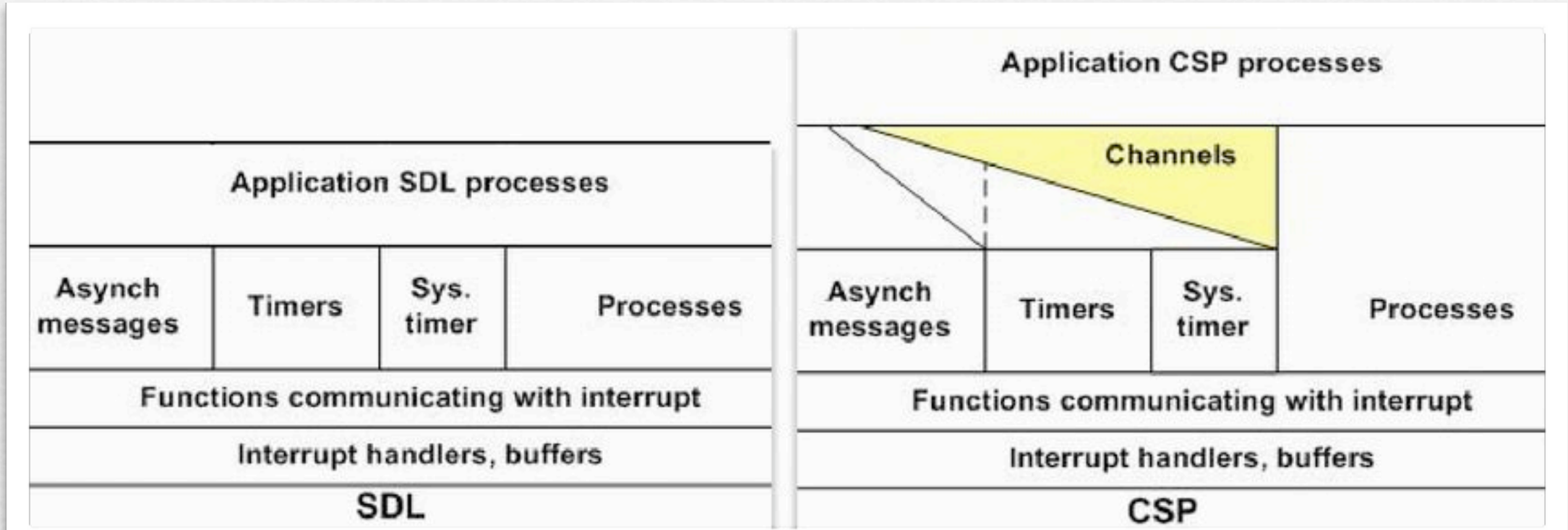
- ✓ Samme system i deler av ny brannsentral:
- ✓ I sløyfeprosessoren, som vi kaller "AutroLooper"

2009-...

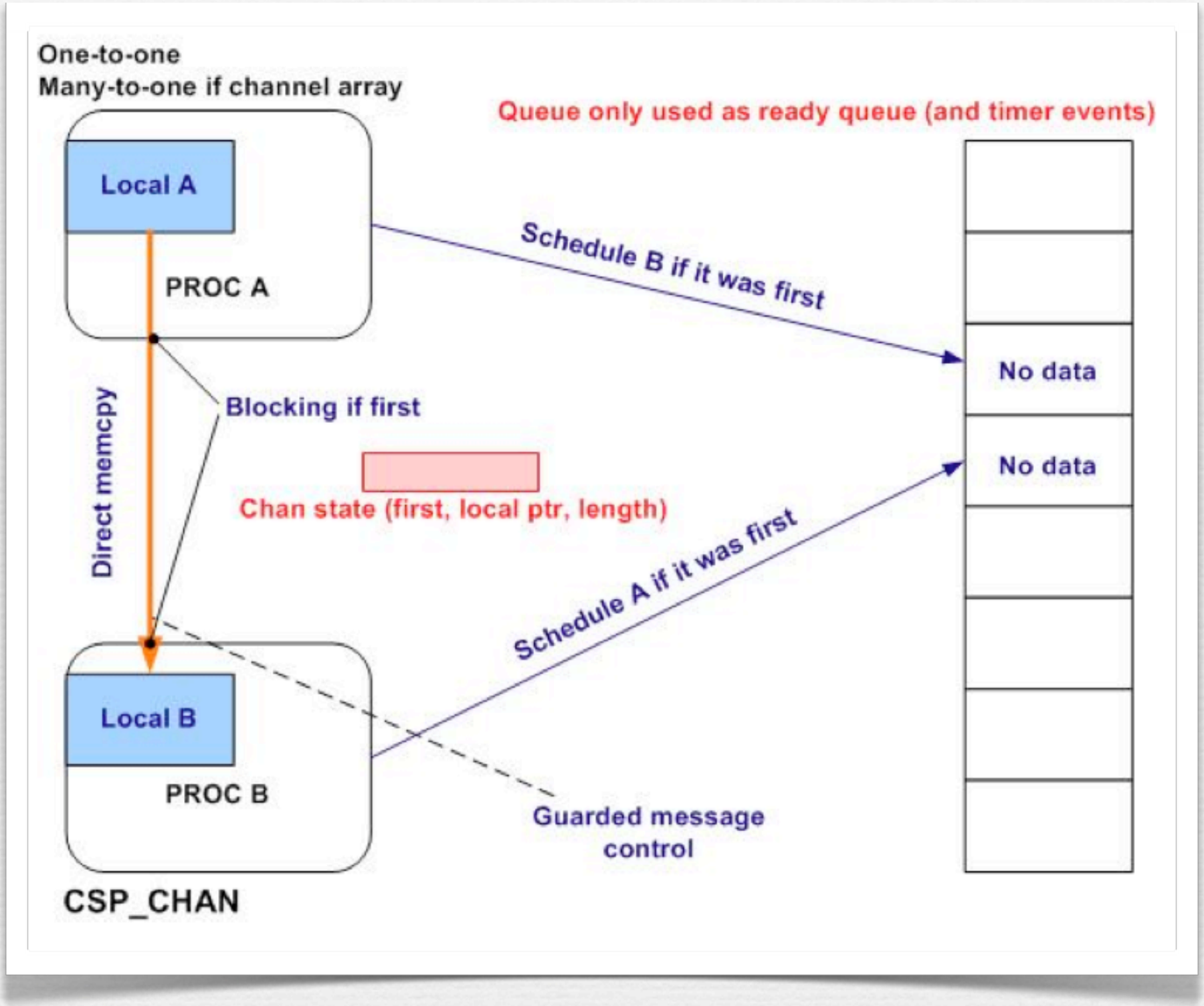
- ✓ I ny enhet for bla.a. cruiseskip



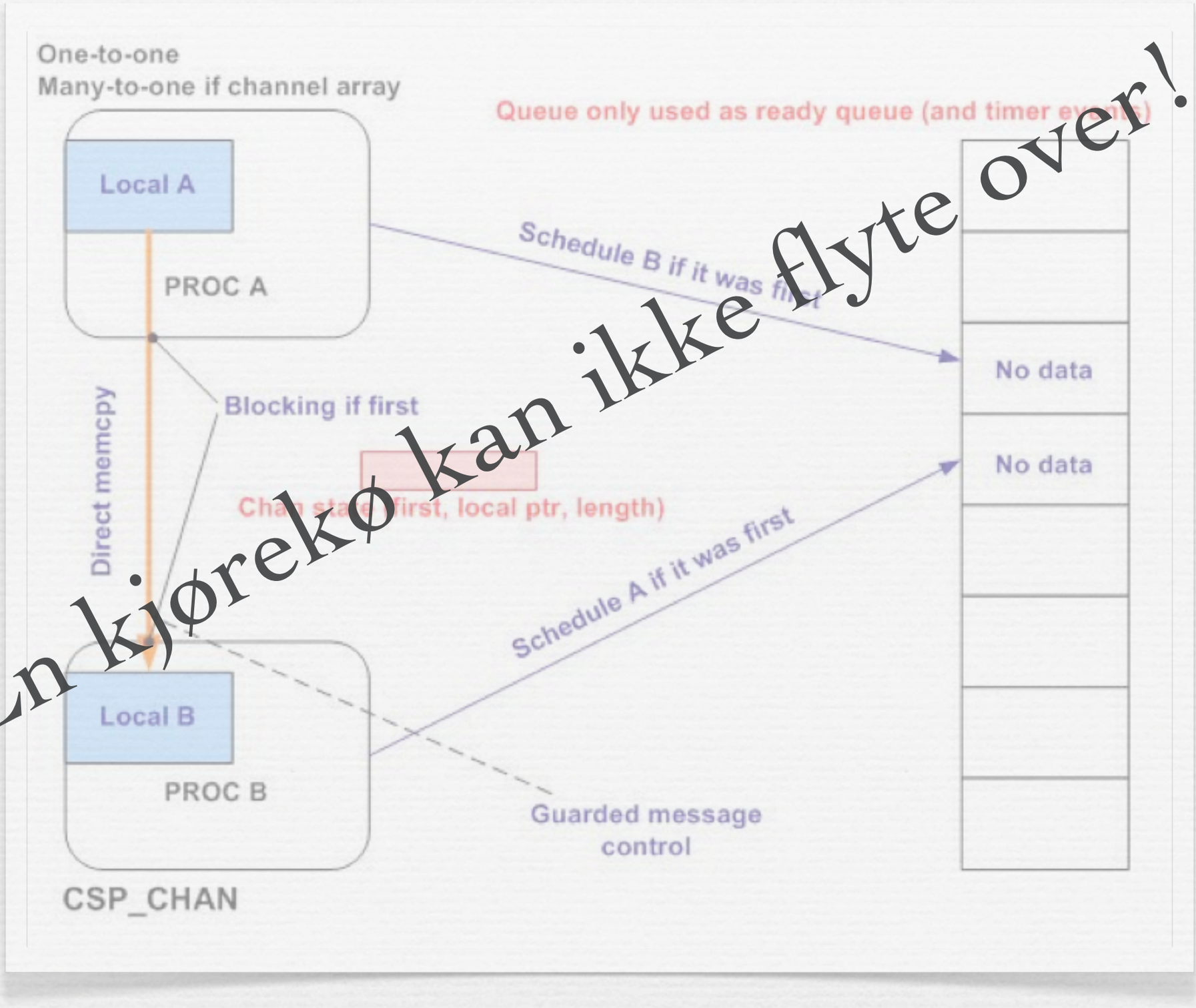
CHAN_CSP


Application SDL processes
Asynch messages
Timers
Sys. timer
Processes
Functions communicating with interrupt
Interrupt handlers, buffers
SDL
Application CSP processes
Channels
Asynch messages
Timers
Sys. timer
Processes
Functions communicating with interrupt
Interrupt handlers, buffers
CSP

Fra meldingskø til kjørekø



Fra meldingskø til kjørekø



En kjørekø kan ikke flyte over!

Meldingskøoverflyt

- ✓ Overflyt er naturlig
- ✓ Det skal håndteres på applikasjonsnivå
- ✓ Meldinger skal kastes bevisst
- ✓ Meldingsflyt skal håndteres bevisst

Meldingsko

- ✓ Overflyt er naturlig
  Men ikke mellom programmer
- ✓ Det skal håndteres på applikasjonsnivå
  Men ikke av operativsystemet
- ✓ Meldinger skal kastes bevisst
  Men kanskje ikke alle
- ✓ Meldingsflyt skal håndteres bevisst
  Så blokkering «ved halvgjort» er flott!

"Blokkerende" prosesser

- ✓ Et system som består av blokkerende kommunikasjon får ikke gjort mer enn et system som ikke blokkerer
- ✓ Heller ikke mindre (som nesten alle tror)
- ✓ Jo flere prosesser og dermed "parallell slakkhet" - desto mer greier et synkront system
- ✓ De fleste funksjoner og prosesser er aktive kort tid om gangen - og passiv venting er hovedbeskjeftigelsen ("run to completion")
- ✓ OBS! Synkrone kanaler er ikke implementert i noen busy-poll mekanisme, og polles heller ikke av kjøresystemet. Derfor har du null overhead. De vurderes bare når partene ankommer

Blokkerende prosesser

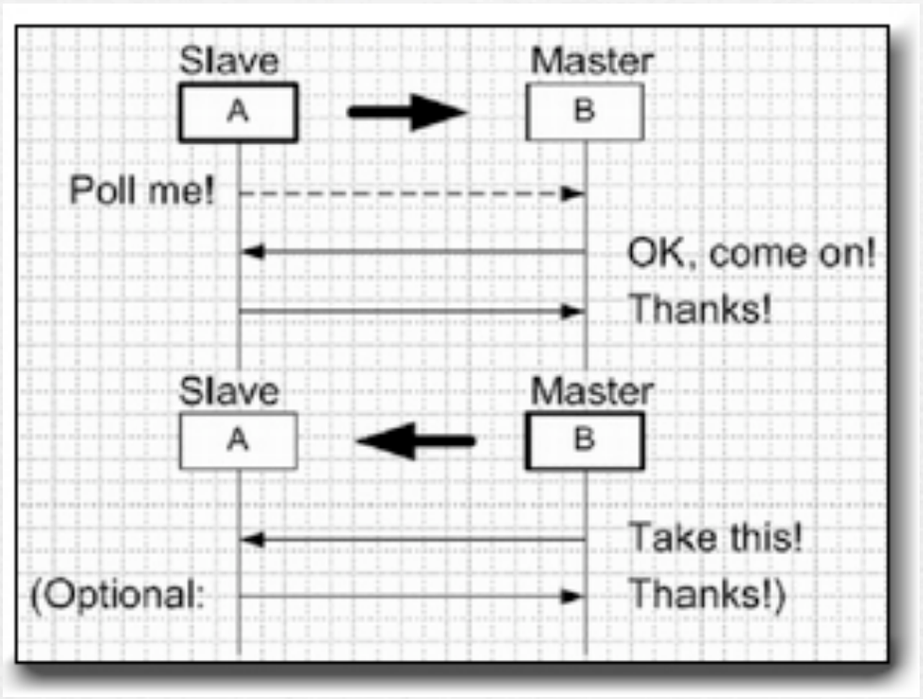
- ✓ Et system som består av blokkerende kommunikasjon får ikke gjort mer enn et system som ikke blokkerer
- ✓ Heller ikke mindre (som nesten alle tror)
- ✓ Jo flere prosesser og dermed "parallell slakkhet" - desto mer greier et synkront system
- ✓ De fleste funksjoner og prosesser er aktive kort tid om gangen - og passiv venting er hovedbeskjeftigelsen ("run to completion")

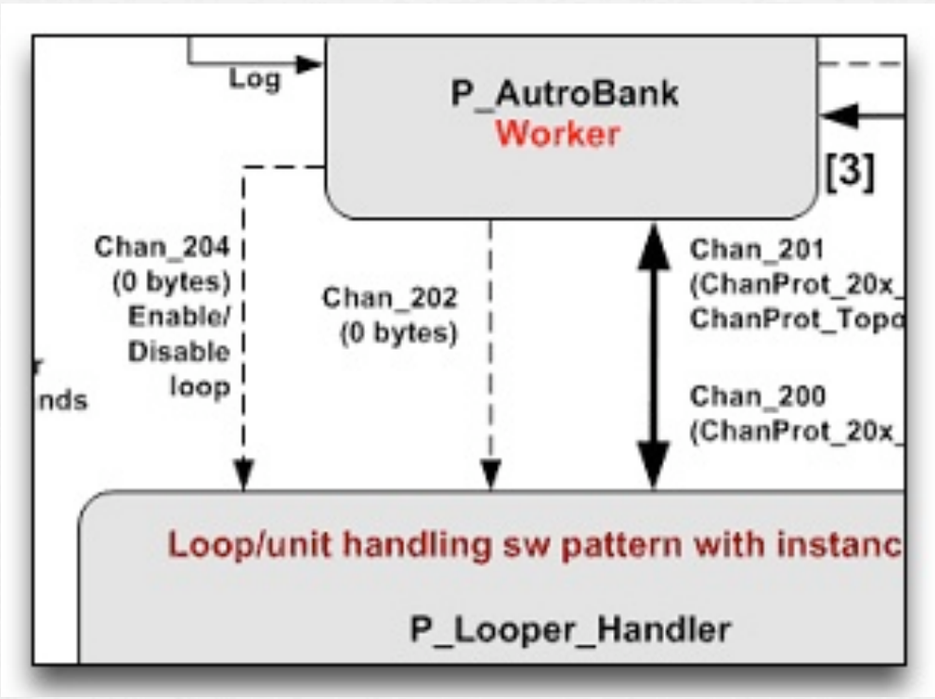
ChanSched

- ✓ OBS! Synkrone kanaler er ikke implementert i noen busy-poll mekanisme, og polles heller ikke av kjøresystemet. Derfor har du null overhead. De vurderes bare når partene ankommer

Data-komm som alltid funker

- ✓ Mønster som garanterer mot vranglås





- ✓ «Knock-come» verifisert i Promela/Spin

http://oyvteig.blogspot.com/2009/03/009-knock-come-deadlock-free-pattern.html

Seriell OO modellering

- ✓ Her benytter vi UML
- ✓ Vi benytter Telelogic Rhapsody
  - ✤ Klassediagrammer
  - ✤ Tilstandsmaskiner
  - ✤ Kodegenerering av ramme og skruktur
- ✓ Håndskrevet:
  - ✤ Aksjonskoden
  - ✤ Lim mot resten av systemet
  - ✤ Prosesser via operativsystem så langt

«modellering»

- ✓ ..er nåtidas «ekspertsystem»-ord
- ✓ Både assembler, C, Java, CSP, occam, Go - og UML er abstraksjoner
- ✓ Altså «modeller»
- ✓ Men i varierende grad er de ovenfor «formelle modeller» - dvs. kan være input til et verktøy som gjør verifisering for korrekthet (null til..?)
- ✓ Men disse er: CSP/FDR2, Promela/Spin, LTSA etc. Rask utvikling, følg med!

Resten av tida skal jeg vise dere noe jeg har lært i de siste årene

New ALT for Application Timers and Synchronisation Point Scheduling

(Two excerpts from a small channel based scheduler)

«ChanSched»

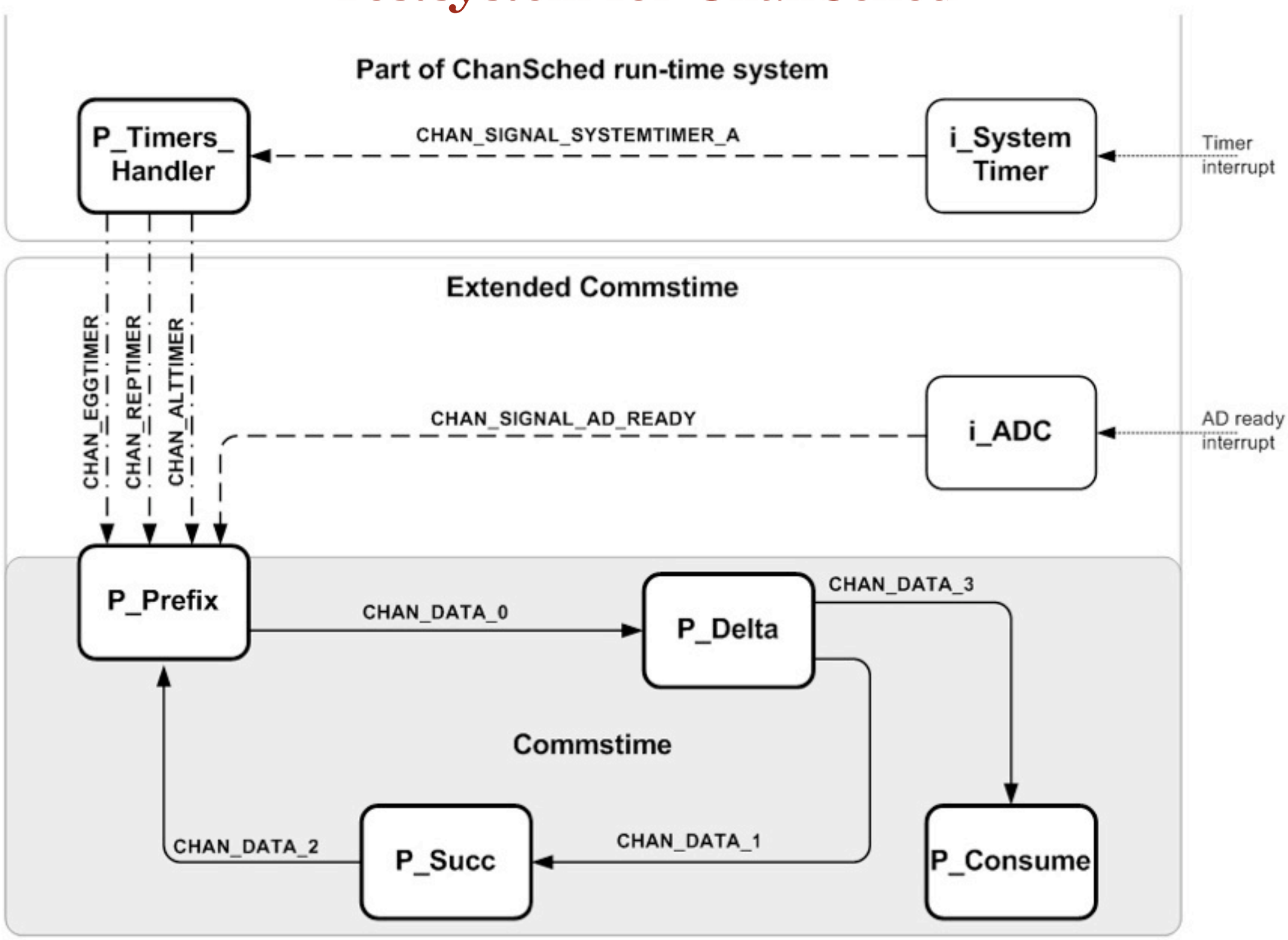
Øyvind TEIG and Per Johan VANNEBO
Autronica Fire and Security①, Trondheim, Norway

{per.vannebo, oyvind.teig}@autronicafire.no

① A UTC Fire & Security Company. NO-7483 Trondheim, Norway
http://www.autronicafire.no

At *Communicating Process Architectures 2009* (CPA-2009), 1-4 November, 2009, Eindhoven, the Netherlands
http://www.wotug.org/cpa2009/

Testsystem for ChanSched



Prosess/dataflyt for utvidet «commstime»

```
void P_Standard_CHAN_CSP (void)
{
  CP_a CP = (CP_a)g_ThisExtPtr; // Application
  switch (CP->State)            // and
                                // communication
                                // state
  {
    case ST_INIT: {/*Init*/ break;}
    case ST_IN:
    {
      CHAN_IN(G_CHAN_IN,CP->Chan_val1);
      CP->State = ST_APPL1;
      break;
    }
    case ST_APPL1:
    {
      // Process val1
      CP->State = ST_OUT;
      break;
    }
    case ST_OUT:
    {
      CHAN_OUT(G_CHAN_OUT,CP->Chan_val1);
      CP->State = ST_IN;
      break;
    }
  }
}
```

```
void P_libcsp2 (Channel *in, Channel *out)
{
  int val3;
  for(;;)
  {
    ChanInInt (in, &val3);
    // Process val3
    ChanOutInt (out, val3);
  }
}
```

```
void P_Extended_ChanSched (void)
{
  CP_a CP = (CP_a)g_ThisExtPtr; // Application
  // Init here                  // state only
  while (TRUE)
  {
    switch (CP->State)
    {
      case ST_MAIN:
      {
        CHAN_IN(G_CHAN_IN,CP->Chan_val2);

        // Process val2

        CHAN_OUT(G_CHAN_OUT,CP->Chan_val2);
        CP->State = ST_MAIN; // option1
        break;
      }
    }
  }
}
```

```
PROC P_occam (CHAN OF INT in, out)

  WHILE TRUE
  INT val4:
    SEQ
      in ? val4
      -- Process val4
      out ! val4

:
```

«En scheduler/ kjøresystem/ operativsystem/ er er ikke så usynlig som jeg trodde»

```
void P_Standard_CHAN_CSP (void)
{
  CP_a CP = (CP_a)g_ThisExtPtr; // Application
  switch (CP->State)            // and
                                // communication
                                // state
  {
    case ST_INIT: {/*Init*/ break;}
    case ST_IN:
    {
      CHAN_IN(G_CHAN_IN,CP->Chan_val1);
      CP->State = ST_APPL1;
      break;
    }
    case ST_APPL1:
    {
      // Process val1
      CP->State = ST_OUT;
      break;
    }
    case ST_OUT:
    {
      CHAN_OUT(G_CHAN_OUT,CP->Chan_val1);
      CP->State = ST_IN;
      break;
    }

  }
}

void P_libcsp2 (Channel *in, Channel *out)
{
  int val3;
  for(;;)
  {
    ChanInInt (in, &val3);
    // Process val3
    ChanOutInt (out, val3);
  }
}
```

```
void P_Extended_ChanSched (void)
{
  CP_a CP = (CP_a)g_ThisExtPtr; // Application
  // Init here                  // state only
  while (TRUE)
  {
    switch (CP->State)
    {
      case ST_MAIN:
      {
        CHAN_IN(G_CHAN_IN,CP->Chan_val2);
        // Process val2
        CHAN_OUT(G_CHAN_OUT,CP->Chan_val2);
        CP->State = ST_MAIN; // option1
        break;
      }
    }
  }
}

PROC P_occam (CHAN OF INT in, out)

  WHILE TRUE
  INT val4:
    SEQ
      in ? val4
      -- Process val4
      out ! val4

:
```

En typisk ChanSched prosess

```
01 Void P_Prefix (void)                          // extended “Prefix”
02 {
03   Prefix_CP_a CP = (Prefix_CP_a)g_CP; // get process Context from Scheduler
04   PROCTOR_PREFIX()                            // jump table (see Section 2)
05   ...  some initialisation
06   SET_EGGTIMER (CHAN_EGGTIMER, LED_Timeout_Tick);
07   SET_REPTIMER (CHAN_REPTIMER, ADC_TIME_TICKS);
08   CHAN_OUT (CHAN_DATA_0, Data_0);  // first output
09   while (TRUE)
10   {
11     ALT();                                    // this is the needed ”PRI_ALT”
12       ALT_EGGREPTIMER_IN (CHAN_EGGTIMER);
13       ALT_EGGREPTIMER_IN (CHAN_REPTIMER);
14       ALT_SIGNAL_CHAN_IN (CHAN_SIGNAL_AD_READY);
15       ALT_CHAN_IN        (CHAN_DATA_2, Data_2);
16       ALT_ALTTIMER_IN    (CHAN_ALTTIMER, TIME_TICKS_100_MSECS);
17     ALT_END();
18     switch (g_ThisChannelId)
19     {
20       ...  process the guard that has been taken, e.g. CHAN_DATA_2
21       CHAN_OUT (CHAN_DATA_0, Data_0);
22     };
23   }
24 }
```

"Fjernet" siden i fjor

```
01 Void P_Prefix (void)                          // extended “Prefix”
02 {
03   Prefix_CP_a CP = (Prefix_CP_a)g_CP; // get process Context from Scheduler
04   PROCTOR_PREFIX()                            // jump table (see Section 2)
05   ...  some initialisation
06   SET_EGGTIMER (CHAN_EGGTIMER, CP->LED_Timeout_Tick);
07   SET_REPTIMER (CHAN_REPTIMER, ADC_TIME_TICKS);
08   CHAN_OUT (CHAN_DATA_0, &CP->Data_0, sizeof(CP->Data_0));  // first output
09   while (TRUE)
10   {
11     ALT();                                    // this is the needed ”PRI_ALT”
12       ALT_EGGREPTIMER_IN (CHAN_EGGTIMER);
13       ALT_EGGREPTIMER_IN (CHAN_REPTIMER);
14       ALT_SIGNAL_CHAN_IN (CHAN_SIGNAL_AD_READY);
15       ALT_CHAN_IN        (CHAN_DATA_2, &CP->Data_2, sizeof (CP->Data_2));
16       ALT_ALTTIMER_IN    (CHAN_ALTTIMER, TIME_TICKS_100_MSECS);
17     ALT_END();
18     switch (g_ThisChannelId)
19     {
20       ...  process the guard that has been taken, e.g. CHAN_DATA_2
21       CHAN_OUT (CHAN_DATA_0, &CP->Data_0, sizeof (CP->Data_0));
22     };
23   }
24 }
```

Nesten usynlig hopp tilbake til der man sist blokkerte

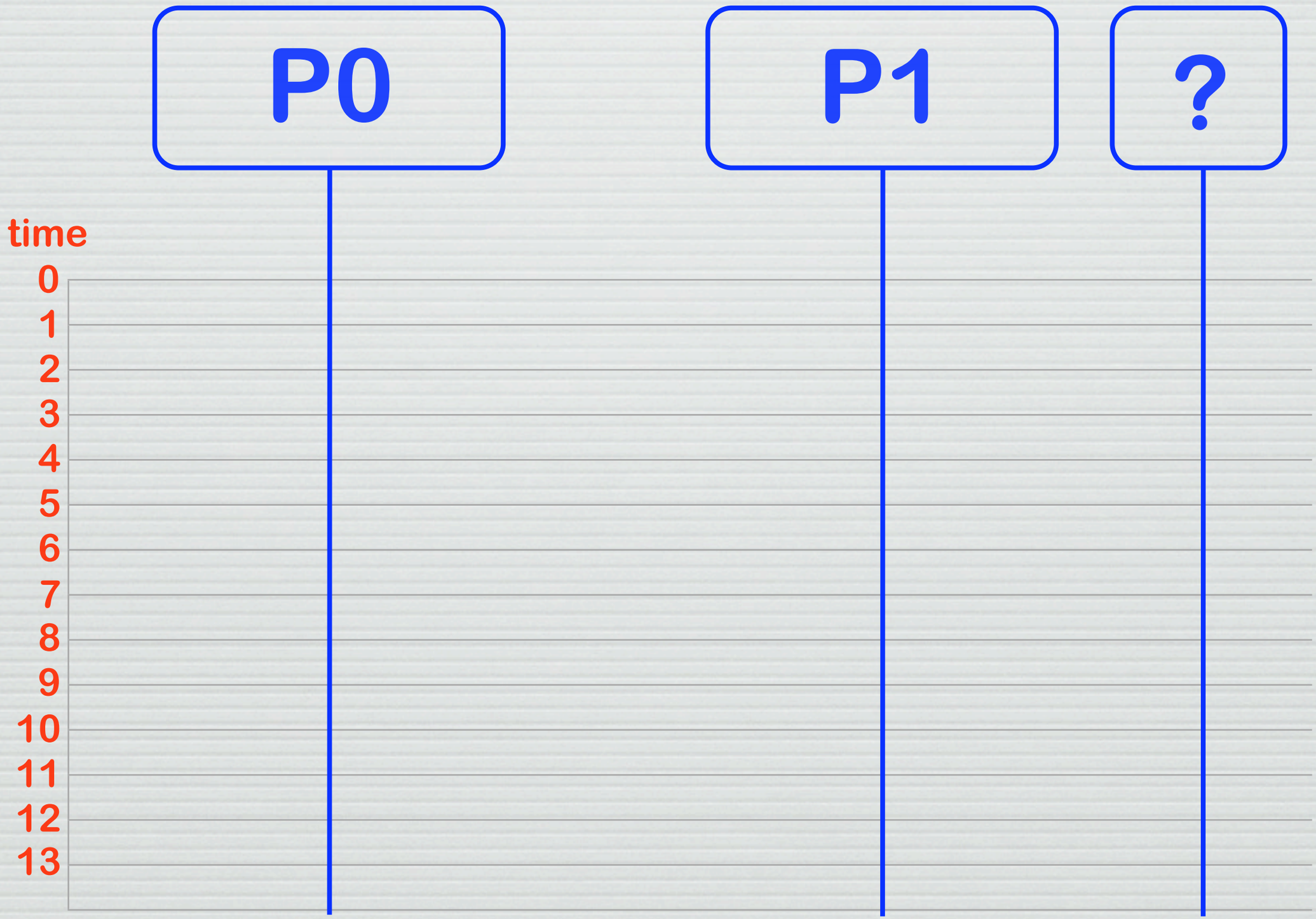
Se http://www.teigfam.net/oyvind/pub/pub_details.html#NewALT

```
64 #define SCHEDULE_AT goto

66 #define CAT(a,b,c,d,e) a##b##c##d##e // Concatenate to f.ex. "SYNCH_8_L"

68 #define SYNCH_LABEL(a,b,c,d,e) CAT(a,b,c,d,e) // Label for Proctor-table

70 #define PROC_DESCHEDULE_AND_LABEL() \
71         CP->LineNo = __LINE__; \
72         return; \
73         SYNCH_LABEL(SYNCH,_,__LINE__,_,L):

75 #define CHAN_OUT(chan,dataptr,len) \
76         if (ChanSched_ChanOut(chan,dataptr,len) == FALSE) \
77         { \
78             PROC_DESCHEDULE_AND_LABEL(); \
79         } \
80         g_ThisAltTaken = FALSE
```

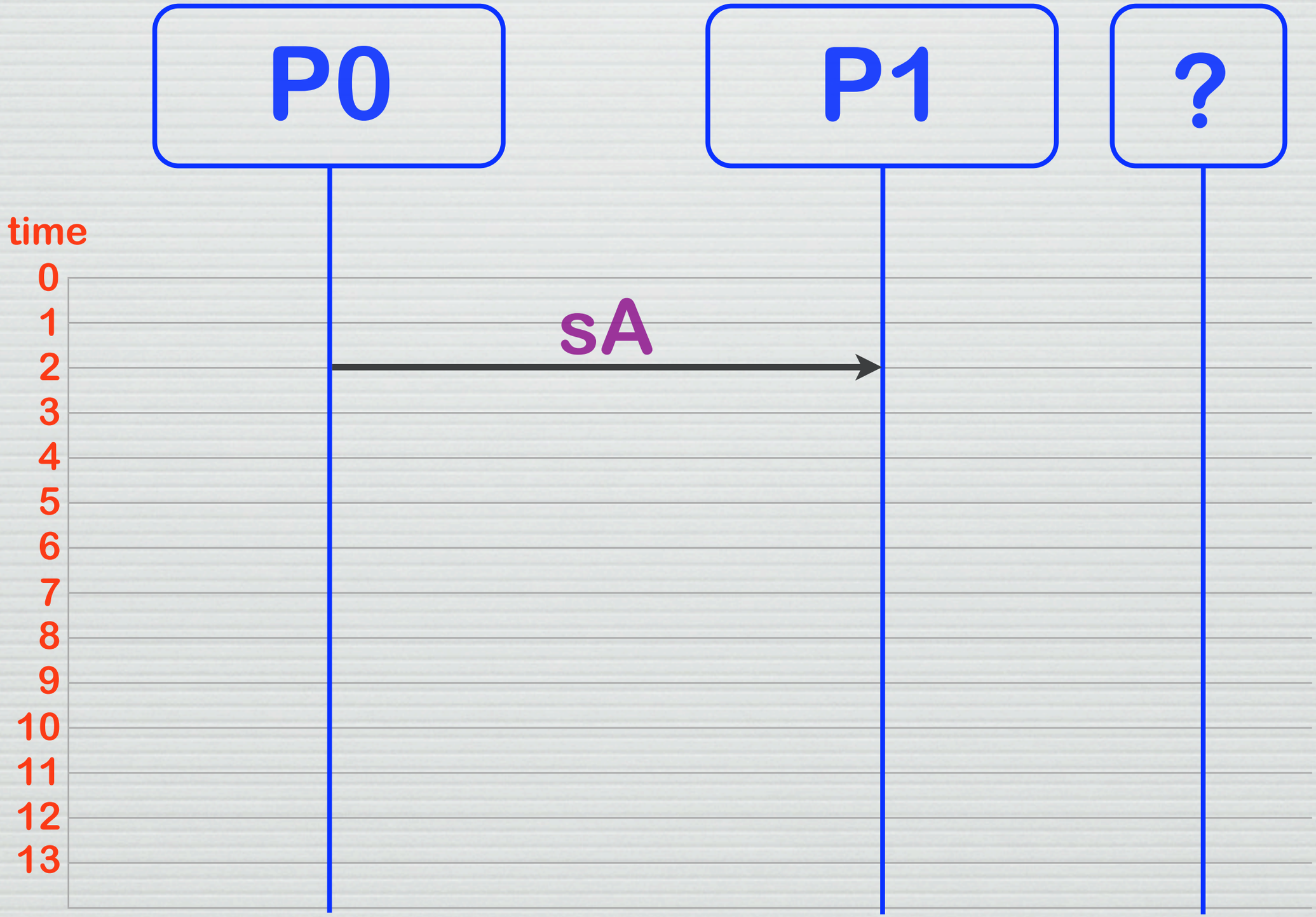
```
81 #define PROCTOR_PREFIX()\
82         switch (CP->LineNo)\
83         {\
84             case 0: break;\
85             case 8: SCHEDULE_AT SYNCH_8_L;\
86             case 17: SCHEDULE_AT SYNCH_17_L;\
87             case 21: SCHEDULE_AT SYNCH_21_L;\
88             DEFAULT_EXIT\
89         }
```

Bare denne er synlig, ref to sider siden

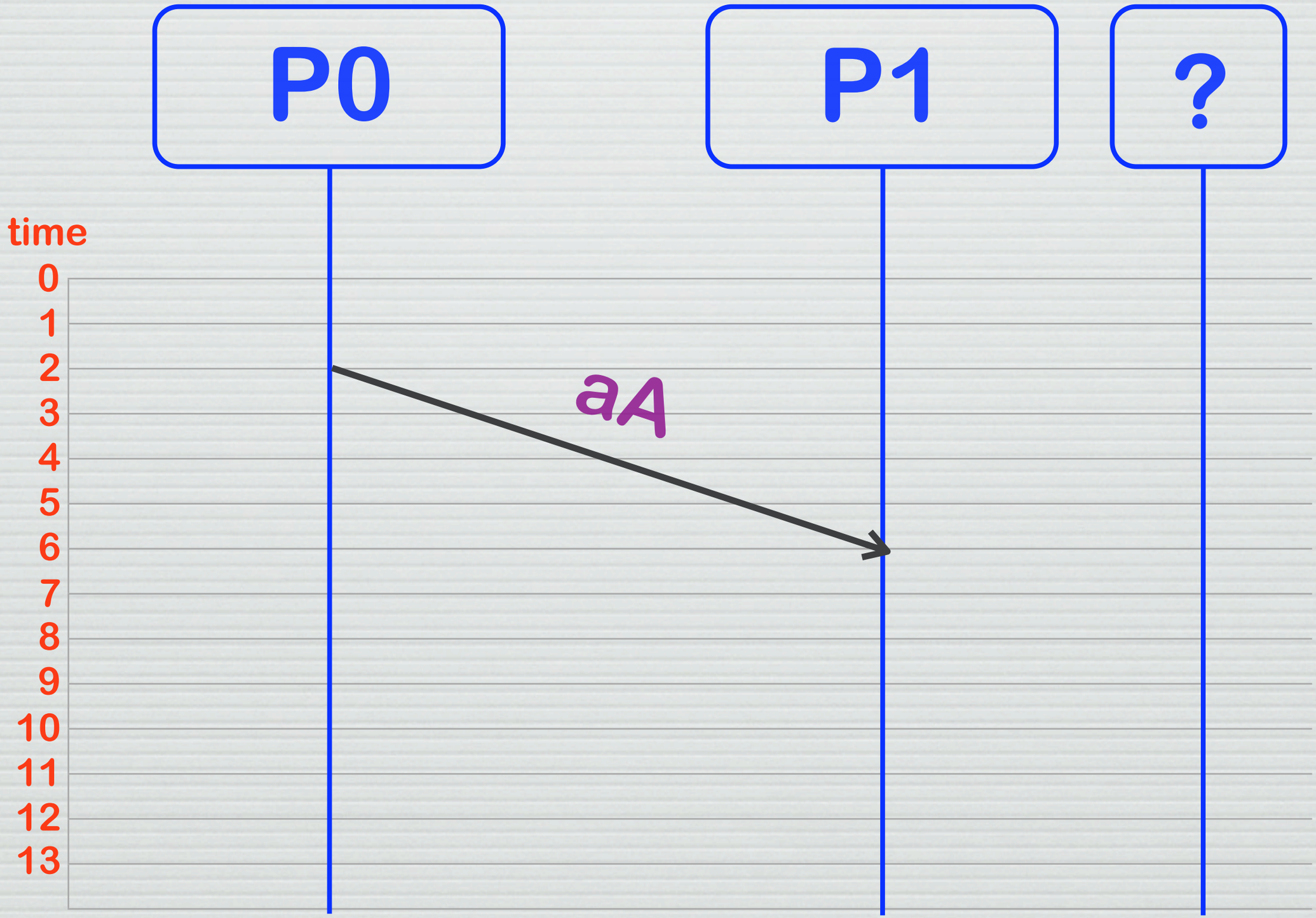
Genereres av verktøy og ligger i egen .h fil



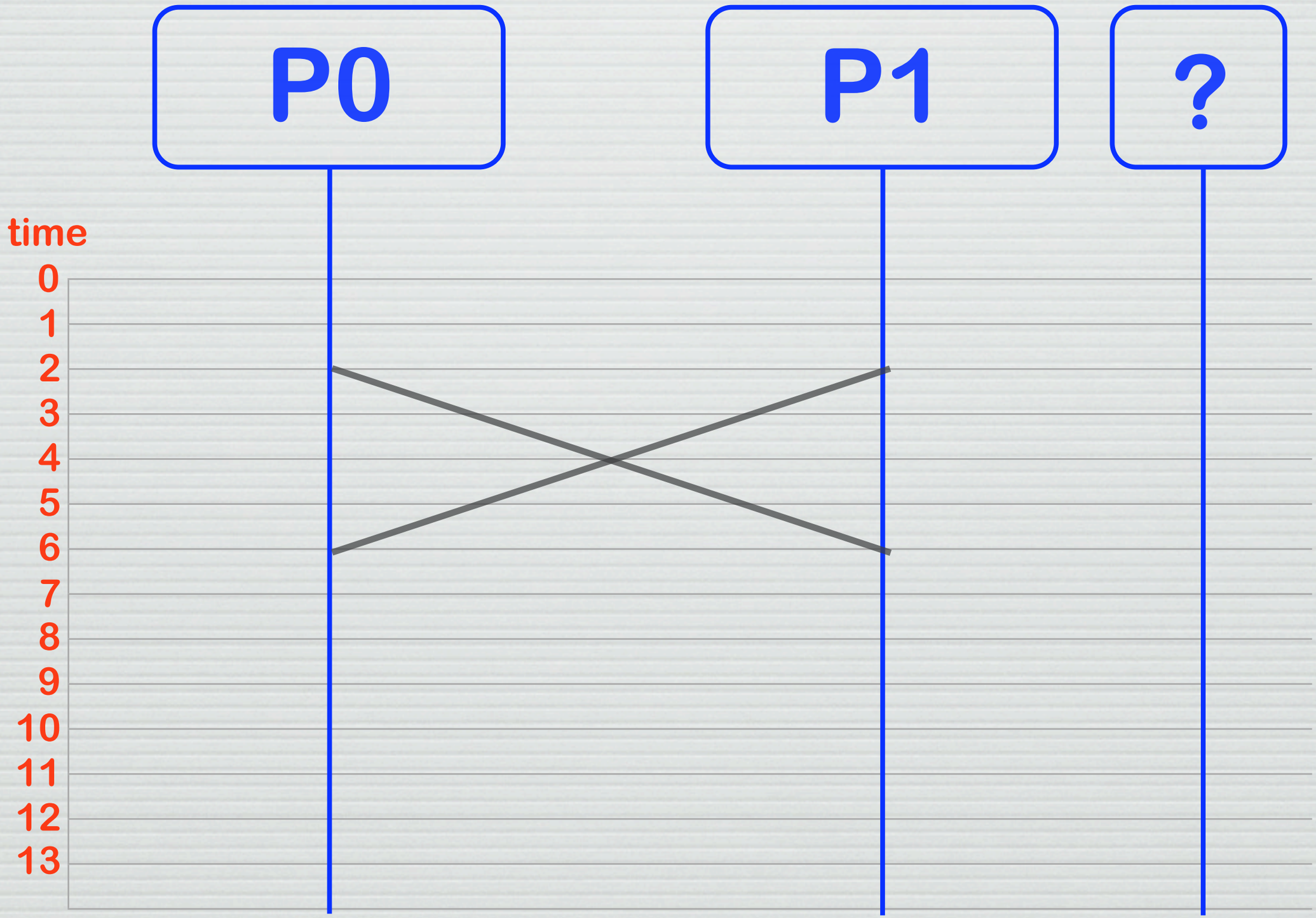
Some patterns



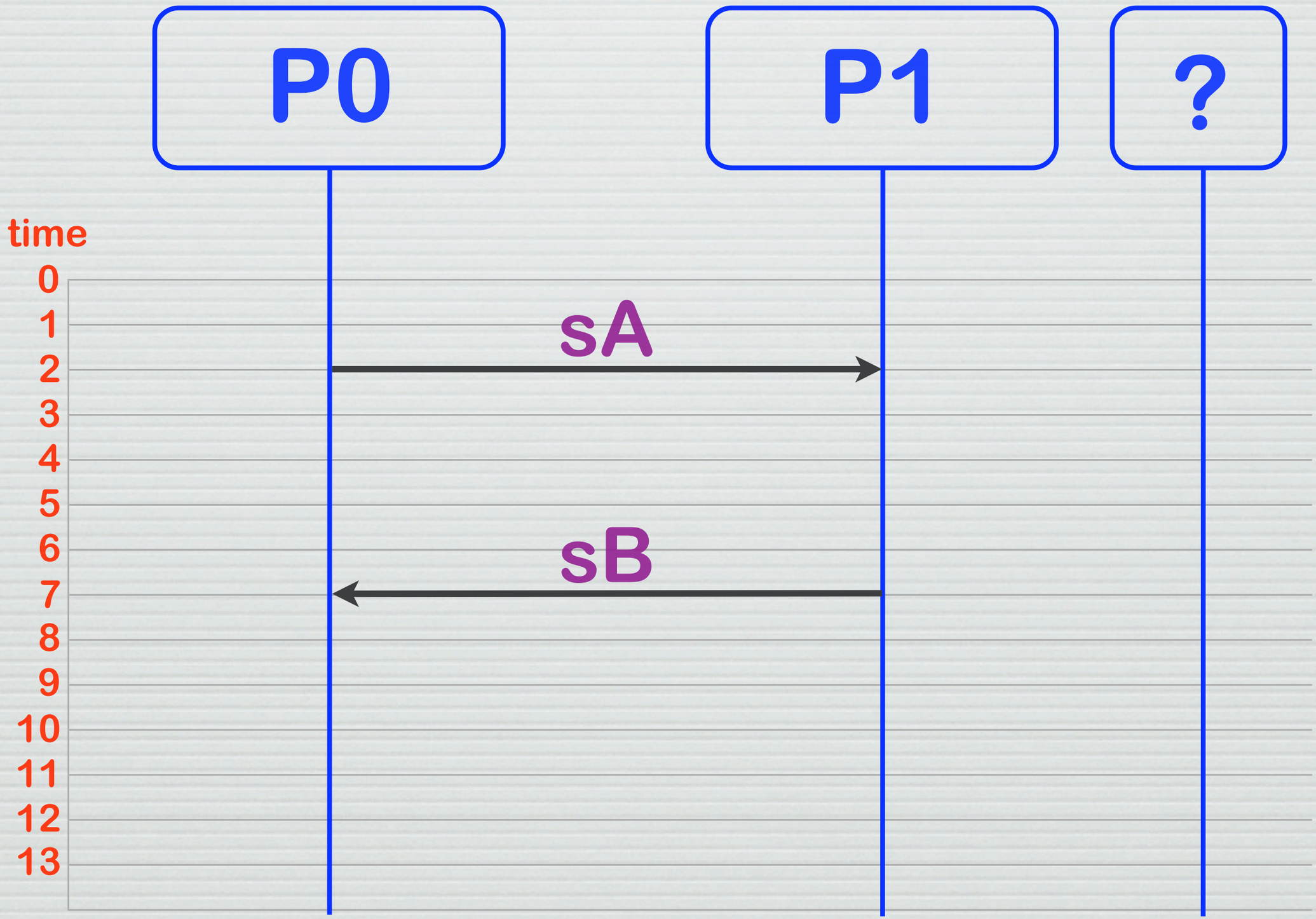
Event without reply



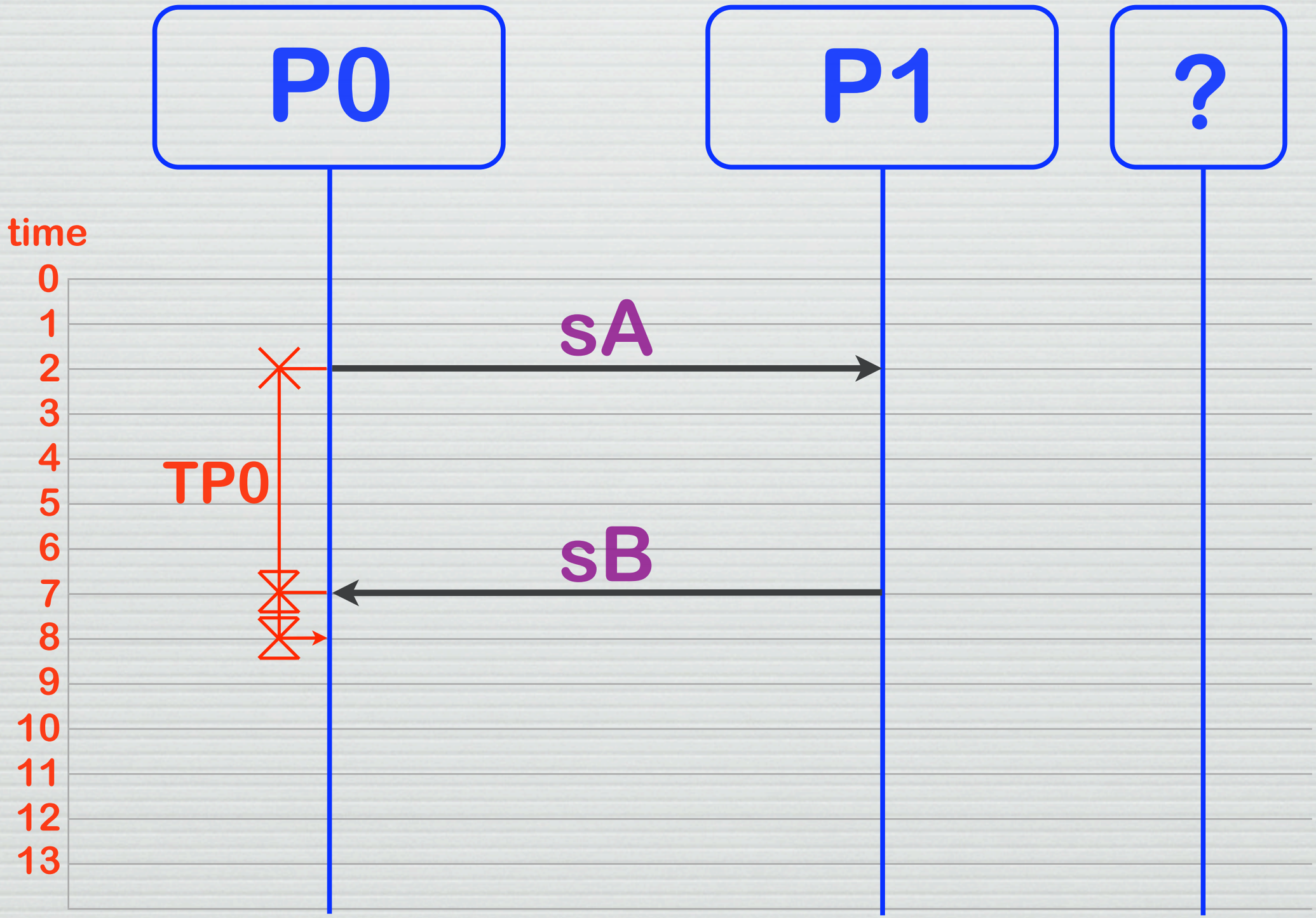
Asynchronous with data



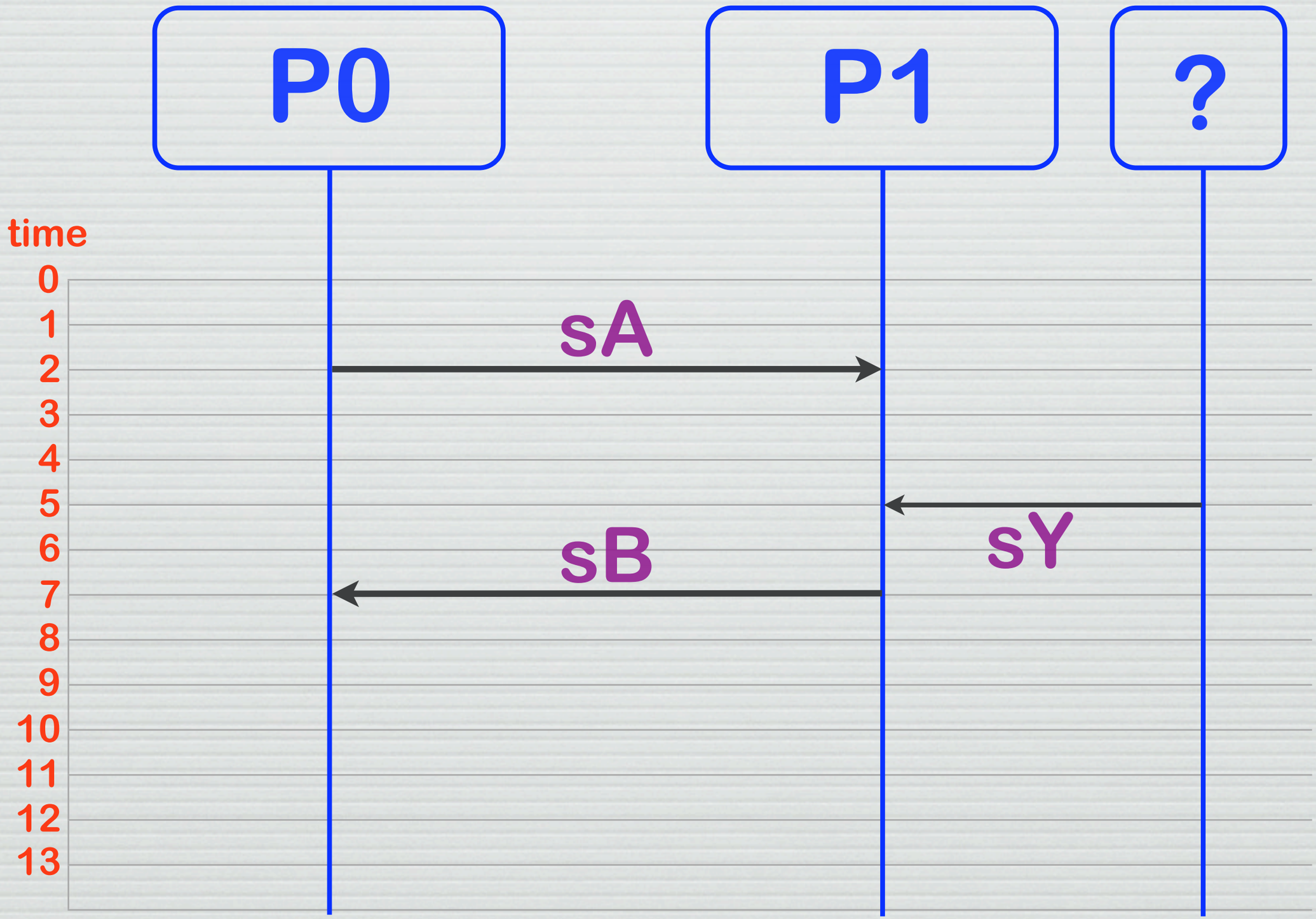
Not here



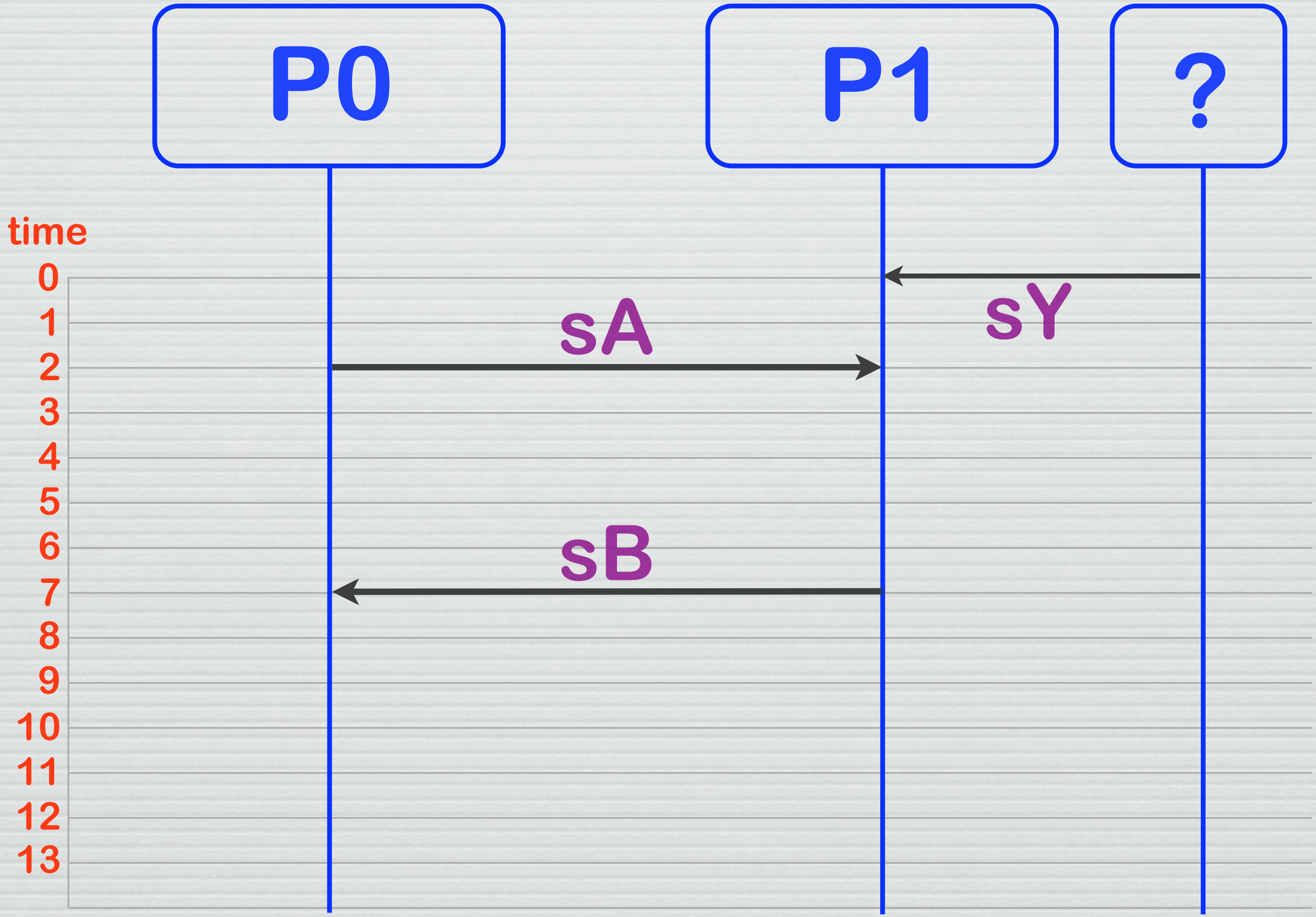
Event with reply



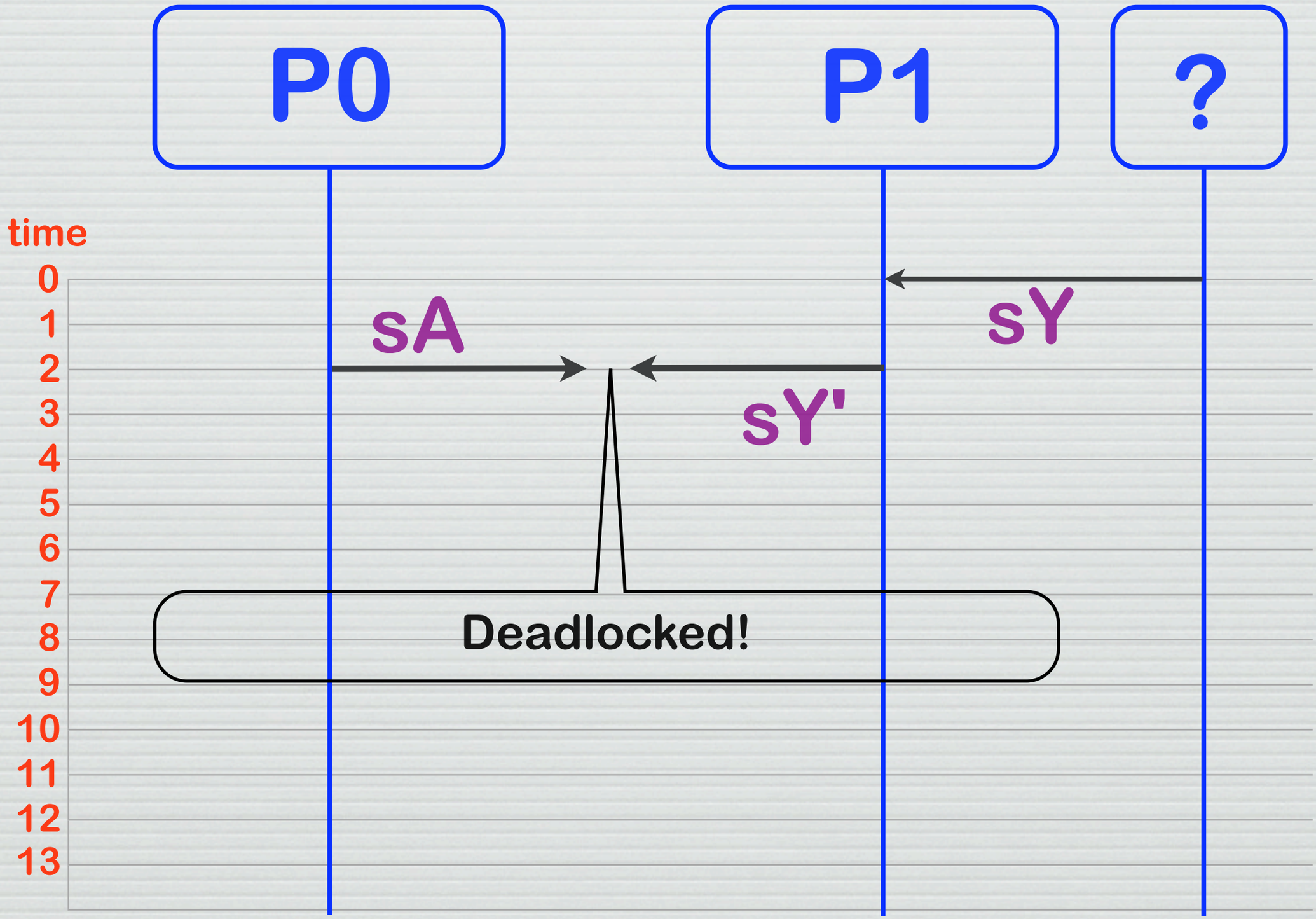
Event with wrong timeout for reply



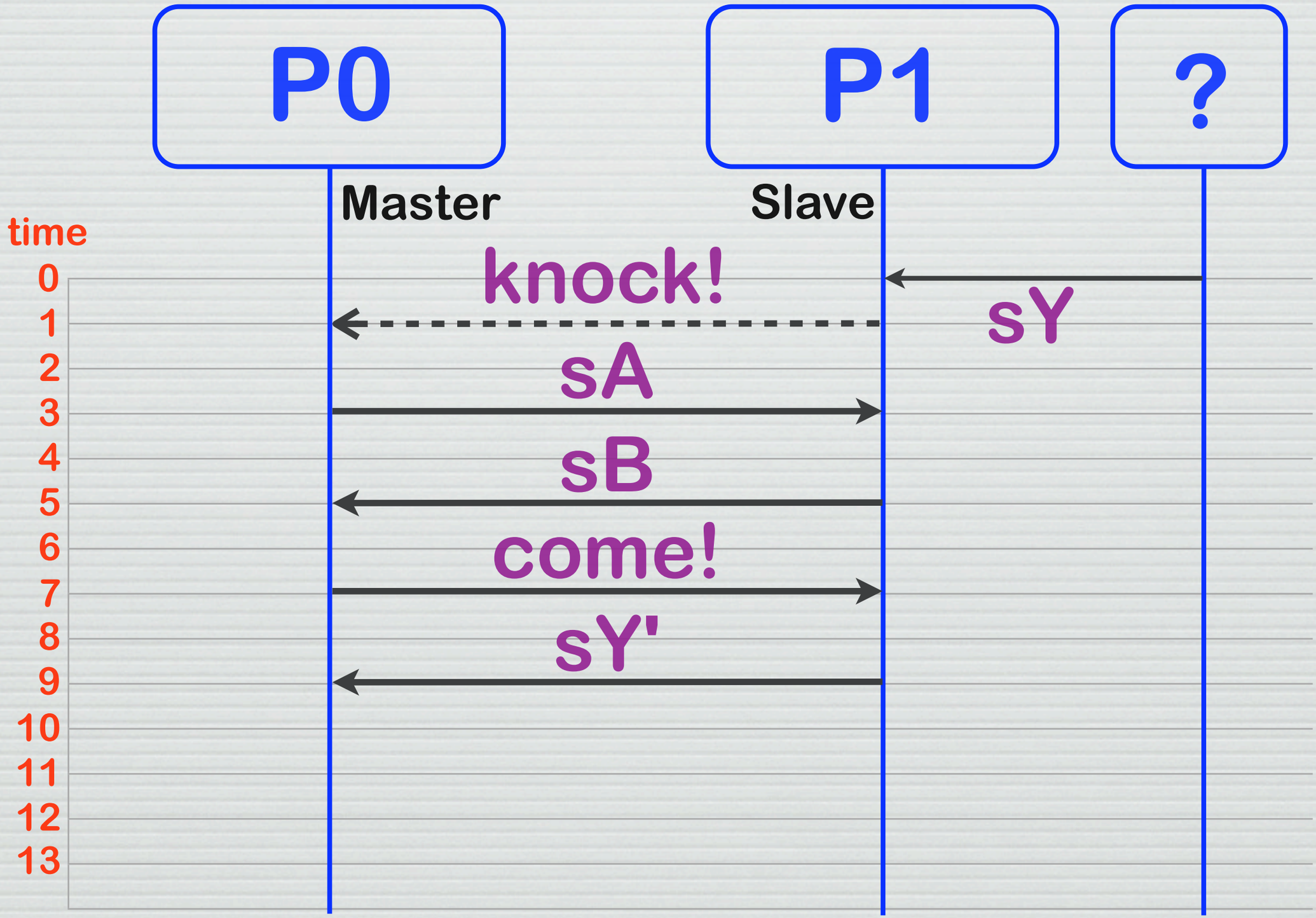
Tell P0 if reply to SA or y



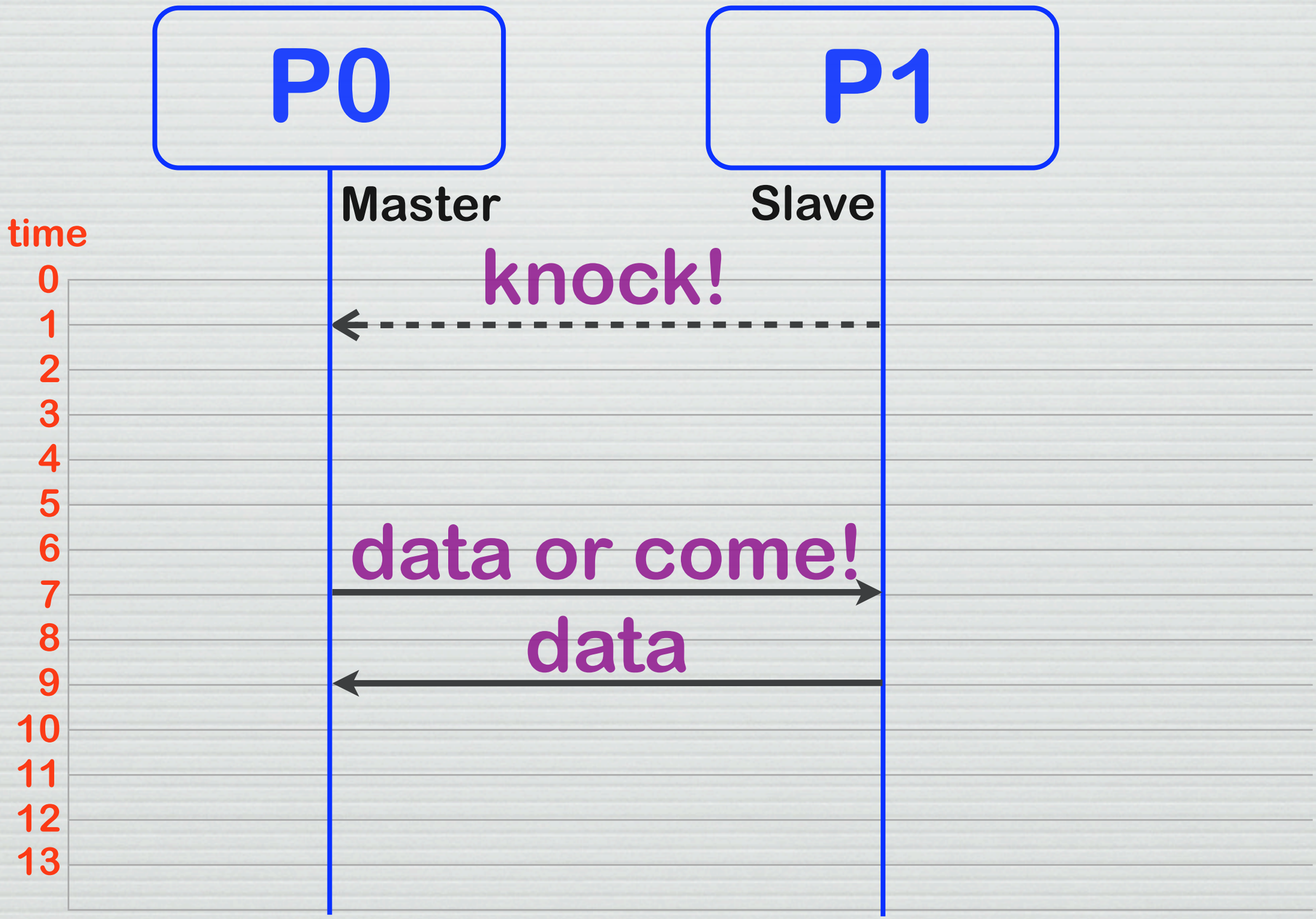
sY is problematic at this time



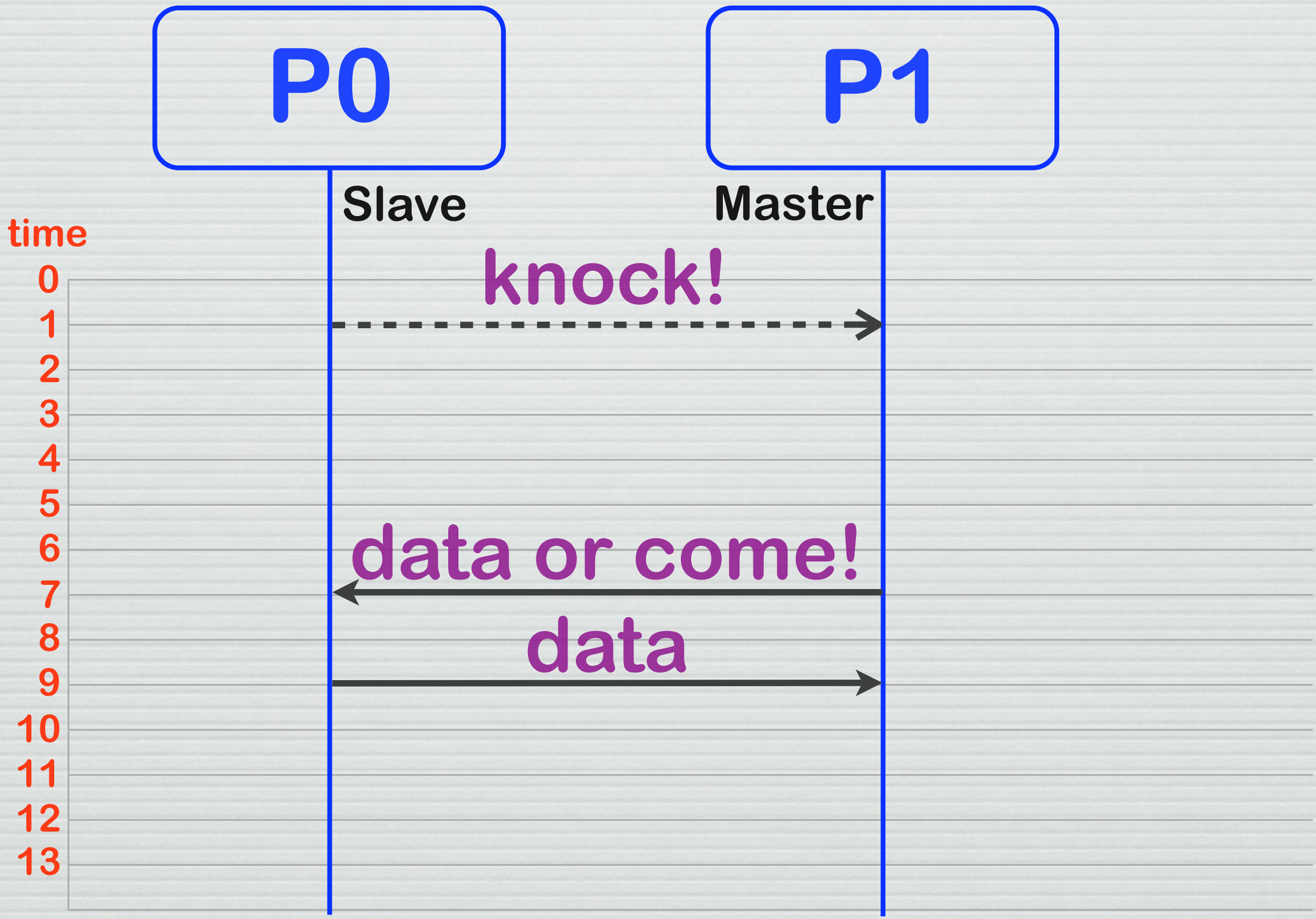
sY is problematic at this time



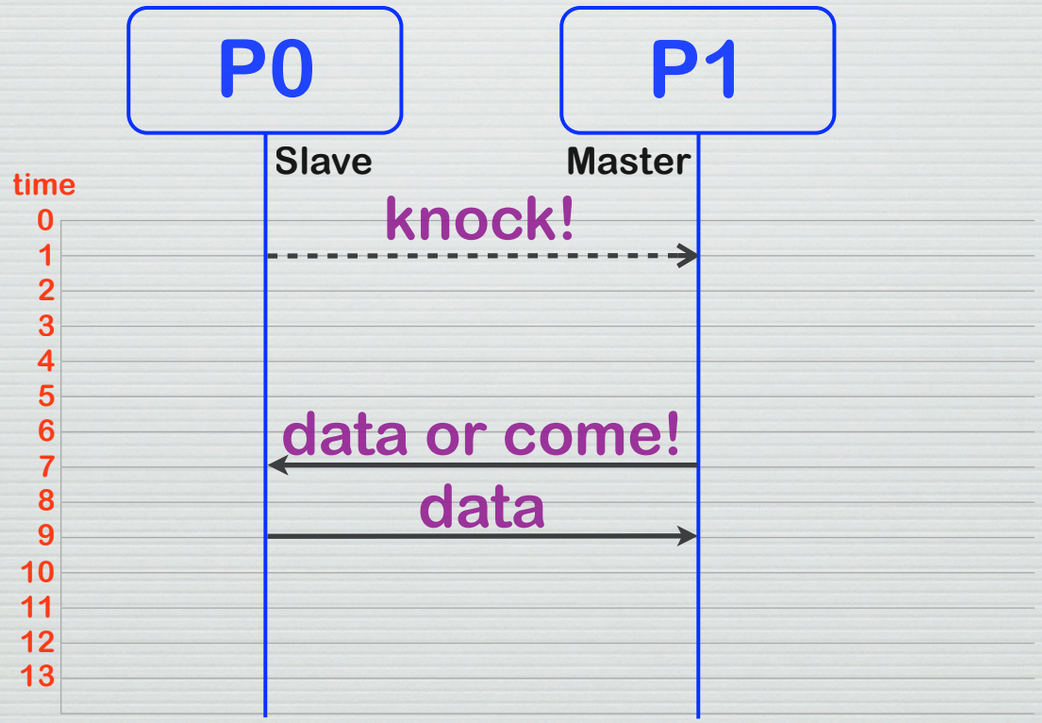
Get rid of sY with knock-come



Master - Slave

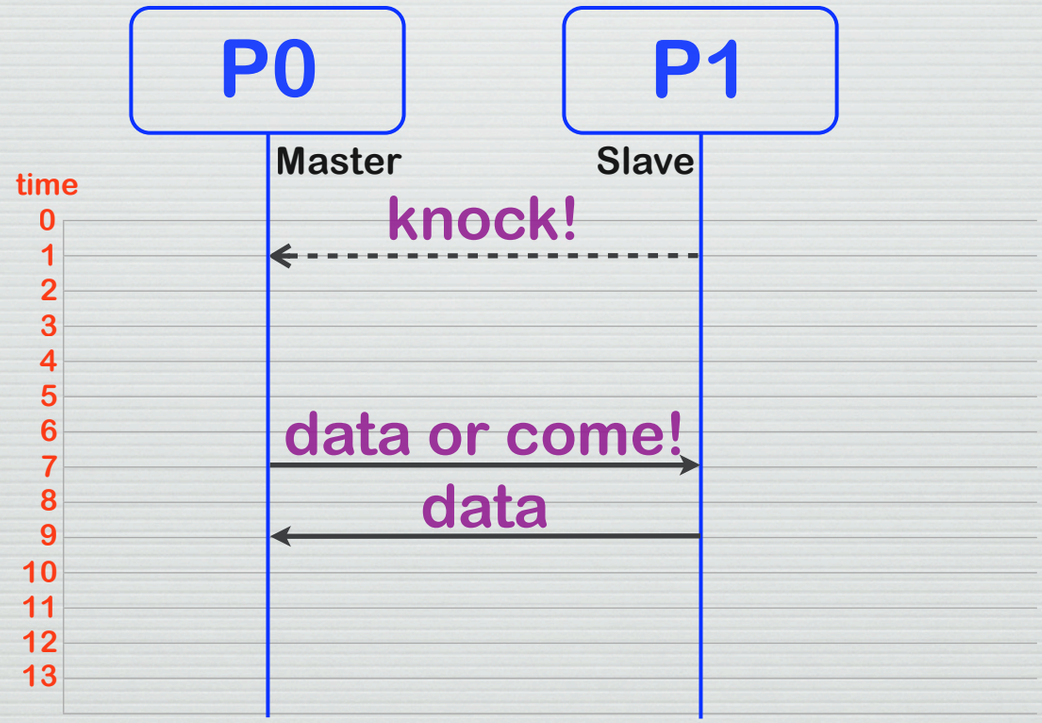


Slave - Master



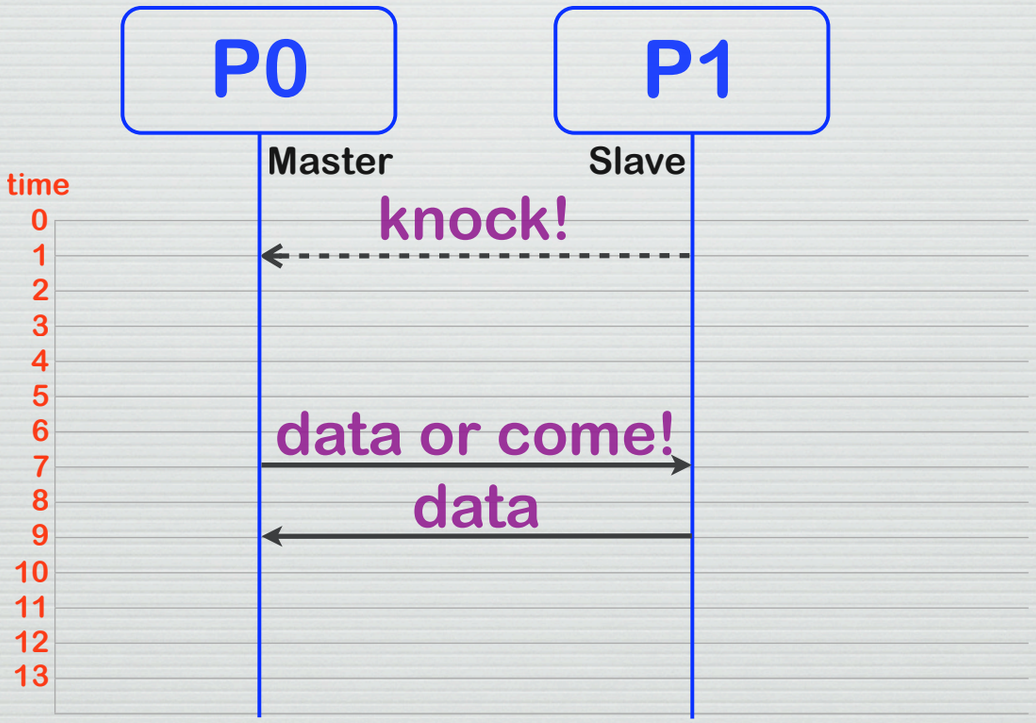
Slave - Master

9



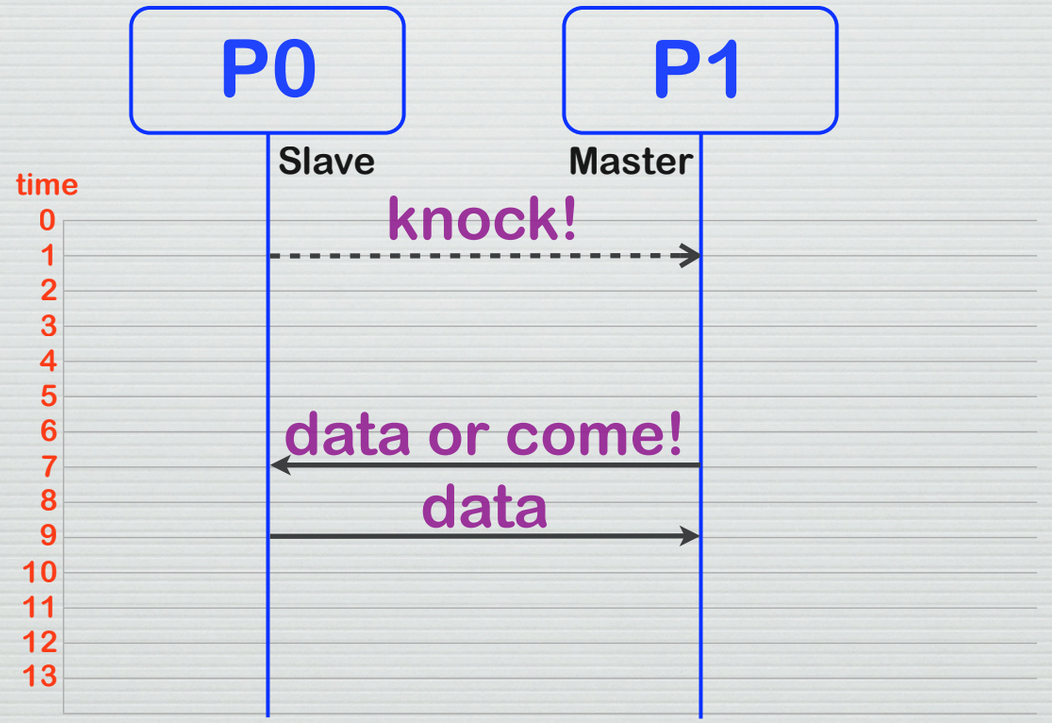
Master - Slave

8



Master - Slave

8



Slave - Master

9

Kontaktinfo

Dette foredraget: http://www.teigfam.net/oyvind/pub/NTNU_2010/foredrag.pdf

Dette faget på NTNU: http://www.itk.ntnu.no/fag/TTK4145/information/

Autronica: http://www.autronicafire.com

oyvind.teig@teigfam.net
oyvind.teig@autronicafire.no

Les dette og mer på http://www.teigfam.net/oyvind/pub/

Kontaktinfo

Dette foredraget: http://www.teigfam.net/oyvind/pub/NTNU_2010/foredrag.pdf

Dette faget på NTNU: http://www.itk.ntnu.no/fag/TTK4145/information/

Autronica: http://www.autronicafire.com

Og - søk jobb hos oss når vi annonserer!

oyvind.teig@teigfam.net
oyvind.teig@autronicafire.no

Les dette og mer på http://www.teigfam.net/oyvind/pub/

- Noen besteforeldre må være på Autronica
  - i 2044 også

- Noen besteforeldre må være på Autronica
- i 2044 også - 34 år fra 2010

Fra harde µ-sekunder via mjuke sekunder til forte år: sann tid i industrien

takk
for meg!