EventTime

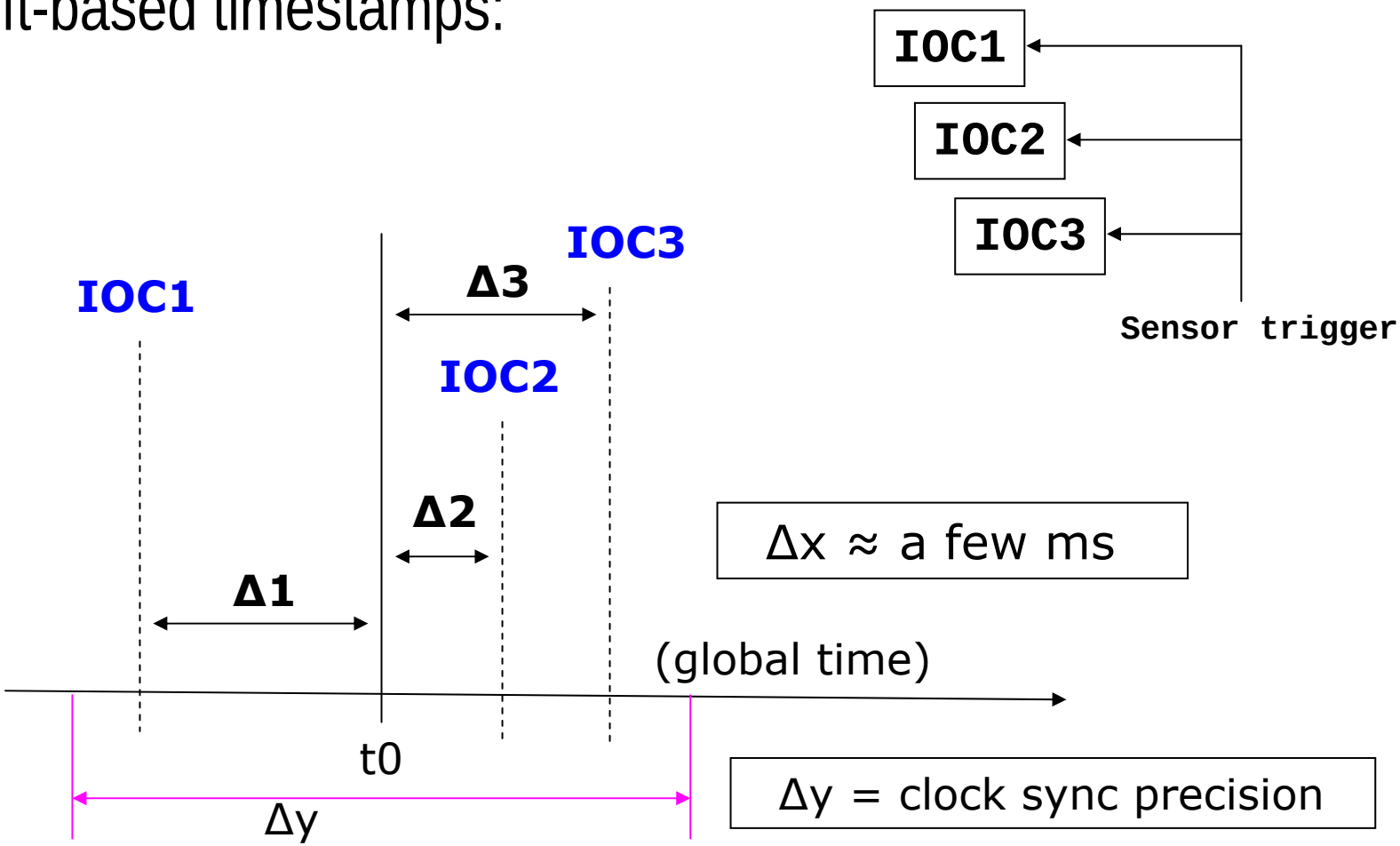
driver for

generalTime

Babak Kalantari

Background

Soft-based timestamps:



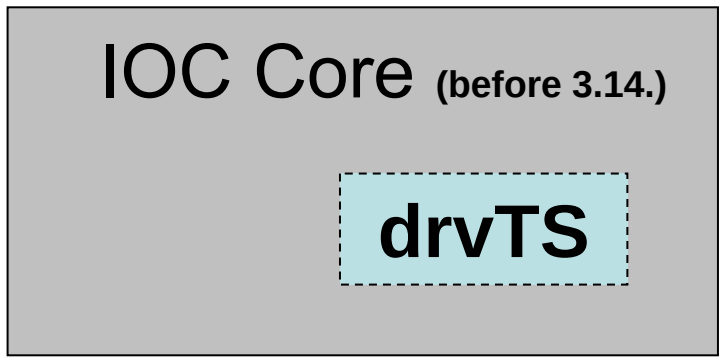
Background

Synchronized timestamps; what does it bring?

- Local (site) time precision
  - what happened first? causality
  - much better results in data correlation
  - other apps.: CA delay measurement
- Global time synchronization
  - Non-EPICS computers (stand-alone systems)
  - External data correlation

Background

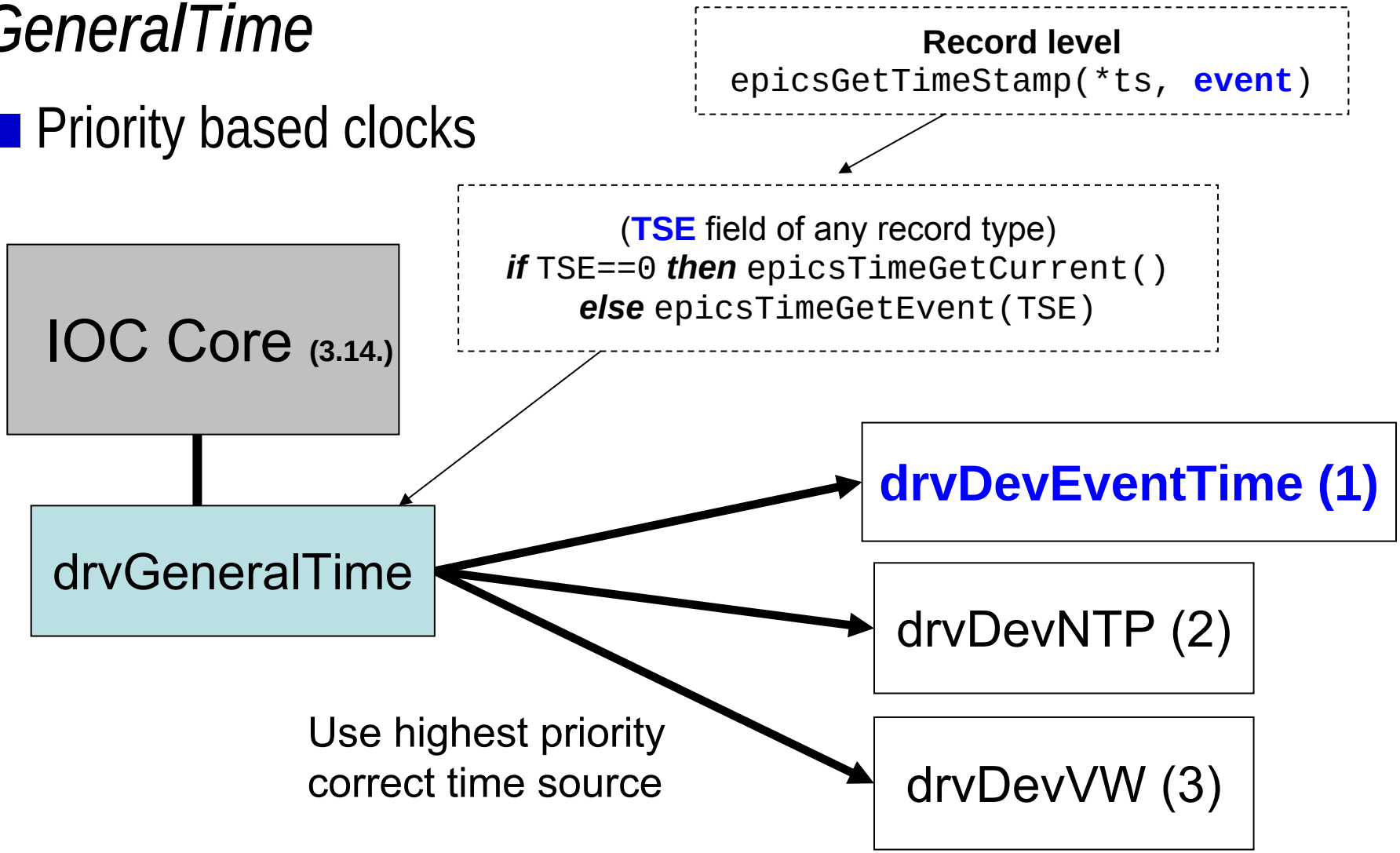
- *drvTS* still serving the community but,
- integral part of *iocCore*
  - development problems
  - customized *iocCore*
  - EPICS version dependent



- Problems with sync TS
  - No failover mechanism (no task switching)
  - No sync with global time
  - Hard-coded interface to HW drivers + OS dependencies

GeneralTime

- Priority based clocks

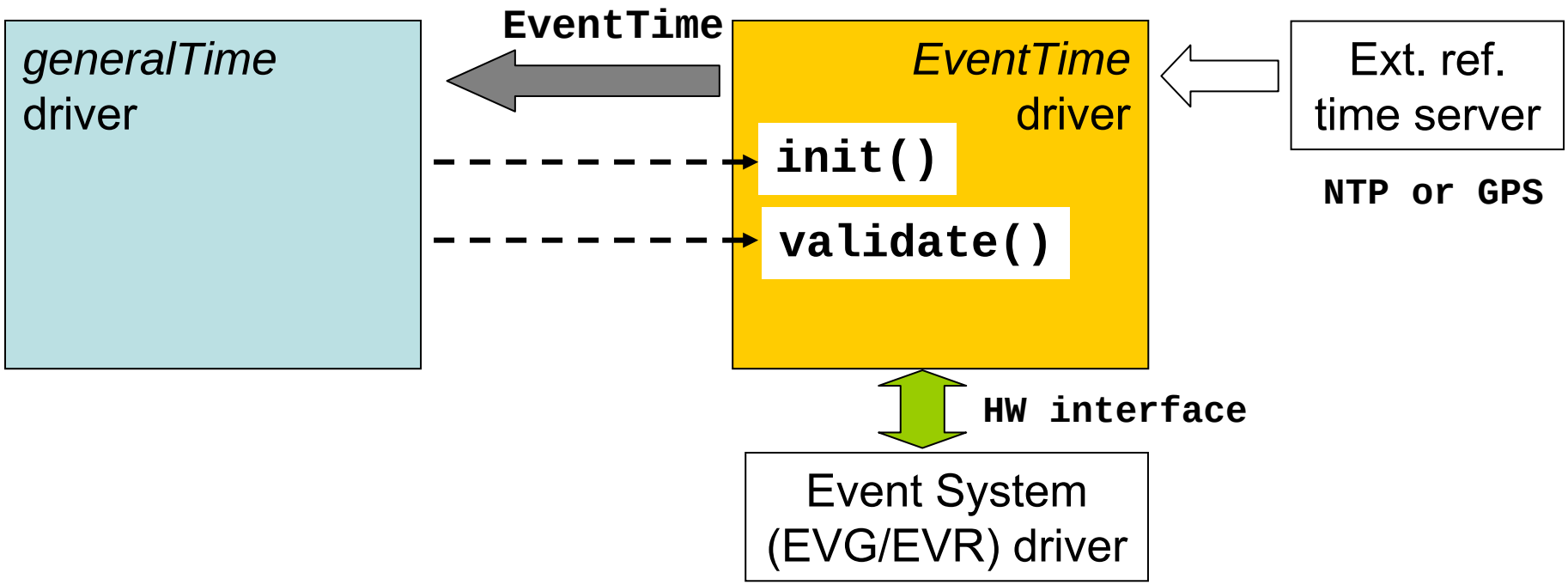


EventTime driver, what does it offer?

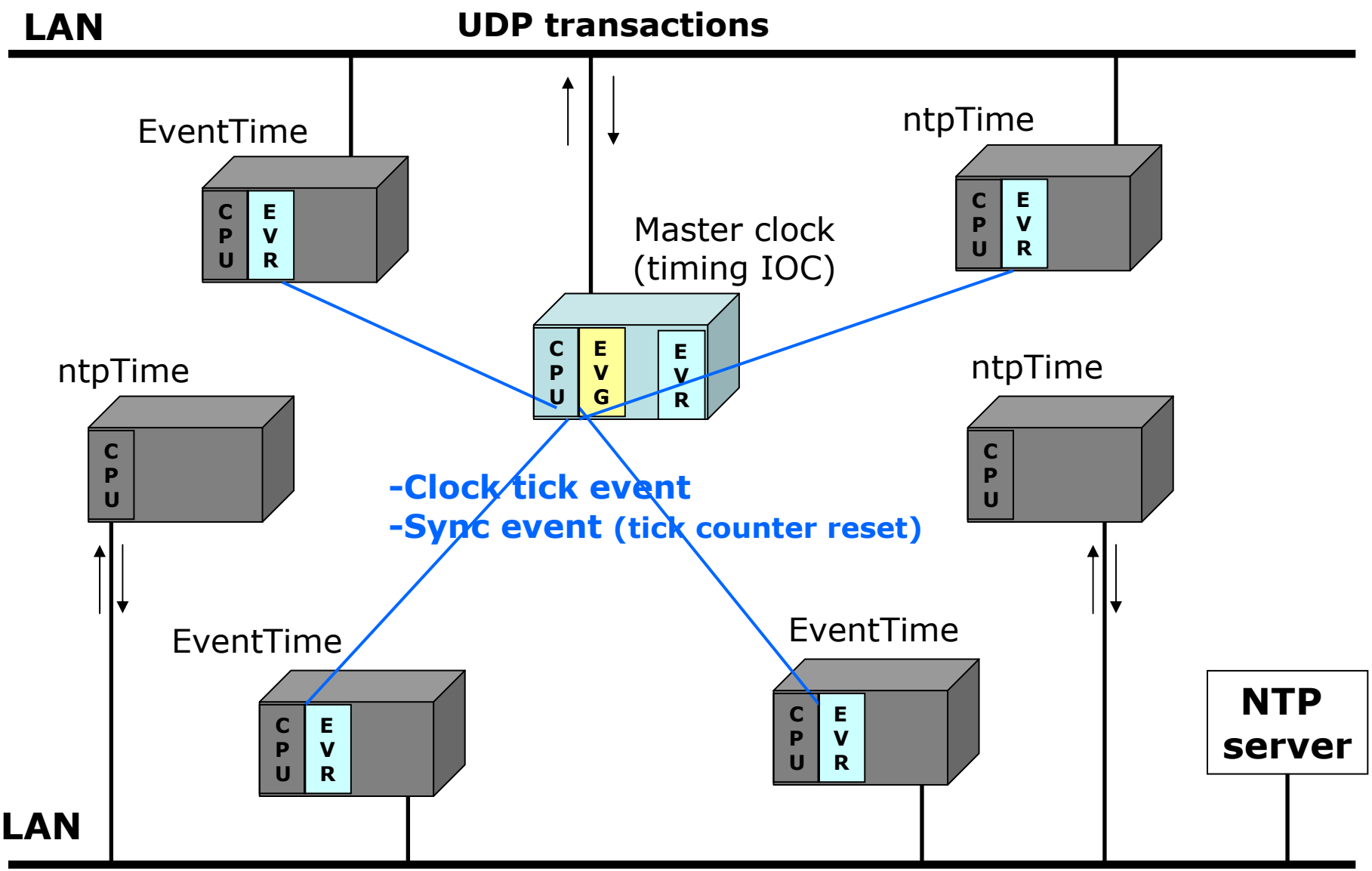
- Tight synchronized clocks / timestamps
- Sync with external time ref.
- Reliability features
- Plug-in design
- No hard-coded dependency
- Uses standard *EPICS* routines (OSI)

EventTime driver

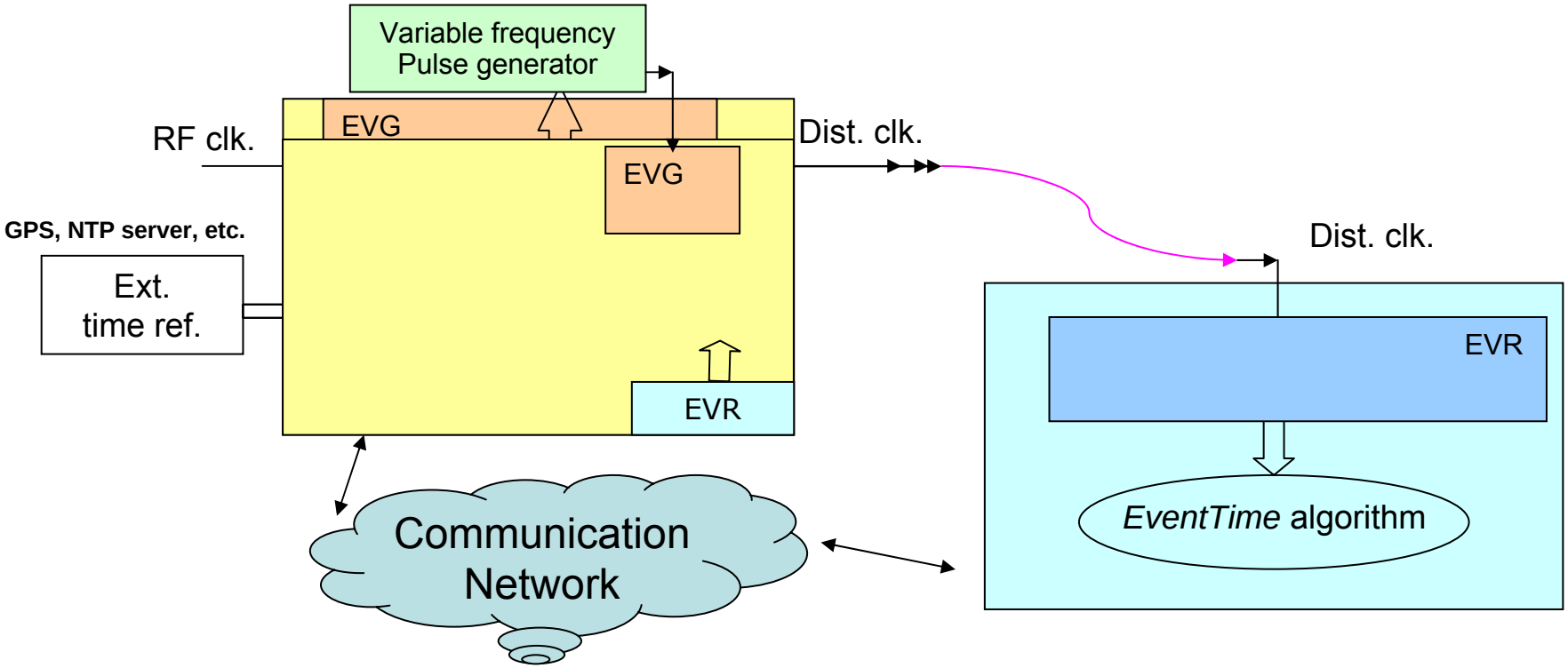
Where does it sit? (relations)



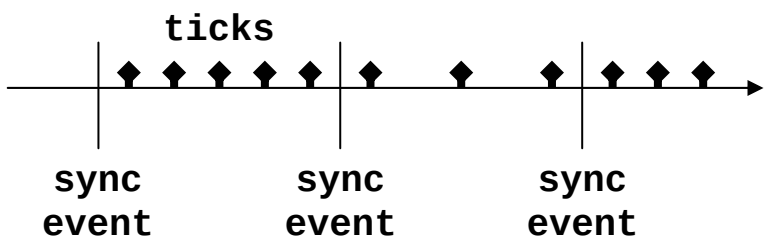
Mechanism


LAN
UDP transactions
EventTime
CPU
EVR
Master clock
(timing IOC)
CPU
EVG
EVR
ntpTime
CPU
EVR
ntpTime
CPU
ntpTime
CPU
-Clock tick event
-Sync event (tick counter reset)
EventTime
CPU
EVR
EventTime
CPU
EVR
NTP
server
LAN

Mechanism



$$T_{current} = T_{last\_sync} + T_{ticks} * \mathbf{conv}$$



EventTime EPICS threads

- Master
  - *EVTstartStampServer()* // listens for timestamp and sync requests from slaves
  - *EVTstartSyncServer()* // broadcasts the sync timestamp at every sync event
  - *EVTstartPLL()* // sync event time with ext. ref. time server (e.g. NTP)
- Slave
  - *EVTstartSyncClient()* // listens for sync timestamp messages from master
- common (master/slave)
  - *EVTstartClkWatch()* // verifies increasing time function

EventTime_Init()

- Called at “initHookAtEnd”
- Checks for required HW
- Creates the required EPICS threads
- Registers itself to *generalTime*
- *EventTime* is validated upon first sync event

Interfaces

- Plug-in interface not limited to event system
  - Timing HW should provide minimal functionalities
  - *drvDevEventTime.h* defines TSrxHW / TStxHW structures

- The master locks to ext. time server if
  - *set_ClkFreq ()* / *get_ClkFreq ()* installed to TStxHW structure

Clock controls & Fault detection

- bo device support to validate *EventTime*
  - Valid / Invalid

- mbbi device support to read *EventTime* status
  - EVT_time_OK,
  - EVT_stream_lost,
  - EVT_sync_timeout,
  - EVT_ntpLock_lost,
  - EVT_tick_fail

Tests

- Long term stability
  - Maintain locking to the ext. time server
- Robustness of the system (fault injection)
  - Breaking event stream (FO links)
  - Master crash or silent fail (no sync messages, no reply)
  - Force master to lose locking to ext. ref.
  - Giving kicks to tick rate
  - Stop master to transmit ticks

How to use it (what is needed)

- Epics 3.14.8 or higher
- *generalTime* & *EventTime* driver
- Master: EVG & EVR
- Slave: EVR
- Generate tick rate & sync events
- At startup
  - *EvtTimeConfigure(master, sync_rate_sec, clock_rate_hz, master_port, slave_port, time_out)*

- Now & Future
  - Putting that into production at SLS
  - To get it working for other platforms e.g. PCI/Linux, soft IOC, RTEMS, ...
  - Fine tuning: *generalTime registration, internal logic/algorithms, API, etc.*

- Acknowledgment:
  - Timo Korhonen
  - Jim Kowalkowsky, Dave Thompson, Sheng Peng, Andrew Johnson