Time Sync Overview

Agenda

- What is Time Sync?
- Time Sync supported features
- Time Sync not supported features
- Time Sync stack diagram
  - PTP stack to application layer
  - Time Sync HAL to PTP stack layer
  - APIs from Low level driver layer
- Time Sync PTP Sequence diagram
- Multi-core Time-synchronization design

What is Time Sync?

- Time Sync is a PDK transport library developed to provide uniform IP abstracted PTP support to CPSW & ICSS IPs.
- Time Sync library utilizes hardware timestamping provided by CPTS(CPSW) or IEP(ICSS) to enable time synchronization with better accuracy
- Time Sync library consists of two parts:
  - Time Sync HAL
  - Time Sync Protocol (PTP)

Time Sync HAL:

- HAL provides an uniform API to the protocol layer by abstracting CPSW and EMAC APIs
- HAL provides Timer and PTP frame transmission/reception APIs

Time Sync Protocol:

- Protocol layer implements TI-RTOS based PTP stack with generic time sync capabilities

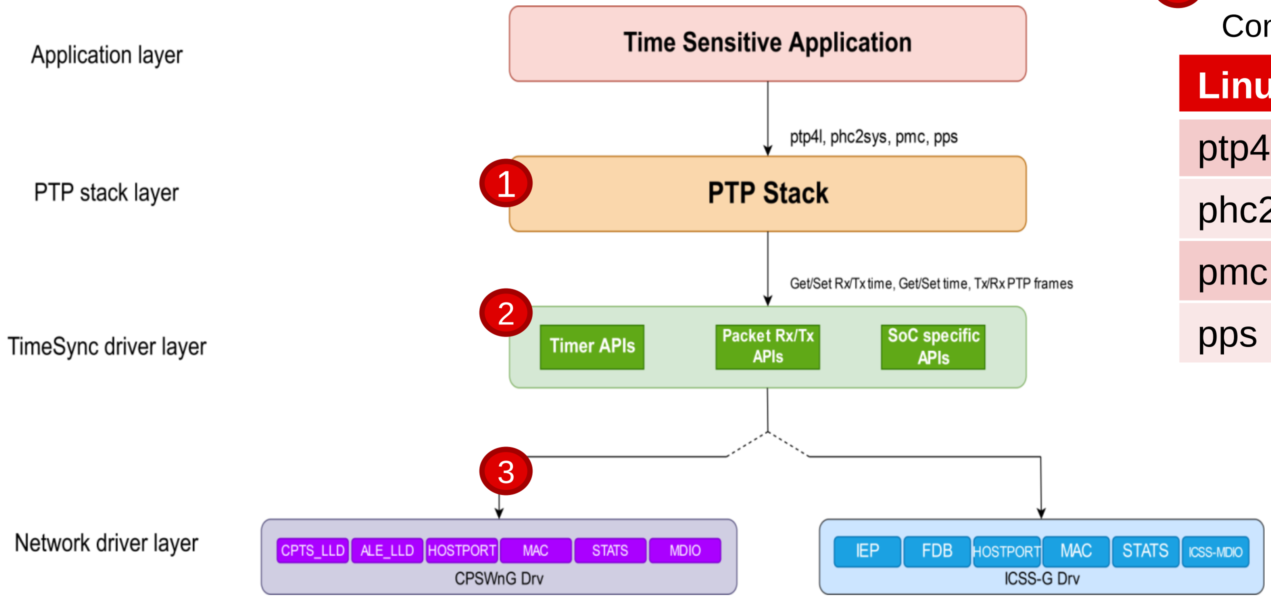
Time Sync supported features

- **Configurations expected from application:**
  - Device configurations: Master/Boundary/Transparent/Ordinary slave
  - Step-mode configuration: Two-step
  - Network protocol: IEEE802.3.
  - Delay mechanism: P2P
  - PPS configuration
- **APIs expected from Low-level network driver:**
  - DMA APIs:
    - Tx & Rx packet submit/retrieve functions
    - Tx & Rx event notify
  - Hardware timer functions:
    - Get Tx/Rx timestamp
    - Get time, Set time
    - Periodic pulse
    - Set compensation (PPM)

Time Sync not supported features

- Best Master Clock algorithm
  - Currently, the first announce message sender is assumed as master
- PTP management frame handling
- IPv4/ IPv6 PTP frame handling
- Not tested on CPSW2G, AM3xxx/AM4xxx/AM5xxx platforms

Time Sync stack diagram



1 **PTP stack to application APIs**
Comparison with Linux PTP

Linux PTP	TimeSync PTP
ptp4l	TimeSync_initPtp
phc2sys	TimeSync_coupleTimer
pmc	TimeSync_ptpMgmt
pps	TimeSync_configPps

TimeSync driver layer refers to Time sync HAL abstracting underlying network driver layer

APIs to PTP stack layer

hTimeSync = TimeSync_open(&timeSyncConfig)

Timer APIs

- TimeSync_getTxTimestamp(hTimeSync, txFrameType, txPort, seqId, &nanoseconds, &seconds)
- TimeSync_getRxTimestamp(hTimeSync, rxFrameType, rxPort,seqId, &nanoseconds, &seconds)
- TimeSync_getCurrentTime(hTimeSync, &nanoseconds, &seconds)
- TimeSync_setClockTime(hTimeSync, nanoseconds, seconds)
- TimeSync_adjTimeSlowComp(hTimeSync, adjOffset, interval)

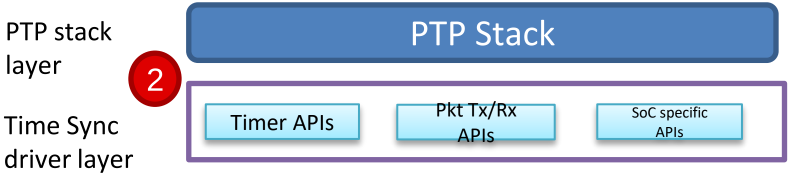
Packet Tx/Rx APIs APIs

- TimeSync_getPtpFrame(hTimeSync, &buff, &size, &portNum)
- TimeSync_sendPtpFrame(hTimeSync, &buff, size, portNum)

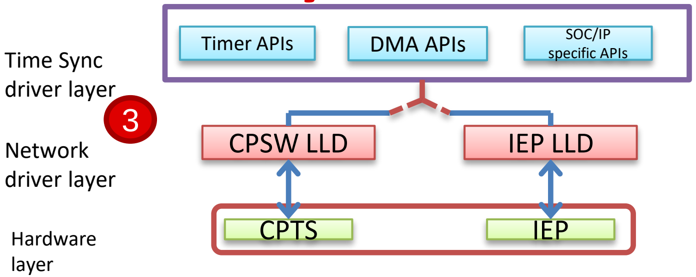
SoC/ Use case specific APIs

For AM335x & AM437x

- TimeSync_ecapEdmaConfig(ecapEdmaConfig)

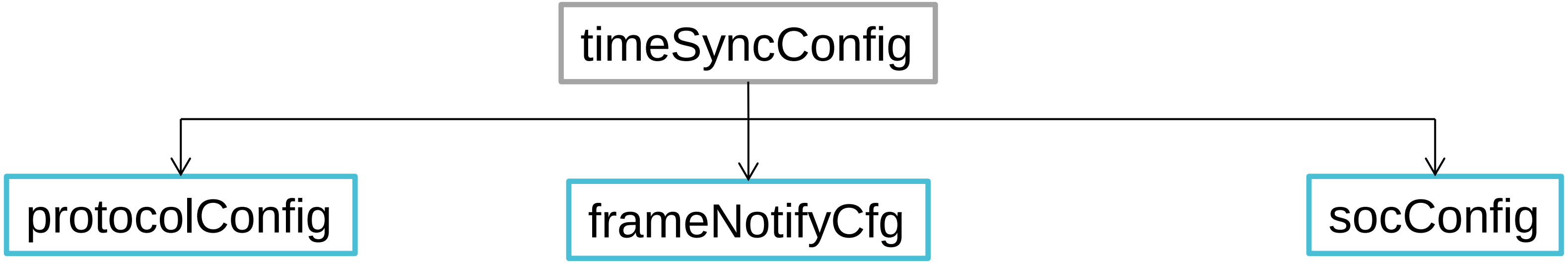


APIs from Low level driver layer



TimeSync API	CPSW API	ICSS API
TimeSync_getTxTimestamp	CPSW_CPTS_IOCTL_LOOKUP_EVENT	IEP register read/writes
TimeSync_getRxTimestamp	CPSW_CPTS_IOCTL_LOOKUP_EVENT	“”
TimeSync_getCurrentTime	CPSW_CPTS_IOCTL_PUSH_TIMESTAMP_EVENT	“”
TimeSync_setClockTime	CPSW_CPTS_IOCTL_LOAD_TIMESTAMP	“”
TimeSync_adjTimeSlowComp	CPSW_CPTS_IOCTL_SET_TS_PPM	“” (For AM335x & AM437x ECAP config API)
TimeSync_getPtpFrame	CpswDma_retrieveRxPackets	ICSS_EmacRxPktGet
TimeSync_sendPtpFrame	CpswDma_submitTxReadyPackets	ICSS_EmacTxPacket

Time Sync Configuration (parameters from PTP to HAL)



timeSyncConfig

protocolConfig

frameNotifyCfg

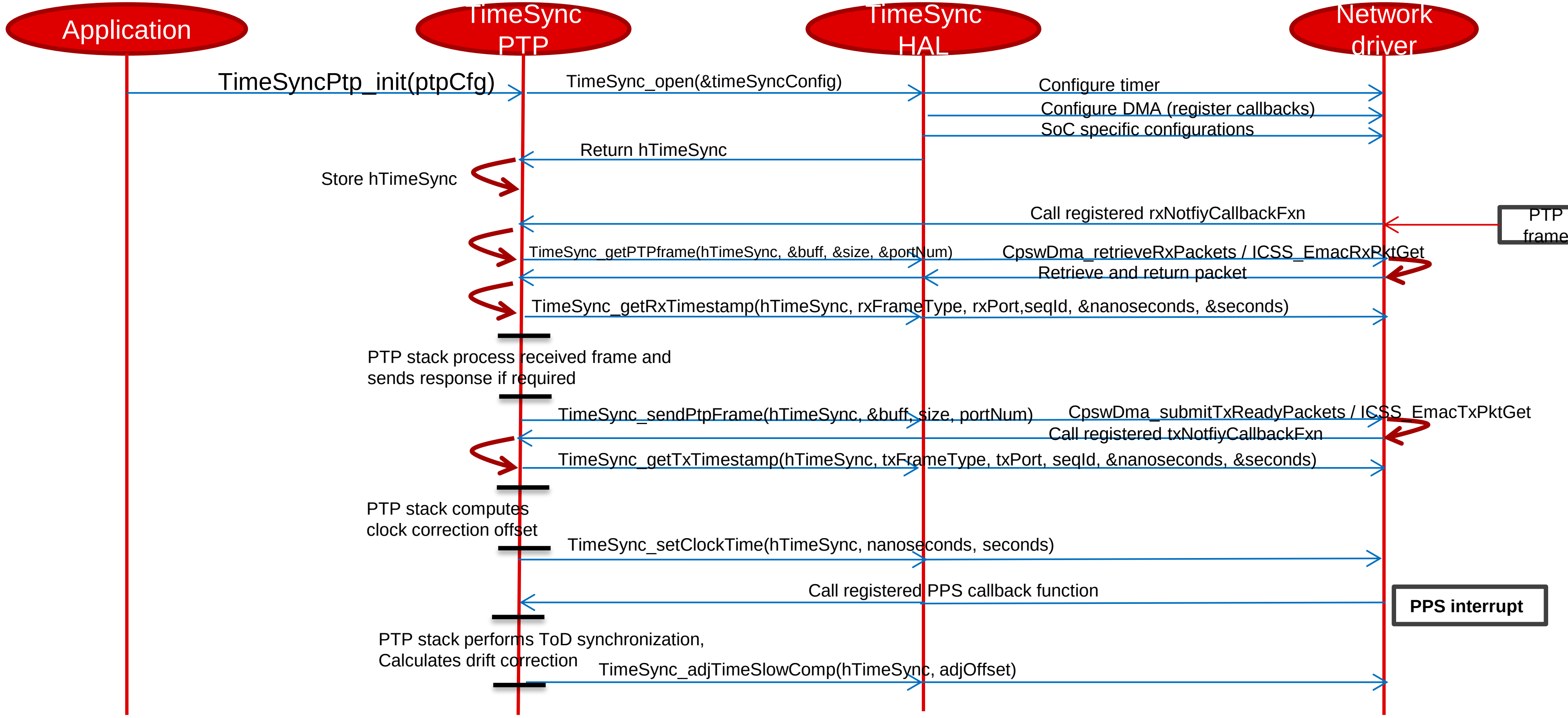
socConfig

- protocol
  (Annex D/E/F)
- deviceCfg
  (MC/OC/TC/BC)
- StepMode
- VLAN Configuration
- PPS configuration
  (PPS: interval, callback
function)
- Port mask

- txNotifyCb
- txNotifyCbArg
- rxNotifyCb
- rxNotifyCbArg

- socVersion
- ipVersion

Sequence diagram



Application

TimeSync PTP

TimeSync HAL

Network driver

TimeSyncPtp_init(ptpCfg)

TimeSync_open(&timeSyncConfig)

Configure timer

Configure DMA (register callbacks)

SoC specific configurations

Return hTimeSync

Store hTimeSync

Call registered rxNotfiyCallbackFxn

PTP frame

TimeSync_getPTPframe(hTimeSync, &buff, &size, &portNum)

CpswDma_retrieveRxPackets / ICSS_EmacRxPktGet

Retrieve and return packet

TimeSync_getRxTimestamp(hTimeSync, rxFrameType, rxPort,seqId, &nanoseconds, &seconds)

PTP stack process received frame and
sends response if required

TimeSync_sendPtpFrame(hTimeSync, &buff, size, portNum)

CpswDma_submitTxReadyPackets / ICSS_EmacTxPktGet

Call registered txNotfiyCallbackFxn

TimeSync_getTxTimestamp(hTimeSync, txFrameType, txPort, seqId, &nanoseconds, &seconds)

PTP stack computes
clock correction offset

TimeSync_setClockTime(hTimeSync, nanoseconds, seconds)

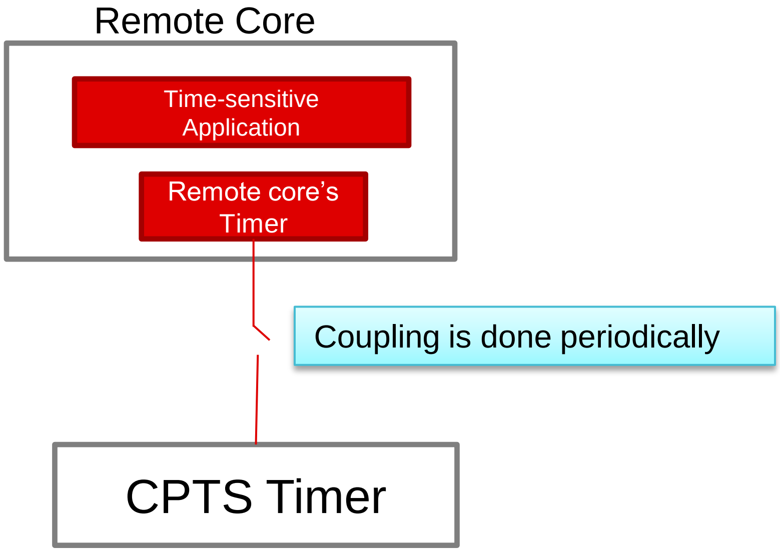
Call registered PPS callback function

PPS interrupt

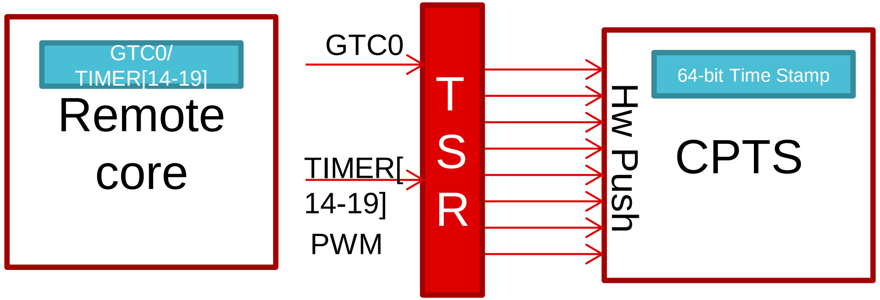
PTP stack performs ToD synchronization,
Calculates drift correction

TimeSync_adjTimeSlowComp(hTimeSync, adjOffset)

Multi – core Time Synchronization (similar to phc2sys in Linux)

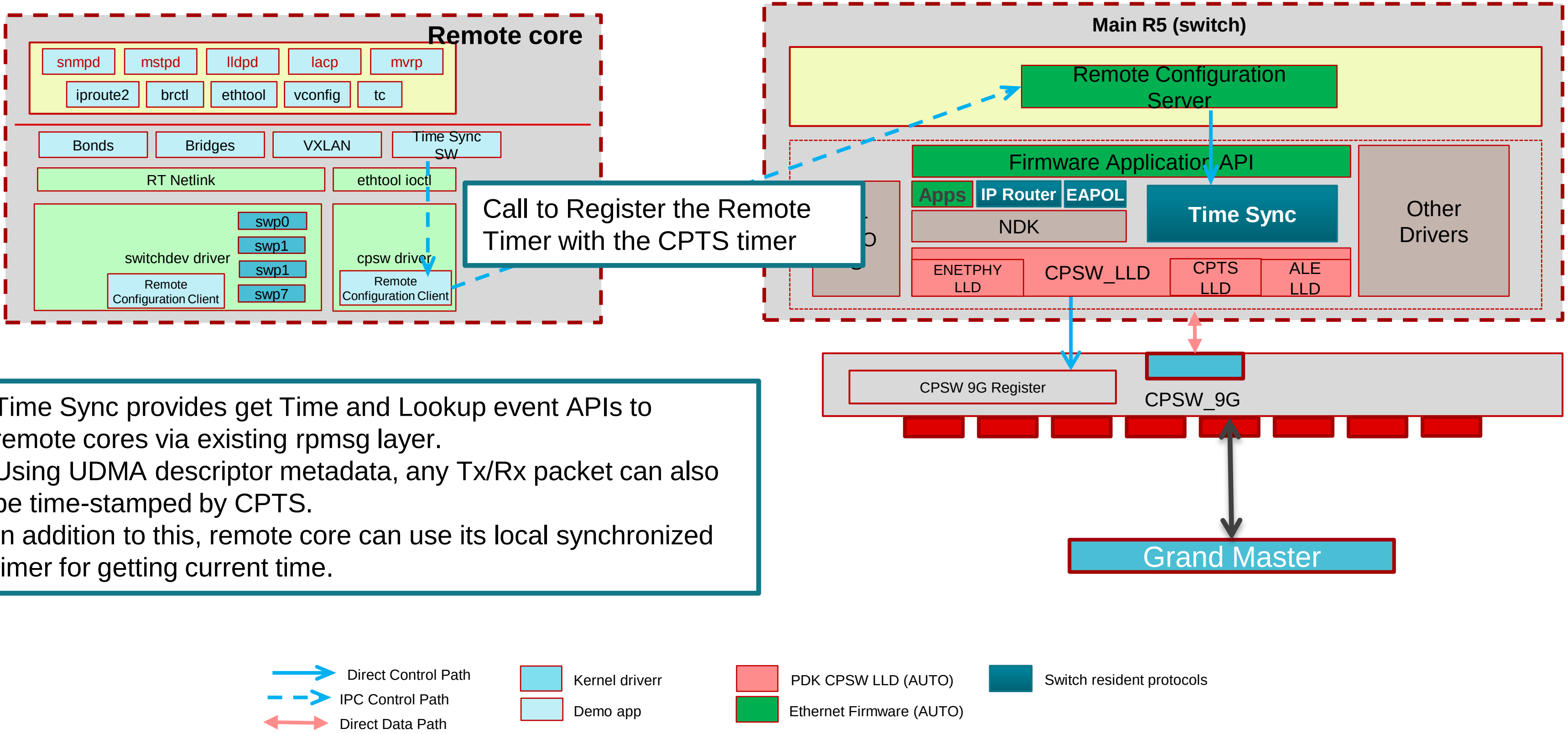


Time Sync Router Configuration



- Coupling between remote core's timer and CPTS timer is done through Hardware push events of CPTS and Time Sync Router(TSR).
- With this a linear relation is established between CPTS & remote timer.
- Once the relation is established, remote core can independently use this coupled timer.

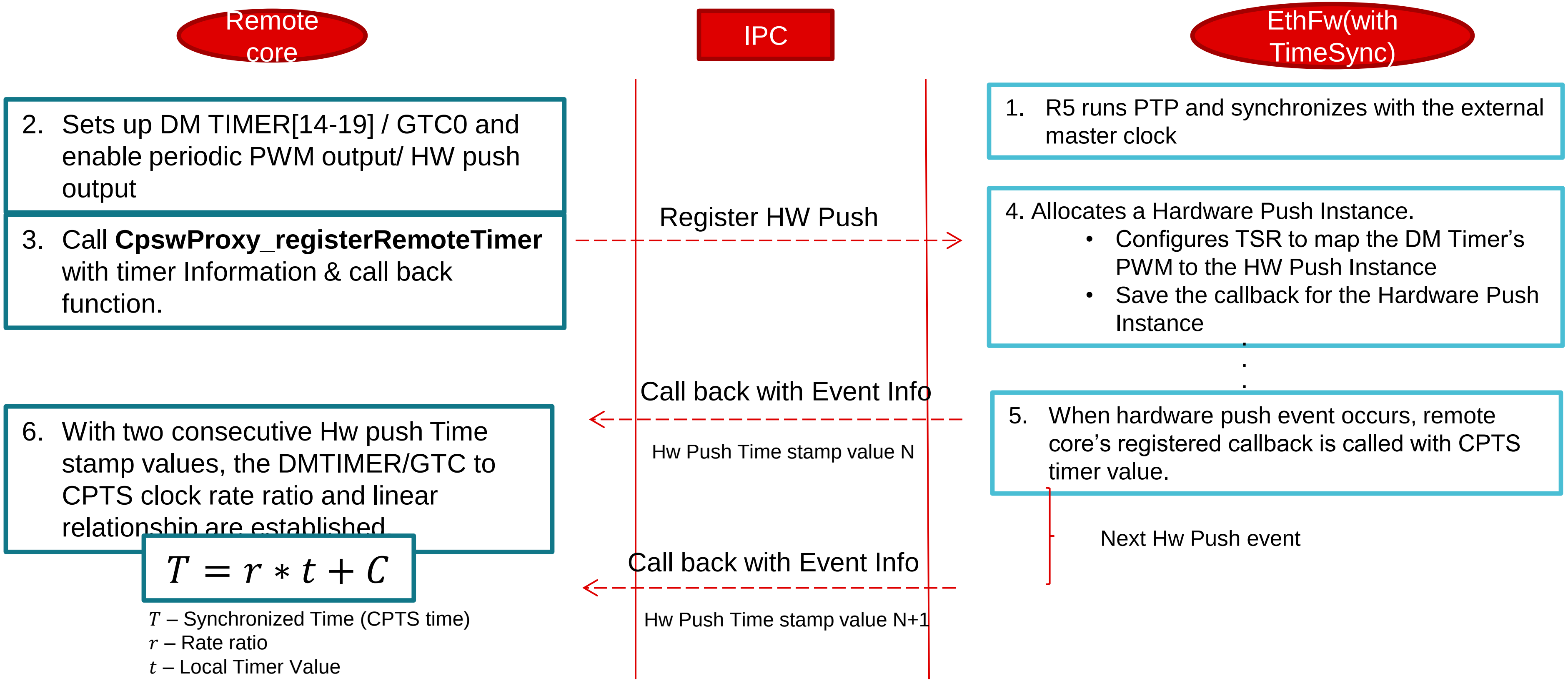
Multi – core Time Synchronization (similar to phc2sys in Linux)



Call to Register the Remote Timer with the CPTS timer

- Time Sync provides get Time and Lookup event APIs to remote cores via existing rpmsg layer.
- Using UDMA descriptor metadata, any Tx/Rx packet can also be time-stamped by CPTS.
- In addition to this, remote core can use its local synchronized timer for getting current time.

Multi – core Time Synchronization sequence



Remote core

IPC

EthFw(with TimeSync)

1. R5 runs PTP and synchronizes with the external master clock

2. Sets up DM TIMER[14-19] / GTC0 and enable periodic PWM output/ HW push output

3. Call **CpswProxy_registerRemoteTimer** with timer Information & call back function.

Register HW Push

4. Allocates a Hardware Push Instance.
   - Configures TSR to map the DM Timer's PWM to the HW Push Instance
   - Save the callback for the Hardware Push Instance

5. When hardware push event occurs, remote core's registered callback is called with CPTS timer value.

Call back with Event Info

Hw Push Time stamp value N

Next Hw Push event

Call back with Event Info

Hw Push Time stamp value N+1

6. With two consecutive Hw push Time stamp values, the DMTIMER/GTC to CPTS clock rate ratio and linear relationship are established

$$T = r * t + C$$

T – Synchronized Time (CPTS time)
r – Rate ratio
t – Local Timer Value
C – Constant value

This material is provided strictly "as-is," for informational purposes only, and without any warranty.
Use of this material is subject to TI's **Terms of Use,** viewable at TI.com