GSM burst transmission in GNU Radio

Piotr Krysik (ptrkrysik@gmail.com)

A FEW WORDS ABOUT ME

- Author of the core part of **gsm-receiver** (part of Airprobe project)
- Main author of **gr-gsm** (successor of Airprobe)
- Author of **Multi-RTL** - a RTL-SDR based multichannel receiver
- Working at Warsaw University of Technology (radar signal processing)

GSM BURSTS TRANSMISSION

What is it for?

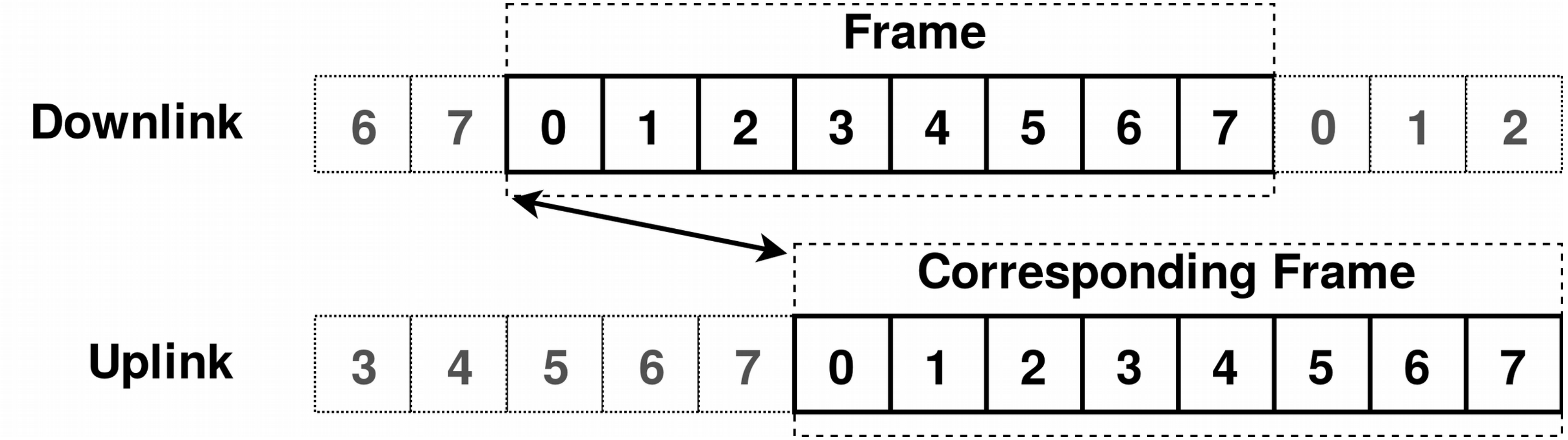
- Implementing Tx side of SDR based burst transceiver for Osmocom-BB
- Could be adapted for BTS TRX
- Could be used for testing of gr-gsm's Rx side performance

gr-gsm ❤ osmocomBB

GSM BURSTS TRANSMISSION

Requirements

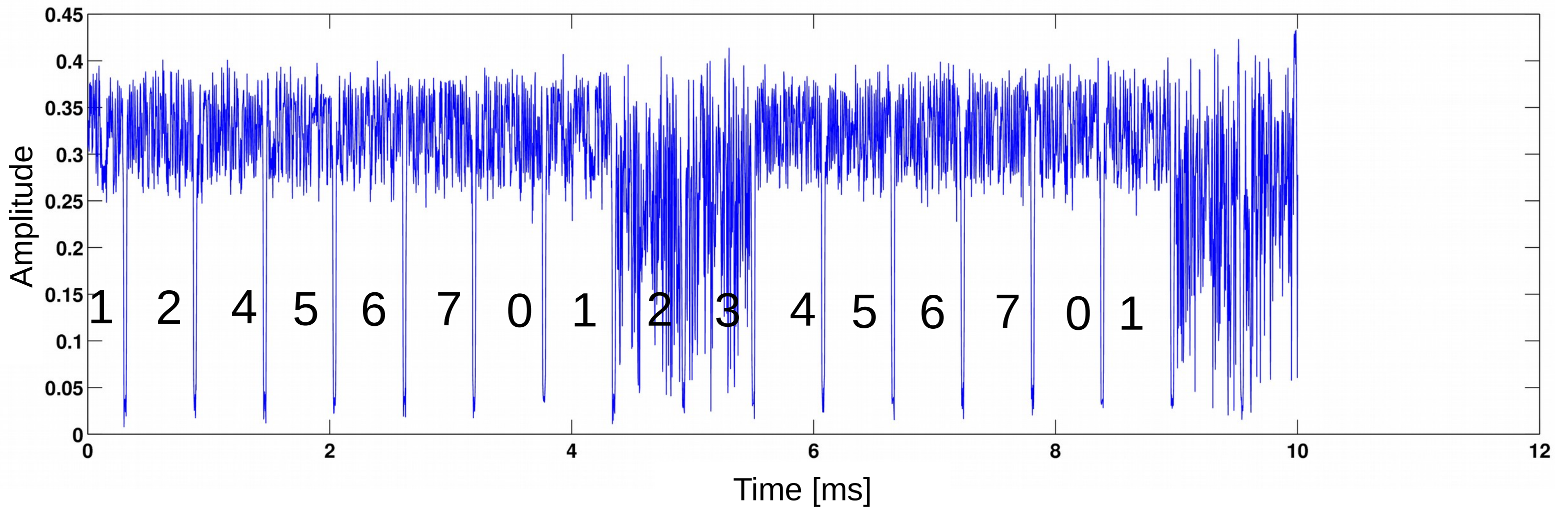
- Input: burst + header with frame number
- Output: GMSK modulated GSM RF signal synchronous with received signal
- Transmitting only when needed (MS most of the time doesn't transmit anything)
- Transmitting in the right time (including Timing Advance)
- Assuring the right signals get transmitted



GSM BURSTS TRANSMISSION

Requirements

- Input: burst + header with frame number
- Output: GMSK modulated GSM RF signal synchronous with received signal
- Transmitting only when needed (MS most of the time doesn't transmit anything)
- Transmitting in the right time (including Timing Advance)
- Assuring the right signals get transmitted

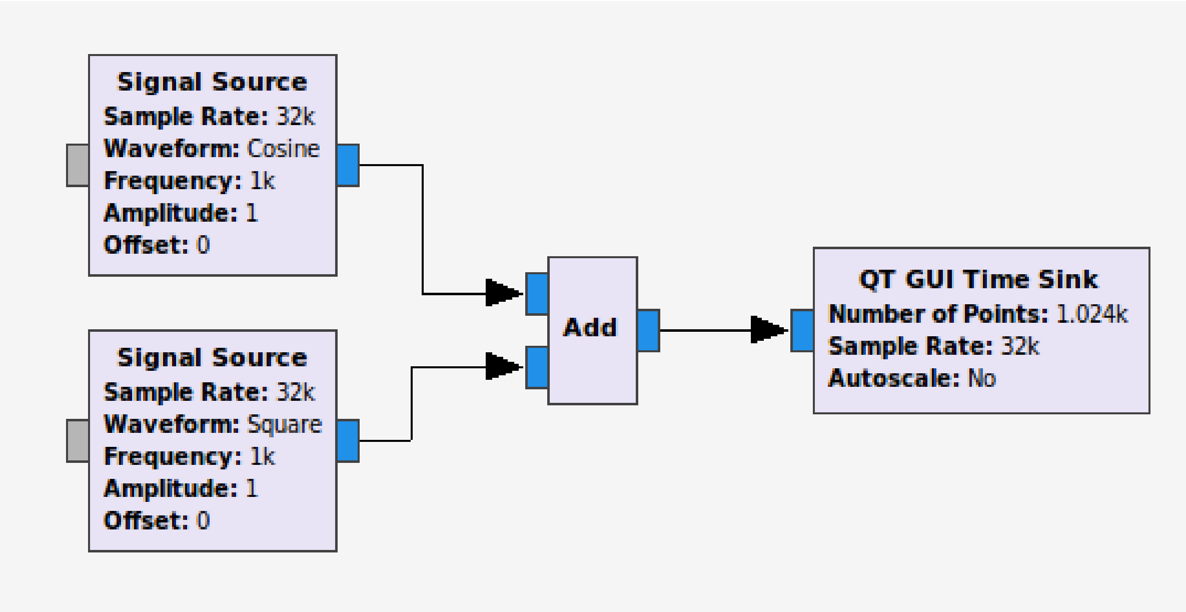


Why use GNU Radio to send GSM bursts?

- Modular - signal processing functions done by reusable blocks
- Advantages
  - No need to reinvent the wheel (filter design, filtering, resampling, de/modulation)
  - No need to know internals of each block
  - Clean architecture of the software
- Disadvantages
  - Some tasks may be easier without limitations imposed by the framework
  - Some tasks might require extending the runtime

Processing of streams of samples

- All samples treated equally
- GNU Radio block see stream of samples through a window
- Block's work:
  - Take some number of samples at the input
  - Apply signal processing function (i.e. FIR filter)
  - Write result to the output



Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0

Signal Source
Sample Rate: 32k
Waveform: Square
Frequency: 1k
Amplitude: 1
Offset: 0

Add

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 32k
Autoscale: No

Processing of streams of samples

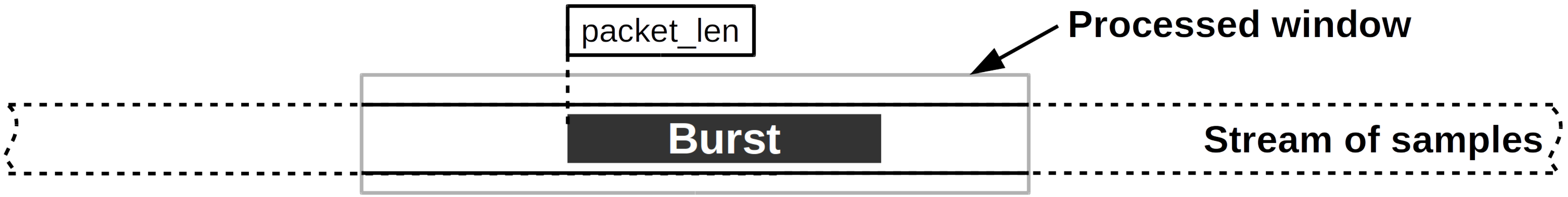
- Limitations
  - No packets
  - No signaling downstream an event has occurrred (i.e. packet preamble detected)
  - No sending data upstream
  - No loops
  - No control mechanism

Stream tags

- Metadata attached to a given sample in a stream
- Travel with samples across multiple blocks
- Any information in PMT (Polymorphic Type) format
  - Current frequency
  - Current time
  - ...
- Can be used to modify block's behavior at given moment
  - Change frequency shift
  - Change re-sampling rate
  - ...

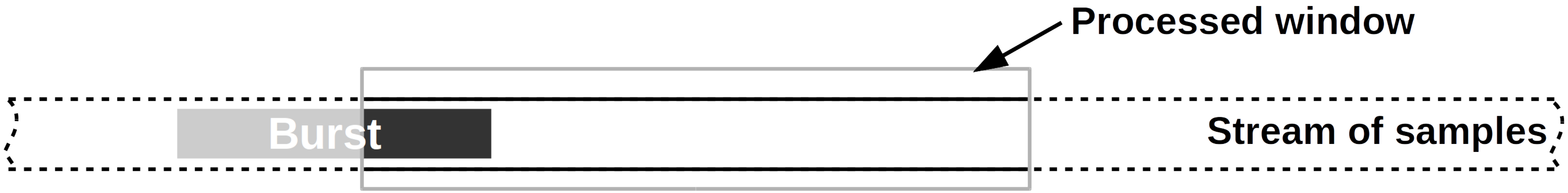
Stream tags

- Can be used to mark start of a burst



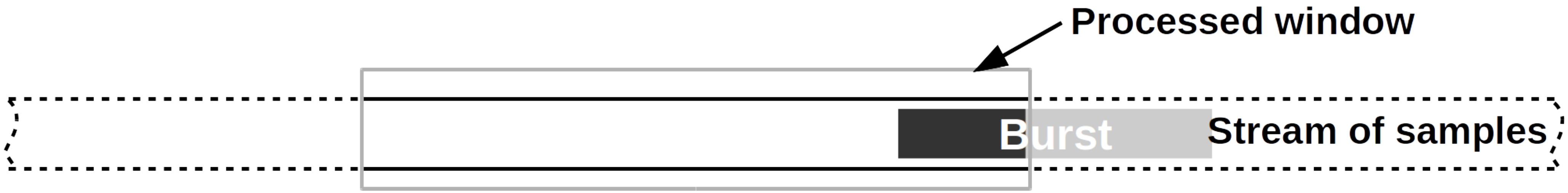
Stream tags

- Can be used to mark start of a burst
- … but there are multiple possible positions of a burst in the block's buffer



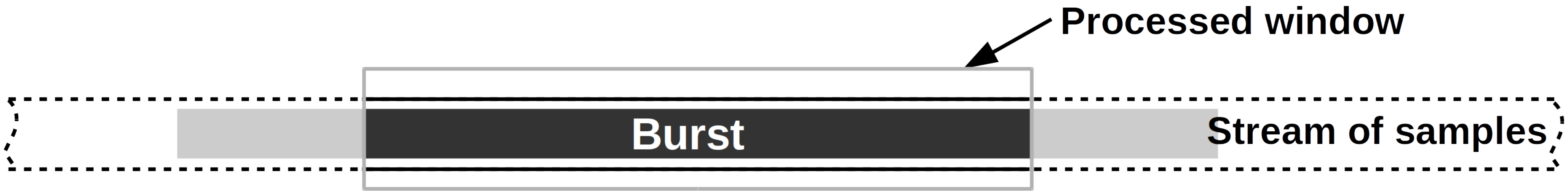
Stream tags

- Can be used to mark start of a burst
- … but there are multiple possible positions of a burst in the block's buffer



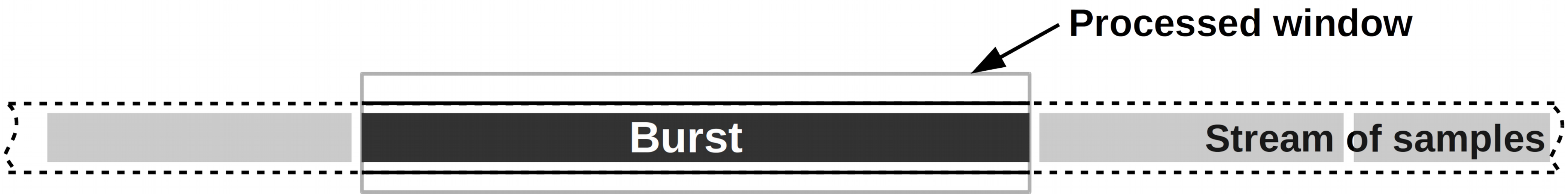
Stream tags

- Can be used to mark start of a burst
- … but there are multiple possible positions of a burst in the block's buffer



Tagged streams

- Streams "packetized" with use of stream tags
- Tag marks start of packet
- Packet length as tag value
- No spaces between packets
- Whole packet processed by a tagged stream block at once (assured by GNU Radio's runtime)



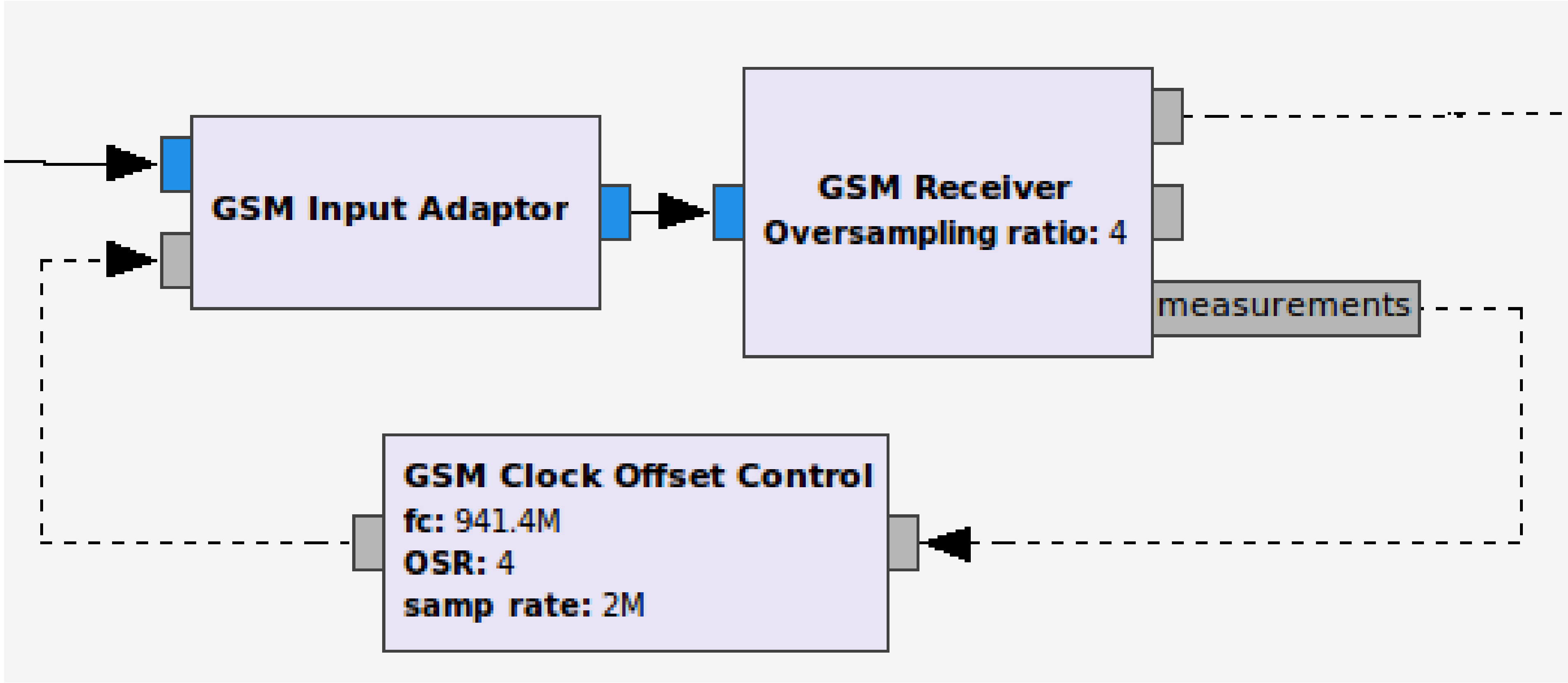
Tagged streams

- Advantages
  - uses buffers preallocated and maintained by GNU Radio
  - no need to check multiple conditions of packet location in the input buffer
  - simpler blocks
- Disadvantages
  - packet size limit - size of GNU Radio buffers
  - packet header is not distinguished from the payload
  - it's a hack

Message Passing

- Mechanism to send asynchronous messages (PMTs) between blocks
- Independent from samples streams
- Blocks can process streams of samples and messages
- Can be used for:
  - Setting parameters of one block by another block
  - Sending packets (special PDU format, pair – header (dictionary) + content (binary blob))
  - Informing about events
  - Implementing loops

Message Passing



Message Passing

- Advantages
  - Flexible - messages can carry multiple data types (PMTs!)
  - Well suited for package representation (i.e. packet with easy to distinguish header fields and data)
- Disadvantages
  - Asynchronism — non-deterministic order of messages in parallel branches
  - No back-pressure (no mechanism to limit how fast message source produces messages)
  - No preallocated space in memory for message content

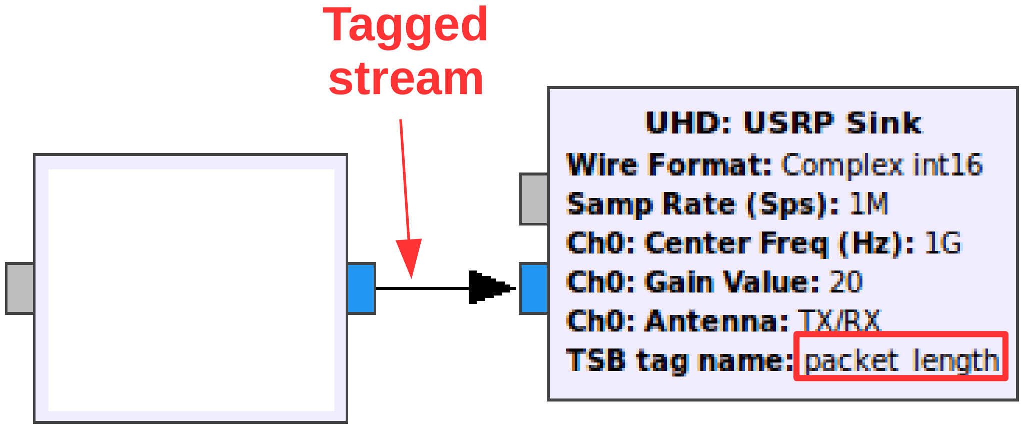
Want more info about GNU Radio features?

See "Stream Tags, PDUs, and Message Passing" talk by Tom Rondeau

BURST TRANSMISSION IN GNU RADIO

How to transmit bursts with GNU Radio?

- UHD (USRP Hardware Driver) provides tag based interface for transmitting bursts
- Access to the interface through “**UHD: USRP Sink**” block
  - Connecting tagged stream to the input (with length tag at start of each packet)
  - Configuring “TSB tag name” (length tag name) in “UHD: USRP Sink”
  - Adding “tx_time” tags at the same positions as the length tags



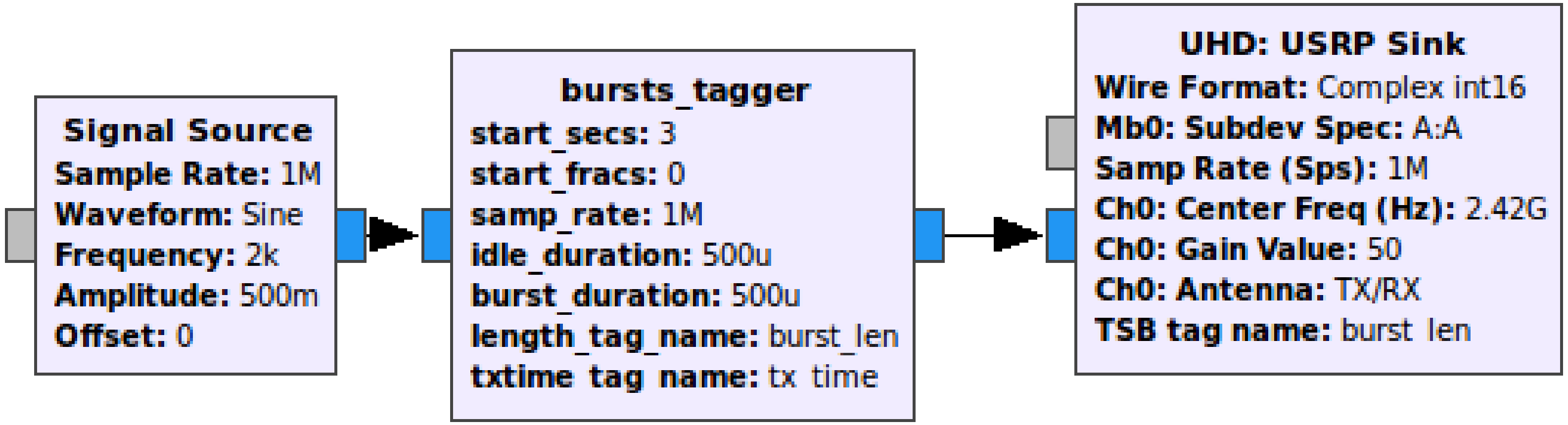
Tagged stream

UHD: USRP Sink
Wire Format: Complex int16
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): 1G
Ch0: Gain Value: 20
Ch0: Antenna: TX/RX
TSB tag name: packet_length

BURST TRANSMISSION IN GNU RADIO

Demo

- Bursts tagger adds “burst_len” (length) and “tx_time” tags



Signal Source
Sample Rate: 1M
Waveform: Sine
Frequency: 2k
Amplitude: 500m
Offset: 0

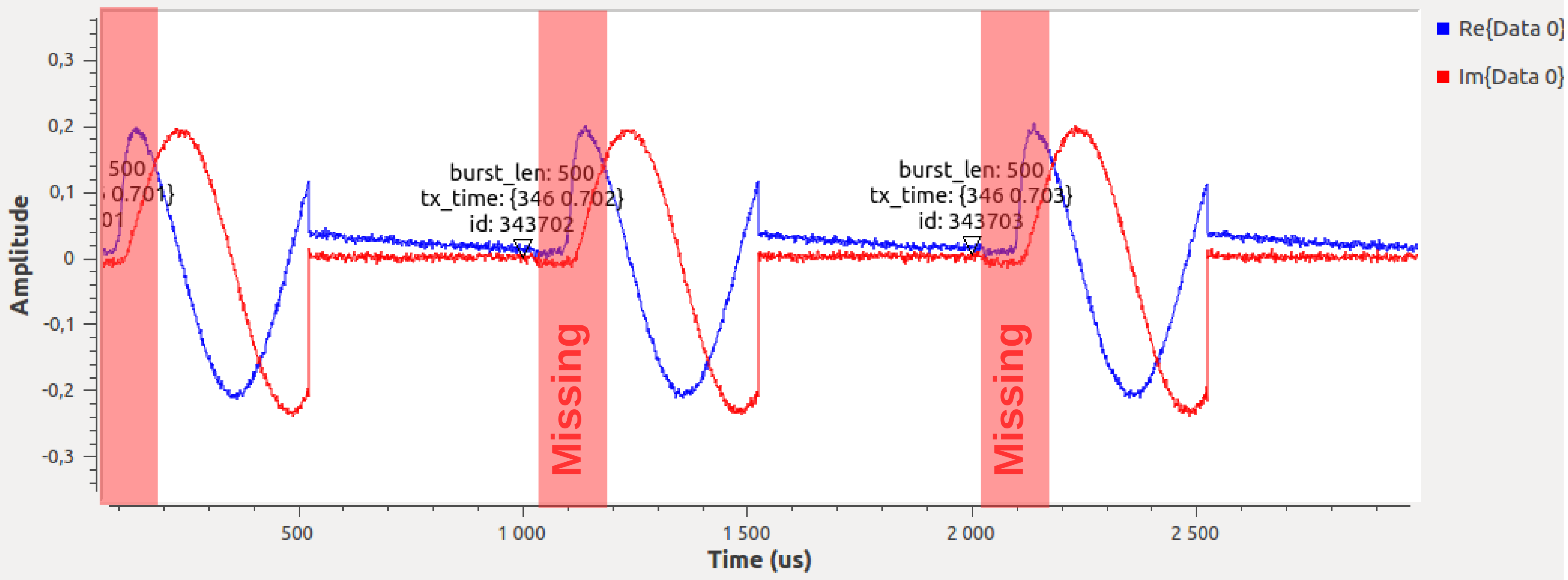
bursts_tagger
start_secs: 3
start_fracs: 0
samp_rate: 1M
idle_duration: 500u
burst_duration: 500u
length_tag_name: burst_len
txtime tag name: tx time

UHD: USRP Sink
Wire Format: Complex int16
Mb0: Subdev Spec: A:A
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): 2.42G
Ch0: Gain Value: 50
Ch0: Antenna: TX/RX
TSB tag name: burst len

WORLD’S NOT PERFECT: HARDWARE ISSUES WITH BURST TRANSMISSION

Corrupted beginnings of bursts

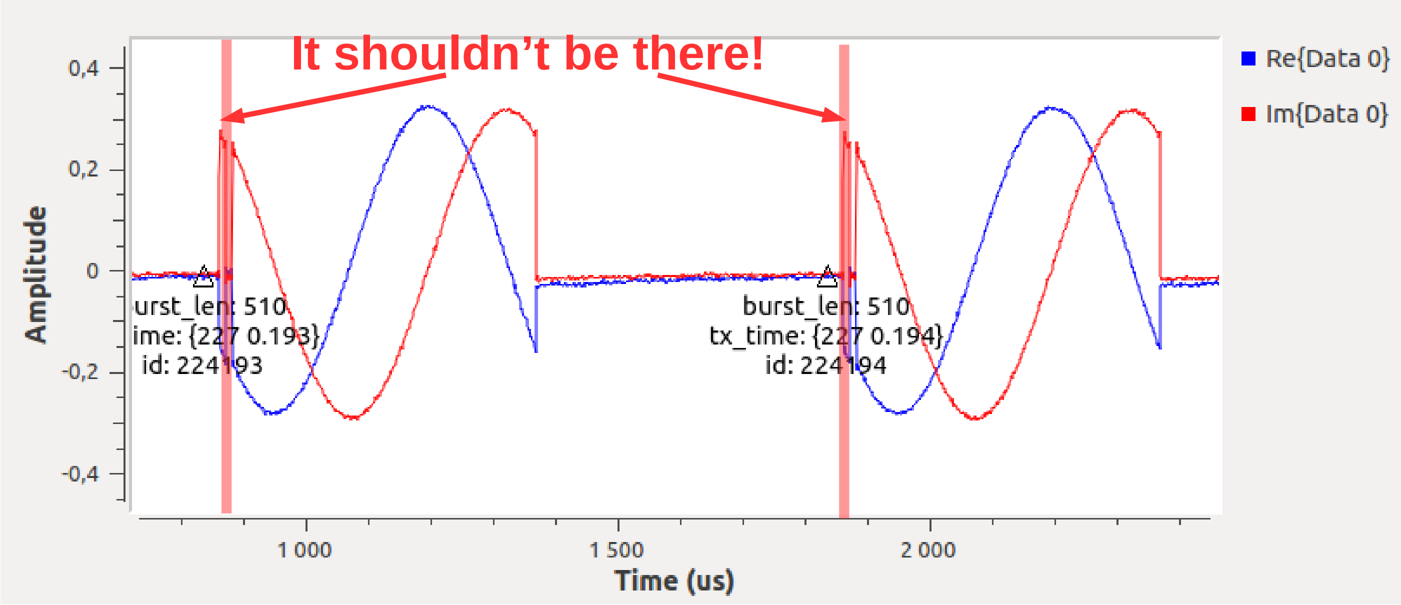
- Problem only with USRP B2x0 when:
  - Transmitting and receiving with the same side of the device
  - There is no connection between active pin and signal ground in the Tx port (i.e. dipole antenna or no antenna)



WORLD’S NOT PERFECT: HARDWARE ISSUES WITH BURST TRANSMISSION

Missing tail of burst – transmitted as start of next burst

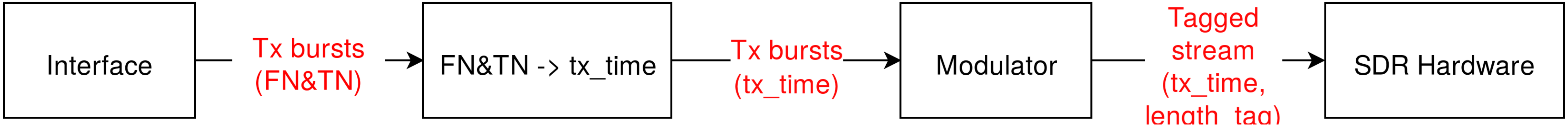
- Appears for many types of USRPs:
  - B2x0
  - X3x0 (for new firmware versions only tail of burst is missing)
- Solution: add enough zero samples at the end of each burst


It shouldn’t be there!

TRANSMITTING GSM BURSTS WITH GNU RADIO

Building GSM transmitter

- Interface - receive bursts with FN (frame number) & TN (Timeslot Number)
- FN&TN → Tx - converts FN&TN pair to tx_time
- Modulator - from PDU messages with bursts to tagged stream with modulated bursts
- SDR Hardware - digital baseband bursts to bursts of RF signal



Interface

Tx bursts (FN&TN)

FN&TN -> tx_time

Tx bursts (tx_time)

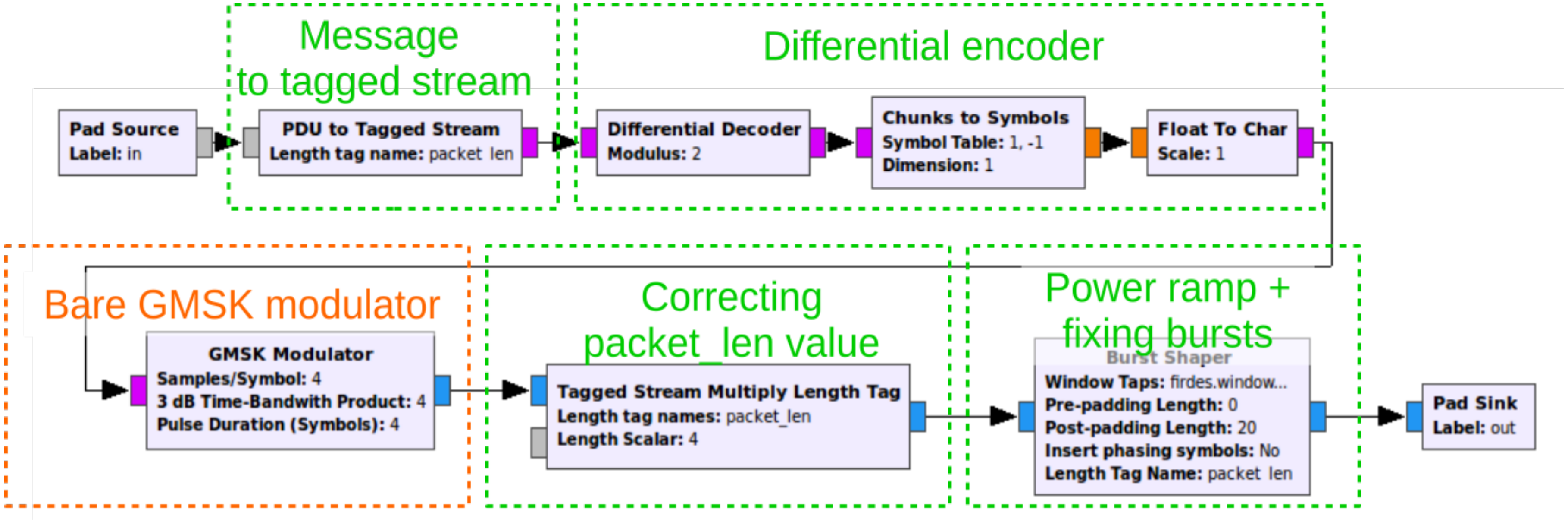
Modulator

Tagged stream (tx_time, length_tag)

SDR Hardware

Modulator with adaptation blocks

- Implemented using blocks available in GNU Radio



Pad Source
Label: in

PDU to Tagged Stream
Length tag name: packet_len

Differential Decoder
Modulus: 2

Chunks to Symbols
Symbol Table: 1, -1
Dimension: 1

Float To Char
Scale: 1

GMSK Modulator
Samples/Symbol: 4
3 dB Time-Bandwith Product: 4
Pulse Duration (Symbols): 4

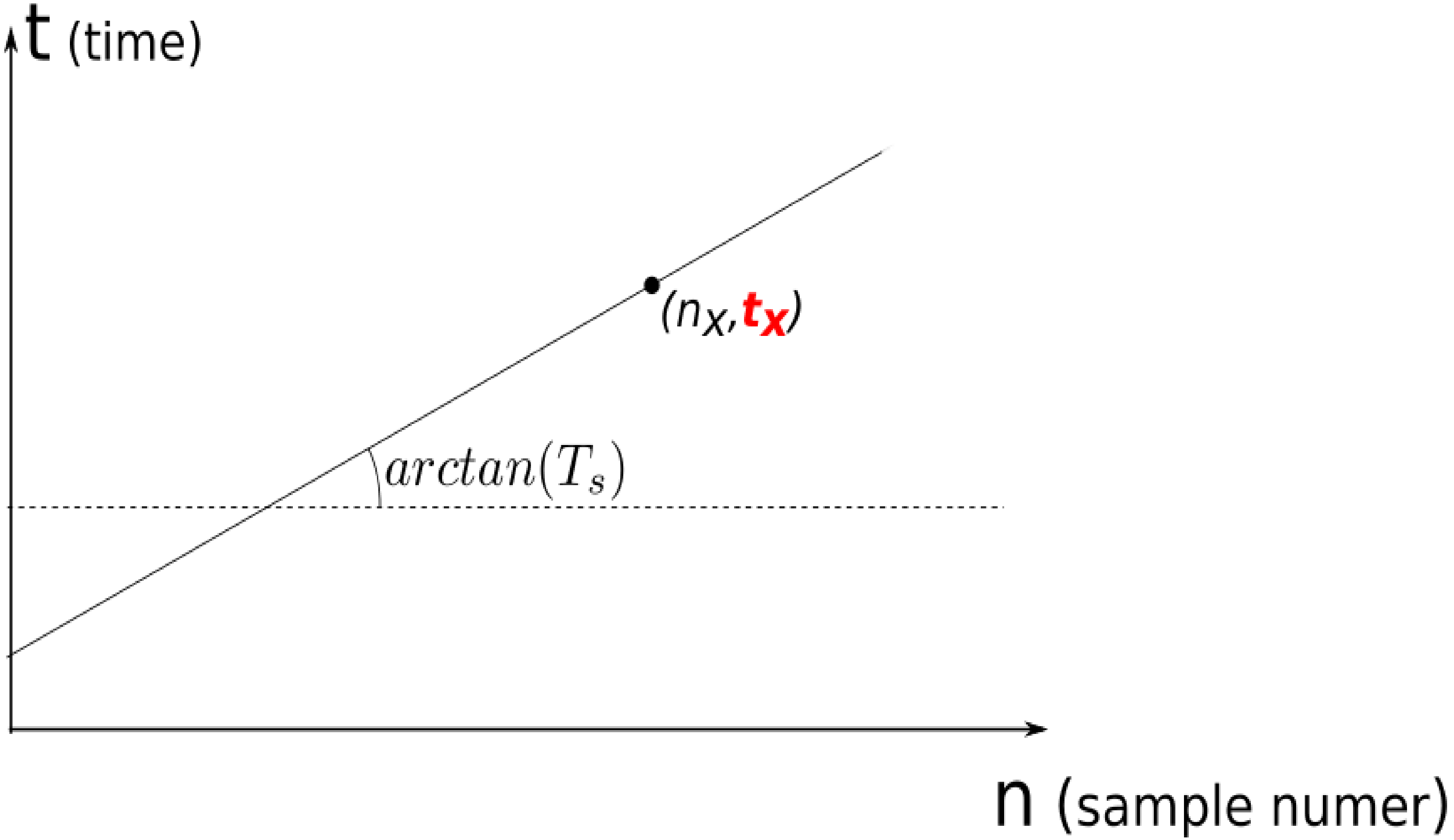
Tagged Stream Multiply Length Tag
Length tag names: packet_len
Length Scalar: 4

Burst Shaper
Window Taps: firdes.window...
Pre-padding Length: 0
Post-padding Length: 20
Insert phasing symbols: No
Length Tag Name: packet_len

Pad Sink
Label: out

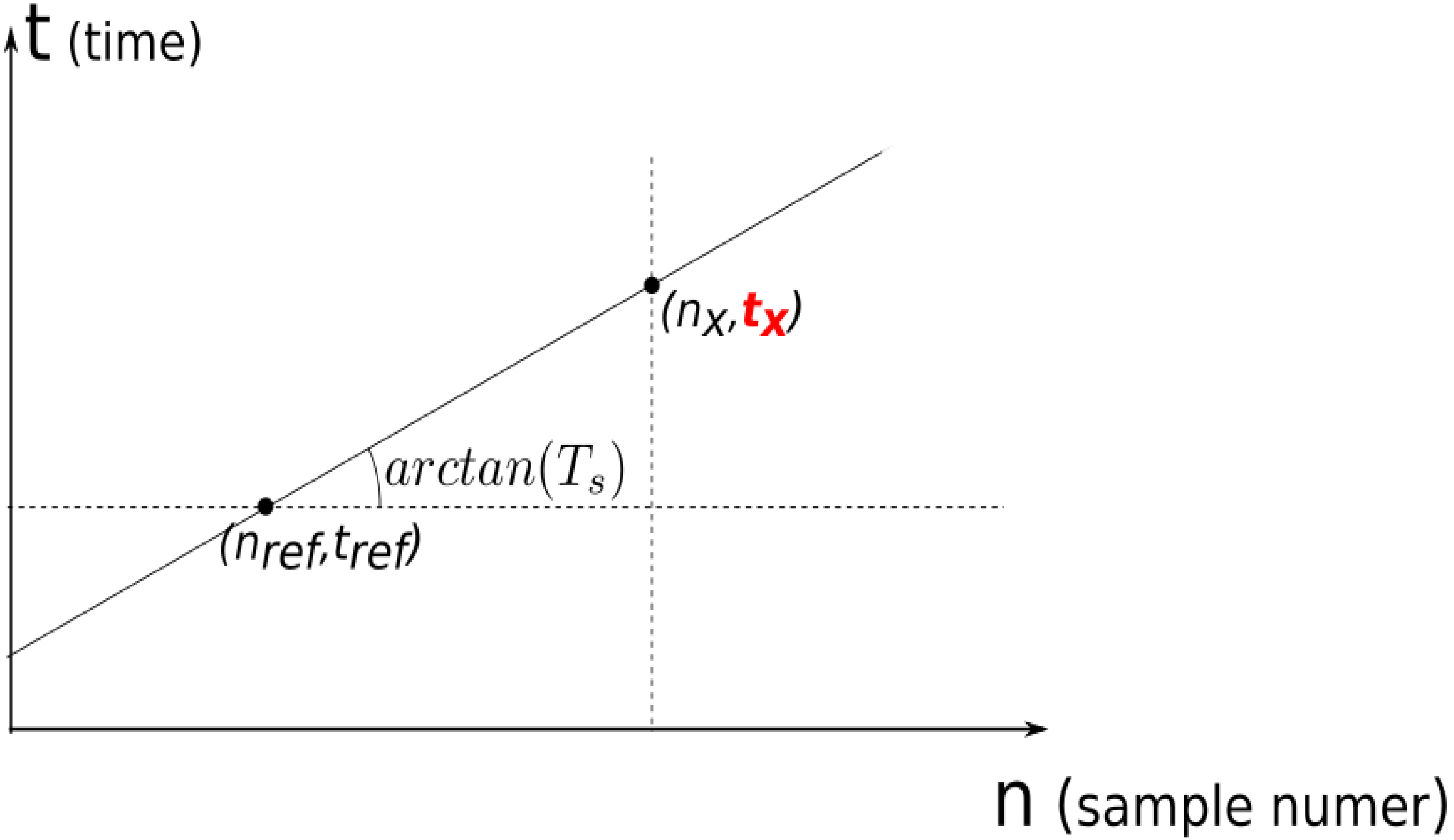
CONVERTING ONE TIME TO ANOTHER

Example: converting number of sample n_x to time t_x (i.e. time since turning on USRP)



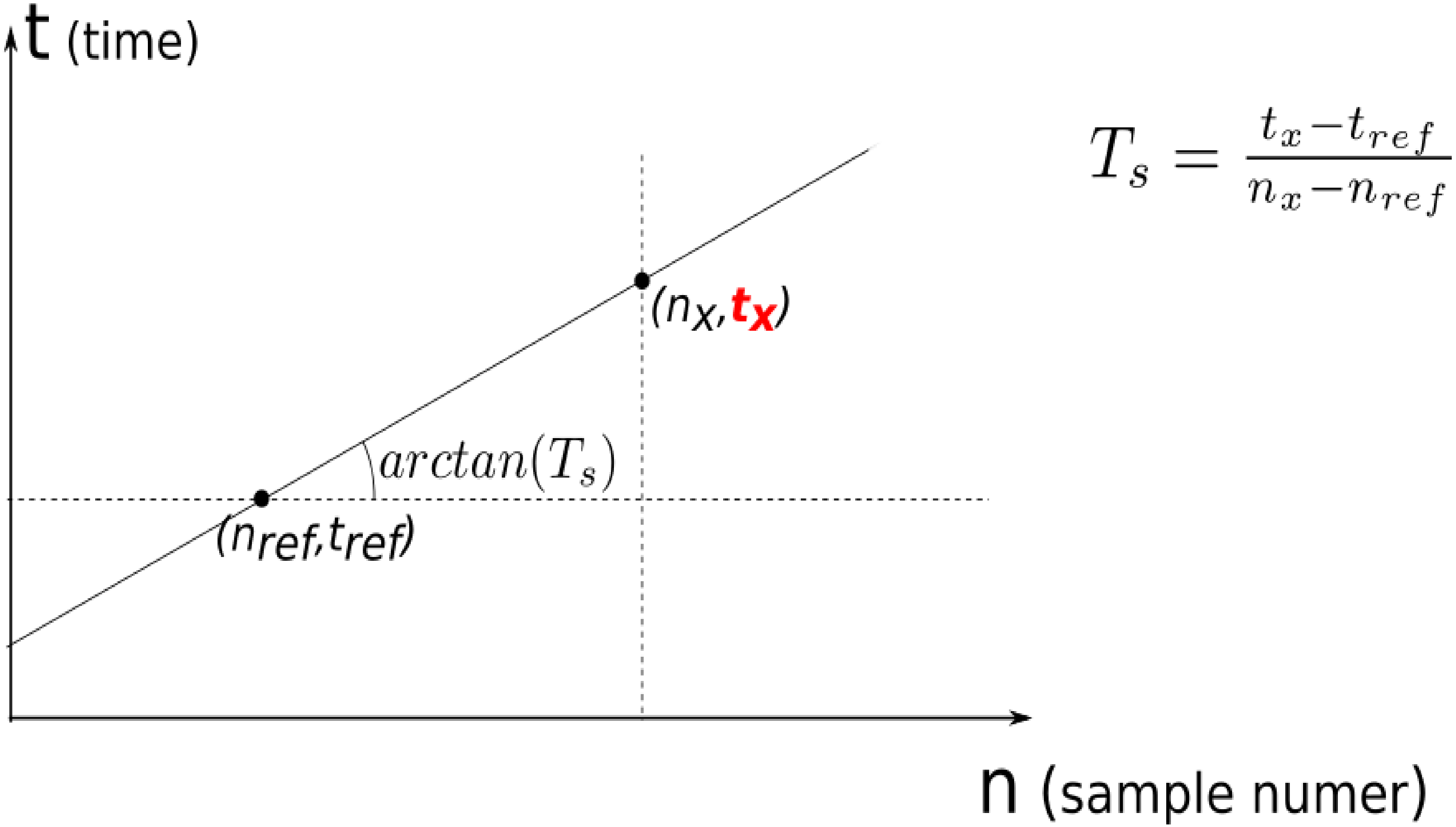
CONVERTING ONE TIME TO ANOTHER

Example: converting number of sample n_x to time t_x (i.e. time since turning on USRP)



CONVERTING ONE TIME TO ANOTHER

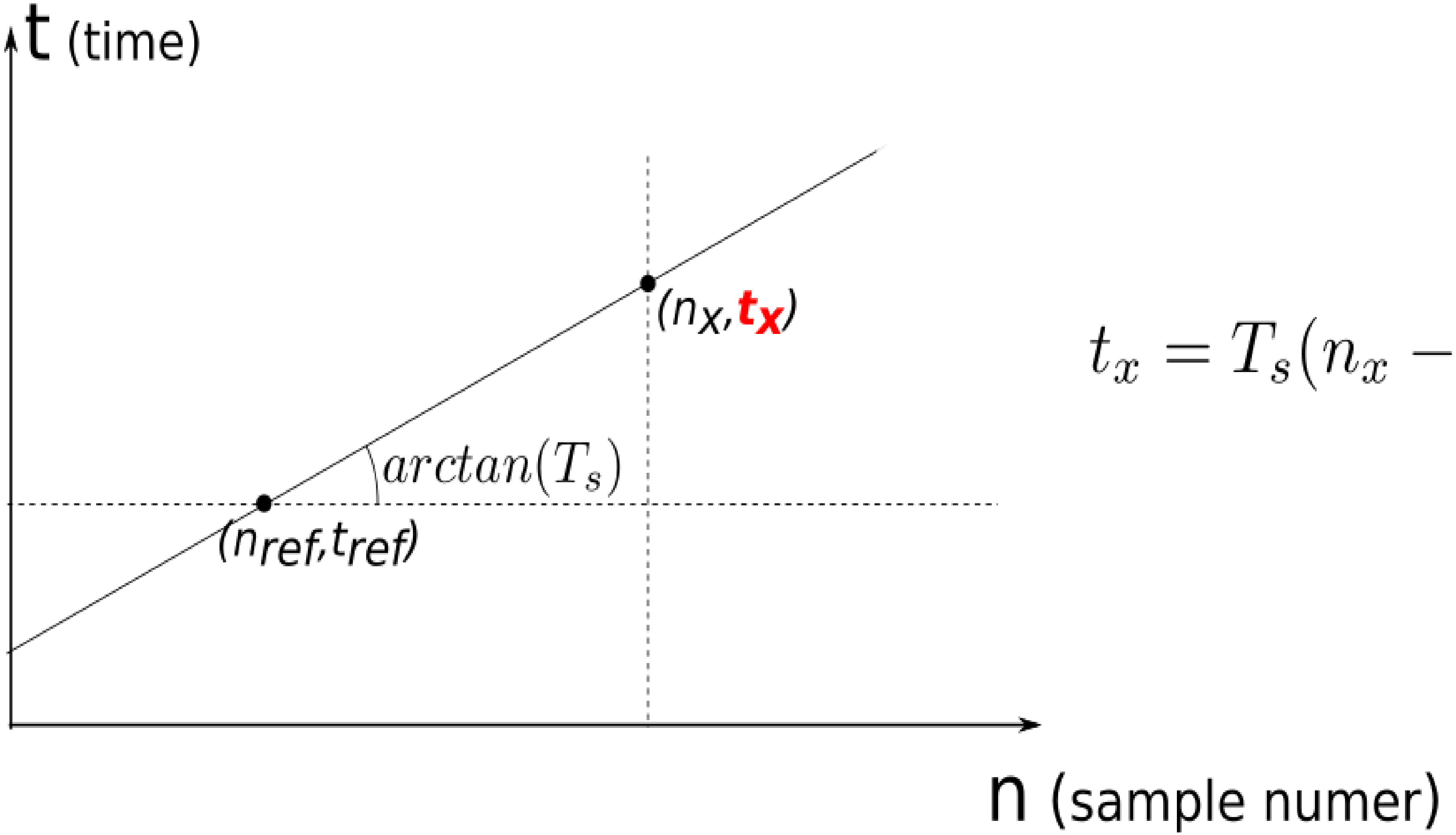
Example: converting number of sample n_x to time t_x (i.e. time since turning on USRP)



$$T_s = \frac{t_x - t_{ref}}{n_x - n_{ref}}$$

CONVERTING ONE TIME TO ANOTHER

Example: converting number of sample n_x to time t_x (i.e. time since turning on USRP)



$$t_x = T_s(n_x - n_{ref}) + t_{ref}$$

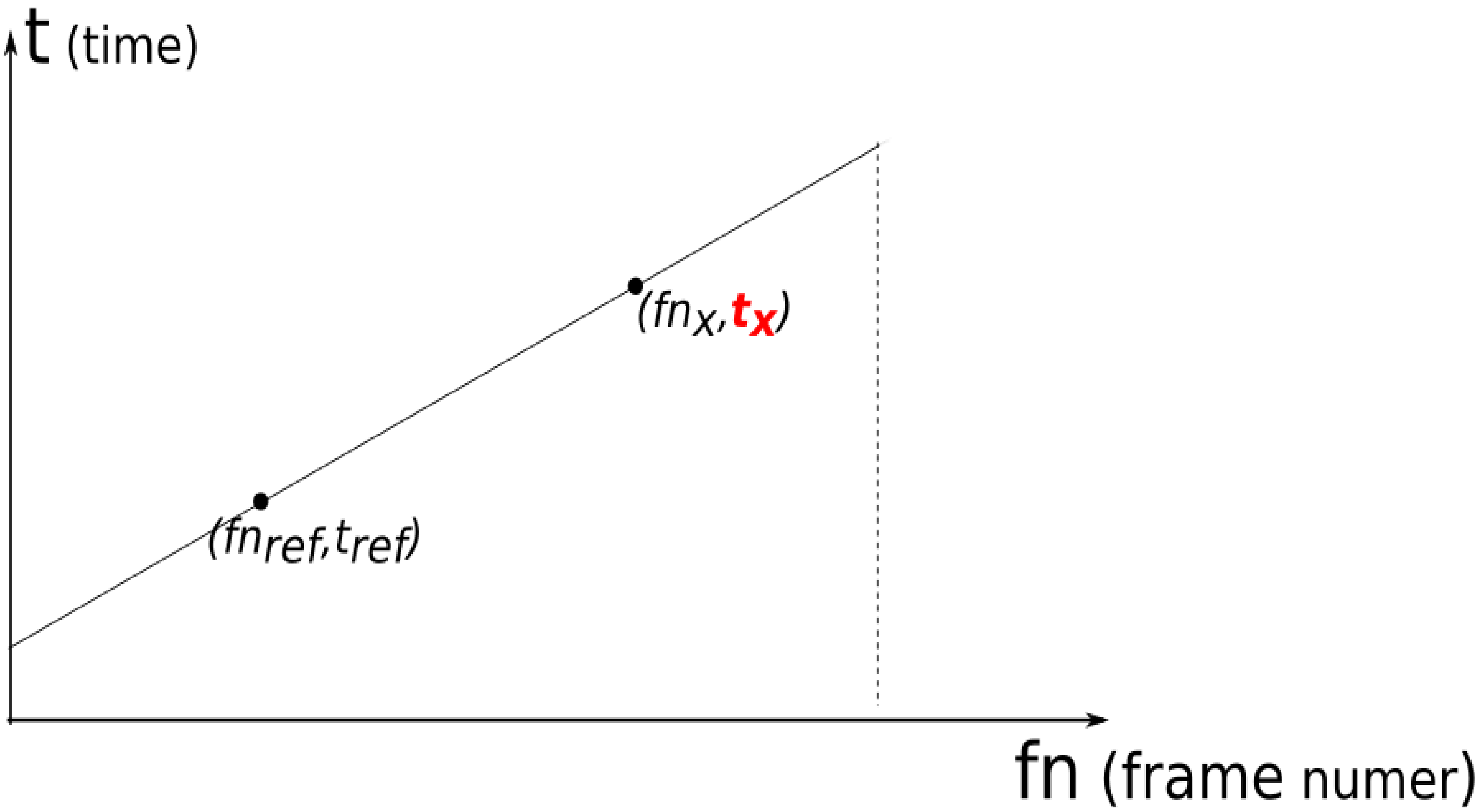
CONVERTING FRAME NUMBER TO TIME

Difficulty with converting frame numbers to time

- Frame number — number modulo hyperframe = 2048*51*26 [frames]
- Repeats every ~3.5 hour
- Computing unambiguous difference between two frame numbers when distance is higher than hyperframe/2

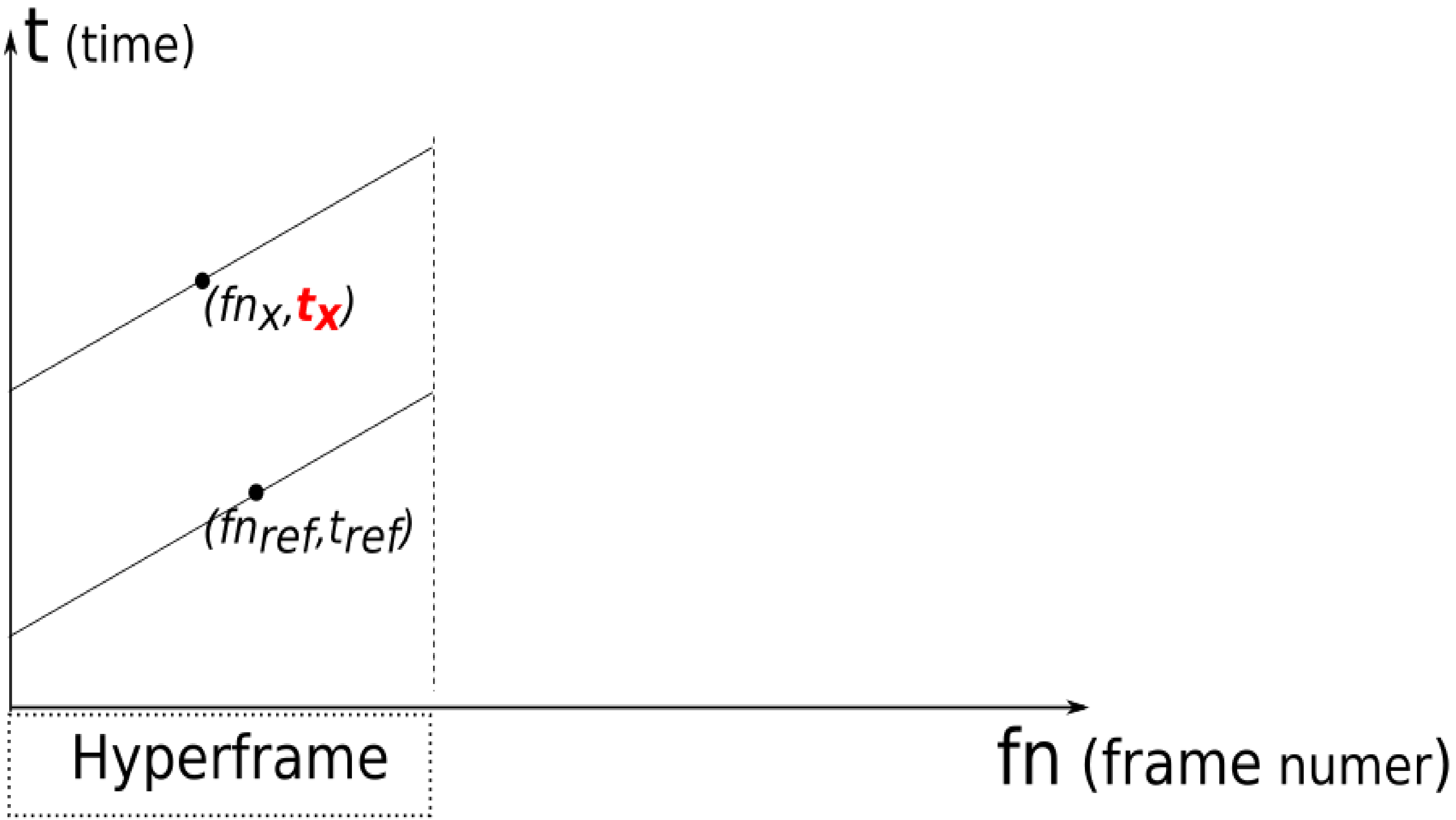
CONVERTING FRAME NUMBER TO TIME

What it would be great to have...



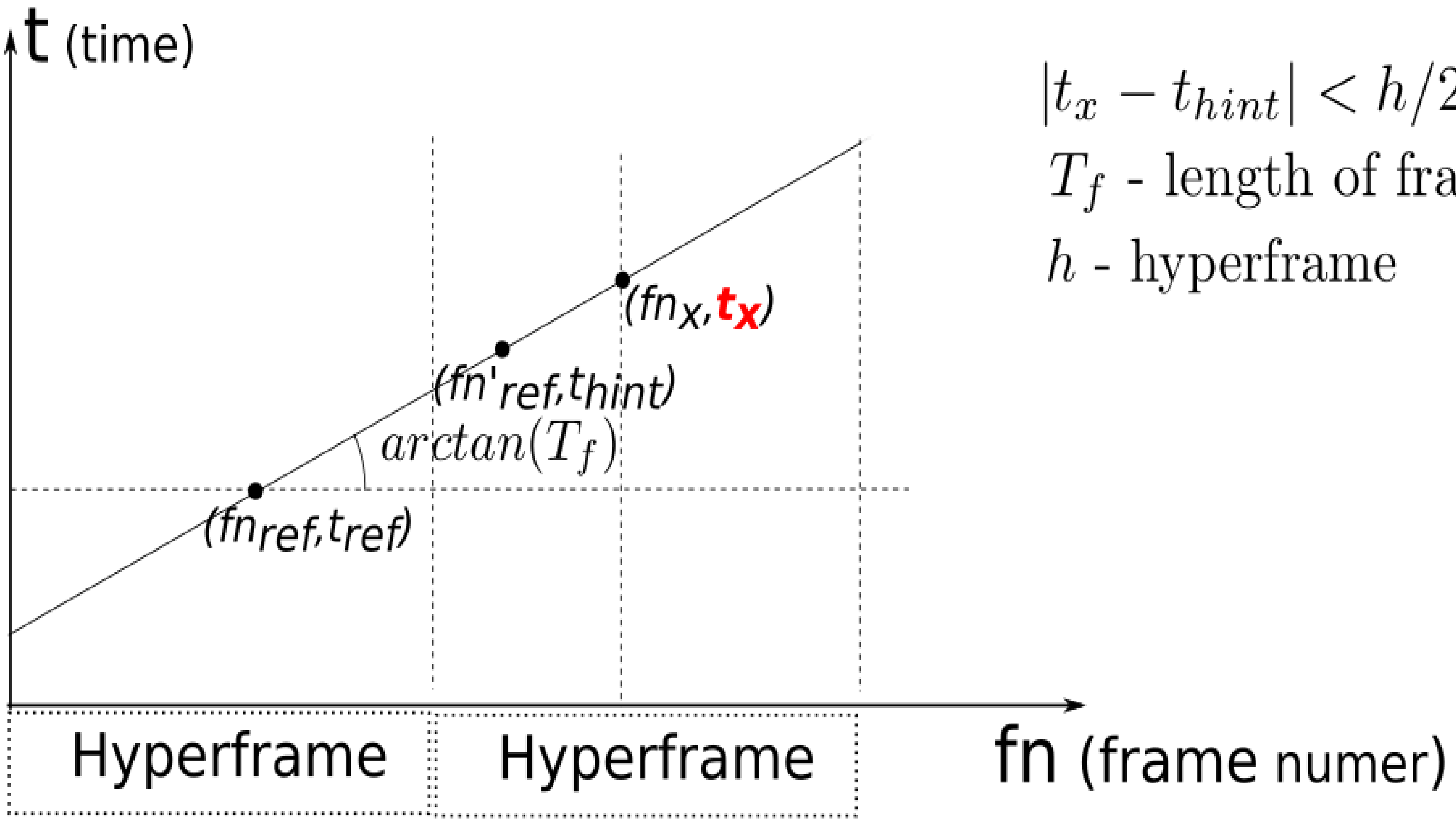
CONVERTING FRAME NUMBER TO TIME

...but this is how it looks like



CONVERTING FRAME NUMBER TO TIME

Let's suppose we have t_{hint} (approximate t_x)...



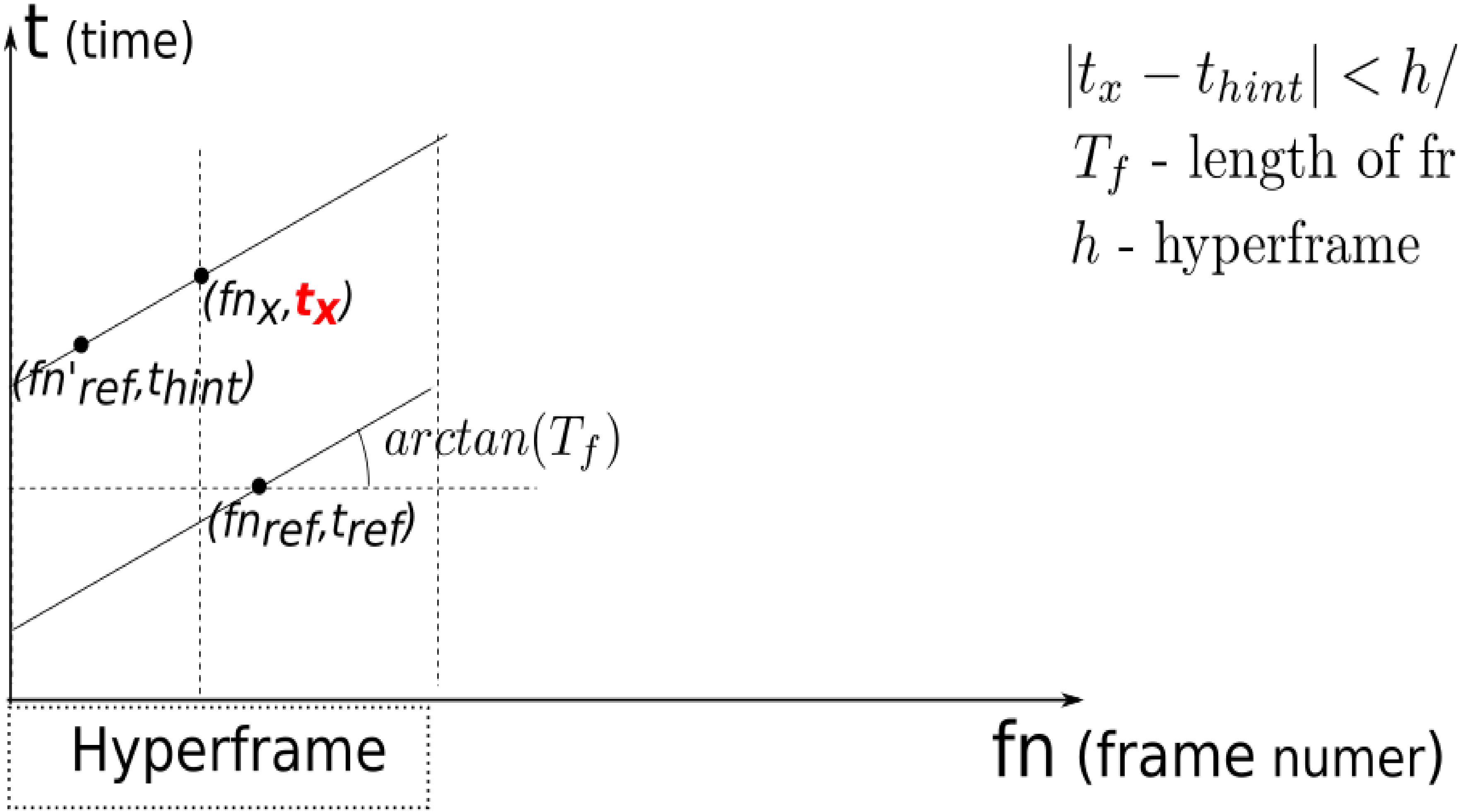
$$|t_x - t_{hint}| < h/2 \cdot T_f$$

T_f - length of frame in [s]

h - hyperframe

CONVERTING FRAME NUMBER TO TIME

...and use it to move (fn_{ref},t_{ref}) closer to (fn_x,t_x)



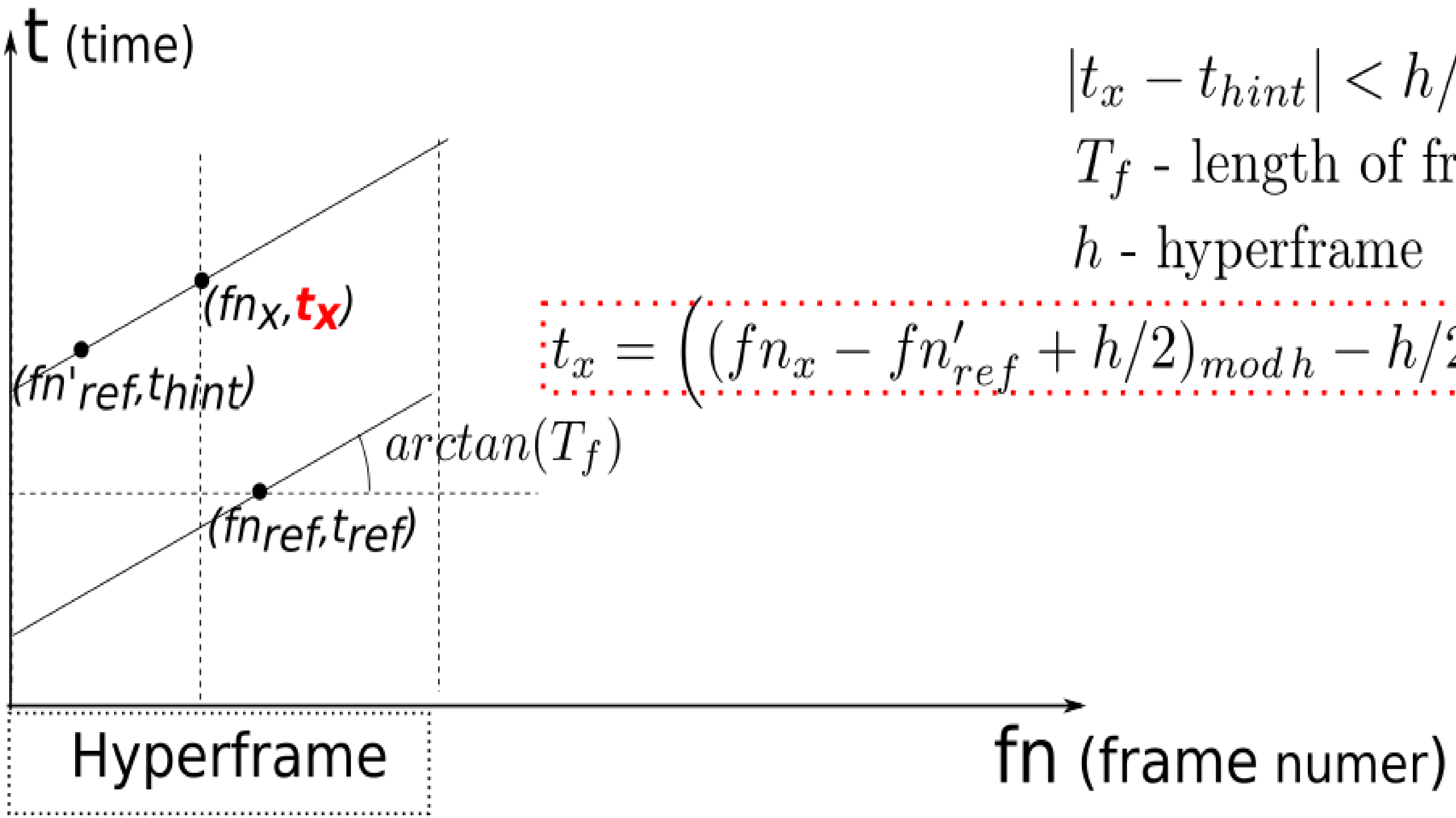
$$|t_x - t_{hint}| < h/2 \cdot T_f$$

T_f - length of frame in [s]

h - hyperframe

CONVERTING FRAME NUMBER TO TIME

Solution...



$$|t_x - t_{hint}| < h/2 \cdot T_f$$

T_f - length of frame in [s]

h - hyperframe

$$t_x = \Big((fn_x - fn'_{ref} + h/2)_{mod\,h} - h/2\Big) \cdot T_f + t_{hint}$$

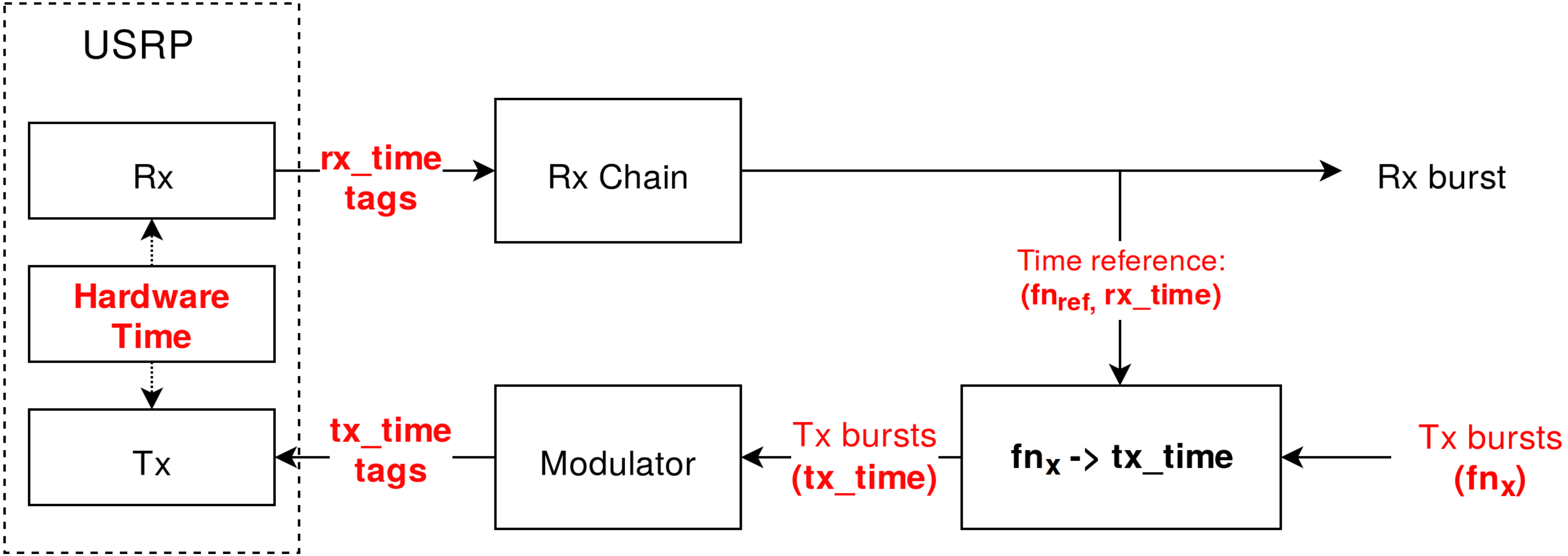
CONVERTING FRAME NUMBER TO TIME

Solution.. that takes into account timeslot numbers (TS)

$$t_x = \Big((fn_x - fn'_{ref} + h/2)_{mod\,h} - h/2\Big)\cdot T_f + t_{hint} + (TS_x - TS_{ref})\cdot TS_{period}$$

CONVERTING FRAME NUMBER TO TIME

Where to get (fn_{ref},t_{ref}) from?



TX FLOWGRAPH DEMO

Demo

Questions?

Piotr Krysik (ptrkrysik@gmail.com)