NTI: A Network Time Interface M-Module for High-Accuracy Clock Synchronization

Martin Horauer

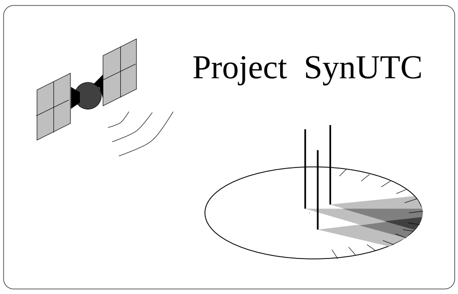
Ulrich Schmid

Klaus Schossmaier

Vienna University of Technology

Dept. of Automation

Dept. of Computer Technology



http://www.auto.tuwien.ac.at/Projects/SynUTC/

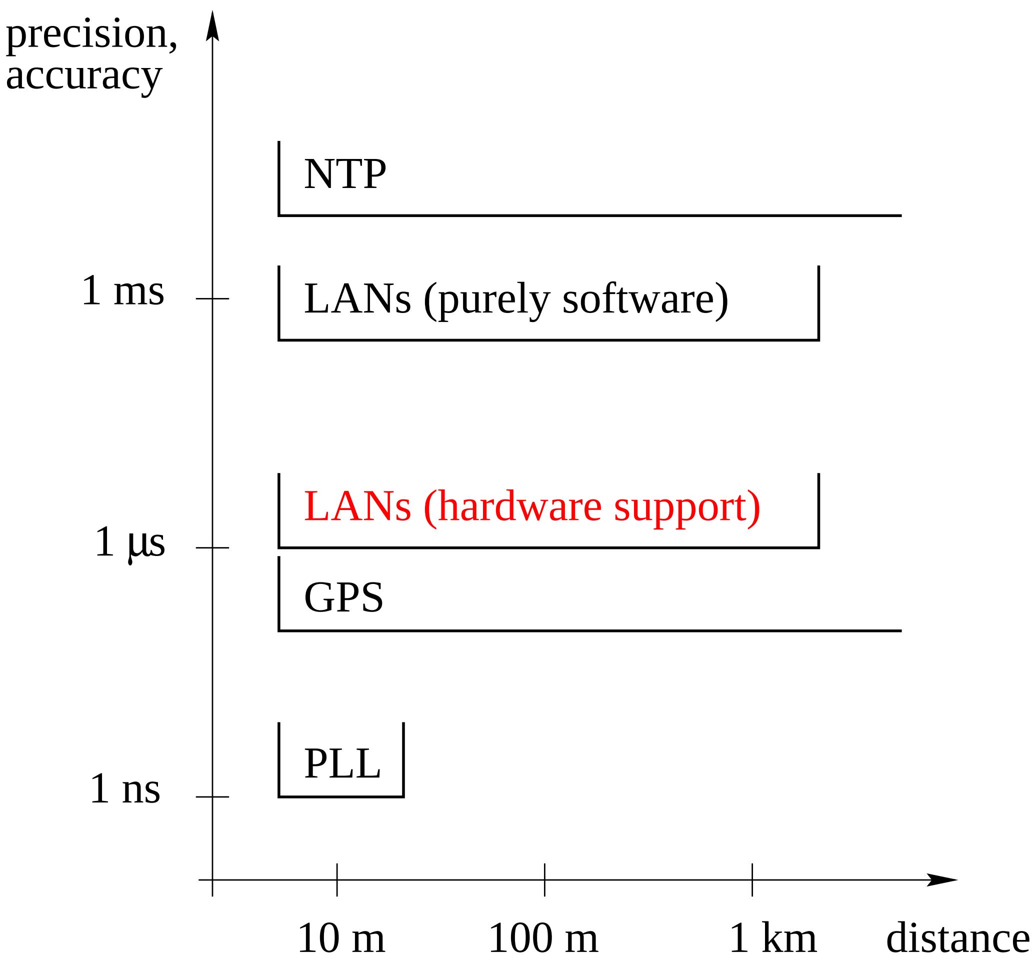
Introduction

Problem: Provide the nodes of a distributed system with

- internal ("precise")
- external ("accurate")

synchronized fault-tolerant clocks

Spectrum of solutions



Our Setting

Distributed system: nodes are

- within a distance of a few 100 meters
- solely interconnected by packet-oriented communication subsystem (shared channel, point-to-point)

Requirements

- fault-tolerant external synchronization
  ⇒ use interval-based clock synchronization
- worst case precision/accuracy in the 1 μs-range
  ⇒ use hardware support
- compatibility with COTS hard- and software
  ⇒ build hardware on top of existing standards
  ⇒ embed software in existing operating system kernel

Interval-Based Clock Synchronization

Replace ordinary clocks $T = C(t)$ by interval clocks

$$\boldsymbol{C}(t) = [C(t) - \alpha^-(t), C(t) + \alpha^+(t)]$$

incorporating

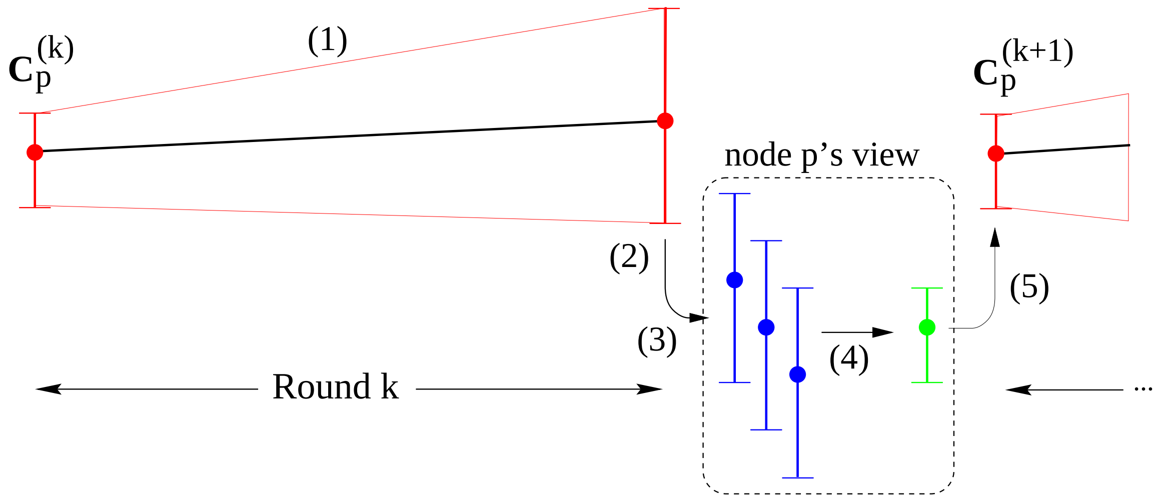
- a usual local clock $C(t)$
  $\Rightarrow$ <u>local time</u> information
- an interval of accuracies $\boldsymbol{\alpha}(t) = [-\alpha^-(t), \alpha^+(t)]$
  $\Rightarrow$ dynamically maintained <u>accuracy bounds</u>

in order to satisfy the

(P) precision requirement: $|C_p(t) - C_q(t)| \leq \pi_{\max}$ for some fixed $\pi_{\max} \geq 0$

(A) accuracy requirement: $-\alpha_p^+(t) \leq C_p(t) - t \leq \alpha_p^-(t)$

Round-based Algorithm



Building blocks

(1) drift compensation

$\Rightarrow$ cope with non-zero clock drifts

(2) periodically exchange timestamped messages

(3) delay compensation

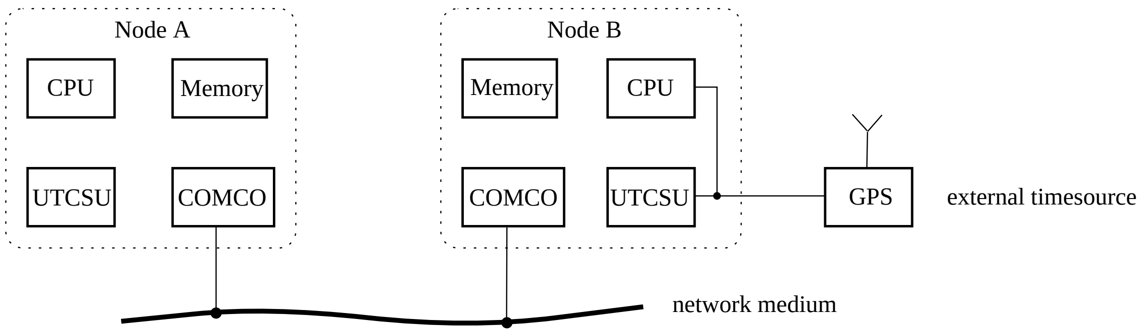
$\Rightarrow$ cope with transmission delays

(4) convergence function

$\Rightarrow$ compute adjustment value

(5) carry out adjustment

Basic Hardware Architecture



Relevant components

- central or dedicated CPU
- DMA-type network communication coprocessor (COMCO)
- shared memory for packet handling
- Universal Time Coordinated Synchronization Unit (UTCSU) = interval clock
- optional external timesources (e.g., GPS receiver)

Clock Synchronization Hardware Support

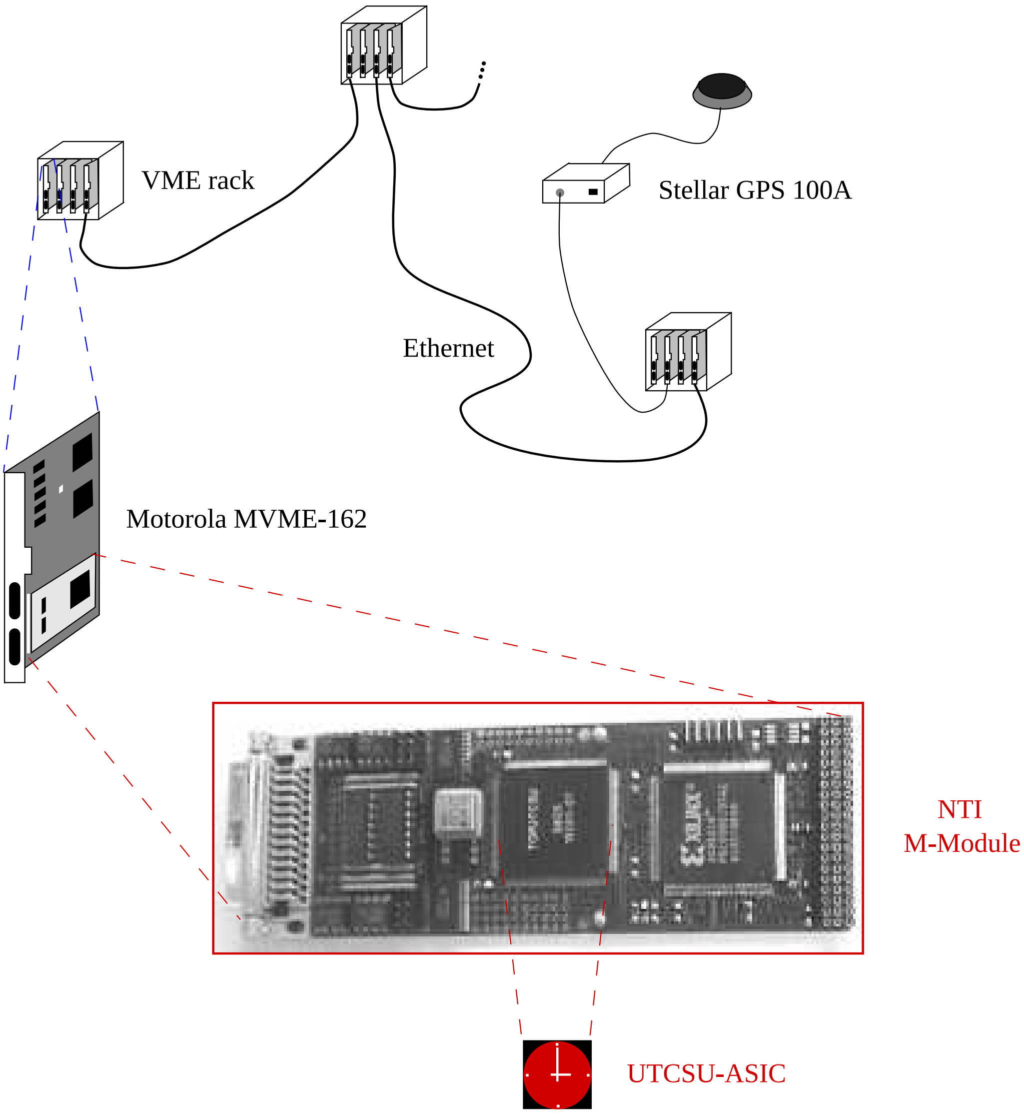
Hardware assistance required for

- rate and state adjustable clock
- maintaining accuracy intervals
- timestamping clock synchronization packets (CSP)
- timer features
- application support

Our solution

- UTCSU-ASIC provides most of the required hardware support for clock synchronization
- NTI M-Module features
  - interfacing to COTS CPUs
  - accurate timestamping for COTS COMCO's

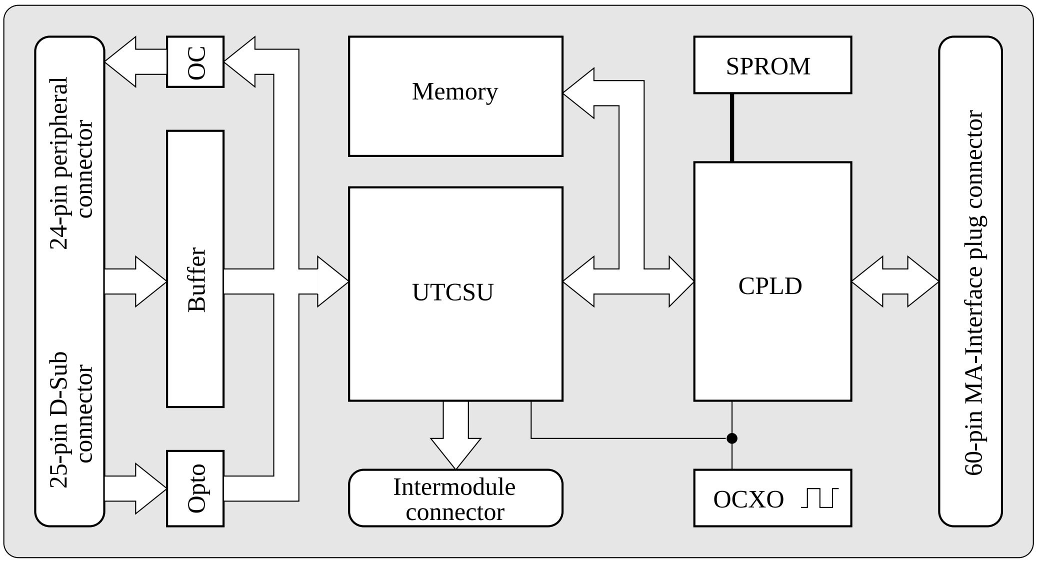
Overview of our Prototype



MA-Module mezzanines

- open bus specification (VITA standard)
- numerous COTS CPU boards with M-Module interface available on the market
- simple, asynchronous bus interface
- 256 bytes I/O space (8, 16 bit accesses)
- 16MB memory space (8, 16, 32 bit accesses)
- small size (146×53 mm), robust unit construction
- connectors

NTI Architecture



Major components

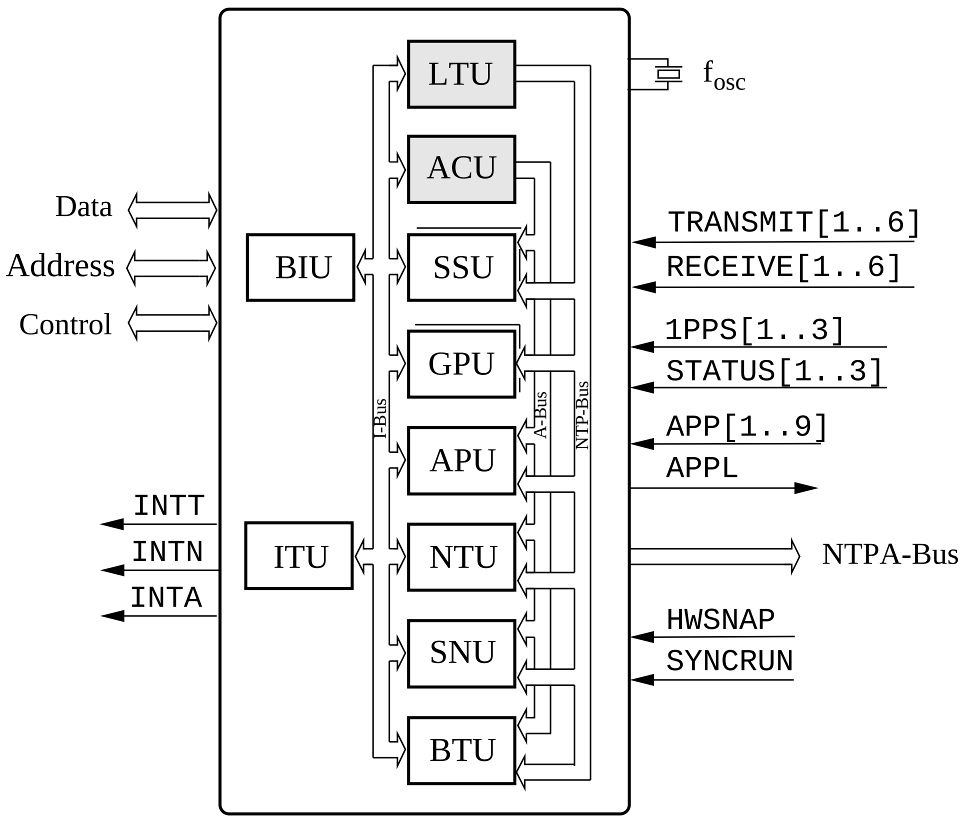
- UTCSU-ASIC
- TCXO or OCXO quartz oscillator
- 512KB SRAM as packet buffer
- CPLD responsible for
  - interfacing the M-Module mezzanine bus
  - special memory addressing/timestamping features

Features of the UTCSU-ASIC (1)

Chip data

- 0.7μm CMOS
- 105 mm^2 die size
- 208 pin MQFP package
- manufactured by Atmel-ES2 (Europractice)

Block diagram of the UTCSU



Features of the UTCSU-ASIC (2)

Local Time Unit (LTU)

- 56 bit NTP time format
  - 32 bit timestamp with $2^{-24} \approx 60$ ns granularity
  - 24 bit macrostamp + 8 bit checksum
- novel adder-based clock design, featuring
  - arbitrary oscillator frequency $1 \ldots 20$ MHz
  - fine-grained rate adjustment
  - state adjustment via continuous amortization
  - optional leap second insertion/deletion

Accuray Unit (ACU)

- two 16 bit accuracy values
  - automatic deterioration
  - no wrap-around

Features of the UTCSU-ASIC (3)

Atomic time/accuracy-stamping

- dedicated TTL trigger inputs
- 32 bit registers for sampled time/accuracies
- 24 independent facilities for
  - CSP timestamping (SSUs)
  - interfacing to 1PPS timing output (GPUs)
  - application-specific purposes (APU)
- extensible via 48 bit multiplexed NTPA-Bus

Features of the UTCSU-ASIC (3)

Duty timers

- raise an interrupt or output a signal at pre-programmable times
- used for triggering
  - actions of the clock synchronization algorithm
  - continuous amortization
  - inserting/deleting leap seconds
  - generating application-specific events

Miscellany

- flexible bus interface
- 3 non-vectored interrupt outputs
- test and debugging features

Accurate CSP Timestamping (1)

Well-known result

worst case precision $\pi_{\max}$ at least $\varepsilon(1 - 1/n)$,
for n nodes in a system with maximum
transmission delay uncertainty $\varepsilon = \delta_{\max} - \delta_{\min}$

Several steps contribute to ε

1. sender-CPU assembles the CSP
2. sender-CPU signals sender-COMCO to take over for transmission
3. sender-COMCO tries to acquire the network medium
4. sender-COMCO reads CSP data from memory and pushes the resulting bit stream onto the medium
5. receiver-COMCO pulls the bit stream from the medium and writes CSP data into memory
6. receiver-COMCO notifies receiver-CPU of packet reception via interrupt
7. receiver-CPU processes CSP

Accurate CSP Timestamping (2)

Software-based solution

- perform CSP timestamping at steps 1 and 7
- incorporates
  - medium access uncertainty ($3 \rightarrow 4$)
  - any variable network delay ($4 \rightarrow 5$)
  - reception interrupt latency ($6 \rightarrow 7$)

Ideal hardware-based solution

- CSP timestamping when, say, first byte is pushed on resp. pulled from medium
- requires support from inside the COMCO
  $\Rightarrow$ not available for COTS ones

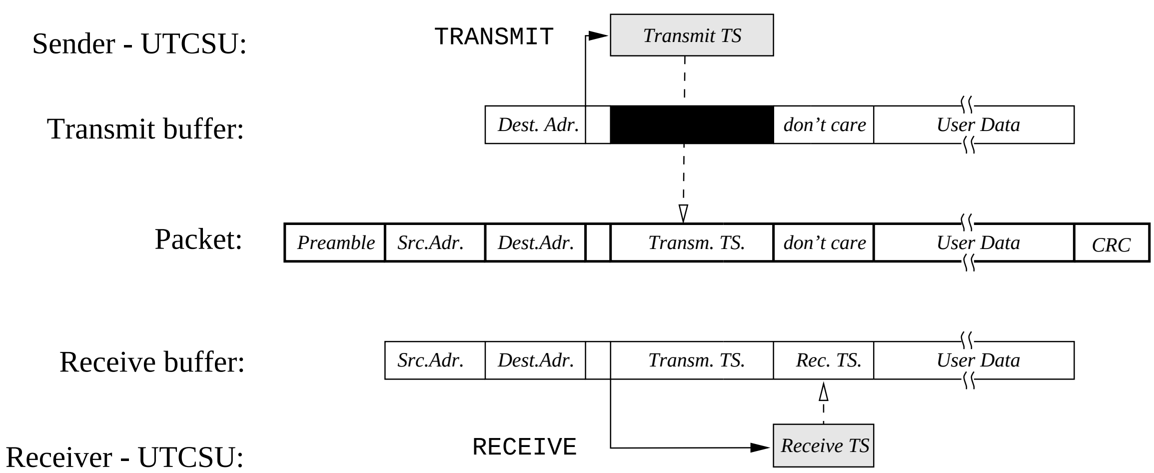
Accurate CSP Timestamping (3)

DMA-based coupling

- sample a timestamp when a certain byte within the transmit resp. receive buffer for a CSP is read resp. written by the COMCO
- transparently map the sampled transmit timestamp into a (successive) portion of the transmit buffer

⇒ timestamping now at steps 4 and 5

Example



Accurate CSP Timestamping (4)

Moving receive timestamps into data packets

- by the CPU within the "Receive Signal ISR"
  - invoked by the NTI
  - need Receive Header Base register
- by the CPU within the "Packet Reception ISR"
  - invoked by the COMCO
- by the COMCO
  - transparent mapping techniques
- by an additional DMA controller

Accurate CSP Timestamping (5)

Restrictions

- does not work for COMCOs with on-chip packet buffer
- internal FIFO buffers are harmful to ε
- evaluation of ε usually needs experimental evaluation

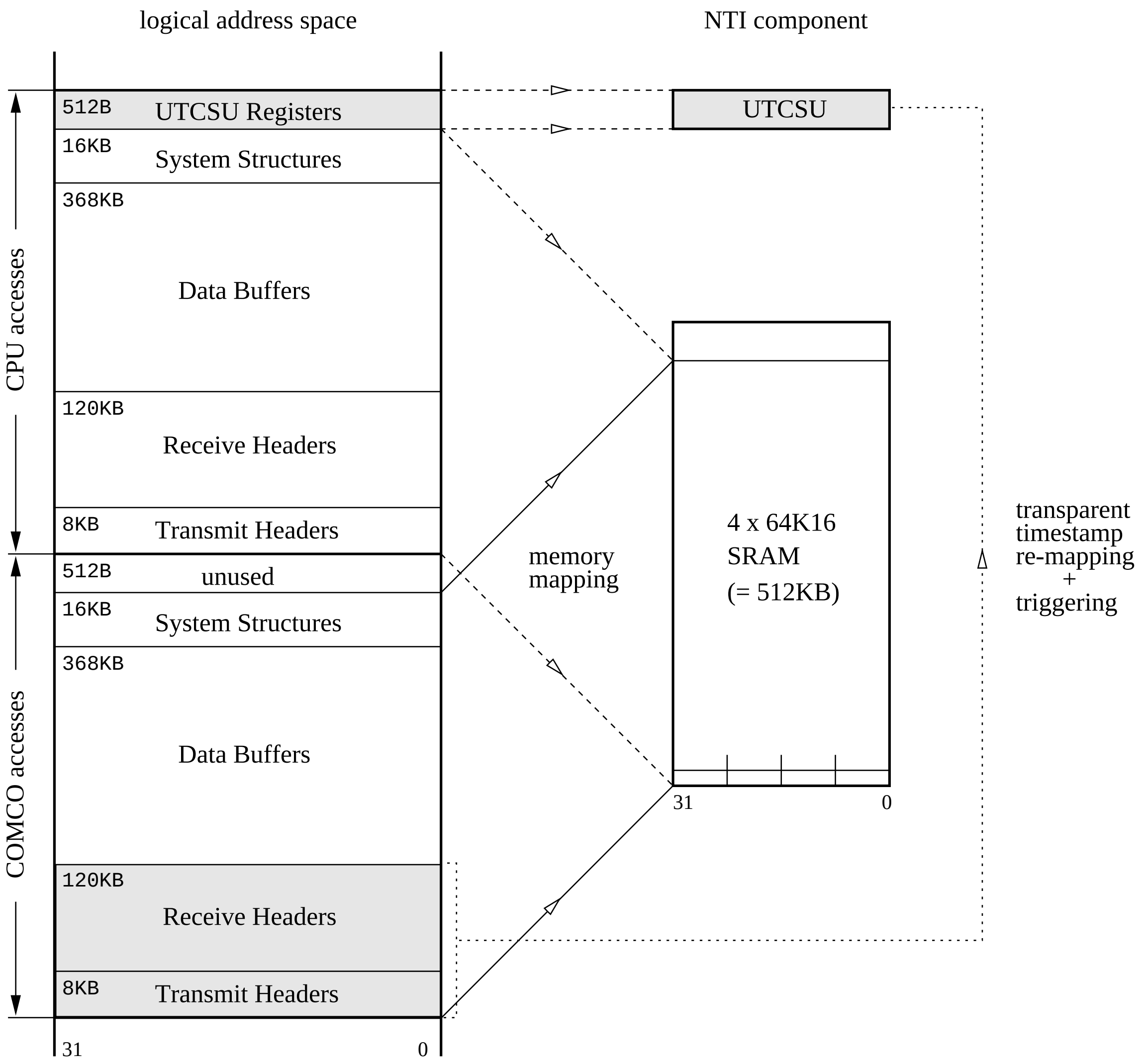
Current NTI design

- targeted for Motorola MVME-162
  - M68040 CPU
  - Intel's 82596CA Ethernet coprocessor
  - passive VME M-Module carrier boards
- used for building up an evaluation system of 16 nodes

$\Rightarrow$ preliminary measurements yielded $\varepsilon \leq 200$ ns

NTI Hardware/Software Interface

M-Module's memory space

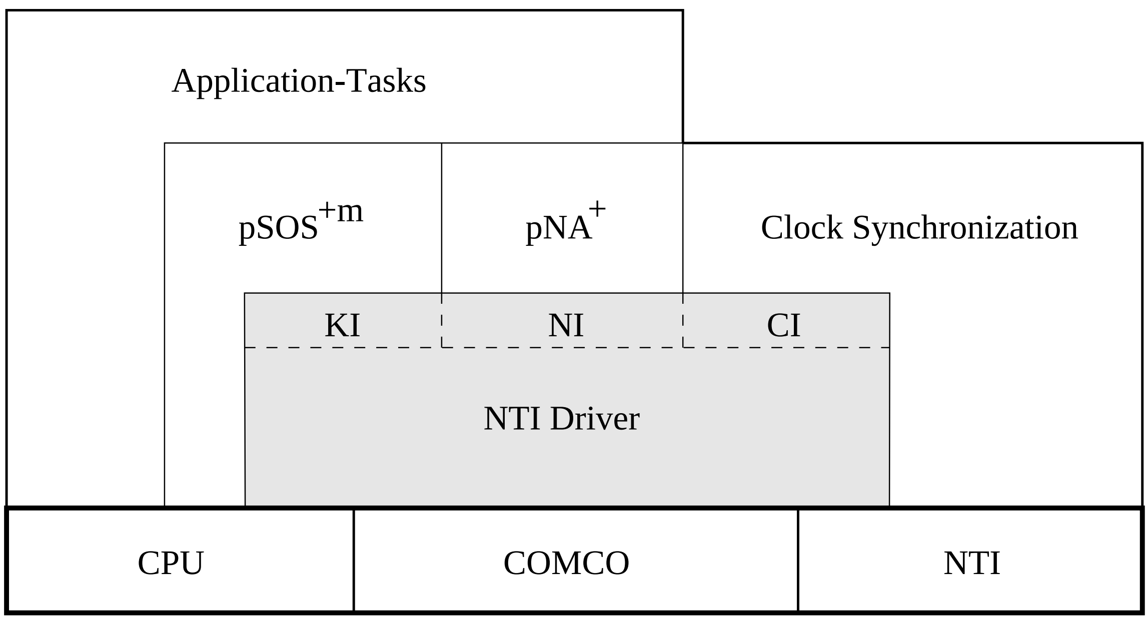


Integration into a real-time system kernel

Adding synchronized clocks to pSOS^{+m}

- clock synchronization
  - must be transparent for application software
  - may generate some CPU and network load
- local time/accuracy available through application support of NTI/UTCSU

Basic system software architecture



Conclusions

Benefits of the NTI

- 1μs external fault-tolerant clock synchronization
- works with COTS
  - CPUs with M-Module interfaces
  - COMCOs with DMA capabilities
  - GPS receivers
  - real-time operating system kernels

Future work

- algorithmic/theoretical issues concerning external clock synchronization
- experimental evaluation including a transition to another target hardware (AcQ's i6040)
- pilot industrial application (on-line fault location for underground power cables)