Keynote ISPCS'07

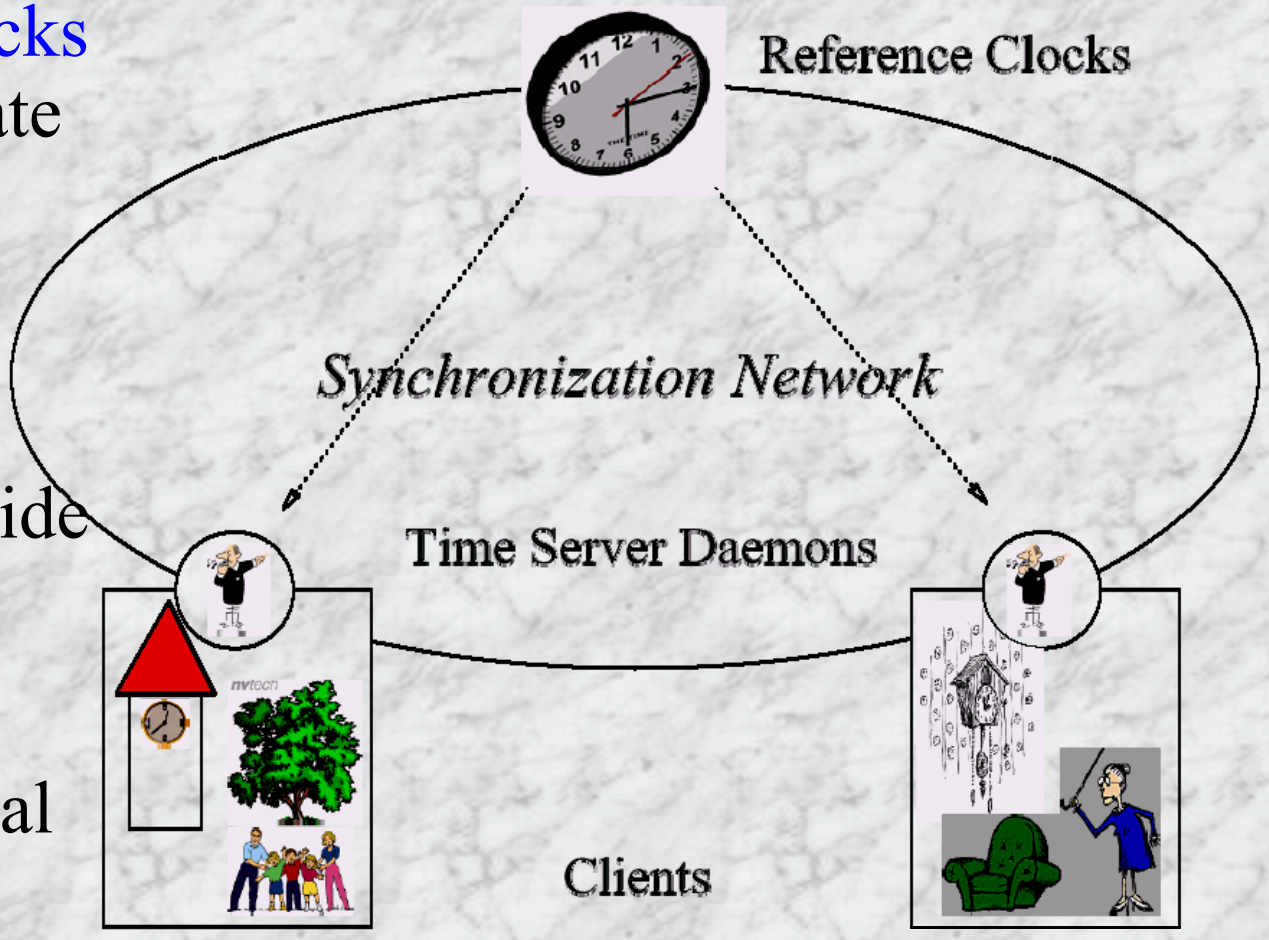
A Perspective of Fault-Tolerant Clock Synchronization

Ulrich Schmid
Technische Universität Wien
s@ecs.tuwien.ac.at

Contents

Time Services

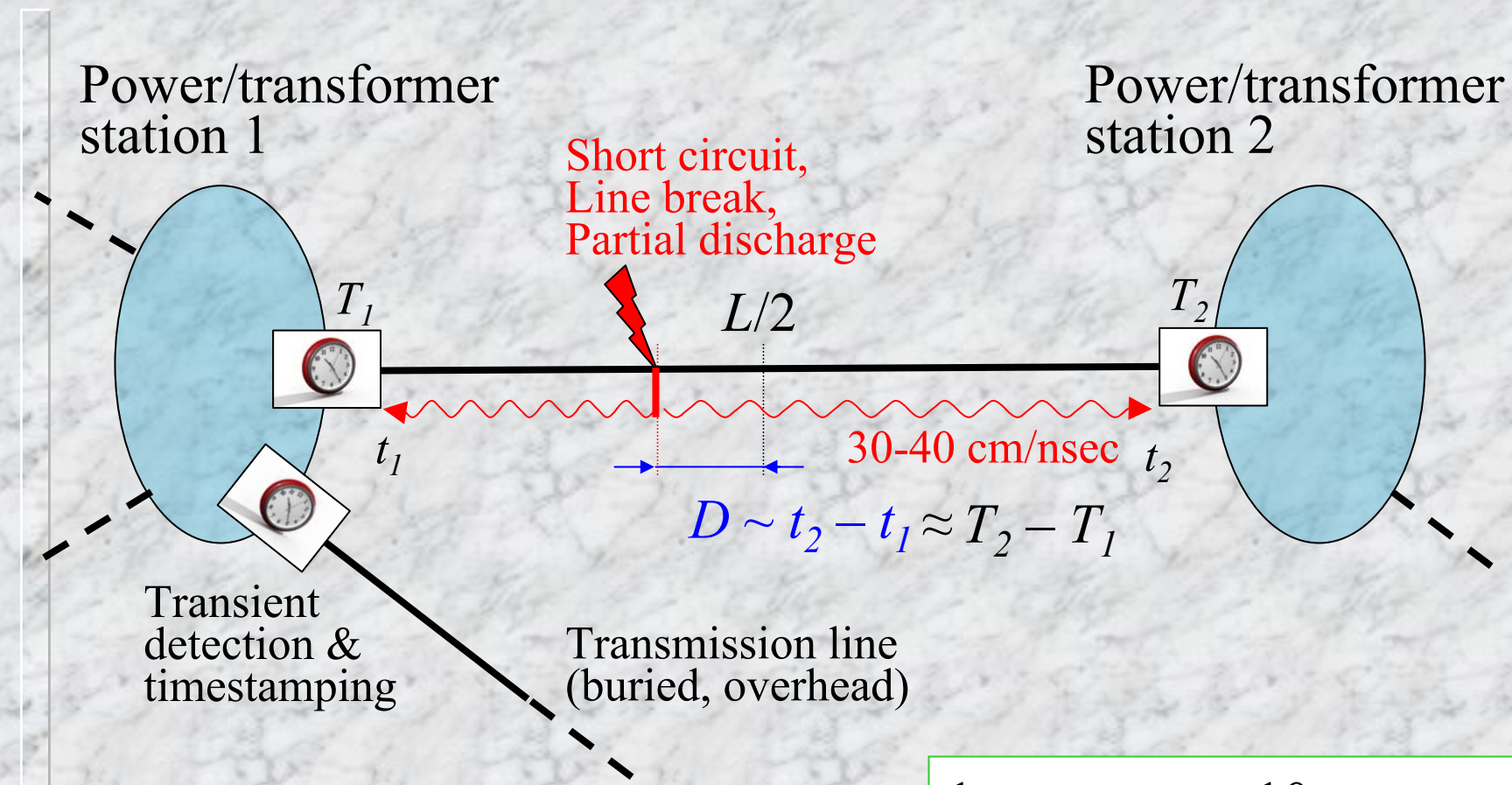
- Reference Clocks provide/generate reference time
- Time Server Daemons provide synchronized local clocks
- Clients use local clocks only



Many Applications need a TS [Lis93]

- Meaningful file creation times in P2P networks
- Data fusion & wakeup synchronization in sensor networks
- Synchronous distributed algorithms
- Communication in high-speed Systems-on-Chip
- …

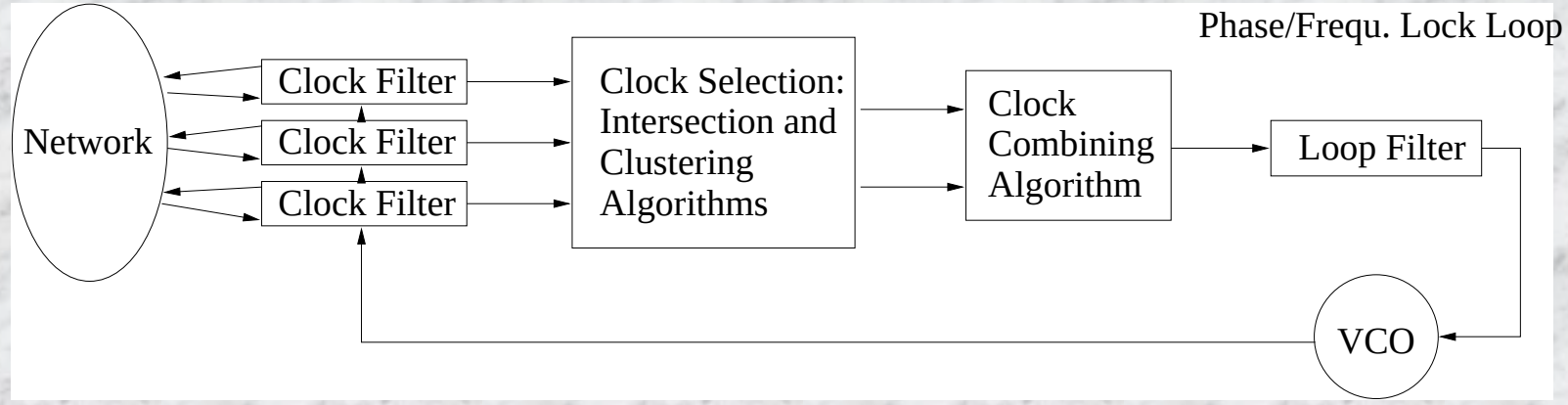
Example: Power Grid Monitoring



1 m range → 10 ns range

Standard TS solutions …

Network Time Prototol (NTP) 1/2



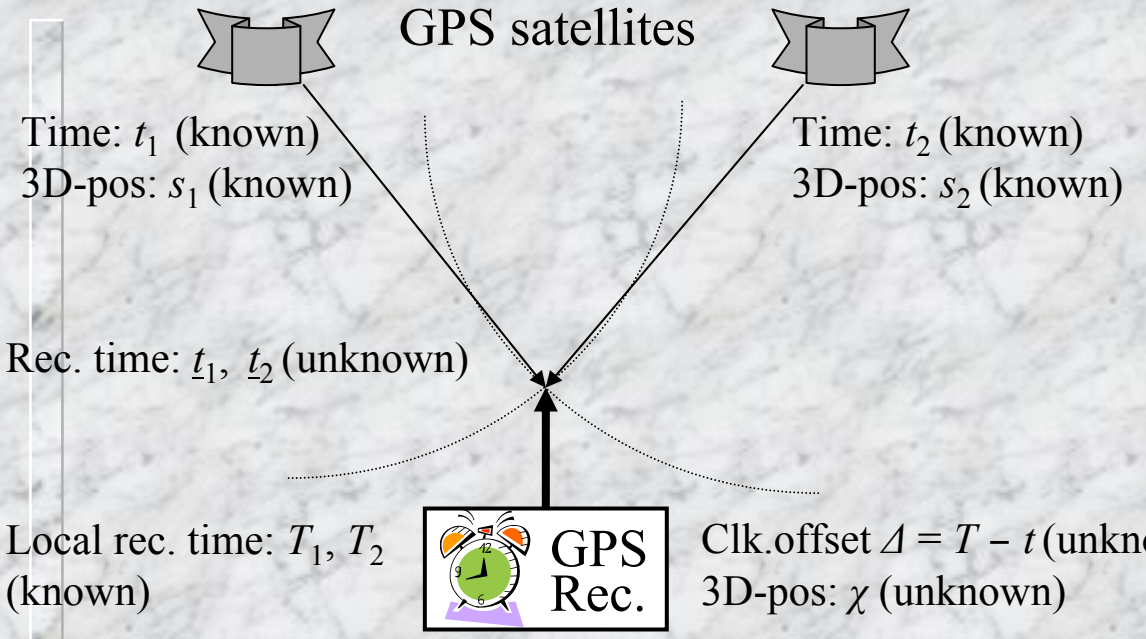
- Distributed maintainance of a minimum-weight spanning tree of hierarchical Reference Clocks
  - Periodically read time from higher “stratum”-level Reference Clocks and adjusts local clock accordingly
  - Employ well-engineered statistical algorithms for data filtering and clock selection [Mil95]

Network Time Prototol (NTP) 2/2

Score of NTP:

+ Readily available, standardized and field-proven
- Designed for ms-range accuracy
- Complex algorithms designed for “bad” (asynchronous) settings like the Internet →
  - complex error modes
  - [probably] sub-optimal behavior in “nicer” (more synchronous) systems

Global Positioning System (GPS) 1/2



- Satellite clocks synchronized to USNO atomic master clock
- GPS-Receiver solves system of equations

$$t_i + |\chi - s_i|/c + \Delta = T_i$$

- 4 satellites required to determine (x, y, z) and Δ
- 1 satellite sufficient for Δ if (x, y, z) is known
- Receiver Autonomous Integrity Monitoring (RAIM)

Global Positioning System (GPS) 2/2

Score of GPS timing receivers:

+ Excellent accuracy (100 ns and below)
+ Usually quite reliable operation [HS97]: 10^{-6}
- Every GPS receiver needs a dedicated (rooftop) antenna
- Very complex overall system → Exhaustive assessment of all error modes impossible
- Possibly large delay until correct time is provided

NTP, GPS, 1588 no Panacea …

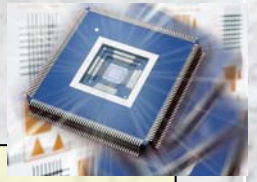
For certain applications, standard TS solutions

- are insufficient in terms of synchronization accuracy
- are too heavy-weight
- are too complex to be rigorously proved correct
- are inacceptably dependent on the correct operation of dedicated Reference Clock(s)

Example: Systems-on-Chip 1/2

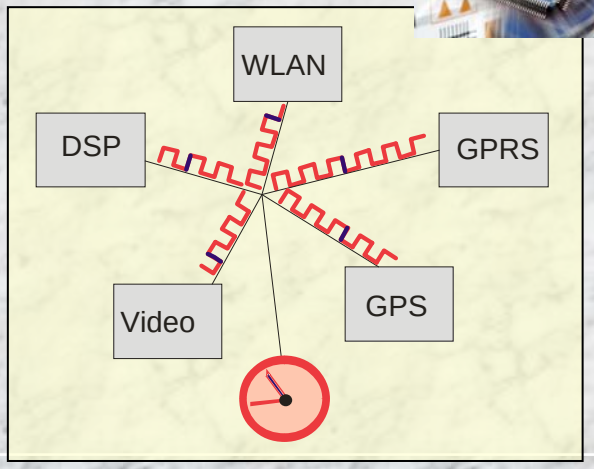
Classic synchronous paradigm:

- ***Concept:*** Common notion of time for entire chip
- ***Method:*** Single crystal oscillator
  Global, “phase accurate” clock tree



Costs:

- Cumbersome clock tree design
  (physical limits!)
- High power consumption
- Clock is single point of failure!

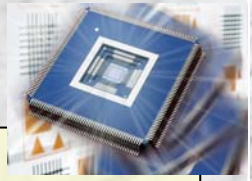


Example: Systems-on-Chip 2/2

Alternative: Fault-tolerant DARTS clock [FSFK06]

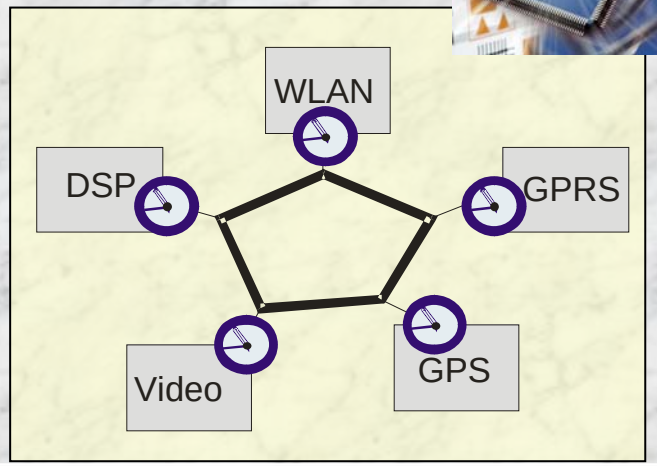
- ***Concept:*** Multiple synchronized tick generators
- ***Method:*** Distributed FT tick generation algorithm
  Implemented in (asynchronous) HW

http://ti.tuwien.ac.at/ecs/research/projects/darts



Result:

- Reasonable synchrony
- Uncritical clock distribution
- Clock is no single point of failure!

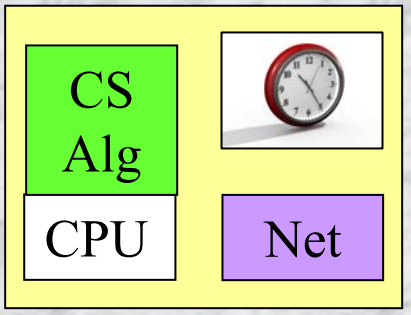


Elementary Fault-tolerant Clock Synchronization

Clock Synchronization Problem 1/3

Node i

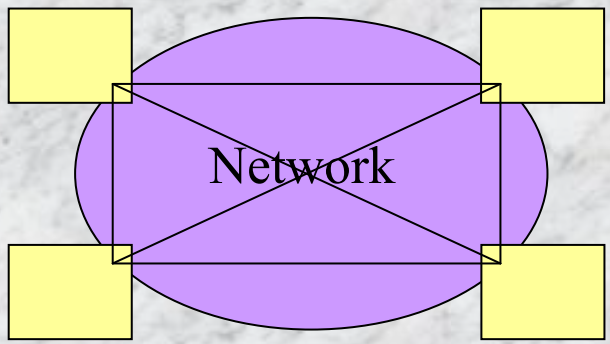
[Deterministic]



Clock C_i: $t \rightarrow T$ [adjustable], max. clock drift ρ SEC/sec

End-to-end delay $\in [d, d+\varepsilon]$

System of n nodes



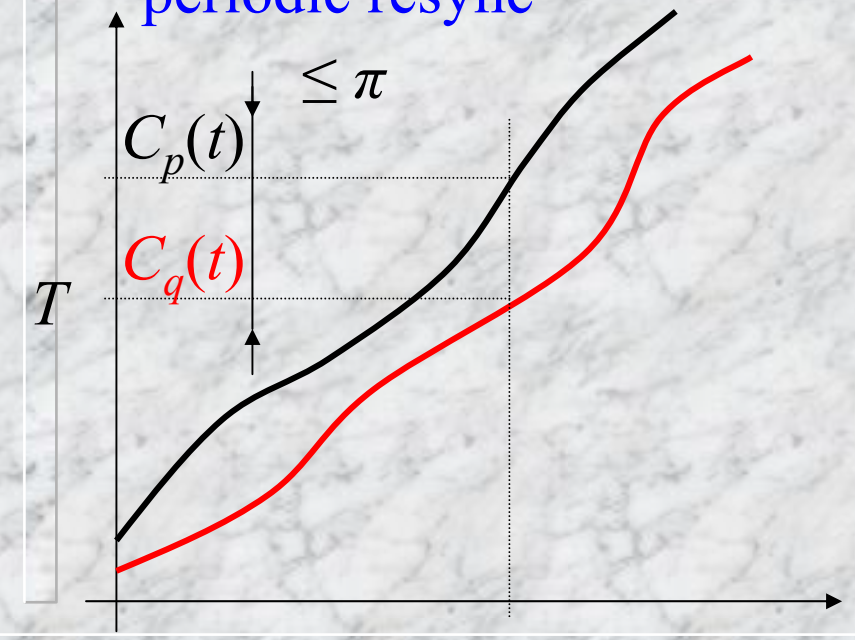
[fully connected]

- Components can fail according to some failure model, ranging from
  - crash
  - arbitrary ("Byzantine")

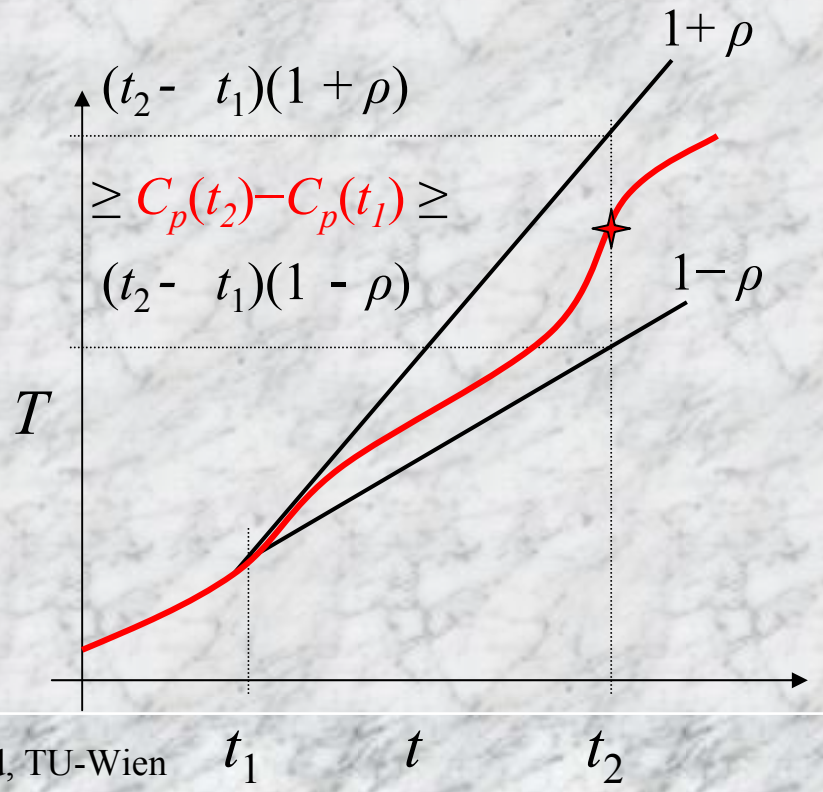
Clock Synchronization Problem 2/3

Internal clock synchronization with precision π

- $|C_p(t) - C_q(t)| \leq \pi$, for all non-faulty p, q and all $t \geq 0$
- Clocks drift apart → need periodic resync

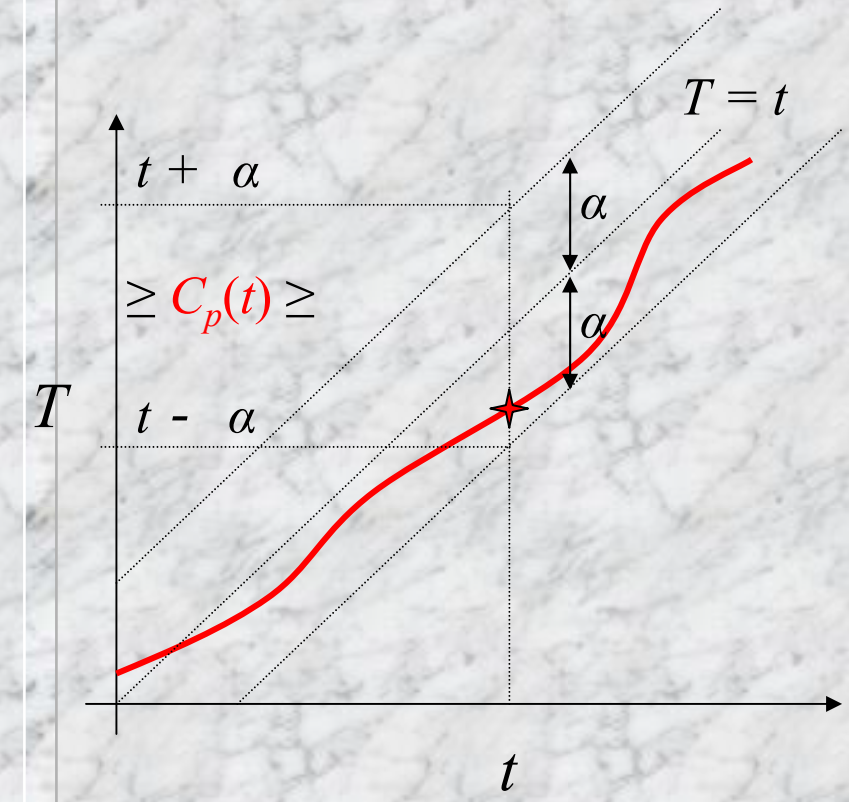


- Linear envelope [bounded drift ρ]



Clock Synchronization Problem 3/3

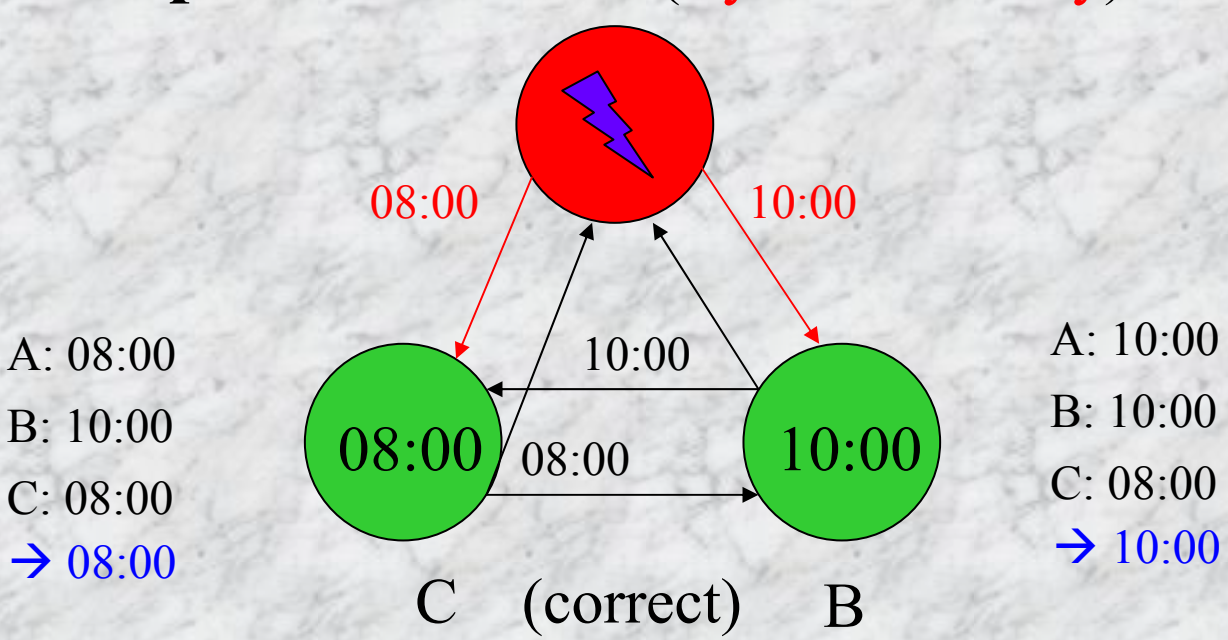
External clock synchronization with accuracy α



- $|C_p(t) - t| \leq \alpha$ for all non-faulty nodes p and all $t \geq 0$
- External CS with accuracy α $\rightarrow$ internal CS with $\pi = 2\alpha$

Why do Failures hurt (so much) ?

Toy example:



- B and C never get closer together
- [DHS86]: $n = 2f + 1$ not enough for f Byz. failures!

Generic Int. Clock Sync Algorithm

Many different internal CS algs exist, but can be reduced to a generic principle [Sch86]: Periodically,

1) Generate ≈simultaneous resync. event at every node
   - via synchronized clocks [LWL88, FC97b, SS97a, etc.]
   - via message exchange [ST87, VRC97]

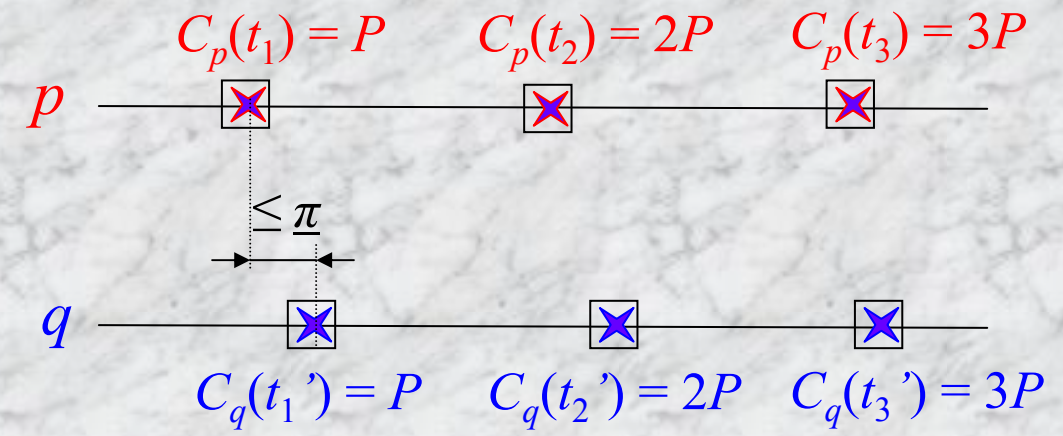
2) Estimate remote clocks
   - one-way [LWL88, SS97a, etc.]
   - round-trip [Cri89]

3) Compute FT convergence function on clock readings
   - fault-tolerant midpoint/avg [LWL88,FC95,Sch97b]
   - a-posteriori agreement [VRC97]
   - non-averaging [ST87]

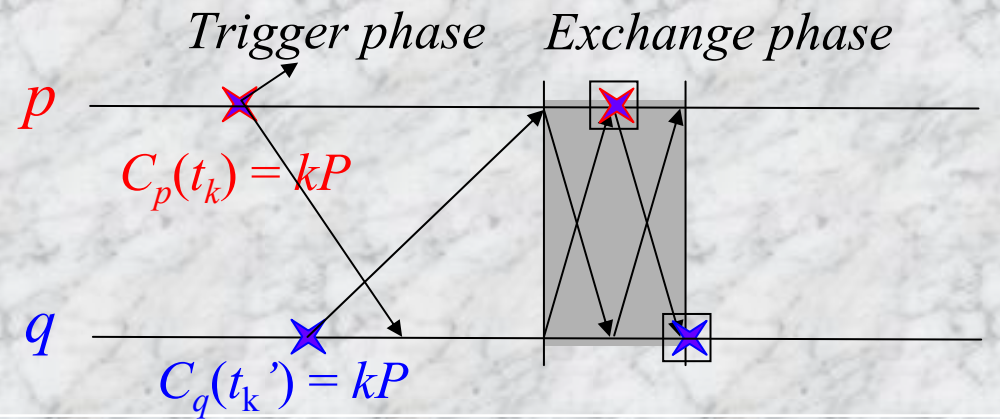
4) Adjust local clock
   - instantaneous correction
   - continuous amortization [SC90, SS97a]

Generate Resynchronization Event

- Via sync. clocks
  - \+ no message overhead
  - – requires initial synchrony

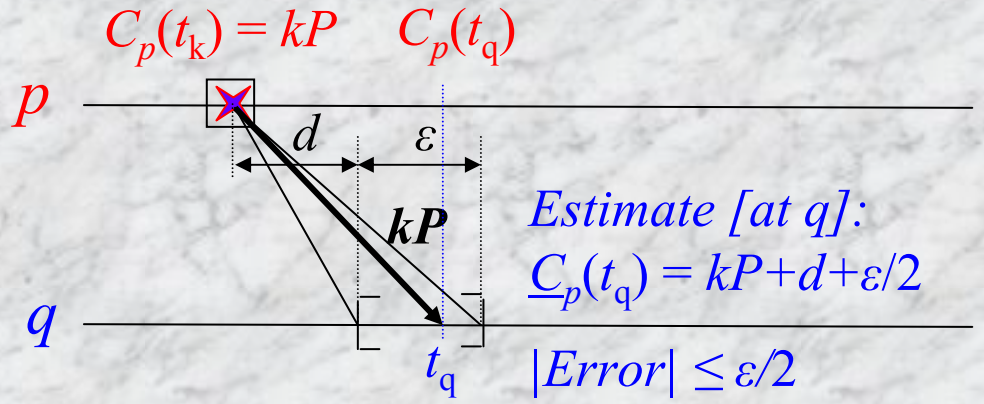


- Via message exchange
  - – message overhead
  - \+ no initial synchrony required

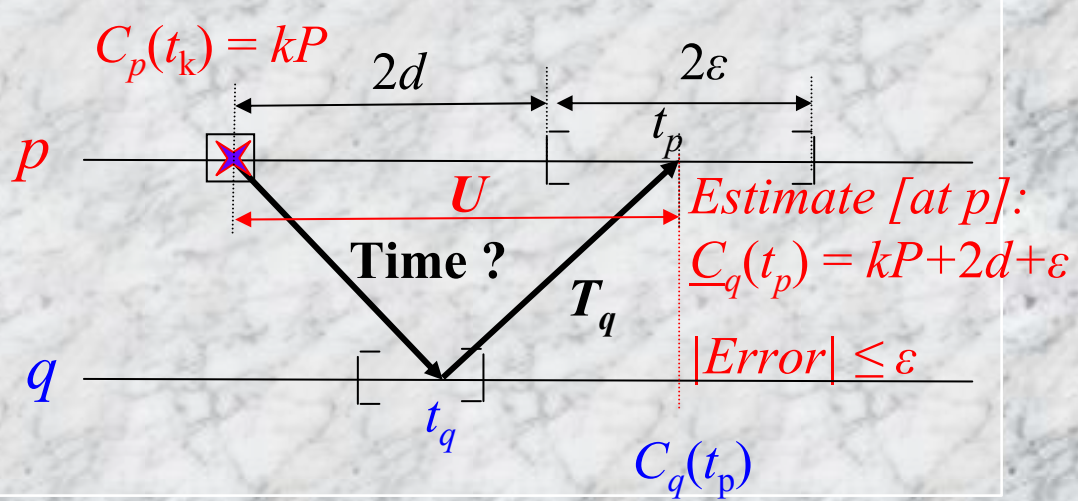


Estimate Remote Clocks

- **One-way:**
  - \+ 1 message only
  - – Must know d and ε (and thus $d_{max} = d + \varepsilon$)

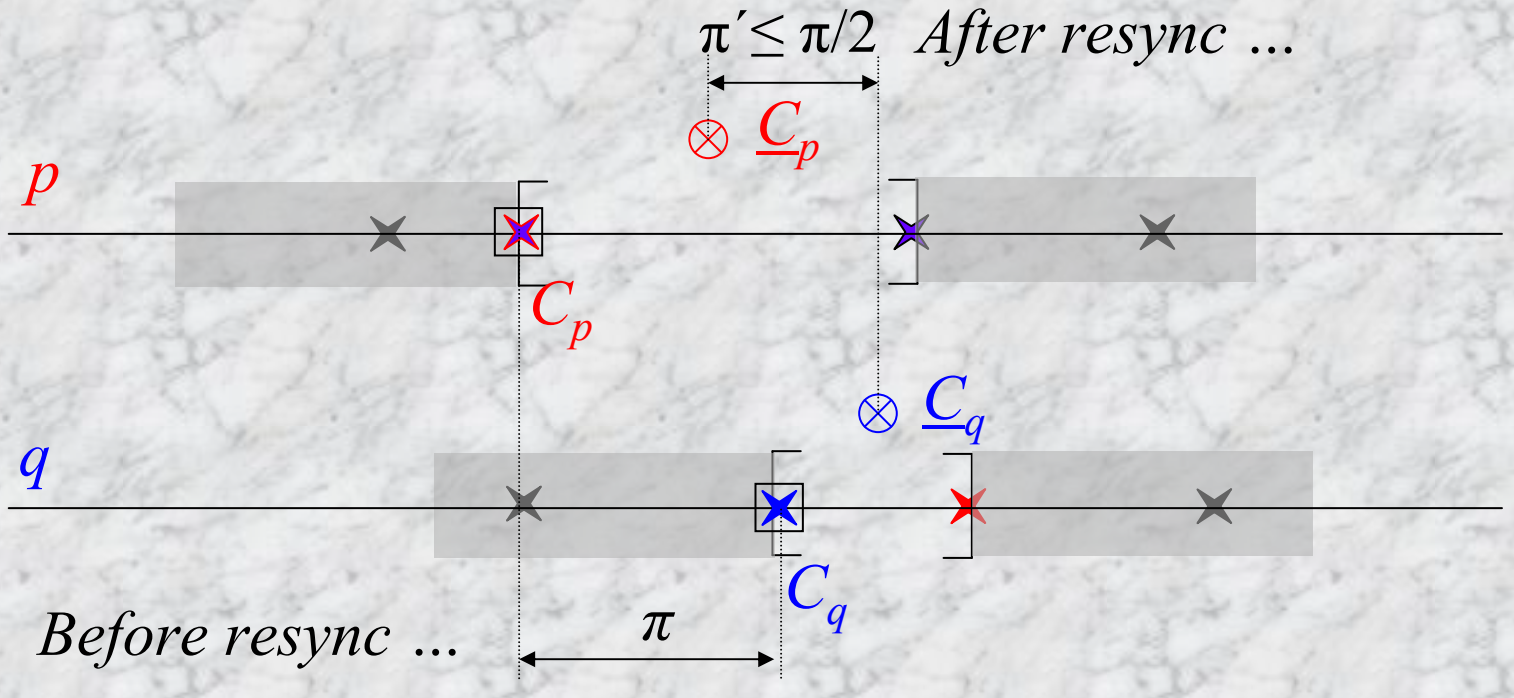


- **Round-trip:**
  - – 2 messages & larger error, BUT
  - \+ Round-trip time ***U*** can be measured locally → need to know d only [compute $\underline{\varepsilon}$]
  - \+ $U \approx 2d \rightarrow \underline{\varepsilon} \approx 0$ "Probabilistic CS" [Cri89]



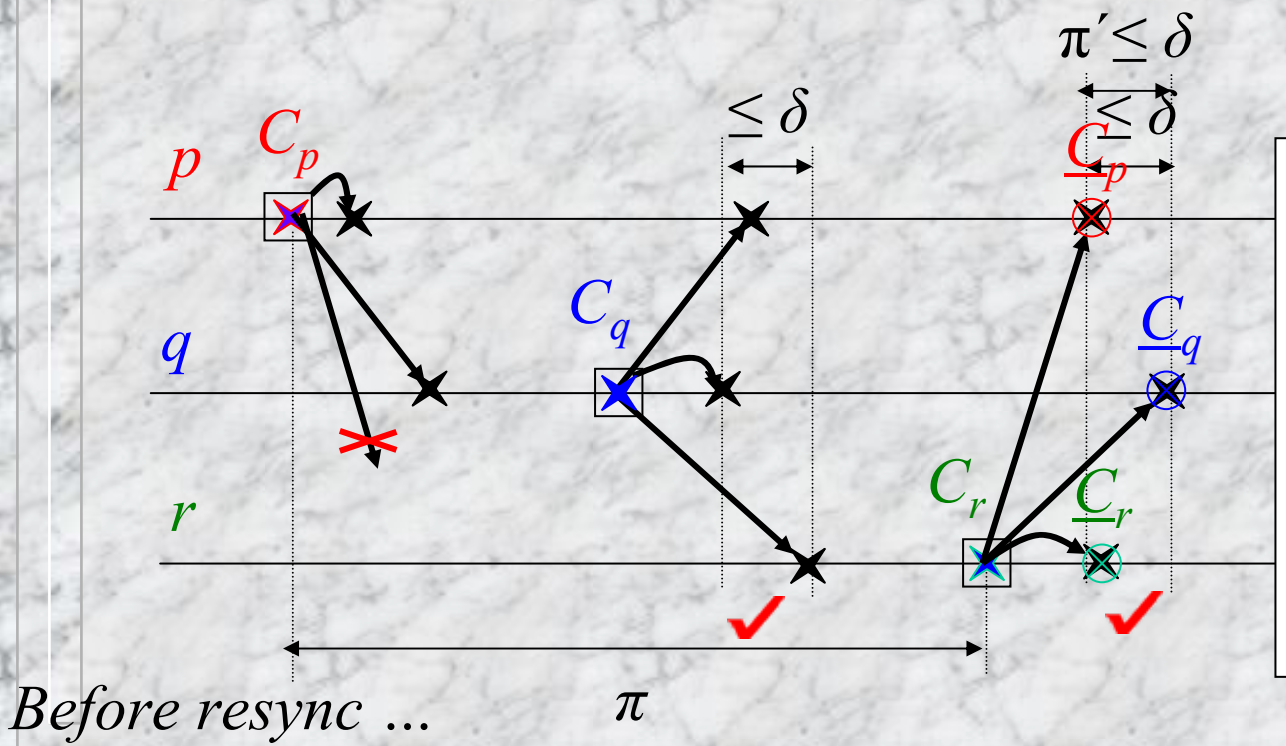
Fault-Tolerant Midpoint CV [LWL88]

- Discard *f* largest and *f* smallest clock readings
- Set local clock to midpoint of remaining interval



A Posteriori Agreement [VRC97]

- Network with efficient HW broadcasting capability
  - At least one faultless bcasts among X successive attempts
  - Reception at all receivers within δ

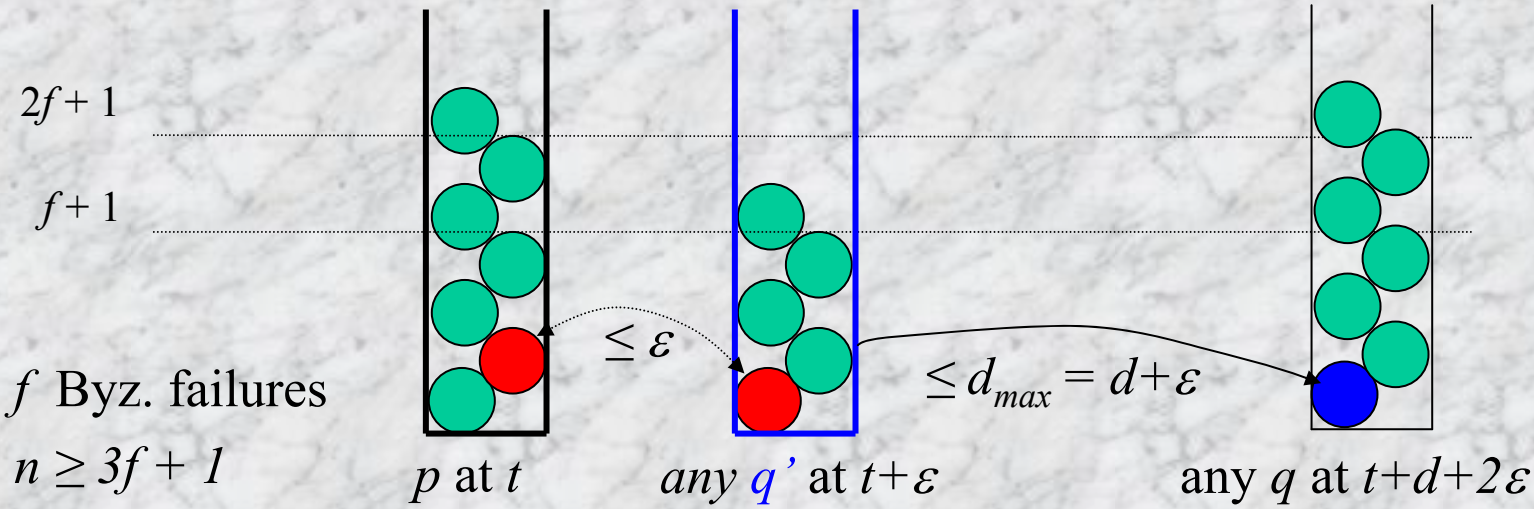


1. Bcast trigger msg; start tentative clock on every reception
2. Detect & select faultless bcasts
3. Run consensus to select winning bcast

Srikanth & Toueg Algorithm [ST87]

if $C_p(t) = kP$
→ send *trig(kP)* to all [Trigger]
if got *trig(kP)* from $f+1$ nodes **or**
got *echo(kP)* from $f+1$ nodes
→ send *echo(kP)* to all [Exchg]
if got *echo(kP)* from $2f+1$ nodes
→ set clock to $kP + \acute{\alpha}$ [Resync]

- Suppose p resyncs at time t
- Then, q resyncs by $t+d+2\varepsilon$
- On resync: Every clock set to same value $kP + \acute{\alpha}$, hence $\pi' = d+2\varepsilon$

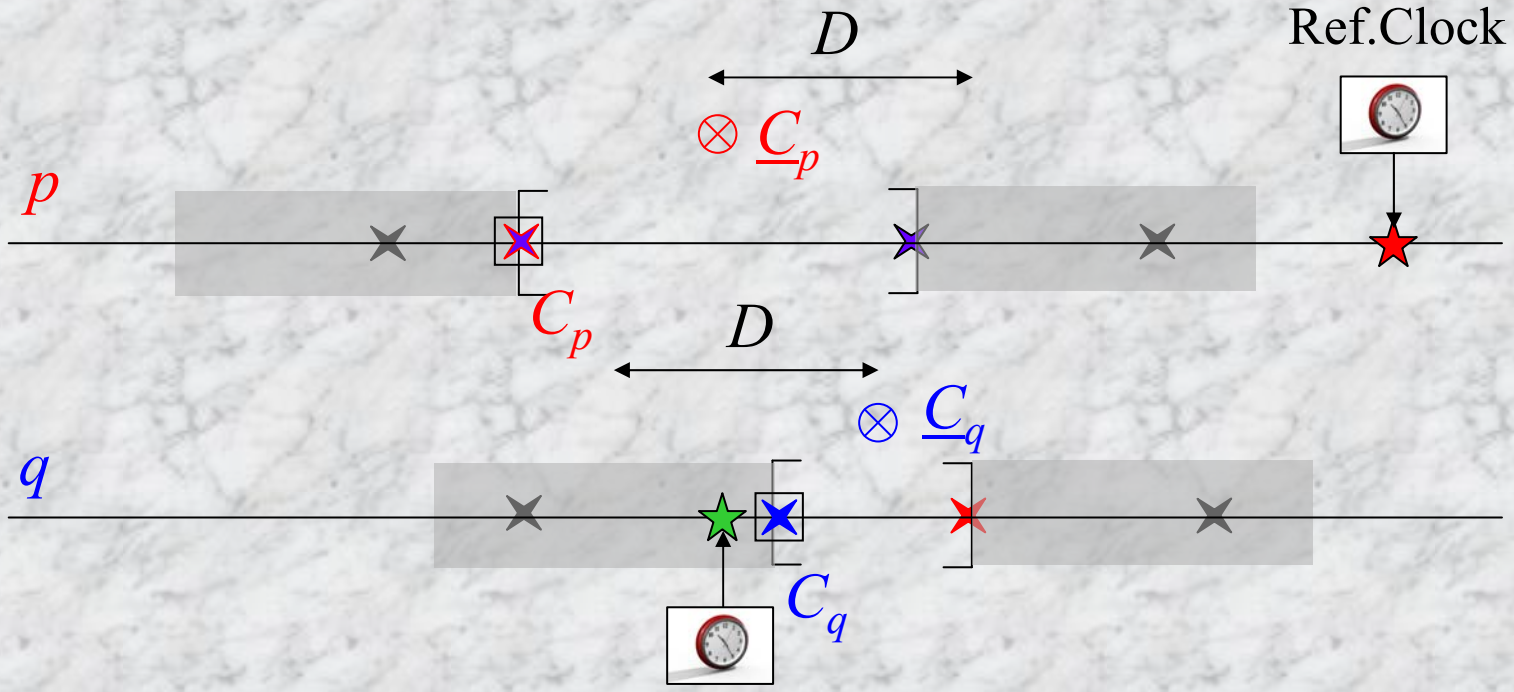


- f Byz. failures
- $n \geq 3f + 1$

Extended FT CS Approaches

Integrating Internal+External CS [FC97b]

- Augment FT midpoint CV, by shifting $\leq D \approx P\rho$ towards external time



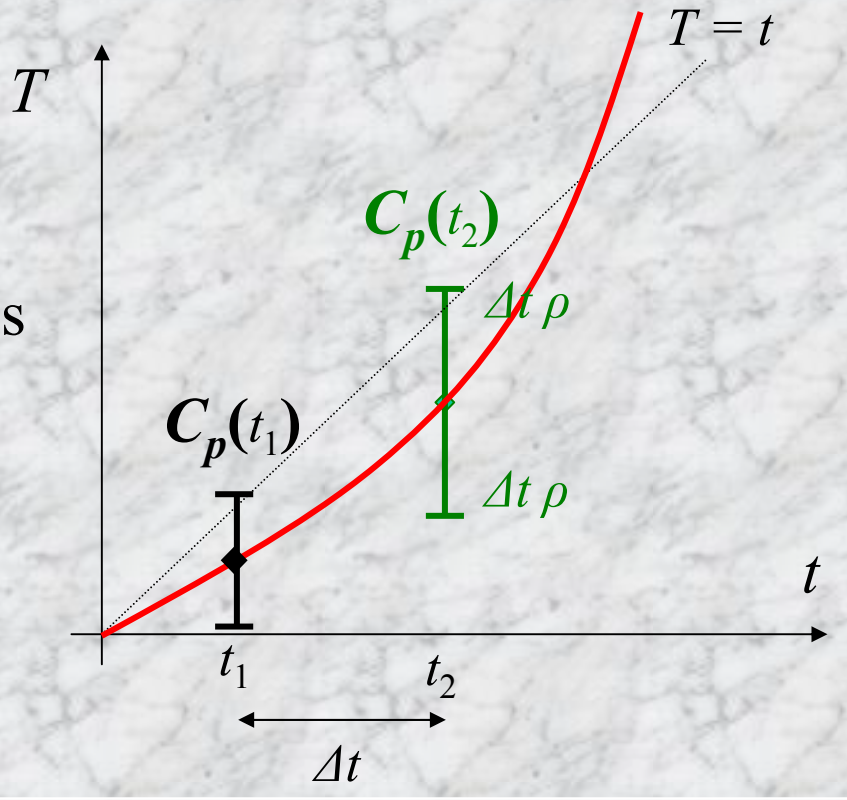
- Tolerates even Byzantine faulty Reference Clocks, at price of somewhat degraded precision

Interval-based Clock Synchronization

Interval-based Clock Synchronization 1/3

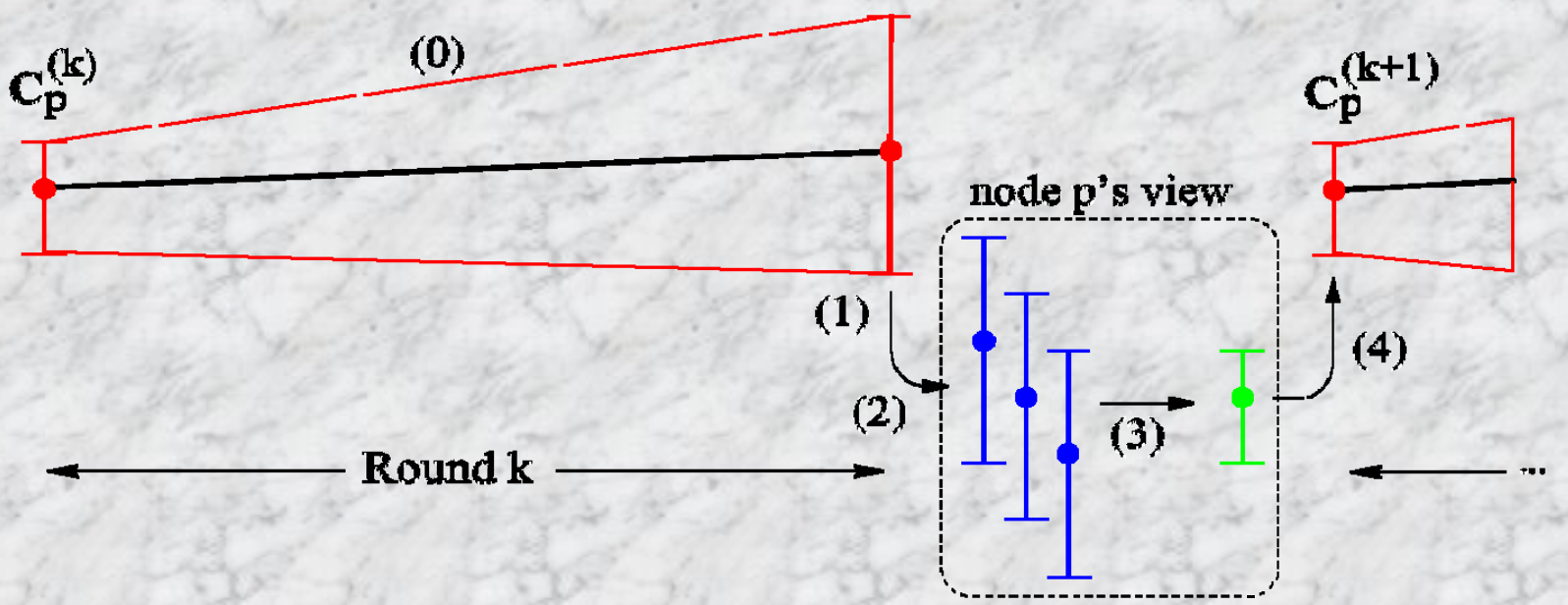
Ordinary clock $C_p(t)$ → **Interval clock $C_p(t)$** [Mar84]

- $\boldsymbol{C_p(t)} = [C_p(t) - \alpha_p(t),\ C_p(t) + \alpha_p(t)]$
  - reference point $C_p(t)$
  - accuracy interval $[-\alpha_p(t),\ \alpha_p(t)]$
- $\boldsymbol{C_p(t)}$ correct iff $t \in \boldsymbol{C_p(t)}$
- Maintaining correct $\boldsymbol{C_p(t)}$ requires interval deterioration
  - Drift compensation ($\pm \Delta t\, \rho$)
  - Delay compensation ($\pm \varepsilon$)
- Elaborate SynUTC framework [SS97, Sch97a, …]



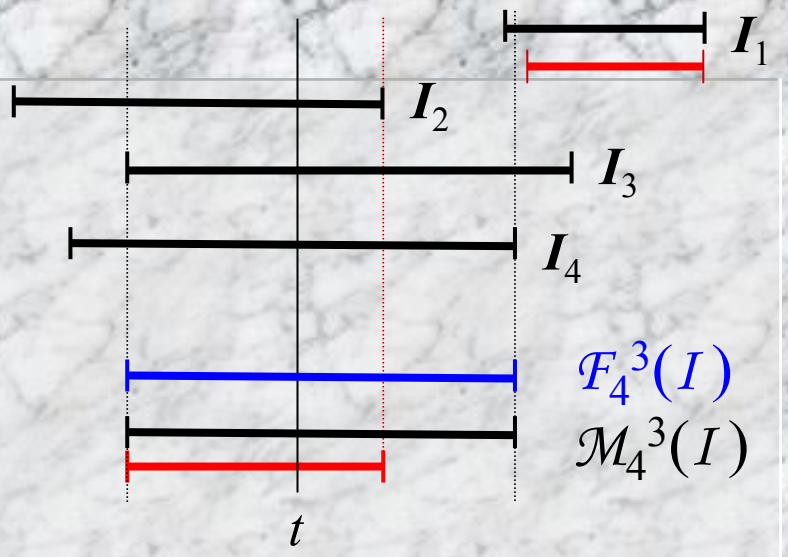
Interval-based Clock Synchronization 2/3

Fits generic CS principle [Sch86]:



Interval-based Clock Synchronization 3/3

- Some interval clock readings may be incorrect ($t \notin I_j$)
  - $\Rightarrow$ Intersection $\cap I_j$ possibly empty
- Need fault-tolerant intersection



- Marzullo function $\mathcal{M}_n^{n-f}(I)$ [Mar84]
  - Maximum interval with end points in n-f of the inputs I_j
  - Worst case optimal solution, but …

- FTI function $\mathcal{F}_n^{n-f}(I)$ [SS99]
  - [f+1-largest left end, f+1-smallest right end] of the inputs I_j
  - No discontinuous output (optimal among functions satisfying Lipschitz condition)

Clock Rate Synchronization

Clock Rate Synchronization 1/2

Clock state sync

- controls/adjusts $C_p(t)$
- ideal: $C_p(t) = t$
- clock drift ρ
- precision

  $|C_p(t) - C_q(t)| \leq \pi$
- accuracy

  $|C_p(t) - t| \leq \alpha$

Clock rate sync [Sch97c]

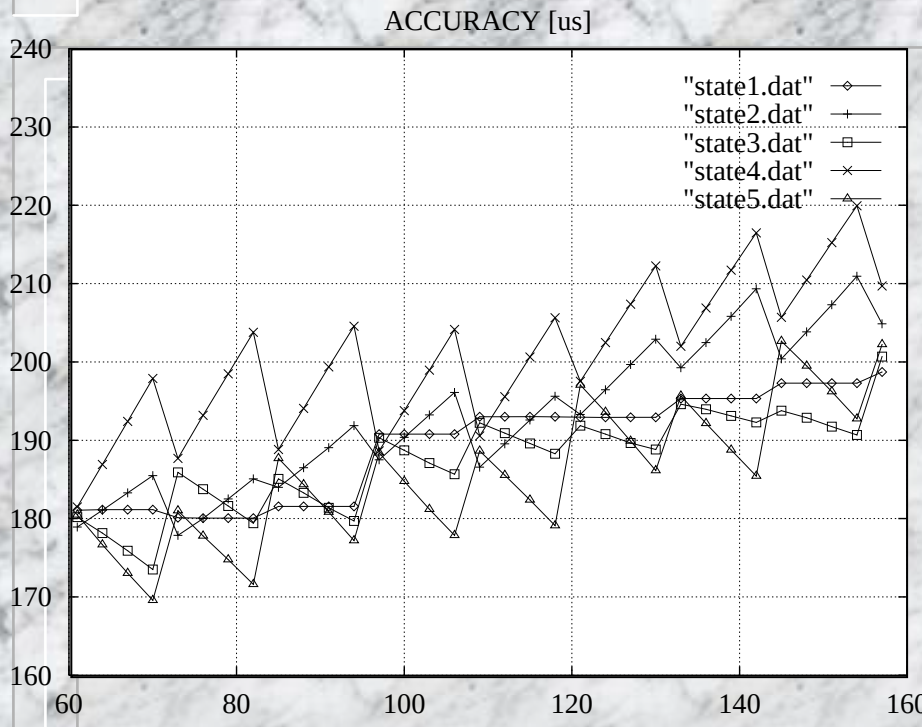
- controls $v_p(t) = dC_p(t)/dt$
- ideal: $v_p(t) = 1$
- long-term clock stability σ
- consonance

  $|v_p(t) - v_q(t)| \leq \gamma$
- drift

  $|v_p(t) - 1| \leq \rho$

- Clock rate sync alg easily piggybacked on clock state sync alg
- Replacing static drift ρ by dynamic $v_p(t)$ enables high-precision CS with low-precision [but high-stability] clocks

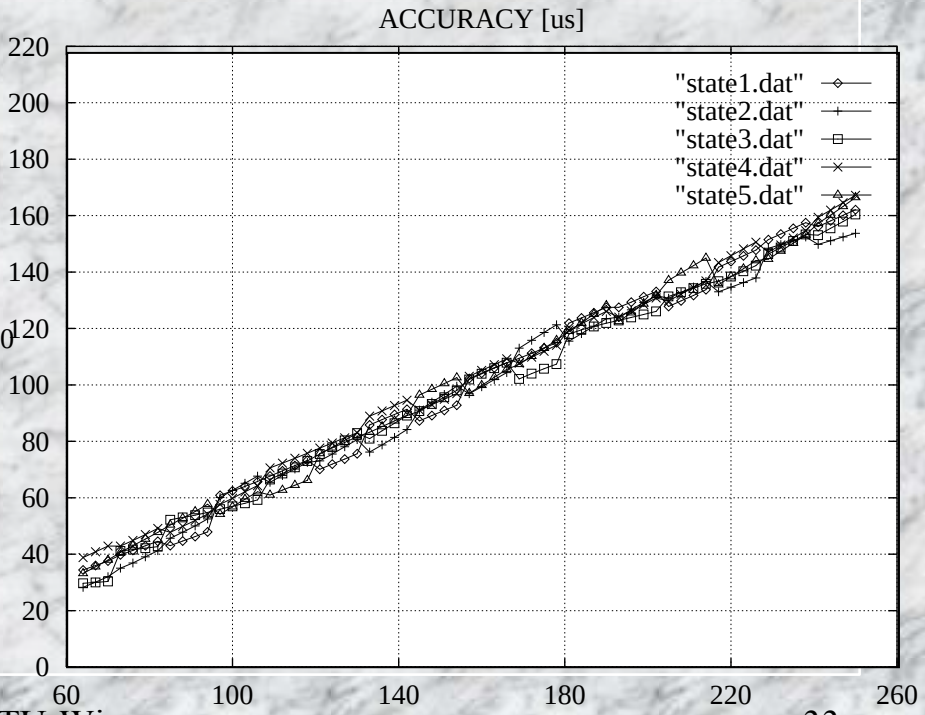
Clock Rate Synchronization 2/2



State sync only

[SW99]

State + rate sync



Hardware Assistance

High-Precision CS - Requirements

Detailed worst case analysis [Sch97] showed:

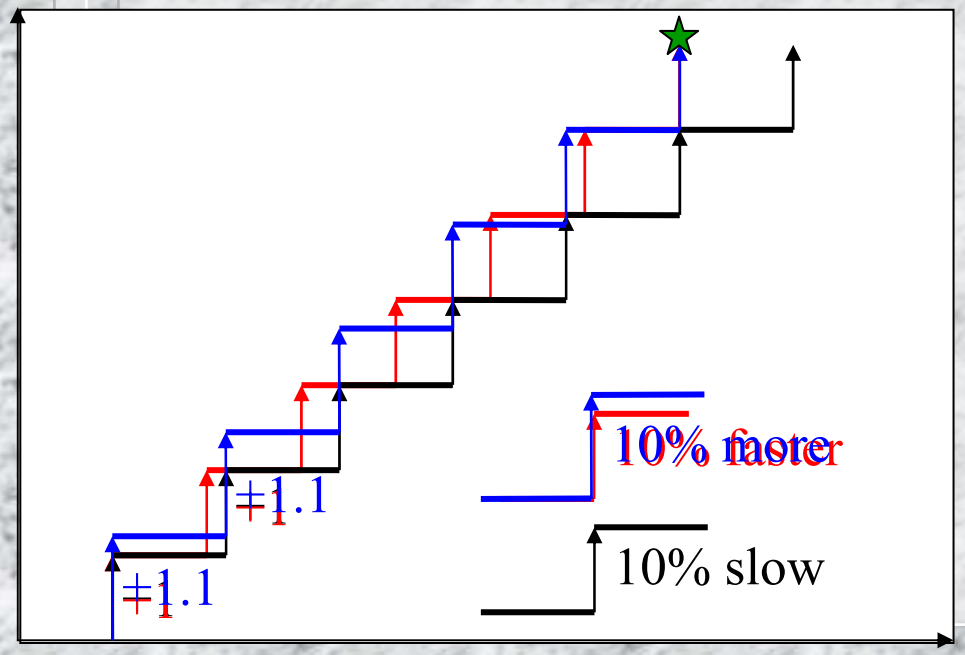
- Precision

$$\pi_{\mathrm{OP}} = 4\varepsilon + 4P\rho + 3G + G_s + 11u$$

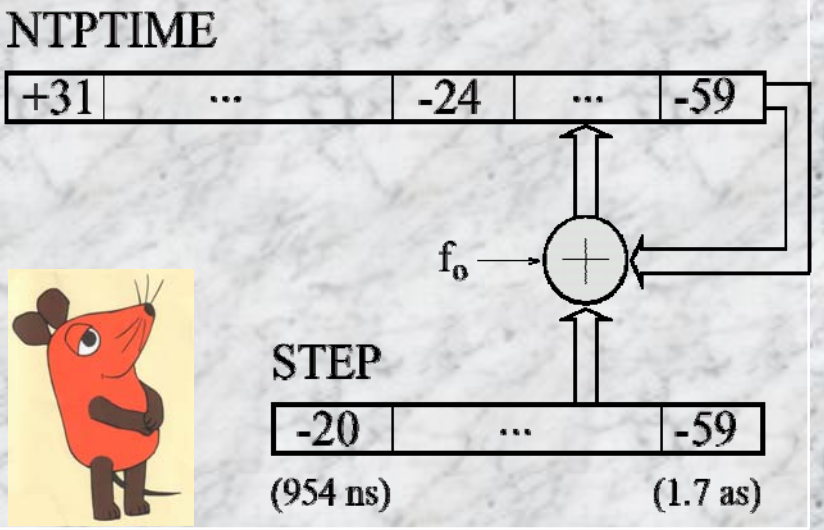
- Dominant terms:
  - End-to-end delay jitter ε
  - Accumulated drift $P\rho$
- Discretization effects also signifikant:
  - Clock reading granularity G
  - Clock setting granularity G_s
  - Clock rate-adjustment uncertainty $u = 1/f_o$

Adder Based Clock

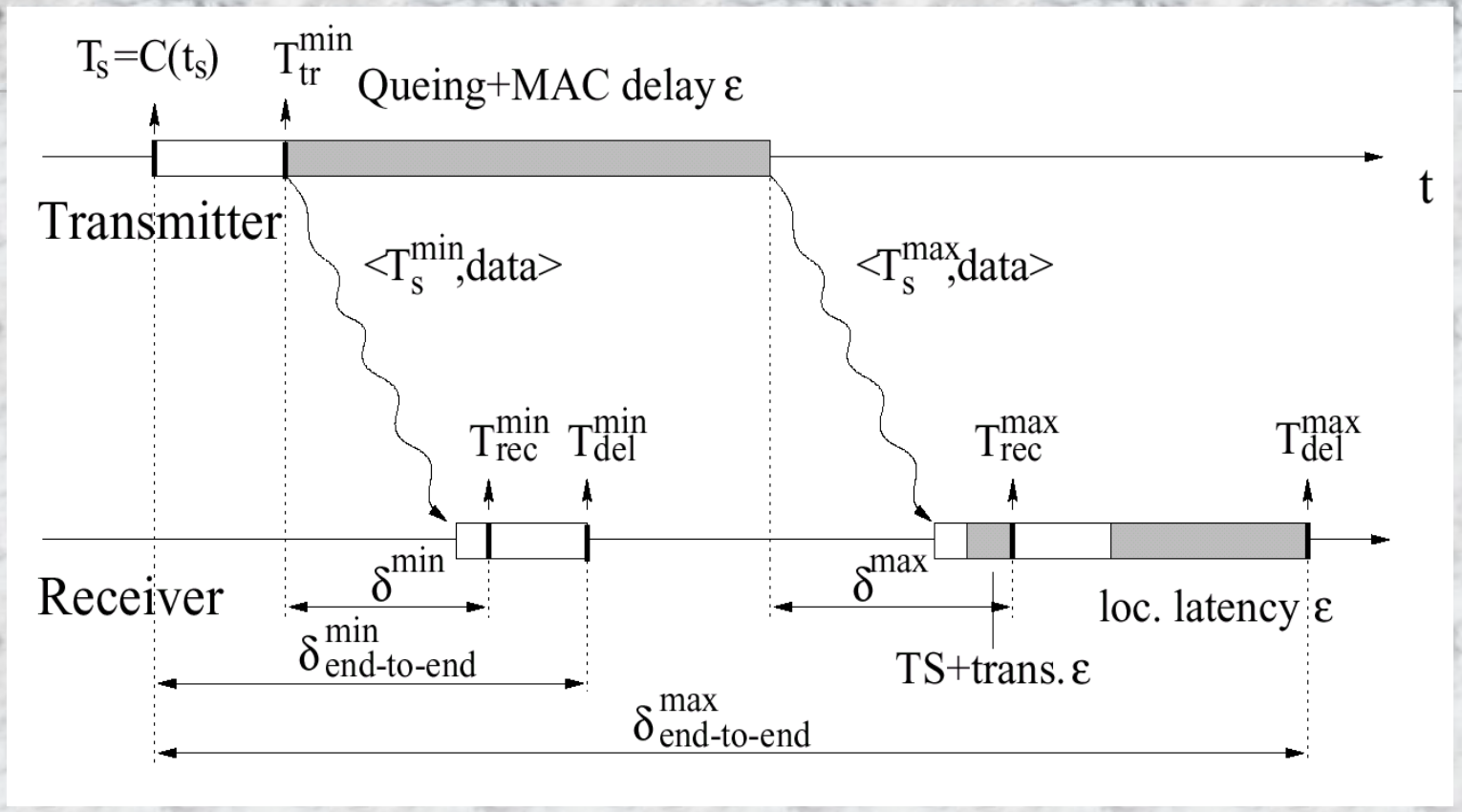
- A clock, driven by a const. high-frequency oscillator f_o, yet
  - produces an arbitrary, user-selectable clock speed
  - facilitates very fine-grained rate-adjustments
  - supports continuous amortization [and even leap seconds]



ABC architecture [SSHL97]:

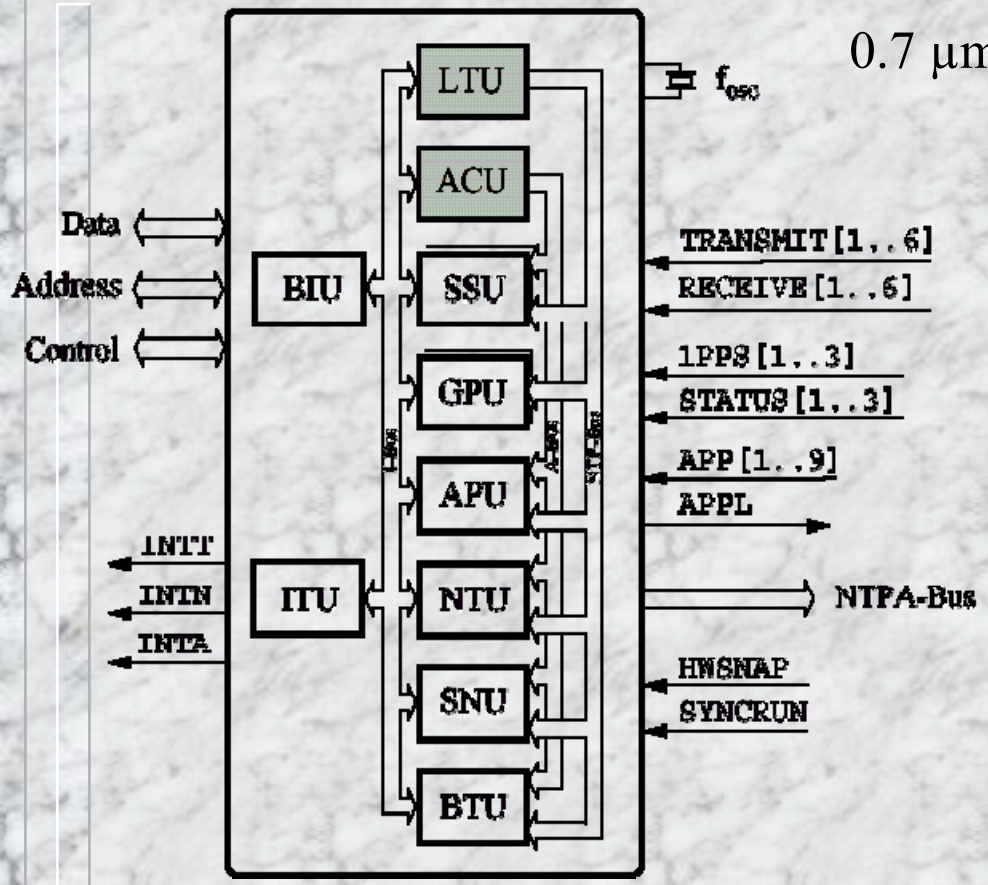


Reduce ε via Msg Timestamping [KO87]



- Large end-to-end jitter $\varepsilon_{end\text{-}to\text{-}end}$: 10 ms range and more
- MAC-layer timestamping greatly reduces ε : 10 ns range

UTCSU for Interval-based CS



0.7 μm CMOS (100 mm²), 208-Pin MQFP

unit	full name
ACU	Accuracy Unit
APU	Application Unit
BIU	Bus Interface Unit
BTU	Built-In Test Unit
GPU	GPS Unit
ITU	Interrupt Unit
LTU	Local Time Unit
NTU	Network Time Interface Unit
SNU	Snapshot Unit
SSU	Synchronization Subnet Unit

[SSHL97]

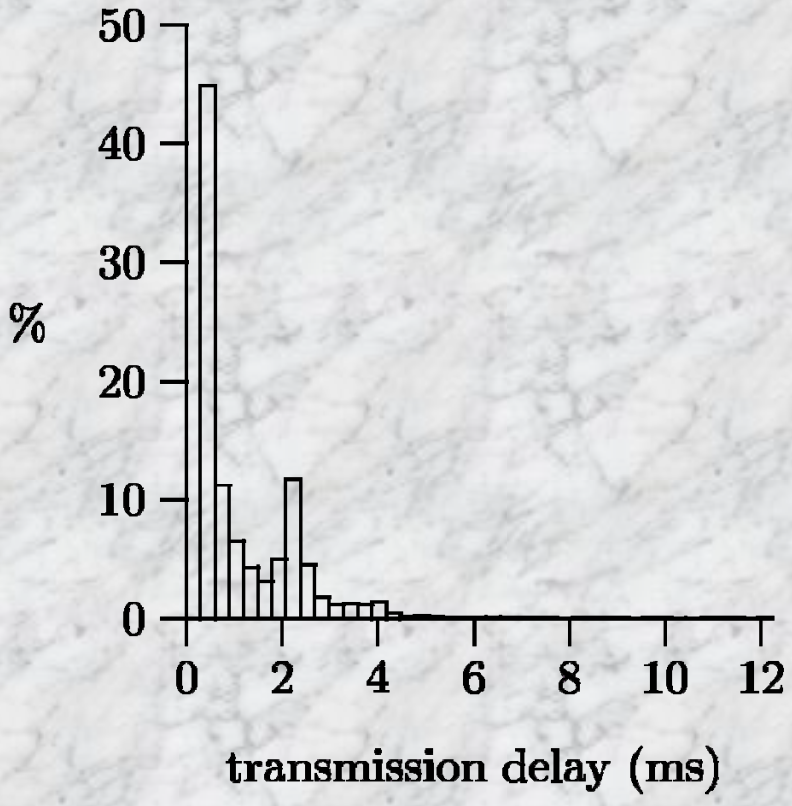
Network Time Interface (NTI)

- Mezzanine bus MA-Module
- UTCSU with 10 MHz OCXO
- 512 KB shared memory with packet timestamping logic
- XILINX 95216-PQFP160 CPLD
- Optically isolated, buffered I/Os

[SKMNCK99]

NTI Evaluation Results 1/2

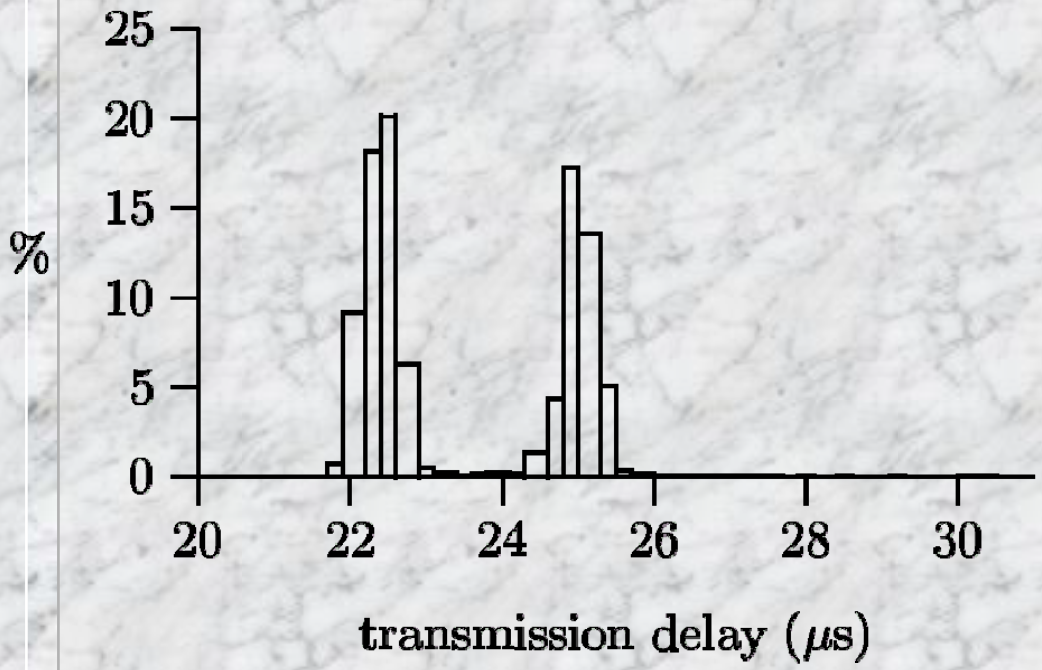
Delay-jitter of message end-to-end delays:



Minimum	0.30 ms
99%-minimum	0.30 ms
95%-minimum	0.30 ms
95%-maximum	4.17 ms
99%-maximum	9.24 ms
Maximum	37.00 ms
Average	1.35 ms
Std.Dev.	1.63 ms

NTI Evaluation Results 2/2

Delay-jitter for NTI packet memory timestamping:

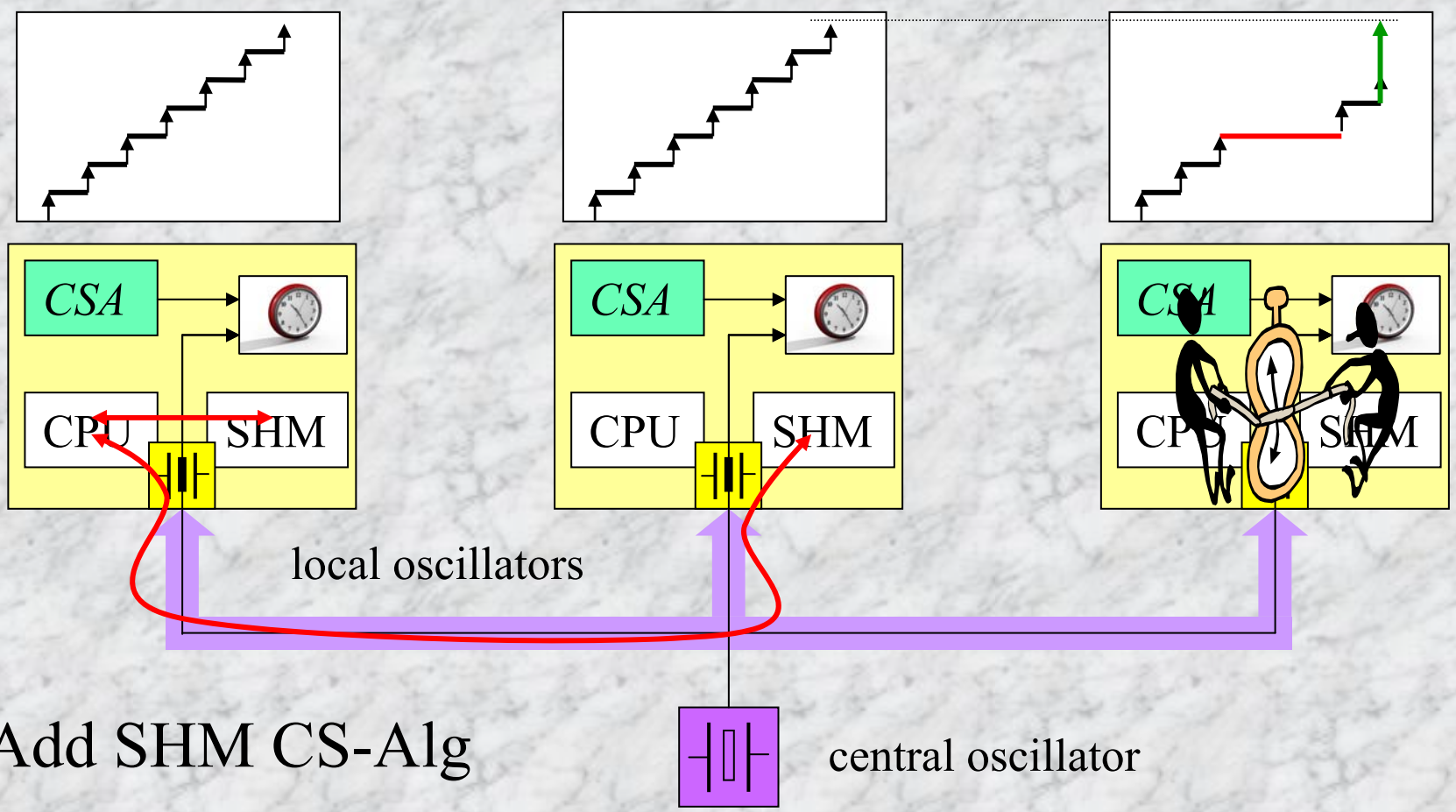


Minimum	20.4 μs
99%-minimum	21.7 μs
95%-minimum	21.9 μs
95%-maximum	25.5 μs
99%-maximum	31.0 μs
Maximum	36.8 μs
Average	23.6 μs
Std.Dev.	1.6 μs

Novel FT CS Approaches

CS in Shared Memory MP Systems

Shared Memory Multiprocessors



k-Wait-free Clock Synchronization [DW93]

- After at least k consecutive clock ticks at node *p*, *q*, the following must hold (irrespectively of other nodes):
  - Regular adjustments at every tick (~ accuracy):

$$C_p(k+1) = C_p(k) + 1$$

  - Agreement at every tick (~ precision):

central clock: $C_p(k) = C_q(k)$

local clocks: $|C_p(k) - C_q(k)| \leq 1$

- Optional: Additional self-stabilization requirement
  - Transient failures can cause memory/registers of (possibly all) processors to take on arbitrary values
  - After finite time, k-wait-free properties are guaranteed to hold

Biological Clock Synchronization

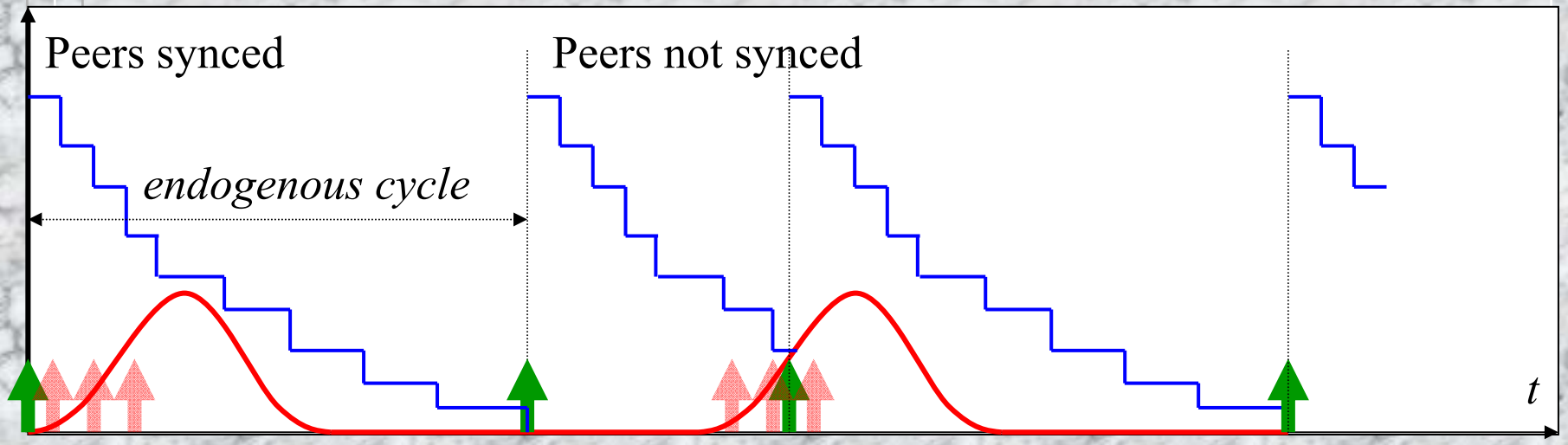
Pulse Generation in Biological Systems

- *Malaccae* Fireflies
  - Male fireflies emit light pulses ~ once per second
  - Swarm of fireflies eventually flash in synchrony

- Cardiac ganglion of lobster heart
  - Heart activation by synchronized firing of 4 interneurons
  - Ganglion controls activation frequency within some bounds, without losing synchrony
  - Evolution-optimized strategy: Fault-tolerance, self-stabilization, etc.

SS+FT Pulse Generation Alg. [DDP03]

- Pulse-coupled oscillators ("integrate-and-fire")
  - Time-decaying refractory function ~ own node's sense of time
  - Time-decaying threshold level ~ perception of pulses from peers
  - Pulse is generated [+ refractory function reset] when refractory function hits threshold level



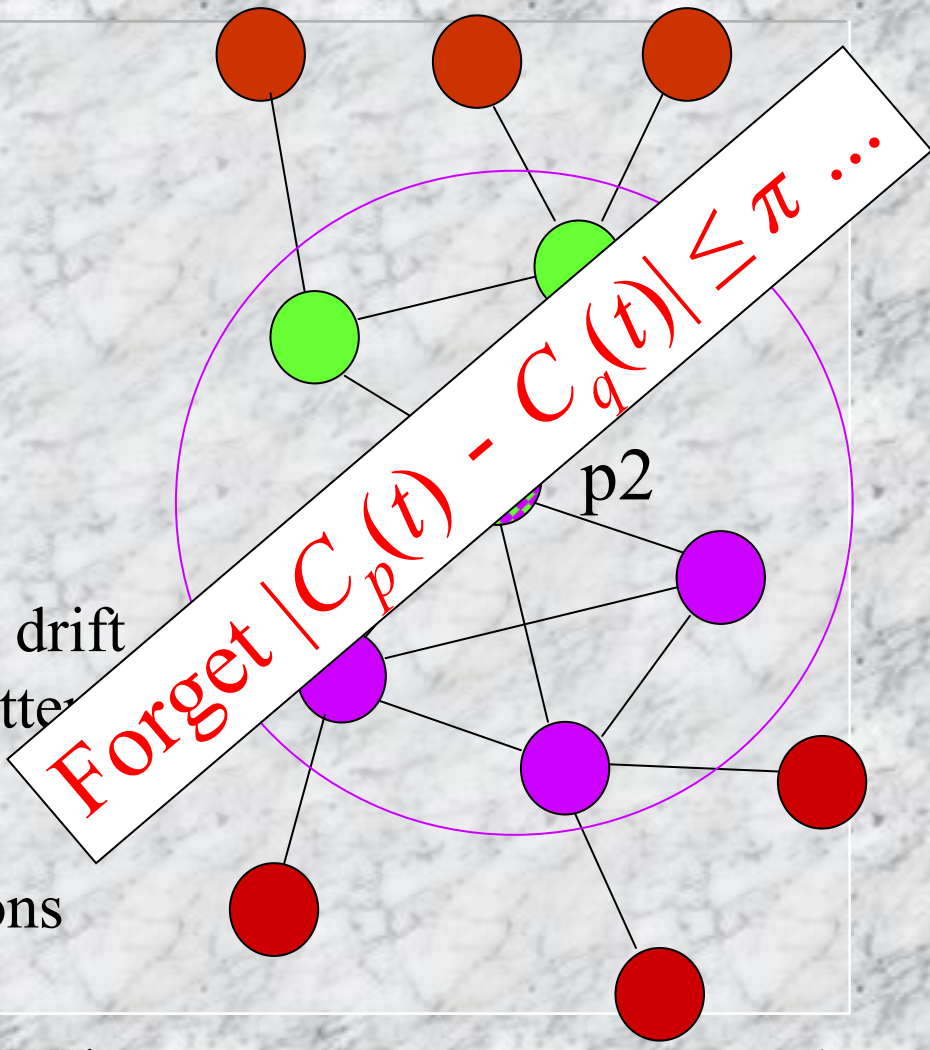
SS+FT Clock Sync Algorithm [DDP03b]

- Allows to build linear-time self-stabilizing, Byzantine fault-tolerant clock sync algorithm, by
  - using synchronized pulses as a pacemaker
  - employing a self-stabilizing Byzantine agreement algorithm acting on clock values
- Also solves initial clock synchronization problem - although not in constant time [WS07]

Clock Sync in Wireless Networks

Wireless Network Challenges

- Special network topology
  - Large number of nodes
  - Sparse connectivity!
  - Mobility of nodes
- Cost and energy constraints
  - Infrequent communication
  - Cheap clocks → accumulated drift dominates end-to-end delay jitter
- Special failure semantics
  - Primarily crashes and omissions
  - But: Partitioning!



Interval-based CS [BMT04]

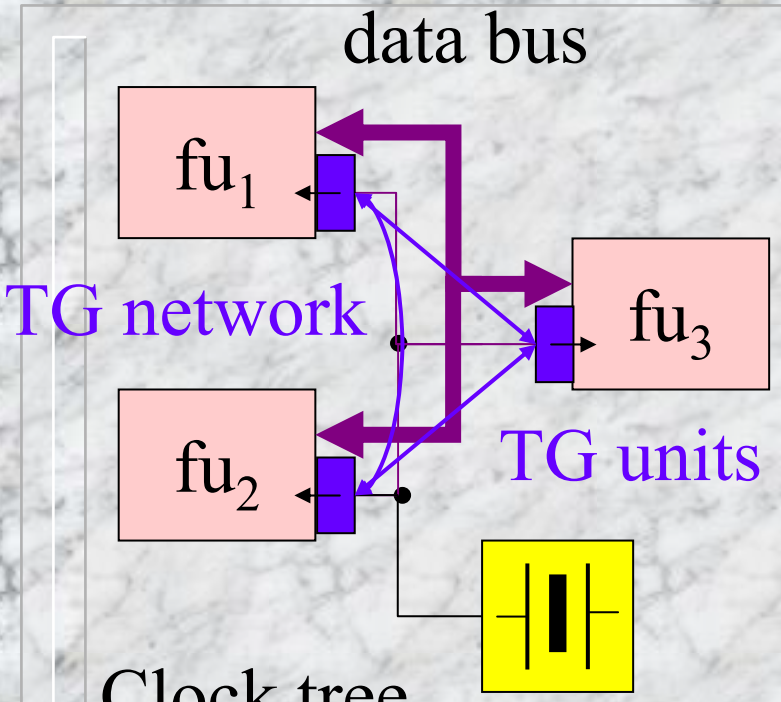
- Essentially external clock synchronization
  - Nodes individually maintain correct interval
  - (FT) Intersection extracts smallest interval from available input intervals
  - Average accuracy benefits from drift diversity & mobility
- No bounded precision π, but guaranteed dynamic accuracy bounds still enable
  - + correct data fusion
  - + correct concerted action schedules
  - + exception on [possibly] excessive inaccuracy

Gradient Clock Synchronization [FL04]

- $| C_i(t) - C_j(t) | \leq \pi(d_{ij})$ with d_{ij} = hop-distance of i and j
- Good synchronization precision $\pi(1)$ only for direct neighbors
- Lower bound: $\pi(d) = \Omega(d + \log Diam / \text{loglog } Diam)$ $\rightarrow \pi(1) >$ const !
- Good algorithms still lacking …

Clock Generation in Systems-on-Chip

Clock Generation in SoCs



Classic synchronous clock

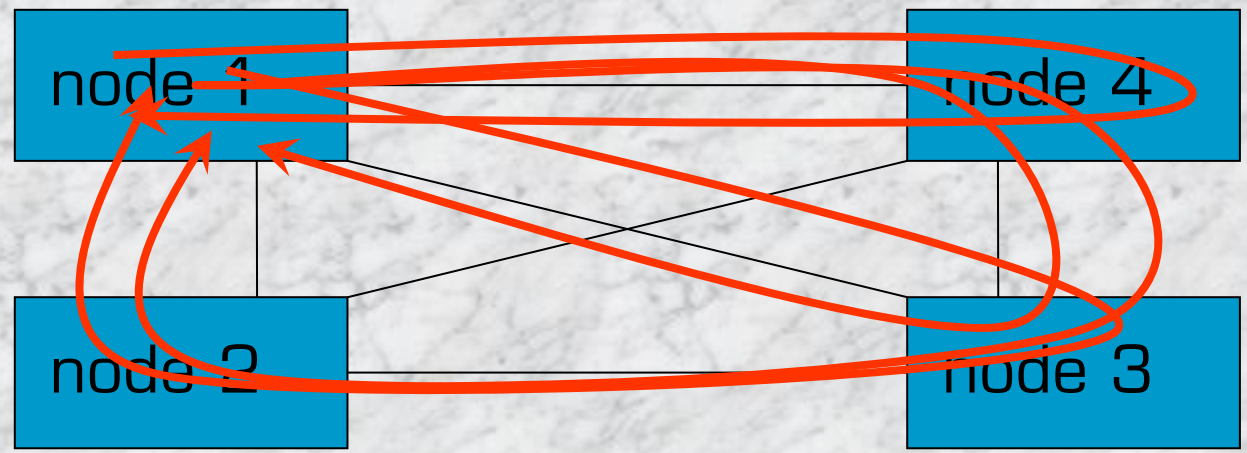
Alternative: Fault-tolerant distributed clock (DARTS)

http://ti.tuwien.ac.at/ecs/research/projects/darts

- Provide every on-chip functional unit with simple fault-tolerant Tick Generation unit [as in GALS]
- Unlike in GALS, TG units communicate via dedicated TG network to generate
  - loosely synchronized micro-ticks for fu-internal clocking [100 Mhz range]
  - synchronous macro-ticks [every Ξ micro-ticks] for communication between fu's

DARTS-Clocks - Principle 1/4

„FT Distributed ring oscillator“:



self-oscillating **Byz. fault tolerant**

DARTS-Clocks - Algorithm 2/4

Modified Srikanth & Toueg algorithm [FSFK06]:

on booting do:

send *tick(0)* to all; clock:= 0;

continuously do:

If received *tick(m)* from at least f+1 remote nodes and m > clock:

send *tick(clock+1),..., tick(m)* to all; clock:= m;

If received *tick(m)* from at least 2f+1 remote nodes and m ≥ clock:

send *tick(m+1)* to all; clock:= m+1;

Tick numbers:

- Cannot reasonably be implemented in HW
- Must be substituted by „up/down“ ticks

DARTS-Clocks - Proofs 3/4

[FSFK06]

Pipe Compare Signal Generators (PCSGs): There exists a dedicated detection circuit for each pair of pipes which generates the status signals $GEQ^{o/e}_{p,q}(t)$ and $GR^{o/e}_{p,q}(t)$. In particular, $GEQ^{o}_{p,q}(t')$ becomes active (i.e., $GEQ^{o}_{p,q}$ previou

(i) $r^{se}_{p,}$

(ii) $[r^{r}_{p}$

Similarl

(i) $r^{self}_{p,q}(t) \in \mathbb{N}_{odd}$ a

Definition 4.1. (Direct Causality). *Let $I(t')$ and $O(t)$ be two events of some specific signal input and output, respectively, of a correct component C. Then $I(t')$ and $O(t)$ are* directly causally related, *denoted by $I(t') \rightarrow O(t)$, if*

(i) they are

(ii) there is i.e., $\nexists I'$

Theorem 4.13. (Precision). *The precision $\pi \geq |b_q(t) - b_p(t)|$ of our algorithm is bounded by $\pi \leq \left\lfloor \frac{T_{sim}}{T^-_{first}} \right\rfloor + 1$.*

Proof. First of established for a $k+1$, i.e., $t^p_k \leq$

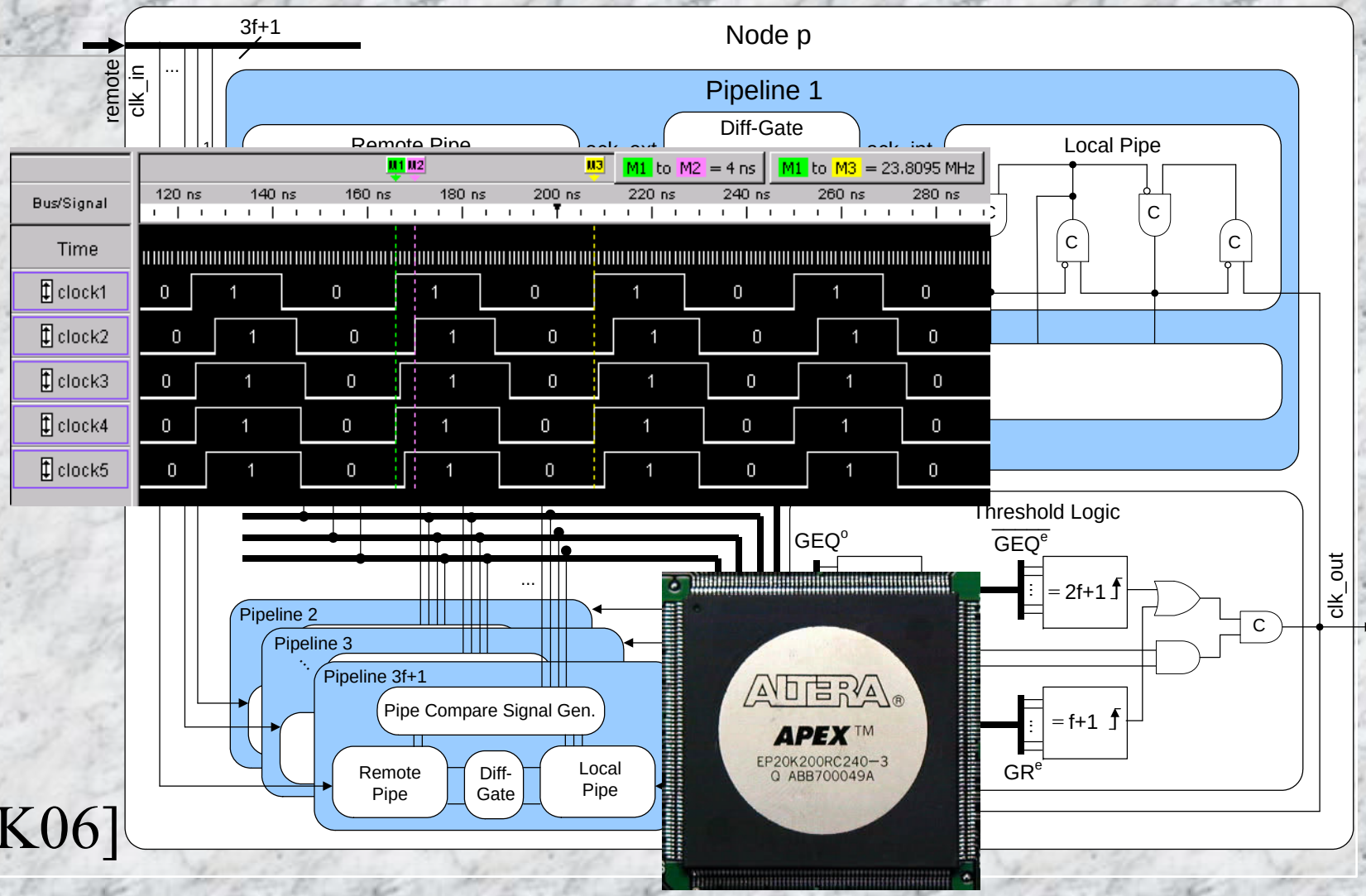
$b^{max}(t')$

Assume that pr

Theorem 4.14. (Accuracy). *Given $\Delta = t_2 - t_1$, the accuracy $|b_p(t_2) - b_p(t_1)|$ of any correct process p is bounded by* $\max\left\{0, \frac{\Delta - T_{sim} - T^+}{T^+}\right\} \leq |b_p(t_2) - b_p(t_1)| \leq \left\lceil \frac{\Delta}{T^-_{first}} \right\rceil + \min\left\{\pi + 1, \left\lceil \frac{\Delta}{D^-} - \frac{\Delta}{T^-_{first}} \right\rceil\right\}$.

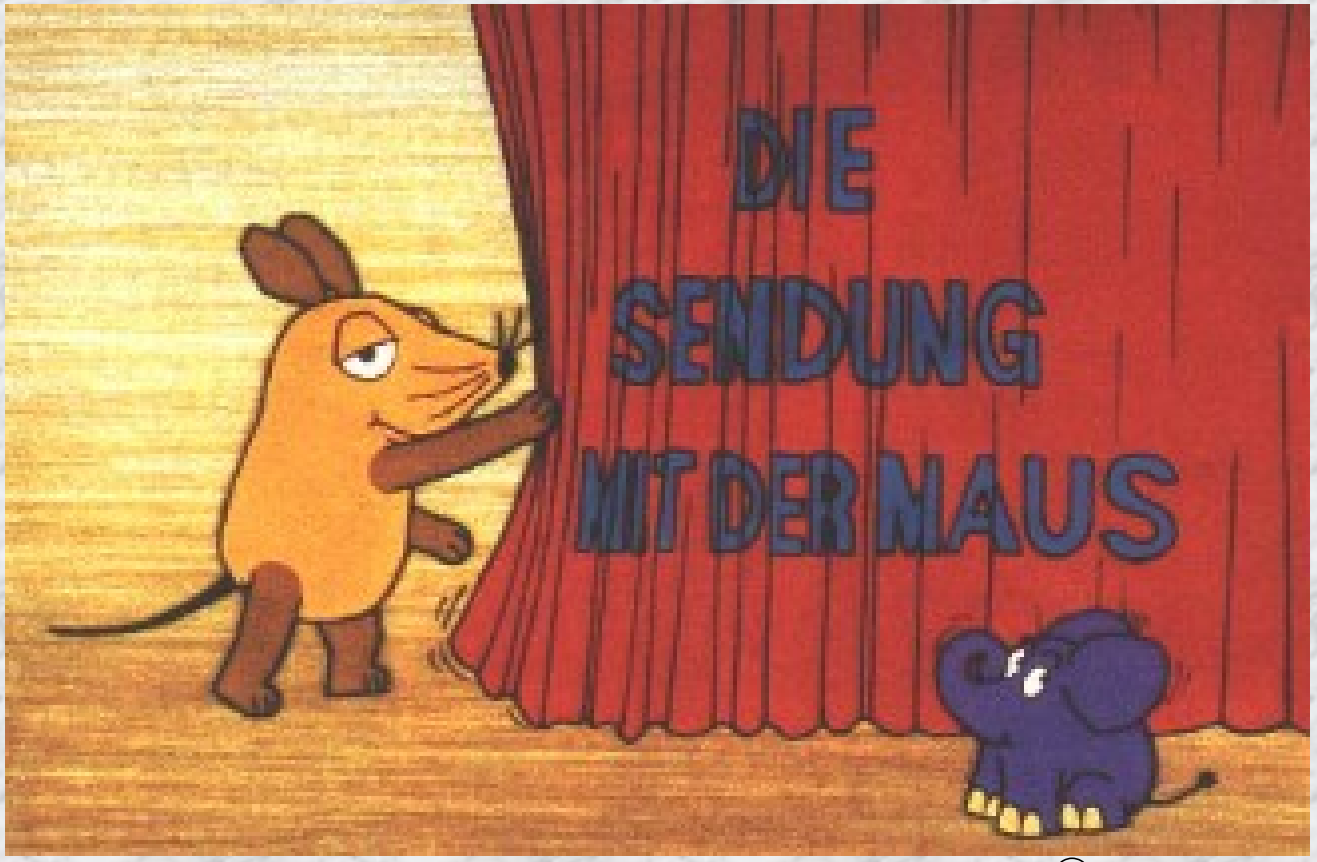
Proof. The upper bound for accuracy will be shown first: It is known that $\forall t : b_p(t) \geq b^{max}(t) - \pi + (1 - I_{usync}(t))$ and $\forall t : b_p(t) \leq b^{max}(t)$ from Lemma 4.13 and Lemma 4.11. Thus $b_p(t_2) - b_p(t_1) \leq b^{max}(t_2) - b^{max}(t_1) + \pi - (1 - I_{usync}(t_1))$. By applying Lemma 4.11, $b_p(t_2) - b_p(t_1) \leq \left\lfloor \frac{t_2 - t_1}{T^-_{first}} \right\rfloor + 2I_{usync}(t_1) - 1 + \pi \leq \left\lfloor \frac{t_2 - t_1}{T^-_{first}} \right\rfloor + \pi + 1 \leq \left\lceil \frac{t_2 - t_1}{T^-_{first}} \right\rceil + \pi + 1$. Moreover, from Lemma 4.7 it follows that $b_p(t_2) - b_p(t_1) \leq \left\lceil \frac{t_2 - t_1}{D^-} \right\rceil$. Hence, $b_p(t_2) - b_p(t_1) \leq \min\left\{\left\lceil \frac{\Delta}{T^-_{first}} \right\rceil + \pi + 1, \left\lceil \frac{\Delta}{D^-} \right\rceil\right\} \leq \left\lceil \frac{\Delta}{T^-_{first}} \right\rceil + \min\left\{\pi + 1, \left\lceil \frac{\Delta}{D^-} \right\rceil - \left\lceil \frac{\Delta}{T^-_{first}} \right\rceil\right\} \leq \left\lceil \frac{\Delta}{T^-_{first}} \right\rceil + \min\left\{\pi + 1, \left\lceil \frac{\Delta}{D^-} - \frac{\Delta}{T^-_{first}} \right\rceil\right\}$ since $\lceil x + y \rceil \leq \lceil x \rceil + \lceil y \rceil$.
To prove the lower bound, first define $b_1 = b_p(t_1)$, $b_2 = b_p(t_2)$ and $t^p_{b_1} \leq t_2$, $t^p_{b_2} \leq t_2$ as the points in time when p sends tick b_1 and b_2. Clearly $t^p_{b_2+1} > t_2$,

DARTS-Clocks - Implementation 4/4



[FFSK06]

The End



References

- [BMT04] P. Blum, L. Meier and L. Thiele. Improved interval-based clock synchronization in sensor networks. In Proceedings of the third international Symposium on Information Processing in Sensor Networks (Berkeley, California), 2004, p. 349-358.
- [Cri89] F. Cristian. Probabilistic clock synchronization. Distributed Computing, 3(3):146--158, 1989.
- [DDP03] Ariel Daliot, Danny Dolev and Hanna Parnas. Self-Stabilizing Pulse Synchronization Inspired by Biological Pacemaker Networks. In Proceedings of the 6th International Symposium on Self-Stabilizing Systems (SSS'03), LNCS 2704, 2003, p. 32-48.
- [DDP03b] Ariel Daliot, Danny Dolev and Hanna Parnas. Linear Time Byzantine Self-Stabilizing Clock Synchronization. In Proceedings of the 7th International Conference on Principles of Distributed Systems, LNCS 3144, 2003, p. 7-9. [A revised version appears in Cornell ArXiv: http://arxiv.org/abs/cs.DC/0608096\/]
- [DHS86] Danny Dolev, Joseph Y. Halpern and H. Raymond Strong. On the Possibility and Impossibility of Achieving Clock Synchronization 32:230-250, 1986.
- [DW93] Shlomi Dolev and Jennifer L. Welch. Wait-Free Clock Synchronization. In Proceeding of the 12th Annual {ACM} Symposium on Principles of Distributed Computing (PODC'93), 1993, p. 97—108.
- [FC95] Christof Fetzer and Flaviu Cristian. An optimal internal clock synchronization algorithm. In Proceedings 10th Annual IEEE Conference on Computer Assurance, Gaithersburg, MD, June 1995.
- [FC97b] Christof Fetzer and Flaviu Cristian. Integrating external and internal clock synchronization. J. Real-Time Systems, 12(2):123--172, March 1997.
- [FL04] Rui Fan and Nancy Lynch. Gradient Clock Synchronization. In Proceedings of the Twenty-Third Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, St. John's, Newfoundland, Canada, 2004, p. 320-327.
- [FFSK06] Markus Ferringer, Gottfried Fuchs, Andreas Steininger and Gerald Kempf. VLSI Implementation of a Fault-Tolerant Distributed Clock Generation. In Proceedings IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT2006), 2006.
- [FSFK06] Matthias Fuegger, Ulrich Schmid, Gottfried Fuchs and Gerald Kempf. Fault-Tolerant Distributed Clock Generation in VLSI Systems-on-Chip. In Proceedings of the Sixth European Dependable Computing Conference (EDCC-6), Coimbra, Portugal, 2006, p. 87—96.

References

- [HS97] Dieter Hoechtl and Ulrich Schmid. Long-term evaluation of GPS timing receiver failures. In Proceedings of the 29th IEEE Precise Time and Time Interval Systems and Application Meeting (PTTI'97), pages 165--180, Long Beach, California, December 1997.
- [KO87] Hermann Kopetz and Wilhelm Ochsenreiter. Clock synchronization in distributed real-time systems. IEEE Transactions on Computers C36(8):933-939, 1987.
- [Lis93] Barbara Liskov. Practical uses of synchronized clocks in distributed systems. Distributed Computing 6:211-219, 1993.
- [LWL88] Jennifer Lundelius-Welch and Nancy A. Lynch. A new fault-tolerant algorithm for clock synchronization. Information and Computation, 77(1):1--36, 1988.
- [Mar84] Keith A. Marzullo. Maintaining the Time in a Distributed System: An Example of a Loosely-Coupled Distributed Service. PhD dissertation, Stanford University, Department of Electrical Engineering, February 1984.
- [Mil95] David L. Mills. Improved algorithms for synchronizing computer network clocks. IEEE Transactions on Networks, pages 245--254, June 1995.
- [SC90] F. Schmuck and F. Christian. Continuous amortization need not affect the precision of a clock synchronization algorithm. In Proceedings of the 9th Annual ACM Symposium on Principles on Distributed Computing (PODC), pages 133--144, Quebec City, Canada, August 1990.
- [Sch86] Fred B. Schneider. A paradigm for reliable clock synchronization. In Proceedings Advanced Seminar of Local Area Networks, pages 85--104, Bandol, France, April 1986.
- [Sch97a] Ulrich Schmid. Interval-based clock synchronization with optimal precision. Information and Computation 186(1), 2003, p. 36-77.
- [Sch97b] Ulrich Schmid. Orthogonal accuracy clock synchronization. Chicago Journal of Theoretical Computer Science 3, 2000, p. 3-77.
- [Sch97] Klaus Schossmaier. An interval-based framework for clock rate synchronization algorithms. In Proceedings 16th ACM Symposium on Principles of Distributed Computing, pages 169--178, St. Barbara, USA, August 1997.
- [SHK99] Ulrich Schmid, Martin Horauer, and Nikolaus Keroe. How to distribute GPS-time over COTS-based LANs. In Proceedings\ of the 31th IEEE Precise Time and Time Interval Systems and Application Meeting (PTTI'99), Dana Point, California, December 1999, p. 545-560

References

- [SKMNCK99] Ulrich Schmid, Johann Klasek, Thomas Mandl, Herbert Nachtnebel, Gerhard R. Cadek, and Nikolaus Keroe. A Network Time Interface M-Module for distributing GPS-time over LANs. J. Real-Time Systems, 18(1), 2000, p. 24-57.
- [SS97] Ulrich Schmid and Klaus Schossmaier. Interval-based clock synchronization. J. Real-Time Systems, 12(2):173--228, March 1997.
- [SS99] Ulrich Schmid and Klaus Schossmaier. How to reconcile fault-tolerant interval intersection with the Lipschitz condition. Distributed Computing 14(2):101-111. 2001.
- [SSHL97] Klaus Schossmaier, Ulrich Schmid, Martin Horauer, and Dietmar Loy. Specification and implementation of the Universal Time Coordinated Synchronization Unit (UTCSU). J. Real-Time Systems, 12(3):295--327, May 1997.
- [ST87] T. K. Srikanth and Sam Toueg. Optimal clock synchronization. Journal of the ACM, 34(3):626--645, July 1987.
- [SW99] Klaus Schossmaier and Bettina Weiss. An Algorithm for Fault-Tolerant Clock State & Rate Synchronization. In Proceedings 18th {IEEE} Symposium on Reliable Distributed Systems (SRDS'99), Lausanne, Switzerland, 1999, p. 36-47.
- [VRC97] Paulo Verissimo, Luis Rodrigues, and Antonio Casimiro. CesiumSpray: a precise and accurate global clock service for large-scale systems. J. Real-Time Systems, 12(3):243--294, 1997.
- [WS07] Josef Widder and Ulrich Schmid. Booting Clock Synchronization in Partially Synchronous Systems with Hybrid Process and Link Failures. Distributed Computing 20(2):115-140, 2007.