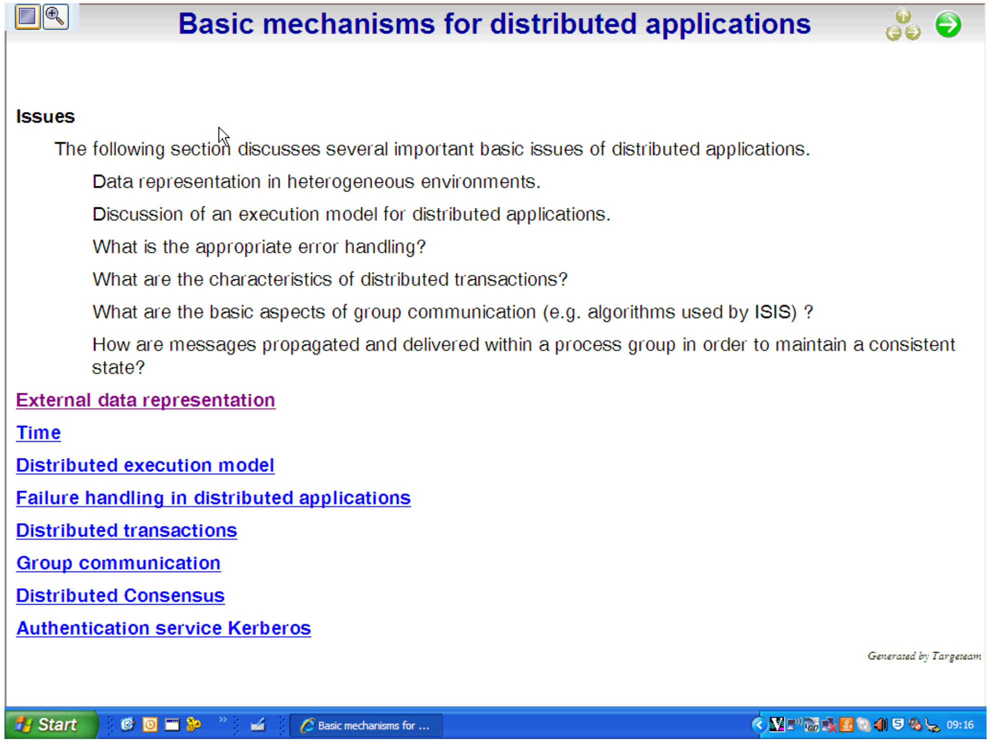
Script **generated by TTT**

Title: Distributed_Applications (04.06.2012)

Date: Mon Jun 04 09:16:14 CEST 2012

Duration: 44:54 min

Pages: 14



Basic mechanisms for distributed applications

Issues

The following section discusses several important basic issues of distributed applications.

- Data representation in heterogeneous environments.
- Discussion of an execution model for distributed applications.
- What is the appropriate error handling?
- What are the characteristics of distributed transactions?
- What are the basic aspects of group communication (e.g. algorithms used by ISIS) ?
- How are messages propagated and delivered within a process group in order to maintain a consistent state?

External data representation

Time

Distributed execution model

Failure handling in distributed applications

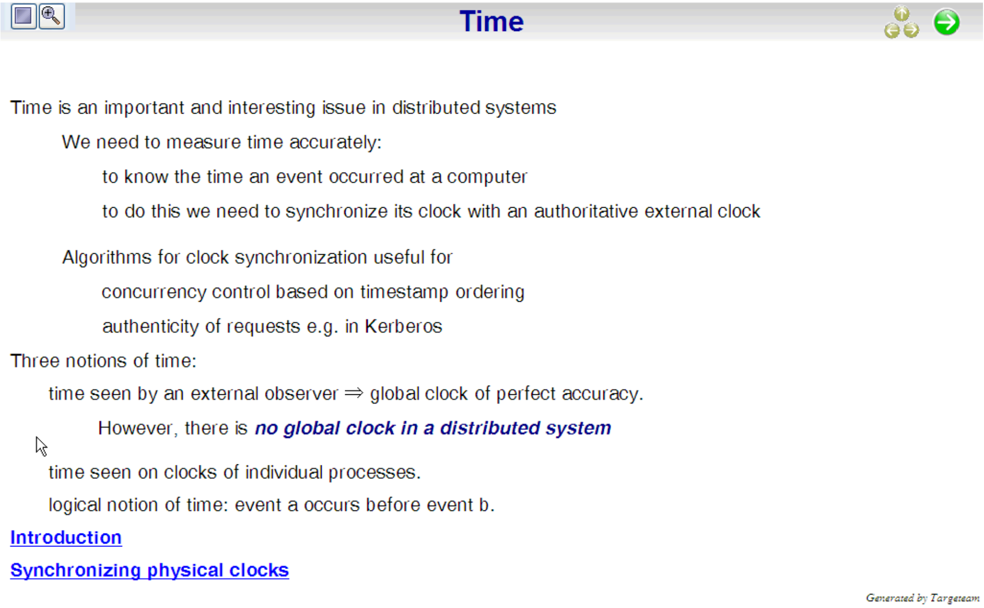
Distributed transactions

Group communication

Distributed Consensus

Authentication service Kerberos

Generated by Targeteam



Time

Time is an important and interesting issue in distributed systems

- We need to measure time accurately:
  - to know the time an event occurred at a computer
  - to do this we need to synchronize its clock with an authoritative external clock
- Algorithms for clock synchronization useful for
  - concurrency control based on timestamp ordering
  - authenticity of requests e.g. in Kerberos

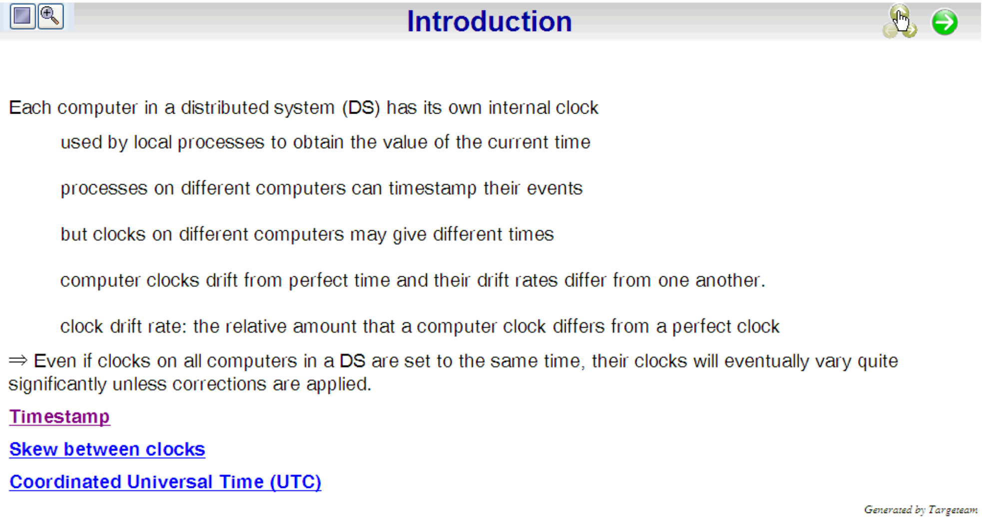
Three notions of time:

- time seen by an external observer ⇒ global clock of perfect accuracy.
  - However, there is ***no global clock in a distributed system***
- time seen on clocks of individual processes.
- logical notion of time: event a occurs before event b.

Introduction

Synchronizing physical clocks

Generated by Targeteam



Introduction

Each computer in a distributed system (DS) has its own internal clock

- used by local processes to obtain the value of the current time
- processes on different computers can timestamp their events
- but clocks on different computers may give different times
- computer clocks drift from perfect time and their drift rates differ from one another.
- clock drift rate: the relative amount that a computer clock differs from a perfect clock

⇒ Even if clocks on all computers in a DS are set to the same time, their clocks will eventually vary quite significantly unless corrections are applied.

Timestamp

Skew between clocks

Coordinated Universal Time (UTC)

Generated by Targeteam

To timestamp events, we use the computer's clock

1. At real time t, the operating system reads the time on the computer's hardware clock H_i (t)
2. It calculates the time on its software clock

   C_i (t)= a H_i (t) + b

   e.g. a 64 bit number giving nanoseconds since some "base time"

   in general, the clock is not completely accurate,

   but if C_i behaves well enough, it can be used to timestamp events at p_i

---

Introduction

Each computer in a distributed system (DS) has its own internal clock

used by local processes to obtain the value of the current time

processes on different computers can timestamp their events

but clocks on different computers may give different times

computer clocks drift from perfect time and their drift rates differ from one another.

clock drift rate: the relative amount that a computer clock differs from a perfect clock

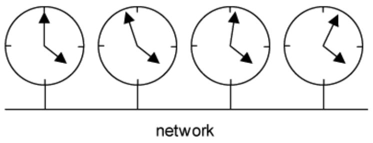
⇒ Even if clocks on all computers in a DS are set to the same time, their clocks will eventually vary quite significantly unless corrections are applied.

Timestamp

Skew between clocks

Coordinated Universal Time (UTC)

Skew between clocks



Computer clocks are not generally in perfect agreement.

Skew: the disagreement between two clocks (at any instant).

Computer clocks are subject to clock drift (they count time at different rates).

Clock drift rate: the difference per unit of time from some ideal reference clock

Ordinary quartz clocks drift by about 1 sec in 11-12 days.

Coordinated Universal Time (UTC)

International Atomic Time is based on very accurate physical clocks (drift rate 10^{-13}).

UTC is an international standard for time keeping

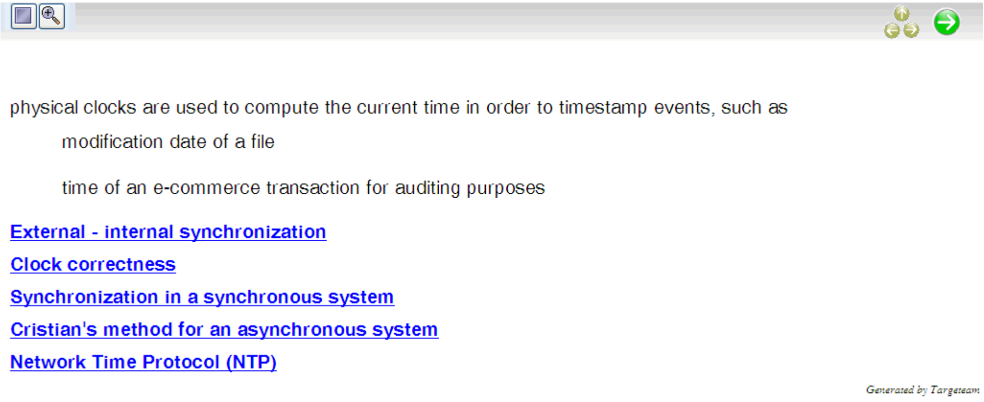
It is based on atomic time, but occasionally adjusted to astronomical time

It is broadcast from radio stations on land and satellite (e.g. GPS)

Computers with receivers can synchronize their clocks with these timing signals

Signals from land-based stations are accurate to about 0.1-10 millisecond

Signals from GPS are accurate to about 1 microsecond



physical clocks are used to compute the current time in order to timestamp events, such as

- modification date of a file
- time of an e-commerce transaction for auditing purposes

External - internal synchronization

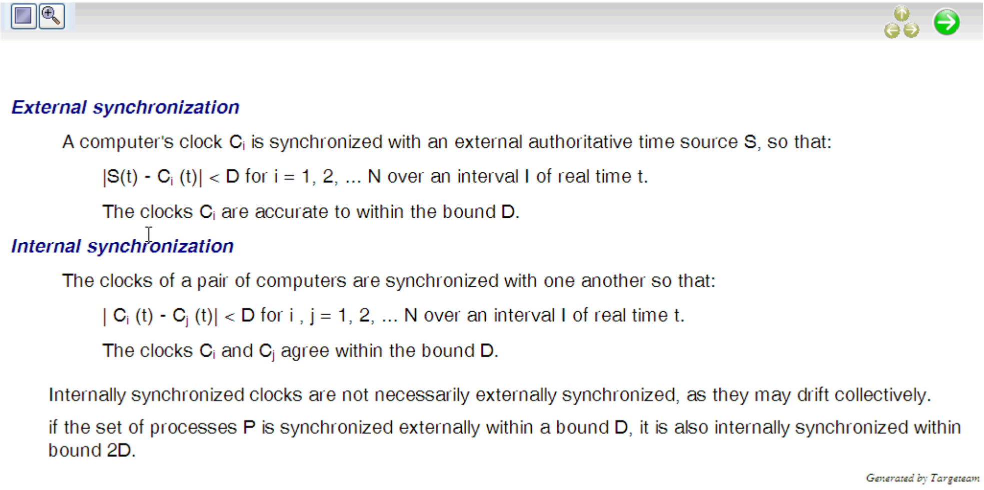
Clock correctness

Synchronization in a synchronous system

Cristian's method for an asynchronous system

Network Time Protocol (NTP)

Generated by Targeteam



External synchronization

A computer's clock C_i is synchronized with an external authoritative time source S, so that:

$|S(t) - C_i(t)| < D$ for i = 1, 2, ... N over an interval I of real time t.

The clocks C_i are accurate to within the bound D.

Internal synchronization

The clocks of a pair of computers are synchronized with one another so that:

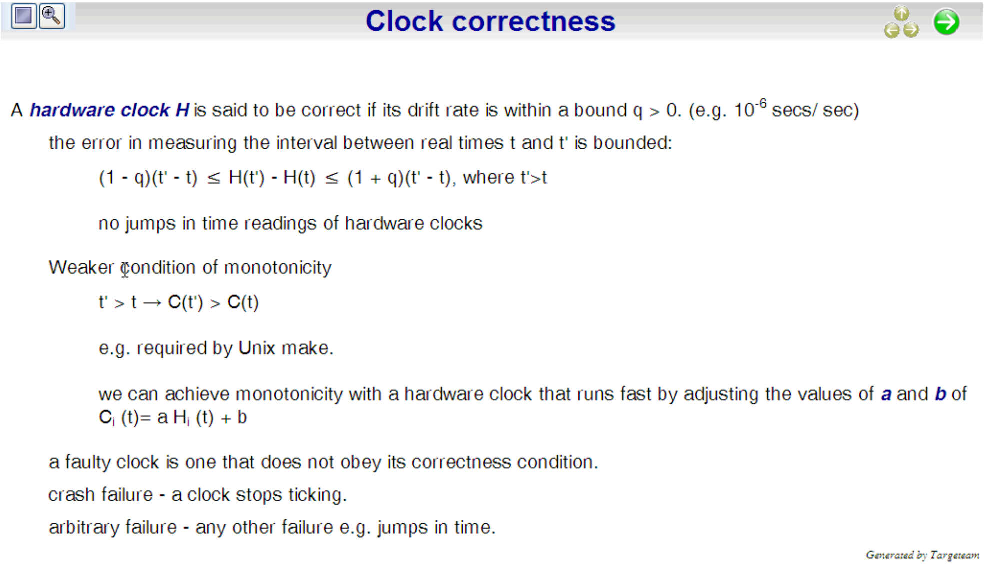
$| C_i(t) - C_j(t)| < D$ for i , j = 1, 2, ... N over an interval I of real time t.

The clocks C_i and C_j agree within the bound D.

Internally synchronized clocks are not necessarily externally synchronized, as they may drift collectively.

if the set of processes P is synchronized externally within a bound D, it is also internally synchronized within bound 2D.

Generated by Targeteam



Clock correctness

A ***hardware clock H*** is said to be correct if its drift rate is within a bound q > 0. (e.g. 10^{-6} secs/ sec)

the error in measuring the interval between real times t and t' is bounded:

$(1 - q)(t' - t) \leq H(t') - H(t) \leq (1 + q)(t' - t)$, where t'>t

no jumps in time readings of hardware clocks

Weaker condition of monotonicity

$t' > t \rightarrow C(t') > C(t)$

e.g. required by Unix make.

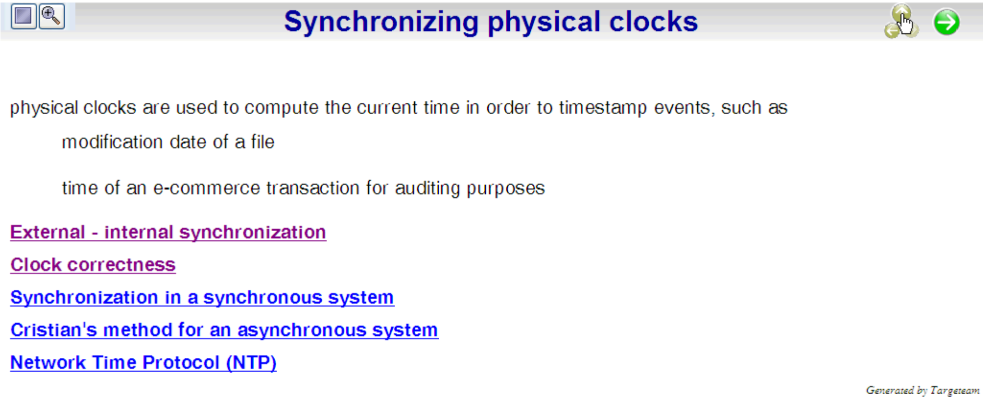
we can achieve monotonicity with a hardware clock that runs fast by adjusting the values of ***a*** and ***b*** of $C_i(t)= a H_i(t) + b$

a faulty clock is one that does not obey its correctness condition.

crash failure - a clock stops ticking.

arbitrary failure - any other failure e.g. jumps in time.

Generated by Targeteam



Synchronizing physical clocks

physical clocks are used to compute the current time in order to timestamp events, such as

- modification date of a file
- time of an e-commerce transaction for auditing purposes

External - internal synchronization

Clock correctness

Synchronization in a synchronous system

Cristian's method for an asynchronous system

Network Time Protocol (NTP)

Generated by Targeteam

Synchronizing physical clocks

physical clocks are used to compute the current time in order to timestamp events, such as

modification date of a file

time of an e-commerce transaction for auditing purposes

External - internal synchronization

Clock correctness

Synchronization in a synchronous system

Cristian's method for an asynchronous system

Network Time Protocol (NTP)

A time server S receives signals from a UTC source

Process p requests time in m1 and receives t in m2 from S.

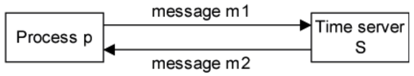
p sets its clock to: t + T_{round} /2

Accuracy is +/- (T_{round} /2 - min).

because the earliest time S puts t in message m2 is min after p sent m1: t + min

the latest time was min before m2 arrived at p: t + T_{round} - min

the time by S's clock when reply message m2 arrives is in the range [t + min, t + T_{round} - min]



Discussion

The approach has several problems. It is only suitable for deterministic LAN environment or Intranet.

- a single time server might fail
  - ⇒ redundancy through group of servers, multicast requests
- it does not deal with faulty time servers
  - how to decide if replies vary (byzantine agreement problems)
- imposter providing false clock readings

Cristian's method for an asynchronous system

Observations:

round trip times between processes are often reasonably short in practice, yet theoretically unbounded

practical estimate possible if round-trip times are sufficiently short in comparison to required accuracy

Approach

Berkeley algorithm

Both algorithms (Cristian and Berkeley) are not really suitable for Internet.