Asynchronous systems

Assume no bounds on:
- clock drift
- processing time
- message passing time
Goals

- How do we make sure that algorithms are correct?
- Why are algorithms correct?
With synchronous rounds:

- Simple proofs by induction
- Local state easily reflects global state

Goals

- How do we make sure that algorithms are correct?
- Why are algorithms correct?
Are solutions obtained with the synchronous system applicable?

Not really...

- The practitioner's argument
- The theoretician's argument
In practice

- Tight synchronous limits are dangerous:
  - Round time proportional to mean delay
  - Low coverage or expensive systems

- Large synchronous limits are not useful:
  - Round time proportional to high percentile delay
  - Taking advantage of synchrony causes a very large performance penalty
In practice

• Solutions for asynchronous systems might have better performance:
  • Round time proportional to mean delay
  • Even if more message exchanges are necessary

Typical delay distribution

- mean
- high percentile

frequency

time
In theory

- Start with a synchronous reliable fully connected network
- Relax the system model:
  - Unbounded message loss
  - Large/unknown graph diameter
  - Dynamic graph
- Example: Leader election
Example: Leader election

- Static known participants
- Synchronous Reliable static
- Synchronous Reliable clique
- Synchronous Bounded unreliable Clique
- Asynchronous Reliable clique
- Synchronous Reliable connected Unknown diameter
- Synchronous Reliable dynamic
- Synchronous Unreliable clique
- Connected
- Disconnected
- Can loose all messages

© 2007-2010 José Orlando Pereira
GSD/DI/U.Minho
Example: Leader election

- **Trivial**
  - Static known participants
- **Possible**
  - Synchronous Reliable static
  - Synchronous Reliable clique
  - Synchronous Bounded unreliable Clique
  - Synchronous Reliable connected Unknown diameter
- **Possible (eventually)**
  - Synchronous Reliable dynamic
  - Synchronous Unreliable clique
  - Asynchronous Reliable clique
  - Can lose all messages

**Impossible**
- Connected
- Disconnected
In theory

- Asynchronism subsumes:
  - Heterogeneity
  - Dynamics
  - Uncertainty

- Much simpler than handling them explicitly

- Often considered an Universal model:
  - Widely applicable solutions
Sample computation

- An alarm clock program:

  ```
  main: // line 1
  cnt:=3 // line 2
  while cnt>0: // line 3
      sleep 1s // line 4
      cnt := cnt-1 // line 5
  ring // line 6
  ```
Observation

- Select model variables and periodically observe the system:
Choose observation that allows reasoning on the desired properties:

- cnt = 3
- cnt = 2
- cnt = 1
- cnt = 0

vcnt := 3
vcnt := 2
vcnt := 1
vcnt := 0  END
Consider all possible sequences of chosen atomic actions:
Safety properties

- Nothing bad ever happens:

\[\text{vcnt}:=3\]
\[\text{vcnt}:=1\]
\[\text{vcnt}:=2\]
\[\text{vcnt}:=3\]
\[\text{vcnt}:=1\]
\[\text{vcnt}:=1\]
\[\text{vcnt}:=1\]
\[\text{vcnt}:=2\]
\[\text{vcnt}:=4\]
\[\text{vcnt}:=2\]
\[\text{vcnt}:=2\]
\[\text{vcnt}:=1\]
\[\text{vcnt}:=1\]
\[\text{vcnt}:=1\]
\[\text{vcnt}:=1\]
\[\text{vcnt}:=0\]
\[\text{END}\]

- OK!

- OK!
Liveness properties

Something good eventually\(^(*)\) happens:

\[\text{vcnt:=3} \rightarrow \text{vcnt:=2} \rightarrow \text{vcnt:=1} \rightarrow \text{vcnt:=0 END}\]

\[\text{vcnt:=3} \rightarrow \text{vcnt:=2} \rightarrow \text{vcnt:=1} \rightarrow \text{vcnt:=1} \rightarrow \text{vcnt:=1} \rightarrow \ldots\]

\(^(*)\) eventually = inevitavelmente ≠ eventualmente
Specification is a set of allowable behaviors:

\[
S = \{ \text{vcnt} = 3 \rightarrow \text{timeout}, \text{vcnt} = 2 \rightarrow \text{timeout}, \text{vcnt} = 1 \rightarrow \text{timeout}, \text{vcnt} = 0 \rightarrow \text{END} \}
\]
Goal 1: Is it correct?

- Is there a convenient representation for specification sets?
  - Compact
  - Practical
- How to prove safety and liveness properties?