What is a distributed system?

“A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.”

Leslie Lamport

Goals (and auto-goals) of a Distributed System

- Connecting Resources and Users
- Transparency
- Openness
- Scalability
**Transparency**

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hides differences in data representation and invocation mechanisms</td>
</tr>
<tr>
<td>Location</td>
<td>Hides where an object resides</td>
</tr>
<tr>
<td>Migration</td>
<td>Hides from an object that object's location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hides from a client the change of location of an object to which the client is bound</td>
</tr>
<tr>
<td>Replication</td>
<td>Hides that an object may be replicated, with replicas at different locations</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hides coordination of activities between objects</td>
</tr>
<tr>
<td>Failure</td>
<td>Hides the failure and recovery of object</td>
</tr>
<tr>
<td>Persistence</td>
<td>Hides whether a resource is in memory or on disk</td>
</tr>
</tbody>
</table>

**Openness**

- Easily interact with other open systems
- Conform to well-defined interfaces
- Achieve independence in heterogeneity wrt
  - Hardware
  - Platform
  - Languages
- Support different app/user-specific policies
  - ideally, provide only mechanisms!

**Scalability**

- Size scalability
  - number of users and processes
- Geographical scalability
  - maximum distance between nodes
- Administrative scalability
  - number of administrative domains
A first course in Distributed Computing...

- Two basic approaches
- cover many interesting systems, and distill from them fundamental principles
- focus on a deep understanding of the fundamental principles, and see them instantiated in a few systems

A few intriguing questions

- How do we talk about a distributed execution?
- Can we draw global conclusions from local information?
- Can we coordinate operations without relying on synchrony?
- For the problems we know how to solve, how do we characterize the “goodness” of our solution?
- Are there problems that simply cannot be solved?
- What are useful notions of consistency, and how do we maintain them?
- What if part of the system is down? Can we still do useful work? What if instead part of the system becomes “possessed” and starts behaving arbitrarily: All bets are off?