Distributed System Challenges  [ can you give an example of each? ]
(1) Relevant information is scattered among multiple machines.
(2) Processes make decisions based only on locally available information.
(3) A single point of failure in the system should be avoided.
(4) No common clock or other precise global time source exists.

(1) - (3) say we can't have a single server for “whatever”.  
(4) says we need some way to determine the “before” relation.  
(Consider 'make' when server's clock and local machine's clock are out of sync.)

Mutual Exclusion in Distributed Systems

**Key point:** no shared memory, so locks, semaphores, and monitors don’t work © Use message passing!

Approach 1: Centralized  
Have a single “lock” machine.  
To *enter* critical section, send lock machine a request and await its reply.  [when is reply sent? ]  
To *exit* send lock machine an “I'm done” message.  
Delay before entry (in messages): 2.  
Problem: lock machine is a single point of failure!

Approach 2: Token Ring  
Pass a single token around a ring of machines.  
To *enter* critical section await the token and then hold it.  
To *exit* pass the token on to the next machine in the ring.  
Delay before entry (in messages): 0 to n-1.  
Problem: lost token.

Election Algorithms to identify a temporary coordinator.
For example, to compute the means of means.  
(Assume all processes have a unique number and that each knows the other processes. We just don't know which processes are up.)
**Bully Algorithm**

For P to hold an election

1. P sends an *I'M_THE_BOSS* message to every process in the group with a higher number than P.
2. Each higher numbered process sends P a *NOPE* message and then starts at (1) (P's job is done).
3. If no one responds, then P is the coordinator!

For example:

![Diagram of Bully Algorithm]

**Ring Algorithm**

For P to hold an election

1. P builds a message with its number.
2. P finds the next process in the list that is up and sends it the message.
3. Upon receipt of a message
   a) if your number is not in the message, add it and then pass the message on as in (2).
   b) otherwise send a *COORDINATOR* message to everyone on the list (the highest number on the list is the new coordinator).

*Note the two processes can start this at the same time with no ill effect.*

![Diagram of Ring Algorithm]

**Final Thought: a Common Implementation Mechanism** - Remove Procedure Call - SOAP