COP 4600 – Summer 2011

Introduction To Operating Systems

Distributed Process Management – Part 1

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop4600/sum2011

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

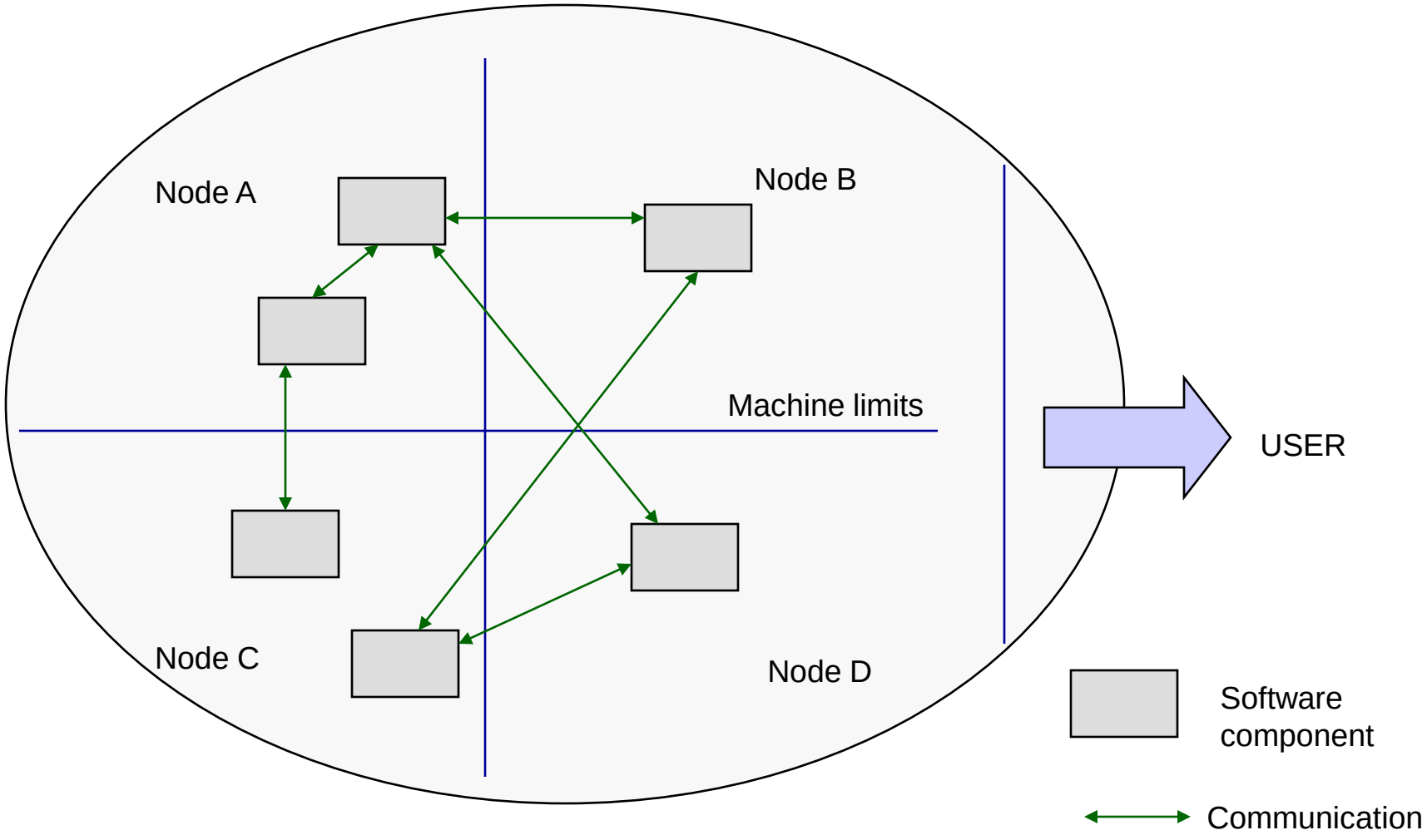
Distributed Process Management

- In this set of notes we'll examine some of the key mechanisms used in distributed operating systems.
- Process migration: the movement of an active process from one machine in the network to another machine in the network.
- Distributed global states: how processes on different systems can coordinate their activities when each is governed by a local clock and when the network imposes a delay in the exchange of information.
- Distributed mutual exclusion: how to ensure mutually exclusive access in a distributed environment.
- Distributed deadlock: how to prevent or detect and resolve deadlock in a distributed environment.

What is a Distributed System?

- While many different definitions of what constitutes a distributed system have been put forth, there is general consensus that there are several central components that a distributed system must contain:
  - A set of autonomous computers.
  - A communication network, connecting those computers.
  - Software which integrates these components with a communication system.

What is a Distributed System?



Important Characteristics of Distributed Systems

- Based on our simple definition, there are several important characteristics of distributed systems that need a closer look.
- All of these characteristics are based on the concept of transparency.
- In the context of information technology, the concept of transparency literally means that certain things should be invisible to the user. The manner in which the problem is solved is largely irrelevant to the user.
- The following transparency properties play a large role in achieving this result for the user:

Transparency Properties of Distributed Systems

Location Transparency – users do not necessarily need to know where exactly within the system a resource is located which they wish to utilize. Resources are typically identified by name, which has no bearing on their location.

Access Transparency – the way in which a resource is access is uniform for all resources. For example, in a distributed database system consisting of several databases of different technologies, there should also be a common user interface (such as SQL).

Replication Transparency – the fact that there may be several copies of a resource is not disclosed to the user. The user has no need to know whether they are accessing the original or the copy. The altering of the resource also must occur transparently.

Transparency Properties of Distributed Systems
(cont.)

Error Transparency – users will not necessarily be informed of all errors occurring in the system. Some errors may be irrelevant, and others may well be masked, as in the case of replication.

Concurrency Transparency – distributed systems are usually used by several users simultaneously. It often happens that two or more users access the same resource at the same time, such as a database table, printer, or file. Concurrency transparency ensures that simultaneous access is feasible without mutual interference or incorrect results.

Migration Transparency – using this form of transparency, resources can be moved over the network without the user noticing. A typical example is today's mobile telephone network in which the device can be moved around freely, without any loss of communication when leaving the vicinity of a sender station.

Transparency Properties of Distributed Systems
(cont.)

Process Transparency – It is irrelevant on which computer a certain task (process) is executed, provided it is guaranteed that the results are the same. This form of transparency is an important prerequisite for the successful implementation of a balanced workload between computers.

Performance Transparency – when increasing the system load, a dynamic reconfiguration may well be required. This measure for performance optimization should be unnoticed by other users.

Scaling Transparency – if a system is to be expanded so as to incorporate more computers or applications, this should be feasible without modifying the system structure or application algorithms.

Language Transparency – the programming language in which the individual subcomponents of the distributed system or application were created must not play any role in the ensemble. This is a fairly new requirement of distributed systems and is only supported by more recently developed systems.

Process Migration

- Process migration is the transfer of a sufficient amount of the state of a process from one computer to another for the process to execute on the target machine.
- Interest in this concept arose from research into methods of load balancing across multiple networked systems.
- True process migration includes the ability to preempt a process on one machine and reactivate it later on another machine.
- Process migration is desirable in a distributed system for a number of reasons, including those listed on the next page.

Motivation For Process Migration

- Load sharing
  - By moving processes from heavily loaded to lightly loaded systems, the load can be balanced to improve overall performance. While empirical data suggests that significant improvements are possible, care must be taken in the design of load-balancing algorithms. The more communication that is necessary for the distributed system to perform the balancing, the worse the performance becomes.
- Communication performance
  - Processes that interact intensively can be moved to the same node to reduce communications cost for the duration of their interaction. Also, if a process is performing data analysis on some file or set of files which are larger than the process's size, it may be advantageous to move the process to the data rather than moving the data to the process.
- Availability
  - Long running processes may need to move to survive in the face of faults for which advance notice is possible or in advance of scheduled down-time. If the OS provides such notification, a process that wants to continue can either migrate to another system or ensure that it can be restarted on the current system at some later time.
- Utilizing special capabilities
  - A process can move to take advantage of unique hardware or software capabilities on a particular node.

Process Migration Mechanisms

- A number of issues need to be addressed in designing a process migration facility.
- Among these issues are the following:
  1. Who initiates the migration?
  2. What portion of the process is migrated?
  3. What happens to outstanding messages and signals?

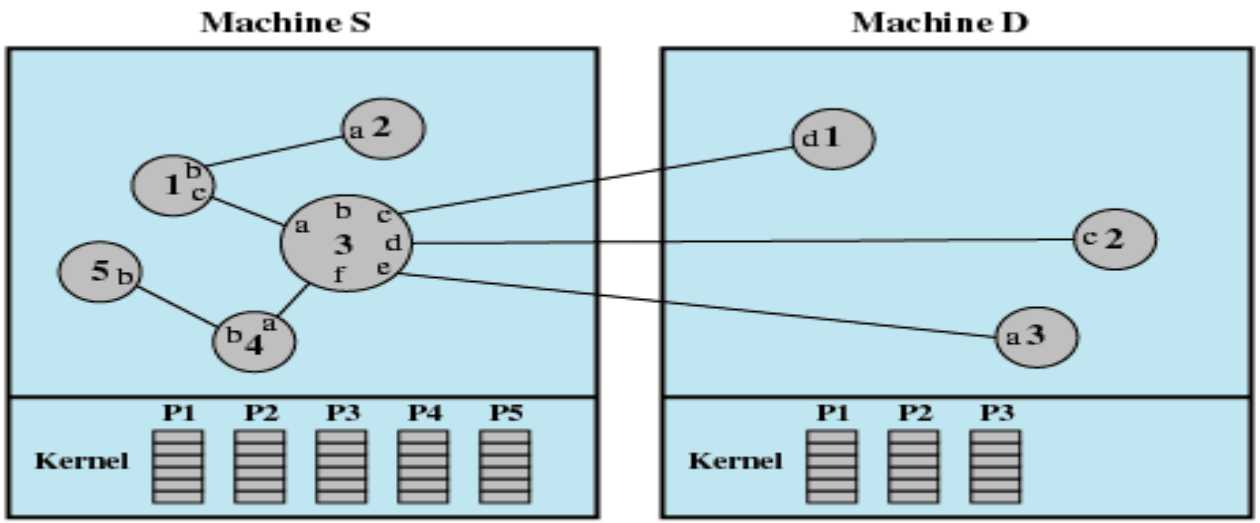
Initiation of Migration

- Who initiates the migration depends on the goal of the migration facility.
- If the goal is load balancing, then some module in the OS that is monitoring system load will generally be responsible for deciding when a migration should take place.
  - The module will be responsible for preempting or signaling a process to be migrated.
  - To determine where to migrate, the module will need to be in communication with peer modules in other systems so that load patterns on other systems can be monitored.
  - In this case, the entire migration function and even the existence of multiple systems may be transparent to the process.
- If the goal is to reach a particular resource, then a process may initiate the migration as the need arises.
  - In this case, the process must be aware of the existence of a distributed system.

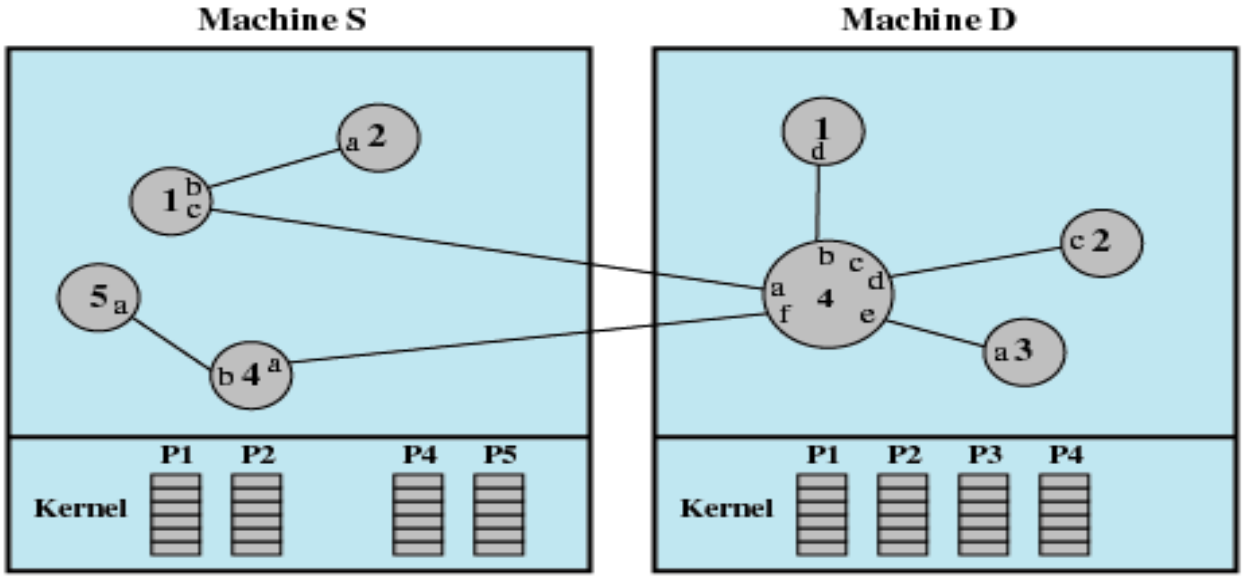
What is Migrated?

- When a process is migrated, it is necessary to destroy the process on the source system and recreate it on the target system.
  - This is a movement of a process and not a replication.
  - Thus, the process image consisting of at least the process control block (PCB) must be moved. In addition, any links between this process and other processes (such as for passing messages and signals) must be updated.
- The figure on the next page illustrates the migration of process 3 out of machine S to become process 4 on machine D.

Example of Process Migration



Before migration



After migration

What is Migrated? (cont.)

- The movement of the PCB is straightforward in process migration. The difficulty, from a performance point of view, concerns the process address space and any open files assigned to the process.
- Let's first consider only the process address space and assume that a virtual memory scheme utilizing paging is being utilized.
- Several different strategies can be employed for what is migrated in this environment.
  - Eager (all) – transfers entire address space
  - Precopy –
  - Eager (dirty) – limited transfer of pages
  - Copy-on-reference
  - Flushing

What is Migrated? (cont.)

- Eager (all): Transfer entire address space at time of migration.
  - No trace of process is left behind in the old system.
  - If address space is large and if the process does not need most of it, then this approach my be unnecessarily expensive.
- Precopy: Process continues to execute on the source node while the address space is copied to the target node.
  - Pages modified on the source during precopy operation have to be copied a second time.
  - Reduces the time that a process is frozen and cannot execute during migration.

What is Migrated? (cont.)

- Eager (dirty): Transfer only the pages of the address space that are in main memory and have been modified.
  - Any additional blocks of the virtual address space are transferred on demand only.
  - While this approach minimizes the amount of data that will be transferred, it does require that the source machine be involved throughout the life of the process by maintaining page table entries. This implies that the process requires remote paging support.

What is Migrated? (cont.)

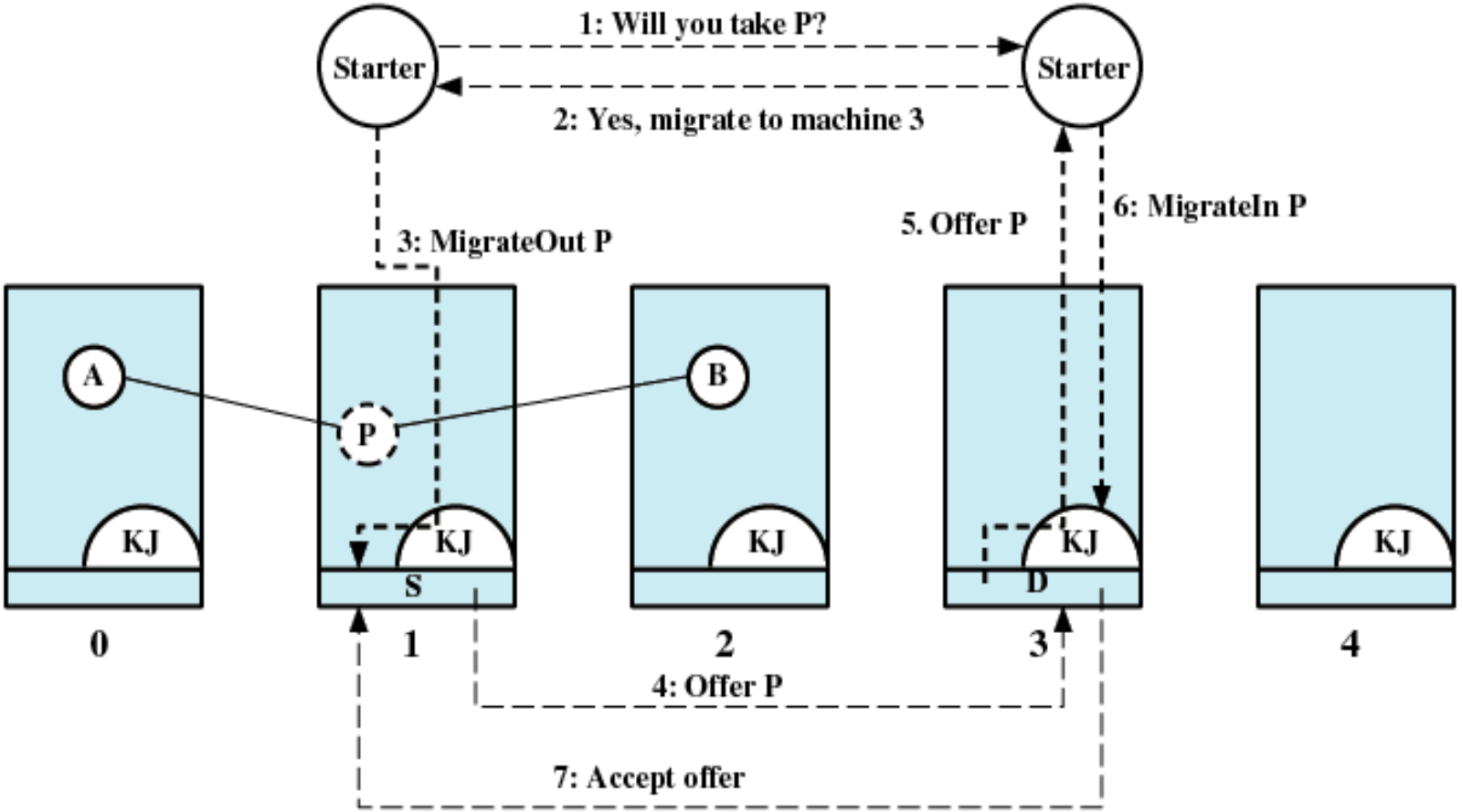
- Copy-on-reference: Pages are only brought over when referenced.
  - Has the lowest initial cost of process migration.

- Flushing: Pages are cleared from main memory of the source by flushing dirty pages to disk.
  - The pages are accessed as needed from disk instead of main memory on the source node.
  - This strategy relieves the source of holding any pages of the migrated process in main memory, immediately freeing a block of memory to be used for other processes.

Negotiation of Migration

- Migration policy is responsibility of Starter utility.
- Starter utility is also responsible for long-term scheduling and memory allocation.
- Decision to migrate must be reached jointly by two Starter processes (one on the source and one on the destination).

Negotiation of Migration (cont.)



Eviction

- Destination system may refuse to accept the migration of a process to itself.
- If a workstation is idle, process may have been migrated to it
  - Once the workstation is active, it may be necessary to evict the migrated processes to provide adequate response time.

Distributed Global States

- Operating system cannot know the current state of all process in the distributed system.
- A process can only know the current state of all processes on the local system.
- Remote processes only know state information that is received by messages.
  - These messages represent the state at some time in the past.
  - Analogous to the situation in astronomy, our knowledge of an object 5 light-years away from the observation point is 5 years-old.

A Running Example

- A bank account is distributed over two branches.
- The total amount in the account is the sum at each branch.
- At 3 PM the account balance is to be determined.
- Messages are sent to request the information.

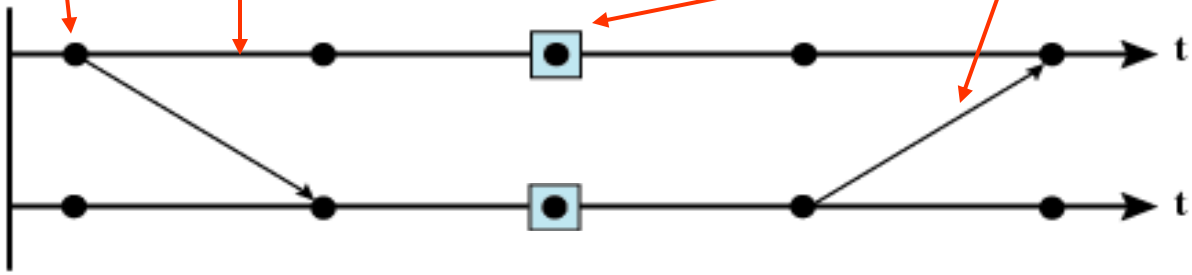
Process State Diagrams

A point on the horizontal line represents an event (e.g. internal process event, message send, message receive.

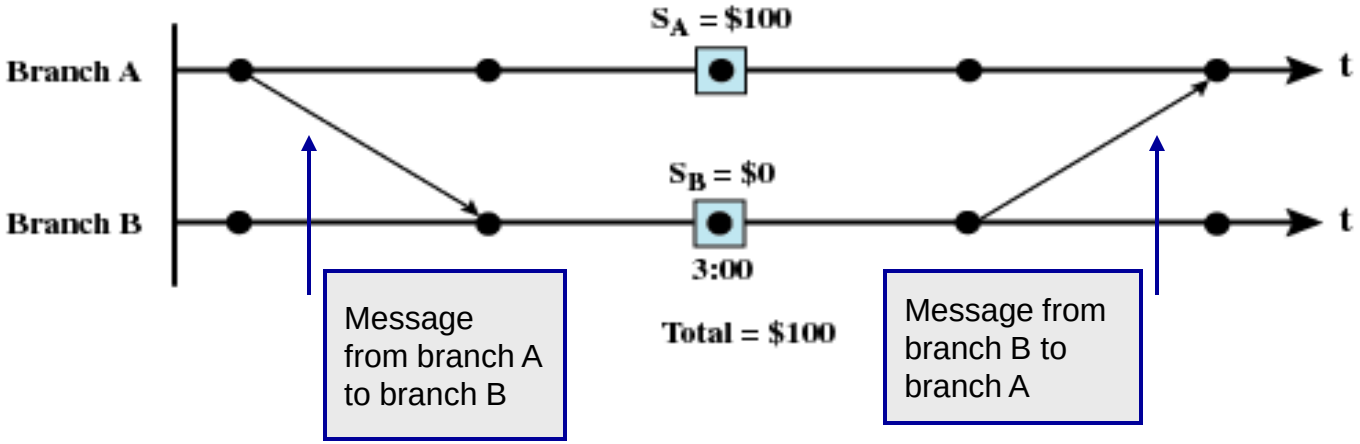
Horizontal lines represent time axis for each process.

A box surrounding a point represents a snapshot of the local process taken at that point in time.

An arrow represents a message between two processes.

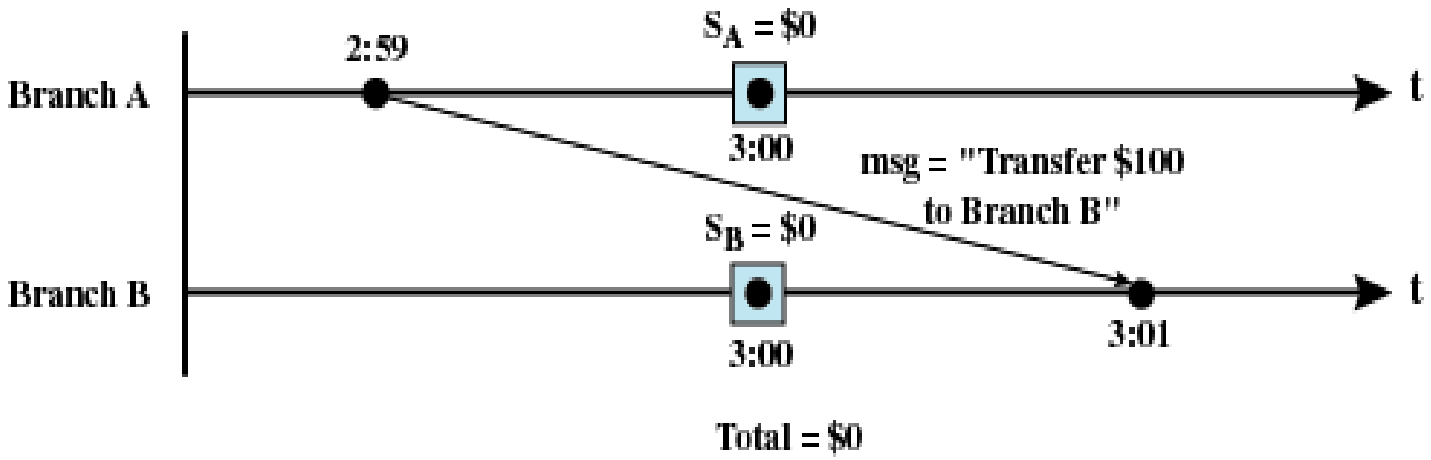


Example Scenarios For Bank Example



One possible scenario – in this case reported account balance is correct

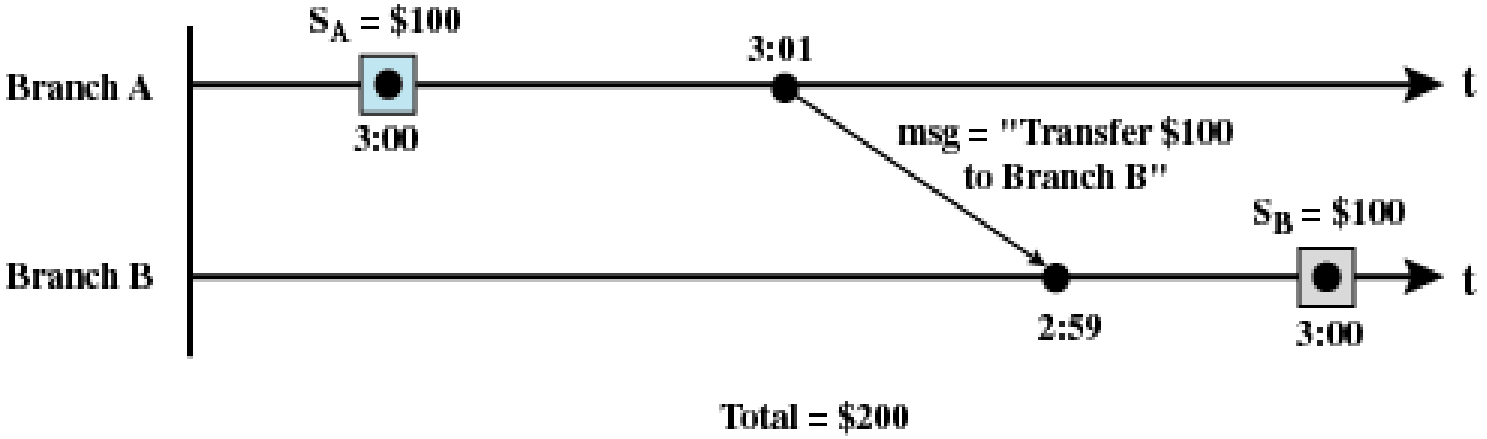
Example Scenarios For Bank Example



If at the time of balance determination, the balance from branch A is in transit to branch B. In this case balance determined at 3:00 pm is incorrect.

All messages in transit must be examined at time of observation. The correct total consists of balance at both branches and amount in any message in transit.

Example Scenarios For Bank Example



If the clocks at the two branches are not perfectly synchronized a problem can arise. Suppose that a transfer message is initiated at branch A at local time 3:01 pm. This message arrives at branch B at 2:59 local time. The balance calculated at 3:00 pm will now show the incorrect amount of $200. The amount is incorrectly counted twice.