CS 351
Design of Large Programs
Concurrent Design Notation

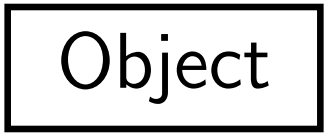
Brooke Chenoweth

University of New Mexico

Spring 2025

Basic Component Notation Revisited

- Passive
  - procedure
  - object
- Active
  - task
  - active object
- Organizational
  - package
- External
  - devices and interfaces



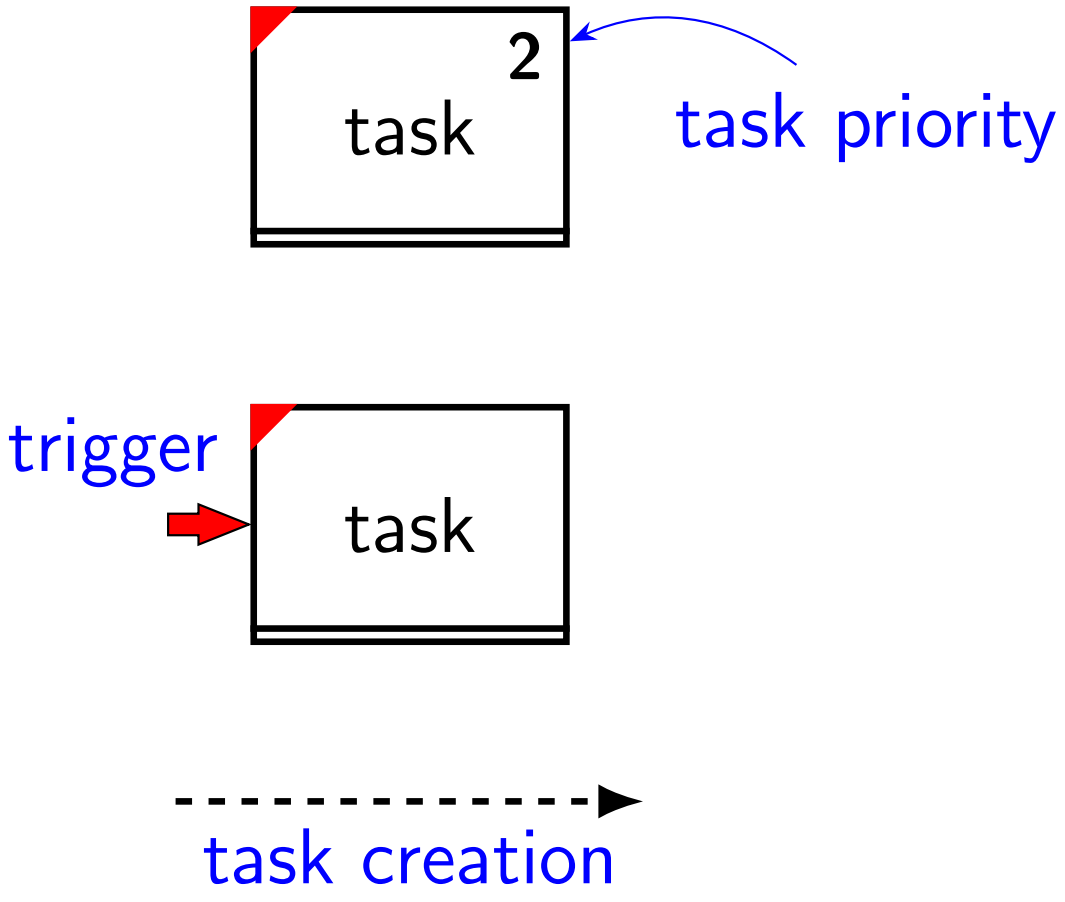
not used so far

used only for main

Extending the Task Notation

- A task is a sequential process having its own independent thread of control
- Tasks may be static (created at initialization) or dynamic (having a life cycle)
- Task execution results in the invocation of methods on objects in the system
- Task execution may be
  - periodic
  - scheduled at a given priority (lower number represents a higher priority)
  - reactive (in response to some actuator)

Extending the Task Notation



Task Activation

- Each actuator is associated with some trigger condition
- Trigger conditions must be defined outside the diagram
- Different notations should be used for different triggers
- Implementability of the trigger mechanism must be established
- Actuators can hide complex implementation details and simplify design understanding

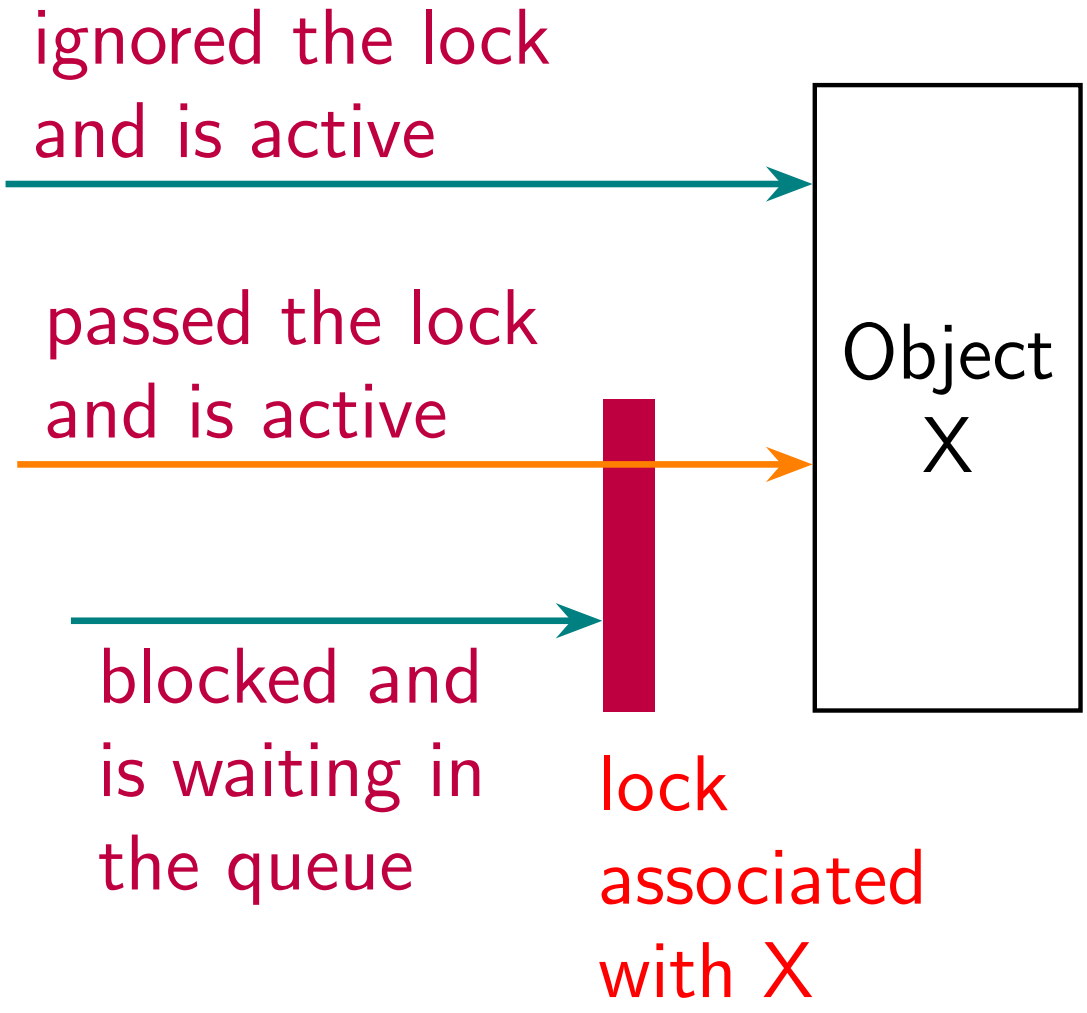
Sample trigger conditions

- clock signal
- timer overflow
- system event (e.g., failure, power up)
- application event
- message arrival
- independently assessed system condition

Synchronization in Java

- Synchronization defines a structured mechanism for coordination among tasks
- Mutual exclusion is a general synchronization mechanism implementable in most systems
- The actual mechanics of synchronization are language specific
- At design level it is convenient to express synchronization by specifying mutual exclusion requirements

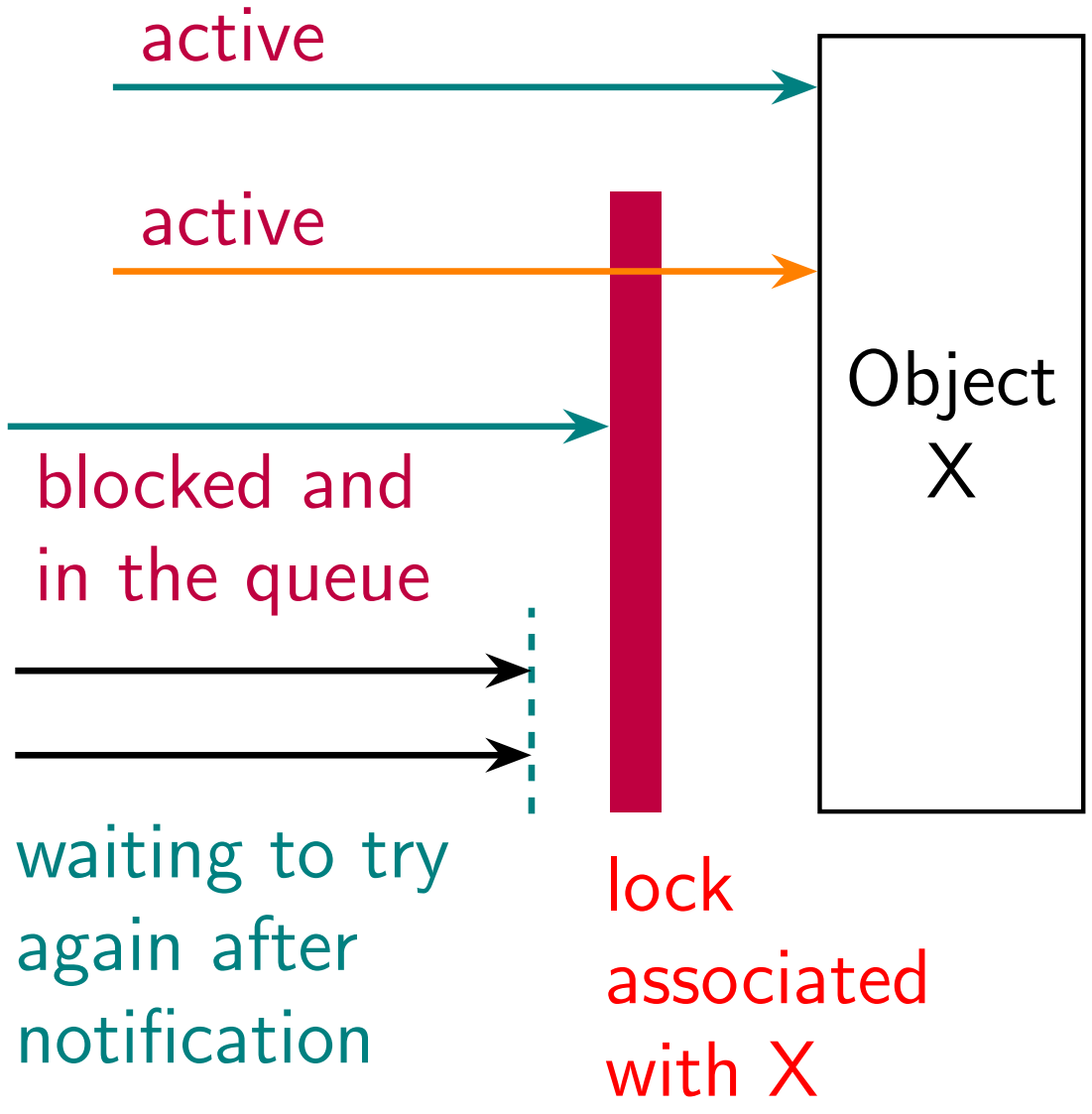
Synchronization in Java



Preview of Java Notification

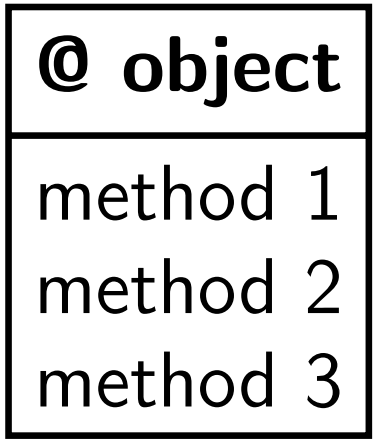
- The `wait(object)` operation places the thread on the wait queue
  - the object lock is released
- A time period may be specified and the thread is removed from the wait queue when the time interval expires
- The `notifyAll(object)` operation removes all the threads from the wait queue
- In both cases the released threads must be scheduled and must acquire the lock

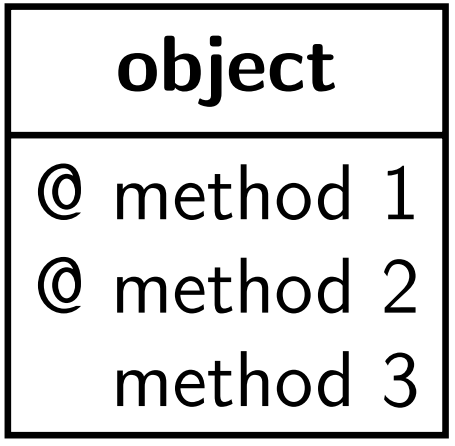
Preview of Java Notification



A Notation for Synchronized Methods

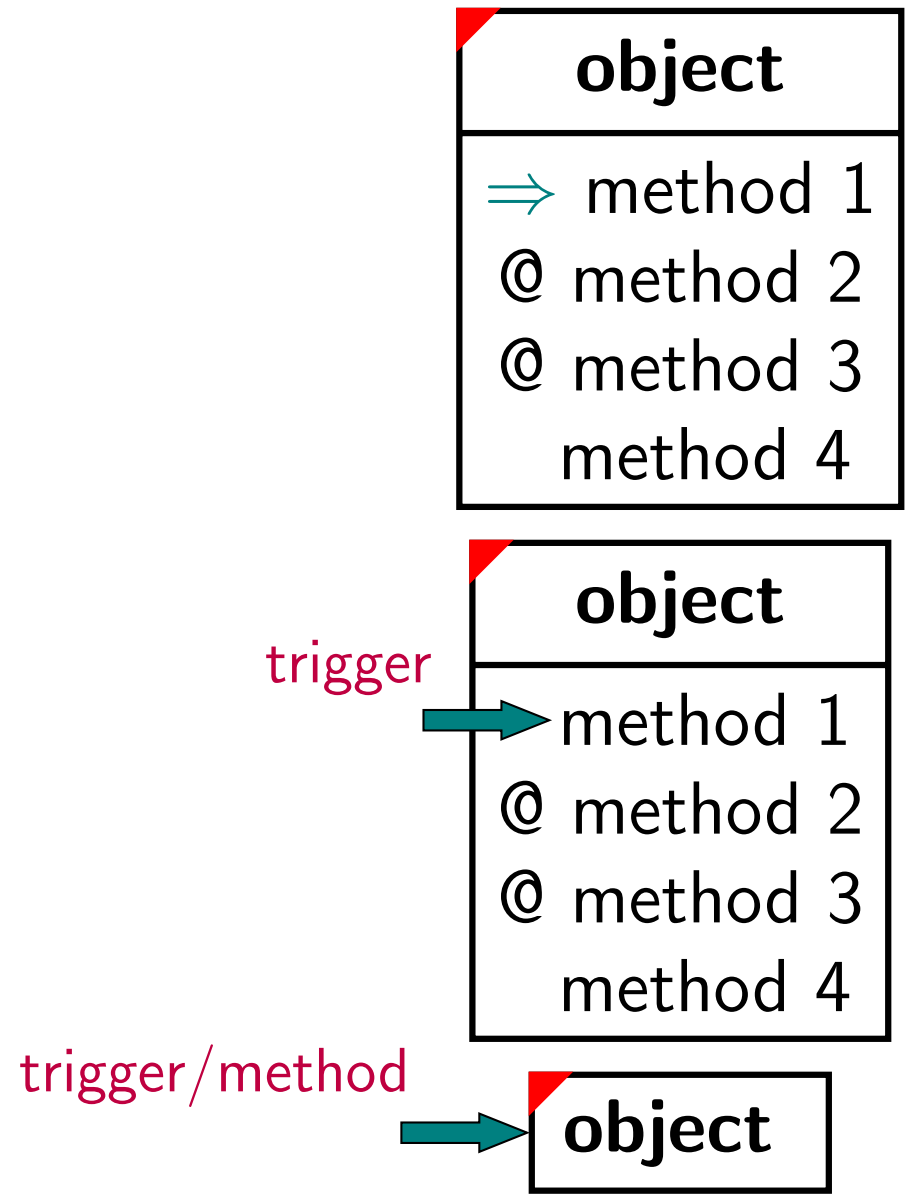
- The notation is the same for classes and objects and so are the semantics
- Notation and semantics may be adjusted for different settings
- Synchronized objects specify mutual exclusion among all methods
- Synchronized methods limit mutual exclusion to identified methods





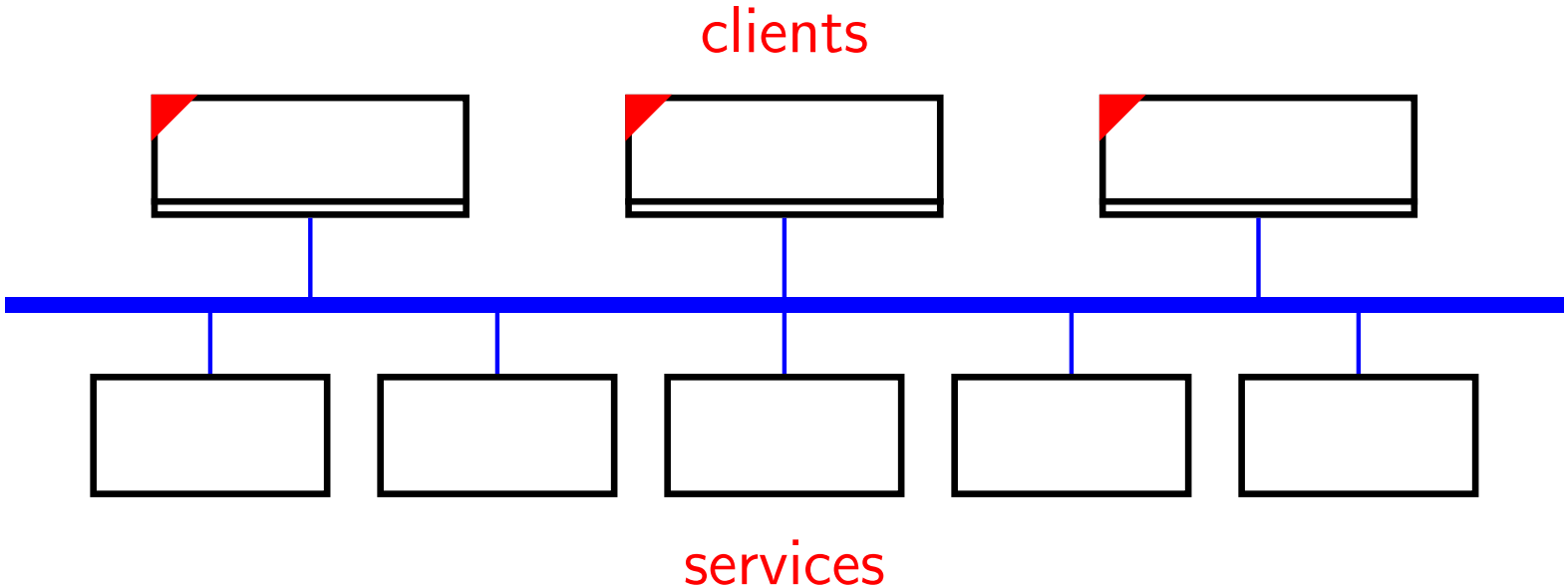
Active Objects

- Active objects exhibit behaviors equivalent to wrapping a task inside an object veneer
  - the goal is to control the execution of a specific method
- The execution of a thread inside an active object may be
  - periodic
  - reactive

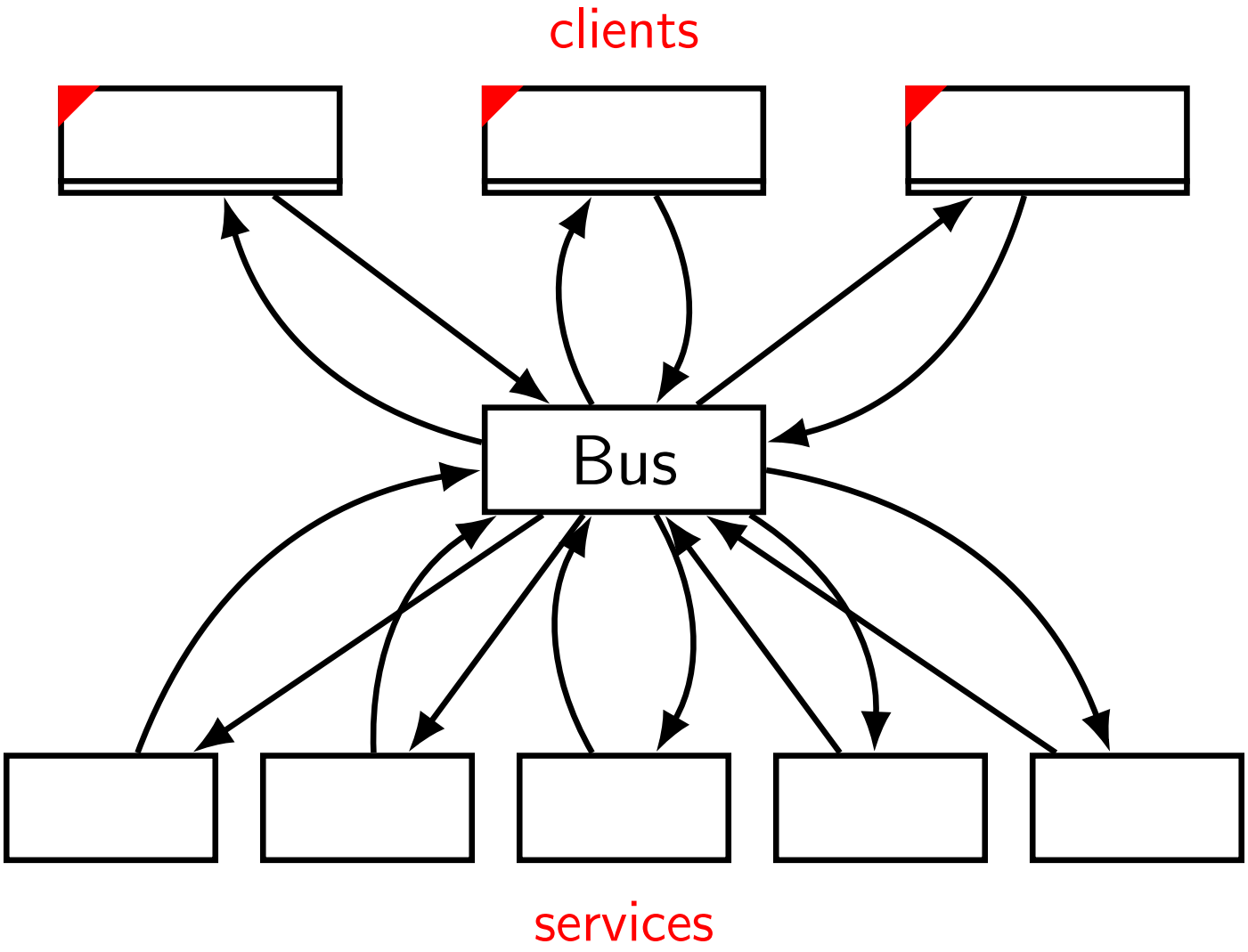


Specialized Connectors

- Triggers eliminate the need for many types of connectors
- Custom connectors can be defined and used in the design diagrams. . . *with the proviso that the semantics of the connector are well defined*
- Example: Software Bus



Software Bus is a Subsystem!



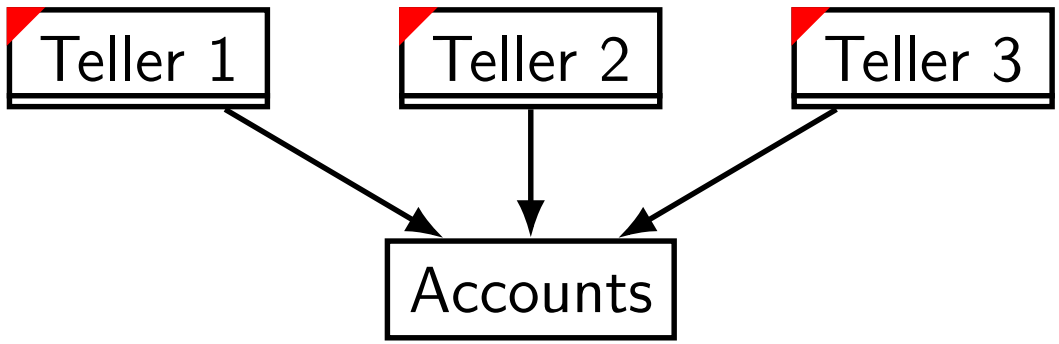
A Simple Banking Example

Consider a bank

- the bank has up to three active tellers
- each teller needs to sign in and sign out
- once signed in, a teller can access one account at a time in order to
  - deposit funds
  - withdraw funds
  - check current account balance
- the accounts are stored in a data base
  - unique account number
  - owner information
  - pin number

Banking: Solution 1

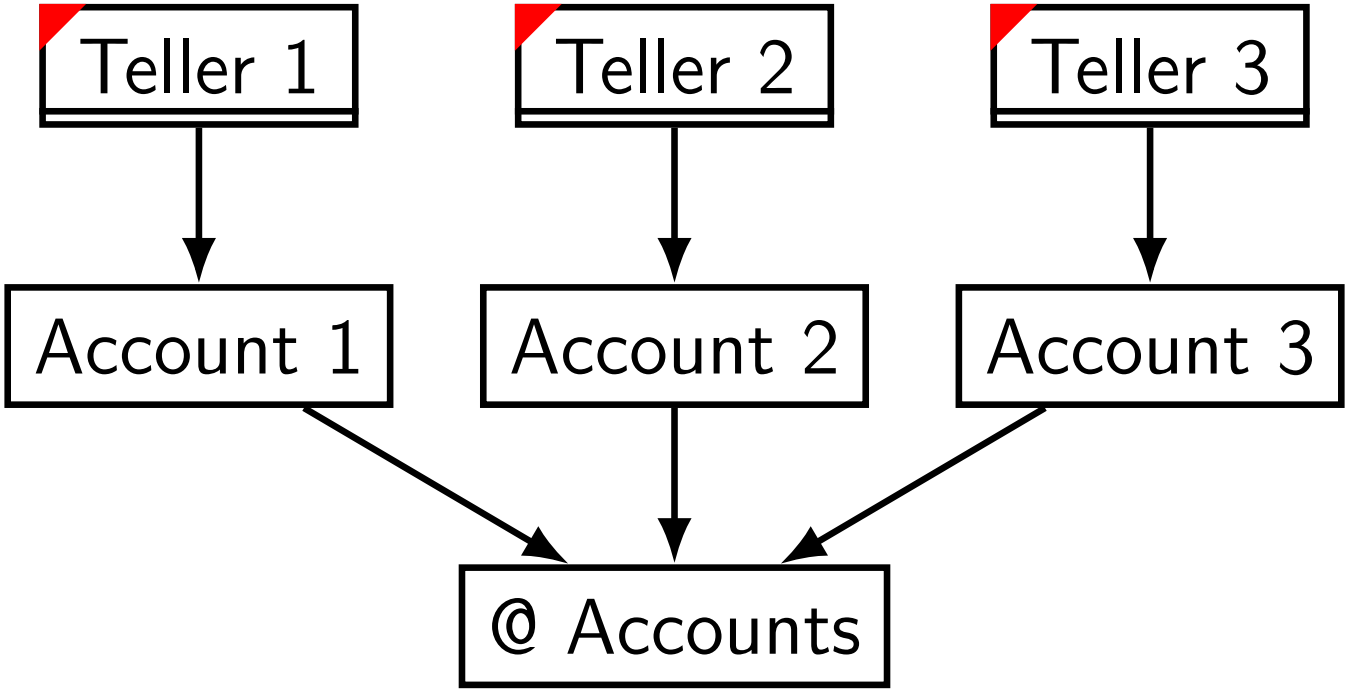
- A separate task is associated with each teller
  - it encapsulates all interactions with the physical tellers (not shown)
- A fixed number of Teller tasks are instantiated at initialization time
- The Teller tasks have direct access to the Accounts object
  - it encapsulates the interactions with the database (not shown)
  - it manages synchronization as needed, on account by account basis



Banking: Solution 2

- The Teller task does not need to know how accounts are managed
- A teller works with a single account at a time
- Between checking the balance and doing a withdrawal the money may not longer be there
- Proposed modifications:
  - lock the account being accessed by the teller and *commit changes at the end*
  - let Teller task work on a single account at a time by creating an *account proxy*

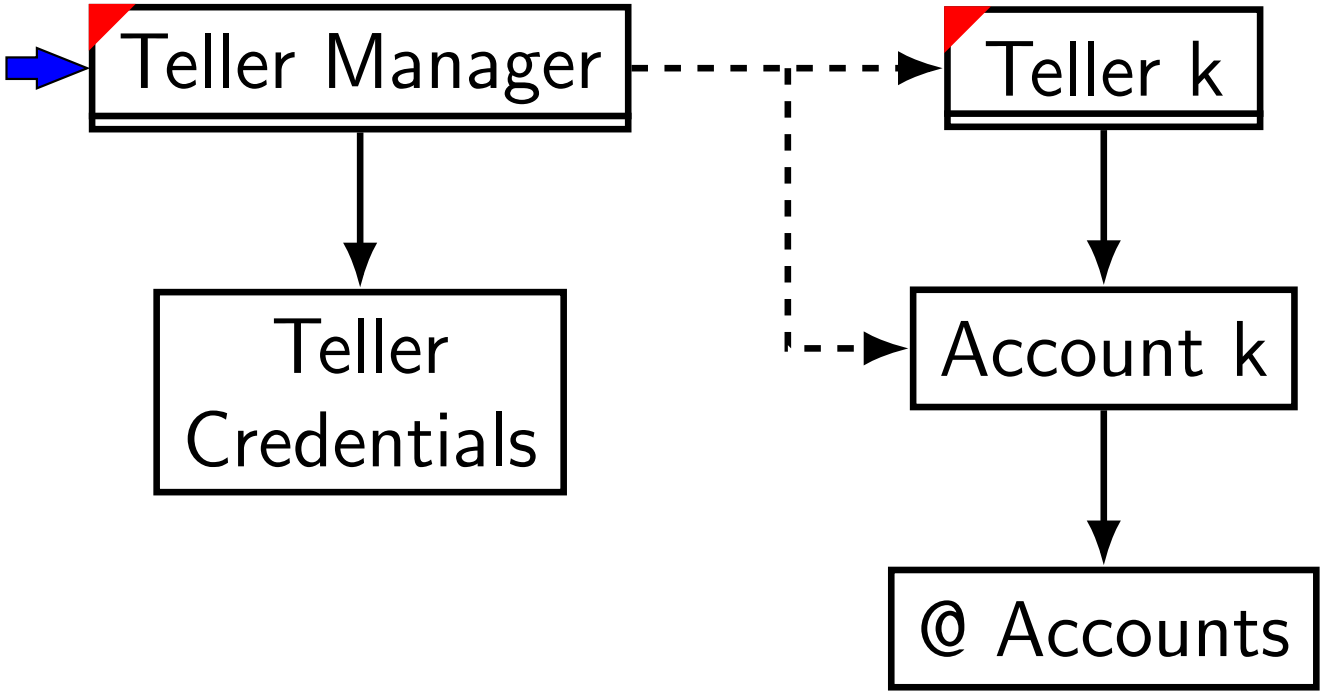
Banking: Solution 2



Banking: Solution 3

- There is no need to run a Teller task when the teller is on vacation
- We should not have to change the initialization code every time the number of tellers changes
- Proposed modifications:
  - add a Teller Manager which sleeps unless a teller wants to sign in
  - create a Teller task and a private Account when a teller signs in
  - no synchronization is needed for Account but the account must be locked in the database

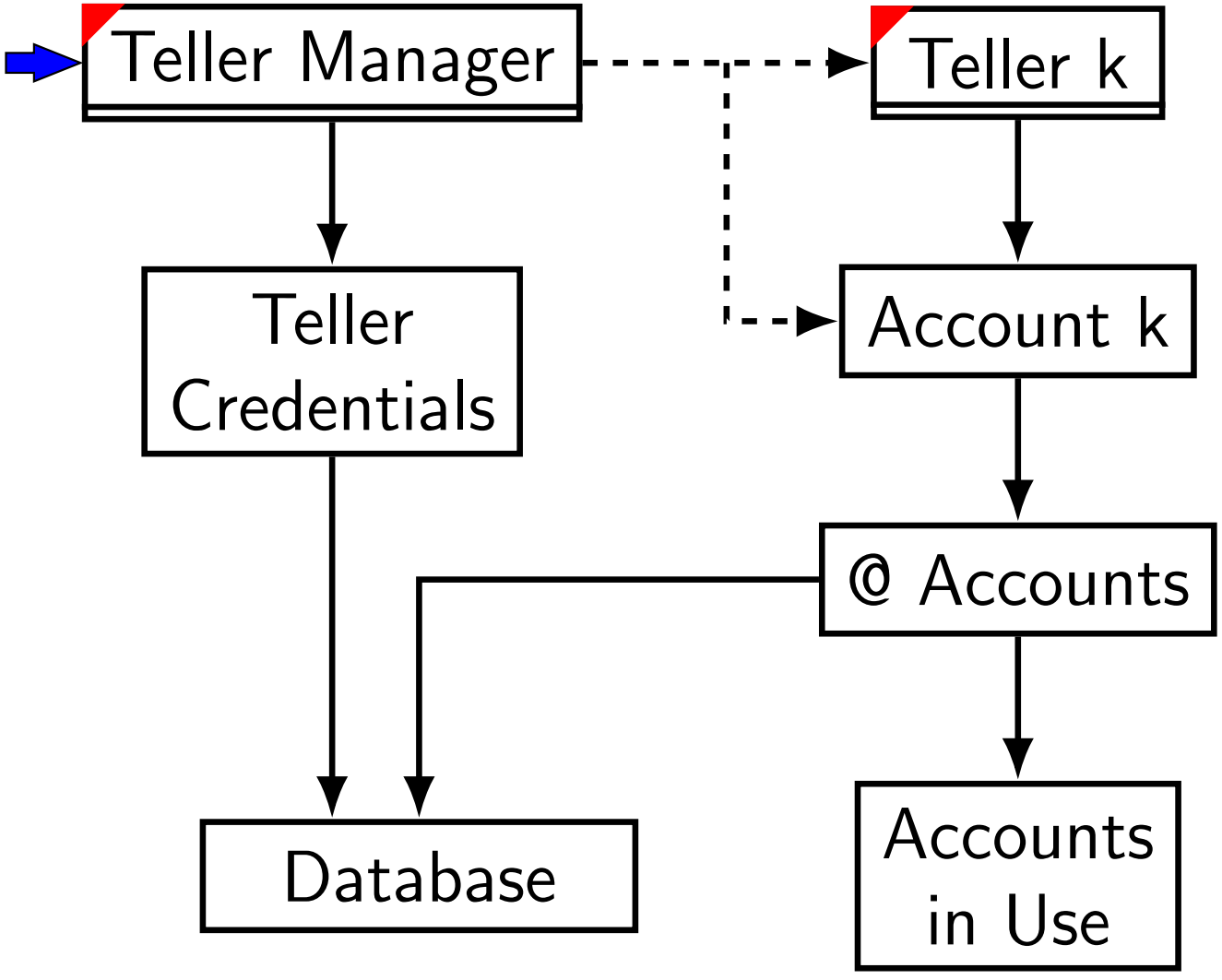
Banking: Solution 3



Banking: Solution 4

- Teller credentials cannot be volatile
- It may be more efficient to save the locks outside the database
  - the database may be backed up in the background
- Proposed modifications:
  - make the database explicit
  - store credentials in the database
  - let Accounts manage the locks

Banking: Solution 4



Asteroid Detection Example

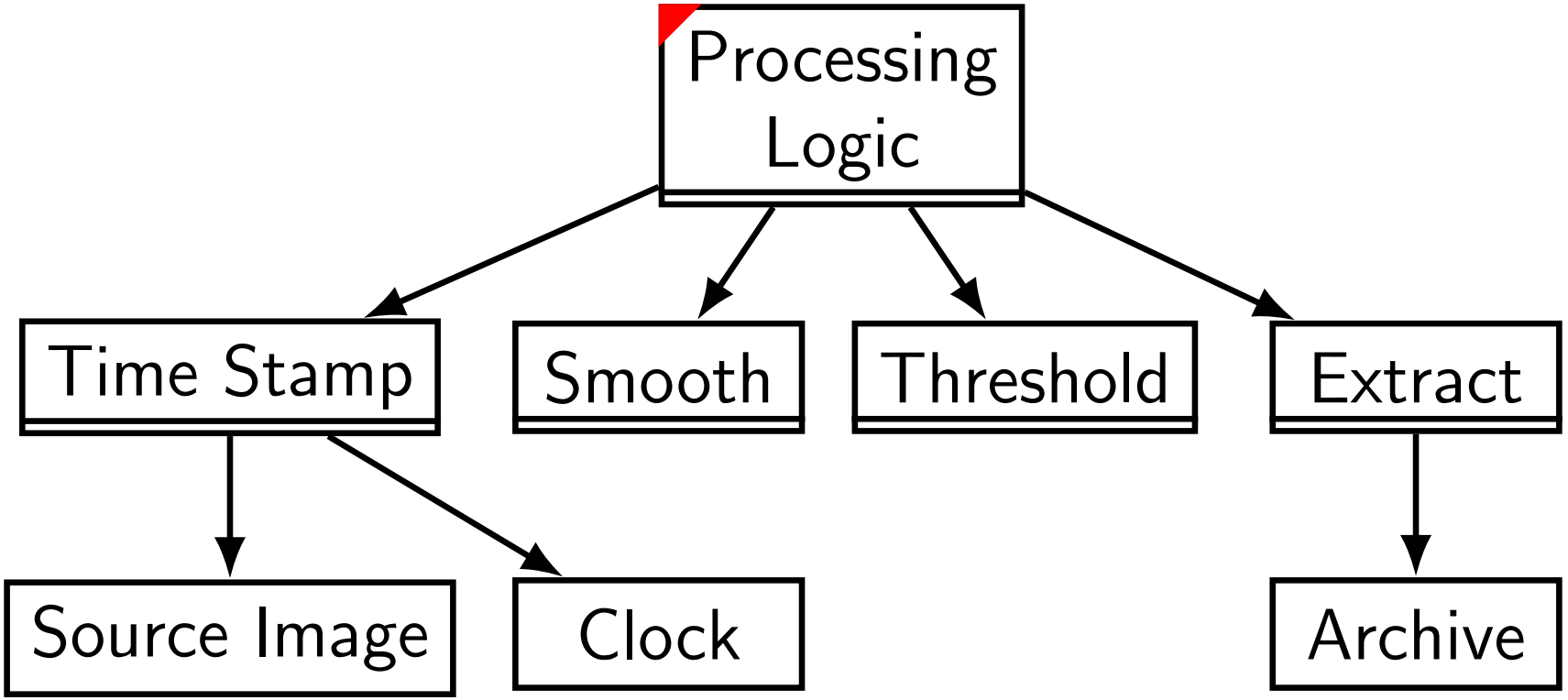
Consider a telescope that scans the sky

- it sends one image at a time for processing
  - fixed size
  - gray scale
  - orientation vector
- the processing steps are as follows
  - time stamp the image
  - apply a Gaussian filter to smooth the image
  - select a threshold value
  - threshold the image
  - compute the center and diameter of large blobs
  - archive the resulting circle and orientation vector

Asteroids: Solution 1

- Associate a procedure with each step along the image processing sequence
- Control the order in which procedures are executed
- Pass the returned values to from one procedure to the next

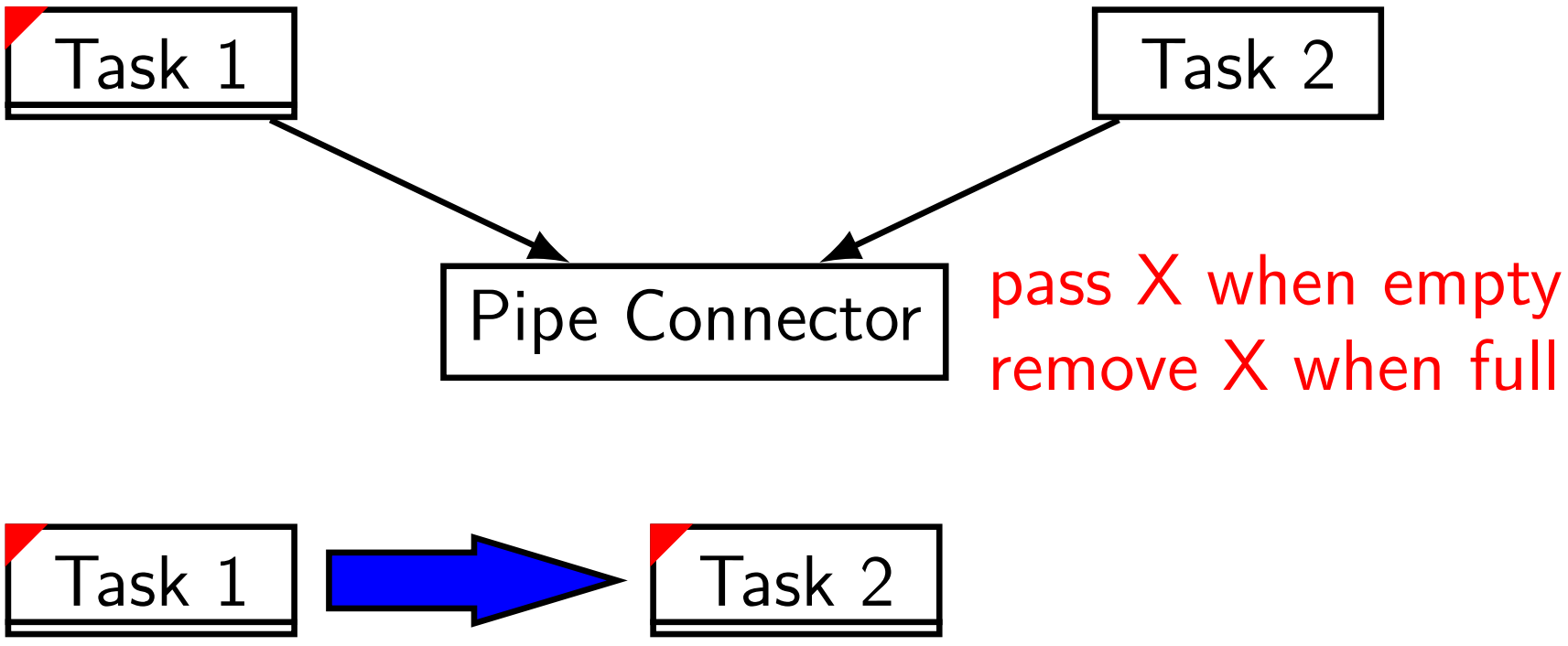
Asteroids: Solution 1



Asteroids: Solution 2

- The design is flexible
  - new steps can be added
  - the ordering of steps can be changed
- However, Processing Logic
  - knows about the data formats being used
  - does a lot of data copying
- Proposed modification:
  - create a pipeline using a new kind of connector

Asteroids: Solution 2



Asteroids: Solution 2

Processing of multiple images can overlap