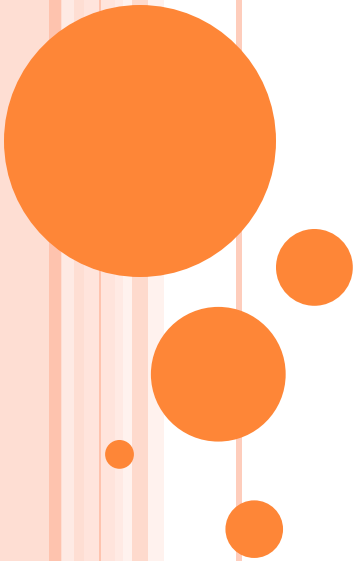


Distributed Systems 1: Lab 2, Distributed Snapshot

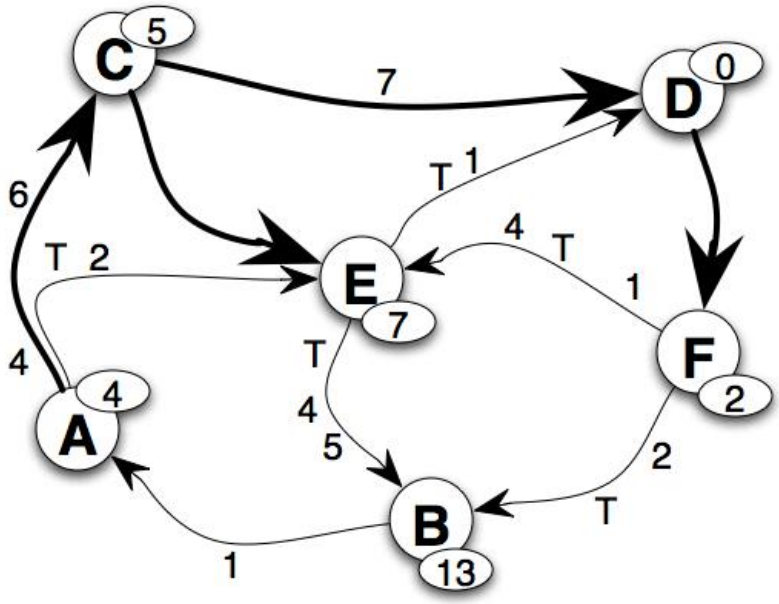
davide.vecchia@unitn.it

timofei.istomin@unitn.it

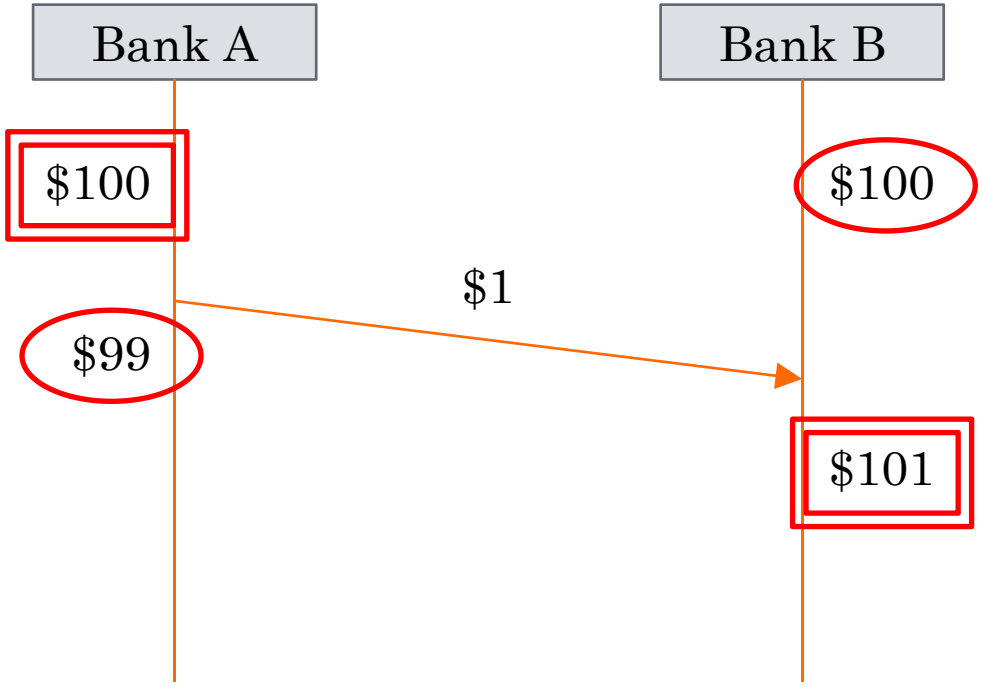
Distributed snapshot

- The "bank" example
- Bank branches send money to each other

- **Question:** what is the total amount of money in the system?
- We'll use the distributed snapshot algorithm of Chandy and Lamport

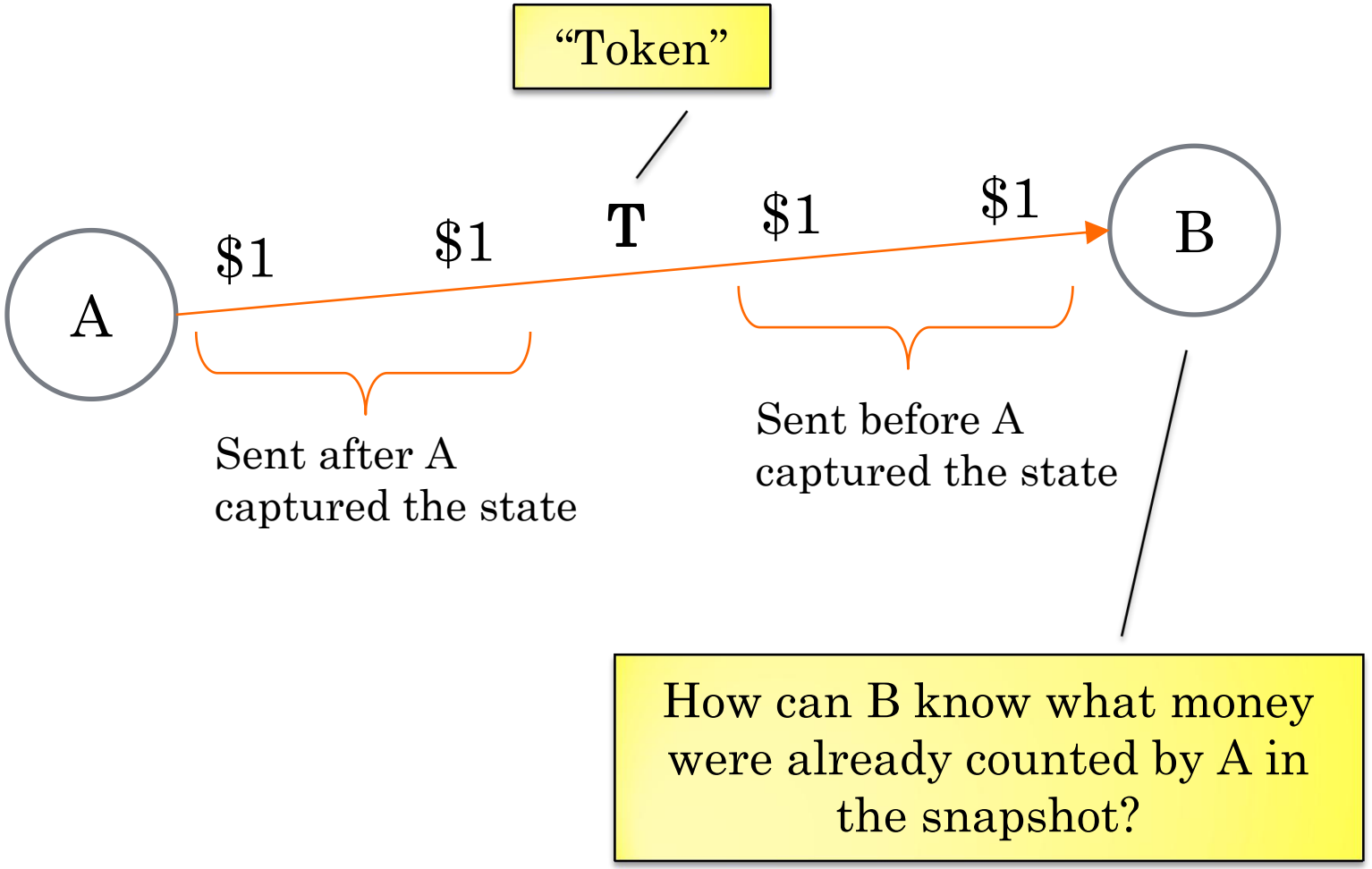


Example: banks and money



Some coordination is needed

Example: banks and money



General algorithm

- **Sending tokens**
  - For each outgoing channel C: process P sends one token along C after P records its state and before P sends further messages along C
- **Receiving a token along channel C**
  - If P has not recorded its state then P records its state and initializes the state of C as the empty sequence
  - Otherwise P records the state of C as the sequence of messages received along C after P's state was recorded and before P received the marker along C
- The algorithm **terminates** when P receives tokens from all incoming channels

Simplified algorithm for our banks

- We can simplify because we know the semantics of the messages and we are only interested in the total amount of money
- The local state of a bank is its balance
- Instead of storing all "messages in transit" we will just have a sum of all "money in transit"

Simplified algorithm for our banks

- **Receiving a token at bank A**
  - If A has not recorded its balance yet, then
    - A records it and sets the "money-in-transit" variable to 0
    - A sends tokens to all peer banks
- **Sending tokens (A→B)**
  - Bank A sends one token to peer bank B after A has recorded its state but before A sends further money transfers to B
- **Receiving money (A→B)**
  - If B recorded its balance but did not receive a token from A yet, it adds the incoming amount to the "money-in-transit" variable
- The algorithm **terminates** when a bank receives tokens from all its peers

Exercise template

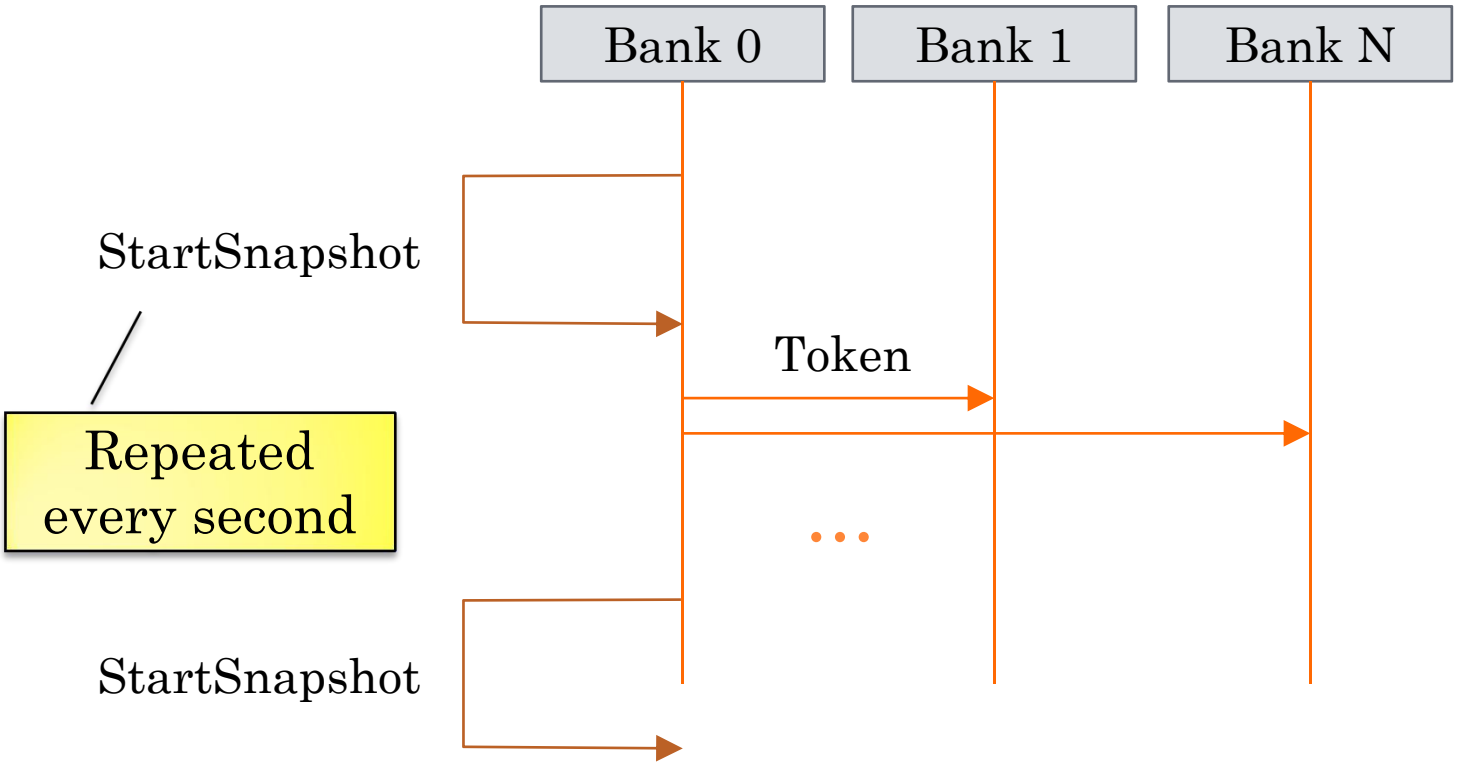
- `BankSystem.java`, `Bank.java` – incomplete snapshot implementation
  - N=10 bank branches (actors) with initial balance of $1000 each that send money continuously to random peers
  - **Bank 0** periodically broadcasts tokens to the rest of the group (once every second)
  - When a bank receives a token, it prints its current balance

Checking the snapshots

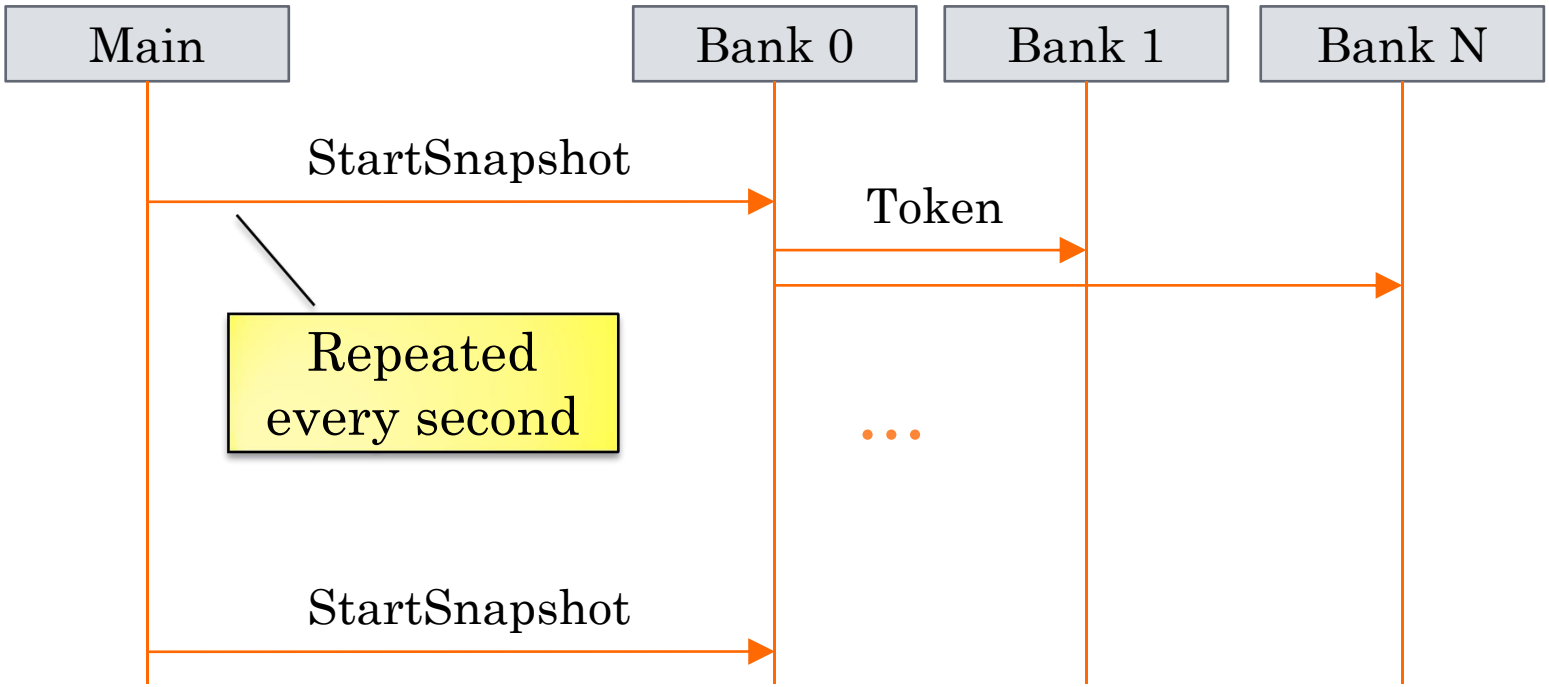
- Start the program and let it run the for some time
  - `gradle run`
- Copy or save the output to `test.log`
  - `gradle run | tee test.log`
- Compile and run the provided `Check.java`
  - `javac Check.java`
  - `java Check test.log`
- You will see that the captured amount of money is not $10000

- Your task is to fix that!

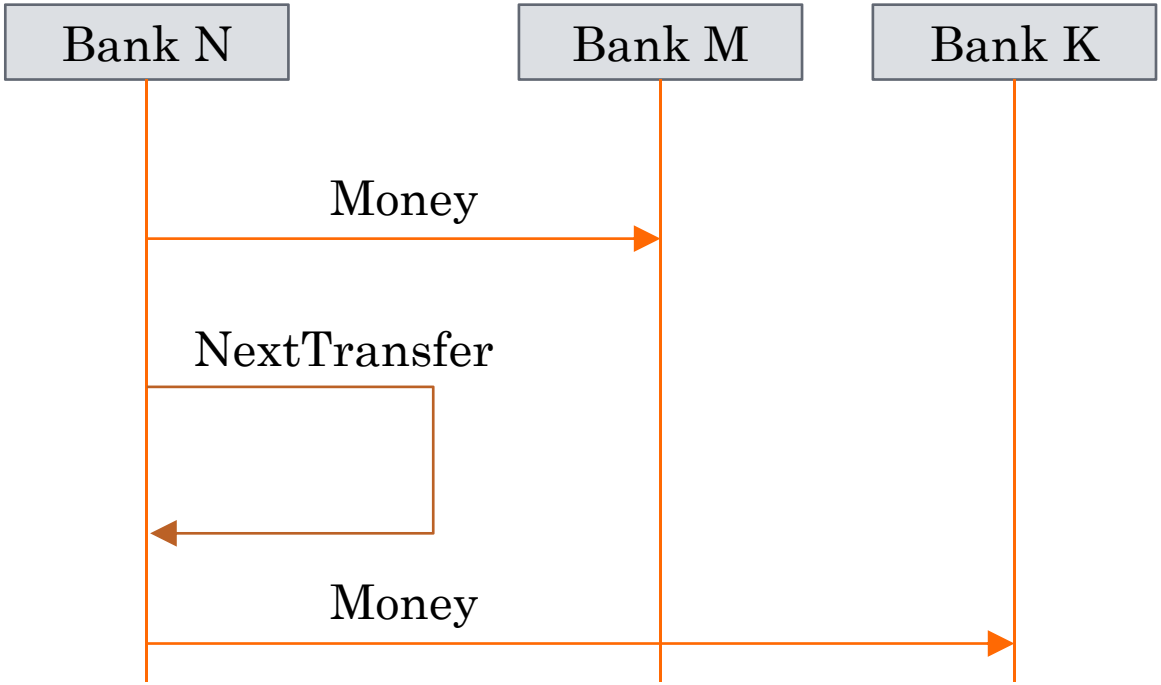
A LOOK AT THE PROGRAM TEMPLATE: SCHEDULING



Another option for scheduling snapshot



Template: periodic money transfer



Hints

- Assume that the previous snapshot completes before a new one starts
- Add a `HashSet<ActorRef>` of banks that you have received a token from
- You can add more info to the output but don't change the beginning of the line:
  - `Bank 0 snapId: 1 state: 984`
  - E.g. you can add the money in transit to the output:
  - `Bank 0 snapId: 1 state: 984 in transit: 10`

Variables you will need

```
// snapshot in progress
boolean stateCaptured;
// captured balance
int capturedBalance;
// in-transit money
int moneyInTransit;
// set of peers we got a token from
Set<ActorRef> tokensReceived =
                        new HashSet<>();
```

Useful methods

`java.util.Set`

- `add()`
- `contains()`
- `containsAll()`
- `size()`
- `clear()`