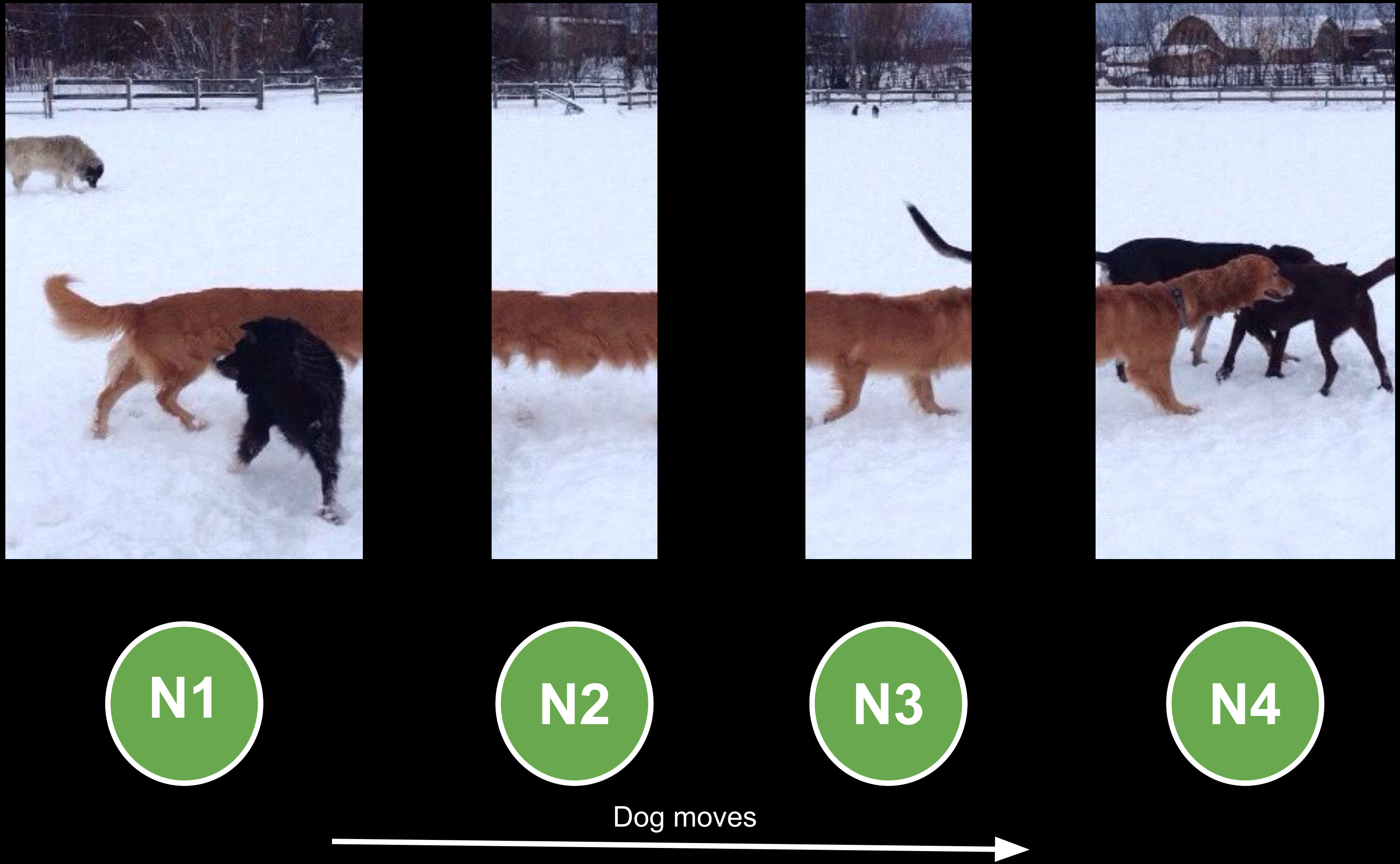
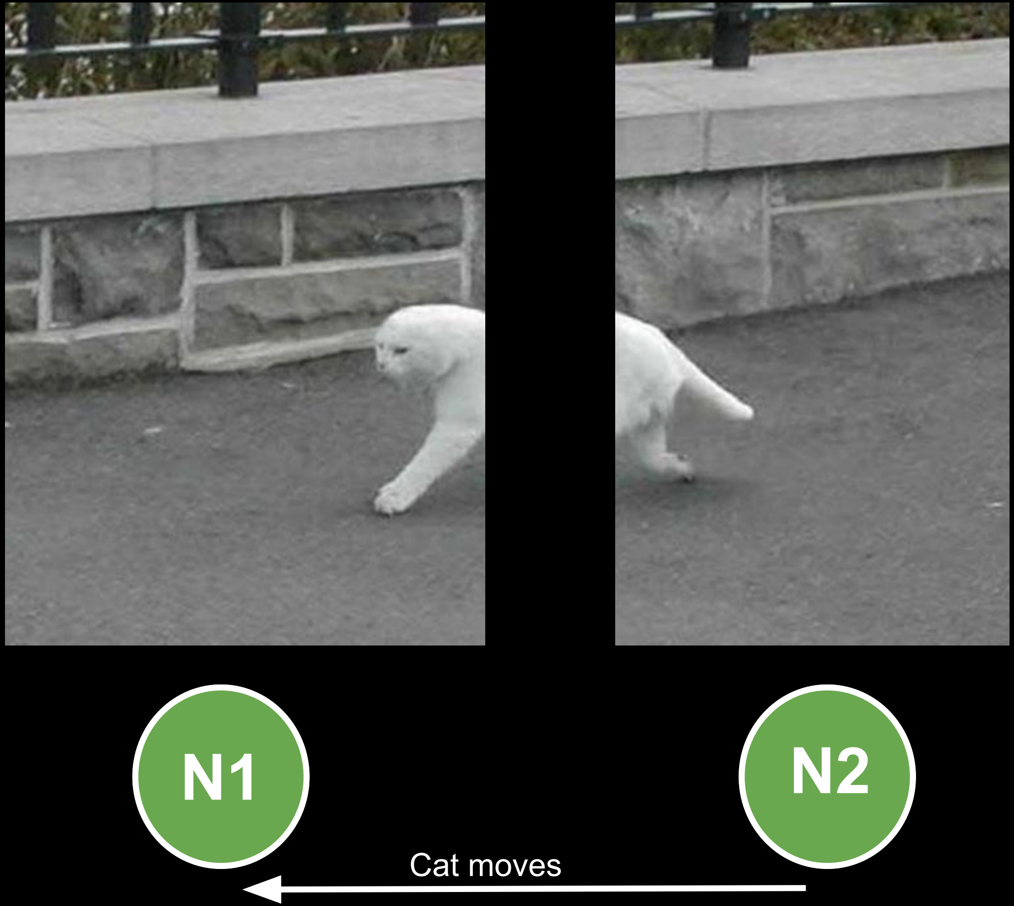
Distributed Snapshots

9/29/22

Need synchronization


N1
N2
N3
N4
Dog moves


N1
N2
Cat moves

Distributed snapshots are easy to screw up

Must ensure state is not duplicated across the cluster

Must ensure state is not lost across the cluster

Messages in flight must also be recorded

Intuition: guarantee zero loss + zero duplication

If you *have* snapshotted your local state:

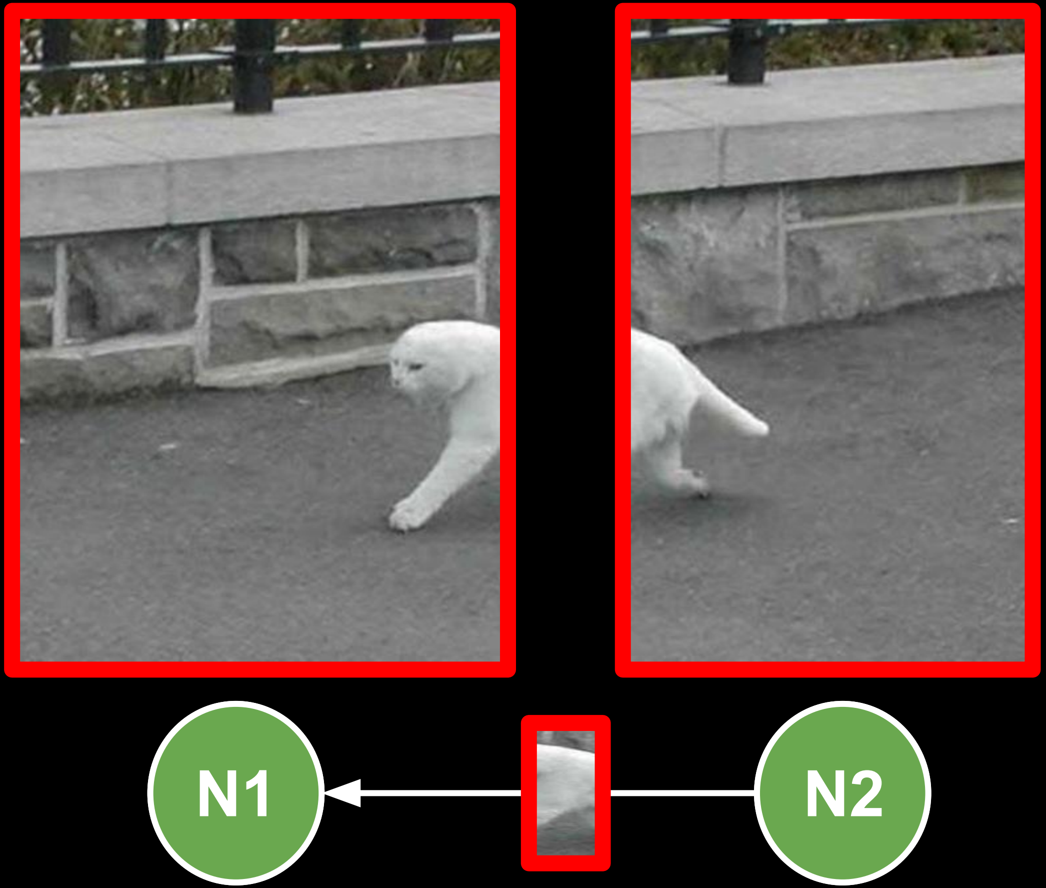
- *Do* record future messages you receive

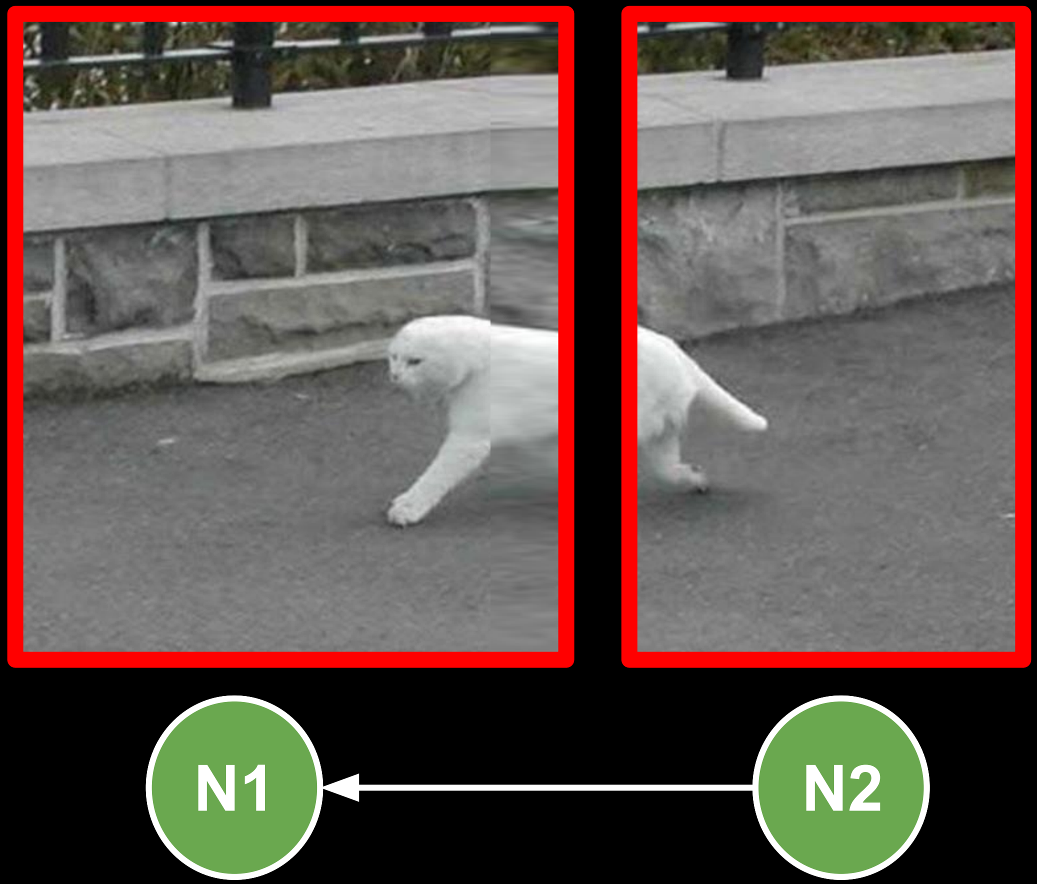
If you *haven't* snapshotted your local state yet:

- *Do NOT* record future messages you receive

Which one guarantees zero loss?

Which one guarantees zero duplication?


N1
N2


N1
N2

Chandy-Lamport snapshot algorithm

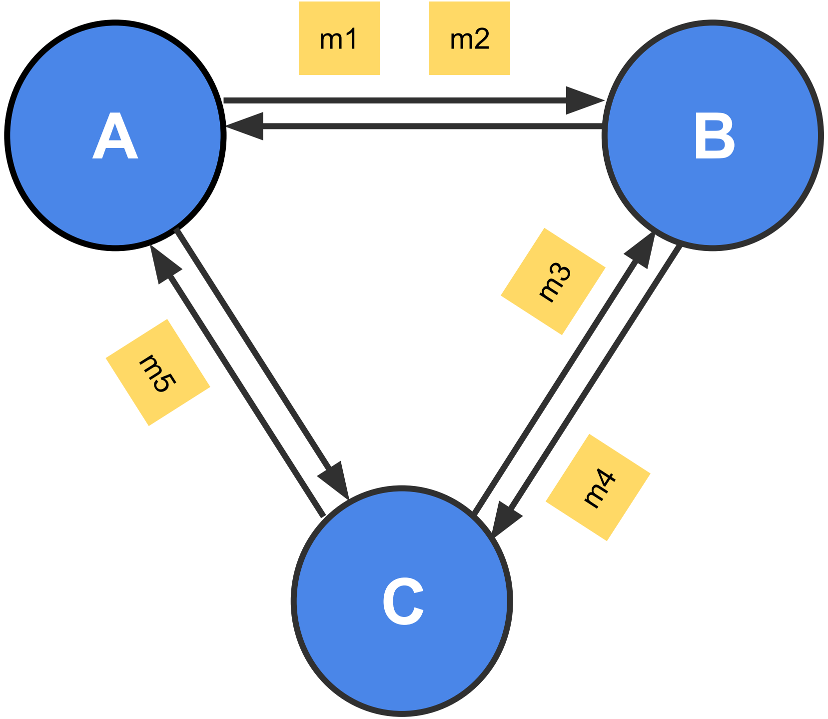
Key idea: Servers send marker messages to each other

Marker messages...

1) Mark the beginning of the snapshot process on the server
2) Act as a barrier (stopper) for recording messages

What is a Global Snapshot?

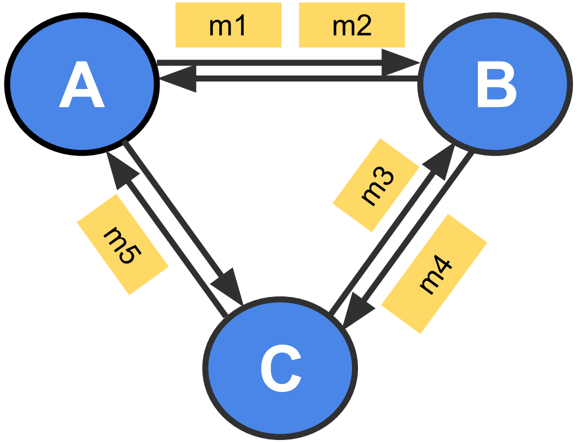
- A global snapshot captures the global state of a distributed system:
    - Local state of each process within the distributed system
    - Local state of each communication channel
- These local states are instantaneous
    - e.g messages in transit one node to another



** Using “process” and “node” interchangeably.

Refresher: system model

- N processes in the system with no process failures
  - Each process tracks some state
- Two FIFO unidirectional channels between every process pair P and Q
  - Channel also has state: the set of messages in the channel
  - All messages sent on channels arrive intact, unduplicated, in order



Chandy-Lamport snapshot algorithm

Starting the snapshot procedure on a server:

- Record local state
- Send marker messages on all outbound interfaces

When you receive a marker message on an interface:

- If you haven't started the snapshot procedure yet:
  - record your local state
  - send marker messages on all outbound interfaces
- Otherwise, stop recording messages you receive on *this* interface, start recording messages you receive on all *other* interfaces

Terminate when all servers have received marker messages on all interfaces

Chandy-Lamport snapshot algorithm

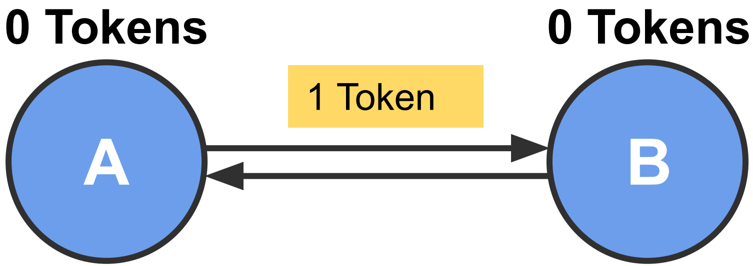
Starting the snapshot procedure on a server:

- Record local state
- Send marker messages on all outbound interfaces

When you receive a marker message on an interface:

- If you haven't started the snapshot procedure yet:
  - record your local state
  - send marker messages on all outbound interfaces
- Otherwise, stop recording messages you receive on *this* interface, start recording messages you receive on all *other* interfaces

Terminate when all servers have received marker messages on all interfaces



Chandy-Lamport snapshot algorithm

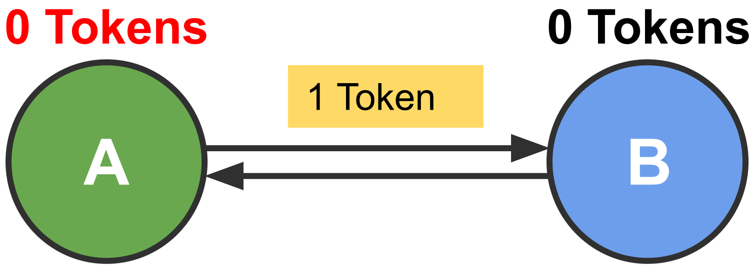
Starting the snapshot procedure on a server:

- **Record local state**
- Send marker messages on all outbound interfaces

When you receive a marker message on an interface:

- If you haven't started the snapshot procedure yet:
    - record your local state,
    - send marker messages on all outbound interfaces
- Otherwise, stop recording messages you receive on *this* interface, start recording messages you receive on all *other* interfaces

Terminate when all servers have received marker messages on all interfaces



Chandy-Lamport snapshot algorithm

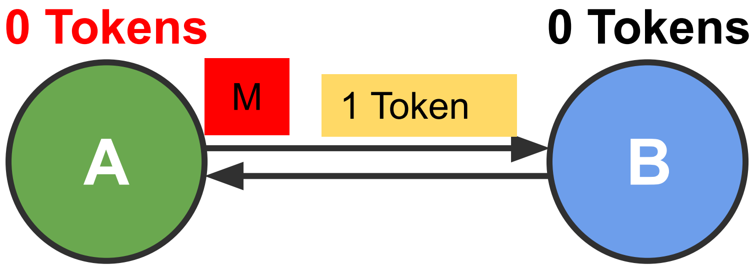
Starting the snapshot procedure on a server:

- **Record local state**
- **Send marker messages on all outbound interfaces**

When you receive a marker message on an interface:

- If you haven't started the snapshot procedure yet:
    - record your local state
    - send marker messages on all outbound interfaces
- Otherwise, stop recording messages you receive on *this* interface, start recording messages you receive on all *other* interfaces

Terminate when all servers have received marker messages on all interfaces



Chandy-Lamport snapshot algorithm

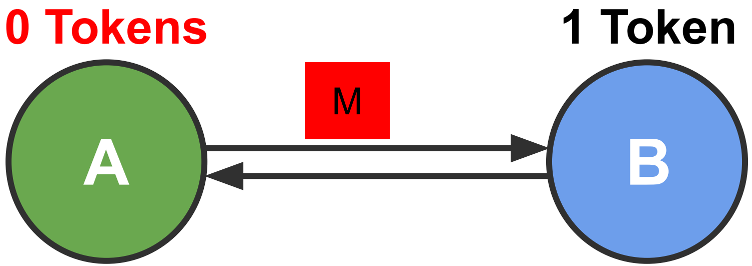
Starting the snapshot procedure on a server:

- **Record local state**
- **Send marker messages on all outbound interfaces**

When you receive a marker message on an interface:

- If you haven't started the snapshot procedure yet:
  - record your local state
  - send marker messages on all outbound interfaces
- Otherwise, stop recording messages you receive on *this* interface, start recording messages you receive on all *other* interfaces

Terminate when all servers have received marker messages on all interfaces



Chandy-Lamport snapshot algorithm

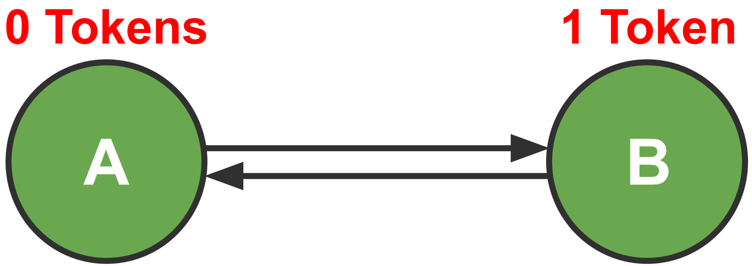
Starting the snapshot procedure on a server:

- **Record local state**
- **Send marker messages on all outbound interfaces**

When you receive a marker message on an interface:

- **If you haven't started the snapshot procedure yet:**
  - **record your local state**
  - send marker messages on all outbound interfaces
- Otherwise, stop recording messages you receive on *this* interface, start recording messages you receive on all *other* interfaces

Terminate when all servers have received marker messages on all interfaces



Chandy-Lamport snapshot algorithm

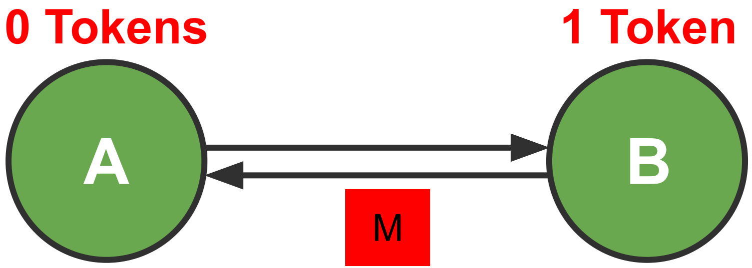
Starting the snapshot procedure on a server:

- **Record local state**
- **Send marker messages on all outbound interfaces**

When you receive a marker message on an interface:

- **If you haven't started the snapshot procedure yet:**
  - **record your local state**
  - **send marker messages on all outbound interfaces**
- Otherwise, stop recording messages you receive on *this* interface, start recording messages you receive on all *other* interfaces

Terminate when all servers have received marker messages on all interfaces



Chandy-Lamport snapshot algorithm

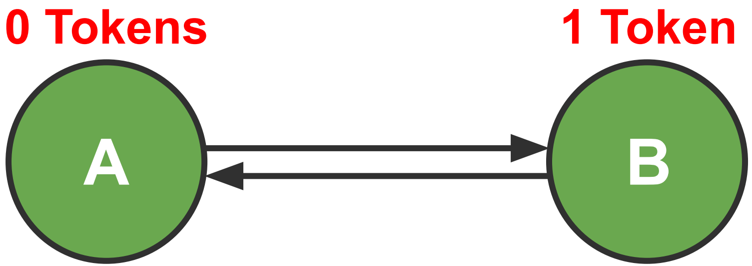
Starting the snapshot procedure on a server:

- **Record local state**
- **Send marker messages on all outbound interfaces**

When you receive a marker message on an interface:

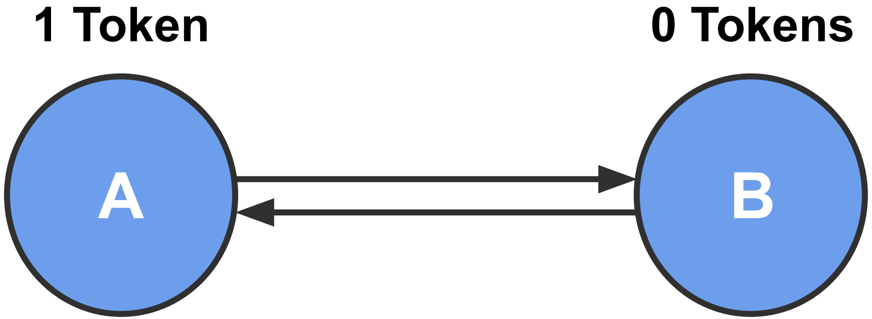
- **If you haven't started the snapshot procedure yet:**
  - **record your local state**
  - **send marker messages on all outbound interfaces**
- Otherwise, stop recording messages you receive on *this* interface, start recording messages you receive on all *other* interfaces

Terminate when all servers have received marker messages on all interfaces

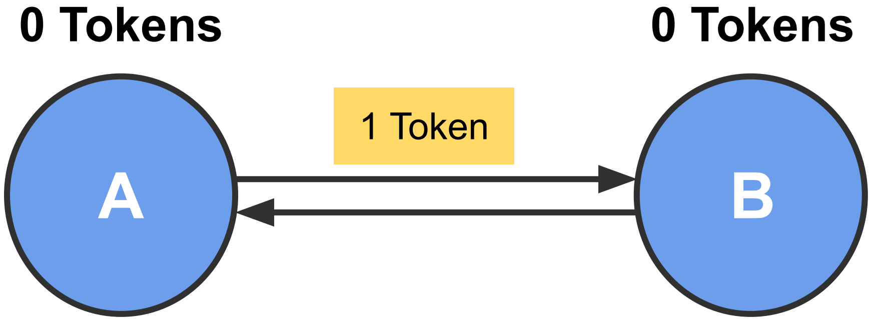


Global Snapshot Finished!

Token passing example 1



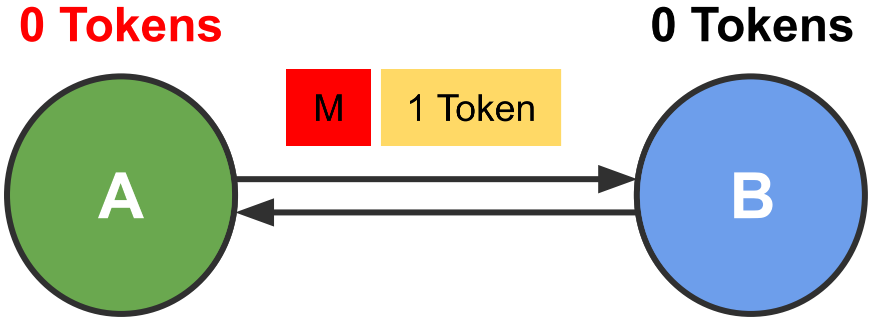
Token passing example 1



Event order:

1. *A* sends 1 token

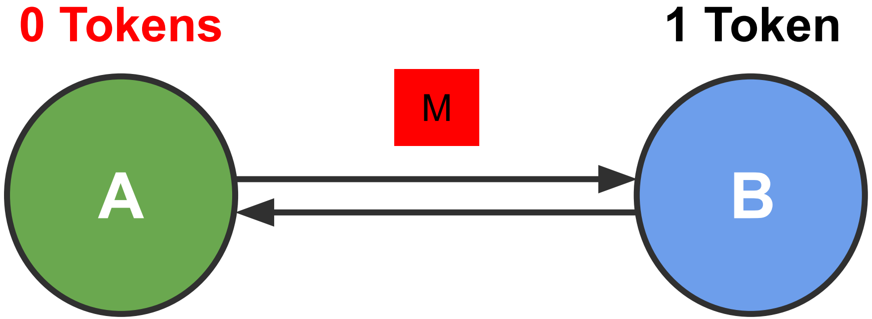
Token passing example 1



Event order:

1. *A* sends 1 token
2. *A* starts snapshot, sends marker

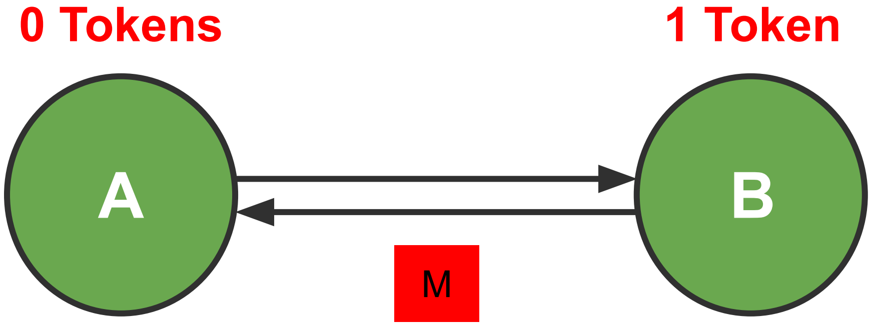
Token passing example 1



Event order:

1. *A* sends 1 token
2. *A* starts snapshot, sends marker
3. *B* receives 1 token

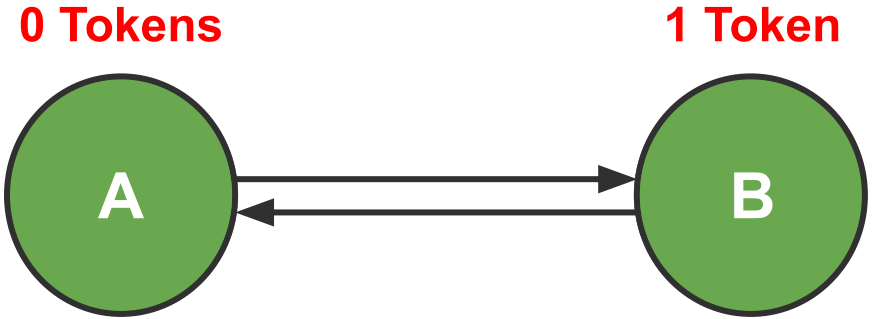
Token passing example 1



Event order:

1. *A* sends 1 token
2. *A* starts snapshot, sends marker
3. *B* receives 1 token
4. *B* receives marker, starts snapshot

Token passing example 1

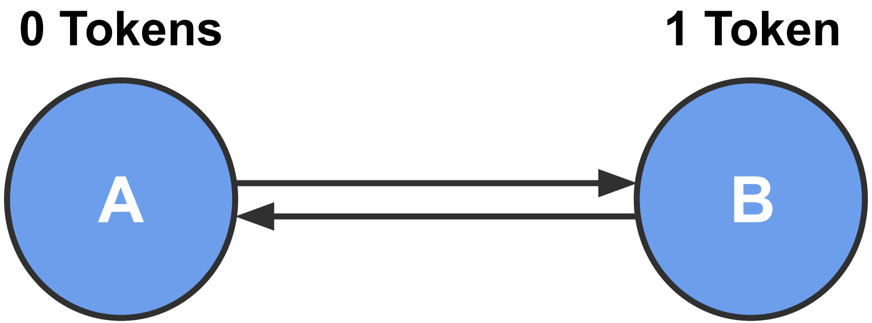


We did not record the token in-flight because B received it before B started the snapshot process

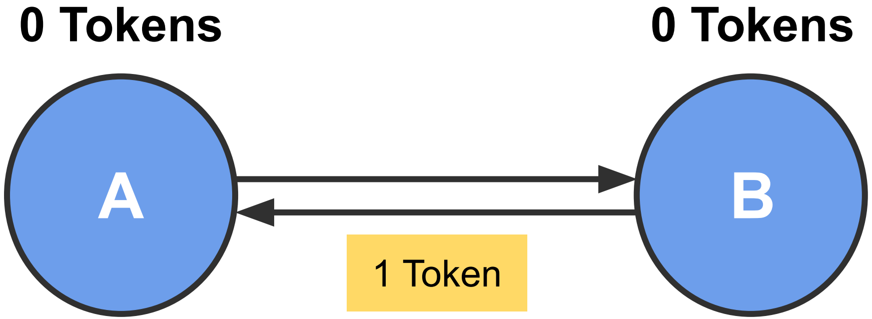
Event order:

1. *A* sends 1 token
2. *A* starts snapshot, sends marker
3. *B* receives 1 token
4. *B* receives marker, starts snapshot
5. *A* receives marker, ends snapshot

Token passing example 2



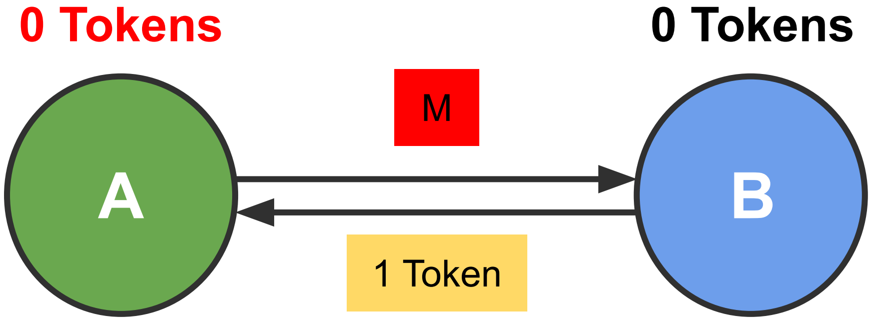
Token passing example 2



Event order:

1. *B* sends 1 token

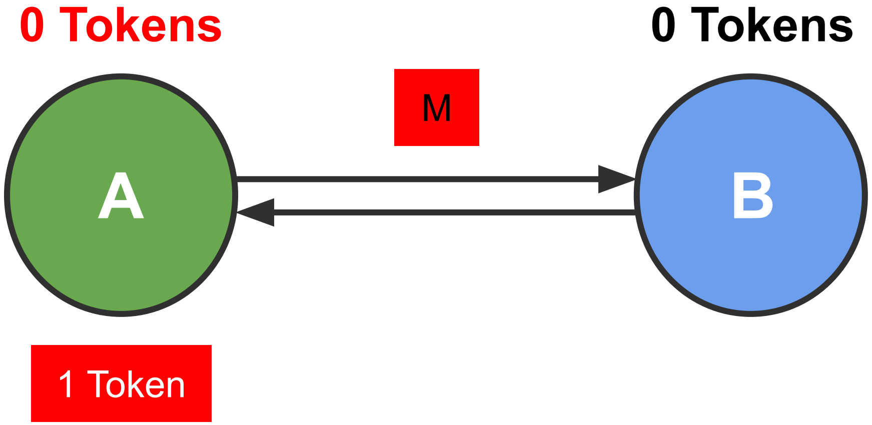
Token passing example 2



Event order:

1. *B* sends 1 token
2. *A* starts snapshot, sends marker

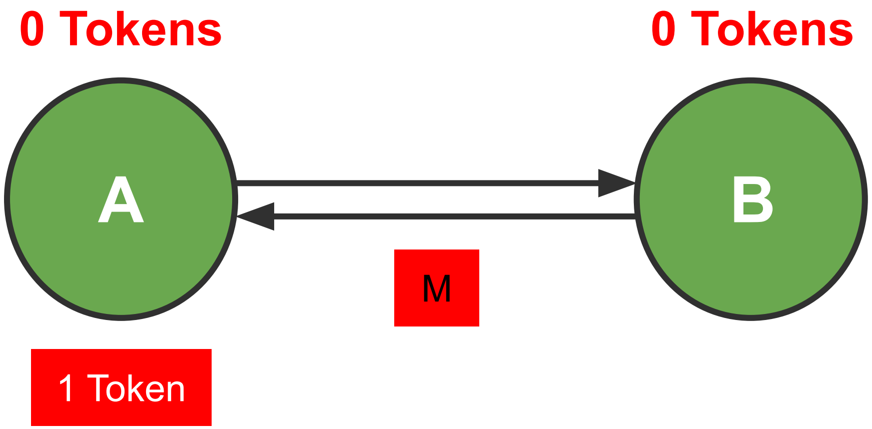
Token passing example 2



Event order:

1. *B* sends 1 token
2. *A* starts snapshot, sends marker
3. *A* receives 1 token, records message

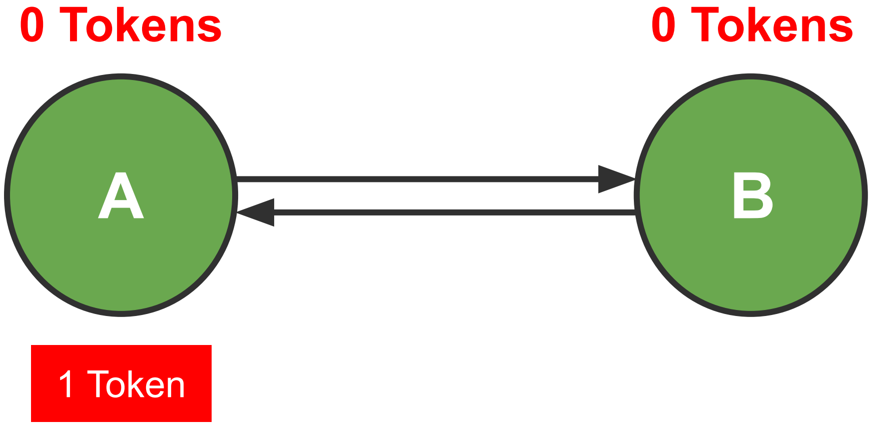
Token passing example 2



Event order:

1. *B* sends 1 token
2. *A* starts snapshot, sends marker
3. *A* receives 1 token, records message
4. *B* receives marker, starts snapshot

Token passing example 2

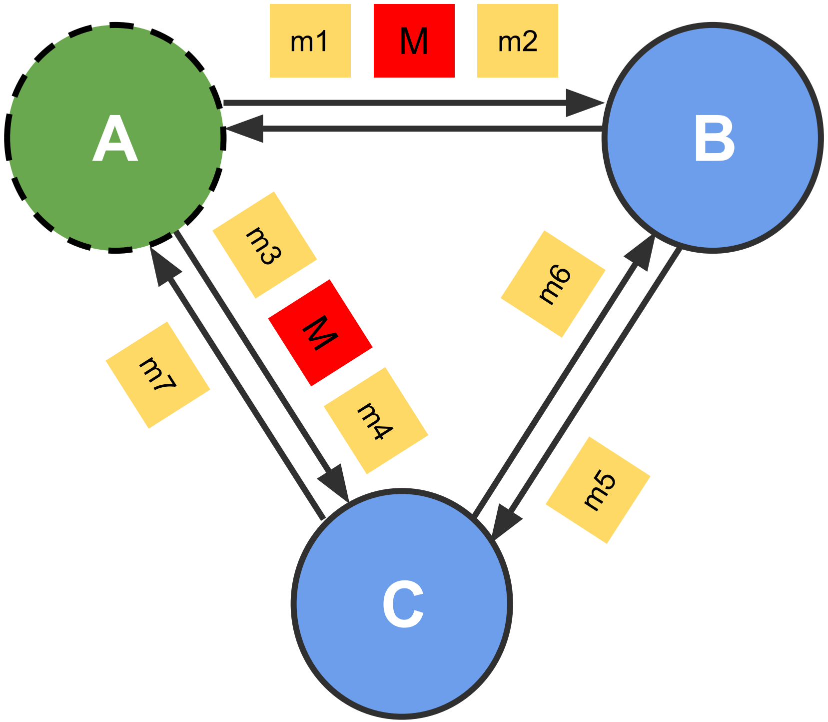


We recorded the token in-flight because A received it **after** *it has already started the snapshot process*

Event order:

1. *B* sends 1 token
2. *A* starts snapshot, sends marker
3. *A* receives 1 token, records message
4. *B* receives marker, starts snapshot
5. *A* receives marker, ends snapshot

Token passing example 3



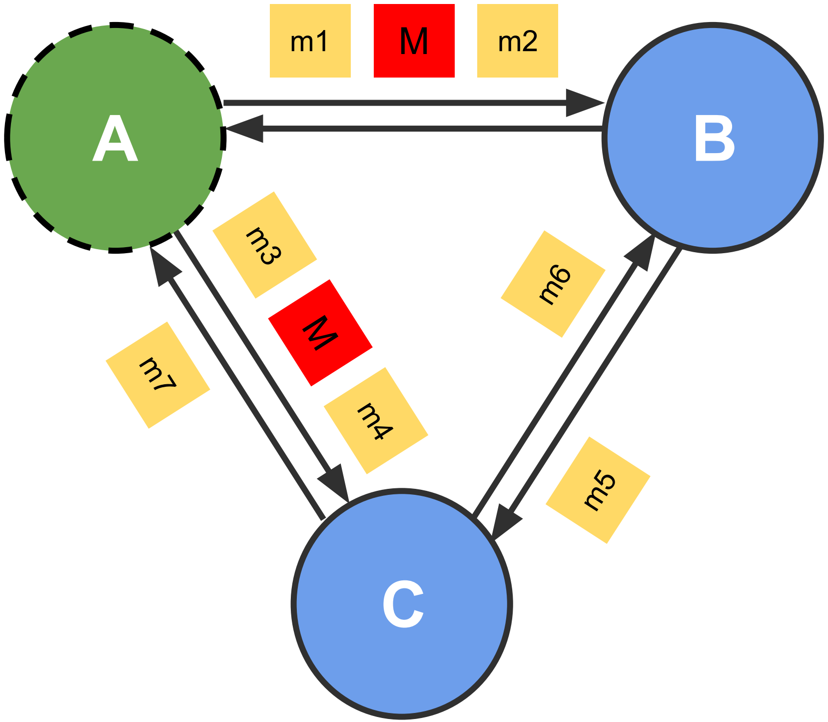
Which messages are definitely recorded in-flight?

Which messages are definitely *not* recorded?

Which messages *might* be recorded?

* recorded as in-flight messages, i.e., as part of *channel state* rather than *process state*

Token passing example 3



Which messages are definitely recorded*?

m7

Which messages are definitely *not* recorded?

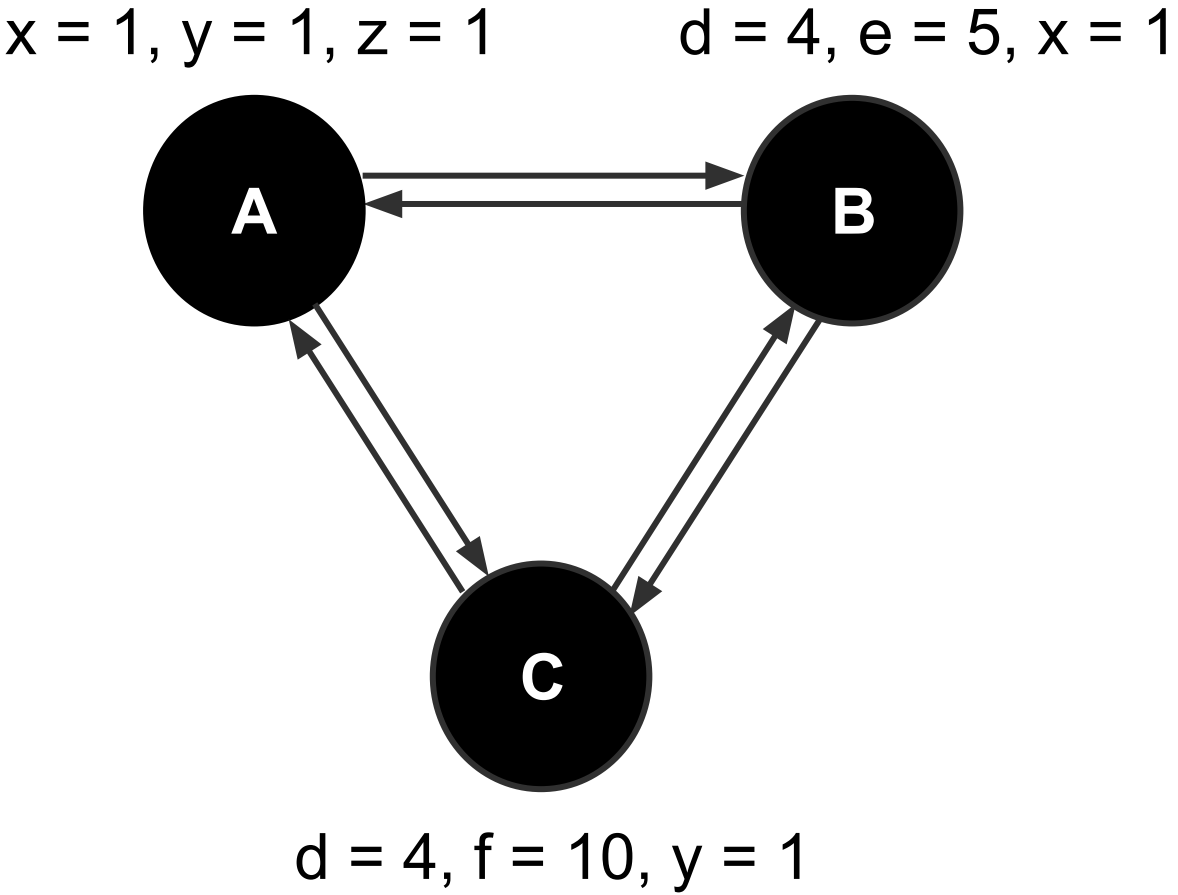
m1, m3

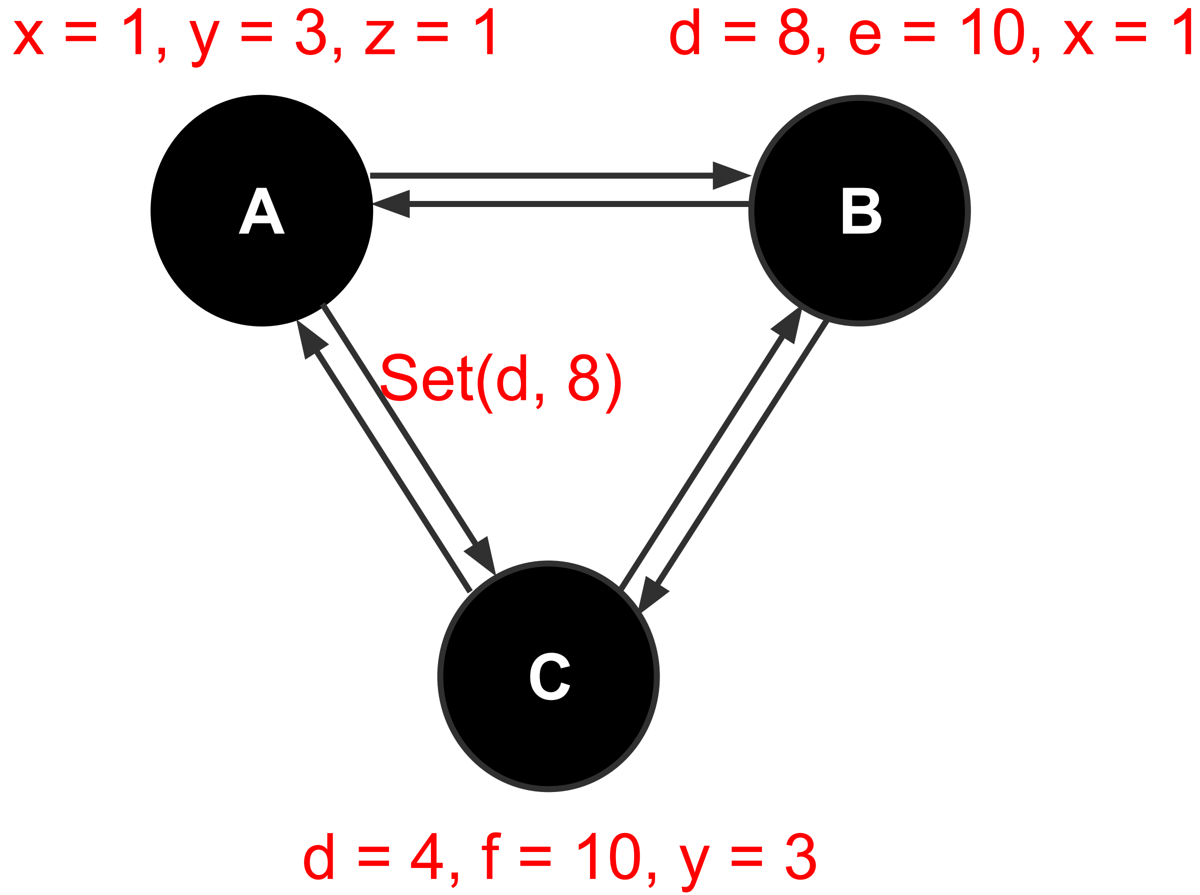
Which messages *might* be recorded?

m2, m4, m5, m6

*recorded as in-flight messages

Distributed database exercise


x = 1, y = 1, z = 1
d = 4, e = 5, x = 1
A
B
C
d = 4, f = 10, y = 1



x = 1, y = 3, z = 1
d = 8, e = 10, x = 1
A
B
Set(d, 8)
C
d = 4, f = 10, y = 3