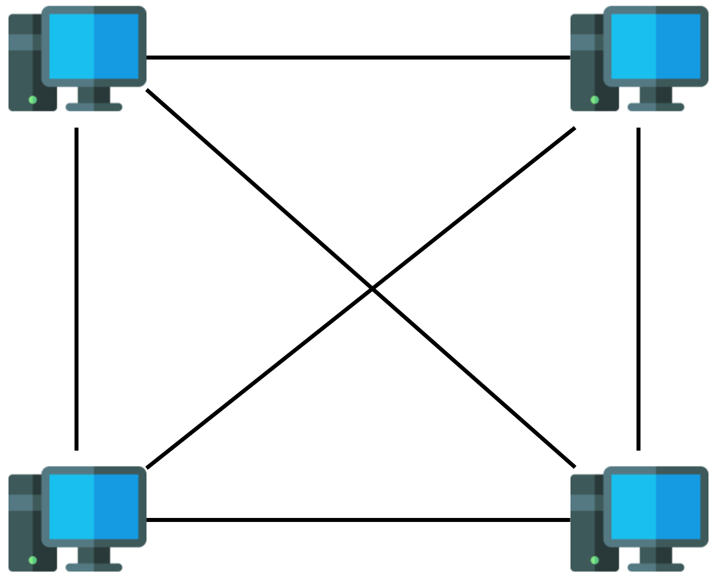
POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Highly available, causally ordered group chat

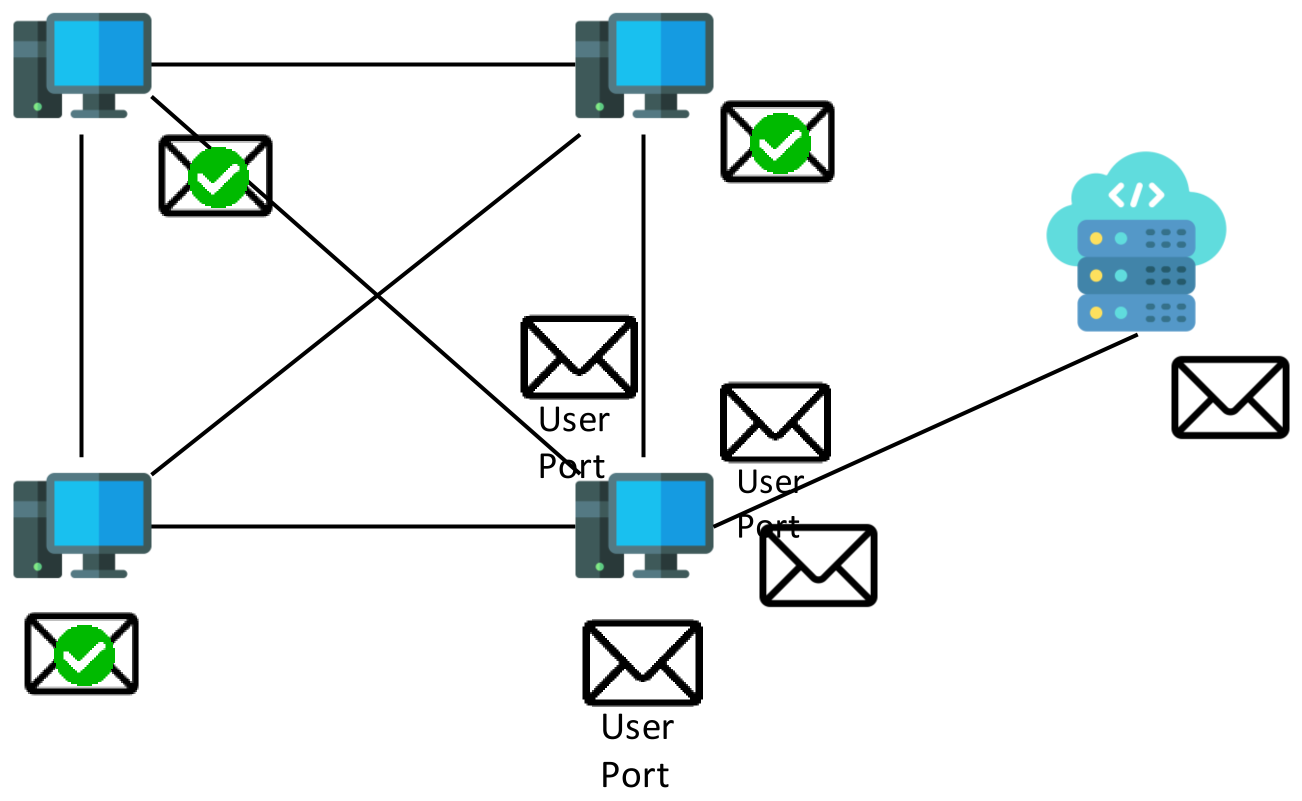
Francesco Spangaro – Giacomo Orsenigo – Federico Saccani

Network



- Peer to peer connection with a discovery server
- Using Java TCP sockets
- Acks to detect network failures

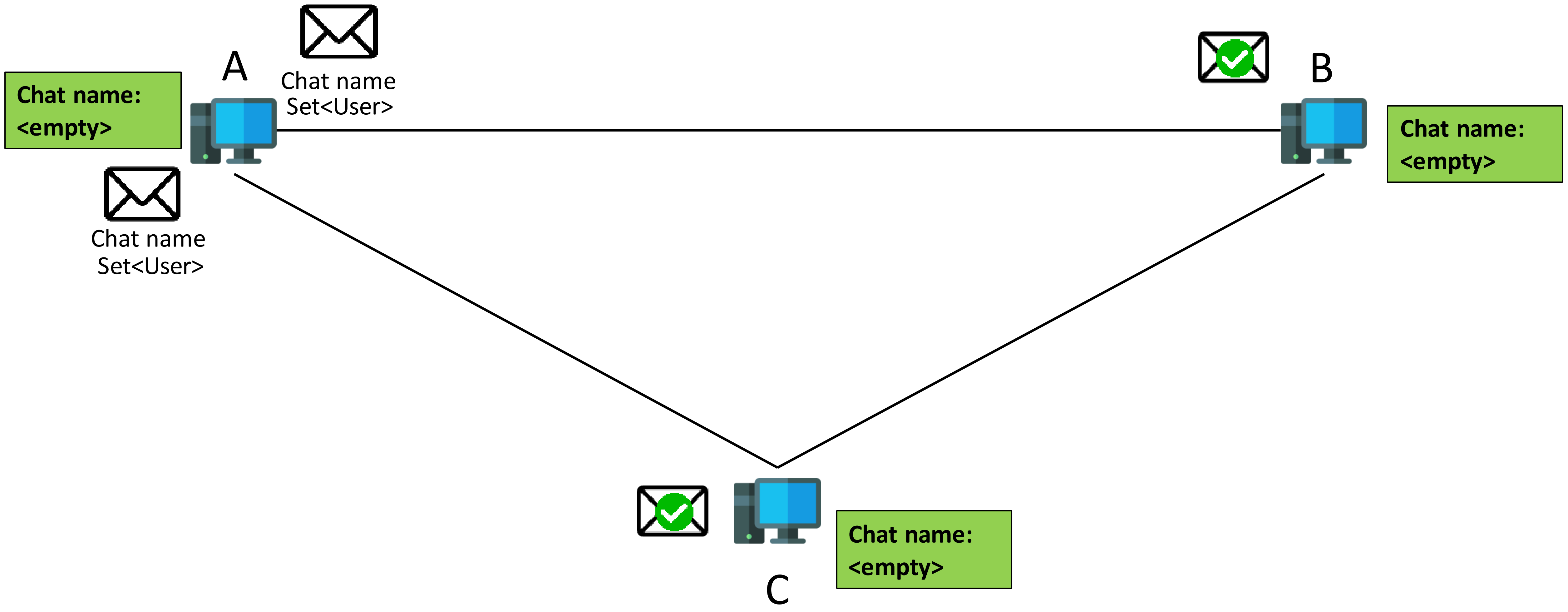
Connection setup



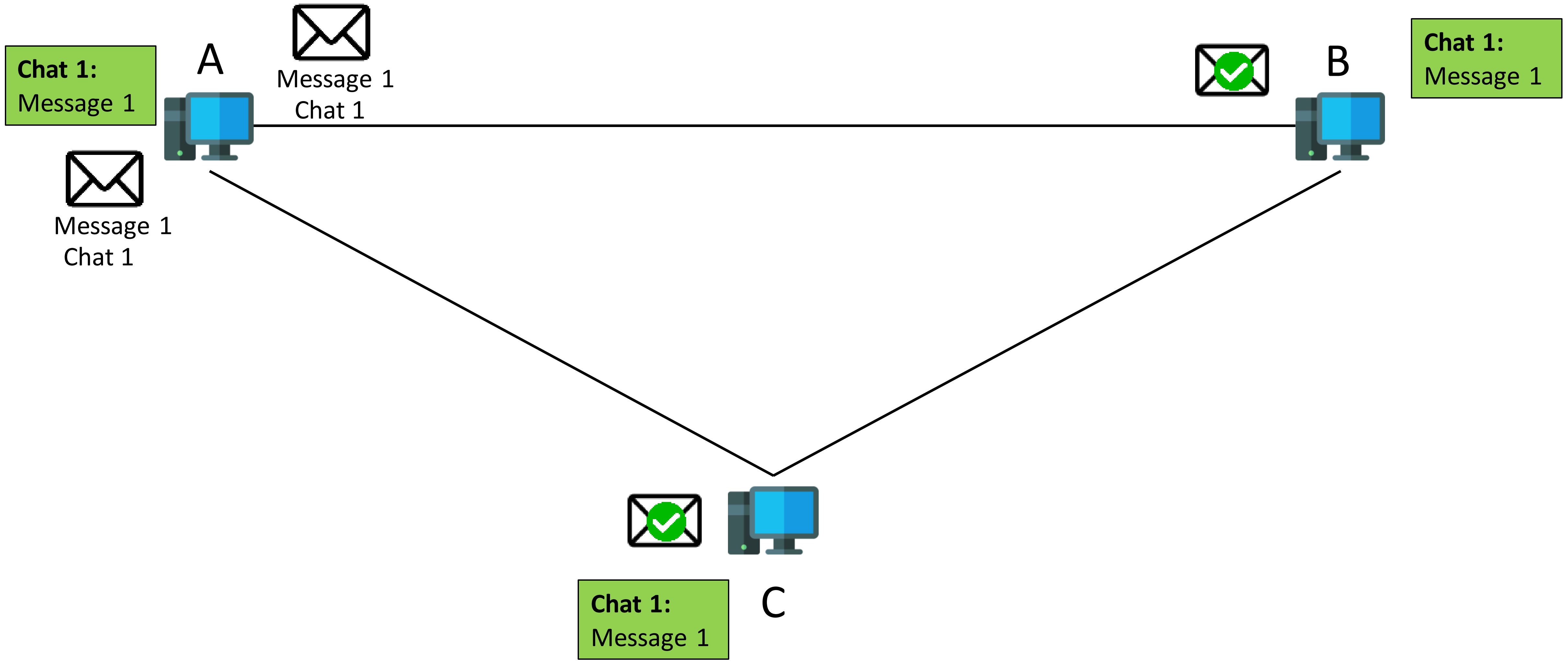
1. The new peer asks the discovery server for the list of addresses.

2. The new peer establishes connections with all other peers.

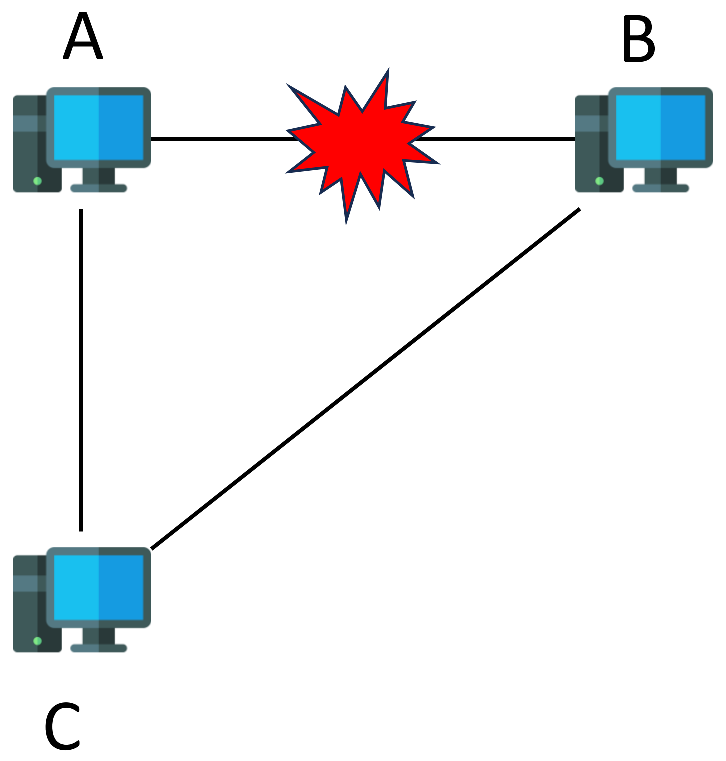
Chat creation



Sending a message (without network faults)

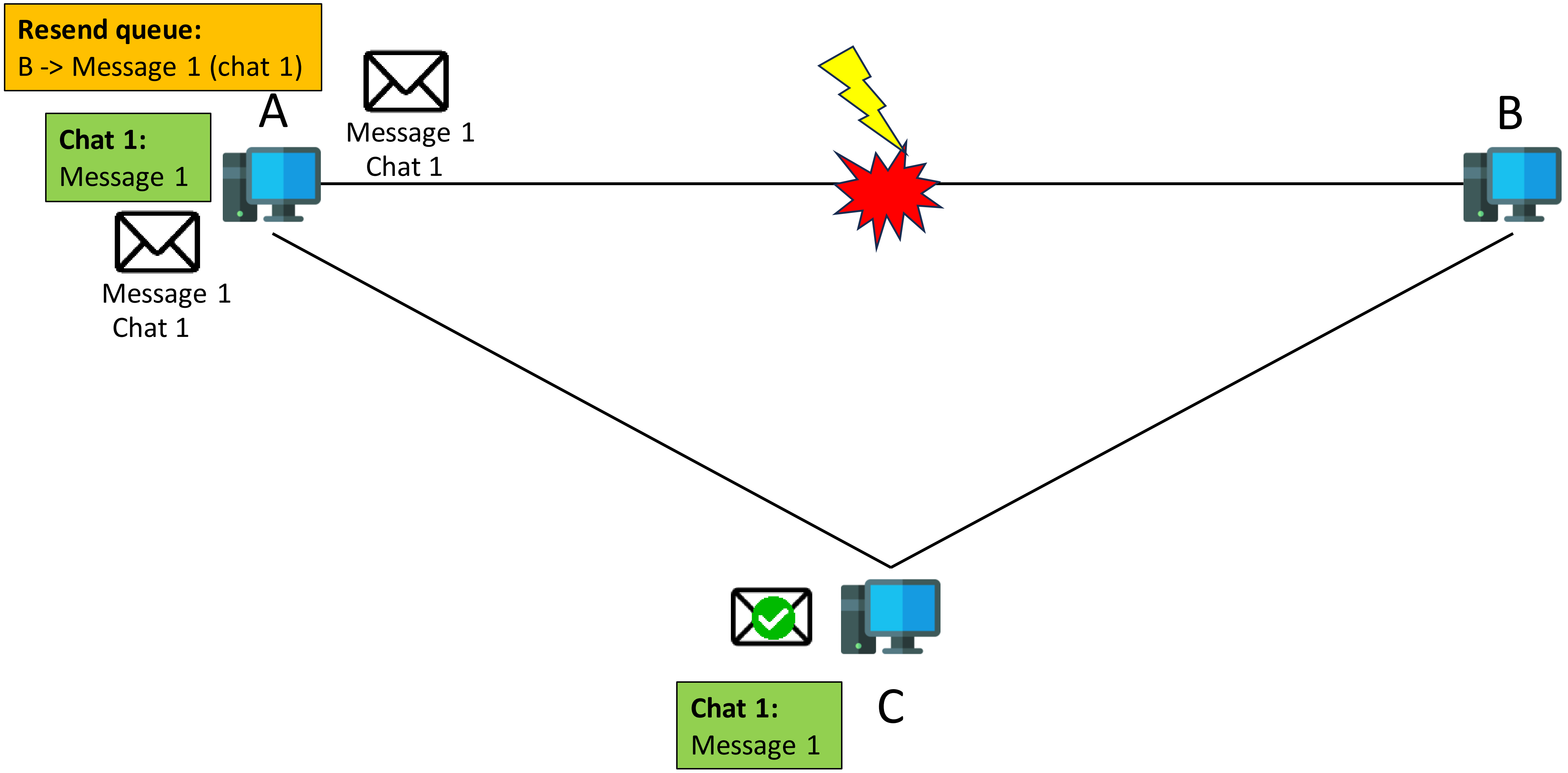


Network faults

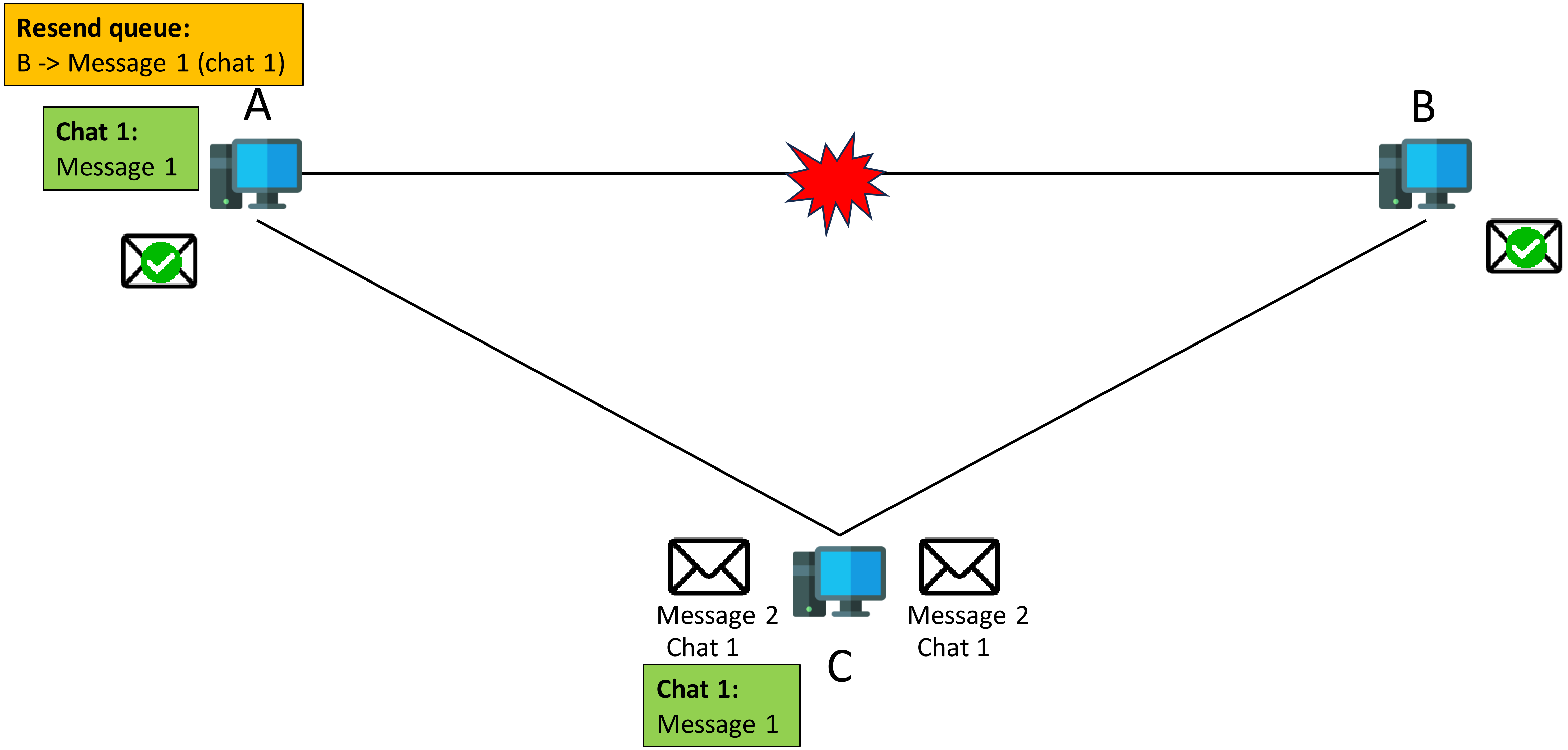


1. All packets are acknowledged to detect network faults
1. All packets sent during network fault are enqueued
3. Automatically retry to reconnect
4. When reconnected send enqueued packets

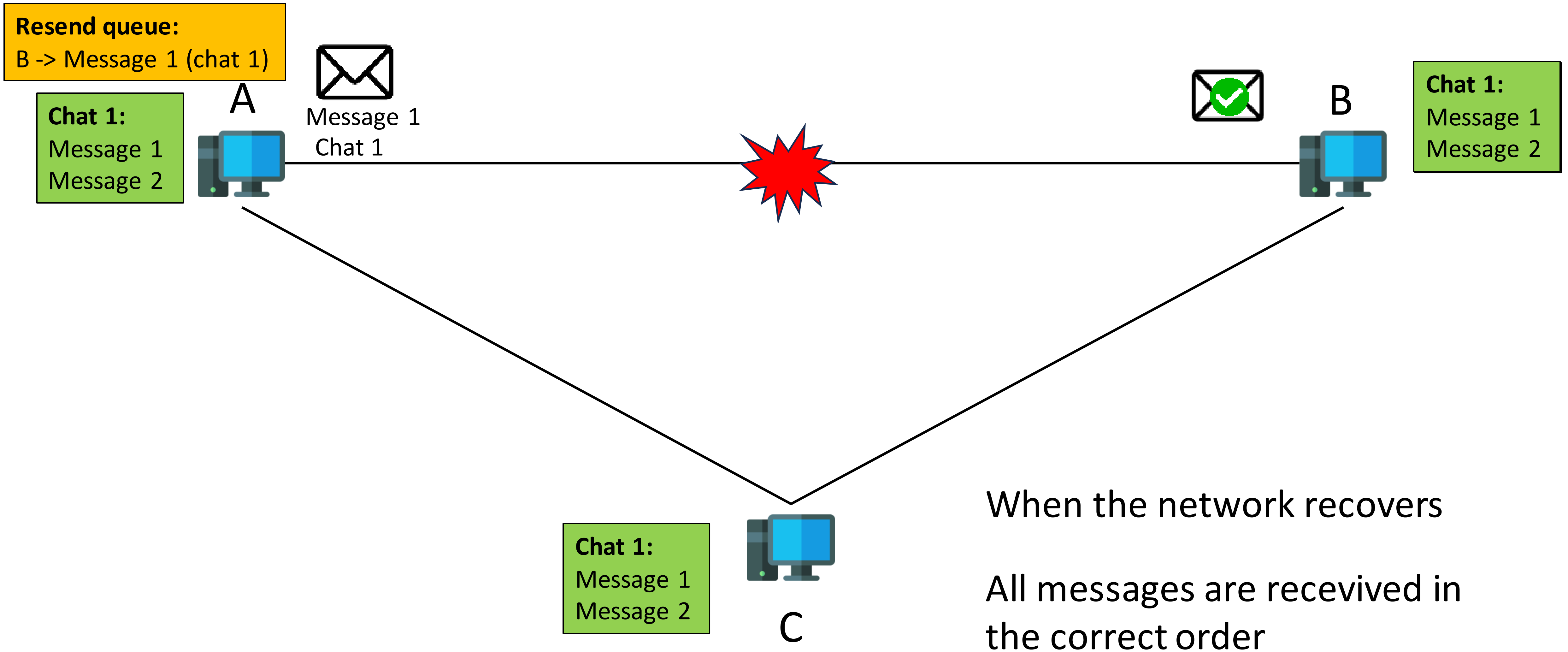
Sending a message (example of network fault)



Sending a message (example of network fault)



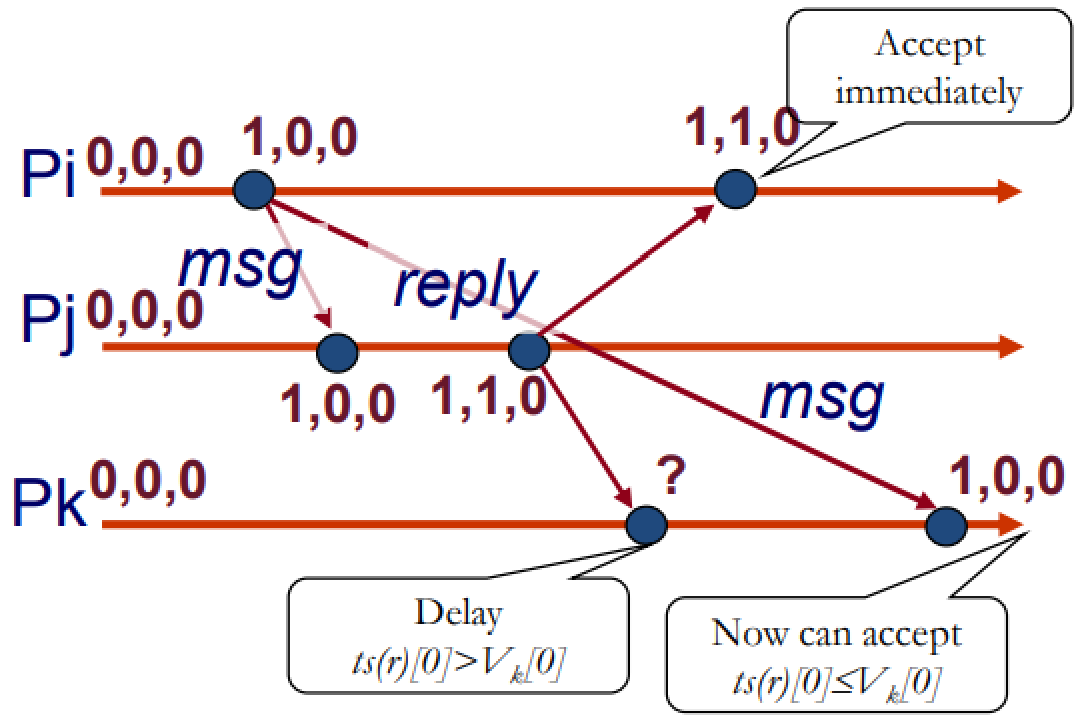
Sending a message (example of network fault)



When the network recovers

All messages are recevived in the correct order

Vector clocks for causal delivery



- Order between messages and replies is preserved
- Increment personal clock only when sending a message
- On message reception check the clocks
- Hold a message until all previous messages are received:
  - $ts(r)[j] = Vk[j]+1$
  - $ts(r)[i] \leq Vk[i]\ \forall\ i \neq j$
- If there are no previous messages accept the message and merge the clocks

Sending a message (code)

```
public Message send(String msg, String sender) {
    try {
        pushLock.lock();
        vectorClocks.put(sender, vectorClocks.get(sender) + 1);
        Message m = new Message(msg, Map.copyOf(vectorClocks), sender);
        msgList.add(m);
        propertyChangeSupport.firePropertyChange( … );
        return m;
    } finally {
        pushLock.unlock();
    }
}
```

Only one message at the time can be add to a chat

Increment the sender's clock

Create the message with updated clocks

Update GUI

Checking vector clocks on reception (code)

Check if one entry in the vector clock map has increased

The first entry that is increased by 1, we assume it's the sender

If any other entry has increased, or if any entry has increased more than 1, enqueue the message

If no clocks incremented, drop the message

Accept the message

```
private int checkVC(Message m) {
    Map<String, Integer> newClocks = Map.copyOf(m.vectorClocks());
    boolean senderFound = false;
    for (String u : users) {
        if (newClocks.get(u) == vectorClocks.get(u) + 1 && !senderFound){
            senderFound = true;
        } else if ((newClocks.get(u) > vectorClocks.get(u))) {
            return -1;
        }
    }
    if (!senderFound) return 0;

    return 1;
}
```