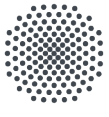


University of Stuttgart
Germany

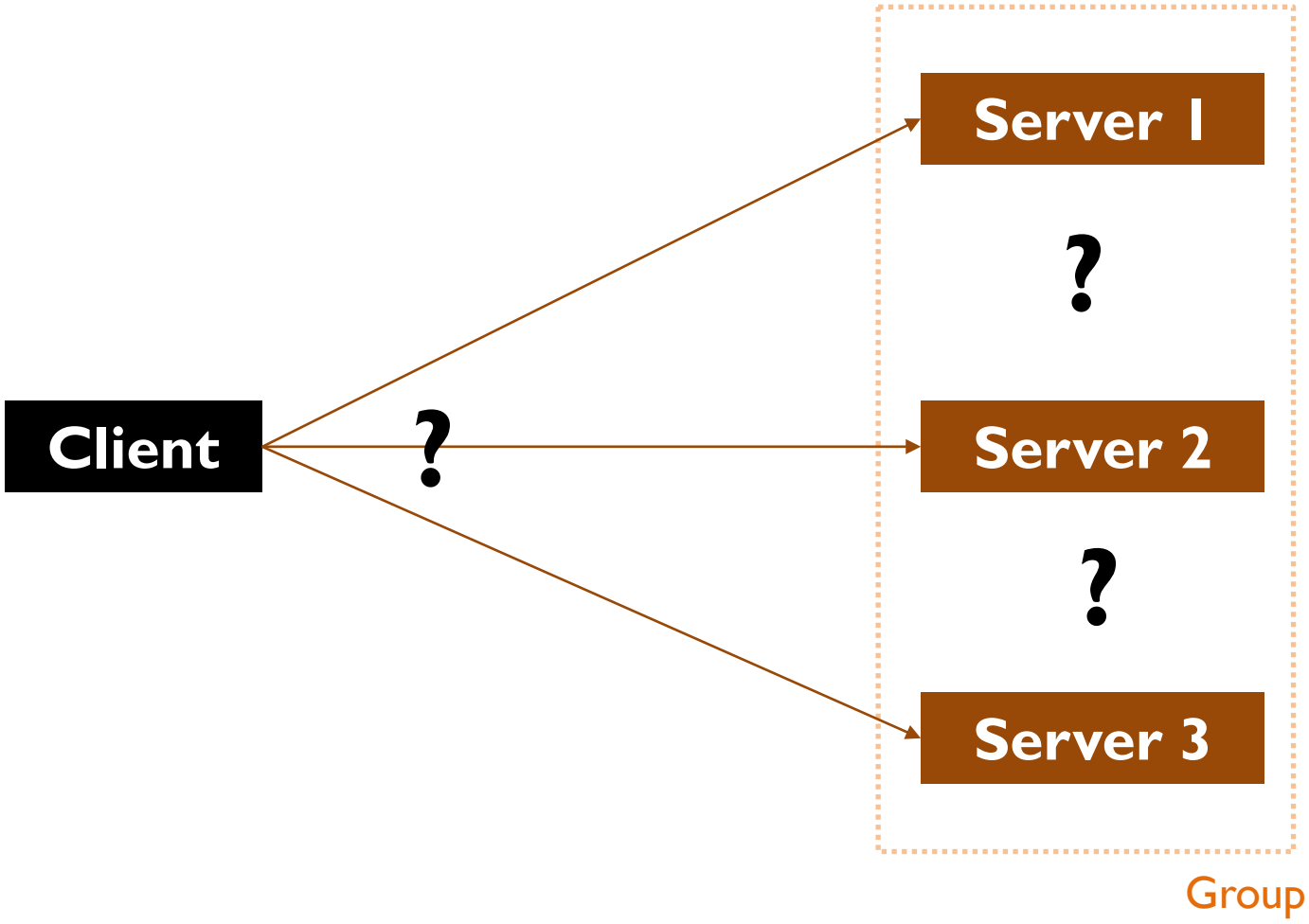
Herman Hollerith Zentrum

Dynamic discovery

Steps and implementation considerations

Ilche Georgievski
ilche.georgievski@iaas.uni-stuttgart.de
Room: U38 0.353

2020
Summer


Server 1
?
Client
?
Server 2
?
Server 3
Group

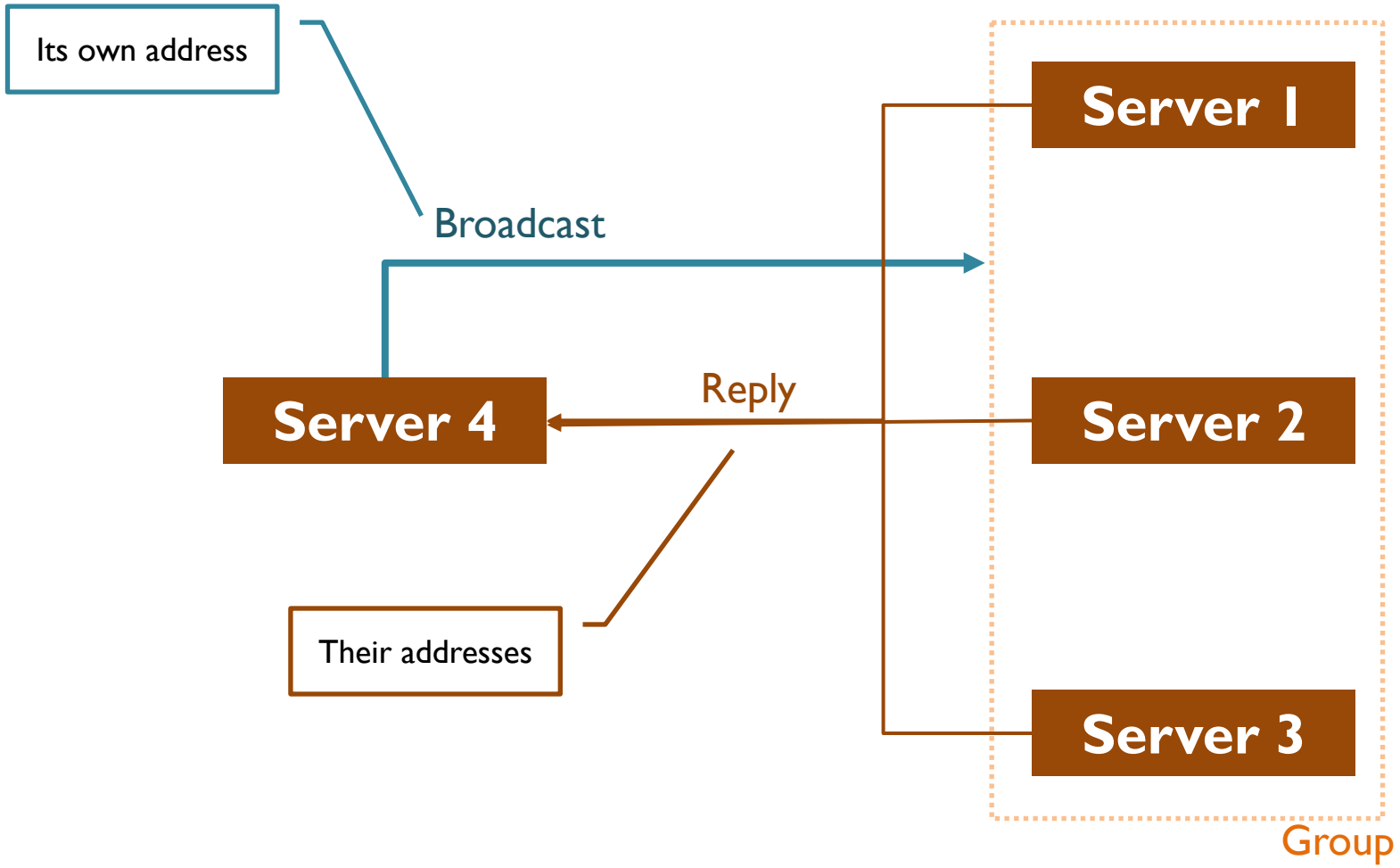
Discovery

- Identification of communication participants is a necessary precondition to any communication
- Two levels:
  - Each participants plays one or more roles
  - New participant initiate communication to participants of a single type (group)
- To establish communication, the initiator must identify a role participant or group that it will send message to
  - Participant addresses should not be hard-coded

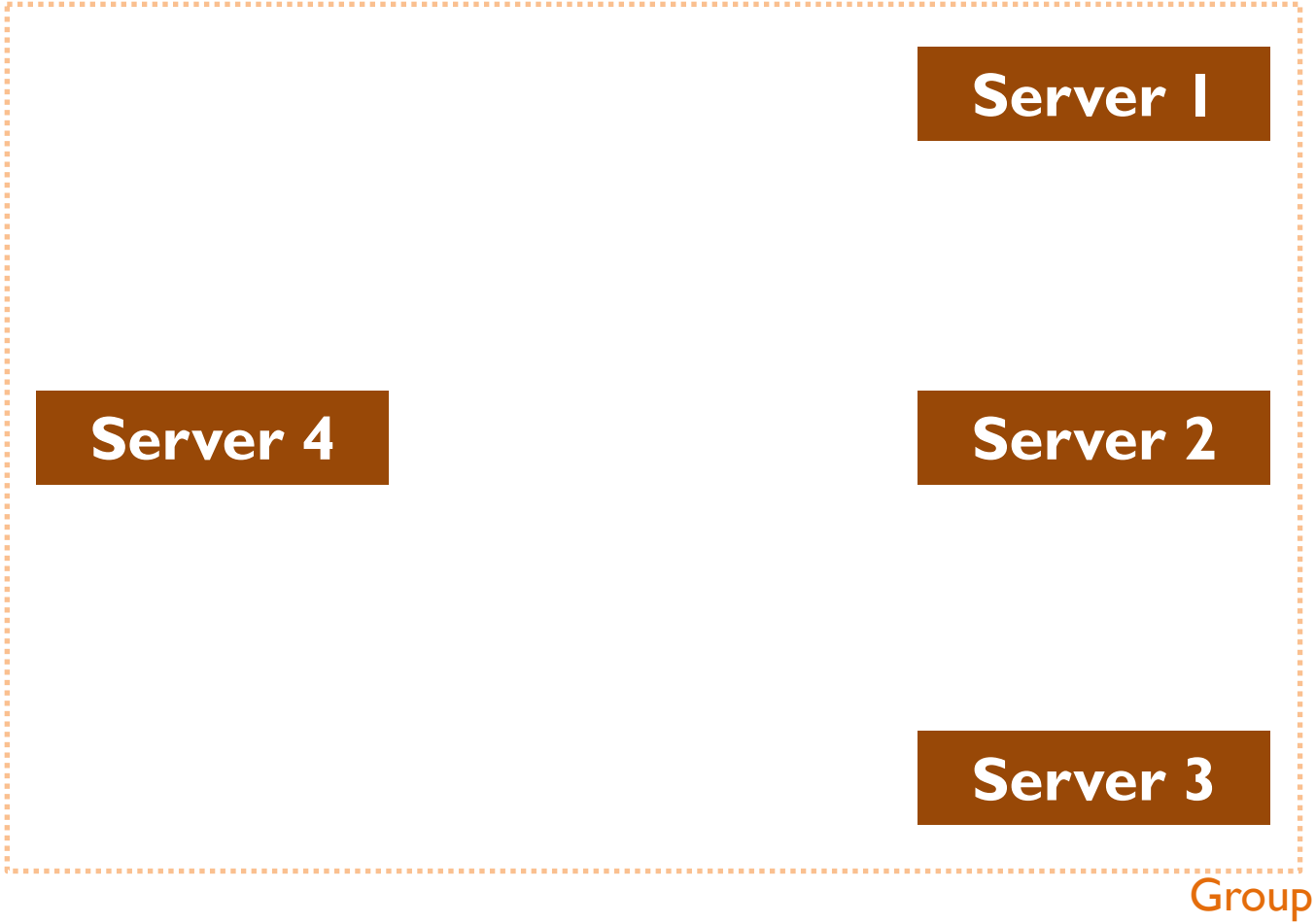
Dynamic discovery

How can a new participant find someone when it has no knowledge about available participants?

Dynamic discovery

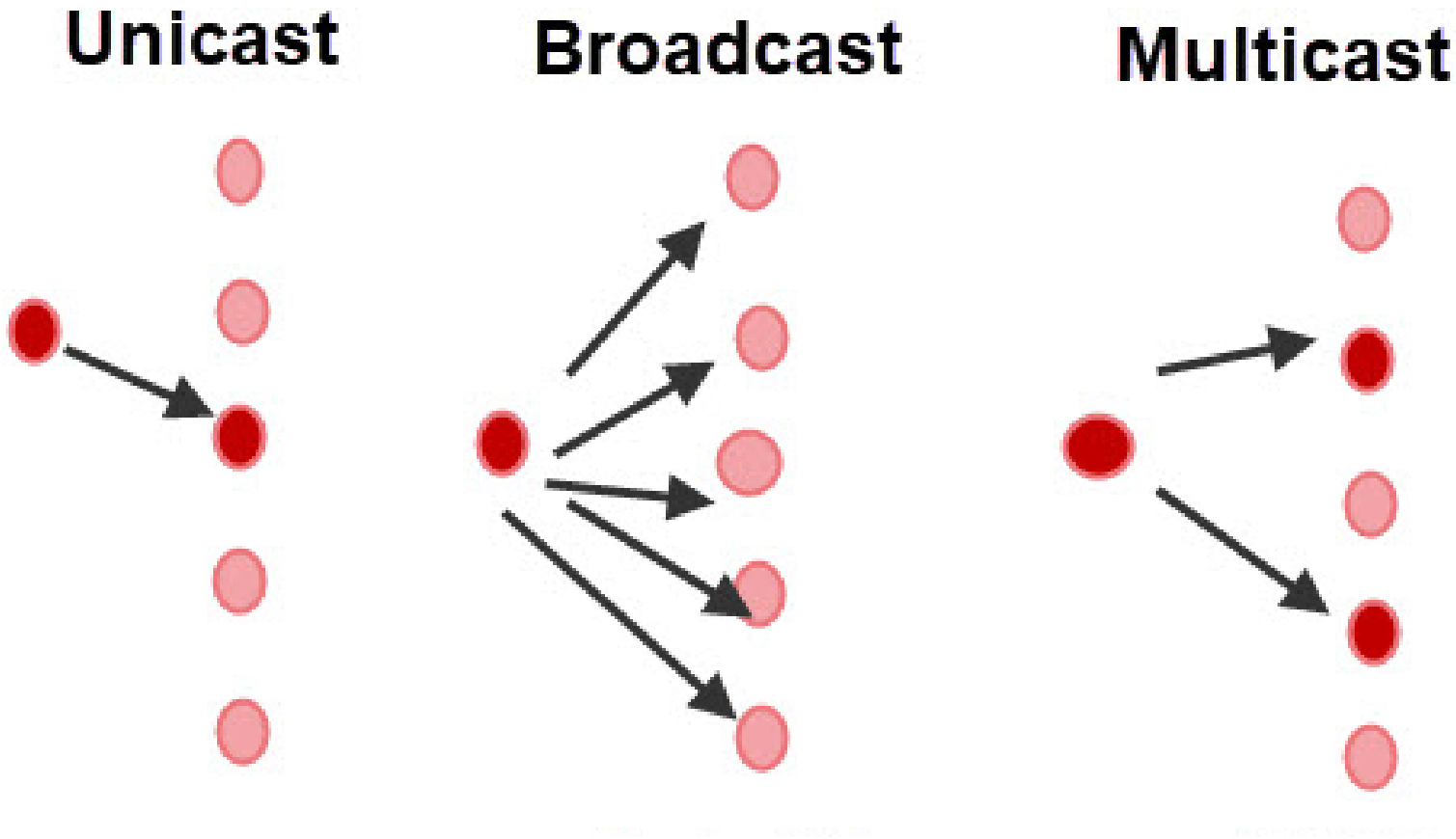


Dynamic discovery



Steps

1. New participant sends a broadcast message
   - No knowledge of any specific recipient
   - Knowledge only of a broadcast address
   - Broadcast message includes the address of the new participant
2. Potential recipients receive the broadcast message
   - Each recipient continuously listens for broadcast messages
   - Each recipient updates its group view
3. Potential recipients respond with a reply message
   - Reply message includes recipient's address
4. New participant collects all replies
   - New participant creates its own group view

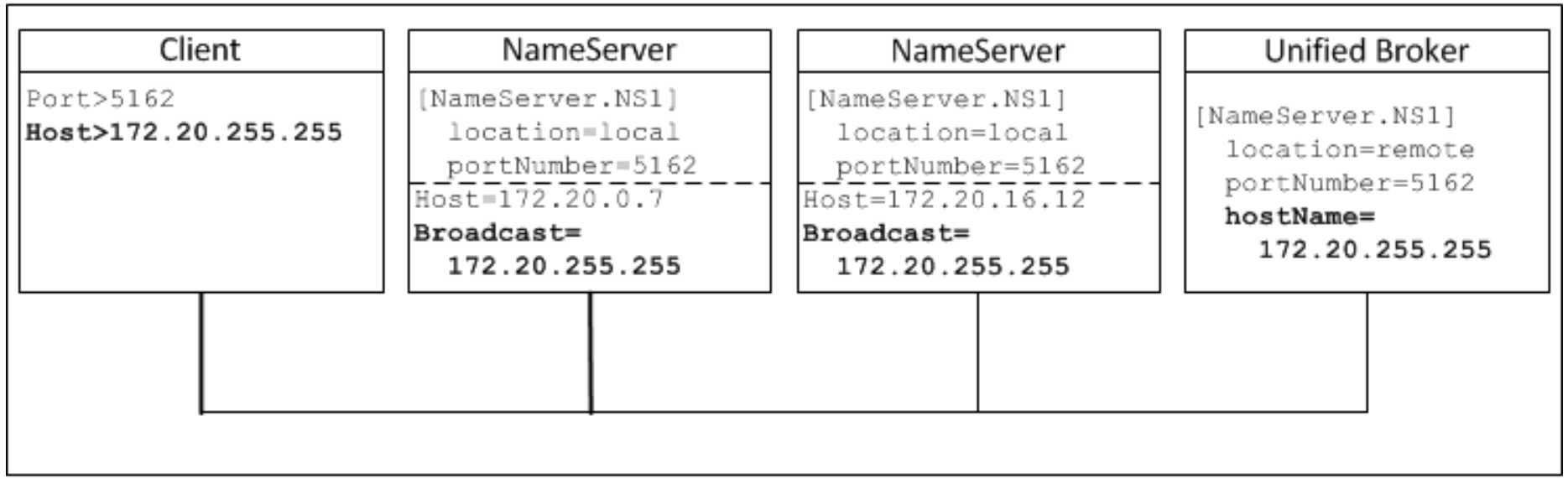

Unicast
Broadcast
Multicast

With UDP broadcast

DYNAMIC DISCOVERY

UDP broadcast

- UDP message to all computers in a LAN
- Broadcast address
  - Highest address in the local subnetwork



Considerations

- `socket.setsockopt(level, optname, None, optlen: int)`
- Relevant options
  - `SO_BROADCAST`
  - `SO_REUSEADDR`
  - `SO_REUSEPORT` (not available in Windows)
- Documentation
  - Linux
  - Windows
  - Mac OS

Broadcast sender

- Create a socket
- Enable the socket for broadcasting
- Send a message to a broadcast address
- Receive a reply message
- Collect all reply messages
- Close the socket
- Create a group view

Broadcast listener

- Create a socket
- Enable the socket to support multiple connections
- Start listening
- Receive a broadcast message
- Update the group view
- Send a reply message
- Continue listening for broadcast messages

Broadcast sender

```
# Broadcast address
BROADCAST_IP = "192.168.0.255"
BROADCAST_PORT = 5972

# Local host information
MY_HOST = socket.gethostname()
MY_IP = socket.gethostbyname(MY_HOST)

...
message = MY_IP + ' sent a broadcast'
broadcast(,, message)

...
broadcast_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
broadcast_socket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
broadcast_socket.sendto(str.encode(broadcast_message), (BROADCAST_IP, BROADCAST_PORT))
broadcast_socket.close()
```

Broadcast listener

```
listen_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# listen_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEPORT, 1)
listen_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
listen_socket.bind((BROADCAST_IP, BROADCAST_PORT))

    while True:
        data, addr = listen_socket.recvfrom(1024)
        if data:
            print("Received broadcast message:", data.decode())
```