Persistent Streams: The Internet with Ephemeral Storage

Oskar Carl, Peter Zdankin, Matthias Schaffeld, Viktor Matkovic,
Yang Yu, Timo Elbers and Torben Weis
Distributed Systems, University of Duisburg-Essen

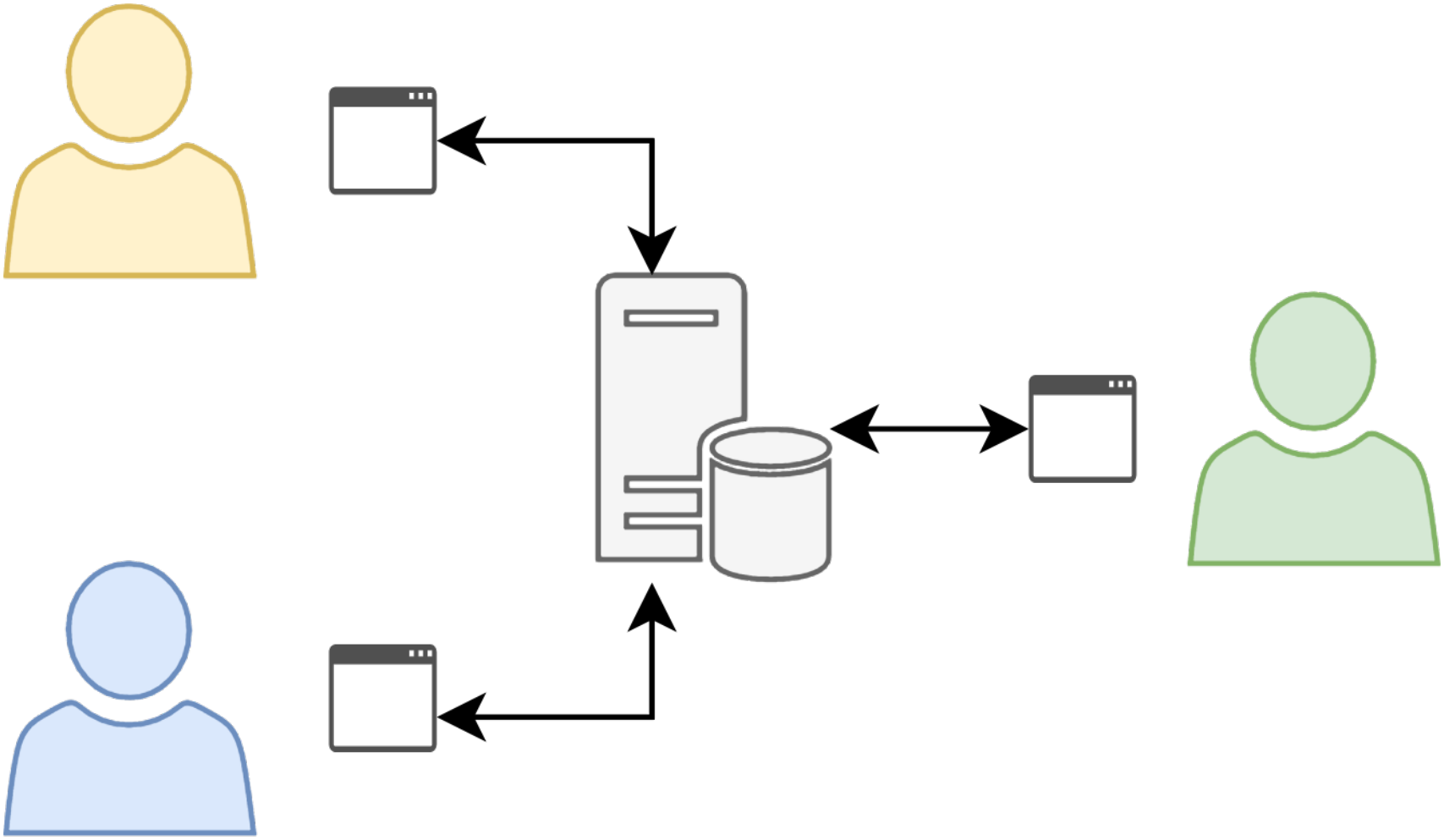
UNIVERSITÄT DUISBURG ESSEN
Open-Minded

oskar.carl@uni-due.de
https://vs.uni-due.de/

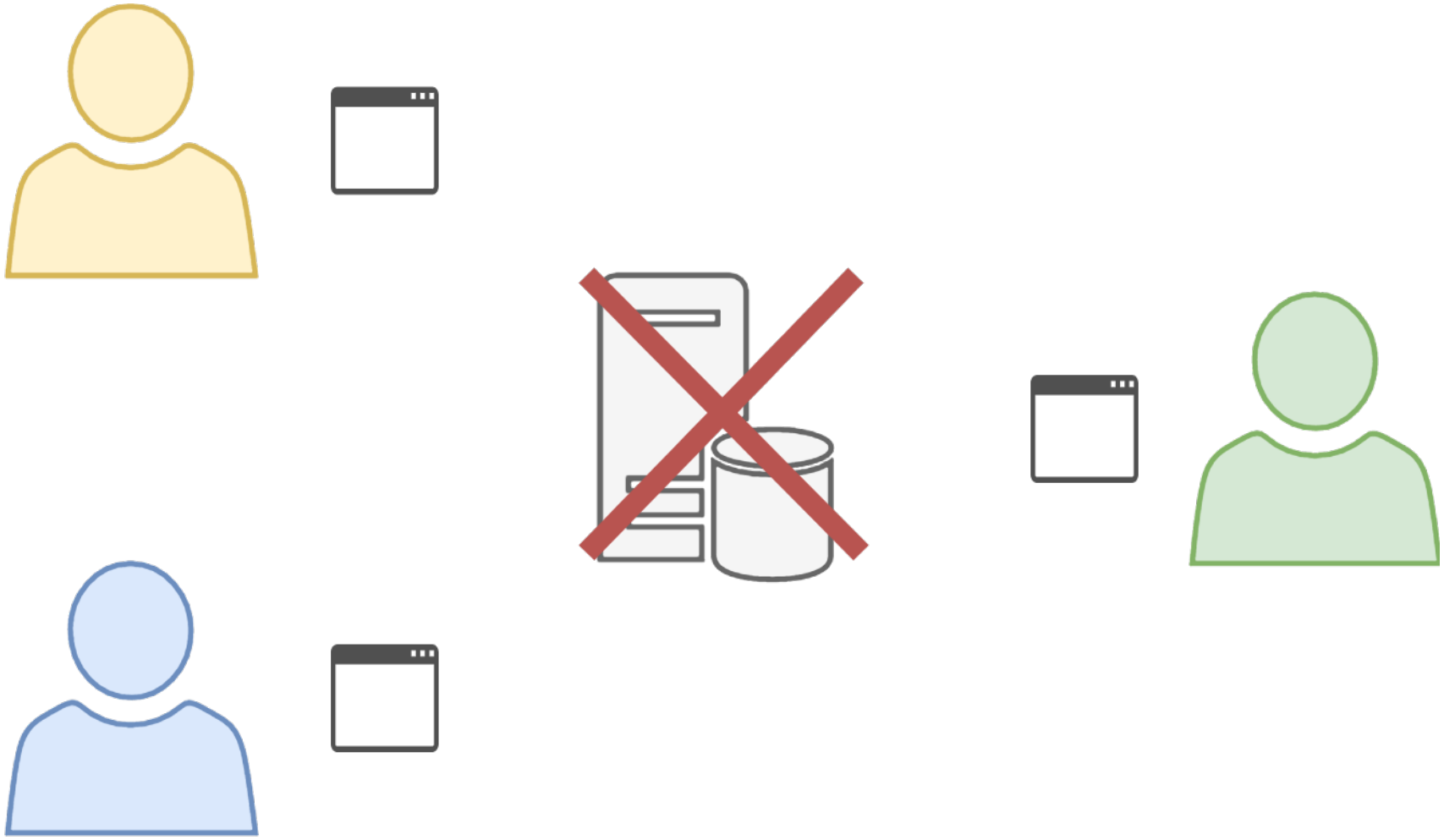
Communication

- Often handled by application-specific servers
  - Client-server designs
  - If the application server is gone, communication is gone

Centralized Communication



Centralized – Server Failure



Using Peer-to-Peer

- Usually complex, slow, and/or unreliable
  - High churn rates cause issues
  - Commonly low upload bandwidth
  - Streaming only feasible if same content for many peers

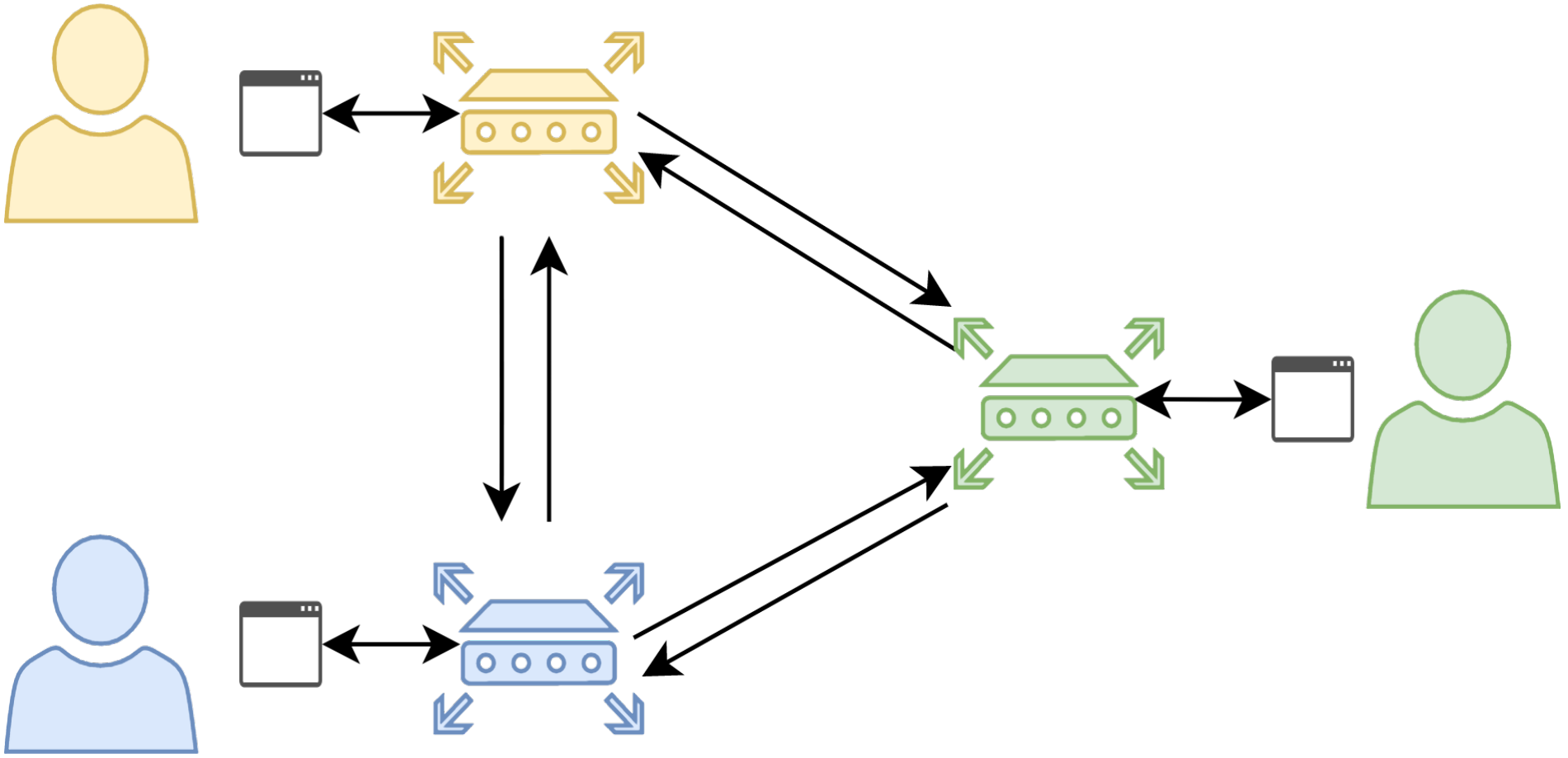
Persistent Streams

- Communication protocol
- Decoupling of application provider and communication
- Temporal decoupling
- Built into the base layer of the internet

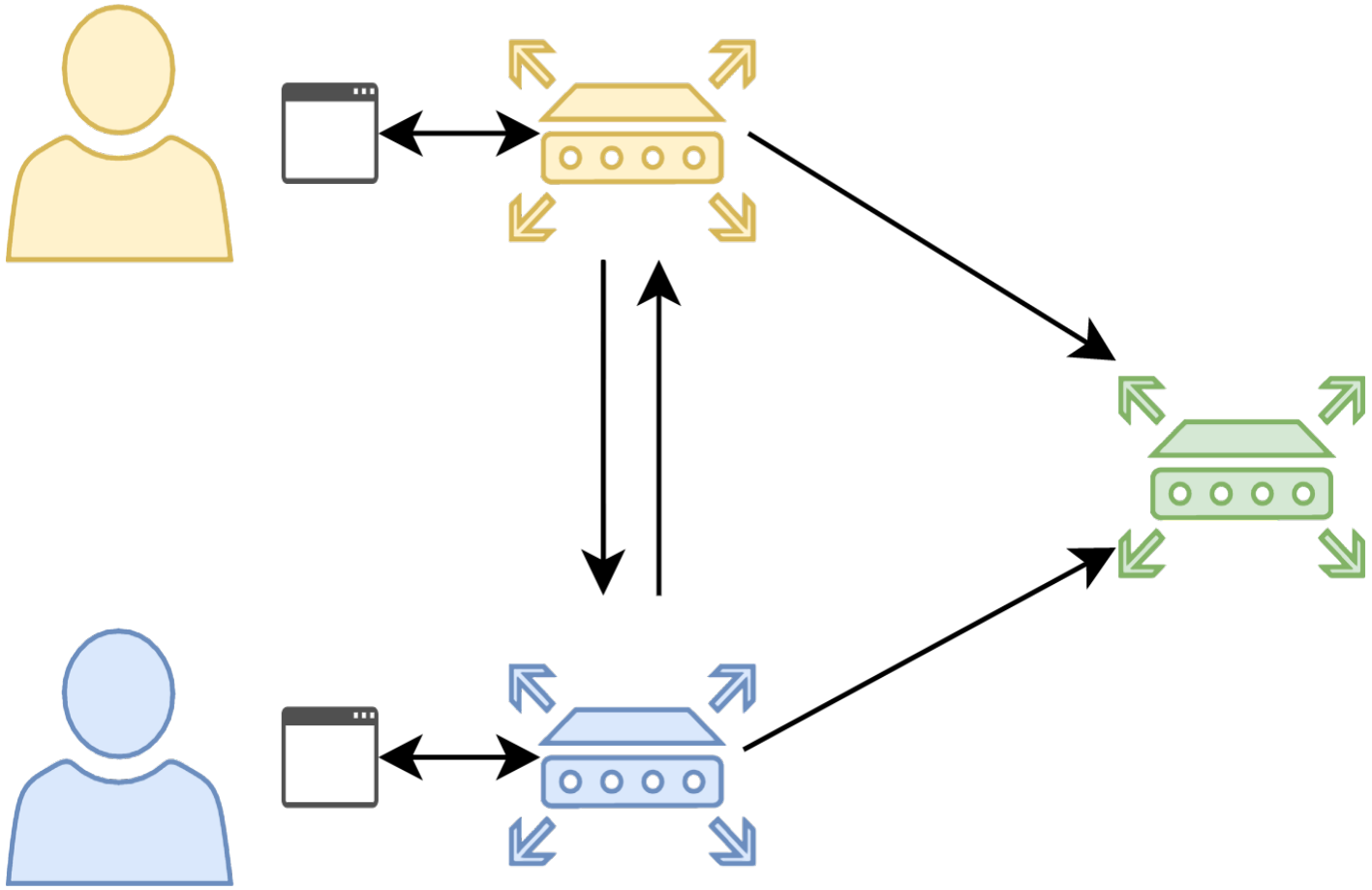
How?

- Applications are *clients*
- Communication flows through *agents*
  - Application-agnostic
  - Hosted anywhere
  - Requirement: ‘always’ online
- Agents forward and cache data
  - *The internet* gains retention
  - No application-specific solution required

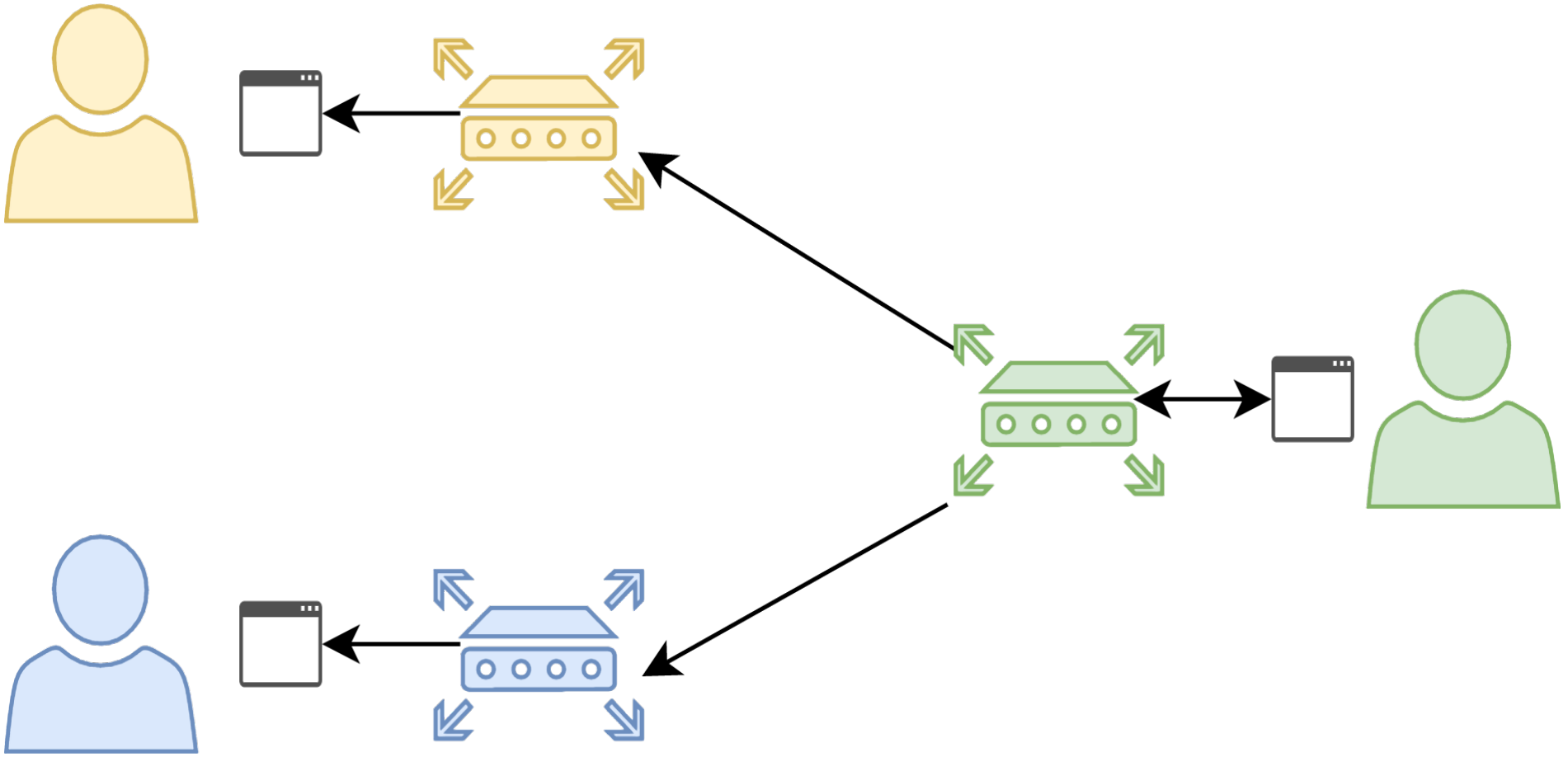
Persistent Streams – Overview



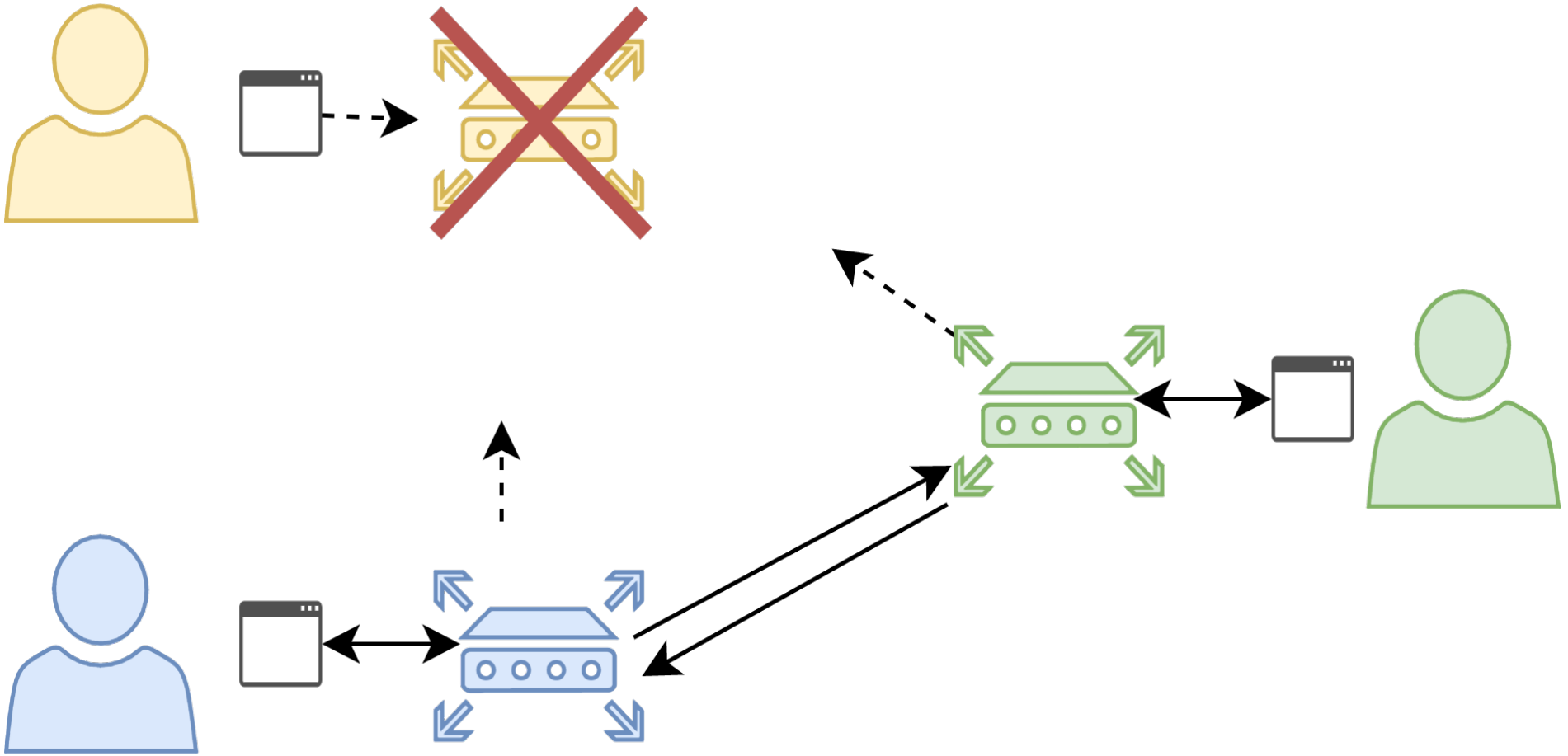
PS – User Offline



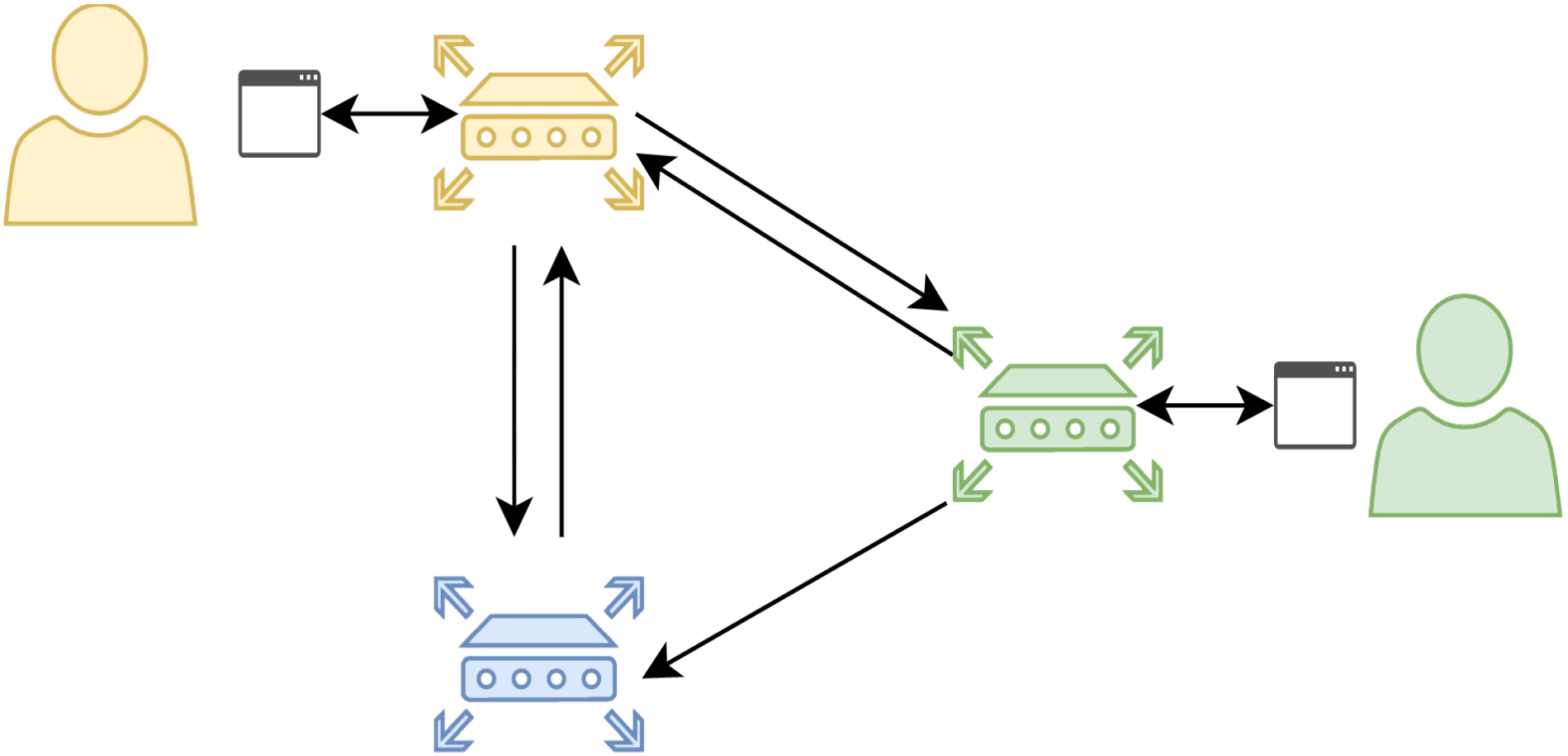
PS – Delayed Transmission



PS – Agent Failure



PS – Agent Recovery

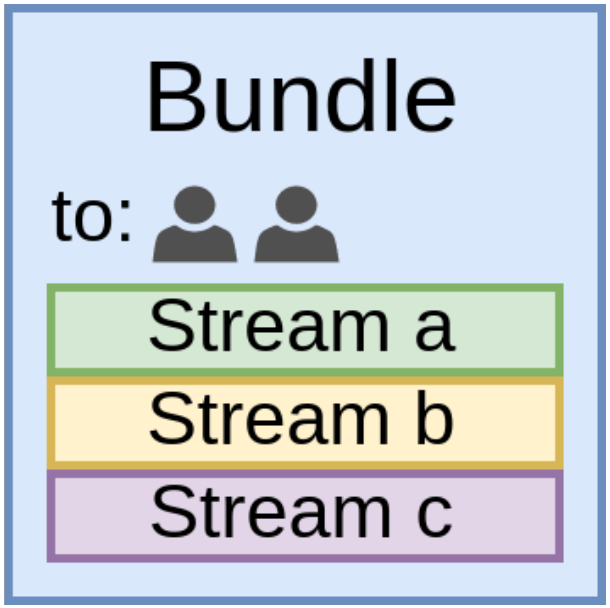


Data Handling

- User data is handled as binary
  - Arbitrary interoperability possible
- No E2E encryption
  - Only transport

Metadata

- Some metadata required
  - *Streams* contain the data
  - *Bundles* group streams together
  - Recipients set per bundle
- Bundles can be public



Data Distribution

- Recipients accept *bundle* once
- *Streams* are pushed to all who accepted
  - No further *accept* required
- Public bundles can be requested

Beyond Communication

- Send data to yourself
  - Communication between *your* clients
- Mark data as *persistent*
  - Not discarded after forwarding
- Grant access to groups of users
  - read/write: multi-user access

Central Issues

How efficient is it?

How does it provide persistence?

Latency

- *Head of line blocking* in agents
  - Causes undesirable delays (twice!)
- → Stop waiting for missing packets
  - Forward instantly (if possible)
  - Can be done transparently in QUIC

Persistence

- Data sent to agent must not be lost after acknowledgement!
  - What if the agent fails?
- Otherwise we get inconsistent behaviour

Persistence

- Store incoming data in NVRAM
  - It's fast enough for 10G Networking
  - Most data can be forwarded directly
  - Other data can be cached
- If not retrieved after some time
  → demote to slower storage

Example

- 10G networking, 2TB of NVRAM
  - ~27m until it's full
- Most data can be discarded
- Or stored on slower tiers after forwarding
- Not retrieved after that time?
  → not time-critical

Thanks for your attention!

Questions?

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

oskar.carl@uni-due.de
https://vs.uni-due.de/