



Application Layer

Prof. Anja Feldmann, Ph.D.

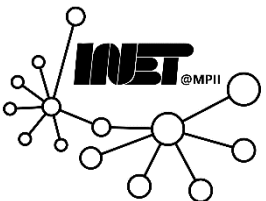
(Based on slide deck of Computer Networking, 7th ed., Jim Kurose and Keith Ross.)



Outline



- Principles of network applications
 - *Examples, design, architecture, and implementation*



Application Layer



Goals

- Conceptual, implementation aspects
- Implementation paradigms
 - *Client-server and Peer-to-peer*
 - *Sockets*
- Transport-layer service models

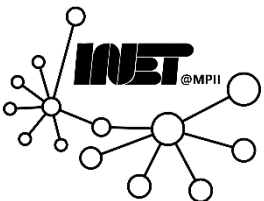


Application Layer



Goals

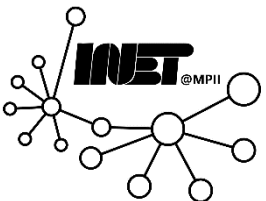
- **Conceptual, implementation aspects**
- Implementation paradigms
 - *Client-server and Peer-to-peer*
 - *Sockets*
- Transport-layer service models



What!?



Image credits: Tim Gouw, www.pexels.com



Network Applications: Examples



To: Anja Feldmann

Cc:

duckduckgo.com

DuckDuckGo — Privacy, simplified.

DuckDuckGo

The search engine that doesn't track you. [Help Spread DuckDuckGo!](#)

mastodon

JOIN THE FEDERATION!
mastodon awaits

Username @mastodon.social

E-mail address

Password

Confirm password

Sign up

By clicking "Sign up" below you agree to follow the rules of the instance and our terms of service.

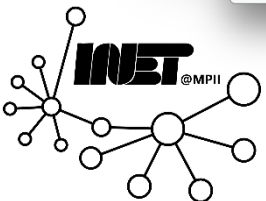
OR

Mastodon

Mastodon hosted on mastodon.social

This page describes the mastodon.social instance - wondering what Mastodon is? Check out [joinmastodon.org](#) instead! In essence, Mastodon is a decentralized, open source social network. This is just one part of

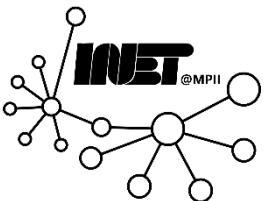
Open "https://mastodon.social" in a new tab



Network Applications: Examples



E-mail, Web, Text messaging, Remote login, P2P file sharing, Multi-user network games, Streaming stored video (e.g., YouTube, Hulu, and Netflix), Voice over IP (e.g., Skype), Real-time video conferencing, Social networking, search, ...



Application Layer



Goals

- Conceptual, implementation aspects
- **Implementation paradigms**
 - *Client-server and Peer-to-peer*
 - *Sockets*
- Transport-layer service models

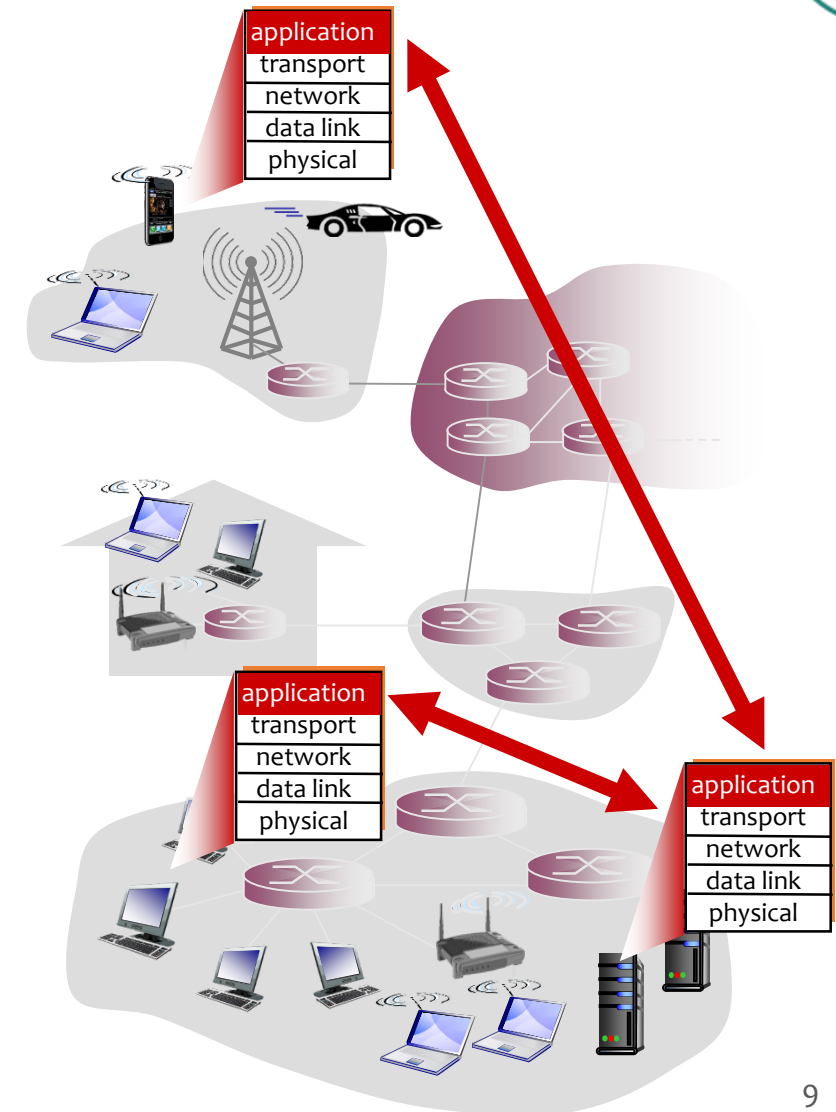


Creating a network application



- Write *programs that*
 - Run on (*different*) end systems
 - Communicate over network

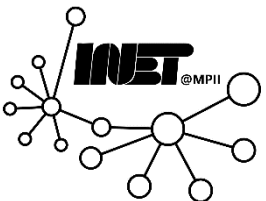
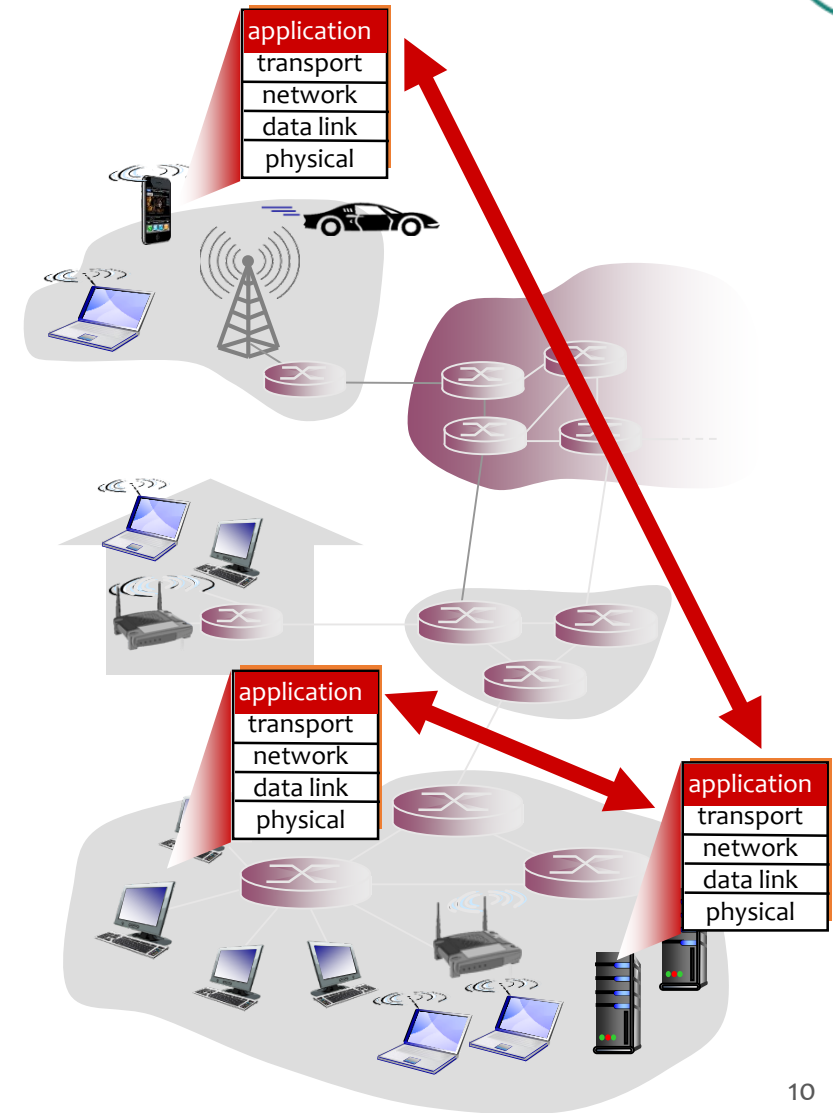
Example: Web server software communicates with browser software



Creating a network application



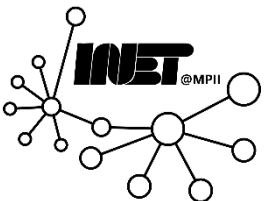
- **No need** to write software for network-core devices
 - Why?
 - Core devices do not run user applications
 - Advantage?
 - Rapid development and deployment



Application Architectures



- Based on how data is exchanged ...
 - **Client-Server**
 - **Peer-to-Peer**

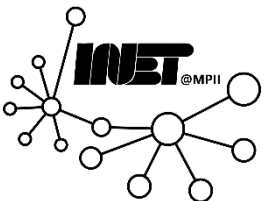
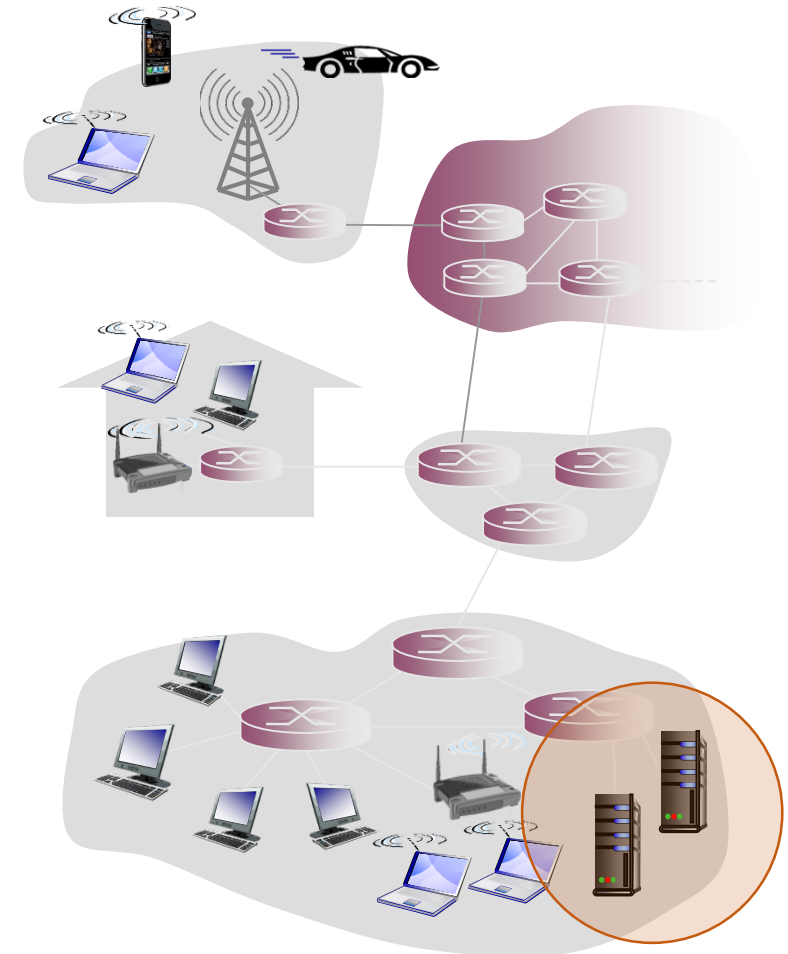


Client-Server Architecture



Server:

- *Always-on* host
- *Permanent* IP address
- Data centers for scaling



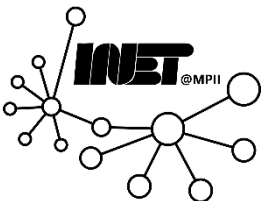
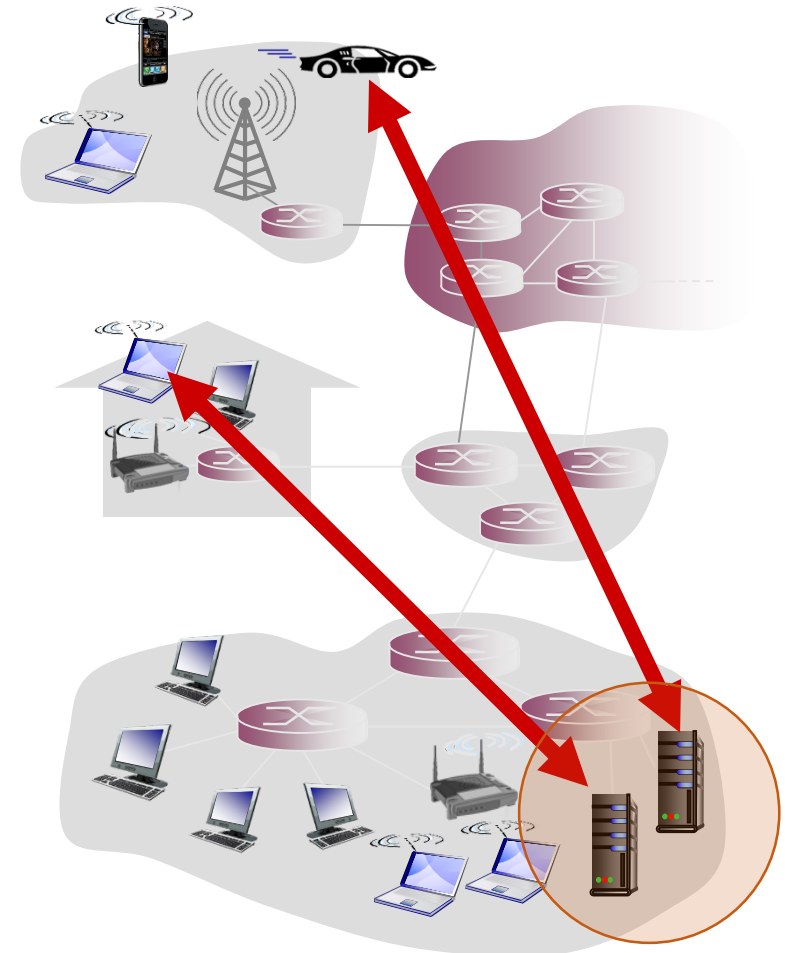
Client-Server Architecture



Client(s):

- Communicate with **server**
- May be intermittently connected
- May have *dynamic* IP addresses

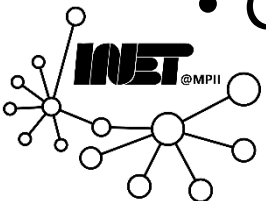
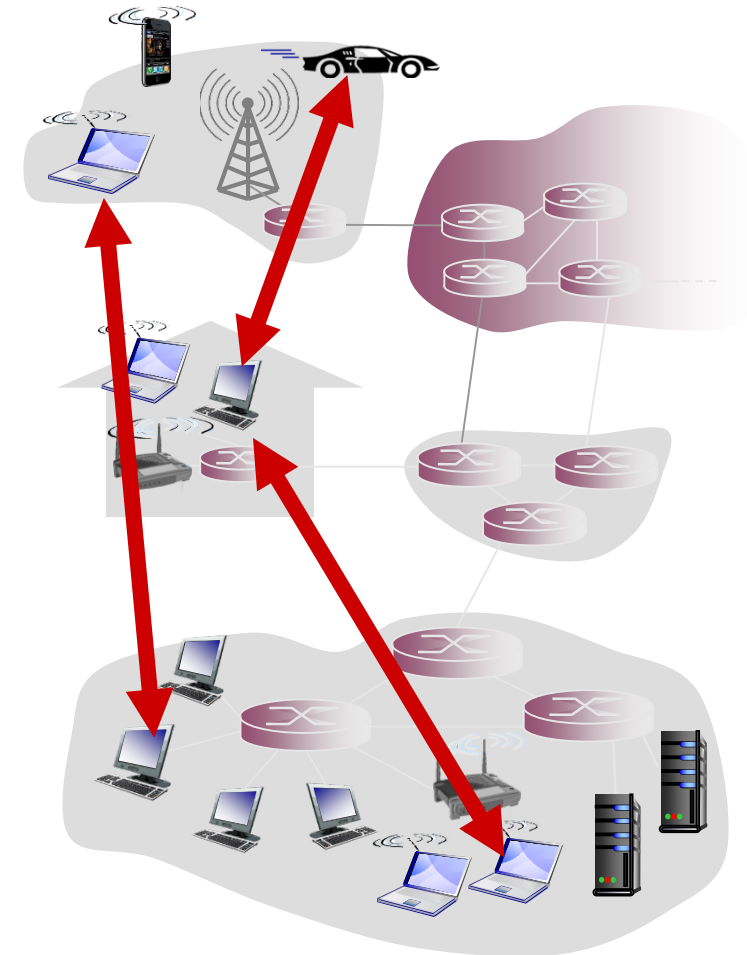
Do **not** communicate directly with each other



Peer-to-Peer Architecture



- **No** always-on server
- *Arbitrary* end systems directly communicate
- **Peers**
 - Request service from other peers
 - Provide service (in return) to other peers
- **Self Scalability** – new peers ...
 - Bring new service capacity **and**
 - New service demands
- Peers are intermittently connected and change IP addresses
- Complex management

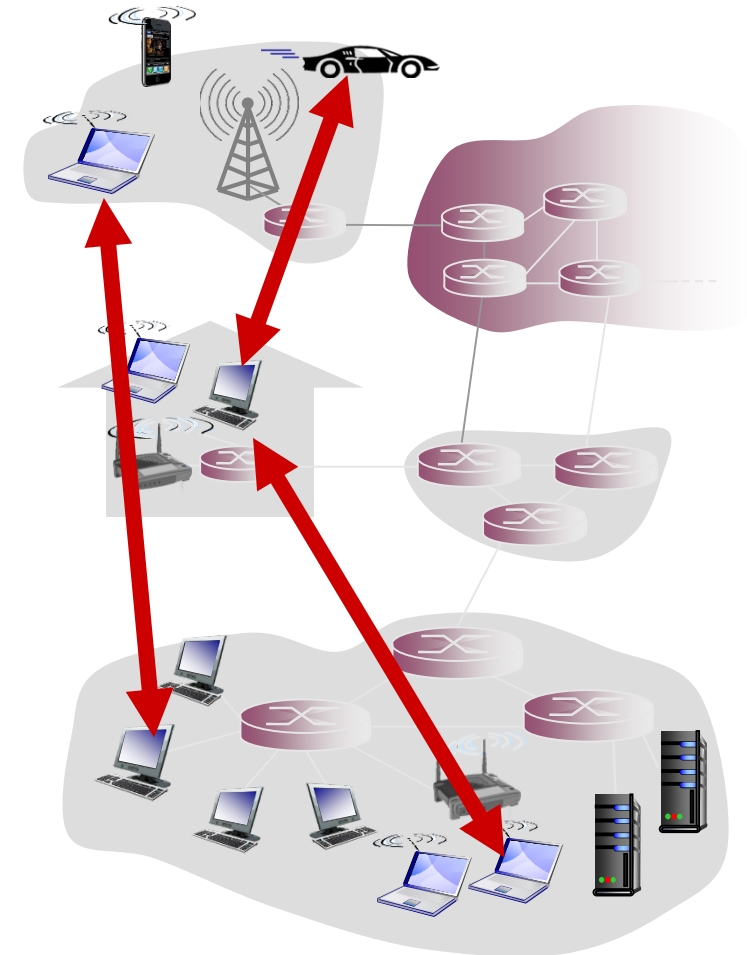


Peer-to-Peer Architecture



- **Peers**

- Request service from other peers
 - Provide service (in return) to other peers
-
- Peers are intermittently connected and change IP addresses
 - **Complex management**



Communicating Processes



- **Process**: program running within a host

Communication between two processes?

- Within **same** host
 - Using **inter-process communication** (defined by OS)
- In **different** hosts
 - By exchanging messages



Processes in Client-Server Arch.



- **Client** process:
 - Process that *initiates* communication
- **Server** process:
 - Process that *waits* to be contacted

OK, how about in *peer-to-peer* architecture?

Have both client and server processes!



Application Layer



Goals

- Conceptual, implementation aspects
- Implementation paradigms
 - *Client-server and Peer-to-peer*
- Interface: Sockets
- Transport-layer service models

