Don’t hesitate to ask questions
also over Slack (email) after the session

Main contact persons:

Tobias  Eng. & Ger.
Alexander  Eng. & Ger.
Rüdiger  Eng. & Ger.
Hendrik  Eng. & Ger.
Eric  Eng. & Ger.
Communication Networks

Exercise 2

Quick recap and task 2.4

Time for you to solve the tasks

More recap and discussion of task 2.2

Solutions will be published next week
routing

How do you guide IP packets from a source to destination?
Think of IP packets as envelopes
Like an envelope, packets have a header.
Like an envelope, packets have a payload
The header contains the metadata needed to forward the packet.

Identify the source and destination of the communication.
Laurent

src: SEAT

dst: Google
Let’s zoom in on what is going on between two adjacent routers
Upon packet reception, routers **locally** look up their forwarding table to know where to send it next.
Forwarding decisions necessarily depend on the destination, but can also depend on other criteria.

<table>
<thead>
<tr>
<th>criteria</th>
<th>destination</th>
<th>mandatory (why?)</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+any other header</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Task 2.4: (Source-and)-Destination-Based Routing

Let’s compare these two in terms of

- required state
- path diversity
With source- & destination-based routing, paths from different sources can differ.
With destination-based routing, paths from different source coincide once they overlap.
While forwarding is a *local* process, routing is inherently a *global* process.

How can a router know where to direct packets if it does not know what the network looks like?
Essentially, there are three ways to compute valid routing state:

<table>
<thead>
<tr>
<th>Intuition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1    Use tree-like topologies</td>
<td>Spanning-tree</td>
</tr>
<tr>
<td>#2    Rely on a global network view</td>
<td>Link-State</td>
</tr>
<tr>
<td></td>
<td>SDN</td>
</tr>
<tr>
<td>#3    Rely on distributed computation</td>
<td>Distance-Vector</td>
</tr>
<tr>
<td></td>
<td>BGP</td>
</tr>
</tbody>
</table>
Communication Networks

Exercise 2

Quick recap and task 2.4

Time for you to solve the tasks

More recap and discussion of task 2.2

Solutions will be published next week
Communication Networks

Exercise 2

Quick recap and task 2.4

Time for you to solve the tasks

More recap and discussion of task 2.2

Solutions will be published next week
To build this global view, routers essentially solve a jigsaw puzzle.
Initially, routers only know their ID and their neighbors. 

D only knows, it is connected to B and C along with the weights to reach them (by configuration).
Each routers builds a message (known as Link-State) and **floods it** (reliably) in the entire network

D’s Advertisement

- edge (D,B); cost: 1
- edge (D,C); cost: 4
At the end of the flooding process, everybody share the exact same view of the network required for correctness see exercise.
Dijkstra will always converge to a unique stable state when run on *static* weights

cf. exercise session for the dynamic case
Task 2.4:
Dynamic weights

\[
\begin{align*}
\text{B} & \quad \text{A} \\
\text{C} & \quad \text{D}
\end{align*}
\]

\[e \quad \text{incoming traffic demand}\]
Unlike before, weights are bidirectional and represent link load.
The problem of oscillation is fundamental to congestion-based routing with local decisions

solution #1  
Use static weights  
i.e. don't do congestion-aware routing

solution #2  
Use randomness to break self-synchronization  
wait(random(0,50ms)); send(new_link_weight);

solution #3  
Have the routers agree on the paths to use  
essentially meaning to rely on circuit-switching
Quick recap and task 2.4

Time for you to solve the tasks

More recap and discussion of task 2.2

Solutions will be published next week