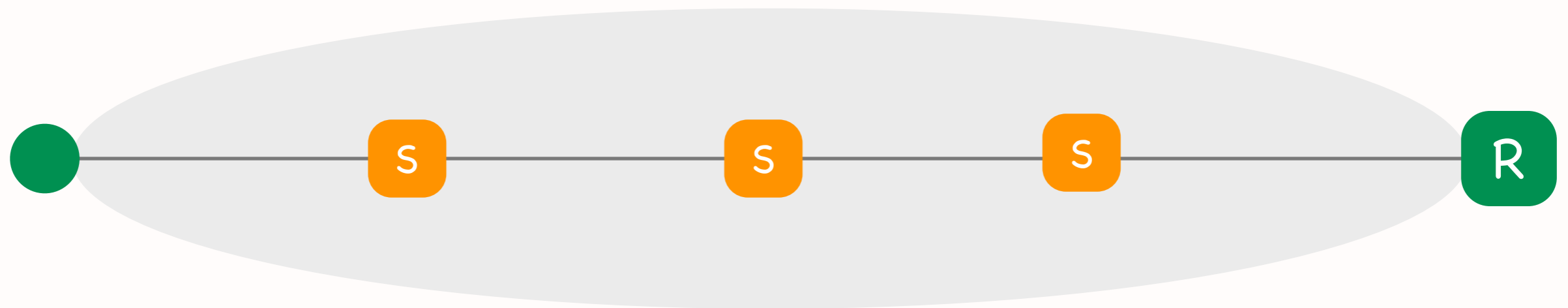
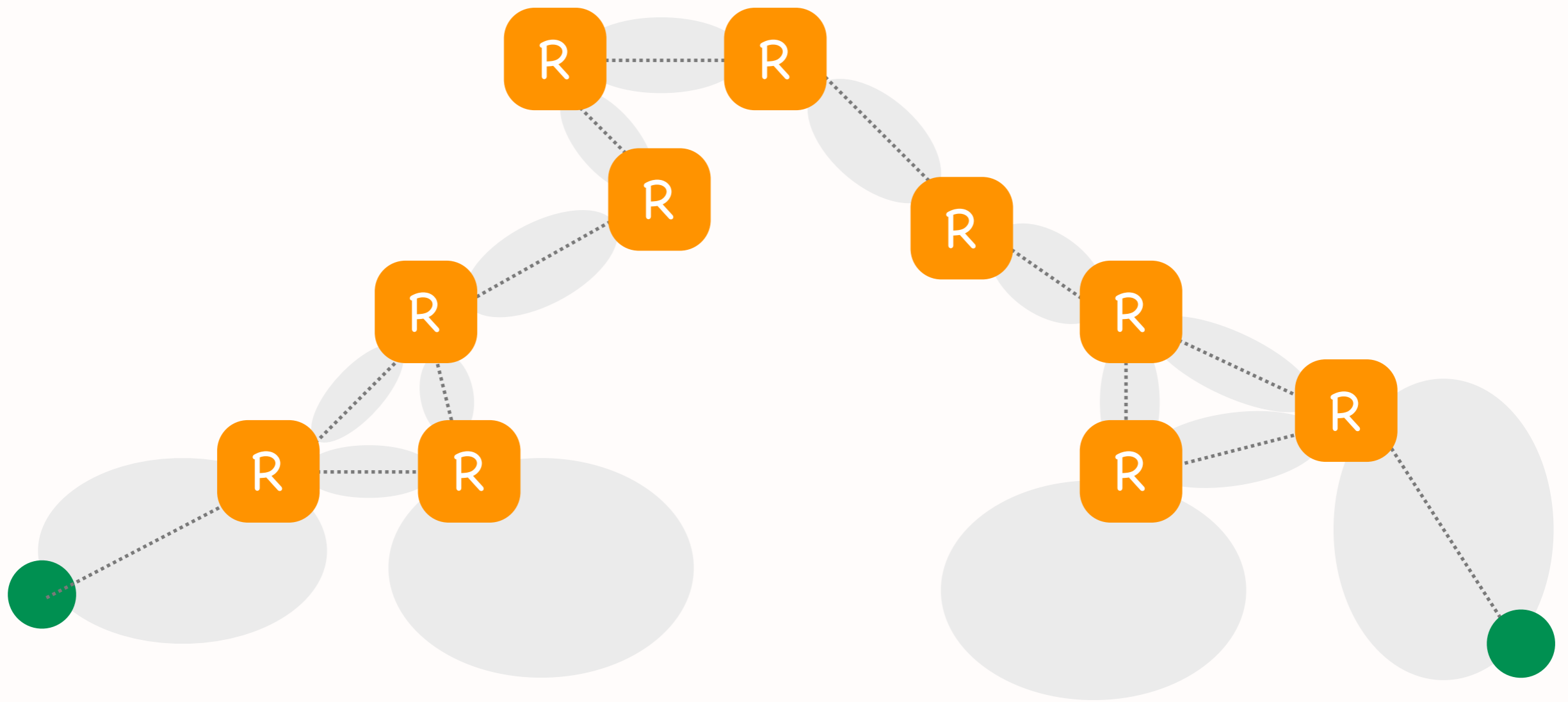


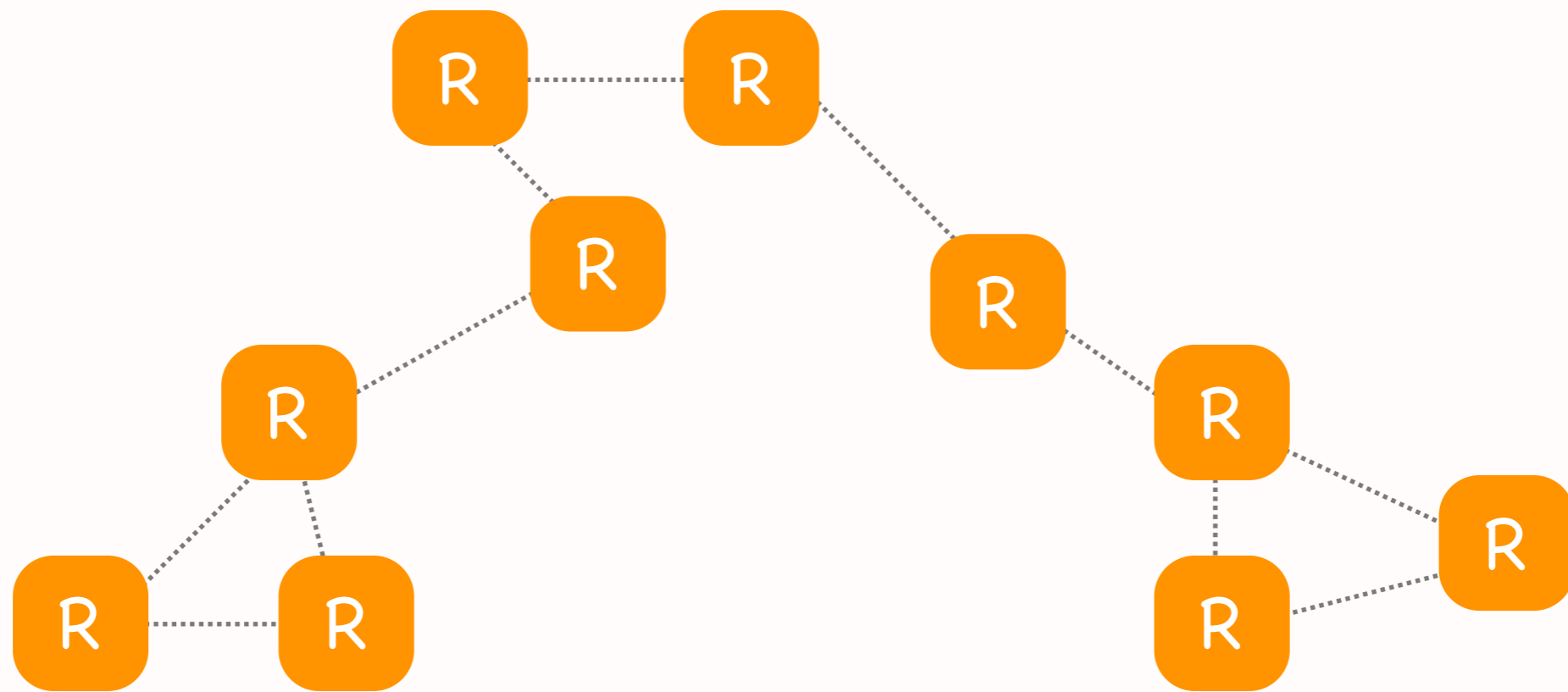
Three levels of hierarchy

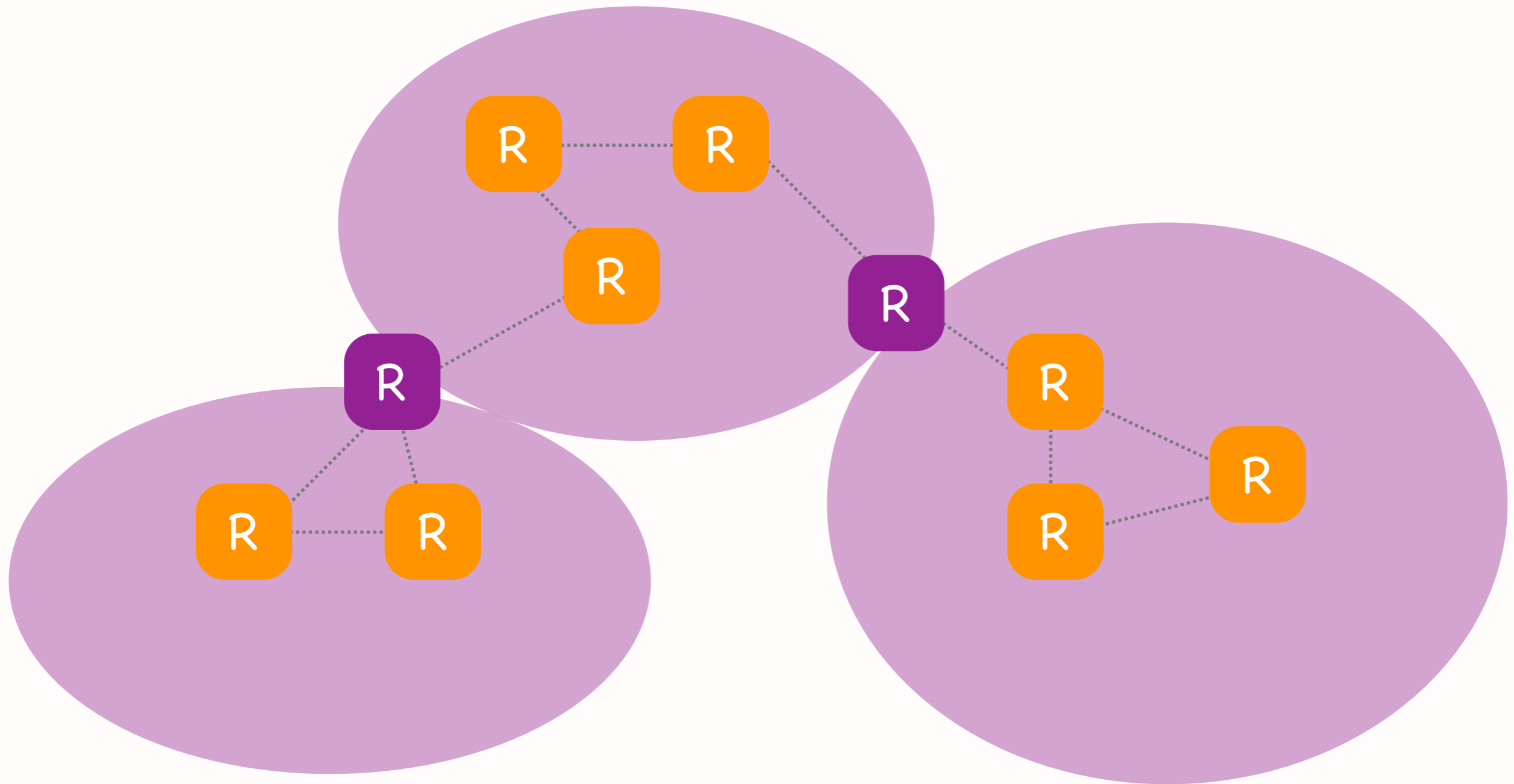
- IP subnet
 - * L2 forwarding
 - * L2 learning
- Autonomous System (AS)
 - * IP (L3) forwarding
 - * intra-domain routing
- Internet
 - * IP (L3) forwarding
 - * inter-domain routing (BGP)











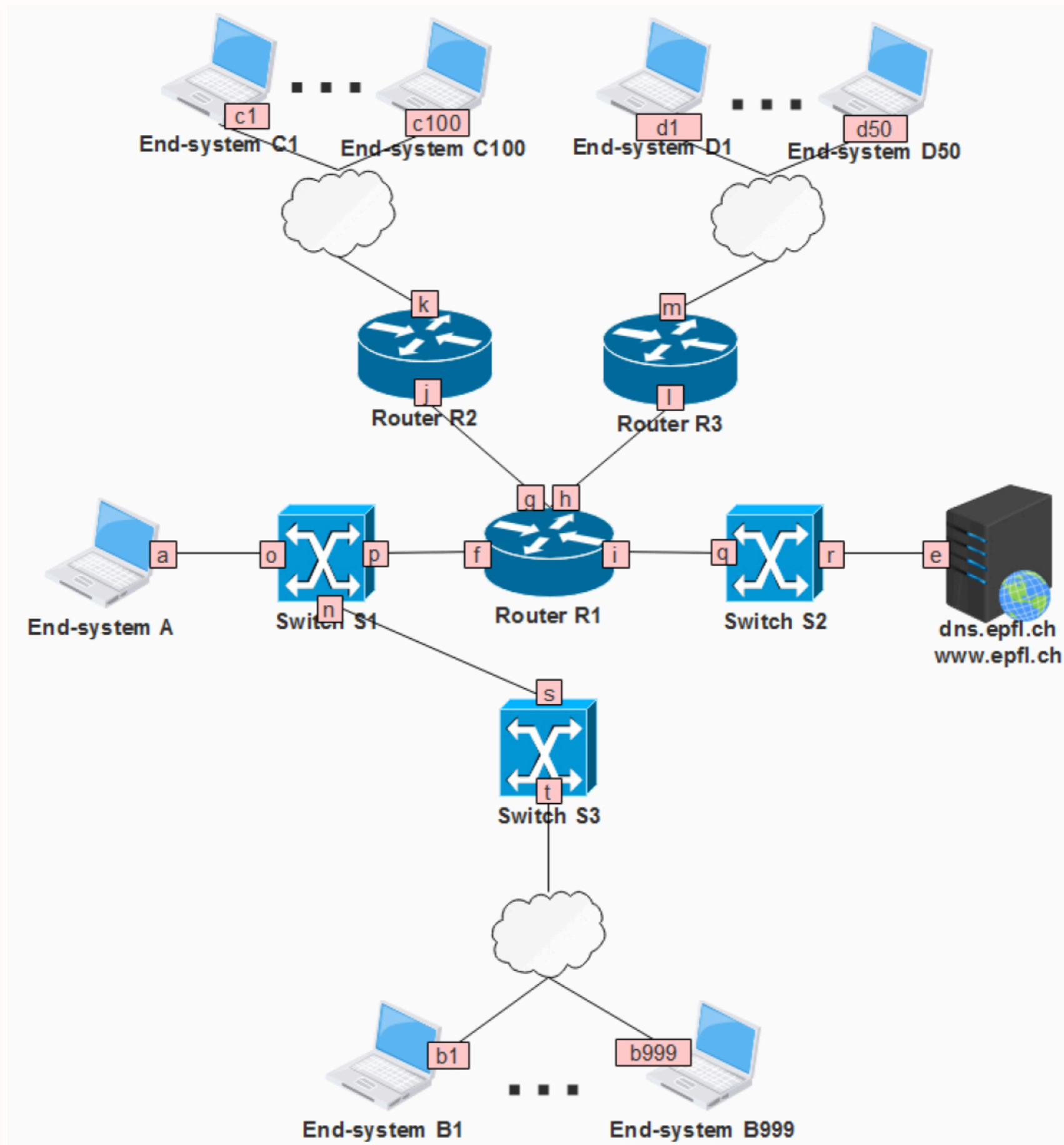
2 x two-level hierarchies

- IP subnet vs. Internet
 - * L2 vs. IP forwarding
 - * different forwarding processes, different layers => different packet headers
- Autonomous System (AS) vs. Internet
 - * intra-domain vs. inter-domain routing
 - * different routing protocols
 - * same forwarding process (IP), same layer

Question: Allocate IP addresses

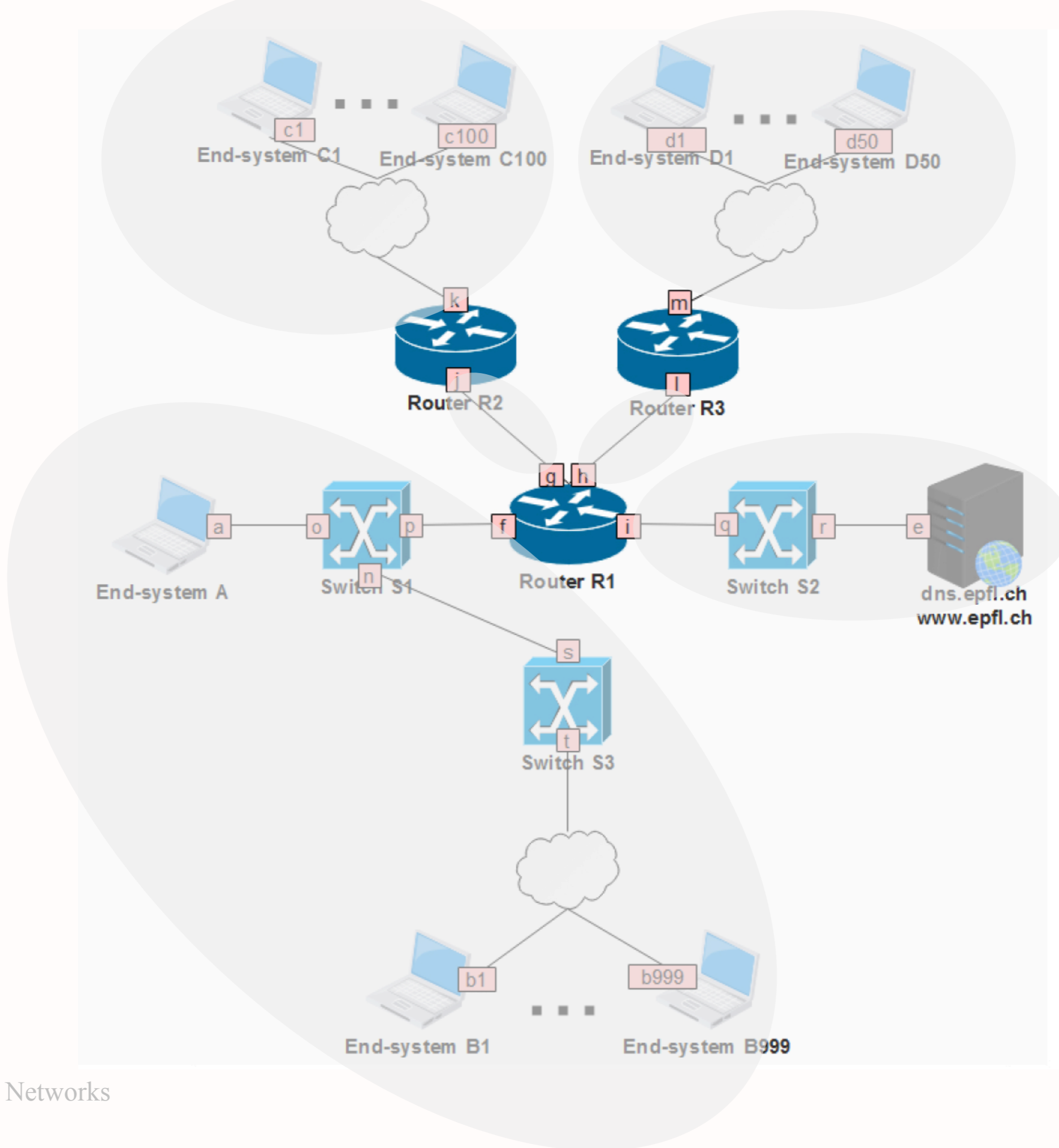
- Given network topology and IP prefix, allocate IP addresses using smallest possible range per IP subnet
- Final 2018, Problem 2, Question 1

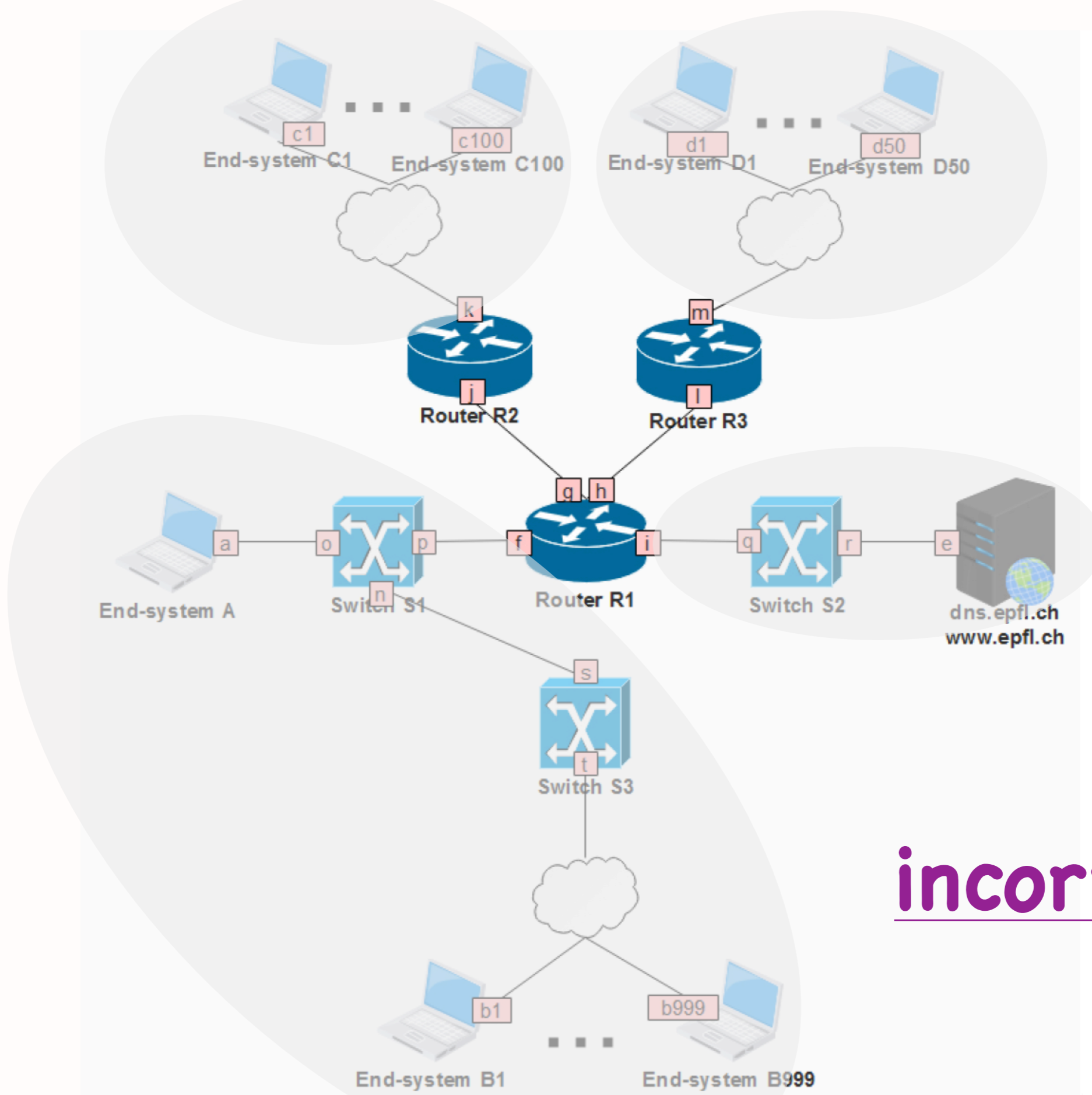
Allocate IP addresses from:
100.0.0.0/16



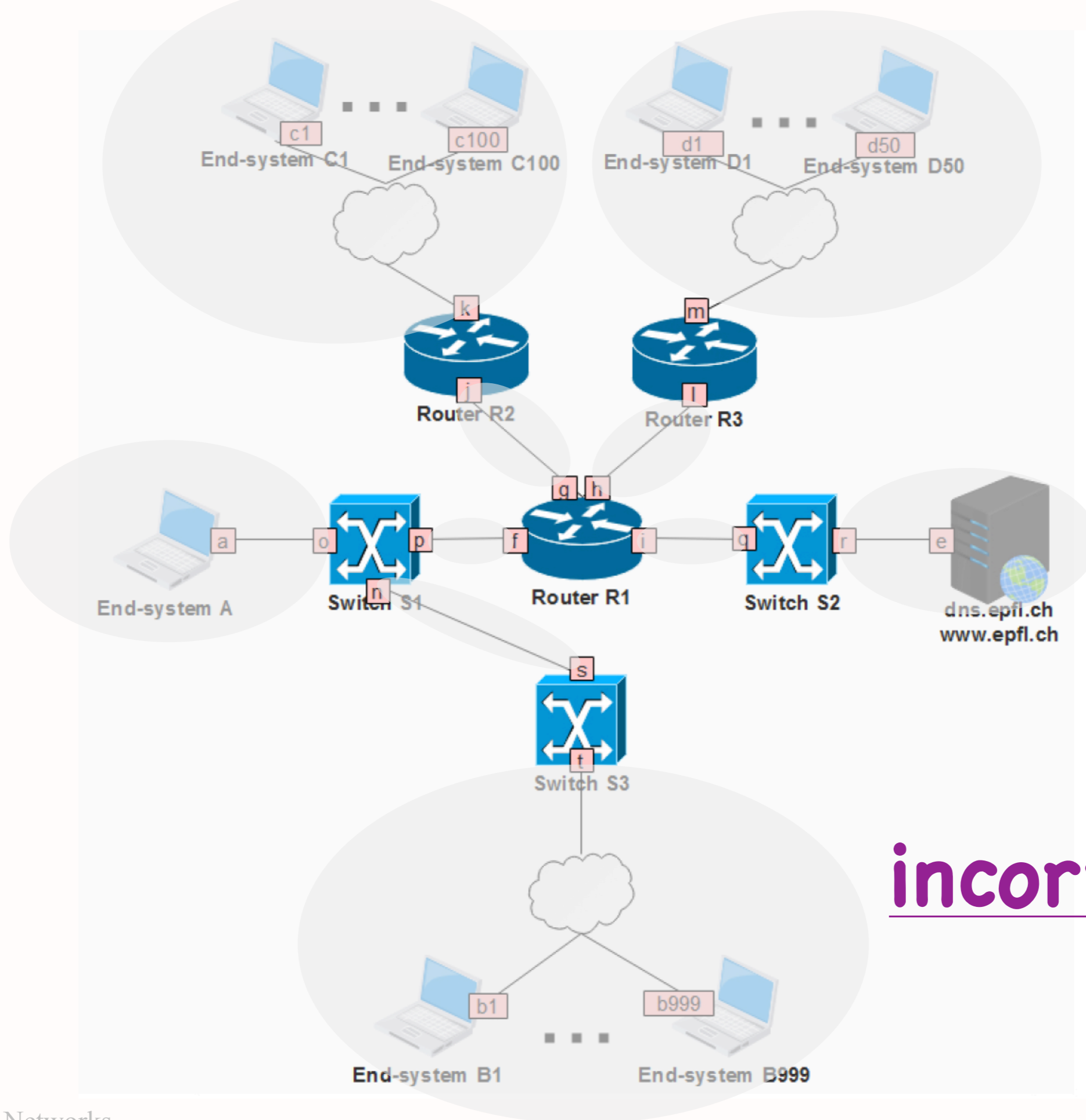
Step 1: Draw the IP subnets

- IP subnet = contiguous network area that has routers only at its boundaries
- Each interface of an IP router belongs to a different IP subnet

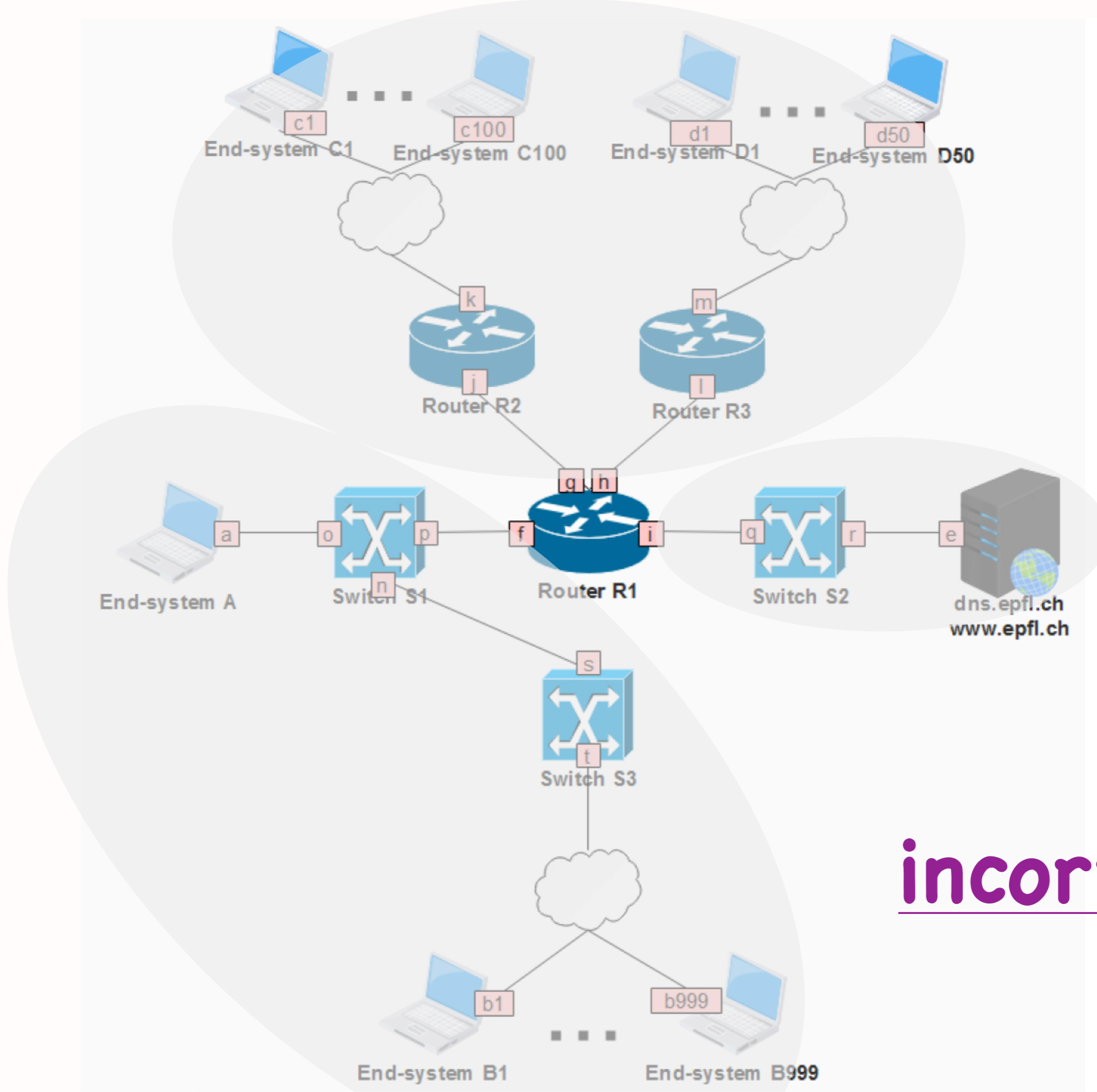




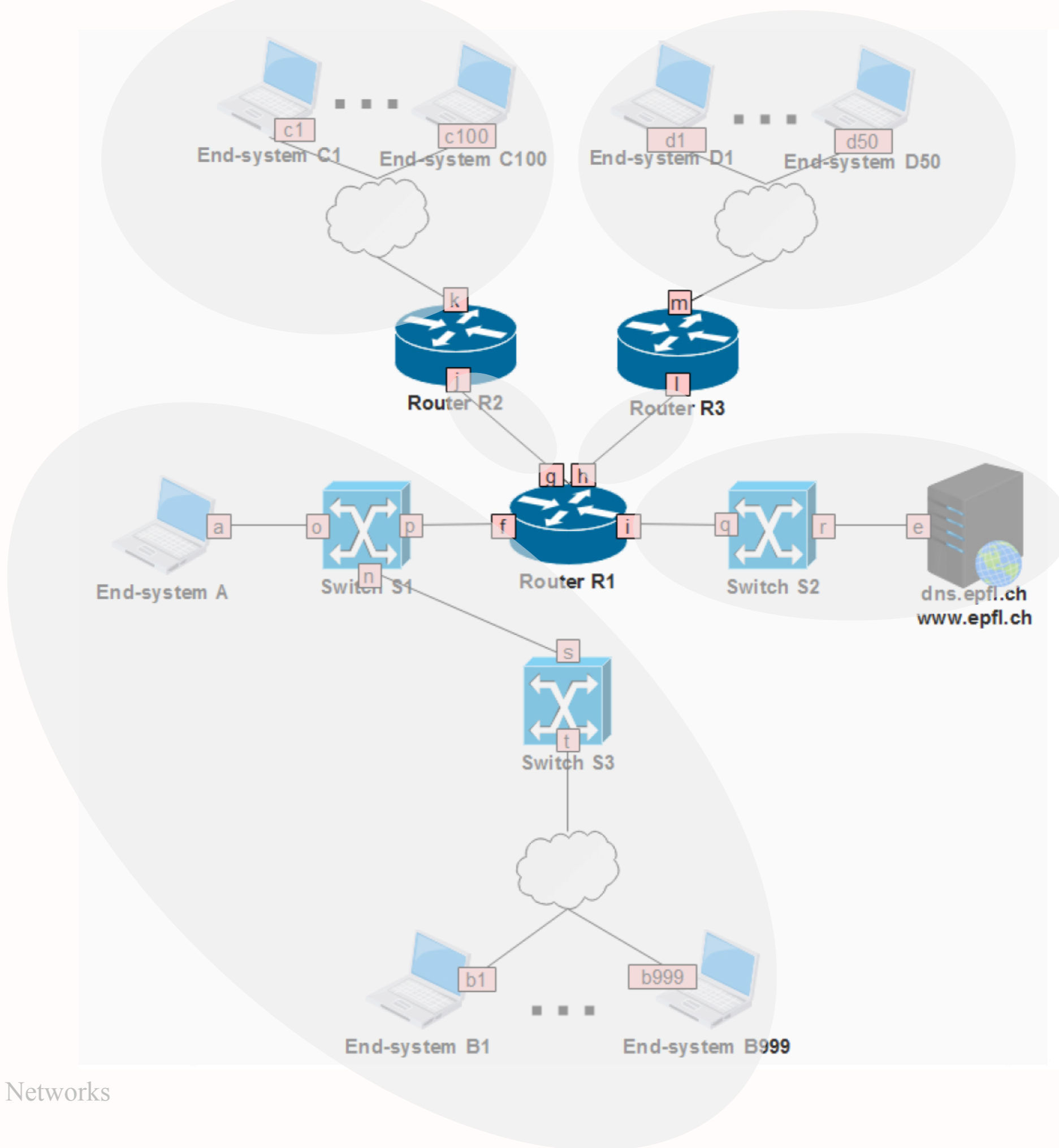
incorrect



incorrect



incorrect

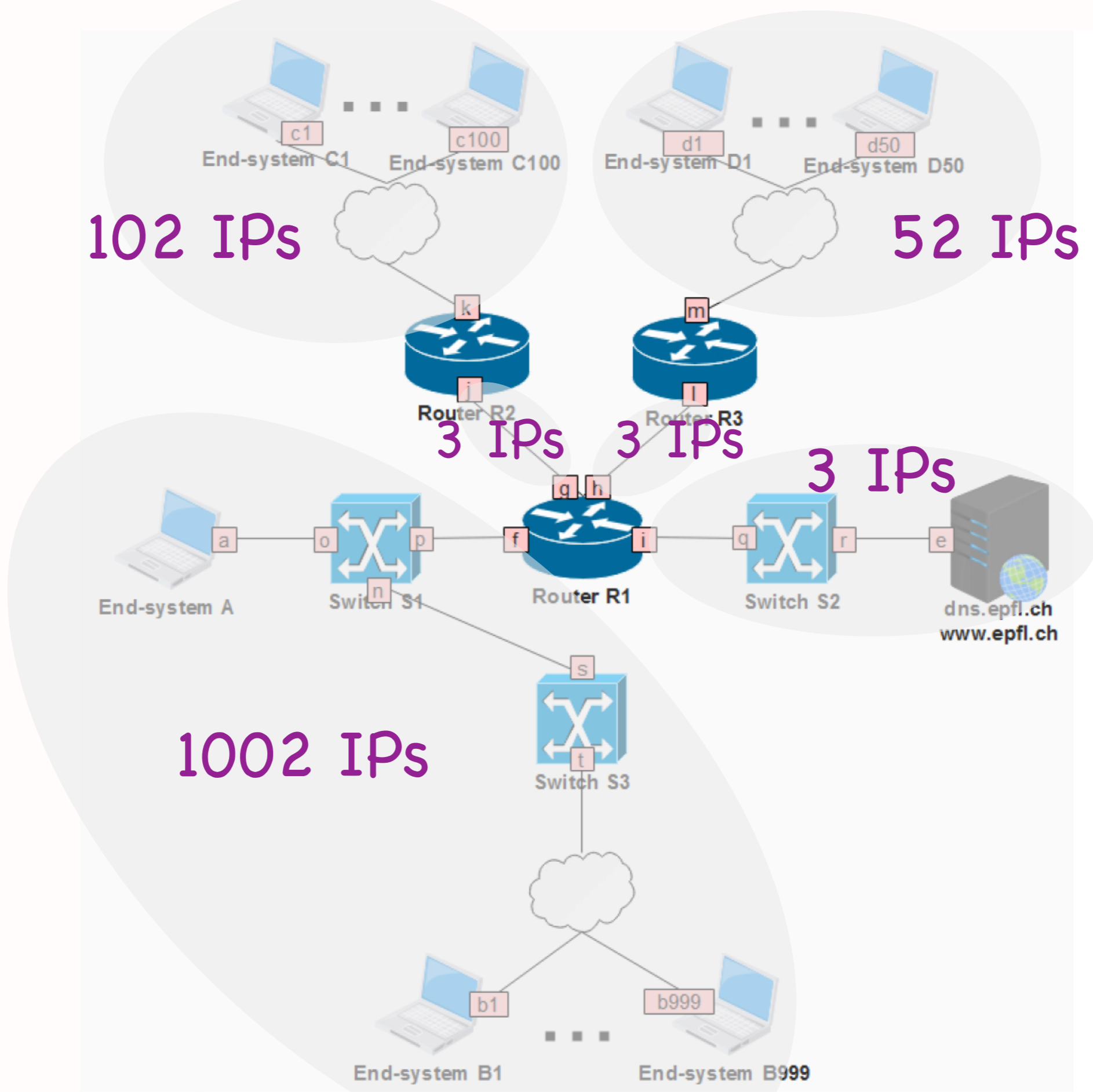


Step 2: Count IPs per subnet

- One IP address per end-system interface
- One IP address per router interface
 - * not needed for IP forwarding,
but needed for other practical reasons
- No IP addresses for link-layer switches
 - * in reality they have IP addresses,
but ignore to simplify exam

Step 2: Count IPs per subnet

- One broadcast IP address
 - * the very last IP address covered by the IP prefix
 - * addresses all entities with an IP address in the local subnet
- No network IP address
 - * the very first IP address covered by the IP prefix
 - * meant to have special meaning, but not typically used



Step 3: Allocate IP prefixes

- One approach: start from the largest IP subnet, allocate consecutive prefixes
- Whatever approach you choose:
IP prefixes allocated to different IP subnets must not overlap

Step 3: Allocate IP prefixes

- 1st IP subnet: 100.0.0.0/22
 - * 1002 IPs => **we need 10 bits** (22-bit mask)
 - * available IP prefix: 100.0.0.0/16
 - * 01100100 00000000 xxxxxxxx xxxxxxxx
 - * **01100100 00000000 000000xx xxxxxxxx**
 - * **allocated IP prefix: 100.0.0.0/22**
 - * we get 1024 addresses
 - * we are "wasting" some address space because the number of addresses is not a power of 2

Step 3: Allocate IP prefixes

- 2nd IP subnet: 100.0.4.0/25
 - * 102 IPs => **we need 7 bits** (25-bit mask)
 - * last allocated IP prefix: 100.0.0.0/22
 - * 01100100 00000000 000000xx xxxxxxxx
 - * 01100100 00000000 000001xx xxxxxxxx
 - * **01100100 00000000 00000100 0xxxxxxx**
 - * **100.0.4.0/25**

Step 3: Allocate IP prefixes

- 3rd IP subnet: 100.0.4.128/26
 - * 52 IPs => **we need 6 bits** (26-bit mask)
 - * last allocated IP prefix: 100.0.4.0/25
 - * 01100100 00000000 00000100 0xxxxxxx
 - * 01100100 00000000 00000100 1xxxxxxx
 - * **01100100 00000000 00000100 10xxxxxx**
 - * **100.0.4.128/25**

Step 3: Allocate IP prefixes

- 4th IP subnet: 100.0.4.192/30
 - * 3 IPs => **we need 2 bits** (30-bit mask)
 - * last allocated IP prefix: 100.0.4.128/26
 - * 01100100 00000000 00000100 10xxxxxx
 - * 01100100 00000000 00000100 11xxxxxx
 - * **01100100 00000000 00000100 110000xx**
 - * **100.0.4.192/30**

Step 3: Allocate IP prefixes

- 5th IP subnet: 100.0.4.196/30
 - * 3 IPs => **we need 2 bits** (30-bit mask)
 - * last allocated IP prefix: 100.0.4.192/30
 - * 01100100 00000000 00000100 110000xx
 - * **01100100 00000000 00000100 110001xx**
 - * **100.0.4.196/30**

Step 3: Allocate IP prefixes

- 6th IP subnet: 100.0.4.200/30
 - * 3 IPs => **we need 2 bits** (30-bit mask)
 - * last allocated IP prefix: 100.0.4.196/30
 - * 01100100 00000000 00000100 110001xx
 - * **01100100 00000000 00000100 110010xx**
 - * **100.0.4.200/30**

Step 4: Allocate IP addresses

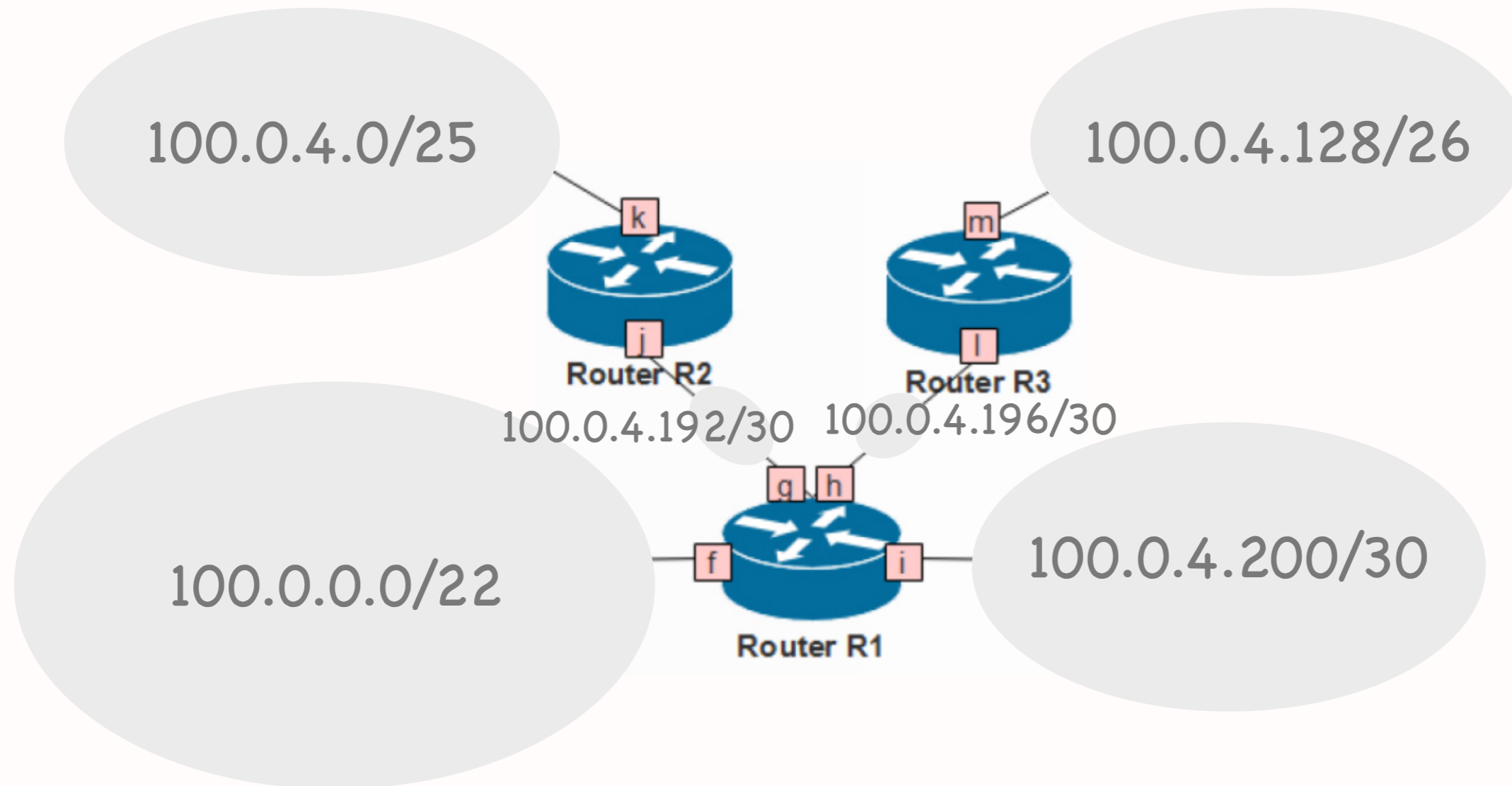
- 1st IP subnet: 1002 addresses from 100.0.0.0/22
 - * 01100100 00000000 000000xx xxxxxxxx
 - * 01100100 00000000 00000011 11111111
 - * broadcast IP address: 100.0.3.255
 - * 100.0.0.0 – 100.0.3.232

Step 4: Allocate IP addresses

- 2nd IP subnet: 102 addresses from 100.0.4.0/25
 - * 01100100 00000000 00000100 0xxxxxxx
 - * 01100100 00000000 00000100 01111111
 - * broadcast IP address: 100.0.4.127
 - * 100.0.4.0 – 100.0.4.100
- ...

Question: Show router tables

- Given network topology and allocated IPs, show router forwarding tables, assuming least-cost path routing protocol that has converged
- Final 2018, Problem 2, Question 2



100.0.0.0/22 f
 100.0.4.0/25 g
 100.0.4.128/26 h
 100.0.4.192/30 g
 100.0.4.196/30 h
 100.0.4.200/30 i

Question: Show packets

- Given a communication scenario, show all the packets transmitted by end-systems and routers
- Final 2018, Problem 2, Question 3

DNS request from A to server

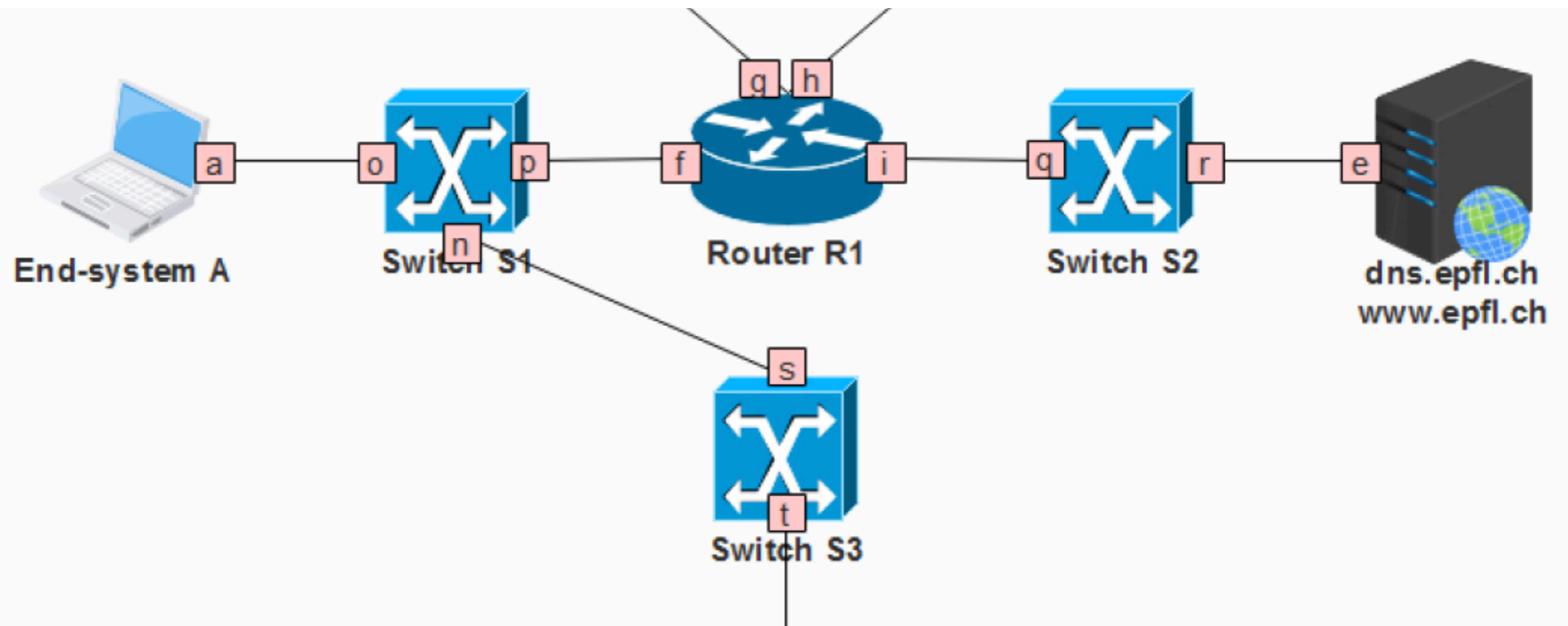
DNS response from server to A

HTTP GET request for base file from A to server

HTTP GET response from server to A

HTTP GET request for image file from A to server

HTTP GET response from server to A



DNS request from A to server

ARP request, MAC: A—broadcast

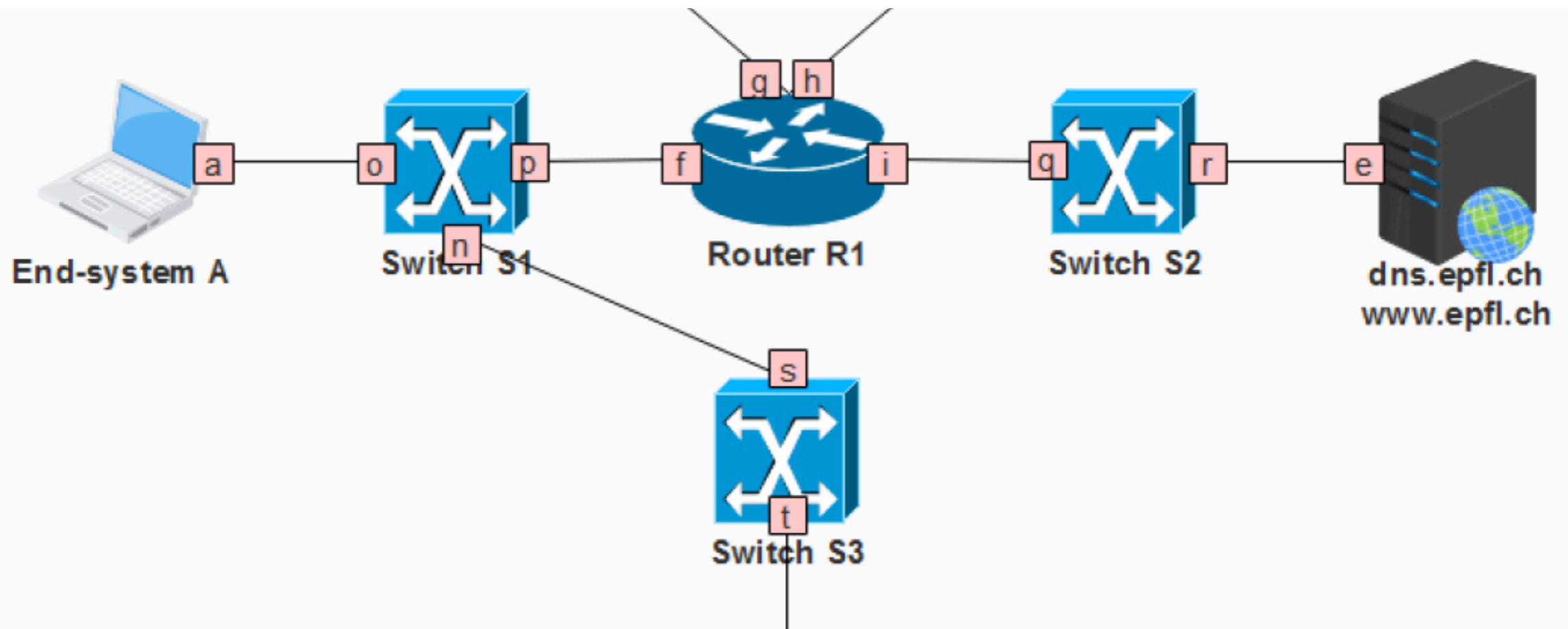
ARP response, MAC: R1—A

DNS request, MAC: A—R1, IP: A—DNS

ARP request, MAC: R1—broadcast

ARP response, MAC: DNS—R1

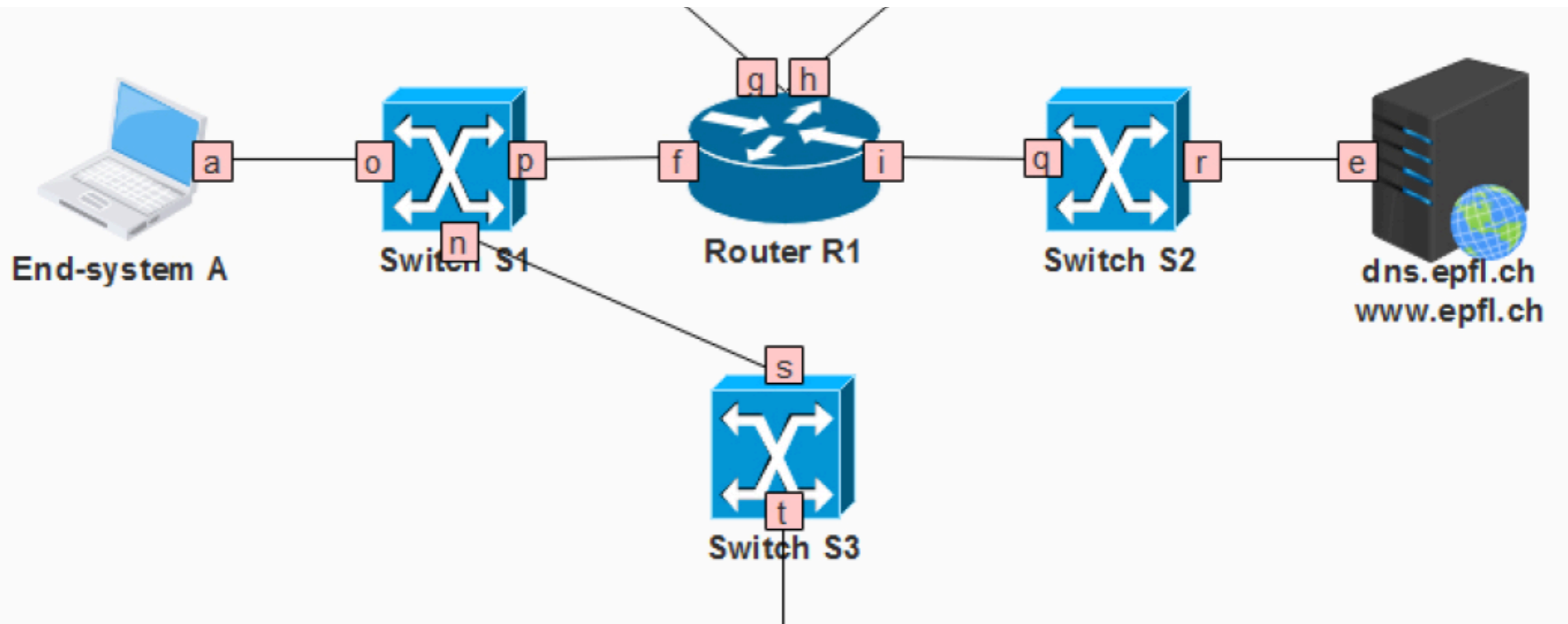
DNS request, MAC: R1—DNS, IP: A—DNS



DNS response from server to A

DNS response, MAC: DNS-R1, IP: DNS-A

DNS response, MAC: R1-A, IP: DNS-A



HTTP GET request from A to server

TCP SYN, MAC: A–R1, IP: A–DNS

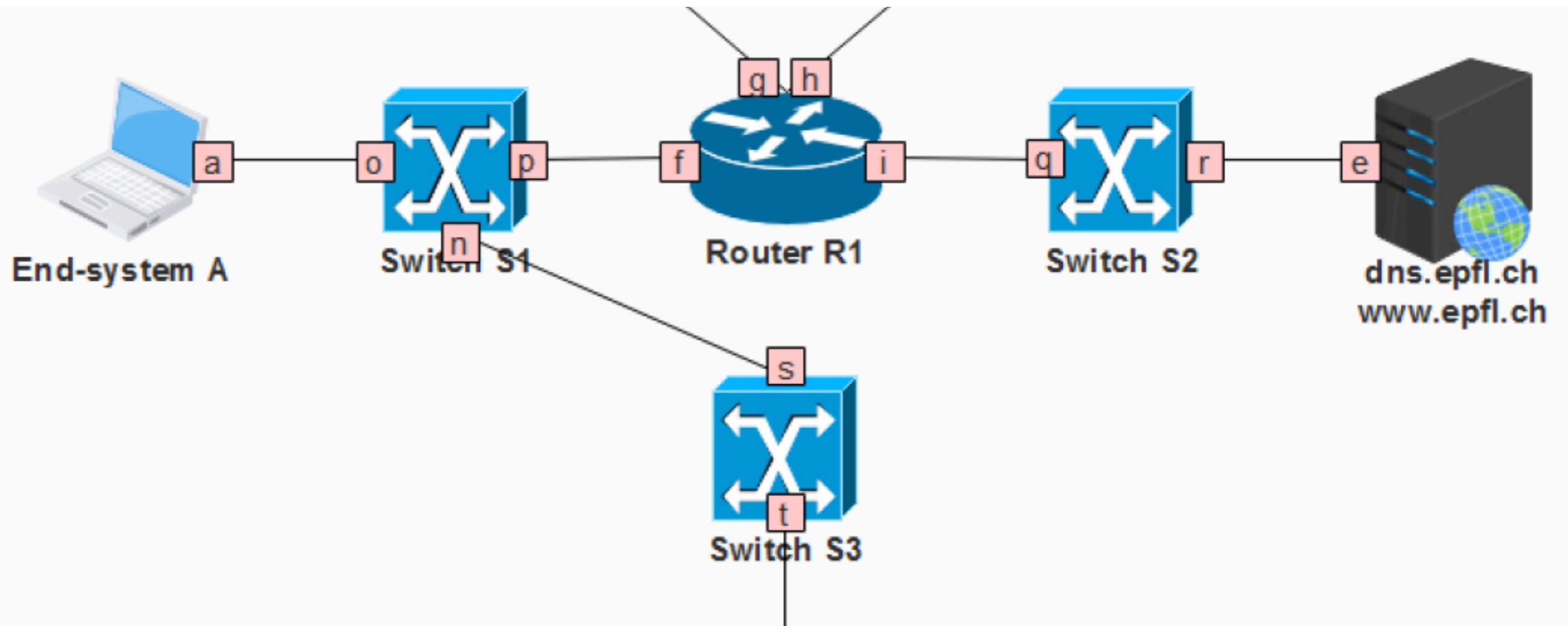
TCP SYN, MAC: R1–DNS, IP: A–DNS

TCP SYN ACK, MAC: DNS–R1, IP: DNS–A

TCP SYN ACK, MAC: R1–A, IP: DNS–A

HTTP GET request, MAC: A–R1, IP: A–DNS

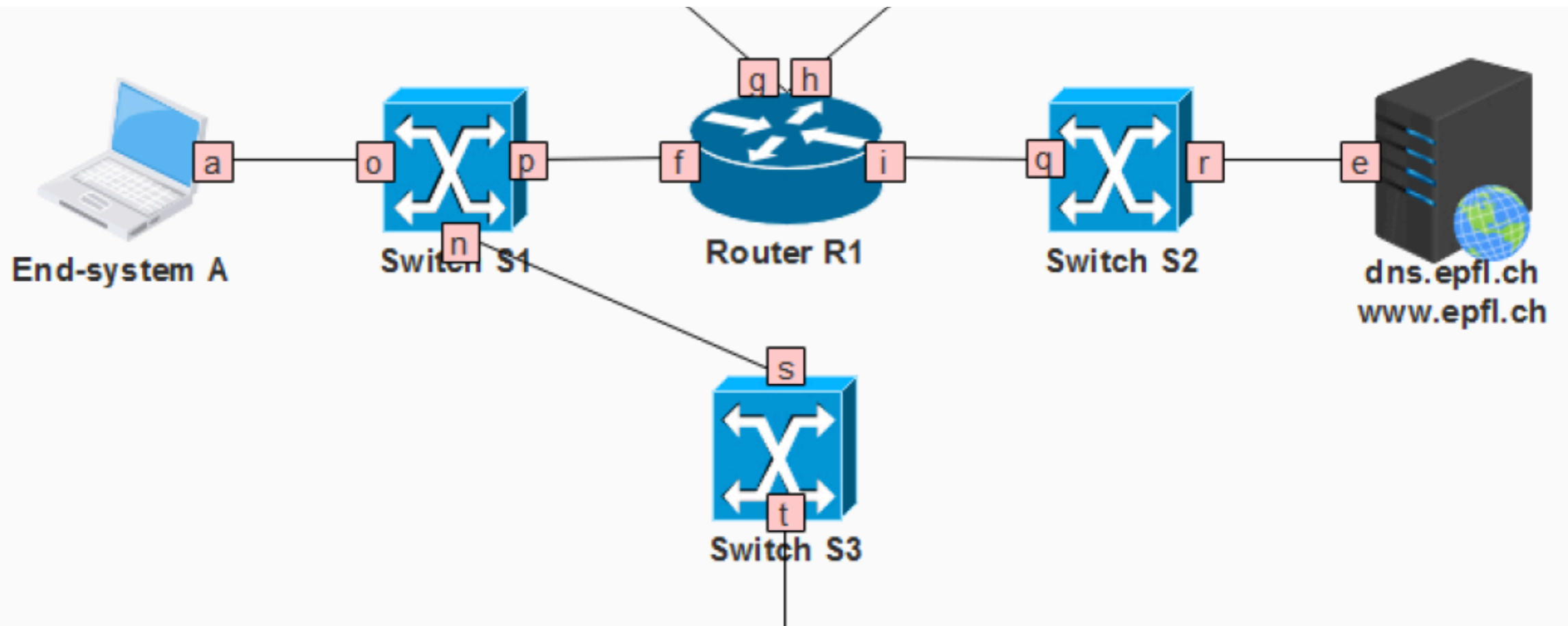
HTTP GET request, MAC: R1–DNS, IP: A–DNS



HTTP GET response from server to A

HTTP GET response, MAC: DNS-R1, IP: DNS-A

HTTP GET response, MAC: R1-A, IP: DNS-A

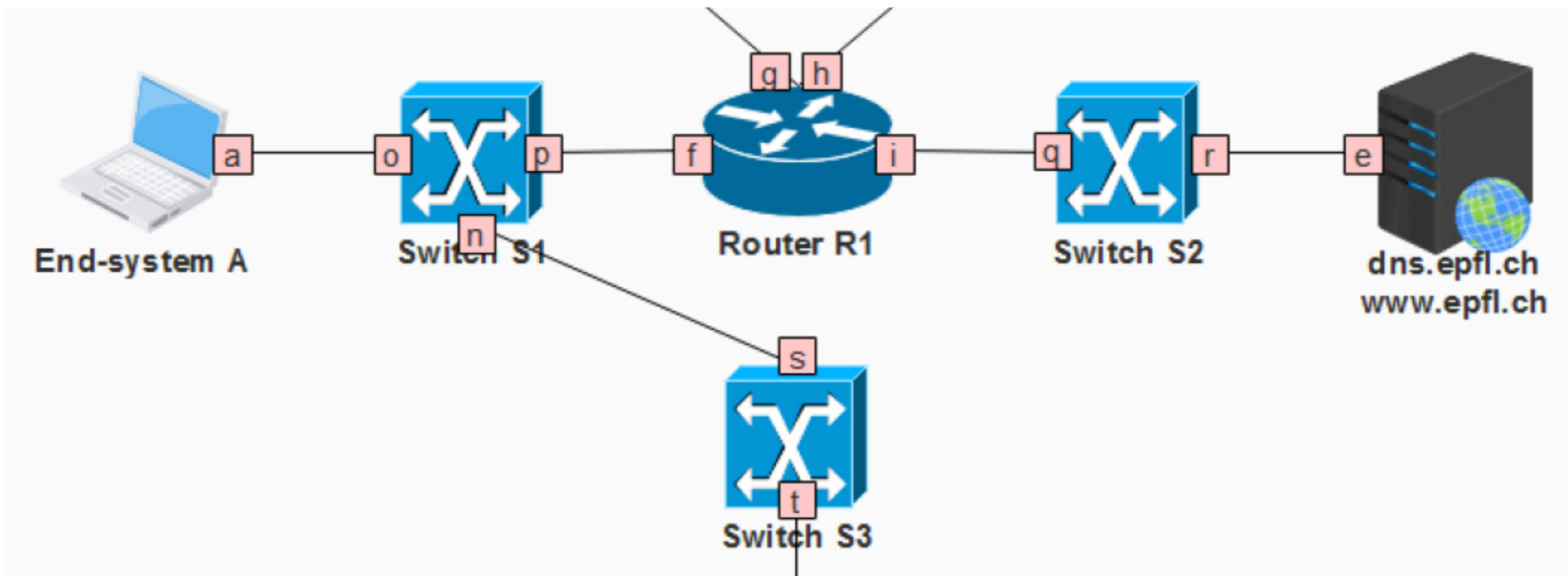


Question: Show switch tables

- Given a communication scenario, show switch forwarding tables
- Final 2018, Problem 2, Question 4

A's MAC o
R1's MAC p

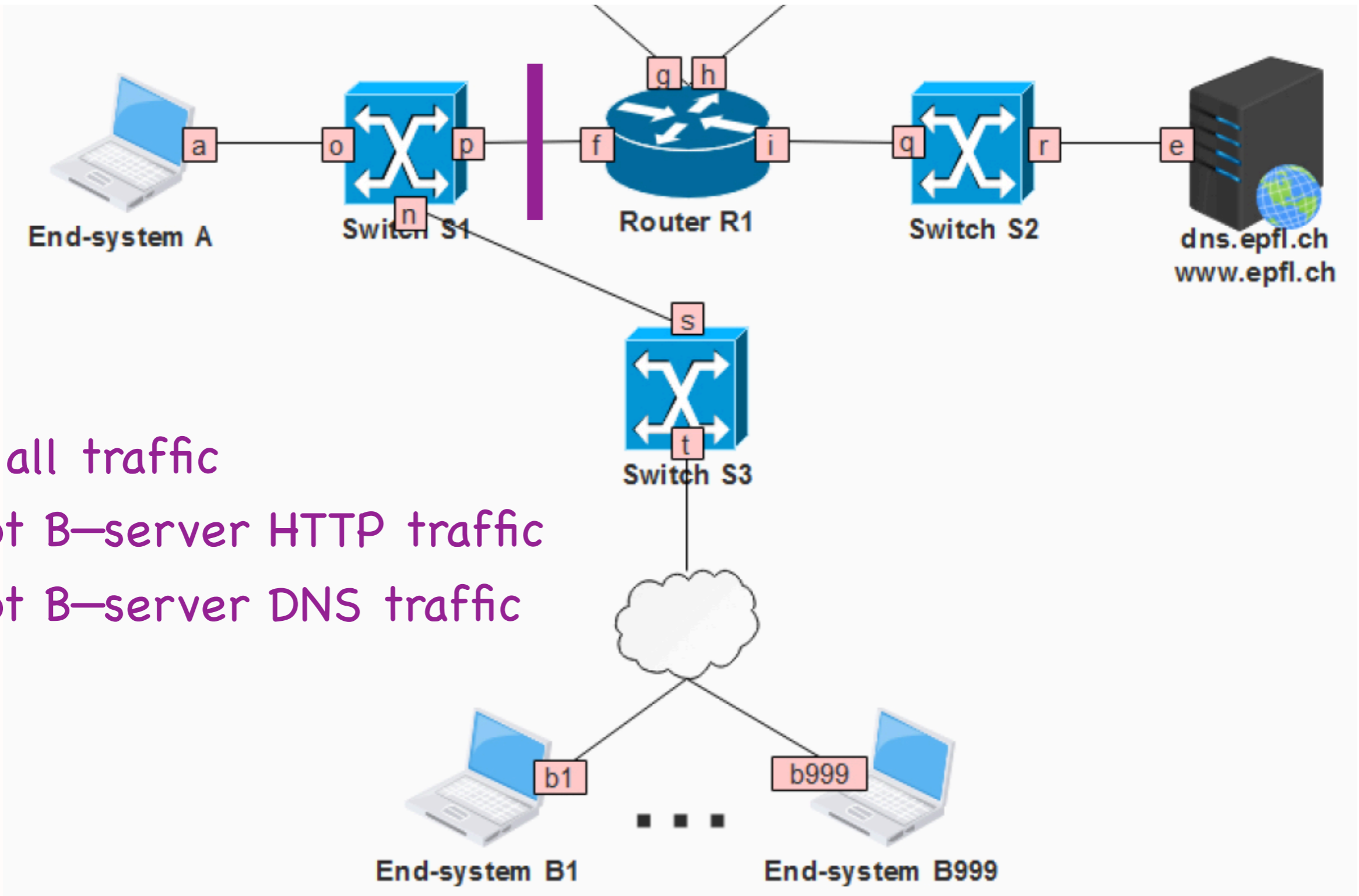
R1's MAC q
DNS's MAC r



A's MAC s

Question: Show filtering table

- Show filtering table that allows a given communication pattern
- Final 2018, Problem 2, Question 5



Deny all traffic
 Except B-server HTTP traffic
 Except B-server DNS traffic

Question: Show filtering table

- List filtering table fields
 - * action, TCP/UDP, src IP, dst IP, src port, dst port
- List entries that achieve given pattern
 - * allow TCP B-prefix server-IP any 80
 - * allow TCP server-IP B-prefix 80 any
 - * allow UDP B-prefix server-IP any 53
 - * allow UDP server-IP B-prefix 53 any
 - * deny any any any any any any

TCP elements

- Connection setup and teardown
- Connection hijacking
- Connection setup (SYN) flooding
- Flow control
- Congestion control

TCP elements

- Connection setup and teardown
- Connection hijacking
- Connection setup (SYN) flooding
- Flow control
- Congestion control

Flow control

- Goal: not overwhelm the **receiver**
 - * not send at a rate that the receiver cannot handle
- How: “**receiver window**”
 - * spare room in receiver's rx buffer
 - * receiver communicates it to sender as TCP header field

Congestion control

- Goal: not overwhelm the **network**
 - * not send at a rate that the network would create network congestion
- How: “**congestion window**”
 - * number of unacknowledged bytes that the sender can transmit without creating congestion
 - * sender estimates it on its own

Self-clocking

- Sender guesses the “right” congestion window based on the ACKs
- ACK = no congestion, increase window
- No ACK = congestion, decrease window

Alice's computer

1 byte

2 bytes

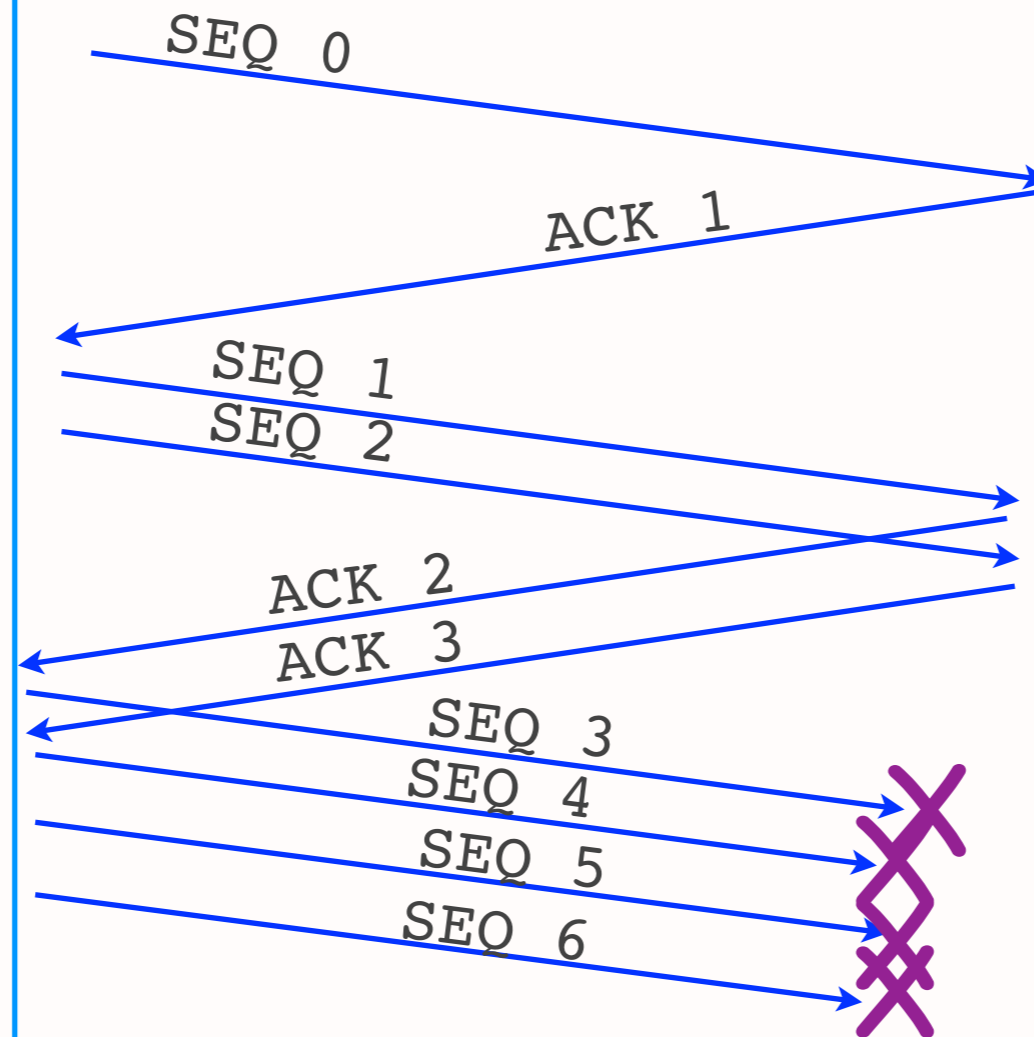
3 bytes

4 bytes

timeout

ssthresh=

2 bytes



Bob's computer

Alice's computer

1 byte

2 bytes

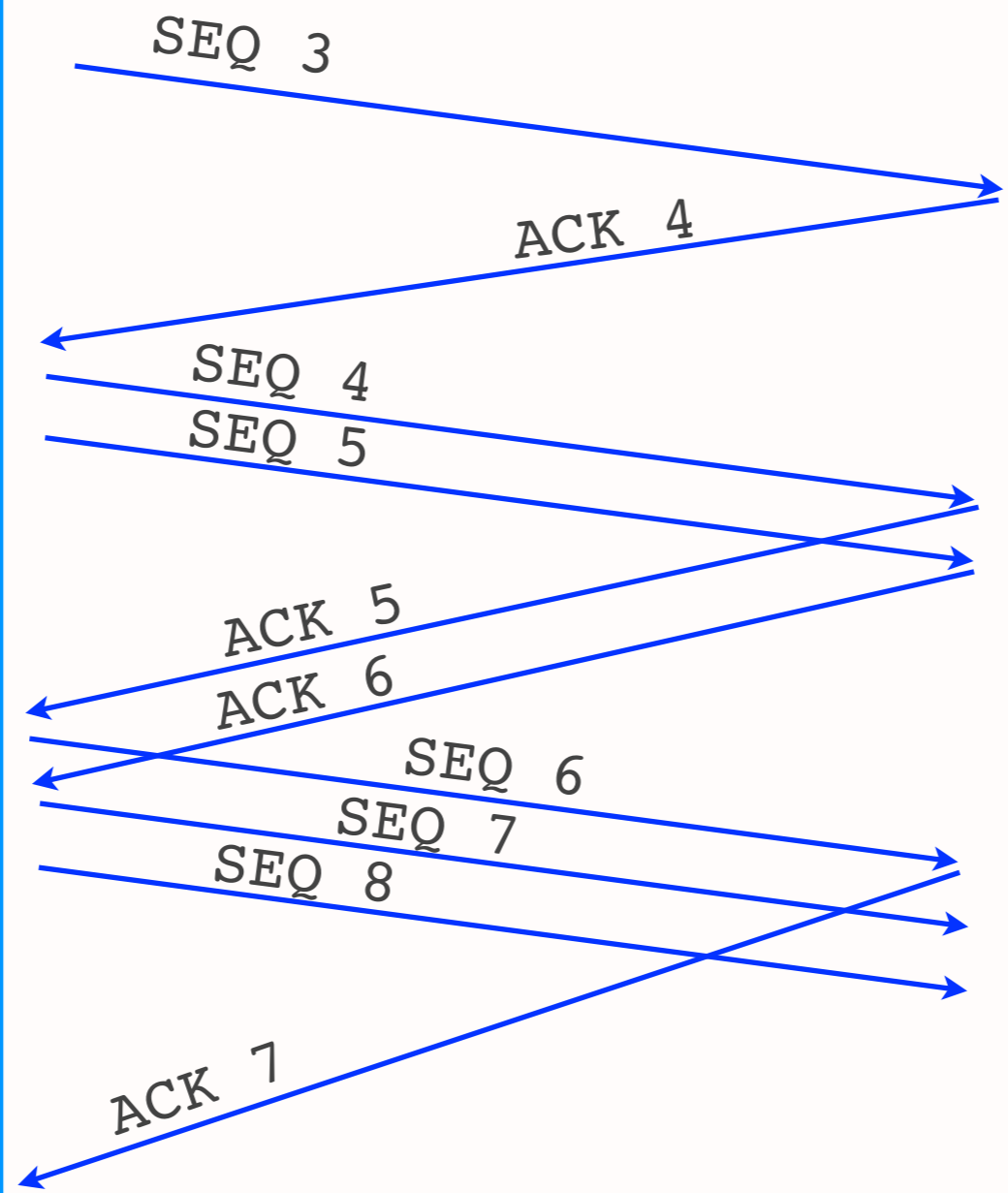
$2 + 1/2 = 2.5$ bytes

$2.5 + 1/2.5 = 3$ bytes

$3 + 1/3$ bytes

ssthresh=
2 bytes

Bob's computer



Basic algorithm (Tahoe)

- Set window to 1 MSS, increase exponentially
- On timeout, reset window to 1 MSS, set ssthresh to last window/2
- On reaching ssthresh, transition to linear increase

Alice's computer

4 bytes

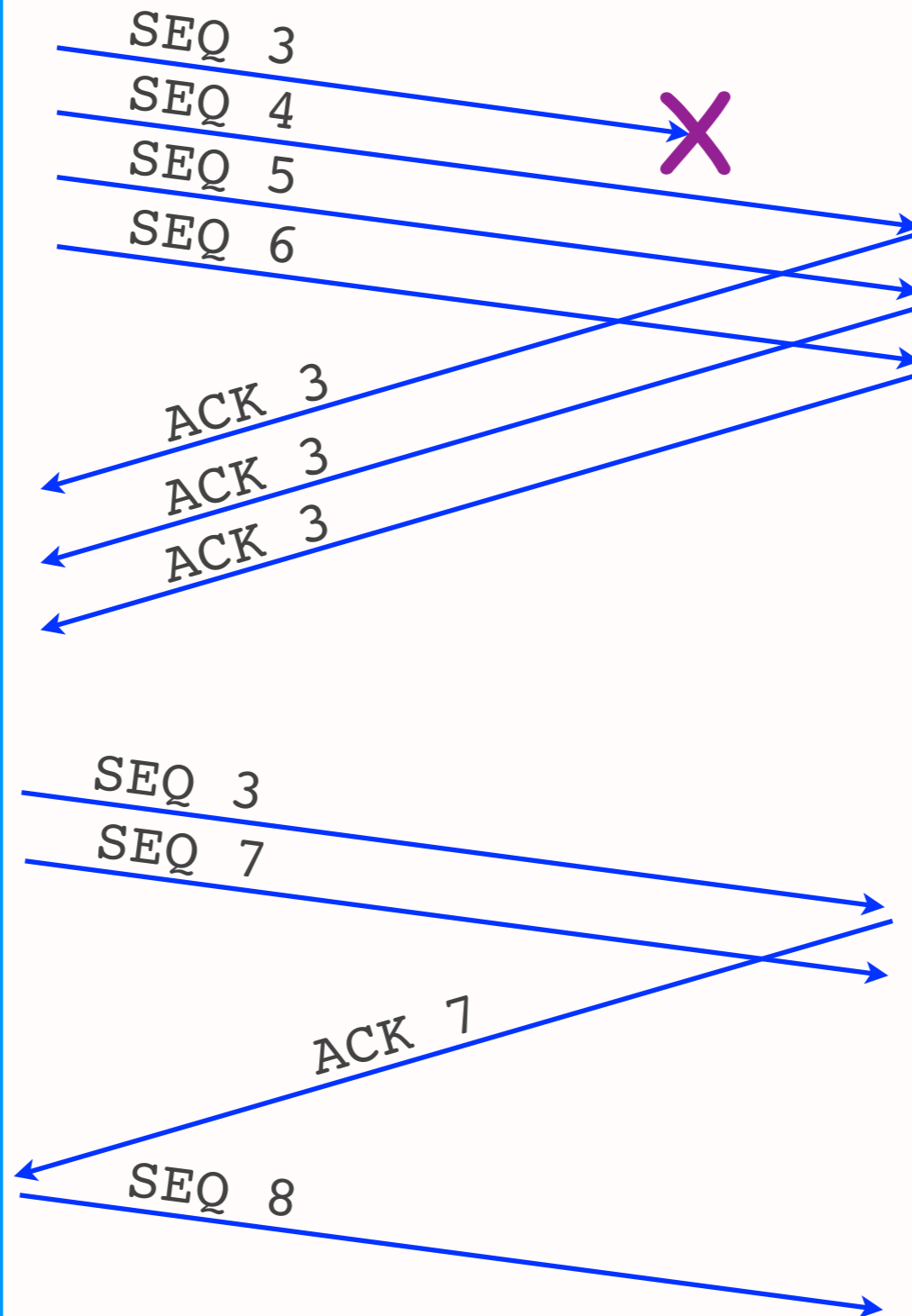
fast
retransmit

5 bytes

2 bytes

ssthresh=
2 bytes

Bob's computer



Basic algorithm (Reno)

- Set window to 1 MSS, increase exponentially
- On timeout, reset window to 1 MSS, set ssthresh to last window/2
- On reaching ssthresh or 3 duplicate ACKs, transition to linear increase

Two retransmission triggers

- **Timeout** \Rightarrow retransmission of oldest unacknowledged segment
- **3 duplicate ACKs** \Rightarrow fast retransmit of oldest unacknowledged segment
 - * avoid **unnecessary wait** for timeout
 - * 1 duplicate ACK not enough \Leftarrow network may have reordered a data segment or duplicated an ACK

TCP terminology

- Exponential increase = **slow start**
 - * on timeout, reset window to 1 MSS
 - * set ssthresh to last window/2
- Linear increase = **congestion avoidance**
 - * on window reaching ssthresh
 - * on receiving 3 duplicate ACKs

Exam material

- All lectures, homework, labs from semester start
- Emphasis on material after midterm + TCP
- Lab-related questions: $\leq 20\%$ of the points

src IP: Alice's, dst IP: Bob's
message



Alice



Bob

src IP: Alice's, dst IP: Charly's
next IP: Bob's
message


Alice


Charly


Bob

src IP: Charly's, dst IP: Bob's
message

src IP: Alice's,
dst IP: Charly's
next IP: Deborah's
next IP: Bob's
message


Charly

src IP: Charly's,
dst IP: Deborah's
next IP: Bob's
message


Alice


Bob


Deborah

src IP: Deborah's,
dst IP: Bob's
message